Вариативная самостоятельная работа №3.

Задание 3.3.3

Основные возможности Scilab, используемые при работе с пользовательскими функциями.

Скрипт-файлы.

Для удобства написания скриптов (функций) в Scilab имеется встроенный редактор – Scinotes. Он позволяет редактировать тексты функций, выполнять их в режиме отладки, содержит функцию автодополнения кода, а также средства непосредственной передачи текста программы в среду Scilab на выполнение. В СКМ зачастую возникает необходимость повторной обработки данных, следовательно, должен иметься механизм для автоматизации этого процесса.

Открыть редактор можно двумя способами:

* подать в консоли Scilab команду scinotes
* выбрать в главном окне последовательно пункты меню Инструменты → Редактор.

В Scinotes предусмотрены классические стандартные команды для работы с файлами, однако ко всему прочему особое внимание следует уделить следующим функциям:

* Execute - содержит пункты, позволяющие передать содержимое редактора в среду Scilab на выполнение или выполнить только выделенную часть.
* Debug - содержит команды для организации и выполнения отладки, такие как включение точек останова (breakpoint), добавление переменных в список наблюдения (watch), настройка запуска функции и т. д. К сожалению, в Scilab 5.1 данный режим недоступен вследствие наличия неустраненной ошибки.

**Функции**

В Scilab реализованы функции, определяемые пользователем. Они, как правило, предназначены для неоднократного использования, имеют входные параметры и не выполняются без их предварительного задания. Рассмотрим несколько способов создания функций в Scilab.

Первый способ — это применение оператора deff, который в общем виде можно записать так:

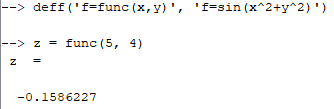


где имя1,...,имяN — список выходных параметров, то есть переменных, которым будет присвоен конечный результат вычислений,

имя\_функции — имя с которым эта функция будет вызываться,

переменная\_1,...,переменная\_M — входные параметры.

Давайте посмотрим на простой пример использования функции deff, используемый для создания и применения функции для вычисления выражения .

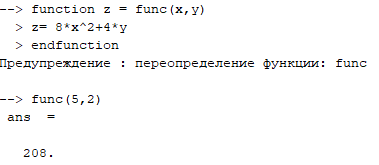


Существует второй способ создания пользовательской функции – при помощи ключевого слова function:



где имя1,...,имяN — список выходных параметров, то есть переменных, которым будет присвоен конечный результат вычислений; имя\_функции — имя с которым эта функция будет вызываться, переменная\_1, ...,переменная\_M — входные параметры.

Пример:



Все имена переменных внутри функции, а также имена из списка входных и выходных параметров воспринимаются системой как локальные, т.е. считаются определенными только внутри функции.

Обращение к функции осуществляется так же, как и к любой другой встроенной функции системы, т.е. из командной строки. Однако функции, хранящиеся в отдельных файлах, должны быть предварительно загружены в систему, например, при помощи оператора exec(имя\_файла) или командой главного меню File – Exec. . . , что, в общем, одно и то же.

При использовании функций необходимо следить за тем, чтобы было передано ровно столько аргументов, сколько было оказано при объявлении функции.

Основные возможности Maxima, используемые при работе с пользовательскими функциями.

**Скрипты.**

Для запуска скрипта необходимо вызвать функцию *batch(имя\_файла)*. Если файл лежит не в стандартных папках программы, может помочь дополнительная функция: *batch(file\_search(имя файла))*.

Для запуска скрипта не обязательно заходить в графическую оболочку программы, можно в консоли ввести команду

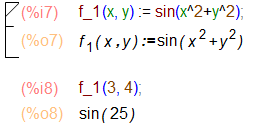
*maxima -b имя\_файла*

и получить тот же результат.

**Функции.**

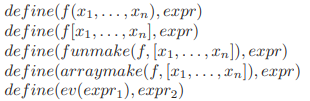
Пользователь может задать собственные функции. Для этого сначала указывается название функции, в скобках перечисляются названия аргументов, после знаков := двоеточие и равно) следует описание функции. После задания пользовательская функция вызывается точно так, как и встроенные функции Maxima.

Например:



Не следует использовать для функций названия, зарезервированные для встроенных функций Maxima. Для создания функций используется также встроенная функция define, которая позволяет преобразовать выражение в функцию.

Синтаксис вызова define довольно многообразен:



Варианты вызова функции define различаются, какой именно объект создаётся: ординарная функция (аргументы в круглых скобках) или массив (аргументы в квадратных скобках). Если первый аргумент — операторы funmake, arraymake, то функция создаётся и вычисляется (аналогично и ev).

Пример:

