МИНОБРНАУКИ РОССИИ

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

Кафедра биомедицинской инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__25__» ___09_____ 2023 г.

ВВЕДЕНИЕ В MATLAB

Методические указания по выполнению практических занятий
для студентов специальности
30.05.03 «Медицинская кибернетика»

Курск 2023

УДК 004.93:61

Составитель: О.В. Шаталова.

Рецензент
Кандидат технических наук, доцент *М.А. Ефремов*

Введение в MATLAB: методические указания по выполнению практических занятий для студентов специальности 30.05.03 «Медицинская кибернетика» / Юго-Зап. гос. ун-т; сост.: О.В. Шаталова. - Курск, 2023. - 125 с.

Предназначено для студентов специальности 30.05.03 «Медицинская кибернетика» по дисциплине «Введение в MATLAB». Может быть использована аспирантами, обучающимися по направлению подготовки 1.5.8. Математическая биология, биоинформатика.

Текст печатается в авторской редакции

Подписано в печать . Формат 60×84 1/16.
Усл. печ. л. 7,27. Уч.-изд. л. 6,58. Тираж 100 экз. Заказ 95 . Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Практическое занятие №1

Знакомство с пакетом MATLAB

1 Структура пакета и принципы работы

MATLAB – система многоцелевого назначения, которая вышла на рынок программных продуктов почти двадцать лет назад и с тех пор непрерывно совершенствовалась фирмой MathWorks. Но первоначально ее основу составляли алгоритмы решения систем линейных уравнений и задач на собственные значения, откуда и произошло ее название «матричная лаборатория» (MATrix LABoratory). Затем система была расширена за счет специальных приложений, таких как Simulink (для моделирования ИС), Wavelet (для применения вэйвлетов), Symbolic Math Toolbox (для проведения символьных вычислений) и пр.

Среда MATLAB включает интерпретатор команд на языке высокого уровня, графическую систему, пакеты расширений и реализована на языке C. Вся работа организуется через командное окно (Command Window), которое появляется при запуске программы matlab.exe. В процессе работы данные располагаются в памяти (Workspace), для изображения кривых, поверхностей и других графиков создаются графические окна. В командном окне в режиме диалога проводятся вычисления. Пользователь вводит команды или запускает на выполнение файлы с текстами на языке MATLAB. Интерпретатор обрабатывает введенное и выдает результаты: числовые и строковые данные, предупреждения и сообщения об ошибках. Строка ввода помечена знаком >>. В командном окне показываются вводимые с клавиатуры числа, переменные, а также результаты вычислений.

Инструментальная панель командного окна позволяет выполнять требуемые действия простым нажатием на соответствующую кнопку. Большинство кнопок имеют стандартный вид и выполняют стандартные, подобные другим программам действия. Следует обратить внимание на кнопку Path Browser, которая позволяет прокладывать пути к разным директориям и делать необходимую директорию текущей, а также

на кнопку Workspace Browser, позволяющую просматривать и редактировать переменные в рабочей области. Принято работать с включенными помимо консоли окнами Command History и Workspace; включить их можно, расставив галочки в раскрытом меню Desktop, а упорядочить – проведя докировку (кривая стрелка около стандартного крестика закрытия окна). Если вы нарушили порядок следования окон, то воспользуйтесь командой Desktop/Desktop Layout/Default.

Все значения переменных, вычисленные в течение текущего сеанса работы, сохраняются в специально зарезервированной области памяти компьютера, называемой рабочим пространством системы MATLAB (Workspace). После окончания сеанса работы с системой MATLAB все ранее вычисленные переменные теряются. Чтобы сохранить в файле на диске компьютера содержимое рабочего пространства системы MATLAB, нужно выполнить команду меню File / Save Workspace As. По умолчанию расширение имени файла mat, поэтому такие файлы принято называть MAT-файлами. Для загрузки в память компьютера ранее сохраненного на диске рабочего пространства нужно выполнить команду меню: File / Load Workspace. Обе операции можно реализовать в режиме командной строки в формате, например, save <имя_файла> (без расширения .mat).

Команда help <имя_функции> позволяет получить на экране справку по конкретной функции. Например, команда help eig позволяет получить оперативную справку по функции **eig** - функции вычисления собственных значений матрицы. С некоторыми возможностями системы можно познакомиться с помощью команды demo. В окне MATLAB помимо собственно команд MATLAB можно использовать системные команды DOS. Удобным свойством системы является возможность использовать клавиши-стрелки ↑↓ для доступа к стеку с ранее введенными командами. Командой clc можно стереть содержимое командного окна, однако это не затронет содержимого рабочего пространства. Когда исчезает необходимость в хранении ряда переменных в текущем сеансе работы, их можно стереть из памяти компьютера командой clear или clear(имя1, имя2, …). Первая команда удаляет из 6 памяти все переменные, а вторая – переменные с именами

имя1 и имя2. Командой who (или более сильной whos) можно вывести список всех переменных, входящих в данный момент в рабочее пространство системы. Для просмотра значения любой переменной из текущего рабочего пространства системы достаточно набрать ее имя и нажать клавишу Enter. Для целей программирования важно знать, что часть строки, следующая за знаком %, является комментарием.

Легче всего протокол сессии получить с помощью команды diary. Вызов команды diary <имя_файла> приведет к тому, что все появившееся далее на экране (кроме графики) будет записано в файл <имя_файла>. Несколько последовательно набранных команд могут быть сохранены в М-файле (т.е. *.m), который с точки зрения операционных систем представляет аналог BAT-файла, а сточки зрения программирования – зародыш будущей программы. Самый простой путь для этого – в окне Command History выделить нужные команды, исполнить команду Create M-file контекстного меню; при этом откроется редактор М-файлов.

Пример:

Для получения информации о магическом квадрате наберем в консоли lookfor magic, получим, подождав некоторое время (для прерывания поиска использовать DOS-овскую комбинацию Ctrl+C):

```
>> lookfor magic
MAGIC  Magic square.
TWEBMAGIC Example standalone test of webmagic function.
WEBMAGIC Magic squares into HTML table.
```

Затем получим справку о команде MAGIC:

```
>> help magic
 MAGIC  Magic square.
    MAGIC(N) is an N-by-N matrix constructed from the integers
```

```
        1 through N^2 with equal row, column, and
diagonal sums.
        Produces valid magic squares for all N >
0 except N = 2.

        Reference page in Help browser
           doc magic
```

Щелкнем по ссылке, открыв окно помощи. Построим уже реальный магический квадрат, набрав в консоли для N=10:

```
>> XMag=magic(10)
XMag =

        92     99      1      8     15     67     74
51     58     40
        98     80      7     14     16     73     55
57     64     41
         4     81     88     20     22     54     56
63     70     47
        85     87     19     21      3     60     62
69     71     28
        86     93     25      2      9     61     68
75     52     34
        17     24     76     83     90     42     49
26     33     65
        23      5     82     89     91     48     30
32     39     66
        79      6     13     95     97     29     31
38     45     72
        10     12     94     96     78     35     37
44     46     53
        11     18    100     77     84     36     43
50     27     59
```

Подсчитаем, сколько занимает в памяти переменная XMag:

```
>> whos XMag
```

```
  Name    Size     Bytes  Class
  XMag   0x10      800  double array
Grand total is 100 elements using 800 bytes
```

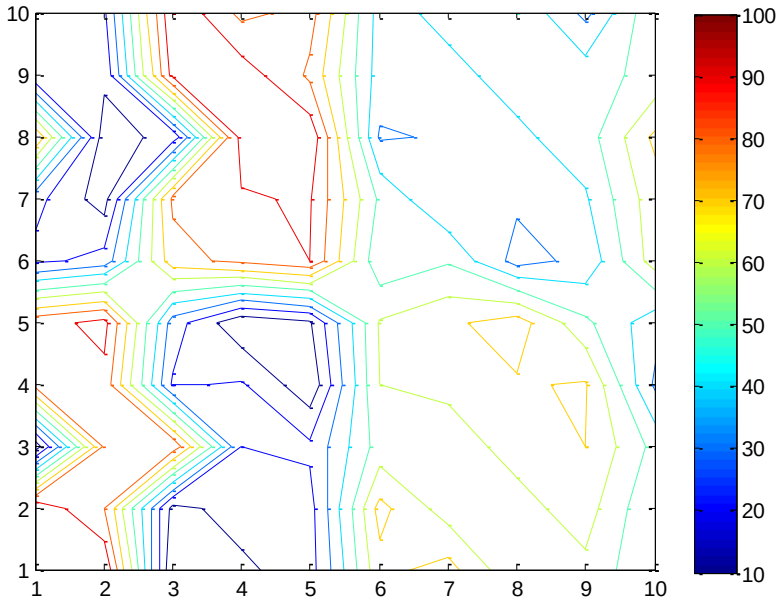

Рисунок 1 – Расшифровка цветов

Выберем закладку Workspace («рабочее пространство»), в контекстном меню исполним команду countour. При этом появится специальное окно, где MATLAB позволяет редактировать графические объекты – в частности, добавим расшифровку цветов (пиктограмма Insert ColorBar) (рисунок 1). Обратим внимание, что наши манипуляции не остались незамеченными средой, и в консоли появилась команда:

```
>> contour (XMag, 'DisplayName', 'XMag', 'ZDataSource', 'XMag'); figure(gcf)
```

Очистим рабочую область командой clear, затем консоль – clc. При этом история команд по-прежнему доступна; выделим шапку со списком сегодняшней даты и запишем в М-файл, вызвав контекстное (с выделенным) меню.

```
lookfor MAGIC
help magic
XMag=magic(10)
```

```
whos XMag
contour (XMag, 'DisplayName', 'XMag',
'ZDataSource', 'XMag'); figure(gcf)
clear
clc
```

Задание:

1. Запустить MATLAB и установите текущий каталог. Получить справку по команде diary. Начать запись дневника в файл.
2. В справочной системе, начав с консоли, получить информацию о гиперболическом котангенсе, а также переводу угла из радианной меры в градусную (и наоборот).
3. Вычислить $\sin(3.3a+b\cdot\mathrm{cth}(a+b))\sin$ для $a=15^0$, $b=23^0$ (предварительно проведя расчет для нулевых a и b).
4. В справочной системе найти информацию о том, какую функцию следует вызывать для вычисления ближайшего к X целого. Предварительно следует показать преподавателю таблицу простейших функций вещественного переменного (перенеся ее в MS Word).

Примечание: необходимо иметь навык работы с HELP-системами (уходя от консоли, вызов, как обычно, по клавише F1 или командой Help – MATLAB Help главного меню). Существует две стратегии поиска: по ключевым словам (индексу) или по дереву содержания (закладка Content окна помощи). Вторая стратегия предпочтительнее, но ее использование требует некоторого эвристического опыта. Разумеется, знание английского языка более чем необходимо.

5. Сформировать матрицу из случайных элементов размера 10*10. Используя пользовательские средства Workspace, построить ее 3D-график, сохранить картинку средствами MATLAB.
6. Записать текущую сессию в mat-файл. Сохранить историю команд в m-файл.
7. Очистить экран и переменные. Закрыть MATLAB.
8. Показать преподавателю все сохраненные файлы. Открыть MATLAB и с консоли загрузить последний mat-файл.
9. Завершить работу в MATLAB окончательно.

Практическое занятие №2

Проведение вычислений без М-файлов

1. Элементарные матричные вычисления

В MATLAB все данные рассматриваются как матрицы. Этот принцип ***векторизации*** имеет как положительные, так и отрицательные стороны. Тип результата определяется автоматически по виду выражения. Для имен обычных переменных принято выбирать строчные, а для имен матриц или даже векторов – прописные (большие). Вычислять можно в режиме калькулятора, но чаще используют присвоение (обычный знак «=», для проверки условий зарезервировано двойное равенство «==») какой-либо переменной. Любое уже определенное значение можно вызвать из рабочей области по имени переменной (либо в консоли, либо контекстным меню в окне Workspace). Помните, что можно провести несколько последовательных вычислений, подавив вывод в консоль промежуточных результатов – для этого после каждой команды нужно поставить «;».

Для чтения/записи данных из файлов (прежде всего ASCII-формата) MATLAB обладает внушительным арсеналом средств (см. раздел помощи MATLAB Programming – Using Import Functions with Text Data). В случае, когда импортируемый файл является перечислением чисел в регулярных колонках и строках (без различных шапок и легенд), достаточно применить команду load. Среди дружественных MATLAB внешних программ прежде всего следует отметить Excel (см. раздел помощи Excel Link – What Is Excel Link?), который со своей стороны имеет средства общения с MATLAB.

При ручном вводе, если компоненты вектора не умещаются на строке, используется троеточие в качестве разделителя. Иногда требуется организовать доступ к блоку внутри имеющейся матрицы, что реализуется через знак «:» – конструкция A(2:5,:) дает нам строки со 2-й по 5-ю матрицы A. Если нужно выполнить одинаковую операцию с каждым элементом матрицы, то нужно перед знаком поставить точку – V.*V, например, возвращает вектор

квадратов, т.е (-1,5) → (2,25). Для создания многомерных матриц использовать функцию cat.

Пример:

Введем скаляр, трехмерные вектор-строку и вектор-столбец, рассматриваемые как матрицы размера 1×n и n×1 (n=3). Сделаем это чуть различающимися способами (разделителем в массиве может либо пробел, либо запятая, либо оба знака вместе).

```
>> x=-1.25e-1

x =

   -0.1250

>> u=[1,2.0 4]

u =

     1     2     4

>> v=[-x; -2.0;7]

v =

    0.1250
   -2.0000
    7.0000
```

Приготовим заранее в рабочем каталоге файл dm3.txt, состоящий из 3 строк по 3 числа, разделенных пробелом (ами) или знаком табуляции. Загрузим его данные в матрицу W.

```
>> Wfrom='dm3.txt';W=load(Wfrom);W

W =
```

```
    1.2000    4.3000    5.0000
  -34.0000    2.1000    4.0000
    3.0000    3.0000   -6.0000
```

Заметим, мы ввели литерную переменную Wfrom, подавили вывод в консоль, и при этом исходный файл был таким:

```
1.2 4.3 5.0
-3.4e+1  2.1 4
3   3   -6
```

Попробуем сгенерировать новую матрицу 10×3, где столбцы являются арифметическими прогрессиями. Вначале создадим матрицу 3×10, а затем ее транспонируем.

```
>>  W1=[-1.4:0.2:0.4;  linspace(-2,5,10);  -1.4:7.7]

W1 =

   -1.4000   -1.2000   -1.0000   -0.8000   -0.6000    -0.4000    -0.2000         0    0.2000    0.4000
   -2.0000   -1.2222   -0.4444    0.3333    1.1111    1.8889    2.6667    3.4444    4.2222    5.0000
   -1.4000   -0.4000    0.6000    1.6000    2.6000    3.6000    4.6000    5.6000    6.6000    7.6000

>> W1=W1'

W1 =

   -1.4000   -2.0000   -1.4000
   -1.2000   -1.2222   -0.4000
   -1.0000   -0.4444    0.6000
   -0.8000    0.3333    1.6000
```

```
   -0.6000    1.1111    2.6000
   -0.4000    1.8889    3.6000
   -0.2000    2.6667    4.6000
         0    3.4444    5.6000
    0.2000    4.2222    6.6000
    0.4000    5.0000    7.6000
```

Заметим, что второй столбец W1 получился в результате применения вызова функции от трех аргументов linspace, а первый – иллюстрирует типичное применения знака «:». Третий столбец иллюстрирует особенности последней операции. Интерпретатор MATLAB выдал бы ошибку, если бы в первой команде мы заменили 0.4 на 0.6 или в последней 7.7 на 9.9. Причина заключается в несоответствии размеров столбцов, ведь второй декларирован linspace как имеющий 10 элементов.

Получим новую матрицу размера 3×3 таким способом: домножим транспонированного двойника W1 на W1 и переставим угловой элемент с центральными местами.

```
>> W2=W1'*W1; Wtemp=W2(1,1); W2(1,1)=W2(end-
1,end-1); W2(end-1,end-1)=Wtemp; clear Wtemp; W2

W2 =

   72.4074    5.3333    1.0000
    5.3333    5.8000  110.6667
    1.0000  110.6667  178.6000
```

Появившаяся выше переменная Wtemp была «мавром, сделавшим свое дело».

Сформируем новую матрицу на основе всех предыдущих – как линейную комбинацию W, W2 и случайной матрицы. Ниже в указанных форматах вызова: функция eigs ищет наименьшее по величине собственное значение матрицы, diag(rand(…)) – формирует почти нулевую матрицу с ненулевой случайной наддиагональю, det и inv – вычисляют определитель матрицы и обратную ей. Очевидно также, что (u*v) – скаляр, (v*u) – матрица

3×3.

```
    >> W3=x*inv(W)+W2/eigs(W2,1,'sm')-
((u*v)+det(v*u))*diag(rand(1,2),1)
    Iteration 1: a few Ritz values of the 3-by-3
matrix:
         0

    W3 =

       -1.4996   -4.9030   -0.0201
       -0.1275   -0.1219   -2.6727
       -0.0304   -2.2877   -3.6801
```

Запишем полученный результат в файл EXCEL в определенный диапазон ячеек на первом листе. Флаг W3status равен 1, если все прошло успешно.

```
    >>
W3status=xlswrite('dm3_res.xls',W3,'B3:H10')

    W3status =

         1
```

Задание

1. Ввести две скалярные величины x=-0.85, y=978.56*10^{-2}.
2. Заготовить XLS-файл с двумя матрицами A и B размера 3×5 и 5×3.
3. Считать их в одноименные структуры MATLAB.

Примечание: Необходимо воспользоваться одним из подразделов помощи MATLAB Programming

4. Изменить центральные элементы обеих матриц. Для A – домножить его на x и вычесть 1, для B – домножить его на siny и прибавить 1.
5. Осуществить поэлементное умножение матрицы A на транспонированную матрицу B.
6. Вычислить произведение AB и прибавить к нему матрицу C, где C – трехдиагональная: на главной диагонали

единицы, наддиагональ – из x-ов, поддиагональ – из y-ков.

7. Результат вывести в виде текстового файла.

8. Вычислить след (trace) матрицы, обратной к последней вычисленной матрице.

9. Сохранить последовательность команд в файле matrix1.m

2 Элементарные функциональные вычисления и построение графиков

Комплексные числа вводятся в виде Re+Imi или Re+Im*j (предпочтительнее вариант без умножения, поскольку может возникнуть путаница с перекрытием имен). Попробуйте набрать >> j=5; [1+j,1+1j] и почувствуйте разницу! Однако в случае, когда коэффициентом перед мнимой единицей является не число, а переменная, между ними следует обязательно использовать знак умножения. Полезно знать три константы MATLAB: pi – число π, inf – бесконечность, NaN – неопределенное значение. Они могут выступать аргументами функций: >> [sin(inf), sin(1/inf), 3*NaN]. В системной переменной ans хранится результат вычислений, а в eps – номинальная точность вычислений, что удобно с точки зрения численных методов, например, решения уравнений способом дихотомии. Значения этих переменных можно изменять: по умолчанию exp = 2^(–52), но очень часто делают присвоение exp = 1e – 6; удобно проводить цепи вычислений (пример: >> ans+1). В MATLAB широкий набор самых разнообразных встроенных математических функций. Вы можете задавать и собственные функции (доступные, однако, только в текущем сеансе). Они называются ***анонимные***, их значение вычисляется только одним оператором, а формат задания и вызова таков:

```
<имя>=@(<arg1>,…) <выражение> …   <имя>
                                (<arg1>,…)
```

При этом инициализируется переменная-указатель на функцию, и как раз по указателю функция и вызывается (полезно посмотреть Workspace). При написании собственных функций весьма желательно предусмотреть матричные операнды, что сильно

облегчает жизнь при построении графиков (так, для вычисления $f(x) = x^3$ конструкция x^3 хуже правильного варианта x.^3 с точкой поэлементной операции). Для реализации более сложных и серьезных функций необходимо обращаться к программированию М-файлов. Приведем лишь некоторые из часто используемых:

log(x) – натуральный логарифм

mod(x,y) – остаток от деления x на y (применима и для нецелых аргументов)

angle(z) – угол поворота комплексного числа z

polyval(p,t) – вычисление полинома от аргумента t с заданными вектором p коэффициентами.

bessel(nu,z) – вычисляет функцию Бесселя, причем если аргумент z матричный, то и результат также матричный. Константа nu нецелая. Пользователь может не только использовать различный синтаксис вызова (например, количество аргументов), но и определять вариант получения результата – в этом состоит важная и удобная особенность функций в MATLAB.

quad1(fun,a,b) – вычисление определенного интеграла Римана по адаптивному методу Гаусса-Лобатто на отрезке [a,b] от функции f(x), заданной указателем fun. Если функция анонимная, то вместо fun подставляется просто <имя>, но если функция задана в М-файле, то нужно провести разыменование и подставлять <@имя>.

Теперь рассмотрим функции вспомогательные для расчета, но весьма полезные:

str2num – переводит строку в число

tic <операции> toc – сколько времени (в сек) потребовалось для совершения операций

clock – возвращает текущую дату и время

Важное значение имеет графическое представление результатов, и MATLAB обладает обширным набором средств (см. раздел помощи в закладке Contents MATLAB – Mathematics - Plotting Mathematical Functions, более подробно изложено в MATLAB – Functions – By Category –Graphics–Specialized Plotting). Каждый раз при получении графика создается графический объект, который автоматически открывается в окне графического редактора (можно открыть несколько объектов – см. команда h=figure(·) и смежные; об иерархии графических объектов (рисунок

2) см. MATLAB – Handle Graphics Object Properties). Там вы можете манипулировать подписями осей, легендами, цветами линий, сохранять изображение в разных форматах.

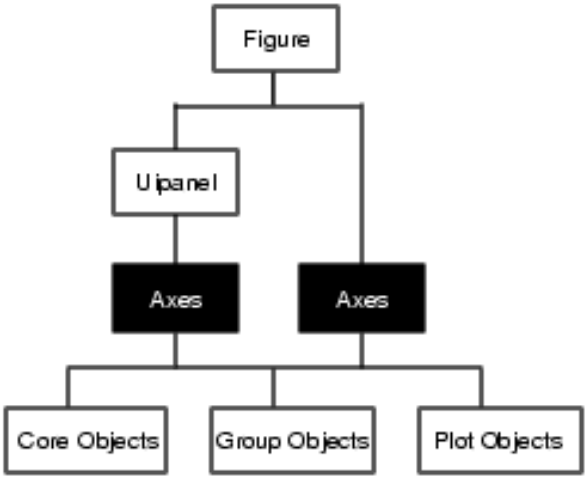

Рисунок 2 – Иерархия графических объектов

Существует сложная иерархия графических объектов, и каждому действию в окне графического редактора соответствует команда, модифицирующая заданный посредством указателя объект. Поэтому не следует забывать, что управлять параметрами изображения можно из консоли (разумеется, большинство инструкций приходится брать из системы помощи, например, set) – этот способ вряд ли можно считать прогрессивным, но в некоторых случаях (поскольку доступные операции в меню окна графического редактора не исчерпывают всего многообразия) незаменим. Нужно помнить, что такие операции, как выбор цветовой гаммы, применяются к тому объекту, который объявлен текущим.

Не меньшее значение имеют пользовательские данные, представленные, как правило, матрицами большого размера (например, 100×300). Часть данных, таких как значения функций, может вычисляться «на лету», но надежнее вычислить все необходимое до построения графика. Всякий график предполагает наличие сетки, прямоугольной или треугольной (если ранее применялись численные методы, то в роли такой сетки может выступать сетка для исчисления конечных разностей, которая изначально и определяет детальность представления результатов; в общем случае обе сетки не совпадают, и тогда дополнительно

следует применять интерполяцию – см. griddata). Самый простой путь для получения сеток для 3D-поверхностей – использование функции meshgrid.

Наиболее общеупотребительны (для целей физико-математических расчетов и/или проектирования РЭА) следующие виды графиков (рядом указано имя специальной функции):

A. Простой график вида $y = f(x)$ с маркерами и без– $plot(x, y, ...)$, $fplot(...)$. Последняя команда удобна тем, что кривая строится по указателю функции, или даже по текстовому описанию $fplot('sin(x).*cos(2x)', [0,5*pi])$.

B. Графики, построенные в преобразованных координатах (например, двойных логарифмических, т.е. данные имеют вид $(x, f(x))$, а координаты – $(\ln x, \ln f(x)))$. В том числе в полярных координатах – $cart2pol$, $loglog(x, y)$, $polar(fi, r)$.

C. Параметрическое семейство кривых $y = f(x, p)$, p – параметр, значение которого указывается в легенде для каждой кривой $y_p = f(x)$. Все кривые имеют разный цвет – hold on, plot, параметр LineSpec функции plot.

D. Контурный график, изображающий линии уровня (изолинии) $z = const$ для поверхности $z = f(x, y)$. Промежутки, отделяющие изолинии, могут заполняться цветом, а могут и нет. Географическая карта, где отображаются океаны и горы, - классический пример контурного графика – contourc, contourf.

E. 3D-график для поверхности $z = f(x, y)$ или траектории, заданной параметрически r(t) – $colormap(hsv128))$, mesh, surf, plot3

F. Анимированные графики. Обычно для траекторий частиц, получили большое распространение в последнее время – comet, getframe…movie.

Пример:

1. Сформировать случайные векторы u и v размера 1×4.

```
>> n=4;u=rand(1,n);v=rand(1,n);[u;v]

ans =
```

```
    0.4740    0.9090    0.5962    0.3290
    0.4782    0.5972    0.1614    0.8295
```

2. На основе этих векторов составить матрицу $A = \|a_{ij}\|$, $a_{ij} = u_i v_j^{j-1}$.

```
>> % Выполним все без циклов. Начнем с формирования вектора из единиц

>> w=[linspace(1,1,n)];w=w';

>> % Сделаем первую заготовку

>> A1=(w*u)'

A1 =

    0.4740    0.4740    0.4740    0.4740
    0.9090    0.9090    0.9090    0.9090
    0.5962    0.5962    0.5962    0.5962
    0.3290    0.3290    0.3290    0.3290
>> % Составим вторую заготовку, содержащие степени

>> q=[0:1:n-1]; A2=w*q
A2 =

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

>>  % Осуществим поэлементную операцию возведения в степень

>> A3=w*v;A3=[A3.^A2]
```

```
A3 =

    1.0000    0.5972    0.0261    0.5707
    1.0000    0.5972    0.0261    0.5707
    1.0000    0.5972    0.0261    0.5707
    1.0000    0.5972    0.0261    0.5707

>> % Поэлементным умножением матриц приходим
к искомому

>> A=A1.*A3

A =
    0.4740    0.2831    0.0124    0.2705
    0.9090    0.5428    0.0237    0.5188
    0.5962    0.3561    0.0155    0.3403
    0.3290    0.1964    0.0086    0.1877

>> % Стираем ненужное

>>clear ('A1','A2','A3','w')
```

3. Для каждого элемента матрицы A вычислить значение полинома с коэффициентами $(u+iv)$. Результаты записать в матрицу B. Превратить ее в вещественную, взяв от каждого элемента его угол поворота. Сколько чистого времени потребовалось на выполнение п.3?

```
>> tic; B=polyval(u+i*v,A);B=angle(B);toc
Elapsed time is 0.003572 seconds.
```

4. Составить анонимную функцию для вычисления площади треугольника по формуле Герона ($S=\sqrt{p(p-a)(p-b)(p-c)}$, p- полупериметр, a,b,c – стороны). Вычислить площадь для египетского треугольника.

```
>> % Возможен только один оператор. Площадь
```

полагаем нулю, если из заданных отрезков нельзя составить треугольник

```
    >>                                  geron=@(a,b,c)
real(0.25*sqrt((a+b+c)*(a+b-c)*(a+c-b)*(b+c-
a)));
    >> geron(3,4,5)

    ans =
         6
```

5. Построить график зависимости S(a) на отрезке [0, πe], приняв b=e, c=π. Оформить подписи, цвет кривой сделать фиолетовым.

```
    >> % Есть два пути решения. 1-й состоит в
переписывании анонимной функции.
    >>% Команда figure избыточна. Hfig –
указатель на текущий графический объект

    >>                                  geron1=@(a,b,c)
real(0.25*sqrt((a+b+c).*(a+b-c).*(a+c-b).*(b+c-
a)));
    >>
x=0:0.1:(pi*exp(1));Hfig=figure(1);plot(x,geron1
(x,exp(1),pi));

>>% Второй путь, более общий, заставляет
обратиться к программированию. Некоторые
конструкции типа for…end доступны и в консоли.
>> % Ниже формируется вектор y, длина которого
постоянно увеличивается. Затем непосредственно
вывод графика (рисунок 3).

>>    x=0:0.1:(pi*exp(1));y=[    ];    for    k=x
y=[y,geron(k,exp(1),pi)]; end;
    >> Hfig_=figure(2);plot(x,y);
```

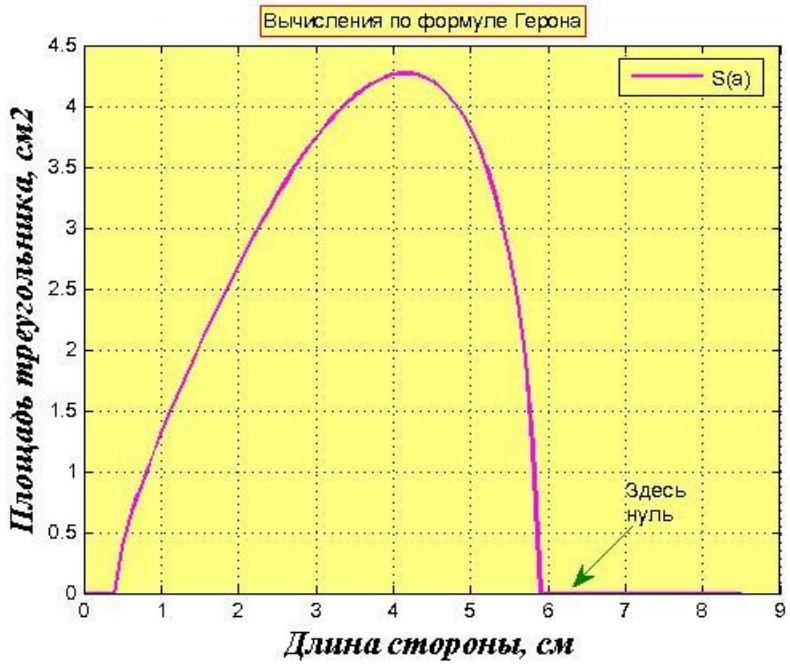

Рисунок 3 – График по результатам вычислений

В окне Figure2 щелкнем мышью по иконке со стрелкой, перейдя к режим редактирования. Вызовем контекстное меню, связанное с кривой – пункт Color, и выберем цвет. Аналогичным способом выберем цвет фона. Сделаем и другие манипуляции с осями, подписями и пр. Полезен пункт главного меню Insert, пункт контекстного меню – Show Properties. Удобно также подключить редактор свойств (View – Property Editor). Исполнив File– Generate M-file, можно просмотреть совершенные нами действия в виде команд.

6. Построить семейство кривых S(a) при пяти значениях b=0,1,2,3,4 и фиксированном c=5. Величина a лежит в пределах [0;8]. Точки маркировать.

```
>> % Закрыли 2-й график, открыли новый с указанием рисовать все там

>> close(Hfig_);figure(3);hold on;

>> % Инициализируем переменные
>> a=0:0.5:8;b=0:1:4;c=5;
```

```
>>% Команды занимают две строки. Организуем цикл, формируем массив указателей на линии, цвет линии в формате RGB (смесь красного, зеленого и синего) случайный

>> Hlines=[ ];for j=1:5, Hline=plot(a,geron1(a,b(j),c)); Hlines=[Hlines Hline]; ...
   set(Hline,'Marker','+','Color',[rand rand rand]); end; hold off

>> %Изменим цвета линий на привычные (рисунок 4) (см. ColorSpec) и добавим подписи к линиям. Вот где понадобились указатели! Названия получаем преобразованием чисел в литералы

>> MyColor=['y' 'm' 'c' 'r' 'g' 'b' 'w' 'k'];
>> for j=1:5, Lname=strcat(num2str(j),': b= ',num2str(b(j)));
set(Hlines(j),'Color',MyColor(j),'DisplayName', Lname); end; legend('show');
```

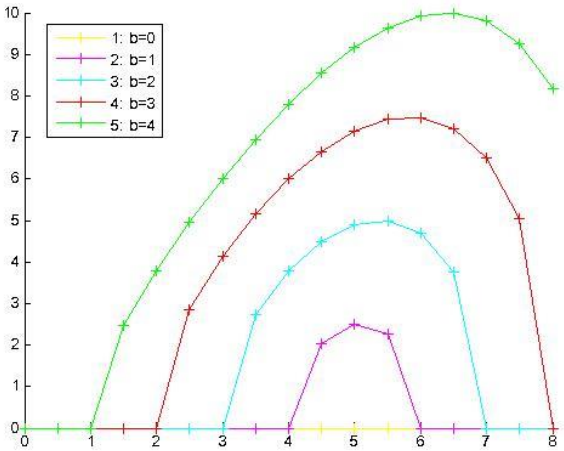

Рисунок 4 – Семейство кривых

7. Построить контурный график на основе базиса (u,v) и матрицы значений B, которая содержит высоты «гор». Площадь построения – [-2;2]× [-2;2].

```
>>  % Вначале создадим функцию растягивания
координат, поскольку 0<u_i & v_i<1
>>stretch=@(x) 4/(max(max(x))-min(min(x)))*x-
2*(max(max(x))+min(min(x)))/(max(max(x))-
min(min(x)));

>> % Построим первоначальную сетку (рисунок 5).

>> [Xbig,Ybig]=meshgrid(u,v)

Xbig =

    0.4740    0.9090    0.5962    0.3290
    0.4740    0.9090    0.5962    0.3290
    0.4740    0.9090    0.5962    0.3290
    0.4740    0.9090    0.5962    0.3290
```

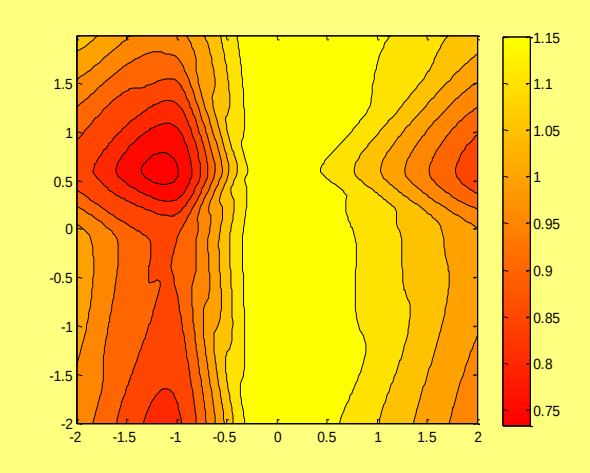

Рисунок 5 – первоначальная сетка

```
Ybig =

    0.4782    0.4782    0.4782    0.4782
    0.5972    0.5972    0.5972    0.5972
    0.1614    0.1614    0.1614    0.1614
    0.8295    0.8295    0.8295    0.8295
```

```
    >>  % Построим подробную сетку в области,
заданной векторами u,v. Эта область обязана не
```

```
превосходить существенно предыдущую область
[Xbig Ybig]
    >>
[X,Y]=meshgrid(linspace(min(u),max(u),100),linsp
ace(min(v),max(v),100));
    >> % Проведем интерполяцию с использование
кубических полиномов
     Z=griddata(Xbig,Ybig,B,X,Y,'cubic');

    >> % Построим контурный график с заливкой для
квадратной площадки

    contourf(stretch(X),stretch(Y),Z)
```

Для того, чтобы скопировать рисунок в буфер обмена, исполнить пункт Edit – Copy Figure меню графического окна, а если хотим скопировать его вместе с фоном, то поставить галочку File – Preferences – File Copy Template – Copy Options – Use Figure Color. Чтобы добавить расшифровку цветов, исполнить Insert – Colorbar. Можно цветовую гамму выбрать монотонной; для этого имеется пункт Edit – Colormap…, а в появившемся окне можно выбрать осеннюю гамму цветов (Tools – Standart colormaps – autumn) – (см. рисунок 6) Сверху-справа. Для того, чтобы обрезать у рисунка неиспользуемые поля – поставить галочку File – Export Setup – (Properties=Size) – Expand Axes….

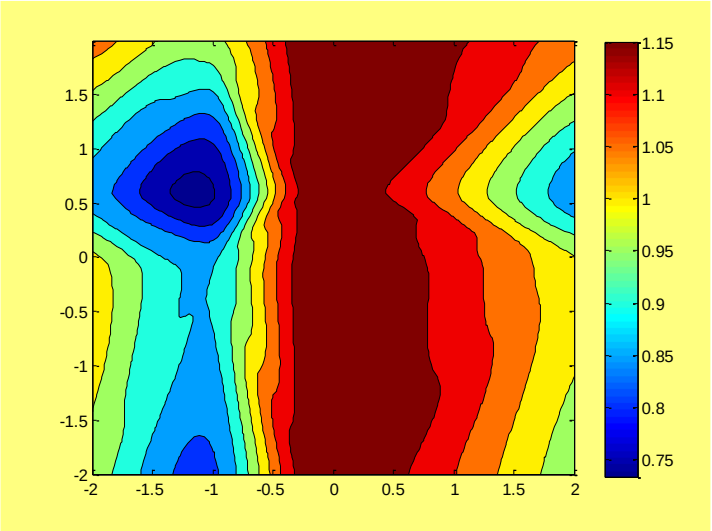

Рисунок 6 – Выбранная гамма цветов

8. Построить поверхность площадей S(a,b,c), вычисленную для изопериметрических треугольников (т.е. a+b+c=1).

```
>> % Подготовим сетки и значения, цвет выберем потом

>> a=0:0.01:1;b=0:0.01:1; [X,Y]=meshgrid(a,b);
Z=geron1(X,Y,1-X-Y);

>> % Инициализируем графический объект и построим в первом приближении (рисунок 7). Функция surf позволяет указывать вектора вместо матриц, т.е. как бы функция meshgrid выполняется автоматически

>> figure(5);Hsur=surf(a,b,Z)

>> % Обычно цветом отмечается высота точки. Введем свои цвета (в палитре RGB) и немного изменим угол зрения (рисунок 8). Также уберем линии сетки поверхности.

>>    C=cat(3,X,Y,1-X-Y);    set(Hsur,'CData',C,
'LineStyle','none'); view([23.5 36]);

>>   xlabel('Сторона   A','Color',[1   0   0]);
ylabel('Сторона  B','Color',[1  0  0]);  zlabel
('Величина площади', 'Color',[1 0 0],'Rotation',
90);

>>title('Площадь','BackgroundColor',[0.9412
0.9412 0.9412],..
'EdgeColor',[1 0 0], 'FontSize',14,
'FontWeight','bold','LineWidth',1);

>>axis([0 1 0 1 0 0.5]); box('on');
```

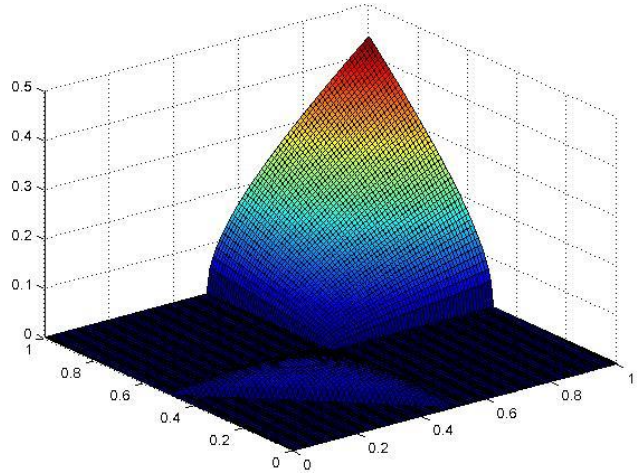

Рисунок 7 – Графический объект в первом приближении

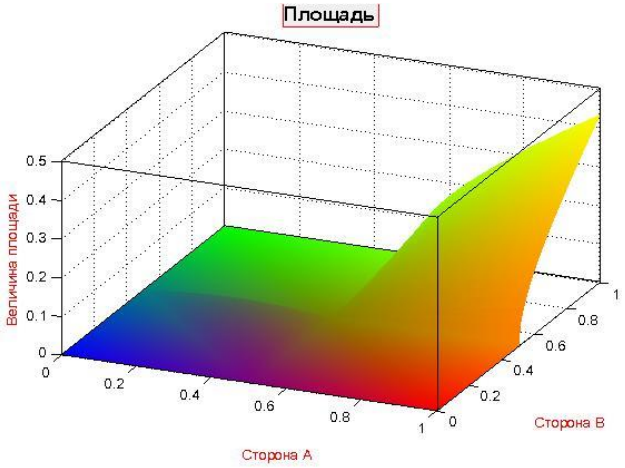

Рисунок 8 – Графический объект с изменённым углом зрения

Примечание: Укажите ошибку вычислений при выполнения п.8. Как ее исправить? См. п.10.

9. Построить в декартовых координатах кривую, заданную параметрически на $[-\pi;\pi]$ (рисунок 9):

$$\vec{r}(t)=\begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}=\tilde{u}\sin t+\tilde{v}\cos t,$$ «тильда»– первые три компонента одноименных векторов.

```
>>      MyCurve1=@(t)      u(1)*sin(t)+v(1)*cos(t);
MyCurve2=@(t) u(2)*sin(t)+v(2)*cos(t);

>> % Обратите внимание, что мы здесь не делаем
матриц, используя облегченный вызов функции-
рисовалки – поэтому не можем применять
индексацию по осям

>> MyCurve3=@(t) u(3)*sin(t)+v(3)*cos(t);
ezplot3(MyCurve1,MyCurve2,MyCurve3,[-pi,pi])
```

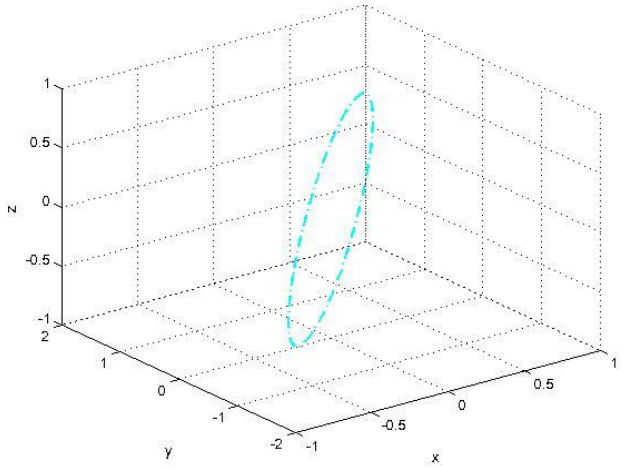

Рисунок 9 – Координатная кривая

10. Создать киноленту, отображающую эволюцию поверхности треугольников при изменении третьей стороны треугольника.

```
>> % Формируем сетку и область переменной времени, она же равна с

>> a=0:0.01:1;b=0:0.01:1;c=0:0.01:2;[X,Y]=meshgrid(a,b);

>> % Последовательно создаем графические объекты: рисунок, пространство осей, поверхность (пока пробную)

>> HFig1=figure(1); axes(); Hsur=surf(X,Y,ones(length(X))); set(HFig1, 'PaperType', 'A4', 'Color','y');

>> % Модифицируем параметры объектов. Убираем сетку, делаем цветовую гамму неподвижной (важен порядок вызова установки пределов третьей оси и цветовой легенды

>> colormap('jet'); set(gca,'Box','on'); caxis([0 0.5]); colorbar; set(Hsur,'LineStyle','none')
```

```
>> % Основной цикл – смена данных построения. На лету формируем массив из скриншотов-картинок

>> for j=1:length(c) set(Hsur,'Zdata',geron1(X,Y,c(j))); MyMovie(j) = getframe(HFig1); end;
>> % Проигрываем кино 5 раз подряд. Записываем avi-файл, его можно проиграть стандартными средствами Windows.

>> movie(MyMovie,5);
movie2avi(MyMovie,'mymovie1.avi','fps',4)
```

Задание:

1. Сформировать случайную комплекснозначную матрицу A размера 3×3, вычислить матричную экспоненту $B = e^A$. Решить систему линейных уравнений $Az = (1,1,1)^T$.
2. Сформировать, не используя циклов, матрицу вида $\begin{pmatrix} AB & B \\ A & [a_{ij} b_{ij}] \end{pmatrix}$ и выделить из нее вещественную (D1) и комплексную (D2) часть
3. Вычислить с помощью анонимной функции электрический потенциал системы точечных электрических зарядов, заданных координатами (D1,D2), в точке с координатами (0,0). Заряды положительные, по модулю равны 1.
4. Построить контурный график потенциала на площади $[-2;2]\times[-2;2]$. Выполнить цветовую легенду в логарифмическом масштабе. В графическом редакторе оформить график красиво.
5. Построить в полярных координатах 2D-график напряженностей поля (velocity plot).
6. Построить 3D-график п.4.
7. Создать анимацию контурного графика в п.4. при движении координат зарядов по закону:

$D1_{ij} = D1_{ij}(1 + a\cos(j + it))$ $D2_{ij} = D2_{ij}(1 + a\sin(i + jt))$, $0<t<2\pi$,

a=0.2

Примечание: поскольку графика требует времени для своего построения, то использовать команду pause для задания задержек. Более предпочтительный путь – команда drawnow до взятия фрейма.

Практическое занятие №3

Проведение вычислений с помощью М-файлов

1. Создание и отладка М-файлов

MATLAB является интерпретирующим языком непосредственных вычислений, т.е. выражения, которые вы вводите, интерпретируются и вычисляются. Поскольку все вычисления в MATLAB выполняются с двойной точностью, формат вывода может управляться с помощью следующих команд (рисунок 10).

format short	с фиксированной точкой и 4 знаками после точки (по умолчанию)
format long	с фиксированной точкой и 14 знаками после точки
format short e	научная нотация с 4 десятичными знаками
format long e	научная нотация с 15 десятичными знаками

Рисунок 10 – Команды управления форматом вывода

После вызова одного из приведенных выше форматов он сохраняется до вызова другого. Команда format compact подавляет большинство пустых строк, позволяя большее количество информации вывести на экран или страницу. Она не зависит от других команд формата. Команда eval чрезвычайна полезна, позволяя обмениваться между процедурами (особенно, внешними) строкой-параметром, которая принуждает систему к выполнению заданной этой строкой команды (или серии команд). Напомним, что MATLAB различает большие и маленькие буквы в именах команд, функций и переменных, а спецзнаки могут быть записаны через обратный слэш.

Для простых операций удобен интерактивный режим, но если вычисления нужно многократно выполнять или необходимо реализовывать сложные алгоритмы, то следует использовать m-файлы MATLAB (расширение файла состоит из одной буквы m). Познакомимся со script-m-файлами (или сценариями)– текстовыми

файлами, содержащими инструкции на языке MATLAB, подлежащими исполнению в автоматическом пакетном режиме. Создать такой файл удобнее с помощью редактора системы MATLAB (не забудем способ создания, исходя из окна истории команд). Он вызывается из командного окна системы MATLAB командой меню File/New/M-file (или самой левой кнопкой на полосе инструментов, на которой изображен чистый белый лист бумаги). Записанные в script-файлы команды будут выполнены, если в командной строке ввести имя script-файла (без расширения). Переменные, определяемые в командном окне и переменные, определяемые в сценариях, составляют единое рабочее пространство системы MATLAB, причем переменные, определяемые в сценариях, являются глобальными, их значения заместят значения таких же переменных, которые были использованы до вызова данного script-файла. Текст реально имеющихся m-файлов (системы MATLAB, например polar, или ваших собственных) можно просмотреть с помощью команды type <имя_функции>.

Однако, большинство m-файлов являются файлами-функциями, т.е. программами. При том же порядке вызова в отличие от простого линейного скрипта, хотя бы тот и содержал, например, циклические конструкции, нормальный m-файл содержит тело, окаймленное следующими декларациями:

```
    function [out1, out2, ...] = funname(in1,
in2, ...)
        <body>
    End
```

Тело может быть представлено в таком формате:

```
    <body>=

    %Строки основного комментария, доступного по
команде help
    % Далее три четыре основных повторяющихся
типа конструкций
```

```
    % Объявление собственных переменных и команды
MATLAB

    <переменная=выражение>; <команда>

    % Объявление подфункций (пишется обычно с
отступом)

        function [out]=nested_funname (in)
            <операторы;>
        end;
    % Конструкции языка типа for, if,….

    If <условие>
        <операторы;>
    end
```

В заголовке явном виде пишутся выходные аргументы, что несколько отличается, например, от Си-программ (там слово function равнозначно procedure). Сходные декларации имеют место для подфункций и пишутся внутри тела, которое по сути тождественно пакету команд. MATLAB рассматривает m-файл как одну «главную» (primary) функцию, причем ее имя обязано совпадать именем m-файла; внутри этой функции могут располагаться декларации локальных (вложенных, nested) функций, доступных только изнутри данного m-файла. Ниже главной функции в редких случаях могут располагаться подфункции, но доступ к ним возможен только из главной – поэтому их лучше все-таки реализовывать как вложенные. Внутри m-файлов можно ссылаться на другие m-файлы, в том числе и на самого себя рекурсивно. Тем самым реализуются, хотя и не полностью, принципы процедурного программирования. Переменные в функциях являются по умолчанию локальными, но в версиях 4.0 и выше разрешено объявлять требуемые переменные глобальными (global). Знак “;” получает двойное толкование: с одной стороны – это подавление вывода в консоль, с другой – привычный разделитель между операторами языка.

Как и в библиотечных функциях MATLAB, используется принцип избыточного задания аргументов. Это позволяет варьировать действия в зависимости от формы и количества аргументов (см. nargin, nargout и особенно параметр декларации функции varargin). Аналогично, доступ к результатам выполнения может быть частичным. При вызове функции для нее создается стек памяти, в который помещаются помимо локальных переменных копии одноименных переменных вызова. Для простого интерфейса, использующего только консоль, предусмотрены операторы disp, input, error. Как и в большинстве языков/сред программирования, MATLAB обладает стандартными операторами ветвления, выбора и цикла (кроме goto). Особенность for заключается в использовании массива/матрицы в качестве индекса, при этом счетчик последовательно пробегает значения, быть может вещественные, элементов/столбцов матрицы. Например, если вам необходимо выполнить <оператор> только для тех элементов матрицы, которые больше 3, то удобно это сделать следующим образом:

```
for i=find(A>3) <оператор> end;
```

Редактор/отладчик предоставляет как средства редактирования текста m-файла, так и средства пошаговой его отладки. Один из способов вызова редактора – вызов из командной строки MATLAB с помощью команды edit. Редактор, используемый в системе, имеет синтаксическую раскраску, т.е. слово или символ по мере ввода приобретают тот цвет, который соответствует их типу. С помощью пункта меню Tools-Fonts можно настроить такие важные параметры, как используемый шрифт. Это особенно важно для работы с русским текстом, поскольку не все шрифты правильно воспроизводят русский текст. Редактор имеет стандартный набор возможностей (запуск M-файла, расстановка точек останова – breakpoints), так и специальные: переключение в Cell Mode (режим ячеек), публикация html, пункт Evaluate Selection, который позволяет вычислять значение выделенного выражения и помещать результат в консоль, пункт Edit-Paste to Workspace. В режиме ячеек легко отлаживать и читать программу; с той же целью введена пиктограмма Show Functions, позволяющая перескакивать по заголовкам вложенных функций.

Более детальные сведения можно прочесть, нажав пункт меню редактора Help – Using the M-file Editor.

Справа от редактируемого файла находится линейка номеров строк, а слева (рядом с полосой прокрутки) – линейка ошибок. Если в строке найдена синтаксическая ошибка, то имеем красный маркер на линейке ошибок; если ошибка некритичная (warning), то цвет маркера желтый. Код может быть предварительно проверен (это чем-то напоминает этап компиляции) - с помощью пункта меню Tools – Check Code with M-Lint (вызывается специальное окно с перечнем найденных ошибок). Помимо синтаксических есть ошибки времени выполнения. Ошибки времени выполнения выявить более сложно, потому что локальная рабочая область m-функции оказывается потерянной, если ошибка приводит к возврату в рабочую область системы MATLAB. Чтобы определить причину такой ошибки, можно использовать один из следующих приемов:

- реализовать вывод результатов промежуточных вычислений на дисплей, удалив в соответствующих операторах точки с запятой, которые подавляют вывод на экран промежуточных результатов;
- добавить в m-файл команды keyboard, которые останавливают выполнение m-файла и разрешают проверить и изменить переменные рабочей области вызываемой m-функции. В этом режиме появляется специальное приглашение K». Возврат к выполнению функции реализуется командой return;
- закомментировать заголовок функции и выполнить m-файл как сценарий. Это позволяет проследить результаты промежуточных вычислений в рабочей области системы;
- использовать отладчик системы MATLAB (см. несколько пиктограмм на панели справа) – только в случае отсутствия входных аргументов.

Пример:

1. Построить и оформить графики трех функций $y(t) = t\ln t + \sqrt[3]{1+t^2\sin t}$, $z(t) = \frac{1+t^3}{1+t^2}$, w(t)=z(t)-y(t) на [0;3]. Из

серии набранных команд собрать М-файл-сценарий.

Создадим анонимные функции

```
    >>                                          funy=@(x)
(x.^2).*log(x+eps)+(1+x.^2).^(1/3);
    >>  funz=@(x)  (1+x.^3)./(1+x.^2);  funw=@(x)
funz(x)-funy(x);
```

Проверим правильность задания

```
    >>  X=[0  1  2];[funy(X);  funz(X);  funw(X)],
clear('X')
    ans =
        1.0000    1.2599    4.4826
        1.0000    1.0000    1.8000
             0   -0.2599   -2.6826
```

Строим графики простейшим путем

```
    >>  figure(1);hold  on;fplot(funy,[0  3],'-
y');fplot(funz,[0        3],'--r');fplot(funw,[0
3],':g');hold off
```

С помощью контекстного меню (Create…) в History Commands формируем М-файл текущей сессии (лишние команды можно затем стереть в редакторе) (рисунок 11)

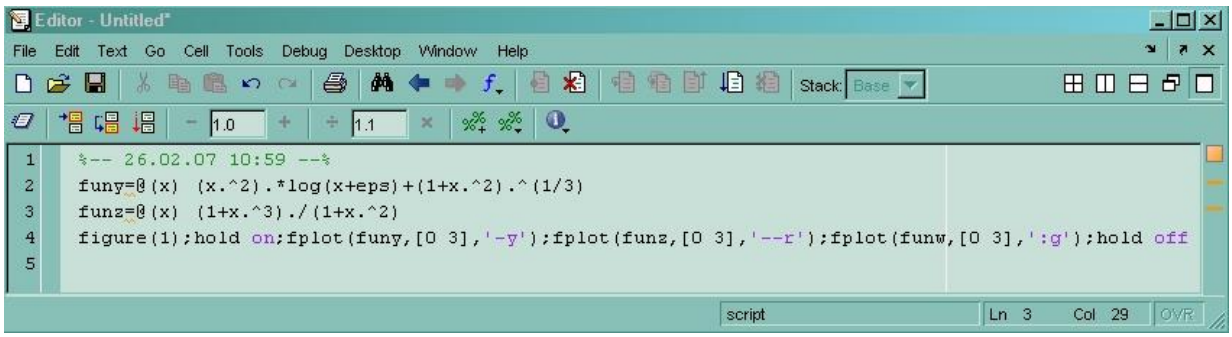

Рисунок 11 – Формирование файла текущей сессии

Проведем украшение графика (скажем, заменим желтый цвет линии на синий и добавим легенду) (рисунок 12). Исполним пункт File/Generate M-file. Откроется новый файл Untitled-2. Скопируем все секции оттуда (кроме первой, начиная со строки %Create Figure). Сохраним файл под именем MyGraph.m в текущий каталог.

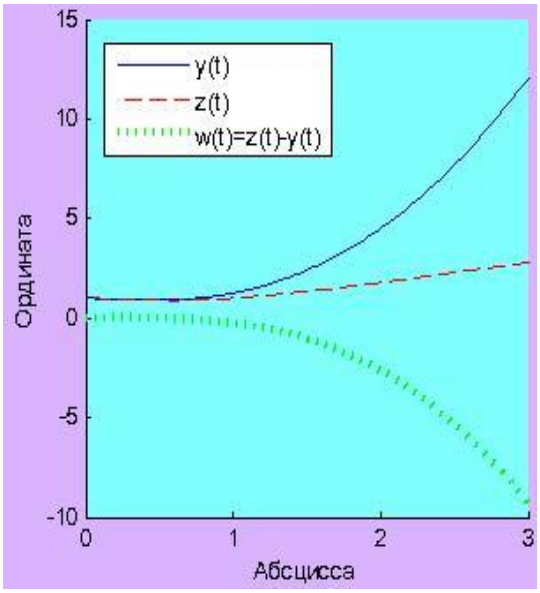

Рисунок 12 – Итоговый график

К сожалению, несмотря на полезные конструкции сгенерированного кода, он был исполнен в виде функции, поэтому придется провести ряд изменений, чтобы сделать его работоспособным. В первой секции заменим figure1 = figure() на set(gcf, - ведь полотно графиков уже создано. Во второй секции первая строка приобретет вид set(gca,'Color',[0.502 1 1]);. Также следует убрать третью ось в команде прорисовки осей axis. С секциями «% Create plot» сложнее, здесь придется операцию создания кривой заменять на поиск указанной кривой и установку ее свойств (см . findobj). Это позволит нам заменить все «пустые» ссылки (дескрипторы).

Общий вид кода и смасштабированного результата его выполнения (по команде MyGraph в консоли) см. на рисунке 13 снизу и слева.

```
%-- 26.02.07 10:59 --%
funy=@(x) (x.^2).*log(x+eps)+(1+x.^2).^(1/3);
funz=@(x) (1+x.^3)./(1+x.^2);
funw=@(x) funz(x)-funy(x);
figure(1);hold on;fplot(funy,[0 3],'-y');...
    fplot(funz,[0 3],'--r');fplot(funw,[0 3],':g');hold off

% Create figure
set(gcf,...
  'Color',[0.8549 0.702 1],...
  'InvertHardcopy','off',...
  'Name','Три графика',...
  'NumberTitle','off',...
  'PaperSize',[20.98 29.68]);

% Create axes
set(gca,'Color',[0.502 1 1]);
axis([0 3 -10 15]);
hold('all');

% Change plots
HLines = findobj(gca,'Type','line');
set(HLines(1),'Color',[0 0 1]);
set(HLines(2),'Color',[1 0 0],'LineStyle','--');
set(HLines(3),'Color',[0 1 0],'LineStyle',':','LineWidth',3);

% Create xlabel
xlabel('Абсцисса');

% Create ylabel
ylabel('Ордината');

% Create legend
legend1 = legend(gca,{'y(t)','z(t)','w(t)=z(t)-y(t)'},...
    'Position',[0.1896 0.7425 0.4257 0.1909]);

```

Рисунок 13 – Общий вид кода

2. Записать в виде М-функции вычисления по формуле Герона в файле geroncore. Вычислить площадь при наборах (1,2,1.5) и (1,2,4).

```
function S=geroncore (a,b,c)
    p=(a+b+c)/2;
    S=sqrt(p.*(p-a).*(p-b).*(p-c));S=real(S);
end
>> geroncore ([1 1],[2 2],[1.5 4])
ans =
    0.7262         0
```

3. Записать в виде М-функции MyTriangle решатель треугольника, т.е. по трем заданным его элементам находящий остальные – число аргументов и результатов переменное.

Приготовим шаблон М-файла, разобьем его на секции (см. иконку «%%+» на второй панели редактора) и сразу его запишем

```
function [MyS, varargout]=MyTriangle (varagrin)
% Эта функция "решает" треугольник. Стандартный
вызов предусматривает
% 4 входных и 4 выходных аргумента. Первые три
аргумента - длины трех
% сторон, четвертый - строка вида 'abA', где
большими буквами указаны
% углы, а малыми - стороны (угол A противолежит
стороне a). Пятый аргумент
% предусматривает дополнительный расчет
биссектрис, медиан или высот.
%%
% Основные внутренние переменные
a=1;b=1;c=1;A=60;B=A;C=A;MyS=0;MyLegend='abc';
%%
%Проверка данных

%%
%Запись данных

%%
%Вычисление результата

%%
%Вывод результатов
varargout(1)=varargin(1);

end
```

Будем наполнять содержание каждой секции. Начнем предварительную проверку данных. При желании формат данных

можно выверять тщательнее. Пока же проверяем число аргументов и формат легенды, которая пишется как массив символов в MyLegend. Для примера мы сохранили закомментированную «точку останова» keyboard. Особенность переменной varargin состоит в том, что она является массивом ячеек (cell array), типом промежуточным между классической записью и массивом. Поэтому для извлечения данных используется функция char. Проще, но менее привычно, было бы записать `varargin {4}` вместо `varargin(4)`.

```
%Проверка данных
switch nargin
  case {0,1,2}
    warning('Слишком мало аргументов'),
flag=false;
  case {6,7,8}
    warning('Слишком много аргументов'),
flag=false;
  otherwise
    flag=true;
    if (nargin~=3) MyLegend=char(varargin(4));
end;
    %keyboard;
end;
if (~flag) return; end;
MyLegend=MyLegend(1:3);flag=true;NAngles=0;
for s=MyLegend
    switch s
        case {'a','b','c'}
        case {'A','B', 'C'}
            NAngles=NAngles+1;
        otherwise
            flag=false;
    end;
    if (~flag) break; end
end;
if (NAngles==3) error('По трем углам нельзя
решить треугольник'),
    return; end;
```

```
if (~flag) error('Неправильный формат легенды
данных'), return; end;
```

Для записи данных используем простой цикл и полезную команду eval.

```
%Запись данных
for (j=1:3)
eval(strcat(MyLegend(j),'=',num2str(varargin{j})
,';')); end;
```

Для расчета нам нужно разделить секцию на две части, в первой – описания функций, а во второй – вызывающий их код. Заключительная команда вызывает функцию geroncore из одноименного M-файла. Мы также оставили точку останова keyboard; поучительно рассмотреть одинаково называющиеся переменные, имеющие разную область действия – см. во время второго останова селектор Workspace-Stack. В стеке функции b=5, а стеке программы b=4 – интерпретатор не путается в действии принципа «локальное имя закрывает глобальное». Теперь программу можно осмысленно запускать, временно раскомментаривая последнюю строчку – например, >> MyTriangle(3,4,5,'abc').

```
% Сначала список функций, математически все
сводится к 5 случаям
%%
% Теорема синусов. Два угла прилежат стороне.
% Находится сторона у первого угла
    function res=SIN_side (a,alfa,beta)
        if (alfa+beta>180) warning('Два тупых
угла невозможны');
            res=NaN;return;
        end;

res=a.*sin(deg2rad(beta))./sin(deg2rad(alfa+beta
));
    end
```

```
% Теорема косинусов. Находится сторона возле
угла
    function res=COS_side (b,c,gamma)
        a1=c.^2-(b.*sin(deg2rad(gamma)))^2;
        if (a1<0) warning('Такого треугольника
не существует');
            res=NaN;return;
        end;
        a2=b+sqrt(a1);a1=b-sqrt(a1);
        if (a1<0) warning('Найдено два таких
треугольника!'); end;
        res=a2;
    end
% Теорема косинусов. Находится сторона против
угла
    function res=COS_opposite (a,b,gamma)
        res=sqrt(a.^2+b.^2-
2*a.*b.*cos(deg2rad(gamma)));
    end
% Теорема косинусов. Находится угол (в градусах)
    function res=COS_angle (a,b,c)
        if ((a<eps)||(b<eps)||(a+b<c))
            warning('Не выполнено неравенство
треугольника');
            res=NaN;return;
        end;
        keyboard;
        res=acosd((a.^2+b.^2-c.^2)./(2*a.*b));
    end
%%
% Главный блок расчета
switch MyLegend
  case 'abc'
    C=COS_angle(a,b,c); if(C==NaN) return; end;
    B=COS_angle(a,c,b); if(B==NaN) return; end;
    A=COS_angle(c,b,a); if(A==NaN) return; end;
  case 'abC'
```

```
    c=COS_opposite(b,a,C);if(c==NaN) return;
end;
    B=COS_angle(a,c,b); if(B==NaN) return; end;
    A=180-B-C;
  case 'abA'
    c=COS_side(b,a,A);if(c==NaN) return; end;
    B=COS_angle(a,c,b); if(B==NaN) return; end;
    C=180-A-B;
  case 'aBC'
    b=SIN_side(a,C,B);if(b==NaN) return; end;
    c=COS_opposite(a,b,C);if(c==NaN) return;
end;
    A=180-B-C
  case 'aAB'
    C=180-A-B; if(C<0) return; end;
    b=SIN_side(a,C,B);if(b==NaN) return; end;
    c=COS_opposite(a,b,C);if(c==NaN) return;
end;
  otherwise
    disp('Будет улучшено в след. версиях')
end

MyS=geroncore(a,b,c);
%[a b c A B C]
```

Проведем дополнительные вычисления медиан, высот и т.д., в соответствии в 5-м параметром вызова. Обратите внимание, что подфункция может вызывать ранее определенную подфункцию; также – несмотря на краткость определений функций, каждая должна занимать три строки.

```
%%
% Дополнительные расчеты по пятому параметру:
% med - медианы, bis - биссектрисы, hhh - высоты
MyS=geroncore(a,b,c);

% Объявление новых функций
    function res=median(a,b,gamma)
```

```
        res=COS_opposite(a,0.5*b,gamma);
    end
    function res=bisectrix(a,b,gamma)
        res=2*MyS./((a+b).*sind(gamma/2));
    end
    function res=height(c)
       res=2*MyS./c;
    end
% Расчет линий
for i=1:3, varargout{i}=i; end;
if (nargin==5)
  switch varargin{5}
      case 'med'

varargout{4}=median(b,a,C);varargout{5}=median(a
,b,C);
          varargout{6}=median(b,c,A);
      case 'bis'

varargout{4}=bisectrix(c,b,A);varargout{5}=bisec
trix(c,a,B);
          varargout{6}=bisectrix(a,b,C);
      case 'hhh'

varargout{4}=height(a);varargout{5}=height(b);
          varargout{6}=height(c);
      otherwise
            ;
  end
end;
```

Последняя секция короткая. Список аргументов varargout, как и varargin, - это массив ячеек, поэтому присвоения стоят в {}, а не в привычных (). При выводе графика оси рисуются после построения графика, в противном случае оси будут масштабироваться по данным.

```
%%
```

```
%Вывод результатов
varargout{1}=[a A];varargout{2}=[b
B];varargout{3}=[c C];
% Как последний штрих - вывод изображения
if (nargin==5)&&(strcmp(varargin{5},'qrh'))
   figure(1);plot([1 a.*cosd(C)+1 b+1 1], [1
a.*sind(C)+1 1 1]);
   axis([0 (a+b+c)/2 0 (a+b+c)/2]);
end
```

Для считывания результата приходится делать вызов программы не слишком красивым, например, `[x,y,z,t,u,v,w]=MyTriangle(3,4,5,'abc','grh')`. Все числа можно было бы расположить компактно в одной переменной-массиве (в качестве результата).

Задание:

1. Построить график функции, заданной в цилиндрических координатах уравнениями: $z = 1 + \sqrt{Ar}\sin\varphi$, $r = \cos^2 A\varphi$, $\varphi \in [0;2\pi]$. Составить М-файл-сценарий и запустить его несколько раз. На что похожа кривая? Сделать ее «кометой».

Указание: Использовать функции pol2cart, plot3, comet3. Параметр A>0 свободный.

2. Оформить функцию myfun1, имеющую два аргумента (вектор и строку) и один результат – скаляр. Вычисление скаляра зависит от вида строки (предусмотреть три варианта расчета) – например, может рассчитываться среднее арифметическое. Протестировать М-файл.

3. Написать программу, вычисляющую средний балл студента. Пользователь должен в консоли вводить имя студента, название предмета и полученный балл. Первым, в качестве базы, вводится название предмета.

Указание: Приветствуется замена циклов встроенными функциями (find,sort), базу данных объявить глобальной переменной (global) и реализовать как массив ячеек (cell array).

2 Решение некоторых стандартных математических задач

В практике математического моделирования часто приходится сталкиваться с необходимостью решать численно некоторые стандартные задачи. К их числу можно отнести:

- Вычислений значений спецфункций
- Задача интерполяции – узнать каково значение функции в некоторой точке интервала, если заданы ее значения в нескольких его точках.
- Поиск решения системы линейных уравнений, вычисление собственных векторов и пр.
- Поиск корней полинома $P_n(x) = 0$ и вообще, нелинейного уравнения f(x)=0
- Вычисление коэффициентов разложения Фурье (в т.ч. быстрое Фурье–преобразование)
- Вычисление значений производных, дивергенций, определенных интегралов от функций
- Задачи поиска наименьшего и наибольшего значений (т.е. локальных и глобальных экстремумов)
- Поиск решений обыкновенных дифференциальных уравнений и их систем (задача Коши)
- Краевые задачи для некоторых уравнений в частных производных

Всю необходимую информацию можно почерпнуть в разделе справки MATLAB-Mathematics, а также в разделе MATLAB–Functions_by_Category–Mathematics. Кратко о некоторых полезных функциях (по их имени легко восстановить формат и особенности вызова):

legendre(n,X) – вычисление значений полиномов Лежандра порядка n в точках вектора X.

yy = spline(x,Y,xx) – интерполяция кубическими сплайнами yy(xx) (с возможной экстраполяцией), xx – как скаляр, так и вектор.

poly(A) – дает коэффициенты характеристического многочлена матрицы A.

roots(p) – дает корни полинома, заданного вектором

коэффициентов.

fzero(fun,x0) – находит нуль функции, заданной указателем fun, в окрестности x0.

optimset – устанавливает параметры оптимизации (например, для fzero – заметим, что решение алгебраических уравнений может быть сведено к оптимизационной задаче; например, $f(x^*)=0 \leftrightarrow \min f^2(x)$). Параметр DisplayLevel='iter' позволяет отображать на экране ход итераций процесса и тем самым «почувствовать» успешность оптимизации.

fminsearch(fun,x0,options) – ищет безусловный экстремум функций многих переменных.

Для решения более сложных оптимизационных задач см. приложение MATLAB Optimization Toolbox.

gradient(F) – дает градиент функции, заданной таблично.

dblquad(fun,xmin,xmax,ymin,ymax) – двукратный интеграл Римана на прямоугольнике.

fft2(X) – быстрое двумерное Фурье-преобразование.

odeset – устанавливает параметры для решения обыкновенных дифференциальных уравнений (ОДУ)

ode45 – решение ОДУ методом Рунге-Кутты 4-го порядка точности.

dde23 – решение ОДУ с запаздыванием.

Пример:

1. Составить М-функцию, содержащую несколько независимых вложенных функций. Параметров вызова два – строковая переменная S, по которой определяется вызываемая подфункция, и массив коэффициентов K.

```
function M=MyMathSolutions(S,K)
%%
    function M1=M1(x,y)
    M1=1;
    end

%%
```

```
    function M2=M2(x,y)
    M2=2;
    end
%%
    function M3=M3(x,y)
    M3=3;
    end

%%
    function M4=M4(x,y)
    M4=4;
    end
%%
    function M5=M5(x,y)
    M5=1;
    end
%%
switch (S)
    case '1'
        ;
    case '2'
        ;
    case '3'
        ;
    otherwise
        disp([S 'не найдено такого кода']);
end
M=1;
end
```

2. Вычислить производную в точке $x=(x_1,x_2)$ следующей функции $W(x)$:

$$W(x) = u(x,b)J_{2/3}(x_1 - a) - \Gamma(bx_2)(ax_1 + bx_2), \quad u(x,b) = x_1 + bx_2 + |x|^b,$$
$$a, b = \text{const}$$

Здесь J – функция Бесселя первого рода, Г – гамма-функция, u – определенная пользователем функция

Прочитаем справку по используемым спецфункциям. Выпишем первую секцию (введем «защиту от дураков» в виде модуля, поскольку Г(x<0) не определено, как и в нуле):

```
%%
    function M1=M1(x,p)
        user_def_fun=@(x,b)
x(1)+b*x(2)+norm(x).^b;
   M1=user_def_fun(x,p(2)).*besselj(2/3,x(1)-
p(1))-gamma(abs(p(2)*x(2)))*(p*x');
    end
```

В последней секции исправим (нижняя строка К – коэффициенты, верхняя – координаты точки):

```
case '1'
        M=M1(K(1,1:2),K(2,1:2));
                    ....%M=1;
```

После вызова-теста

```
>> MyMathSolutions('1',[1 1;1 -1])
```

получим верный ответ 0. Пробный расчет дает

```
MyMathSolutions('1',[2 3;2 1])= -14.
```

3. В пределах одного полотна построить два графика: один – контурный для W(x), второй – график скоростей для градиента gradW(x). Принять a=1.5, b=2.3, площадка – первая четверть круга радиуса π в центром в (0,0). Шаг сетки для графика скоростей равен π/4.

Для простоты можно было бы использовать циклы, поскольку фактическая трехмерность матрицы данных мешает использовать ezcontourf. И все-таки имеет место такая короткая формулировка (данные транспонируются):

```
%%
    function M2=M2(x,y)
        if (x.^2+y.^2>pi^2) M2=NaN;
           else M2=real(M1([x.' y.'],K(2,1:2)));
        end
    end
```

И в первом приближении нужно добавить в последнюю секцию:

```
case '2'
        ezcontourf(@M2,[0,pi,0,pi]);M=2;
```

В качестве проверки запустим

```
>> MyMathSolutions('2',[2 3;2 1])
```

и получим график (рисунок 14). Неровная белая полоса снизу получается по причине Г(+0)=-∞.

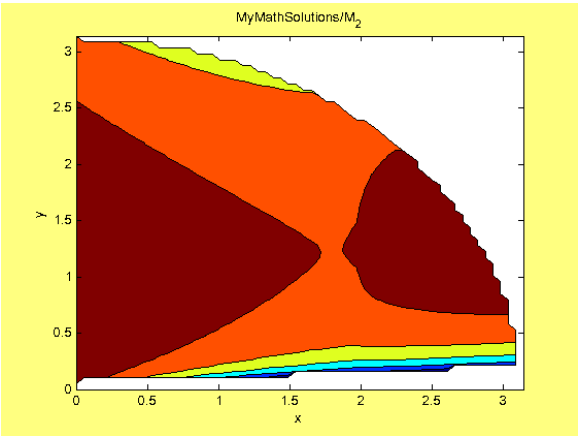

Рисунок 14 – Проверочный график

В ядре MATLAB отсутствует функция по вычислению градиента по заданному указателю функции, что вынуждает нас использовать матрицы. Поскольку градиент берется из данных интерполяции, то нужно сначала построить подробную сетку, взять градиент в ее узлах, затем полученные матрицы вновь интерполировать по нужным нам узлам; в противном случае точность не будет высокой (рисунок 15).

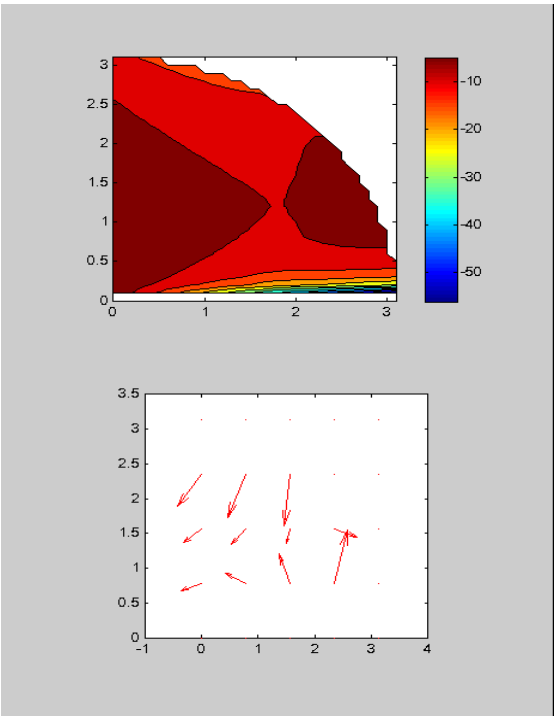

Рисунок 15 - Вычисление градиента по заданному указателю функции

```
%%

function M3=M3(x,y)
    [X,Y]=meshgrid(0:0.1:pi);Z=X;
    for i=1:length(X),
        for j=1:length(Y),
            Z(i,j)=real(M2(X(i,j),Y(i,j)));
        end
    end
    [Zx,Zy]=gradient(Z,0.05,0.05);
    xbase=0:pi/4:pi;[xx,yy]=meshgrid(xbase);
    Fx = griddata(X,Y,Zx,xx,yy);Fy =
griddata(X,Y,Zy,xx,yy);
    scrsz = [1 1 0.4
0.8].*get(0,'ScreenSize');figure('Name',...
        'Simulation Plot
Window','NumberTitle',...
        'off','Position',scrsz);
```

```
subplot(2,1,1);contourf(X,Y,Z);colorbar;axis
square;

subplot(2,1,2);quiver(xx,yy,Fx,Fy,'Color','r');a
xis square;
        M3=0;
    end
```

В данном случае вызов >> MyMathSolutions('3',[2 3;2 1]). Неровность снизу контурного графика исчезает; вектора на втором графике показывают направление возрастания функции. Попробуйте сделать шаг вдвое меньше и рассчитать для других параметров.

4. Вычислить объем n-мерной сферы радиуса 1. Вывести график зависимости V(n)

Математически задача сводится к вычислению n-кратного интеграла, а в отношении программирования более чем уместна рекурсия, т.е. вызов функцией копии самой себя. Однако, MATLAB позволяет вычислять только одно, дву- и троекратные интегралы, а рекурсия в нем не предусмотрена, поэтому воспользуемся методом Монте-Карло:

$$V(n)=\underset{\sum_k x_k^2\le 1}{\iint} dx_1\cdot\ldots\cdot dx_n \quad V(2)=\pi \quad V(3)=\frac{4}{3}\pi$$

$$V(n)=2^n\iiint_{R_+^n} f(x)\rho(x)dx=\overline{f(\xi)} \quad \xi-\text{случайная многомерная величина с плотностью } \rho(x)$$

$$\rho(x)\equiv 1 \quad f(x)=\begin{cases} 0,(x,x)>1 \\ \mu(x)\equiv 1,(x,x)\le 1 \end{cases}$$

Заметим, что метод Монте-Карло обладает плохой сходимостью, пропорциональной корню из числа испытаний √N.

Соответствующие фрагменты выглядят так (была проведена нормировка на число π):

```
   function M4=MySphereVolume(x,p,flag)
    % x- вещественный аргумент, p - целые
параметры, например, размерность
    % при начальном вызове x равен радиусу, а p
- скаляр
        MassOfPoint=@(x)    1;
        function Region=Region(x)
            if (sum(x.*x)>1) Region=0; else
Region=MassOfPoint(x); end
        end

N=2;summa=1;medsumma=0;medsumma1=0.5;eps=1e-8;
        while ((abs(medsumma-
medsumma1)>eps)&&(N*sqrt(eps)<1))

ksi=rand(1,p);summa=summa+Region(ksi);N=N+1;

medsumma=medsumma1;medsumma1=summa/N;
            %если захотим увидеть ход процесса,
каждый сотый шаг
            if (flag&&(~mod(N,100)))
['процессинг: ' num2str(medsumma) ...
                    '  ' num2str(medsumma1)],
end;
        end
        %вторым аргументом идет погрешность
расчета
        M4=[medsumma1*((2*x).^p) 1/sqrt(N)];
    end
….
case '4'
        Z=[];
```

```
        for n=1:10,
res=MySphereVolume(1,n,false), Z=[Z res(1)/pi];
end;
        plot(Z);
```

5. а) Вычислить все корни уравнения $x^3 - 6x + 1 = 0$.

б) Вычислить все корни трансцендентного уравнения на отрезке [-5;5]: $x * \cos(x) + \ln(x^2) = A$ (для некоторых A).

Примечание: излагается решение без M-файла

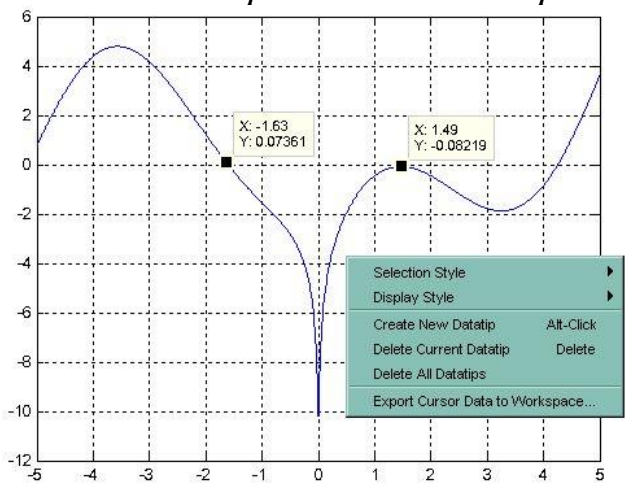

Рисунок 16 – Отрезок $x * \cos(x) + \ln(x^2) = A$

```
>> format long;roots([8 0 -6 1])

ans =

  -0.93969262078591
   0.76604444311898
   0.17364817766693
```

Интересующий нас раздел помощи – MATLAB->Mathematics->Function Functions->Minimizing Functions and Finding Zeros. Чтобы «почувствовать» задачу «на лету» построим график:

```
    A=1; transcen=@(x) x.*cos(x)+log(x.^2)-A; x=-
5:0.01:5;plot(x,transcen(x)), grid on
```

Можно попробовать вручную, мышкой, найти корень. Для этого в графическом редакторе выбрать пиктограмму  , в контекстном меню на кривой выбрать Selection Style – Mouse Position, Display Style – DataTip, выбрать стартовую точку, и, не отпуская кнопку мыши, провести по кривой. График позволяет нам выделить участки перемены знака. Найдем третий по возрастанию корень:

```
>> fzero(transcen,[4 5])
ans =
   4.25053115278136
```

Однако график не позволяет нам определить, является ли второй корень двойным, или мы имеем два близких корня. Здесь, несмотря на многочисленность параметров (см. optimset) fzero, нам придется использовать функцию оптимизации fminbnd:

```
>> transcen1=@(x) -
transcen(x);[x,fx]=fminbnd(transcen1,1,2,optimse
t('Display','iter'))

 Func-count     x          f(x)
Procedure
     1        1.38197    0.0935767        initial
     2        1.61803      0.11398        golden
     3        1.23607     0.170065        golden
     4        1.47297    0.0815729
parabolic
     5        1.47129    0.0815597
parabolic
     6        1.46961    0.0815551
parabolic
     7        1.46955    0.0815551
parabolic
     8        1.46952    0.0815551
parabolic

Optimization terminated:
```

```
 the current x satisfies the termination
criteria using OPTIONS.TolX of 1.000000e-004

x =
   1.46955240774355
fx =
   0.08155506454236
```

Если бы уравнение действительно имело корень, то ордината fx была бы отрицательной. Таким образом, впечатление от графика (о наличии корня) ложное.

6. а) Графически решить систему уравнений (a,b – параметры):

$$\begin{cases} x^2 - ax + y^2 = b^2 \\ x + by = a \end{cases}$$

б) Найти все решения системы:

$$\begin{cases} ax + by = z \\ ax^2 + by^2 = z^2 \\ (x+a)(y+b)(z+1) = 1 \end{cases}$$

По пункту (а) – мы используем контурные графики для линий уровня, соответствующих нулю. Добавим в наш М-файл секцию:

```
function M5=M5(a,b)
        %Решение графически системы двух
уравнений
        %Сетка адаптирована к параметрам a,b
        f1=@(x,y) x.^2+y.^2-a*x-b^2;f2=@(x,y)
x+b*y-1;
        x=fix(-b+a/2-
1):b/100:fix(b+a/2+1);y=fix(-b-
1):0.001:fix(b+1);
```

```
[X,Y]=meshgrid(x,y);Z1=f1(X,Y);Z2=f2(X,Y);
        figure(1);hold on;contour(X,Y,Z1,[0
0],'-r','DisplayName','1');
        contour(X,Y,Z2,[0 0],'-
g','DisplayName','2');
        axis equal; grid on; legend show; hold
off;
        M5=0;
    end
...
case '5'
        M5(K(1,1),K(2,1));
```

По пункту (б) – задача сводится к поиску нулевого экстремума. Добавим еще секцию:

```
%%
function M6=M6(a,b)
        %Решение системы нескольких уравнений
        %Область поиска локальных экстремумов
разбивается на
        %несколько случайным числом
        f1=@(x) a*x(1)+ b*x(2)-x(3);
        f2=@(x) a*x(1).^2+b*x(2).^2-x(3).^2;
        f3=@(x) (x(1)+a).*(x(2)+b).*(x(3)+1)-1;
        %Критерий корня - экстремум суммы
квадратов
        OurFun=@(x)
sqrt(f1(x).^2+f2(x).^2+f3(x).^2);
        Res=[];
        for n=1:100,
            display(['Номер итерации: '
int2str(n)]), pause(0.05);

opt=optimset('Display','off');p=10*rand(1,3)-5;
            [x,fx]=fminsearch(OurFun,p,opt);
```

```
            if (abs(fx)<1e-4) Res=[Res;x fx];
end;
        end;
        %убрать повторы
        if (size(Res,1)==0) display('Решений
нет'),
        else
            Res=Res';Rest=Res(:,1);%keyboard;
            for p=Res,
                flag=true;
                for q=Rest,
                    %keyboard;
                    if (norm(p-q)<1e-3)
flag=false; end;
                end;
                if (flag) Rest=[Rest p]; end;
            end;
        end;
        %проверка правых частей, 3 последних
строки
        Rest=[Rest(1:3,:);Rest(1:3,:)];n=0;
        for p=Rest,
p(4)=f1(p(1:3));p(5)=f2(p(1:3));p(6)=f3(p(1:3));
            n=n+1;Rest(:,n)=p;
            display([int2str(n) '-Аналитическое
решение #(x,y,z): ' num2str(p(1:3)')]),
            %display(['Погрешности от уравнений:
' num2str(p(4:6)')]),
        end;
        M6=Rest;
    End
…
case '6'
        M=M6(K(1,1),K(2,1));
```

При a=1 и b=2 вызов W=MyMathSolutions('6',K) дает всего 5 решений (решения №3,4 можно получить аналитически – при y=0

$x = z = \pm 2^{-1/2} - 1)$:

```
    …
    Номер итерации: 100
    1-Аналитическое решение #(x,y,z): -0.93181
1.8636      2.7954
    2-Аналитическое решение #(x,y,z): 0.16215
-0.32434     -0.4865
    3-Аналитическое решение #(x,y,z): -0.29292
1.8644e-005    -0.29288
    4-Аналитическое решение #(x,y,z): -1.7071 -
2.6909e-006     -1.7071
    5-Аналитическое решение #(x,y,z): 1.103     -
2.2059     -3.3089
```

7. Решить систему обыкновенных дифференциальных уравнений и начертить траекторию движения частицы (параметр a изменяется от -1 до 1 с шагом 0.5):

$$\begin{cases} dx/dt = ay - z\cdot\cos t \\ dy/dt = az - x\cdot\sin t \\ dz/dt = ax - yt \end{cases} \quad \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{t=0} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad 0 < t < \pi$$

Особенность решения ОДУ состоит в том, что выходные вектора многокомпонентны, и точки расположены неравномерно (последняя проблема решается, однако, командой deval). По параметру a мы создаем семейство решений (рисунок 17) и записываем их в 3-х-мерную матрицу. Входная матрица K равна [0 1;3 2].

```
%%

function M7=M7(a,b)
    function RateX=RateX(x,y,z,t)
        RateX=a*y-z.*cos(t);
    end
    function RateY=RateY(x,y,z,t)
```

```
        RateY=a*z-x.*sin(t);
    end
    function RateZ=RateZ(x,y,z,t)
        RateZ=a*x-y.*t;
    end
    function Rate=Rate(time,arg)

Rate=[RateX(arg(1),arg(2),arg(3),time);...
           RateY(arg(1),arg(2),arg(3),time);
RateZ(arg(1),arg(2),arg(3),time)];
    end
    Data=[];Ti=linspace(0,pi,100)';
    for a=-1:0.5:1,
        [T,Y]=ode45(@Rate,[0 pi],b);
        %Шаг интегрирования нельзя задавать
постоянным.
        %Интерполируем данные
        Y1 = interp1(T,Y(:,1),Ti,'spline'); Y2 =
interp1(T,Y(:,2),Ti,'spline');
        Y3 = interp1(T,Y(:,3),Ti,'spline');
Data=cat(3,Data,[Y1 Y2 Y3]);
    end;
    %Данные подсчитаны. Приступим к построению
графиков на одном полотне
    scrsz = get(0,'ScreenSize');

figure('Name','Траектории','Position',[0.1*scrsz
(3:4) 0.8*scrsz(3:4)]);
    for j=1:5,
       subplot(2,3,j);
       plot(Ti,Data(:,1,j),'-',Ti,Data(:,2,j),'-
.', Ti,Data(:,3,j),':');
       grid
on;legend('X(t)','Y(t)','Z(t)');legend('show');
       title(gca,['Параметр равен: a='
num2str(j/2-1.5)]);
    end;
```

```
    % На лишнем месте построим 3D-кривую для
a=0.5

subplot(2,3,6);h=plot3(Data(:,1,4),Data(:,2,4),D
ata(:,3,4));
    set(h,'Color','m'); axis square;grid
on;box('on');title('Параметр 0.5');

legend({'Траектория'},'Orientation','horizontal'
,'location','NorthEast');
    M7=1;
end
…
case '7'
        M=M7(K(1,1),[K(1,2);K(2,2);K(2,1)]);
```

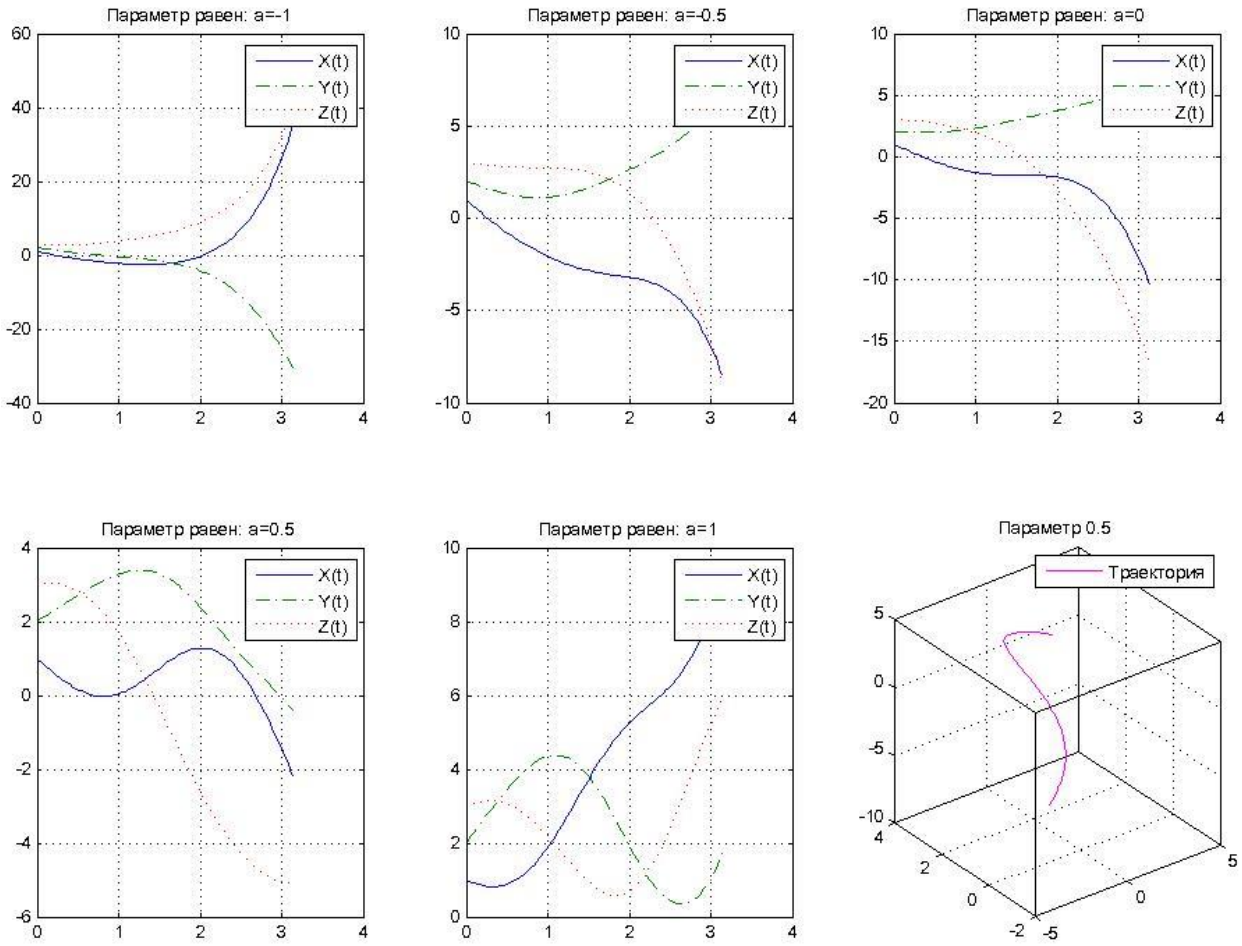

Рисунок 17 – Семейство решений

Задание:

1. Попробуйте п.4. Примера пересчитать с большей точностью, а результаты пересчета отразить на одном графике (для этого немного измените второй фрагмент кода). Поясните полученные результаты. Улучшите код и постройте единый график, адекватный истинным результатам (отобразить их отдельной кривой). Они определяются аналитическим выражением через гамма-функцию: $V(n) = \pi^{n/2} / \Gamma(n/2+1)$.

2. Создать М-файл MyMath.m, вычисляющий значение функции $P(x) = 1 + 2x + 3x^2 + 4x^3 + 5x^5$, а также дающий его корни. На отрезке [-1;1] вычислить коэффициенты быстрого Фурье-разложения P(x) (см. fft) кол-ве 10 шт. Сравнить их с коэффициентами классического Фурье-разложения c_n.

Замечание: Для функции f(x) коэффициенты ряда Фурье задаются формулами (для отрезка [a;b]):

$$a_n = \frac{2}{b-a}\int_a^b f(x)\cos\left(\frac{2\pi}{b-a}kx\right)dx, \quad b_n = \frac{2}{b-a}\int_a^b f(x)\sin\left(\frac{2\pi}{b-a}kx\right)dx,$$

$$c_n = 0.5\sqrt{a_n^2 + b_n^2}$$

3. В рамках того же М-файла реализовать вычисление лапласиана от функции:

$$U_1(x,y,z) = x^2yz - (1+x^2)\frac{\partial}{\partial z}(a \cdot xz^2)$$

$$U_2(x,y,z) = a \cdot P(xy) + \mathrm{erf}(x - y + bz) - b * \frac{\partial}{\partial z}\left(zL_5{}^{(0)}(\sin(zx + ay))\right)$$

Здесь erf – функция ошибок, L_5 – полином Лежандра 5-й степени, a,b- параметры.

Указание: вычислить лапласиан для U_1 аналитически, но не программировать ее. Сверить результаты аналитики и

программирования. Затем адаптировать процедуру для U_2. Раздел справки (а также математические определения) см. в MATLAB-> Functions -- By Category-> Mathematics->Data Analysis-> Finite Differences and Integration, а также Mathematics-> Specialized Math.

4. В рамках того же M-файла реализовать построение 4-х графиков на одном полотне: 1,2,3D-мерного и одного параметрического. На базе данных от функции $U_{1,2}(x,y,z)$.

5. Графическим путем (через пересечение двух кривых) с точностью до 4-го знака найти все корни уравнения $0.1P(x)=U_1(x,1,1)$ – при a=2. На отрезке [-1;2] аппроксимировать $U_1(x,1,1)$ полиномом 5-й степени Q(x). Найти корни уравнения $0.1P(x)=Q(x)$.

6. Пользуясь результатами п.6 Примера, найти все локальные экстремумы функции $U_2(x,y,z)$ при наборах: (a,b)=(1,1),(1,0),(0,1),(-1,0).

7. a) Решить следующую систему на отрезке [-5;5] при $a=\pi$, b=2,c=-3:

$$\begin{cases} dx/dt = \cos(ax+by+cz) \\ dy/dt = \cos(ay+bz+cx) \\ dz/dt = \cos(az+bx+cy) \end{cases} \quad \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{t=0} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \quad 0 < t < 2\pi$$

b) Найти точки Пуанкаре сечения траекторией плоскости z=0.3. Соединить эти точки с соседними с помощью функции gplot.

Практическое занятие №4

Реализация вычислений с помощью графического интерфейса (GUI)

Наличие графического интерфейса служит косвенным признаком коммерциализуемости программного обеспечения. По существу современные стандарты и пользовательские ожидания требуют присутствия GUI (Graphical User Interface) в любой программе. Разумеется, вопрос о том, является ли М-файл полноценной программой, остается дискуссионным – мы отвечаем на него положительно. В научно-исследовательской работе можно обходиться без каких-то оболочек, довольствуясь командной строкой, но лучше затратить сразу немного времени на создание оболочки, чтобы далее в цивилизованных условиях проводить вычислительные эксперименты и отладку содержания математических моделей.

Описание работы по созданию GUI превосходно описано в мультимедийной справке: см. закладку Help-> Demos->MATLAB->Creating Graphical User Interfaces или файл Macromedia Player \demos\CreatingaGUIwithGUIDE.swf. См. также раздел справки Getting Started->Creating Graphical User Interfaces. Существует три основных шага: зарисовка желаемого интерфейса (кнопочки, тестовые боксики и пр.) в редакторе GUIDE, генерация М-кода графической оболочки и модификация последнего и текста своей расчетной программы с целью их связывания. Последний шаг наиболее существенен. Отметим, что вызванной из графического интерфейса программе не возбраняется писать служебные сообщения по-прежнему в командное окно. Ниже рассмотрим простейший пример GUI-программы, вычисляющей сумму двух чисел A+B.

Войдем в редактор -  на главном меню. Уменьшим размеры будущего окна до 200*300 пиксел (пользуясь сеткой). Формат сетки можно задать через Tools->Grid and Rulers. Полезно установить флажок Snap Grid – «Выравнивание по сетке». Теперь отобразим два окна – Инспектора Свойств и Обозревателя Объектов (View->Property Inspector, Object Browser); это можно сделать и через

иконки  .

Каждая кнопка или флажок, перетаскиваемая на полотно с палитры слева, имеют такие свойства как имя (tag), цвет и вызываемую функцию.

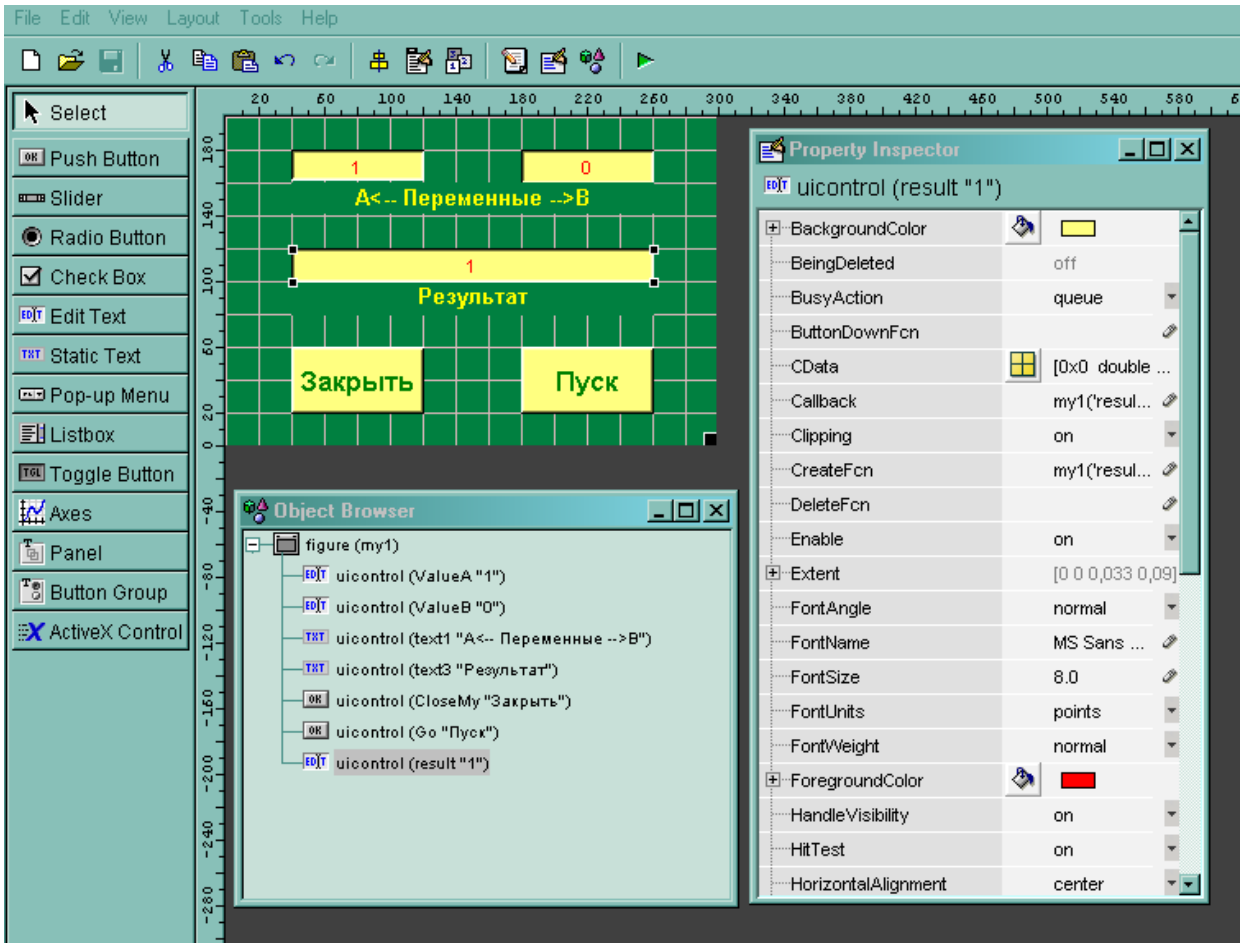

Рисунок 18 - Простейший пример GUI-программы

Менять свойства объекта можно либо через контекстное меню, связанное с ним, либо щелкая по его названию в древе на Обозревателе (свойства автоматически отображаются в Инспекторе). Изменим цвет на зеленый, имя на Summa. Добавим два текстовых бокса под именами ValueA и ValueB. Для этого перетащим первый из Edit Box слева, поменяем его цвета, tag и value. Щелкнув правой кнопкой мыши по нему, выберем Dublicate. На новом объекте изменим его tag и остальные параметры. Создадим статический текст – не путать имя и текстовую метку! Тем же способом создадим бокс (имя – result) и надпись результат и две кнопки Push Button – «Закрыть» и «Пуск».

Перейдем ко второму шагу. Сначала посмотрим опции – Tools->GUI Option. Мы видим, что по умолчанию стоит «Сгенерировать и М-файл, и рисуночный файл *.fig». Поменяем только параметр масштабируемости окна (на proportional). Сохраним файлы под именем my1 (это делается за одну операцию), М-файл откроется автоматически. Готовясь к третьему шагу, подготовим простенький демофайл my1exec.m:

```
function q=my1exec(p)
    q=p(1)+p(2);
end
```

Верхнюю часть кода my1.m-файла редактировать нецелесообразно, о чем нас предупреждают комментарии (рекомендуется внимательно прочитать эти сгенерированные автоматически комментарии). Внесем изменения согласно таблице 1.

Сделаем ряд замечаний. Во-первых, структура файла несколько отличается от обычного М-файла, приближаясь к Си-коду. Все подфункции написаны как бы параллельно, отражая специфику ориентированного на события программирования. Во-вторых, за небольшими уточнениями оказывается, что созданное окно тождественно объекту класса figure. В-третьих, за некоторыми исключениями каждому компоненту окна соответствуют две по названию связанные с ним процедуры – Callback и CreateFcn. Вторая функция автоматически вызывается при его создании (в языке Си аналогично «конструктору» объекта при его инициализации), а вторая – при наступлении некоторых событий, с ним связанных. Напомним, что событием в Windows называется нажатие клавиши, щелчок мыши в определенном месте экрана и прочее вызванное внешними причинами (например, пользователем) изменение среды. Полезно посмотреть Инспектор до и после сохранения М-файла, сравнить изменения полей/свойств, которые заканчиваются на Fcn; кроме того, мы видим, что можно определять не только две функции, связанные с компонентом, а более. В-четвертых, функция OpeningFcn, идущая вверху и относящаяся к figure целиком, является конструктором окна. Путем промежуточного переприсвоения указатель на нее выводится вовне (OutputFcn).

Таблица 1 – Вносимые изменения в программу

Название процедуры	Добавленный код
1	2
my1_OpeningFcn	my.A=0;my.B=0;setappdata(hObject,'mydata',my);

ValueA_Callback	my=getappdata(handles.Summa,'mydata');
	my.A=str2double(get(hObject,'String'));
	setappdata(handles.Summa,'mydata',my);

ValueB_Callback	my=getappdata(handles.Summa,'mydata');
	my.B=str2double(get(hObject,'String'));
	setappdata(handles.Summa,'mydata',my);

CloseMy_Callback	rmappdata(handles.Summa,'mydata'); delete(handles.Summa);

Go_Callback	my=getappdata(handles.Summa,'mydata');
	res=my1exec([my.A my.B]);
	set(handles.result,'String',num2str(res));

Важнейшее значение имеет уникальная для конкретного окна GUI-структура, имя которой автоматически генерируется как handles. По сути это объект класса структура, поля которого являются указателями на все дочерние объекты, включая и сам figure. Она передается параметром в каждую процедуру и позволяет программисту изнутри обмениваться данными между дочерними

компонентами окна. В нее можно записывать и пользовательские данные; поскольку значении переменных среды регистрируются циклически, то при изменении handles (путем прибавления лишнего поля, не обязательно указателя) простого присвоения недостаточно, а следует применять дополнительно команду guidata.

Ключевой момент в создании GUI-программы – это организация обмена данными между расчетной программой и вызывающей ее оболочкой. В первую очередь для этого нужны те переменные, доступ к которым является общим. Один из путей слишком очевиден – обращение к глобальным переменным (см. маркер global). В разбираемом примере реализован путь добавления данных не путем наращения handles, а с помощью организации т.н. пользовательских данных. Они непосредственно не видны из различных функций (например, Callback-ов), но к ним можно получить доступ парой функций – getappdata, setappdata. Поэтому их удобнее всего инкапсулировать в одну структуру; тем не менее, при желании пользователь может ввести несколько таких структур (в отличие от GUI-данных, которые уникальны). В руководстве описан еще один способ – через свойство UserData, ассоциированным с каждым компонентом окна и фундаментальные функции set(handles.NNN,…) и get (здесь NNN – имя компонента, задаваемое в свойстве Tag).

Подробнее разберем пример. Назначение GUI-программы – по числам, вводимым пользователем в два бокса, вычислить при нажатии на кнопку «Пуск» их сумму и отобразить ее в третьем боксе. При нажатии на кнопку «Закрыть» окно закрывается. В функции my1_OpeningFcn на лету создаем двупольную структуру my, затем данные в ней копируем/инкапсулируем в пользовательскую структур mydata. Эта структура впредь будет ассоциирована с figure, указатель на которую временно совпадает с hObject, но статически свопадает с handles.Summa. Следующие две строки таблицы описывают изменение пользовательских данных – функция Callback вызывается после «отщелкивания» курсора от бокса. Триада команд является типической: считывание пользовательских данных (Application Data) в локальную переменную, ее изменение и запись их обратно. Во второй команде по указателю hObject=handles.ValueA(B) мы получаем доступ к

тексту, набранному пользователем в боксе, затем преобразуем строку в число. В предпоследней строке таблицы «для очистки совести» уничтожаем пользовательские данные и закрываем окно традиционной командой уничтожения динамической переменной по указателю. Последняя строка таблицы собственно и содержит то, ради чего писалась оболочка. Считываем пользовательские данные, числа A и B, вызываем параметрически файл my1exec.m с кодами расчета; результаты, возвращаемые в переменную res, напрямую подаются в свойство компонента result (изменение свойств происходит визуально в третьем боксе).

Пример:

1. Реализовать усеченную версию игры «О, счастливчик» в виде последовательности окон. Файл MS Excel содержит базу вопросов-ответов в формате колонок: A – вопрос, B – правильный ответ, CDE – неправильные ответы, F – начисляемые очки за вопрос.

При решении примера вспомним, как мы поступали ранее при зарисовке графиков, создавая множество figure. Теперь наши возможности расширены за счет команд вызова стандартных типов окон – см. MATLAB-> Functions -- By Category-> Creating Graphical User Interfaces-> Predefined Dialog Boxes. И в данном случае не графика довлеет над расчетной программой, а наоборот.

Сценарий программы таков:

1) Спросить пользователя, использовать ли настройки по умолчанию?

2) Yes – задать путь к новой базе, новый шрифт и вид окна ответов, затем показать шкалу прогресса процесса (waitbar);No – самим задать

3) В цикле показывать вопросное окно, верность/неверность ответа и суммарный балл и предложение продолжить

Соответственно после задания пробной базы в Excel (OLbase.xls) представим макет программы Olucky.m:

```
%Primary function
function Olucky=Olucky()

%Default param
function [OLbasepath,OLFont,OLStyle]=OluckyDef()
   OLbasepath=0,OLFont=1,OLStyle=2,
end

%New param
function [OLbasepath,OLFont,OLStyle]=OluckyNew()
   OLbasepath=0,OLFont=-1,OLStyle=-2,
end

%Quest/Answer
function Result=OluckyRes(Number,Summa)
    Result=9,
end

    [a,b,c]=OluckyDef();a,
    [a,b,c]=OluckyNew();b,
    c=OluckyRes(3,4),
    Olucky=777;
end
```

Разумеется, пока синтаксически верная программа ничего не делает. Начнем заполнять подфункцию `OluckyNew`. Соответствующие GUI-вызовы основаны на uigetfile, uisetfont, msgbox:

```
function
[OLbasepath,OLFont1,OLStyle]=OluckyNew()
   [f,fpath]=uigetfile('*.xls','Где
вопросы?','C:\');
   OLbasepath=[fpath,f];
   button=questdlg('Ответы по
вертикали?','Кнопки или
списки...','Да','Нет','Да');
   if (strcmp(button,'Да')), OLStyle=true; else
OLStyle=false; end;
```

```
   h=msgbox('Сейчас вам будет предложено выбрать
шрифт',...
        'Или шрифты','help','non-
modal');waitfor(h);OLFont1=uisetfont(OLFont);
end
```

Отметив, что переменная OLFont есть структура, пишем OluckyDef

```
function [OLbasepath,OLFont,OLStyle]=OluckyDef()
   OLbasepath='OLbase.xls';OLStyle=true;
   OLFont=struct('FontName','Times New
Roman','FontUnits','points',...

'FontSize',12,'FontWeight','normal','FontAngle',
'italic');
end
```

Чтобы код был исполняемым, произведем замену в запускающей части кода.

```
button=questdlg('Применить параметры по
умолчанию?','Вы ленивы?','Да','Нет','Да');
[OLbasepath,OLFont,OLStyle]=OluckyDef();
if(strcmp(button,'Нет')),
[OLbasepath,OLFont,OLStyle]=OluckyNew(); end;
    xlsread(OLbasepath,-1);
```

Составим новую функцию чтения xls-данных и соответственно заменим последнюю строку на вызов этой вложенной фукнции:

```
%Read xls
function [N,Quest,Ans,Weight]=OLread(path)

[Weight,NQuestAns]=xlsread(OLbasepath,'A:F');
    N=size(NQuestAns);N=N(1);
```

```
Quest=NQuestAns(2:N,1);Ans=NQuestAns(2:N,2:5);N=
N-1;
end
    ....
[N,Quest,Ans,Weight]=OLread(OLbasepath);
```

Теперь обратимся к функции, которая должна выполняться в цикле - OluckyRes. Если пользователь решает прервать игру, то она возвращает -1.

```
%Quest/Answer
function Result=OluckyRes(Number,Summa)
    message={{};{};{}};
    message{1}=strcat('После
',int2str(Number(1)),'-го вопроса');
    message{2}=strcat('у вас в кошельке
',int2str(Summa),' тысяч рублей');
    message{3}='Не пора ли забрать деньги?';
    button=questdlg(message,'Отвечать иль не
отвечать?','Да','Нет','Нет');
    %QuestN+AnsN+TrueAnsN - все, что знаем о N-м
вопросе

AnsN={Ans{Number(1),Number(2)},Ans{Number(1),Num
ber(3)},...

Ans{Number(1),Number(4)},Ans{Number(1),Number(5)
}};

QuestN=Quest{Number(1)};TrueAnsN=Ans{Number(1),1
};WN=Weight(Number(1));
    if (strcmp(button,'Да')),
        Result=-1;
    elseif (OLStyle)

[ind,ok]=listdlg('ListString',AnsN,'SelectionMod
e','single',...
```

```
'PromptString',QuestN,'ListSize',[400 100]);
        if (ok==0), ind=1; end;
        if (strcmp(AnsN{ind},TrueAnsN))
            Result=Summa+WN;
        else
            Result=Summa;
        end
    else
        message=strvcat('Мудрецы
говорят:',AnsN{1},'Соглашаясь, нажмите
крестик');

button=questdlg(message,QuestN,AnsN{2},AnsN{3},A
nsN{4},AnsN{3});
        if (strcmp(button,'')), button=AnsN{1};
end;
        if (strcmp(button,TrueAnsN))
            Result=Summa+WN;
        else
            Result=Summa;
        end;
    end;
end
```

По параметру OLStyle определяем, что применяем – listdlg или questdlg. Перед этим, однако, загружаем из основной базы данных вопрос под номером QuestN и соответствующие ответы и цену ответа. Если ответ оказывается правильным, то сумма приращивается на цену ответа, выводясь в переменную Result.

После того, как мы определились с реализацией опроса по некоторому вопросу из базы данных, нужно написать функцию сортировки ответов (чтобы пользователь не привык к правильности первого варианта).

```
%SortAnswer
function Num=OluckySort(Num)
    for i=1:20,
```

```
ind=floor(1+4*rand(1));ind1=floor(1+4*rand(1));

v=Num(ind1);Num(ind1)=Num(ind);Num(ind)=v;
    end
end
```

Теперь в основное тело программы осталось добавить цикл.

```
%Number - номер вопроса по счету, номера ответов
в колонке xls... если равен 1, то
%правильный
Summa=0;Numb=[1 2 3 4];
for n=1:N,
    Numb=OluckySort(Numb);Number=[n
Numb];Result=OluckyRes(Number,Summa);
    if (Result<0), break, end;
    Summa=Result;
end;
Olucky=Summa,
```

Мы показали, что даже с помощью стандартных диалоговых окон в MATLAB можно создать нечто практически полезное. Разумеется, с помощью созданного самими интерфейса можно было бы сделать игру более элегантной и «отшлифованной».

2. В масштабируемом окне создать интерфейс, содержащем большинство элементов палитры, с целью адаптировать его к программе MyTriangle (см. пример №3 Практического занятия №3, часть 1).

3.

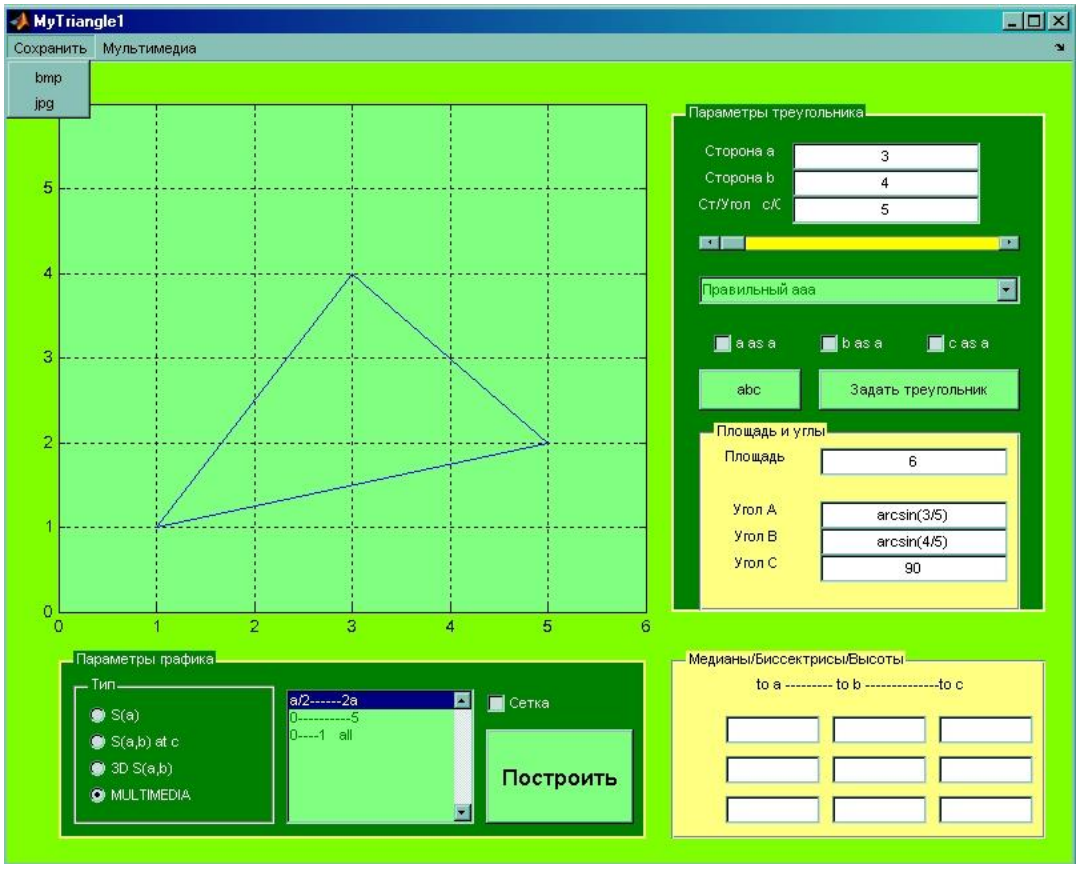

Рисунок 19 – Создание интерфейса, содержащего большинство элементов палитры

Нулевым шагом здесь является просмотр файта MyTriangle.m и зарисовка на бумаге эскиза будущего GUI-окна.

Окно имеет четыре больших компонента: оси, панель выбора параметров треугольника (внутри нее не поместившаяся снизу подпанель «Площадь и углы»), панель «Медианы/ Биссектрисы/Высоты» и панель «Параметры графика». Сценарий таков.

Пользователь задает длины сторон треугольника, причем третью он может задать через полосу прокрутки. Дополнительно в развертывающемся меню он может выбрать тип треугольника, если не хочет вводить лишних сторон (например, если треугольник прямоугольный). Три «галочки» даны, если он захочет поменять порядок следования сторон. «Утопленная» кнопка с надписью «abc» играет роль четвертой галочки на случай, если треугольник задается через две стороны и угол между ними. После этого нажимается кнопка «Задать треугольник», он решается, и

полупустые боксы ниже заполняются значениями углов, площади, медиан и т.п. При этом также сам треугольник рисуется на осях. Затем пользователь может перейти к более сложным задачам, связанным с расчетом площадей при разных длинах сторон – это последняя панель. Здесь он выбирает: какой тип графика ему нужен, нужно ли отображать линии сетки и внешний бокс, способ задания пределов изменения. Разумеется, все завершает кнопка «Построить».

Отметим некоторые особенности. Для масштабируемости окна выбрать GUI-options свойство Resize behaviour как Propotional до его сохранения Для установки размеров окна нужно в его (figure) Инспекторе свойств задать: Units=pixels, Position=[100, 100, 800, 600]. Для задания позиций popup-меню и/или Listbox нужно в Инспекторе найти свойство String и щелкнуть по пиктограмме  . Все панели, кроме панели «Тип», имеют тип Panel; указанная же панель имеет тип buttongroup. Для создания меню окна необходимо исполнить пункт меню GUIDE Tools->Menu Editor. Пункты меню «Сохранить->bmp» и «Сохранить->jpg» добавлены с тем, чтобы пользователь мог сохранять картинку, изображенную на осях. Заметим, что можно было поступить более просто, выведя обычное меню окна figure (проставив в Инспекторе его свойств Menubar=figure), но гибче составить свое меню. Что касается создания панелей инструментов, тосредствами GUIDE это невозможно, однако, в режиме командной строки можно использовать команду uitoolbar.

Что касается зарисовки «парадного треугольника», то здесь пришлось программировать. Мы выбрали несколько более сложный путь, пренебрегая советом писать код в функцию OpeningFcn окна. Вместо этого, не найдя функции CreateFcn для осей axes1 в сгенерированном коде MyTriangle1.m, нужно создать ее следующим универсальным способом – нужно выделить в GUIDE оси, в контекстном меню выбрать View->CreateFcn; при этом во-первых, в М-файл дописывается кусок кода, и во-вторых, связь с этим фрагментом задается явно внутри fig-файла (посмотрите в Инспекторе свойсв для axes1 поле CreateFcn). Осталось только для верности сохранить оба файла – это делается стандартным способом.

В сгенерированный фрагмент кода необходимо дописать несколько строк, причем обратите внимание на последнюю строчку. Ее присутствие вызвано тем, что структура handles заполняется указателями только после исполнения всех CreateFcn от дочерних объектов окна.

```
function axes1_CreateFcn(hObject, eventdata,
handles)
x=[1 3 5 1];y=[1 4 2 1];plot(x,y);
axis([0 6 0 6]);axis manual;grid on;
handles.axes1=hObject;guidata(gcf, handles);
```

Заполним сразу и код-обработчик пункта меню «Сохранить». Для примера рассмотрим сохранение bmp-файла (для простоты он сохраняется с именем photo в текущий каталог, легко можно было бы сделать сохранение гибче – с помощью диалоговых окон). Итак, читается указатель на оси, вычисляется абсолютная позиция осей axes1, из-за особенностей фукнции getframe нам необходимо немного расширить прямоугольную область, чтобы в «снимок» уместились шкалы осей, поэтому немного изменяем вектор rect. Далее связка трех функций: снятия снимка в переменную photo, затем ее перекодировка (с расщеплением) в пиксельный образ и сохранение последнего в файл. Написание Callback-а для подпункта jpg очевидно.

```
function bmp_Callback(hObject, eventdata,
handles)
h=handles.axes1;rect=get(h,'Position');rect1=get
(h,'OuterPosition');
rect=[rect1(1)-rect(1) rect1(2)-rect(2) rect1(3)
rect1(4)];
photo=getframe(h,rect);[photo,cmp]=frame2im(phot
o);
imwrite(photo,'photo.bmp','bmp');beep;
```

4. Реализовать «решение треугольника» и более сложные функции в GUI-программе

Помимо fig-файла будем использовать три M-файла: MyTriangle1.m (он уже сгенерирован и является управляющим), MyTriangle0exec.m (копия MyTriangle.m, решающая единичный треугольник) и MyTriangle1exec.m (здесь проводятся все действия по построению графиков, включая, например, параметрические расчеты кривых). Управляющий файл, разумеется, будет содержать прикладные данные, но не следует «втаскивать» в них какие-то многомерные массивы, нужные для построения графиков – достаточно просто передать при вызове исполняемых файлов указатель на те оси, в пределах которых и будут строиться кривые.

Начнем с изначальной установки данных приложения. Добавим в функцию MyTriangle1_OpeningFcn GUI-файла блок:

```
%Задание данных приложения
mt=struct('Square',6,'Sides',[3 4 5],'Angles',[0
0 90],...
    'Medians',[1 1 1],'Bisectors',[2 2 2],
'Heights',[3 3 3]);
setappdata(hObject,'MTriangle',mt);
```

Обратимся к панели задания треугольника и запишем обработчики текстовых боксов через View->Callbacks (поскольку боксы находятся на панели, то, к сожалению, функции вызова для них не были сгенерированы ранее, но, тем не менее, как не трудно убедиться, указатели на каждый бокс присутствуют в структуре hahdles). Например, для бокса, соответствующего стороне b:

```
function edit2_Callback(hObject, eventdata,
handles)
mt=getappdata(handles.output,'MTriangle');
mt.Sides(2)=str2double(get(hObject,'String'));
setappdata(handles.output,'MTriangle',mt);
```

Что касается стороны c, то вначале запишем обработчик для полосы прокрутки:

```
function slider1_Callback(hObject, eventdata,
handles)
mt=getappdata(handles.output,'MTriangle');
pos=get(hObject,'Value');flag=get(handles.toggle
button1,'Value');
if (flag)
    pos1=abs(mt.Sides(1)-
mt.Sides(2));pos2=mt.Sides(1)+mt.Sides(2);
    pos=pos1+pos*(pos2-pos1);mt.Sides(3)=pos;
else
    pos=180*pos;mt.Angles(3)=pos;
end;
set(handles.edit3,'String',num2str(pos));
setappdata(handles.output,'MTriangle',mt);
```

Первая ветка условного оператора соответствует ненажатой кнопке «abc» (возвращаемое ею значение равно 1) и ситуации ввода трех сторон. Отметим, что мы не стали принудительно изменять для слайдера его предельные значения (они равны 0 и 1).

Обработчик для третьего бокса выглядит так:

```
function edit3_Callback(hObject, eventdata,
handles)
mt=getappdata(handles.output,'MTriangle');
pos=str2double(get(hObject,'String'));flag=get(h
andles.togglebutton1,'Value');
if (flag)
    mt.Sides(3)=pos;
    pos1=abs(mt.Sides(1)-
mt.Sides(2));pos2=mt.Sides(1)+mt.Sides(2);
    pos=(pos-pos1)/(pos2-pos1);
else
    mt.Angles(3)=pos;pos=pos/180;
end;
set(handles.slider1,'Value',pos);
setappdata(handles.output,'MTriangle',mt);
```

Напишем обработчик для кнопки-триггера (начальное значение установим как 1). Обратим внимание, что поле Value обновляется автоматически, и мы лишь сменяем надпись и цвет кнопки:

```
function togglebutton1_Callback(hObject,
eventdata, handles)
q=get(hObject,'Value');
if (~q)
    set(hObject,'String','abC');
    set(hObject,'BackgroundColor',[1 1 0.5]);
else
    set(hObject,'String','abc');
    set(hObject,'BackgroundColor',[0.5 1 0.5]);
end;
```

Напишем фрагмент обработчика всплывающего меню (с учетом показаний «галочек» - здесь есть некая избыточность). Выбранный тип треугольника влияет на показания полей боксов сторон a,b,c, но еще не на сами данные приложения.

```
function popupmenu1_Callback(hObject, eventdata,
handles)
pos=get(hObject,'Value');mt=getappdata(handles.o
utput,'MTriangle');
flags=[get(handles.checkbox1,'Value')
get(handles.checkbox2,'Value')
get(handles.checkbox3,'Value')],
switch (pos)
    case 1
        side=1;for i=1:3, if flags(i)
side=mt.Sides(i); end; end;

set(handles.edit1,'String',num2str(side));

set(handles.edit2,'String',num2str(side));

set(handles.edit3,'String',num2str(side));
```

```
    case 4
        side=(mt.Sides(1)^2+mt.Sides(2)^2)^0.5;

set(handles.edit3,'String',num2str(side));
    otherwise ;
end
```

Теперь перейдем к обработке кнопки запуска. Имеющиеся данные, если поля менялись, должны составить данные приложения. Структуру данных MTriangle надлежит также дозаполнить вызовом уже расчетных процедур. Затем результаты следует отобразить в боксах панели «Медианы/…» и построить треугольник.

```
function pushbutton1_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    structure with handles and user
data (see GUIDATA)
%Фиксация входных данных
mt=getappdata(handles.output,'MTriangle');
ra=str2num(get(handles.edit1,'String'));mt.Sides
(1)=ra;
rb=str2num(get(handles.edit2,'String'));mt.Sides
(2)=rb;
rc=str2num(get(handles.edit3,'String'));
if (get(handles.togglebutton1,'Value'))
    mt.Sides(3)=rc;
else
    mt.Angles(3)=rc;
end
rflag=get(handles.togglebutton1,'String');
%Решение треугольника
[S,aA,bB,cC,Ma,Mb,Mc]=MyTriangle0exec(ra,rb,rc,r
flag,'med');
```

```
mt.Sides=[aA(1) bB(1) cC(1)];mt.Angles=[aA(2)
bB(2) cC(2)];
mt.Medians=[Ma Mb Mc];mt.Square=S;
[S,aA,bB,cC,Ma,Mb,Mc]=MyTriangle0exec(ra,rb,rc,r
flag,'bis');mt.Bisectors=[Ma Mb Mc];
[S,aA,bB,cC,Ma,Mb,Mc]=MyTriangle0exec(ra,rb,rc,r
flag,'hhh');mt.Heights=[Ma Mb Mc];
setappdata(handles.output,'MTriangle',mt);
%Обновление полей
set(handles.edit3,'String',num2str(mt.Sides(3)))
;
set(handles.edit4,'String',num2str(mt.Square));h
=0;
for i=1:3,
    command=['h=handles.edit' int2str(4+i)
';'];eval(command);
    set(h,'String',num2str(mt.Angles(i)));
    command=['h=handles.edit' int2str(7+i)
';'];eval(command);
    set(h,'String',num2str(mt.Medians(i)));
    command=['h=handles.edit' int2str(10+i)
';'];eval(command);
    set(h,'String',num2str(mt.Bisectors(i)));
    command=['h=handles.edit' int2str(13+i)
';'];eval(command);
    set(h,'String',num2str(mt.Heights(i)));
end;
%Построение треугольника
x=zeros(1,4);y=x;x(2)=mt.Sides(1);x(3)=x(2)-
mt.Sides(2)*cosd(mt.Angles(3));
y(3)=y(2)+mt.Sides(2)*sind(mt.Angles(3));
x=x+abs(mt.Sides(1)-
mt.Sides(2))*ones(1,4);y=y+ones(1,4);
axes(handles.axes1);cla;plot(x,y,'g-
');p=sum(mt.Sides)/2+1;
set(gca,'Color',[1 1 0.5]);axis([0 p 0 p]);grid
on;box on;
```

Чтобы как-то сократить число однотипных выводов в боксы нижней панели, мы воспользовались функцией eval. При построении треугольника удобно вначале сделать текущими осями оси axes1 (см. команду axes). Для лучшего вида окна добавим в функцию OpeningFcn начальные значения popup-меню и «галочек», а также начального значения в группе радиокнопок «Тип»:

```
% Доустановка параметров
set(handles.popupmenu1,'Value',6);
set(handles.checkbox1,'Value',1);
set(handles.radiobutton1,'Value',1);
set(handles.uipanel4,'UserData',1);
set(handles.listbox1,'UserData',[0 5]);
```

Работа с панелями кнопок имеет особенности. Хотя существуют указатели на каждую радиокнопку отдельно, равно как и указатель на группу в целом, функция обработчик составляется на группу и имеет название не Callback, а SelectionChangeFcn (это название проявляется в контекстном меню View Инспектора свойств для группы). В частности, такой механизм гарантирует единственность выбранного варианта. Шаблон функции перепишем из примера:

```
function uipanel4_SelectionChangeFcn(hObject,
eventdata, handles)
% hObject    handle to uipanel4 (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    structure with handles and user
data (see GUIDATA)
switch get(hObject,'Tag')   % Get Tag of
selected object
    case 'radiobutton1'
        set(handles.uipanel4,'UserData',1);
    case 'radiobutton2'
        set(handles.uipanel4,'UserData',2);
    case 'radiobutton3'
```

```
        set(handles.uipanel4,'UserData',3);
    case 'radiobutton4'
        set(handles.uipanel4,'UserData',4);
end
```

Здесь нам важно сохранить выбор в какой-то переменной; удобным путем представляется использование такого способа сохранения данных, как поле UserData, в нашем случае, принадлежащего объекту buttongroup.

Теперь настало время создать второй исполняемый файл MyTriangle.m. Достаточно будет вызывать его с 4-мя параметрами: hax – указатель на оси (handles.axes1), mt – номер процедуры, sides – вектор-строка со сторонами треугольника, limits – вектор пределов изменения (его трактовка зависит от функции). Тело файла, разумеется, содержит 4 вложенных функции, вызываемых из главной через оператор switch.

```
function mt=MyTriangle1exec
(hax,mt,sides,limits)
    function MT1=MT1()
    MT1=0;
    end
    function MT2=MT2()
    MT2=0;
    end
    function MT3=MT3()
    MT3=0;
    end
    function MT4=MT4()
    MT4=0;
    end

switch mt
    case 1
        MT1()
    case 2
        MT2()
    case 3
```

```
        MT3()
    case 4
end
mt=0;
end
```

Некоторым видоизменением решенного ранее примера №5-10 Работы №2 (часть 2) легко наполнить тела трех вложенных функций. Эффективным оказывается не циклический вызов MyTriangle0exec.m, а векторный вызов файла geroncore.m.

Перед оператором switch нужно расположить две команды, стирающие с осей прежние графики и устанавливающие оси на GUI-окне текущими: `axes(hax);cla reset;`

```
function MT1=MT1()
    hold on;
a=linspace(limits(1),limits(2),21);c=sides(3);
    b=min([abs(c-limits(1)) abs(c-limits(2))]);
    b=linspace(b,c+limits(2),8);MyColor=['y' 'm'
'c' 'r' 'g' 'b' 'w' 'k'];
    Hlines=zeros(1,8);
    for j=1:8,
        Hline=plot(a,geroncore(a,b(j),c));
Hlines(j)=Hline;
        Lname=strcat(num2str(j),': b=
',num2str(b(j)));

set(Hlines(j),'Color',MyColor(j),'DisplayName',
Lname);
    end;

legend('show');xlabel('a');ylabel('S(a)');axis
on;axis tight;hold off;
    MT1=0;
    end
    function MT2=MT2()
```

```
a=linspace(limits(1),limits(2),21);b=linspace(li
mits(3),limits(4),21);

[X,Y]=meshgrid(a,b);contourf(X,Y,geroncore(X,Y,s
ides(3)));colorbar;
    Sname=strcat('S(a,b) at c=
',num2str(sides(3)));title(Sname);
    xlabel('a');ylabel('b');axis on;axis tight;
    MT2=0;
    end
    function MT3=MT3()

a=linspace(limits(1),limits(2),21);b=linspace(li
mits(3),limits(4),21);

[X,Y]=meshgrid(a,b);surf(X,Y,geroncore(X,Y,sides
(3)));colorbar;
    Sname=strcat('S(a,b) at c=
',num2str(sides(3)));title(Sname);

xlabel('a');ylabel('b');zlabel('S(a,b)');axis
on;axis tight;
    MT3=0;
    end
    function MT4=MT4()

a=linspace(limits(1),limits(2),101);b=linspace(l
imits(3),limits(4),101);

c=linspace(limits(5),limits(6),101);[X,Y]=meshgr
id(a,b);
    S=geroncore(a(50),b(50),c(50));MM=[];

Hsur=surf(X,Y,S*ones(101),'LineStyle','none');co
lorbar; colormap jet;
    axis manual;axis([a(1) a(101) b(1) b(101) 0
2*S+2]);title('');
```

```
xlabel('a');ylabel('b');zlabel('S(a,b)');grid
on;box on;

rect=get(hax,'Position');rect1=get(hax,'OuterPos
ition');
    rect=[rect1(1)-rect(1) rect1(2)-rect(2)
rect1(3) rect1(4)];
    for t=c,
        [S,rect1]=sprintf('Время C= %2.3f
сек',t);title(S);
        set(Hsur,'ZData',geroncore(X,Y,t));
drawnow;
        MM=[MM,getframe(hax,rect)];
    end;
    movie2avi(MM,'mt4.avi','fps',2);
    MT4=0;
    end
```

Вернемся к GUI-файлу, точнее, к обработчику списка на панели «Параметры графика». Содержательную часть мы можем либо оставить для Callback-а кнопки пуска, либо частично обработать данные здесь же, используя вновь UserData для списка. Выбирая второе, будем иметь для обработчика:

```
function listbox1_Callback(hObject, eventdata,
handles)
flag=get(hObject,'Value');mt=getappdata(handles.
output,'MTriangle');
switch flag
    case 1
        a=mt.Sides(1);a=[a/2
2*a];b=mt.Sides(2);b=[b/2 2*b];
        c=mt.Sides(3);c=[c/2 2*c];
    case 2
        a=[0 5];b=[0 5];c=[0 5];
    case 3
        a=[0 1];b=[0 1];c=[0 1];
```

```
end;
flag=get(handles.uipanel4,'UserData');
switch flag
    case 1
        set(hObject,'UserData',a);
    case 2
        set(hObject,'UserData',[a b]);
    case 3
        set(hObject,'UserData',[a b]);
    case 4
        set(hObject,'UserData',[a b c])
end;
```

Теперь с обработчиком кноки пуска стало проще. Заметим, однако, что это сделало работу GUI-интерфейса менее корректной – ведь мы стали зависимыми от порядка нажатия на группу переключателей и список. Более правильный путь состоит в том, чтобы опрос всех параметров (переключателей, «галочек» и пр.) осуществлялся бы обработчиком кнопки пуска.

```
function pushbutton2_Callback(hObject,
eventdata, handles)
mt=getappdata(handles.output,'MTriangle');
hax=handles.axes1;mt_=get(handles.uipanel4,'User
Data');
sides=mt.Sides;limits=get(handles.listbox1,'User
Data');
mt=MyTriangle1exec(hax,mt_,sides,limits);
```

Осталось только дописать простенький обработчик проверочного бокса «Сетка» и оставить ремарку пользователю:

```
function checkbox4_Callback(hObject, eventdata,
handles)
grid;
function figure1_DeleteFcn(hObject, eventdata,
handles)
disp('Приятных сновидений')
```

Задание:

1. Рассмотреть стандартные типы окон, включая диалоговые. Адаптировать окно (quidetemplate1.fig в каталоге toolbox\matlab\uitools\guitemplates) под какую-либо простую программу.
2. Создать GUI-окно, обладающее панелью инструментов, меню с тремя уровнями вложенности, которое возможно было бы масштабировать (включая пропорциональное увеличение осей и шрифтов). На панель инструментов вынести по крайней мере три пиктограммы. Предусмотреть небольшой логотип на окне

Указание: Возможности GUIDE здесь ограничены, в функции OpeningFcn можно предусмотреть команды постановки дополнительных элементов в окно.

3. Реализовать на базе Примера игру «О, счастливчик» с собственным GUI-интерфейсом.

Дополнительные примеры

Матричные комплекснозначные вычисления

Создаем случайную комплекснозначную матрицу и вычисляем несколько значений функции Бесселя.

```
>> x=rand(5,5)+(0+1i)*rand(5,5)

x =

   0.6649  +  0.6020i   0.6739  +  0.4222i
0.5485 + 0.8580i   0.7009 + 0.4983i   0.6343 +
0.8983i
   0.3654  +  0.2536i   0.9994  +  0.9614i
0.2618 + 0.3358i   0.9623 + 0.4344i   0.8030 +
0.7546i
   0.1400  +  0.8735i   0.9616  +  0.0721i
0.5973 + 0.6802i   0.7505 + 0.5625i   0.0839 +
0.7911i
   0.5668  +  0.5134i   0.0589  +  0.5534i
0.0493 + 0.0534i   0.7400 + 0.6166i   0.9455 +
0.8150i
   0.8230  +  0.7327i   0.3603  +  0.2920i
0.5711 + 0.3567i   0.4319 + 0.1133i   0.9159 +
0.6700i

>> bessel(0.6,x)

ans =

   0.6513  +  0.2129i   0.5987  +  0.1500i
0.7188 + 0.3544i   0.6288 + 0.1674i   0.7707 +
0.3259i
   0.4243  +  0.1477i   0.8920  +  0.1439i
0.3871 + 0.2205i   0.6758 + 0.0800i   0.7521 +
0.2024i
   0.5298  +  0.5571i   0.6231  +  0.0137i
```

```
0.6576 + 0.2658i    0.6641 + 0.1711i    0.4623 +
0.5360i
      0.5860  +  0.2142i      0.3515  +  0.4157i
0.1348 + 0.0727i    0.6799 + 0.1904i    0.8091 +
0.1503i
      0.7475  +  0.1884i      0.4303  +  0.1697i
0.5443 + 0.1510i    0.4387 + 0.0612i    0.7420 +
0.1374i
```

Немного меняем ее минор поэлементным умножением:

```
    >>              x(2:3,2:3)=[1              2;3
4];x(2:3,2:3)=x(2:3,2:3).*x(2:3,2:3)

    x =

      0.6649  +  0.6020i      0.6739  +  0.4222i
0.5485 + 0.8580i    0.7009 + 0.4983i    0.6343 +
0.8983i
      0.3654    +    0.2536i            1.0000
4.0000              0.9623 + 0.4344i    0.8030 +
0.7546i
      0.1400    +    0.8735i            9.0000
16.0000             0.7505 + 0.5625i    0.0839 +
0.7911i
      0.5668  +  0.5134i      0.0589  +  0.5534i
0.0493 + 0.0534i    0.7400 + 0.6166i    0.9455 +
0.8150i
      0.8230  +  0.7327i      0.3603  +  0.2920i
0.5711 + 0.3567i    0.4319 + 0.1133i    0.9159 +
0.6700i
```

Решения

Практическое занятие №2, п.1.

3.

```
 % Книга расположена в текущем каталоге 21.xls
 %Для корректного задания ссылок проверить в
Excel(2003) стиль ссылок
 %Убрать флаг Excel->Сервис->Параметры->Общие-
>Стиль ссылок R1C1
 >> W=xlsread('21.xls', 1, 'A1:I5')

 W =

      1      1      2    NaN      1      3     -1
0     1
      3      5      5    NaN      1     -5      2
3     4
     -1      2     -2    NaN      2      5     -2
1    11
      0      3      1    NaN    NaN    NaN    NaN
NaN   NaN
      1      4    -11    NaN    NaN    NaN    NaN
NaN   NaN
 >> B=W(:,1:3),A=W(1:3,5:9)

 B =

      1     1     2
      3     5     5
     -1     2    -2
      0     3     1
      1     4   -11

 A =
```

```
      1     3    -1     0     1
      1    -5     2     3     4
      2     5    -2     1    11
```

4.

```
>> A(2,3)=x*A(2,3)-1, B(3,1)=sin(y)*B(3,1)+1
```

5.

```
>> A.*B'

ans =

    1.0000     9.0000    -0.3530              0
1.0000
    1.0000   -25.0000    -5.4000    9.0000
16.0000
    4.0000    25.0000     4.0000    1.0000  -
121.0000
```

6.

```
>>                  v=linspace(1,1,2);C=diag([v
1])+diag(x*v,1)+diag(y*v,-1),

C =

    1.0000   -0.8500         0
    9.7856    1.0000   -0.8500
         0    9.7856    1.0000

>> A*B+C

ans =

   11.6470   17.1500    8.0000
   -1.1676   -3.4000  -59.4500
```

```
    27.2939   79.7856  -86.0000
```

7.

```
 >>  dlmwrite('myfile.txt',  ans,  'delimiter',
'\t', 'precision', 6)
```

8.

```
 >> trace(inv(ans))

 ans =

     0.1305
```

Практическое занятие №2, п.2.

1.

% Ваши численные результаты в практическом занятии могут отличаться

% ввиду случайности значений элементов матриц. Для самопроверки можно

% принудительно ввести именно данные матрицы A и B

```
    >>       A=rand(3)+i*rand(3),       B=expm(A),
z=B\ones(3,1)
    A =

       0.9501  +  0.4447i      0.4860  +  0.9218i
0.4565 + 0.4057i
       0.2311  +  0.6154i      0.8913  +  0.7382i
0.0185 + 0.9355i
       0.6068  +  0.7919i      0.7621  +  0.1763i
0.8214 + 0.9169i


    B =
```

```
       0.7557  +  1.3942i   -1.0258  +  2.5609i   -
1.2289 + 0.8812i
      -1.7591  +  1.0176i    0.0147  +  1.6752i   -
2.2026 + 0.9689i
      -1.0854  +  2.3869i   -0.8267  +  1.9390i   -
0.1626 + 2.1086i


    z =

      -0.0997 - 0.1304i
       0.0039 - 0.3194i
      -0.0813 + 0.0232i
```

2.

```
    % Тех же результатов можно было бы достичь
через команду diag или даже без нее
    >>
C=cat(1,cat(2,A*B,B),cat(2,A,A.*B));D1=real(C);D
2=imag(C)
    D1 =

       -3.1588   -4.8146   -4.4527    0.7557   -
1.0258   -1.2289
       -5.2554    -4.8658    -5.4803    -1.7591
0.0147   -2.2026
       -5.2456   -5.3915   -5.3599   -1.0854   -
0.8267   -0.1626
        0.9501    0.4860    0.4565    0.0980   -
2.8592   -0.9185
        0.2311    0.8913    0.0185   -1.0329   -
1.2235   -0.9472
        0.6068    0.7621    0.8214   -2.5489   -
0.9718   -2.0669
```

```
    D2 =

        1.1829      3.3544     -0.3721      1.3942
2.5609    0.8812
       -0.5754      0.7272     -1.4280      1.0176
1.6752    0.9689
        2.8754      2.8557      1.4947      2.3869
1.9390    2.1086
        0.4447      0.9218      0.4057      1.6607
0.2990   -0.0963
        0.6154      0.7382      0.9355     -0.8474
1.5040   -2.0425
        0.7919      0.1763      0.9169      0.5889
1.3320    1.5829
```

3.

```
    >>    Electr=@(x,y)    sum(sum(((x-D1).^2+(y-
D2).^2).^(-0.5))); Electr(0,0)
    ans =

       21.4684
```

4.

```
    % при нехватке памяти/мощности подождать,
уменьшить число точек до 100 или 500
    >> [X,Y]=meshgrid(linspace(-2,2,1000));
    >> Z=[];for k=1:length(X), for n=1:length(Y),
Z(k,n)=Electr(X(1,k),Y(n,1)); end ; end
    >>                              contourf(X,Y,Z,
logspace(0,1.8,10));colorbar;
```

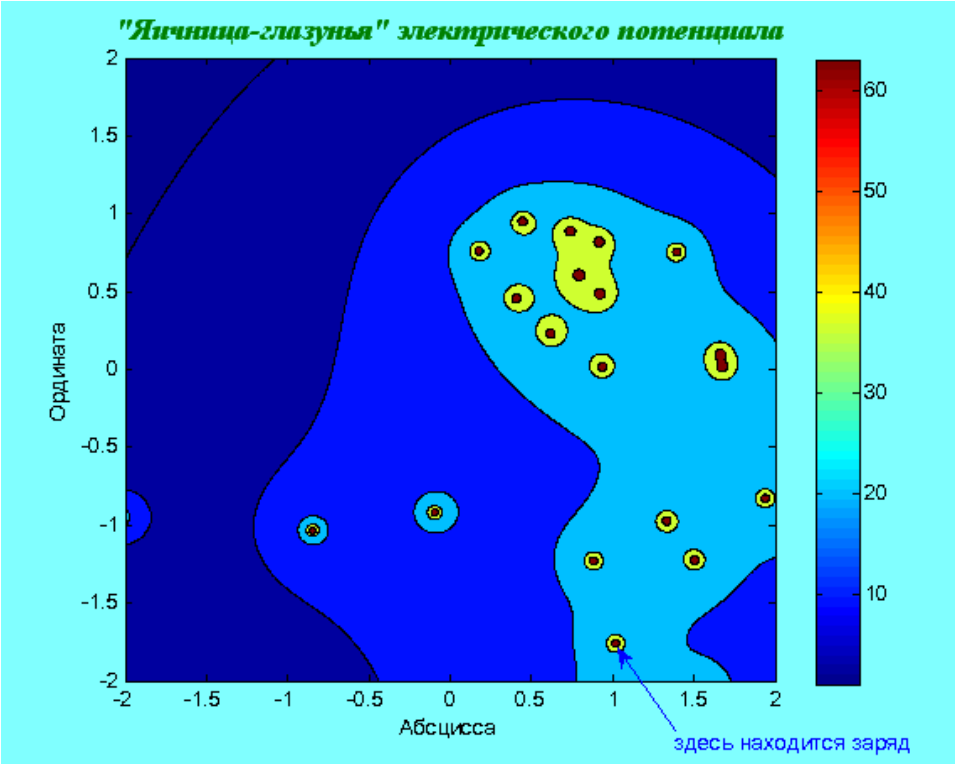

Рисунок 20 – «Яичница-глазунья» электрического потенциала

5.

```
>> h=4/1000;[U,V] = gradient(Z,h,h);
% Делаем матрицы разреженными, чтобы стрелки не сливались
>> N=20;NN=1000/N;x=[];y=[];z=[];u=[];v=[];
>> for i=1:N, for j=1:N, m=NN*(i-1)+1;n=NN*(j-1)+1;…
x(i,j)=X(m,n);y(i,j)=Y(m,n);z(i,j)=Z(m,n);u(i,j)=U(m,n);v(i,j)=V(m,n); end; end
% Строим лишний график ради осей полярных координат
h = polar([0 2*pi], [0 2]); delete(h);hold on;
% График скоростей и контурный, т.е. напряженностей и потенциала (рисунок 21).
h=quiver(gca,x,y,z,v,8.0);set(h,'Color','r'); contour(x,y,z);legend show;
```

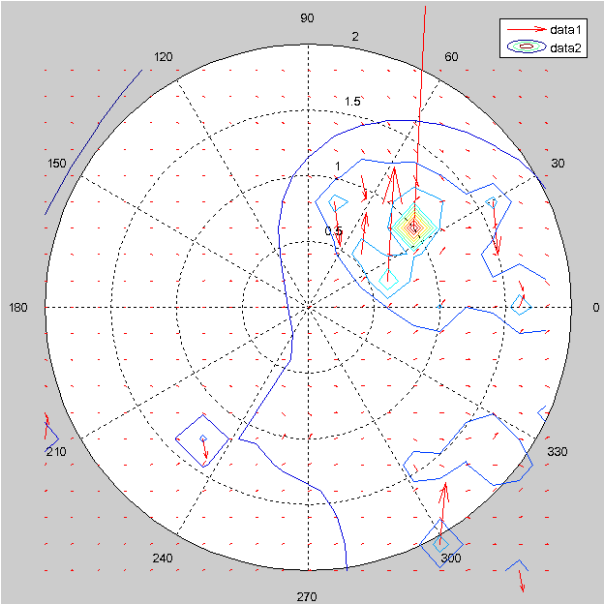

Рисунок 21 - График скоростей, напряженностей и потенциала

6.

```
    % Строим логарифм потенциала, а цвет рациональнее изменять по обратной величине (рисунок 22)
    >>surf(X,Y,log10(Z),Z.^(-1),'LineStyle','none');colormap hsv;colorbar
```

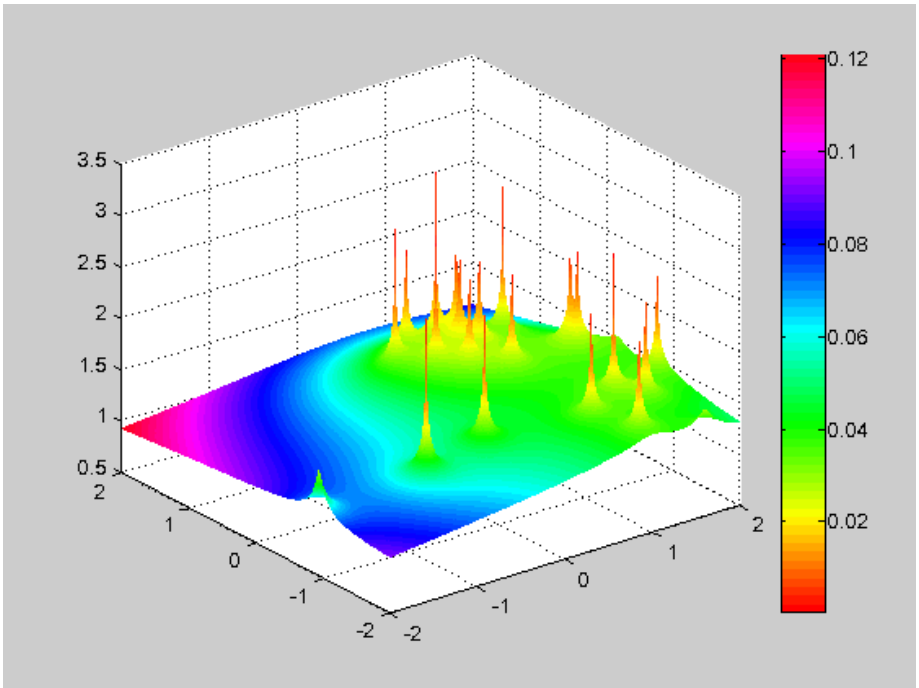

Рисунок 22 - Логарифм потенциала

7.

```
% Оформим в виде M-файла.
% Обратите внимание, что функция Electr вводится
внутри цикла
% Потенциал нормируется на минимальное значение,
и строится его логарифм
function M=MyCin(a)
tt=0:0.05:2*pi;[x,y]=meshgrid(linspace(-
2,2,501));z=[];MM=[];
a=0.5;I=ones(6,1)*linspace(1,6,6); J=I';
D1=6*rand(6)-3;D2=6*rand(6)-3; D10=D1;D20=D2;
flex1=@(t) D10.*(1+a*cos(I+t*J));
flex2=@(t) D20.*(1+a*sin(I+t*J));
Hfig=figure(1);Hax=axes();Hsur=surf(x,y,ones(len
gth(x)),'LineStyle','none');
axis manual;axis([-2 2 -2 2 0 6]);caxis([0
6]);title(''); colorbar; colormap jet;
for t=tt,
    D1=flex1(t);D2=flex2(t);
    Electr=@(x,y) sum(sum(((x-D1).^2+(y-
D2).^2).^(-0.5)));
    for k=1:length(x),
        for n=1:length(y),
            z(k,n)=Electr(x(1,k),y(n,1));
        end ;
    end;
    z=z-min(min(z))+1;title(strcat('Время:
',num2str(t),' сек'));
    set(Hsur,'ZData',log(z),'CData',log(z));
drawnow;
    MM=[MM,getframe(Hfig)];
end;
movie2avi(MM,'mym1.avi','fps',4);
M=1;
end
```

Практическое занятие №3, п.1.

1.

```
function Lab3=Lab31(A)
rrr=@(t) cos(A*t).^2;
zzz=@(t) 1+sqrt(A*rrr(t)).*sin(t);
fi=linspace(0,2*pi,201);ro=rrr(fi);Z=zzz(fi);
[X,Y,Z]=pol2cart(fi,ro,Z);figure(1);plot3(X,Y,Z)
;grid on;
figure(2);comet3(X,Y,Z);Lab3=0;
end
```

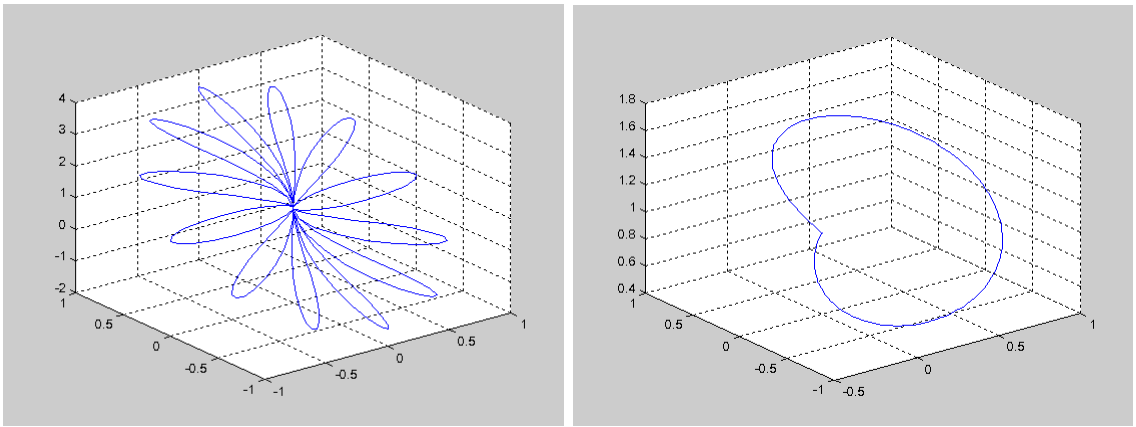

Рисунок 23 – Пример графиков

2.

```
function res=myfun1(a,s)
    if ((isvector(a))&(ischar(s)))
        switch s(1)
            case 'a'
                w=sprintf('Среднее значение:
%f',mean(a));
            case 'p'
                w=sprintf('Полином в точке:
%f',polyval(a,1.1));
             case 'g'
                w=sprintf('График');plot(a);
            otherwise
```

```
            w=sprintf('Неверные данные!');
        end;
    disp(w);res=0;
    else display('Неверный формат
данных!');res=1;
    end;
end
>> t=10*rand(10,1)-6;myfun1(t,'pbx');
Полином в точке: -50.535823
```

3.

```
    %Подготовьте  xls-файл,  столбцы  A,B,C  –
предмет, имя, оценка
    % Массив ячеек D имеет ту же структуру. Шапки
не делать
function M=myfun2(flag)
global Dbase
global Dip
    function R=reading()
        %Заполните D сами
        R=0;
    end
D={};
%Определение способа ввода - ручного или из
Ексел-файла
if (isnumeric(flag))
   reading();
else
   [D1,D2]=xlsread('1234.xls',1,'A:C');D=cat(2,
D2, num2cell(D1));clear 'D2' 'D1';
end
%D-столбцы данных, их в массив записей Dbase
Dbase=cell2struct(D,{'Subject','Name','Value'},2
);
%Данные из массива ячеек в вектора
Subject=char(D{:,1});Name=char(D{:,2});
Value=cell2mat(D(:,3)); clear('D');
```

```
[Name, Permut]=sortrows(Name);E=Dbase;
for i=1:length(E), Dbase(i)=E(Permut(i)); end;
clear 'E';
%База данных пересортирована
%Теперь вычисляем среднее, делаем структуру Dip
[UnN, NN]=unique(Name,'rows');
NN=[0;NN];dip=struct('Name',{'xxx'},'Merit',{5})
;Dip=[];
for i=1:size(UnN,1), dip.Name=UnN(i,:);db=0;
    for j=(NN(i)+1):NN(i+1),
        db=db+Dbase(j).Value;
    end;
    dip.Merit=db/(NN(i+1)-NN(i));Dip=[Dip;dip];
end;
%Усложняя:Структуру в вектора, их - на печать
DipCell=(struct2cell(Dip))',
DipCellName=char(DipCell{:,1});DipCellVal=cell2m
at(DipCell(:,2));
for i=1:size(Dip),
    disp(strcat('Студент: ',DipCellName(i,:),'
Балл:  ',num2str(DipCellVal(i)))),
end;
M=0;
end
```

Практическое занятие №3, п.2.

```
    % Перед выполнением работы составить файл
MyMath.m по шаблону п.1 Примера
```

1.

```
    % Внесено изменение в алгоритм. Итерации идут
группами, примерно по сто (рисунок 24).
function M1=M1(x,p,flag)
        MassOfPoint=@(x)    1;
        function Region=Region(x)
```

```
            if (sum(x.*x)>1) Region=0; else
Region=MassOfPoint(x); end
        end
        N=1;NN=100*2^(p);eps=p*(1e-6);
        med1=0;med=1;media=[];summa=0;
        while ((abs(med1-
med)>eps)&&(N*sqrt(eps)<1))
            for i=1:NN,
ksi=rand(1,p);summa=summa+Region(ksi); end;

media(N)=summa/NN;med=med1;med1=mean(media);
summa=0;
            %если захотим увидеть ход процесса,
каждый десятый шаг
            if (flag&&(~mod(N,10)))
['процессинг: ' num2str(med1) ...
                        '  ' num2str(med)],  end;
            N=N+1;
        end
        %вторым аргументом идет погрешность
расчета
        M1=[med1*((2*x).^p) 1/sqrt(N*NN)];
    end
    ….
case '1'
        tic;Z=[];Z1=[];
        for n=1:K, disp(n),res=M1(1,n,false);
Z=[Z;res(1),res(2)];end;
        for n=1:K, disp(n),res=M1(1,n,false);
Z1=[Z1;res(1),res(2)];end;
        VolSp=@(x)
pi^(x/2).*(1./gamma(0.5*x+1));

E=Z(:,2)'*ones(K);E1=Z1(:,2)'*ones(K);norm(Z(:,1
)-Z1(:,1)),
        x=1:K;figure(1);hold on;
        plot(x,Z(:,1),'-
r');errorbar(x,Z(:,1),E,'-r');
```

```
        plot(x,Z1(:,1),'-
g');errorbar(x,Z1(:,1),E1,'-g');
        fplot(VolSp,[1 K],'-m');
        legend('Монте-Карло1','dy','Монте-
Карло2','dy','Теория');
        grid on;legend show;
        M=Z;toc,
```

```
>> MyMath('1',12)
…
ans =

    0.3075
```

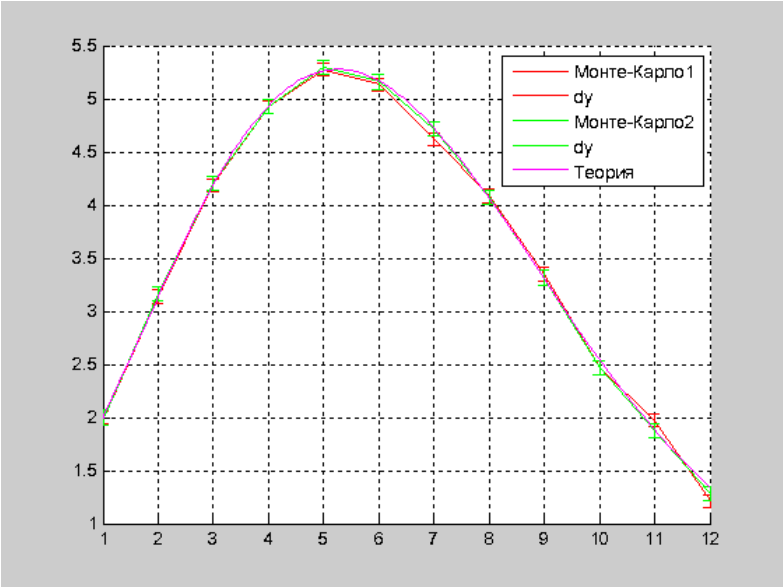

Рисунок 24 – Группы итераций

```
Elapsed time is 222.764131 seconds.

ans =

    2.0000    0.0408
    3.1900    0.0151
    4.1597    0.0060
    4.9331    0.0025
    5.2497    0.0021
    5.2142    0.0022
```

```
    4.6321    0.0023
    4.0785    0.0017
    3.3158    0.0012
    2.6167    0.0016
    1.8250    0.0013
    1.3400    0.0009
```

2.

```
function M2=M2(a,b)
        w=2*pi/(b-a);
        function Fsin=Fsin(x)
            Fsin=P5(x).*sin(w*n*x);
        end;
        function Fcos=Fcos(x)
            Fcos=P5(x).*cos(w*n*x);
        end;
        for n=0:10,

A(n+1)=quadl(@Fsin,a,b);B(n+1)=quadl(@Fcos,a,b);
        end;
        B(1)=B(1)/2;M2=abs(A+i*B)*2/(b-a);
    end
    …
P5=@(x) polyval([5 0 4 3 2 1],x);
switch
    …
case '2'
        roots([5 0 4 3 2 1]),x0=[-1 -0.5 0.5
1];[x0;P5(x0)],
        cf=abs(fft(P5(linspace(-
1,1,11))'));tf=M2(-1,1)';
        [cf tf],
        M=2;

    >> MyMath('2',12)

    ans =
```

```
   0.4421 + 0.9076i
   0.4421 - 0.9076i
  -0.1798 + 0.5845i
  -0.1798 - 0.5845i
  -0.5247


ans =

   -1.0000   -0.5000    0.5000    1.0000
   -7.0000    0.0938    3.4063   15.0000


ans =

   24.2000    2.0000
   22.9582    3.1683
   21.3636    2.6417
   18.1306    2.0586
   15.9210    1.6313
   14.9136    1.3387
   14.9136    1.1311
   15.9210    0.9776
   18.1306    0.8601
   21.3636    0.7673
   22.9582    0.6924
```

% Расхождение левой и правой колонок обусловлено несовпадением частоты

% т.е. особенностью вычисления быстрого Фурье-преобразования

3.

$U_1 = xz(xy - 2a(1+x^2))$ $\Delta U_1 = 2z(y - 6a \cdot x)$

```
%%
```

```
%Тестовая функция
```

```
    function M3=M3(x,y,z)
        A=K(1,1);h=1e-4;
        m3=((x.^2).*y).*z;
        function Dz=Dz(x,y,z)
           part=@(x,z) A*x.*(z.^2);
           partZ=cat(4,part(x,z-
h),part(x,z),part(x,z+h));
           % Матрица по 4-му измерению, ведь
x,y,z - могут быть трехмерными
           % Градиент по переменной берется по
2-му измерению
           partZ=permute(partZ,[1 4 3 2]);
           Dz=gradient(partZ,h);Dz=permute(Dz,[1
4 3 2]);

Dz=Dz(:,:,:,2);Dz=reshape(Dz,size(m3));
        end
        M3=m3-(1+x.^2).*Dz(x,y,z);
    end
%%
    %Рабочая функция
    function M4=M4(x,y,z)
        A=K(2,1);B=K(2,2);h=1e-4;
        m4=A*P5(x.*y)+erf(x-y+B.*z);
        function Dz=Dz(x,y,z)
            function part0=part0(x,y,z)

p=legendre(5,sin(x.*z+A*y));p=p(1,:,:,:);
              part0=reshape(p,size(m4));
            end;
           part=@(x,y,z) z.*part0(x,y,z);
           partZ=cat(4,part(x,y,z-
h),part(x,y,z),part(x,y,z+h));
           partZ=permute(partZ,[1 4 3 2]);
           Dz=gradient(partZ,h);Dz=permute(Dz,[1
4 3 2]);
           Dz=reshape(Dz(:,:,:,2),size(m4));
        end;
```

```
        M4=m4-B*Dz(x,y,z);
    end
%%
    %Строим графики по заданному указателю
функции MM
    function M5=M5(MM,Lim,Name,flg)
        %Ниже - процедура 3Д-графики, flg, если
режим теста
        function
Mygraph3d=Mygraph3d(X,Y,Z,U,aver,s1,flag)
            if flag
               figure('Name',['3D-графики: '
Name]);axis(Lim);grid on;box on;

xlabel('x');ylabel('y');zlabel('z');
               view(3);light('Position',[Lim(2)
Lim(4) Lim(6)]);camlight;lighting flat;
            else
               hold on;
            end;
            p =
patch(isosurface(X,Y,Z,U,aver));isonormals(X,Y,Z
,U,p);

set(p,'FaceColor',s1,'EdgeColor','none');
            Mygraph3d=gca;
        end;
        %Построение линий уровня на 2D
        function Mygraph2d=Mygraph2d(level)
            %level - либо задаем число линий,
либо вектор значений уровней
            Hfig=figure('Name',['Семейство линий
уровня для лаплассиана ' Name]);
            axis([Lim(1) Lim(2) Lim(5)
Lim(6)]);hold on;
            xlabel('x');ylabel('z');LSpec={'-
r',':b','-.g'};
```

```
            %Фиксируем y, строим линии уровня
LU(x,const,z)=const
            [X,Z]=meshgrid(x,z);
            for yy=1:3,
                n=1+ceil((yy-
1)*N/2);Ly=squeeze(L(n,:,:))';

[C,Hcg]=contour(gca,X,Z,Ly,level,LSpec{yy});
                set(Hcg,'DisplayName',['at y='
num2str(y(n))],'ShowText','off');
                if (yy==1)
set(Hcg,'ShowText','on'); end;
            end;
            grid on;legend show;
            Mygraph2d=Hfig;
        end;
        %Расчеты

N=100;x=linspace(Lim(1),Lim(2),N+1);y=linspace(L
im(3),Lim(4),N+1);

z=linspace(Lim(5),Lim(6),N+1);[X,Y,Z]=meshgrid(x
,y,z);U=MM(X,Y,Z);
        L=6*del2(U,(Lim(2)-Lim(1))/N,(Lim(4)-
Lim(3))/N,(Lim(6)-Lim(5))/N);

Um=mean(mean(mean(U)));Lm=mean(mean(mean(L)));

s{1}=['U(r)=',num2str(Um)];s{2}=['d2U(r)=',num2s
tr(Lm)];
        %Данные и лапласcиан сформированы. Далее
графика
        if flg

Mygraph3d(X,Y,Z,U,Um,'red',true);title('Тестовый
пример - функция и ее лапласcиан');
           Mygraph3d(X,Y,Z,L,Lm,'blue',false);
```

```
            legend
show;legend(s,'Location','NorthEast');
            hold off;
        else

Mygraph3d(X,Y,Z,U,Um,'green',true);title('Рабоча
я функция');
             legend
show;legend(s{1},'Location','NorthEast');

Mygraph3d(X,Y,Z,L,Lm,'magenta',true);title('Лапл
ассиан рабочей функции');
             legend
show;legend(s{2},'Location','NorthEast');
        end;
        %Построены две поверхности. Теперь
строим линии уровня лаплассиана
        Mygraph2d(3);
        M5=0;
    end

%%
    %Тестовая функция
    function M3=M3(x,y,z)
        A=K(1,1);h=1e-4;
        m3=((x.^2).*y).*z;
        function Dz=Dz(x,y,z)
            part=@(x,z) A*x.*(z.^2);
            partZ=cat(4,part(x,z-
h),part(x,z),part(x,z+h));
            % Матрица по 4-му измерению, ведь
x,y,z - могут быть трехмерными
            % Градиент по переменной берется по
2-му измерению
            partZ=permute(partZ,[1 4 3 2]);
            Dz=gradient(partZ,h);Dz=permute(Dz,[1
4 3 2]);
```

```
Dz=Dz(:,:,:,2);Dz=reshape(Dz,size(m3));
        end
        M3=m3-(1+x.^2).*Dz(x,y,z);
        %M3=A*sin(z)-x.^3-y.^4;
    end

%%
    %Рабочая функция
    function M4=M4(x,y,z)
        A=K(2,1);B=K(2,2);h=1e-4;
        m4=A*P5(x.*y)+erf(x-y+B.*z);
        function Dz=Dz(x,y,z)
            function part0=part0(x,y,z)

p=legendre(5,sin(x.*z+A*y));p=p(1,:,:,:);
               part0=reshape(p,size(m4));
            end;
           part=@(x,y,z) z.*part0(x,y,z);
           partZ=cat(4,part(x,y,z-
h),part(x,y,z),part(x,y,z+h));
           partZ=permute(partZ,[1 4 3 2]);
           Dz=gradient(partZ,h);Dz=permute(Dz,[1
4 3 2]);
           Dz=reshape(Dz(:,:,:,2),size(m4));
        end;
        M4=m4-B*Dz(x,y,z);
        %M4=P5(x.*y);
    end
%%
    %Строим графики по заданному указателю
функции MM
    function M5=M5(MM,Lim,Name,flg)
        %Ниже - процедура 3Д-графики, flg, если
режим теста
        function
Mygraph3d=Mygraph3d(X,Y,Z,U,aver,s1,flag)
            if flag
```

```
                figure('Name',['3D-графики: '
Name]);axis(Lim);grid on;box on;

xlabel('x');ylabel('y');zlabel('z');
                view(3);light('Position',[Lim(2)
Lim(4) Lim(6)]);camlight;lighting flat;
            else
                hold on;
            end;
            p =
patch(isosurface(X,Y,Z,U,aver));isonormals(X,Y,Z
,U,p);

set(p,'FaceColor',s1,'EdgeColor','none');
            Mygraph3d=gca;
        end;
        %Построение линий уровня на 2D
        function Mygraph2d=Mygraph2d(level)
            %level - либо задаем число линий,
либо вектор значений уровней
            Hfig=figure('Name',['Семейство линий
уровня для лаплассиана ' Name]);
            axis([Lim(1) Lim(2) Lim(5)
Lim(6)]);hold on;
            xlabel('x');ylabel('z');LSpec={'-
r',':b','-.g'};
            %Фиксируем y, строим линии уровня
LU(x,const,z)=const
            [X,Z]=meshgrid(x,z);
            for yy=1:3,
                n=1+ceil((yy-
1)*N/2);Ly=squeeze(L(n,:,:))';

[C,Hcg]=contour(gca,X,Z,Ly,level,LSpec{yy});
                set(Hcg,'DisplayName',['at y='
num2str(y(n))],'ShowText','off');
                if (yy==1)
set(Hcg,'ShowText','on'); end;
```

```
            end;
            grid on;legend show;
            Mygraph2d=Hfig;
        end;
        %Расчеты

N=100;x=linspace(Lim(1),Lim(2),N+1);y=linspace(L
im(3),Lim(4),N+1);

z=linspace(Lim(5),Lim(6),N+1);[X,Y,Z]=meshgrid(x
,y,z);U=MM(X,Y,Z);
        L=6*del2(U,(Lim(2)-Lim(1))/N,(Lim(4)-
Lim(3))/N,(Lim(6)-Lim(5))/N);

Um=mean(mean(mean(U)));Lm=mean(mean(mean(L)));

s{1}=['U(r)=',num2str(Um)];s{2}=['d2U(r)=',num2s
tr(Lm)];
        %Данные и лаплассиан сформированы. Далее
графика
        if flg

Mygraph3d(X,Y,Z,U,Um,'red',true);title('Тестовый
пример - функция и ее лаплассиан');
           Mygraph3d(X,Y,Z,L,Lm,'blue',false);
           legend
show;legend(s,'Location','NorthEast');
           hold off;
        else

Mygraph3d(X,Y,Z,U,Um,'green',true);title('Рабоча
я функция');
            legend
show;legend(s{1},'Location','NorthEast');

Mygraph3d(X,Y,Z,L,Lm,'magenta',true);title('Лапл
ассиан рабочей функции');
```

```
            legend
show;legend(s{2},'Location','NorthEast');
        end;
        %Построены две поверхности. Теперь
строим линии уровня лаплассиана
        Mygraph2d(3);
        M5=0;
    end
    …
case '3'
        tic;v=[1 2 5 10 0
2];M5(@M3,v,'Test',true);
        v=[-1 1 0 1 -1
0];M5(@M4,v,'Work',false);
        M=3;toc,
```

```
>> K=[1 -1;0.1 -1];
>> MyMath('3',K);
Elapsed time is 30.782295 seconds.
```

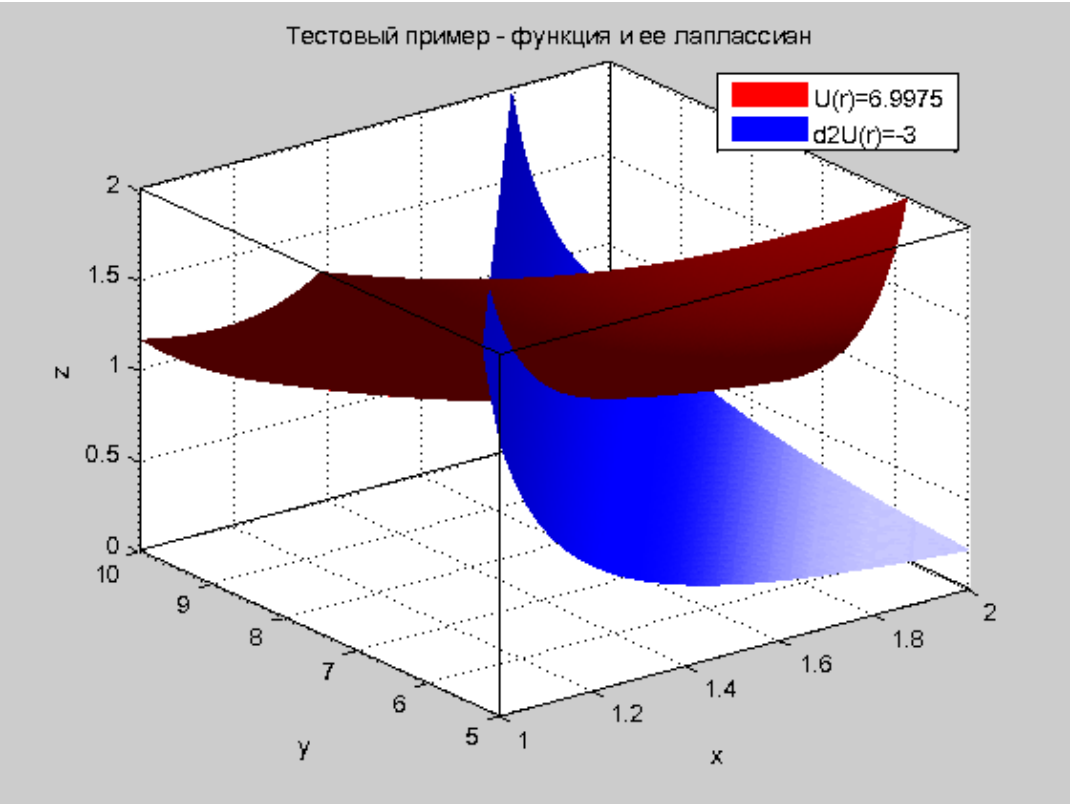

Рисунок 25 – Тестовая функция

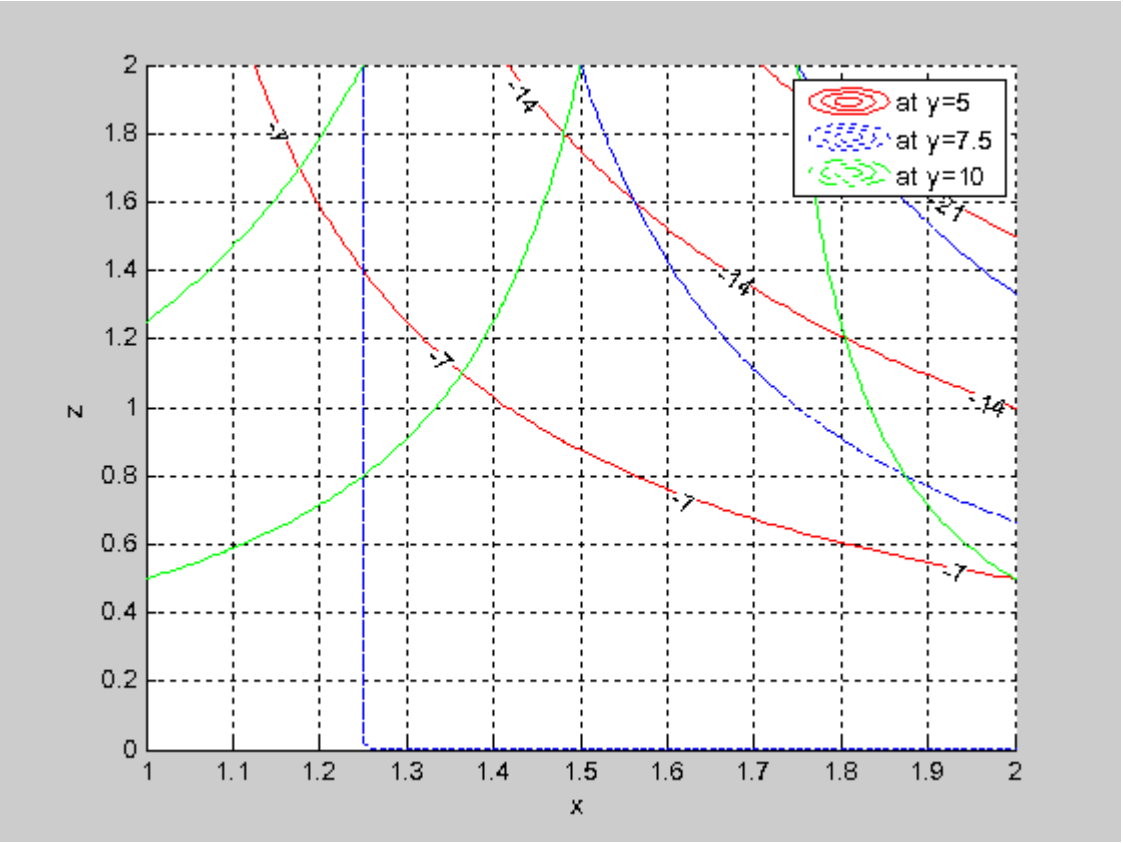

Рисунок 26 – Линии уровня лапласиана тестовой функции

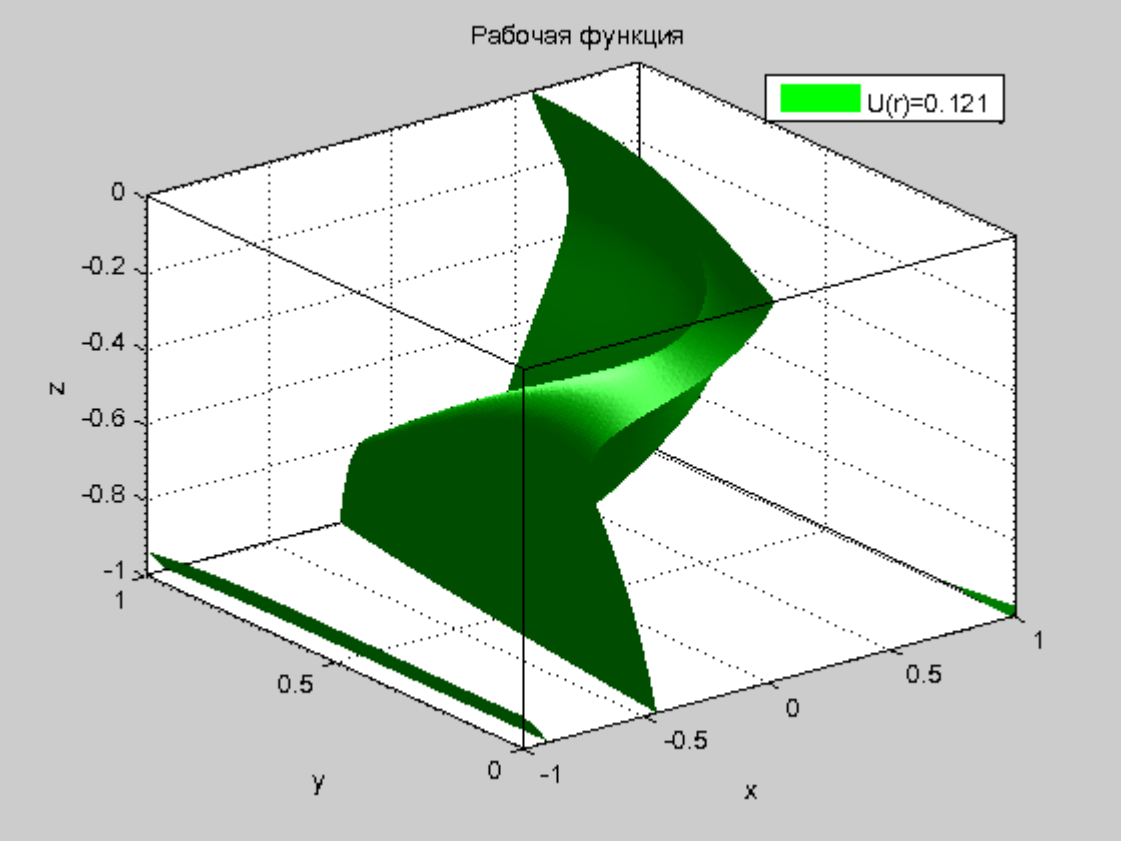

Рисунок 27 – Рабочая фукция

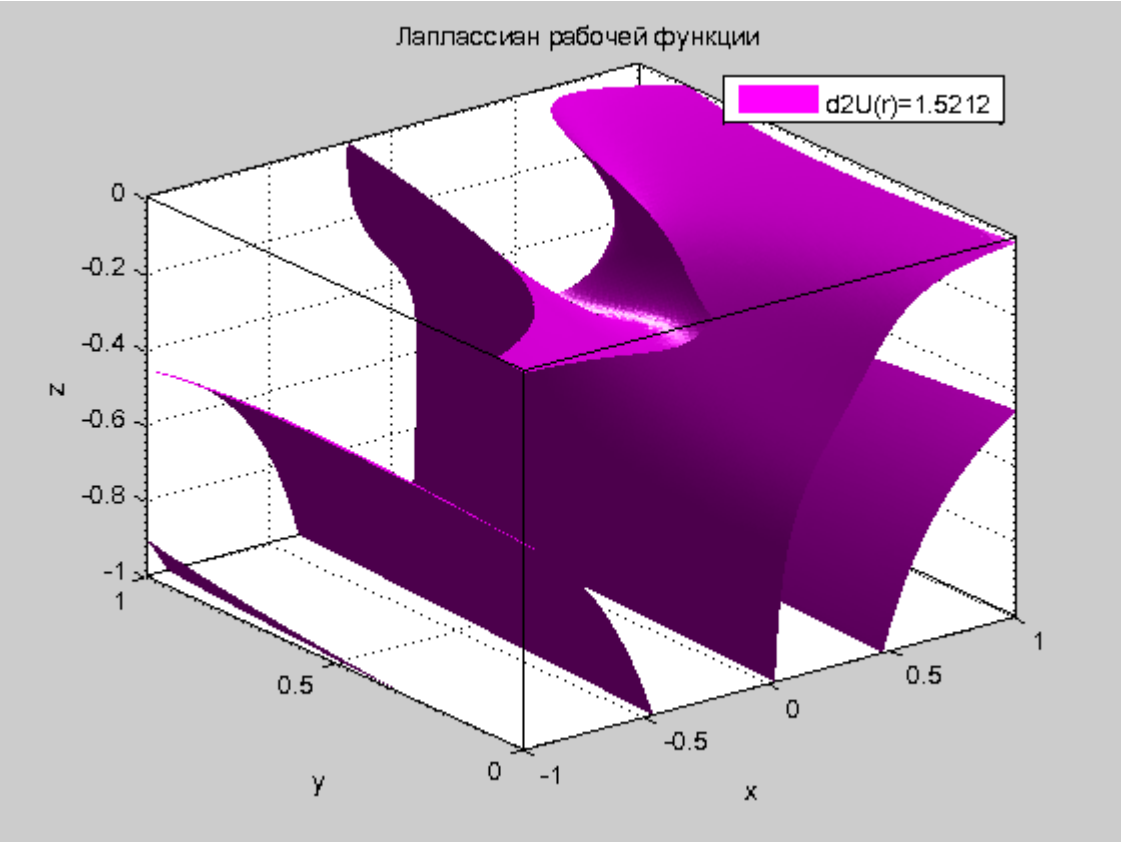

Рисунок 28 – Лапласиан рабочей функции

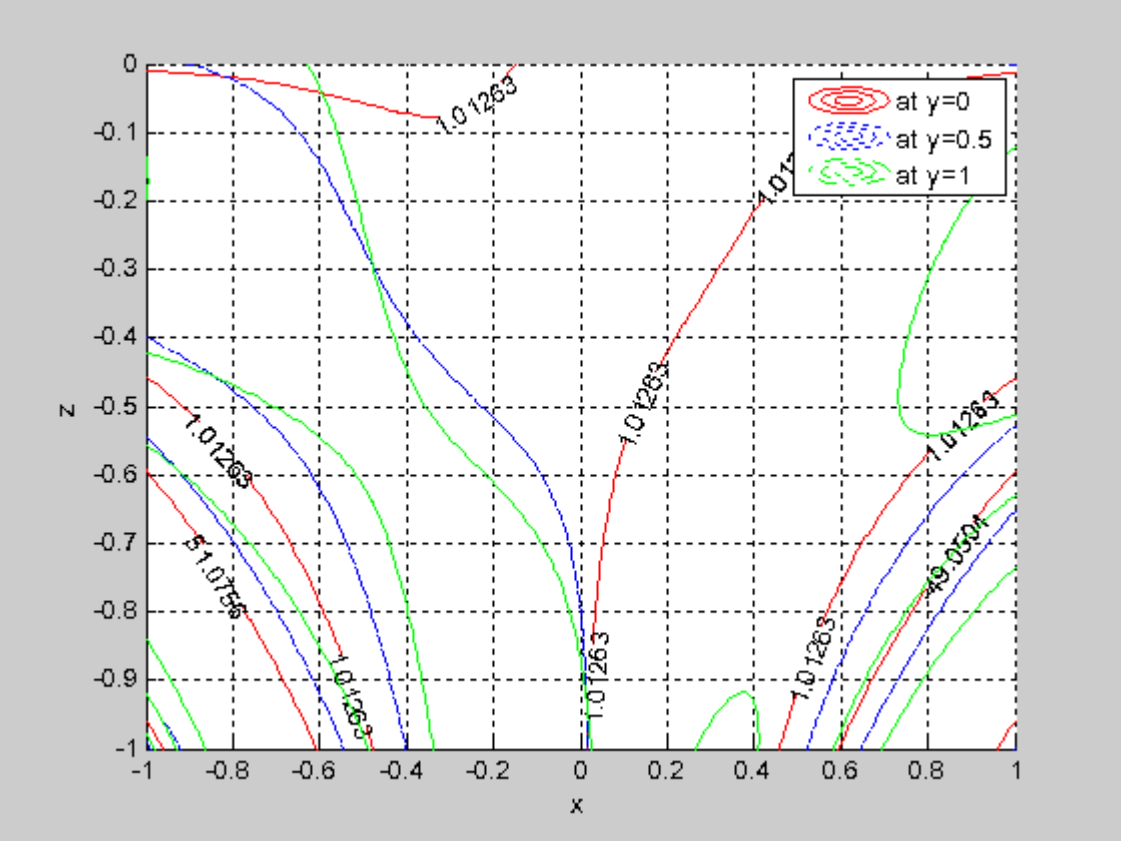

Рисунок 29 - Линии уровня лапласиана рабочей функции

4.

```
%%
    function M6=M6(MM)
        N=101;scrsz = [1 1 0.8
0.8].*get(0,'ScreenSize');s='';
        figure('Name','Simulation Plot
Window','NumberTitle','off','Position',scrsz);
        %1

subplot(2,2,1);x=linspace(K(3,1)/2,2*K(3,1),N);
        [s,error]=sprintf('U(x,y0,z0) at
y0=%3.2f, z0=%3.2f',K(3,2),K(3,3));
        if (func2str(MM)=='MyMath/M3')

[leg,error]=sprintf('a=%3.2f',K(1,1));
        else

[leg,error]=sprintf('(a,b)=(%3.2f,%3.2f)',K(2,1)
,K(2,2));
        end

U=MM(x,K(3,2),K(3,3));plot(x,U,'r','DisplayName'
,leg);
        title(s);grid
on;legend('Location','NE');legend
show;xlabel('x');ylabel('U(x)');
        %2
subplot(2,2,2);y=linspace(K(3,2)/2,2*K(3,2),N);[
X,Y]=meshgrid(x,y);U=MM(X,Y,K(3,3));
        [s,error]=sprintf('U(x,y,z0) at
z0=%3.2f',K(3,3));
contourf(x,y,U,'DisplayName',leg);
        title(s);xlabel('x');ylabel('y');grid
on;legend('Location','NE');legend show;colorbar;
        %3
subplot(2,2,3);z=linspace(K(3,3)/2,2*K(3,3),N);[
X,Y,Z]=meshgrid(x,y,z);U=MM(X,Y,Z);
```

```
d=linspace(0.5,2,4);xd=K(3,1)*d;yd=K(3,2)*d;zd=K
(3,3)*d(2:3);
        h =
slice(x,y,z,U,xd,yd,zd);set(h,'FaceColor','inter
p','EdgeColor','none');
        axis tight;title('Объемный график
сечений для U(x,y,z)');
        xlabel('x');ylabel('y');zlabel('z');grid
on;colorbar;
        %4
        PU=[];
        if (func2str(MM)=='MyMath/M3')
           [s,error]=sprintf('f(x,a)=U(x,y0,z0)
at y0=%3.2f, z0=%3.2f',K(3,2),K(3,3));

a=K(1,1);y=K(3,2);z=K(3,3);aa=linspace(a/2,2*a,N
);
           for k=aa,
K(1,1)=k;d=MM(x,y,z);PU=[PU;d]; end;K(1,1)=a;

[X,Y]=meshgrid(x,aa);subplot(2,2,4);surfc(X,Y,PU
,'LineStyle','None');

xlabel('x');ylabel('a');zlabel('U(x,a)');colorba
r;axis tight;axis on;box on;
           title(s);
        else
           [s,error]=sprintf('f(a,b)=U(x0,y0,z0)
at x0=%3.2f, y0=%3.2f,
z0=%3.2f',K(3,1),K(3,2),K(3,3));

a=K(2,1);b=K(2,2);x=K(3,1);y=K(3,2);z=K(3,3);

aa=linspace(a/2,2*a,N);bb=linspace(b/2,2*b,N);
           for m=aa, d=[];K(2,1)=m;for n=bb,
K(2,2)=n;d=[d;MM(x,y,z)]; end; PU=[PU d]; end;
           K(2,1)=a;K(2,2)=b;
```

```
[X,Y]=meshgrid(aa,bb);subplot(2,2,4);surfc(X,Y,P
U,'LineStyle','None');

xlabel('a');ylabel('b');zlabel('U(a,b)');colorba
r;axis tight;axis on;box on;
            title(s);
         end
         M6=0;
    end
    …
case '4'
         tic;M6(@M3);M6(@M4);M=4;toc,
```

```
>> K=[2 0 0;0.2 1 0;-0.5 0.5 1];
>> MyMath('4',K);
Elapsed time is 36.239721 seconds.
```

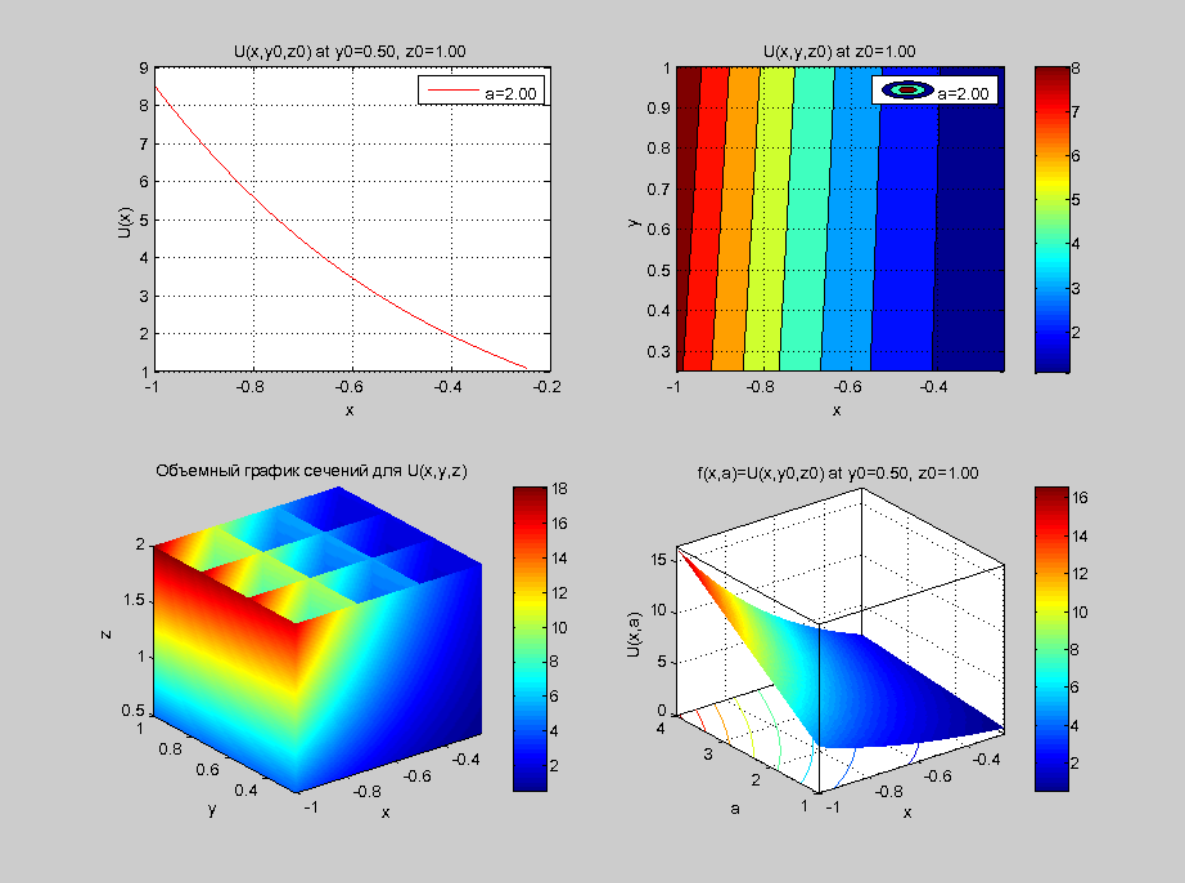

Рисунок 30 – 3-D график функции

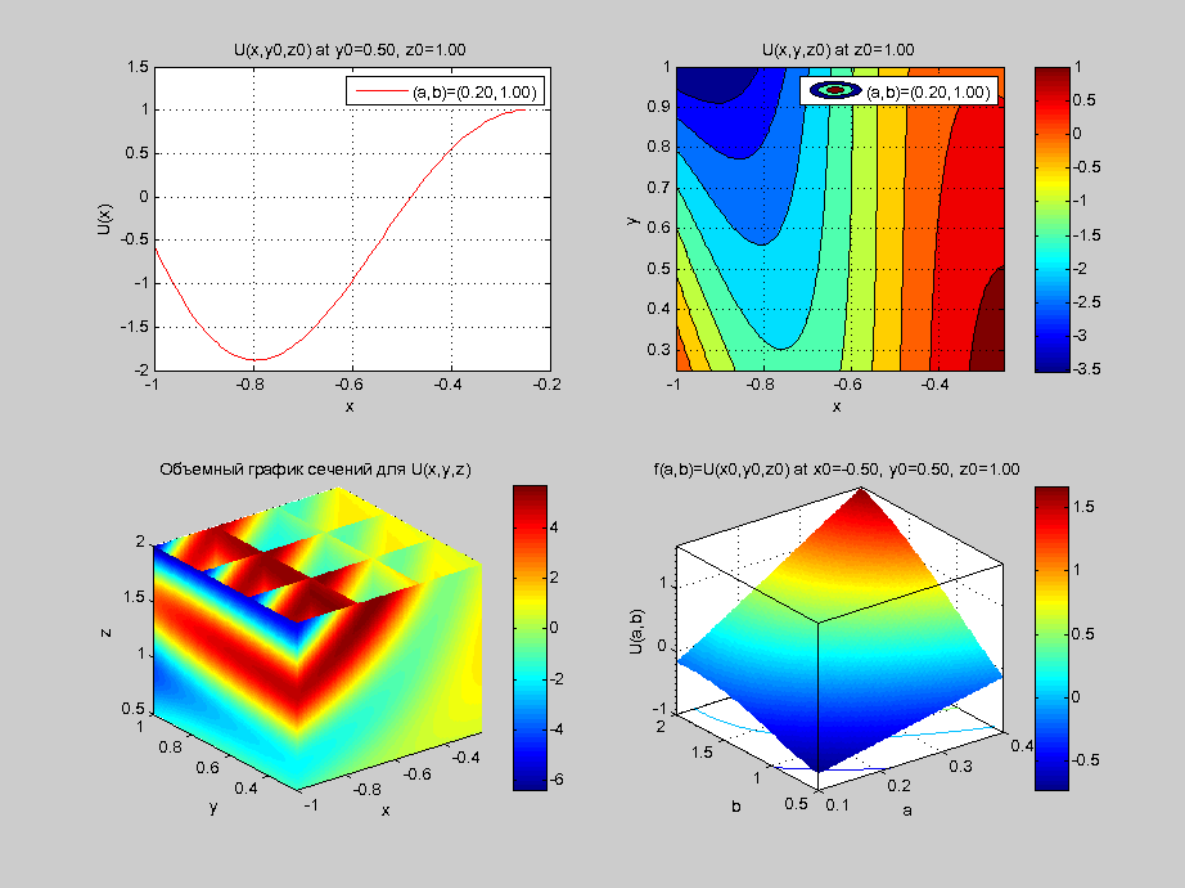

Рисунок 31 - 3-D график функции

5.

```
case '5'

N=101;x=linspace(K(2,1),K(2,2),N);U=M3(x,K(3,2),
K(3,3))-0.1*P5(x);
        plot(x,U);grid on;
        %повторить несколько раз предыдущие две
строки
        M=5;
```

```
    >> K=[2 0 0;-1 2 0;0 0.5 1]

    K =

        2.0000         0         0
       -1.0000    2.0000         0
```

```
            0    0.5000    1.0000

    >> MyMath('5',K);
    >> K(2,1)=-0.1;K(2,2)=0.1;
    >> MyMath('5',K);
    >> K(2,1)=-0.03;K(2,2)=-0.02;
    >> MyMath('5',K);
    >> K(2,1)=-0.024;K(2,2)=-0.023;
    >> K(2,1)=-0.024;K(2,2)=-0.023;
    >>
    >> MyMath('5',K);
    >> K(2,1)=-0.0238;K(2,2)=-0.0236;
    >> MyMath('5',K);
    >> x=-0.0238;
    …
U=M3(x,K(3,2),K(3,3));q=polyfit(x,U,5),q=q-
0.1*[5 0 4 3 2 1];roots(q),
        M=5;

    >> K=[2 0 0;-1 2 0;0 0.5 1]

    K =

        2.0000         0         0
       -1.0000    2.0000         0
            0    0.5000    1.0000

    >> MyMath('5',K);

    q =

       -0.0000    0.0000   -4.0000    0.5000   -
4.0000    0.0000


    ans =

      -0.0321 + 2.7775i
```

```
      -0.0321 - 2.7775i
       0.0440 + 1.0434i
       0.0440 - 1.0434i
      -0.0238
    % Нетрудно видеть, что корни и коэффициенты
полинома найдены верно
```

Практическое занятие №4

2.

Для масштабируемости окна и его элементов устанавливаем GUI-options значение свойства масштабируемости Resizable, но сверх того во всех элементах в GUIDE свойство Units как Proportional. Средствами GUIDE размечаем саму figure, оси, статический текст, текст в боксе и кнопку пуска. Средствами GUIDE размечаем само меню, затем сохраняем оба файла (lab42.fig и lab42.m). Далее редактируем последний файл самостоятельно, чтобы: во-первых, доопределить обработчики, а во-вторых, создать логотип, панель инструментов и задать их обработчики.

Листинг файла lab42.m без избыточных комментариев и процедур см. ниже (рисунок 32):

```
function varargout = lab42(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename,
...
                   'gui_Singleton',
gui_Singleton, ...
                   'gui_OpeningFcn',
@lab42_OpeningFcn, ...
                   'gui_OutputFcn',
@lab42_OutputFcn, ...
                   'gui_LayoutFcn',   [], ...
                   'gui_Callback',    []);
if nargin && ischar(varargin{1})
```

```
    gui_State.gui_Callback =
str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] =
gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

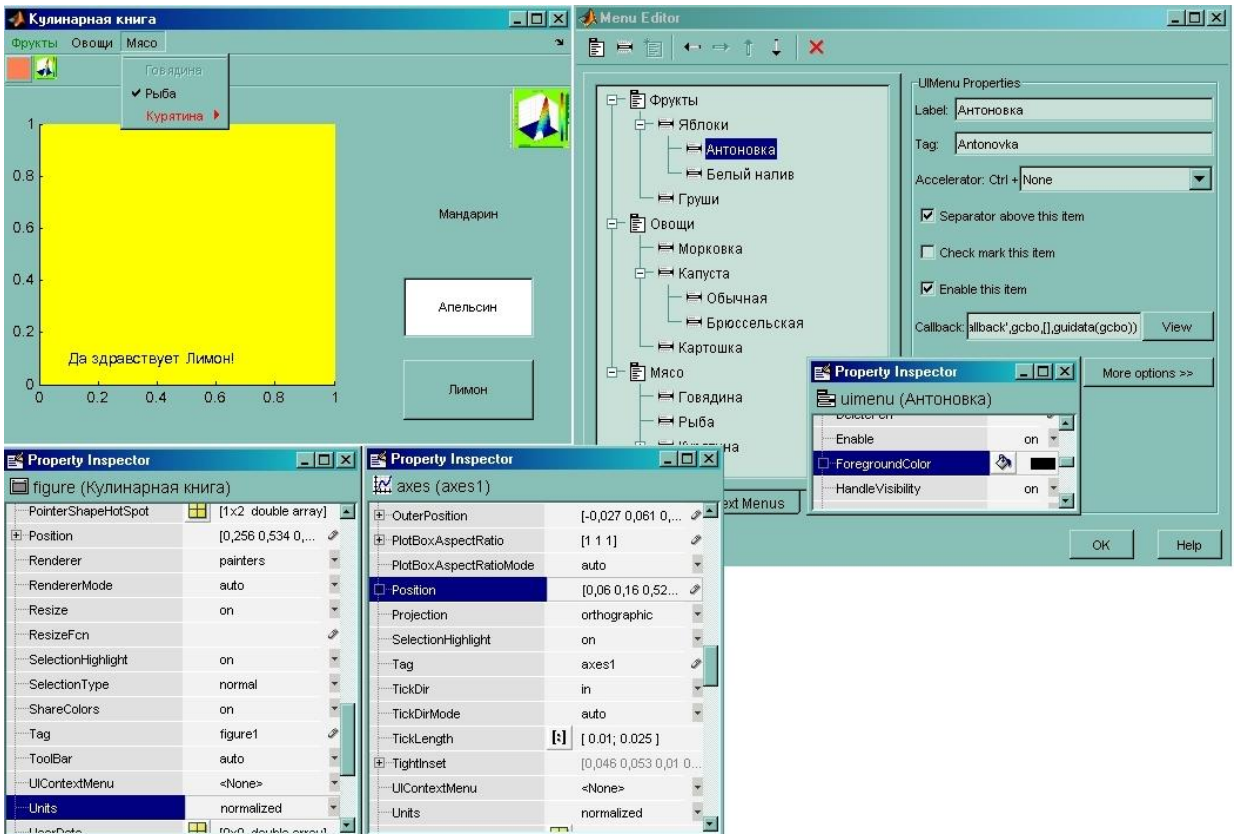

Рисунок 32 - Листинг файла lab42.m без избыточных комментариев и процедур

```
% --- Executes just before lab42 is made
visible.
function lab42_OpeningFcn(hObject, eventdata,
handles, varargin)
% Choose default command line output for lab42
handles.output = hObject;
```

```
%Добавление элементов
handles.toolbar1=uitoolbar('Tag','toolbar1');
handles.tbr_button1=uipushtool(handles.toolbar1,
'Tag','tbr_button1');
handles.tbr_button2=uipushtool(handles.toolbar1,
'Tag','tbr_button2');
handles.tbr_button3=uipushtool(handles.toolbar1,
'Tag','tbr_button3');
%1-й способ - создание картинки на лету
x=ones(20);CData=cat(3,x,x/2,x/3);set(handles.tb
r_button1,'CData',CData);
%2-й способ - чтение из bmp-файла. Функция
imresize не из ядра МАТЛАБ
x=imread('photo.bmp','bmp');CData=imresize(x,[20
20],'bilinear');
set(handles.tbr_button2,'CData',CData,'TooltipSt
ring','2-я кнопка');
%3-й способ - чтение из ico-файла с помощью
утилиты iconRead
% работа утилиты зависит от формата файла и не
всегда корректна
icopath=fullfile(matlabroot,'help','techdoc','cr
eating_guis','examples');
path(icopath,path);CData=iconRead('my.ico');rmpa
th(icopath);
set(handles.tbr_button3,'CData',CData,'TooltipSt
ring','3-я кнопка');
%Поместим логотип в верхнем правом углу
mn=[50
50];x=imread('photo.bmp','bmp');CData=imresize(x
,mn,'bilinear');
h=handles.output;set(h,'Units','pixels');pos=get
(h,'Position');
pos=[pos(3)-mn(1) pos(4)-mn(2) mn];
haxes_im=axes('Tag','axes_im','Units','pixels','
Position',pos,'Parent',h);
set(h,'Units','normalized');set(haxes_im,'Units'
,'normalized');
```

```
handles.haxes_im=haxes_im;handles.im=image(CData
,'Parent',haxes_im,'Tag','im');
set(haxes_im,'Visible','off');
%Установка вызова обработчика для первой кнопки
панели инструментов
set(handles.tbr_button1,'ClickedCallback',{@tbr1
_Callback, handles,'Апельсины'});
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to
the command line.
function varargout = lab42_OutputFcn(hObject,
eventdata, handles)
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.

function pushbutton1_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    structure with handles and user
data (see GUIDATA)
axes(handles.axes1);cla
reset;set(gca,'Color','Yellow');
text(0.1,0.1,'Да здравствует Лимон!');

% ----------------------------------------------
--
function Kapusta_Callback(hObject, eventdata,
handles)
% hObject    handle to Kapusta (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    structure with handles and user
data (see GUIDATA)
beep;
```

```
function tbr1_Callback(hObject, eventdata,
handles, label)
axes(handles.axes1);cla reset;set(gca,'Color',[1
1/2 1/3]);
text(0.1,0.1,label);title(label);set(hObject,'To
olTipString','Apelsin');
```