

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

КАФЕДРА «Робототехника и мехатроника»

**Практикум**

по выполнению практических работ

по дисциплине

«Применение специализированного ПО для интеллектуальной обработки данных»

Ростов-на-Дону

2023

Составители: доцент Изюмов А.И,

Практикум по выполнению лабораторной работы по дисциплине «Применение специализированного ПО для интеллектуальной обработки данных». ДГТУ, г. Ростов-на-Дону, 2023 г.

В практикуме кратко изложены теоретические вопросы, необходимые для успешного выполнения лабораторной работы, рабочее задание и контрольные вопросы для самопроверки.

Предназначено для обучающихся по направлению подготовки (код, название):

15.04.06 Мехатроника и робототехника

Ответственный за выпуск:

И.о. зав. кафедрой (руководитель структурного подразделения, ответственного за реализацию ОПОП) Изюмов Андрей Игоревич

© Издательский центр ДГТУ, 2023г.

**Лабораторная работа №1 «Классы данных в системе Matlab. Инструментальное средство Editor/Debugger. Инструментальное средство Profiler.»**

**Теоретическая часть**

**Классы данных в Matlab**

В Matlab существует множество классов данных, которые позволяют эффективно решать различные задачи:

* **Массивы и матрицы**: Matlab (Matrix Laboratory) изначально разработан для работы с матрицами. Массивы могут быть одномерными (векторами) или многомерными.
* **Структуры**: Переменные, которые могут содержать данные различных типов и размеров.
* **Ячейки**: Это массивы, элементы которых могут содержать любые типы данных.
* **Функциональные дескрипторы**: Позволяют использовать функции как переменные.
* **Классы и объекты**: Matlab поддерживает объектно-ориентированное программирование, что позволяет создавать свои собственные классы данных.

**Editor/Debugger**

Editor в Matlab предоставляет удобный интерфейс для написания, редактирования и отладки кода:

* **Точки останова (Breakpoints)**: Позволяют приостановить выполнение программы в определенной точке для анализа.
* **Шаговое выполнение**: Позволяет выполнять код построчно, облегчая процесс отладки.
* **Просмотр переменных**: В процессе отладки можно просматривать текущие значения переменных.

**Profiler**

Profiler - это инструмент для анализа производительности программ:

* **Анализ времени выполнения**: Позволяет определить, какие части кода занимают больше всего времени.
* **Вызовы функций**: Profiler показывает, какие функции были вызваны и сколько раз, а также время их выполнения.
* **Оптимизация**: С помощью Profiler можно определить узкие места в коде и оптимизировать их для увеличения производительности.

**Инструкция по технике безопасности при выполнении лабораторной работы**

При выполнении лабораторной работы необходимо соблюдать правила техники безопасности при работе с электрооборудованием, включение электрооборудования, в том числе компьютеров, производить только после разрешения преподавателя или сопровождающего инженера.

**Цель работы**

Приобрести навыки работы с основными классами данных в системе Matlab, освоить функциональность инструментального средства Editor/Debugger для отладки программ, познакомиться с инструментом Profiler для анализа производительности кода.

**Рабочее задание**

1. Изучить основные классы данных системы Matlab.
2. Создать простой скрипт на языке Matlab, используя различные типы данных.
3. Отладить созданный скрипт с помощью инструмента Editor/Debugger.
4. Проанализировать производительность скрипта с использованием инструмента Profiler.

**Материально-техническое обеспечение работы**

Компьютер с установленным пакетом Matlab.

**Порядок выполнения работы**

1. **Изучение классов данных**
   * В главном меню Matlab перейдите в раздел "Help" или "Справка".
   * В поисковой строке введите "Data types in Matlab" или "Типы данных в Matlab".
   * Изучите предоставленную информацию о основных классах данных.
2. **Создание скрипта**
   * В главном окне Matlab выберите "New Script" или "Новый скрипт".
   * В открывшемся редакторе кода напишите простой скрипт, где используются разные классы данных (например, векторы, матрицы, структуры).
   * Сохраните скрипт, выбрав "File" -> "Save" или "Файл" -> "Сохранить".
3. **Отладка с использованием Editor/Debugger**
   * В редакторе кода найдите кнопку "Run and Advance" или "Запустить и продолжить".
   * Если код содержит ошибки, Editor/Debugger подсветит их красным.
   * Наведите курсор на подсвеченную ошибку, чтобы увидеть описание проблемы.
   * Исправьте ошибку и продолжайте отладку, пока все ошибки не будут устранены.
4. **Анализ производительности с использованием Profiler**
   * В главном меню Matlab выберите "Profiler".
   * В открывшемся окне Profiler нажмите на кнопку "Start Profiling" или "Начать профилирование".
   * Запустите ваш скрипт.
   * После завершения работы скрипта проанализируйте отчёт в Profiler. Обратите внимание на участки кода, которые требуют больше времени на выполнение, и рассмотрите возможность их оптимизации.
5. **Остановка Profiler и сохранение результатов**
   * В окне Profiler нажмите на кнопку "Stop Profiling" или "Остановить профилирование".
   * Результаты можно сохранить, выбрав "File" -> "Save" или "Файл" -> "Сохранить".

**Контрольные вопросы**

1. Что такое классы данных в системе Matlab?
2. Какие основные классы данных существуют в Matlab?
3. Для чего предназначен инструмент Editor/Debugger в Matlab?
4. Какие основные функции выполняет инструмент Profiler?
5. Каким образом можно оптимизировать код на основе данных из Profiler?

**Лабораторная работа №2 «Работа с текстовыми и двоичными файлами в Matlab.»**

**Теоретическая часть**

**Текстовые файлы в Matlab**

Текстовые файлы - это файлы, содержащие текстовую информацию, которую можно прочитать и редактировать с помощью стандартных текстовых редакторов.

* **Открытие и закрытие текстовых файлов**:
  + Для открытия файлов используется функция **fopen**, а для их закрытия - **fclose**.
* **Чтение из текстовых файлов**:
  + Можно использовать функции **fgetl** или **fgets** для чтения строк или **fscanf** для чтения определенного формата данных.
* **Запись в текстовые файлы**:
  + Функция **fprintf** позволяет записывать форматированный текст в файл.

**Двоичные файлы в Matlab**

Двоичные файлы содержат информацию в форме, которая не предназначена для чтения человеком, но может быть быстро и эффективно обработана компьютером.

* **Открытие и закрытие двоичных файлов**:
  + Подобно текстовым файлам, для открытия используется **fopen** с соответствующим режимом, например, **'rb'** (чтение двоичного файла).
* **Чтение из двоичных файлов**:
  + Функция **fread** позволяет читать данные из двоичного файла.
* **Запись в двоичные файлы**:
  + Функция **fwrite** используется для записи данных в двоичный файл.

**Обработка ошибок при работе с файлами**

При работе с файлами важно обрабатывать возможные ошибки, такие как отсутствие файла, ошибки при чтении или записи. Matlab предоставляет инструменты, такие как **ferror** и **try-catch**, чтобы упростить этот процесс.

**Инструкция по технике безопасности при выполнении лабораторной работы**

При выполнении лабораторной работы необходимо соблюдать правила техники безопасности при работе с электрооборудованием, включение электрооборудования, в том числе компьютеров, производить только после разрешения преподавателя или сопровождающего инженера.

**Цель работы**

Освоить методы работы с текстовыми и двоичными файлами в Matlab, научиться записывать в них данные и считывать данные из них.

**Рабочее задание**

1. Изучить методы работы с текстовыми файлами в Matlab.
2. Изучить методы работы с двоичными файлами в Matlab.
3. Создать текстовый и двоичный файл, записать в них данные, а затем считать данные обратно.

**Материально-техническое обеспечение работы**

Компьютер с установленным пакетом Matlab.

**Порядок выполнения работы**

1. **Изучение работы с текстовыми файлами**
   * В справке Matlab или предоставленных учебных материалах найдите раздел о работе с текстовыми файлами.
   * Ознакомьтесь с функциями **fopen**, **fprintf**, **fscanf** и **fclose** для работы с текстовыми файлами.
2. **Создание и запись в текстовый файл**
   * Создайте новый скрипт в Matlab.
   * Используйте функцию **fopen** для создания нового текстового файла или открытия существующего.
   * Запишите некоторые данные в файл с помощью функции **fprintf**.
   * Закройте файл с помощью функции **fclose**.
3. **Чтение данных из текстового файла**
   * Откройте текстовый файл для чтения с помощью функции **fopen**.
   * Используйте функцию **fscanf** для считывания данных из файла.
   * Закройте файл.
4. **Изучение работы с двоичными файлами**
   * Ознакомьтесь с функциями **fread** и **fwrite** для работы с двоичными файлами в справке Matlab.
5. **Создание и запись в двоичный файл**
   * В новом скрипте используйте **fopen** для создания двоичного файла.
   * Запишите некоторые данные в файл с помощью функции **fwrite**.
   * Закройте файл.
6. **Чтение данных из двоичного файла**
   * Откройте двоичный файл для чтения.
   * Считайте данные из файла с помощью функции **fread**.
   * Закройте файл.

**Контрольные вопросы**

1. Какие основные функции используются для работы с текстовыми файлами в Matlab?
2. В чём особенность работы с двоичными файлами по сравнению с текстовыми?
3. Какие режимы открытия файлов вы знаете?
4. Как сохранить матрицу данных в текстовом файле?
5. Каковы преимущества и недостатки использования двоичных файлов по сравнению с текстовыми?

**Лабораторная работа №3 «Процедуры ответных вызовов и S-функции системы Matlab.»**

**Теоретическая часть**

**Процедуры ответных вызовов (Callback Functions) в Matlab**

Процедуры ответных вызовов в Matlab - это функции, которые автоматически выполняются в ответ на определенные действия или события, например, при нажатии кнопки в графическом интерфейсе.

* **Создание и использование**: Callback-функции можно привязать к различным элементам графического интерфейса, таким как кнопки, ползунки и др. Это позволяет создавать интерактивные программы.
* **Типы ответных вызовов**: В зависимости от элемента графического интерфейса и действия, которое нужно выполнить, существуют разные типы callback-функций, например, **ButtonDownFcn**, **KeyPressFcn** и так далее.

**S-функции в Matlab**

S-функции (System-functions) в Matlab позволяют создавать пользовательские динамические системы, которые можно использовать в Simulink.

* **Структура S-функции**: Основная часть S-функции - это switch-case структура, которая обрабатывает различные этапы выполнения динамической системы, такие как инициализация, вычисления на каждом шаге, завершение и так далее.
* **Создание S-функций**: S-функции могут быть написаны на языке программирования Matlab (m-файлы) или на языке C/C++ (mex-файлы). Они позволяют реализовать динамические системы с произвольной структурой и поведением.

**Интеграция S-функций и Callback-функций**

При создании сложных программ и моделей в Matlab и Simulink можно комбинировать S-функции и callback-функции, чтобы обеспечить интерактивность и точную реализацию желаемого поведения системы.

**Инструкция по технике безопасности при выполнении лабораторной работы**

При выполнении лабораторной работы необходимо соблюдать правила техники безопасности при работе с электрооборудованием, включение электрооборудования, в том числе компьютеров, производить только после разрешения преподавателя или сопровождающего инженера.

**Цель работы**

Освоение методов создания процедур ответных вызовов и использования S-функций в Matlab для расширения функциональных возможностей и управления интерфейсами.

**Рабочее задание**

1. Изучить основы создания процедур ответных вызовов в Matlab.
2. Познакомиться с методами создания и использования S-функций в Matlab.
3. Продемонстрировать применение процедур ответных вызовов и S-функций в практическом примере.

**Материально-техническое обеспечение работы**

Компьютер с установленным пакетом Matlab.

**Порядок выполнения работы**

1. **Изучение процедур ответных вызовов**
   * В справке Matlab или учебных материалах найдите информацию о процедурах ответных вызовов.
   * Ознакомьтесь с методами создания графических интерфейсов, которые используют процедуры ответных вызовов для реагирования на действия пользователя.
2. **Создание простого графического интерфейса с процедурой ответного вызова**
   * Используйте инструмент GUIDE или App Designer для создания простого графического интерфейса.
   * Добавьте кнопку или другой элемент управления.
   * Назначьте этому элементу процедуру ответного вызова, которая будет выполняться при его активации.
3. **Основы S-функций**
   * В справке Matlab ознакомьтесь с разделом о S-функциях.
   * Изучите, как создавать и использовать S-функции для создания пользовательских блоков в Simulink.
4. **Создание и использование S-функции**
   * Создайте простую S-функцию, которая, например, вычисляет квадрат входного сигнала.
   * Добавьте вашу S-функцию в модель Simulink и соедините ее с другими блоками.

**Контрольные вопросы**

1. Что такое процедура ответного вызова в контексте Matlab?
2. Какие события могут инициировать процедуры ответных вызовов?
3. В чем заключается основная идея S-функций в Matlab?
4. Каковы преимущества использования S-функций в Simulink?
5. Могут ли S-функции взаимодействовать с базовым рабочим пространством Matlab?

**Лабораторная работа №4 «Создание окна с параметрами задачи управления, привязка к модели событий и запуск перерасчёта и симуляции»**

**Теоретическая часть**

**Введение в графическую анимацию**

Графическая анимация в MATLAB представляет собой визуализацию изменений объектов или данных во времени. Это мощный инструмент для демонстрации динамики систем, визуализации результатов моделирования или создания интерактивных учебных приложений.

**Создание базовой анимации**

* **Обновление графических объектов**: Наиболее простой способ анимации заключается в изменении свойств графических объектов, таких как точки данных в графике, с использованием цикла.
* **Функция drawnow**: Эта функция обновляет графическое окно и выводит последние изменения объектов. Она часто используется внутри циклов для создания анимации.

**3. Использование таймерных объектов для анимации**

Таймерные объекты MATLAB позволяют выполнять функции в заданные промежутки времени, что идеально подходит для создания анимации с регулярным обновлением.

**Сохранение анимации**

* **Функция getframe**: Эта функция захватывает текущее состояние графического окна в виде кадра, который можно затем использовать для создания видео.
* **Функции VideoWriter и writeVideo**: Эти функции позволяют создавать видеофайлы из набора кадров, полученных с помощью функции **getframe**.

**Использование управляемых объектов для интерактивной анимации**

С помощью элементов графического интерфейса, таких как слайдеры или кнопки, можно контролировать анимацию, внося изменения в реальном времени.

**Цель работы**

Освоение инструментов и методик создания графической анимации в среде программирования Matlab.

**Рабочее задание**

1. Изучить инструменты Matlab, предназначенные для создания графической анимации.
2. Создать базовую анимацию, демонстрирующую принципы работы с графическими объектами в Matlab.
3. Продемонстрировать применение анимации для визуализации динамических процессов.

**Материально-техническое обеспечение работы**

Компьютер с установленным пакетом Matlab.

**Порядок выполнения работы**

1. **Ознакомление с графическими объектами Matlab**
   * В справке Matlab найдите раздел посвященный графическим объектам.
   * Изучите основные классы объектов, такие как **figure**, **axes**, **line** и другие.
2. **Создание базовой анимации**
   * Создайте новую фигуру (**figure**) и оси (**axes**).
   * Используя функцию **line**, создайте линию или другой графический объект.
   * В цикле изменяйте свойства графического объекта (например, координаты **XData** и **YData** для линии), чтобы создать анимационный эффект.
   * Используйте функцию **drawnow** внутри цикла для обновления графического отображения.
3. **Применение анимации для визуализации динамических процессов**
   * Создайте сценарий, моделирующий динамический процесс (например, движение тела под действием гравитации).
   * Используя анимацию, визуализируйте этот процесс на графике.
4. **Сохранение и экспорт анимации**
   * Ознакомьтесь с инструментами Matlab для сохранения анимации в виде видеофайла или GIF.
   * Экспортируйте созданную анимацию в выбранном формате.

**Контрольные вопросы**

1. Какие графические объекты доступны в Matlab для создания анимации?
2. Что делает функция drawnow и почему она важна для анимации?
3. Каким образом можно управлять скоростью анимации в Matlab?
4. Как сохранить созданную анимацию в видеоформате или GIF в Matlab?
5. Какие дополнительные инструменты и функции Matlab могут быть полезны для создания более сложной анимации?