МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«15» 02 2021 г.

ЦИФРОВАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ

Методические указания к выполнению лабораторных работ
для студентов направления подготовки 09.04.01

Курск 2021

УДК 004
Составитель: В.С. Панищев

Рецензент
Кандидат технических наук, доцент *Ю.А. Халин*

Цифровая обработка изображений: методические указания к выполнению лабораторных работ по дисциплине «Цифровая обработка и анализ изображений в информационных системах» / Юго-Зап. гос. ун-т; сост.: В.С. Панищев; Курск, 2020. 28 с.

Методические указания предназначены для ознакомления с студентов с принципами создания эффектов трехмерной графики и приобретения навыков в написании программ обработки изображений.

Предназначены для студентов направления подготовки 09.04.01 Информатика и вычислительная техника.

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60х84 1/16.
Усл. печ. л. 1,2. Уч. – изд.л. 1,1. Тираж 30 экз. Заказ . Бесплатно.
Юго-Западный государственный университет
305040, Курск, ул. 50 лет Октября, 94.

Содержание

Работа №1 Моделирование эффекта трехмерного ландшафта в трехмерной графике

1. Цель работы

Целью данной работы является изучение принципов создания эффектов трехмерной графики и приобретение навыков в написании программ, обеспечивающих создание эффекта трехмерных ландшафтов.

2. Основные теоретические положения

В настоящее время существует несколько основных принципов представления данных для хранения информации о ландшафтах:

Первый - использование регулярной сетки высот (или другое название Карта Высот - HeightMap).

Второе - использование иррегулярной сетки вершин и связей, их соединяющих (т.е. хранение простой триангулизированной карты).

Третий - хранение карты ландшафта, но в данном случае хранятся не конкретные высоты, а информация об использованном блоке. В этом случае создается некоторое количество заранее построенных сегментов, а на карте указываются только индексы этих сегментов.

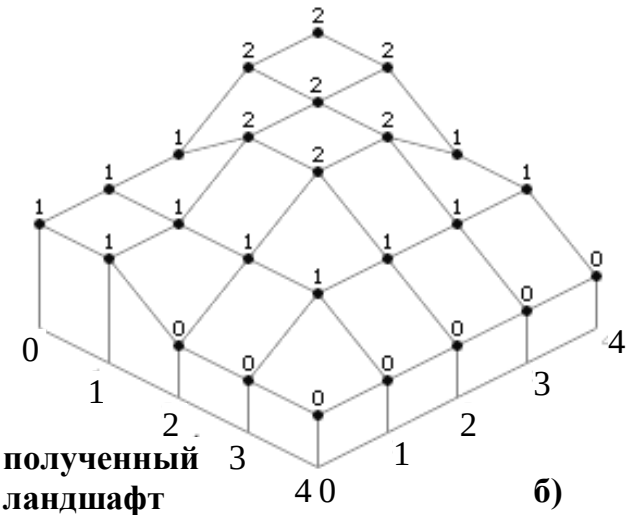
2.1. Построение ландшафтов с помощью регулярной сетки высот

В данном способе данные представлены в виде двухмерного массива. Уже заданы две координаты: x – по высоте и y по ширине массива, а третья координата, заданная значением в конкретной ячейке – это высота точки будущего ландшафта. Пример карты высот представлен на рис. 1а, а соответствующий, а соответствующий ландшафт, построенный на ее основе – на рис. 1б.

	0	1	2	3	4
0	1	1	1	2	2
1	1	1	2	2	2
2	0	1	2	2	1
3	0	1	1	1	1
4	0	0	0	0	0

карта высот

а)



полученный ландшафт

б)

Рис. 1. Пример карты высот и соответствующего трехмерного ландшафта для метода регулярной сетки высот

Обычно карту высот хранят в файлах картинок. Это позволяет легко вносить изменения и более-менее наглядно просматривать данные. Тогда двумя координатами будет положение конкретного пикселя на картинке, а третья координата будет представлена цветом (чем выше значение, прямая зависимость от яркости пикселя - тем больше значение высоты для этой точки). Обычно такие картинки содержатся в монохромном варианте, но можно использовать и все цвета радуги. Второй вариант дает нам больше

градаций высоты, чем предполагаемые 256 градаций в случае монохромного представления.

Примерный вид такого ландшафта выглядит следующим образом (Рис. 2):

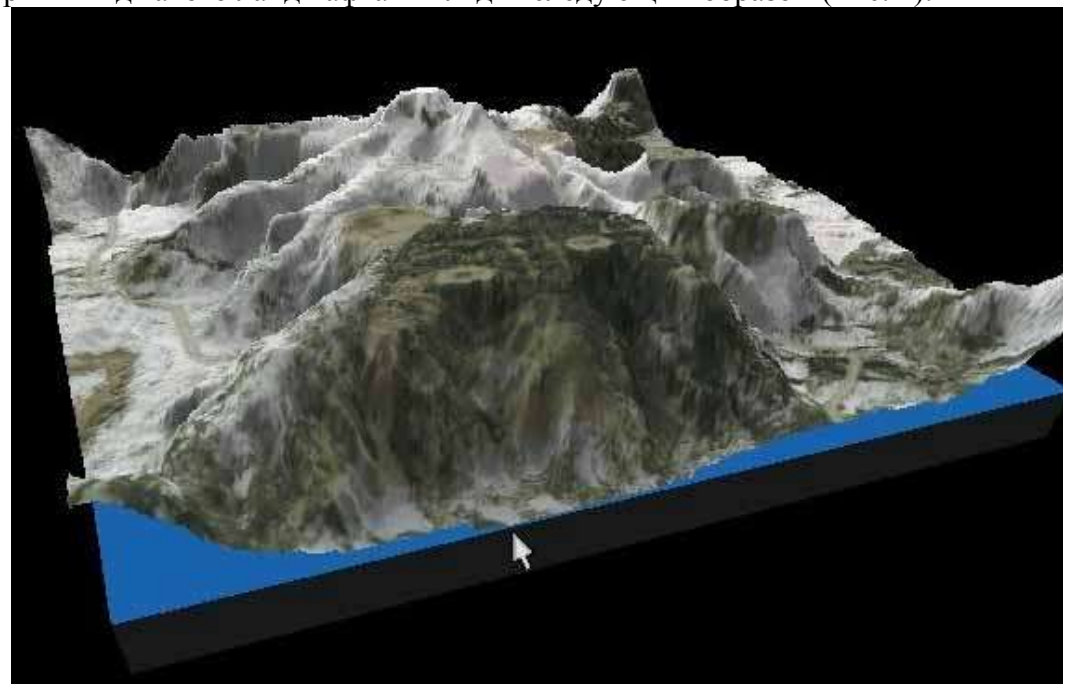


Рис. 2. Пример трехмерного ландшафта

Один из алгоритмов для построения ландшафтов состоит из следующих основных этапов:
Создаем двухмерный массив размером $n \times n$, где величина *n* зависит от размера будущего ландшафта и определяется в конечном счете количеством точек по ширине и высоте экране. Например, массив может быть размером 100×100 или 200×200;
Генерируем в этот массив, во все его ячейки случайные значения;
Проходимся простым сглаживающим фильтром:
1. Берем точку (все по порядку);
2. Вокруг этой точки берем значения всех восьми точек + значение выбранной точки;
3. Суммируем все эти девять значений;
4. Делаем арифметическое усреднение с помощью деления полученного на шаге 3 значений на 9;
5. Полученный результат записываем в исходную точку;
6. Пробегам весь массив;
7. Заполняем случайными значениями еще несколько точек в исходном массиве, конкретная точка выбирается случайным образом;
8. Заново проходимся сглаживающим фильтром.
9. При необходимости получения более сглаженного ландшафта повторяем пункты 3-4 еще несколько раз.

В приведенном алгоритме необходимо учитывать, что выполнение пунктов 2–4 для первой и последней строки и столбца не будет иметь смысла, в следствие того, что вокруг каждой точки будет не восемь значений, а только пять. В этом случае следует либо игнорировать усреднение в данных точках массива и заранее установить их в ноль, либо для первой строки и первого столбца суммировать с *n*-м, а для *n*-х – соответственно с 1-й.

2.2. Построение ландшафтов с помощью иррегулярной сетки

Другой способ построения трехмерных ландшафтов – использование иррегулярной сетки вершин и связей их соединяющих. Зачастую такие решения применяются в

специализированных пакетах для игр или специальных пакетах для работы с трехмерной графикой, типа 3Dmax. Данные хранятся в виде трехмерных моделей.
Трехмерный ландшафт, сгенерированный по данному способу выглядит следующим образом (Рис. 3):

Алгоритм построения такого ландшафта крайне прост:

1. Случайным образом на предполагаемой карте ландшафта располагаем вершины будущего ландшафта. Количество вершин также как и в предыдущем способе зависит от размера рабочей области экрана и размерности необходимого ландшафта;

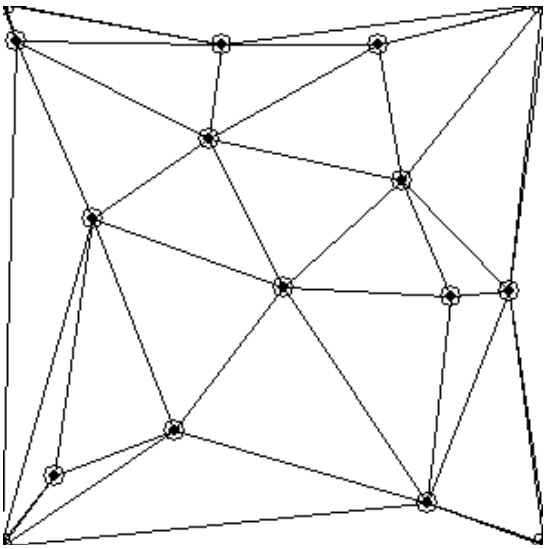


Рис. 3. Ландшафт на основе иррегулярной сетки вершин и связей

2. Соединяем все вершины связями таким образом, чтобы каждая вершина была соединена связью с каждой из соседних вершин;
3. При необходимости окрашиваем готовый ландшафт каким-либо цветом.

2.3. Сферические ландшафты

Другой тип ландшафтов представляют сферические ландшафты, которые применяются в области космической науки и техники, например, для создания неискаженных континентов на планетах.
Этот метод крайне прост, но труден в реализации и состоит из следующих шагов:
1. Взять сферу.
2. Разделить ее пополам следующим образом (Рис. 4)

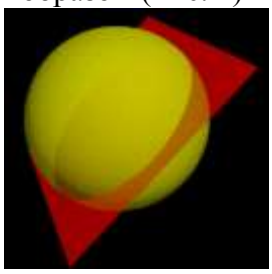


Рис. 4. Пример разделения сферы

3. Сделать одну половину немного больше
4. Сделать другую половину немного меньше.
5. Повторить этот процесс много раз, выбирая при этом каждый раз случайный путь обхода половин сферы.
В результате после одной итерации вид сферы будет следующим (Рис. 5):

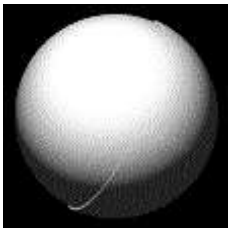


Рис. 5. Ландшафт после одной итерации

После двух итераций следующим (Рис. 6):

Рис. 6. Сфера после двух итераций

После трех итераций вид сферы будет таким (Рис. 7):

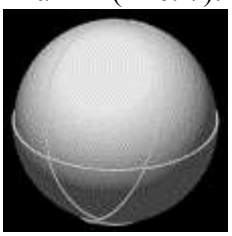


Рис. 7. Ландшафт после трех итераций

После большого количества итераций, небольшие горные хребты на сфере начинают приобретать форму континентов. В конечном счете, небольшие горные хребты исчезают, постепенно превращаясь в горные цепи. Если задать большое количество итераций, то образец земли через некоторое время приобретет вид планеты.

После всего 100 итераций уже видны очертания континентов (Рис. 8).

Рис. 8. Ландшафт после 100 итераций

После 1000 итераций начинают появляться очертания хребтов (Рис. 9).

Рис. 9. Ландшафт после 1000 итераций

10000 итераций дает вид достаточно больших гор (Рис. 10).

Рис. 10. Ландшафт после 10000 итераций

Для реализации этого метода возможно два способа, выбор которого зависит от того, что необходимо получить в конечном итоге. Один метод заключается в генерации карты текстуры, которая может быть «обернута» вокруг сферы без искажений. Другой метод основан на создании 3D модели планеты, используя полигоны.

2.4. Метод 3D полигонов

Этот метод дает более качественные результаты, по сравнению с предыдущим методом, но требует больших затрат памяти. Планета в этом случае содержат более 10000 полигонов.
В первую очередь необходимо выбрать вектор нормали к точке, из которой ведется наблюдение. Процедура для этого может выглядеть, например, следующим образом.

```
        procedure Normals //случайным образом выбирает проекцию //нормали
 VECTOR: n
  loop
     n = VECTOR( random(-1,1), random(-1,1), random(-1,1) )
     magnitude = mag(n)
   until magnitude(n) < 1
```

Сначала необходимо задать сферу. Чем больше количество вершин в ней будет, тем лучше. Также, лучше иметь сферу с распределением вершин поперек ее поверхности. После того как задана сфера, выбран вектор нормали, метод может быть реализован следующим образом.

```
loop
  n = random vector    ;см. Пример выше
          loop         ;сквозь каждую вершину сферы
    d = dotproduct( n , this vertex)  ;если вершина впереди или
                                                    ;позади точки наблюдения

        if d > 0 then                       ; если впереди, то
           move this vertex out a little bit
        else                                ; если позади, то
           move this vertex in a little bit
        end if
   end loop
end loop
```

Для обеспечения скорости и точности работы алгоритма можно сохранять дополнительный номер к каждой вершине. Когда происходит перемещение «в» или «из» точки, достаточно добавить или вычесть 1 из номера вместо перемещения положения вершины.
Небольшой модификацией к методу, описанному в п.3 является применение нормали, выбранной в данном методе, как расстояние от точки наблюдения до центра сферы.
Центр сферы определен вектором n. Это является и нормальным вектором к точке наблюдения и одновременно расстоянием от центра сферы до точки наблюдения (Рис. 11).

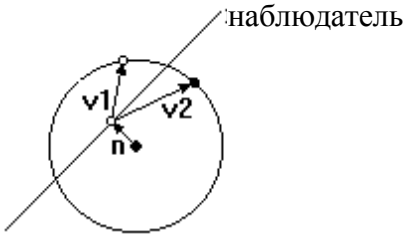


Рис. 11. Выбор нормали к точке наблюдения

На диаграмме отмечены две точки: одна из них впереди точки наблюдения, а другая позади этой нее. Из диаграммы видно, что вектор v1, начинающийся из конца вектора n и заканчивающийся в точке, отмеченной на диаграмме белым кружочком, больше единицы, а вектор v2 и вектор n меньше единицы. Тогда соответствующая процедура может выглядеть следующим образом:

```
loop
        n = random vector                ;выбрать случайный вектор по методу,
                                         описанному выше
  m = randomly either -1 or 1

  loop                                   ;по всем вершинам сферы

     p = координаты данной вершины
     v = p - n
   d = dotproduct(n , v)             ; вершина впереди
                                         ; или позади точки наблюдения

     if d > 0 then                       ; если впереди, то
        move this vertex by m
     else                                ; если позади, то
        move this vertex by -m
     end if
  end loop

end loop
```

В данной процедуре введен новый вектор m. Все точки сферы позади точки наблюдения «чуть-чуть» смещены за сферу, а точки впереди сферы – соответственно «чуть-чуть» перемещены в сферу. Всякий раз, когда выбирается новая точка наблюдения, центр сферы находится позади точки наблюдения. В данном случае делается исключение для случая, когда точка наблюдения проходит через середину. Это означает, что большинство точек находятся позади точки наблюдения. В результате планета состоит полностью из земли. Для того, чтобы это исправить в процедуре, описанной выше случайным образом проверяется, являются ли точки сферы впереди или позади точки наблюдения. В случае, если являются, делается замена положения точки сферы. Переменная m является индикатором: ‘-1’ означает «немного внутри», а ‘+1’ – «немного вне».

3. Задание на лабораторную работу

Написать программу для моделирования трехмерного ландшафта с помощью любого из описанных в теоретической части методов. Программа может быть написана на языке программирования Pascal, C, C++, Java. Программа обязательно должна содержать следующие элементы:

- В случае использования метода карт высот должен задаваться коэффициент сглаживания поверхности, который должен быть задан пользователем;
- Необходимо предусмотреть возможность повторной прорисовки ландшафта путем добавления в форму проекта программы специальной кнопки, которой должна быть назначена эта функция;
- Необходимо предусмотреть возможность явного задания размеров ландшафта путем добавления в форму проекта программы специальных полей, в которую пользователь может задавать размер предполагаемого ландшафта;
- Ландшафт должен быть изображен в цвете.

4. Контрольные вопросы

1. Какие существуют основные методы для построения трехмерных ландшафтов?
2. В чем состоит основной принцип сглаживания ландшафтов?
3. Охарактеризуйте метод регулярной карты высот. В чем состоят его достоинства и недостатки?
4. Охарактеризуйте метод иррегулярной сетки вершин и связей. Опишите его достоинства и недостатки.
5. Сравните достоинства и недостатки метода регулярной карты высот и иррегулярной сетки вершин и связей.
6. На чем основан метод сферических ландшафтов?
7. В чем отличие метода 3D полигонов от метода сферических ландшафтов?
8. Для чего в методе 3D полигонов применяется вектор нормали?
9. В чем состоит модификация метода 3D полигонов?
10. Для чего в модификации метода 3D полигонов введена дополнительная переменная m?

5. Содержание отчета

Результаты выполнения лабораторной работы оформляются в виде отчета, который должен содержать следующие разделы:

1. тема лабораторной работы;
2. цель работы;
3. общая часть и индивидуальный вариант задания;
4. описание алгоритма (граф-схема алгоритма);
5. текст программы;
6. результаты моделирования;
7. анализ результатов и выводы.

Работа №2 Обработка цифровых изображение в Octave

Цель работы: ознакомление с средствами и методическими приемами анализа изображений.

Задача: провести анализ изображений в соответствии с вариантом.

Выполнить детальные задания в конце файла. Подготовить содержательный отчет, сделать выводы.

Для выполнения настоящей работы используется бесплатная программа GNU Octave (http://www.octave.org). Существуют версии языка для различных дистрибутивов GNU Linux (ALT Linux, Debian, Ubuntu, Mandriva и др.) и для ОС Windows.На наш взгляд, GNU Octave больше ориентирован на работу в Linux. Работа в ОС Windows возможна, но пользователю Windows надо быть готовым работать с простым текстовым редактором и командной строкой.

1 Установка Octave

1.1 Установка Octave в Windows

Установка Octave в ОС Windows проходит стандартным образом. Необходимо с официального сайта http://www.gnu.org/software/octave/ скачать версию программы для Windows и установить её.На первом этапе нужно выбрать папку для установки программы. На следующем – определить, правильно ли выбрана платформа (параметр – ATLAS Libraries), под которую будет оптимизирована программа Octave, и на этом же этапе — выбрать пакеты расширений (параметр Octave Forge), которые будут установлены вместе с программой.

В результате будет установлен текстовый редактор Notepad++, интерпретатор Octave, пакеты расширений и англоязычная документация по Octave. Принципы работы с интерпретатором описаны ранее. Существует и графическая оболочка (среда) для работы с Octave – Octave Workshop. Программу можно скачать с официального сайта http://sourceforge.net/projects/octave-workshop/

1.2. Установка Octave 15

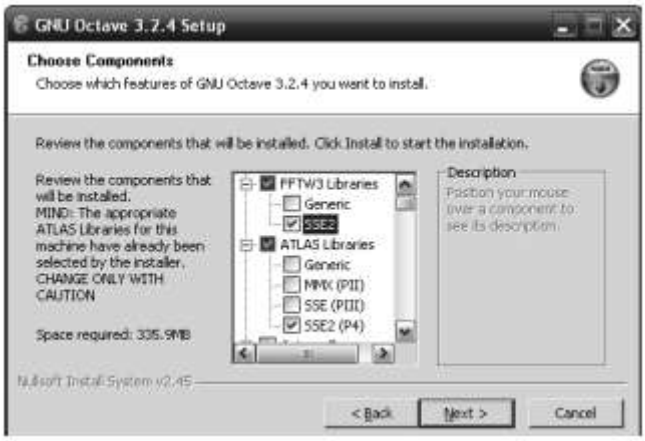


Рис. 1. Установка Octave. Выбор архитектуры процессора.

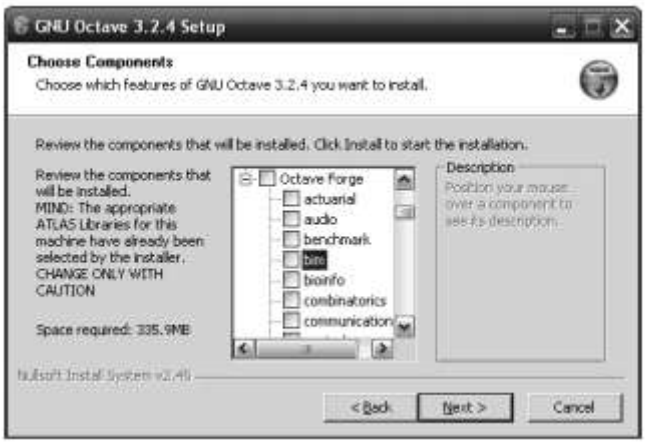


Рис. 2 Установка Octave. Выбор пакетов расширений.

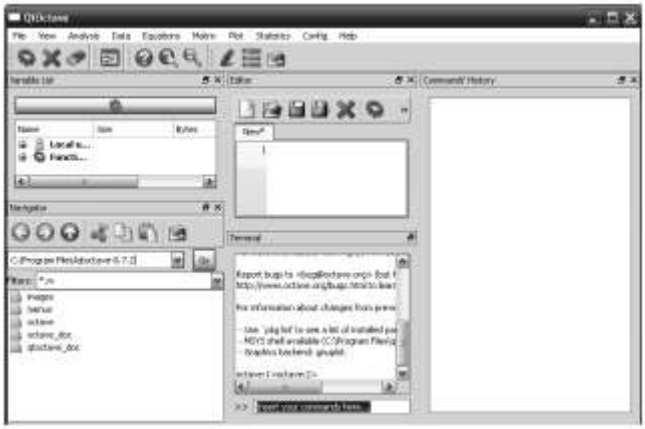


Рис. 3 Окно QtOctave под управлением ОС Windows

Её установка проходит стандартным для Windows способом. В состав программы Octave Workshop входит графическая оболочка, сам интерпретатор Octave и некоторые пакеты расширений. Наиболее мощной графической оболочкой для работы с Octave является программа QtOctave, которая разработана для ОС Linux. Она портирована в ОС Windows в виде portable-версии, которую можно скачать по адресу http://portableapps.com/node/14720 Или скачать с (http://www.octave.org).

Программу нужно разархивировать и запустить файл qtoctave.exe. Окно QtOctave в ОС Windows представлено на рис. 3.

1.3 Графическая оболочка QtOctave

После запуска QtOctave на экране появляется основное окно приложения (рис. 4). Это окно содержит меню, панель инструментов и рабочую область Octave Terminal. Окно может иметь другой внешний вид, в зависимости от предпочтений пользователя.

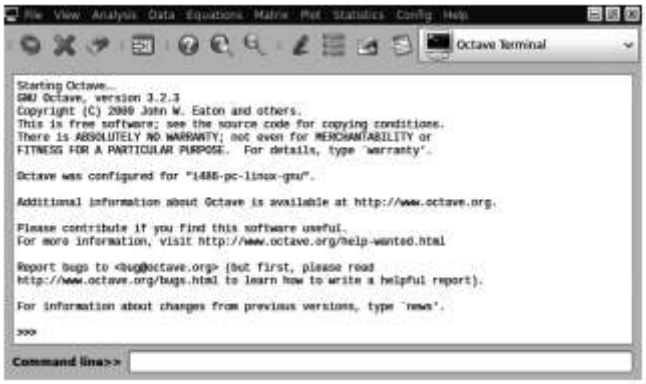


Рис. 4 Окно QtOctave

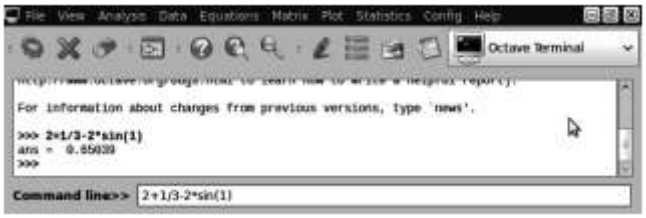


Рис. 5 Выполнение элементарной команды

Признаком того, что система готова к работе, является наличие знака приглашения >>>. Ввод команд осуществляется с клавиатуры в командной строке Command line. Нажатие клавиши Enter заставляет систему выполнить команду и вывести результат, что проиллюстрировано на рис. 5.

Понятно, что все выполняемые команды не могут одновременно находиться в поле зрения пользователя. Поэтому просмотреть информацию, которая покинула видимую часть окна, можно с помощью полос прокрутки и клавиш Page Up и Page Down.

Клавиши ↑ и ↓ позволяют вернуть в командную строку ранее введённые команды или другую входную информацию, так как вся эта информация сохраняется в специальной области памяти.

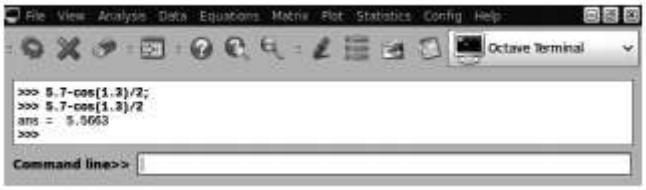


Рис. 6 Примеры вывода результатов вычислений

Так, если в пустой активной командной строке нажать клавишу ↑, в ней появится последняя вводимая команда. Повторное нажатие вызовет предпоследнюю команду, и так далее. Клавиша ↓ выводит команды в обратном порядке.

Таким образом можно сказать, что вся информация в рабочей области находится в зоне просмотра. Важно знать, что в зоне просмотра нельзя ничего исправить или ввести. Единственная допустимая операция — это выделение информации с помощью мыши и копирование её в буфер обмена, к примеру, для дальнейшего помещения в командную строку.

В командной строке действуют элементарные приёмы редактирования:

- → — перемещение курсора вправо на один символ;
- ← — перемещение курсора влево на один символ;
- Home — перемещение курсора в начало строки;
- End — перемещение курсора в конец строки;

• Del — удаление символа после курсора;

• Backspace — удаление символа перед курсором.

Кроме того, существуют особенности ввода команд. Если команда заканчивается точкой с запятой (;), то результат её действия не отображается в рабочей области. В противном случае, при отсутствии знака «;», результат действия команды сразу же выводится в рабочую область (рис. 6).

Работа в среде QtOctave может осуществляться в так называемом программном режиме. В этом случае в командной строке указывается имя программы, составленной из управляющих команд и функций Octave и имеющей расширение .m. Это достаточно удобный режим, так как он позволяет сохранить разработанный вычислительный алгоритм в виде файла и повторять его при других исходных данных и в других сеансах работы.

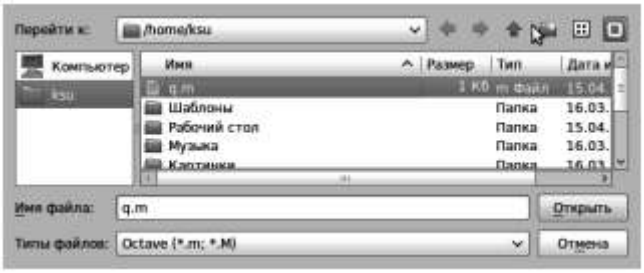


Рис. 7. Выбор файла для выполнения в Octave

Выполнить команды Octave, хранящиеся в файле с расширением **.m**, позволяет команда главного меню **File → Run an Octave Script**.

Эта команда продублирована кнопкой в панели инструментов и открывает окно диалога, представленное на рис. 7 Смена текущей директории осуществляется командой **File → Change Directory**. Команда также открывает диалоговое окно, предназначенное для выбора нового каталога.

Выход из программы выполняет команда **File → Quit.** Очистить рабочую область от введённых ранее команд можно, обратившись к пункту меню **View → Clear terminal.** Команда продублирована кнопкой в виде ластика на панели инструментов.

Команда **View → Dock Tools → Variable List** открывает окно, показанное на рис. 8 Здесь пользователю доступны значения всех переменных, вычисленные в течение текущей сессии. Они сохраняются в специально зарезервированной области памяти и при желании, определения всех переменных и функций, входящих в текущую сессию, можно сохранить на диске в виде файла.

Окно, представленное на рис. 9 содержит список выполненных команд и открывается командой **View → Dock Tools → Command List.** Выполнить поиск, просмотр, открытие файлов и каталогов, осуществить смену текущей директории, установить путь к файлу и так далее можно в окне, показанном на рис. 10 Это окно появится, если выполнить команду **View → Dock Tools → Navigator.** Текстовый редактор в QtOctave вызывает команда **View → Dock Tools → Editor.**

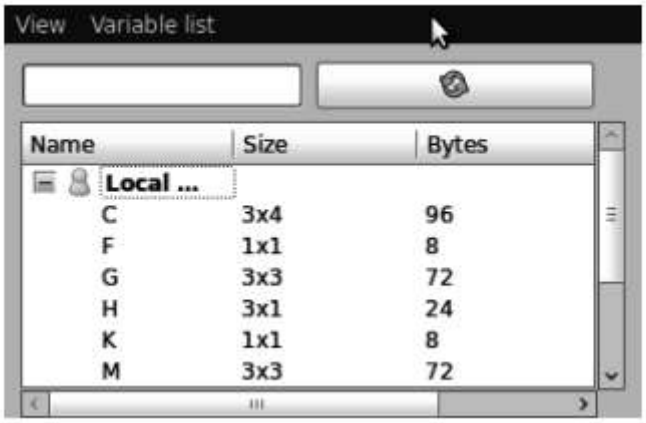


Рис. 8 Список переменных, определённых в процессе работы

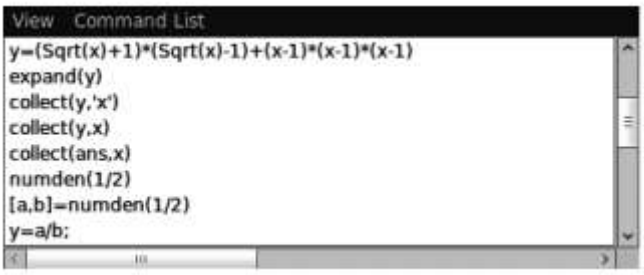


Рис. 9 Список выполненных команд

Ввод текста в окно редактора осуществляется по правилам, принятым для команд Octave. Рис. 11 содержит пример ввода команд для решения биквадратного уравнения 2**x**4 - 9**x**2 + 4 = 0. Точка с запятой «;» ставится после тех команд, которые не требуют вывода значений.

Для сохранения введённой информации необходимо выполнить команду File → Save из главного меню редактора. Если информация сохраняется впервые, появится окно Save file As....Выполнить команды, набранные в текстовом редакторе, может команда меню редактора Run → Run. Кроме того, как было сказано выше, можно набрать имя созданного в текстовом редакторе файла в командной строке и нажать ENTER.

Все эти действия приведут к появлению в рабочей области результатов вычислений, что видно на рис. 11

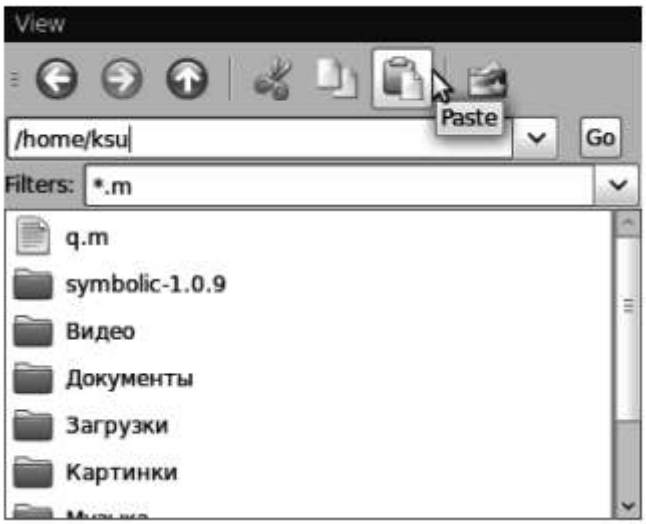


Рис. 10 Смена каталога

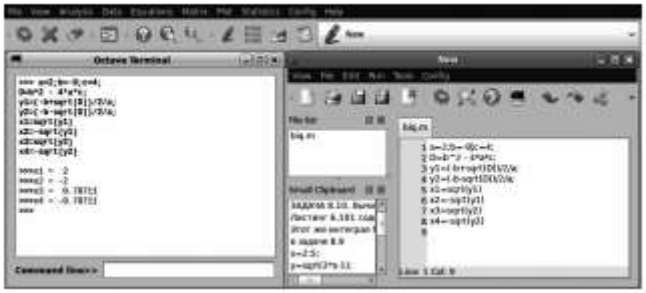


Рис.11 Пример работы в текстовом редакторе

Отметим, что текстовый редактор имеет возможность работы с множеством окон и обладает принятыми для текстовых редакторов приёмами редактирования, подробно останавливаться на которых мы не будем.

Для выхода из режима редактирования можно просто закрыть окно или использовать команду File → Close. Ранее созданный файл открывает команда главного меню редактора File → Open. Окна, представленные на рис. 8-11, обладают общим свойством. Команды View → Show inside of main window и View →Show outside of main window позволяют выводить окна внутри основного окна QtOctave (8) и за его пределами соответственно.

Управлять положением окон в среде QtOctave можно командой View → Windows Layout. А команда View → Show позволяет отображать или удалять кнопки на панели инструментов. Далее представлено краткое описание других пунктов главного меню QtOctave:

• Analysis — решение некоторых задач матанализа (интегрирование и решение обыкновенных дифференциальных уравнений);

• Data — работа с матрицами (ввод, форматированный ввод, ввод из файла, запись в файл);

• Equations — решение линейных и нелинейных уравнений;

• Matrix — действия над матрицами (сложение, вычитание, умножение, транспонирование, инвертирование, вычисление определителя);

• Plot — работа с графикой (построение двумерных и трёхмерных графиков, форматирование графической области, запись графического изображения в файл);

• Statistics — вычисление некоторых статистических функций;

• Config — настройка конфигурации системы, установка пакетов расширений;

• Help — справочная информация.

2 Принципы работы с интерпретатором

Octave — высокоуровневый интерпретируемый язык программирования, предназначенный для решения задач вычислительной математики. В состав пакета входит интерактивный командный интерфейс (интерпретатор Octave). Интерпретатор Octave запускается из терминала ОС Linux или из его порта в Windows. После запуска Octave пользователь видит окно интерпретатора (см. рис. 12).

В окне интерпретатора пользователь может вводить как отдельные команды языка Octave, так и группы команд, объединяемые в программы. Если строка заканчивается символом «;», результаты на экран не выводятся. Если же в конце строки символ «;» отсутствует,

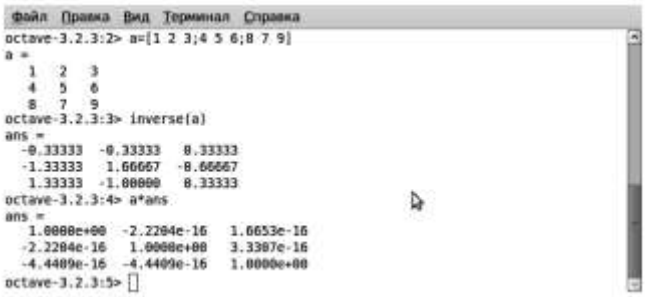


Рис. 12 Окно интерпретатора Octave

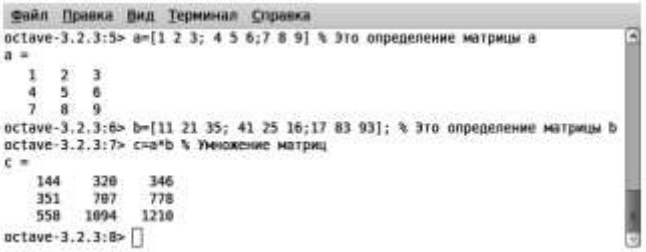


Рис. 13. Использование символов «;» и «%» в Octave

результаты работы выводятся на экран (см. рис. 13). Текст в строке после символа % (процент) является комментарием и интерпретатором не обрабатывается1 (см. рис. 13). Рассмотрим несколько примеров.

Пример 1. Решить систему линейных алгебраических уравнений (СЛАУ)

$$\begin{cases} 3x_1 + 5x_2 - 7x_3 = 11 \\ 3x_1 - 4x_2 + 33x_3 = 25 \\ 22x_1 - 11x_3 + 17x_3 = 22 \end{cases}$$

Возможны два варианта решения любой задачи в Octave:

1. Терминальный режим. В этом режиме в окно интерпретатора последовательно вводятся отдельные команды.

2. Программный режим. В этом режиме создаётся текстовый файл с расширением .m, в котором хранятся последовательно выполняемые команды Octave. Затем этот текстовый файл (программа на языке Octave) запускается на выполнение в среде Octave. Для решения СЛАУ в окне интерпретатора Octave последовательно введём следующие команды (листинг 1):

```
% Определение матрицы коэффициентов системы линейных уравнений.
A=[3 5 −7;3 −4 33;22 −11 17];
b=[11; 25; 22]; % Вектор правых частей СЛАУ.
x=A^(−1)*b % Решение системы методом обратной матрицы.
x =
  1.56361
  2.55742
```

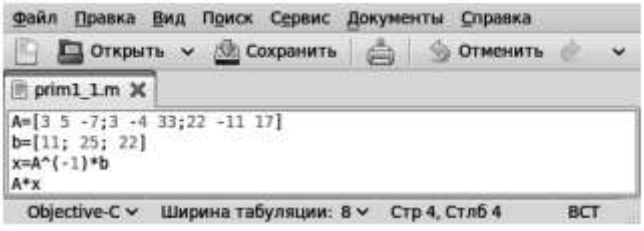


Рис. 14 Программа для решения примера 1

```
  0.92542
octave-3.2.3:27> A*x % Проверка.
ans =
  11.000
  25.000
  22.000
```

В переменной ans хранится результат последней операции, если команда не содержит знака присваивания. Следует помнить, что значение переменной ans изменяется после каждого вызова команды без операции присваивания.

Теперь рассмотрим, как решить эту же задачу в программном режиме. Вызовем любой текстовый редактор, например gedit, в окне которого последовательно введём следующие команды:

```
A=[3 5 -7;3 -4 33;22 -11 17]
b=[11; 25; 22]
x=A^(-1)*b
A*x
```

Сохраним введённые команды в виде файла с расширением .m, например, /home/evgeniy/prim1_1.m (рис. 14). Теперь эту программу необходимо запустить на выполнение из интерпретатора. Для этого в окне интерпретатора введём команды

```
cd '/home/evgeniy'% Переход в каталог, где хранится программа.
prim1_1 % Запуск программы.
```

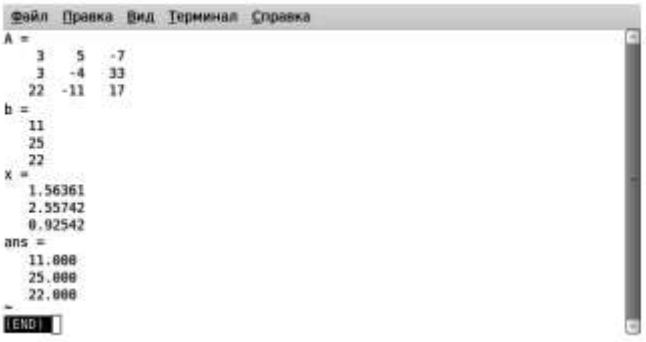


Рис. 15 Окно терминала после запуска программы prim1_1

Окно интерпретатора примет вид, представленный на рис. 15. Просмотрев результаты работы программы, нажмите q для возвращения в режим ввода команд терминала.

3 Обработка изображений в GNU OCTAVE (MATLAB)

Внимание! Некоторые коды примеров содержат небольшие неточности, препятствующие их корректному выполнению. Это означает, что необходимо найти и исправить «неточность», что требует понимания работы кода программы. Разумеется, предоставляемый по итогам работы отчет должен содержать корректный код программы, без ошибок, с которыми программа работать не сможет.

3.1 Представление об изображениях

Изображения рассматриваются, как прямоугольные матрицы, каждый элемент которых соответствует яркости пикселя изображения (“pixel” от англ. “picture element”). Открытие изображений в средах научного программирования позволяют применять к ним средства, доступные для анализа матриц. Представление матрицы в виде изображения показано на рисунке (рис. 16)

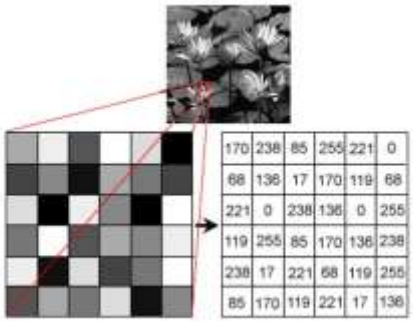


Рис. 16 Представление фрагмента серого изображения в виде матрицы

При этом следует помнить, что «серые», одноканальные изображения представляют собой одну матрицу, тогда как изображения «цветные» представляют собой три канала R,G,B – красный (λ=625—740), зеленый (λ=500—565), голубой (λ=485—500), т.е. представляются тремя матрицами. Это нужно учитывать при открытии цветного изображения.

3.2. Фильтры изображений в Octave (Matlab)

Для ознакомления с методами обработки изображений в Octave рассмотрим следующий пример. Для занятия необходимо использовать изображения в папке files архива лабораторной работы. Внимание! Для корректной работы Octave имя файла и путь к нему не должны содержать кириллических символов.

Пример 1. Откроем изображение lily.jpg из архива.

```
%открытие файла изображения в матрицу A
My_lily=imread('lily.jpg');

%посмотрим картинку
imshow(My_lily);
```

Обратим внимание, что изображение содержит три канала, а значит открытие создаст трехмерную матрицу. Для изучения переменных и объектов, находящихся в памяти, можно использовать окно «Область переменных» Octave. Наряду с этим, свойства переменной можно узнать с помощью команды whosVAR_NAME. Это выдаст нам свойства переменной VAR_NAME. Узнаем свойства матрицы my_lily.

```
%узнаем свойства переменной My_lily
whos My_lily

Variables in the current scope:

   Attr Name        Size                     Bytes  Class
   ==== ====        ====                     =====  =====
        My_lily     600x800x3                1440000  uint8
```

Переменная My_lily представляет собой трехмерный массив размерностью 600x800 (**600x800x**3), занимающий в памяти 1440000 байт, элементы массива представляют собой данные класса **uint8** (8-ми битное целое). В элемент класса uint8 можно записать данные от 0 до 28=255, т.е. всего 256 значений. Ясно, что этот тип данных наиболее приемлем для хранения в них матриц изображений. Octave (Matlab) доступны любые методы, применимые для преобразования изображений и, зачастую, используемые в графических редакторах, такие как обрезка, масштабирование, фильтры и т.д. Однако, неоспоримым достоинством Octave является воспроизводимость и контроль действий.

Пример 2. Масштабирование изображения lily.jpg и сохранение файла с меньшим разрешением. Кадрирование.

Для масштабирование используется функция new_image_mat=imresize(image_matrix,scale). Функция imresize использует два параметра. В качестве первого, image_matrix,

используется исходное изображение, в качестве второго scale, используется масштабный коэффициент. Если он меньше 1, изображение уменьшается, если больше, увеличивается. Масштабированную матрицу изображения new_image_mat можно пересохранить, используя функцию, imwrite(new_image_mat, filepath). Здесь new_image_mat – матрица, которую требуется сохранить в изображение, filepath – полный путь к файлу, включая имя

```
%открытие файла изображения в матрицу My_lily
My_lily=imread('lily.jpg');

%масштабирование на 50%
My_lily_small=imrsize(My_lily, 0,5);

%посмотрим картинку после масштабирования
imshow(My_lily_small);
title('После уменьшения на 50%');

%путь к файлу и его имя в переменной
filepath=('C:\my_folder\lily_sm.jpg');

%сохраняем уменьшенное на 50% изображение в файл
imwrite(My_lily_small, filepath);
```

В результате выполнения кода, в папке C:\my_folder будет сохранено новое изображение lily_sm.jpg, ширина и высота которого уменьшены на 50% по сравнению с исходным изображением. Изображение иногда приходится кадрировать, т.е. выбрать фрагмент исходной матрицы, удалив все лишнее. Код ниже показывает, как это может быть сделано:

```
%открытие файла изображения в матрицу My_lily
My_lily= imread('lily.jpg');

%исходная картинка
figure;
imshow(My_lily);
title('Исходное изображение');

%кадрирование одного из цветков
My_lily_crop= My_lily(140:285,433:617,:);

%посмотрим кадрированный фрагмент
figure;
imshow(My_lily_cro);
title('Кадрированный фрагмент');

%путь к файлу и его имя в переменной
filepath=('C:\my_folder\lily_crop.jpg');

%сохраняем уменьшенное на 50% изображение в файл
imwrite(My_lily_crop, filepath);
```

Для проведения анализа изображение часто должно пройти этапы обработки, направленные на его размытие или повышение резкости, распознавание границ, а также преобразование изображения (например, из цветного в оттенки серого или в двухцветное, черно-белое). Это может быть достигнуто в Octave (Matlab) с помощью пользовательских или предустановленных функций. Достижение целого ряда графических эффектов возможно с помощью применения так называемой свертки изображения – применение матрицы коэффициентов (матрицы свертки), которая «умножается» на значение пикселей изображения для получения требуемого результата.

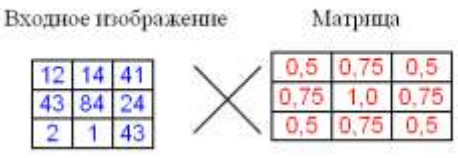


Рис. 17 Свертка изображения по матрице коэффициентов

Например, для создания фильтра размытия (blur) требуется выполнить свертку изображение по «матрице Гаусса» – матрице, в которой элементы распределены по нормальному закону (распределение Гаусса). Чтобы создать такую матрицу, может быть использована функция fspecial().

```
>> B=fspecial('Gaussian',5,5)
B =
    0.0369    0.0392    0.0400    0.0392    0.0369
    0.0392    0.0416    0.0424    0.0416    0.0392
    0.0400    0.0424    0.0433    0.0424    0.0400
    0.0392    0.0416    0.0424    0.0416    0.0392
    0.0369    0.0392    0.0400    0.0392    0.0369
```

Код выше показывает создание «гауссовой матрицы». Пространственная сетка для этой матрицы (создается вызовом функции mesh()) представлена (рис.18).

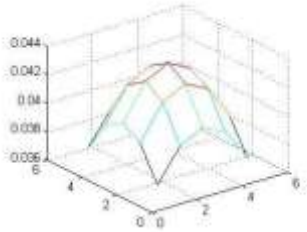


Рис. 18 Пространственная сетка «гауссовой матрицы»

Необходимо знать, что свертка матриц (конволюция) может быть выполнена только для одноканальной матрицы – серого изображения. Если матрица 3-х канальная надлежит сначала разложить матрицу по отдельным каналам, применить к ней фильтр, а затем снова совместить каналы в цветное изображение.

Пример 3. *Разложение цветного изображения по отдельным каналам и вывод его в совмещенном графическом окне (subplot).*

```
A=imread('lily.jpg');
%выбор в отдельные матрицы каналы
изображения %красный
A_R=A(:,:,3);
%зеленый
A_G=A(:,:,2);
%голубой
A_B=A(:,:,1);
figure;
%создание новой картинки
subplot(2,2,1); %значит, что создается изображение
2x2, текущий сегмент 1
imshow(A);
title('RGB');
subplot(2,2,2); %значит, что создается изображение
2x2, текущий сегмент 2
imshow(A_B);
title('Blue channel');
subplot(2,2,3); %значит, что создается изображение
2x2, текущий сегмент 3
imshow(A_G);
title('Green channel');
subplot(2,2,4); %значит, что создается изображение
2x2, текущий сегмент 4
imshow(A_R);
title('Red channel');
saveas(gcf,'MyFigure.png'); %сохраним картинку в файл png
figure; %create new figure
%combine channels into older picture - nothing been changed
```

```
comb_lily(:,:,1)=A_R; comb_lily(:,:,2)=A_G;
comb_lily(:,:,3)=A_B;
comb_lily=uint8(comb_lily);
imshow(comb_lily);
title('lily combined from separated channels');
```

Результат вывода каналов изображения показан на рис. 5.4.

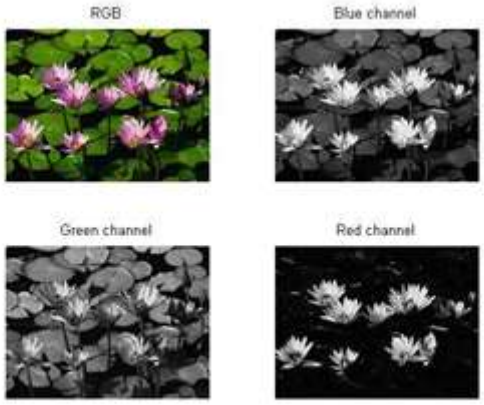
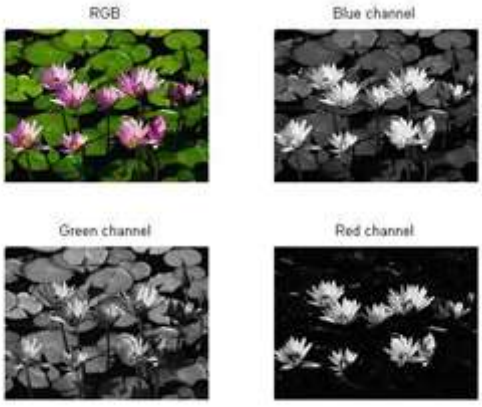


Рис. 5.4. Совмещенная графика *subplot*

Пример 4. *Графические фильтры изображения: размытие. Рассмотрим применение фильтра размытия по гауссовой матрице для серого изображения.*




```
%открытие файла изображения в матрицу A
My_lily= imread('lily.jpg');
%посмотрим картинку
imshow(My_lily);
%преобразуем картинку в оттенки серого
My_gray_lily=rgb2gray(My_lily);
%отобразим серую картинку в новом окне
figure;
imshow(My_gray_lily);
title('Изображение в оттенках серого';
%преобразуем серое изображение в формат с
плавающей запятой %из uint8
My_gray_lily_d=im2double(My_gray_lily);
%создаем «гауссову матрицу»
B=fspecial('Gaussian',5,5);
%свертка по матрице B
My_gray_lily_blur=conv2(My_gray_lily_d,B);
%вывод размытого изображения
figure;
```

```
imshow(My_gray_lily_blur);
title('Применен фильтр blur');
```

Пример 5. *Графические фильтры изображения: усиление резкости (sharpen).*

```
%открытие файла изображения в матрицу A
My_lily= imread('lily.jpg');
%посмотрим картинку
imshow(My_lily);
%выберем только красный канал
изображения My_lily_red=
My_lily(:,:,3);
%отобразим серую картинку в новом окне
figure;
imshow(My_lily_red);
title('Красный канал исследуемого фото');
%преобразуем серое изображение красного
канала в формат с %плавающей запятой
%из uint8
My_lily_rd=im2double(My_lily_red);
%создаем матрицу повышения резкости
S=[-1 -1 -1; -1 9 -1; -1 -1 -1];
%свертка по матрице B
Lily_rd_blur=conv2(My_lily_rd,S);
%вывод размытого изображения
figure;
imshow(Lily_rd_blur);
title('Применен фильтр sharpen');
```

5.3. Подстройка изображений в Octave (Matlab)

Иногда необходимо выполнить коррекцию изображения, сделать его светлее или темнее. Для того, чтобы это сделать в Octave/Matlab необходимо помнить, что матрица изображения содержит значения его спектральной яркости. Это значит, что коррекция изображения может означать поэлементное умножение матрицы на некоторый коэффициент n. Для того, чтобы сделать изображение темнее, надо умножить на коэффициент n<1, для того, чтобы осветлить изображение, коэффициент должен быть n>1.

```
%открытие файла изображения в матрицу A
My_lily= imread('lily.jpg');
%преобразуем в оттенки серого
My_lily=rgb2gray(My_lily);
%коэффициенты осветления и затемнения
n_d=0.5; n_l=1.8;
%затемним изображение
Ml_dark=My_lily*n_d;
%осветлим изображение
Ml_lighter=My_lily*n_l;
%вывод изображений и их гистрограмм
subplot(2,3,1); imshow(Ml_dark);
title('If darker');
subplot(2,3,2);
imshow(My_lily);
title('Initial image');
```

```
subplot(2,3,3);
imshow(Ml_lighter);
title('If lighter');
subplot(2,3,4); imhist(Ml_dark);
title('If darker');
subplot(2,3,5);
imhist(My_lily);
title('Initial image');
subplot(2,3,6);
imhist(Ml_lighter);
title('If lighter');
```

Рекомендации. В Matlab/Octave для вывода строковых сообщений используются следующие функции. Преобразование числа в строку, выполняемое в Scilab функцией *string(Число),* в Matlab/Octave требует применения функции *num2str(Число)*. Объединение строк, осуществляемое в Scilab оператором сложения, в Matlab/Octave требует strcat(Строка1, Строка2, …Строка n). При возникновении технических затруднений, необходимо читать справку по функциям, применяемым в Matlab/GNU Octave.

Прервать выполнение программы можно, нажав в командном окне комбинацию клавиш «Ctrl+C».

Начало координат изображения в Matlab/Scilab/Octave – левый верхний угол. Столбцы и строки графического изображения развернуты на 90 относительно графиков и диаграмм.

4 Задание для самостоятельной работы

Исправить коды примеров Matlab, найдя ошибки, препятствующие их нормальной работе. Указать в отчете, что и где было исправлено; Составить глоссарий применяемых в уроке функций Matlab, с кратким описанием;

Выполнить приводимые примеры, с внимательным изучением комментариев в коде, выполнить самостоятельные задачи; Основываясь на полученном опыте и литературном материале, выполнить самостоятельные задания;

В таблице вариантов найти координаты кадрирования исходного изображения. Кадрированное изображение, фрагмент файла **lily.jpg** будет использоваться для обработки в рамках своего варианта; Для кадрированного цветного изображения выполнить фильтры **blur** и **sharpen.** В качестве изображений можно использовать собственные.

На совмещенной графике **subplot** размером 4x4 клетки показать (1) исходное изображение фрагмента, (2) изображение в оттенках серого, (3) цветное изображение с фильтром blur, (4) цветное изображение с фильтром **sharpen**. Сохранить совмещенную графику в файл в виде изображения и заем вставить его в отчет; Для анализируемого фрагмента сделать два преобразования – затемнить и осветлить, для чего составить программу. По результатам выполнения этой программы вывести затемненное и осветленное серые изображения фрагмента и их гистограммы.

5. Контрольные вопросы

1. Почему кадрирование в коде выше выполняется как
My_lily_crop= My_lily(140:285,433:617,:);
а не как: My_lily_crop= My_lily(140:285,433:617);
2. В чем состоит принцип программирования в Octave?
3. Какие фильтры можно применять к изображениям?
4. Как сделать изображение светлее?
5. Приведите примеры фильтров резкости?

6. Что такое свертка с маской?
7. Как реализовать разложение изображения по каналам?

6. Содержание отчета

Результаты выполнения лабораторной работы оформляются в виде отчета, который должен содержать следующие разделы:

8. тема лабораторной работы;
9. цель работы;
10. общая часть и индивидуальный вариант задания;
11. описание алгоритма (граф-схема алгоритма);
12. текст программы;
13. результаты моделирования;
14. анализ результатов и выводы.

Работа №3 графика поверхностей в Scilab (Octave)

Цель работы: ознакомление со средствами и методами программирования вычисления функций для поверхностей.

Задача: провести анализ поверхности в соответствии с вариантом.

Представить результаты в виде трехмерных графиков с использованием графических функций. Подготовить отчет, сделать выводы.

Задание 1. Трехмерная графика (функции plot3d, mesh, surf, contour): ввести исходные данные, вычислить функцию, вывести функцию в виде трехмерных графиков разного типа.

Задание 2. Повторить задание 1 с отображением графиков в подокнах.

Варианты заданий.

№	Функция	Пределы изменения	
		x	y
1.	z=sin(x)cos(y)	от -2pi до 2pi	от -2pi до 2pi
2.	z=sin(x/2)cos(y)	от -2pi до 2pi	от -2pi до 2pi
3.	z=sin(2x)cos(y)	от -2pi до 2pi	от -2pi до 2pi
4.	z = sin(x)cos(y/2)	от -2pi до 2pi	от -2pi до 2pi
5.	z = sin(x/2)cos(2y)	от -2pi до 2pi	от -2pi до 2pi
6.	z = sin(2x)cos(2y)	от -2pi до 2pi	от -2pi до 2pi
7.	z = (1+sin(x)/x)(sin(y)/y)	от -2pi до 2pi	от -2pi до 2pi
8.	z = (sin(x)/x)cos(y)	от -2pi до 2pi	от -2pi до 2pi

Формирование задач. В работе предусмотрены 2 задачи, в каждой из которых вычисляется двумерная функция, описывающая объемную фигуру, и строятся поверхностные и контурные графики с использованием различных графических функций. В первой задаче каждый график выводится в свое окно, во второй в подокна общего окна.

Поверхностный и контурный графики. Для формирования поверхностного или контурного графика необходимо:

1. задать число точек по координатам X иY,
2. создать вложенные циклы по X и Y, вычислить функцию Z=f(X,Y),
3. ввести номер графического окна, вывести туда график выбранного типа.

Следует использовать графики:

1. трехмерный с аксонометрией, функция plot3(X,Y,Z),
2. трехмерный с функциональной окраской, функция mesh(X,Y,Z),
3. трехмерный с функциональной окраской и проекцией, функция surf(X,Y,Z),
4. контурный, функция contour(X,Y,Z),

Пример 1

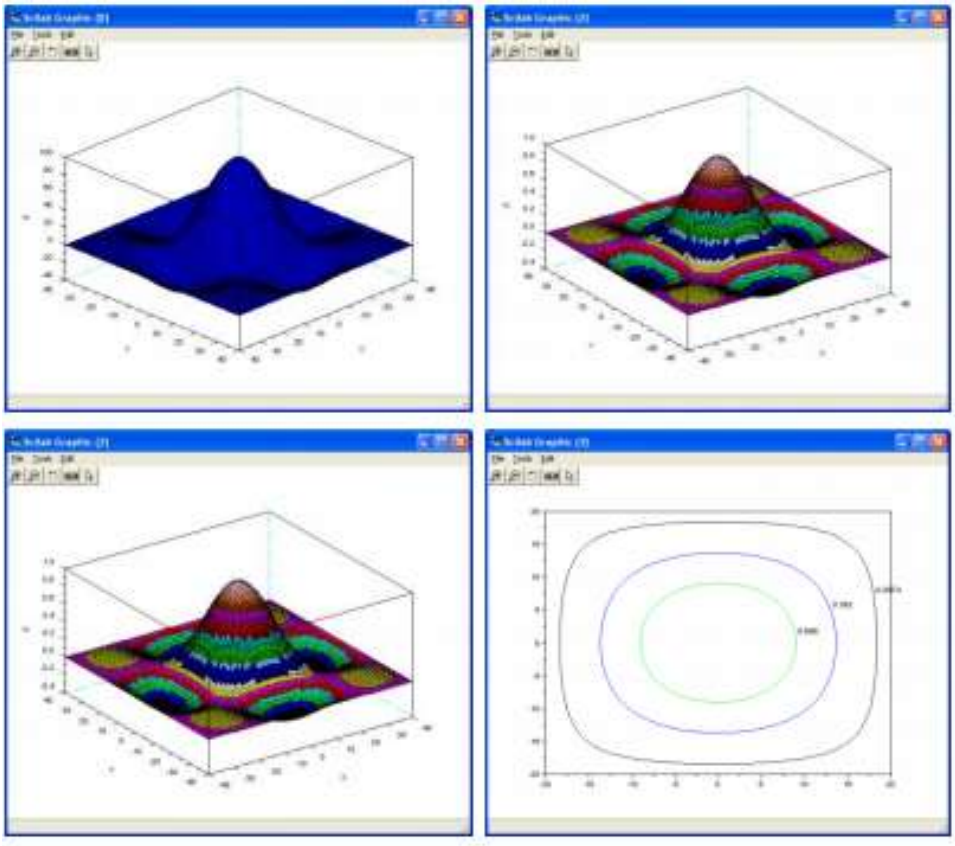
Задание Функция $z = \frac{\sin(x)}{x} \cdot \frac{\sin(y)}{y}$

Пределы изменения аргументов -2pi...2pi

Листинг программы

```
// Функции plot3d, mesh, surf, contour
N=40;
h=%pi/20;
// Расчет матрицы
for n=1:2*N+1
if n==N+1 A(n)=1; else A(n)=sin(h*(n-N-1))/(h*(n-N-1)); end;
end;
for n=1:2*N+1
        for m=1:2*N+1
                Z(n,m)=A(n)*A(m);
        end;
end;
// Задание площадки
X=-N:1:N;
Y=-N:1:N;
scf();
plot3d(X,Y,Z*100); // Окно 1. 3d график с монотонной окраской
scf();
mesh(X,Y,Z); // Окно 2. 3d график, каркас
scf();
surf(X,Y,Z); // Окно 3. 3d график с функциональной окраской
scf();
contour(X,Y,Z,3); // Окно 4. 3d график, контуры
```

Запуск программы командой Execute => Load Into Scilab. Результат – графики в отдельных окнах.

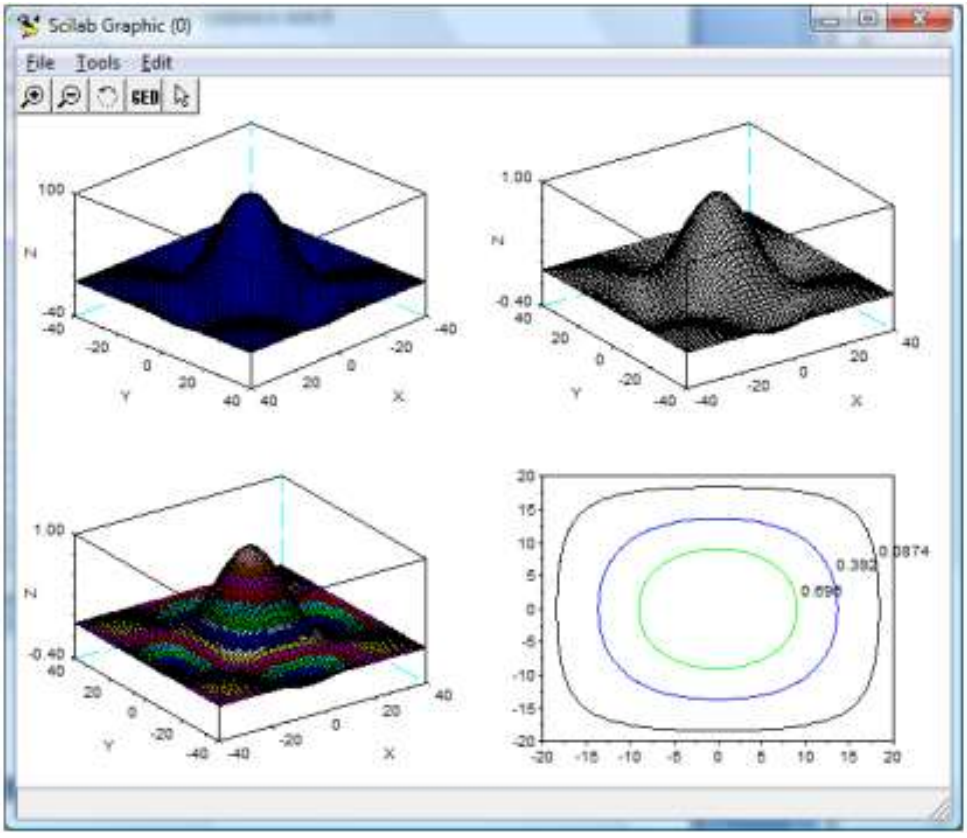


Пример 2

Листинг программы

```
// Функции plot3d, mesh, surf, contour. Используются подокна.
N=40;
h=%pi/20; 30
// Расчет матрицы
for n=1:2*N+1
if n==N+1 A(n)=1; else A(n)=sin(h*(n-N-1))/(h*(n-N-1)); end;
end;
for n=1:2*N+1
        for m=1:2*N+1
                Z(n,m)=A(n)*A(m);
        end;
end;
// Задание площадки
X=-N:1:N;
Y=-N:1:N;
subplot(2,2,1); // Подокно 1. 3d график с монотонной окраской
plot3d(X,Y,Z*100);
subplot(2,2,2); // Подокно 2. 3d график, каркас
mesh(X,Y,Z);
subplot(2,2,3); // Подокно 3. 3d график с функциональной окраской
surf(X,Y,Z);
subplot(2,2,4); // Подокно 4. 3d график, контуры
contour(X,Y,Z,3);
```

Запуск программы командой Execute => Load Into Scilab. Результат – графики в отдельных подокнах общего окна.



Контрольные вопросы

1. Как отобразить трехмерную поверхность в Octave?

2. Что делает функция surf?
3. Что делает функция contour?
4. Что делает функция subplot?
5. Как можно отобразить функцию, заданную точками?
6. Как получить срез поверхности?

Содержание отчета

Результаты выполнения лабораторной работы оформляются в виде отчета, который должен содержать следующие разделы:

1. тема лабораторной работы;
2. цель работы;
3. общая часть и индивидуальный вариант задания;
4. описание алгоритма (граф-схема алгоритма);
5. текст программы;
6. результаты моделирования;
7. анализ результатов и выводы.

Библиографический список

1. Ю. Тихомиров. Программирование трехмерной графики, BHV, СПб.– 1999 г.
2. http://freespace.virgin.net/hugo.elias/models/m_landsp.htm
3. www2.unil.ch/biomapper/opengl/Landscapes.html
4. Е.В. Шикин, А.В. Боресков. Компьютерная графика. Динамика, реалистические изображения.– М.: «Диалоги–Мифи», 1995 г.
5. Н. Томпсон. Секреты программирования трехмерной графики для Windows 95. СПб: Питер, 1997. – 352 с.
6. Д. Роджерс. Алгоритмические основы машинной графики. Пер. с англ. – М.: Мир, 1989. (2-ое издание – 2001 год).
7. Е.А. Никулин. Компьютерная геометрия и алгоритмы машинной графики.: БХВ-Петербург, 2003.– 560 с.
8. М.В. Михайлюк. Основы компьютерной графики.: ИТЦ «МАТИ», 2001.– 194 с.
9. Романычева Э.Т., Соколова Т.Ю., Шандурина Г.Ф. Инженерная и компьютерная графика.: ДМК Пресс, 2001.– 592 с.
10. Френсис Хилл. OpenGl. Программирование компьютерной графики. Для профессионалов.: Питер, 2002.– 1088 с.
11. 1. Поршнев С. В. Компьютерное моделирование физических процессов в пакете MATLAB / М.:, 2003. 593 с.
12. 2. Алексеев Е.Р. , Чеснокова О.В. Введение в Octave для инженеров и математиков: / Е.Р. Алексеев, О.В.Чеснокова М.: ALT Linux, 2012. 368 с.
13. Материалы по продуктам MATLAB & Toolboxes // [Электронный ресурс]: Математический сайт Exponenta.ru. Веб-сайт. URL: http://matlab.exponenta.ru/index.php