Министерство образования и науки Украины
Севастопольский национальный технический университет

РАЗРАБОТКА MATLAB–ПРИЛОЖЕНИЙ С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ ПОЛЬЗОВАТЕЛЯ

Методические указания
к выполнению лабораторной работы по дисциплине
"Разработка прикладного программного обеспечения ЭВМ"
для студентов дневной формы обучения
направления подготовки 050201 – Системная инженерия

Севастополь
2007

УДК 681.5

Разработка MATLAB-приложений с графическим интерфейсом пользователя: Методические указания к выполнению лабораторной работы по дисциплине "Разработка прикладного программного обеспечения ЭВМ" для студентов дневной формы обучения направления 050201 – Системная инженерия /Разраб. В.А. Карапетьян. – Севастополь: Изд-во СевНТУ, 2007. – 24 с.

В методических указаниях приводится описание лабораторной работы, предназначенной для изучения и практической разработки приложений Matlab с графическим интерфейсом пользователя (GUI).

Методические указания предназначены для студентов дневной формы обучения направления подготовки 050201 – Системная инженерия.

Методические указания рассмотрены и утверждены

на заседании кафедры Технической кибернетики,

протокол № 8 от " 24 " апреля 2007 г.

Допущено учебно-методическим центром СевНТУ

в качестве методических указаний.

Рецензент Крамарь В.А. канд. техн. наук, доцент.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Изучение методов, способов и приемов программирования графического интерфейса пользователя (GUI – Graphic User Interface) в приложениях Matlab. Знакомство с визуальной средой проектирования интерфейса – GUIDE – конструктором графического интерфейса пользователя.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Matlab – многофункциональная интегрированная система автоматизации математических и научно-технических расчетов. Система Matlab включает в себя собственно пакет Matlab и его расширения: пакет моделирования и анализа динамических систем Simulink; набор дополнительных блоков BlockSet для пакета Simulink; несколько десятков специализированных наборов программных средств – Toolbox – для решения задач в различных областях науки и техники и визуализации вычислений; графические интерактивные средства анализа, проектирования и моделирования динамических систем [1 – 4]. Matlab имеет в своем составе специальные инструментальные средства для взаимодействия с автономными приложениями; обладает обширной и удобной справочной системой в текстовом и гипертекстовом форматах, а также набором электронных pdf-книг.

Вся совокупность этих средств позволяет решать весьма сложные и крупные задачи. Для программирования таких задач обычно требуется разработка множества собственных m-функций. Часто необходимо осуществлять ввод и вывод числовых и текстовых данных, хранящихся в файлах. В большинстве случаев промежуточные и окончательные результаты расчетов следует представлять в графической форме. Инженерные методы расчетов по природе своей являются итерационными, т.е. предполагают неоднократное решение задачи с различными

начальными данными, измененными параметрами, модифицированными алгоритмами и т.п. Поэтому использование мощных средств системы Matlab для решения крупных задач требует, помимо хорошего понимания предметной области, весьма глубоких знаний, как самой системы, так и наличия необходимого опыта работы в ней. Обычному пользователю весьма непросто реализовывать столь сложные вычислительные проекты. В этом смысле на помощь приходят программные средства быстрой визуальной разработки приложений.

2.1. Основы среды разработки GUIDE

Система Matlab имеет такое инструментальное средство – GUIDE (Graphical User Interface Development Environment) – специальная утилита для создания графического интерфейса пользователя [3 – 6]. В результате работы в этой программе пользователь может создавать оконные приложения с типовыми для Windows компонентами управления – кнопками, текстовыми полями ввода/вывода, списками, полосами прокрутки, меню, индикаторами состояния и др. "Носителями" этих компонентов – стандартных окон Windows – выступают графические объекты Matlab типа figure. В таблице 2.1 представлены основные интерфейсные компоненты GUIDE [4].

Таблица 2.1 – Интерфейсные компоненты GUIDE Matlab 6.x

Наименование компонента	Вид	Назначение	Аналог в VCL Delphi
Push Button		Обычная кнопка	Button
Toggle Button		Кнопка, фиксирующаяся в нажатом состоянии	SpeedButton
Radio Button		Переключатель – индикатор альтернатив	RadioButton
Check Box		Флажок – фиксация состояния	CheckBox
Edit Text		Редактируемое однострочное текстовое поле	Edit
Static Text		Текстовое поле для вывода	Label
Slider		Полоса прокрутки (ползунок)	ScrollBar
Frame		Прямоугольный контейнер для компонентов	Panel, Frame
List Box		Список из нескольких строк	ListBox
Popup Menu		Контекстное (всплывающее) меню	PopupMenu
Axes		Изображение графиков	TeeChart

Как и в известных системах визуального программирования, каждый компонент GUIDE имеет набор свойств и методов, определяющих его внешний вид и поведение на стадии выполнения программы.

В Matlab предусмотрены два способа создания и работы оконных приложений – динамический и статический [3–6].

Динамический способ предусматривает формирование графических элементов управления в процессе работы Matlab-программы. Здесь можно провести аналогию с созданием оконных приложений Windows исключительно средствами библиотек WinAPI. Все элементы управления являются объектами дескрипторной графики типа uicontrol и создаются одноименным конструктором. Параметрами функции-конструктора uicontrol определяются владелец элемента управления, его тип, свойства и функция поведения. Возвращает функция описатель (уникальное число) созданного объекта, с помощью которого в программе можно получить доступ к данному объекту интерфейса и, тем самым, динамически его модифицировать.

Статический способ формирования интерфейсной части приложения предполагает стандартную, для систем визуального программирования, процедуру – элементы управления с палитры компонентов GUIDE "перетаскиваются" на форму, размещаются в нужном месте и, через инспектор объектов, получают необходимые значения своих свойств. Таким образом, внешний вид элементов управления можно наблюдать и корректировать до запуска приложения. Облегчается также и написание обработчиков событий – шаблоны обработчиков формируются визуальной средой автоматически.

Статический вариант разработки оконного приложения в системе Matlab обеспечивается утилитой GUIDE, которую можно вызывать либо командой guide из командного окна Matlab, либо через меню File|New|GUI, либо через кнопку Start|Matlab|GUIDE (GUI Builder). Глобальные настройки утилиты производятся в диалоговом окне File|Preferences|GUIDE [3].

После вызова утилиты GUIDE появляется стартовое окно GUIDE Quick Start (рисунок 2.1), в котором имеется две закладки: Create New GUI и Open Existing GUI.

Для создания нового графического приложения пользователь должен выбрать один из четырех предлагаемых вариантов, который ляжет в основу проектируемого интерфейса:

– Blank GUI (Default) – пустую форму без элементов управления;

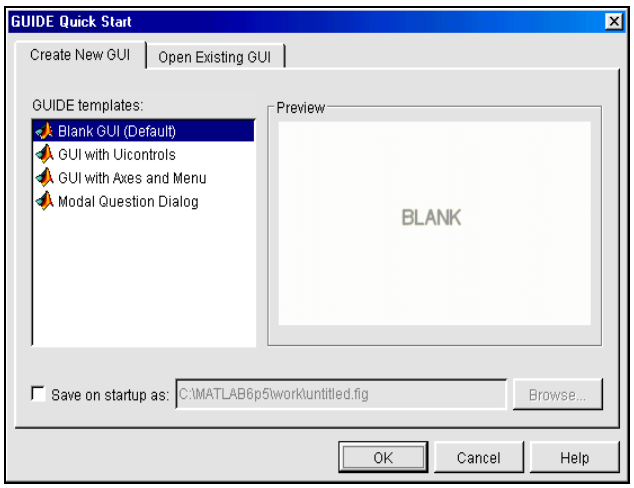


Рисунок 2.1 – Стартовое окно утилиты GUIDE

– GUI with Uicontrols – типовой интерфейс с набором из кнопок, полей ввода, меток, панелей и переключателей. Функционально это приложение рассчитывает массу объекта по введенным пользователем значениям плотности и объема;

– GUI with Axes and Menu – интерфейс с координатными осями и меню. Данное приложение предназначено для построения графика одной из шести функций, которую можно выбрать в меню;

– Modal Question Dialog – модальное диалоговое окно.

На закладке Open Existing GUI пользователь может выбрать уже существующее оконное приложение и продолжить работу над ним.

Если пользователь начинает проектирование нового интерфейса, то ему предоставляется окно среды GUIDE – конструктора графического интерфейса с пустой формой – редактором компоновки (рисунок 2.2). Конструктор содержит: строку главного меню; панель инструментов управления (toolbar); палитру компонентов интерфейса (component palette); пустую заготовку окна приложения (layout editor).

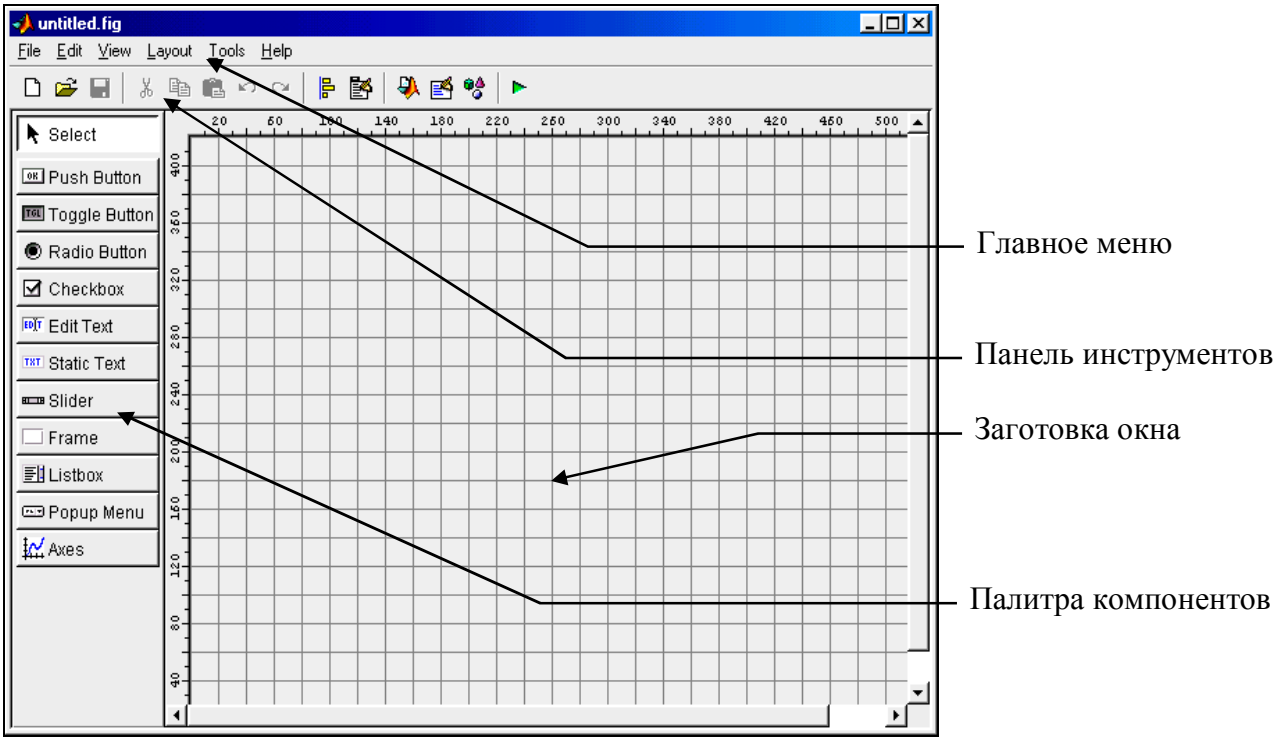


Рисунок 2.2 – Окно конструктора интерфейса с пустой формой

Кнопочная панель инструментов управления включает в себя более десяти кнопок. Первые 8 кнопок – стандартные командные кнопки для работы с файлами и выполнения операций редактирования.

Далее идут кнопки: выравнивание компонентов (Align Objects); редактор меню (Menu Editor); редактор m-файлов (m-file Editor); инспектор свойств объектов (Property Inspector); обозреватель объектов (Object Browser); выполнить GUI-приложение (Run).

Процедура формирования интерфейса приложения состоит в перетаскивании элементов интерфейса с палитры компонентов на поле окна и придания им необходимых свойств и реакций на действия пользователя. Обычно на рабочее поле нанесена настраиваемая вспомогательная сетка для простоты выравнивания компонентов интерфейса по линиям и узлам. Свойства компонентов устанавливаются в окне инспектора свойств (Property Inspector). Двойной щелчок на компоненте приводит к появлению окна инспектора (рисунок 2.3). Окно состоит из двух частей – левая половина содержит список имен свойств, а правая – значения свойств.

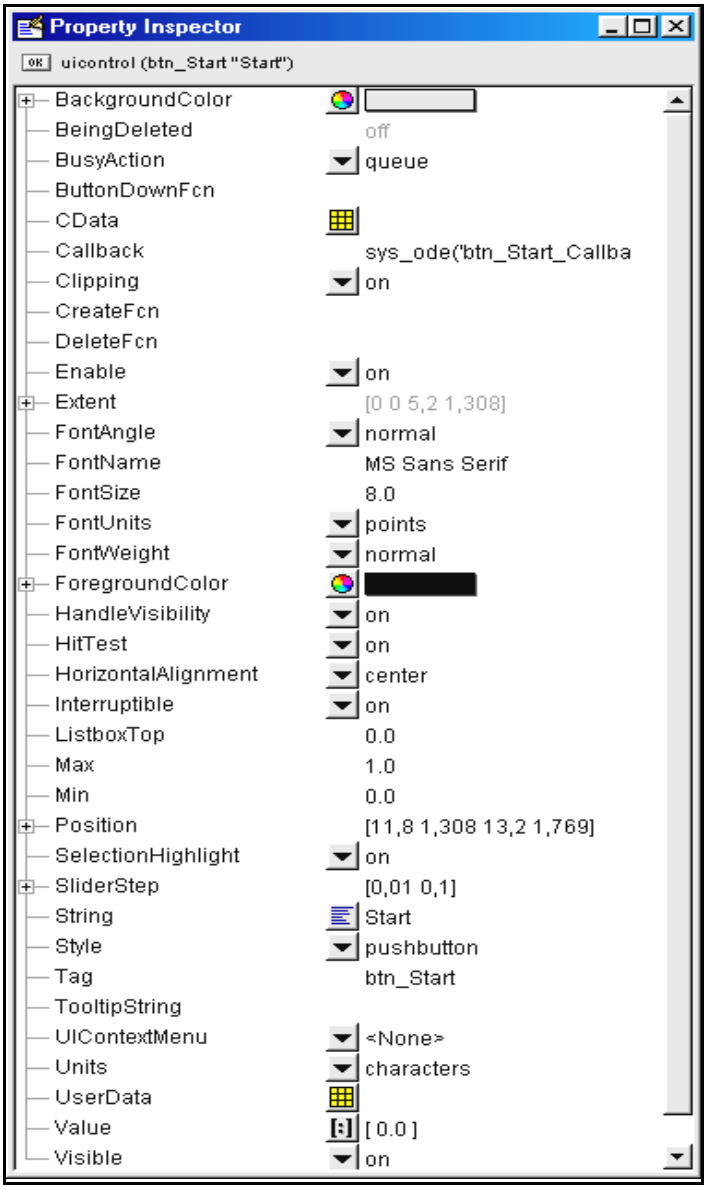


Рисунок 2.3 – Окно инспектора свойств для кнопки Push Button

Утилита GUIDE в процессе работы автоматически формирует два файла – бинарный файл с расширением fig и текстовый с расширением m. Первый файл хранит информацию об интерфейсных компонентах, их положении и размерах, реакциях. Второй – программные реализации реакций компонентов на действия пользователя. Оба файла автоматически редактируются при внесении программистом изменений в интерфейс приложения. В любой момент в процессе проектирования приложение можно запустить на исполнение. При первом запуске необходимо дать имя файлу приложения. Перед первым запуском приложения открывается редактор m-файлов, в котором содержится файл-функция и несколько подфункций.

Среда GUIDE вставляет в текст m-файла множество комментариев, поясняющих назначение функций, подфункций, их параметров и некоторых переменных. Программист на языке Matlab кодирует реакции приложения как отклики компонент на действия пользователя. В списке свойств каждого элемента управления имеется особое свойство Callback. Это свойство процедурного типа – имя функции, тело которой выполняется при возникновении *основного* для данного компонента события. Для кнопок, переключателей и флажков это – щелчок мышью на компоненте; для списков – выбор элемента списка; для полосы прокрутки – перемещение движка и т.п. Для некоторых компонент можно запрограммировать реакции и на другие (*неосновные*) события.

В функцию-обработчик события через ее параметры передаются все необходимые данные: описатель само*го* компонента и структура с описателями всех компонентов интерфейса приложения. Таким образом, в теле функции-обработчика реализован простой доступ к любому компоненту интерфейса.

В Приложении А приведено описание несложного иллюстративного GUIDE-приложения – вычисления и построения собственного движения линейной стационарной САУ.

2.2. Работа с файлами данных в Matlab

Во многих задачах, связанных с проведением сложных и объемных расчетов, необходимо сохранять полученные данные в файлах для дальнейших манипуляций – последующей обработки, визуализации и др. Система Matlab, в этом смысле, обеспечивает работу с двумя типами файлов – бинарными (двоичными) и текстовыми [3, 4].

Бинарные файлы предназначены для хранения произвольных данных в виде последовательности байтов. Сам файл не содержит никакой информации о типе находящихся в нем данных. Интерпретация данных бинарного файла происходит только на этапах операций записи/чтения. Данные бинарных файлов могут быть интерпретированы произвольно. Изменять трактовку данных можно динамически в процессе одной сессии работы с файлом.

Текстовые файлы логически интерпретируются как набор символьных строк, разделенных парами управляющих символов – 0Dh (**CR** Carriage Return) "возврат каретки" и 0Ah (**LF** Line Feed) "перевод строки".

Основные этапы работы с файлами – открытие файла, операции ввода/вывода, закрытие файла. Независимо от типа файла операции открытия и закрытия файлов в Matlab для бинарных и текстовых файлов осуществляются одними и теми же функциями – fopen и fclose.

Спецификация команды открытия файла:

```
hFile = fopen( 'имя_файла' , 'флаг' );
```

hFile – числовой описатель (handle) открытого файла;

имя_файла – имя или полное имя файла;

флаг – атрибут, указывающий возможность дальнейшей работы с файлом. В случае отсутствия данного параметра, файл откроется как двоичный и только для чтения.

Если операция открытия файла прошла удачно, то описатель файла принимает значение целого положительного числа, иначе – значение, равное -1. Для получения дополнительной (поясняющей) информации о неудачной попытке открыть файл необходимо использовать полную спецификацию команды в виде [hFile err_txt] = fopen('имя_файла' , 'флаг'). В переменную err_txt будет помещено текстовое сообщение о причине ошибки при попытке открытия файла.

Второй параметр, если он присутствует, может принимать вид строки из одного, двух или трех символов. В строке атрибутов обязательно должен присутствовать один из трех символов: r – для чтения; w – для записи; a – для дозаписи в конец файла. Следует осторожно обращаться с атрибутом, в котором присутствует символ w (открытие файла для записи) – если открываемый файл уже существует, то его содержимое теряется и создается новый файл.

Наличие знака плюс (+) в строке атрибута открытия файла означает возможность выполнения операций и чтения и записи. Символ t в строке атрибута указывает, что файл открывается как текстовый. Если символа t нет в атрибуте или в нем присутствует символ b, то файл открывается для работы как бинарный (двоичный).

После завершения операций с данными файла его следует закрыть. Для этого служит функция fclose(hFile).

При открытии файла формируется связанный с файлом логический объект – указатель файла. Указатель показывает ("смотрит") на то место файла, откуда будут считаны (куда будут помещены) байты при очередной операции чтения (записи). После выполнения операции считывания/записи указатель перемещается в новое положение ближе к концу файла.

Положение (текущая позиция или координата) указателя определяется как расстояние (смещение) в байтах от начала файла. Значение смещения возвращает функция ftell: ofs = ftell(hFile).
Сразу после открытия файла это смещение равно нулю (ofs=0).

Для повторного чтения данных из файла необходимо установить указатель в начало файла. Это можно сделать, например, так: закрыть файл и открыть его заново. Однако проще воспользоваться функцией frewind, которая возвращает указатель в начало файла: frewind(hFile).

При чтении данных из текстовых или двоичных файлов иногда необходимо знать, достиг ли указатель конца файла (end_of_file). Для этого используют функцию feof: k = feof(hFile).
Если k=0 (false), то данные (байты) между указателем и концом файла еще есть, иначе (при достижении конца файла) – будет k=1 (true).

Произвольные перемещения указателя на определенное количество байтов внутри файла обеспечивает функция fseek. В отличие от аналогичных функций в языках программирования, в системе Matlab функцию fseek можно использовать как с бинарными файлами, так и с текстовыми. Величина перемещения указывается в байтах относительно одной из трех возможных позиций: начала файла (-1 или 'bof'), текущей позиции указателя (0 или 'cof') или конца файла (1 или 'eof').

Например, для открытого файла с описателем hFile:
инструкция flag = fseek(hFile,n,'bof') или, что то же самое, flag = fseek(hFile,n,-1) означает перевод указателя на n (здесь n>0) байтов вперед от начала файла;
инструкция flag = fseek(hFile,n,'cof') или, что то же самое, flag = fseek(hFile,n,0) означает перевод указателя вперед (n>0) или назад (n<0) на n байтов от текущего положения указателя в файле;
инструкция flag = fseek(hFile,n,'eof') или, что то же самое, flag = fseek(hFile,n,1) означает перевод указателя на n (здесь n<0) байтов назад от конца файла.

2.2.1. Запись и чтение двоичных файлов

Операции ввода/вывода данных для двоичных (бинарных) файлов осуществляются с помощью функций fwrite и fread.

Для записи данных в файл, открытый как бинарный с описателем hFile, используется функция fwrite в виде count = fwrite(hFile,A,'precision').

Здесь count – количество записанных в файл элементов данных; A – массив записываемых данных. Данные массива A записываются в файл последовательно по столбцам; 'precision' – символьный параметр, определяющий *тип* элемента записываемого массива (задает количество байтов, необходимое для хранения одного элемента массива A). Следующая таблица представляет возможные значения параметра 'precision'.

Таблица 2.2 – Значения параметра 'precision'

Значение	Числовой тип		Размер в байтах
'uchar' 'uint8'	unsigned integer	целое без знака	1
'schar' 'int8'	signed integer	целое со знаком	1
'int16', 'int32', 'int64'	integer	целое	2, 4, 8
'uint16', 'uint32', 'uint64'	unsigned integer	целое без знака	2, 4, 8
'single' 'float32'	floating point	вещественное	4
'double' 'float64'	floating point	вещественное	8

В принципе, Matlab позволяет рассматривать содержимое файла с точностью до бита, для этого необходимо указывать на месте параметра precision следующие текстовые константы [4]:

– 'bitN' – целое число из N бит со знаком ($1 \le N \le 64$);

– 'ubitN' – целое число из N бит без знака ($1 \le N \le 64$).

Если параметр 'precision' не указан, то по умолчанию тип элемента массива A устанавливается как uint8.

В качестве первого параметра могут быть указаны числовые значения (1 или 2) описателей открытых по умолчанию текстовых файлов для вывода – стандартный вывод (standard output = 1) или стандартный вывод для сообщений об ошибках (standard error = 2). В любом случае вывод осуществляется в командное окно Matlab.

Функция fwrite допускает наличие четвертого целочисленного параметра (n), который обеспечивает запись элементов массива в файл не подряд, а предворяя каждый указанным количеством (n) нулевых байтов.

Для считывания данных из файла, открытого как бинарный с описателем hFile, используется функция fread в виде

```
[A  count] = fread(hFile,[n,m],'precision')
```

Здесь A – имя массива, размерностью n×m, в который помещаются считываемые из файла данные. Массив A заполняется по столбцам; count – количество действительно считанных элементов типа precision из файла. Если данных в файле не хватает для заполнения последнего столбца массива A элементами типа precision, то этот столбец массива дополняется нулями. Для считывания всех элементов файла удобно в конструкции второго параметра использовать константу inf. Например: [inf] или просто inf; [2,inf].

Возможно использовать функцию fread в двух упрощенных формах:

A = fread(hFile) – содержимое файла воспринимается как вектор-столбец из однобайтовых беззнаковых чисел, при этом в массив A данные поступают в вещественном формате double;

A = fread(hFile,n) – в A считывается заданное количество (n) байтов.

Функция fread позволяет проводить считывание данных из бинарного файла с пропусками. Для этого служит четвертый (необязательный) параметр функции, показывающий количество пропускаемых *байтов* файла, например, по команде v=fread(hF,inf,'float32',4) происходит считывание всех (inf) данных файла hF в одномерный массив v вещественных чисел одинарной точности (float32). При этом после

каждого считанного в массив v числа, следующее число (4 байта) пропускается. Если в качестве типа считываемых данных указан битовый ('bitN', 'ubitN'), то четвертый параметр функции означает число пропускаемых при чтении бит. Возможен вариант использования функции fread и для считывания нескольких подряд идущих элементов файла (порций элементов) с последующими пропусками: v=fread(hF,inf,'3*float32',4) – чтение трёх чисел с пропуском последующих двух и так до конца файла (inf).

Следует еще раз напомнить, что считывание происходит с того места, на которое показывает указатель в файле. После считывания указатель перемещается на соответствующее число байтов к концу файла.

2.2.2. Запись и чтение текстовых файлов

Физическая структура текстового файла, как и любого другого файла – последовательность байтов. Логическая структура текстового файла – одномерный массив из символьных строк. Строки в текстовом файле разделяются парой управляющих символов **CR LF** (0Dh 0Ah) "возврат каретки" и "перевод строки".

Запись данных в текстовый файл осуществляется функцией fprintf. Общий формат применения функции:

count=fprintf(hFile,'format',a,b,…).

Здесь hFile – указатель (описатель) открытого для записи файла; 'format' – строка с указанием форматов записываемых данных; a, b, … – записываемые данные в виде переменных или значений; count – возвращает количество записанных в файл байтов.

Первый параметр может также принимать значения 1 и 2, в этом случае вывод (запись) осуществляется в стандартный системный выходной файл – командное окно Matlab. Если первый параметр в функции fprintf отсутствует – данные выводятся в командное окно.

Всё многообразие возможностей вывода данных в текстовый файл определяется структурой и содержанием второго параметра – строкой **форматов**, оговаривающей форму и тип записываемых в файл данных. Строка форматов может содержать как управляющие последовательности символов, так и обычные символы. Управляющие последовательности символов либо задают формат записываемых в файл данных (начинаются с символа **%**), либо вставляют в файл специальные управляющие символы, форматирующие текст (начинаются с символа \).

Перечислим некоторые управляющие последовательности символов:

– для записи в текстовый файл **целочисленных** данных применяются следующие символы с префиксом **%**:

d или i – десятичное целое число со знаком;
o – восьмеричное целое число без знака;
u – десятичное целое число без знака;
x – шестнадцатеричное целое число без знака с малыми буквами a,b,c,d,e,f;
X – шестнадцатеричное целое число без знака с большими буквами A,B,C,D,E,F;

– для записи в текстовый файл **вещественных числовых данных** применяются следующие символы с префиксом **%**:

f – число с фиксированной запятой;
e или E – число с плавающей запятой и признаком порядка – e или E;

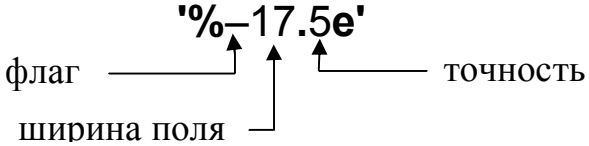
g или G – число с плавающей или фиксированной запятой в зависимости от того, какой формат (f или e) требует меньшего числа байтов для представления числа;

– для записи в текстовый файл **символьных данных** применяются следующие символы с префиксом **%**:

s – запись строки символов;

c – запись одного символа.

Между символом-признаком формата % и одной из перечисленных выше букв могут находится необязательные опции – флаг, ширина поля, точность. Например,



флаг – один из трёх символов: *плюс* – добавление знака плюс перед неотрицательными числами; *минус* – выравнивание выводимых данных по левому краю; **#** - добавление префиксов для восьмеричных (o) или шестнадцатеричных (ox или OX) чисел;

ширина поля – целое число, определяющее общее количество символов, отводимое под представление записываемых данных;

точность – количество значащих цифр после запятой в представлении вещественного числа.

Управляющие последовательности для вставки специальных символов или выполнения специальных символьных операций — Esc–последовательности:

\b – удаление предыдущего символа (BackSpace);
\f – перевод страницы (Form Feed);
\n – переход на новую строку (New Line);
\r – возврат каретки – переход в начало текущей строки (Carriage Return);
\t – вставка символа горизонтальной табуляции (Horizontal Tab);
%% – вставить в строку символов один символ процента %;
\\ – вставить в строку символов один символ обратной наклонной черты \;
' ' – вставить в строку символов один символ одиночной кавычки ';
\" – вставить в строку символов один символ двойной кавычки ".

Чтение данных из текстового файла может быть выполнено с помощью одной из трёх функций: fgetl, fgets и fscanf.

Функция s=fgetl(hFile) предназначена для чтения символов очередной строки (get line) файла (hFile) от позиции указателя до конца строки, исключая чтение символов признака конца строки CR и LF. Возвращается прочитанная строка символов s. Если функция возвращает число минус 1, то это означает, что все данные файла уже прочитаны и перед вызовом функции указатель в файле показывал за последний символ файла.

Функция s=fgets(hFile,n) предназначена для последовательного чтения из файла (hFile) нескольких (n) символов (get symbol), в том числе и символов CR и LF. Чтение начинается с позиции указателя и осуществляется в пределах текущей строки, включая и символы признака конца строки. Попытка прочитать

несуществующие данные за концом файла приводит к возврату функцией числа минус 1.

Когда текстовый файл помимо собственно символьной информации содержит также и числовые данные, разумно использовать функцию fscanf.

Общий вид оператора вызова функции таков:

```
[A,count] = fscanf(hFile,'format',size);
```

Функция запрашивает (читает) из файла hFile данные в количестве size, преобразует эти данные из символьного представления в указанный формат format и помещает их в возвращаемый массив A, а параметр count (необязательный) содержит количество фактически считанных данных. Параметр size может отсутствовать или принимать одно из трёх значений: inf, n или [n,m]. Если параметр size отсутствует или равен inf, то функция fscanf пытается прочитать данные до конца файла. Когда третий параметр функции принимает значение n – считываемые данные располагаются в вектор-столбце A размерности n×1, а когда [n,m] – в матрице A размером [n,m]. В последнем случае число m (но не n) может принимать значение inf.

При считывании из файла числовых данных любых типов (вещественных или целочисленных) в массив A все они помещаются в формате double. Если в указателе формата встречается префикс **%***, то соответствующее значение из файла считывается, однако в возвращаемый массив A не помещается. Эта возможность позволяет пропускать данные при считывании.

Для работы с текстовыми данными и файлами в Matlab существуют еще несколько функций, в частности textread, textscan, sscanf и др. Например, функция textread читает данные из текстового файла, который не нужно открывать.

3. ЗАДАНИЕ НА РАБОТУ

Задание студенту на разработку графического приложения Matlab формулируется, конкретизируется и выдается преподавателем. Приложение должно обладать функционально необходимым и интуитивно понятным интерфейсом и визуально представлять в числовой или графической форме, как исходные данные задачи, так и результаты вычислений.

В лабораторной работе исследуются характеристики линейных стационарных объектов управления, заданных своими математическими моделями:

– в виде передаточной функции $W(s) = \dfrac{b_0 + b_1 s + b_2 s^2 + \ldots + b_m s^m}{a_0 + a_1 s + a_2 s^2 + \ldots + a_n s^n}$; (3.1)

– набором нулей (z_i), полюсов (p_i) и общего коэффициента усиления (k)

$$W(s) = k \frac{(s - z_1)\cdot(s - z_2)\cdot\ldots\cdot(s - z_m)}{(s - p_1)\cdot(s - p_2)\cdot\ldots\cdot(s - p_n)}; \quad (3.2)$$

– в пространстве состояний $\begin{cases} \dfrac{dx(t)}{dt} = Ax(t) + Bu(t), \\ y(t) = Cx(t). \end{cases}$ (3.3)

Примерный перечень заданий

Разработать Matlab-приложение, которое реализует:

1. Построение единичной переходной функции *h(t)* для объекта управления, заданного своей передаточной функцией (3.1);

2. Построение единичной переходной функции *h(t)* для объекта управления, заданного своей передаточной функцией в виде набора нулей, полюсов и общего коэффициента усиления (3.2);

3. Построение единичной переходной функции *h(t)* для объекта управления, заданного моделью в пространстве состояний (3.3) при условии скалярного входа (*u*) и скалярного выхода (*y*);

4. Построение импульсной переходной функции *g(t)* для объекта управления, заданного своей передаточной функцией (3.1);

5. Построение импульсной переходной функции *g(t)* для объекта управления, заданного своей передаточной функцией в виде набора нулей, полюсов и общего коэффициента усиления (3.2);

6. Построение импульсной переходной функции *g(t)* для объекта управления, заданного моделью в пространстве состояний (3.3) при условии скалярного входа (*u*) и скалярного выхода (*y*);

7. Построение собственного движения для объекта управления, заданного своей передаточной функцией (3.1);

8. Построение собственного движения для объекта управления, заданного своей передаточной функцией в виде набора нулей, полюсов и общего коэффициента усиления (3.2);

9. Построение собственного движения для объекта управления, заданного моделью в пространстве состояний (3.3) при условии скалярного входа (*u*) и скалярного выхода (*y*);

10. Построение реакции на произвольное управляющее воздействие при нулевых начальных условиях для объекта управления, заданного своей передаточной функцией (3.1);

11. Построение реакции на произвольное управляющее воздействие при нулевых начальных условиях для объекта управления, заданного своей передаточной функцией в виде набора нулей, полюсов и общего коэффициента усиления (3.2);

12. Построение реакции на произвольное управляющее воздействие при нулевых начальных условиях для объекта управления, заданного моделью в пространстве состояний (3.3) при условии скалярного входа (*u*) и скалярного выхода (*y*);

13. Построение амплитудно-частотной характеристики для объекта управления, заданного своей передаточной функцией (3.1);

14. Построение амплитудно-частотной характеристики для объекта управления, заданного своей передаточной функцией в виде набора нулей, полюсов и общего коэффициента усиления (3.2);

15. Построение амплитудно-частотной характеристики для объекта управления, заданного моделью в пространстве состояний (3.3) при условии скалярного входа (*u*) и скалярного выхода (*y*);

16. Построение частотного годографа Найквиста для объекта управления, заданного своей передаточной функцией (3.1);

17. Построение частотного годографа Найквиста для объекта управления, заданного своей передаточной функцией в виде набора нулей, полюсов и общего коэффициента усиления (3.2);

18. Построение частотного годографа Найквиста для объекта управления, заданного моделью в пространстве состояний (3.3) при условии скалярного входа (u) и скалярного выхода (y);

19. Изображение на комплексной плоскости нулей и полюсов передаточной функции (3.1) объекта управления;

20. Изображение на комплексной плоскости нулей и полюсов передаточной функции (3.2) объекта управления;

21. Изображение на комплексной плоскости собственных чисел матрицы динамических коэффициентов (A) объекта управления (3.3).

Во всех вариантах необходимо предусмотреть файловый ввод/вывод параметров модели исследуемой динамической системы, а также сохранение в файлах результатов расчетов.

4. СОДЕРЖАНИЕ ОТЧЕТА И ПОРЯДОК ЗАЩИТЫ РАБОТЫ

В отчете по лабораторной работе должны быть отражены цель, необходимые сведения о методах решения рассматриваемых задач и средствах системы Matlab, тексты программ, численные и графические результаты расчетов и построений, выводы по работе.

Защита результатов лабораторной работы производится за компьютером при наличии работающей программы и полностью оформленного отчета. В ходе защиты студент может получить дополнительные практические задания по существу работы.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие Вы знаете способы создания графических приложений Matlab?
2. Как формируется интерфейс графического приложения при динамическом способе его создания?
3. Как формируется интерфейс графического приложения при статическом способе его создания?
4. Каковы состав, структура и содержание файлов графического приложения Matlab?
5. Как программируются реакции приложения на действия пользователя?
6. Почему после завершения работы приложения рабочая область Matlab очищается?
7. Каким образом можно оставить в рабочей области Matlab требуемые данные после отработки приложения?
8. С помощью каких компонентов и как можно вводить и выводить числовые матрицы, представленные в их естественном (прямоугольном) виде?
9. Поясните работу с текстовыми файлами в Matlab.
10. Поясните работу с бинарными (двоичными) файлами в Matlab.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Дьяконов В.П. Matlab 6.5 SP1/7 + Simulink 5/6. Основы применения /В.П. Дьяконов. – М.: СОЛОН-Пресс, 2005. – 800 с.
2. Лазарев Ю.Ф. Моделирование процессов и систем в Matlab. Учебный курс /Ю.Ф. Лазарев. – СПб.: Питер; Киев: BHV, 2005. – 512 с.
3. Ануфриев И.Е. Matlab 7 /И.Е. Ануфриев, А.Б. Смирнов, Е.Н. Смирнова. – СПб.: БХВ-Петербург, 2005. – 1104 с.
4. Кетков Ю.Л. Matlab 7: программирование, численные методы /Ю.Л. Кетков, А.Ю. Кетков, М.М. Шульц. – СПб.: БХВ-Петербург, 2005. – 752 с.
5. http://matlab.exponenta.ru/gui/index.php – Приложения с GUI и дескрипторная графика.
6. MATLAB. Creating Graphical User Interface. Version 1. – MathWorks, Inc., 2000. – 180p.

ПРИЛОЖЕНИЕ А
(обязательное)
Пример графического приложения

Ниже приведен пример графического приложения Matlab с подробными комментариями. Назначение приложения – построение собственного движения произвольной устойчивой линейной стационарной системы. С помощью функции rss пакета Control Toolbox формируется устойчивая линейная стационарная системы заданной размерности. В программе жестко заданы начальные условия в виде вектора начального состояния с единичным значением первой компоненты и нулевыми значениями остальных. Собственное движение системы строится на указанном пользователем временн*о*м интервале. В программе предусмотрены возможности изменения толщины линий и управления координатной сеткой на поле графика.

Интерфейс приложения в среде GUIDE представлен на рисунке А.1. Надписи "**Порядок системы**", "**Нач. время**", "**Кон. время**" и "**Толщина графиков**" реализованы с помощью компонентов Static Text. Поля для ввода данных о порядке системы и временн*о*м интервале реализованы компонентами Edit Text. Для выбора вида координатной сетки использованы две компоненты Checkbox, расположенные на компоненте Frame. Управление толщиной линий графика осуществляется с помощью компоненты Popup Menu. Сам процесс вычислений и построений инициируется щелчком на кнопке Start (компонент Push Button). Графики строятся на поле компоненты Axes.

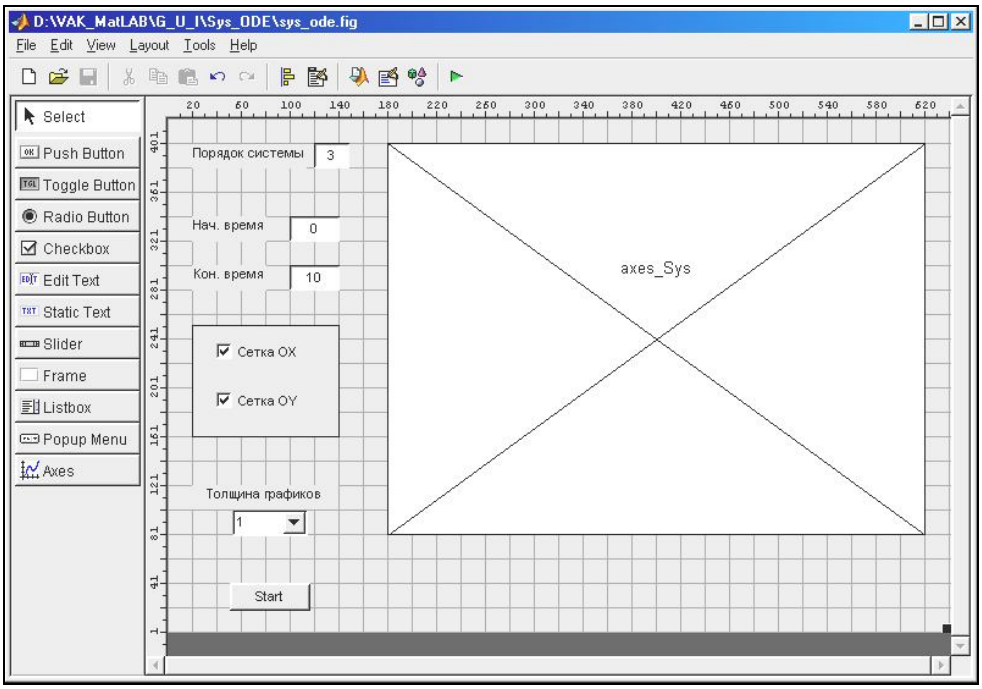


Рисунок А.1 – Интерфейс приложения в среде GUIDE

Таким образом, интерфейс приложения реализован с помощью тринадцати компонентов, из которых четыре обладают индивидуальными реакциями-откликами на основные события. Это флажки для управления координатной сеткой (Checkbox), список выбора толщины линий графика (Popup Menu) и кнопка инициализации вычислений и построений Start.

На рисунках А.2 и А.3 представлены варианты выполнения приложения с различными исходными данными и неодинаковым оформлением.

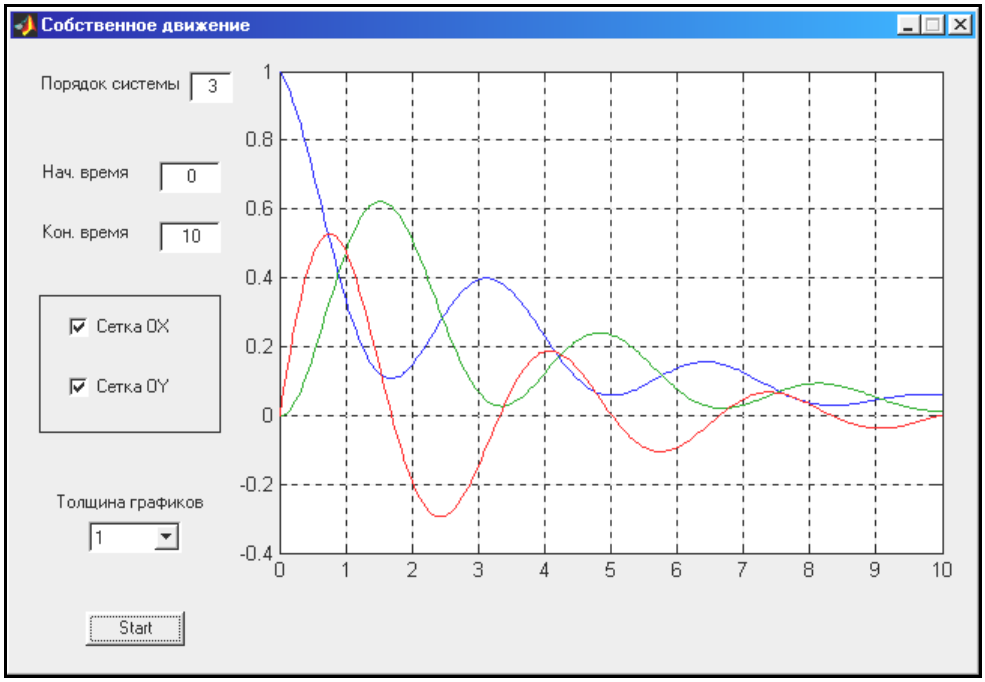


Рисунок А.2 – Результаты работы программы
(порядок системы – 3; толщина графиков – 1; полная сетка)

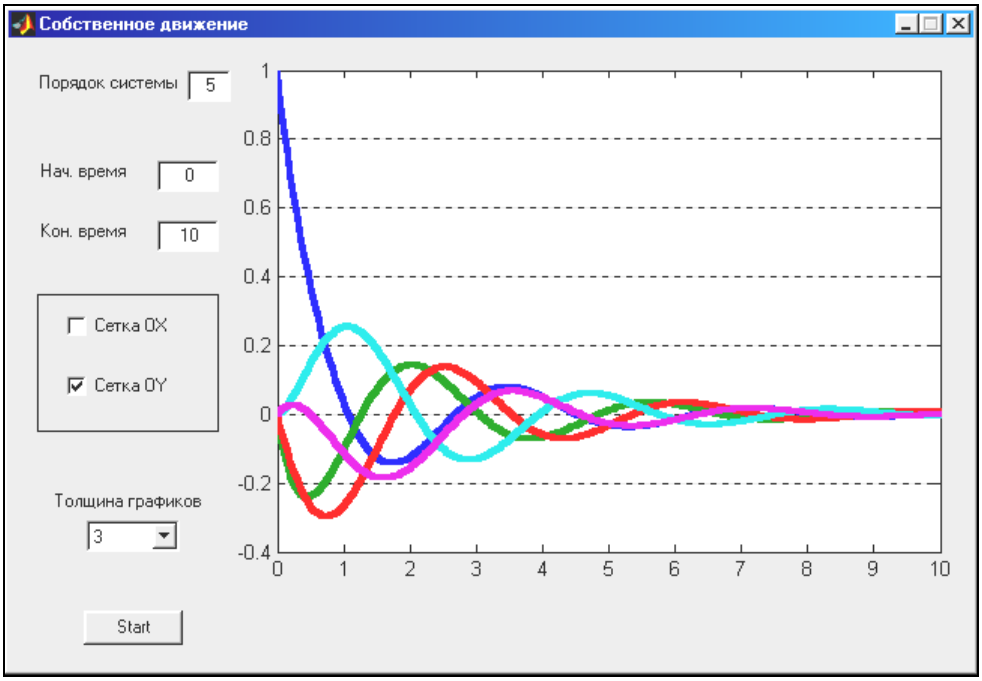


Рисунок А.3 – Результаты работы программы
(порядок системы – 5; толщина графиков – 3: нет сетки по оси OX)

В таблице А.1 приведен практически полный текст, соответствующего данному приложению, m-файла (особенности форматирования наши). Код приложения содержит четырнадцать функций. Заголовок главной функции приведен в первой строке. Строки 2÷19 и 35 – это комментарий среды проектирования. В строках 19 и 35 особо указывается, что тело данной функции программисту редактировать не ре-

комендуется. Вначале каждой функции идут комментарии о назначении параметров и некоторые объяснения и рекомендации, специфические для конкретной функции. В принципе, для сокращения текста, естественно, можно убрать комментарии, и даже функции, не имеющие исполняемых инструкций, например, текст строк 70÷75, 88÷93, 106÷111.

Первые три функции (строки 1, 37 и 50) относятся к самому приложению. Первую (главную) функцию, как мы уже указывали, трогать вообще не надо. Вторая и третья функции редактируются программистом в случае, когда необходимо произвести действия в начале исполнения программы и перед ее завершением.

Функции с заголовками в строках 59, 77, 95 и 164 сформированы средой. Исполняемые инструкции этих функций связаны с традициями системы Windows, а именно – фоновый цвет элементов ввода данных в ОС Windows обычно принимается белым. Поэтому здесь выясняется тип операционной системы и, если это Windows (функция ispc), то устанавливается белый цвет фона полей компонентов ввода данных – однострочных редакторов (59, 77, 95) и полей элементов выпадающего меню (164).

Рассмотрим более детально содержание тех функций, инструкции которых вводились программистом. Это четыре функции – строки 113, 142, 153 и 176. Главная смысловая функция приложения – обработчик основного события для кнопки Start – функция btn_Start_Callback (строка 113). Остальные три функции реализуют оформительские возможности приложения: управление координатной сеткой по осям OX (функция cb_X_Callback, строка 153) и OY (функция cb_Y_Callback, строка 142); управление толщиной линий графика – функция pm_Width_Callback.

Функция btn_Start_Callback

Строки 118 и 119 – считывание текстовых данных о начальном (t0) и конечном (tk) времени моделирования и преобразование их в числовые значения вещественного типа double. В строке 120 производятся аналогичные действия и, вдобавок, полученное число преобразуется в целое – dim_X – порядок моделируемой системы. В строках 121÷124 задается шаг интегрирования (step_ODE), формируются интервал интегрирования (t) и вектор начального состояния системы (x0).

В строке 125 с помощью функции rss производится формирование устойчивой линейной стационарной системы (sys) соответствующего порядка и структуры. Затем (строка 126) вычисляется (initial) реакция системы (sys) на ненулевые начальные условия (x0) на заданном интервале (t). Полученные данные сохраняются в массивах Y, T, X.

Далее (строки 127÷140) производится построение графиков и оформление координатной плоскости в зависимости от текущих установок.

Строка 127 – очистка (инициализация) осей поля графиков.

Строки 128 и 129 – построение графиков компонент вектора состояния (plot(T,X)) и запоминание массива указателей на линии графиков в структуре handles. На эту строку кода следует обратить особое внимание. Здесь реализованы сразу несколько стандартных приемов программирования в Matlab. Во-первых, нам необходимо передать описатели линий графиков в наши специальные функции cb_X_Callback и cb_Y_Callback для возможного изменения толщины этих линий.

Передача возможна только через параметры функций. Это достигается через добавление поля line в структуре handles, которая передается как параметр в любую функцию приложения. В строке 129 производится сохранение обновленной структуры, что и позволяет использовать ее в дальнейшем в других функциях. Во-вторых, здесь использовано уникальное свойство функции plot – возвращение по умолчанию указателей на построенные линии графиков. Если строится сразу несколько линий (у нас это величина размерности системы dim_X), то функция plot возвращает массив указателей на эти линии. Этот факт проявляется в строках 184, 186, 188, 190 и 192 функции pm_Width_Callback, где толщина линий изменяется сразу для всех – оператор двоеточие (:).

Строка 130 – установка *первого* поля компонента выбора толщины линии pm_Width в активное состояние.

Строки 131÷140 – нанесение координатной сетки на поле графика в соответствии с текущими настройками пользователя. Здесь идентификатор gca (graphic current axis) указывает, что мы работаем с текущими осями.

Функции cb_X_Callback и cb_Y_Callback

Эти функции предназначены для отработки событий, связанных с действиями пользователя по управлению видом координатной сетки графика. Строки 147÷151 и 158÷162 в соответствующих функциях обрабатывают основные события компонент Checkbox – снятие и установку флажков ☑.

Свойство Value у компоненты Checkbox принимает два значения 1 и 0 – соответственно флажок установлен или снят. Поэтому в конструкции if использована функция get(hObject,'Value'), которая и возвращает значение данного свойства компонента. Идентификатор hObject в текущей функции указывает на родительский компонент, в данном случае на компонент Checkbox.

Функции pm_Width_Callback

Здесь запрограммирована реакция приложения на действия пользователя с управляющим элементом Popup Menu. Выбор пользователем определенного поля меню приводит к изменению толщины линий на графике. Так как вариантов (полей меню) пять, то для программирования соответствующего действия выбрана конструкция языка switch – многоальтернативный выбор (строки 182÷193). Каждая альтернатива связана с установкой определенной толщины линий графика. Изменение толщины осуществляется одновременно для всех линий. Используется функция set, которая устанавливает всем линиям handles.line(:) толщину LineWidth.

Таблица А.1 – Исходный код графического приложения

№	Текст m-файла графического приложения
1	**function** varargout = sys_ode(varargin)
2	*% SYS_ODE M-file for sys_ode.fig*
3	*% SYS_ODE, by itself, creates a new SYS_ODE or raises the existing*
4	*% singleton*.*
5	*% H = SYS_ODE returns the handle to a new SYS_ODE or the handle to the*
6	*% existing singleton*.*
7	*% SYS_ODE('CALLBACK',hObject,eventData,handles,...) calls the local*
8	*% function named CALLBACK in SYS_ODE.M with the given input arguments.*

№	Текст m-файла графического приложения
9	% SYS_ODE('Property','Value',...) creates a new SYS_ODE or raises the
10	% existing singleton*. Starting from the left, property value pairs are
11	% applied to the GUI before sys_ode_OpeningFunction gets called. An
12	% unrecognized property name or invalid value makes property application
13	% stop. All inputs are passed to sys_ode_OpeningFcn via varargin.
14	% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
15	% instance to run (singleton)".
16	% See also: GUIDE, GUIDATA, GUIHANDLES
17	% Edit the above text to modify the response to help sys_ode
18	% Last Modified by GUIDE v2.5 05-Aug-2006 22:38:46
19	% **Begin initialization code - DO NOT EDIT**
20	gui_Singleton = 1;
21	gui_State = struct('gui_Name', mfilename, ...
22	'gui_Singleton', gui_Singleton, ...
23	'gui_OpeningFcn', @sys_ode_OpeningFcn, ...
24	'gui_OutputFcn', @sys_ode_OutputFcn, ...
25	'gui_LayoutFcn', [] , ...
26	'gui_Callback', []);
27	**if** nargin & isstr(varargin{1})
28	gui_State.gui_Callback = str2func(varargin{1});
29	**end**
30	**if** nargout
31	[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
32	**else**
33	gui_mainfcn(gui_State, varargin{:});
34	**end**
35	% **End initialization code - DO NOT EDIT**
36	% --- Executes just before sys_ode is made visible.
37	**function** sys_ode_OpeningFcn(hObject, eventdata, handles, varargin)
38	% This function has no output args, see OutputFcn.
39	% hObject handle to figure
40	% eventdata reserved - to be defined in a future version of MATLAB
41	% handles structure with handles and user data (see GUIDATA)
42	% varargin command line arguments to sys_ode (see VARARGIN)
43	% Choose default command line output for sys_ode
44	handles.output = hObject;
45	% Update handles structure
46	guidata(hObject, handles);
47	% UIWAIT makes sys_ode wait for user response (see UIRESUME)
48	% uiwait(handles.main_window);
49	% --- Outputs from this function are returned to the command line.
50	**function** varargout = sys_ode_OutputFcn(hObject, eventdata, handles)
51	% varargout cell array for returning output args (see VARARGOUT);
52	% hObject handle to figure
53	% eventdata reserved - to be defined in a future version of MATLAB
54	% handles structure with handles and user data (see GUIDATA)
55	% Get default command line output from handles structure
56	varargout{1} = handles.output;
57	
58	% --- Executes during object creation, after setting all properties.
59	**function** ed_Dim_Sys_CreateFcn(hObject, eventdata, handles)
60	% hObject handle to ed_Dim_Sys (see GCBO)
61	% eventdata reserved - to be defined in a future version of MATLAB

№	Текст m-файла графического приложения
62	% handles empty - handles not created until after all CreateFcns called
63	% Hint: edit controls usually have a white background on Windows.
64	% See ISPC and COMPUTER.
65	if ispc
66	set(hObject,'BackgroundColor','white');
67	else
68	set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
69	end
70	**function** ed_Dim_Sys_Callback(hObject, eventdata, handles)
71	% hObject handle to ed_Dim_Sys (see GCBO)
72	% eventdata reserved - to be defined in a future version of MATLAB
73	% handles structure with handles and user data (see GUIDATA)
74	% Hints: get(hObject,'String') returns contents of ed_Dim_Sys as text
75	% str2double(get(hObject,'String')) returns contents of ed_Dim_Sys as a double
76	% --- Executes during object creation, after setting all properties.
77	**function** ed_Time_beg_CreateFcn(hObject, eventdata, handles)
78	% hObject handle to ed_Time_beg (see GCBO)
79	% eventdata reserved - to be defined in a future version of MATLAB
80	% handles empty - handles not created until after all CreateFcns called
81	% Hint: edit controls usually have a white background on Windows.
82	% See ISPC and COMPUTER.
83	if ispc
84	set(hObject,'BackgroundColor','white');
85	else
86	set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
87	end
88	**function** ed_Time_beg_Callback(hObject, eventdata, handles)
89	% hObject handle to ed_Time_beg (see GCBO)
90	% eventdata reserved - to be defined in a future version of MATLAB
91	% handles structure with handles and user data (see GUIDATA)
92	% Hints: get(hObject,'String') returns contents of ed_Time_beg as text
93	% str2double(get(hObject,'String')) returns contents of ed_Time_beg as a double
94	% --- Executes during object creation, after setting all properties.
95	**function** ed_Time_end_CreateFcn(hObject, eventdata, handles)
96	% hObject handle to ed_Time_end (see GCBO)
97	% eventdata reserved - to be defined in a future version of MATLAB
98	% handles empty - handles not created until after all CreateFcns called
99	% Hint: edit controls usually have a white background on Windows.
100	% See ISPC and COMPUTER.
101	if ispc
102	set(hObject,'BackgroundColor','white');
103	else
104	set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
105	end
106	**function** ed_Time_end_Callback(hObject, eventdata, handles)
107	% hObject handle to ed_Time_end (see GCBO)
108	% eventdata reserved - to be defined in a future version of MATLAB
109	% handles structure with handles and user data (see GUIDATA)
110	% Hints: get(hObject,'String') returns contents of ed_Time_end as text

№	Текст m-файла графического приложения
111	`%  str2double(get(hObject,'String')) returns contents of ed_Time_end as a double`
112	`% --- Executes on button press in btn_Start.`
113	**`function`** `btn_Start_Callback(hObject, eventdata, handles)`
114	`% hObject    handle to btn_Start (see GCBO)`
115	`% eventdata  reserved - to be defined in a future version of MATLAB`
116	`% handles    structure with handles and user data (see GUIDATA)`
117	`clc;`
118	`t0=str2double(get(handles.ed_Time_beg,'String'));`
119	`tk=str2double(get(handles.ed_Time_end,'String'));`
120	`dim_X=round(str2double(get(handles.ed_Dim_Sys,'String')));`
121	`step_ODE=(tk-t0)/250;`
122	`t=t0:step_ODE:tk;`
123	`x0=zeros(dim_X,1);`
124	`x0(1)=1;`
125	`sys=rss(dim_X);`
126	`[Y,T,X]=initial(sys,x0,t);`
127	`cla;`
128	`handles.line=plot(T,X);`
129	`guidata(gcbo,handles);`
130	`set(handles.pm_Width,'Value',1);`
131	**`if`** `get(handles.cb_X,'Value')`
132	`    set(gca,'XGrid','on')`
133	**`else`**
134	`    set(gca,'XGrid','off')`
135	**`end`**
136	**`if`** `get(handles.cb_Y,'Value')`
137	`    set(gca,'YGrid','on')`
138	**`else`**
139	`    set(gca,'YGrid','off')`
140	**`end`**
141	`% --- Executes on button press in cb_Y.`
142	**`function`** `cb_Y_Callback(hObject, eventdata, handles)`
143	`% hObject    handle to cb_Y (see GCBO)`
144	`% eventdata  reserved - to be defined in a future version of MATLAB`
145	`% handles    structure with handles and user data (see GUIDATA)`
146	`% Hint: get(hObject,'Value') returns toggle state of cb_Y`
147	**`if`** `get(hObject,'Value')`
148	`    set(gca,'YGrid','on')`
149	**`else`**
150	`    set(gca,'YGrid','off')`
151	**`end`**
152	`% --- Executes on button press in cb_X.`
153	**`function`** `cb_X_Callback(hObject, eventdata, handles)`
154	`% hObject    handle to cb_X (see GCBO)`
155	`% eventdata  reserved - to be defined in a future version of MATLAB`
156	`% handles    structure with handles and user data (see GUIDATA)`
157	`% Hint: get(hObject,'Value') returns toggle state of cb_X`
158	**`if`** `get(hObject,'Value')`
159	`    set(gca,'XGrid','on')`
160	**`else`**
161	`    set(gca,'XGrid','off')`
162	**`end`**

№	Текст m-файла графического приложения
163	`% --- Executes during object creation, after setting all properties.`
164	**`function`** `pm_Width_CreateFcn(hObject, eventdata, handles)`
165	`% hObject    handle to pm_Width (see GCBO)`
166	`% eventdata  reserved - to be defined in a future version of MATLAB`
167	`% handles    empty - handles not created until after all CreateFcns called`
168	`% Hint: popupmenu controls usually have a white background on Windows.`
169	`%       See ISPC and COMPUTER.`
170	**`if`** `ispc`
171	`    set(hObject,'BackgroundColor','white');`
172	**`else`**
173	`    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));`
174	**`end`**
175	`% --- Executes on selection change in pm_Width.`
176	**`function`** `pm_Width_Callback(hObject, eventdata, handles)`
177	`% hObject    handle to pm_Width (see GCBO)`
178	`% eventdata  reserved - to be defined in a future version of MATLAB`
179	`% handles    structure with handles and user data (see GUIDATA)`
180	`% Hints: contents = get(hObject,'String') returns pm_Width contents as cell array`
181	`%  contents{get(hObject,'Value')} returns selected item from pm_Width`
182	**`switch`** `get(hObject,'Value')`
183	`    case 1`
184	`        set(handles.line(:),'LineWidth',1);`
185	`    case 2`
186	`        set(handles.line(:),'LineWidth',2);`
187	`    case 3`
188	`        set(handles.line(:),'LineWidth',3);`
189	`    case 4`
190	`        set(handles.line(:),'LineWidth',4);`
191	`    case 5`
192	`        set(handles.line(:),'LineWidth',5);`
193	**`end`**`;`

Заказ №_____ от "____" _____ 200__г. Тираж _____ экз.

Изд-во СевНТУ