Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Промышленная электроника»

ОСНОВЫ ТЕОРИИ СИСТЕМ

ПРАКТИКУМ
по выполнению лабораторных работ
по одноименной дисциплине
для студентов специальности 1-53 01 07
«Информационные технологии и управление
в технических системах»
дневной формы обучения

Гомель 2020

УДК 681(075.8)
ББК 32.81я73
О-75

Рекомендовано научно-методическим советом автоматизированных и информационных систем ГГТУ им. П. О. Сухого (*протокол № 10 от 03.06.2019 г.*)

Рецензент: доц. каф. «Автоматизированный электропривод» ГГТУ им. П. О. Сухого канд. техн. наук, доц. *В. С. Захаренко*

О-75 **Основы** теории систем : практикум по выполнению лаборатор. работ по одноим. дисциплине для студентов специальности 1-53 01 07 «Информационные технологии и управление в технических системах» днев. формы обучения / сост.: Э. М. Виноградов, В. А. Хананов. – Гомель : ГГТУ им. П. О. Сухого, 2019. – 197 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

Содержит 13 лабораторных работ с порядком их выполнения, основными теоретическими сведениями, заданиями для самостоятельной работы и контрольными вопросами.

Для студентов специальности 1-53 01 07 «Информационные технологии и управление в технических системах» дневной формы обучения.

УДК 681(075.8)
ББК 32.81я73

Введение

Понятие «система» является одним из ключевых определений в общеобразовательной программе подготовки инженеров по информационным технологиям и управлению. Рассматривая систему как совокупность взаимодействующих объектов, можно широко использовать преимущества структурного отображения исследуемых явлений и процессов для их математического описания и моделирования на компьютерах.

Современные вычислительные средства позволяют без особого труда и затрат времени решать сложные задачи по исследованию и расчету систем управления не инженерными (адаптированными) методами, а с использованием математических аппаратов любой степени сложности. При этом не требуется помощь программиста для реализации методов и визуализации проводимых исследований.

Примером могут служить программные пакеты MathCad, Mathematica, Maple, Matlab, которые используются студентами, учеными и инженерами во всем мире. Названные коммерческие программы универсальны и рассчитаны на любых пользователей. К сожалению, они имеют существенный недостаток – слишком большая цена за одну версию программы. Однако существуют и свободно распространяемые (некоммерческие) программные пакеты для подобных целей. Примером может служить программный пакет Scilab.

Свободно распространяемый программный пакет Scilab был разработан во Франции, в Национальном исследовательском институте информатики и автоматизации INRIA. В настоящее время он распространяется через Интернет. Последнюю версию пакета всегда можно скачать на официальном сайте программы www.scilab.org.

Scilab позволяет производить численные вычисления с различными функциями, имеет средства для построения графиков. В состав пакета входит инструмент для визуального исследования различных устройств. Scilab имеет свой язык программирования, встроенный редактор текста и средства для отладки программ. Поэтому по современной терминологии можно определить Scilab как интегрированную среду разработки для выполнения инженерных расчетов и научных исследований.

В представленном пособии последовательно изложены функции программного пакета Scilab, которые применяются для проведения моделирования и анализа различных систем, с примерами их использования. Во второй части пособия, для моделирования и исследования систем автоматического регулирования вводится понятие структурных блоков, схем и их описания с помощью полиномиальных передаточных функций. Так же описываются возможности программного пакета Scilab для создания пользовательского интерфейса программы исследования характеристик систем регулирования.

Лабораторная работа № 1

НАЧАЛО РАБОТЫ В СРЕДЕ РАЗРАБОТКИ SCILAB

1 Цель работы

Ознакомиться с интерфейсом интегрированной среды разработки Scilab. Исследовать работу Scilab в режиме калькулятора. Изучить и исследовать выполнение математических вычислений со скалярами. Научиться создавать простейшие скрипт-файлы.

2 Порядок выполнения работы

2.1 Создание папки для работы

2.1.1. Для выполнения лабораторных работ Вам необходимо создать на компьютере свою рабочую папку. Сделать это можно следующим образом.

На панели TotCom выберите диск F. Затем выберите папку ОТС и раскройте ее. Внутри нее создайте новую папку (с помощью клавиши F7) с именем, соответствующим вашей фамилии (буквы обязательно латинские), например: Ivanov.

Примечание. При использовании в именах папок русских букв возможна неправильная работа исследуемых программ.

В дальнейшем Вы будете записывать и хранить в вашей папке все файлы в процессе выполнения лабораторных работ. Полный путь к ней:

F:\OTC\Ivanov

2.1.2. При выполнении лабораторных работ Вы будете создавать много различных файлов. Чтобы было легче ориентироваться в них, желательно для каждой лабораторной работы иметь свою отдельную папку. Допустим, что для выполнения лабораторной работы № 1 мы будем использовать папку с именем Lab1, которую необходимо предварительно создать. С этой целью откройте вашу рабочую папку (с именем, соответствующим вашей фамилии) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab1.

В дальнейшем Вы будете записывать и хранить в этой папке все файлы в процессе работы с Matlab. Полный путь к ней:

F:\OTC\Ivanov\Lab1

2.2 Запуск и настройка среды разработки Scilab

2.2.1. Запустить Scilab можно щелчком по кнопке мыши, поместив курсор на ярлык Scilab на рабочем столе компьютера. После запуска программы вы увидите на экране рабочий стол среды разработки Scilab, который имеет следующие основные окна:

1) Обозреватель файлов, в котором отображается список файлов и вложенных папок активного в данный момент каталога;

2) Командное окно, является основным окном, в котором вводятся исполняемые команды и отображаются результаты вычислений;

3) Обозреватель переменных служит для отображения переменных, которые используются в данном сеансе работы;

4) Журнал команд используется для отображения содержимого буфера, в котором хранятся выполненные ранее команды.

2.2.2. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа рабочий стол среды разработки Scilab.

2.2.3. Необходимо установить в качестве активного (текущего) каталог (папку) Lab1. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка: «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем ОТС, потом папку с вашей фамилией (Ivanov) и затем папку Lab1. В строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab1. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab1

2.2.4. При необходимости очистите командное окно и журнал команд, используя пункты меню Правка.

2.3 Исследование работы Scilab в режиме калькулятора

2.3.1. **Задание**. Наберите в командном окне следующие выражения. Для выполнения нажмите клавишу Enter.

1)

--> 7 + 8 / 2

2)

--> 27 ^ (1 / 3) + 32 ^ 0.2

3) Если вычисляемое выражение слишком длинное, то перед нажатием клавиши Enter следует набрать две или более точек. Это будет означать продолжение в командной строке:

--> 1 + 2 + 3 + 4 + ..

--> 5 + 6 + 7

4) Текстовый комментарий в Scilab – это строка, начинающаяся с символов // (как в языке Си):

--> // This is my comment

--> // Это мой комментарий

5) Если символ точки с запятой «;» указан в конце выражения, то результат вычислений не выводится, а активизируется следующая командная строка:

- - > 1 + 2;

- - > 1 + 2

2.3.2. **Задание**. Требуется подсчитать выражение:

$$\left(3^{-1}\cdot\left(\frac{1}{2}\right)^{-2}-27^{-1/3}\right)\cdot\left(3\frac{2}{5}-0{,}9\right)$$

Наберите в командной строке соответствующее выражение (по правилам Scilab) и вычислите его.

2.4 Переменные в Scilab

В среде разработки Scilab любая переменная до использования в формулах и выражениях должна быть определена. Для определения переменной необходимо набрать имя переменной, потом символ «=», и затем значение переменной. Здесь знак равенства – это оператор присваивания. То есть, если в общем виде оператор присваивания записать как

имя_переменной = значение_выражения,

то в переменную, имя которой указано слева. будет записано значение выражения, указанного справа. Выражение в правой части оператора присваивания может быть числом, арифметическим выражением, строкой символов или символьным выражением.

2.4.1. **Задание**. Выполните следующие примеры определения переменных.

1) Присваивание значений переменным x и y:

```
--> x = 2.3
--> y = -34.7
```

Обратите внимание, что имена переменных x и y, а также их значения появились в окне обозревателя переменных на рабочем столе Scilab.

2) Для просмотра значения любой переменной из текущего рабочего пространства можно набрать ее имя и нажать клавишу Enter. Проверим, существуют ли в рабочем пространстве в данной сессии (сеансе работы) переменные с именами x, y и z:

```
--> x
--> y
--> z
```

Обратите внимание на сообщении об ошибке – переменная z не определена.

3) Определим переменные u и v, присвоим им значения, и вычислим значение переменной w:

```
--> u = 1;  v = 2;
--> w = (u + v) – u / v
```

Обратите внимание, что имя переменной w появилось в окне обозревателя переменных.

2.4.2. Когда отпадает необходимость в хранении ряда переменных в текущем сеансе работы, их можно удалить из памяти компьютера. Для очистки рабочего пространства используется команда clear в разных формах, например:

clear x – уничтожение переменной с именем x;
clear a b c – уничтожение переменных с именами a, b, c;
clear – уничтожение всех переменных.

Задание. Выполните команду clear в различных формах. Обратите внимание, как имена переменных исчезают из окна обозревателя переменных.

1) Удалить переменную y:
--> clear y;
2) Удалить переменные x и z:
--> clear x z;
3) Очистить текущее рабочее пространство:
--> clear;
4) Проверим, имеются ли в рабочем пространстве переменные с именами:
--> x
--> u
Сделайте выводы по результатам выполнения команд.

2.4.3. Среда разработки Scilab имеет несколько предопределенных переменных и констант, которые называются системными.

Основные системные переменные и константы:

ans – (от англ. answer – ответ) – результат вычисления, имя которого не определено пользователем;
%e – число e (основание натурального логарифма);
%pi – число π;
%i – мнимая единица ($\sqrt{-1}$);
%eps – условный нуль, равный 2.220D-16;
%inf – «бесконечность»;

%nan – «не число» (от англ. not a number);
%t или %T – логическая единица, «истина» (от англ. true);
%f или %F – логический нуль, «ложь» (от англ. false).

Задание. Исследуйте системную переменную (константу) %pi.
1) Введите в командную строку;
--> %pi
2) Вычислите выражение:
--> y = 2 * %pi / 6
3) Вычислите выражение:
--> z = pi / 8
Сделайте выводы по использованию системной константы %pi в выражениях.

2.5 Управление форматом отображения чисел

2.5.1. По умолчанию в Scilab в качестве результата выводится только 10 символов цифр (включая знак числа и десятичную точку).
Задание. Выведите значение системной константы %pi
--> %pi
Подсчитайте количество значащих цифр.

2.5.2. Можно изменить формат отображения выводимых чисел с помощью команды format(n), где n–количество выводимых символов.
Задание. Выполните вывод числа π с 20 значащими цифрами:
--> format(20)
--> %pi

2.5.3. Заданный в команде format(n) вид отображения чисел сохраняется до окончания текущей сессии (сеанса работы). Для возврата к первоначальному формату необходимо исполнить команду format(10).
Задание. Проверьте формат вывода числа e, а затем вернитесь к формату вывода по умолчанию:
--> %e
--> format(10)
--> %pi

2.6 Выполнение математических вычислений со скалярами

Таблица 1 – Основные элементарные математические функции Scilab

Запись в Matlab	Запись в математике
sqrt(x)	$\sqrt{x}$ - корень квадратный из x
abs(x)	|x| - абсолютная величина
exp(x)	e^x - экспонента
log(x)	ln x – логарифм натуральный
log10(x)	lg x – логарифм десятичный
cos(x), sin(x), tan(x), cot(x), x – в радианах	cos x, sin x, tg x, ctg x
cosd(x), sind(x), tand(x), cotd(x), x – в градусах	
acos(x), asin(x), tan(x), acot(x) – x в радианах	arccos x, arcsin x, arctg x, arcctg x
acosd(x), asind(x), atand(x),acot(x) – x в градусах	
round(x)	Округляет x до ближайшего целого
factorial(x)	x! – выдает факториал числа (x – целое, положительное)

2.6.1. **Задание**. Для вычисления выражения

$$z = \sin\left(\frac{25\pi}{6}\right) - \cos\left(-\frac{\pi}{3}\right)$$

наберите в командной строке и нажмите Enter:

$$z = \sin(25 * \%pi / 6) - \cos(-\%pi / 3)$$

2.6.2. **Задание**. Вычислите выражение

$$y = \frac{\cos x^2}{x + \sin^3 x} + e^{-2,1}$$

при значении x = 1,46.

2.6.3. **Задание**. Вычислите выражение

$$w = \frac{1{,}2 - 2x}{\lg(x + 3{,}2)} - \sqrt{|x - 5{,}8|}$$

при значении x = 0,478.

2.7 Операции с комплексными числами

В Scilab системная переменная %i является мнимой единицей $\sqrt{-1}$.

Комплексное число z можно записать в алгебраической форме

$$z = \mathrm{Re}(z) + \mathrm{Im}(z)i$$

или в экспоненциальной форме

$$z = Ze^{i\arg(z)},$$

где Re(z) – действительная часть числа;

Im(z) – комплексная часть числа;

Z – модуль числа;

arg(z) – аргумент числа, который определяется соотношением:

$$\arg(z) = arctg\frac{\mathrm{Im}(z)}{\mathrm{Re}(z)}.$$

Для записи комплексного числа в алгебраической форме надо ввести в командную строку:

--> x = 5 + 6 * %pi

или

--> y = 10 - %i * 5

Таблица 2 – Функции комплексного переменного в Scilab

Функция	Описание
abs(x)	Вычисляет модуль комплексного числа x
conj(x)	Возвращает число комплексно-сопряженное с x
imag(x)	Выдает мнимую часть числа x
real(x)	Выдает действительную часть числа

2.7.1. **Задание.** Выполните следующие примеры работы с комплексными числами.

1) Значения переменной i

--> %i

……

--> i

……....

2) Задайте комплексные числа: $a = 1 + i, b = 2 - 3i$

……....

3) Вычислите сумму комплексных чисел: $a + b$

……..

4) Вычислите произведение комплексных чисел: $a * b$

……

5) Вычислите модуль комплексного числа a

……....

6) Вычислите действительную и мнимую части комплексного числа: Re(b) и Im(b)

……

7) Вычислите аргумент комплексного числа b (через arctg)

……

8) Вычислите число, комплексно сопряженное числу b:

……..

9) Вычислите синус комплексного числа a

……

2.8 Ведение дневника сессии

В Scilab имеется возможность сохранить в файле всего текста сессии (содержимого командного окна), т.е. сеанса работы в Scilab. Для этого используется специальные команды для ведения дневника сессии:

diary('file_name') – ведет запись на диск всех команд в строках ввода и полученных результатов в виде текстового файла с указанием имени. Файл с именем file_name – это обычный текстовый файл (не исполняемый). Обычно используется встроенный в Windows редактор типа Блокнот. Для этого файл должен иметь расширение .txt. Запись файла ведется в папку (директорию), которая в данный момент является текущей;

diary(0) – прекращает запись в файл.

2.8.1. **Задание.** Выполните сохранение нескольких команд сессии в текстовом файле с именем primer.txt в папке Lab1, которая в данный момент является текущей. Для этого выполните следующие команды:

-- > diary(primer.txt)

-- > x = 1 + 2

*-- > y = x*5*

diary(0)

Обратите внимание, что после выполнения команды diary(‘primer.txt’) в командном окне появляется сообщение ans = 1. Это значение идентификатора id, которое программа присвоила файлу primer.txt.

Чтобы прочитать содержимое дневника, щелкните мышью по надписи Lab1 в окне обозревателя файлов. Должно появиться название файла дневника primer.txt. Щелкните дважды по этому имени. Должно раскрыться окно блокнота с текстом –содержимым командного окна Scilab.

Убедитесь, что все заданные команды были записаны.

2.8.2. Иногда требуется записывать в дневник не все команды сессии, а только некоторые из них. Для управления процессом записи в текстовый файл Scilab имеет специальные команды вида:

diary(id, ‘off’) или diary(‘file_name’, ‘off’) – приостанавливают запись в файл;

diary(id, ‘on’) или diary(‘file_name’, ‘on’) – вновь начинает запись в файл.

Здесь параметр id – это идентификатор файла file_name.

Таким образом, чередуя команды diary(‘off’) и diary(‘on’), можно сохранять нужные фрагменты сессии в их формальном виде.

Задание. Выполните сохранение нескольких команд сессии в текстовом файле с именем primer_2.txt в папке Lab1, которая в данный момент является текущей. Для этого выполните следующие команды:

-- > diary(primer_2.txt)

-- > $x = 4 + 6$

*-- > diary(*1*, off)*

-- > $y = x*7$

*-- > diary(*1*, on)*

-- > $z = x/25$

*-- > diary(*0*)*

Здесь параметр 1 – это номер идентификатора, который программа присвоила файлу primer_2.txt при запуске дневника.

Чтобы прочитать содержимое дневника, щелкните мышью по надписи Lab1 в окне обозревателя файлов. Должно появиться название файла дневника primer_2.txt. Щелкните дважды по этому имени. Должно раскрыться окно блокнота с текстом – содержимым командного окна Scilab.

Убедитесь, что команда y = x * 7 не была записана в дневник.

Полученные текстовые файлы с расширением .txt можно прочитать с помощью текстового редактора Windows (выделив его на панели TotCom, а затем нажав клавишу F3, либо двойным щелчком мыши по его названию). Эти файлы можно распечатать или встраивать в различные документы.

Создание скрипт-файлов

Скрипт-файл (в буквальном переводе файл-сценарий) – это список команд Scilab, сохраненный на диске. Для сокращения обычно эти файлы называют просто скриптами. Для подготовки, редактирования и отладки скриптов служит специальный редактор среды Scilab, называемый SciNotes (в более ранних версиях он назывался SciPad). Редактор можно вызвать из главного меню рабочего стола Scilab, выполнив команду: Инструменты=>Текстовый редактор Scilab. В результате работы этой команды будет создан новый скрипт-файл. По умолчанию он имеет имя Безымянный документ 1.

Окно редактора скрипт-файлов выглядит стандартно, т.е. имеет заголовок, меню, панель инструментов.

Ввод текста в окно редактора скрипт-файла осуществляется по правилам, принятым для команд Scilab.

2.9.1. **Задание.** Вызовите редактор SciNotes с помощью команды главного меню среды Scilab: Инструменты =>Текстовый редактор Scilab. В окне редактора наберите текст скрипта для решения квадратного уравнения $ax^2 + bx + c = 0$.

Текст скрипта:

// Решение уравнения $ax^2 + bx + c = 0$

```
a = 1;   b = 5;   c = 4;
d = b ^ 2 – 4 * a * c;
x1 = (-b + sqrt(d)) / (2 * a)
x2 = (-b – sqrt(d)) / (2 * a)
```

Нетрудно заметить, что точка с запятой «;» ставится после тех команд, которые не требуют вывода значений.

Для сохранения введенной информации необходимо выполнить команду: Файл =>Сохранить как… из меню редактора. Открывается окно с заголовком Сохранить как… Если открыта папка Lab1, то в строке Filename нужно ввести имя сохраняемого скрипт-файла, например, roots (от англ. корни). Расширение указывать не нужно, оно добавляется автоматически. Щелчок по кнопке Save приведет к сохранению информации, находящейся в окне редактора. Скрипт-файлы сохраняются с расширением .sce.

Проверить, что скрипт-файл с именем roots.sce появился в папке Lab1, можно щелкнув мышью по строке Lab1 в дереве папок в окне обозревателя файлов.

2.9.2. Запуск (исполнение) скрипт-файла может быть выполнен в Scilab двумя основными способами.

1. Скрипт-файл может быть запущен непосредственно из окна редактора SciNotes. С этой целью выполните команды из меню редактора: Выполнить => …файл с отображением команд. В командном окне появятся строки с командами скрипта и результаты вычислений (в данном примере значения корней уравнения x1 и x2).

2. Скрипт-файл может быть запущен из командного окна. Для этого надо выполнить команды из главного меню: Файл =>

Выполнить. При этом появляется окно для выбора файла с заголовком Select a file to execute. Нужно выбрать из папки Lab1 файл с именем roots (щелкнув по нему кнопкой мыши), файл появится в строке File name. Затем надо щелкнуть по кнопке Open.

В командном окне появятся значения переменных x1 и x2 (результаты вычислений в скрипте). В конце будет выведено сообщение Execution done (исполнение выполнено).

Задание. Запустите скрипт-файл roots.sce двумя способами:

1) Из окна редактора SciNotes. После появления результатов в командном окне закройте окно редактора.

2) Из командного окна Scilab.

3 Задания для самостоятельной работы

Указание. Отчет по лабораторной работе № 1 выполняется в виде дневника, т.е. содержимого командного окна Scilab в ходе выполнения заданий для самостоятельной работы. Файл дневника создается в папке Lab1, рекомендуемое имя для файла дневника – self_work.txt. Каждое задание должно в начале иметь строку комментария, например, вида:

// Задание 1

Тексты заданий приводить не нужно.

Задание 1. Вычислите следующие выражения:

1) e^{14} и 38280π, каждый результат с точностью до 20 знаков. Какое из значений больше? (*Подсказка*: предварительно задайте необходимый формат для отображения результата)

2) $\ln 2$

3) *arctg* 0,5

Замечание. После выполнения заданных вычислений нужно вернуться к формату (по умолчанию) отображения с 10 символами.

Задание 2. Вычислите выражение

$$\left(\frac{1}{3}\right)^3 \cdot e^{a+b} + \frac{\sqrt{15-kx^2-0{,}41}}{10^{-2}\cdot|a+b|} + \frac{\ln(a+b)^2}{a+kx^2} - \sqrt[3]{3}$$

при значениях: a = 2,5 b = -5,25 x = 1,25 k = 4

Задание 3. Вычислите выражение

$$z = \left(\frac{2}{9}\right)^2 + \frac{\sqrt[4]{x^2+b}}{0{,}4x} - 10^4 e^{kx} + \cos\sqrt{x^3+b} + \frac{\sin 3}{(x^2+b)\cdot n}$$

при значениях: x = 2,5 b = 0,04 k = 4 n = 5

Задание 4. Дано тригонометрическое тождество

$$\sin^3 x = \frac{1}{4}(3\sin x - \sin 3x)$$

Проверьте правильность тождества путем вычисления по отдельности левой и правой части при значении x = 12°.

Задание 5. Определите две переменные: $alpha = 5\pi/8$ и $beta = \pi/8$. Используя эти переменные, покажите, что следующее тригонометрическое тождество справедливо, вычисляя значения левой и правой части

$$\sin\alpha\cdot\cos\beta = \frac{1}{2}\left[\sin(\alpha-\beta) + \sin(\alpha+\beta)\right]$$

Задание 6. Дано тригонометрическое тождество

$$tg\,4\gamma = \frac{4tg\,\gamma - 4tg^3\gamma}{1 - 6tg^2\gamma + tg^4\gamma}$$

Проверьте правильность тождества путем вычисления по отдельности его левой и правой чисти при γ = 20°. (Подсказка: вместо греческой буквы γ используйте имя gamma).

Задание 7. Вычислите выражение

$$\frac{4}{3}n^3\cdot\sin^2\frac{\alpha}{2}\cdot\sqrt[3]{\cos\alpha}$$

при $n = 1{,}7\cdot10^3$ $\alpha = 18^\circ$.

Задание 8. Выполните следующие действия с комплексными числами.

1) Число z_1, заданное в алгебраической форме $z_1 = 4 + 3i$, перевести в экспоненциальную $Ze^i\varphi$.

2) Число z_2, заданное в экспоненциальной форме $z_2 = 2{,}71e^{i\pi/12}$, перевести в алгебраическую.

3) Вычислить выражение

$$z_1^2 \cdot z_2 : z_3 + z_4,$$

где

$$z_3 = 1{,}82e^{-1,2i},$$
$$z_4 = \sqrt{3} - 2i.$$

Задание 9. Объект с начальной температурой T_0, который помещается в начальное время t = 0 внутрь камеры с постоянной температурой T_s, будет изменять свою температуру по закону

$$T = T_s + (T_0 - T_s)\, e^{-kt},$$

где T – температура объекта за время t; k – константа.

Предположим, что бутылка напитка с начальной температурой 30^o C помещается в холодильник, где температура 4^o C. Определите, с точностью до ближайшего целого числа, температуру, которую будет иметь напиток через 3 часа. Константа k = 0,45. Сначала определите все переменные, а затем вычислите температуру.

Разработайте скрипт и сохраните его в файле с именем zadanie9.sce в вашей папке Lab1. Запустите скрипт на исполнение из окна редактора с выводом текста скрипта и убедитесь в его работоспособности.

Задание 10. Снаряд запускается под углом θ и начальной скоростью V. Время полета снаряда до приземления обозначим как t_{travel}, максимальное расстояние до точки падения обозначим как x_{max}, и максимальная высота подъема как h_{max}. Эти величины определяются соотношениями:

$$t_{travel} = 2\frac{V}{g}\sin\theta$$

$$x_{\max} = 2\frac{V^2}{g}\sin\theta\cdot\cos\theta$$

$$h_{\max} = 2\frac{V^2}{g}\sin^2\theta,$$

где g=9,81 m/c^2 – ускорение свободного падения.

Рассмотрите случай, когда V = 200 m/c и θ = 54^o. Определите V и θ как переменные (для греческой буквы θ используйте слово theta) и вычислите значения t_{travel}, x_{max}, h_{max}.

Разработайте скрипт и сохраните его в файле с именем zadanie10.sce в вашей папке Lab1. Запустите скрипт на исполнение из окна редактора с выводом текста скрипта и убедитесь в его работоспособности.

Задание 11. Треугольник имеет стороны a = 16 см, b = 18 см, c = 15 см и углы α, β, γ.

Определите a, b, c как переменные и затем:

1) Вычислите углы (в градусах) с использованием закона косинусов для треугольника:

$$c^2 + b^2 - 2\cdot a\cdot b\cdot\cos\gamma$$

$$a^2 = b^2 + c^2 - 2\cdot b\cdot c\cdot\cos\alpha$$

$$b^2 = a^2 + c^2 - 2\cdot a\cdot c\cdot\cos\beta$$

2) Проверьте, что сумма углов должна быть равна 180^o.

3) Разработайте скрипт для решения задачи и сохраните его в файле с именем zadanie11.sce в папке Lab1. Запустите скрипт на исполнение с выводом текста скрипта и убедитесь в его работоспособности.

4 Содержание отчета

Отчет по лабораторной работе № 1 выполняется в виде дневника, т.е. содержимого командного окна Scilab в ходе выполнения заданий для самостоятельной работы

Разработанный дневник необходимо продемонстрировать преподавателю для оценки объема выполненной работы и анализа качества выполнения заданий.

Контрольные вопросы

1. Какие системные переменные и константы существуют в Scilab?
2. Как приводятся текстовые комментарии?
3. Какие имеются основные тригонометрические функции в Scilab?
4. Как сообщается об ошибках в Scilab?
5. Как задается формат отображения чисел?
6. Что такое скрипт?
7. Как можно создать скрипт-файл?

Лабораторная работа № 2

СОЗДАНИЕ МАССИВОВ

1 Цель работы

Изучить способы создания массивов – векторов и матриц. Освоить методы индексации массивов. Исследовать встроенные функции для работы с массивами.

2 Порядок выполнения работы

2.1 Создание папки для работы

При выполнении лабораторной работы № 2 вы будете использовать папку с именем Lab2, которую необходимо предварительно создать.

С этой целью откройте вашу рабочую папку (с вашей фамилией) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab2.

В дальнейшем вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 2. Полный путь к этой папке будет такой:

F:\OTC\Ivanov\Lab2

2.2 Запуск и настройка среды разработки Scilab

2.2.1. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа – рабочий стол среды разработки Scilab.

2.2.2. Нужно установить в качестве активной (текущей) папку Lab2. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем OTC, затем папку с вашей фамилией (условно Ivanov) и затем папку Lab2. В строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab2. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab2

2.2.3. При необходимости очистите командное окно и журнал команд, использую пункты меню Правка.

2.3 Создание векторов методом поэлементного ввода

2.3.1. Для создания вектора-строки нужно ввести его элементы с пробелами или запятыми между элементами внутри квадратных скобок.

Задание. Создайте вектор-строку следующими способами:

1) используя пробелы между элементами

--> v1 = [1 2 3]

2) используя запятые между элементами

--> v2 = [2, 1, -1]

2.3.2. Для создания вектора-столбца нужно ввести элементы с разделением их символом «;» или нажатием клавиши Enter после ввода каждого элемента.

Задание. Создайте вектор-столбец следующими способами:

1) используя символ «;» после ввода каждого элемента

--> u1 = [2; 4; 5]

2) используя Enter после ввода каждого элемента

--> u2 = [4
8
10]

В Scilab возможен ввод элементов векторов в виде арифметических выражений, содержащих любые доступные системе функции. В этом случае лучше для зрительного разделения элементов использовать символы «,» или «;».

Задание. Создайте вектор-строку вида:

$--> v = \left[2 + 2/\left(3 + 4\right)\quad \exp(5)\quad sqrt(10)\right]$

2.4 Создание вектора с помощью оператора «:» - двоеточие

Для создания векторов с элементами, представляющими собой упорядоченную числовую последовательность с постоянным промежутком (разностью между значениями соседних элементов) в Scilab используется оператор «:» (двоеточие) в виде:

Начальное_значение : Шаг : Конечное_значение

Если шаг не задан, то по умолчанию он считается равным 1. Если шаг – величина положительная, то последовательность чисел возрастающая, если же шаг – величина отрицательная, то последовательность чисел – уменьшающаяся.

При создании векторов с помощью оператора «:» квадратные скобки при определении вектора можно не использовать.

Задание. Выполните примеры использования оператора «:» для векторов:

1)

$--> 1:5$

2)

$--> i = 0:2:10$

3)

$--> j = [10:-2:2]$

4)

$--> w = 0:\%pi/2:2*\%pi$

5)

$--> y = 0:8$

$--> \cos(y)$

2.5 Создание матриц

Матрица – это двумерный массив размером m × n, где m – количество строк, а n – количество столбцов. Обычно матрица создается путем ввода элементов строка за строкой внутри квадратных скобок. Элементами матриц могут быть числа или математические выражения, включающие числа, определенные ранее переменные и функции. Все строки должны иметь одно и то же количество элементов.

2.5.1. **Задание**. Создайте матрицу A размером 3 × 3 с разделением строк символом «;»

--> $A = [5 \quad 35 \quad 43; \quad 4 \quad 76 \quad 81; \quad 21 \quad 32 \quad 40]$

2.5.2. **Задание**. Создайте матрицу B размером 3 × 5 с разделением строк вводом Enter

--> $B = [7 \quad 2 \quad 76 \quad 33 \quad 8$

$1 \quad 98 \quad 6 \quad 25 \quad 6$

$5 \quad 54 \quad 68 \quad 9 \quad 0]$

2.5.3. **Задание**. Создайте матрицу C размером 2 × 3. Для зрительного разделения элементов используйте символ «,»

--> cd = 6; e = 3; h = 4;

--> C = [e, cd*h, cos(%pi/3); h^2, sqrt(h*e/cd), 14]

2.5.4. **Задание.** Создайте матрицу D размером 3 × 6, в которой некоторые строки вводятся как векторы с элементами в виде числовых значений с постоянным шагом изменения

--> D = [1 : 2 : 12; 0 : 5 : 25; 67 2 43 68 4 13]

2.6 Операция конкатенации массивов

Матрицы и векторы можно формировать, составляя их из ранее заданных матриц и векторов. Эта операция называется конкатенацией (объединением).

Задание. Исследуйте операции конкатенации векторов и матриц для заданных векторов:

--> $v1 = [1 \quad 2 \quad 3]; \; v2 = [4 \quad 5 \quad 6]; \; v3 = [7 \quad 8 \quad 9];$

1) горизонтальная конкатенация векторов-строк

--> $V = [v1 \quad v2 \quad v3]$

2) вертикальная конкатенация векторов-строк

--> $V = [v1; \quad v2; \quad v3]$

3) горизонтальная конкатенация матриц

--> $M = [V \quad V \quad V]$

4) вертикальная конкатенация матриц

--> $M = [V; \quad V]$

2.7 Операция транспонирования

Операция транспонирования обозначается знаком «’» - апостроф. Она меняет строки матрицы с ее столбцами. Для квадратных матриц операция транспонирования имеет наглядную геометрическую интерпретацию: на своих местах остаются элементы главной диагонали, а остальные «отражаются симметрично» относительно этой диагонали. Вектор-строки операцией транспонирования преобразуются в вектор-столбцы и наоборот.

Задание. Исследуйте операцию транспонирования:

1) для матрицы А размером 2 × 3

--> $A = [1 \quad 2 \quad 3; \quad 4 \quad 5 \quad 6]$

--> $B = A'$

2) для квадратной матрицы

--> $C = [1 \quad 1 \quad 1; \quad 2 \quad 2 \quad 2; \quad 3 \quad 3 \quad 3]$

--> $D = C'$

3) для вектора-строки

--> $v = [1 \quad 2 \quad 3]$

--> $u = v'$

4) для вектора-столбца

--> w = [1; 2; 3]

--> z = w’

2.8 Адресация элементов массивов

Элементы в массиве (в векторе или в матрице) могут адресоваться индивидуально или в группах.

2.8.1. Адрес элемента вектора – это его позиция в строке (или в столбце), указанная в круглых скобках.

Задание. Исследуйте способы адресации для вектора:

--> $v = [35 \quad 46 \quad 78 \quad 23 \quad 5 \quad 14 \quad 81 \quad 3 \quad 55]$

1) значение 4-го элемента вектора

--> $v(4)$

2) значение 7-го элемента вектора

--> $v(7)$

3) использование элементов вектора в математическом выражении

$--> v(2)*v(5)+sqrt(v(7))$

4) присвоение элементу вектора нового значения

$--> v(1)=77;\ v(5)=99;$

$--> v$

2.8.2. Адрес элемента матрицы – это позиция, определяемая номером строки и номером столбца, где он расположен, указанная в круглых скобках.

Задание. Исследуйте способы адресации для матрицы:

$--> M=\begin{bmatrix}3 & 11 & 6 & 5;\ 4 & 7 & 10 & 2;\ 13 & 9 & 0 & 8\end{bmatrix}$

1) значения определенных элементов

$--> M(1,2)$

$--> M(3,4)$

2) использование элементов матрицы в математическом выражении

$--> M(2,4)-M(1,2)$

3) присвоение элементам матрицы новых значений

$--> M(1,2)=35;\ M(2,4)=66;$

$--> M$

2.9 Применение оператора «:» - двоеточие для адресации массивов

Символ «:» - двоеточие может использоваться для адресации диапазона элементов в векторе или матрице.

Обычно при обращении к элементам матрицы отсчет индексов ведется от первой строки и первого столбца. Оператор “$” позволяет адресовать элементы матрицы, отсчитывая от последней строки или столбца в зависимости от контекста, например:

A(k, $) – элемент на пересечении к-ой строки и последнего столбца;

A($, j) – элемент на пересечении последней строки и j-го столбца.

Задание. Исследуйте различные варианты адресации для доступа к группе элементов массивов.

1) создайте вектор:

$--> v=\begin{bmatrix}4 & 15 & 8 & 12 & 34 & 2 & 50 & 32 & 11\end{bmatrix}$

и выделите из него элементы с 3-го по 7-й:

--> $u = v(3:7)$

2) создайте матрицу:

--> $X = [3,\ 4,\ 8,\ 12;\ 2,\ 5,\ 7,\ 11;\ 1,\ 6,\ 9,\ 10]$

3) для доступа к первой строке введите команду:

--> $X(1,:)$

4) для доступа к последней строке введите:

--> $X(\$,:)$

5) для доступа к последним двум строкам введите команду:

--> $X(\$-1:\$,:)$

6) для доступа к 1-й и 3-й строкам введите команду:

--> $X([1,3],:)$

7) для доступа к 1-му столбцу введите:

--> $X(:,1)$

8) для доступа к последнему столбцу введите:

--> $X(:,\$)$

9) для доступа к 1-му и 3-му столбцам введите команду:

--> $X(:,[1,3])$

10) для доступа к элементам 2-й строки X(2, 2), X(2, 3), X(2,4):

--> $X(2,[2,\ 3,\ 4])$

11) для доступа к элементам из нескольких строк и столбцов:

--> $X([2,3],[2,3,4])$

2.10 Добавление элементов к массиву

2.10.1. Элементы могут быть добавлены к вектору путем присвоения значений новым элементам.

Задание.

1) Создайте вектор-строку из 4-х элементов:

--> $w = 1:4$

добавьте к нему еще 6 элементов:

--> $w(5:10) = 10:5:35$

2) создайте вектор-строку:

--> $z = [5\quad 7\quad 2]$

добавьте к нему 8-й элемент:

--> $z(8) = 4$

2.10.2. Строки и/или столбцы могут быть добавлены к существующей матрице путем присваивания значений новым строкам или столбцам. Однако следует помнить, что размеры добавляемых строк и столбцов должны соответствовать существующей матрице.

Задание.

1) Создайте матрицу:

$--> D = [1 \quad 2 \quad 3 \quad 4; \quad 5 \quad 6 \quad 7 \quad 8]$

2) добавьте элементы 3-й строки:

$--> D(3,:) = [10 : 4 : 22]$

3) добавьте элемент:

$--> D(4,5) = 17$

2.11 Удаление элементов массивов

Для удаления элемента или группы элементов массива используется символ «[ ]» - пустые квадратные скобки.

2.11.1. **Задание.** Исследуйте способы удаления элементов вектора:

1) создайте вектор:

--> v = [2 8 40 65 3 55 23 154 75 80]

2) удалите 6-й элемент:

$--> v(6) = [\ ]$

3) удалите элементы с 3-го по 6-й:

$--> v(3 : 6) = [\ ]$

2.11.2. Исследуйте способы удаления строк и столбцов матрицы:

1) создайте матрицу:

$--> N = [1 \quad 2 \quad 3; \quad 4 \quad 5 \quad 6; \quad 7 \quad 8 \quad 9]$

2) Удалите 2-й столбец матрицы:

$--> N(:,2) = [\ ]$

3) удалите 2-ю строку матрицы:

$--> N(2,:) = [\ ]$

2.12 Специальные матричные функции

Для работы с матрицами и векторами в Scilab существуют специальные функции. Рассмотрим некоторые функции, которые могут формировать матрицы и определять их основные параметры.

Функция matrix(A [, n, m]) – преобразует матрицу A в матрицу другого размера.

Задание. Исследуйте работу функции matrix( ):

1) создание матрицы D размера 3×2

$--> D = [1 \quad 2; \quad 3 \quad 4; \quad 5 \quad 6]$

2) создание матрицы размера 2×3

$--> matrix(D, 2, 3)$

3) создание матрицы размера 1×6 (вектора-строки)

$--> matrix(D, 1, 6)$

4) Создание матрицы размера 6×1 (вектора-столбца)

$--> matrix(D, 6, 1)$

Функция length(X) – определяет количество элементов массива X; если X – вектор, то его длину; если X – матрица, то вычисляет общее число ее элементов.

Задание. Исследуйте работу функции length( ):

Создайте вектор-строку

$--> v = [-1 \quad 0 \quad 3 \quad -2 \quad 1 \quad -1 \quad 1]$

Определите его длину

$--> length(v)$

Создайте матрицу

$--> M = [1 \quad 2 \quad 3; \quad 4 \quad 5 \quad 6]$

Определите количество элементов матрицы

$--> length(M)$

Функция size(X [, f1]) – определяет размер массива X; если X – матрица, то size(X, 1) или size(X, ‘r’) определяет число строк матрицы X, а size(X, 2) или size(X, ‘c’) – число столбцов.

Задание. Исследуйте работу функции size( ):

Создайте матрицу

$--> M = [1 \quad 2; \quad 3 \quad 4; \quad 5 \quad 6; \quad 7 \quad 8]$

Определите количество строк m и столбцов n матрицы

--> $[m,n]=size(M)$

Определите количество строк

--> $size(M,1)$

Определите количество столбцов

--> $size(M,2)$

Функция ones(m, n) – создает матрицу из единиц, размером m строк и n столбцов.

Задание. Исследуйте работу функции ones( ):

1) формирование квадратной матрицы размером 2 × 2

--> $ones(2,2)$

2) формирование вектора-строки

--> $ones(1,3)$

3) формирование матрицы размера m × n

--> $m=3; n=2;$

--> $X=ones(m,n)$

4) формирование матрицы Y из единиц с размерами матрицы M

--> $M=[1 \quad 2 \quad 3; \quad 4 \quad 5 \quad 6]$

--> $Y=ones(M)$

Функция zeros(m, n) – создает матрицу из нулей, размером m строк и n столбцов.

Задание. Исследуйте работу функции zeros( ):

1) формирование матрицы размером 3 × 2

--> $zeros(3,2)$

2) формирование нулевой матрицы Z с размером вектора v

--> $v=[1 \quad 2 \quad 3 \quad 4 \quad 5]$

--> $Z=zeros(v)$

Функция eye(m, n) – формирует единичную матрицу из m строк и n столбцов.

Задание. Исследуйте работу функции eye( ):

1) создание квадратной матрицы

--> $eye(3,3)$

2) создание вектора-столбца

--> $eye(5,1)$

3) формирование единичной матрицы E с размерами матрицы M

--> $M = [0 \quad 1; \quad 2 \quad 3]$

--> $E = eye(M)$

3 Задания для самостоятельной работы

Указание. Отчет по лабораторной работе № 2 выполняется в виде дневника, т.е. содержимого командного окна Scilab в ходе выполнения заданий для самостоятельной работы. Файл дневника создается в папке Lab2, рекомендуемое имя для файла дневника – self_work.txt. Каждое задание должно в начале иметь строку комментария, например, вида:

// Задание 1

Тексты заданий приводить не нужно.

Задание 1. Используйте одну команду для создания вектора-строки (с присвоением его значений переменной с именем w) с 19 элементами:

$w = [1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 9 \quad 8 \quad 7 \quad 6 \quad 5 \quad 4 \quad 3 \quad 2 \quad 1]$

Не вводите вектор поэлементно.

Задание 2. Для заданной матрицы $M = [2 \quad 4 \quad 1; \quad 6 \quad 7 \quad 2; \quad 3 \quad 5 \quad 9]$ введите команды, которые выполняют следующее:

а) присваивают первую строку матрицы M вектору с именем x1;

б) присваивает последние 2 строки матрицы M новой матрице с именем Y.

Задание 3. Для заданной матрицы

$A = [2, \quad 7, \quad 9, \quad 7; \quad 3, \quad 1, \quad 5, \quad 6; \quad 8, \quad 1, \quad 2, \quad 5]$ напишите команды, которые выполняют следующее:

а) присваивают четные столбцы матрицы A новой матрице с именем B;

б) присваивают нечетные строки матрицы A новой матрице с именем C;

в) преобразуют матрицу A в матрицу размером 4×3;

г) извлекают корень квадратный из всех элементов матрицы A.

4 Содержание отчета

Отчет по лабораторной работе № 2 выполняется в виде дневника, т.е. содержимого командного окна Scilab в ходе выполнения заданий для самостоятельной работы.

Разработанный дневник необходимо продемонстрировать преподавателю для оценки объема выполненной работы и анализа качества выполнения заданий.

Контрольные вопросы

1. Какими способами можно создать вектор-строку?
2. Каким способами можно создать вектор-столбец?
3. Какими способами можно создать матрицу?
4. Как можно использовать оператор двоеточие для создания вектора?
5. Объясните термин «конкатенация массивов».
6. Объясните термин «транспонирование массивов».
7. Как можно транспонировать вектор-строку и вектор-столбец?
8. Как выполняется адресация для векторов?
9. Как выполняется адресация для матриц?

Лабораторная работа № 3

МАТЕМАТИЧЕСКИЕ ОПЕРАЦИИ С МАССИВАМИ

1 Цель работы

Изучить и исследовать методы математических операций с векторами и матрицами. Исследовать встроенные функции для операций с массивами.

2 Порядок выполнения работы

2.1 Создание папки для работы

При выполнении лабораторной работы № 3 вы будете использовать папку с именем Lab3, которую необходимо предварительно создать.

С этой целью откройте вашу рабочую папку (с вашей фамилией) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab3.

В дальнейшем вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 3. Полный путь к этой папке будет такой:

F:\OTC\Ivanov\Lab3

2.2 Запуск и настройка среды разработки Scilab

2.2.1. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа – рабочий стол среды разработки Scilab.

2.2.2. Нужно установить в качестве активной (текущей) папку Lab3. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем ОТС, затем папку с вашей фамилией (условно Ivanov) и затем папку Lab3. В строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab3. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab3

2.2.3. При необходимости очистите командное окно и журнал команд, использую пункты меню Правка.

2.3 Исследование матричных математических операций

2.3.1. **Задание.** Исследуйте выполнение операций сложения и вычитания с векторами и матрицами:

1) создайте векторы $vectA = [8 \quad 5 \quad 4]$ и $vectB = [10 \quad 2 \quad 7]$. Выполните сложение векторов:

vectC = vectA + vectB,

а затем вычитание:

vectD = vectA-vectB

2) создайте две матрицы:

$A = [5 \ -3 \ 8; \ 9 \ 2 \ 10]$ и $B = [10 \ 7 \ 4; \ -11 \ 15 \ 1]$

Выполните операции сложения матриц A + B и вычитания A-B.

Затем выполните операцию сложения вектора vectA с матрицей A:

vectA + A

Объясните полученные результаты.

2.3.2. **Задание.** Исследуйте операции сложения и вычитания скаляров с векторами и матрицами:

1) создайте вектор $vectA = [1 \ 5 \ 8 \ -10 \ 2]$ и затем выполните операцию:

vectA + 4

2) создайте матрицу $A = [6 \ 21 \ -15, \ 0 \ -4 \ 8]$ и затем выполните:

A-5

Объясните полученные результаты.

2.3.3. **Задание.** Выполните операции матричного умножения:

1) создайте две матрицы:

$A = [1 \ 4 \ 2, \ 5 \ 7 \ 3, \ 9 \ 1 \ 6, \ 4 \ 2 \ 8]$

$B = [6 \ 1, \ 2 \ 5, \ 7 \ 3]$

и выполните следующие операции:

$C = A * B$

$D = B * A$

Объясните полученные результаты.

2) создайте две квадратные матрицы:

$F = [1\ \ 3,\ \ 5\ \ 7]$

$G = [4\ \ 2,\ \ 1\ \ 6]$

и выполните следующие операции:

$F * G$

$G * F$

Объясните полученные результаты.

3) создайте два вектора:

$v = [2\ \ 5\ \ 1]$

$u = [3,\ \ 1,\ \ 4]$

и выполните следующие операции:

$v * u$

$u * v$

Объясните полученные результаты.

4) создайте матрицу:

$A = [2\ \ 5\ \ 7\ \ 0;\ \ 10\ \ 1\ \ 3\ \ 4;\ \ 6\ \ 2\ \ 11\ \ 5]$

и выполните операции умножения ее на скалярную величину:

$3 * A$

$A * 3$

Объясните полученные результаты.

2.3.4. Операция деления массивов не определена в линейной алгебре. В Scilab операция деления массивов используется для решения матричных уравнений. Используется два варианта операции деления: « \ » - левое деление и « / « - правое деление.

Операция левого деления используется для решения матричного уравнения

$$A \cdot X = B,$$

где X и B являются векторами-столбцами.

Решением этого уравнения будет:

$$X = A \setminus B$$

Операция правого деления используется для решения матричного уравнения

$$X \cdot C = D,$$

где X и D являются векторами-строками.

Решением этого уравнения будет:

$$X = D / C$$

Задание. Используйте матричные операции для решения следующей системы линейных алгебраических уравнений (СЛАУ):

$$\begin{aligned} 4x - 2y + 6z &= 8 \\ 2x + 8y + 2z &= 4 \\ x + 10y + 3z &= 0 \end{aligned} \quad , \qquad (1)$$

Эту систему можно записать в матричной форме как:

$$A \cdot X = B, \qquad (2)$$

где

$$A = \begin{bmatrix} 4 & -3 & 6 \\ 2 & 8 & 2 \\ 6 & 10 & 3 \end{bmatrix},$$

$$X = \begin{bmatrix} x \\ y \\ z \end{bmatrix},$$

$$B = \begin{bmatrix} 8 \\ 4 \\ 0 \end{bmatrix}.$$

Выполните решение уравнения (2) несколькими способами:

1) с использованием операции левого деления:

--> A =[4 -2 6; 2 8 2; 6 10 3]

--> B =[8; 4; 0]

--> X = A \ B

2) с использованием функции вычисления обратной матрицы:

--> Xa = inv(A) * B

3) с использованием символа инверсии матрицы A^{-1} :

--> Xb = A ^ (-1) * B

Сделайте выводы по полученным результатам.

4) Запишите уравнение (1) в форме:

$X \cdot C = D$, (3)

где X – вектор-строка неизвестных,

C = A' – матрица коэффициентов системы,

D = B' – вектор-строка свободных членов.

Выполните решение уравнения (3) несколькими способами:

1) с использованием операции правого деления:

--> C = A'

--> D = B'

--> Xc = D / C

2) с использованием функции вычисления обратной матрицы:

--> Xd = D * inv(C)

3) с использованием символа инверсии матрицы C^{-1} :

--> Xe = D * C ^ (-1)

Сделайте выводы по полученным результатам.

2.4 Исследование поэлементных математических операций с массивами

2.4.1. **Задание.** Исследуйте поэлементные операции умножения, деления и возведения в степень матриц:

1) создайте матрицы:

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

--> B = [1 4 10; 3 2 7]

2) выполните операции:

--> A .* B

--> C = A ./ B

--> B .^ 3

--> A * B

Сделайте выводы по полученным результатам.

2.4.2. **Задание.** Исследуйте поэлементные операции с векторами:

1) создайте вектор-строку:

--> x = [1:8]

и вычислите выражение:

В Scilab для этого нужно ввести:

--> y = x .^ 2 – 4 * x

2) создайте вектор-строку:

--> t = 1 : 5

3) вычислите функцию:

--> y = cos(t)

4) вычислите функцию:

--> z = y / t

Результатом будет единственное число, а не функция $z(t) = \cos(t)/t$ от нескольких значений аргумента t ;

5) исправьте вычисление функции z, используя поэлементную операцию деления:

--> z = y ./ t

Сделайте выводы по полученным результатам.

2.5 Встроенные функции для операций с массивами

Для работы с матрицами и векторами в Scilab существуют специальные функции. Рассмотрим некоторые функции, которые могут определять основные параметры массивов:

mean(A) – если A – вектор, то функция возвращает среднее (арифметическое) значение элементов;

max(A) – если A – вектор, то функция возвращает наибольший элемент;

min(A) – если A – вектор, то функция возвращает наименьший элемент;

sum(A) – если A – вектор, то функция возвращает сумму элементов;

length(A) – если A – вектор, то функция возвращает длину вектора (количество элементов).

Подробную информацию о функциях можно получить из справочной системы Scilab, выполнив команду:

help function_name

3 Задания для самостоятельной работы

Указание. Отчет по лабораторной работе № 3 выполняется в виде дневника, т.е. содержимого командного окна Scilab в ходе выполнения заданий для самостоятельной работы. Файл дневника создается в папке Lab3, рекомендуемое имя для файла дневника – self_work.txt. Каждое задание должно в начале иметь строку комментария, например, вида:

// Задание 1

Тексты заданий приводить не нужно.

Задание 1. Выполните следующие команды и объясните, почему Scilab генерирует ошибку:

1)

```
x = [1, 2, 3];
y = [4, 5, 6];
z = x * y
```

2)

```
x = [1, 2, 3];
y = [4, 5, 6];
z = y * x'
```

3)

```
x = [1, 2, 3];
y = [4, 5, 6];
z = y' * x
```

Задание 2. Пусть известны округленные среднесуточные значения температуры воздуха в течение недели (в ^{o}C). Эти данные заданы в массиве *temp* =[-1 0 2 3 -5 -7 -4]. Требуется определить:

1) минимальную температуру;

2) максимальную температуру;

3) среднюю температуру.

Указание. Для вычисления значений необходимо использовать встроенные функции Scilab для работы с массивами.

Задание 3. Создайте три матрицы:

$$A=\begin{bmatrix}2 & 4 & -1\\ 3 & 1 & -5\\ 0 & 1 & 4\end{bmatrix},\ B=\begin{bmatrix}-2 & 5 & 0\\ -3 & 2 & 7\\ -1 & 6 & 9\end{bmatrix},\ C=\begin{bmatrix}0 & 3 & 5\\ 2 & 1 & 0\\ 4 & 6 & -3\end{bmatrix}$$

и вычислите:

1) выражения $A+B$ и $B+A$, чтобы показать, что сложение матриц подчиняется переместительному закону;

2) выражения $A * (B + C)$ и $A * B + A * C$, чтобы показать, что умножение матриц подчиняется распределительному закону.

Задание 4. Используйте матрицы A, B и C из задания 4, чтобы ответить на следующие вопросы:

1) справедливо ли равенство: $A * B = B * A$?

2) справедливо ли равенство: $A * (B * C) = (A * B) * C$?

3) справедливо ли равенство: $(A * B)' = B' * A'$?

4) справедливо ли равенство: $(A + B)' = A' + B'$?

Задание 5. Определите вектор $x = [1 : 10]$ и напишите команды для выполнения следующих операций:

1) прибавить число 10 к каждому элементу вектора x. Присвоить результат новому вектору m ;

2) прибавить число 3 только к нечетным элементам вектора x. Присвоить результат новому вектору r ;

3) вычислить квадратный корень из каждого элемента вектора x. Присвоить результат новому вектору s ;

4) возвести в квадрат каждый элемент вектора x. Присвоить результат новому вектору t.

Задание 6. Определите два вектора-столбца:

$x = [1;\ 2;\ 3;\ 4]$ и $y = [5;\ 6;\ 7;\ 8]$.

Напишите команды, которые выполняют следующие операции:

1) прибавить сумму элементов вектора x к элементам вектора y (используйте для суммирования встроенную функцию sum( )). Присвоить полученный результат новому вектору m ;

2) возвести каждый элемент вектора x в степень, соответствующую элементу вектора y. Присвоить результат новому вектору s;

3) разделить каждый элемент вектора y на соответствующий элемент вектора x. Присвоить результат новому вектору z ;

4) умножить каждый элемент вектора x на соответствующий элемент вектора y. Присвоить результат новому вектору r ;

5) найти сумму элементов вектора x и присвоить результат переменной с именем t ;

6) вычислить выражение: $x' * y\text{-}t$

Задание 7. Создайте вектор $t = 1:0.2:2$, затем напишите и выполните следующие математические выражения:

1) $\ln\left(2 + t + t^2\right)$
2) $\lg\left(2 + t + t^2\right)$
3) $e^{1+\cos 3t}$
4) $\cos^2 t + \sin^2 t$
5) *arctg t*
6) *ctg t*

Задание 8. Докажите, что $\lim\limits_{x \to 0} \dfrac{\sin x}{x} = 1$. Сделайте это путем создания вектора x с элементами: 1.5, 1.0, 0.5, 0.1, 0.01, 0.001, 0.0001, 0.00001. Затем создайте новый вектор y, в котором каждый элемент определяется из элементов x с помощью вычисления выражения $\dfrac{\sin x}{x}$. Сравните полученные значения элементов из y со значением 1 (используйте формат для отображения чисел с 20 символами).

Задание 9. Используйте Scilab для доказательства, что сумма членов бесконечного ряда

$$\sum_{n=1}^{\infty}\frac{1}{2^n}=\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3}+\ldots$$

стремится к 1. Сделайте это путем вычисления суммы для

1) n =10
2) n =20
3) n= 30
4) n = 40

Для каждого варианта создайте вектор с именем n, в котором 1-й элемент будет 1, шаг равен 1, а последний элемент соответственно 10, 20, 30 или 40. Затем используйте поэлементные операции для создания вектора, в котором элементы равны $\frac{1}{2^n}$. И наконец, используйте встроенную функцию sum( ) для сложения членов ряда. Сравните значения, полученные в вариантах 1), 2), 3) и 4) со значением 1 (не забудьте поставить в конце команд символ «;», чтобы не выводить на дисплей множество элементов векторов).

Задание 10. Решите следующую СЛАУ:

$3x-2y+2z=7.5;$

$-4.5x+2y+3z=5.5;$

$5x+y-2.5z=4.5.$

Проверьте правильность вычисления неизвестных x, y, z.

Задание 11. Решите следующую СЛАУ:

$3u+1.5v+w+0.5x+4y=-11.75;$

$-2u+v+4w-3.5x+2y=19;$

$6u-3v+2w+2.5x+y=-23;$

$u+4v-3w+0.5-2y=-1.5;$

$3u+2v-w+1.5x-2y=-3.5.$

Проверьте правильность вычисления неизвестных.

4 Содержание отчета

Отчет по лабораторной работе № 3 выполняется в виде дневника, т.е. содержимого командного окна Scilab в ходе выполнения заданий для самостоятельной работы

Разработанный дневник необходимо продемонстрировать преподавателю для оценки объема выполненной работы и анализа качества выполнения заданий.

Контрольные вопросы

1. Как выполняется матричное сложение и вычитание?
2. Как выполняется матричное умножение?
3. Для каких целей в Scilab введены операции левого деления “ \ ” и правого деления “ / “ ?
4. Как выполняются операции поэлементного умножения и деления с массивами?
5. Как выполняется операция поэлементного возведения в степень с массивом?
6. Какие основные встроенные функции для операций с массивами имеются в Scilab?

Лабораторная работа № 4

УПРАВЛЕНИЕ ВВОДОМ И ВЫВОДОМ ДАННЫХ В СКРИПТ-ФАЙЛАХ

1 Цель работы

Изучить и исследовать функции ввода input( ) и вывода disp( ) информации в скрипт-файлах. Исследовать функцию форматированного вывода данных mprintf( ).

2 Порядок выполнения работы

2.1 Создание папки для работы

При выполнении лабораторной работы № 4 вы будете использовать папку с именем Lab4, которую необходимо предварительно создать.

С этой целью откройте вашу рабочую папку (с вашей фамилией) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab4.

В дальнейшем вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 4. Полный путь к этой папке будет такой:

F:\ОТС\Ivanov\Lab4

2.2 Запуск и настройка среды разработки Scilab

2.2.1. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа – рабочий стол среды разработки Scilab.

2.2.2. Нужно установить в качестве активной (текущей) папку Lab4. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем ОТС, затем папку с вашей фамилией (условно Ivanov) и затем папку Lab4. В

строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab4. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab4

2.2.3. При необходимости очистите командное окно и журнал команд, использую пункты меню Правка.

2.3 Исследование метода задания данных для скрипта из командного окна

2.3.1. Переменные из скрипт-файлов и переменные из командного окна находятся в едином рабочем пространстве Scilab, т.е. используют одну и ту же область памяти. Поэтому можно определять переменные и присваивать им различные значения непосредственно в командном окне перед исполнением скрипта, не редактируя сам скрипт-файл.

Задание. Откройте редактор SciNotes и создайте следующий скрипт:

// Скрипт вычисляет среднее значение трех переменных x1, x2, x3

// Переменные определяются в командном окне

average = (x1 + x2 + x3) / 3

Сохраните скрипт-файл под именем, например, ave.sce в папке Lab4.

Закройте окно редактора SciNotes.

1) Введите в командном окне данные для скрипта:

--> x1 = 10;

--> x2 = 20;

--> x3 = 30;

Запустите скрипт с помощью команды:

--> exec(‘ave.sce’)

Будет вычислено среднее значение:

average =

20.

2) Выполните скрипт-файл при новых исходных данных, например, x1 = 15, x2 = 25, x3 = 35.

2.4 Использование команды input для ввода данных в скрипт-файл

2.4.1. В этом случае переменная определяется в скрипте, а когда файл исполняется, пользователю предлагается присвоить значение переменной.

Формат команды input:

variable_name = input(‘string-message’)

Когда идет прогон скрипта, выполняется команда input и строка-сообщение отображается в командном окне. Обычно эта строка сообщает, что пользователь должен ввести значение, которое присваивается переменной. Строка-сообщение должна быть заключена в одиночные «’ ‘» или двойные «” “» кавычки. Текст сообщения может быть на английском или русском языках.

Как и для любой переменной в Scilab, переменная и присвоенное ей значение может быть отображены в командном окне, если не напечатан символ « ; « в конце команды input.

Задание. Откройте редактор SciNotes и создайте скрипт:

// Скрипт вычисляет среднее значение трех переменных x1, x2, x3

// Переменные объявляются в скрипте, но значения им присваиваются

// с помощью команды input при исполнении скрипта в командном окне

x1 = (‘Введите значение переменной x1: ‘);

x2 = (‘Введите значение переменной x2: ‘);

x3 = (‘Введите значение переменной x3: ‘);

average = (x1 + x2 + x3) / 3

Сохраните набранный скрипт-файл в папке Lab4 под именем ave2.sce. Закройте редактор SciNotes.

1) Запустите скрипт из командного окна с параметром 2, чтобы не отображались команды файла:

--> exec(‘ave2.sce’, 2)

Введите последовательно запрашиваемые значения переменных, например, x1 = 10, x2 = 20, x3 = 30.

2) Повторно запустите скрипт ave2.sce и введите значения переменных x1 = 15, x2 = 25, x3 = 35.

2.4.2. В рассмотренном примере присваивались значения скалярным переменным. Однако с помощью команды input можно присваивать значения векторам и матрицам (вводом левой квадратной скобки, затем строка за строкой, в заключение правая квадратная скобка).

Задание. Откройте блокнот SciNotes и создайте следующий скрипт:

```
// Скрипт вычисляет среднее значение элементов вектора-строки
X = input ('Введите произвольное количество чисел как ..
элементов вектора-строки X = ');
// Количество элементов вектора
lenX = length(X)
// Сумма элементов вектора
sumX = sum(X)
      // Среднее значение элементов вектора
averageX = sumX / lenX
```

Замечание. Обратите внимание, каким образом в команде input выполнен ввод длинной строки.

Сохраните скрипт-файл в папке Lab1 под именем ave3.sce. Закройте редактор SciNotes.

Запустите скрипт на исполнение с помощью команды

```
--> exec(ave3.sce, 2)
```

Введите вектор-строку с произвольным количеством чисел (5…10). Убедитесь в правильности вычисленных данных.

2.5 Команда вывода на дисплей disp

2.5.1. Команда disp используется для отображения элементов переменной без отображения ее имени, или для отображения текста.

Формат команды:

disp (variable_name)

или

disp(‘text’)

Всякий раз, когда исполняется команда disp, выводимое сообщение появляется с новой строки.

Задание. Выполните несколько примеров вывода при помощи команды disp:

1) вывод значения скаляра:

-- > *x*1 = 10,

-- > *disp(x*1*)*

2) вывод элементов матрицы:

-- > *A* =[5 9 1; 7 2 4];

-- > *disp(A)*

3) вывод строки

-- > *disp* (‘The problem has no solution’)

-- > *disp* (‘Задача не имеет решения’)

2.5.2. Команда disp может использоваться в скрипт-файлах для вывода данных.

Задание. Отройте редактор SciNotes и запишите в него следующий скрипт:

// Скрипт вычисляет среднее значение трех переменных x1, x2, x3

*x*1 = *input* (‘Введите значение переменной x1 = ‘);

*x*2 = *input* (‘Введите значение переменной x2 = ‘);

*x*3 = *input* (‘Введите значение переменной x3 = ‘);

average = *(x*1 + *x*2 + *x*3*) / 3,*

disp(' '); // вывод пустой строки

disp (‘Среднее значение трех переменных равно: ‘)

disp(average)

Сохраните набранный скрипт-файл в папке Lab4 под именем ave4.sce. Закройте редактор SciNotes.

Запустите скрипт из командного окна с параметром 2, чтобы не отображались команды файла:

-- > *exec(ave*4*.sce,* 2*)*

Введите последовательно запрашиваемые значения переменных, например, x1 = 10, x2 = 20, x3 = 30.

Убедитесь в правильности выполнения программы.

2.5.3. Во многих случаях возникает необходимость выводить данные в виде таблицы. Так как с помощью команды disp может быть выведена только одна переменная, то необходимо выводимые в виде таблицы переменные определить как столбцы матрицы, которая будет являться аргументом в команде disp.

Задание. Наберите в командном окне программу, которая выводит на дисплей таблицу из двух столбцов: в левом столбце будут значения целых чисел, а в правом столбце – их квадраты.

```
--> x = 1 : 6;          // создать вектор-строку из шести чисел
--> y = x .^ 2;         // вычислить квадраты этих чисел
--> table(:, 1) = x';   // сформировать матрицу, у которой 1-й
                        //столбец будет иметь значения элементов
вектора x
--> table(:, 2) = y';   // сформировать матрицу, у которой 2-й
                        //столбец будет иметь значения элементов
вектора y
disp (‘ Числа:     Их квадраты:’)
disp(' ')
disp(table)
```

Объясните выведенные на дисплей данные.

2.6 Команда форматируемого вывода данных mprintf

Команда mprintf может быть использована для отображения вывода числовых данных или текста на дисплее. С помощью этой команды (в отличие от команды disp) вывод может быть отформатирован. Например, текст и числовые значения данных могут быть произвольно перемешаны и отображаться в одной строке. В дополнение формат отображения чисел может быть управляем.

2.6.1. Для вывода текста формат команды mprintf имеет вид:

```
mprintf(‘text as string’)
```

Задание. Выполните вывод в командное окно текста:

```
--> mprintf(‘The problem has no solution’)
--> mprintf(‘Задача не имеет решения’)
```

Если выводимая строка очень длинная, то можно разбить ее на несколько строк при выводе вставкой символа “ \n “ перед символом в строке, который должен начинать новую строку вывода.

Задание. Выполните вывод «длинной» строки:

--> mprintf(‘The problem has no solution.\n Please check the input data.’)

2.6.2. Для отображения смеси текста и числа (значения переменной) команда mprintf имеет формат:

mprintf(‘text %-5.2f additional text\n’, variable_name)

Здесь символ % указывает место, где число должно быть вставлено в текст, а -5.2f – это форматирующие элементы.

Подробную информацию о форматирующих элементах можно получить из справочной системы Scilab по команде:

help mprintf

Так, например, символ f означает вывод числа в формате с плавающей точкой, символ d – вывод числа в формате целого, .2f - вывод числа в формате с плавающей точкой и двумя дробными цифрами.

Задание. Выполните вывод числа π в разных форматах:

--> mprintf(‘Вывод целого числа: %d\n’, %pi)

--> mprintf(‘Вывод дробного числа: %f\n’, %pi)

--> mprintf(‘Вывод дробного числа: %.2f\n’, %pi)

Объясните полученные результаты вывода.

2.6.3. С помощью команды mprintf возможно вставить более одного числа (значения переменной) в текст. Это делается с помощью наборов знаков % с последующими элементами форматирования в тех местах текста, где числа должны быть вставлены.

В общем случае команда mprintf выглядит так:

mprintf(‘text …… %d ……. %f …….%f …text\n’, var1, var2, var3)

Задание. Введите в командное окно следующую программу:

--> x1 = 10; x2 = 20.2; x3 = 30.3333; x4 = 40.44444;

--> mprintf(‘x1 = %d, x2 = %d, x3 = %f, x4 = %.2f\n’, x1, x2, x3, x4)

Объясните полученные результаты вывода данных.

2.6.4. Во многих случаях возникает необходимость выводить данные в виде таблицы. Для этой цели переменные для выводимых данных нужно представить в виде векторов-столбцов.

Задание. Введите в командное окно программу, которая выводит таблицу значений целых чисел и их квадратов:

--> x = 1 : 6; // создание вектора-строки из целых чисел

--> y = x .^2; // вычисление квадратов чисел

// вывод таблицы чисел и их квадратов как элементов векторов-столбцов

--> mprintf(‘Число равно %d, его квадрат равен %d\n’, x’, y’)

Объясните полученные результаты вывода данных.

2.6.5. Команда mprintf часто используется для вывода данных в скрипт-файлах.

Задание. Необходимо создать скрипт-файл, в котором задать путем присваивания необходимые исходные данные. Записать выражения для решения задачи. Вывести результаты расчета и исходные данные в командное окно Scilab с помощью функции mprintf. Вывод осуществить так, как указано в задании. вывести все заданные и рассчитанные числовые значения с двумя знаками после десятичной точки. Вывести рядом с числовыми значениями название единиц измерения.

Условие задачи. Задать мощности P1, P2, P3 трех потребителей электроэнергии на участке промышленного предприятия и их координаты x1, y1, x2, y2, x3, y3. Рассчитать координаты x0, y0 центра нагрузок по формулам:

$$x_0 = \frac{P_1 x_1 + P_2 x_2 + P_3 x_3}{P_1 + P_2 + P_3};$$

$$y_0 = \frac{P_1 y_1 + P_2 y_2 + P_3 y_3}{P_1 + P_2 + P_3}.$$

Вывести рассчитанные и исходные значения в виде:

КООРДИНАТЫ ЦЕНТРА ЭЛЕКТРИЧЕСКИХ НАГРУЗОК:

$x0 = 21{,}02м \qquad y0 = 34{,}03м$

ИСХОДНЫЕ ДАННЫЕ:

Номер	X, м	Y, м	P, кВт
1	13.50	45.00	55.80
2	20.00	32.00	17.40
3	35.00	15.60	31.30

Откройте редактор SciNotes и наберите следующий текст:

```
// Задаем исходные данные
x1 = 13.5; y1 = 45; P1 = 55.8;
x2 = 20; y2 = 32; P2 = 17.4;
x3 = 35; y3 = 15.6; P3 = 31.3;
// Выполняем расчет
x0 = (P1 * x1 + P2 * x2 + P3 * x3) / (P1 + P2 + P3);
y0 = (P1 * y1 + P2 * y2 + P3 * y3) / (P1 + P2 + P3);
// Выводим в командное окно
mprintf('\n')
mprintf('КООРДИНАТЫ ЦЕНТРА ЭЛЕКТРИЧЕСКИХ НАГРУЗОК:\n')
mprintf('----------------------------------------------------------------\n')
mprintf('x0 = %.2f м   y0 = %.2f м\n', x0, y0)
mprintf('\n')
mprintf('ИСХОДНЫЕ ДАННЫЕ:\n')
mprintf('----------------------------------------------------------------\n')
mprintf('Номер      X, м      Y, м      P, кВт\n')
mprintf('    1      %.2f      %.2f      %.2f \n', x1,y1,P1)
mprintf('    2      %.2f      %.2f      %.2f \n', x2,y2,P2)
mprintf('    3      %.2f      %.2f      %.2f \n', x3,y3,P3)
mprintf('\n')
```

Сохраните скрипт-файл под именем koord.sce. Выполните скрипт из окна редактора без отображения команд. Сравните выведенный текст с заданным. Если будут существенные отличия, то проведите корректировку скрипта.

3 Задания для самостоятельной работы

Указание. Отчет по лабораторной работе № 4 выполняется в виде дневника, т.е. содержимого командного окна Scilab в ходе выполнения заданий для самостоятельной работы. Файл дневника создается в папке Lab4, рекомендуемое имя для файла дневника – self_work.txt. Каждое задание должно в начале иметь строку комментария, например, вида:

// Задание 1

Тексты заданий приводить не нужно.

Задание 1. Создайте скрипт-файл для решения следующей задачи.

Условие задачи. Для динамической колебательной системы, содержащей груз с массой m, пружину с коэффициентом жесткости k и демпфер с коэффициентом демпфирования c, нужно вычислить резонансную частоту системы по формуле:

$$f_r = f_n\sqrt{1-2s^2},$$

где коэффициент затухания системы s вычисляется по формуле:

$$s = \frac{c}{2\sqrt{km}}.$$

Собственная частота незатухающих колебаний системы рассчитывается по формуле:

$$f_n = \frac{1}{2\pi}\sqrt{\frac{k}{m}}$$

Подсказка. Исходными данными для задачи являются:

m – масса системы;

k – коэффициент жесткости пружины;

c – коэффициент демпфирования.

Эти параметры вводятся с помощью трех команд input, в которых должен быть текст вида: “Задайте массу системы: “.

Промежуточными являются параметры:

s – коэффициент затухания системы;

fn – собственная частота незатухающих колебаний.

Результаты вычисления этих параметров не отображаются на дисплее.

Результатом выполнения программы должен быть параметр:

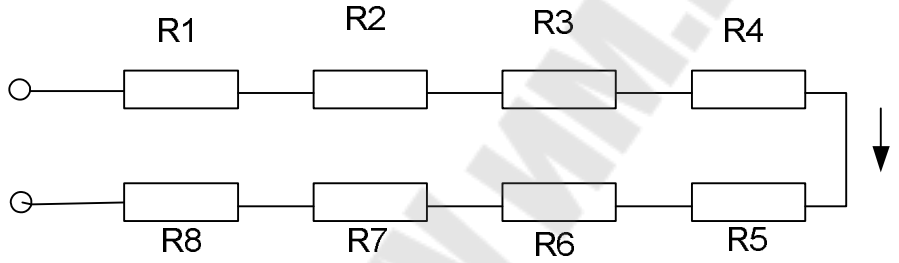
fr – резонансная частота системы.

Этот параметр должен выводиться на дисплей посредством команды disp. Перед выводом значения fr необходимо вывести на дисплей текст вида: “Резонансная частота системы равна: “.

Разработанный скрипт-файл сохраните в папке Lab4 под именем, например, zadanie1.sce. Затем закройте редактор SciNotes.

Запустите скрипт из командного окна без отображения строк программы. Введите для тестирования параметры: m = 10, k = 10, c = 0.6. Должно получиться fr = 0.1590.

Задание 2. Имеется электрическая цепь с 8-ю последовательно соединенными резисторами



Падение напряжения на отдельном резисторе:

$$U_n = R_n I = \frac{R_n}{R_{eq}} U_s ,$$

где U_s – напряжение источника питания;

$I = U_s / R_{eq}$ – ток в цепи;

$R_{eq} = \sum R_n = R1 + R2 + R3 + R4 + R5 + R6 + R7 + R8$ – эквивалентное (общее) сопротивление цепи.

Мощность, рассеиваемая в каждом резисторе, равна:

$$P_n = I^2 R_n = \frac{R_n}{R_{eq}} U^2$$

Требуется написать программу (скрипт), которая вычисляет падение напряжения на каждом резисторе и мощность, рассеиваемую в нем. Когда скрипт начинает исполняться, выдается запрос пользователю сначала ввести величину напряжения источника, а

затем ввести сопротивления резисторов как элементов вектора-строки.

Программа выводит таблицу со значениями сопротивлений резисторов в первом столбце; падением напряжения на резисторах во втором столбце; мощность, рассеиваемую на резисторах, в третьем столбце. Вслед за таблицей программа выводит величину тока в цепи и общую рассеиваемую мощность.

Исходные данные для расчета:

Us = 24 В, R1 = 20 Ом, R2 = 14 Ом, R3 = 12 Ом, R4 = 18 Ом, R5 = 8 Ом, R6 = 15 Ом, R7 = 10 Ом, R8 = 16 Ом.

Разработанный скрипт-файл сохраните в папке Lab4 под именем, например, zadanie2.sce. Затем закройте редактор SciNotes.

Запустите скрипт из командного окна без отображения строк программы.

Задание 3. Зная величины r12, r23, r31 треугольника сопротивлений требуется рассчитать значения сопротивлений r1, r2, r3 эквивалентной звезды:

$$r_1 = \frac{r_{12}r_{31}}{r_{12}+r_{23}+r_{31}}, \quad r_2 = \frac{r_{23}r_{12}}{r_{12}+r_{23}+r_{31}}, \quad r_3 = \frac{r_{23}r_{31}}{r_{12}+r_{23}+r_{31}}.$$

Вывести исходные и расчетные значения в виде:

СОПРОТИВЛЕНИЯ ТРЕУГОЛЬНИКА:

---

r12 = 3.40 Ом
r23 = 5.50 Ом
r31 = 2.90 Ом

СОПРОТИВЛЕНИЯ ЭКВИВАЛЕНТНОЙ ЗВЕЗДЫ:

---

r1 = 0.84 Ом
r2 = 1.58 Ом
r3 = 1.35 Ом

Решение задачи оформите в виде скрипт-файла с именем, например, zadanie3.sce. Выполните скрипт из окна редактора без

отображения команд. Сравните выведенный текст с заданным. Если будут существенные отличия, то проведите корректировку скрипта.

Задание 4. Задать высоту H, внешний R и внутренний r радиусы цилиндрического кольца. Вычислить его объем V и площадь S всей поверхности по формулам:

$$S = 2\pi H(R + r) + 2\pi(R^2 - r^2),$$

$$V = \pi H(R^2 - r^2).$$

Вывести заданные и рассчитанные значения в виде:

ОТВЕТ ЗАДАЧИ:

V = 54.924 куб.см
S = 219.733 кв.см

ЗАДАЧА РЕШЕНА

ИСХОДНЫЕ ДАННЫЕ: H = 5.49 см
R = 3.17 см
r = 2.62 см

Решение задачи оформите в виде скрипт-файла с именем, например, zadanie4.sce. Выполните скрипт из окна редактора без отображения команд. Сравните выведенный текст с заданным. Если будут существенные отличия, то проведите корректировку скрипта.

4 Содержание отчета

Отчет по лабораторной работе № 4 выполняется в виде списка скрипт-файлов, разработанных в ходе выполнения заданий для самостоятельной работы.

Разработанный дневник необходимо продемонстрировать преподавателю для оценки объема выполненной работы и анализа качества выполнения заданий.

Контрольные вопросы

1. Какими способами можно загрузить скрипт-файл в рабочее пространство Scilab?
2. Для каких целей применяется команда input( )?
3. Для каких целей используется команда disp( )?
4. Каким образом выполняется форматируемый вывод данных в Scilab?
5. Как можно вывести на дисплей «длинную» строку?
6. Как можно вывести на дисплей целое число?
7. Как можно вывести на дисплей дробное число с двумя цифрами после точки?

Лабораторная работа № 5

ПОСТРОЕНИЕ И ОФОРМЛЕНИЕ ГРАФИКОВ ФУНКЦИЙ

1 Цель работы

Изучить основные функции интегрированной среды Scilab для построения двумерных графиков. Изучить и исследовать возможности оформления графиков и графических окон. Приобрести навыки построения графиков в Scilab.

2 Порядок выполнения работы

2.1 Создание папки для работы

При выполнении лабораторной работы № 5 вы будете использовать папку с именем Lab5, которую необходимо предварительно создать.

С этой целью откройте вашу рабочую папку (с вашей фамилией) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab5.

В дальнейшем вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 5. Полный путь к этой папке будет такой:

F:\OTC\Ivanov\Lab5

2.2 Запуск и настройка среды разработки Scilab

2.2.1. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа – рабочий стол среды разработки Scilab.

2.2.2. Нужно установить в качестве активной (текущей) папку Lab5. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем ОТС, затем папку с вашей фамилией (условно Ivanov) и затем папку Lab5. В

строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab5. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab5

2.2.3. При необходимости очистите командное окно и журнал команд, использую пункты меню Правка.

2.3 Исследование функции plot

2.3.1. Для построения двумерных графиков функции одной переменной вида $y = f(x)$ в Scilab существует команда plot, обращение к которой в простейшем случае выглядит следующим образом:

plot(x, y)

Здесь x – список значений независимой переменной, а y – список значений функции f(x) в этих точках. Следовательно, для того, чтобы построить график в Scilab, необходимо создать два списка: значений переменной и значений функции в этих точках.

При этом следует учесть, что график, построенный при помощи функции plot, представляет собой ломаную линию, соединяющую значения функции, вычисленные в заданных точках (x, y). Значит, чтобы получить гладкий график, необходимо вычислить значение функции в большом количестве точек.

Задание. Для построения графика функции $y = e^{\cos x}$ на промежутке [-2π; 2π] с шагом 0.1 наберите в командном окне Scilab следующее:

--> *x = -2*%pi : 0.1 : 2*%pi*;

--> *y = exp(cos(x))*;

--> *plot(x, y)*

2.3.2. Для построения нескольких графиков в одном окне можно использовать обращение к функции plot следующего вида:

plot(x1, y1, x2, y2, x3, y3, …)

Здесь x1, y1 – список значений переменной и первой функции, x2, y2 – второй, x3, y3 – третьей и так далее. Таким образом строятся графики практически какого угодно количества функций.

Задание. Для построения графиков функций $y = e^{\cos x}$ и $z = \sin x$ на промежутке [-2π; 2π] с шагом 0.1 введите в командное окно Scilab:

--> *x = -2*%pi : 0.1 : 2*%pi*;

--> *y = exp(cos(x))*;

--> *z = sin(x)*;

--> *plot(x, y, x, z)*

2.4 Видоизменение графиков при помощи функции plot

Вид графика можно изменять, добавив при обращении к функции plot помимо основных, еще один аргумент – строку, состоящую из трех символов, определяющих цвет линии, тип маркера, которым будет нарисован график, и тип линии. Обращение к функции plot в этом случае будет выглядеть так:

plot(x1, y1, string1, x2, y2, string2, …)

Строка string выглядит следующим образом:

‘параметр1параметр2параметр3’

Параметры пишутся один за другим без разделителей.

Параметр 1 определяет цвет графика:

Таблица 4

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

Параметр 2 задает маркер для рисования графика:

Таблица 5

Символ	Описание маркера
.	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

Параметр 3 устанавливает тип линии графика:

Таблица 6

Символ	Описание
-	Сплошная
:	Штрих-пунктирная
- -	Штриховая

Если один из символов не указан, его значение выбирается по умолчанию (как правило, синяя сплошная линия). При неуказанном типе символа – значение будет отсутствовать.

Задание. Выполните построение трех графиков:

1) $f1(x) = |3x|$ на интервале [-3; 3] – сплошная синяя линия;

2) $f2(x) = 8\sin x$ на интервале $[-\pi; \pi]$ – пунктирная красная линия;

3) $f3(x) = e^x / 10$ на интервале [-1; 4] – график, нарисованный зелеными кружочками.

Шаг изменения аргументов функций примите 0.2.

Наберите в командном окне следующее:

--> *x*1 = -3 : 0.2 : 3; *f*1 = *abs(*3 * *x*1*)*;

--> *x*2 = -*%pi* : 0.2 : *%pi*; *f* 2 = 8 * sin*(x*2*)*;

--> *x*3 = -1 : 0.2 : 4; *f* 3 = exp*(x*3*)* / 10;

--> *plot(x*1, *f* 1, '*b*-', *x*2, *f* 2, '*r*–', *x*3, *f* 3, '*go*' *)*

Убедитесь в правильности работы программы.

2.5 Оформление графиков

Оформление графиков выполняется следующими средствами.

Для вывода названия графика и осей используется команда:

xtitle(‘plot_name’, ‘x_name’, ‘y_name’),

где plot_name – название (заголовок) графика, x_name – название оси X, y_name – название оси Y.

Чтобы отобразить сетку на графике, следует воспользоваться оператором

xgrid(n)

Здесь n – целое число, отвечающее за цвет сетки. По умолчанию, если параметр n не указан, то сетка имеет черный цвет.

Описание линий (так называемая легенда) осуществляется командой

legend(line1, line2, …, linen, place, frame),

где line1 и другие – названия графиков;

place – местоположение описания:

1 – верхний правый угол графического окна,

2 – верхний левый угол,

3 – нижний левый угол,

4 – нижний правый угол,

5 – определяется пользователем после изображения графика;

frame – задает рамку для описания: %t – описание заключено в рамку, %f – рамки нет.

Задание. Нужно построить график функций f1(x) = sin(x) и f2(x) = cos(x) на промежутке [-2π; 2π] с шагом 0.2. Оформите его, подписав сам график, оси и отобразив описание линий и сетку.

Для разработки программы удобно создать скрипт-файл:

```
x = -2*%pi : 0.2 : 2*%pi;
f1 = sin(x);
f2 = cos(x);
plot(x, f1, x, f2);
xtitle ('Графики функций f1(x) и f2(x)', 'Ось X', 'Ось Y');
xgrid( );
legend('f1(x)', 'f2(x)', 3, %f);
```

Сохраните скрипт-файл в папке Lab5 под именем, например, grafic1.sce. Запустите скрипт из окна редактора. Убедитесь в правильности работы программы.

2.6 Построение нескольких графиков в одном графическом окне

В Scilab можно выводить несколько графиков в одном окне, не совмещая их в одних координатных осях. Например, если графическое окно должно содержать 4 самостоятельных графика, все окно разбивается на 4 области (подокна), а затем в каждую из них выводится график функции.

Наиболее простым способом изображения нескольких графиков в одном окне является использование функции subplot. Обращение к ней имеет вид:

```
subplot(m, n, p)
```

или

```
subplot(mnp)
```

Выполнение этой функции приводит к тому, что графическое окно разбивается на m окон по вертикали и n окон по горизонтали, текущим окном становится окно с номером p. Для формирования графика в каждом окне можно использовать функцию plot.

Пример. Построить графики шести функций: y = sin(x), z = cos(x), u = cos(sin(x), v = sin(cos(x), w = exp(sin(x)), r = exp(cos(x)) в одном графическом окне, каждый в своей системе координат. Аргумент x изменяется на интервале [-10; 10] с шагом 0.01.

С помощью функции subplot разбиваем графическое окно на 6 областей. Примем, что в каждом столбце по вертикали должно быть

3, а по горизонтали 2 области для вывода графиков. Третье число в записи функции указывает, в какую из областей (счет ведется по порядку: слева направо и сверху вниз) выводится график, формируемый функцией plot(x, y).

Задание. Напишите программу для рассмотренного примера. В программе ожидается много команд, поэтому удобно оформить ее в виде скрипт-файла. С этой целью откройте редактор SciNotes и запишите текст:

```
// Скрипт формирования шести графиков функций
x = [-10 : 0.01 : 10];
y = sin(x), z = cos(x);
u = cos( sin(x)), v = sin( cos(x));
w = exp( sin(x)); r = exp( cos(x));
subplot(3, 2, 1);
plot(x, y);
subplot(3, 2, 2);
plot(x, z);
subplot(3, 2, 3);
plot(x, u);
subplot(3, 2, 4);
plot(x, v);
subplot(3, 2, 5);
plot(x, w);
subplot(3, 2, 6);
plot(x, r);
```

Сохраните скрипт-файл в папке Lab5 под именем, например, grafic2.sce.

Затем запустите скрипт из окна редактора. По содержимому графического окна убедитесь в правильности решения поставленной задачи.

2.7 Функция plot2d

Функция plot2d предоставляет пользователю больше возможностей для оформления графиков. Простейшее обращение к этой функции осуществляется командой

plot2d(x, y)

За внешний вид графика отвечают ключи команды, задаваемые в виде key = value:

plot2d(x, y, key1=value1, key2=value2, key3=value3, …)

Возможные значения ключей:

1) style – определяет цвет графика, который задается указанием RGB-кода: style = color(r, g, b). Каждое из значений r, g, b соответствует «количеству» красного, зеленого и синего, и может меняться в пределах от 0 до 255. Есть более простая возможность – указать непосредственно имя цвета: style = color('имя цвета'). Наиболее распространенные названия:

Таблица 7

yellow	желтый	blue	синий
cyan	голубой	black	черный
red	красный	brown	коричневый
green	зеленый	grey	серый

2) rect – устанавливает размер окна вокруг графика: значение этого ключа задается в виде вектора $rect = [x\min, y\min, x\max, y\max]$, где xmin, xmax – интервал отображения графического окна по оси x, ymin, ymax – по оси y.

3) axesflag – задает наличие рамки вокруг графика; базисные значения этого параметра: 0 – нет рамки, 1 – изображение рамки, ось слева, 2 – изображение рамки без подписи осей, 3 – изображение осей, ось справа, 5 – изображение осей в виде перекрестка в начале координат. По умолчанию принято значение 1.

4) nax – устанавливает число значений координатных осей; его значение представляет собой массив $nax = [nx, Nx, ny, Ny]$, где Nx (Ny) – число основных делений оси X (Y), которые подписаны на графике, nx (ny) – число промежуточных делений.

5) leg – определяет легенду для каждого графика; задается в виде

leg = ‘leg1@leg2@leg3…’, где leg1 – описание первого графика, leg2 – второго и так далее.

Пример. Построение графика функции f(x) = exp(sin(x)) на промежутке [-2π; 2π] с шагом 0.2 со следующим оформлением: линия графика зеленого цвета; размеры графического окна: от -10 до 10 по оси x и от -1 до 4 по оси y; оси должны быть изображены в виде перекрестия; количество основных меток: 11 по оси x и 6 по оси y, вспомогательных – по 2 для обеих осей; отобразить легенду.

Задание. Наберите в командном окне программу для построения графика из рассмотренного примера:

--> *x = -2*%pi : 0.2 : 2*%pi; f = exp(sin(x));*

--> *plot2d(x, f, style = color(' green'), rect = [-10, -1, 10, 4], …*

--> *axesflag = 5, nax = [2, 11, 2, 6], leg = ' f(x) = exp(sin(x))'*

Сравните полученный график с заданным в примере.

2.7 Построение точечных графиков

Многие прикладные задачи (например, обработка и анализ экспериментальных данных) требуют построения точечного графика. Для этой цели можно использовать функцию plot2d в следующей форме:

plot2d(x, y, d),

где x, y – массивы координат точек, которые нужно отобразить на графике, d – отрицательное число, определяющее тип маркера:

Таблица 8

Число	Описание
-0	Точка
-1	Плюс
-2	Крестик
-3	Плюс, вписанный в окружность

Окончание таблицы 8

-4	Закрашенный ромб

-5	Незакрашеный ромб
-6	Треугольник вершиной вверх
-7	Треугольник вершиной вниз
-8	Плюс, вписанный в ромб
-9	Кружок
-10	Звездочка
-11	Квадрат
-12	Треугольник вершиной вправо
-13	Треугольник вершиной влево
-14	Пятиконечная звезда

Массивы координат могут быть вычислены или введены непосредственно.

Пример 1. Построить точечный график функции $y = \sin(x)$ с типом маркера «плюс, вписанный в ромб». Диапазон изменения аргумента $-2\pi \le x \le 2\pi$ с шагом 0.25.

Задание. Введите в командное окно программу для построения графика:

--> *x = -2*%pi : 0.25 : 2*%pi*;

--> *y = sin(x)*;

--> *plot2d(x, y, -8)*

Сравните полученный график с заданным в примере 1.

Пример 2. Требуется построить точечный график функции $y = \sin(x)$ на промежутке [0; 3] с шагом 0.15 крестиками и нанести на этот же график точки с координатами (0.1;0.3), (0.4;0.8), (1;0.6), (2;0.1), (2.5;0.4) кружками.

Задание. Наберите в командном окне программу для построения графика из примера 2:

--> *x = 0 : 0.15 : 3; y = sin(x)*;

--> *x*1 = [0.1, 0.4, 1, 2, 2.5]; *y*1 = [0.3, 0.8, 0.6, 0.1, 0.4];

--> *plot2d(x, y, -2)*

--> *plot2d(x*1, *y*1, *-9)*

Сравните полученные графики с заданными в примере 2.

3 Задания для самостоятельной работы

Указание. Отчет по лабораторной работе № 5 выполняется в виде списка скрипт-файлов, разработанных в ходе выполнения заданий для самостоятельной работы.

Задание 1. Разработайте скрипт для построения графика функции

$$y = 3.5^{-0.5x} \cos 6x$$

для диапазона $-2 \le x \le 4$ с шагом 0.01. Используйте функцию plot. Скрипт-файл сохраните в папке Lab5 под именем zadanie1.sce. Проверьте работу скрипта, запустив его из окна редактора.

Задание 2. Разработайте скрипт для построения графиков функций

$$y1(x) = x \ln \frac{1}{|x|}, \quad y2(x) = 1 + 10 \frac{\sin x}{x}$$

на интервале $0.3 \le x \le 10$ с шагом 0.1. Линия графика y1 – красного цвета, сплошная, линия графика y2 – зеленого цвета, пунктирная. Сделайте заголовок графика и заголовки осей. Нанесите сетку черного цвета. Разместите легенду в правом верхнем углу с рамкой.

Разработанный скрипт-файл сохраните в папке Lab5 под именем zadanie2.sce. Проверьте работу скрипта, запустив его из окна редактора.

Задание 3. Разработайте скрипт для построения четырех графиков функций

$$y = \sin(2x),\ z = \cos(3x),\ u = \cos(\sin(2x)),\ v = \sin(\cos(3x))$$

в одном графическом окне, каждый в своей системе координат. Примите, что x изменяется на интервале [-10 : 10] с шагом 0.01. Скрипт-файл сохраните в папке Lab5 под именем zadanie3.sce. Проверьте работу скрипта, запустив его из окна редактора.

4 Содержание отчета

Отчет по лабораторной работе № 5 выполняется в виде списка скрипт-файлов, разработанных в ходе выполнения заданий для самостоятельной работы.

Разработанные скрипты необходимо продемонстрировать преподавателю для оценки объема выполненной работы и анализа качества выполнения заданий.

Контрольные вопросы

1. Как с помощью функции plot( ) вывести несколько графиков в одном графическом окне?
2. Как можно задать цвет и тип линии графика в функции plot( )?
3. Каким способом можно задать заголовок графика, выведенного с помощью функции plot( )?
4. Как сделать надписи на осях графика, выведенного с помощью функции plot( )?
5. Какие возможности для вывода имеет функция plot2d( )?
6. Как можно построить график точечной функции в Scilab?

Лабораторная работа № 6

ПРОГРАММИРОВАНИЕ В SCILAB

1 Цель работы

Изучить и практически исследовать методы программирования в интегрированной среде разработки Scilab.

2 Порядок выполнения работы

2.1 Создание папки для работы

При выполнении лабораторной работы № 6 Вы будете использовать папку с именем Lab6, которую необходимо предварительно создать.

С этой целью откройте Вашу рабочую папку (с Вашей фамилией) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab6.

В дальнейшем Вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 6. Полный путь к этой папке будет такой:

F:\OTC\Ivanov\Lab6

2.2 Запуск и настройка среды разработки Scilab

2.2.1. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа – рабочий стол среды разработки Scilab.

2.2.2. Нужно установить в качестве активной (текущей) папку Lab6. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем ОТС, затем папку с вашей фамилией (условно Ivanov) и затем папку Lab6. В строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab6. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab6

2.3 Операторы отношения и логические

2.3.1. Операции отношения сравнивают между собой два операнда по величине. Способы задания этих операций:

<	Меньше
<=	Меньше или равно
>	Больше
>=	Больше или равно
==	Равно
<> или ~=	Не равно

Истинность операции отношения в Scilab обозначается символом %T или %t (True – истина), а ложность – символом %F или %f (False – ложь). Для того, чтобы можно было использовать операции отношения в арифметических выражениях, принимается, что при истинности операции ее величина (то есть результат выполнения выражения) равна 1, а при ложности – равна 0. Операции отношения имеют более низкий приоритет, чем арифметические операции. Изменить порядок вычисления можно с помощью круглых скобок.

Задание. Выполните несколько примеров в командном окне Scilab:

1)
--> 5 > 8
2)
--> a = 5 < 10
3)
--> 3 + 4 < 16 / 2
4)
--> 3 + (4 < 16) / 2
5)
--> y = (6 < 10) + (7 > 8) + (5 * 3 == 60 / 4)

Объясните полученные результаты в каждом примере.

2.3.2. Логические операции над вещественными числами обозначаются следующими символами:

&	Логическое И
|	Логическое ИЛИ
~	Логическое НЕ

Первые две операции являются двухоперандными (бинарными), а операция НЕ является унарной (одноопернадной). Логические операции трактуют свои операнды как «истинные» (не равные нулю) или «ложные» (равные нулю). Результат логической операции: истина (обозначается как %T, а ее значение равно 1) или ложь (обозначается как %F, а ее значение равно 0).

В одном выражении можно использовать арифметические, логические и операции отношения. Последовательность выполнения операций определяется их расположением внутри выражения, их приоритетом и наличием круглых скобок.

Задание. Выполните несколько примеров в командном окне Scilab:

1)
--> 3 & 7
2)
--> a = 5 | 0
3)
--> ~25
4)
--> t = 25 * ((12 & 0) + (~ 0) + (0 | 5))
5)
--> x = -2; y = 5;
--> -5 < x < -1
6)
--> (-5 < x) & (x < -1)
7)
--> (y >= 8) | (x < -1)

8)
--> ~ (y >= 8) | (x < -1)

Объясните полученные результаты.

Операции отношения и логические применимы и к массивам (векторам и матрицам). Эти операции выполняются поэлементно, поэтому массивы (операнды) должны быть одинаковой размерности.

Задание. Выполните несколько примеров в командном окне Scilab:

1)
--> b = [15 6 9 4 11 7 14];
--> c = [8 20 9 2 19 7 10];
--> b == c
2)
--> b <> c
3)
--> b – c > 0
4)
--> A = [2 9 4; -3 5 2; 6 7 -1]
--> B = A <= 2

Объясните полученные результаты для каждого примера.

2.3.3. Функция find(<test v>) возвращает индексы элементов вектора, удовлетворяющих условию теста.

Задание. Выполните следующие примеры в командном окне Scilab:

1) чтобы найти все элементы вектора w, которые меньше, чем 5:
--> w = [1, 5, 3, 8, 15, 7, 3, 2, 12, 6];
--> find(w<5)
Убедитесь в правильности выполнения поиска.
2) чтобы найти все элементы вектора w, равные 3:
--> find(w==3)
Убедитесь в правильности выполнения поиска.

2.4 Операторы управления программой

Операторы управления программой – это средства, с помощью которых можно изменять порядок выполнения программы. Они позволяют выполнять ветвление, циклическое повторение одного или нескольких операторов, передачу управления в нужное место кода программы.

Язык Scilab (Sci-язык) предоставляет три категории операторов управления программой:

1) операторы выбора – это if и select;
2) операторы цикла – это for и while;
3) операторы перехода – это break и continue.

2.5 Исследование операторов выбора

Sci-язык поддерживает два типа операторов выбора: if (если) и select (выбор). Эти операторы позволяют проверять выполнение определенных условий и выбирать возможное продолжение вычислительного процесса. Возможны следующие конструкции этих операторов:

1) оператор if единственного выбора;
2) оператор if … else двойного выбора;
3) оператор if … elseif … else множественного выбора;
4) оператор select множественного выбора.

2.5.1. Оператор if единственного выбора для одного выполняемого оператора имеет вид:

```
if условие then
    оператор
end
```

Для нескольких (блока) выполняемых операторов:

```
if условие then
оператор1
оператор2
…….
операторN
end
```

В случае истинности условия выполняются операторы, следующие за then. Если условие ложно, то операторы не выполняются.

Замечание. Ключевое слово then может не использоваться, если после него вводится ENTER.

Задание. Наберите в командном окне программу для исследования оператора if:

```
--> x = 1;
--> if x > 0 then
-->    x = 10;
--> end
--> x
```

Так как начальное значение x больше 0, то условие истинно, и выполняется оператор присвоения x = 10.

Теперь исследуйте работу программы при начальном значении x = -1:

```
--> x = -1;
--> if x > 0 then
-->    x = 10;
--> end
--> x
```

Так как в этом случае начальное значение x меньше 0, то условие ложно, и оператор присвоения x = 10 не выполняется. Поэтому начальное значение x не изменится.

2.5.2. Оператор if … else двойного выбора обеспечивает две альтернативы продолжения выполнения программы. Выбор осуществляется исходя из проверяемого условия.

Общий вид оператора if …else двойного выбора для одного выполняемого оператора:

```
if условие then
оператор1
else
оператор2
end
```

Вместо одиночных операторов 1 и 2 могут быть блоки операторов. В случае истинности условия выполняется оператор1, если условие ложно, то выполняется оператор 2.

Пример. Заданы два числа a и b. Вычислить их сумму, если a > b, и их произведение в противном случае.

Задание. Наберите в командном окне следующую программу:

```
--> a = 2; b = 6;
--> if a > b then
-->    c = a + b;
--> else
-->    c = a * b;
--> end
-->c
```

Объясните полученный результат.

Теперь исследуйте работу программы при других значениях входных параметров:

```
--> a = 8; b = 6;
--> if a > b then
-->    c = a + b;
--> else
-->    c = a * b;
--> end
--> c
```

Объясните полученный результат.

В случае сложных условий составляющие его элементарные условия заключаются в скобки.

Пример. Заданы два числа a и b. Вычислить их сумму, если одновременно a > 0 и b > 0, и их произведение в противном случае.

Задание. Наберите в командном окне следующую программу:

```
--> a = 2; b = 3;
--> if (a > 0) & (b > 0) then
-->    c = a + b;
```

```
--> else
-->    c = a * b;
--> end
--> c
```

Объясните полученный результат.

2.5.3. Общий вид оператора if … elseif … else множественного выбора:

```
if условие1 then
        оператор1
elseif условие2 then
        оператор2
……..
elseif условиеN then
        операторN
else
        оператор(N+1)
end
```

Последняя else-часть в операторе может и отсутствовать.

Оператор if … elseif множественного выбора выполняет серию последовательных проверок до тех пор, пока не будет установлено следующее:

1. Одно из условий в if-части или в частях elseif является истиной. В этом случае выполняются соответствующие им операторы.

2. Ни одно из вложенных условий не является истиной. Программа выполнит операторы в последней else-части, если она имеется.

Пример. Заданы два числа a и b. Вычислить их сумму, если одновременно a > 0 и b > 0, или произведение, если одно из них отрицательно, или среднее значение, если отрицательны оба.

Задание. Для разработки программы удобно создать скрипт-файл в редакторе SciNotes:

```
a = input('Введите значение a = ');
```

```
b = input('Введите значение b = ');
if (a > 0) & (b > 0) then
        c = a + b;
elseif ((a > 0) & (b < 0)) | ((a <0) & (b > 0)) then
        c = a * b;
else
        c = (a + b) / 2;
end
disp(c)
```

Сохраните набранный скрипт-файл в папке Lab6 под именем primer1.sce.

Выполните тестирование скрипта в следующей последовательности.

1. Перейдите в командное окно и запустите скрипт командой:

--> exec('primer1.sce', 2)

Введите последовательно запрашиваемые значения переменных:

a = 2; b = 3;

По выведенному результату переменной с убедитесь в правильности выполнения программы.

2. Вновь запустите скрипт командой:

--> exec('primer1.sce', 2)

Введите последовательно запрашиваемые значения переменных:

a = -2; b = 3;

По выведенному результату переменной с убедитесь в правильности выполнения программы.

3. Вновь запустите скрипт командой:

--> exec('primer1.sce', 2)

Введите последовательно запрашиваемые значения переменных:

a = -2; b = -3;

По выведенному результату переменной с убедитесь в правильности выполнения программы.

2.5.4. Оператор select используется для выбора одного варианта из многих. Он проверяет, совпадает ли значение выражения с одним из значений, входящих в некоторое множество констант, и выполняет соответствующий этому значению выбор.

Общий вид оператора select:

```
select выражение
        case константа1 then
        оператор1
        case константа2 then
                оператор2
        ……..
        case константаN then
                операторN
        else
                оператор(N+1)
end
```

Важное замечание. В связи с тем, что в ветвях, помеченных словом case, можно указывать только константу, для анализа принадлежности к диапазону значений следует использовать оператор if … elseif множественного выбора.

Задание. Создайте скрипт-файл в редакторе SciNotes:

```
n = input('Введите целое число от 1 до 3 : ');
select n
        case 1 then
                disp('Один');
        case 2 then
                disp('Два');
        case 3 then
                disp('Три');
        else
                disp('Иное значение');
end
```

Сохраните набранный скрипт-файл в папке Lab6 под именем primer2.sce.

Выполните тестирование скрипта в следующей последовательности.

1. Перейдите в командное окно и запустите скрипт командой:

```
--> exec('primer2.sce', 2)
```

Введите число 2.

По выведенной строке убедитесь в правильности выполнения программы.

2. Вновь запустите скрипт командой:

--> exec('primer2.sce', 2)

Введите число 5.

По выведенному сообщению убедитесь в правильности выполнения программы.

2.6 Исследование операторов цикла

В Sci-языке имеются два оператора цикла – это for и while.

2.6.1. Оператор цикла for предназначен для программирования циклических алгоритмов, когда переменная цикла явно выражена и изменяется от начального значения до конечного с постоянным шагом.

Общий вид оператора цикла for:

```
for x = xn : dx : xk
   операторы
end
```

Здесь x – переменная цикла;
xn – начальное значение;
dx – шаг изменения переменной цикла;
xk – конечное значение.

Если dx =1, то его можно не писать:

```
for x = xn : xk
   операторы
end
```

Задание. Наберите в командном окне пример цикла for:

```
--> for k = 1 : 3 : 10     // Значение переменной цикла k будет: 1, 4, 7, 10
--> k                      // Вывод на дисплей значения k
--> x = k ^ 2              // Вычисление и вывод значения x
--> end                    // Конец цикла
```

По полученным результатам сделайте вывод о работе программы.

2.6.2. Оператор цикла while предназначен для программирования циклических алгоритмов, где проверка условия повторения цикла выполняется перед выполнением рабочей части цикла.

Общий вид оператора:

```
while логическое_выражение
   операторы
end
```

Порядок выполнения оператора следующий:

- Проверяется истинность логического выражения.
- До тех пор, пока оно истинно, выполняется оператор рабочей части цикла.
- Если логическое выражение стало ложным, то выполняется следующий за оператором цикла оператор программы.

Пример. Вывести на экран дисплея четные числа от 2 до 12.

Задание. Наберите в командном окне следующую программу:

```
--> k = 2;
--> while k <= 12
-->    k = k +2
-->end
```

Объясните полученные результаты работы программы.

2.7 Операторы перехода break и continue

2.7.1. Оператор break применяется для принудительного окончание цикла, минуя стандартную проверку условия. Когда оператор break встречается в теле цикла, цикл немедленно заканчивается и выполнение программы переходит на строку, следующую за циклом.

Рассмотрим пример использования оператора break для досрочного прекращения работы цикла for:

```
x = 1;
for k = 1 : 20
    k                          // Вывод значения k
    x = 2 * k                  // Вычисление и вывод значения x
    if x > 10 then
       break
    end                        // Конец оператора if
end                            // Конец оператора for
```

Пока значение x не превышает 10, выполняется цикл for, в котором выводится на дисплей значение переменной цикла k, а также вычисляется и выводится переменная x. Как только значение x становится больше величины 10, Scilab заканчивает выполнение цикла for вне зависимости от достижения конечного значения переменной цикла k.

Задание. Наберите в командном окне предыдущую программу:

```
--> x = 1;
--> for k = 1 : 20
-->    k
-->    x = 2 * k
-->    if x > 10 then
-->           break
-->    end
--> end
```

Сделайте выводы по результатам работы программы.

2.7.2. Работа оператора continue чем-то похожа на работу оператора break. Но вместо форсированного окончания цикла оператор continue переходит к следующей итерации цикла, пропуская оставшийся код тела цикла.

Рассмотрим пример использования оператора continue в следующей программе:

```
x =1;
for k = 1 : 3
    k                  // Отображение значения k
    if k == 2
       continue
    end
    x = x + 1                  // Вычисление и отображение
значения x
end
```

Первоначально значение x = 1. На первой итерации (первом проходе цикла) k = 1, поэтому условие k == 2 ложно. Вследствие этого команда continue не выполняется. Scilab исполняет оператор x = x + 1. Теперь значение x равно 2. На второй итерации k инкрементируется и теперь равно k = 2. Поэтому условие k == 2 истинно. Команда continue теперь исполняется. Scilab пропускает оставшиеся команды тела цикла for, и поэтому оператор x = x + 1 не исполняется. Вследствие этого значение x остается равным 2. На третьей итерации k = 3, поэтому условие k == 2 ложно. Команда continue вследствие этого не выполняется. Scilab исполняет оператор x = x + 1. Теперь значение x равно 3.

Задание. Наберите в командном окне текст предыдущей программы:

```
--> x = 1;
--> for k = 1 : 3
-->    k
-->    if k == 2
-->           continue
-->    end
-->    x = x +1
--> end
```

Сделайте выводы по результатам работы программы.

2.8 Пример разработки скрипт-файла

Функция $f(x) = e^x$ может быть представлена рядом Тейлора в виде:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Требуется написать программу в виде скрипт-файла, которая использует представление e^x в виде ряда Тейлора. Программа вычисляет e^x путем добавления членов ряда и останавливается, когда абсолютное значение члена, который был добавлен последним, будет меньше, чем 0.0001. Используйте цикл while, но ограничьте предельное значение числа проходов значением 30. Если в 30-м проходе значение члена ряда, который добавляется, будет не меньше, чем 0.0001, то программа останавливается и выводится сообщение, что требуется более 30 членов ряда.

Первые несколько членов ряда Тейлора:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots$$

Используйте программу для вычисления значений e^2, e^{-4} и e^{21}.

Задание. Откройте редактор Scinotes и наберите текст скрипта:

```
x = input('Введите значение x = ');
n = 1; an = 1; S = an;
while (abs(an) >= 0.0001) & (n <= 30)
    an = x^n / factorial(n);
    S = S + an;
    n = n + 1;
end
if n >= 30 then
    disp('Программа прервана, так как требуется более 30 членов');
    else
    mprintf('exp(%f) = %f\n', x, S)
```

```
mprintf(‘Количество использованных членов ряда: %d\n', n)
end
```

Сохраните скрипт-файл в папке Lab6 под именем primer3.sce. Проверьте работу программы при заданных значениях степеней x.

3 Задания для самостоятельной работы

Задание 1. Разработайте скрипт для анализа температурных данных. Среднесуточная температура воздуха (в градусах по Цельсию) за первые 14 дней месяца составила:

15 17 20 22 19 16 12 14 16 17 18 21 22 20

Требуется определить:

1) количество дней, в которых температура больше 20^oC;

2) количество дней, в которых температура находится между 16^o и 18^oC;

3) дни (по порядку), в которых температура находится между 20^o и 25^oC.

Разработанный скрипт-файл сохраните в папке Lab6 под именем zadanie1.sce. Проверьте работу скрипта, запустив его из окна редактора.

Задание 2. Напишите программу в скрипт-файле, которая определяет действительные корни квадратного уравнения $ax^2 + bx + c = 0$. Имя файла zadanie2.sce. Когда скрипт запускается, он запрашивает пользователя ввести значения констант a, b, c. Чтобы вычислить корни уравнения, программа вначале определяет дискриминант D по формуле:

$$D = b^2 – 4ac$$

Если $D > 0$, то программа выдает сообщение: Уравнение имеет два корня. Затем два корня отображаются в следующих двух строках.

Если $D = 0$, то программа выдает сообщение: Уравнение имеет один корень. Затем значение корня отображается в следующей строке.

Если же $D < 0$, то программа выдает сообщение: Уравнение не имеет действительных корней. Затем работа программы прекращается.

Запустите разработанный скрипт-файл из командного окна три раза для получения решения трех следующих уравнений:

а) $3x^2 + 6x + 3 = 0$

б) $-3x^2 – 4x - 6 = 0$

в) $-3x^2 + 7x + 5 = 0$

Задание 3. Разработайте скрипт, в котором вычисляется сумма всех вводимых с клавиатуры положительных чисел. Условие окончания суммирования – ввод не положительного числа.

Указание. Для прекращения цикла ввода используйте оператор break. Для ввода числа используйте команду input с текстом. После окончания суммирования на дисплей должно выводиться с помощью команды disp сообщение о значении полученной суммы в виде одной строки.

Скрипт-файл сохраните в папке Lab6 под именем zadanie3.sce. Проверьте работу скрипта, запустив его из окна редактора.

Задание 4. Напишите программу в виде скрипта для вычисления значения y согласно выражению:

$$y = \sum_{k=-10}^{10} \frac{1}{k^2 + 2k},$$

где k – целое число, не равное 0 и не равное -2 (k 0, k -2).

Указание. Для пропуска итераций суммирования используйте оператор continue. На дисплей должно выводиться значение суммы y для каждого значения k. Для вывода используйте команду mprintf.

Скрипт-файл сохраните в папке Lab6 под именем zadanie4.sce. Проверьте работу скрипта, запустив его из окна редактора.

Задание 5. Значение числа π может быть определено выражением:

$$\sqrt{6(\sum_{n=1}^{\infty} \frac{1}{n^2})}$$

Напишите программу в виде скрипт-файла, которая вычисляет выражение. Программа вначале запрашивает пользователя ввести значение n. После выполнения вычислений программа выводит значение суммы членов ряда. Для сравнения выводится значение числа %pi (в формате с 20 символами). Вывод значения суммы и точного значения числа π произвести с помощью команды disp.

Сохраните скрипт в файле в папке Lab6 под именем zadanie5.sce. Проверьте работу программы для n = 100, n = 10000, n = 1000000.

4 Содержание отчета

Отчет по лабораторной работе № 6 выполняется в виде списка скрипт-файлов, разработанных в ходе выполнения заданий для самостоятельной работы.

Разработанные скрипты необходимо продемонстрировать преподавателю для оценки объема выполненной работы и анализа качества выполнения заданий.

5 Контрольные вопросы

1. Опишите действия операций отношения.
2. Опишите действия логических операций.
3. Какие операторы управления программой имеются в Scilab?
4. Какие операторы выбора имеются в Scilab?
5. Опишите действия оператора выбора if.
6. Опишите работу оператора выбора select.
7. Какие операторы цикла имеются в Scilab?
8. Какие операторы перехода имеются в Scilab?

Лабораторная работа № 7

ФУНКЦИИ В SCILAB

1 Цель работы

Изучить и практически исследовать использование функций в программах среды разработки Scilab.

2 Порядок выполнения работы

2.1 Создание папки для работы

При выполнении лабораторной работы № 7 Вы будете использовать папку с именем Lab7, которую необходимо предварительно создать.

С этой целью откройте Вашу рабочую папку (с Вашей фамилией) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab7.

В дальнейшем Вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 7. Полный путь к этой папке будет такой:

F:\OTC\Ivanov\Lab7

2.2 Запуск и настройка среды разработки Scilab

2.2.1. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа – рабочий стол среды разработки Scilab.

2.2.2. Нужно установить в качестве активной (текущей) папку Lab6. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем ОТС, затем папку с вашей фамилией (условно Ivanov) и затем папку Lab7. В строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab7. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab7

2.3 Создание и исполнение функций

Для определения новой (пользовательской) функции в Scilab используются ключевые слова function и endfunction в конструкции:

```
function [имя1, …, имяN] = имя_функции(перем1, …, перемM)
    тело_функции
endfunction
```

Здесь обозначено: имя1, …, имяN – список выходных параметров; имя_функции – имя, с которым эта функция будет вызываться; перем1, …, перемM – входные параметры (аргументы). Определение выходных параметров происходит в теле функции. Если функция имеет только один выходной параметр, то квадратные скобки не используются.

В качестве примера рассмотрим создание функции с именем myfunction, которая принимает единственный параметр x, умножает его на 2 и возвращает результат в качестве выходного параметра y.

```
function y = myfunction(x)
y = 2 * x;
endfunction
```

В Scilab можно использовать два способа определения этой функции:

- можно ввести тело функции непосредственно в командном окне Scilab, инструкция за инструкцией. Встретив запись наподобие function y = myfunction(x), интерпретатор переходит в режим ожидания тела функции. Завершается ввод командой endfunction, после чего Scilab возвращается в обычный режим.
- Более удобным вариантом является определение функции в отдельном файле. Именно этот способ применяется в большинстве случаев. При использовании редактора SciNotes имя файла-функции с

расширением .sci предлагается по имени, записанном в строке с ключевым словом function. Для рассматриваемого примера это будет myfunction.sci. Можно, при желании, изменить это имя.

Для того, чтобы использовать файл-функцию, созданную в редакторе SciNotes, ее нужно загрузить в рабочее пространство Scilab. Это можно сделать прямо из окна редактора, используя команду меню: **Выполнить => … выполнить без отображения команд**.

Также для загрузки функции в рабочее пространство Scilab можно использовать команду exec. Функция exec предназначена для исполнения инструкций, содержащихся в некотором файле, так, как если бы они вводились непосредственно в командное окно Scilab.

После того как функция создана и загружена в рабочее пространство Scilab, ее можно использовать подобно любой встроенной функции. Например для рассматриваемой функции myfunction(x):

```
--> y = myfunction(3)
y=
    6.
```

Задание 1. Наберите в командном окне Scilab определение следующей функции:

```
--> function y = myfunction(x)
--> y = 2 * x;
--> endfunction
```

Выполните проверку работы функции:

```
--> y = myfunction(3)

--> z = myfunction(5)

--> myfunction(10)
```

Задание 2. Нужно создать файл-функцию для вычисления математической функции:

$$f(x) = \frac{e^{x^2}}{\sqrt{x^2 + 5}}$$

Функцию надо написать таким образом, чтобы входной параметр x мог быть вектором.

В окне редактора SciNotes запишите следующее:

```
function y = func(x)
    y = exp(x.^2) ./ sqrt(x.^2 + 5);
endfunction
```

Сохраните файл в папке Lab7 под именем func.sci.

Загрузите файл-функцию func.sci в рабочее пространство Scilab из командного окна:

```
--> exec('func.sci', 2)
```

Выполнить проверку работы функции или, по-другому, тестирование функции можно следующими способами.

1) Вычисление функции в случае, когда входной параметр x является скаляром (числом) можно набором в командном окне следующих вариантов:

а)

```
--> x = 6;
--> y = func(x)
```

б)

```
-->z = func(6)
```

в)

```
--> func(6)
```

2) Чтобы вычислить функцию для нескольких значений x, нужно создать вектор с этими значениями, а затем использовать этот вектор в качестве аргумента функции:

```
--> x = 1 : 2 : 11
```

--> y = func(x)

3) Другой путь – это ввод вектора непосредственно в виде аргумента функции:

--> z = func([1 : 2 : 11])

2.4 Создание функций «на лету»

В случаях, когда значение относительно простого математического выражения должно быть использовано много раз в программе, Scilab предоставляет возможность создания функций «на лету». Этим термином называется пользовательская функция, которая определена и записана внутри компьютерного кода (не в отдельном файле-функции), и затем используется в программе. Подобные функции могут быть созданы в любой части среды Scilab: в командном окне, в скрипте, внутри обычной пользовательской функции.

Для создания функции «на лету» используется оператор deff, имеющий в общем случае вид:

deff (‘[имя1, … , имяN] = имя_функции(перем1, … , перемM)’, ‘имя1 = выражение1; … ; имяN = выражениеN’)

Здесь имя1, … , имяN – список выходных параметров, т.е. переменных, которым будет присвоен конечный результат вычислений; имя_функции – это имя, с которым функция будет вызываться; перем1, … , перемM – входные параметры; имя1 = выражение1; ….; имяN = выражениеN – определение (вычисление) выходных переменных. Если выходной параметр один, то квадратные скобки не нужны.

Пример 1. Определить «на лету» функцию f(x) = sin(cos(x)) и вычислить ее значение в точке 1.3.

Задание. Наберите в командном окне следующую программу:

--> deff(‘y = fun(x)’,’y = sin(cos(x))’)
--> y = fun(1.3)

--> fun(1.3)

Пример 2. Определить «на лету» функцию, в которой одновременно задаются f(x, y) = cos(x + y) и g(x, y) = sin(x – y), и вычислить ее значение в точке (-2.5; 1.7).

Задание. Наберите в командном окне следующую программу:

--> deff('[f, g] = fun2(x, y)','f = cos(x + y); g = sin(x – y)')

--> [f, g] = fun2(-2.5, 1.7)
--> [f1, g1] = fun2(-2.5, 1.7)
--> fun2(-2.5, 1.7)

3 Задания для самостоятельной работы

Задание 1. Напишите файл-функцию для вычисления математической функции

$y(x) = (-0.2\,x^3 + 7\,x^2)\,e^{-0.3x}$

Дайте имя функции, например, math. Входной параметр функции – это x, а выходной – y. Напишите функцию таким образом, чтобы входной параметр x мог быть вектором. Разработанную файл-функцию сохраните в папке Lab7 под именем math.sci.

Далее напишите скрипт-файл, который использует разработанную функцию math( ). Вначале скрипт запрашивает ввод двух элементов вектора x. Нужно ввести значения -1.5 и 5. Затем вычисляется y(x) и выводятся значения y для этих элементов. Потом скрипт запрашивает ввод вектора x с начальным значением -2, шагом 0.1 и конечным значением 6. Программа строит график функции y(x). График должен иметь заголовок, подписи осей.

Скрипт сохраните в папке Lab7 под именем zadanie1.sce.

Выполните тестирование разработанного скрипта.

Задание 2. Поверхность тела человека BSA (Body Surface Area) в кв. метрах (используется в медицине для дозирования медпрепаратов) может быть вычислена по эмпирической формуле:

$$BSA = 0.007184 \cdot w^{0.425} \cdot h^{0.75},$$

где w – масса тела в кг, h – рост в м.

Требуется написать в Scilab функцию пользователя, которая вычисляет поверхность тела человека. Для имени функции и ее параметров используйте следующую формулу:

BSA = BodySurA(w, h)

Входные параметры функции w и h – это масса и рост человека. Выходной параметр BSA – это поверхность тела. Сохраните файл-функцию в папке Lab7 под именем BodySurA.sci.

Затем надо создать в SciNotes скрипт, в котором вначале запрашиваются входные параметры: масса тела в кг и рост в м. Затем при помощи функции BodySurA( ) вычисляется значение BSA. После этого выводится с помощью команды mprintf величина BSA в кв. метрах при заданных значениях веса и роста. Скрипт-файл сохраните в папке Lab7 под именем zadanie2.sce.

Выполните тестирование программы при следующих входных параметрах:

1) вес – 95 кг, рост – 187 см (мужчина);
2) вес – 61 кг, рост – 158 см (женщина).

Задание 3. RL-фильтр нижних частот (фильтр, который пропускает сигналы низких частот), имеет коэффициент передачи К – отношение амплитуд выходного и входного напряжений в виде:

$$K = \left| \frac{U_{вых}}{U_{вх}} \right| = \frac{1}{\sqrt{1 + \left(\frac{\omega L}{R}\right)^2}},$$

где w – это угловая частота входного сигнала.

Требуется написать функцию пользователя, которая вычисляет значение коэффициента К. Для записи функции в Scilab используйте следующую формулу:

K = filtr(R, L, w)

Входные параметры функции: R – сопротивление резистора в омах; L – индуктивность катушки в генри; w – угловая частота входного сигнала в 1/с.

Напишите функцию таким образом, чтобы w могла быть вектором. Файл-функцию сохраните в папке Lab7 под именем filtr.sci.

Затем напишите программу в виде скрипт-файла, которая использует функцию filtr( ) для построения графика зависимости коэффициента К от частоты w для диапазона частот от 10 до 1000000 с шагом 10. Для построения функции используйте команду plot2d с заданием логарифмической горизонтальной оси. График доложен иметь заголовок, подписи осей, сетку. Когда скрипт начинает исполняться, он запрашивает пользователя ввести значения параметров R и L.

Скрипт-файл сохраните в папке Lab7 под именем zadanie3.sce.

Выполните тестирование скрипта при R = 600 Ом, L = 0.14e-6 Гн.

4 Содержание отчета

Отчет по лабораторной работе № 7 выполняется в виде списка файлов, разработанных в ходе выполнения заданий для самостоятельной работы.

Разработанные файлы необходимо продемонстрировать преподавателю для оценки объема выполненной работы и анализа качества выполнения заданий.

5 Контрольные вопросы

1. Какие виды функций используются в среде разработки Scilab?
2. Какие в Scilab имеются методы создания функций пользователя?
3. Как определяется функция с помощью оператора functioin?
4. Как определяется функция с помощью оператора deff?
5. Что такое файл-функция в Scilab?
6. Как выполняется загрузка функций в рабочее пространство Scilab?

Лабораторная работа № 8

ПОЛИНОМЫ В SCILAB

1 Цель работы

Изучить и исследовать методы создания полиномов и математические операции с ними. Исследовать встроенные функции для операций с полиномами.

2 Порядок выполнения работы

2.1 Создание папки для работы

При выполнении лабораторной работы № 8 Вы будете использовать папку с именем Lab8, которую необходимо предварительно создать.

С этой целью откройте Вашу рабочую папку (с Вашей фамилией) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab8.

В дальнейшем Вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 8. Полный путь к этой папке будет такой:

F:\OTC\Ivanov\Lab8

2.2 Запуск и настройка среды разработки Scilab

2.2.1. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа – рабочий стол среды разработки Scilab.

2.2.2. Нужно установить в качестве активной (текущей) папку Lab8. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем OTC, затем папку с вашей фамилией (условно Ivanov) и затем папку Lab8. В строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab8. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab8

2.3 Полиномы в Scilab

Полином – это функция в форме алгебраической суммы многочленов вида:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$$

Здесь a_n, a_{n-1}, …, a_0 – коэффициенты (только числа); x – символьная переменная (аргумент) функции. Число n, которое является неотрицательным целым, называется степенью полинома.

Примеры полиномов:

$f(x) = 2x^6 + 3x^3 + x + 1$ полином степени 6

$f(x) = 4x - 1$ полином степени 1

$f(x) = 7$ константа, как полином степени 0

В Scilab принят обратный порядок следования коэффициентов полиномов, а именно:

$$a_0, a_1, \ldots, a_{n-1}, a_n$$

Первый элемент полинома – это коэффициент при аргументе, например, x со степенью 0 ($x^0 = 1$). Таким образом, в Scilab полином записывается в виде:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} + a_n x^n$$

Примеры полиномов:

в обычной математике, в Matlab	в Scilab
$2x^6 + 3x^3 + x + 1$	$1 + x + 3x^3 + 2x^6$
$x^2 + 4x + 6$	$6 + 4x + x^2$

В Scilab очень часто используется векторное представление полиномов. Вектор должен включать все коэффициенты полинома, даже те, которые равны нулю:

полином в Scilab	вектор в Scilab
$5 + 3x$	[5 3]
$3x + x^3 = 0 + 3x + 0x^2 + x^3$	[0 3 0 1]
$1 + x^5 + 2x^6 = 1 + 0x + 0x^2 + 0x^3 + 0x^4 + x^5 + 2x^6$	[1 0 0 0 0 1 2]

Корни полинома – это значения аргумента, при которых полином равен нулю. Например, корни полинома $f(x) = -3 – 2x + x^2$ – это значения x, для которых $-3 – 2x + x^2 = 0$. Это будут x = 3 и x = -1.

2.4 Методы создания полиномов в Scilab

Имеется несколько методов создания полиномов.

2.4.1. Сначала надо определить символьную переменную – аргумент полинома. Это делается с помощью команды poly( ) в следующей форме:

--> x = poly(0, ‘x’)

Затем можно задать полином с использованием «естественной» математической записи (порядок следования коэффициентов произвольный):

--> p1 = x^3 – 6*x^2 + 11*x – 6

Можно даже задать полином в виде:

--> p2 = (x^2 – 1) * (x + 3)

Задание. Наберите в командном окне следующую программу для создания полиномов:

```
--> poly(0, ‘x’)
--> p1 = x^3 – 6*x^2 + 11*x -6
--> p2 = (x^2 – 1)*(x + 3)
```

В Scilab имеется предопределенная (системная) переменная %s, которая специально предусмотрена для использования в качестве символьной переменной. При ее применении задать полином можно так:

```
--> p3 = %s^3 – 6*%s^2 + 11*%s – 6
p3 =
   – 6 + 11s – 6s^2 + s^3
```

Использовать символьную переменную в виде %s не очень удобно, поэтому обычно ее предварительно переопределяют:

```
--> s = %s
--> p4 = s^3 – 6*s^2 + 11*s – 6
p4 =
```

$-6 + 11s - 6s^2 + s^3$

Задание. Выполните следующую программу для определения полиномов с использованием системной переменной %s:

```
--> p3 = %s^3 – 6*%s^2 + 11*%s – 6
--> clear
--> s = %s
--> p4 = s^3 – 6*s^2 + 11*s – 6
```

2.4.2. Другой метод создания полиномов – это использование команды poly( ) в формате:

poly(A, ‘name’, ‘flag’)

Здесь

A – вектор или матрица;

name – строка, содержащая имя символьной переменной;

flag – строка, которая может иметь вид:

‘coeff’, обычно сокращенно ‘c’ или

‘roots’, обычно сокращенно ‘r’;

Флаг ‘c’ означает, что параметр A состоит из коэффициентов поли- нома, флаг ‘r’ означает, что параметр A состоит из корней полинома.

Задание. Выполните программу, в которой задается полином с использованием его коэффициентов в виде вектора-строки:

```
--> mycoef = [-6 11 -6 1]
--> p5 = poly(mycoef, ‘x’, ‘c’)
--> p6 = poly([-6 11 -6 1], ‘z’, ‘c’)
```

Задание. Выполните программу, в которой задается полином с использованием его корней в виде вектора-строки:

```
--> clear
--> myroots = [1 2 3]
--> p7 = poly(myroots, ‘x’, ‘r’)
--> p8 = poly([1 2 3], ‘z’, ‘r’)
```

2.5 Алгебраические операции с полиномами

В Scilab возможны алгебраические, то есть символьные операции с полиномами. Могут использоваться арифметические операции: сложение, вычитание, умножение, деление и их комбинации. Важно соблюдать требование – все полиномы в этих операциях должны иметь одни и те же символьные переменные.

Задание. Выполните арифметические символьные операции с полиномами:

```
--> clear
--> p1 = poly([-6 11 -6 1], 'x', 'c')
--> p2 = poly([1 2 3], 'x', 'r')
```

1) сложение полиномов:

```
--> p3 = p1 + p2
```

2) вычитание полиномов:

```
--> p4 = p1 – p2
```

3) умножение полиномов:

```
--> p5 = p1 * p2
```

4) деление полиномов:

```
--> p6 = p5 / p1
```

Scilab имеет две встроенные функции для выполнения деления полиномов – это pdiv( ) и ldiv( ). Ознакомиться с ними можно из меню help.

2.6 Анализ полиномов

Scilab имеет несколько встроенных функций для выполнения специальных действий с полиномами.

Функция coeff( ) возвращает вектор, содержащий коэффициенты заданного полинома p1:

```
coeff(p1)
```

Можно продифференцировать полином p1 с помощью функции derivat( ):

```
derivat(p1)
```

Можно вычислить корни заданного полинома p1 с помощью функции roots( ), которая возвращает вектор-столбец из значений корней:

roots(p1)

Задание. Выполните следующие функции для заданного полинома p1:

```
--> clear
--> p1 = poly([-6 11 -6 1], ‘x’, ‘c’)
--> coef = coeff(p1)
--> p2 = derivat(p1)
--> rts = roots(p1)
```

Объясните полученные результаты.

2.7 Полиномиальные дроби

Полиномиальную дробь, или, по-другому, дробно-рациональный полином можно задать путем записи дроби, в которой числитель и знаменатель являются полиномами.

Задание. Выполните программу для задания полиномиальной дроби:

```
--> clear
--> s = %s
--> p1 = (3*s^2 + 2*s + 1) / (6*s^2 + 5*s + 4)
--> p2 = poly([1 2 3], ‘x’, ‘c’) / poly([4 5 6], ‘x’, ‘c’)
```

Можно вычислить (в символьном виде) числитель и знаменатель дробно-рационального полинома с помощью функций numer( ) и denom( ).

Задание. Определите числитель и знаменатель дроби p1:

```
--> num = numer(p1)
--> den = denom(p1)
```

2.8 Вычисление значения полинома

Заданный полином или полиномиальная дробь могут быть вычислены для определенного значения аргумента с помощью функции horner( ). Ее синтаксис:

horner(p, x),

где p – полином или дробь; x – массив чисел.

Задание. Выполните вычисление значений заданного полинома:

```
--> clear
--> s = %s
--> p = s^2 – 2*s + 15
--> horner(p, 2)
--> horner(p, [2 4 6 8 10])
```

Объясните полученные результаты.

Удобно использовать функцию horner( ) для построения графика значений полинома в определенной области изменения аргумента. Для этого сначала создается вектор для аргумента, затем используется функция horner( ) для вычисления значений полинома, а затем строится график с помощью функции plot( ).

Пример. Требуется построить график значений полинома

$y = 0.1x^5 – 0.2x^4 – x^3 + 5x^2 – 41.5x + 235$

для области $– 6 \le x \le 6$. Для решения задачи удобно создать скрипт.

Задание. В окне редактора SciNotes наберите текст скрипта:

```
clear, clc
p = poly([235 -41.5 5 -1 -0.2 0.1], ‘x’, ‘c’);
x = linspace(-6, 6, 200);                  // 200 значений аргумента x
y = horner(p, x);
plot(x, y);
```

xlabel(‘x -->’);
ylabel(‘y -->’);

Сохраните набранный скрипт в папке Lab8 под именем, например, grafic.sce. Запустите скрипт на выполнение из окна редактора. Объясните полученный график.

3 Задания для самостоятельной работы

Задание 1. Создайте скрипт, который использует Scilab для умножения двух полиномов:

$$(-x^3 + 5x - 1)\,(x^4 + 2x^3 - 16x + 5)$$

Для выполнения задания 1 нужно создать в текстовом редакторе SciNotes скрипт-файл и сохранить его в папке Lab8 под именем zadanie1.sce. Проверку работы скрипта произведите из окна редактора SciNotes.

Задание 2. Создайте скрипт, в котором выполняется умножение следующих полиномов:

$$x\,(x - 1.7)\,(x + 0.5)\,(x - 0.7)\,(x + 1.5)$$

Затем необходимо построить график результата умножения полиномов для области $-1.6 \le x \le 1.8$ (всего 200 точек).

Для выполнения задания 2 нужно создать в текстовом редакторе SciNotes скрипт-файл и сохранить его в папке Lab8 под именем zadanie2.sce. Проверку работы скрипта произведите из окна редактора SciNotes.

Задание 3. Создайте скрипт, в котором выполняется деление полинома $-10x^6 - 20x^5 + 9x^4 + 10x^3 + 8x^2 + 11x - 3$ на полином $2x^2 + 4x - 1$.

Для выполнения задания 3 нужно создать в текстовом редакторе SciNotes скрипт-файл и сохранить его в папке Lab8 под именем zadanie3.sce. Проверку работы скрипта произведите из окна редактора SciNotes.

Задание 4. Создайте скрипт для построения графика полинома

$$y = 0.008\ x^4 - 1.8\ x^2 - 5.4\ x + 54$$

в области $-14 \le x \le 16$. Сделайте подписи осей.

Для выполнения задания 4 нужно создать в текстовом редакторе SciNotes скрипт-файл и сохранить его в папке Lab8 под именем zadanie4.sce. Проверку работы скрипта произведите из окна редактора SciNotes.

4 Содержание отчета

Отчет по лабораторной работе № 8 выполняется в виде списка скрипт-файлов, разработанных в ходе выполнения заданий для самостоятельной работы.

Разработанные скрипты необходимо продемонстрировать преподавателю для оценки объема выполненной работы и анализа качества выполнения заданий.

5 Контрольные вопросы

1. Как в Scilab можно определить алгебраический полином?
2. Как получить полиномиальную дробь?
3. Объясните функцию poly(V, ‘x’, ‘c’).
4. Объясните функцию poly(V, ‘x’, ‘r’).
5. Объясните функцию roots(p).
6. Как производить операции в символьном виде?
7. Что такое корни полинома?
8. Как можно вычислить корни полинома?

Лабораторная работа № 9

ИССЛЕДОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ В SCILAB

1 Цель работы

Изучить и практически исследовать методы расчета и построения временных и частотных характеристик, способы преобразования структурных схем, определения устойчивости систем автоматического управления.

2 Порядок выполнения работы

2.1 Создание папки для работы

При выполнении лабораторной работы № 9 вы будете использовать папку с именем Lab9, которую необходимо предварительно создать.

С этой целью откройте вашу рабочую папку (с вашей фамилией) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab9.

В дальнейшем вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 9. Полный путь к этой папке будет такой:

F:\OTC\Ivanov\Lab9

2.2 Запуск и настройка среды разработки Scilab

2.2.1. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа – рабочий стол среды разработки Scilab.

2.2.2. Нужно установить в качестве активной (текущей) папку Lab6. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем ОТС, затем папку с вашей фамилией (условно Ivanov) и затем папку Lab9. В

строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab9. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab9

2.3 Исследование функции определения системы syslin( )

В пакете CACSD интегрированной среды Scilab, предназначенном для исследования систем управления, имеется функция syslin( ), которая определяет передаточную функцию линейной системы в Scilab по ее передаточной функции W(s), эаписанной с помощью преобразования Лапласа. Если задана передаточная функция звена или системы управления в виде отношения полиномов:

$$W(s) = \frac{N(s)}{D(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + ... + b_1 s + b_0}{a_m s^m + a_{m-1} s^{m-1} + ... + a_1 s + a_0}$$

то Scilab определяет передаточную функцию следующим образом:

W = syslin(‘c’, W(s))

или

W = syslin(‘c’, N(s), D(s))

Здесь параметр ‘c’ указывает, что система с непрерывным временем, а s – символьная переменная.

Задание. Требуется исследовать основные варианты определения передаточной функции линейной системы в Scilab с помощью функции syslin( ). Примеры выполняются в командном окне.

1. Допустим, что передаточная функция звена или системы имеет вид:

$$W(s) = \frac{s+2}{s^2 + 2s + 1}$$

Сначала определим переменную с именем s как символьную %s (в Scilab переменная с именем %s является системной предопределенной символьной переменной):

--> s = %s;

Затем надо набрать:

--> W = syslin('c', (2 + s) / (s^2 + 2*s + 1))

Обратите внимание, что в полученной с помощью syslin( ) передаточной функции системы W коэффициенты полиномов числителя и знаменателя располагаются в порядке возрастания степени символьной переменной s (как это принято в Scilab).

2. Пусть передаточная функция звена или системы имеет вид:

$$W(s) = \frac{1 + 0.1s}{0.2s^2 + 0.4s + 1}$$

Сначала очистим рабочее пространство (область переменных) командой:

--> clear

Затем определим символьную переменную:

--> s = %s;

Определим передаточную функцию следующим образом:

--> W = syslin('c', 1 + 0.1*s, 0.2*s^2 + 0.4*s + 1)

3. Передаточная функция звена или системы имеет вид:

$$W(s) = \frac{s^2 + 3s + 1}{2s^3 + 4s^2 + 5}$$

Очистим рабочее пространство командой:

--> clear

Определим символьную переменную:

--> s = %s;

Определим числитель передаточной функции:

--> num = s^2 + 3*s +1;

Определим знаменатель передаточной функции:

--> den = 2*s^3 + 4*s^2 + 5;

Определим передаточную функцию системы в виде:
--> W = syslin(‘c’, num, den)

2.4 Построение временных характеристик линейных систем

В пакете CACSD среды Scilab имеется функция csim( ), с помощью которой можно рассчитать временные характеристики системы по ее передаточной функции. Функция csim( ) вычисляет реакцию системы на определенные входные сигналы. Имеется два вида стандартных входных сигналов и соответственно два варианта функции csim( ):

1)

$$h = csim(‘step’, t, W),$$

где ‘step’ – это входной единичный ступенчатый сигнал 1(t), а h – это переходная функция h(t);

2)

$$g = csim(‘impulse’, t, W),$$

где ‘impulse’ – это входной бесконечный импульс δ(t) (единичная функция Дирака), а g – это импульсно-переходная функция g(t).

В обоих случаях переменная t – это вектор, определяющий время вычисления функции, а W – это передаточная функция системы, определяемая с помощью функции syslin( ).

Задание. Требуется вычислить и построить графики переходной функции h(t) и импульсно-переходной функции g(t) для звена САУ, заданного передаточной функцией:

$$W(s) = \frac{0.1s + 1}{0.1s^2 + 0.2s + 1}$$

Набираем в командном окне следующую программу.
Очистим рабочее пространство Scilab:
--> clear
Определим символьную переменную:
--> s = %s;
Определим передаточную функцию линейной системы:

--> W = syslin('c', 0.1*s + 1, 0.1*s^2 + 0.2*s + 1)

Задаем временной интервал от 0 до 10 секунд в виде вектора с шагом 0.1 с:

--> t = 0 : 0.1 : 10;

Вычисляем переходную функцию h(t):

--> h= csim('step', t, W);

Строим график переходной функции в окне с номером 0:

--> plot(t, h)

Вычисляем импульсно-переходную функцию:

-->g = csim('impulse', t, W);

Открываем новое графическое окно с номером 1 командой:

--> scf(1);

Строим график импульсно-переходной функции в этом окне:

--> plot(t, g)

2.5 Построение частотных характеристик линейных систем

В пакете CACSD среды Scilab имеется несколько функций, с помощью которых можно вычислить и построить графики различных частотных характеристик линейных систем, заданных передаточными функциями. Наиболее часто используются следующие функции:

1) bode( ) – диаграмма Боде. Эта функция вычисляет и строит в одном графическом окне две функции: логарифмическую амплитудную частотную характеристику (ЛАЧХ) и фазовую частотную характеристику (ФЧХ). Функция наиболее часто применяется в формате

bode(W, fmin, fmax),

где W – передаточная функция линейной системы, определенной посредством функции syslin( ); fmin, fmax – диапазон частот вычисления функции в Гц.

Так как диапазон частот обычно очень большой, то по оси абсцисс откладывается частота в логарифмическом масштабе (lg f), чтобы получить линейную шкалу.

2) nyquist( ) – диаграмма (годограф) Найквиста. Эта функция вычисляет и строит график амплитудно-фазовой частотной

характеристики системы (АФЧХ). Функция наиболее часто применяется в формате

nyquist(W, fmin, fmax),

где W – передаточная функция линейной системы, определенной посредством функции syslin( ); fmin, fmax – диапазон частот вычисления функции в Гц.

3) gainplot( ) – функция вычисляет и строит логарифмическую амплитудную частотную характеристику (ЛАЧХ).). Функция наиболее часто применяется в формате

gainplot(W, fmin, fmax),

где W – передаточная функция линейной системы, определенной посредством функции syslin( ); fmin, fmax – диапазон частот вычисления функции в Гц.

Так как диапазон частот обычно очень большой, то по оси абсцисс откладывается частота в логарифмическом масштабе (lg f), чтобы получить линейную шкалу.

4) phaseplot( ) - функция вычисляет и строит фазовую частотную характеристику (ФЧХ).). Функция наиболее часто применяется в формате

phaseplot(W, fmin, fmax),

где W – передаточная функция линейной системы, определенной посредством функции syslin( ); fmin, fmax – диапазон частот вычисления функции в Гц.

Так как диапазон частот обычно очень большой, то по оси абсцисс откладывается частота в логарифмическом масштабе (lg f), чтобы получить линейную шкалу.

Возможно применение рассмотренных функций в сокращенном формате:

bode(W), nyquist(W), gainplot(W), phaseplot(W).

В этом случае, по умолчанию, минимальная частота fmin = 10^{-3} Гц, а fmax = 10^{3} Гц. Этого диапазона частот обычно вполне достаточно для большинства реальных систем.

Задание. Требуется вычислить и построить графики частотных функций для звена САУ, заданного передаточной функцией:

$$W(s) = \frac{0.1s + 1}{0.1s^2 + 0.2s + 1}$$

Наберите в командном окне следующую программу.
Очистим рабочее пространство Scilab:
--> clear
Определим символьную переменную:
--> s = %s;
Определим передаточную функцию линейной системы:
--> W = syslin('c', 0.1*s + 1, 0.1*s^2 + 0.2*s + 1)
--> scf(2); // открываем новое графическое окно с номером 2
Строим диаграмму Боде в окне 2:
--> bode(W);
--> scf(3); // открываем новое графическое окно с номером 3
Строим диаграмму Найквиста в окне 3:
--> nyquist(W);
--> scf(4); // отрываем новое графическое окно с номером 4
Строим ЛАЧХ в окне 4:
--> gainplot(W);
--> scf(5); // открываем новое графическое окно с номером 5
Строим ФЧХ в окне 5:
--> phaseplot;

Обратите внимание, что некоторые графики имеют заголовки и названия осей. В заключение закройте все графические окна.

2.6 Определение устойчивости САУ

Пример. Требуется определить устойчивость САУ, имеющей передаточную функцию вида:

$$W(s) = \frac{0.1s + 1}{0.1s^2 + 0.2s + 1}$$

Характеристическое уравнение САУ будет:

$$D(s) = 0.1s^2 + 0.2s + 1$$

Необходимо найти корни этого уравнения.

Задание. Наберите в командном окне следующую программу.

Очистим рабочее пространство Scilab:

```
--> clear
```

Определим символьную переменную:

```
--> s = %s;
```

Определим передаточную функцию линейной системы:

```
--> W = syslin('c', 0.1*s + 1, 0.1*s^2 + 0.2*s + 1)
```

Вычислим знаменатель передаточной функции:

```
--> den = denom(W);
```

Вычислим корни характеристического уравнения:

```
--> r = roots(den)
```

Должен получиться результат:

```
r =
    -1. + 3. i
    -1. - 3. i
```

Корни получились комплексно-сопряженные с отрицательной вещественной частью. Поэтому можно сделать вывод, что САУ с заданной передаточной функцией будет устойчивой.

2.7 Преобразование структурных схем САУ

Структурная схема САУ показывает, из каких из каких элементарных динамических звеньев она состоит и как эти звенья соединены между собой. В результате на основании передаточных функций отдельных звеньев становится возможным определить передаточную функцию всей системы в целом. На структурной схеме каждое звено изображается в виде прямоугольника, внутри которого указывается его передаточная функция. Направление передачи сигналов обозначается стрелками. Элементарные динамические звенья в структурных схемах САУ могут соединяться лишь тремя различными способами: последовательным, параллельным и встречно-параллельным (с обратной связью).

В пакете CACSD среды Scilab для построения моделей соединения динамических звеньев используются знаки арифметических операций. При этом передаточные функции

отдельных звеньев должны быть предварительно определены посредством функции syslin( ).

1. Последовательное соединение звеньев

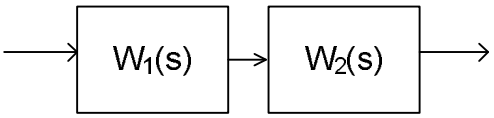


Рисунок 1 – Последовательное соединение звеньев

Если обозначить

W1 = syslin(‘c’, W1(s)), W2 = syslin(‘c’, W2(s)),

то эквивалентная передаточная функция W12 при таком соединении будет:

$$W12 = W1 * W2$$

2. Параллельное соединение звеньев

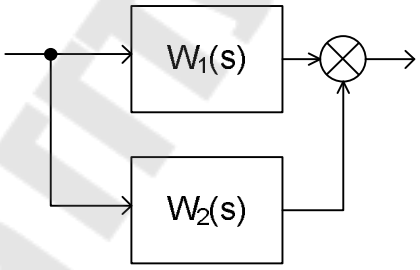


Рисунок 2 – Параллельное соединение звеньев

Если обозначить

W1 = syslin(‘c’, W1(s)), W2 = syslin(‘c’, W2(s)),

то эквивалентная передаточная функция W12 при таком соединении будет:

$$W12 = W1 + W2$$

3. Соединение звеньев с обратной связью (отрицательной)

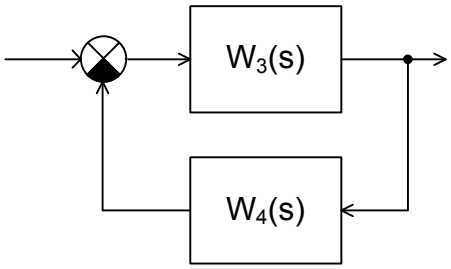


Рисунок 3 – Соединение звеньев с обратной связью (отрицательной)

Если обозначить

W3 = syslin(‘c’, W3(s)), W4 = syslin(‘c’, W4(s)),

то эквивалентная передаточная функция W34 при таком соединении будет:

W34 = W3 /. W4

Примечание. Соединение звеньев с положительной обратной связью практически не используется из-за возможности появления автоколебаний (самовозбуждения) в САУ.

4. Соединение звеньев с единичной отрицательной обратной связью

В системах автоматического управления возможен вариант встречно-параллельного соединения, когда отсутствует звено в цепи отрицательной обратной связи. Этот вариант называется соединением с единичной отрицательной обратной связью.

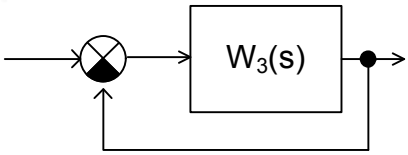


Рисунок 4 – Соединение с единичной отрицательной обратной связью

Если обозначить

W3 = syslin(‘c’, W3(s)),

то эквивалентная передаточная функция W при таком соединении будет:

$$W = W3 / (1 + W3)$$

2.8 Пример анализа САУ

Структурная схема САУ имеет следующий вид:

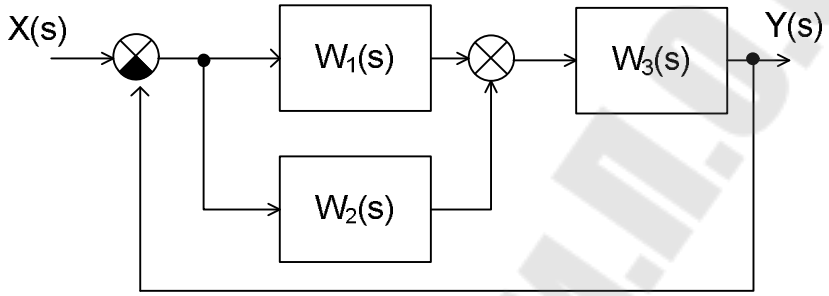


Рисунок 5 – Структурная схема САУ

Передаточные функции звеньев:

$$W_1(s) = 4\,; W_2(s) = \frac{0.05s + 1}{0.1s + 1}\,; W_3(s) = \frac{1}{s(0.5s + 1)}\,.$$

Требуется произвести исследование САУ и выполнить следующее:

1) преобразовать структурную схему и найти эквивалентную передаточную функцию САУ;
2) вычислить и построить графики переходной и импульсно-переходной характеристик САУ;
3) вычислить и построить диаграмму Боде, диаграмму (годограф) Найквиста, логарифмическую амплитудную частотную характеристику (ЛАЧХ), фазовую частотную характеристику (ФЧХ);
4) определить устойчивость САУ.

Задание. Для решения поставленной задачи создадим несколько скриптов. С этой целью откройте редактор SciNotes.

1. Запишите в окне редактора текст скрипта для преобразования структурной схемы САУ и нахождения эквивалентной передаточной функции системы.

```
clear, clc;
s = %s;                    // объявление символьной переменной
      // определение передаточных функций отдельных звеньев
W1 = syslin('c', 4, 1);
W2 = syslin('c', 0.05*s + 1, 0.1*s + 1);
W3 = syslin('c', 1, s * (0.5*s + 1);
                    // преобразование структурной схемы
W12 = W1 + W2;
W123 = W12 * W3;
W = W123 / (1 + W123);
                    // эквивалентная передаточная функция САУ
disp('Передаточная функция САУ: W = ');
disp(W);
```

Созданный скрипт-файл сохраните в папке Lab9 под именем, например, w_sau.sce. Запустите скрипт, используя команду меню: Выполнить => … без отображения команд. При отсутствии ошибок закройте файл, выполнив команду: Файл => Закрыть.

2. Запишите в окне редактора SciNotes текст скрипта для вычисления и построения графиков временных функций САУ.

```
t = 0 : 0.01 : 10;          // диапазон времени и шаг
h = csim('step', t, W);
plot(t, h);          // график h(t) в окне с номером 0
xtitle('Переходная характеристика h(t)', 'Время, c',
'Амплитуда');
xgrid( );            // отобразить сетку
scf(1);              // открыть новое графическое окно с номером 1
g = csim('impulse', t, W);
plot(t, g);          // график g(t) в окне 1
xtitle('Импульсно-переходная характеристика g(t)', 'Время, c', ..
    'Амплитуда');
xgrid( );            // отобразить сетку
```

Созданный скрипт-файл сохраните в папке Lab9 под именем, например, t_sau.sce. Запустите скрипт, используя команду меню: Выполнить => … без отображения команд. При отсутствии ошибок закройте файл, выполнив команду: Файл => Закрыть. Закройте также все графические окна.

3. Запишите в окне редактора SciNotes текст скрипта для вычисления и построения графиков частотных функций САУ.

```
bode(W);          // построение графика в окне с номером 0
title('Диаграмма Боде');
scf(1);           // открыть новое графическое окно с номером 1
nyquist(W);       // построение графика в окне 1
scf(2);           // открыть новое графическое окно с номером 2
gainplot(W);            // построение графика в окне 2
title('Логарифмическая амплитудная частотная характеристика (ЛАЧХ)');
scf(3);           // открыть новое графическое окно с номером 3
phaseplot(W);     // построение графика в окне 3
title('Фазовая частотная характеристика (ФЧХ)');
```

Созданный скрипт-файл сохраните в папке Lab9 под именем, например, f_sau.sce. Запустите скрипт, используя команду меню: Выполнить => … без отображения команд. При отсутствии ошибок закройте файл, выполнив команду: Файл => Закрыть. Закройте также все графические окна.

4. Запишите в окне редактора SciNotes текст скрипта для вычисления корней характеристического уравнения САУ с целью определения устойчивости САУ.

```
den = denom(W);       // определение знаменателя передаточной
                      // функции W
r = roots(den);       // вычисление корней
disp('Корни характеристического уравнения САУ равны: ');
disp(r);
```

Созданный скрипт-файл сохраните в папке Lab9 под именем, например, r_sau.sce. Запустите скрипт, используя команду меню: Выполнить => … без отображения команд. При отсутствии ошибок закройте файл, выполнив команду: Файл => Закрыть. По значениям корней характеристического уравнения сделайте вывод об устойчивости САУ.

3 Задания для самостоятельной работы

Перед выполнением заданий для самостоятельной работы вам необходимо создать в редакторе WORD файл, например, с именем sau.doc и сохранить его в папке Lab9. В этом файле вы будете сохранять скрипты и графики функций, которые необходимы для оформления пояснительной записки к курсовому проекту по ОТС.

Задание 1. Разработайте скрипт для преобразования структурной схемы и нахождения эквивалентной передаточной функции САУ для вашего индивидуального задания к курсовому проекту по ОТС.

Созданный скрипт-файл сохраните в папке Lab9 под именем, например, w_proj.sce. Запустите скрипт, используя команду меню: Выполнить => … без отображения команд.

При отсутствии ошибок скопируйте скрипт-файл в буфер обмена, выполнив последовательно команды в редакторе SciNotes: Правка => Выделить все, затем Правка => Копировать. После этого раскройте файл sau.doc и вставьте в него скопированный скрипт из буфера обмена. Затем сверните файл sau.doc. В редакторе SciNotes выполните команду: Файл => Закрыть.

Задание 2. Запишите в окне редактора SciNotes текст скрипта для вычисления и построения графиков временных функций для САУ по заданию курсового проекта. Выполните оформление графиков функций: название, подписи осей, сетка.

Скрипт-файл сохраните в папке Lab9 под именем, например, t_proj.sce. Запустите скрипт на выполнение. При необходимости сделайте коррекцию временного диапазона. Затем скопируйте скрипт-файл в буфер обмена, выполнив последовательно команды в

редакторе SciNotes: Правка => Выделить все, затем Правка => Копировать. После этого раскройте файл sau.doc и вставьте в него скопированный скрипт из буфера обмена.

Далее раскройте графическое окно 0 и с помощью команды: Файл => Копировать в буфер обмена выполните копирование графика переходной функции. Затем раскройте файл sau.doc и вставьте в него график переходной функции из буфера обмена.

Далее раскройте графическое окно 1 и с помощью команды: Файл => Копировать в буфер обмена выполните копирование графика импульсно-переходной функции. Затем раскройте файл sau.doc и вставьте в него график импульсно-переходной функции из буфера обмена.

Затем сверните файл sau.doc. В редакторе SciNotes выполните команду: Файл => Закрыть.

Задание 3. Запишите в окне редактора SciNotes текст скрипта для вычисления и построения графиков частотных функций для САУ по заданию курсового проекта. Выполните оформление графиков функций: название, подписи осей, сетка.

Скрипт-файл сохраните в папке Lab9 под именем, например, f_proj.sce. Запустите скрипт на выполнение. Затем скопируйте скрипт-файл в буфер обмена, выполнив последовательно команды в редакторе SciNotes: Правка => Выделить все, затем Правка => Копировать. После этого раскройте файл sau.doc и вставьте в него скопированный скрипт из буфера обмена.

Далее раскройте графическое окно 0 и с помощью команды: Файл => Копировать в буфер обмена выполните копирование диаграммы Боде. Затем раскройте файл sau.doc и вставьте в него диаграмму Боде из буфера обмена.

Раскройте графическое окно 1 и с помощью команды: Файл => Копировать в буфер обмена выполните копирование диаграммы (годографа) Найквиста. Затем раскройте файл sau.doc и вставьте в него годограф Найквиста из буфера обмена.

Далее раскройте графическое окно 2 и с помощью команды: Файл => Копировать в буфер обмена выполните копирование графика ЛАЧХ. Затем раскройте файл sau.doc и вставьте в него график ЛАЧХ из буфера обмена.

Раскройте графическое окно 3 и с помощью команды: Файл => Копировать в буфер обмена выполните копирование графика ФЧХ. Затем раскройте файл sau.doc и вставьте в него график ФЧХ из буфера обмена.

Затем сверните файл sau.doc. В редакторе SciNotes выполните команду: Файл => Закрыть.

Задание 4. Запишите в окне редактора SciNotes текст скрипта для вычисления корней характеристического уравнения САУ с целью определения устойчивости САУ.

Созданный скрипт-файл сохраните в папке Lab9 под именем, например, r_proj.sce. Запустите скрипт, используя команду меню: Выполнить => … без отображения команд. По значениям корней характеристического уравнения сделайте вывод об устойчивости САУ.

При отсутствии ошибок скопируйте скрипт-файл в буфер обмена, выполнив последовательно команды в редакторе SciNotes: Правка => Выделить все, затем Правка => Копировать. После этого раскройте файл sau.doc и вставьте в него скопированный скрипт из буфера обмена. Затем сверните файл sau.doc. В редакторе SciNotes выполните команду: Файл => Закрыть.

4 Содержание отчета

Отчет по лабораторной работе № 9 выполняется в виде списка скрипт-файлов, разработанных в ходе выполнения заданий для самостоятельной работы, а также файла в формате *.doc, в котором записаны скрипты и графики функций.

Контрольные вопросы

1. Какая функция используется в Scilab для определения передаточной функции линейной системы управления?
2. Какие функции используются в Scilab для расчета и построения временных характеристик систем управления?
3. Какие функции используются в Scilab для расчета и построения частотных характеристик систем управления?
4. Как выполняется в Scilab преобразование структурных схем?

5. Как в Scilab можно определить устойчивость системы управления?

Лабораторная работа № 10

ВИЗУАЛЬНОЕ МОДЕЛИРОВАНИЕ В SCILAB

1 Цель работы

Изучить и практически исследовать методы визуального математического моделирования систем автоматического управления.

2 Порядок выполнения работы

2.1 Создание папки для работы

При выполнении лабораторной работы № 10 вы будете использовать папку с именем Lab10, которую необходимо предварительно создать.

С этой целью откройте вашу рабочую папку (с вашей фамилией) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab10.

В дальнейшем вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы №10. Полный путь к этой папке будет такой:

F:\OTC\Ivanov\Lab10

2.2 Запуск и настройка среды разработки Scilab

2.2.1. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа – рабочий стол среды разработки Scilab.

2.2.2. Нужно установить в качестве активной (текущей) папку Lab9. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем ОТС, затем папку с вашей фамилией (условно Ivanov) и затем папку Lab10. В строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab10. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab10

2.3 Знакомство с интерфейсом Xcos

Xcos (в первых версиях называлась Scicos, от “Scilab Connected Object Simulator”) – это часть интегрированной среды разработки Scilab. Xcos позволяет осуществить визуальное математическое моделирование динамических систем различных объектов. Моделируемые объекты могут описываться как непрерывные и как дискретные. В данной лабораторной работе будут использоваться только непрерывные объекты.

Перед вызовом Xcos необходимо, чтобы была открыта интегрированная среда Scilab. Для вызова Xcos следует выбрать из главного меню: Инструменты => Визуальное моделирование Xcos или набрать слово xcos в командном окне Scilab.

Задание. Требуется произвести запуск Xcos.

После запуска Xcos по умолчанию на дисплее открываются два окна:

- окно Палитры блоков;
- окно графического редактора.

В окне Палитры блоков представлены группы блоков, из которых строится диаграмма (блочная модель) Xcos. Выделив нужную группу щелчком левой кнопкой мыши, вы увидите графическое изображение входящих в нее блоков. Чтобы построить диаграмму, нужно выделить нужные блоки в окне Палитры блоков и разместить их в окне графического редактора. Блоки перемещаются с помощью мыши (щелчок левой кнопкой, перетаскивание в нужное место). Затем блоки соединяются друг с другом, используя их входы и выходы.

Назначение некоторых наиболее употребительных блоков для моделирования систем управления приведены в таблице 1.

Таблица 1 – Блоки для моделирования систем управления

Название	Выполняемая функция	Палитра
STEP_FUNCTION	Генератор ступенчатого сигнала	Источник сигналов и воздействий
GENSIN_f	Генератор синусоидального сигнала	Источник сигналов и воздействий
CLOCK_c	Счетчик времени (часы активации)	Источник сигналов и воздействий
CLR	Непрерывная передаточная функция	Системы с непрерывным временем
CSCOPE	Одноканальный осциллограф	Регистрирующие устройства
CMSCOPE	Многоканальный осциллограф	Регистрирующие устройства
BIGSOM_f	Блок суммирования сигналов	Математические операции
SUMMATION	Блок вычитания сигналов	Математические операции
GAIN_f	Блок с коэффициентом передачи K	Математическая операция

Создание новой модели в Xcos предусматривает выполнение следующих действий:

- запуск Xcos с пустым окном графического редактора;
- последовательное открытие нескольких палитр;
- копирование нужных блоков из палитр в окно графического редактора;
- соединение входов и выходов блоков;
- установка параметров нужного значения в блоках;
- наименование и сохранение диаграммы модели (в файле с расширением .xcos);
- запуск процесса моделирования.

2.4 Пример построения и исследования простой САУ

2.4.1. Рассмотрим пример построения блочной диаграммы САУ, которая моделирует и визуализирует выходной сигнал y(t) при подаче на ее вход единичного ступенчатого сигнала (т.е. строит переходную характеристику h(t) системы).

Используемые блоки: STEP-FUNCTION, CLR, CSCOPE, CLOCK_c. Найдите блоки по их названию в Палитре блоков (для поиска используйте данные из таблицы 1) и перетащите каждый с помощью мыши в окно графического редактора, расположив в порядке, показанном на рисунке 1.

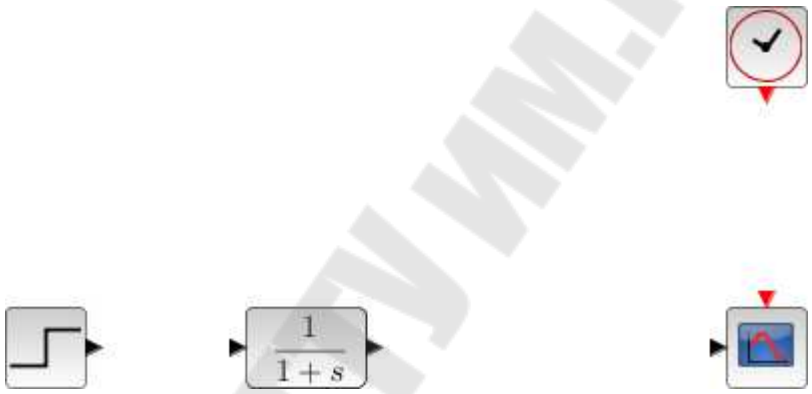


Рисунок 1 – Расположение блоков диаграммы

Диаграммы Xcos содержат два типа соединений: регулярные (синего цвета) и управляющие (красного цвета). По регулярным соединениям передаются сигналы данных, а по управляющим - сигналы активации. Блоки могут иметь регулярные и управляющие входы и выходы. Регулярные входы и выходы блоков располагаются слева и справа (черного цвета), а управляющие (красного цвета) – сверху и снизу относительно блока.

Блоки соединяются друг с другом при помощи линий, создаваемых щелчком левой кнопкой мыши на выходе одного блока и перемещением указателя мыши при нажатой левой кнопке на вход другого блока. Обратите внимание на то, что соединятся могут только блоки с выходами и входами одного цвета (черного с черным и красного с красным). Разрешенные к соединению входы и выходы при наведении курсора мыши подсвечиваются зеленым цветом.

Соедините блоки между собой согласно рисунка 2.

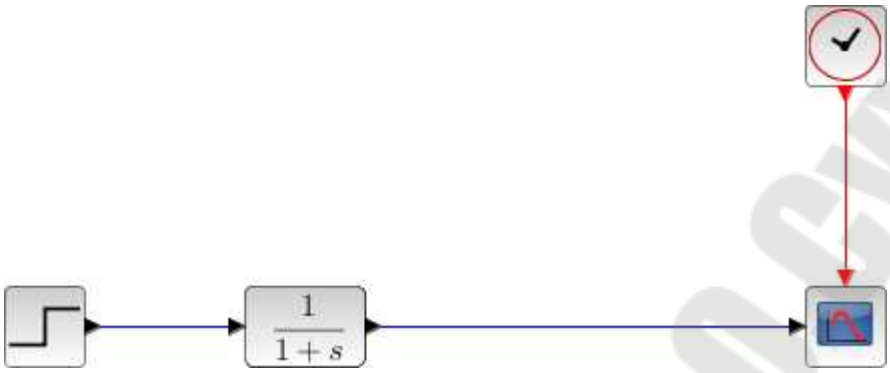


Рисунок 2 – Диаграмма (блок-схема) САУ

Запуск процесса моделирования может быть выполнен двумя способами:

1) с помощью команды из меню редактора: Моделирование => Запустить;

2) нажатием кнопки на панели инструментов редактора.

Остановить процесс моделирования можно также двумя способами:

1) с помощью команды из меню редактора: Моделирование => Остановить;

2) нажатием кнопки на панели инструментов редактора.

Задание. Запустите процесс моделирования блок-схемы, созданной в окне графического редактора.

На экране автоматически появляется графическое окно (осциллограмма), в котором отображается процесс моделирования. В нижней части окна будет строка: «Выполняется моделирование».

По графику в окне вы можете видеть, что время моделирования слишком велико, и переходный процесс давно закончился. Остановите процесс моделирования и закройте графическое окно.

Требуется модифицировать параметры процесса моделирования. С этой целью выполните команду меню редактора: Моделирование => Установка.

Откроется окно с заголовком «Параметры моделирования». Надо установить время моделирования 5 секунд. Для этого установите в строке «Конечное время интегрирования» значение 5.

Остальные параметры менять не нужно. В заключение щелкните по кнопке OK.

Далее надо модифицировать параметры блока CSCOPE (осциллограф). С этой целью дважды щелкните левой кнопкой мыши по блоку на диаграмме. Откроется окно с заголовком «Ввод значений». Установите значения ординат в пределах от 0 до 1,5 и период активации 5 секунд. Чтобы сделать это, измените следующие установки в окне:

Y_{min} : 0

Y_{max} : 1.5

Refresh period: 5

В заключение щелкните по кнопке OK.

Затем дважды щелкните мышью по блоку STEP_FUNCTION. Откроется окно с заголовком «Ввод значений». Установите параметр «Начальное время скачка» равным 0. Для этого измените установку:

Step time: 0

Щелкните по кнопке OK.

И, наконец, дважды щелкните мышью по блоку CLOCK_c (счетчик времени). Откроется окно с заголовком «Ввод значений». Установите параметр «Время инициализации», т.е. начало отсчета времени, равным 0. Для этого измените установку в окне:

Время инициализации: 0

Щелкните по кнопке OK.

Вновь запустите процесс моделирования. В графическом окне должна появиться осциллограмма переходного процесса (переходная характеристика), приведенная на рисунке 3.

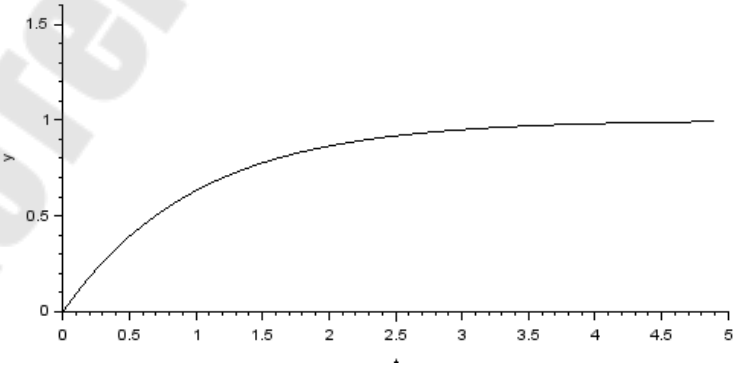


Рисунок 3 – Переходная характеристика звена

В заключение закройте графическое окно и сохраните разработанную блок-схему моделирования в папке Lab10 под именем, например, x_sau.xcos.

2.4.2. В рассмотренном примере в блоке CLR была использована стандартная передаточная функция вида:

$$W(s) = \frac{1}{1+s}$$

Имеется возможность изменять пользователем вид передаточной функции блока CLR.

Допустим, что требуется задать передаточную функцию блока в виде:

$$W(s) = \frac{0.1s+1}{0.1s^2+0.2s+1}$$

С этой целью дважды щелкните мышью по блоку CLR на диаграмме в окне редактора. Откроется окно с заголовком «Ввод значений». В окне будут приведены параметры блока по умолчанию. Модернизируйте эти параметры следующим образом:

Numerator: 1 + 0.1*s

Denominator: 1 + 0.2*s + 0.1*s^2

Затем щелкните по кнопке OK.

На диаграмме в блоке CLR появилось значение новой передаточной функции. Так как значение функции не помещается в границы блока, то щелкните по блоку и растяните его границы влево и вправо. Окончательный вид диаграммы должен соответствовать рисунку 4.

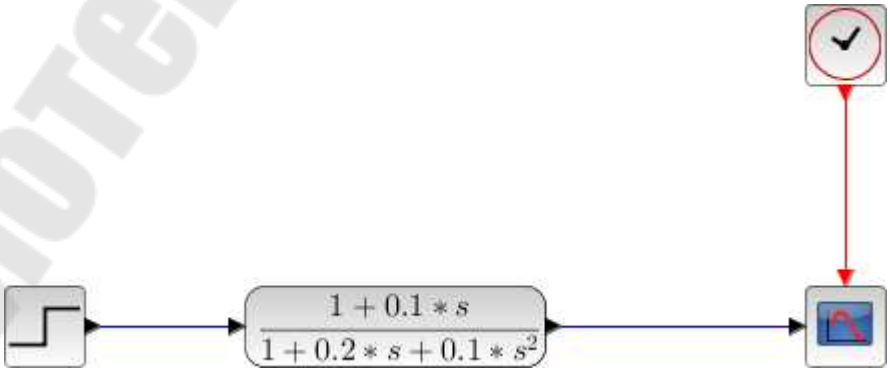


Рисунок 4 – Диаграмма модернизированной САУ

Запустите процесс моделирования и проанализируйте полученный график переходной функции. По графику в окне вы можете видеть, что переходный процесс имеет колебательный характер, и он не заканчивается при достижении заданного времени моделирования 5 секунд. Закройте графическое окно.

Требуется модифицировать параметры процесса моделирования. С этой целью выполните команду меню редактора: Моделирование => Установка. Откроется окно с заголовком «Параметры моделирования». Надо установить время моделирования 10 секунд. Для этого установите в строке «Конечное время интегрирования» значение 10. Остальные параметры менять не нужно. В заключение щелкните по кнопке [OK]. Далее надо модифицировать параметры блока CSCOPE (осциллограф). С этой целью дважды щелкните левой кнопкой мыши по блоку на диаграмме. Откроется окно с заголовком «Ввод значений». Установите период активации 10 секунд. Для этого запишите в строке Refresh period значение 10. В заключение щелкните по кнопке [OK].

Вновь запустите процесс моделирования. В графическом окне должна появиться осциллограмма переходного процесса (переходная характеристика), приведенная на рисунке 5.

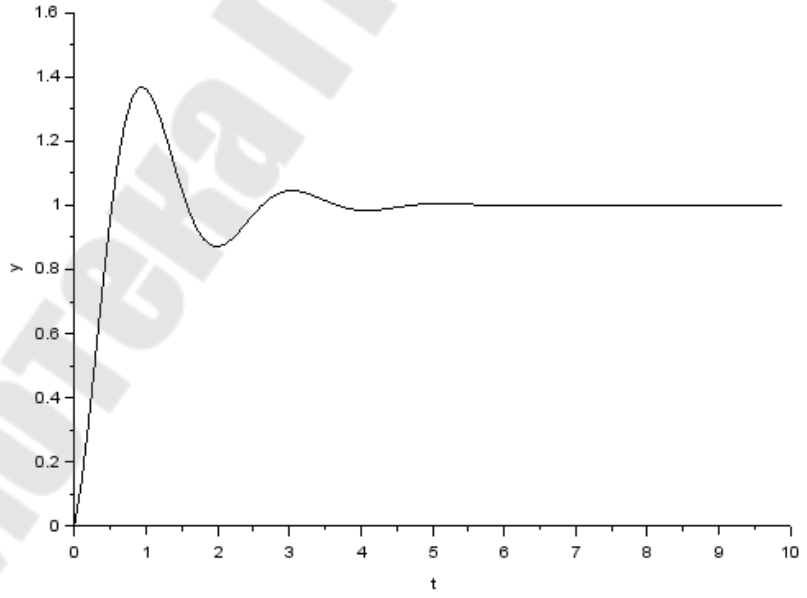


Рисунок 5 – Переходная характеристика h(t) звена

В заключение сохраните блок-схему в папке Lab10 под именем. например, x_sau_2.xcos. Очистите окно редактора, выполнив команду: Файл => Закрыть.

2.4.3. Для исследования частотных свойств САУ нужно в качестве источника входного воздействия использовать генератор синусоидального сигнала. В Xcos для этой цели имеется блок GENSIN_f. На рисунке 6 приведена блок-схема САУ для исследования частотных характеристик. В качестве регистрирующего взят блок CMSCOPE (многоканальный осциллограф).

Задание. В окне графического редактора Xcos выполните блок-схему по рисунку 6 для исследования частотных характеристик САУ.

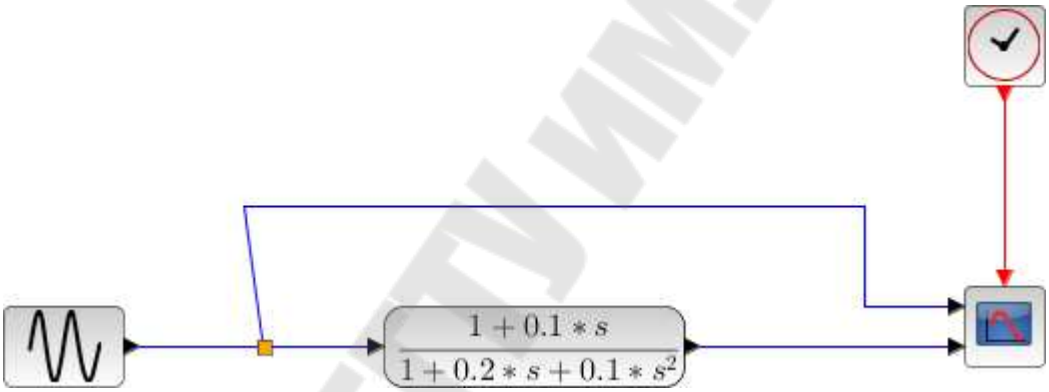


Рисунок 6 – Диаграмма САУ для исследования частотных характеристик

Установите следующие параметры модели.

В редакторе выполните команду: Моделирование => Установка. Откроется окно с заголовком «Параметры моделирования». Надо установить время моделирования 5 секунд. Для этого установите в строке «Конечное время интегрирования» значение 5. Остальные параметры менять не нужно. В заключение щелкните по кнопке OK.

Далее надо установить параметры блока CMSCOPE (многоканальный осциллограф). С этой целью дважды щелкните левой кнопкой мыши по блоку на диаграмме. Откроется окно с заголовком «Ввод значений». По умолчанию блок CMSCOPE моделирует двухканальный осциллограф. Его параметры задаются вектором из двух элементов. Первый элемент вектора относится к

первому (верхнему на диаграмме) каналу, а второй элемент – ко второму каналу (нижнему на диаграмме). Измените следующие установки в окне:

Y_{min} vector: -1 -0.5

Y_{max} vector: 1 0.5

Refresh period: 5 5

В заключение щелкните по кнопке OK.

Затем дважды щелкните мышью по блоку GENSIN_f. Откроется окно с заголовком «Ввод значений». Установите параметр:

Частота: 10

Щелкните по кнопке OK.

И, наконец, дважды щелкните мышью по блоку CLOCK_c (счетчик времени). Откроется окно с заголовком «Ввод значений». Установите следующие параметры:

Period: 0.01

Время инициализации: 0

Щелкните по кнопке OK.

Запустите процесс моделирования. Откроется графическое окно с двумя осциллограммами. На верхней будет входное синусоидальное воздействие, а на нижней – выходная величина (рисунок 7).

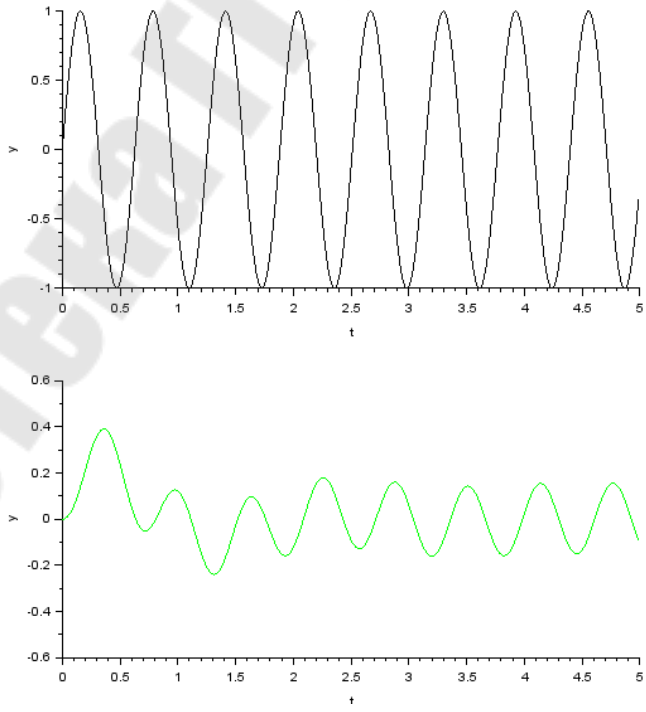


Рисунок 7 – Реакция звена на гармонический сигнал

Закройте графическое окно. Сохраните блок-схему в папке Lab10 под именем, например, x_sau3.xcos. Затем очистите окно редактора с помощью команды: Файл => Закрыть.

2.5 Пример анализа САУ

Рассмотрим САУ, структурная схема которой приведена на рисунке 8.

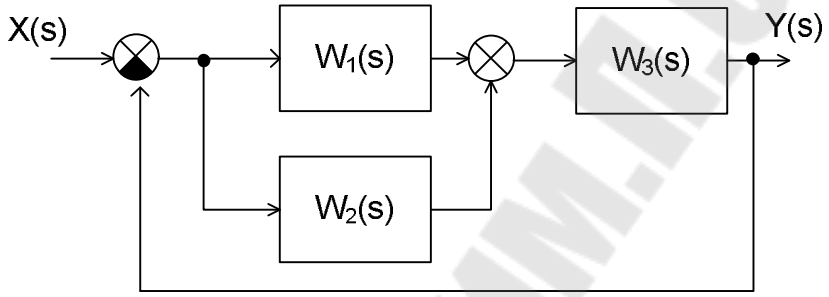


Рисунок 8 – Структурная схема САУ

Передаточные функции звеньев:

$$W_1(s) = 4\,; W_2(s) = \frac{0.05s + 1}{0.1s + 1}\,; W_3(s) = \frac{1}{s(0.5s + 1)}\,.$$

Задание. Создайте блок-схему по рисунку 8 в Xcos для исследования САУ и построения графика переходной функции h(t).

Найдите необходимые блоки в Палитре блоков (для поиска используйте данные из таблицы 1) и переместите каждый с помощью мыши в окно редактора, расположив в порядке, показанном на рисунке 9.

Пояснение. Передаточная функция первого звена $W_1(s) = 4$. Это пропорциональное (безынерционное) звено с коэффициентом передачи 4. Для его моделирования лучше использовать блок GAIN_f, который выполняет функции блока с коэффициентом передачи К. В данном случае К = 4.

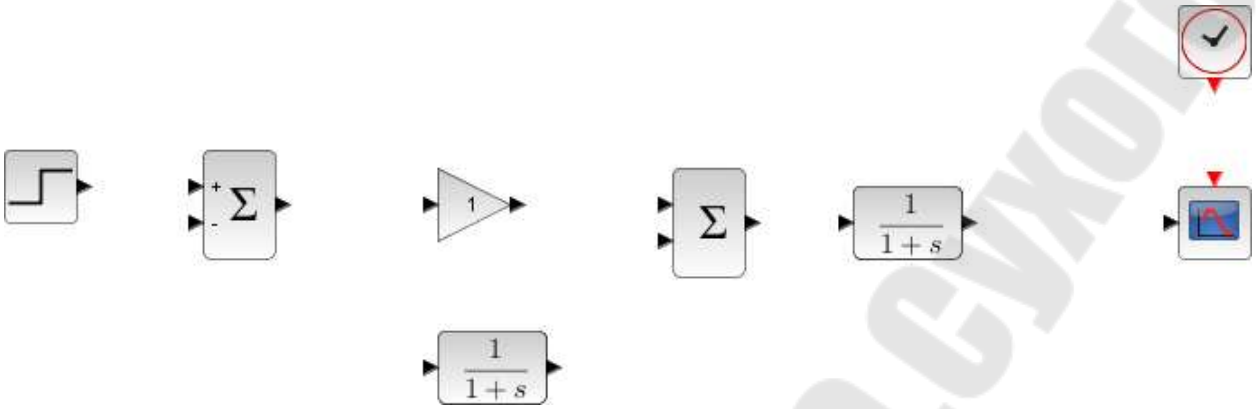


Рисунок 9 – Расположение блоков САУ для построения диаграммы

Далее нужно выполнить корректировку передаточных функций блоков на рисунке 9 согласно исходным данным. После этого надо выполнить соединение блоков согласно блок-схеме рисунка 8. Полученная диаграмма для моделирования САУ приведена на рисунке 10.

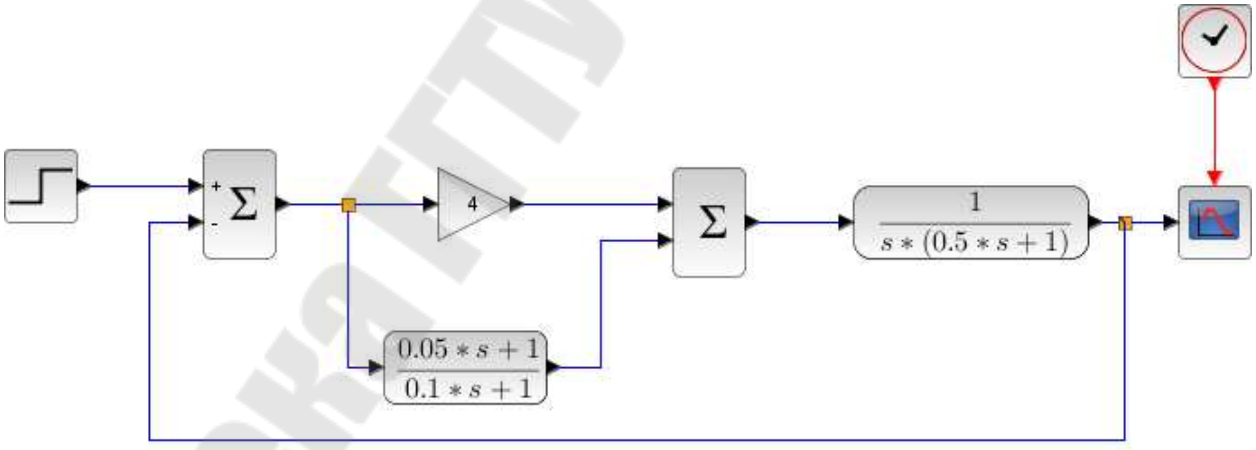


Рисунок 10 – Диаграмма (блок-схема) САУ для исследования переходной характеристики

Запустите процесс моделирования. При необходимости произведите корректировку временных параметров блоков. Сохраните файл диаграммы в папке Lab10 под именем, например, x_sau4.xcos.

Задание. Создайте блок-схему по рисунку 11 в Xcos для исследования частотных характеристик САУ.

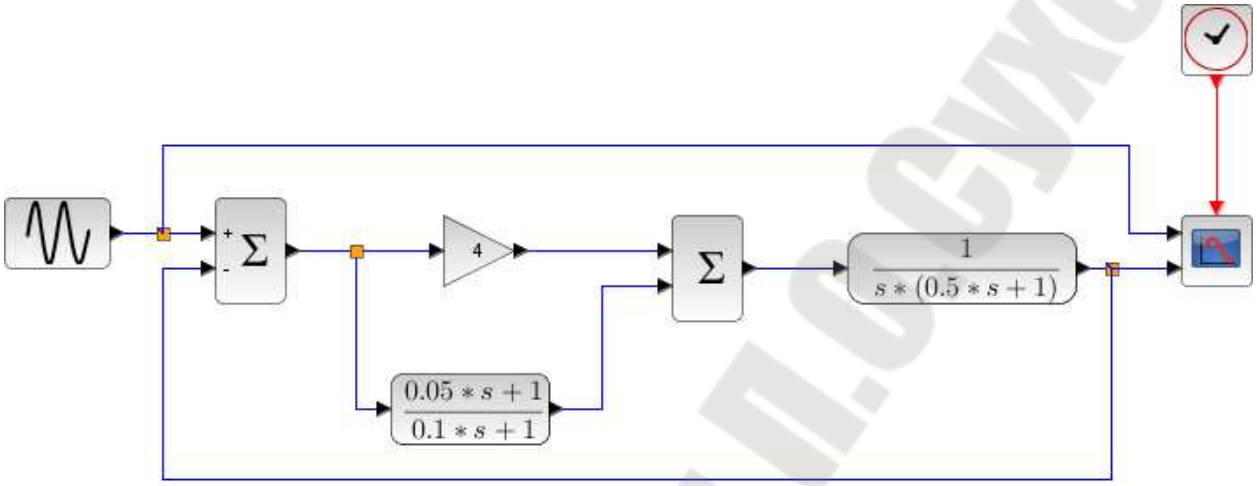


Рисунок 11 – Диаграмма САУ для исследования частотных характеристик

Установите необходимые параметры блоков модели (по аналогии с п. 2.4). Запустите процесс моделирования. Откроется графическое окно с двумя осциллограммами. На верхней будет входное синусоидальное воздействие, а на нижней – выходная величина. При необходимости откорректируйте временные параметры блоков.

Сохраните блок-схему в папке Lab10 под именем. например, x_sau5.xcos.

3 Задания для самостоятельной работы

Перед выполнением заданий для самостоятельной работы вам необходимо создать в редакторе WORD файл, например, с именем sau_2.doc и сохранить его в папке Lab10. В этом файле вы будете сохранять диаграммы и графики функций, созданные в Xcos, которые необходимы для оформления пояснительной записки к курсовому проекту по ОТС.

Задание 1. Создайте в окне графического редактора Xcos диаграмму (блок-схему) для исследования и построения графика переходной функции для САУ по заданию курсового проекта.

Созданный файл сохраните в папке Lab10 под именем, например, t_proj.xcos. Запустите процесс моделирования. При необходимости сделайте коррекцию временного диапазона. Затем скопируйте файл диаграммы в буфер обмена, выполнив последовательно команды в редакторе Xcos: Правка => Выбрать все, затем Правка => Копировать. После этого раскройте файл sau_2.doc и вставьте в него скопированную диаграмму из буфера обмена.

Далее с помощью команды: Файл => Копировать в буфер обмена выполните копирование графика переходной функции. Затем раскройте файл sau_2.doc и вставьте в него график переходной функции из буфера обмена.

Затем сверните файл sau_2.doc. В редакторе Xcos выполните команду: Файл => Закрыть.

Задание 2. Создайте в окне графического редактора Xcos диаграмму (блок-схему) для исследования частотных характеристик для САУ по заданию курсового проекта. Входное воздействие – генератор синусоидального сигнала.

Созданную диаграмму сохраните в папке Lab10 под именем, например, f_proj.xcos. Запустите процесс моделирования. При необходимости сделайте коррекцию временного диапазона. Затем скопируйте файл диаграммы в буфер обмена, выполнив последовательно команды в редакторе Xcos: Правка => Выбрать все, затем Правка => Копировать. После этого раскройте файл sau_2.doc и вставьте в него скопированную диаграмму из буфера обмена.

Далее с помощью команды: Файл => Копировать в буфер обмена выполните копирование осциллограмм процессов. Затем раскройте файл sau_2.doc и вставьте в него осциллограммы из буфера обмена.

Затем сверните файл sau_2.doc. В редакторе Xcos выполните команду: Файл => Закрыть.

4 Содержание отчета

Отчет по лабораторной работе № 10 выполняется в виде списка файлов, разработанных в ходе выполнения заданий для самостоятельной работы, а также файла в формате .doc, в котором сохранены диаграммы и графики функций.

Контрольные вопросы

1. Объясните назначение Xcos.
2. Структура библиотеки Xcos.
3. Назовите основные блоки Sours-Источники, используемые при моделировании систем автоматического управления.
4. Назовите основные блоки Sinks-Получатели, используемые при моделировании систем автоматического управления.
5. Правила построения моделей систем управления в Xcos.
6. Объясните правила моделирования в Xcos систем управления.

Лабораторная работа № 11

РАЗРАБОТКА ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ В SCILAB

1 Цель работы

Изучить и практически исследовать методы разработки графического интерфейса пользователя в интегрированной среде Scilab.

2 Порядок выполнения работы

2.1 Создание папки для работы

При выполнении лабораторной работы № 11 вы будете использовать папку с именем Lab11, которую необходимо предварительно создать.

С этой целью откройте вашу рабочую папку (с вашей фамилией) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab11.

В дальнейшем вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 11. Полный путь к этой папке будет такой:

F:\OTC\Ivanov\Lab11

2.2 Запуск и настройка среды разработки Scilab

2.2.1. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа – рабочий стол среды разработки Scilab.

2.2.2. Нужно установить в качестве активной (текущей) папку Lab11. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем ОТС, затем папку с вашей фамилией (условно Ivanov) и затем папку

Lab11. В строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab11. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab11

2.3 Исследование функции вывода сообщения в стандартное окно

В Scilab имеется функция messagebox( ) для вывода различных сообщений в стандартное окно. Все варианты использования этой функции можно получить по справке:

help messagebox

В данной лабораторной работе мы рассмотрим только простейшие варианты использования этой функции. Демонстрационные примеры надо выполнить в командном окне Scilab.

1. Стандартное окно с заголовком (по умолчанию):

Сообщение

Для вывода текста сообщения в виде одной строки используется формат:

messagebox(‘Single line message’);

Пример. Вывести в окно сообщение с текстом:

Введите исходные параметры

Задание 1. В командном окне наберите следующее:

--> messagebox(‘Введите исходные параметры’);

Для закрытия окна с сообщением следует щелкнуть мышью по кнопке [OK].

2. Для вывода текста сообщения в виде нескольких строк используется формат:

messagebox([‘line1’ ‘line2’ ‘line3’]);

Пример. Нужно вывести сообщение в виде двух строк:

Введите уравнение в виде:

a * x + b

Задание 2. В командном окне наберите следующее:

--> messagebox([‘Введите уравнение в виде’ ‘a * x + b’]);

3. Пользователь может изменить заголовок окна, используя следующий формат:

messagebox(‘Message’, ‘user defined title’);

Пример. Нужно определить заголовок окна:

Получение результата

и вывести сообщение:

Для получения результата нажмите кнопку

Задание 3. В командном окне наберите следующий текст:

--> messagebox(‘Для получения результата нажмите кнопку’, …

--> ‘Получение результата’);

4. Для вывода сообщения об ошибке можно использовать окно со значком ошибки. Формат такого сообщения:

messagebox(‘An error message’, ’Error’, ‘error’)

Пример. Требуется вывести сообщение об ошибке в окне с заголовком:

Сообщение об ошибке

и вывести текст:

Ошибка при вводе параметров

Задание 4. В командном окне наберите следующее:

--> messagebox(‘Ошибка при вводе параметров’, ‘Сообщение об ошибке’,…

--> ‘error’);

Задание 1 для самостоятельной работы. Требуется вывести в стандартное окно следующие сообщения.

1. Текст с двумя строками:

Введите передаточную функцию

в виде: (s+1)/(s^2+2*s+1)

2. Текст в виде:

Не все параметры введены

Заголовок окна:

Сообщение об ошибке

В окне должен быть значок ошибки.

2.4 Создание графического окна

В визуальных приложениях среды Scilab основным объектом является графическое окно (англоязычный термин – figure). Каждое графическое окно имеет идентификатор, который называется дескриптором графического окна или просто дескриптором (англоязычный термин – handle).

Для создания графического окна служит функция figure( ), например,

hFig = figure( )

В результате выполнения этой команды будет создано пустое графическое окно. Дескриптор окна будет записан в переменную с именем hFig.

Обычно при создании графического окна задаются его параметры. В этом случае обращение к функции figure( ) будет иметь вид:

hFig = figure(‘свойство_1’, ‘значение_1’, ‘свойство_2’, ‘значение_2’, …)

Здесь ‘свойство_1’ – название первого параметра, а значение_1 – его значение; ‘свойство_2’ – название второго параметра, а значение_2 – его значение и т. д.

Примечание. Текст ‘значение’ будет заключен в кавычки, если значением параметра является строка, если же значением параметра является число, то кавычки использовать не надо.

Размер и положение окна на экране компьютера можно задавать с помощью параметра

‘position’, [x, y, dx, dy],

где x, y – положение верхнего левого угла окна (по горизонтали и вертикали соответственно) относительно верхнего левого угла экрана;

dx – размер окна по горизонтали (ширина окна) в пикселях;

dy – размер окна по вертикали (высота окна) в пикселях.

Для изменения заголовка окна используется параметр

‘figure_name’, ‘имя_окна’

Например, создадим графическое окно с именем FIRST WINDOW командой:

```
hFig = figure('position', [20, 40, 500, 400], ..
    'figure_name', 'FIRST WINDOW');
```

Графическое окно можно закрыть с помощью функции

```
close(hFig),
```

где hFig – дескриптор окна.

Задание 1. Создайте графическое окно без имени размером 400 × 300 пикселов, расположив его на расстоянии 100 × 200 пикселов от левого верхнего угла дисплея. Затем закройте его с помощью команды.

Замечание. Для подавления вывода в командное окно информации о созданном графическом окне нужно после команды figure( ) ставить ";".

Задание 2. Создайте графическое окно с именем Мое окно размером 700 × 300 пикселов и расположенное на расстоянии 50 × 100 от левого верхнего угла дисплея. Затем закройте это окно.

Полную информацию о параметрах графического окна можно получить из справочной системы Scilab, набрав в командном окне

```
--> help figure
```

2.5 Динамическое создание интерфейсных компонентов

В Scilab используется динамический способ создания интерфейсных компонентов. Он заключается в том, что на стадии выполнения программы могут создаваться (и удаляться) те или иные элементы управления (кнопки, метки, флажки и т.д.) и их свойствам присваиваться соответствующие значения.

Для создания любого интерфейсного компонента с заданными свойствами используется функция uicontrol( ), возвращающая дескриптор формируемого компонента. Компонент, как правило, создается внутри текущего графического окна, которое будет являться родителем этого компонента. Синтаксис функции uicontrol( ):

hComp = uicontrol(hFig, ‘style’,‘тип_компонента’, ..
‘свойство_1’,‘значение_1’, … ‘свойство_k’, ‘значение_k’)

Здесь hComp – дескриптор создаваемого компонента;

hFig – дескриптор объекта, внутри которого будет создаваться компо- нент (чаще всего этом объектом является графическое окно); первый аргумент функции uicontrol( ) не является обязательным, и если он отсутствует, то родителем создаваемого компонента является текущий графический объект – текущее графическое окно;

‘style’ – служебная строка, которая указывает на стиль создаваемого компонента;

‘тип_компонента’ – определяет, к какому классу принадлежит создавае- мый компонент. Это может быть Pushbutton, Text, Edit, Radiobutton или

другие компоненты;

‘свойство_k’, ‘значение_k’ – определяют свойства и значения отдельных компонентов.

У существующего интерфейсного компонента можно изменить те или иные свойства с помощью функции set( ):

set(hComp, ‘свойство_1’, ‘значение_1’, … , ‘свойство_k’, ‘значение_k’)

Здесь hComp – дескриптор динамического компонента, параметры которого будут меняться.

Получить значение параметра компонентов можно с помощью функции get( ) следующей структуры:

get(hComp, ‘свойство’)

Здесь hComp – дескриптор динамического интерфейсного компонента, значе- ние параметра которого необходимо узнать;

‘свойство’ – имя параметра, значение которого нужно узнать.

Функция get( ) возвращает значение параметра.

Полную информацию о свойствах компонентов Scilab можно получить по справке

help uicontrol properties

2.6 Исследование программы с компонентом «Командная кнопка»

Компонент командная кнопка типа PushButton создается с помощью функции uicontrol( ), в которой параметру ‘style’ необходимо присвоить значение ‘pushbutton’. По умолчанию она не снабжена никакой надписью, имеет серый цвет и располагается в левом нижнем углу графического окна. Надпись на кнопке можно установить с помощью свойства ‘string’. Размер кнопки и положение ее в графическом окне можно задавать с помощью параметра

‘position’, [x y dx dy],

где x, y – положение нижнего левого угла кнопки (по горизонтали и вертикали соответственно) относительно нижнего левого угла графического окна;

dx – размер кнопки по горизонтали в пикселах;

dy – размер кнопки по вертикали (в пикселах).

Главным назначением командной кнопки является вызов функции, реагирующей на щелчок по кнопке. Щелчок по кнопке генерирует событие callback, которое указывается как параметр функции uicontrol( ). Значением параметра callback является строка с именем функции, вызываемой при щелчке по кнопке. В таком случае функция uicontrol( ) принимает вид:

```
hPbut = uicontrol(‘style’, ‘pushbutton’, ‘string’, ‘Кнопка’, ..
    ‘callback’, ‘function’)
```

Здесь function – имя вызываемой при наступлении события callback функции.

В качестве примера рассмотрим окно с кнопкой, при щелчке по которой производится вывод сообщения в стандартное окно.

Задание. В редакторе Scinotes наберите текст следующего скрипта.

```
clear; clc;  // очистить рабочее пространство, очистить командное окно
                // создаем графическое окно
hFig=figure('position', [100, 100, 400, 200], ..
    'figure_name', 'Кнопка с сообщением');
      // создаем командную кнопку
hPbut=uicontrol('style', 'pushbutton', 'position', [50, 50, 100, 20],..
    'string', 'Вызов', 'callback', 'call_msg');
// функция вывода сообщения при нажатии на кнопку
function call_msg( )
messagebox('Нужно ввести входные параметры');
endfunction
```

Сохраните скрипт в папке Lab11 под именем, например, but_msg.sce. Запустите скрипт на выполнение из окна редактора командой: Запустить => без отображения команд. Созданное программой графическое окно приведено на рисунке 1.

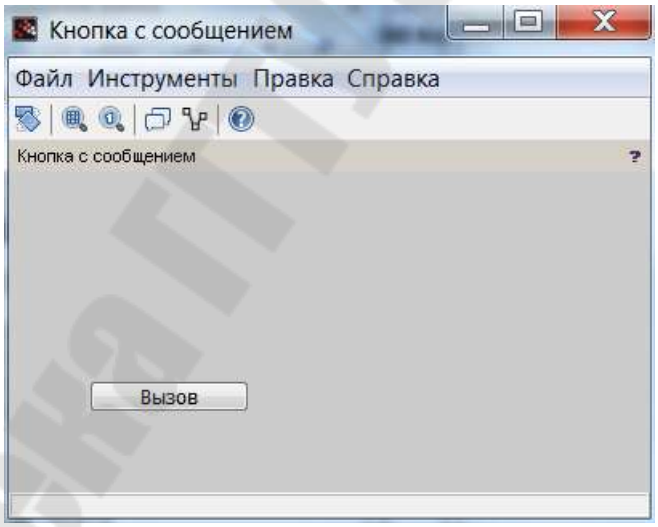


Рисунок 1 – Графическое окно с кнопкой

В появившемся графическом окне щелкните по кнопке. Наблюдайте за появлением окна сообщения. Закройте его щелчком по кнопке [OK]. Затем несколько раз щелкните по кнопке «Вызов», каждый раз закрывая окно сообщения. В заключение закройте графическое окно и файл скрипта.

2.7 Исследование программы с компонентом «Метка»

Следующим наиболее часто используемым компонентом является метка – текстовое поле для отображения символьной информации. Для определения метки значение параметра ‘style’ в функции uicontrol( ) должно иметь значение ‘text’. Компонент предназначен для вывода символьной строки (или нескольких строк). Выводимый на метку текст – значение параметра ‘string’ – может быть изменен только из программы.

Рассмотрим пример создания текстового поля (метки).

Задание. Наберите в командном окне следующую программу:

```
--> hFig = figure(‘position’, [20, 20, 500, 400]);
--> hText = uicontrol(‘style’, ‘text’, ‘position’, [50, 50, 100, 20], ..
    ‘string’, ‘Метка’);
```

Закройте созданное графическое окно с помощью мыши.

Одним из основных свойств метки является горизонтальное выравнивание текста, которое определяется свойством HorizontalAlignment. Это свойство может принимать одно из следующих значений:

left – выравнивание текста по левому краю (значение по умолчанию);

center – выравнивание текста по центру;

right – выравнивание текста по правому краю.

В качестве примера рассмотрим графическое окно, содержащее 4 текстовых поля с разными значениями свойств HorizontalAlignment.

Задание. В редакторе SciNotes наберите текст следующего скрипта.

```
clear, clc;
hFig = figure(‘position’, [50, 50, 300, 200]);
hText1 = uicontrol(‘style’, ‘text’, ‘position’, [30, 30, 150, 20], ..
    ‘string’, ‘Метка 1’);
hText2 = uicontrol(‘style’, ‘text’, ‘position’, [30, 60, 150, 20], ..
    ‘string’, ‘Метка 2’, ‘HorizontalAlignment’, ‘center’);
hText3 = uicontrol(‘style’, ‘text’, ‘position’, [30, 90, 150, 20], ..
    ‘string’, ‘Метка 3’, ‘HorizontalAlignment’, ‘right’);
```

```
hText4 = uicontrol('style', 'text', 'position', [30, 120, 150, 20], ..
    'string', 'Метка 4', 'HorizontalAlignment', 'left');
```

Скрипт сохраните в папке Lab11 под именем, например, text.sce. Запустите скрипт на выполнение из окна редактора. Созданное программой графическое окно с метками приведено на рисунке 2.

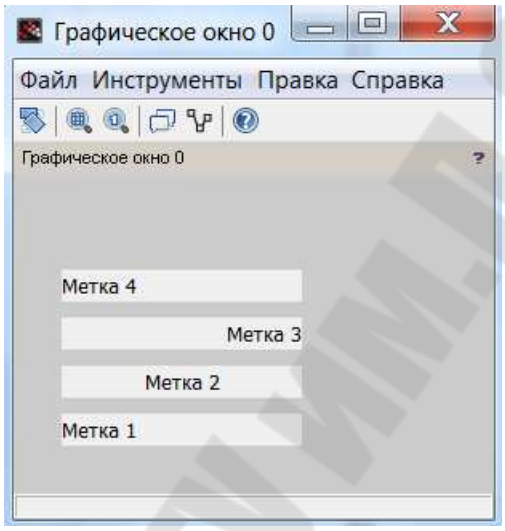


Рисунок 2 – Графическое окно с метками

Проанализируйте влияние свойства HorizontalAlignment на расположение текста в полях меток.

В заключение закройте графическое окно и выполните команду редактора: Файл => Закрыть.

2.8 Исследование компонента «Окно редактирования»

Интерфейсный компонент «Окно редактирования» может использоваться для ввода и вывода символьной информации. У этого компонента свойство 'style' должно принимать значение 'edit'. Текст, набираемый в окне редактирования, можно корректировать. При работе с компонентом можно использовать операции с буфером обмена. Процедура ввода, завершаемая нажатием клавиши [ENTER], генерирует событие callback.

Строка ввода определяется параметром 'string', который определяет находящийся в компоненте текст. Для нормального

функционирования компонента этот параметр необходимо обязательно задавать при определении компонента с помощью функции uicontrol( ). Изменить значение этого свойства можно с помощью функции set( ), а считать его значение – с помощью функции get( ).

Вводимый текст может быть прижат к левому или правому краю окна ввода, если задать соответствующее значение свойства HorizontalAlignment (по аналогии с компонентом «Метка»). Если вводимый текст представляет собой числовое значение, которое должно быть использовано в работе программы, то содержимое свойства ‘string’ переводится в числовой формат с помощью функции eval( ) или функции evstr( ).

2.8.1. Для исследования компонента «Окно редактирования» выполните следующее.

Задание. В редакторе SciNotes наберите текст скрипта, в котором создаются два поля меток и два окна редактирования.

```
clear, clc;
hFig = figure(‘position’, [20, 20, 300, 300]);
hText1 = uicontrol(‘style’, ‘text’, ‘position’, [100, 230, 100, 20], ..
    ‘string’, ‘Ввод числа’);
hEdit1 = uicontrol(‘style’, ‘edit’, ‘position’, [100, 200, 100, 20], ..
    ‘string’, ‘ ‘);
hText2 = uicontrol(‘style’, ‘text’, ‘position’, [100, 130, 100, 20], ..
    ‘string’, ‘Ввод текста’);
hEdit2 = uicontrol(‘style’, ‘edit’, ‘position’, [100, 100, 100, 20], ..
    ‘string’, ‘ ‘);
```

Скрипт сохраните в папке Lab11 под именем, например, edit.sce. Запустите скрипт на выполнение из окна редактора. Созданное программой графическое окно приведено на рисунке 3.

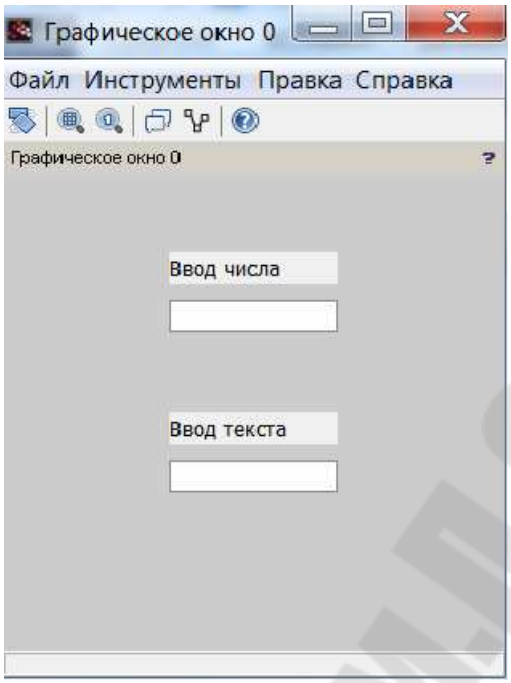


Рисунок 3 – Графическое окно с полями меток и редактирования

Для исследования работы программы щелкните мышью по полю редактирования, расположенному ниже метки «Ввод числа». После появления курсора введите с клавиатуры произвольное число.

Щелкните мышью по полю редактирования, расположенному ниже метки «Ввод текста». После появления курсора введите с клавиатуры произвольный текст.

Затем удалите введенные числа и текст из полей редактирования и введите новые данные.

В заключение закройте графическое окно и выполните команду редактора: Файл => Закрыть.

2.8.2. Исследуем программу, которая вычисляет сумму двух чисел, введенных с клавиатуры. Для работы программу создается графическое окно, в котором расположены: поле редактирования (дескриптор hEdit1) для ввода первого числа и поле редактирования (дескриптор hEdit2) для ввода второго числа. Каждое поле редактирования имеет соответствующие метки (дескрипторы hText1 и hText2). Результат операции суммирования заносится в поле метки (дескриптор hText3). Для управления работой программы имеются две кнопки: кнопка с надписью «Сложить» (дескриптор hPbut_add)

вызывает функцию суммирования чисел add( ); кнопка с надписью «Выход» (дескриптор hPbut_close) вызывает функцию закрытия приложения, при этом закрывается графическое окно программы.

Расположение компонентов программы вместе с именами их дескрипторов приведено на рисунке 4.

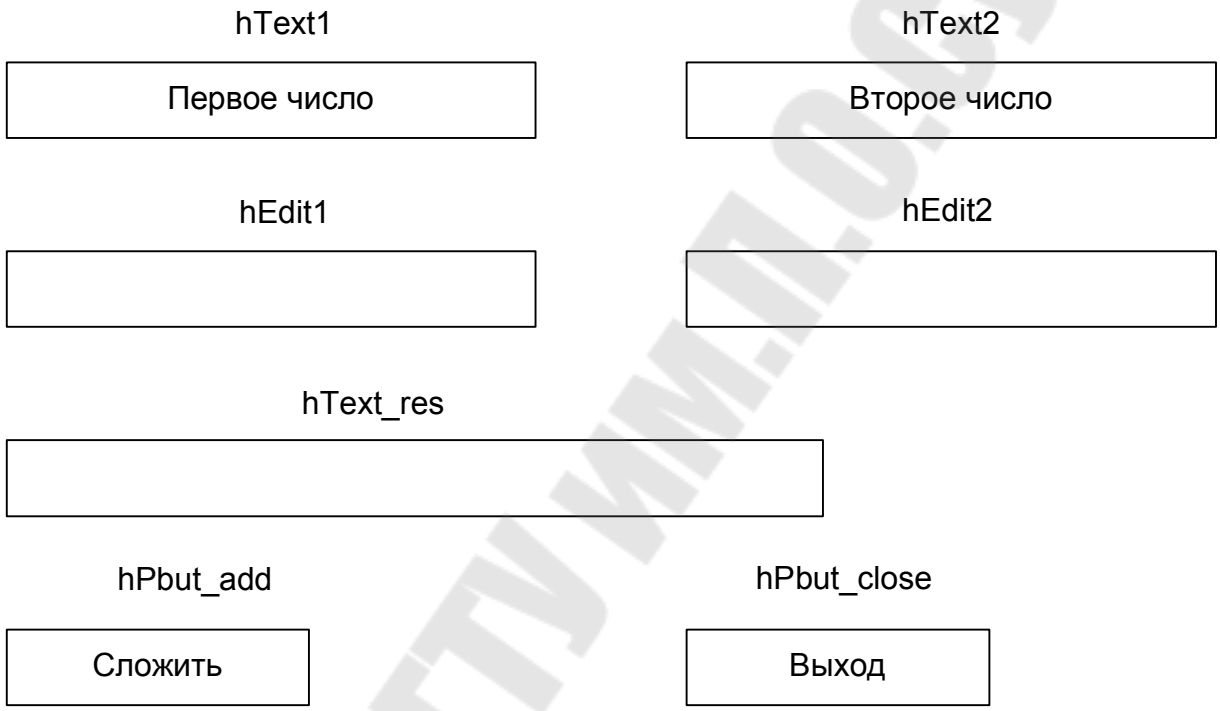


Рисунок 4 – Расположение компонентов в графическом окне

Задание. В редакторе SciNotes наберите текст скрипта для суммирования чисел:

```
clear, clc;
hFig = figure('position', [10, 10, 600, 400], ..
    'figure_name', 'Суммирование чисел');
hText1 = uicontrol('style', 'text', 'position', [100, 250, 100, 20],..
    'string', 'Первое число');
hText2 = uicontrol('style', 'text', 'position', [300, 250, 100, 20],..
    'string', 'Второе число');
hEdit1 = uicontrol('style', 'edit', 'position',[100, 200, 100, 20], ..
    'string', ' ');
hEdit2 = uicontrol('style', 'edit', 'position', [300, 200, 100, 20], ..
    'string', ' ');
```

```
hText_res = uicontrol('style','text', 'position',[100, 150, 250, 20], ..
    'string', ' ');
                    // кнопки управления
hPbut_add = uicontrol('style','pushbutton', 'position', [100, 100, 120,
20],..
    'string', 'Сложить', 'callback', 'add');
hPbut_close = uicontrol('style', 'pushbutton', 'position', [300, 100, 120,
20],..
    'string', 'Выход', 'callback', 'finish');
                    // функция суммирования чисел
function add( )
                    // получение численных значений введенных чисел
    num1 = eval(get(hEdit1, 'string'));
    num2 = eval(get(hEdit2, 'string'));
                    // суммирование и вывод
    sum = num1 + num2;
    set(hText_res, 'string', mprintf('Сумма равна %.2f\n', sum));
endfunction
                    // функция завершения программы
function finish( )
    close(hFig);
endfunction
```

Скрипт сохраните в папке Lab11 под именем, например, add.sce. Запустите скрипт на выполнение из окна редактора. Графическое окно программы приведено на рисунке 5.

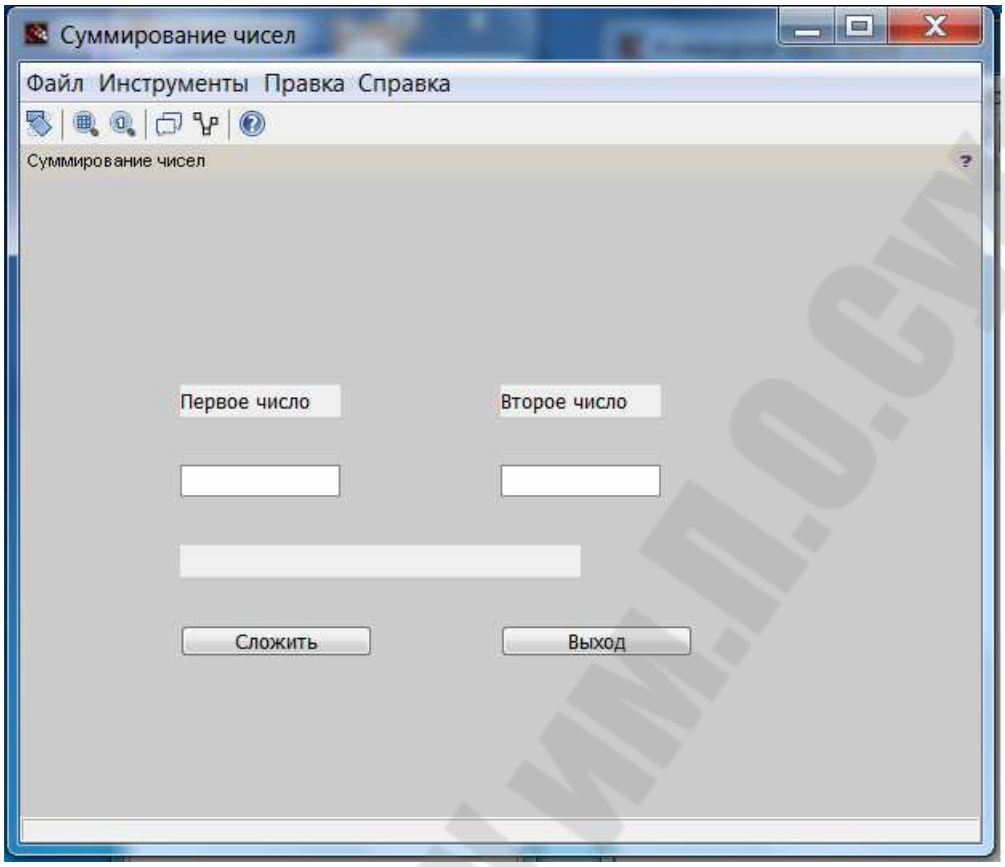


Рисунок 5 – Графическое окно для программы суммирования чисел

Для исследования работы программы введите в поля редактирования два произвольных числа (целые или дробные). Затем щелкните по кнопке «Сложить». Наблюдайте за текстом в окне результата.

Затем введите новые данные и щелкните по кнопке «Сложить».

В заключение щелкните по кнопке «Выход», что должно привести к закрытию графического окна (закрытию приложения). Затем закройте файл скрипта, выполнив команду редактора: Файл => Закрыть.

2.9 Исследование компонента «Список строк»

Интерфейсный компонент «Список строк» в простейшем случае можно рассматривать как окно с массивом строк в нем. Если длина списка превышает высоту окна, то для перемещения по списку может

использоваться вертикальная полоса прокрутки, которая генерируется автоматически.

Создание списка строк производится с помощью функции uicontrol( ) при задании параметра ‘style’ – ‘listbox’.

Список позволяет пользователю выбрать одну или несколько строк, и в зависимости от выбора произвести то или иное действие. Выбор строки осуществляется щелчком левой кнопкой мыши в тот момент, когда курсор указывает на выбираемую строку. Одновременно с подсветкой строки ее номер заносится в свойство ‘value’, и генерируется событие ‘callback’. Строки в списке нумеруются от 1.

Рассмотрим создание списка и работу с ним на простом примере.

Задание. В окне редактора SciNotes наберите текст следующего скрипта.

```
clear; clc; /
/ очистить рабочее пространство и командное окно
                    // создаем графическое окно
hFig = figure(‘position’, [100, 100, 300, 300], ..
    ‘figure_name’, ‘Окно со списком’);
                    // создаем компонент «Окно со списком»
hLbox = uicontrol(‘style’, ‘listbox’, ‘position’, [10, 10, 100, 100], ..
    ‘string’, ‘Строка 1 | Строка 2 | Строка 3’, ..
    ‘value’, 0, ‘callback’, ‘call_msg’);
                    // функция вывода сообщения при выборе строки из
списка
function call_msg( )
n = get(hLbox, ‘value’);
if n == 1 then
        messagebox(‘Выбрана строка 1’);
elseif n == 2 then
        messagebox(‘Выбрана строка 2’);
elseif n == 3 then
        messagebox(‘Выбрана строка 3’);
end;
endfunction
```

Сохраните набранный скрипт в папке Lab11 под именем, например, list.sce. Запустите скрипт из окна редактора. Графическое окно со списком приведено на рисунке 6.

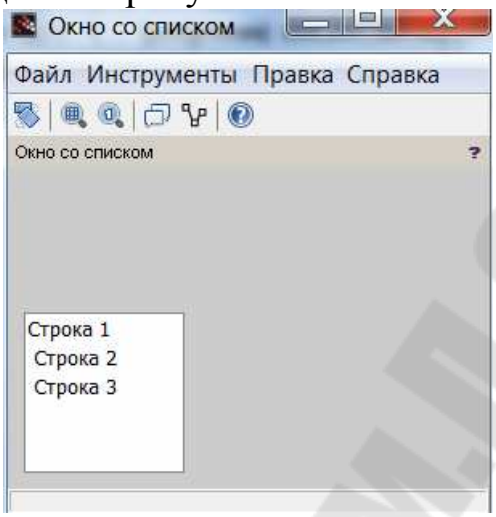


Рисунок 6 – Графическое окно со списком

При отсутствии ошибок исследуйте работу программы, щелкая мышью по строкам списка. Сделайте вывод о правильности работы программы. В заключение закройте графическое окно и закройте файл скрипта.

2.10 Исследование элемента управления «Переключатель»

2.10.1. Элемент управления Переключатель (Radiobutton) является индикатором альтернативных комбинаций. Он позволяет переключаться между состояниями или выключать одно из свойств. При создании переключателя его свойство ‘style’ должно иметь значение ‘radiobutton’.

Пример создания переключателя:

```
hFig = figure( );                    // создаем графическое окно
hRbut = uicontrol(‘style’, ‘radiobutton’, ‘position’, [25, 150, 100, 20], ..
    ‘string’, ‘Переключатель’);
```

При создании переключателя должно быть задано его состояние (параметр ‘value’). Переключатель может быть активен (значение

‘value’ равно 1) или нет (значение ‘value’ равно 0). Задать значение свойства ‘value’ можно также и с помощью функции set( ). Например:

set(hRbut, ‘value’, 0);

Получить значение свойства ‘value’ можно с помощью функции get( ):

n = get(hRbut, ‘value’);

Переключатель может реагировать на событие ‘callback’ и вызывать на выполнение определенную функцию.

Допустим, например, переключатель служит для вызова функции построения графика синусоиды. В этом случае переключатель создают с помощью следующего вида функции uicontrol( ):

hRbut = uicontrol(‘style’, ‘radiobutton’, ‘string’, ‘sin(x)’, ‘value’, 0,…

‘callback’, ‘call_sin’);

Здесь call_sin – это имя функции, которая будет вызываться при щелчке по переключателю. Однако следует помнить, что при щелчке по переключателю автоматически происходит смена его состояния.

2.10.2. Проиллюстрируем применение переключателей на примере программы, в которой с помощью переключателя можно выбрать функцию, график которой будет воспроизводиться в отдельном графическом окне.

Задание. Наберите в окне редактора SciNotes текст следующего скрипта.

```
                                // создаем первое графическое окно
hFig1 = figure('position', [50 50 300 300]);
                        // создание переключателей
hRbut1= uicontrol('style', 'radiobutton', 'position', [25, 140, 80, 20], ..
    'string', 'sin(x)', 'value', 0, 'callback', 'Radio1');
hRbut2 = uicontrol('style', 'radiobutton', 'position', [25, 100, 80, 20],..
    'string', 'cos(x)', 'value', 0, 'callback', 'Radio2');
                        // создаем окно для графиков функций
hFig2 = figure('position', [400, 50, 300, 300]);
                    // задаем диапазон изменения аргумента функций
x = -2*%pi : 0.1 : 2*%pi;
```

```
            // функция, реагирующая на щелчок по 1-й кнопке
function Radio1( )
    if get(hRbut1, 'value') == 1 then
    set(hRbut2, 'value', 0);
    end;
    clf(hFig2);              // очищаем графическое окно
    plot(x, sin(x), 'r');    // построение синусоиды красного цвета
    xgrid( );
endfunction
                    // функция, реагирующая на щелчок по 2-й кнопке
function Radio2 ( )
    if get(hRbut2, 'value') == 1 then
    set(hRbut1, 'value', 0);
    end;
    clf(hFig2);                  // очищаем графическое окно
    plot(x, cos(x), 'g');        // построение косинусоиды зеленого цвета
    xgrid( );
endfunction
```

Сохраните скрипт в папке Lab11 под именем radio.sce. Запустите скрипт на выполнение из окна редактора. Щелкайте мышью по переключателям и наблюдайте за отображением графиков функций. Изменение состояния переключателей происходит автоматически при щелчке по ним. Графические окна программы при различных состояниях переключателей приведены на рисунках 7 и 8.

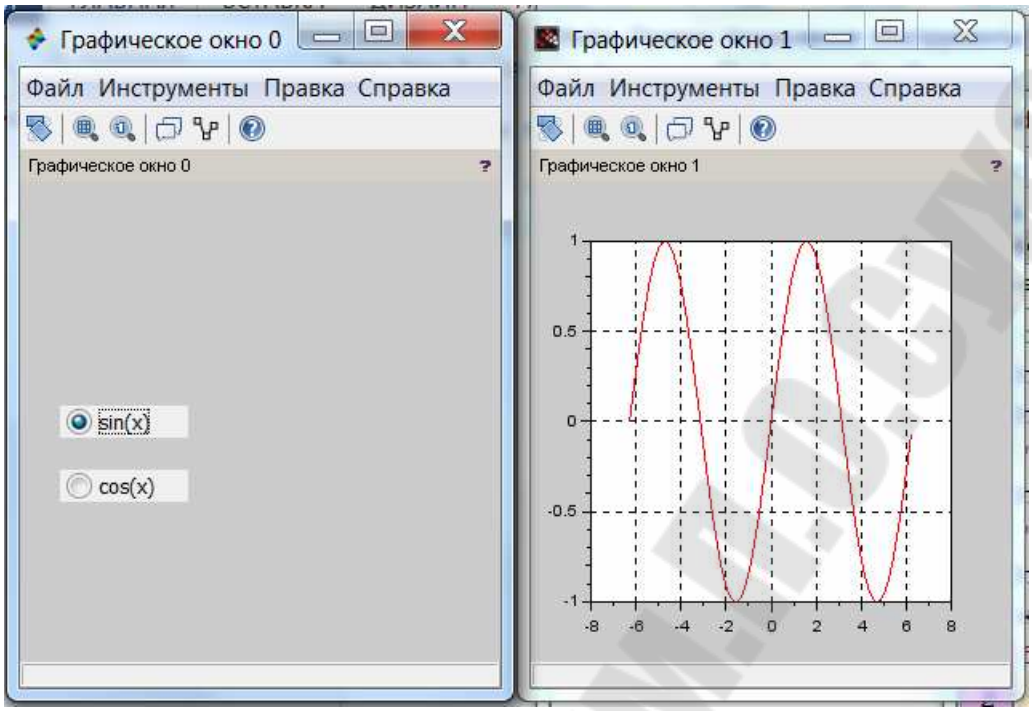


Рисунок 7 – Выбор переключателя для графика sin(x)

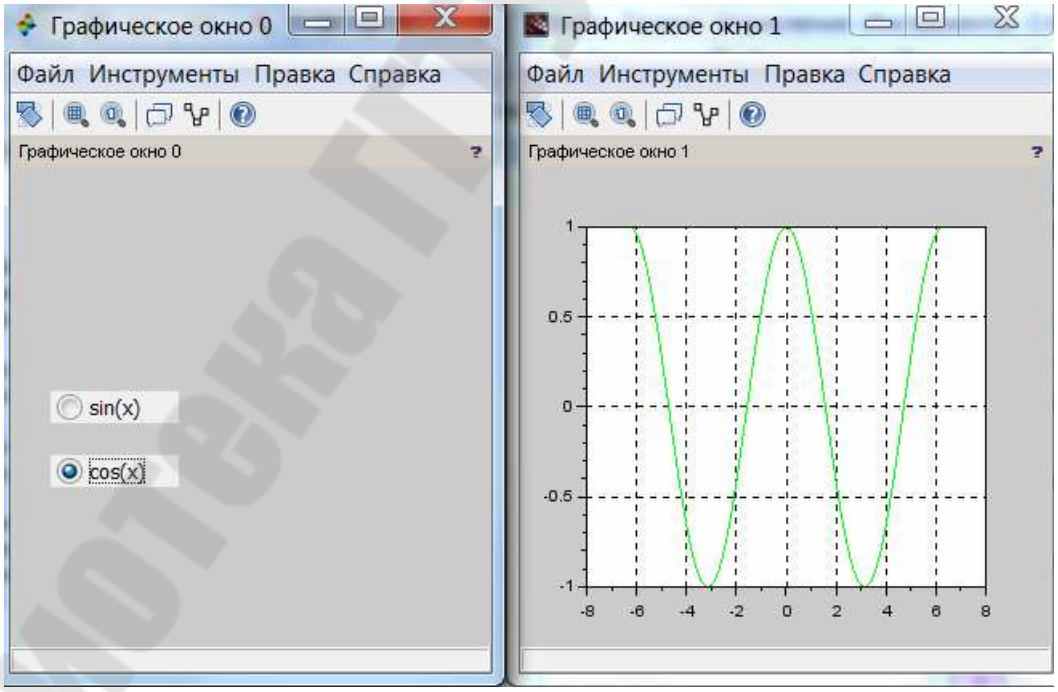


Рисунок 8 – Выбор переключателя для графика cos(x)

В заключение закройте файл radio.sce в окне редактора и графические окна на дисплее.

2.10.3. Теперь рассмотрим программу, в которой с помощью переключателя можно выбрать функцию, график которой будет воспроизводиться при щелчке по кнопке Построить. Кроме того, в программе будет кнопка Выход, с помощью которой происходит закрытие приложения.

Задание. Наберите в окне редактора SciNotes текст следующего скрипта.

```
                    // создаем первое графическое окно
hFig1 = figure('position', [10, 10, 300, 300]);
                    // создание переключателей
hRbut1 = uicontrol('style', 'radiobutton', 'position', [25, 140, 80, 20],
..
    'string', 'sin(t)', 'value',0, 'callback', 'Radio1');
hRbut2 = uicontrol('style', 'radiobutton', 'position', [25, 100, 80, 20],
..
    'string', 'cos(t)', 'value',0, 'callback', 'Radio2');
                    // создание командных кнопок
hPbut1 = uicontrol('style', 'pushbutton', 'position', [50, 50, 150, 20], ..
    'string', 'Построить', 'callback', 'call_plot');
hPbut2 = uicontrol('style', 'pushbutton', 'position', [50, 10, 150, 20], ..
    'string', 'Выход', 'callback', 'finish');
                    // создаем второе графическое окно
hFig2 = figure('position', [400, 10, 300, 300]);
// функция, реагирующая на щелчок по 1-му переключателю
function Radio1( )
    if get(hRbut1, 'value') == 1 then
    set(hRbut2, 'value', 0);
    end;
endfunction
// функция, реагирующая на щелчок по 2-му переключателю
function Radio2( )
    if get(hRbut2, 'value') == 1 then
    set(hRbut1, 'value', 0);
    end;
```

```
endfunction
  // функция построения графиков
function call_plot( )
t = 0 : 0.01 : 10;                          // диапазон изменения
аргумента
if get(hRbut1,'value') ==1 then  // если активен 1-й переключатель
    clf(hFig2);
    plot(t, sin(t),'r');                    // построение синусоиды
    xgrid( );
elseif get(hRbut2, 'value') ==1 then       // если активен 2-й
переключатель
    clf(hFig2);
    plot(t, cos(t), 'g');                   // построение косинусоиды
    xgrid( );
end;
endfunction
                    // функция, отвечающая за кнопку Выход
function finish()
    close(hFig1);                           // закрытие 1-го окна
    close(hFig2);                           // закрытие 2-го окна
endfunction
```

Сохраните скрипт в папке Lab11 под именем radio_2.sce. Запустите скрипт на выполнение из окна редактора. Щелкая по переключателям и командным кнопкам, убедитесь в правильности работы программы.

В заключение закройте файл radio_2.sce в окне редактора.

2.11 Исследование элемента управления «Слайдер»

2.11.1. Элемент управления «Слайдер» (полоса прокрутки) используется для задания некоторой величины путем перемещения «вручную» бегунка скролинга. При создании слайдера с помощью функции uicontrol( ) параметру ‘style’ необходимо присвоить значение ‘slider’.

Задание. Наберите в командном окне следующий текст:

--> hFig = figure(‘position’, [10 10 300 300]);
--> hSlid = uicontrol(‘style’, ‘slider”, ‘position’, [20, 200, 250, 30],…
‘min’, 0, ‘max’, 10, ‘value’, 5, ‘sliderstep’, [0.1 1]);

На дисплее откроется графическое окно с изображением слайдера. Перемещать бегунок можно щелчком мыши по полосе прокрутки или нажатием клавиш “ -> ” или “ <- “ клавиатуры.

Соответствие между положением бегунка и числовым значением свойства ‘value’ слайдера устанавливается следующими параметрами в функции uicontrol( ).

1. Свойства ‘min’ и ‘max’ определяют границы значений, записываемых в ‘value’, при перемещении бегунка. В нашем случае это будет 0 в крайнем левом положении, и 10 – в крайнем правом положении бегунка.

2. Начальное значение и, соответственно, положение бегунка определяется числом, записанном в свойство ‘value’. В нашем случае это число 5 (половина диапазона изменения ‘value’), поэтому бегунок будет находиться в середине полосы прокрутки.

3. Свойство ‘sliderstep’ определяет шаг перемещения бегунка и, соответственно, изменения значения ‘value’. Этот параметр является вектором из двух компонентов. Первый компонент определяет шаг изменения ‘value’ при нажатии на клавиши с гравировкой “ <- “ или “ -> “ клавиатуры, а второй компонент – при щелчке мышью по полосе прокрутки. В нашем случае при установке значений вектора [0.1 1] свойства ‘sliderstep’ нажатие на клавиши “ <- “ или “ -> “ изменит ‘value’ на величину 0.1 (1%), а щелчок мыши слева или справа от бегунка изменит ‘value’ на 1, т. е. на 10% от максимального значения.

Задание. Выполните перемещение бегунка с помощью мыши и клавиатуры.

Для получения текущей координаты бегунка слайдера необходимо выполнить команду:

--> val = get(hSlid, ‘value’)

Результатом будет неизменное число при любом положении бегунка. Причина этого заключается в том, что значение ‘value’ не обновляется автоматически при перемещении бегунка. Для его обновления нужна функция, которая будет вызываться событием ‘callback’, возникающем при перемещении ползунка.

2.11.2. Исследуем программу, в которой на дисплей выводится значение текущей координаты бегунка слайдера.

Задание. Наберите в окне редактора SciNotes текст следующего скрипта.

```
hFig = figure(‘position’, [10, 10, 300, 300]);
hSlid = uicontrol('style', 'slider', 'position', [20, 200, 250, 30],...
    'min', 0, 'max', 10, 'value', 5, 'sliderstep', [0.1 1], 'callback', 'val_slider');
                    // функция получения текущего значения бегунка слайдера
function val_slider( )
    val = get(hSlid, 'value');
    disp('Значение слайдера:');
    disp(val);
endfunction
```

Сохраните скрипт в папке Lab11 под именем slider.sce. Запустите скрипт на выполнение из окна редактора. Созданное программой графическое окно со слайдером приведено на рисунке 9.

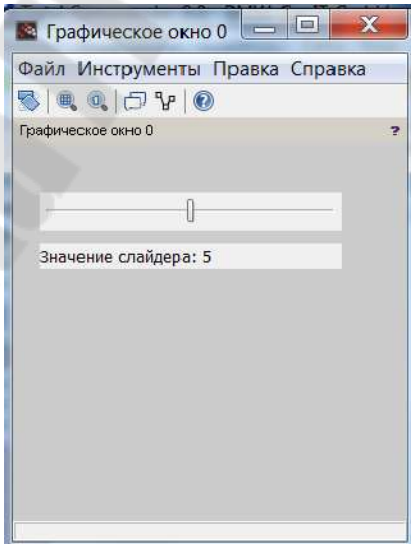


Рисунок 9 – Слайдер с текущей координатой бегунка

Перемещая бегунок слайдера с помощью мыши и клавиатуры, наблюдайте за величиной val в командном окне.

В заключение закройте файл slider.sce в окне редактора и графическое окно на дисплее.

2.11.3. В предыдущей программе числовое значение координаты бегунка не отображается в графическом окне. Рассмотрим программу, в которой этот недостаток устранен. В ней выполнена синхронизация изменения положения бегунка и отображения числового значения координаты.

Задание. Наберите в окне редактора SciNotes текст следующего скрипта.

```
hFig = figure('position', [10, 10, 300, 300]);
hSlid = uicontrol('style', 'slider', 'position', [20, 230, 230, 30], ..
   'min', 0, 'max', 10, 'value', 5, 'sliderstep', [0.1 1], 'callback', 'slider_update');
  // текстовое поле для отображения координаты бегунка с начальным
  // значением, равном 5
hText = uicontrol(‘style’, ‘text’, ‘position’, [20, 200, 230, 20],...
    ‘string’, ‘Значение слайдера: ‘+string(5));
  // функция обновления значения координаты
function slider_update( )
    val = get(hSlid, ‘value’);
    set(hText, ‘string’, ‘Значение слайдера: ‘ + string(val));
endfunction
```

Сохраните скрипт в папке Lab11 под именем slider_update.sce. Запустите скрипт на выполнение из окна редактора. Перемещая бегунок слайдера с помощью мыши и клавиатуры, наблюдайте за численным значением координаты в текстовом поле.

В заключение закройте графическое окно на дисплее и файл slider_update.sce в окне редактора.

2.11.4. Рассмотрим программу, в которой слайдер используется для задания аргумента функции y = sin(x), график которой строится в окне.

Задание. Наберите в окне редактора SciNotes текст следующего скрипта.

```
                        // создание графического окна
hFig1 = figure('position', [10, 10, 300, 300]);
                        // создание слайдера
hSlid = uicontrol('style', 'slider', ‘position’, [20, 200, 250, 30], ..
    'min', 0, 'max', 10, 'value', 5, 'sliderstep', [0.1 1]);
                        // текст заголовка над слайдером
hText= uicontrol('style', 'text', 'position', [20, 250, 250, 20], ..
    'string', 'Установка значения x от 0 до 10');
                        // создание командной кнопки
hPbut = uicontrol('style', 'pushbutton', 'position', [100, 100, 100, 20], ..
    'string', 'Построить', 'callback', 'call_plot');
                        // создание окна для графика
hFig2 = figure('position', [350, 10, 300, 300]);
                        // функция построения графика
function call_plot( )
    n = get(hSlid, 'value');      // получить значение от слайдера
    x0 = round(n);                // округлить полученное значение
    x = -x0 : 0.1 : x0;           // диапазон изменения аргумента
    y = sin(x);
    clf(hFig2);                   // очистить окно для графика
    plot(x,y);                    // построить график
endfunction
```

Сохраните скрипт в папке Lab11 под именем slider_sin.sce. Запустите скрипт на выполнение из окна редактора. Созданные программой графические окна приведены на рисунке 10.

Исследуйте работу программы (построение графика синусоиды) при различных положениях бегунка слайдера. Обратите внимание на изменение координаты горизонтальной оси (величины аргумента x).

В заключение закройте файл slider_sin.sce в окне редактора и графические окна на дисплее.

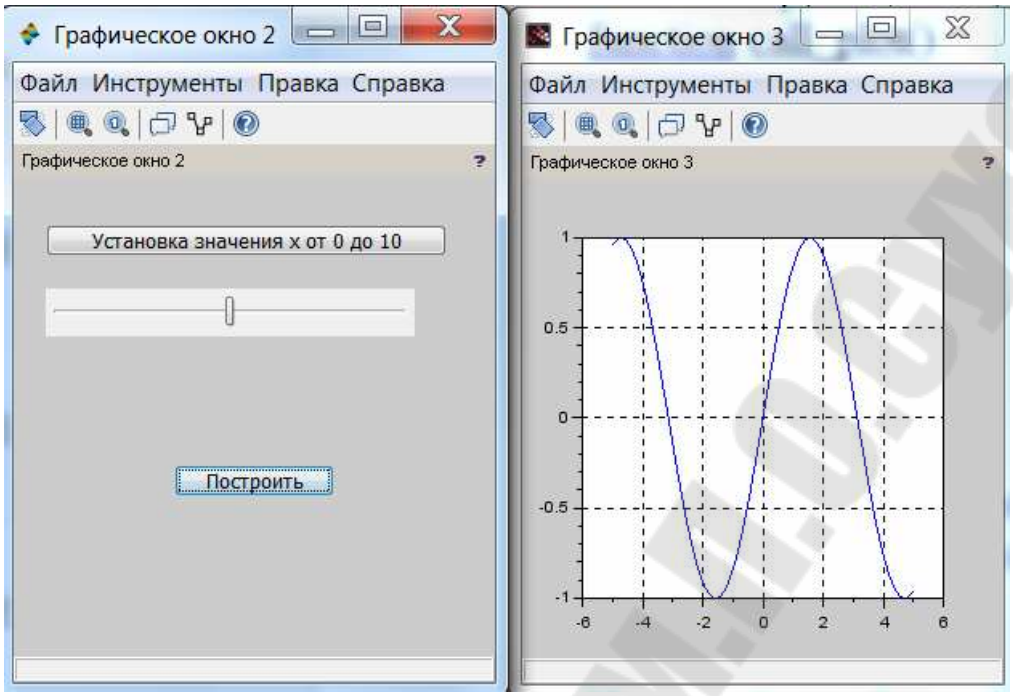


Рисунок 10 – Применение слайдера для задания значений координат оси x графика

3 Содержание отчета

Отчет по лабораторной работе № 11 выполняется в виде списка скрипт-файлов, которые были исследованы при выполнении работы.

Контрольные вопросы

1. Назначение графического интерфейса пользователя (GUI).
2. Объясните назначение функции messagebox( ).
3. Как создается графическое окно в Scilab?
4. Основные работы с интерфейсными компонентами в Scilab.
5. Опишите интерфейсный компонент Pushbutton.
6. Опишите интерфейсный компонент Text.
7. Опишите интерфейсный компонент Edit.
8. Опишите интерфейсный компонент Radiobutton.
9. Опишите интерфейсный компонент Listbox.
10. Опишите интерфейсный компонент Slider.

Лабораторная работа № 12

Разработка графического интерфейса для исследования характеристик систем управления

1 Цель работы

Практически исследовать методы разработки графического интерфейса пользователя для построения временных и частотных характеристик систем автоматического управления.

2 Порядок выполнения работы

2.1 Создание папки для работы

При выполнении лабораторной работы № 12 вы будете использовать папку с именем Lab12, которую необходимо предварительно создать.

С этой целью откройте вашу рабочую папку (с вашей фамилией) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab12.

В дальнейшем вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 12. Полный путь к этой папке будет такой:

F:\OTC\Ivanov\Lab12

2.2 Запуск и настройка среды разработки Scilab

2.2.1. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа – рабочий стол среды разработки Scilab.

2.2.2. Нужно установить в качестве активной (текущей) папку Lab12. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем ОТС, затем папку с вашей фамилией (условно Ivanov) и затем папку

Lab12. В строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab12. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab12

2.3 Исследование программы GUI для ввода передаточной функции САУ

Задание. Наберите в окне редактора SciNotes текст следующего скрипта.

```
clear;  clc;   // очистка рабочего пространства и командного окна
               // создание графического окна
hFig=figure('position', [100,  100,  400,  300],…
         'figure_name', 'Окно ввода параметров');
               // текст заголовка окна редактирования
hText1=uicontrol('style', 'text', 'position', [50, 250, 200, 20], ..
        'string', 'Функция W(s)');
               // окно редактирования
hEdit1=uicontrol('style', 'edit',  'position', [50, 200, 250, 20], 'string', '   ');
               // командная кнопка для ввода параметра
hPBut=uicontrol('style', 'pushbutton',  'position', [50, 150, 100, 20], ..
        'string', 'Ввод',  ‘callback', 'call_enter');
               // функция ввода и вычисления передаточной функции
function   call_enter( )
      s = %s;       // определение символьной переменной
// получение значения передаточной функции из окна редактирования
      W = get(hEdit1, 'string');
               // вывод в командное окно параметра из окна редактирования
      disp('Передаточная функция W(s) равна:');
      disp(W);
      G = evstr(W);        // преобразование строки в числовое значение
      TF = syslin('c', G);   // получение передаточной функции системы
          // определение числителя передаточной функции системы
      num = numer(TF);
      disp('Числитель TF равен:');          // вывод в командное окно
```

```
    disp(num);
        // определение знаменателя передаточной функции системы
    den = denom(TF);
    disp('Знаменатель TF равен:');    // вывод в командное окно
    disp(den);
endfunction
```

Сохраните скрипт в папке Lab12 под именем param_enter.sce. Запустите скрипт на выполнение из окна редактора. Графическое окно, созданное программой, приведено на рисунке 1.

Протестируйте работу программы, введя в окно редактирования передаточную функцию:

s /(s^2 + s + 1)

Убедитеь по выводимым данным в командном окне в правильности работы программы. Редактор SciNotes пока не закрывайте!

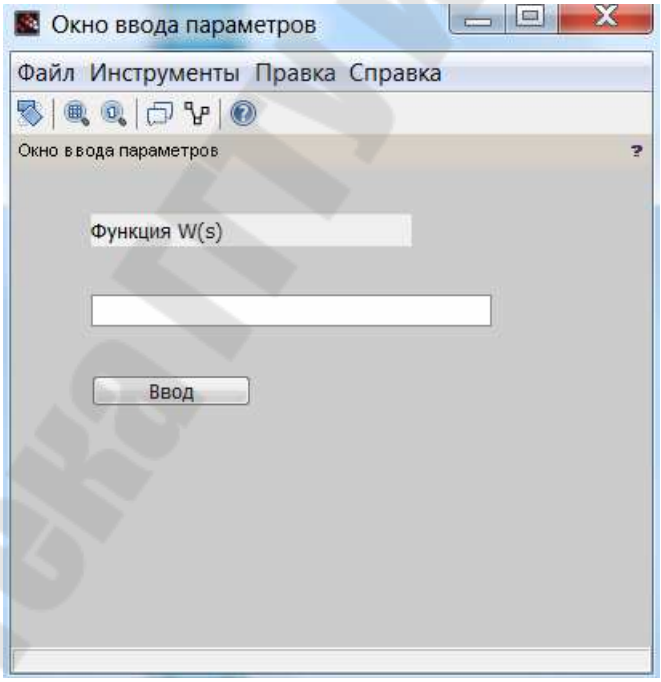


Рисунок 1 – Графическое окно скрипта param_enter.sce

2.4 Локальные и глобальные переменные в функциях

Все имена переменных внутри функции, а также из списка входных и выходных параметров воспринимаются средой Scilab как локальные, т. е. определенными только внутри данной функции. Они

недоступны из других функций или из командного окна. Однако иногда возникает необходимость использовать внутренние переменные из функций, например, чтобы прочитать значение переменной из командного окна. Для этой цели необходимо определить эту переменную внутри функции как глобальную. Например, для внутренней переменной с именем z записать в функции

global z;

Если требуется прочитать значение переменной z из командного окна, то необходимо предварительно определить z как глобальную в рабочем пространстве Scilab. Для этого надо записать в командном окне:

global z;

После выполнения этой команды переменная с именем z будет находиться в рабочем пространстве Scilab, и ее значение можно прочитать или изменить из командного окна.

Задание. Допустим, что требуется прочитать значение переменной с именем TF (передаточной функции системы), вычисленной в функции call_enter( ) события callback предыдущей программы.

С этой целью наберите в командном окне:

--> TF

Программа Scilab выдаст ошибку:

! error 4

Неизвестная переменная TF

Для устранения этой ошибки необходимо объявить переменную TF как глобальную в функции call_enter( ). С этой целью откройте в редакторе SciNotes файл param_enter.sce и запишите в функции call_enter( ), например, перед строкой TF = syslin('c', G) команду:

global TF; // объявление переменной TF как глобальной

Сохраните изменения в редакторе SciNotes. Вновь запустите скрипт на выполнение. Введите в окно редактирования передаточную функцию:

s/(s^2 + s + 1)

и щелкните по кнопке Ввод.

В командном окне наберите:

--> global TF; // объявление переменной TF как глобальной

--> TF // чтение TF

Значение переменной TF должно появиться в командном окне.

Объявление переменных как глобальных позволяет обмениваться данными между различными файлами в сложных программах.

В заключение закройте графическое окно на дисплее и файл скрипта в окне редактора.

2.5 Исследование программы анализа устойчивости САУ

2.5.1. **Задание.** Наберите в окне редактора SciNotes текст следующего скрипта.

```
clear; clc;          // очистка рабочего пространства и командного окна
                  // создание графического окна
hFig = figure('position', [10, 10, 600, 400],  'figure_name', 'Анализ САУ');
// текст заголовка окна редактирования для передаточной
// функции W(s)
hText_param = uicontrol('style', 'text',  'position', [20, 300, 250, 20], ..
        'string', 'Передаточная функция W(s)');
// окно редактирования передаточной функции
hEdit_param=uicontrol('style', 'edit',  'position', [20,  270,  250,  20],...
         'string', '   ');
// вывод сообщения в стандартное окно
messagebox(['Введите значение передаточной функции'
         'в виде:  (s+1)/(s^2+2*s+1)']);
// командная кнопка для ввода параметров
hPbut_vvod = uicontrol('style', 'pushbutton',   'position', [300, 300, 100,
20],...
         'string', 'Ввод',  'callback', 'calc_ust');
// заголовок окна результата расчета устойчивости САУ
hText_ust = uicontrol('style', 'text',  'position', [300, 250, 200, 20],...
         'string', 'Расчет устойчивости САУ');
// окно вывода результата расчета устойчивости САУ
hText_res = uicontrol('style', 'text', 'position', [300, 220, 200, 20], 'string', '
');
```

```
// Командная кнопка для выхода из программы
hPbut_fin = uicontrol('style', 'pushbutton',  'position', [300, 100, 100, 20]
,...
        'string', 'Выход',  'callback', 'finish');
// функция расчета устойчивости системы
function  calc_ust( )
      set(hText_res, 'string', 'Система устойчива');
endfunction
// функция выхода
function  finish()
      close(hFig);
endfunction
```

Сохраните скрипт в папке Lab12 под именем ust_sau.sce. Запустите скрипт на выполнение из окна редактора. Графическое окно, созданное скриптом, приведено на рисунке 2.

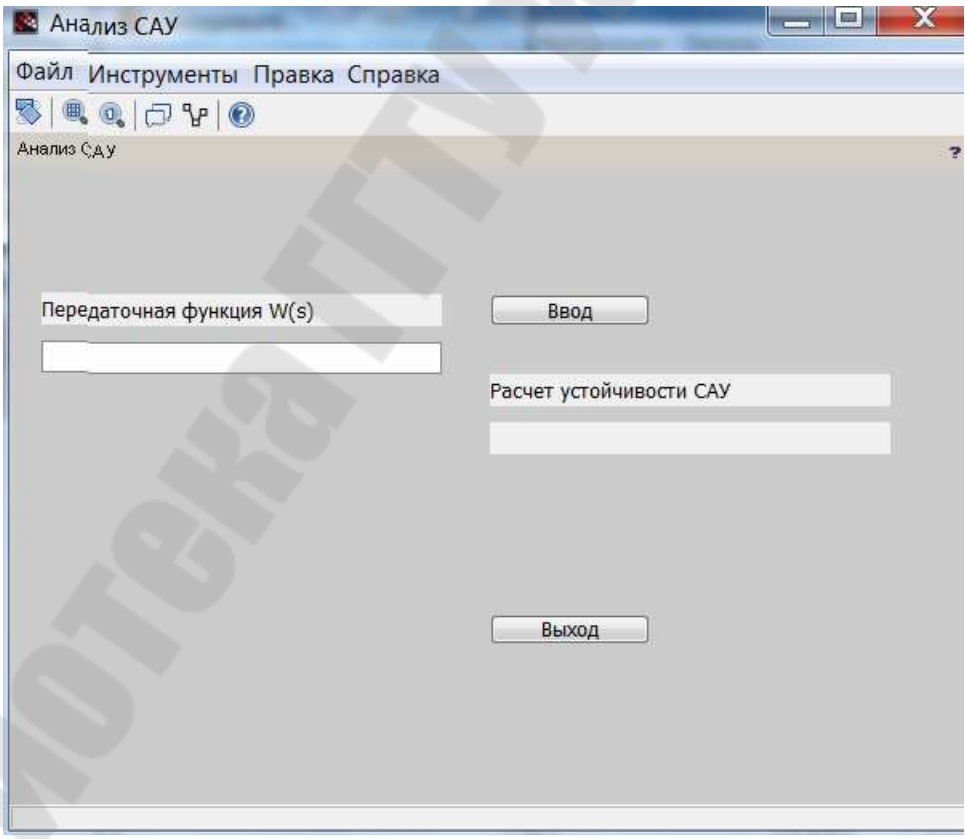


Рисунок 2 – Графическое окно скрипта ust_sau.sce

Протестируйте работу программы при входном параметре – передаточной функции W(s):

s/(s^2 + s +1)

Щелкните по кнопке [Ввод] и наблюдайте за текстом в окне результата. В заключение щелкните по кнопке [Выход].

2.5.2. **Задание 2 для самостоятельной работы.** Функция call_ust( ) в предыдущей программе просто выводит строку «Система устойчива" без анализа САУ.

Необходимо модернизировать функцию calc_ust( ), чтобы она выполняла следующие действия:

1) ввод значения входного параметра – передаточной функции W(s) из окна редактирования;

2) вычисление системной передаточной функции TF – (по аналогии со скрипт-файлом param_enter.sce);

3) расчет устойчивости САУ. Расчет выполняется по методике, изложенной в лабораторной работе № 9. Для определения устойчивости САУ необходимо рассчитать значения корней характеристического уравнения (знаменателя системной передаточной функции TF):

r = roots(den);

Это будет вектор, содержащий корни. Количество корней можно определить с помощью функции:

len = length(r);

Затем в цикле

for i=1 : len

необходимо выяснить, являются ли вещественные части корней отрицательными числами. Для вычисления вещественной части служит функция real( ), например:

real(r(1)

вычисляет вещественную часть корня r(1).

Если вещественные части всех корней отрицательны, то система устойчива. В противном случае – неустойчива. Результат расчета необходимо вывести в окно редактирования результата.

Разработанный файл назовите ust_sau_2.sce.

Запустите скрипт ust_sau_2.sce на выполнение из окна редактора. Проведите тестирование работы программы для двух вариантов передаточных функций:

1)

W(s) = s /(s^2 +s + 1)

2)

W(s) = s /(s^2 +s - 1)

Подсказка. В первом варианте система должна быть устойчивой, а во втором – неустойчивой.

2.6 Работа с текстовыми файлами в Scilab

2.6.1. Интегрированная среда Scilab обеспечивает работу с двумя типами файлов: двоичными и текстовыми. Двоичные файлы предназначены для хранения произвольных данных в виде последовательности байтов. Текстовые файлы логически интерпретируются как набор символьных строк. Основные этапы работы с файлами: открытие файла, операции ввода или вывода, закрытие файла.

Для открытия файла предназначена функция mopen( ), которая имеет вид:

fd = mopen(file, mode)

Здесь file – строка, в которой хранится имя файла; mode – режим работы с файлом. Мы будем использовать только один из двух режимов: ‘r’ – текстовый файл открывается в режиме чтения; ‘w’ – открывается пустой текстовый файл, который предназначен только для записи информации.

Функция mopen( ) возвращает идентификатор (дескриптор) открытого файла fd, с помощью которого функции работы с файлом будут обращаться к нему.

Функция записи в текстовый файл mfprintf( ) имеет вид:

mfprintf(fd, s1, s2)

Здесь fd – дескриптор файла; s1 – строка вывода; s2 – список выводимых переменных. В строке вывода вместо выводимых переменных указывается строка преобразования (по аналогии с функцией mprintf( ) вывода в командное окно).

Функция чтения данных из текстового файла mfscanf( ) имеет следующий вид:

A = mfscanf(fd, s1)

Здесь fd – дескриптор файла; s1 – строка форматов.

Функция mfscanf( ) работает следующим образом: из файла с дескриптором fd считываются в переменную A значения в соответствии с форматом s1.

Функция закрытия файла mclose( ) имеет вид:

mclose(fd)

Рассмотрим работу с текстовыми файлами на примерах записи и чтения строк в виде передаточных функций.

2.6.2. Предположим, что необходимо записать передаточную функцию

W = 1/(0.5*s+1)

в текстовый файл с именем file_w.txt, который будет храниться в папке Lab12. Полный путь к этому файлу будет:

F:\OTC\Ivanov\Lab12\file_w.txt

Запишите в окне редактора SciNotes текст скрипта для записи в файл.

```
fd1=mopen('F:\OTC\Ivanov\Lab12\file_w.txt', 'w');  // открыть файл для
                                                   // записи
str1='1/(0.5*s+1)';                  // строка для записи
mfprintf(fd1, '%s\n', str1);         // запись строки в файл
mclose(fd1);                         // закрыть файл
```

Сохраните скрипт в текущей папке Lab12 под именем wr_w.sce. Запустите скрипт из окна редактора. Чтобы прочитать содержимое файла, щелкните мышью по надписи Lab12 в окне обозревателя файлов. Должно появиться название файла file_w.txt. Щелкните дважды по этому имени. Должно раскрыться окно блокнота с текстом – строкой, представляющей собой передаточную функцию. Убедитесь, что значение функции записано правильно.

В заключение закройте файл скрипта.

Теперь запишите в окне редактора SciNotes текст скрипта для чтения из файла.

```
fd2=mopen('F:\OTC\Ivanov\Lab12\file_w.txt', 'r');// открыть файл
// для чтения
str2=mfscanf(-1, fd2, '%s');   // чтение строки из файла. Параметр  -1
                               // обеспечивает чтение до конца файла
mclose(fd2);                   // закрыть файл
```

Сохраните скрипт в текущей папке Lab12 под именем rd_w.sce. Запустите скрипт из окна редактора. Чтобы прочитать содержимое строки str2, наберите в командном окне:

--> str2

Убедитесь, что значение передаточной функции прочитано правильно.

2.6.3. Предположим, что необходимо записать четыре передаточные функции:

W1 = 1/(s^2+s+1)

W2 = 1/(s+1)

W3 = s/(s^2+4*s+5)

W4 = 4/(1 + 0*s)

в текстовый файл с именем file_4w.txt, который будет храниться в папке Lab12. Полный путь к этому файлу будет:

F:\OTC\Ivanov\Lab12\file_4w.txt

Запишите в окне редактора SciNotes текст скрипта для записи в файл.

```
fd3=mopen('F:\OTC\Ivanov\Lab12\file_4w.txt', 'w');// открыть файл для
  // записи
WX1=['1/(s^2+s+1)';  '1/(s+1)';  's/(s^2+4*s+5)';  '4/(1+0*s)'];// вектор
// из четырех строк для записи
mfprintf(fd3, '%s\n', WX1);          // запись вектора в файл
mclose(fd3);                         // закрыть файл
```

Сохраните скрипт в текущей папке Lab12 под именем wr_4w.sce. Запустите скрипт из окна редактора. Чтобы прочитать содержимое файла, щелкните мышью по надписи Lab12 в окне обозревателя файлов. Должно появиться название файла file_4w.txt. Щелкните дважды по этому имени. Должно раскрыться окно блокнота с текстом – тремя строками, представляющими собой передаточные функции. Убедитесь, что значения функций записано правильно.

В заключение закройте файл скрипта.

Теперь запишите в окне редактора SciNotes текст скрипта для чтения из файла.

```
fd4=mopen('F:\OTC\Ivanov\Lab12\file_4w.txt', 'r');// открыть файл для
                                                  // чтения
WX2=mfscanf(-1, fd4, '%s'); // чтение вектора из файла. Параметр -1
                            // обеспечивает чтение до конца файла
mclose(fd4);                // закрыть файл
```

Сохраните скрипт в текущей папке Lab12 под именем rd_4w.sce. Запустите скрипт из окна редактора. Чтобы прочитать содержимое вектора WX2, наберите в командном окне:

```
--> W1 = WX2(1)
--> W2 = WX2(2)
--> W3 = WX2(3)
--> W4 = WX2(4)
```

Убедитесь, что все четыре передаточные функции прочитаны правильно.

2.7 Задание 3 для самостоятельной работы

Создайте скрипт для GUI, выводящего на дисплей графическое окно по рисунку 3. В окне пустой прямоугольник вверху слева – это окно редактирования (ввода пути для текстового файла, содержащего заданную передаточную функцию). Ниже расположено текстовое поле, в которое выводится передаточная функция, прочитанная из файла. При нажатии на кнопку Ввод программа вводит из файла передаточную функцию, вычисляет системную передаточную функцию, затем производит расчет устойчивости системы. Результат расчета (система устойчива или неустойчива) выводится в текстовое поле, расположенное справа. При нажатии на кнопку Выход производится закрытие графического окна.

В программе необходимо предусмотреть вывод окна сообщения после запуска скрипта. Текст сообщения:

Введите полный путь к файлу в виде:

F:\OTC\Ivanov\Lab12\file_w.txt

Разработанный скрипт сохраните в папке Lab12 под именем, например, ust_sau_3.sce. Перед запуском скрипта нужно убедиться,

что текстовый файл, указанный в скрипте, существует, и в нем записана строка с передаточной функцией.

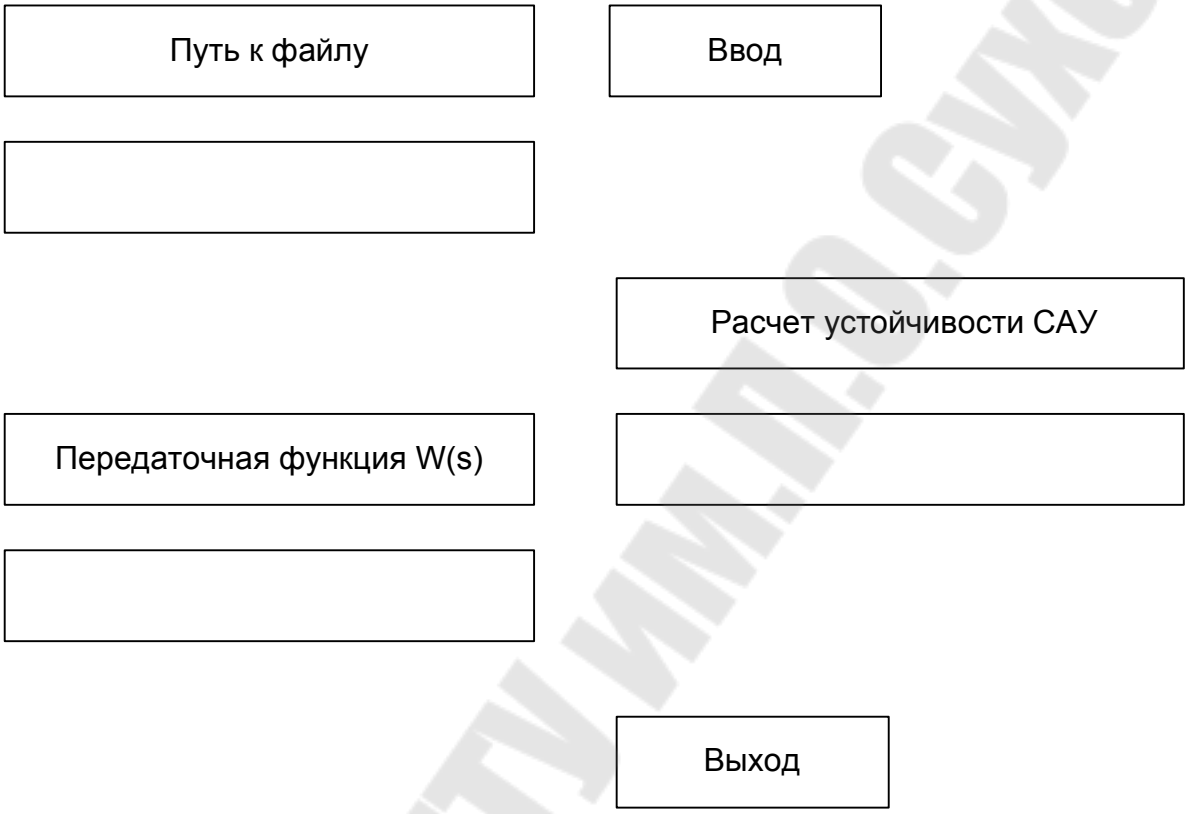


Рисунок 3 – Графическое окно к заданию 3 для самостоятельной работы

Выполните тестирование разработанного скрипта.

2.8 Задание 4 для самостоятельной работы

Это задание делают студенты, у которых в индивидуальном задании для курсового проекта по ОТС определен ввод передаточных функций системы «вручную», т.е. набором с клавиатуры.

В окне четыре пустых прямоугольников слева – это окна редактирования (ввода передаточных функций, определенных по заданию на курсовой проект). При нажатии на кнопку Ввод программа вводит из окон редактирования все передаточные функции, выполняет структурное преобразование, вычисляет

системную передаточную функцию, затем производит расчет устойчивости системы. Результат расчета (система устойчива или неустойчива) выводится в текстовое поле, расположенное справа. При нажатии на кнопку Выход производится закрытие графического окна.

В программе необходимо предусмотреть вывод окна сообщения после запуска скрипта. Текст сообщения:

Введите значение передаточной функции
в виде: (s+1)/(s^2+2*s+1)

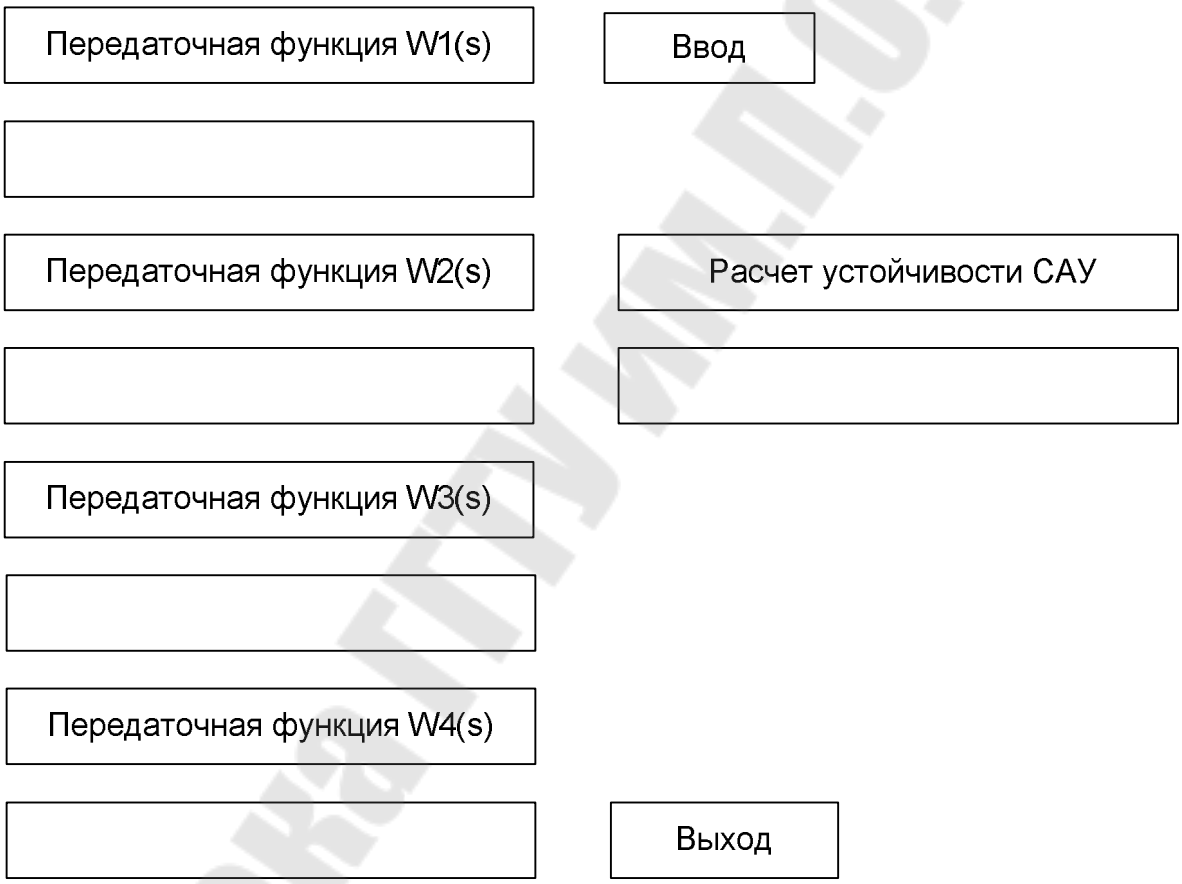


Рисунок 4 – Графическое окно к заданию 4 для самостоятельной работы

Разработанный скрипт сохраните в папке Lab12 под именем, например, ust_sau_4.sce. Запустите скрипт из окна редактора. Введите передаточные функции, определенные в задании на курсовой проект, и проанализируйте работу программы.

2.9 Задание 5 для самостоятельной работы

Это задание делают студенты, у которых в индивидуальном задании для курсового проекта по ОТС определен ввод передаточных функций системы из файла.

В окне пустой прямоугольник вверху слева – это окно редактирования (ввода пути для текстового файла, содержащего заданную передаточную функцию). Ниже расположены четыре текстовых поля, в которые выводятся передаточные функции, прочитанные из файла. При нажатии на кнопку Ввод программа вводит из файла передаточные функции, выполняет структурное преобразование, вычисляет системную передаточную функцию, затем производит расчет устойчивости системы. Результат расчета (система устойчива или неустойчива) выводится в текстовое поле, расположенное справа. При нажатии на кнопку Выход производится закрытие графического окна.

В программе необходимо предусмотреть вывод окна сообщения после запуска скрипта. Текст сообщения:

Введите полный путь к файлу в виде:

F:\OTC\Ivanov\Lab12\file_4w.txt

Разработанный скрипт сохраните в папке Lab12 под именем, например, ust_sau_5.sce. Перед запуском скрипта нужно записать в текстовый файл, указанный в скрипте, строки с четырьмя передаточными функциям, определенными в задании на курсовой проект. Для записи используйте скрипт, рассмотренный в п. 2.6.3.

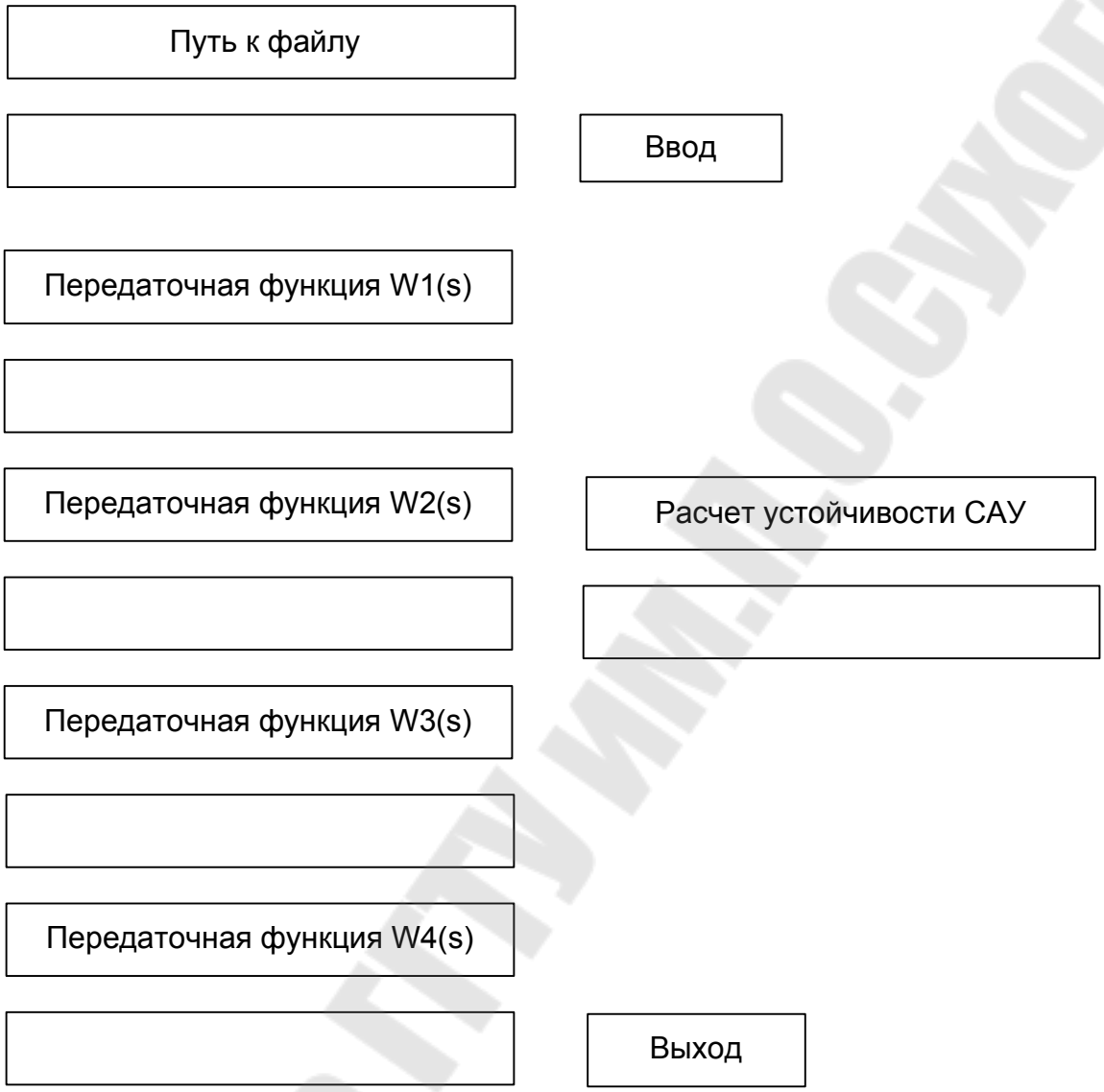


Рисунок 5 – Графическое окно к заданию 5 для самостоятельной работы

3 Содержание отчета

Отчет по лабораторной работе № 12 выполняется в виде списка скрипт-файлов, которые были разработаны при выполнении заданий для самостоятельной работы.

Контрольные вопросы

1. Объясните термин «характеристическое уравнение системы управления».

2. Как определяется устойчивость системы по корням характеристического управления?

3. Что такое локальные и глобальные переменные?

4. Как можно обмениваться данными между различными функциями в программах?

5. Какие функции используются в Scilab для работы с текстовыми файлами?

Лабораторная работа № 13

РАЗРАБОТКА ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ДЛЯ ИССЛЕДОВАНИЯ СИСТЕМ УПРАВЛЕНИЯ

1 Цель работы

Практически исследовать методы разработки графического интерфейса пользователя для расчета и анализа систем автоматического управления.

2 Порядок выполнения работы

2.1 Создание папки для работы

При выполнении лабораторной работы № 13 вы будете использовать папку с именем Lab13, которую необходимо предварительно создать.

С этой целью откройте вашу рабочую папку (с вашей фамилией) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab13.

В дальнейшем вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 13. Полный путь к этой папке будет такой:

F:\OTC\Ivanov\Lab13

2.2 Запуск и настройка среды разработки Scilab

2.2.1. После запуска Scilab необходимо разместить на экране дисплея слева текст данной лабораторной работы, а справа – рабочий стол среды разработки Scilab.

2.2.2. Нужно установить в качестве активной (текущей) папку Lab13. С этой целью подведите курсор мыши к иконке слева от строки со списком папок. Появится подсказка «Выберите папку». Щелкните кнопкой мыши по иконке. Раскроется окно с заголовком «Выберите папку». Сначала выберите диск F, затем папку с именем ОТС, затем папку с вашей фамилией (условно Ivanov) и затем папку

Lab13. В строке Folder name должно быть следующее: F:\OTC\Ivanov\Lab13. Если это так, то щелкните по кнопке Open.

В результате этих действий в строке обозревателя файлов должно появиться:

F:\OTC\Ivanov\Lab13

2.3 Исследование программы построения временных характеристик САУ

Рассмотрим программу, которая строит временные характеристики САУ – переходную h(t) и импульсно-переходную g(t).

Задание. Наберите в окне редактора SciNotes текст следующего скрипта.

```
// ввод передаточной функции из командного окна
W = input('Введите передаточную функцию W = ');
// создание системной передаточной функции
s = %s;
TF = syslin('c', W);
global TF;                                  // объявление переменной TF как глобальной
// создание графического окна
hFig1 = figure('position', [10, 10, 300, 300]);
// текст заголовка для слайдера
hText1= uicontrol('style', 'text',  'position', [20, 260, 230, 20], ..
  'string', 'Установка времени от 0 до 10 секунд');
// создание слайдера
hSlid = uicontrol('style', 'slider',  'position', [20, 220, 230, 30], ..
  'min', 0,  'max', 10,  'value', 5,  'sliderstep', [0.1  1]);
// текст заголовка для переключателей
hText2 = uicontrol('style', 'text',  'position', [20, 180, 200, 20], ..
  'string', 'Выбор временной функции');
// создание переключателей
hRbut1= uicontrol('style', 'radiobutton',  'position', [25, 140, 80, 20], ..
  'string', 'h(t)',  'value', 0,  'callback', 'Radio1');
```

```
        hRbut2 = uicontrol('style', 'radiobutton',  'position', [25, 100, 80, 20],
..
          'string', 'w(t)',  'value', 0,  'callback', 'Radio2');
                              // создание командной кнопки Построить
        hPbut1 = uicontrol('style', 'pushbutton',  'position', [50, 50, 100, 20],
..
          'string', 'Построить',   'callback', 'grafic_t');
                              // создание командной кнопки Выход
        hPbut2 = uicontrol('style', 'pushbutton',  'position', [50, 10, 100, 20],
..
          'string', 'Выход',  'callback', 'finish');
                              // создание окна для вывода графиков
        hFig2 = figure('position', [400, 10, 300, 300]);
                              // функция реакции на щелчок по 1-му
переключателю
        function  Radio1( )
        if  get(hRbut1, 'value') ==1 then
           set(hRbut2, 'value', 0);
        end;
        endfunction
                              // функция реакции на щелчок по 2-му
переключателю
        function  Radio2( )
        if  get(hRbut2, 'value') ==1 then
           set(hRbut1, 'value', 0);
        end;
        endfunction
                              // функция построения графиков
        function  graphic_t( )
        global  TF;

        time = get(hSlid, 'value');          // получение значения от слайдера
        t0 = round(time);                              // округление полученного
значения
        t = 0 : 0.01 : t0;                   // диапазон изменения аргумента
        if  get(hRbut1, 'value') ==1 then // если активен 1-й переключатель
           clf(hFig2);                                 // очистить графическое окно
           h = csim('step', t, TF);  // вычислить переходную функцию
```

```
        plot(t, h);                          // построить график функции
h(t)
        xgrid( );
           elseif  get(hRbut2, 'value') ==1 then  // если активен 2-й
переключатель
        clf(hFig2);
        g = csim('impulse', t, TF);   // вычислить импульсно-
переходную
                                              // функцию
        plot(t, g);                          // построить график функции
g(t)
        xgrid( );
      end;
      endfunction
                        // функция закрытия приложения
      function  finish( )
          close(hFig1);
          close(hFig2);
      endfunction
```

Сохраните скрипт в папке Lab13 под именем t_grafic.sce. Запустите скрипт на выполнение из окна редактора.

Введите в командном окне передаточную функцию в виде:

(100 + 9*s) / (100 + 29*s + 12*s^2 + s^3)

На рисунке 1 приведены графические окна, созданные программой t_grafic.sce.

Выполните исследование работы программы. С этой целью щелкните мышью по значку переключателя с надписью h(t) – выбор переходной функции. Затем щелкните по кнопке [Построить]. Должен появиться график переходной характеристики (функции) h(t), как показано на рисунке 2. Исследуйте влияние слайдера на временной диапазон графика. С этой целью переместите бегунок слайдера на некоторое расстояние и вновь щелкните по кнопке [Построить]. На графике переходной характеристики h(t) должен измениться диапазон времени в соответствии с новым положением бегунка слайдера.

Далее щелкните мышью по значку переключателя с надписью g(t) – выбор импульсно-переходной функции. Затем щелкните по

кнопке [Построить]. Должен появиться график импульсно-переходной характеристики (функции) g(t). Исследуйте влияние слайдера на временной диапазон графика.

В заключение щелкните по кнопке [Выход] для закрытия графических окон и завершения работы программы. Также закройте файл t_grafic.sce в окне редактора.

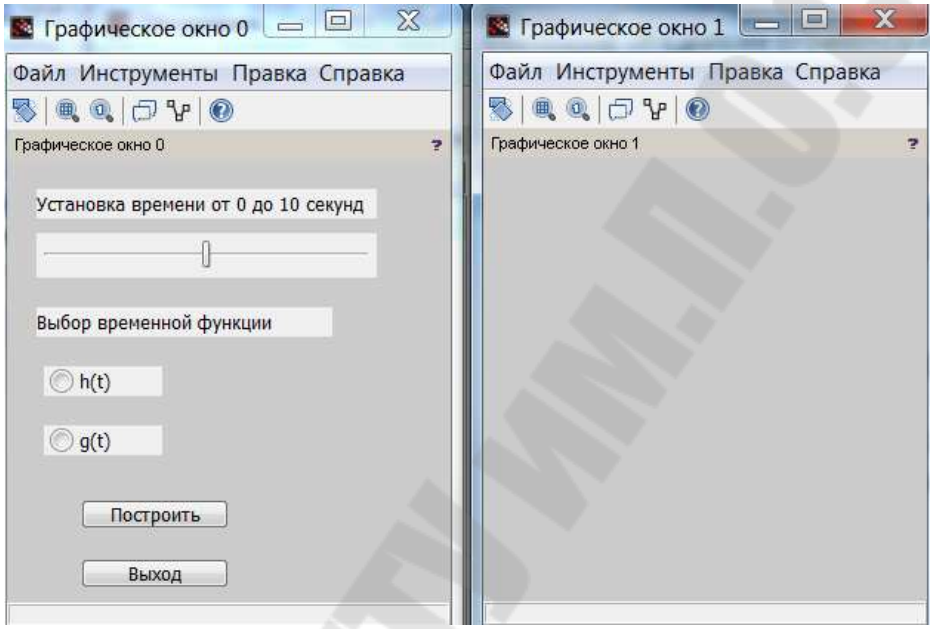


Рисунок 1 – Графические окна скрипта t_grafic.sce

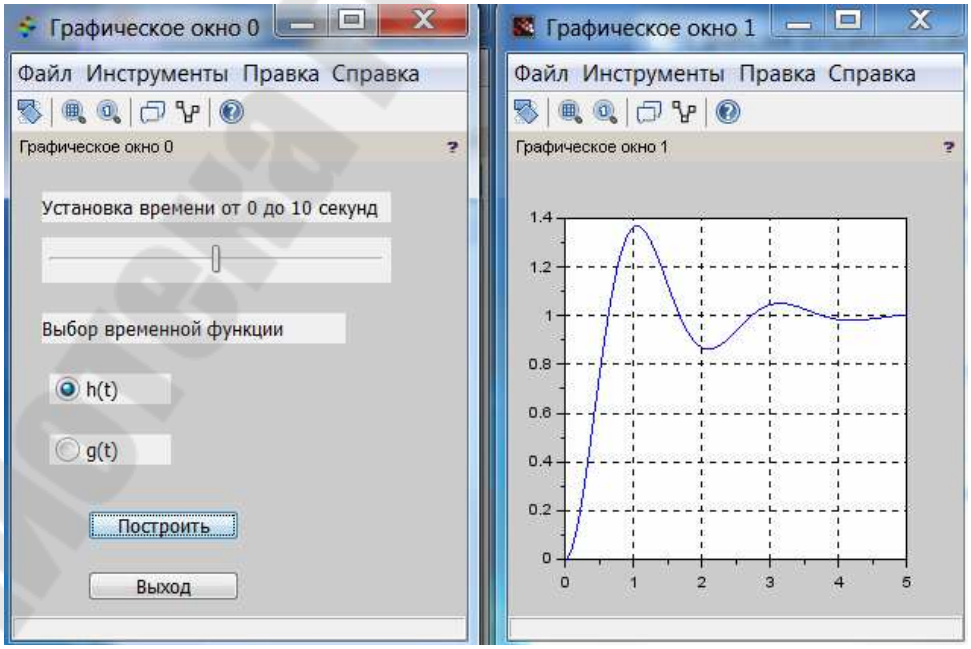


Рисунок 2 – Результаты выбора временной функции h(t)

2.4 Исследование программы построения частотных характеристик САУ

Задание. Наберите в окне редактора SciNotes текст следующего скрипта.

```
                    // ввод передаточной функции из командного
окна
    W = input('Введите передаточную функцию W = ');
            // создание системной передаточной функции
    s = %s;
    TF = syslin('c', W);
    global  TF;                 // объявление переменной TF как
глобальной
            // создание графического окна
    hFig1= figure('position', [10, 10, 300, 300]);
            // заголовок для окна списка
    hText1 = uicontrol('style', 'text',  'position', [20, 270, 250, 20], ..
    'string', 'Выбор частотной характеристики');
            // создание компонента Список из четырех строк
    hLbox = uicontrol('style', 'listbox',  'position', [20, 140, 150, 100], ..
    'string', 'Диаграмма Боде | Диаграмма  Найквиста | ЛАЧХ | ФЧХ',
..
    'value', 0);
                    // создание командной кнопки Построить
    hPbut1 = uicontrol('style', 'pushbutton',  'position', [50, 100, 100, 20],
..
    'string', 'Построить',  'callback', 'grafic_f');
                    // создание командной кнопки Выход
    hPbut2 = uicontrol('style', 'pushbutton',  'position', [50, 50, 100, 20],
..
    'string', 'Выход',  'callback', 'finish');
                    // создание окна для графиков функций
    hFig2 = figure('position', [400, 10, 300, 300]);
                    // функция построения графиков
    function   graphic_f( )
    global TF;          // объявление переменной TF как глобальной
    n = get(hLbox, 'value'); // получение зн-я пар-а ‘value’ из списка
```

```
                    // оператор выбора по значению  n
        select  n
    case  0          // если не выбрана строка из списка
       return   // возврат из функции без каких-либо действий
    case  1     // если выбрана 1-я строка  (Диаграмма Боде)
       clf(hFig2);                    // очистить графическое окно
       bode(TF);                      // построение диаграммы Боде
    case  2   // если выбрана 2-я строка (Диаграмма Найквиста)
       clf(hFig2);
       nyquist(TF);             // построение диаграммы Найквиста
    case  3          // если выбрана 3-я строка (ЛАЧХ)
      clf(hFig2);
       gainplot(TF);          // построение графика ЛАЧХ
    case  4          // если выбрана 4-я строка (ФЧХ)
     clf(hFig2);
     phaseplot(TF);          // построение графика ФЧХ
        end;
endfunction
// функция закрытия окон приложения (выхода из программы)
function  finish( )
 close(hFig1);
 close(hFig2);
endfunction
```

Сохраните скрипт в папке Lab13 под именем f_grafic.sce. Запустите скрипт на выполнение из окна редактора. Введите в командном окне передаточную функцию в виде:

```
(100 + 9*s) / (100 + 29*s + 12*s^2 + s^3)
```

На рисунке 3 приведены графические окна, созданные программой f_grafic.sce.

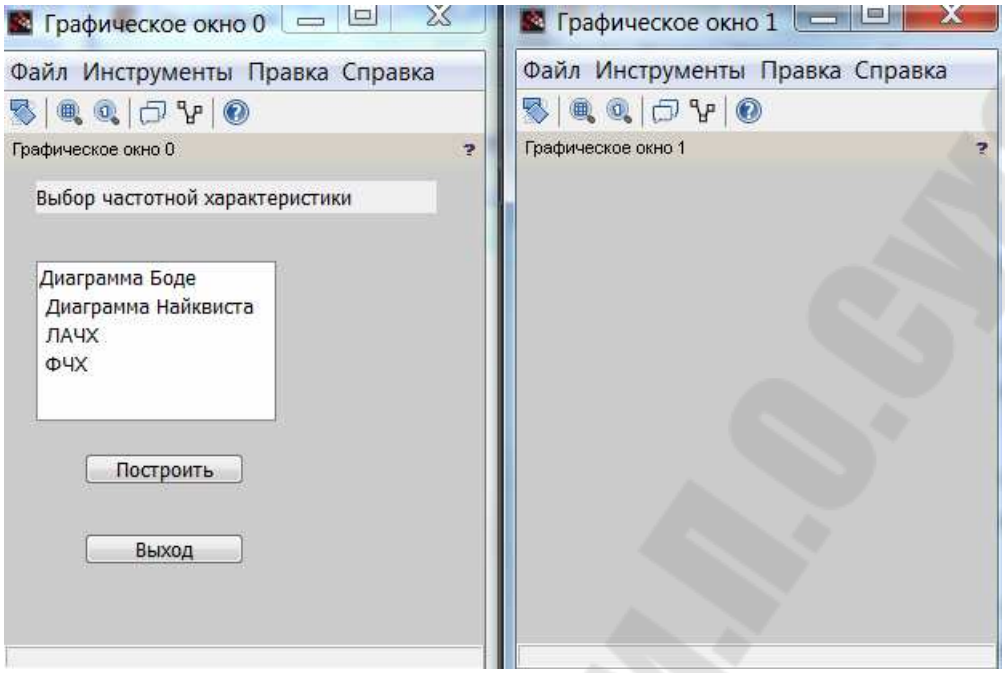


Рисунок 3 – Графические окна программы f_grafic.sce

Выполните исследование работы программы. С этой целью щелкните мышью по строке с названием частотной характеристики, например, «Диаграмма Боде», а затем по кнопке [Построить]. Должен появиться график выбранной частотной характеристики, как показано на рисунке 4. Протестируйте работу программы для всех частотных характеристик из списка.

В заключение щелкните по кнопке [Выход] для закрытия графических окон и завершения работы программы. Закройте также файл скрипта в окне редактора.

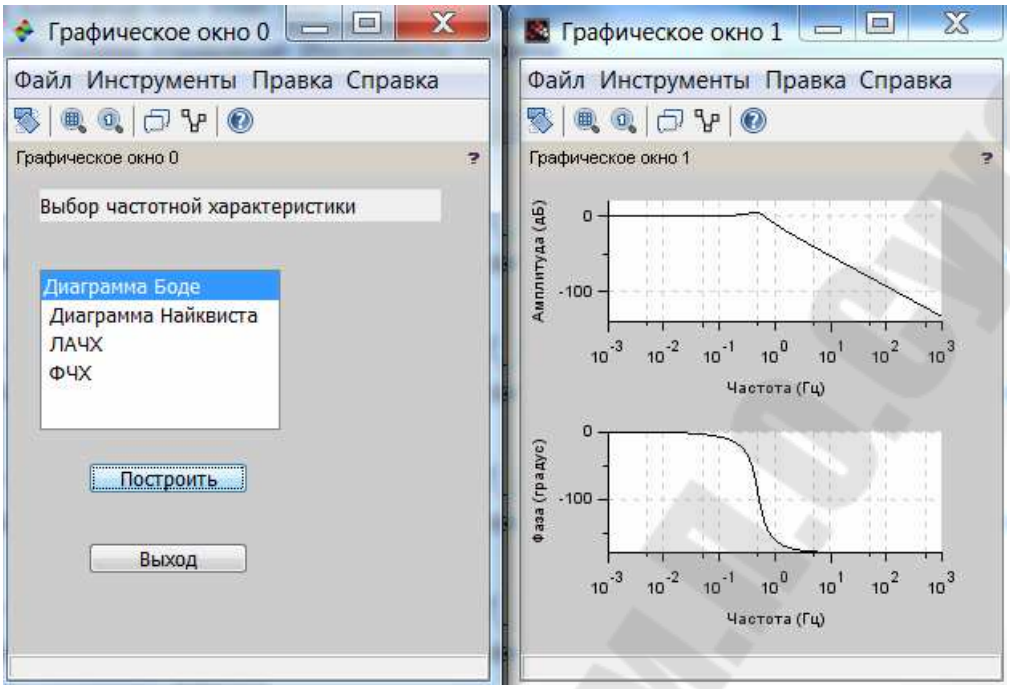


Рисунок 4 – Результаты работы программы f_grafic.sce

2.5 Методика компоновки элементов графического интерфейса для исследования системы автоматического управления

Рассмотрим методику компоновки элементов графического интерфейса для исследования системы автоматического управления в соответствии с заданием на курсовой проект по дисциплине «Основы теории систем».

В лабораторной работе № 12 в заданиях для самостоятельной работы необходимо было разработать скрипт-файлы для ввода исходных данных (передаточных функций отдельных звеньев САУ) и расчета устойчивости САУ для вариантов:

1) ввод исходных передаточных функций звеньев «вручную» с клавиатуры – скрипт-файл с именем ust_sau_4.sce;

2) ввод исходных передаточных функций из текстового файла file_4w.txt – скрипт-файл с именем ust_sau_5.sce.

Для выполнения компоновки элементов интерфейса нужно скопировать эти файлы в папку с именем Lab13.

Рассмотрим последовательность компоновки элементов графического интерфейса для варианта ввода исходных передаточных функций «вручную» с клавиатуры компьютера (для

варианта ввода из текстового файла последовательность компоновки будет аналогична).

2.5.1. Запустите текстовый редактор SciNotes и откройте скрипт-файл ust_sau_4.sce. Сразу же сохраните его под новым именем, например, sau.sce, выполнив команду: Файл => Сохранить как…

Необходимо увеличить размеры основного графического окна по вертикали. Для этого откорректируйте строку, например, так:

hFig1 = figure('position', [10, 10, 600, 700]);

Запустите скорректированный скрипт из окна редактора и проанализируйте графическое окно. Так как координаты всех компонентов отсчитываются от нижнего левого угла графического окна, то необходимо поднять все элементы вверх. Допустим, что нужно поднять на 300 пикселей. С этой целью измените второй параметр (расстояние по вертикальной оси) в свойстве 'position' для всех компонентов графического окна (дескриптор hFig1) на величину 300. Вновь запустите скорректированный скрипт из окна редактора. Если расположение компонентов будет соответствовать требованиям интерфейса, то щелкните по кнопке [Выход] для закрытия приложения.

Далее нужно удалить из графического окна кнопку с надписью [Выход] и функцию, которая вызывается при ее нажатии. Для удаления этих компонентов лучше «закомментировать» соответствующие строки программы:

```
//hPbut2 = uicontrol('style','pushbutton', …….
                //     ….                    'callback', 'finish');
//function  finish( )
                //     close(hFig1);
//endfunction
```

Затем следует сохранить файл sau.sce. и закрыть его.

2.5.2. Откройте в редакторе скрипт-файл t_grafic.sce для построения временных характеристик САУ. Нам требуется получить его копию в буфере обмена. С этой целью выполните следующие команды:

1) Правка => Выделить все;
2) Правка => Копировать;
3) Файл => Закрыть.

2.5.3. Откройте в редакторе SciNotes скрипт sau.sce. Щелкните мышью в конце файла, должен появиться символ курсора. Затем вставьте копию файла t_grafic.sce из буфера обмена, выполнив команду Правка => Вставить. Сохраните полученный файл.

Необходимо выполнить коррекцию в новом варианте файла sau.sce.

1) Надо удалить несколько строк программы в добавленном файле t_grafic.sce. Эти строки располагаются между строками:

endfunction

……

hText1 = uicontrol(……);

Должны остаться только две строки

s = %s;

global TF;

Сама системная функция TF была определена ранее при нажатии кнопки [Ввод].

2) У всех графических компонентов, находящихся в программе после строки endfunction, нужно изменить имена дескрипторов, чтобы не было повторяющихся имен. Проще всего это сделать добавлением к именам суффикса _t, например:

hText1_t = ……..

hSlid_t = ……

…….

hPbut2_t = ……

Необходимо также сделать коррекцию имен дескрипторов в функциях Radio1( ) и Radio2( ).

3) Надо откорректировать позицию и размеры графического окна для вывода графиков временных функций САУ, расположив это окно вверху справа от основного окна, например

hFig2 = figure('position', [650, 10, 400, 500]);

После коррекции элементов интерфейса надо сохранить файл sau.sce и запустить его на выполнение из окна редактора без ввода исходных передаточных функций. Если расположение компонентов и графических окон будет соответствовать поставленной задачи, щелкните по кнопке [Выход] для закрытия окон и завершения приложения.

Далее нужно удалить из графического окна кнопку с надписью [Выход] и функцию, которая вызывается при ее нажатии. Для

удаления этих компонентов лучше «закомментировать» соответствующие строки программы:

//hPbut2_t = uicontrol('style','pushbutton', …….

// …. 'callback', 'finish');

//function finish( )

// close(hFig1);

// close(hFig2;

//endfunction

Сохраните полученный файл, а затем закройте его.

2.5.4. В редакторе SciNotes откройте файл f_grafic.sce построения частотных характеристик САУ. Затем выполните команды редактора:

Правка => Выделить все

Правка => Копировать

Файл => Закрыть

2.5.5. В окне редактора откройте файл sau.sce. В конец его вставьте файл из буфера обмена командой: Правка => Вставить. Затем надо сохранить полученный файл sau.sce.

Необходимо выполнить коррекцию в новом варианте файла sau.sce.

1) Надо удалить несколько строк программы в добавленном файле f_grafic.sce. Эти строки располагаются между строками:

endfunction

……

hText1 = uicontrol(……);

Должны остаться только две строки

s = %s;

global TF;

Сама системная функция TF была определена ранее при нажатии кнопки [Ввод].

2) У всех графических компонентов, находящихся в программе после строки endfunction, нужно изменить имена дескрипторов, чтобы не было повторяющихся имен. Проще всего это сделать добавлением к именам суффикса _f, например:

hText1_f = ……..

hLbox_f = ……

…….

hPbut2_f = ……

Необходимо также сделать коррекцию имен дескрипторов в функции grafic_f( ).

3) Нужно сделать коррекцию в аргументах функций uicontrol( ), указав первым аргументом имя дескриптора главного графического окна hFig1:

```
hText1_f = uicontrol(hFig1, ‘style’, ….);
hLbox_f = uicontrol(hFig1, ‘style’, ….);
……
hPbut2_f = uicontrol(hFig1, ‘style’, ….);
```

4) Надо откорректировать имя, позицию и размеры графического окна для вывода графиков частотных характеристик САУ, расположив это окно вверху справа от окна для вывода графиков временных характеристик, например

```
hFig3 = figure(‘position’, [1100, 10, 400, 500]);
```

После этого надо записать новое имя дескриптора hFig3 в соответствующие строки функции grafic_f( ).

5) Необходимо сместить все графические компоненты скрипта f_grafic.sce в правую часть окна hFig1, чтобы они не накладывались на компоненты области построения временных характеристик. Допустим, что надо сместить все компоненты на 300 пикселей вправо в окне hFig1. Надо увеличить на 300 значение первой координаты (расстояния от левого края графического окна) в значении свойства ‘position’ у всех компонентов:

```
hText_f = uicontrol(hFig1, ‘style’, ‘text’, ‘position’, [320, 270, 250, 20], …);
hLbox_f = uicontrol(hFig1, ‘style’, ‘listbox’, ‘position’, [320, 140, 150, 100], …);
```

и так далее.

6) Надо добавить функцию закрытия close( ) графического окна hFig3 в функцию finish( ) в конце файла:

```
function finish( )
    close(hFig1);
    close(hFig2);
    close(hFig3);
endfunction
```

После коррекции всех элементов интерфейса надо сохранить файл sau.sce и запустить его на выполнение из окна редактора без

ввода исходных передаточных функций. Если расположение компонентов и графических окон будет соответствовать поставленной задачи, щелкните по кнопке [Выход] для закрытия окон и завершения приложения.

2.5.6. Теперь надо исследовать работу разработанного скрипта sau.sce с исходными передаточными функциями. Введите передаточные функции W1(s) – W4(s), заданные в курсовом проекте, в соответствующие окна редактирования. Затем щелкните по кнопке [Ввод]. Должен появиться результат расчета устойчивости САУ.

После этого исследуйте работу программы построения временных характеристик САУ. Постройте переходную характеристику h(t). Исследуйте влияние положения бегунка слайдера на величину времени t графика. Переходный процесс должен полностью закончится (для устойчивых САУ) при максимальном значении времени, задаваемом слайдером. При необходимости требуется откорректировать параметры слайдера (значения свойств ‘min’ и ‘max’) в программе. Также исследуйте построение импульсно-переходной характеристики g(t).

Затем исследуйте работу программы при построении частотных характеристик САУ.

2.5.7. В заключение разработки графического интерфейса пользователя нужно запрограммировать текстовую информацию в графических окнах и графиках.

1) Нужно сделать заголовки графических окон:

для hFig1 – Анализ САУ,

для hFig2 – Временные характеристики САУ,

для hFig3 – Частотные характеристики САУ.

2) Нужно сделать заголовок для окна графического интерфейса. Текст заголовка необходимо разместить в верхней части окна hFig1:

Исследование системы автоматического управления

. В файле sau.sce это может быть строка вида:

hText_title = uicontrol(‘style’, ’text’, ‘position’, [50, 650, 500, 30], ..

‘string’, ’Исследование системы автоматического управления’, ..

‘fontsize’, 14, ‘fontweight’, ‘bold’, ‘horizontalalignment’, ‘center’);

Здесь выбран жирный шрифт размера 14 pt, выравнивание текста по центру.

Подробные сведения о различных параметрах графических компонентов можно получить из справочной системы Scilab, набрав в командном окне:

--> help uicontrol_properties

3) Нужно сделать заголовки и подписи осей для всех графиков временных и частотных характеристик САУ.

4) При желании можно изменить цвет фона графических окон и компонентов. Подробные сведения о необходимых параметрах можно получить из справочной системы Scilab.

2.5.8. После выполнения всех надписей и тщательной проверке работы программы следует «почистить» файл скрипта sau.sce. Надо удалить закомментированные строки программы, продумать и внести нужные комментарии.

3 Содержание отчета

Отчет по лабораторной работе № 13 выполняется в виде списка скрипт-файлов, которые были исследованы при выполнении заданий.

Контрольные вопросы

1. Какие временные характеристики используются в Scilab для исследования систем управления?

1. Какие частотные характеристики используются в Scilab для исследования систем управления?

3. Для чего используется интерфейсный компонент Slider при исследовании временных характеристик?

4. Опишите работу интерфейсного компонента Listbox при исследовании частотных характеристик.

5. Как создаются в Scilab графические окна для исследования временных и частотных характеристик систем автоматического правления?

Литература

1. Алексеев, Е.Р. Scilab: Решение инженерных и математических задач / Е.Р. Алексеев, О.В. Чеснокова, Е.А. Рудченко. – М.: ALT Linux; Бином. Лаборатория знаний, 2008. – [Электронный ресурс]. – Режим доступа: https://docs.altlinux.org/books/2008/altlibrary-scilab-20090409.pdf.

2. Панкратов, И.А. Scilab. Первые шаги: учеб. пособие / И.А. Панкратов – Саратов, СГУ, 2012. – [Электронный ресурс]. – Режим доступа: http://www.sgu.ru/sites/default/files/textdocsfiles /2013/12/10/ scilab_first_steps.pdf.

3. Андриевский, А.Б. Решение инженерных задач в среде Scilab: учеб. пособие / А.Б. Андриевский и др.– СПб.: НИУ ИТМО, 2013. – [Электронный ресурс]. – Режим доступа: http://www.boors.ifmo.ru/file/pdf/1366.pdf.

Содержание

ОСНОВЫ ТЕОРИИ СИСТЕМ

Практикум
по выполнению лабораторных работ
по одноименной дисциплине
для студентов специальности 1-53 01 07
«Информационные технологии и управление
в технических системах»
дневной формы обучения

Составители: **Виноградов** Эдуард Михайлович
Хананов Валентин Андреевич

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 06.06.20.
Рег. № 57Е.
http://www.gstu.by