Дальневосточный федеральный университет
Инженерная школа

С.Л. Шевырев

# ИНФОРМАТИКА В НЕФТЕГАЗОВОЙ ОТРАСЛИ

Учебное электронное издание
Учебно-методическое пособие

Владивосток
Дальневосточный федеральный университет
2018

Дальневосточный федеральный университет
Инженерная школа

С.Л. Шевырев

# ИНФОРМАТИКА В НЕФТЕГАЗОВОЙ ОТРАСЛИ

Учебное электронное издание

Учебно-методическое пособие

Владивосток
Дальневосточный федеральный университет
2018

УДК 004:378.4
ББК 32.81р30
Ш38

Автор **Шевырев Сергей Леонидович,** к.г.-м.н.,
доцент кафедры нефтегазового дела и нефтехимии Инженерной школы
*Дальневосточный федеральный университет*

**Шевырев С.Л.** Информатика в нефтегазовой отрасли: учебно-методическое пособие [Электронный ресурс] / Инженерная школа ДВФУ. – Электрон. дан. – Владивосток: Дальневост. федерал. ун-т, 2018. – [70 с.]. – 1 CD. – Систем. требования: процессор с частотой 1,3 ГГц (Intel, AMD); оперативная память от 1 ГБ, Windows (XP; Vista; 7 и т.п.); Acrobat Reader, Foxit Reader либо любой другой их аналог.

ISBN 978-5-7444-4382-5

Рассматриваются основные вычислительные средства, необходимые для моделирования и обработки результатов лабораторных измерений, а также анализа изображений, применяемых в нефтегазовой отрасли. Приводимые методики могут быть также использованы для подготовки научных глав квалификационных и дипломных работ с применением современных систем научного программирования Scilab и GNU Octave (бесплатный аналог Matlab). Приводятся методы решения типовых задач научного программирования.

Рассматривается также распространяемая свободно географическая информационная система Quantum GIS (QGIS), ее способность к созданию схематических изображений местности и ситуационных планов, обработки аэрофото- и космоснимков.

Рекомендуется для студентов направления «Нефтегазовое дело», аспирантов, молодых ученых и преподавателей в областях естественных и инженерных наук.

*Ключевые слова*: информатика, научное программирование, транспортная задача, клеточные автоматы, обработка изображений, геоинформатика.

*Публикуется по решению кафедры нефтегазового дела и нефтехимии Инженерной школы ДВФУ*

Редактор Т.В. Рябкова
Верстка Г.П. Писаревой
Дизайн CD Г.П. Писаревой
Опубликовано: 30.10.2018

Формат PDF
Объем 3,3 МБ [Усл. печ. л. 8,1]
Тираж 30 экз.

Издание подготовлено редакционно-издательским отделом Инженерной школы ДВФУ
[Кампус ДВФУ, корп. С, каб. С 714]

Дальневосточный федеральный университет
690091, г. Владивосток, ул. Суханова, 8

Изготовитель CD: Дальневосточный федеральный университет
(типография Издательства ДВФУ
690091, г. Владивосток, ул. Пушкинская, 10)

ISBN 978-5-7444-4382-5

**Содержание**

# ВВЕДЕНИЕ

Практика современной научно-исследовательской работы предполагает усиливающуюся специализацию применяемых вычислительных аппаратов и инструментальных средств, а также усиливающуюся информатизацию исследовательского процесса. Это требует от студентов и молодых специалистов владения навыками использования ЭВМ для выполнения научной работы.

В качестве такого инструментария могут выступать среды научного программирования, позволяющие ученому самостоятельно разрабатывать приложения. От готовых решений такие программы отличаются оптимизацией и адаптацией к конкретному кругу задач. Кроме того, прилагаемые к использованию среды научного программирования Scilab и GNU Octave бесплатны для всех категорий пользователей.

Эффективность применения сред научного программирования к решению научных и научно-производственных проблем обусловливается наличием большого количества библиотек различных функций, как встроенных, так и загружаемых с интернет-сайта программ. Функции обеспечивают расширение возможностей программного обеспечения, включают средства статистического анализа, построения графиков и диаграмм, анализа изображений, методы применения средств искусственного интеллекта и многое другое. Программы Scilab и GNU Octave являются текстовыми интерпретаторами. Это означает, что их приложения сохраняются в текстовых файлах как сценарии, выполняемые построчно, и могут быть переданы и запущены на компьютерах, на которых установлены среды программирования.

Указанные среды программирования могут быть использованы для прототипирования, т.е. создания и отработки алгоритма информационной системы, реализуемой затем в средах программирования высокого уровня в виде самостоятельных приложений.

Настоящее учебно-методическое пособие, однако, не может рассматриваться как учебник или самоучитель для начального освоения средств научного программирования. Для этих целей можно рекомендовать официальное справочное сопровождение программы, а также книги из списка литературы.

# 1. ОБЩИЕ СВЕДЕНИЯ ОБ ИНТЕРПРЕТАТОРЕ SCILAB

При выполнении научных исследований перед студентами, аспирантами или молодыми учеными часто встает вопрос о выборе среды моделирования и автоматизации для выполнения научных расчетов.

Выбор визуальных специальных сред моделирования, часто дорогостоящих и требующих лицензирования, может содержать «подводные камни». Окончание действия лицензии в этом случае означает прекращение выполнения работы. Использование же нелицензионного софта неэтично и может привести к вопросам о его происхождении при подаче статьи, описывающей результаты его применения, в редакцию рецензируемого журнала.

В визуальных средах разработки зачастую используются модели типа «черный ящик». От конечного пользователя не требуется знаний о принципах их работы и устройстве. В результате такой деятельности формируется представительная модель, есть результаты, но часто нет понимания о работе узлов модели.

Еще один «подводный камень» – возможность прекращения поддержки разработчиками сред моделирования. Прекращение разработчиками поддержки и обновления программного обеспечения ставит в затруднительное положение специалистов, которые его использовали. Можно говорить, что программы визуального имитационного моделирования подойдут скорее ***производственникам***, для экономии ресурсов при замене вещественных моделей виртуальными, нежели молодым ученым.

По этой причине хотелось бы обратить внимание студентов на модели, создаваемые в средах научного программирования. В качестве примеров последних можно привести Matlab, GNU Octave и Scilab. Matlab – платное приложение, разрабатываемое компанией Mathworks, а две другие программы совершенно бесплатны. Язык программирования Matlab появился в 1970-х годах в университете Нью-Мексико. Бесплатная программа Octave практически полностью заимствует синтаксис Matlab. У Scilab, разработанной в 1994 г. в Национальном исследовательском институте информатики и автоматизации (INRIA, Франция) и Национальной школе дорожного ведомства, есть некоторые отличия, но в целом они довольно похожи.

Достоинством приведенных сред является возможность сразу начать программирование с небольшой подготовкой или без оной, толерантность программ к погрешностям синтаксиса. Программа набирается в виде текстового файла (есть собственный редактор) и исполняется построчно.

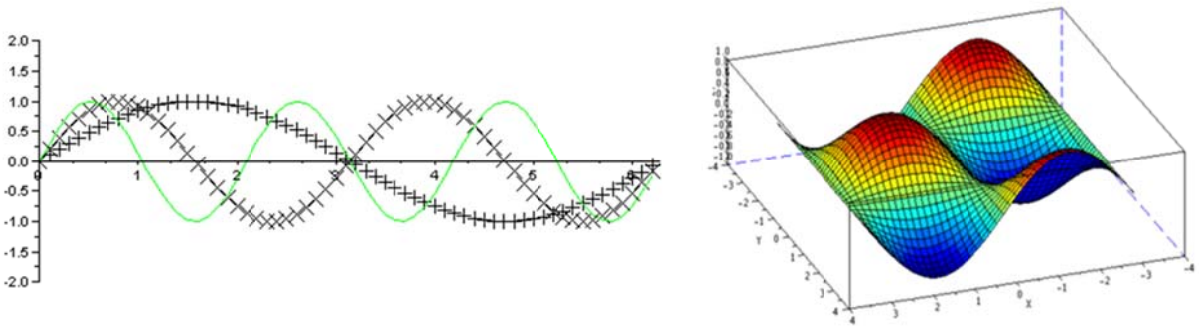

Рис. 1. Пример графиков и поверхностей в Scilab

Возможности программы Scilab: анализ данных, визуализация – построение графиков, диаграмм, поверхностей (рис. 1). Преимущества перед программами своего класса

(Matlab и т.п.): бесплатность, небольшой размер дистрибутива (100 Мб против более 2 Гб у Matlab). Преимущества перед другими средствами анализа данных (например, Statistica): высокая скорость работы, контроль выполнения программы не ограничен графическим интерфейсом.

Поскольку Scilab – язык-интерпретатор команд, программа на нем выполняется медленнее, чем в языках-компиляторах (Delphi, Pascal, Fortran). Однако реализованы отладка программы во время выполнения, открытость кода, возможность работы там, где есть Scilab, возможность транслирования из Matlab в Scilab.

Для установки Scilab на компьютер, работающий под управлением операционных систем Windows и Macintosh, надо скачать дистрибутив с сайта Scilab.org. Для установки программы на компьютер Linux необходимо набрать в окне консоли:

*sudo apt-get install scilab*

Для запуска программы Scilab в Windows надо щелкнуть левой кнопкой мыши соответствующий ярлык в меню «Пуск» → scilab-5.4.0 → scilab-5.4.0.exe. Вызов программы приведет к появлению командного окна (рис. 2). Командное окно имеет боковые панели: «обозреватель файлов», «обозреватель переменных», «журнал команд».

Для работы и ввода команд можно использовать собственно командное окно. Наряду с ним можно использовать текстовый редактор SciNotes, предназначенный для написания и открытия программ SciLab (текстовые файлы с расширением *.sce и *.sci).

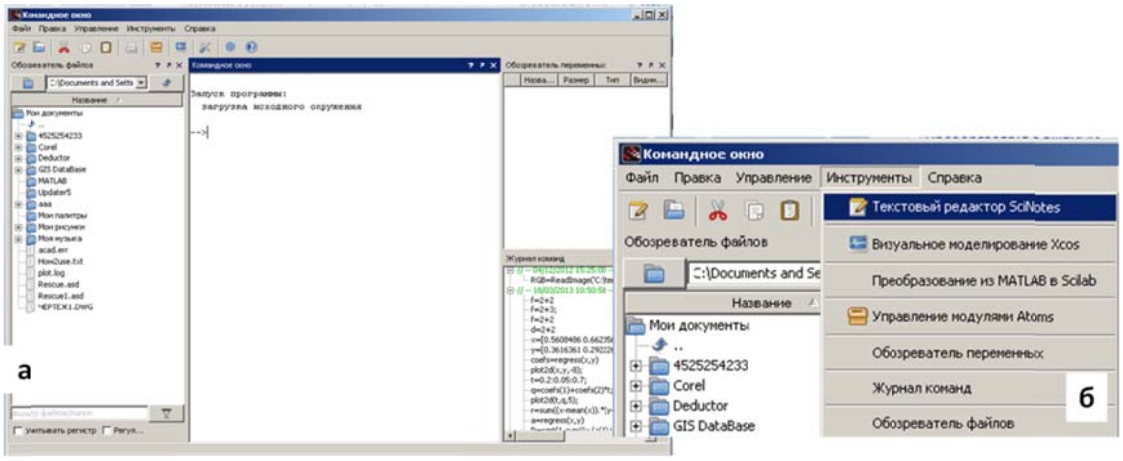


Рис. 2. Командное окно Scilab (а) и вызов редактора Scinotes (б)

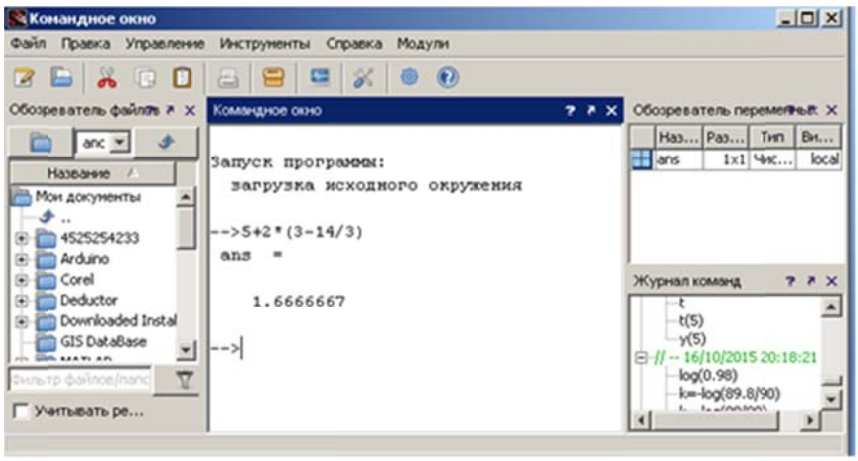


Рис. 3. Вычисление выражения в командном окне Scilab и его результат

Ввод команд Scilab осуществляется в командном окне, а написание программ – в редакторе кода. В случае, если после команды следует «;», вывод результата выполнения команды блокируется. Рис. 3 демонстрирует ввод выражения в окне Scilab и вывод результата в виде служебной временной переменной *ans*, т.е. использование Scilab в качестве инженерного калькулятора.

**Операторы Scilab:**

| | | |
|---|---|---|
| + сложение | – вычитание | * умножение |
| / деление | ^ степень | \ деление (справа налево) |
| = присваивания | < меньше | > больше |

Если в сценарий Scilab мы хотим добавить поясняющий текст (он называется комментарием) или сделать невыполняемыми некоторые строки, то перед ними в Scilab пишут // (а в Matlab знак %). Такой код игнорируется интерпретатором (рис. 4).

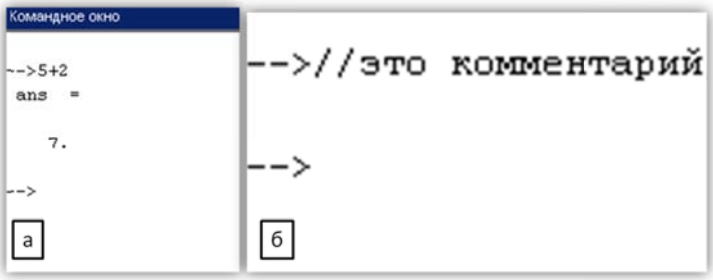


Рис. 4. Выражение в Scilab и текст комментария

Для хранения информации и совершения операций в Scilab используются переменные. Объявить переменные можно непосредственно в командном окне программы:

```
Fluid_density = 1.2; // читаемо
fluidDensity = 1.2 // "венгерская" запись
i=1; // переменная-буква подойдет для временных
данных, счетчиков цикла и т.д.
word = 'User '; // данные типа String
```

Также нужно помнить о некоторых правилах объявления переменных:

- переменные начинаются с латинских букв;
- запрещены операторы;
- символы пунктуации и пробелы;
- case sensitive (символы в разном регистре – разные символы).

Главная цель, которой следует придерживаться при именовании переменных, – хорошая читаемость кода.

Удаление переменных необходимо для очистки оперативной памяти и ускорения работы программы. Для удаления переменных в командном окне Scilab используется команда *clear var_name*, где *var_name* – переменная для удаления:

```
clear Fluid_density  //удалил одну переменную

clear fluidDensity  i word          // удалил три
```

Наряду с *пользовательскими переменными* существуют системные переменные Scilab, которые содержат часто используемые постоянные. Ввод системных переменных Scilab предваряется знаком «%»:

```
%i      //мнимая единица
%pi     //число π=3.141592653589793
%e      //число e=2.7182818
%inf    //машинный символ бесконечности (∞)
%NaN    //неопределенный результат(0/0, ∞/∞ и т.п.)
%eps    //условный ноль %eps=2.220E-16
```

Кроме системных переменных Scilab содержит *функции*. Функция – последовательность действий над передаваемыми в скобках переменными. Функции могут быть предустановленными *системными* и *пользовательскими* (созданными пользователем с целью автоматизации). Поиск нужной функции и порядка ее вызова осуществляется в справке Scilab (клавиша F1). Наиболее употребляемые системные функции Scilab представлены в табл. 1.

Таблица 1

**Наиболее употребляемые системные функции Scilab**

| Функция | Описание функции |
|---|---|
| *Тригонометрические* | |
| sin(x) | Синус числа x |
| cos(x) | Косинус числа x |
| tan(x) | Тангенс числа x |
| cotg(x) | Котангенс числа x |
| asin(x) | Арксинус числа x |
| acos(x) | Арккосинус числа x |
| atan(x) | Арктангенс числа x |
| *Экспоненциальные* | |
| exp(x) | Экспонента числа x |
| log(x) | Логарифм числа x |
| *Другие* | |
| sqrt(x) | Квадратный корень числа x |
| abs(x) | Модуль числа x |
| log10(x) | Десятичный логарифм от x |
| log2(x) | Логарифм по основанию 2 от x |

**Применение Scilab для вычисления выражений**

Наиболее простое применение Scilab для вычисления выражений реализуется в командной строке. Вычислим выражение:

$$z = \sqrt{|\sin\frac{x}{y}|}e^{x^y}$$

для x = 1,2; y = 0,3.

Стоит обратить внимание, что в Scilab разделитель целой и дробной части не «,», а «.» независимо от системных настроек операционной системы. Для вычисления выражения выше введем в командном окне построчно:

```
x=1.2;y=0.3;
z=sqrt(abs(sin(x/y)))*exp(x^y)
```

В результате Scilab выведет (рис. 5):

```
z  =
2.5015073
```

В этом примере использованы предустановленные функции sqrt(), abs(); sin(), exp():

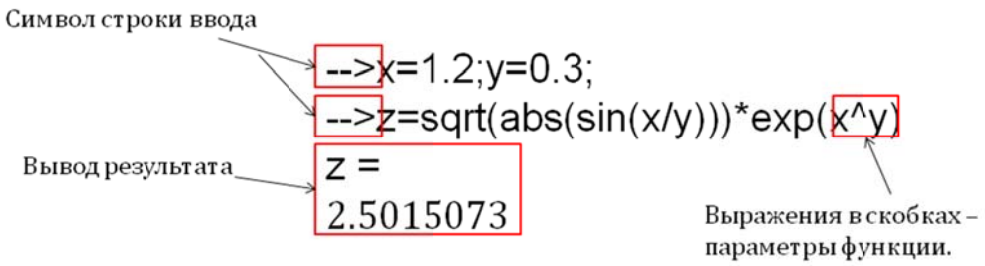


Рис. 5. Вычисление значения выражения в Scilab

Кроме того, вычислить выражение возможно, объявив для этого пользовательскую функцию. Это можно сделать двумя способами – в командном окне и в редакторе сценариев.

1-й способ. Объявление функций с помощью функции deff(), что означает “define function” – объявить функцию. Синтаксис:

```
deff('[имя1,…,имяN] =
имя_функции(переменная_1,…,переменная_M)',
'имя1=выражение1;…;имяN=выражениеN')
```

Пример объявления функции:

```
deff('z=fun1(x,y)','z=sqrt(abs(sin(x/y)))*exp(x^y)');
x=1.2;y=0.3;z=fun1(x,y)
```

Также можно после объявления передать параметры функции fun1.

2–й способ. Объявление в редакторе Scinotes (рис. 6). Необходимо создать файл сценария программы – текстовый файл с расширением имени *.sce. SciNotes подсвечивает синтаксис, выделяет предполагаемые ошибки:

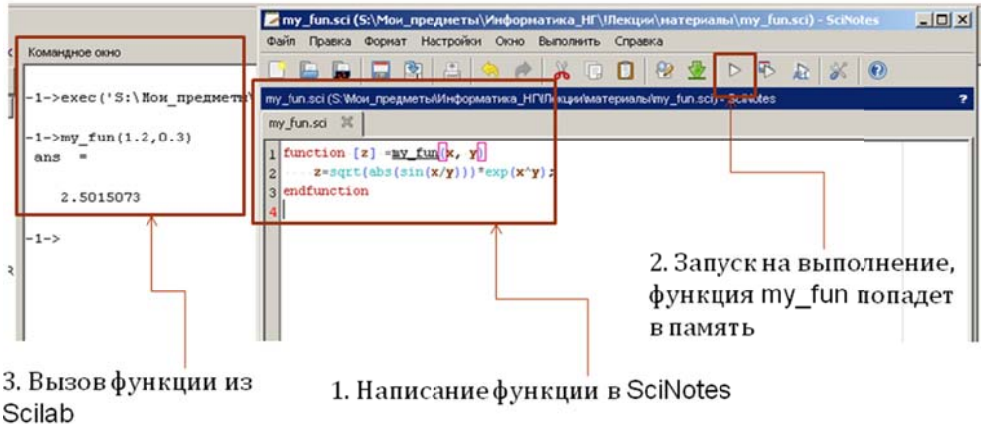


Рис. 6. Последовательность объявления и вызова функции в Scilab

Второй способ является более предпочтительным при объявлении более сложных функций, особенно если их планируется использовать неоднократно. Для объявления функции нужно создать отдельный файл сценария. Пример объявления функции в Scilab:

```
function [z] =my_fun(x, y)
    z=sqrt(abs(sin(x/y)))*exp(x^y);
endfunction
```

После сохранения и запуска этого файла функцию можно вызвать из командного окна, указав в скобках с параметры ввода:

```
my_fun(1.2,0.3)
```

В результате переменной *ans* будет присвоен ответ, который тут же будет выведен в командном окне.

```
-->ans = 2.5015073
```

Для начала работы с данными необходимо понимание различий между данными и информацией. *Данные* являются результатом инструментальных наблюдений и измерений, представляя собой исходный материал для получения новых знаний. *Информация* – знание, т.е. результат анализа и осмысления данных. Данные в средах научного программирования Scilab и Octave (Matlab) представлены «обычными» переменными и массивами (векторами и матрицами).

Простейшее устройство получения данных – температурный датчик, подключенный к программируемому микроконтроллеру *Arduino*. Такое устройство собирает и отправляет на компьютер через COM-порт напряжение с датчика, пересчитанное в значение температуры, и время выполнения измерений.

Данные в средах научного программирования представлены в виде массивов, т.е. векторов и матриц.

*Массив* – последовательность данных какого-либо типа (численного или символьного). Используется вместо объявления многих переменных. Массив имеет имя, а данные в нем – порядковый номер, или индекс.

```
//объявление одномерного массива, содержащего строковые значения
names = ['Mikhail', 'Anna', 'Boris', 'Han Li'];
```

Массивы объявляются в квадратных скобках “[ ]”, элементы указываются через запятую или пробел. Нумерация элементов (индексы массива) начинается с 1. Для обращения к элементу № 1 массива *names* нужно указать его в круглых скобках после имени массива:

```
//вызов первого элемента массива
-->names(1)  //names - имя массива, 1 в скобках - порядковый номер
ans  =
 Mikhail
```

Одномерные (т.е. содержащие только одну строку или столбец) массивы носят название векторов. Создавать массивы можно несколькими путями: с помощью поэле-

ментного ввода, указанием начального и конечного элемента и шага увеличения значения, а также с помощью специальных функций.

1. Объявление массива с помощью поэлементного ввода (как было показано на примере names):

```
//ввод  вектора строки с значениями от 1 до 5
My_array=[1 2 3 4 5]
```

В результате выполнения кода в примере выше будет создан массив *My_array*, содержащий целые числа от 1 до 5.

2. Объявление массива с указанием его начального и конечного элементов и шага между ними.

```
//ввод  вектора строки с значениями от 1 до 5
My_array=1:2:10
```

В результате такого ввода в командном окне появится:

```
-->My_array =
1.  3.  5.  7.  9.
```

3. Объявление массива с помощью специальных функций. Например, функция *rand (2,3)* создаст в Scilab матрицу случайных чисел от 0 до 1 размером 2 строки на 3 столбца. Такая матрица называется двумерной. Ввод в командном окне создаст соответствующую матрицу:

```
--> A=rand(2,3)
A  =
0.2113249   0.0002211   0.6653811
0.7560439   0.3303271   0.6283918
```

Одномерные матрицы, или векторы, разделяются на матрицы-строки, аналогичные объявленным выше *names* и *My_array* и матрицы-столбцы.

```
--> A=rand(2,3)
A  =
0.2113249   0.0002211   0.6653811
0.7560439   0.3303271   0.6283918
--> //Массив-строка
--> String_arr=[1 2 3 4 5] //объявление
--> //Вывод Scilab:
--> String_arr =
--> 1.  2.  3.  4.  5.
```

Матрицы-столбцы объявляются аналогично:

```
--> //Массив-столбец:
--> Column_arr=[1; 2; 3]   //объявление
Column_arr  =
1.
2.
3.
```

**Обработка экспериментальных данных в Scilab**

Сведения о природных объектах и промышленных образцах получаются в ходе замеров (опробования) (рис. 7). В качестве объекта, показанного на рисунке, может выступать пятно нефтяного загрязнения, образец вещества или материала и т.д.

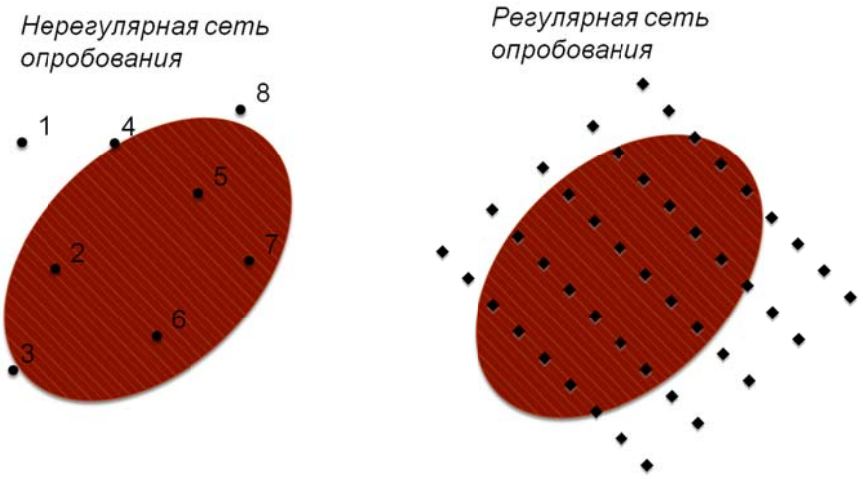


Рис. 7. Схемы регулярного и нерегулярного опробования

Исходные данные наблюдений часто представлены в виде таблиц, разделяющихся на строки (единичные замеры) и столбцы (измеренные параметры) (рис. 8).


| _ID | X_UTM | Y_UTM | Pl | Int | $D_m$ | Is |
|---|---|---|---|---|---|---|
| 1 | 15002,0000 | 5019,0000 | 0,0101 | 0 | 1,0989867 | 0 |
| 2 | 15012,0000 | 5019,0000 | 0,0573 | 0 | 1,1905419 | 0 |
| 3 | 15022,0000 | 5019,0000 | 0,0801 | 0 | 1,1184548 | 0 |
| 4 | 15032,0000 | 5019,0000 | 0,1807 | 1 | 1,1152221 | 0 |
| 5 | 15042,0000 | 5019,0000 | 0,3228 | 0 | 1,1564515 | 0 |
| 6 | 15052,0000 | 5019,0000 | 0,3872 | 1 | 1,1244395 | 1 |
| 7 | 15062,0000 | 5019,0000 | 0,4864 | 5 | 1,1228726 | 23 |
| 8 | 15072,0000 | 5019,0000 | 0,3716 | 2 | 1,1632857 | 8 |
| 9 | 15082,0000 | 5019,0000 | 0,4693 | 2 | 1,1322016 | 9 |
| 10 | 15092,0000 | 5019,0000 | 0,3290 | 0 | 1,1225641 | 10 |
| 11 | 15102,0000 | 5019,0000 | 0,3736 | 3 | 1,1097711 | 52 |

Рис. 8. Табличное представление экспериментальных данных

Для задания такой таблицы в Scilab надо применять двумерные массивы, иначе называемые *матрицами*.

Ввод элементов матрицы также осуществляется в квадратных скобках, при этом элементы строки отделяются друг от друга пробелом или запятой, а строки разделяются между собой точкой с запятой:

$$Name = [x_{11}, x_{12}, \ldots, x_{1n}; x_{21}, x_{22}, \ldots, x_{2n}; \ldots; x_{m1}, x_{m2}, \ldots, x_{mn}] ;$$

Обратиться к элементу матрицы можно, указав после имени матрицы в круглых скобках через запятую номер строки и номер столбца, на пересечении которых элемент расположен (рис. 9)


Рис. 9. Обращение к элементам массива по индексам

Для работы с элементами массива необходимо указание индексов в скобках (после имени массива). Объявим матрицу и произведем действия с отдельными ее элементами.

```
//Объявление матрицы
A=[1 2 3;4 5 6;7 8 9]
```

В результате Scilab выведет:

```
-->A = 1. 2. 3.
    4. 5. 6.
    7. 8. 9.
```

Выполнение действия с элементами матрицы, как с числами:

```
-->A(1,2)^A(2,2)/A(3,3)
 ans  =
   3.5555556
```

В результате этих действий Scilab вычислил результат и присвоил его временной переменной *ans*.

Объединение матриц носит название «конкатенация». Она может быть горизонтальной (построчно) и вертикальной (по столбцам).

Рассмотрим пример горизонтальной конкатенации строк:

```
-->//объявление матриц v1, v2, v3
-->v1=[1 2 3];
-->v2=[4 5 6];
-->v3=[7 8 9];
-->//Горизонтальная конкатенация векторов-строк:
-->V=[v1 v2 v3]
```

В результате выполнения кода выше Scilab выведет:

```
-->V = [v1 v2 v3]
 V  =
    1.   2.   3.   4.   5.   6.   7.   8.   9.
```

Вертикальная конкатенация представлена ниже:

```
-->//Вертикальная конкатенация векторов-строк:
 -->V=[v1; v2; v3]
 V  =
   1.   2.   3.
   4.   5.   6.
   7.   8.   9.
```

Для выбора части матрицы, подматрицы или отдельного ряда применяется символ «:». Указывая его вместо индекса при обращении к массиву, можно получать доступ к группам элементов.

Исходный код:

```
-->select1=V(:,1)  //Выбор 1-го столбца в переменную select1
 select1  =
   1.
   4.
   7.
-->select2=V(1,:)  //Выбор 1-й строки
 select2  =
    1.   2.   3.
```

Для выбора части матрицы (субматрицы) необходимо указать диапазоны по рядам и столбцам:

```
-->select3=V(2:3,1:2) //Выбор части матрицы (обрезка)
 select3  =
  4.   5.
  7.   8.
```

Действия между матрицами возможны при совпадении их размерностей, они выполняются с помощью операторов:

```
  + – сложение
  - – вычитание
  ‘ – транспонирование
  * – матричное умножение (умножение на число)
  ^ – возведение в степень
  \ – левое деление
  / – правое деление
   .* – поэлементное умножение
  .^ – поэлементное возведение в степень
  .\ –поэлементное левое деление
./ – поэлементное правое деление
```

Выполнение приведенных операций возможно при совпадении размерности матриц, т.е. равном количестве строк и столбцов. На матричных операциях основаны многие графические фильтры, приемы обработки изображений и т.д.

Существуют также специальные матричные функции, служащие целям преобразования и создания специальных матриц: *matrix(A,n,m)*. Эта функция преобразует матрицу A в матрицу другого размера, n – количество строк, m – количество столбцов. Пример выполнения:

```
—>D=[1 2;3 4;5 6];
—>matrix(D,2,3)
ans = 1. 5. 4. 3. 2. 6.
—>matrix(D,3,2)
ans = 1. 2. 3. 4. 5. 6.
—>matrix(D,1,6)
ans = 1. 3. 5. 2. 4. 6.
—>matrix(D,6,1)
ans = 1. 3. 5. 2. 4. 6.
```

Функции ones(n,m) и zeros(n,m) используются для создания матриц нулей и единиц:

```
-->ones(1,3) //Формируется вектор-строка
 ans = 1. 1. 1.
-->ones(2,2) //Формируется квадратная матрица
ans = 1. 1. 1. 1.
-->m=3; n=2;
-->zeros(3,2)
 ans = 0. 0. 0. 0. 0. 0.
 -->M=[1 2 3 4 5];
-->Z=zeros(M)
 Z= 0. 0. 0. 0. 0.
```

# 2. ОПРЕДЕЛЕНИЕ ОСНОВНЫХ ЧИСЛОВЫХ ХАРАКТЕРИСТИК СОВОКУПНОСТИ СЛУЧАЙНЫХ ВЕЛИЧИН

При определении свойств нефтепродуктов и природного газа, как правило, изучается совокупность случайных величин, которая может быть определена основными числовыми характеристиками: **средним, дисперсией, коэффициентом вариации, квадратической неровнотой** и другими.

Числовые характеристики меняются от одной выборки к другой и являются случайными величинами, изменяющимися с некоторой доверительной вероятностью в определенном интервале. Чем больше погрешность при определении числовой характеристики, тем шире этот интервал. Точность в определении конкретной численной характеристики определяется **ошибкой**, а надежность определения – **доверительной вероятностью**. При известной дисперсии случайной величины для требуемых точности и надежности можно найти **доверительный объем** измерений оценки числовой характеристики.

## 2.1. Получение совокупности случайных величин

Для ознакомления с методикой определения основных числовых характеристик совокупности случайных величин необходимо получить данную совокупность. Она может быть получена применением лабораторного измерительного и аналитического оборудования, полевыми наблюдениями и т.д. (рисунки 10, 11).

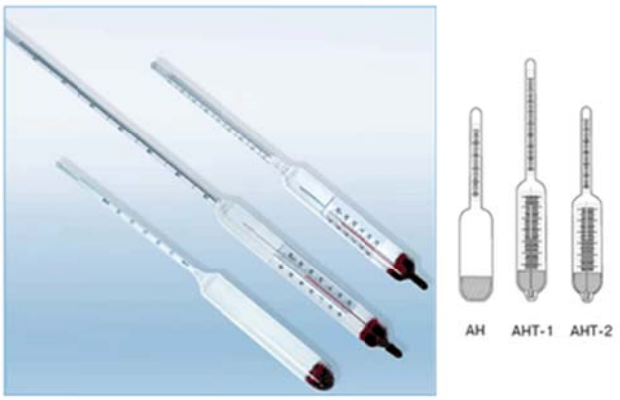


Рис. 10. Ареометры для лабораторного контроля плотности

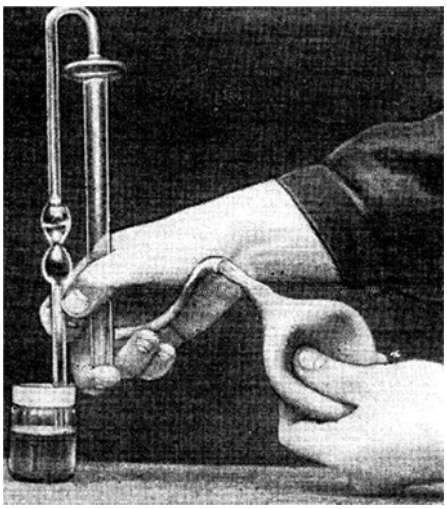

Рис. 11. Прибор для определения кинематической вязкости – вискозиметр Пинкевича

Для выполнения исследования воспользуемся совокупностями наблюдений, приведенными в приложении I.

### 2.2. Оценка математического ожидания, дисперсии и среднего квадратического отклонения

Математическое ожидание *M(x)*, называемое также средним значением, определяет центр распределения случайных величин, около которого группируется большая их часть. Характеристиками рассеяния случайной величины *x* около центра распределения *M(x)* является дисперсия $S^2(x)$ и среднее квадратическое (стандартное) отклонение *S(x)*.

Вычисление значений **математического ожидания, дисперсии и среднего квадратического отклонения** для исследуемой выборки производится по формулам:

$$M(x) = \frac{1}{n}\sum^{n} x_i \tag{1}$$

$$S^2(x) = \frac{1}{n-1}\sum_{i=1}^{n}(x - M(x))^2 \tag{2}$$

$$S(x) = \sqrt{S^2(x)} \tag{3}$$

### 2.3. Определение выбросов выборки экспериментальных данных

Случается, что выборка экспериментальных данных включает значения, резко и существенно выделяющиеся относительно других. Причины появления таких значений различны: ошибка оборудования, нарушение технологии проведения эксперимента исследователем и др. Для получения адекватных результатов такие значения необходимо исключать из дальнейшей обработки, т.к. они могут существенно исказить числовые характеристики рассматриваемого объекта. Однако необоснованное исключение значений также влияет на числовые характеристики.

Если существует такая возможность, необходимо проанализировать условия эксперимента (замера), при которых данные были получены. Если выясняется, что условия не соответствовали стандарту, то данные следует исключить из выборки независимо от их значений. Однако бывает, что условия эксперимента невозможно или затруднительно определить, тогда применяется статистический метод исключения значений данных.

Примером такой оценки является метод, основанный на анализе критерия Смирнова-Граббса. Он выполняется в следующем порядке:

1. В выборке находят минимальные *min(x)* и максимальные *max(x)* значения.

2. Для минимальных и максимальных значений находят расчетные значения критерия Смирнова–Граббса по формулам (4) и (5):

$$G_{\max} = \frac{\max(x) - M(x)}{S(x)} \cdot \sqrt{\frac{n}{n-1}}; \tag{4}$$

$$G_{\min} = \frac{M(x) - \min(x)}{S(x)} \cdot \sqrt{\frac{n}{n-1}}. \tag{5}$$

3. Затем полученные значения сравниваются с табличным значением G критерия Смирнова-Граббса (приложение II). В случае, если $G_{max}$ или $G_{min}$ больше G, то соответ-

ствующее минимуму или максимуму значение $x$ необходимо исключить из экспериментальных данных. Затем следует повторить расчет оценок $M(x)$, $S^2(x)$ и $S(x)$.

Эти действия повторяют до полного исключения резко выделяющихся значений из экспериментальных данных.

### 2.4. Относительные характеристики рассеяния случайной величины

Относительной характеристикой рассеяния случайной величины является коэффициент вариации $CV(x)$.

$$CV(x) = \frac{S(x)}{M(x)}. \tag{6}$$

*Квадратической неровнотой* $C(x)$ называется выражение значения $CV(x)$ в процентах.

$$C(x) = \frac{S(x)}{M(x)} \cdot 100\%. \tag{7}$$

### 2.5. Погрешности измерений и границы доверительного интервала

В результате измерений исследуемого параметра неизбежно возникают ошибки измерений, для описания которых введены оценки абсолютной $\varepsilon(x)$, и относительной $\delta(x)$ погрешностей. Абсолютная и относительная доверительные ошибки, допущенные при оценке математического ожидания, определяются по формулам:

$$\varepsilon(x) = \frac{2 \cdot S(x)}{\sqrt{n}}; \tag{8}$$

$$\delta(x) = \frac{4 \cdot S(x)}{n}. \tag{9}$$

Двусторонний доверительный интервал покрывает распределение измеряемой величины с установленной доверительной вероятностью $P_x$:

$$M(x) - \varepsilon(x) \le M(x) \le M(x) + \varepsilon(x). \tag{10}$$

В практике лабораторных исследований при статистической обработке принимают доверительную вероятность, выраженную в долях единицы, $P_x = 0{,}95$. Также выделяют уровень значимости, вычисляемый как $\alpha = 1 - P_x$. Доверительная вероятность и уровень значимости также выражаются в процентах.

### 2.6. Доверительный объем испытаний

Рассматривая точность оценки среднего значения можно сделать вывод о достаточности объема измерений. При этом, установив значение относительной ошибки $\delta(x)=3\%$ и приняв квадратическую неровноту $C(x)$ по данным произведенных ранее расчетов, можно рассчитать доверительный объем выборки $n_d(x)$:

$$n_d(x) = \left( \frac{u(P_x) C(x)}{\delta(x)} \right)^2. \tag{11}$$

В выражении (11) величина $u(P_x)$ называется *квантилем нормального распределения случайной величины*. При доверительной вероятности $P_x = 0{,}95$ значение квантиля нормального распределения $u(P_x) = 1{,}65$. В прочих случаях следует воспользоваться таблицами квантилей.

## 2.7. Определение числовых характеристик объема испытаний в Scilab (Matlab, Octave)

При составлении программы обработки данных в Scilab необходимо руководствоваться некоторыми соображениями:

- После начала ввода данных Scilab никаких действий пользователя по подготовке/обработке данных вне программы не допускается (кроме реализации диалоговых команд input()).
- Статистические формулы можно заменить встроенными функциями Scilab.
- Исходные данные варианта могут быть подготовлены в формате csv/txt.
- Для каждого подпункта удобно готовить отдельную программу Scilab.

В процессе работы при фильтрации данных с помощью критерия Смирнова–Граббса необходимо будет сохранить новую таблицу в формате csv.

Некоторые применимые при написании программы решения в Scilab рассмотрены ниже.

*Листинг 1. Чтение данных файла test.csv в матрицу*

```
my_dir='C:/Labour_folder/';
  filename = fullfile(my_dir , 'test.csv');
// ; -разделитель ячеек , - делитель разрядов, double – формат данных
 my_mat=csvRead(filename, ';',',', 'double');
//результат – матрица с данными
```

*Листинг 2. Запись данных матрицы в файл csv*

```
my_dir=' 'C:/Labour_folder/';

 CSV = ["1,0.3";
 "2,0.5";
 "3,0.2";
 "4,0.6";
 "5,0.2";
 "6,0.1";
 "7,0.3";
 ];
 filename = fullfile(my_dir , 'test.csv');
 mputl(CSV, filename);
```

**Важное замечание!** В Scilab версии 5.3 функция *readCSV* работать не будет. Вместо нее необходимо использовать *read_csv('путь_к_файлу', 'разделитель')*. Подробнее необходимо читать официальную справку.

Пример вызова:

*Листинг 3. Вызов функции чтения таблицы csv в Scilab 5.3*

```
my_dir='C:/Labour_folder/';
  filename = fullfile(my_dir , 'test.csv');
// ; -разделитель ячеек , - делитель разрядов, double – формат данных
 my_mat= read_csv (filename, ';' );
```

**Требования к отчету**

Отчет о выполнении лабораторной работы должен содержать:

– тему и цель лабораторной работы на титульной странице;

– оглавление;

– необходимые теоретические сведения по теме;

– исходную совокупность случайных величин (по заданию преподавателя);

– поэтапный расчет основных числовых характеристик для заданной совокупности случайных величин в виде вставок кода Scilab в объектах «Подпись»;

– результаты выполнения кода Scilab в виде вывода в командном окне;

– интерпретацию результатов расчета основных числовых характеристик для заданной совокупности случайных величин;

– отметку преподавателя о выполнении лабораторной работы.

# 3. РЕШЕНИЕ ЗАДАЧ ОПТИМИЗАЦИИ ТРАНСПОРТА НЕФТЕПРОДУКТОВ

Суть *транспортной задачи* (ТЗ) – минимизация полной стоимости распределения (транспортировки) бензина с нефтебаз на несколько автозаправочных станций (АЗС) в соответствии с существующей потребностью при различном наличии топлива и стоимости доставки до определенных потребителей.

Кроме **описания хода решения**, ответом является **указание объемов бензина, перевозимого с каждой нефтебазы на каждую АЗС**, и **общие затраты** на транспортировку всего объема топлива.

## 3.1. Основные сведения

Стандартная ТЗ определяется как *задача разработки наиболее экономичного плана перевозки продукции одного вида из нескольких пунктов отправления в пункты назначения*. При этом величина транспортных расходов прямо пропорциональна объему перевозимой продукции и задается с помощью тарифов на перевозку единицы продукции. В практике логистики надо определить количество продукции (в нашем случае тонн топлива), перевозимого с отдельной нефтебазы на одну из АЗС для полного покрытия потребности в топливе (рис. 12).

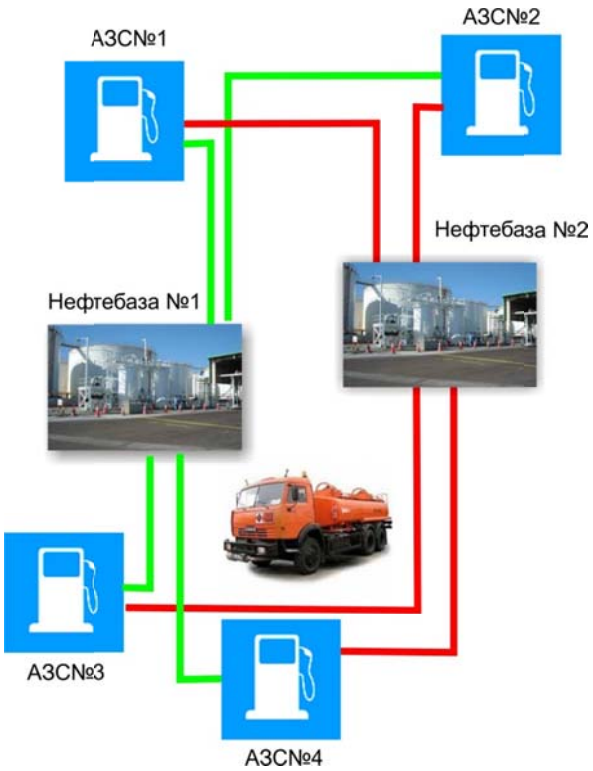


Рис. 12. Сущность транспортной задачи – определение источника и пункта назначения товарной продукции

Входные параметры модели решения ТЗ:

- n – количество пунктов отправления (нефтебаз);
- m – количество пунктов назначения (АЗС);
- $a_i$ – запас продукции (топлива) в пункте отправления $A_i$ ($i = \overline{1,n}$) [единиц товара, т];

- $b_j$ – спрос на продукцию (топливо) в пункте назначения $B_j$ ($j=\overline{1,m}$) [единиц товара, т].
- $c_{ij}$ – транспортный тариф или стоимость перевозки единицы продукции из пункта отправления $A_i$ в пункт назначения $B_j$ [руб./т], зависит от расстояния и способа транспортировки.

Выходные параметры модели решения ТЗ:

- $x_{ij}$ – количество продукции, перевозимой из нефтебазы $A_i$ на АЗС $B_j$ [т];
- $C_\Sigma$ – транспортные расходы на перевозку всей продукции [руб.].

**Стадийность построения модели** транспортной задачи:

- определение переменных;
- проверка сбалансированности задачи;
- построение сбалансированной транспортной матрицы;
- задание целевой функции;
- задание ограничений.

Модель транспортной задачи может быть представлена в следующем виде:

$$C_\Sigma = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \to \min;$$

$$\begin{cases} \sum_{j=1}^{m} x_{ij} = a_i, i \in [1,n] \\ \sum_{i=1}^{n} x_{ij} = b_j, j \in [1,m] \\ \forall x_{ij} \ge 0 \left(i \in [1,n], j \in [1,m]\right) \end{cases}. \qquad (12)$$

*Целевая функция* (12) представляет собой транспортные расходы на осуществление всех перевозок в целом. Первая группа ограничений указывает, что запас продукции в любом пункте отправления должен быть равен суммарному объему перевозок продукции из этого пункта. Вторая группа ограничений указывает, что суммарные перевозки продукции в какой-либо пункт потребления должны полностью удовлетворить спрос на продукцию в этом пункте. Формой представления модели ТЗ является транспортная матрица (табл. 2).

*Таблица 2*

**Общий вид транспортной матрицы**

| Пункты отправления (нефтебазы) $A_i$ | Пункты потребления (АЗС) $B_j$ | | | | Запасы, ед. прод., т |
|---|---|---|---|---|---|
| | $B_1$ | $B_2$ | … | $B_m$ | |
| $A_1$ | $c_{11}$ | $c_{12}$ | … | $c_{1m}$ | $a_1$ |
| $A_2$ | $c_{21}$ | $c_{22}$ | … | $c_{2m}$ | $a_2$ |
| … | … | … | … | … | … |
| $A_n$ | $c_{n1}$ | $c_{n2}$ | … | $c_{nm}$ | $a_n$ |
| Потребность, ед. прод., т | $b_1$ | $b_2$ | … | $b_m$ | $\sum_{i=1}^{n} a_i = \sum_{j=1}^{m} b_j$ |

Из модели (12) следует, что сумма запасов продукции во всех пунктах отправления должна равняться суммарной потребности во всех пунктах потребления, то есть

$$\sum_{i=1}^{n} a_i = \sum_{j=1}^{m} b_j . \qquad (13)$$

Если условие (13) выполняется, то ТЗ называется *сбалансированной*, если не выполняется – несбалансированной. Поскольку ограничения модели (12) могут быть выполнены только при сбалансированной ТЗ, то при построении транспортной модели необходимо проверять условие баланса (13). В случае, когда суммарные запасы на нефтебазах превышают суммарные потребности АЗС, необходим дополнительный фиктивный пункт потребления, который будет формально потреблять существующий излишек запасов, то есть

$$b_{ф} = \sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j . \qquad (14)$$

Если суммарные потребности АЗС превышают суммарные запасы, то необходимо ввести в модель фиктивный пункт отправления (нефтебазу), формально восполняющий существующий недостаток продукции в пунктах отправления:

$$a_{ф} = \sum_{j=1}^{m} b_j - \sum_{i=1}^{n} a_i . \qquad (15)$$

Введение фиктивного потребителя или отправителя повлечет необходимость формального задания фиктивных тарифов $c_{ij}^{ф}$ (реально не существующих) для фиктивных перевозок. Поскольку нас интересует определение наиболее выгодных реальных перевозок, то необходимо предусмотреть, чтобы при решении задачи (при нахождении опорных планов) фиктивные перевозки не рассматривались до тех пор, пока не будут определены все реальные перевозки. Для этого надо фиктивные перевозки сделать невыгодными, то есть наиболее дорогими, чтобы при поиске решения задачи их рассматривали в самую последнюю очередь. Таким образом, величина фиктивных тарифов должна превышать максимальный из реальных тарифов, используемых в модели, то есть

$$c_{ij}^{ф} > \max c_{ij} \left( i \in [1, n], j \in [1, m] \right). \qquad (16)$$

На практике возможны ситуации, когда в определенных направлениях перевозки продукции невозможны, например, по причине ремонта транспортных магистралей. Такие ситуации моделируются с помощью введения так называемых запрещающих тарифов $c_{ij}^{з}$. Запрещающие тарифы должны сделать невозможными, то есть совершенно невыгодными, перевозки в соответствующих направлениях. Для этого величина запрещающих тарифов должна значительно превышать максимальный из реальных тарифов, используемых в модели:

$$c_{ij}^{з} >> \max c_{ij} \left( i \in [1, n], j \in [1, m] \right). \qquad (17)$$

### 3.2 Методика решения задач оптимизации в среде Scilab

Для решения задач линейного программирования в Scilab предназначена функция *linpro* следующей структуры[1]:

[1] В системе Matlab для этих целей служит функция *linprog*

**[x,kl,f] = *linpro*(c, A, b [,ci, cs] [, k] [ ,x0])**

Здесь c – массив (вектор-столбец) коэффициентов при неизвестных функции цели, длина вектора n совпадает с количеством неизвестных x.

A – матрица коэффициентов из левой части системы ограничений, количество строк матрицы равно количеству ограничений m, а количество столбцов совпадает с количеством неизвестных коэффициентов n.

b – массив (вектор-столбец), содержит свободные члены системы ограничений, длина вектора m.

ci – массив (вектор-столбец) размерности n содержит нижнюю границу переменных (cij < = xj); если таковая отсутствует, указывают [ ].

cs – массив (вектор-столбец) длиной n, содержит верхнюю границу переменных (csj > = xj); если таковая отсутствует, указывают [ ].

k – целочисленная переменная, используется, если в систему ограничений кроме неравенств входят и равенства, в матрице они будут находиться в k первых строках, оставшиеся l строк займут неравенства, т.е. *m = k + l (Включать обязательно для решения транспортной задачи, k равно числу условных равенств)*.

x0 – вектор-столбец начальных приближений длиной n.

Функция *linpro* возвращает массив неизвестных x и минимальное значение функции C.

Рассмотрим использование функции *linpro* на примере решения следующей задачи линейного программирования.

***Пример решения задачи оптимизации***

Планируется выпуск двух видов костюмов – мужских и женских. На женский костюм требуется 1 м шерстяной ткани, 2 м хлопка и 1 день трудозатрат. На мужской костюм – 3,5 м шерстяной ткани, 0,5 м хлопка и 1 день трудозатрат. Всего имеется 350 м шерстяной ткани, 240 м хлопка и 150 дней трудозатрат.

По плану предусматривается выпуск не менее 110 костюмов, причем необходимо обеспечить прибыль не менее 1400 рублей. Требуется определить оптимальное число костюмов каждого вида, обеспечивающее максимальную прибыль, если прибыль от реализации женского костюма составляет 10 руб., а от реализации мужского – 20 руб.

**Постановка условий и решение**

Пусть $x_1$ – число женских костюмов, $x_2$ – число мужских костюмов. Прибыль от реализации женских костюмов составляет $10*x_1$ руб., от реализации мужских костюмов $20*x_2$ руб., то есть необходимо максимизировать целевую функцию

$f(x)=10*x_1+20*x_2$

Расход шерсти составляет $1*x_1+3,5*x_2$, хлопка $2*x_1+0,5*x^2$, трудовых ресурсов $x_1+x_2$. Поэтому ограничения задачи имеют вид

1) $x_1 + 3,5*x_2 \le 350$;

2) $2*x_2 + 0,5*x_2 \le 240$;

3) $x_1 + x_2 \le 150$;

4) $x_1 + x_2 \ge 110$;

5) $10*x_1+20*x_2 \ge 1400$;

6) $x_1 \ge 0$;

7) $x_2 \ge 0$.

Первые три неравенства описывают ограничения по ресурсам, четвертое и пятое – соответственно плановое задание по общему числу костюмов и ограничение по прибыли.

Код Scilab для решения приведенной задачи показан ниже.

Для **максимизации** задачи, т.е. наиболее полного использования трудовых и сырьевых ресурсов, необходимо взять f(x) со знаком «−», т.е. −f(x). **Для ее минимизации, т.е. определения количества костюмов, необходимого сделать, наоборот (**взять f(x) с +).

```
// Сперва введем коэффициенты целевой функции f, переменных в условиях A,
ограничения по условиям <= b, нижние границы ci >=0
c = [-10;- 20];
A = [1 3.5
   2 0.5
   1 1
   -1 -1
   -10 -20];
b = [350; 240; 150; -110; -1400];
//для условий >= в A и b выставляются - коэффициенты
ci = zeros(2,1);
k=0; //в этом примере не используется условных равенств
// Далее обратимся к программе линейного программирования
[x,kl,f] = linpro(c,A,b,ci,[],k);
 // Вводя x, lambda.ineqlin и lambda.lower получим
disp('значения Xij');
x
disp('значение целевой функции f');
f
```

Выполнив вышеуказанный код, получим:

```
x =
  70
  80
f =
   −2300
```

T.к. f = −f(x), то f(x)=2300.

Этот вывод следует интерпретировать следующим образом: для максимизации прибыли при имеющихся ресурсах следует выпустить 70 мужских и 80 женских костюмов.

```
// Сперва введем коэффициенты целевой функции f, переменных в условиях A, ограниче-
ния по условиям <= b, нижние границы ci >=0
c = [10; 20];
A = [1 3.5
   2 0.5
   1 1
   -1 -1
   -10 -20];
b = [350; 240; 150; -110; -1400];
ci = zeros(2,1);  //нижнее ограничение у c=0, для может быть задано большим числом
 // Далее обратимся к программе линейного программирования
k=0; //в этом примере не используется условных равенств
[x,kl,f] = linpro(c,A,b,ci,[],k);
 // Вводя x, lambda.ineqlin и lambda.lower получим
disp('значения Xij');
x
disp('значение целевой функции f');
f
```

**Выполнение листинга возвращает:**

```
x =
  80
  30
fval =
1400
```

Таким образом было получено решение минимизации, удовлетворяющее условиям. Т.е. $x_1 = 80$; $x_2 = 30$; значение функции $f(x) = 1400$.

## Самостоятельная работа

Для усвоения материала необходимо самостоятельно разобрать пример решения задачи оптимизации, приведенный выше. Полученный опыт использовать для решения задачи лабораторной работы транспортировки нефтепродуктов в соответствии с вариантом (приложение III).

Перед выполнением работы и отражения ее результатов в отчете, прорешать в Scilab вариант, разобранный в примере решения транспортной задачи графическим способом.

Для дополнительной практики и улучшения понимания рекомендуется использовать практические упражнения из книги [1]: Алексеев Е.Р., Чеснокова О.В., Рудченко Е.А. «Scilab: Решение инженерных и математических задач».

## Требования к отчету

Отчет о выполнении лабораторной работы должен содержать:

– тему и цель лабораторной работы на титульной странице;

– оглавление;

– необходимые теоретические сведения по теме;

– исходные данные и последовательность их обработки (преобразования);

– поэтапный расчет в виде вставок кода Scilab в объектах «Подпись»;

– результаты выполнения кода Scilab в виде вывода в командном окне;

– интерпретацию результатов расчета объемов топлива, поставляемых с i нефтебазы на j АЗС, $x_{ij}$ в т;

– отметку преподавателя о выполнении лабораторной работы.

# 4. ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ И МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

Целью настоящего раздела является ознакомление с применением обыкновенных дифференциальных уравнений (ОДУ) в решении элементарных задач математического моделирования, а также способами решения ОДУ (задач Коши и краевых задач) в программах научного программирования (на примере среды *Scilab*).

Необходимо разобрать примеры 1–2, выполнить задачи 1 и 2 с учетом вариантов. По результатам выполнения задачи представить научный отчет.

## 4.1. Понятие об обыкновенных дифференциальных уравнениях

Математическое моделирование служит для постановки и решения задач реального мира в математических терминах (рис. 13).


Рис. 13. Схематический процесс математического моделирования

Обыкновенным дифференциальным уравнением (ОДУ) называется уравнение, связывающее между собой значения независимой переменной x, неизвестной функции y = f(x) и её производных (или дифференциалов):

$$F(x, y, y', y'', ..., y^{(n)}) = 0 . \qquad (18)$$

Порядком уравнения называется максимальный порядок n входящей в него производной (или дифференциала).

Решением (интегралом) дифференциального уравнения порядка n называется функция *y(x)*, имеющая на некотором интервале (a, b) производные до порядка n включительно и удовлетворяющая этому уравнению. Процесс решения дифференциального уравнения называется интегрированием.

Все дифференциальные уравнения можно разделить на *обыкновенные* (ОДУ), в которые входят только функции (и их производные) от одного аргумента, и *уравнения с частными производными* (УРЧП), в которых входящие функции зависят от многих переменных.

## 4.2. Геометрический смысл уравнения первого порядка

Уравнение $y'=f(x,y)$ или $dy/dx=f(x,y)$ в каждой точке (x, y) области, в которой задана функция f(x, y), определяет угловой коэффициент касательной к решению, проходящему через точку (x, y), т.е. направление, в котором проходит решение через эту точку. На рис. 14 показаны две связные ветви графика y=1/(1-x). Стрелкой слева отмечено решение ОДУ $y'=y^2$, определенное на множестве (-∞; 1). Справа показана ветвь графика, соответствующая другому решению ОДУ, которое определено и непрерывно на другом интервале (1;+∞).

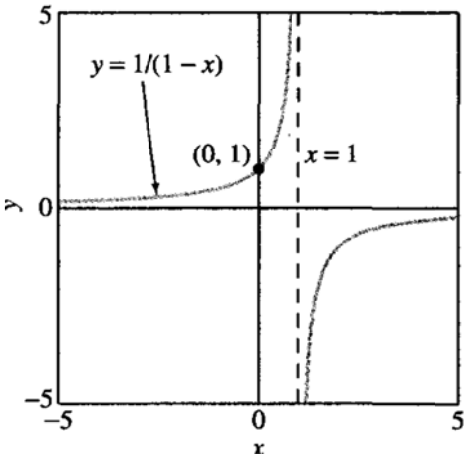


Рис. 14. Решение ОДУ y’ = $y^2$, функция y = 1/(1-x)

### 4.3. Приложения ОДУ в информатике

Иногда для моделирования производственных процессов в нефтегазовой отрасли промышленности интересны не столько *функция f(x)*, сколько *скорость f’(x)* и *ускорение f’’(x)* ее изменения. В некоторых случаях только скорость и ускорение бывают известны. В этом случае требуется найти функцию (интеграл).

**Пример № 1. Постановка модели и решение простейшей задачи.** Для рекультивации нефтезагрязненных почв зачастую используются живые организмы, штамм которых вносится в грунт. По условию задачи, колония живых организмов, разлагающих углеводороды, находится в благоприятных условиях, благодаря чему прирост $\alpha > 0$ (рождаемость $\gamma$ выше, чем смертность $\sigma$, $\alpha = \gamma - \sigma$; $\alpha > 0$). Для упрощения, пространство, занимаемое колонией, и ее пищевые ресурсы (т.е. пропитывающие почву углеводороды) будем считать неограниченными. Предположим также, что естественных врагов, питающихся организмами данной колонии, нет. Найти закон изменения численности организмов в зависимости от времени, если при t = 0 их число равнялось $y_0$.

**Решение:** будем считать, что скорость изменения численности организмов пропорциональна этой численности и $\alpha$ – коэффициент пропорциональности (рождаемость $\gamma$ выше, чем смертность $\sigma$, $\alpha = \gamma - \sigma$; $\alpha > 0$):

$V = \alpha \cdot y$.

Так как $V = y'$, то численность y организмов в колонии в момент времени t удовлетворяет уравнению: $y' = \alpha \cdot y$.

Отсюда $\frac{dy}{dt} = \alpha y$.

Разделяем переменные: $\frac{dy}{y} = \alpha dt$. Начальное условие: при t = 0 численность особей $x = x_0$.

Интегрируя, получим общее решение:

$$\int_{x0}^{x} \frac{dy}{y} = \int_{0}^{t} \alpha dt;$$

$$\ln y - \ln y_0 = \ln \frac{y}{y_0} = \alpha t;$$

$$y = y_0 \cdot e^{\alpha t}.$$

Значит, число организмов в колонии изменяется в соответствии с законом

$$y = y_0 \cdot e^{\alpha t}.$$

Ответ: число организмов в колонии изменяется по закону $y = y_0 \cdot e^{\alpha t}$.

### 4.4. Решение ОДУ в Scilab

Для решения обыкновенных дифференциальных уравнений в Scilab применяется функция *ode()*. С помощью этой функции можно решать граничные задачи.

Синтаксис команды:

y = *ode*(y0,t0,t,f),

здесь: y – вектор (массив-строка) решений дифференциального уравнения,

y0 – начальное значение функции y(t),

t0 – начальное значение аргумента,

t – вектор значений аргумента,

f – производная исследуемой функции.

Выполним постановку, табличное и графическое решение задачи о численности популяции (см. пример № 1) в Scilab для определенного коэффициента прироста популяции $\alpha = 0.3$ и на множестве $t = \overline{0,10}$ (т.е. с момента времени 0 до 10).

*Код Scilab для примера 1:*

```
function ydot=f(y, t)
   ydot=alpha*y;
endfunction;
y0=3;
t0=0;
alpha=0.3;
t=0:0.1:10;
y=ode(y0,t0,t,f);
plot(t, y);
```

**Пример № 2. Решение простой задачи теоретической механики.** Для планирования действий по ликвидации последствий возможных чрезвычайных ситуаций (ЧС) на нефтегазовых предприятиях иногда важно знать не только направление, но и скорость движения ветра, которая изменчива и зависит от внешних условий.

Проходя через лес и испытывая сопротивление деревьев, ветер теряет часть своей скорости. На бесконечно малом пути эта потеря пропорциональна скорости в начале этого пути и его длине. Найти скорость ветра, прошедшего в лесу **150 м**, зная, что до вступления в лес начальная скорость ветра $V_0 = 12\ м/с$; после прохождения пути $\boldsymbol{S = 1}$ **м** скорость ветра **уменьшилась** до величины $V_1 = 11{,}8\ м/с$.

**Решение:** Пусть на расстоянии $S$ от начала леса скорость ветра равна $V$, потеря скорости на пути $dS$ равна $-dV$ (**процесс убывающий**). Эта потеря пропорциональна $V$, и поэтому дифференциальное уравнение процесса примет вид: $-\frac{dV}{dS} = kV$.

Разделяем переменные:

$$\frac{dV}{V} = -k \cdot dS.$$

Интегрируя, получим общее решение задачи:

$$V = c \cdot e^{-ks} \qquad \textbf{(1)}.$$

Найдем частное решение, используя начальное условие: при S = 0; $V = V_0$. Подставим это условие в уравнение **(1)**: $V_0 = c \cdot e^0 \Rightarrow c = V_0$.

Закон процесса:

$$V = V_0 \cdot e^{-kS} \quad \textbf{(2)}.$$

Для определения коэффициента пропорциональности $k$ используем дополнительное условие: при S = 1м, $V = V_1 = 11{,}8 м/с$.

Откуда: $V_1 = V_0 \cdot e^{-k}$, или $e^{-k} = \frac{V_1}{V_0} = \frac{11{,}8}{12} = 0{,}983$.

Подставляя числовые значение в уравнение **(2)**, получим искомую скорость: $V = 12 \cdot (0{,}983)^{150} = 12 \cdot 0{,}0776 \approx 0{,}92 м/с$.

Итак, скорость ветра, углубившегося на 150м в лес, составит 0,93 м/с.

Ответ: $V(150) = 0{,}92 м/с$.

Для использования программы *ode() Scilab* необходимо найти коэффициент пропорциональности $k$. В коде ниже показано его вычисление.

*Код Scilab для решения примера 2:*

```
clear; //очистка памяти
function Vdot=f(S, V)
   Vdot=-k*V;
endfunction;
V0=12;
t0=0;
k=-log(11.8/12);
//k=0.016;
t=0:1:150;
V=ode(V0,t0,t,f);
plot2d(t, V);
ylabel('V(S),[м/с]');
xlabel('S,[м]');
title('график изменения V(S)');
disp ('Скорость ветра через 150 м = ');
disp (V(150));
```

Вызов программы *в Scilab* возвращает графическое решение уравнения (рис. 15) и результат в командной строке.

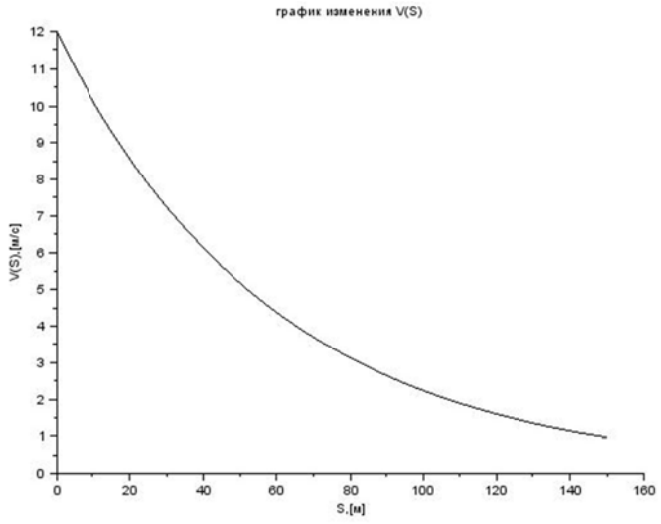


Рис. 15. График изменения скорости ветра *V(S)* по мере его продвижения в лесу на расстояние *S*

### Задачи для самостоятельного решения

**Задача 1. Разрядка аккумуляторной батареи.** Требуется сформулировать задачу разрядки батареи, питающий лампу. Электрическая схема цепи состоит из источника постоянного тока с напряжением *U*, выключателя и лампы.

Чем дольше горит лампа, тем меньшее напряжение должна вырабатывать батарейка модели и тем слабее должна светить лампочка. В этом и состоит динамика модели: напряжение и связанная с ним яркость свечения лампы зависят от времени (рис. 16).


Рис. 16. Электрическая схема цепи и электролампы

Уменьшение напряжения батарейки со временем опишем уравнением:

$$U_{Б}(t) = U_0 \cdot e^{-t/T} = U_0 \cdot e^{-\alpha t},$$

где $U_Б(t)$ – зависимость напряжения батарейки от общего времени ее работы;

$U_0$ – напряжение, создаваемое новой батарейкой;

$t$ – общее время работы батарейки в фонарике;

$T$ – срок службы батарейки, работающей в фонарике. Это характерное время, в течение которого батарейка разрядится почти на две трети, точнее, ее напряжение составит 37% от начального. У многих батареек для фонариков время Т составляет 2–3 часа.

Программа, моделирующая фонарик, должна учитывать время работы батарейки и уменьшать ее напряжение в соответствии с формулой выше. Отметим, что при решении этой задачи мы пренебрежем сопротивлением проводов, поэтому напряжение на лампочке будет равно напряжению батарейки.

Для решения задачи нам потребуется представить уравнение разряда батарейки в дифференциальном виде:

$$\frac{dU}{U} = -\alpha \cdot dt.$$

Вариант задачи (приложение IV) содержит: $U_0$; $U_n$ – напряжение через время $n$; $n$ – время работы до уменьшения напряжения до $U_n$.

**Требуется:**

1. Найдите численное решение уравнения этой задачи, задав характеристики аккумуляторной батареи (см. вариант задачи, приложение IV). Как найти коэффициент пропорциональности $\alpha$?

2. По аналогии с предыдущими упражнениями составьте код Scilab. Когда разрядится аккумулятор?

3. Постройте график изменения напряжения $U(t)$ в Scilab для вставки в отчет.

**Задача 2. Зарядка электроконденсатора.** Электрический конденсатор изначально не заряжен. Процесс зарядки электрического конденсатора в цепи (рис. 17), к которому в момент времени 1 = 0 приложено напряжение *U*, описывается ОДУ первого порядка

$$R\frac{dQ}{dt} = V - \frac{Q}{C}$$

с начальным условием $Q(0) = 0$.

**Требуется:**

1. Найдите численное решение уравнения этой задачи, задав следующие характеристики цепи: сопротивление $R$, емкость $C$, напряжение $V$ (см. вариант задачи, приложение V).

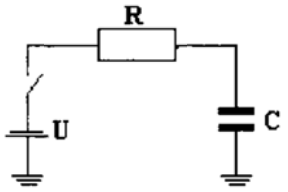


Рис. 17. Схема зарядки конденсатора

2. Ответьте на вопросы. Как зависит заряд конденсатора от времени? Заряжается ли он бесконечно, или в какой-то момент времени происходит его насыщение?

3. Постройте график зарядки конденсатора для вставки в отчет.

**Указание.** Для увеличения точности решения можно брать время с меньшим шагом. Рекомендуемый шаг времени $t$ наблюдений 0,0001 с, диапазон наблюдений $t$ изменяется от 0 до 0,05 с.

**Рекомендация.** Для проверки незначительности увеличения заряда конденсатора с дальнейшим течением времени можно использовать код, аналогичный приведенному ниже.

*Код Scilab для обхода вектора значений заряда Q в момент времени t и проверки малости отличий между Q(i) и Q(i-1):*

```
//проверка малости изменений прибл. значений вектора Q
for i=2:length(Q)
   i_min=i-1;
if Q(i)-Q(i_min)<1e-9 then
   disp('Конденсатор заряжен в момент времени t~');
   disp(t(i));
   disp ('Заряд конденсатора достиг Q = ');
   disp (Q(i));
   break;
   end;
end
```

# 5. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ И КЛЕТОЧНЫЕ АВТОМАТЫ В GNU OCTAVE (MATLAB)

**Цель работы:** знакомство с применением клеточных автоматов в решении задач математического моделирования, настройка клеточного автомата.

**Задание:** изучить теоретическую часть и пример кода клеточного автомата, найти вкравшуюся в код ошибку. Изменяя входные параметры системы в соответствии с заданием, изучить их влияние на результат работы модели. Подготовить содержательный отчет, сделать выводы.

**Примечание.** Для выполнения настоящей работы используется бесплатная программа GNU Octave (http://www.octave.org), имеющая существенную совместимость с Matlab и близость с Scilab.

## 5.1. Представление о клеточных автоматах

Клеточный автомат (КА) – дискретная модель, изучаемая в математике, теоретической биологии, физике, гидравлике и т.д. Включает регулярную решётку ячеек, каждая из которых может находиться в одном из конечного множества состояний, таких как 1 и 0.

Решетка может быть любой размерности (соотношение длины сторон, рис. 18). Для каждой ячейки определено множество ячеек, называемых окрестностью. К примеру, окрестность может быть определена как все ячейки на расстоянии не более 2 от текущей (*окрестность фон Неймана ранга 2*).

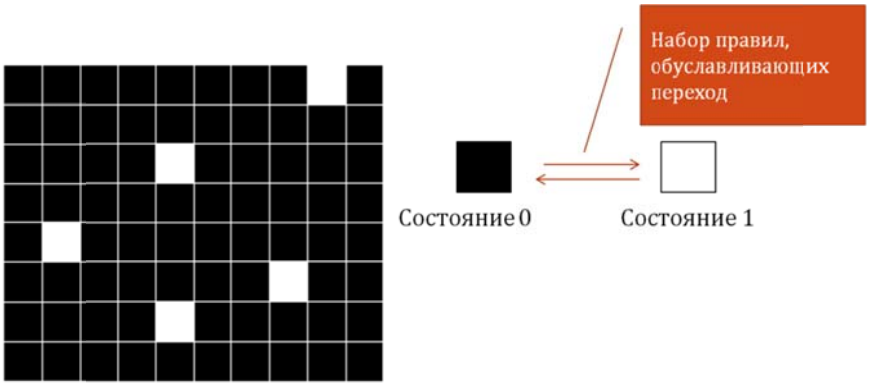


Рис. 18. Схематическое представление простейшего клеточного автомата

Для работы клеточного автомата требуется задание начального состояния всех ячеек и правил перехода ячеек из одного состояния в другое. На каждой итерации (ходе), используя правила перехода и состояния соседних ячеек, определяется новое состояние каждой ячейки.

Обычно правила перехода одинаковы для всех ячеек и применяются сразу ко всей решётке.

**Пример: игра «Жизнь» Дж Конвея.** Для моделирования поведения популяции одноклеточных простейших используется решетка (матрица) размерностью100x100 элементов. Решетка заселяется некоторым исходным количеством организмов. Затем, с течением времени для всей решетки выполняется ряд правил, управляющих поведением «организмов», т.е. элементов матрицы.

1. Выживание. Каждая фишка, имеющая вокруг себя две или три соседние фишки, выживает и переходит в следующее поколение.

2. Гибель. Каждая фишка, у которой больше трёх соседей, погибает, то есть снимается с доски из-за перенаселённости.

Каждая фишка, вокруг которой свободны все соседние клетки или же занята всего одна клетка, погибает от одиночества.

3. Рождение. Если число фишек, с которыми граничат какая-нибудь пустая клетка, в точности равно трём (не больше и не меньше), то на этой клетке происходит рождение нового «организма», то есть следующим ходом на неё ставится одна фишка.

4. Возраст. Предельный возраст клетки – 2 поколения.

*Код игры «Жизнь» (по Iain Haslam, http://exolete.com/life/):*

```
len=50; GRID=int8(rand(len,len));
up=[2:len 1]; down=[len 1:len-1]; %the world is round
colormap(gray(2));
for i=1:100
    neighbours=GRID(up,:)+GRID(down,:)+GRID(:,up)+GRID(:,down)+...
        GRID(up,up)+GRID(up,down)+GRID(down,up)+GRID(down,down);
    GRID = neighbours==3 | GRID & neighbours==2;
    image(GRID*2); pause(0.02);
end
```

После начала игры «популяция» быстро начинает сокращаться, уступая место долгоживущим структурам – «глайдерам», которые способны двигаться. Их столкновения друг с другом приводят к разрушению стабильности.

**Самостоятельно.** Разобрать структуру программы «Жизнь». Изучить поведение системы при изменении входных параметров – **размеров поля**, **времени жизни клеток** и других.

**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 (Строка 1, Строка 2, …Строка n )*.

Существуют и другие несовместимости. 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».

**Внимание!** Начало координат изображения в Matlab/Scilab/Octave – левый верхний угол.

### 5.2. Создание клеточного автомата-модели в GNU Octave / Matlab

Для создания клеточного автомата возьмем следующую ситуацию. В резервуаре с жидкостью на дно падает дробинка. На нее воздействуют Архимедова сила $F_a$, стремящаяся вытолкнуть дробинку и сила тяжести $F_g$, направленная ко дну сосуда. Если плотность жидкости больше плотности дробинки, последняя плавает, в противном случае – тонет (рис. 19). Сопротивлением жидкости и ее вязкостью при падении в ней дробинки следует пренебречь.

Модель, реализуемая в виде клеточного автомата, должна удовлетворять следующим условиям:

1. Дробинка движется с ускорением.

2. Если $F_a > F_g$, дробинка не тонет, выдается сообщение о плавании дробинки.

3. Максимальный срок выполнения программы – 100 кадров (шагов) или момент касания дна.

4. По завершении работы модели выдается график изменения глубины дробинки и ее ускорения (рис. 20).

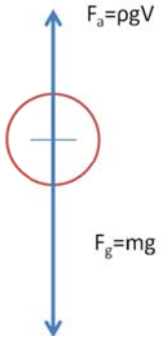


Рис. 19. Схема сил, действующих на падающих в жидкости дробинку:
$F_g$ – сила тяжести, $F_a$ – Архимедова сила

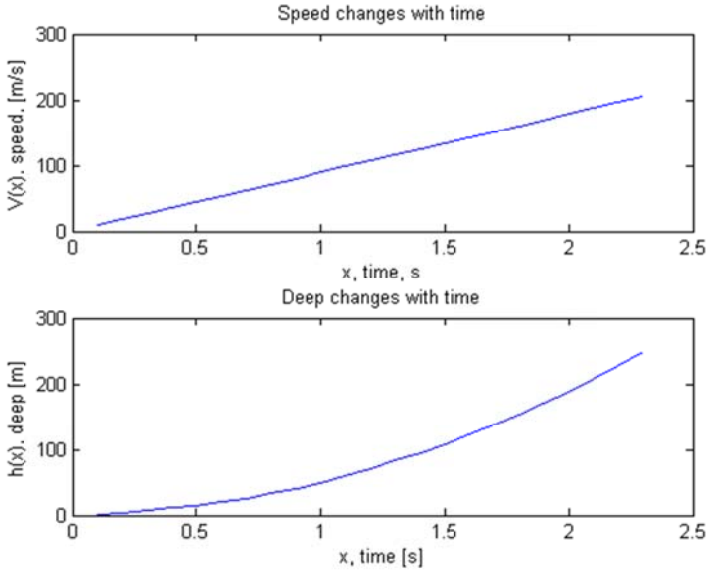


Рис. 20. Графики, создаваемые программой по завершении ее выполнения:
верхний – изменение скорости со временем, нижний – увеличение глубины со временем.

Код Matlab с ошибками для клеточного автомата, реализующего падение дробинки в резервуар, дан в приложении VII.

**Задание для самостоятельной работы**

- Исправить код Matlab, найдя ошибки, препятствующие его нормальной работе;
- Используя комментарии в коде программ, детально выяснить порядок их работы, значение переменных и алгоритмы применяемые в функциях.
- Использовать клеточный автомат для исследования падения дробинки плотности, заданной в варианте (приложение VI), в жидкости № 1 и 2.
- Описать проводимый эксперимент, сравнить результаты и обосновать выводы;.
- Ответить на вопрос: «Как поведет себя дробинка из свинца в ртути?». Обосновать ответ аналитически.
- Составить блок-схему работы клеточного автомата «Падение дробинки».

# 6. ОБРАБОТКА ИЗОБРАЖЕНИЙ В GNU OCTAVE (MATLAB)

**Цель работы:** ознакомление с средствами и методическими приемами анализа дистанционных изображений в средах научного программирования.

**Задача:** провести анализ изображений в соответствии с вариантом. Выполнить детальные задания в конце файла. Подготовить содержательный отчет, сделать выводы.

**Примечание.** Для выполнения настоящей работы используется бесплатная программа GNU Octave (http://www.octave.org), имеющая существенную совместимость с Matlab. Для работы программы обработки изображений в Octave, требуется скачать модуль *image* с сайта программы, а затем установить его с помощью консольной команды Octave:

```
pkg install C:\my_folder\image-2.4.1.tar.gz
```

Здесь `C:\my_folder\` – папка, в которой находится скачанный архив с модулем image, `image-2.4.1.tar.gz` – имя файла архива.

После установки необходимо перезапустить программу. Загрузка модулей не автоматизирована и требует ввода консольной команды:

```
pkg load all
```

**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.

## 6.1. Представление об изображениях

В информатике 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. 21).

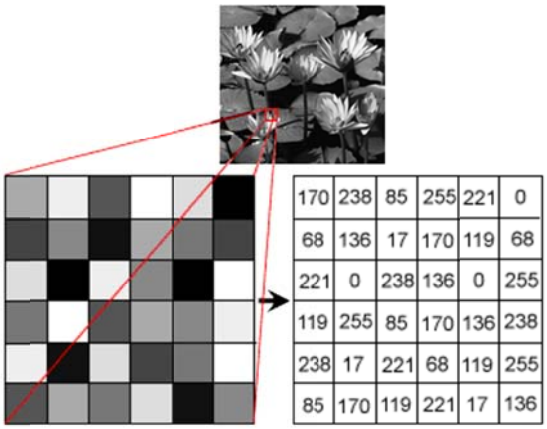


Рис. 21. Представление фрагмента серого изображения в виде матрицы

При этом следует помнить, что «серые», одноканальные изображения представляют собой одну матрицу, тогда как изображения «цветные» представляют собой три канала Red – красный (λ = 625–740), Green – зеленый (λ = 500–565), Blue – голубой (λ = 485–500),

т.е. представляются тремя матрицами. Это нужно учитывать при открытии цветного изображения.

### 6.2. 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 или «Workspace» Matlab. Наряду с этим, свойства переменной можно узнать с помощью команды whos VAR_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
```

**Что мы узнали?** Переменная My_lily представляет собой трехмерный массив размерностью 600x800 (`600x800x`**3**), занимающий в памяти 1440000 байт, элементы массива представляют собой данные класса `uint8` (8-битное целое). В элемент класса uint8 можно записать данные от 0 до $2^8$=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 масштабирования 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[2].

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

%масштабирование на 50%
My_lily_small= imresize(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);
```

**Самостоятельно:** 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);
```

Проверить, отразить в отчете и обосновать свой ответ.

Для проведения анализа изображение часто должно пройти этапы обработки, направленные на его размытие или повышение резкости, распознавание границ, а также пре-

[2] О способах применения функции *imwrite( )*, дополнительных параметрах и способах применения следует читать официальную справку к функции

образование 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. 22).

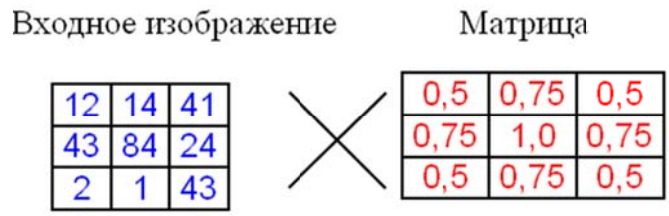


Рис. 22. Свертка изображения по матрице коэффициентов

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. 23).

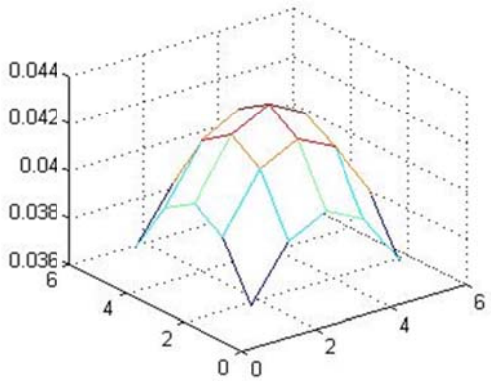


Рис. 23. Пространственная сетка «гауссовой матрицы»

Необходимо знать, что свертка матриц (конволюция) может быть выполнена только для одноканальной матрицы – серого изображения. Что же делать, если матрица 3-канальная, как в случае нашей цветной картинки? Логично предположить, что необходимо 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*).

**Самостоятельно:** детально разобрать код выше и понять как он работает**.** Результат вывода каналов изображения показан на рис. 24.

```
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); imshow(A); title('RGB');
subplot(2,2,2); imshow(A_B);title('Blue channel');
subplot(2,2,3); imshow(A_G);title('Green channel');
subplot(2,2,4); 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');
```


Рис. 24. 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);
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);
```

### 6.3. 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 (Строка 1, Строка 2, …Строка n)*.

Существуют и другие несовместимости. 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».

**Внимание!** Начало координат изображения в Matlab/Scilab/Octave – левый верхний угол.

**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;
- Прорешать приводимые примеры, внимательно изучив комментарии 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*.
- На совмещенной графике *subplot* размером 4×4 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 затем 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.

# 7. ОБРАБОТКА ИЗОБРАЖЕНИЙ ДЛЯ ОПРЕДЕЛЕНИЯ КОЛЛЕКТОРСКИХ СВОЙСТВ ПОРОД В GNU OCTAVE (MATLAB)

**Цель работы:** ознакомление с методическими приемами анализа микроизображений горных пород в средах научного программирования.

**Задание:** провести анализ изображений в соответствии с вариантом с применением приводимых методик. Применить фильтры изображения, описать достигаемый эффект. Бинаризировать изображение, выделить кластеры и провести их простейший анализ.

Подготовить содержательный отчет, сделать выводы.

**Примечание.** Для выполнения настоящей работы используется бесплатная программа GNU Octave 4.0 (http://www.octave.org), имеющая существенную совместимость с Matlab. Для работы программы обработки изображений в Octave требуется скачать модуль *image* с сайта программы, а затем установить его с помощью консольной команды Octave:

```
pkg install C:\my_folder\image-2.4.1.tar.gz
```

Здесь C:\my_folder\ – папка, в которой находится скачанный архив с модулем image, image-2.4.1.tar.gz – имя файла архива. Перед выполнением программы с применением модуля его необходимо загрузить командой *pkg load all.*

**Внимание!** Некоторые коды примеров могут содержать 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.

## 7.1. Коллекторские свойства горных пород

По своему происхождению горные породы могут быть разделены на осадочные, магматические и метаморфические.

Для прогноза, поисков и эксплуатации нефтегазовых месторождений, проектирования подземных хранилищ природного газа бывает важно определить, является горная порода коллектором (т.е. может вмещать жидкость или газ) или флюидоупором (т.е. не способна вмещать жидкость или газ).

В качестве главных свойств пород, отвечающих за их коллекторские качества, выступают *пористость* и *проницаемость*.

*Пористость* горной породы φ выражается как отношение объема пор $V_{пор}$ к объему образца $V_{обр}$ и может быть выражена в %:

$$\varphi = \frac{V_{пор}}{V_{обр}}. \quad (18)$$

Пористость соответствует емкости породы – т.е. ее способность вмещать флюид – нефть, газ или природные воды. Пористость также может быть выражена как функция давления, так как при увеличении давления пористость уменьшается за счет сокращения объема пор. В практике геологических исследований обычно применяют лабораторные методы измерения пористости.

Однако в случае однородного распределения пор, каналов или трещин в породе, пористость может определена в результате анализа микрофотоизображения породы – *шлифа* (рис. 25).

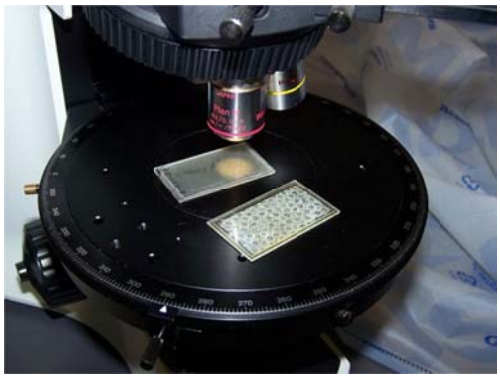

Рис. 25. Шлифы на предметном столике микроскопа

Шлифом называется тонкая (доли миллиметра) полированная пластинка породы, которая приклеивается к препаровальному стеклу с помощью канадского бальзама (прозрачная смесь смол хвойных растений).

Изучение шлифов и их описание выполняется в специальных оптических приборах, называемых поляризационными микроскопами (рис. 26).

Рис. 26. Поляризационный микроскоп Nikon Eclipse E200POL

Для приближенного расчета пористости при анализе плоского изображения можно заменить отношение объема пор.

*Проницаемость* горной породы рассматривается неотрывно от течения в ней жидкости и может быть выражена с помощью отношения Кармана-Козени [1]:

$$K = \frac{1}{8\tau A_v^2}\frac{\varphi^3}{(1-\varphi)^2}. \qquad (19)$$

Проницаемость зависит от текстуры породы, выраженной в качестве показателя кривизны $\tau$ и удельной площадью поверхности частиц породы $A_v$.

### 7.2. Методы морфологического анализа изображений в Octave (Matlab)

Для ознакомления с методами обработки изображений в Octave рассмотрим следующий пример. Для занятия необходимо использовать изображение в папке files архива лабораторной работы.

**Внимание!** Для корректной работы Octave имя файла и путь к нему не должны содержать пробелов и символов, отличающихся от латинских.

Рассмотрим изображение неких изометрических (округлых) объектов, находящихся на плоской поверхности (рис. 27).

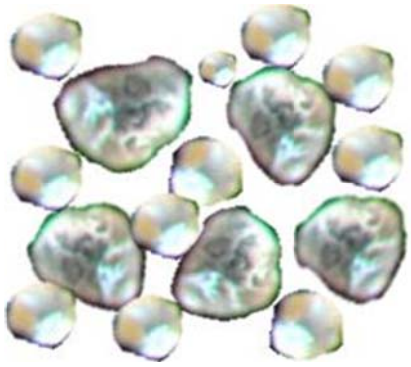

Рис. 27. Изображение песчинок под увеличением

Что можно понять на первый взгляд, смотря на эти объекты? Прежде всего, то, что приемы анализа должны зависеть от сущности наблюдаемых объектов. Некоторые объекты на рисунке *circles_foto.png* содержат внутри светлую область. Это может быть или пустота или блик, т.е. засветка фото, вызванная более интенсивным отражением определенных частей изображения. Если это блик, а не пространство, то, например, для задач учета площади объектов нам необходимо заполнить его пикселями. Как это сделать, будет рассмотрено в примерах этой лабораторной работы.

*Бинаризация* – преобразование цветного (RGB) изображения в логическое, матрица которого состоит из 0 и 1. Это бывает нужно для выделение полезного сигнала, т.е. необходимой для текущего исследования части изображения.

**Пример 1.** Открытие изображения и его бинаризация. Обратите внимание на комментарии в тексте программы.

```
clear; %очистка памяти
pkg load all; %загружаем модули, только для Octave!
A=imread('sand.png'); %загрузка цветного изображения
A_gray=rgb2gray(A); %преобразование изображения в оттенки серого
imshow(A_gray); %вывод серого изображения
figure; %создание нового графического окна
A_bw=(A_gray<180); %бинаризация изображения
imshow(A_bw); %вывод бинаризованного изображения в графическом
окне
```

В результате выполнения программы будет выведено два графического изображения, серого A_gray и черно-белого (логического) A_bw. Принципиально важно понять, как в примере изображение было бинаризовано:

```
A_bw=(A_gray<180); %бинаризация изображения
```

Запись A_gray<180 означает ,что мы выбираем пиксели изображения A_gray, яркость которых имеет значения меньше **180,** и рассматриваем их как логическую 1 для нового изображения A_bw. Логические 1 показываются функцией imshow() как белый цвет, 0 как черный цвет. Напомню, что яркость изображения 255 – это белый цвет (фон) исследуемой фотографии. Понятно, что в выборку A_gray<180 попадут самые и умеренно темные участки изображения (рис 28, слева). А если сделать выбор A_gray<120, то мы выберем уже наиболее темные объекты, а остальные будут игнорироваться (рис. 28, справа).

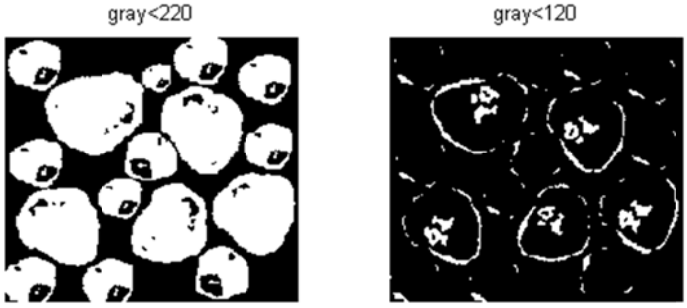


Рис. 28. Результат формирования бинарного изображения с различным порогом бинаризации: а – порог бинаризации 180; б – порог бинаризации **1**20.
Светлые части изображения соответствуют логической 1

Изменяющееся значение предельного значения оттенка серого называется *порогом бинаризации*. Порог бинаризации подбирается для каждой фотографии экспериментально.

После бинаризации исходного изображения заметно, что теряются внутренние части объектов. Необходимо найти способ их заполнить. Для этой цели нам помогут следующие морфологические операции логических изображений.

*Морфологическая эрозия* – морфологическое удаление пикселей, границ объектов логического изображения, равных 1 с помощью «стирателя» – структурного элемента, особой матрицы, создаваемого функцией *strel(n),* где n – размерность объекта.

*Морфологическая дилатансия* – морфологическое наращивание пикселей, прилегающих к объектам, с помощью аналогичного структурного элемента.

**Пример 2.** Заполнение внутренних частей объектов изображения.

Внимательное рассмотрение бинарного изображения из примера № 1 позволяет сделать вывод, что существенная часть объектов изображения потеряна.

Ввиду того, что Matlab и Octave не вполне совместимы, приводим код для них отдельно.

*Код Octave (Matlab):*

```
A_bw2 = edge(A_gray,'canny', 0.2, 2); %применяем алгоритм Canny
для
%поиска границ на изображении
%создаем структурный элемент, прямоугольник 2x2, которым будем
%воздействовать на изображение
se = strel('square',2);
%выполнение операции дилатансии,
%т.е. морфологического наращивания элементом se
A_bw2=imdilate(A_bw2,se);
%заполнение пустот на изображении
A_bw3 = bwfill( A_bw2, 'holes',8);
%создаем структурный элемент большего размера
se = strel('square',3);
%убираем имеющийся на изображении пиксельный "мусор"
%морфологической эрозией
A_bw3=imerode(A_bw3,se);
%выводим результат обработки изображения
imshow(A_bw3);
```

Код Matlab (в Octave 4 не выполняется):

```
%применяем алгоритм Canny для
%поиска границ на изображении
A_bw2 = edge(A_gray,'canny', 0.15, 2);
A_bw2 = imfill(A_bw2,'holes');
%strel – структурный элемент
se = strel('disk',1);
 %imopen морфолгическое открытие, эрозия после дилатансии
A_bw3 = imopen(A_bw2,se);
imshow(A_bw3);
```

В результате применения кода, показанного выше, будет получено бинарное изображение A_bw3, аналогичное (рис. 29).

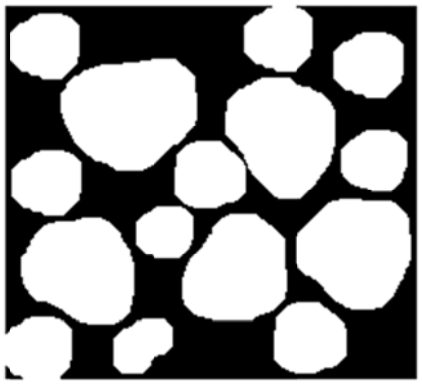

Рис. 29. Результат комбинации действий, направленных на выделение краев, заполнение границ, эрозии и дилатансии

Рассмотрение изображения (см. рис. 27) показывает, что цели преобразования изображения достигнуты. Выделена большая часть объектов, заполнены пустоты, обусловленные бликами. Специальная функция выделения кластеров бинарного изображения *bwlabel()* возвращает в качестве результата матрицу подписанных объектов и их количество.

Знание масштаба изображения (соответствие пикселя, например, мм$^2$) позволит нам рассчитать площадь объектов.

**Пример 3.** Функция *bwlabel.*

Эта функция позволяет на основе анализа бинарного (логического) изображения получить матрицу подписанных кластеров (взаимосвязанных точек) и их количество (число объектов):

```
[B,L] = bwlabel(A_bw2);
%B-матрица нумерованных кластеров, L-их число
imagesc(B); %это функция показа матрицы, а не изображе-
ния
title('Regions matrix');
disp('Number of regions=');
      disp(L); %вывод в командном окне количества кластеров
```

В результате выполнения кода выше будет выведено количество объектов на изображении и матрица пронумерованных кластеров (рис. 30)

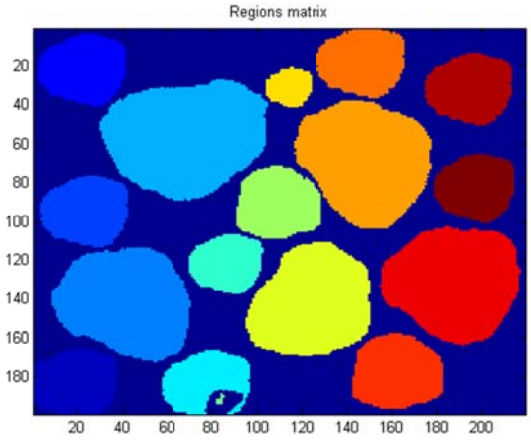


Рис. 30. Матрица кластеров объектов – результат выполнения функции *bwlabel()*. Объекты показаны цветами в соответствии с номером

Теперь можно использовать функцию *regionprops(),* (от англ. *region properties* – свойства региона) для вычисления свойств объектов. Эта функция может вычислить многие свойства, но нам потребуется только два: Centroid (координаты центра), Area (площадь кластера в пикселях). Свойства передаются особому объекту типа Structure (структура), который может их хранить.

**Пример 4.** Функция *regionprops ()*.

Эта функция позволяет на основе анализа бинарного (логического) изображения получить матрицу подписанных кластеров (взаимосвязанных точек) и их количество (число объектов):

```
stats = regionprops(B,'Centroid','Area');
imshow(A);
hold on; %это значит, следующий график будет поверх кар-
тинки
xc=[]; %пустые массивы для координат центров
yc=[];
%пустая переменная для хранения суммарной площади
sum_area=0;
for k = 1:L  %обход объекта stats
  xc = [xc stats(k).Centroid(1)]; %добавляем координаты
цента в их векторы xc и yc
 yc = [yc stats(k).Centroid(2)];
 sum_area=sum_area+stats(k).Area; %добавляем площадь k-
того объекта к sum_area
 end
%выводим график координат центра поверх
plot(xc,yc,'+');

disp ('Total area=');
disp (sum area);
```

Выполнение этого кода выведет нам координаты центра обработанных объектов поверх исходной картинки и общую площадь в командном окне.

**Пример 5.** Инвертирование.

Иногда полезным является инвертирование изображений. Операция инвертирования ~ превращает 1 изображения в 0 и наоборот.

Пример операции инвертирвоания в командном окне:

```
A= [0 1 0]
A =
   0   1   0

>> ~A
ans =
   1   0   1
```

**Рекомендации**

- Параметры приводимых функций могут изменяться в зависимости от ситуации, сжатый объем учебно-методического пособия не позволил привести их здесь вполне. Читателю рекомендуется обратиться к встроенной в программу справке и материалам интернета.
- 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 ).*
- Существуют и другие несовместимости. 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».

**Внимание!** Начало координат изображения в Matlab/Scilab/Octave – левый верхний угол.

**Задания для самостоятельной работы**

- Проверить код Matlab, найдя ошибки, препятствующие его нормальной работе (если такие имеются).
- Составить глоссарий применяемых в уроке функций Matlab с кратким описанием.
- Используя материалы примеров (приложение VIII), в зависимости от варианта **рассчитать** или **пористость образца**, или **количество объектов на микрофотографии**. Необходимо учитывать, что поры в породе могут быть заполнены, а могут быть и не заполнены нефтью. Вне зависимости от этого поры подлежат учету.

# 8. ОСНОВЫ РАБОТЫ С ГЕОИНФОРМАЦИОННЫМИ СИСТЕМАМИ (НА ПРИМЕРЕ ГИС QGIS)

**Цель работы:** ознакомление с некоторыми операциями, направленными на построение цифровой карты на примере географической информационной системы Quantum GIS (QGIS).

**Задание:** провести последовательное выполнение примеров, изучая интерфейс программы. Результаты выполнения самостоятельных разделов включить в отчет.

**Примечание.** Лабораторная работы выполняется с помощью бесплатной ГИС-программы QGIS (http://www.qgis.org), поддерживающей основные картографические форматы MapInfo (tab) и ArcMap (shp).

## 8.1. Пространственные данные. Понятия «*картографический слой*» и «*картографический проект*»

В компьютерной графике применяются различные форматы изображений – векторные (wmf, emf, ai, cdr, dxf и др.) и растровые (png, jpg, tiff, bmp, pcx и др). Перечень задач, решаемых с помощью этих форматов изображений, достаточно широк, они могут быть как дизайнерские, или изобразительные, так и инженерные проектные.

Коренное отличие программ цифровой картографии от дизайнерских состоит в наличии специфических функций, таких как возможность работать с картографическими системами координат и проекциями, наличии связей с базами данных и языка структурированных запросов (SQL), наличие специализированных форматов, содержащих геодезическую пространственную привязку (табл. 3).

Таблица 3

**Форматы изображений цифровой картографии**

| Тип файлов | Форматы |
|---|---|
| Векторный | TAB (MapInfo), MIF/MID (MapInfo); DXF (AutoCAD); SHP (ArcView) |
| Растровый | BMP; TIFF; GEOTIFF (формат tiff с геопривязкой в заголовке); JPEG; GIF, GRID (форма представления количественных данных по регулярной сети) |
| Табличные данные | XLS (Excel); DBF (dBase FoxPro); SLK; DAT; ASCII; TXT |

Цифровая карта в ГИС представляет собой файл *проекта* (в MapInfo *.wor, в ArcMap – *.mxd, QGIS – *.qgs) и подключенные файлы слоев, как векторных, так и растровых.

*Проект* содержит сведения о подключенных к проекту слоях, ссылки на их файлы на жестком диске, порядке расположения слоев (порядке прорисовке) и стилях их отображения, используемой картографической проекции.

Несмотря на значительное количество картографических программ, основными форматами, применяемыми для векторных слоев карт являются форматы ArcView SHP (англ. “shape” – форма) и MapInfo TAB (англ. “table” – таблица).

Один векторный слой в Mapinfo представлен четырьмя файлами, имеющими одно имя и расширения: *.tab, *.dat, *.map, *.id. Векторный слой ArcView представлен тремя файлами с расширением *.shp, *.shx, *.dbf.

Важным отличием SHP формата и QGIS является *топология слоя*. Это означает, что тот или иной векторный слой карты, а также связанные с ним векторные файлы спе-

циализированы на отдельный тип данных – точечный, полигональный, линейный. Не может быть слоя, содержащего точки и линии или линии и полигоны. **При обдумывании структуры проекта** это надо учитывать.

Слои *линейной топологии* могут быть использованы для объектов, ширина которых не учитывается (реки, дороги, ЛЭП и др.), *полигоны* – для объектов, имеющих выраженную площадь (леса, населенные пункты в масштабе карты, озера), *точки* – для объектов, площадь которых не рассматривается (населенные объекты, точки отбора проб и наблюдений и т.д.).

Достоинством картографических программ является экономия времени на верстку карты за счет гибкого изменения стилей объектов (рис. 31) на карте послойно, в том числе в соответствии со свойствами объектов, указанными в связанной базе данных – *атрибутивной таблице (АТ)*. Значительная производительность достигается, кроме того, за счет подписывания объектов на карте из *АТ*.

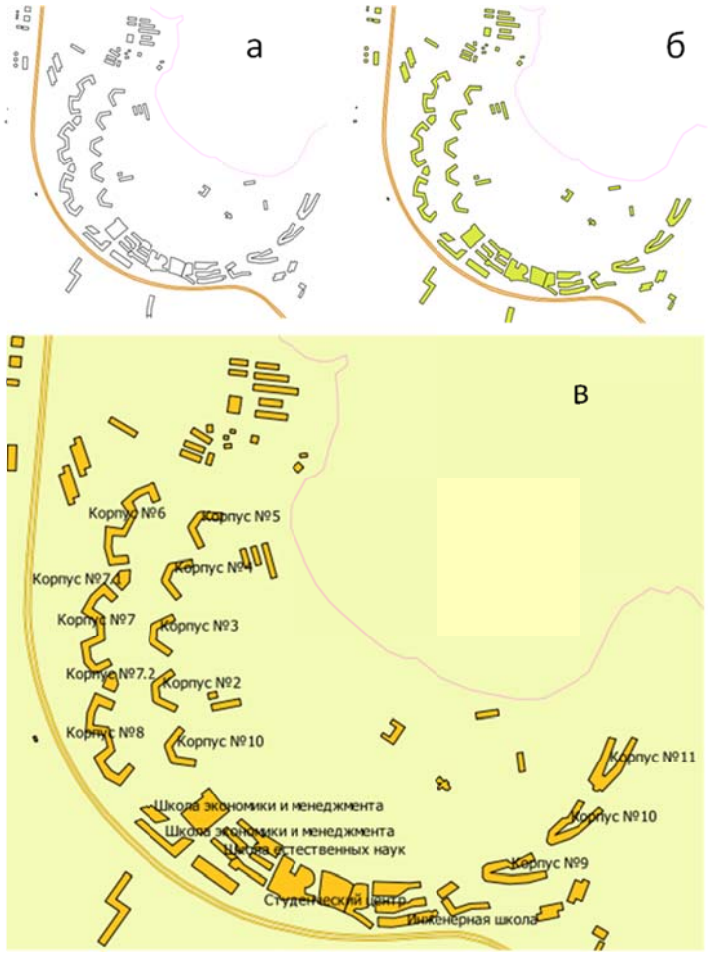


Рис. 31. Изменение стилей отображения карты (а, б)
и подписывание из атрибутивной таблицы (в) в ГИС QGIS

## 8.2. Интерфейс программы Quantum GIS

Несмотря на значительное наличие платных картографических программ на рынке, следует выделить бесплатную ГИС QGIS. Ее достоинством является поддержка форматов как MapInfo TAB, так и ArcView SHP и нетребовательность к ресурсам компьютера.

Интерфейс QGIS представляет собой типичное для программ такого вида расположение областей и панелей (рис. 32).

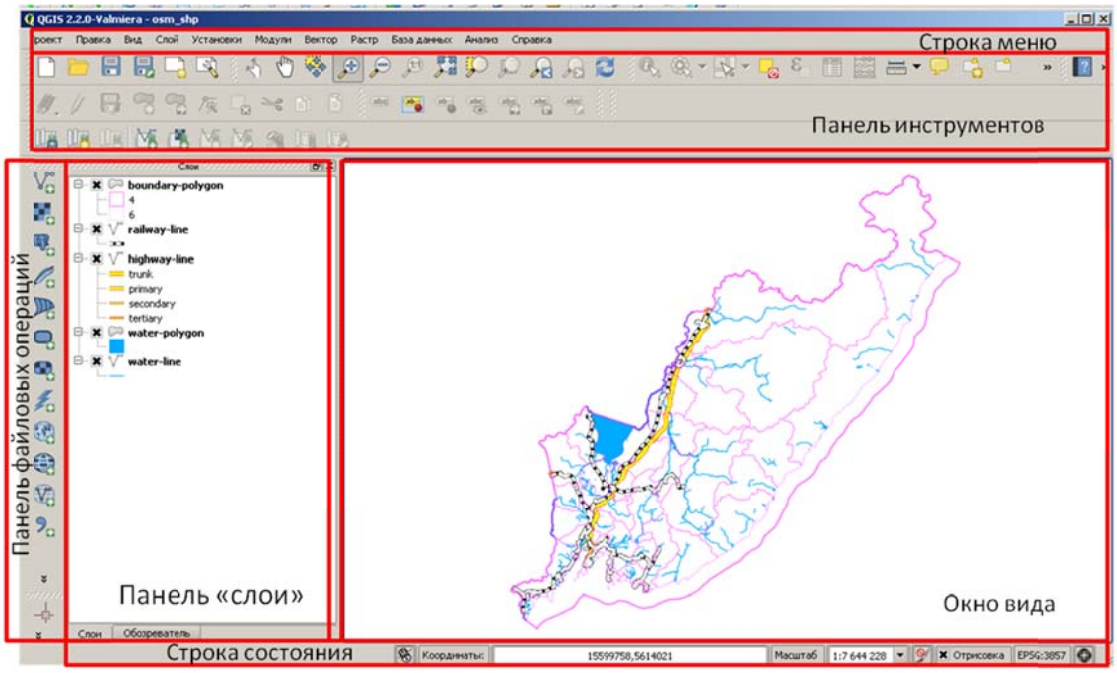


Рис. 32. Основные элементы интерфейса окна программы QGIS

**Строка меню** программы служит для выполнения операций с файлами, слоями проекта и настройки самого проекта. Объекты **панели инструментов** позволяют осуществлять навигацию проекта, изменение и создание новых слоев. **Панель «Слои»** предназначена для управления порядком прорисовки слоев, стилями их отображения. Многие операции осуществляются с помощью контекстного меню, выводимого по нажатию правой кнопкой мыши (ПКМ) на слое. **Окно вида** программы содержит компоновку слоев в виде карты таким образом, как она может быть экспортирована или напечатана. **Строка состояния** отражает координаты текущего положения курсора и масштаб карты.

Интерфейс программы претерпевает некоторые изменения от версии к версии, однако интуитивно управление меняется слабо.

### 8.3. Создание картографического проекта

**Важно!** Из сказанного выше очевидно, что даже единственный векторный слой карты представляет собой несколько файлов. Это значит, что проект карты будет представлять собой множество файлов, помимо собственно файла проекта (*.qgs). Для файла учебного картографического проекта, создаваемого в рамках самостоятельной работы, следует выделить отдельный каталог с подкаталогами, соответствующий следующей рекомендуемой структуре (рис. 33).


Рис. 33. Рекомендуемая структура каталогов проекта QGIS

Перед началом работы в отдельном каталоге необходимо создать следующие подкаталоги целевого назначения: shp – для векторных слоев, geotiff – для растровых слоев

с геопривязкой, doc – для документации и пояснительных файлов формата Microsoft Word, qgs – для файлов проектов QGIS.

Главное правило – отсутствие кириллических символов, пробелов и специальных символов (кроме заменяющего пробел знака «_») в именах файлов и каталогов, а также пути проекта.

Такого рода организация файлов позволит легко отыскивать требуемые файлы, препятствует захламлению каталогов и предупредит возникающие ошибки в работе программы.

Для начала работы QGIS необходимо запустить. В ОС Windows он находится в группе QGIS или OSGEO4W в меню «Пуск». После запуска программы с ярлыка и ее загрузки новый проект будет создан и открыт по умолчанию. Если запуск программы осуществлялся щелчком по имени уже существующего проекта, необходимо создать новый проект, нажав сочетание клавиш Ctrl+N или выполнив команду строки меню *Файл → Новый проект*.

Система координат проекта устанавливается в меню *Файл → Свойства проекта*. В наших упражнениях мы будем использовать систему координат, основанную на геоиде (модель Земли) WGS84.

Однако прежде чем настраивать систему координат проекта, следует добавить какие-либо слои в проект. Отметим, что для добавления картографического слоя SHP-формата надо выбрать файл с расширением *.shp (таблица 4).

Таблица 4

**Описание слоев, прилагаемых к проекту, проекция WGS 84/UTM zone 53N**

| Папка, имя «главного» файла слоя | Описание | Топология данных |
|---|---|---|
| shp\DNL_polyline.shp | Реки Приморского края | Линия |
| shp\dvfo_p_region.shp | Некоторые субъекты Дальневосточного ФО | Полигон |
| shp\PPP_point.shp | Точечные символы населенных пунктов Приморского края | Точка |
| shp\vsto2_polyline.shp | Ветка магистрального нефтепровода ВСТО-2 | Линия |
| geotiff\dv1.tif | Топографическая карта юга Приморского края, геопривязка | Растр |

**Пример 1.** Добавление данных в проект и выбор их стилей

В созданный при открытии программы проект добавим векторные данные shp\dvfo_p_region.shp и shp\DNL_polyline.shp

Выберем в меню *Слой → Добавить векторный слой*. Возникнет диалоговое окно доступа к файловой системе. Необходимо выбрать нужные слои и нажать кнопку «Открыть». После этого слои возникнут на панели «Слои», а их отображение – в **окне вида** (рис. 34).

Сразу после открытия слои могут выглядеть весьма своеобразно и далеко от привычной системы обозначений топокарт. Это означает, что необходимо выбрать их стили. Чтобы это сделать, надо произвести двойной щелчок на имени слоя, затем в появляющемся окне перейти на вкладку «Стиль» и выбрать подходящие цвета символа, заливки и контура (зависит от типа данных) (рис. 35).

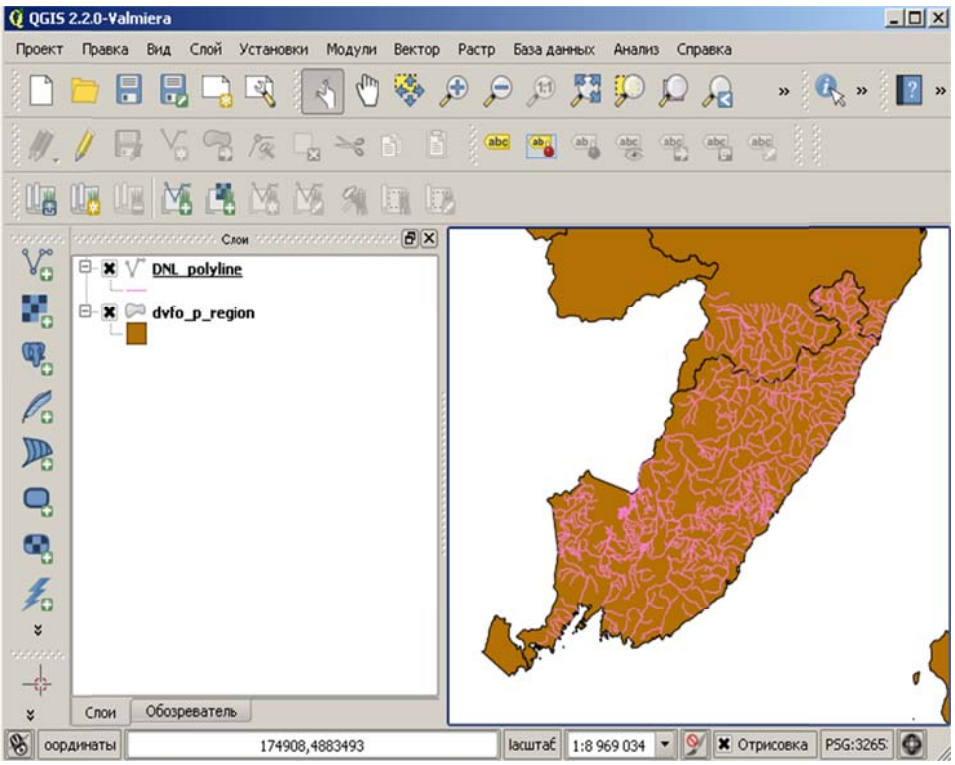


Рис. 34. Открытые слои в окне QGIS

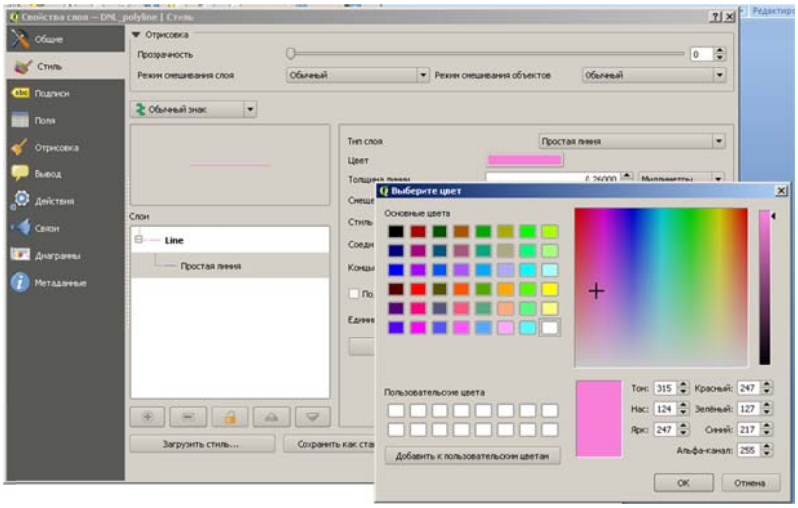


Рис. 35. Вкладка «Стиль» свойств слоя проекта

**Самостоятельно.** Установить для рек Приморья следующий стиль: сплошная тонкая линия, толщина 1 пикс, цвет RGB (0,45, 247). Для полигонов субъектов ДВФО убрать заливку, контур тонкий черный RGB (0,0,0)

**Пример 2.** Подписи слоев из АТ.

Объекты удобно подписывать из АТ (данные файла с расширением *.dbf). Это избавляет от необходимости подписывать каждый объект вручную. Выполним подписывание рек из таблицы атрибутов. Рассмотрим таблицу атрибутов. Ее вызов осуществляется нажатием ПКМ на имени слоя и выбором опции «Открыть таблицу атрибутов» или выделением слоя левой кнопкой мыши (ЛКМ) и выбором в меню *Слой → Открыть таблицу атрибутов* (рис. 36).

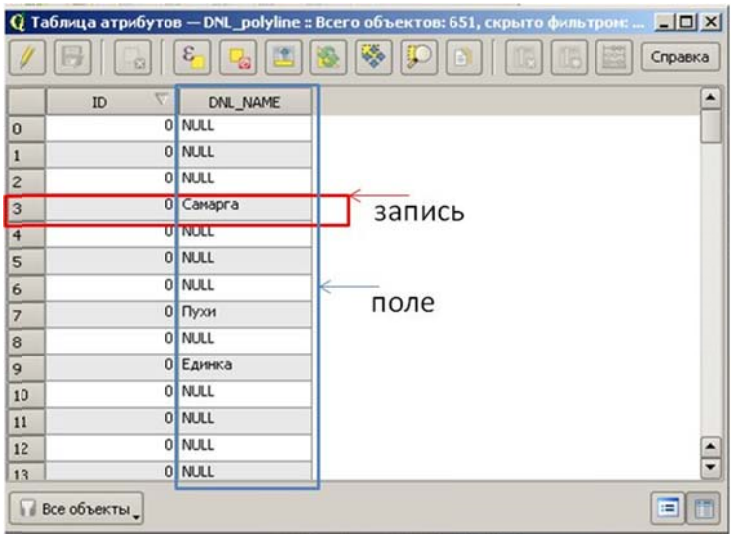


Рис. 36. Таблица атрибутов слоя DNL_polyline.shp. Выделено поле DNL_NAME

Анализ АТ слоя показывает, что она содержит два поля (столбца) и 651 записей (строк). Фактически, строка таблицы, т.е. запись в ней соответствует единичному объекту на карте, а столбец – какому-нибудь его свойству. Видно, что не все объекты поля DNL_NAME содержат названия рек. Значения типа NULL означают отсутствие данных (рис. 37).

Для подписывания названий рек слоя **DNL_polyline.shp** необходимо выделить его на панели слоев и в строке меню выбрать *Слой → Свойства.* В появившемся окне свойств выбрать вкладку «Подписи», там поставить галочку на опции «Подписывать объекты значением поля» и выбрать поле «DNL_name».

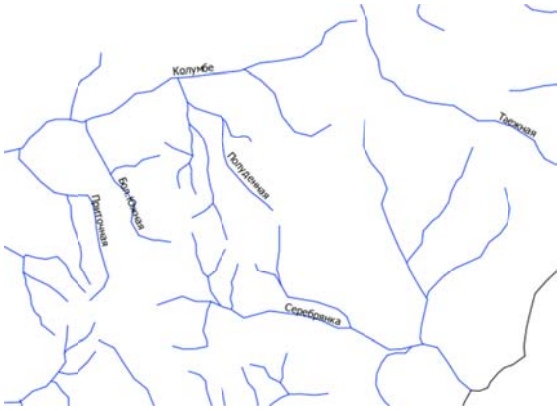


Рис. 37. Подпись на слое DNL_polyline.shp при приближении
(колесом мыши или инструментами масштабирования)

**Самостоятельно.** Добавить в проект слой *shp\PPP_point.shp* и подписать названия населенных пунктов из его атрибутивной таблицы.

### 8.4. Запросы к атрибутивной таблице слоя

Иногда бывает нужно выбрать объекты, соответствующие некоторому набору признаков, т.е. определенным значениям записей в таблице атрибутов. В ГИС-программах для этого бывают предусмотрены встроенные функции, такие как SQL-запросы (SQL, англ. *Structured Query Language* – язык структурированных запросов). В QGIS используется упрощенная система запросов с помощью графического интерфейса и составления простых выражений.

При формировании выражений необходимо учитывать тип данных (численный или строковый).

**Пример 3.** Поиск данных в АТ слоя. Выборка объектов по их атрибутам.

Выберем линейный объект слоя реки (файл **DNL_polyline.shp)** по его имени и приблизим карту к нему. Откроем атрибутивную таблицу слоя **DNL_polyline.shp** (строка меню *Слой → Открыть таблицу атрибутов*). Откроется окно атрибутивной таблицы (см. рис. 36). Выберем реку Единка по названию и приблизим к ней карту. Для этого надо нажать на кнопку  («Выделить объекты, удовлетворяющие условию») на панели инструментов атрибутивной таблицы. Появится окно составления запроса под названием «Выделение выражением» (рис. 38).

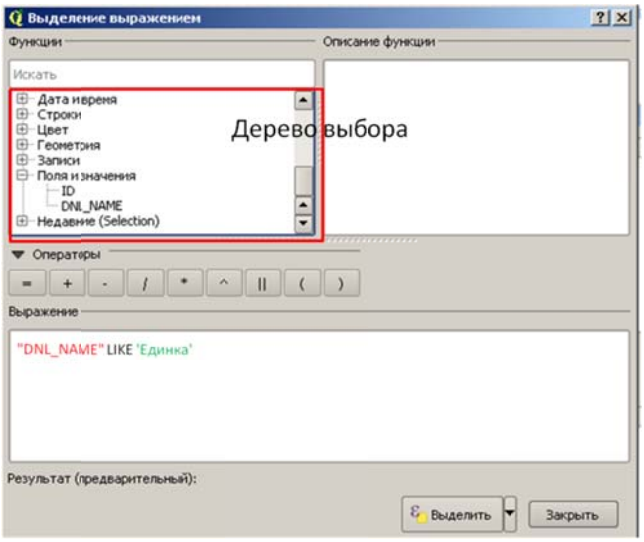


Рис. 38. Выделение выражением – выбор реки Единка слоя DNL_polyline.shp

Запрос вводится по следующему шаблону: ИМЯ_ПОЛЯ ОПЕРАТОР ЗНАЧЕНИЕ ЗАПИСИ в текстовом поле «Выражение». Для того чтобы посмотреть совместимые операторы и список имеющихся полей, можно воспользоваться «деревом выбора». Для выбора текстовых значений используется оператор «LIKE», для выбора чисел используется оператор «=». Введем выражение:

```
"DNL_NAME" LIKE 'Единка'
```

После нажатия на кнопку «Выделить» произойдет выделение строки в таблице, при условии, что в выражении отсутствуют ошибки и правильно введено значение атрибута. При этом в *строке состояния* программы (нижний левый угол) будет написано «*В слое DNL_polyline выделен 1 объект*» (рис. 39).

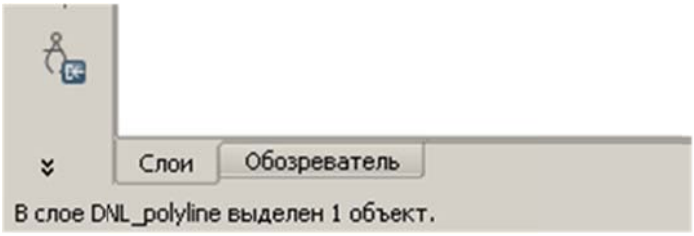


Рис. 39. Сообщение о выделении объекта в строке состояния

С помощью инструментов центрирования и увеличения можно управлять отображением выделенных объектов на карте, увеличивать их и приближать (рис. 40).

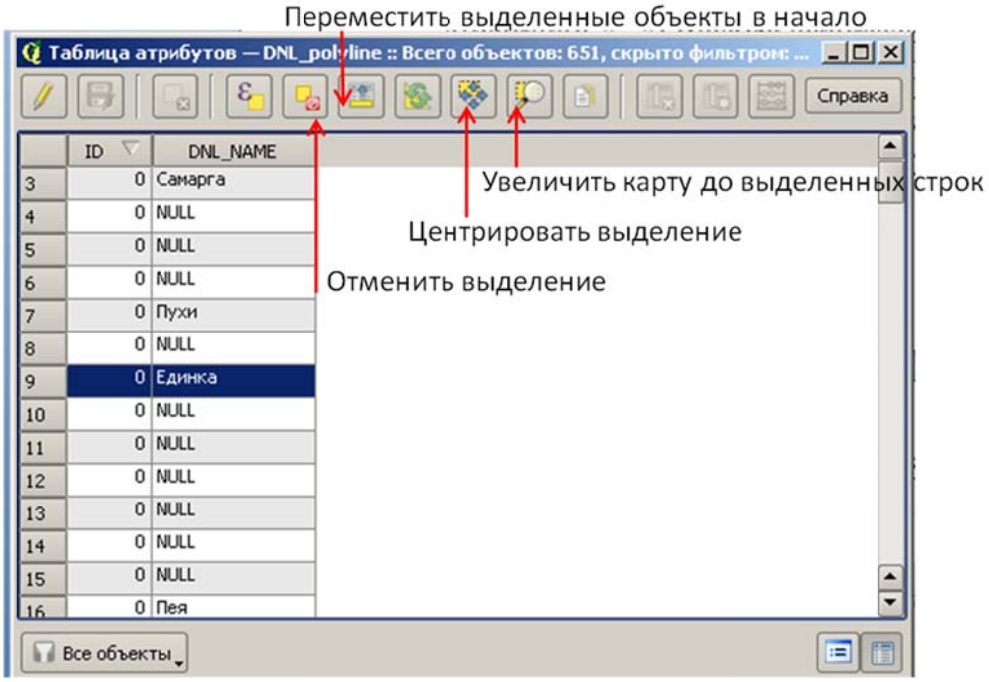


Рис. 40. Выделенный слой в таблице

### 8.5. Калькулятор полей слоя

Объекты слоя, выбор которых осуществлялся в п. 8.4, оцифровывались без простановки порядковых номеров объектов в поле ID, которое заполнено 0. Это можно исправить с помощью *Калькулятора полей* слоя. В результате работы калькулятора полей слой будет изменен. Для начала редактирования необходимо перевести слой в специальный режим. Для этого надо нажать на кнопку  панели инструментов основного окна программы или окна атрибутивной таблицы. Кнопка притапливается, что означает перевод слоя в *режим редактирования*. В этом режиме можно вносить изменения в объекты карты, создавать и удалять их, изменять атрибуты объектов (рис. 41).

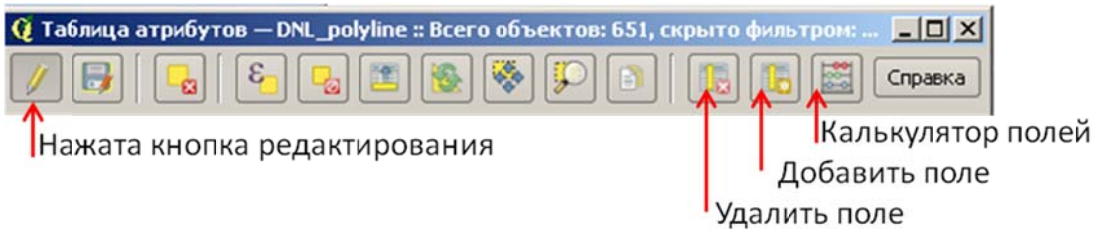


Рис. 41. Дополнительные кнопки режима редактирования слоя

Окно *калькулятора полей* содержит дерево выбора функций, геометрических параметров объектов, а также областей выбора параметров редактирования поля (рис. 40).

Следует обратить внимание на область «Настройка режима редактирования», содержащую выбор «Обновление выделенных объектов», «Создание нового поля», «Обновление существующего поля».

Возможно автоматизированное заполнение поля какими-либо параметрами объектов, выбираемыми в дереве объектов, – геометрическими свойствами (площадью $area, длиной $length), а также параметрами записи, например, порядковым номером строки

в таблице, отражающим последовательность создания объектов, – $rownum. Выбор опций, показанных на рис. 42, приведет к перезаписи значений поля ID значениями номера строки $rownum.

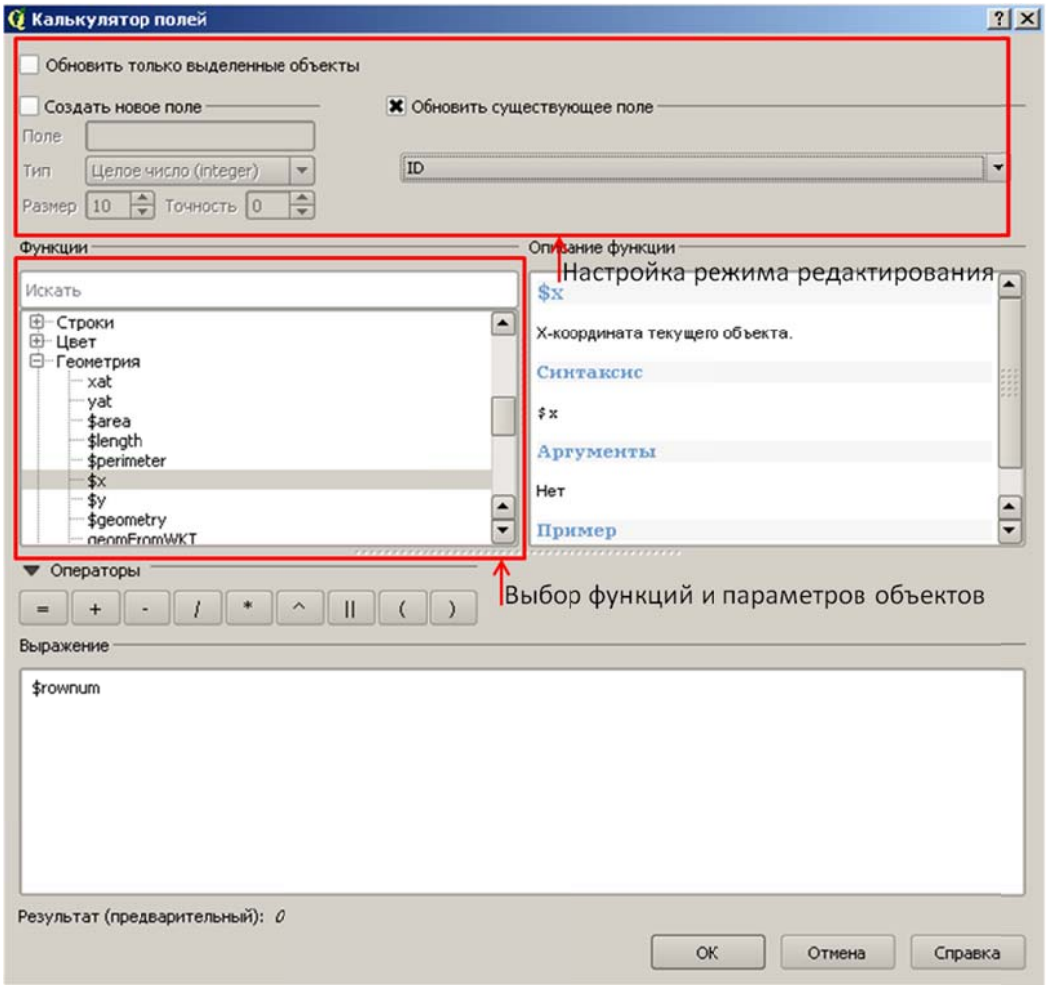


Рис. 42. Окно средства калькулятор полей

**Самостоятельно**. Использовать калькулятор полей для обновления значений поля ID номерами строк, а также создания нового поля LEN, содержащего значения длины реки. Какой формат поля рационально задать при его создании? В каких единицах программа возвращает данные, включаемые в таблицу? От чего зависит формат данных?

### 8.6. Пространственные запросы

Пространственные запросы в ГИС-программах выполняются для проверки пересечения объектов карты одного слоя другим слоем и служат для решения множества прикладных задач. Для вызова окна запроса необходимо выполнить действия меню основного окна программы *Вектор →Пространственный запрос → Пространственный запрос.* Возникает окно запроса (рис. 43).

Для проектирования магистральных трубопроводов с применением цифровой картографии может потребоваться выяснить количество пересекаемых трассой рек и получить их список. После выполнения этого пространственного запроса в атрибутивной таблице выделяется список объектов. Кроме того, они показаны на карте особым цветом.

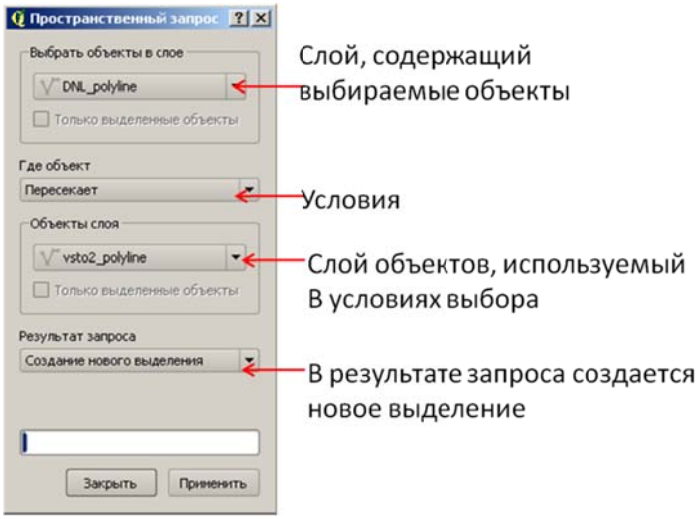


Рис. 43. Окно пространственного запроса

### 8.7. Создание слоя и редактирование его объектов

Создание нового слоя осуществляется с помощью команды *Слой →Создать → Создать шейп-файл.* Это приводит к выводу окна создания нового шейп-файла (SHP) (рис. 44).

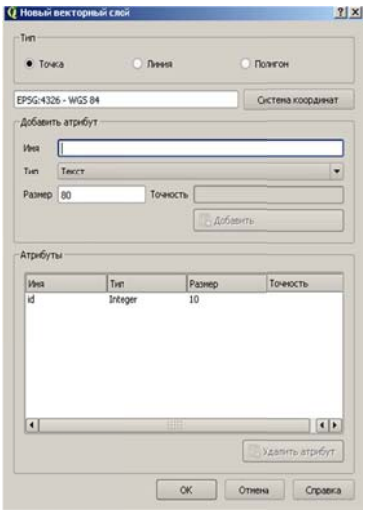


Рис. 44. Окно создания нового слоя карты

В этом окне необходимо задать тип данных (топологию), соответствующих природе оцифровываемых объектов, систему координат слоя, а также список атрибутов слоя (полей атрибутивной таблицы) и их типы данных.

Редактирование слоя производится в соответствующем режиме с помощью инструментов панели редактирования  .

Для создания новых объектов важно достичь **«прилипания» (Snapping)** вершин для создания топологически корректной модели. Это выражается, например, в том, что линии пересекающихся трубопровода и его отвода или главной реки и ее притока на карте должны иметь общую точку.

Несмотря на очевидность задачи, поставить общую точку «вручную» невозможно, при увеличении карты обязательно будет заметен «недоввод» или «перехлест» объектов. Поэтому, важно, чтобы при создании/редактировании объектов, курсор как бы прилипал

к стороне или вершине линии, обеспечивая точный ввод. Настройка параметров прилипания настраивается в пункте меню *Установки →Параметры прилипания* (рис. 45).

Это окно также позволяет включать режим «топологического редактирования» установкой соответствующей галочки. Этот режим позволяет одновременно перемещать вершину, принадлежащую сразу нескольким объектам, что не нарушает целостность карты.

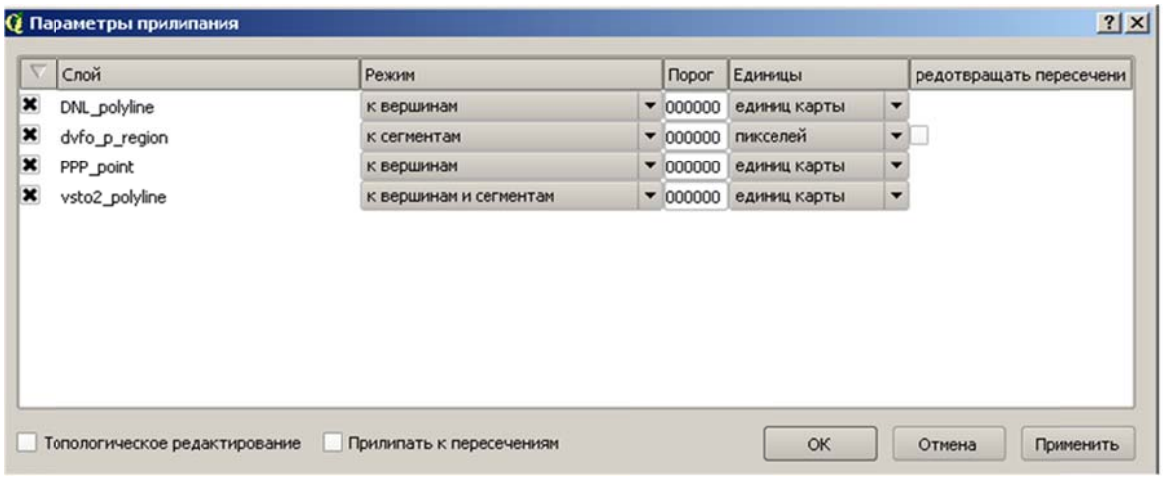


Рис. 45. Окно «Параметры прилипания»

### Рекомендации

- Учитывать, что операция отмены или сочетание клавиш Ctrl+Z в ГИС-системах малополезна, т.к. при редактировании проекта обрабатываются множество файлов и отменить ошибочное действие невозможно. Решение – внимательно читать предупреждающие сообщения программы.
- Редактирование и сохранение проектов QGS и добавленных в них файлов происходит независимо друг от друга, сохранение проекта не приводит к сохранению слоя, и наоборот.
- Можно редактировать несколько слоев одновременно, но это не рекомендуется для начинающих пользователей.

### Задания для самостоятельной работы

- Кроме приведенных примеров, выполнить основные задания под руководством преподавателя.
- Ход работы и результаты конспектировать, фиксируя основные методические приемы.
- Задания для самостоятельной работы должны включать оцифровку объектов космического снимка, указанных преподавателем.

### Требования к отчету

Отчет о выполнении лабораторной работы должен содержать:

- тему и цель лабораторной работы на титульной странице;
- оглавление;
- необходимые теоретические сведения по теме решаемых задач;
- исходные данные и последовательность их обработки (преобразования);
- поэтапный расчет в виде вставок кода Scilab в объектах «Подпись»;
- результаты выполнения кода Scilab в виде вывода в командном окне;
- графики, выполненные в Scilab;
- интерпретацию результатов работы моделей;
- отметку преподавателя о выполнении лабораторной работы.

## Литература


1. Алексеев Е.Р., Чеснокова О.В., Рудченко Е.А. Scilab: Решение инженерных и математических задач. М.: ALT Linux; БИНОМ. Лаборатория знаний, 2008. 260 с.
2. Павлова М.И. Руководство по работе с пакетом SciLab 2.6. 2003. 200 с.
3. Поршнев С. В. Компьютерное моделирование физических процессов в пакете MATLAB. М.: Горячая линия – Телеком, 2003. 593 с.
4. Эдвардс Ч., Пенни Д. Дифференциальные уравнения и краевые задачи: моделирование и вычисление с помощью *Mathematica, Mapple и Matlab*. М.: Вильямс, 2008. 1104 с.
5. Алексеев Е.Р., Чеснокова О.В. Введение в Octave для инженеров и математиков. М.: ALT Linux, 2012. 368 с.
6. Материалы по продуктам MATLAB & Toolboxes // Математический сайт Exponenta.ru. URL: http://matlab.exponenta.ru/index.php (дата обращения: 05.01.2018).
7. An Introduction to Reservoir Simulation Using MATLAB. User Guide for the Matlab Reservoir Simulation Toolbox (MRST). Department of Applied Mathematics Oslo, Norway. SINTEF ICT, 2014. 213 p.
8. QGIS User guide. Официальное руководство к QGIS 2.8. URL: https://docs.qgis.org/2.8/pdf/en/QGIS-2.8-UserGuide-en.pdf (дата обращения: 20.01.2018).

**Приложение I**

## Варианты совокупностей случайных величин

$X_1$, $X_2$ и $Y$ – соответственно плотность (кг/м$^3$), кинематическая вязкость (сСт) и содержание парафинов (%); m – номер замера.

| | **Вариант 1** | | | **Вариант 2** | | | **Вариант 3** | | | **Вариант 4** | | | **Вариант 5** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***m*** | $X_1$ | $X_2$ | $Y$ | $X_1$ | $X_2$ | $Y$ | $X_1$ | $X_2$ | $Y$ | $X_1$ | $X_2$ | $Y$ | $X_1$ | $X_2$ | $Y$ |
| 1 | 952 | 13,7 | 9,9 | 931 | 13,5 | 8,4 | 983 | 13,7 | 11,3 | 983 | 13,7 | 9,3 | 980 | 13,7 | 10,5 |
| 2 | 952 | 13,4 | 8,7 | 922 | 13,4 | 10,8 | 980 | 13,4 | 13,3 | 1133 | 13,4 | 13,3 | 840 | 13,4 | 9,4 |
| 3 | 982 | 11,3 | 11,3 | 953 | 15,3 | 11,4 | 950 | 11,3 | 10,9 | 940 | 11,3 | 8,3 | 1213 | 11,3 | 11,3 |
| 4 | 930 | 12,5 | 7,4 | 910 | 14,8 | 9,0 | 950 | 13,5 | 11,3 | 950 | 13,5 | 10,4 | 1103 | 13,5 | 9,4 |
| 5 | 922 | 15,1 | 8,1 | 950 | 13,1 | 10,8 | 923 | 15,1 | 10,9 | 1033 | 15,1 | 9,8 | 933 | 15,1 | 9,4 |
| 6 | 920 | 17,4 | 9,9 | 932 | 10,3 | 10,3 | 983 | 17,4 | 11,3 | 853 | 17,4 | 8,3 | 890 | 17,4 | 8,3 |
| 7 | 982 | 14,0 | 10,3 | 920 | 13,3 | 11,4 | 930 | 14,0 | 8,3 | 1173 | 14,0 | 10,4 | 953 | 14,0 | 10,5 |
| 8 | 901 | 14,5 | 8,7 | 953 | 14,3 | 13,0 | 973 | 14,5 | 13,3 | 853 | 14,5 | 11,5 | 930 | 14,5 | 10,0 |
| 9 | 950 | 15,3 | 11,3 | 923 | 12,2 | 10,3 | 933 | 15,3 | 13,3 | 1173 | 15,3 | 11,5 | 953 | 15,3 | 9,4 |
| 10 | 950 | 14,8 | 9,3 | 903 | 13,4 | 8,4 | 943 | 14,8 | 9,0 | 1030 | 14,8 | 10,4 | 1173 | 14,8 | 11,3 |
| | **Вариант 6** | | | **Вариант 7** | | | **Вариант 8** | | | **Вариант 9** | | | **Вариант 10** | | |
| ***m*** | $X_1$ | $X_2$ | $Y$ | $X_1$ | $X_2$ | $Y$ | $X_1$ | $X_2$ | $Y$ | $X_1$ | $X_2$ | $Y$ | $X_1$ | $X_2$ | $Y$ |
| 1 | 822 | 12,6 | 19,1 | 1161 | 12,7 | 9,2 | 1122 | 15,1 | 9,1 | 962 | 11,8 | 9,1 | 1221 | 12,6 | 9,6 |
| 2 | 1172 | 12,2 | 10,7 | 1121 | 12,2 | 8,7 | 1191 | 19,1 | 11,2 | 1172 | 16,7 | 11,9 | 1172 | 12,2 | 9,1 |
| 3 | 1192 | 16,2 | 11,1 | 821 | 11,2 | 9,2 | 1472 | 12,2 | 11,2 | 1181 | 12,1 | 9,1 | 1172 | 16,2 | 11,2 |
| 4 | 1112 | 12,8 | 13,7 | 1122 | 12,6 | 11,2 | 1031 | 18,1 | 9,6 | 1272 | 17,1 | 12,7 | 1312 | 12,8 | 11,2 |
| 3 | 1122 | 12,1 | 11,2 | 1171 | 16,1 | 12,2 | 1041 | 12,7 | 8,2 | 971 | 12,1 | 13,2 | 911 | 12,1 | 8,2 |
| 6 | 1122 | 11,2 | 14,1 | 1262 | 17,2 | 13,2 | 1072 | 17,7 | 9,1 | 1172 | 12,9 | 9,7 | 792 | 11,2 | 9,1 |
| 7 | 992 | 12,2 | 10,7 | 1231 | 12,1 | 11,2 | 1092 | 13,2 | 11,8 | 1172 | 12,1 | 11,2 | 822 | 12,2 | 11,8 |
| 8 | 882 | 12,2 | 9,2 | 922 | 12,6 | 8,2 | 1072 | 16,2 | 9,6 | 1176 | 12,2 | 9,1 | 962 | 15,2 | 13,2 |
| 9 | 966 | 16,1 | 8,8 | 1691 | 16,6 | 9,9 | 796 | 9,3 | 7,8 | 1116 | 16,2 | 11,9 | 1166 | 15,1 | 9,1 |
| 10 | 1176 | 16,2 | 12,1 | 1061 | 12,8 | 14,6 | 1201 | 12,2 | 11,2 | 1191 | 16,2 | 12,1 | 1121 | 16,2 | 11,2 |
| | **Вариант 11** | | | **Вариант 12** | | | **Вариант 13** | | | **Вариант 14** | | | **Вариант 15** | | |
| ***m*** | $X_1$ | $X_2$ | $Y$ | $X_1$ | $X_2$ | $Y$ | $X_1$ | $X_2$ | $Y$ | $X_1$ | $X_2$ | $Y$ | $X_1$ | $X_2$ | $Y$ |
| 1 | 1310 | 16,7 | 9,6 | 1136 | 16,0 | 11,9 | 966 | 10,8 | 8,3 | 760 | 9,1 | 8,8 | 1176 | 16,3 | 11,1 |
| 2 | 1270 | 16,9 | 11,3 | 996 | 13,1 | 10,6 | 1136 | 13,8 | 9,0 | 966 | 16,6 | 11,9 | 906 | 11,1 | 8,6 |
| 3 | 936 | 13,3 | 7,7 | 936 | 11,6 | 8,8 | 1070 | 16,6 | 11,0 | 860 | 11,3 | 10,6 | 1210 | 13,6 | 10,4 |
| 4 | 1120 | 16,1 | 11,9 | 900 | 11,3 | 10,6 | 1046 | 13,6 | 9,6 | 17,30 | 17,6 | 11,9 | 1336 | 16,3 | 10,6 |
| 5 | 1060 | 16,6 | 11,3 | 1160 | 16,1 | 11,9 | 1065 | 13,8 | 10,1 | 1072 | 14,6 | 11,0 | 1526 | 16,0 | 11,7 |
| 6 | 1130 | 16,7 | 9,6 | 1162 | 17,0 | 11,6 | 1076 | 18,6 | 11,3 | 1020 | 13,1 | 9,3 | 970 | 11,7 | 9,1 |
| 7 | 1066 | 16,3 | 10,1 | 1060 | 13,7 | 11,3 | 966 | 11,1 | 9,0 | 1076 | 13,6 | 11,3 | 1020 | 16,0 | 11,3 |
| 8 | 1301 | 13,1 | 11,6 | 836 | 11,6 | 9,3 | 910 | 11,6 | 10,1 | 1106 | 16,5 | 11,6 | 1010 | 16,3 | 11,7 |
| 9 | 1130 | 16,9 | 8,9 | 860 | 13,1 | 9,3 | 865 | 13,3 | 10,8 | 970 | 13,1 | 9,3 | 836 | 10,6 | 9,1 |
| 10 | 876 | 11,7 | 8,3 | 1116 | 16,8 | 11,9 | 1010 | 13,0 | 10,8 | 1066 | 13,8 | 10,0 | 960 | 13,7 | 9,9 |

**Приложение II**

## Критические значения критерия Смирнова–Граббса для уровня доверительной вероятности

| Количество элементов совокупности, *n* | Уровень доверительной вероятности, $P_D$ | | |
|---|---|---|---|
| | **0,99** | **0,95** | **0,90** |
| 3 | 1,414 | 1,412 | 1,406 |
| 4 | 1,723 | 2,689 | 1,645 |
| 5 | 1,955 | 1,869 | 1,791 |
| 6 | 2,130 | 1,996 | 1,894 |
| 7 | 2,265 | 2,093 | 1,974 |
| 8 | 2,374 | 2,172 | 2,041 |
| 9 | 2,464 | 2,237 | 2,097 |
| 10 | 2,540 | 2,294 | 2,146 |
| 11 | 2,606 | 2,343 | 2,190 |
| 12 | 2,663 | 2,387 | 2,229 |
| 13 | 2,714 | 2,426 | 2,664 |
| 14 | 2,739 | 2,461 | 2,297 |
| 15 | 2,800 | 2,493 | 2,326 |
| 16 | 2,837 | 2,523 | 2,354 |
| 17 | 2,871 | 2,551 | 2,380 |
| 18 | 2,903 | 2,577 | 2,404 |
| 19 | 2,932 | 2,600 | 2,426 |
| 20 | 2,959 | 2,623 | 2,447 |
| 21 | 2,934 | 2,644 | 2,4 67 |
| 22 | 3,008 | 2,664 | 2,486 |
| 23 | 3,030 | 2,683 | 2,504 |
| 24 | 3,051 | 2,701 | 2,502 |
| 25 | 3,071 | 2,717 | 2,537 |

**Приложение III**

## Варианты исходных данных для решения транспортной задачи

$a_1$, $a_2$ – наличие ДТ на первой и второй нефтебазах, т; $b_1$, $b_2$, $b_3$ – потребность 1, 2 и 3-й АЗС в топливе, т; $c_{ij}$ – стоимость доставки топлива с i склада на j АЗС.

| Вари-ант | $a_1$ | $a_2$ | $b_1$ | $b_2$ | $b_3$ | $c_{11}$ | $c_{12}$ | $c_{13}$ | $c_{21}$ | $c_{22}$ | $c_{23}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 108 | 162 | 81 | 135 | 54 | 6 | 3 | 4 | 6 | 10 | 8 |
| 2 | 160 | 240 | 120 | 200 | 80 | 5 | 6 | 7 | 7 | 4 | 8 |
| 3 | 132 | 198 | 99 | 165 | 66 | 4 | 7 | 6 | 4 | 7 | 10 |
| 4 | 120 | 180 | 90 | 150 | 60 | 9 | 4 | 5 | 5 | 4 | 5 |
| 5 | 104 | 156 | 78 | 130 | 52 | 6 | 4 | 5 | 10 | 9 | 10 |
| 6 | 128 | 192 | 96 | 160 | 64 | 9 | 8 | 6 | 7 | 6 | 6 |
| 7 | 132 | 198 | 99 | 165 | 66 | 8 | 3 | 8 | 9 | 5 | 7 |
| 8 | 120 | 180 | 90 | 150 | 60 | 3 | 3 | 8 | 5 | 10 | 10 |
| 9 | 120 | 180 | 90 | 150 | 60 | 7 | 9 | 8 | 8 | 4 | 4 |
| 10 | 148 | 222 | 111 | 185 | 74 | 3 | 5 | 4 | 5 | 10 | 7 |
| 11 | 112 | 168 | 84 | 140 | 56 | 9 | 6 | 8 | 7 | 4 | 9 |
| 12 | 116 | 174 | 87 | 145 | 58 | 3 | 3 | 3 | 8 | 5 | 10 |
| 13 | 128 | 192 | 96 | 160 | 64 | 8 | 8 | 5 | 8 | 8 | 4 |

**Приложение IV**

## Варианты исходных данных для решения задачи разрядки батареи

| Вариант | $U_0$, V | $U_n$,V | n, мин |
|---|---|---|---|
| 1 | 19 | 17 | 8 |
| 2 | 20 | 18 | 5 |
| 3 | 21 | 19 | 5 |
| 4 | 22 | 20 | 8 |
| 5 | 16 | 14 | 8 |
| 6 | 16 | 14 | 7 |
| 7 | 20 | 18 | 7 |
| 8 | 19 | 17 | 8 |
| 9 | 18 | 16 | 8 |
| 10 | 21 | 19 | 5 |
| 11 | 17 | 15 | 11 |
| 12 | 19 | 17 | 7 |
| 13 | 21 | 19 | 12 |
| 14 | 17 | 15 | 9 |

**Приложение V**

**Варианты исходных данных для решения задачи зарядки конденсатора**

| Вариант | R, Ом | C, $10^{-n}$ Ф | V, В |
|---|---|---|---|
| 1 | 2300 | 4 | 5 |
| 2 | 2400 | 4 | 5 |
| 3 | 1900 | 5 | 10 |
| 4 | 2400 | 4 | 11 |
| 5 | 1600 | 5 | 7 |
| 6 | 2100 | 5 | 5 |
| 7 | 1500 | 5 | 7 |
| 8 | 2400 | 5 | 8 |
| 9 | 1700 | 5 | 8 |
| 10 | 1700 | 5 | 7 |
| 11 | 2000 | 5 | 6 |
| 12 | 1800 | 6 | 9 |
| 13 | 1700 | 4 | 6 |
| 14 | 1500 | 5 | 8 |

**Приложение VI**

**Варианты исходных данных для решения самостоятельной задачи**

Параметры: $\rho_{f,}$ № 1 – плотность жидкости № 1, $\rho_{f,}$ № 2 – плотность жидкости № 2, $\rho_p$ – плотность дробинки.

| Вариант | $\rho_{f,}$ № 1 | $\rho_{f,}$ № 2 | $\rho_p$ |
|---|---|---|---|
| 1 | 550 | 2500 | 9000 |
| 2 | 825 | 4480 | 10083 |
| 3 | 597 | 3999 | 9912 |
| 4 | 526 | 3205 | 8012 |
| 5 | 425 | 2324 | 8561 |
| 6 | 756 | 2293 | 10051 |
| 7 | 582 | 4066 | 9714 |
| 8 | 408 | 3282 | 7919 |
| 9 | 579 | 2143 | 9800 |
| 10 | 368 | 4026 | 9507 |
| 11 | 581 | 2392 | 10306 |
| 12 | 386 | 4149 | 8249 |
| 13 | 464 | 2877 | 7142 |
| 14 | 678 | 2407 | 9185 |

**Приложение VII**

**Код программы «Падение дробинки вжидкости»**

```
%высота "стакана" h=250 м. Масса дробинки m=0.010 кг. Rho [kg/m3]
glass=zeros(250,50); %reservoir size of 250x50 px  environment, gravity
%gravity constant fluid init, density of fluid pellet initial state and position
g=9.81; w_Rho=1000; p_Rho=11000;
%coordinates initial speed  size  mass volume
p_xy=[25 1]; p_sp=0; p_sz=2E-6; p_m=0.050; p_V=p_m/p_Rho; %
imshow(glass); %output on start
 %data logging
%arrays for pellet state logging
sec_x=[]; speed_y=[]; deep_y=[];
 %2 forces act in that model
%Fp=mg  Fa=Rho_g_V - gravity and Archimeds forces description
Fp=p_m*g; Fa=w_Rho*g*p_V; delta_F=Fa-Fp;
%<0 sinks, >0 floats
p_a=abs(delta_F)/p_m; %pellet's acceleration, nonzero, abs(delta_F)
t=0.1; %discretisation of time in experiment 0.1 second
size_glass=size(glass); %reservoirs height and width
for i=1:100 %100 falling of 100 frames max
 p_sp=p_sp+p_a; %speed acceleration
 dS=p_sp*t;  %speed delta
 if (delta_F<0) %pellet sinks, because Fa<Fg
 p_xy=p_xy+[0 round(dS)]; %enlarge depth
 end;
 if (delta_F>0) %pellet sinks, because Fa<Fg
    disp('Pellet is floating!)
  if p_xy(2)>size_glass(1) %checking of bottom reaching
    disp ('bottom is reached');
    break; %break up the cycle in that case
 end;
 %preparation of animated frame before output
 pict_out=glass;  pict_out(p_xy(2),p_xy(1))=1;
  %changing frame in visual picture
 imagesc(pict_out);
 %data logging
 sec_x=[sec_x (i*t)]; speed_y=[speed_y p_sp]; deep_y=[deep_y p_xy(2)];
 %pict heading
 heading=strcat('Frame ' num2str(i),' of 100; ', 'T [sec] =', num2str((i*t)));
 title(heading);
 pause (1);
end;
 %final graphics output
figure; subplot(2,1,1); plot(sec_x,speed_y);
title ('Speed changes with time'); xlabel('x, time, s'); ylabel('V(x), speed, [m/s]');
subplot(2,1,2); plot(sec_x,deep_y); title ('Deep changes with time'); xlabel('x, time [s]');
ylabel('h(x), deep [m]');
```

**Приложение VIII**

## Варианты исходных данных для решения самостоятельной задачи

п – поры, о – объекты. Номер варианта соответствует номеру файла jpg

| Вариант | Описание | Примечание |
|---|---|---|
| 1 | Высокопористый водорослевый известняк. Глубина 1700 м. Материал – багряные водоросли. Поры – «прозрачные области изображения». | п |
| 2 | Мрамор с керитом (битуминозное вещество). Поляризованный свет. Поры – «темные области изображения». | п |
| 3 | Материал – пенобетон. Рассчитать пористость по наиболее контрастной части изображения. | п |
| 4 | Трещиноватый нефтеносный известняк. Определить пористость (трещиноватость) породы. | п |
| 5 | Комковатый известняк с вторичным цементом. Битум, заполняющий поры, темно-коричневый. | п |
| 6 | Перекристаллизованный известняк битуминозный. Битум темно-коричневого цвета заполняет поры. | п |
| 7 | Пемза вулканическая. | п |
| 8 | Золотины из россыпи р. Анабар. | о |
| 9 | Раковинки фораминифер – морских одноклеточных простейших. | о |
| 10 | Трещиноватый металл. Разрушенный коррозией материал. | п |
| 11 | Трещиноватый металл. Разрушенный усталостью материал. | п |
| 12 | Битуминозный песчаник. Битум темно-коричневый. | п |