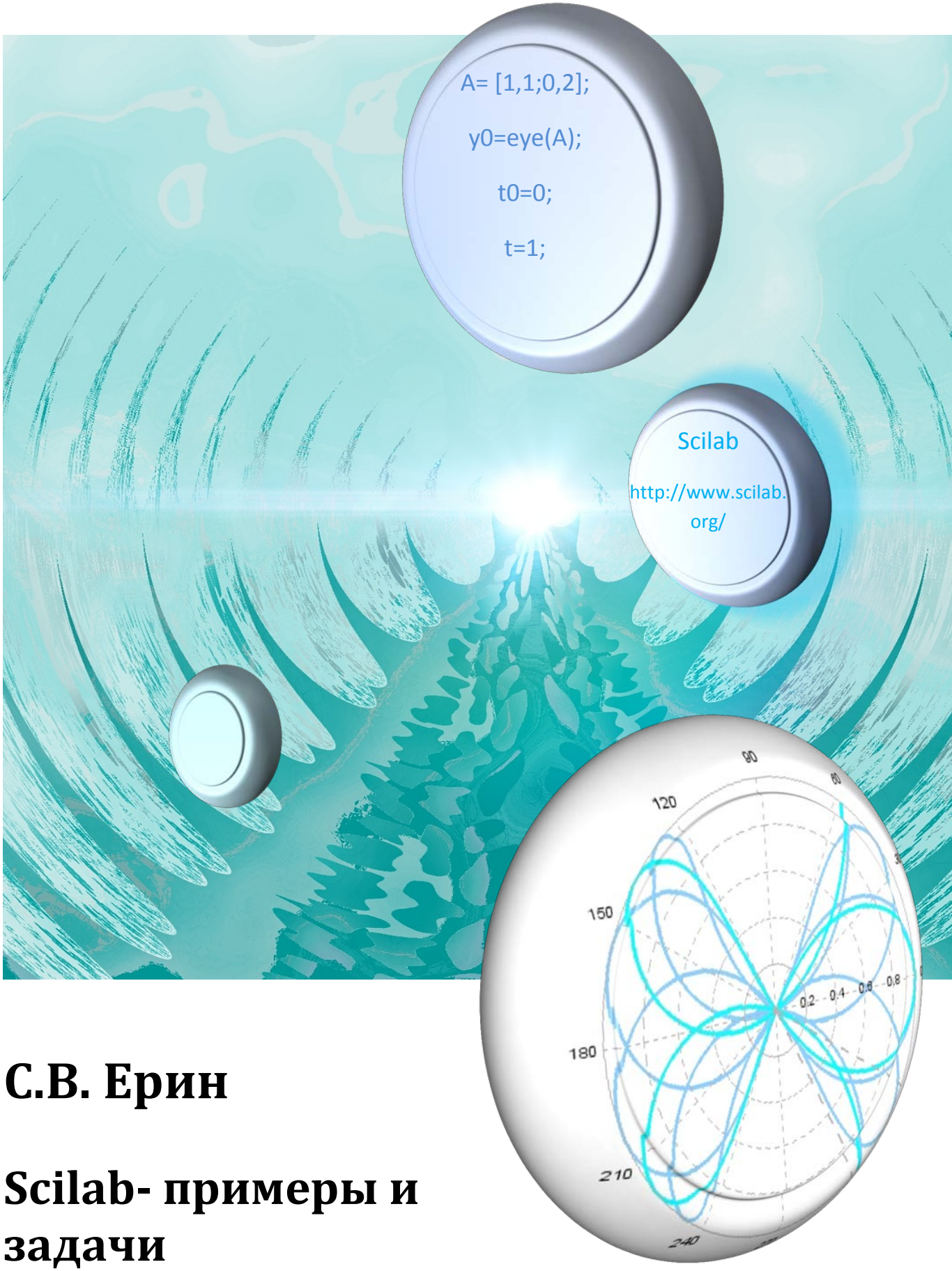


С.В. Ерин

Scilab- примеры и задачи

Москва 2017

С.В. Ерин

Scilab- примеры и задачи

Сборник примеров и задач

Серия «Свободные математические пакеты»

Москва
2017

УДК 002.6(075.8)
ББК 32.811я73
Е71

Рецензент: Битюков С.И, доктор ф-м.наук, в.н.с.

Ерин С. В.
Е71 Scilab- примеры и задачи: практическое пособие/С.В. Ерин — М.: Лаборатория «Знания будущего», 2017. — 154 с.: ил.

В учебном пособии рассмотрены примеры и задачи, которые позволят освоить возможности применения свободно распространяемого математического пакета Scilab.

Примеры решения математических задач сопровождаются текстами сценариев на языке Scilab и комментариями, поясняющими ход решения. По разделам пособия даны задачи для самостоятельного решения.

Учебное пособие предназначено для подготовки студентов и аспирантов технических университетов, а также в помощь инженерным и научным работникам, выполняющим значительный объем вычислений и аналитических расчетов.

УДК 002.6(075.8)
ББК 32.811я73
Е71

ОГЛАВЛЕНИЕ

Предисловие

В настоящее время стремительно развиваются программные системы для проведения математических расчётов. Можно отметить наиболее известные, такие как: Matlab, Mathematica, Mathcad, Maxima, Octave, Scilab. У каждой из перечисленных программ есть свои плюсы и минусы, но не будем их обсуждать, а просто рассмотрим такую программу как Scilab.

Открытая система Scilab является мощным математическим пакетом, используемым для решения различных математических, инженерных и экономических задач. Основным преимуществом этой системы является то, что пользователю не обязательно быть программистом, чтобы решать вычислительные задачи в различных областях науки и техники. Scilab работает в режиме интерпретатора, а также позволяет обрабатывать программы, написанные на встроенном языке. Система Scilab очень удобна при проведении инженерных расчетов.

Стоит отметить, что алгоритмы, использованные в Scilab, оптимизированы для вычислений, проводимых с матрицами и комплексными числами (аналогично Matlab, Octave). В Scilab реализовано большое количество встроенных функций, позволяющих пользователю выполнять сложные численные расчёты, моделировать поведение физических процессов и технических систем. Развитая графическая система встроенных команд позволяет наглядно представлять полученные данные, и проводить графический анализ результатов. Scilab является открытой и расширяемой системой, что позволяет использовать не только встроенные функции, но и добавлять собственные (пользовательские), причём они могут быть написаны на таких языках, как Фортран, Си. Для написания программ (функций) в Scilab имеется встроенный редактор SciNotes. Кроме того, вычисления можно проводить в командном режиме (используя Scilab в качестве "калькулятора"), получая результат сразу после ввода команды. Аналогично Matlab, для решения специальных задач в Scilab разработаны пакеты с дополнительными функциями ATOMS (AuTomatic mOdules Management for Scilab). Эти дополнительные пакеты позволяют существенно расширить возможности пакета Scilab.

В заключение необходимо сказать, что пакет Scilab всё шире используется в процессе обучения студентов, и возникает настоятельная необходимость в учебно-методической литературе на русском языке, обеспечивающей процесс изучения и использования данного пакета. Стоит отметить наиболее популярную существующую литературу по

данному вопросу [1-5]. В ней рассматриваются возможности Scilab и приведены многочисленные иллюстрирующие примеры.

Данная книга является дополнительной к уже выпущенной книге авторов С.В. Ерина и Ю.Л. Николаева «Автоматизация инженерных расчётов с использованием пакета Scilab». В книге подробно разобраны решения многих представленных примеров, собраны примеры и задачи, которые можно использовать при изучении системы Scilab.

Краткое введение в Scilab 1

Введение.

В этой главе, кратко рассмотрим возможности и команды Scilab, которые будут использоваться на протяжении всей книги. Для полного изучения возможностей пакета лучше обратиться к уже опубликованной литературе [1-5] и встроенной справочной системе самого пакета. Существуют версии Scilab для различных операционных систем. В нашем случае будет использоваться версия Scilab 5.5.2 для ОС Windows. Вышла более свежая версия Scilab 6 Beta 2, но пока это нестабильная версия и находится в процессе бета-тестирования.

Запустив Scilab, на экране монитора увидим изображение, представленное на рисунке 1.

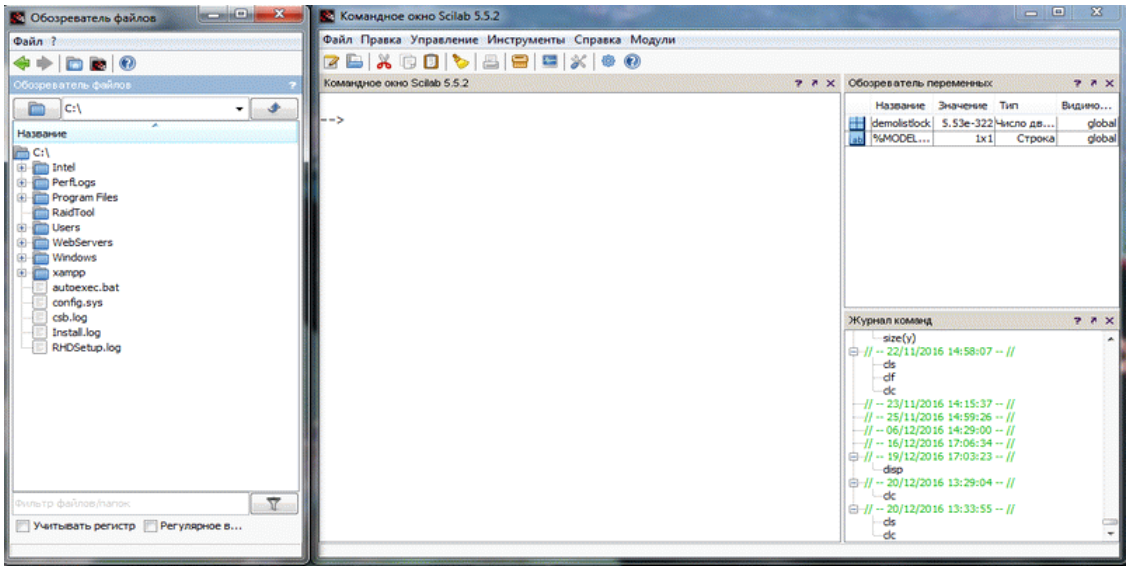


Рис.1 Окна Scilab

Назначение окон ясно из их названий (впрочем, можно посмотреть справочную систему для более углублённого понимания). Все вычисления проводятся в командном окне. Как отмечалось выше, в пакете Scilab возможна работа в интерактивном и в пакетном режиме.

Вначале рассмотрим работу в интерактивном режиме, но перед этим рассмотрим объекты, с которыми работает программа. Внутренняя структура этих объектов заранее задана и скрыта от глаз пользователей.

Существуют следующие типы объектов: *переменная, матрицы и векторы, списки, полиномы, rational (производный от полинома, который образуется делением одного*

полинома на другой), математическая функция, особый тип данных - struct (структура) и другой тип структурных данных - cell (ячейка).

В своей основе перечисленные объекты являются массивами, в которых сохранены определенные данные. Как работать с этими объектами рассмотрим в дальнейшем.

1.1 Интерактивный режим

Рассмотрим работу в Scilab в интерактивном режиме. В этом режиме системой можно пользоваться, как мощным калькулятором. Причём, пользователь в этом режиме может использовать весь функционал Scilab. Работа в интерактивном режиме называется также работой в режиме прямых вычислений или командном режиме, и носит диалоговый характер. Чтобы получить ответ пользователь набирает выражение в командном окне, редактирует его и после нажатия клавиши ENTER получает ответ. Например, команда

--> v= [5 6 9 4];

задает четырехэлементный вектор v (возможна также запись через запятую v=[5,6,9,4]). Знак **;** (точка с запятой) в конце выражения блокирует немедленный вывод результата вычислений. Далее вычислим cos(v):

-->cos(v)

ans =

0.2836622 0.9601703 - 0.9111303 - 0.6536436

Система вычислила функцию cos от векторного аргумента. Функция cos является встроенной функцией (список см. Help). Встроенные функции записываются строчными буквами, их аргументы указываются в круглых скобках. Если не указана переменная для присвоения значения результата, то Scilab назначает переменную с именем **ans.**

Выражения записываются согласно правилам языка Scilab. Если выражение оказывается длинным, то его часть можно перенести на новую строку с помощью знака **..** (многоточие). Текстовые комментарии вводятся с помощью знака // перед текстом строки. Необходимо подчеркнуть, что все переменные в Scilab рассматриваются как матрицы, вектора или массивы. Для матриц и векторов поэлементные операции (они обозначаются добавлением знака **.** (точка) перед соответствующей операцией, например, **.*** и **./**) возможны лишь для операндов с одинаковыми размерами, например:

```
--> a1=[3 6 9 ];
--> a2=[1,2,3];
--> a1/a2
ans=3
```

```
--> a1.*a2
ans=3 12 27
--> a1./a2
ans=3 3 3
```

Операция обычного умножения (без знака .) выполняется по правилам линейной алгебры. Используются также операции \ и .\ (левое матричное деление), ^ и .^ (возведение в степень и поэлементное возведение в степень).

Для формирования упорядоченных последовательностей применяется операция : (двоеточие) (начальное значение : шаг : конечное значение, по умолчанию шаг=1),

```
--> 1:5
ans =
    1 2 3 4 5
```

Кроме арифметических, в Scilab существуют операции отношения **(= =, ~=, <, >, <= , >=),** логические операции (**&**, **and**, **or**, **not**, **~**,), а также операции = (присваивание), **’** (транспонирование), **[,]** - (горизонтальное соединение), **[;]** -(вертикальное соединение).

1.2 Переменные и функции системы Scilab

Рассмотрим, как работать с таким объектом Scilab, как переменная. Переменная в Scilab — это именованный массив с одним полем, хранящим данные определённого типа. Перечислим эти типы данных:

```
Числа:
   Целые
   Вещественные
   Комплексные
Строки
Логические переменные
```

Чтобы создать переменную, достаточно ввести ее имя и присвоить ей какое-либо начальное значение. Чтобы посмотреть текущее значение переменной, достаточно ввести имя переменной в командном окне, либо воспользоваться редактором переменных. Чтобы посмотреть тип переменной в Scilab нужно воспользоваться командой **type()** или **typeof()** Полное описание по этим командам (впрочем, как и по другим) можно получить с помощью команд: **help type**, **help typeof**.

Существуют правила, по которым присваиваются имена переменных и вообще любых объектов среды:

В имени переменной можно использовать буквы латинского алфавита (верхнего и нижнего регистра) и цифры;

Имя переменной может начинаться с символов '!', '$', '%', '_', '#', '?', но не с цифр;

Имя переменной, записанное строчными и прописными буквами, задаёт разные переменные, например: переменные temp,TEMP ,Temp разные;

Нельзя называть переменную именем переменной с зарезервированными словами, такими как имена объявленных функций, констант и др.

В Scilab имеются системные переменные, которые начинаются с символа %:

%i – мнимая единица (√-1);

%pi – число π (3.141592653589793);

%e – число e=2.7182818;

%inf – машинный символ бесконечности (∞);

%NaN – неопределенный результат (0/0, ∞/∞, 1^{∞} и т.п.);

%eps – условный ноль %eps=2.220E-16.

Все перечисленные переменные можно использовать в математических выражениях

Итак, создадим в командном окне несколько переменных:

```
--> x1=2, y2=0.33, z3=-6.05e-2, X4=0.5-%i*7
```

В результате мы получили 4 переменных:

```
-->x1=2, y2=0.33, z3=-6.05e-2, X4=0.5-%i*7
 x1  =

   2.
 y2  =

   0.33
 z3  =

  - 0.0605
 X4  =

   0.5 - 7.i
```

три из которых хранят вещественные числа и одна — комплексное число.

Для создания комплексных чисел существует также функция **complex(a,b)**. Аргументы **a** и **b** - действительная и мнимая части соответственно. Способ задания с использованием ввода знаков **+** и **%i** не всегда удобен.

Для работы с комплексными числами также существуют функции:

conj(X) — возвращает комплексное число сопряженное числу X;

real(X) — возвращает действительную часть комплексного числа X;

imag(X) — возвращает мнимую часть комплексного числа X;

isreal(X) — возвращает логическое %T, если передаваемый аргумент является комплексным числом, %F в противном случае.

imult(X) — умножает мнимую единицу на аргумент. Согласно справочной информации, данную функцию рекомендуется использовать, когда приходится иметь дело с константами типа %nan и %inf, появляющихся в комплексном числе.

Что касается целых чисел, то необходимо обратить внимание, что эти числа в Scilab создаются только через специальные функции. В любом другом случае числовому значению будет присваиваться вещественный тип данных.

Под хранение в памяти целого числа и его знака используют разное число битов: 8, 16, 32, 64. Диапазон целых чисел зависит от количества используемых битов.

В таблице ниже приведены функции для создания целых чисел и диапазоны возможных значений. Во всех случаях функция возвращает целое число, указанное в качестве аргумента, с определенным способом его хранения в памяти.

Таблица 1

Имя функции	**Тип целого числа**	**Диапазон**	**Код представления**
int8()	знаковое 8-битное число	$[-2^7,2^7-1]$	1
uint8()	беззнаковое 8-битное число	$[0,2^8-1]$	11
int16()	знаковое 16-битное число	$[-2^{15},2^{15}-1]$	2
uint16()	беззнаковое 16-битное число	$[0,2^{16}-1]$	12
int32()	знаковое 32-битное число	$[-2^{31},2^{31}-1]$	4
uint32()	беззнаковое 32-битное число	$[0,2^{32}-1]$	14

Возьмём функцию uint8() и поэкспериментируем с ней. Создадим целочисленную переменную s:

```
--> s=uint8(10)
s  =
  10
```

В записи s =10 исчезла точка, разделяющая целую и дробную части.

Объявим целочисленную переменную s=257:

```
s=uint8(257)
 s  =
     1
```

Вместо 257 мы получили 1. В данном случае, значение переменной вышло за диапазон возможных значений. Такая ситуация называется переполнением.

Создадим символьную (строковую) переменную следующей командой:

```
-->z='Hello World'
 z  =
 Hello World
```

Теперь создадим логическую переменную. Надо отметить, что логические переменные хранят в себе одну из двух предопределенных констант: **%T** (от англ. True — истина) и **%F** (от англ. False — ложь). Для логических типов данных применяются логические операции. Но в Scilab возможно смешение типов данных, поэтому в логических операциях можно использовать числовой тип данных. В этом случае, любое ненулевое значение будет иметь значение **%T**, а нулевое значение — **%F**.

Зададим две логические переменные, две числовые и посмотрим значение после логических операций: И (**&**) и ИЛИ(|):

```
--> x=%T; y=%F;
--> 4 & x, 1 | y
ans  =
 T
ans  =
 T
```

Приведём примеры работы Scilab в качестве калькулятора:

-->3+3 ans = 6. -->2/3 ans =	-->2^4 ans = 16. -->a = 22 a =	-->pi = atan(1.0)*4 pi = 3.1415927 -->sin(pi/4) ans =

.6666667	22.	0.7071068 -->exp(0.1) ans = 1.1051709

Необходимо отметить, что в Scilab для работы с любыми объектами в среде существуют функции, которые позволяют пользователю получить ответы на вопросы: " Какое число переменных находится в памяти ?", "Какие имена свободны, а какие заняты?", "Как очистить память от ненужных объектов?" и другие.

Значения переменных хранятся в рабочей области, и для её очистки используется команда **clear** в разных формах: **clear X** (стирается одна переменная **X**), clear **a b c** (несколько переменных), **clear all** (все переменные). Команда **who** без аргументов возвращает несортированный список объявленных переменных (локальных и глобальных) с текущими значениями параметров памяти. Более подробную информацию выдает команда **whos**, которая выводит список переменных с указанием типа и размера, если вызывается без аргументов. Если вызывается с аргументами, то выводит список переменных, хранящих определенный тип данных (если указан параметр **-type**) или/и обладающих определенным именем или фрагментом этого имени (если указан параметр **-name**). Команда **who_user** выводит только переменные, созданные пользователем во время текущей сессии.

В Scilab есть предопределенные математические функции (так называемые компилированные функции). К ним относятся тригонометрические, экспоненциальные и другие. Пользователь системы также имеет возможность определить собственную функцию. Существует два способа определить эту функцию: через написание некой программы-функции или объявить через специальную функцию **deff( ).**

Выбор способа определения функции остаётся за пользователем.

Рассмотрим определение функции через специальную функцию deff( ).

Общий синтаксис имеет вид:

```
deff('[Y1,Y2...]=Fname(X1,X2,...)',['Y1=выражение1';'Y2=выражение2;...'])
// [Y1,Y2...] — вектор возвращаемых переменных (имена назначает пользователь)
// Fname — имя функции, назначает пользователь
// (X1,X2,...) — список из аргументов функции
// 'Y1=выражение_1;Y2=выражение_2;...' — для каждой выходной переменной должно быть определено выражение,
// которое может зависеть от аргументов или не зависеть.
```

Пример задания такой функции приведён ниже:

```
-->deff('y=f(x)','y=x.^3+cos(x)')
-->x=1:5;
-->f(x)
 ans  =
    1.5403023    7.5838532    26.010008    63.346356    125.28366
```

Усложним функцию:

```
-->deff('[x,y]=f(z,u)',['x=z-u';'y=u.^2+sin(z)']);
-->f(5,6)
 ans  =
   -1.
->[a1,a2]=f(5,6)
 a2  =


   35.041076
 a1  =


 - 1.
```

В случае попытки вычислить значение функции в виде f(5,6), ans не переопределилась системой в вектор, и был записан только результат расчета переменной x. Для получения полного ответа необходимо явно определить вектор, куда будет записан результат. Необходимо отметить, что для вызова функции все аргументы обязательны.

В случае сложной функции целесообразнее использовать программируемую функцию в виде конструкции **function...endfunction**. Но об этом поговорим попозже.

1.3 Работа с векторами и матрицами

Для задания вектора из n элементов, следует перечислить их значения в квадратных скобках через пробел или через запятую, например:

```
->a=[1 -3 0 -%i]
 a  =
  1.  - 3.    0  - i
```

Для задания вектора, элементы которого представляют собой числовую последовательность в виде арифметической прогрессии, можно использовать следующую конструкцию:

<начальное значение>:<шаг>:<конечное значение>

```
--> -2:2:8
ans  =
 - 2.  0.  2.    4.    6.    8.
-->1:5
 ans  =
   1.    2.    3.    4.    5.
```

В первом случае создан вектор с начальным значением -2, конечным значением 8 и шагом между элементами 2. Если шаг не указан, то его значение по умолчанию принимается равным 1, и его начальное значение должно быть меньше конечного, в противном случае Scilab создаст пустой вектор.

Из линейной алгебры известно, что вектор может быть задан как вектор-столбец или— вектор-строка.

Посмотрим как можно задать вектор-столбец:

```
-->[2; 2; 7]
 ans  =
   2.
   2.
   7.
-->[2 2 7]'
 ans  =
   2.
   2.
   7.
```

В первом случае каждый элемент вектора разделяется точкой с запятой. Во-втором - вектор-столбец получен операцией транспонирования (оператор в виде апострофа).

Посмотрим, как обратится к любому элементу внутри вектора. Для этого необходимо записать имя вектора, указав в круглых скобках индекс элемента.

```
-->x=1:4
 x =
   1.    2.    3.    4.
-->x(3) // обращение к элементу 3  вектора x
 ans  =
```

```
   3.
-->x=x';
-->x(3) //операция транспонирования не влияет на индексацию
 ans  =
   3.
-->x($)
 ans  =
   4.
```

Последний элемент вектора можно узнать также, используя служебный символ '$'. Рассмотрим, как можно удалить элемент вектора. Для этого используется конструкция в виде пустой матрицы '**[ ]**':

```
-->x=1:4
 x  =
   1.    2.    3.    4.
-->x(2)=[] // удаление элемента 2  вектора x
 x  =
   1.    3.    4.
```

Теперь рассмотрим операции, которые можно проводить с матрицами. Матрица является двухмерным массивом однотипных элементов, и её можно рассматривать как несколько векторов-строк, записанных столбцом.

В Scilab создать матрицу можно несколькими способами:

```
-->A=[3 2; 5 4]
 A  =
   3.    2.
   5.    4.
--> [2:5;4:-1:1]
ans =
   2.  3.  4.  5.
   4.  3.  2.  1.
--> [(3:5)' (3:-1:1)']
ans =
   3.  3.
   4.  2.
   5.  1.
```

Первый способ - создание векторов-строк, которые отделяются точкой с запятой. Второй случай - матрицу можно собрать из векторов. И третий способ - транспонирование векторов, и присоединение их столбцами друг к другу.

Каждый элемент матрицы имеет два индекса: первый указывает число строк, а второй — столбцов. Элементы векторов и матриц могут задаваться не только числами, но и в виде арифметических выражений, содержащих доступные системе функции. Элементы могут быть также комплексными. Для указания отдельного элемента используется выражение вида A(k) или A(m,n):

```
-->A(1,2)   // 1-ая строка, 2-ий столбец
 ans  =
        2
>> A(4)  //A(k) дает доступ к элементам матрицы, развернутым в один столбец
ans=
      4
>> A(A> 2)  //Индекс  k  может быть  логическим  выражением:
ans  =
   3.
   5.
   4.
 ->A([1,2],2)  // A([1,2],2) обозначает первый и второй элементы второго столбца
 ans  =


   2.
   4.
```

В качестве индекса может быть вектор. Для указания на m-ю строку или n-й столбец матрицы используются выражения вида A(m,:) или A(:,n). Запись A(k:m;n) обозначает вектор-столбец, сформированный из нескольких элементов n-го столбца матрицы A (с k по m-й).

```
-->A(:,$) // $ -обозначает последний элемент
 ans  =
        2.
        4.
```

В пакете Scilab имеется ряд функций для задания особых векторов и матриц, например: **zeros(m,...,q)**- многомерный массив из нулей, **ones(m,...,q)**-массив из единиц, **rand(m,...,q)**

-массив из случайных чисел, **eye(n)** - единичная матрица, **linspace(a,b,n)**- вектор-строка из n чисел, равномерно расположенных на отрезке [a,b], (по умолчанию n=100), **logspace(a,b,n)** - узлы логарифмической сетки по десятичному логарифму (по умолчанию n=50).

Вектора и матрицы также могут быть элементами матриц, что позволяет формировать большие матрицы, объединяя малые. Для удаления строк или столбцов удобно использовать пустые квадратные скобки **[ ]**:

```
-->A(1,:)=[ ]
 A  =
     5.    4.
```

Основную информацию о векторах и матрицах можно получить с помощью функций **size(A)** - вектор-строка с размерами матрицы A, **length(B)**- число компонентов вектора B, **ndims(A)** - размерность массива A, **disp(A)** - массив A без его имени, **isempty(A)** - T, если A=[], **isequal(X,Y)**-T, если X=Y.

Более подробно с операциями и функциями над матрицами можно познакомиться по справочной системе Scilab или в соответствующей литературе.

1.4 Списки

Перейдём к следующему типу объектов - спискам.

Дадим определение, что такое список в Scilab — это массив (или структура), состоящий из элементов разных типов данных.

Этот объект можно создать вручную или с помощью следующих функций:

list() — создает список, поля которого могут содержать произвольный тип данных;
tlist() — создает типизованный список;
mlist() — создает матричноориентированный типизованный список.

Создадим список, используя функцию list(), и вызвав функцию, где в качестве аргументов перечислим объекты списка:

```
-->lt=list(complex(1,-5),[2 3 4],[1 2 5;3 4 6]);
-->lt(1),lt(2)(3),lt(3)($,$)
 ans  =
       1. - 5.i
 ans  =
       4.
 ans  =
       6.
```

Чтобы обратится к элементу списка, необходимо указать индекс списка, а затем индекс элемента. В список можно добавлять элементы, как с начала, так и с конца. Чтобы добавить элемент в список в начало, необходимо обратиться к нулевому элементу списка, после чего добавленный элемент станет первым, а все остальные сместятся на позицию вправо. Для добавления элемента в конец следует обратиться к элементу списка с индексом **($+1)**:

```
-->lt(0)=[2 3];
-->lt(1)
 ans  =
    2 3
-->lt($+1)=50;
-->lt($)
 ans  =
    50.
```

Чтобы удалить элемент списка нужно воспользоваться специальной функцией **null()**.

```
-->lt(2)=null()
 lt  =
      lt(1)
    2.    3.
     lt(2)
    2.    3.    4.
      lt(3)
    1.    2.    5.
    3.    4.    6
```

Список **list()** обычно используется, когда данные в нем обезличены.

Типизованный список.

Типизованные списки используются в Scilab в основном для определения новых структур данных. Они позволяют создавать пользовательские типы данных и являются надстройками над списками типа **list**. К элементам типизованного списка можно обращаться по их уникальным именам, что удобно для создания пользовательского интерфейса. Отличие типизованного списка от простого заключается в том, что у него есть поля. Поле такой же элемент списка, только имеющий уникальный идентификатор или, попросту, имя. Для работы с типизованными списками в Scilab определены следующие функции:

tlist() — создает типизованный список.

fieldnames() — позволяет вывести все поля типизованного списка.

definedfields() — позволяет вывести поля, в которых есть данные.

setfield() — позволяет установить поле для типизованного списка.

getfield() — позволяет извлечь из поля данные.

Для примера создадим типизованный список, который будет хранить информацию о некоем пользователе. Пусть у этого списка будут следующие поля: год, месяц и день, и сам список описывается именем ‘Name’.

```
->tlt=tlist(['Name','Year','month','day'],2000,'june',15)
 tlt  =
      tlt(1)
 !Name  Year  month  day  !
      tlt(2)
2000
     tlt(3)
    june
      tlt(4)
    15
```

Наличие имени отличает этот вид списков от обычных list-списков. Первый аргумент в вызове является вектором, в котором последовательно передаются имя списка и типы элементов этого списка.

Таким образом, типу Year было присвоено значение 2000, типу month — строка 'june' и типу day — число 15. Обратиться к конкретному значению списка можно следующим образом:

```
->tlt.Year
 ans  =


    2000.
или
-->tlt ('Year')
 ans  =
    2000.
```

Работа с типизованными списками ничем не отличается от работы с list-спискам. Например:

```
-->tlt(2)                // к любому элементу списка можно обратиться по индексу
 ans  =
 2000
-->tlt(1)                // смысловая часть начинается со второго индекса
 ans  =                  // в первой позиции хранятся имя списка и имена его полей
! 'Name','Year','month','day'!

-->tlt(1)(1)             // чтобы получить имя списка надо обратиться к элементу вектора,
                         //  находящемуся в первой позиции
 ans  =
  Name

-->tlt(3)=null();        // элементы можно удалить функцией null()
                         // при этом все элементы, находящиеся после удаляемого, сдвигаются к
                        // голове списка
```

Посмотрим, как можно добавить поля в типизованные списки. Например, к существующему списку добавим поле номер телефона.

```
-->tlt(1)($+1)='tel.'; setfield(5,111333,tlt);
-->tt.Voltage
 ans  =
    111333.
-->fieldnames(tlt)       // проверяем имена полей списка
 ans  =

!Year   !
!       !
!month  !
!       !
!day    !
!       !
!tel.   !
```

1.5 Объекты: полином и rational

Полиномом называется алгебраическое уравнение вида

$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0$, $a_n \neq 0$, $n \geqslant 1$

В Scilab он определен, как объект polynomial. Для работы с этим объектом предназначены специальные функции. Рассмотрим их. Прежде всего, полином можно задать через функцию **poly()**.

Синтаксис:

```
poly(a,vname,['flag'])
// a — число или матрица (смысл зависит от используемого флага).
// vname — имя символьной переменной в виде строки.
// ['flag'] — флаг, определяет способ задания полинома
// варианты:
// 'r' или 'roots' — в 'a' задаются корни полинома (по умолчанию, если не указан флаг, //также задаются корни полинома);
// 'c' или 'coeff' — если указан этот флаг, то задаются коэффициенты полинома, которые //указаны в 'a'.
```

```
->y=poly([1 2],'x','r')
 y  =
               2
   2 - 3x + x
->y=poly([1 2 3],'x','c')
 y  =
                 2
   1 + 2x + 3x
```

Функции для работы с полиномами:

varn() — позволяет поменять или узнать символическую переменную указанного полинома;

```
-->y=varn(y,'s')
 y  =
               2
 1 + 2s + 3s   // поменяем символьную переменную на 's'.
```

coeff() — позволяет получить вектор-строку с коэффициентами полинома, где коэффициенты расположены, начиная с младшей степени при символической переменной

```
-->coeff(y)
 ans  =


   1.    2.    3.
```

Чтобы найти корни полинома используется функция **roots().**

```
-->R=roots(y)
 ans  =
  - 0.3333333 + 0.4714045i
  - 0.3333333 - 0.4714045i
```

Для вычисления значения полинома при любом значении аргумента используется функция **horner()**.

```
-->horner(y,5)
 ans  =
   86.
->horner(y,1:5)
 ans  =
   6.    17.    34.    57.    86.
```

Объект rational («рациональное число») является производным от объекта polynomial. Он образуется делением одного полинома на другой, но, несмотря на это, имеет свой собственный идентификатор и инициируется как rational.

```
-->y=poly([1 2 3],'x','c')/poly([-1 2 -3],'x','c')
 y  =
                  2
    1 + 2x + 3x
       -----------
                   2
    -1 + 2x - 3x
-->typeof(y)
 ans  =
 rational
```

Значения объекта rational от любого аргумента можно вычислить, используя функцию **horner()**.

1.6 Структуры struct, cell

Структуры **struct**, **cell** используются в Scilab в основном для переносимости сценариев из одной среды в другую (MATLAB и Octave), но бывают случаи, когда удобно пользоваться такой структурой непосредственно в Scilab, например, при обработке массивов данных. Объект **struct** очень похож на типизованный список, но в отличие от него именем структуры является переменная, в которую и записывается эта структура. Внутри **struct** все составляющие могут быть выделены по именам. Создадим переменную s как **struct** с полями: «day» (день), «month» (месяц), «year» (год).

```
-->s=struct('day',06,'month','june','year',2000)
 s  =
   day: 6
   month: "june"
   year: 2000
->s.day //выделение необходимой переменной
 ans  =
    6.
```

Из примера видно, что на нечетных позициях в аргументах стоят имена полей, а на четных — присваиваемые значения.

К сожалению, в **struct** нельзя обращаться к полям по индексам. Чтобы объединить структуры, которые хранят однородные данные, например пользователей, и пользователей несколько, то единственный способ - это записать их в вектор.

Другой тип структурных данных это **cell** (ячейка), это матрица, которая способна хранить данные разных типов, в отличие от матрицы. Для создания **cell**, необходимо вызвать одноименную функцию:

```
-->q = cell(2,3)           // создаем cell-матрицу 2x3
 q  =
!{}  {}  {}  !
!              !
!{}  {}  {}  !
```

Поля этой матрицы могут содержать разные типы данных: **double**, **integer**, **string** и т.д.

Для того, чтобы заполнить позицию в cell-матрице используется специальное служебное слово:

```
-->q(1,3).entries = 1
-->q(2,2).entries ='Пример'
 q  =
!{}  {}           1   !
!                     !
!{}  "Пример"  {}  !
```

Для работы с элементами массива **cell** можно использовать те же операции, что и с обычными матрицами.

```
-->q(2,2)
ans  =
 "Пример"
-->q(1,:)
 ans  =
!{}  {}  1  !
-->q (1 ,3). entries =[] //удалить элемент
 c  =
!{}  {}           {}  !
!                     !
!{}  "Пример"  {}  !
```

С помощью **cell** можно создавать многоиндексные массивы (гиперматрицы). Но в отличие от гиперматриц, созданных с помощью функции **hypermat()**, cell-гиперматрицы способны хранить данные разных типов.

```
-->c = cell(1,2,3)
 c  =
(:,:,1)
!{}  {}  !
(:,:,2)
!{}  {}  !
(:,:,3)
!{}  {}  !
```

Cell удобно использовать, если нужно получить объект, обладающий свойствами матриц и list-списков.

1.7 Создание и отладка программных файлов.

Как отмечалось раннее, в Scilab можно работать в моде командной строки, и (или) программном режиме, поскольку Scilab содержит структурный язык программирования.

Программами (в Scilab её называют ещё сценарием) являются файлы текстового формата с расширением .sce (scenario), содержащие запись программ в виде программных кодов. Основными средствами программирования являются:

- данные различных типов,
- операторы,
- функции пользователя,
- управляющие структуры,
- системные операторы и функции,
- средства расширения языка.

В языке программирования системы Scilab имеются следующие управляющие структуры (перечислим важнейшие):

Операторы ветвления

- условный оператор **if...elseif...else...end,** простейший вариант: **if** условие инструкции **end**

```
--> x=2
x =
2.
--> if x>0 then,y=-2*x,else,y=x,end
y =
- 4.
--> x=-2
x =
- 2.
--> if x>0 then,y=-2*x,else,y=x,end
y =
- 2.
```

- Конструкция **select ... case**

```
--> x=1
x =
  1.
--> select x,case 1,y=x+5,case -1,y=sqrt(x),end
y =
6
```

Операторы ветвления

- цикл типа **for...end,** конструкция цикла имеет вид: **for** var=n:k:m , инструкции, **end,** где n – начальное значение переменной цикла var, k – шаг (по умолчанию k=1), m – конечное значение переменной var

```
-->x=1;for k=1:3,x=2*x*k,end
 x  =
   2.
 x  =
   8.
 x  =
   48.
```

- цикл типа **while...end,** конструкция цикла имеет вид: **while** условие инструкции **end**

```
->x=1; while x<10,x=2.5*x,end
 x  =
   2.5
 x  =
   6.25
 x  =
   15.625
```

Досрочный выход из цикла может быть реализован с помощью операторов **break** и **continue**

```
-->a=0;for i=1:5:1000,a=a+1;if i > 20 then break,end; end
-->a
 a  =
   5.
```

Имеются также функции диалогового ввода и вывода: **a**=**input** (‘String’), **disp(X)**

```
-->x=input('n=');
n=2
-->x
```

```
x  =
   2.
-->disp(x)
   2.
```

Сценарии вызываются по имени, под которым записаны в виде файла с расширением **.sce**. При первом обращении сценарий ищется на диске. Путь к программе можно указать командой **path**.

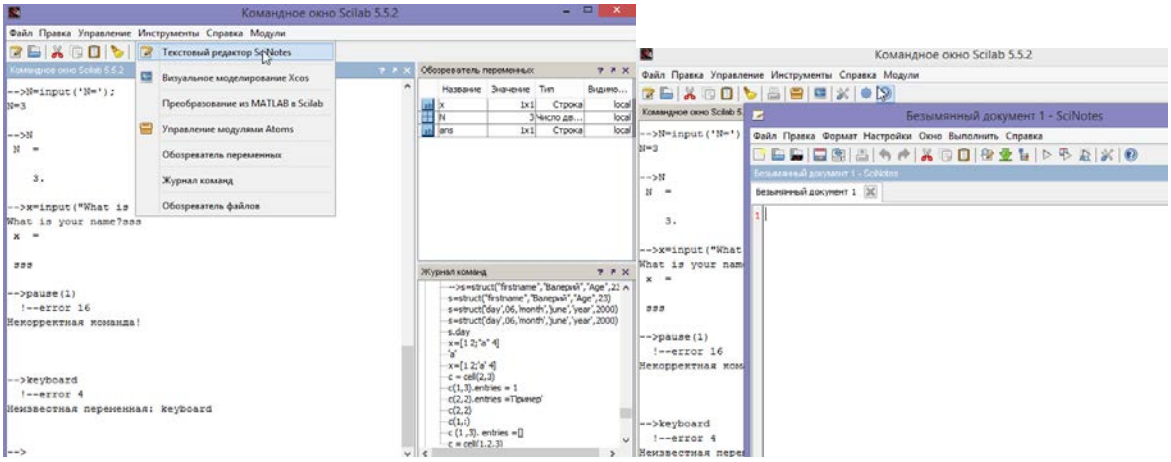


Рис.2 Командное окно и окно редактора SciNotes

Программы бывают двух видов: сценарий (script) и программируемая функция (function). Сценарий - это набор команд без входных и выходных параметров. Любой сценарий состоит из инструкций, которые описывают конкретные действия с объектами Scilab. В свою очередь, инструкция может быть представлена несколькими операндами, которые связаны определённым способом. Программируемая функция также является сценарием, но отличается тем, что она имеет уникальное имя. Благодаря этому, её можно вызвать из любого места сценария.

Для создания программы необходимо:

1. Вызвать команду «текстовый редактор» из меню (см. рис. 2).
2. В окне редактора SciNotes набрать текст программы.
3. Сохранить текст программы с помощью команды **File – Save** в виде файла с расширением .sce , например, name.sce .
4. После этого программу можно вызвать несколькими способами:

   -набрав в командной строке **exec**, например, **exec("name.sce")**.

   -воспользоваться командой меню **File – Exec**. . . командного окна

   -в окне **SciNotes** выполнить команду **Execute – Load into Scilab**

Пример программы-скрипта (определение локальных максимумов вещественного вектора):

```
m=length(x);  // вычисление длины вектора  x
k=2:m-1;    // формирование  вектора  индексов
i=find(x(k-1)<x(k)&x(k)>x(k+1))+1;   // поиск индексов  локальных максимумов
y=x(i);  // значения локальных максимумов
```

Набранную программу следует сохранить, например, под именем test.sce. Теперь вызовем её из командной строки:

```
>> ->x=[7,2,-5,3,-2,8]; exec('test.sce');, [y;i]
 ans  =
    3.
    4.
```

Рассмотрим, как создать программируемую функцию. В зависимости от предпочтений пользователя, такое определение функции можно использовать вместо функции deff(). Общий синтаксис выглядит следующим образом:

```
function <выходные переменные>=имя_функции(аргументы)
...
инструкции
...
endfunction
```

Первая строка программы-функции имеет вид: **function [y1,y2,...yj]=fn(x1,x2,...xi)**, где **fn** – имя функции, **x1**,**x2**,...**xi** – входные формальные параметры, **y1**,**y2**,...**yj** – выходные формальные параметры. Вызов функции осуществляется следующим образом: **[v1,v2,...vj]=ff(u1,u2,...ui)**, где **u1**,**u2**,...**ui**,**v1**,**v2**,...**vj** – фактические параметры. Входных аргументов у функции может и не быть. Все переменные программы-функции имеют локальный характер и недоступны для их использования извне. Для связи с другими модулями программы можно использовать глобальные переменные, определив их командой: **global var1 var2**... Рассмотрим пример создания программы-функции, используя SciNotes.

```
function [x]=fact(k) //вычисление факториала
k=int(k)
if k<1 then k=1,end
x=1;
for j=1:k,x=x*j;end
endfunction
```

Файлы-функции могут вызываться так же из других sce-файлов.

1.8 Ввод и вывод данных

Очень часто необходимо конечные или промежуточные результаты вычислений или большие объемы информации сохранять в файлы. Для этого в Scilab существует набор команд, который позволяет, как записывать данные в файлы, так и читать их из файлов. Рассмотрим их использование на примерах:

```
--> x=[1 2 %pi;0 5 4]
x =
 1. 2. 3.1415927
 0  5. 4.
--> write('x.dat',x)
--> clear x
--> xnew=read('x.dat',2,3)
xnew =
 1. 2. 3.1415927
 0  5. 4.
```

Также существуют более мощные Си-подобные функции **mfscanf** и **mfprintf**. Рассмотрим использование этих функции на предыдущем примере:

```
--> x=[1 2 %pi;0 5 4]
x =
  1. 2. 3.1415927
  0  5. 4.
--> fd=mopen('x_c.dat','w') // открывает файл x_c.dat на запись
--> mfprintf(fd,'%f %f %f\n',x) // выполняет форматный вывод данных в файл или
                                  //командное окно
--> mclose(fd) // закрывает открытый файл
--> clear x
--> fd=mopen('x_c.dat','r') // открывает файл x_c.dat на чтение
--> xnew(1,1:3)=mfscanf(fd,'%f %f %f\n') ;
--> xnew(2,1:3)=mfscanf(fd,'%f %f %f\n')
xnew =
 1. 2. 3.141593
 0  5. 4.
--> mclose(fd)
```

1.9 Графика

Система Scilab имеет развитые возможности графического представления данных – от построения простых графиков функций до трехмерных комбинированных и презентационных графиков. Вначале, кратко рассмотрим, какие команды используются в Scilab для создания и работы с графическими окнами.

Команды **scf(i)**, **figure(i)** – делают активным i–е окно при i<=k (если открыто k окон) или открывает окно с номером i при i>k

clf – очищает текущее графическое окно (по умолчанию окно очищается при построении очередного графика)

mtlb_hold on – запрещает очистку окна до подачи команды **mtlb_hold off**

winsid- позволяет посмотреть весь перечень открытых окон

xdel(i)-позволяет закрыть графические окна, где i вектор номеров этих окон

subplot(m,n,p) – разбивает окно на m*n подокон, p – номер текущего подокна.

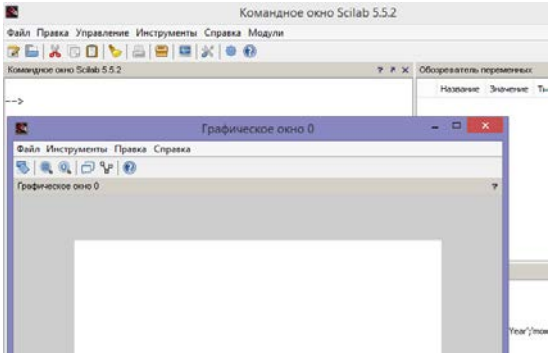


Рис.3 Графическое окно

На рисунке 3 представлено графическое окно, которое открывается при выполнении команд **scf(i)**, **figure(i)** или по умолчанию, при выполнении команд построения графиков.

Рассмотрим построение графиков одномерных функций.

Для построения графика функции одной переменной в декартовых координатах в Scilab используются следующие команды:

plot(x,y) – строит график функции y(x), где y и x – вектора одинаковой длины и является аналогом семейства plot2d (служит для совместимости с MATLAB).

plot2d — строит график, определяемым пользователем;

plot2d2 — строит график в виде ступенчатой функции;

plot2d3 — строит график в виде вертикальных полосок;

plot2d4 — строит график с указанием направления;

fplot2d()— строит график по заранее определенной функции;

Построим график одномерной функции, используя команду **plot(x,y)**.

Полный прототип функции выглядит следующим образом:

plot(y,<LineSpec>,<GlobalProperty>)

plot(x,y,<LineSpec>,<GlobalProperty>)

plot(x1,y1,<LineSpec1>,x2,y2,<LineSpec2>,...xN,yN,<LineSpecN>,<GlobalProperty1>,<GlobalProperty2>,..<GlobalPropertyM>)

plot(<axes_handle>,...)

где:

x-вектор или матрица рациональных чисел. Если отсутствует, то по умолчанию предполагается вектор **1:n**, где n –число точек, задаваемых параметром y.

y- вектор или матрица рациональных чисел y

<**LineSpec**> Опциональный аргумент строкового типа. Используется для задания свойств линий графиков: цвет линии, тип маркера данных, стиль линии.

<**GlobalProperty**>**{PropertyName,PropertyValue**}- опциональный аргумент строкового типа, который определяет общие свойства графика.

<**axes_handle**> -эта опция позволяет позиционировать вершину графика

Более подробное описание можно посмотреть в справочной системе Scilab.

Пример.

```
--> x=-2*pi:0.1*pi:2*pi;
--> y1=cos(x);
--> y2=cos(x).^2;
--> y3=cos(x).^3;
--> plot(x,y1,'-Y',x,y2,'-+R',x,y3,'-.ogr')
```

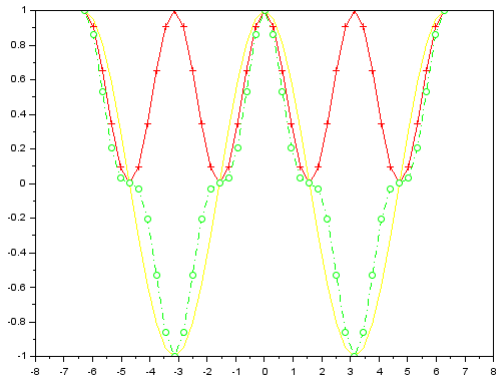


Рис 4. Графики функций $\cos(x),\cos^2(x),\cos^3(x)$

Теперь, поместим начало координат в середине графика, используя следующую последовательность команд:

Результат представлен на рисунке 5.

```
-->a=gca();
-->a.x_location="middle";
 -->a.y_location="middle";
```

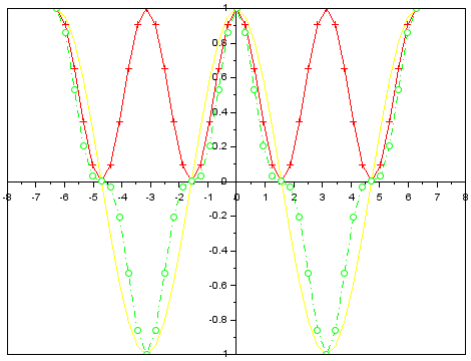


Рис 5. Графики функций cos(x),cos^2(x),cos^3(x)

Воспользуемся следующими командами, чтобы сделать подписи к самому графику и осям. Результат представлен на рисунке 6.

Для справки, приведём значения параметров функции plot(), которые используются для задания таких параметров графика, как : цвет, тип маркеров данных и тип линии графика (см. таблица 1).

```
a=gca();
a.title.text='графики cos(x),cos^2(x),cos^3(x))';// подпись
//графика
a.x_label.text='X'; // подпись для оси абсцисс
a.y_label.text='Y'; //  подпись для оси ординат
a.grid=[0 0]; // включение сетки и задание её цвета
//(черный)
a.children(1).children.thickness=4;//задание толщины
//линии
```

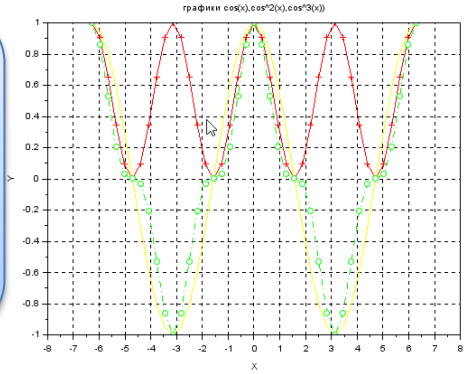


Рис 6. Графики функций cos(x),cos^2(x),cos^3(x)

Значения параметра функции **plot**, определяющего цвет графика:

Таблица 1

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

Значения параметра, определяющего тип линии графика:

Таблица 2

Символ	Описание
-	сплошная
:	пунктирная
-.	штрихпунктирная
--	штриховая

Теперь обратимся к функции **plot2d**, которая позволяет пользователю полностью определять внешний вид графика. Для этого используются параметры **keyn=valuen.**

Полный прототип функции **plot2d** выглядит следующим образом:

plot2d([logflag],x,y',[key1=value1,key2=value2,...,keyn=valuen]

Рассмотрим некоторые значения параметра keyn=valuen (подробнее см. руководство):

• массив styleопределяет цвета графика. Количество элементов массива совпадает с количеством изображаемых графиков.Имеется два способа задания цветов графиков. Воспользоваться функцией color, которая по названию (color("имя цвета")) формирует цвет, или использовать код rgb (color(r,g,b)) (см. справку).

Значение параметра, определяющего тип маркеров (точек) графика:

Таблица 3

Символ	Описание
.	точка
o	кружок
x	крестик
+	знак "плюс"
*	звездочка
s	квадрат
d	ромб
v	треугольник вершиной вниз
^	треугольник вершиной вверх
<	треугольник вершиной влево
>	треугольник вершиной вправо
p	пятиконечная звезда
h	шестиконечная звезда

Пример:

```
x=[-2:0.1:2];
y=[x;x^2-1];
plot2d(x,y',style=[color("red"),color(0,176,0)]);
```

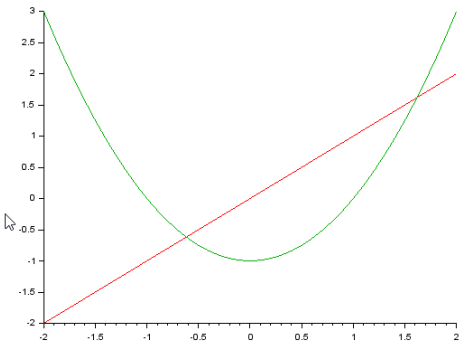


Рис 7. Графики функций x, x^2-1

• **axesflag** определяет наличие рамки вокруг графика. Значения этого параметра:

0 нет рамки;

1 изображение рамки, ось y слева (по умолчанию);

3 изображение рамки, ось y справа;

5 изображение осей, проходящих через точку (0,0)

Пример:

```
x=[-2:0.1:2];
y=[x;x^2-1];
plot2d(x,y',style=[color("red"),color(0,176,0)],axesflag=5);
```

Построенный график представлен на рисунке 8.

Остальные параметры, позволяющие управлять свойствами графика можно посмотреть в справочной системе пакета Scilab.

```
x=[-2:0.1:2];
y=[x;x^2-1];
plot2d(x,y',style=[color("red"),color(0,176,0)],axesflag=5);
```

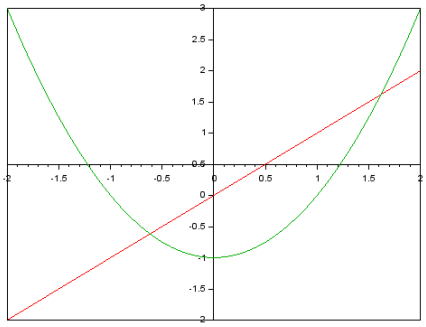


Рис 8. Графики функций x, x^2-1

Кроме управления свойствами графиков с помощью команд, в пакете Scilab предусмотрена намного более удобная возможность управлять свойствами с помощью графического редактора. В графическом окне, как видно из рис.3, находятся следующие пункты меню: Файл, Инструменты, Правка, Справка. В пункте «Правка» возможно управление свойствами графика, отображением данных и их редактированием,

изменением масштабов осей, созданием надписей и т.д. Для более полного представления о возможностях графического редактора рекомендуется познакомиться со встроенной справкой и поэкспериментировать с каким-нибудь графиком.

Для построения графиков в полярной системе координат служит команда **polarplot()**. Полный прототип функции выглядит так:

polarplot(theta,rho,[style,strf,leg,rect])

где: **theta** — азимут, **rho** — радиальная координата, **[style,strf,leg,rect]** — настройки внешнего вида

Пример:

```
t= 0:.01:2*%pi;
polarplot(sin(5*t),cos(10*t))
```

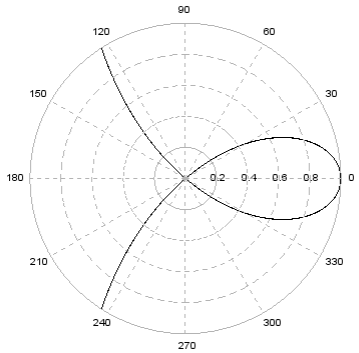


Рис 9. График в полярной системе координат *teta= sin(5*t), ro=cos(10*t)*

Кроме графиков в декартовой и полярной системах координат в Scilab возможно построение и других типов. Рассмотрим примеры построения некоторых из них.

Функция **plot2d2( )** позволяет строить ступенчатые графики.

Пример:

```
x=[0:0.1:2*%pi]';
clf()
plot2d2(x,[x x^2 x^3])
```

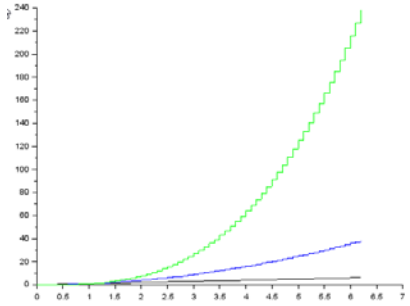


Рис 10. Ступенчатые графики функций x, x^2,x^3

С помощью функции **plot2d3( )** можно построить линейчатые графики, состоящие из отрезков прямых.

Пример:

```
x=[0:0.1:2*%pi]';
clf()
plot2d3(x,[cos(x) cos(2* x) cos(3* x)])
```

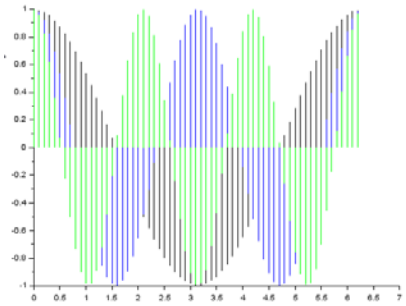


Рис 11. Линейчатые графики функций cos x, cos 2x,cos 3x

Функция **plot2d4( )**, позволяет строить графики вида, где соседние точки соединяются векторами:

```
x=[0:0.1:2*%pi]';
clf()
plot2d4(x,[cos(x) cos(2* x) cos(3* x)])
```

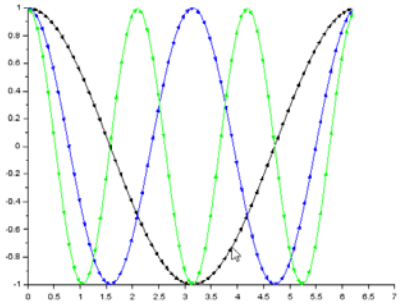


Рис 12. Графики функций *f(x)=cos x, f(x)=cos 2x, f(x)=cos 3x*

Функция **subplot(m,n,p)** или **subplot(mnp)** делит графическое окно на **m x n** частей, в каждой из которых можно поместить свой график. Где **m** – количество частей по вертикали, а **n** – по горизонтали, параметр **p** задаёт текущее окно. Построим четыре графика в одном графическом окне -sin(x), cos(x), x, x^2:

```
x=[-5:0.1:5];
y=sin(x); z=cos(x);
u=x; v=x^2;
subplot(3,2,1);plot(x,y);
subplot(3,2,2);plot(x,z);
subplot(3,2,3);plot(x,u);
subplot(3,2,4);plot(x,v);
```

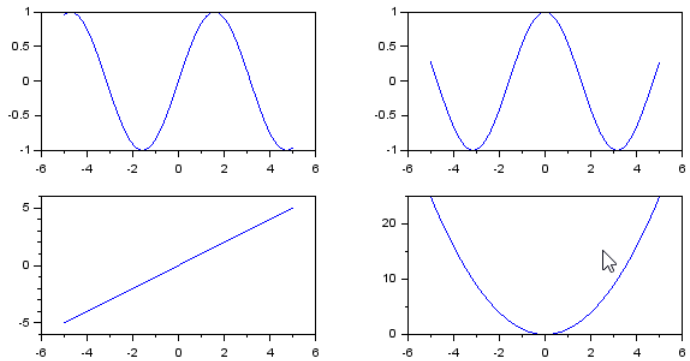


Рис 13. Графики функций *f(x)=sin x, f(x)=cos x, f(x)=x, f(x)=x^2*

Рассмотрим ещё несколько типов графиков, которые возможны в Scilab. Это прежде всего гистограммы, столбчатые диаграммы, круговые. Вначале построим гистограмму,

используя функцию **histplot()**. Полное описание всех, используемых параметров этой функции можно посмотреть в справке пакета Scilab. Для генерации массива данных из 1000 значений воспользуемся функцией **rand()**, и возьмём количество бинов равное 20.

```
-->a=rand(1,10000,'normal');  //
генерация нормального распределения
-->clf(); histplot(20,a)
```

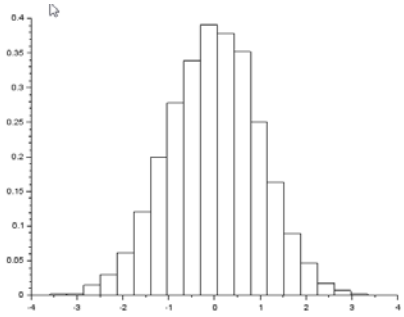


Рис 14. Гистограмма гауссова распределения

Для построения столбчатых диаграмм воспользуемся командой **bar()**.

Пример:

```
x=[1 3 7];y=[1  -8 9;2 -2 5;4  -4 3];
bar(x,y);
```

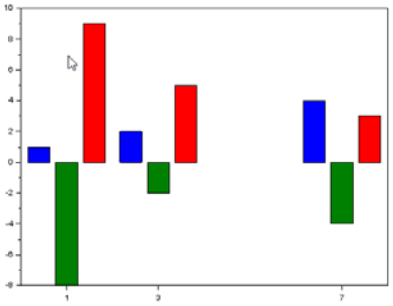


Рис 15. Столбчатая диаграмма

Посмотрим, как можно построить круговую диаграмму. В Scilab такая диаграмма строится с помощью функции **pie(x[,sp[,txt]])**.

Пример построения такой диаграммы:

```
// три входных параметра, x=[ 4 8 2 1 ], sp=[0 1 0 1],
//txt=["часть1","часть2","часть3","часть4"]
pie([4 8 2 1],[0 1 0 1],["часть1","часть2","часть3","часть4"]);
```

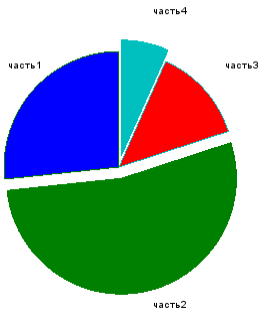


Рис 16. Круговая диаграмма

Помимо инструментария для построения двумерных графиков пакета Scilab содержит большой набор средств трехмерной графики**.** Ниже приведён полный список функций,

которые можно использовать при построении 3d графиков. Полную информацию, касающуюся приведённых функций можно посмотреть в справке Scilab. Некоторые из них рассмотрим поподробнее.

mesh — график трёхмерной сетки

comet3d — 3D анимированный график

contour —построение изолиний(линии одинаковых уровней значений исследуемой величины)

eval3d — значения функции на сетке

eval3dp — вычисление граней 3D-параметрической поверхности

fac3d — 3D график поверхности

fcontour — построение изолиний на 3D поверхности, задаваемой функцией

fplot3d — 3D график поверхности, задаваемой функцией

fplot3d1 — 3D монохромный или цветной график изолиний поверхности, заданной с помощью функции

genfac3d —преобразует поверхность заданную через координаты векторов x,y, и матрицу z(x, y) в набор трёх 4 x N матриц, дающих координаты N связанных прямоугольных граней(фасет)

geom3d — даёт 2D проекцию 3D графика, после того как построен 3D график

hist3d — 3D гистограмма

nf3d — преобразует координаты в прямоугольные грани (фасеты) для plot3d

param3d — 3D график параметрических кривых

plot3d — 3D график

plot3d1 — 3D монохромные или цветные изолинии поверхности

plot3d2 — график поверхности, заданной прямоугольниками (фасетами)

plot3d3 — график в виде сетки из прямоугольников (фасет)

surf — поверхностный 3D график

Прежде всего, это различные средства **построения поверхностей**, описываемых функцией двух переменных

например:

meshgrid(X,Y) –задание опорной области для построения трехмерной поверхности (команда **meshgrid(X,X)** эквивалентна команде **meshgrid(X)**);

Пример:

```
//поверхность, изображенная методом сеток
-->[X,Y]=meshgrid([-2:.5:2,0:0.5:5]);
-->Z=X.^2-Y.^2;
-->mesh(X,Y,Z,'edgecol','blu')
```

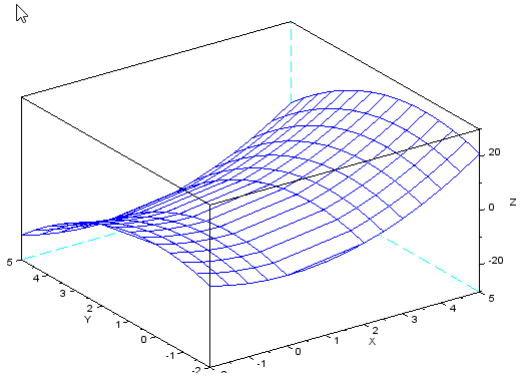


Рис 17. График функции $Z(X,Y)=X^2-Y^2$

Функция **surf**.

Позволяет отобразить сетку с окраской поверхности.

Пример.

```
[X,Y]=meshgrid(-1:.1:1,-1:.1:1);
Z=X.^2-Y.^2;
surf(X,Y,Z);
```

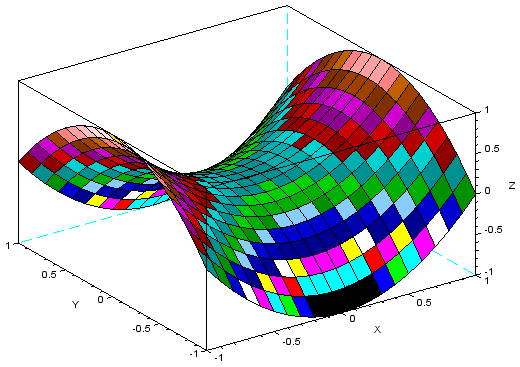


Рис 18. График функции $Z(X,Y)=X^2-Y^2$

Рассмотрим действие функций семейства **plot3d().**

Пример с использованием функции **plot3d**:

```
x=[0:0.2:2*%pi]';
y=[0:0.2:2*%pi];
z=tan(x)*cos(y);
plot3d(x,y,z)
```

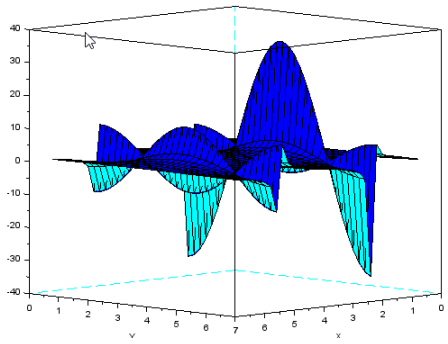


Рис 19. 3d график функции $Z(x,y)=tg(x)\cdot cos(y)$

Пример с использованием функции **plot3d1**:

```
//plot3d1 - поверхность с линиями
уровней.
x=[0:0.2:2*%pi]';
y=[0:0.2:2*%pi];
z=tan(x)*cos(y);
plot3d1(x,y,z)
```

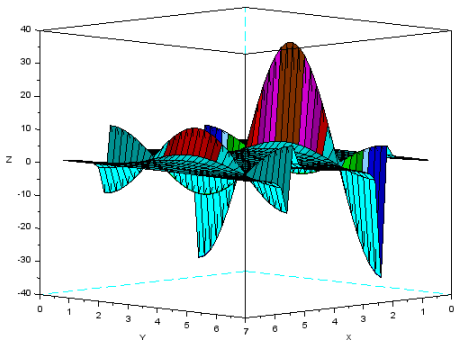


Рис 20. 3d график функции $Z(x,y)=tg(x)\cdot cos(y)$

Функция **plot3d2**.

Как отмечалось ранее, эта функция предназначена для построения поверхности, которая задается набором граней. С помощью функции **plot3d** по входным данным можно построить лишь отдельно стоящие друг от друга плоские грани, тогда как, использование функций plot3d2, (plot3d3) позволит, учитывая взаимное расположение этих граней, отобразить 3d график в виде цельного геометрического тела.

```
u = linspace(-%pi/2,%pi/2,40);
v = linspace(0,2*%pi,20);
X = sin(u)'*sin(v);
Y = cos(u)'*cos(v);
Z = cos(u)'*ones(v);
plot3d2(X,Y,Z);
```

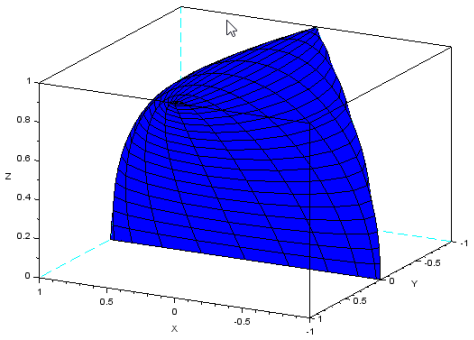


Рис 21. 3d поверхность из примера

Функция **plot3d3** позволяет отобразить поверхность с прямоугольными гранями без закраски поверхности.

```
u = linspace(-%pi/2,%pi/2,40);
v = linspace(0,2*%pi,20);
X = sin(u)'*sin(v);
Y = cos(u)'*cos(v);
Z = cos(u)'*ones(v);
plot3d3(X,Y,Z);
```

В обоих, рассмотренных примерах для большей наглядности использовалась функция 2d/3d вращения фигуры из меню графического окна «Инструменты».

```
u = linspace(-%pi/2,%pi/2,40);
v = linspace(0,2*%pi,20);
X = sin(u)'*sin(v);
Y = cos(u)'*cos(v);
Z = cos(u)'*ones(v);
plot3d3(X,Y,Z);
```

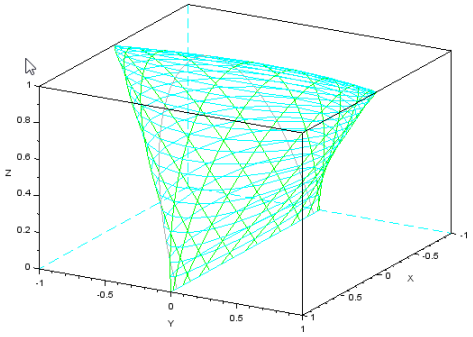


Рис 22. 3d поверхность из примера

Функция contour отображает контуры на 3d поверхности.

```
t=linspace(-%pi/2,%pi/2,15);
function z=my_surface(x,y)
z=x*sin(x)^2*cos(y),
endfunction
contour(t,t,my_surface,10)
```

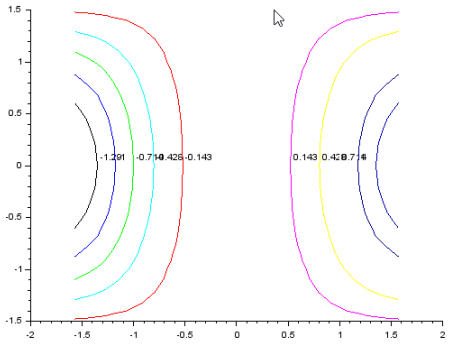


Рис 23. Контурный график поверхности $z(x,y)=x \cdot sin^2(x) \cdot cos(y)$

Функция **hist3d** позволяет отобразить данные в виде объёмной 3d гистограммы.

```
hist3d(10*rand(10,10,'normal'));
```

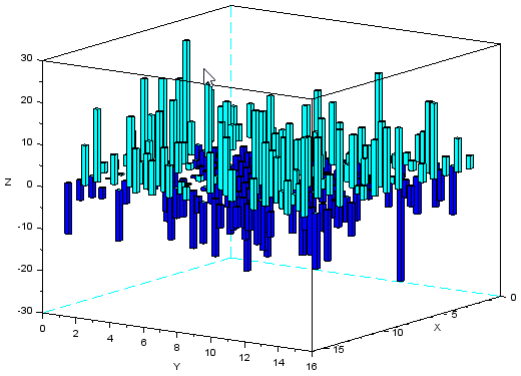


Рис 24. 3d гистограмма матрицы из данных, распределённых нормально

Функция **colorbar()** позволяет для функций **plot3d**(), **fec()**, **Sgrayplot()** и др. отобразить рядом с графиками цветовую шкалу.

Пример:

```
x = linspace(0,1,81);
z = cos(2*%pi*x)'*sin(2*%pi*x);
zm = min(z); zM = max(z);
xset("colormap",jetcolormap(64))
colorbar(zm,zM)
Sgrayplot(x,x,z)
```

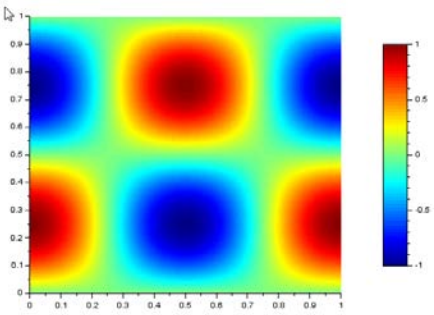


Рис 25. Сглаженный контурный график с использованием цвета и цветовой шкалой

В заключение, познакомимся с дополнительными удобными возможностями для анализа данных, которые представляет Scilab. Рассмотрим их на примере. Создадим матрицу случайных чисел с нормальным распределением : x=10*rand(15,15,"normal"). Она отобразится в командном окне в обозревателе переменных в виде переменной x.

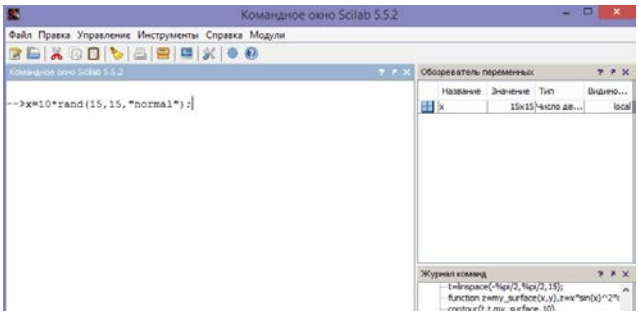


Рис 26. Командное окно

Два раза щёлкнем левой кнопкой мыши по синему значку рядом с x. У нас отобразится матрица значений переменной x в редакторе переменных, где можно редактировать значения переменных.

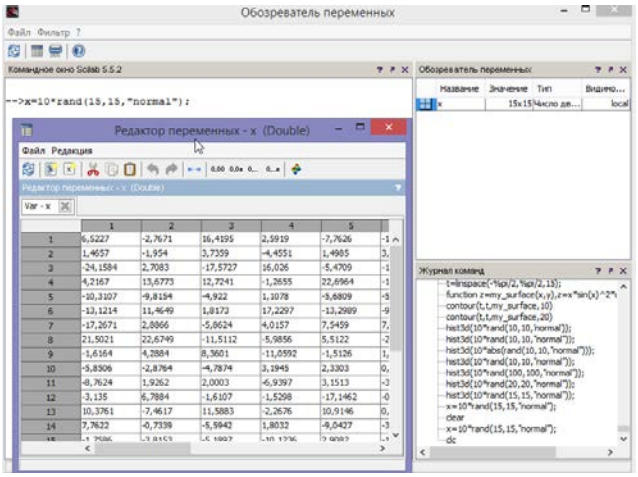


Рис 27. Окно редактора переменных

Выбрав необходимые значения или все значения переменной левой кнопкой мыши, нажмём правую кнопку мыши. Мы увидим дополнительное меню, в котором указаны действия, которые можно провести с данными. Выберем пункт «Построить графическое изображение матрицы». Появится дополнительное меню, где можно выбрать вид графика, который хотите построить. Выберем Matplot , и получим нужный график.

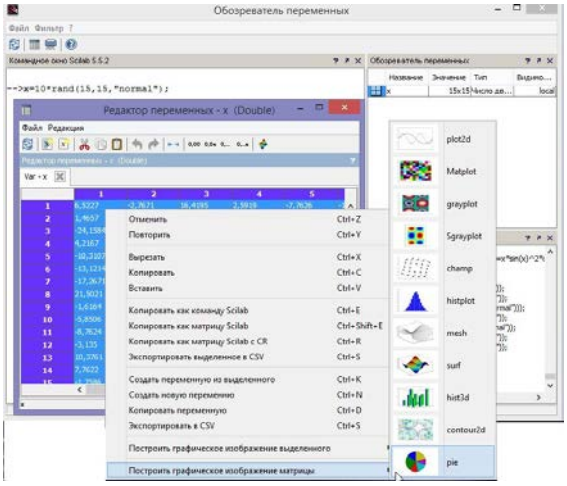


Рис 28. Окно редактора переменных с дополнительным меню

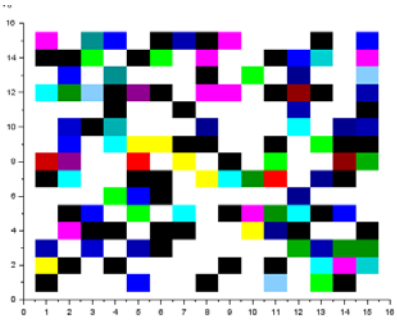


Рис 29. Матрица переменной x, где цвет отображает значение данных

Стоит поэкспериментировать с перечисленными в меню командами и изучить, какое действие они оказывают на переменную.

В заключение этой главы, хотелось бы обратить внимание, что арсенал функций и возможностей, встроенных в Scilab, не ограничивается, рассмотренными здесь. Пользователю Scilab необходимо привыкать интенсивно пользоваться встроенной справкой системы и интернет-ресурсами для успешной работы с этим пакетом. Тем не менее, изложенного в первой главе будет достаточно для решения задач и примеров, которые будут представлены ниже.

Вычисления в Scilab

2

Наиболее очевидная вещь – это использование системы Scilab для вычисления значений математических выражений, построение графиков различных функций. Поэтому, в этой главе рассмотрим на примерах способы проведения вычислений, задание формата данных и его влияние на конечный результат.

2.1 Вычисления с одной числовой переменной

Начнём с простейших операций.

Пример 1. Вычислить значения следующих выражений: *c=3+a+2·b; c=2·(a+b)-4* при *a=1*, *b=1*.

```
->clear
-->a=1;b=1;
-->z=3+a+2*b
 z  =
      6.
 -->z=2*(a+b)-4
 z  =
   0.
-->format('e',25) //формат числа распечатываемого и выводимого на экран
 -->a=1;b=1;
-->z=2*(a+b)-4
 z  =
   0.0000000000000000000D+00
```

Пример 2.Умножить и разделить числа *a=3.7* и *b=5* по следующим формулам: *z=a·3·b*, z=$\frac{(a+b)-2}{a-b}$. Результаты вывести на экран в экспоненциальном формате с максимальным количеством значимых цифр. В Scilab используются числа с двойной точностью, а машинная точность хранится в глобальной переменной %eps.

```
->clear
-->format('e',25)//формат числа распечатываемого и выводимого на экран
-->a=3.7;b=5;
-->z1=a*3*b
 z1  =
   5.5500000000000000711D+01
-->z2=((a+b)-2)/(a-b)
 z2  =
 - 5.1538461538461538188D+00
-->c=%eps
 c  =
   2.2204460492503130808D-16
```

Пример 3. Возвести число *a=5.1* в степень *b=3/4*. Результат вывести на экран в формате с плавающей запятой. Количество значимых цифр, включая знак 8.

```
-->format('v',8)
-->a=5.1
 a  =
   5.1
-->b=3/4
 b  =
   0.75
-->z=a^b
 z  =
   3.39373
```

Пример 4. Возвести число π в степень ***e***. Результат представить в формате числа с плавающей запятой и в экспоненциальном формате с максимальным количеством значимых цифр.

```
-->format('v',25)
-->%pi^%e
 ans  =
   22.45915771836104113390
-->format('e',25)
-->%pi^%e
 ans  =
   2.24591577183610411 3D+01
```

Пример 5. Вычислить значение тригонометрических функций : *sin(x), cos(x), tg(x), ctg(x)* при x=π/4. Результат представить в формате числа с плавающей запятой с количеством значимых цифр принятых в Scilab по умолчанию.

```
-->format('v',10)
-->x=%pi/4;
-->a=sin(x)
 a  =
   0.7071068
-->b=cos(x)
 b  =
   0.7071068
-->c=tan(x)
 c  =
   1.
-->f=cotg(x)
 f  =
   1.
```

Пример 6. Вычислить значение тригонометрических функций : *arcsin(x), arcos(x), arctg(x), arcctg(x)* при x=0.5. Результат представить в градусах в формате числа с плавающей запятой с количеством значимых цифр принятых в Scilab по умолчанию.

```
->x=0.5;
-->a=asind(x)
 a  =
   30.
-->b=acosd(x)
 b  =
   60.
-->c=atand(x)
 c  =
   26.565051
-->f=acotd(x)
 f  =
   63.434949
```

Пример 7. Вычислить значение выражения $f(x) = sin^2(x) \cdot \left(x - \sqrt{exp(\sqrt{x})}\right)^3$ при значениях x = 1,4,30.

```
-->x=[1,4,30]
 x  =
   1.    4.    30.
-->y1=(sin(x)).^2
 y1  =
   0.7080734    0.5727500    0.9762065
-->y2=(x-sqrt(exp(sqrt(x)))).^3
 y2  =
 - 0.2730074    2.1056085    3070.4274
-->y=y1.*y2
 y  =
 - 0.1933093    1.2059873    2997.3711
```

Отметим, что при решении этой задачи вычисляемое выражение было разбито на части. Таким образом, можно уменьшить количество вероятных ошибок, если вычисляются значения достаточно сложного выражения.

Пример 8. Построить график функции, приведённой в предыдущей задаче в интервале x от 1 до 30 с интервалом x через 0.4.

```
x=[1:0.4:30]
y1=(sin(x)).^2;
y2=(x-sqrt(exp(sqrt(x)))).^3;
y=y1.*y2
plot(x,y)
```

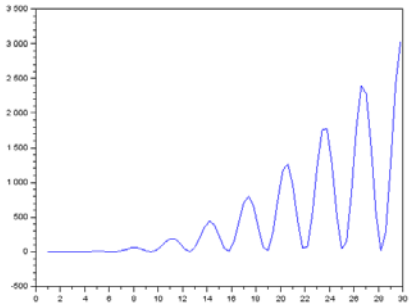


Рис 30. График функции к задаче 8

Задачи.

Вычислить N значений функции на отрезке. На экран вывести значения аргумента и значения функции. Построить график заданной функции.

1. Функция $f(x)=\frac{\sin x\cos x}{x+\cos x}$, N=15, на отрезке [0, 2π].

2. Функция $f(x)=\ln(2x-1)\sqrt{e^{-x}+4e^{x}}$, N=10, на отрезке [0.7, 4].

3. Функция $f(x)=(\sin x-1)\sqrt{e^{-\cos x}+tg\ x\ e^{x}}$, N=20, на отрезке [0.05, 1].

4. Функция $f(x)=x\sin x+\frac{e^{-x}-e^{x}}{e^{-x}+e^{x}}$, N=30, на отрезке [0, 1].

5. Функция $f(x)=x\,(\sin x+\cos x)\frac{x-ctg\ 2x}{1+sin^{2}\ x}$, N=10, на отрезке [0, π/2].

6. Функция $f(x)=\frac{x}{1+\sqrt[3]{x+1}}$., N=30, на отрезке [10,100].

7. Функция $f(x)=\frac{sh\ x+ch\ x}{th\ x+5}$, N=20, на отрезке [-3,3].

8. Функция $f(x)=x^{3}\,|\sin x+\cos x\,|+x-tg\ 2x$, N=10, на отрезке [0, π/2].

9. Функция $f(x)=\log_{2}(\sin x+1)\sqrt{e^{-\cos x}}$, N=30, на отрезке [0, π/2].

10. Функция $f(x)=\frac{\sqrt[3]{x^{2}+1}}{\sqrt{|x|+0.5}}\frac{x}{1+sin^{2}\ x}$, N=10, на отрезке [-π/2, π/2].

11. Функция $f(x)=\frac{sin^{2}(2x+\frac{\pi}{2})}{\sqrt{\frac{|x|+1}{4}}}+\frac{xln\ x}{1+x^{2}}$, N=20, на отрезке [-π/2, π/2].

12. Функция $f(x)=\frac{\sqrt{(3x+1)^{3}}}{x+2}+\sin^{2}(\frac{2x-1}{x+1})$, N=15, на отрезке [0, 1].

13. Функция $f(x)=\sqrt{\log_{2}(\sin x+1)}+\sqrt{1+x^{3.5}}$, N=30, на отрезке [0, π/2].

14. Функция $f(x)=4cos^{2}\left(\frac{x+\frac{\pi}{3}}{x+2}\right)-\ln(\frac{e^{-x}-e^{x}}{e^{-x}+e^{x}})$, N=20, на отрезке [0, 1].

15. Функция $f(x)=\ln\left(\sqrt{\frac{2x-1}{x+1}}\right)+\sin(\sqrt{2+x})$, N=10, на отрезке [0.7, 4].

16. Функция $f(x)=4-\cos^{2}(\frac{e^{-x}-e^{x}}{e^{-x}+e^{x}})$, N=20, на отрезке [0, 1].

17. Функция $f(x)=\frac{e^{-x}-e^{x}}{2}-\frac{e^{-x}-e^{x}}{e^{-x}+e^{x}}$, N=30, на отрезке [-3, 3].

18. Функция $f(x)=\left(\frac{e^{-x}-e^{x}}{2}\right)^{2}+\left(\frac{e^{-x}+e^{x}}{2}\right)^{2}$, N=30, на отрезке [-3, 3].

19. Функция $f(x) = e^{\frac{x+5}{x^2}} - \sqrt{x}$, N=20, на отрезке [0, 3].

20. Функция $f(x) = \text{tg}\left(\sqrt{\frac{2x-1}{x+1}}\right) + \sqrt{2+x}$, N=15, на отрезке [0.7, 4].

21. Функция $f(x) = exp(-x^2/2)$, N=15, на отрезке [-20 ;20].

22. Функции $y = sin(x), y = cos(x), y = tg(x)$ построить на одном графике на интервале x=[-π/2;π/2].

2.2 Вычисления с несколькими числовыми переменными.

Пример 1. Вычислить значения функции $z=x\cdot y$ в области $[0\leq x \leq 10]$, $[0\leq y \leq 10]$ и построить график этой функции.

```
x =(0:0.1:10)';
y=x; z=x*y';
plot3d(x,y,z)
```

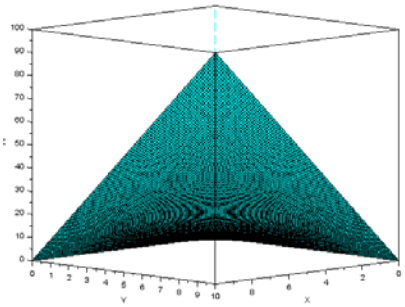


Рис 31. График функции $z=x\cdot y$

Пример 2. Вычислить значения функции $z(x, y) = sin\, x \cdot e^{-3y}$ в трёх точках **x**=0, π/2, π, **y**=0, 0.5, 1 и построить график в области $[0\leq \mathbf{x} \leq 2\pi]$, $[0\leq \mathbf{y} \leq 1]$.

Для разбиения переменных x и y на равномерные интервалы воспользуемся функцией **linspace(x1,x2,n)**, где: **x1**, **x2** – начало и конец интервала соответственно, **n**- число интервалов, на которые необходимо разбить отрезок. Размерность полученного вектора ***x*** можно проверить с помощью функции **size(x).**

```
-->x=[0,%pi/2,%pi]';
-->y=[0,0.5,1];
-->z=sin(x)*exp(-y);
-->z
 z  =
   0.          0.          0.
   1.          0.6065307   0.3678794
   1.225D-16   7.428D-17   4.505D-17
```

Построим график нашей функции

```
-->x=linspace(0,2*%pi,51)';
-->y=linspace(0,1,51);
-->[X,Y]=ndgrid(x,y);
-->z=sin(X)*exp(-Y);
-->plot3d(x',y,z)
```

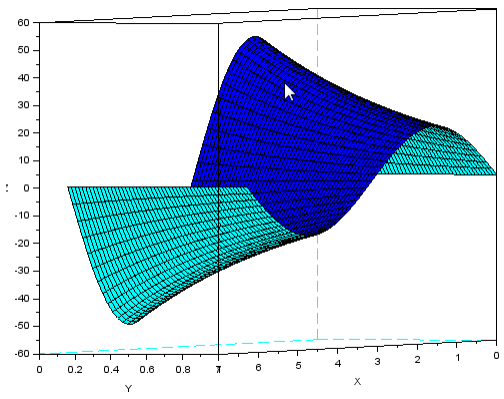


Рис 32. График функции $z(x)=sin(x)\cdot e^{-y}$

Пример 3. Вычислить значения функции $z(x, y) = sin\ x$ в трёх точках **x**=0, π/4, π/2, и построить график в области [0≤ **x** ≤ 2π], [0≤ **y** ≤ 2π].

```
->x=[0,%pi/4,%pi/2]'
 x  =
   0.
   0.7853982
   1.5707963
-->z=sin(x) * ones(x)';
-->z
 z  =
   0.          0.          0.
   0.7071068   0.7071068   0.7071068
   1.          1.          1.
```

```
-->x=[0:%pi/16:2*%pi]';
-->z=sin(x) * ones(x)';
-->plot3d(x, x, z);
```

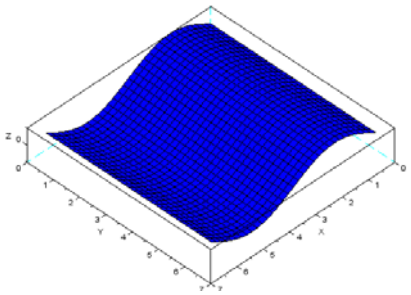


Рис 33. График функции $z(x,y)=sin\ x$ в области [0≤ **x** ≤ 2π], [0≤ **y** ≤ 2π].

Пример 4. Построить график функции $z(x, y) = sin\ x$ в области [0≤ **x** ≤ 2π], [0≤ **y** ≤ 9].

```
-->x=[0:%pi/16:2*%pi]';
-->y=[0:0.5:9];
-->z=sin(x)*ones(y);
-->plot3d(x,y,z)
```

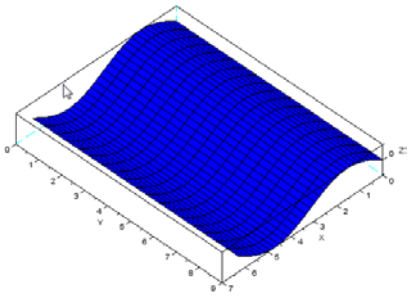
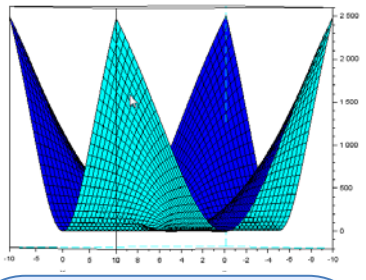
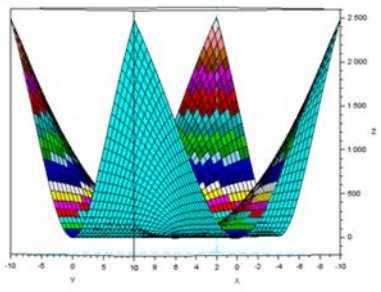


Рис 34. График функции *z(x,y)=sin x* в области [0≤ **x** ≤ 2π], [0≤ **y** ≤ 9]

Пример 5. Вычислить значения функции $z(x,y)=\frac{x^2y^2-2xy-5}{x^2+y^2+1}$ в трёх точках [0,0],[-1,0], [0, -1] и построить график в области [-10≤ x ≤ 10], [-10≤ y ≤ 10].

```
-->deff('[Z]=fun(X,Y)','Z=(X.^2.*Y.^2-2*X.*Y-5)./(X.^2.+Y.^2+1)')
-->fun(0,0)
 ans  =
  - 5.
-->fun(-1,0)
 ans  =
  - 2.5
-->fun(0,-1)
 ans  =
  - 2.5
```





```
clear,clc,clf;
x = linspace(-10,10,50);
y = linspace(-10,10,50);
[X,Y] = ndgrid(x,y);
Z=(X.^2.*Y.^2-2*X.*Y-
5)./(X.^2.+Y.^2+1);
plot3d(x,y,Z)
plot3d1(x,y,Z)
surf(x,y,Z)
```

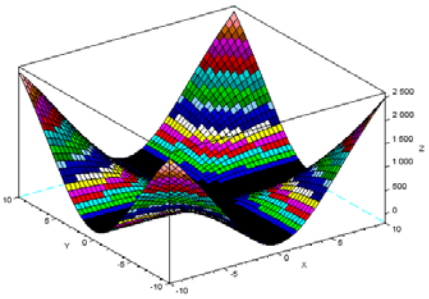


Рис.35 График функции $z(x,y)=\frac{x^2y^2-2xy-5}{x^2+y^2+1}$

Пример 6. Построить график функции $z(x,y)=(2x^2-y^2)e^{(-x^2-0.5y^2)}$ в области

[-2≤ x ≤ 2], [-3≤ y ≤ 3].

```
clear,clc,clf;
x=linspace(-2,2,30); // Linear
spacing
y=linspace(-3,3,30);
[X,Y]=meshgrid(x,y); // Surface
mesh
Z=(2*X.^2-Y.^2).*exp(-X.^2-
0.5*Y.^2);
surf(X,Y,Z) //
```

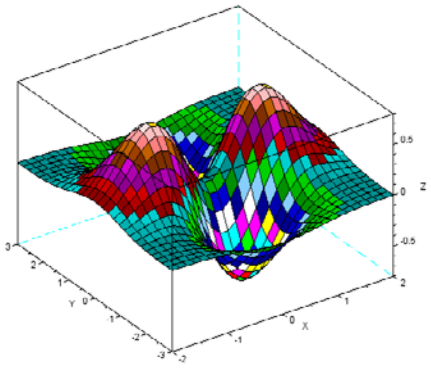


Рис.36 График функции $z(x,y) = (2x^2 - y^2)e^{(-x^2-0.5y^2)}$

Пример 7. Построить контурный график функции $z(x,y) = (2x^2 - y^2)e^{(-x^2-0.5y^2)}$ в области [-2≤ x ≤ 2], [-3≤ y ≤ 3].

```
clear,clc,clf;
x=linspace(-2,2,30); // Linear
spacing
y=linspace(-3,3,30);
[X,Y]=meshgrid(x,y); // Surface
mesh
Z=(2*X.^2-Y.^2).*exp(-X.^2-
0.5*Y.^2);
-->contour(x,y,Z,10)
```

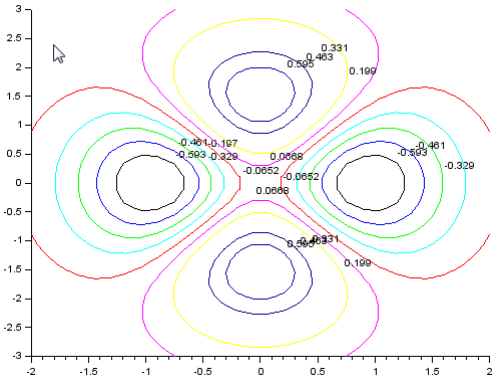


Рис.37 Контурный график функции $z(x,y) = (2x^2 - y^2)e^{(-x^2-0.5y^2)}$

Пример 8. Построить кривую, заданную параметрически, где *0≤ t ≤5π, x=sin t, y=cos 2t, z=t²/10*:

```
-->t=[0:0.1:5*%pi]';
-->param3d(sin(t),cos(2*t),t.^2/10,20,30)
```

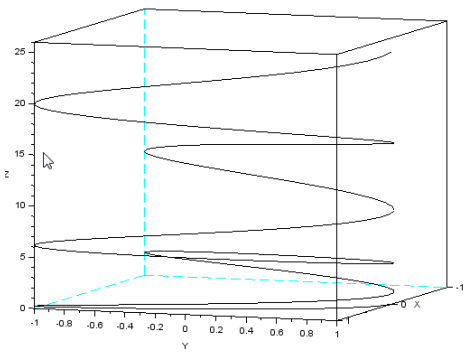


Рис.38 График параметрической линии

Рассмотрим применение функции [Xf,Yf,Zf]=eval3dp(fun,p1,p2) для создания прямоугольной сетки 3d графика параметрически заданной поверхности.

Пример 9. Построить конус, заданный параметрически, где радиус 0≤ ro ≤5, высота 1≤ p2 ≤5, угол -π≤φ≤π. Конус задаётся линейным увеличением радиуса ro, x и y:
x = ro_{p2}·cos(φ), y = ro_{p2}·sin(φ).

```
clear,clc,clf;
ro=[0,1,2,2.3,3,5];// радиус
function [x,y,z]=cone(p1,p2)// задаёт конус
 x=ro(1,p2).*cos(p1)
 y=ro(1,p2).*sin(p1)
 z=ro(1,p2).*ones(p1)
endfunction
[xv,yv,zv]=eval3dp(cone,linspace(-%pi,%pi,20),1:6);
plot3d(xv,yv,zv,theta=60,alpha=70)
```

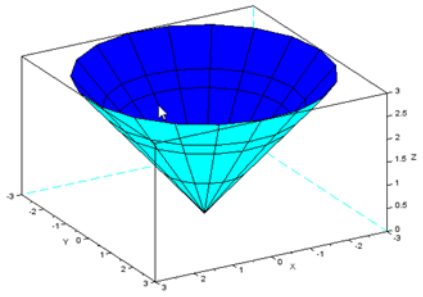


Рис.39 Конус

Пример 10. Построить график функции $z(x,y)= 7x^2-y^2$ в области [-2≤ x ≤ 2], [-3≤ y ≤ 3].

```
-->[x y]=meshgrid(-2:0.1:2,-3:0.1:3);
-->z=7*x.^2-y.^2;
-->mesh(x,y,z);
```

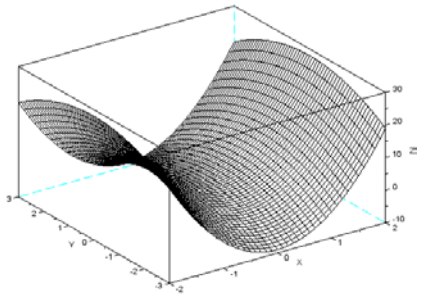


Рис.40 График функции $z(x,y)= 7x^2-y^2$

***Задачи*.**

Вычислить 3 значений функции на заданном отрезке. Вывести на экран значения аргумента и значения функции. Построить график заданной функции.

1. Функция $f(x,y)=\frac{\sin x\cos y}{x+\cos x}$, на отрезке x∈ [0, π],y∈[0, π]

2. Функция $f(x,y)=x\sin xy+\frac{e^{-x}-e^{y}}{e^{-y}+e^{x}}$, на отрезке x∈ [0, π],y∈[0, 1]

3. Функция $f(x,y)=\left(\frac{e^{-x}-e^{y}}{2}\right)^2+\left(\frac{e^{-y}+e^{x}}{2}\right)^2$, на отрезке x,y∈[-3, 3]

4. Функция $f(x,y)=4xy-\cos^2(\frac{e^{-y}-e^{x}}{e^{-x}+e^{y}})$, на отрезке x,y∈[0, 1]

5. Функция $f(x,y)=e^{\frac{y+5}{x^2}}-\sqrt{x+y}$, на отрезке x,y∈[0, 3]

6. Функция $f(x,y)=\mathrm{tg}\left(\sqrt{\frac{2x-y}{x+y+1}}\right)+\sqrt{y+x}$, на отрезке x,y∈ [0.7, 4]

8. Функция $f(x,y)=\frac{\sqrt[3]{x^2+1}}{\sqrt{|y|+1}}\frac{xy}{1+sin^2 x}$, на отрезке x,y∈[-π/2, π/2]

9. Функция $f(x,y)=\frac{x+2y}{1+\sqrt[3]{xy+1}}$., на отрезке x,y∈[10,100]

10. Функция $f(x,y)=\frac{sin^2(2xy+\frac{\pi}{2})}{\sqrt{\frac{|xy|+1}{4}}}$, на отрезке x,y∈[-π/2, π/2]

11. Функция $f(x,y)=\log_2(\sin x+\sin y+1)\sqrt{e^{-\cos x}}$, на отрезке x,y∈[0, π/2]

12. Функция $f(x,y)=|\sin x+\cos y|+x+y$, на отрезке x,y∈[0, π/2]

13. Функция $f(x,y)=\frac{sh\,x+ch\,y}{th\,xy+5}$, на отрезке x,y∈[-3,3]

14. Функция $f(x,y)=4cos^2\left(\frac{x+y+\frac{\pi}{3}}{xy+2}\right)$, на отрезке x,y∈[0, 1]

15. Функция $f(x,y)=x\,(\sin y+\cos x)\frac{y}{1+sin^2\,y}$, на отрезке x,y∈[0, π/2]

16. Функция $f(x,y)=5x^2-2y^2+xy$, на отрезке x,y∈[0, 3]

17. Функция $f(x,y)=(\sin x^2+\cos y^2)^y$, на отрезке x,y∈[-1, 1]

18. Функция $f(x,y)=(xy)^{x+y}$, на отрезке x,y∈[-1, 1]

19. Функция $f(x,y)=e^{\frac{-(x^2+y^2)}{2}}$, на отрезке x,y∈[-2, 2]

20 Функция $f(x,y)=(1+xy)(5-y)$, на отрезке x,y∈[0, 4]

21. Функция $f(x,y)=(2x^2-y^2+1)e^{(-x^2-y^2)}$, на отрезке [-2≤ x ≤ 2], [-3≤ y ≤ 3]

22. Функция $f(x,y)=(xy+x+y)^{x+y}$, на отрезке x,y∈[-1, 1]

2.3 Вычисления с массивами данных (вектора и матрицы).

Пример 1. Сформировать массив x=1, 3, 5, 7, 9, 11 и вычислить массив элементов ***y=x²***.

```
-->x=[1,3,5,7,9,11]
 x  =
   1.   3.   5.   7.   9.   11.
-->y=x.^2
 y  =
   1.   9.   25.   49.   81.   121.
```

Пример 2. Сформировать массив x=0, 10, 15, 20, 25, 30 (задан в градусах) и вычислить массив элементов: *y=sin(x), y=cos(x), y=sin(x)cos(x).*

```
-->x=0:5:30
 x  =
   0.   5.   10.   15.   20.   25.   30.
-->y=sind(x)
```

```
y  =
   0.   0.0871557   0.1736482   0.2588190   0.3420201   0.4226183   0.5
-->y=cosd(x)
y  =
   1.   0.9961947   0.9848078   0.9659258   0.9396926   0.9063078   0.8660254
-->y=sind(x).*cosd(x)
y  =
   0.   0.0868241   0.1710101   0.25   0.3213938   0.3830222   0.4330127
```

Пример 3. Сформировать массив в виде вектор- столбца x=5, 10, 15, 20, 25, 30 (задан в градусах) и вычислить массив элементов: *y=tg(x), y=ctg(x), y=tg(x)ctg(x).*

```
-->x=5:5:30
 x  =
   5. 10. 15. 20. 25. 30.
>x=x'//транспонирование
 x  =
   5.
   10.
   15.
   20.
   25.
   30.
```

```
-->y=tand(x)
 y  =
   0.0874887
   0.1763270
   0.2679492
   0.3639702
   0.4663077
   0.5773503
```

```
-->y=cotd(x)
 y  =
   11.430052
   5.6712818
   3.7320508
   2.7474774
   2.1445069
   1.7320508
```

```
-->y=tand(x).*cotd(x)
 y  =
   1.
   1.
   1.
   1.
   1.
   1.
```

Пример 4. Сформировать массив в виде матрицы, если заданы три вектор - столбца **x**=0.5, 1, 1.5, 2, 2.5, 3, **$y=x^2$, z=x/2**, и вычислить сумму элементов матрицы и сумму элементов каждой строки.

```
-->x=0.5:0.5:3;
-->y=x.^2;
-->z=x/2;
-->a=[x;y;z]
 a  =
0.5   1.   1.5   2.  2.5   3.
0.25  1.   2.25  4.  6.25  9.
0.25  0.5  0.75  1.  1.25  1.5
```

```
-->sum(a(1,:))
 ans  =
   10.5
```

```
-->sum(a(2,:))
 ans  =
   22.75
```

```
-->sum(a(3,:))
 ans  =
   5.25
```

```
-->sum(a)
ans  =
   38.5
```

Пример 5. Сформировать матрицу размерности [2x6] и [4x3], если задана матрица c=$\begin{pmatrix}1 & 2 & 3\\4 & 5 & 6\end{pmatrix}$

```
->c=[1 2 3;4 5 6]
 c  =
   1.   2.   3.
   4.   5.   6.
-->a=[c c]
 a  =
   1.   2.   3.   1.   2.   3.
   4.   5.   6.   4.   5.   6.
```

```
-->z=[c; c]
 z  =
   1.    2.    3.
   4.    5.    6.
   1.    2.    3.
   4.    5.    6.
```

Пример 6. Сформировать матрицу **A** размерности [6x6] и выделить из неё подматрицу **B** размерности [3x3]. Удалить из матрицы **A** первые два столбца.

```
 -->x1=(1:6)';
-->x2=2*x1;
 -->x3=0.5*x1;
 -->x4=3*x1;
 -->x5=3.5*x1;
-->x6=4*x1;
-->A=[x1 x2 x3 x4 x5 x6]
 A  =
   1.    2.    0.5    3.    3.5    4.
   2.    4.    1.     6.    7.     8.
   3.    6.    1.5    9.    10.5   12.
   4.    8.    2.     12.   14.    16.
   5.    10.   2.5    15.   17.5   20.
   6.    12.   3.     18.   21.    24.
-->B=A(4:6,2:4)
 B  =
   8.    2.    12.
   10.   2.5   15.
   12.   3.    18.
-->A(:,1:2)=[]
 A  =
   0.5   3.    3.5    4.
   1.    6.    7.     8.
   1.5   9.    10.5   12.
   2.    12.   14.    16.
   2.5   15.   17.5   20.
   3.    18.   21.    24.
```

Пример 7. Вычислить значение выражения $\mathbf{2\cdot (A+B^T)^2 + 5\cdot A\cdot (B-A^T)}$, где $\boldsymbol{A} = \begin{pmatrix} 1 & 2 & -3 \\ 5 & -6 & 0 \\ 3 & 2 & 7 \end{pmatrix}$, $\boldsymbol{B} = \begin{pmatrix} 10 & -5 & 0 \\ 1 & -4 & 7 \\ 12 & 9 & 6 \end{pmatrix}$

```
-->clear
-->A=[1 2 -3;5 -6 0;3 2 7]
 A  =
   1.    2.  - 3.
   5.  - 6.    0.
   3.    2.    7.
-->B=[10 -5 0;1 -4 7;12 9 6]
 B  =
   10.  - 5.    0.
   1.   - 4.    7.
   12.    9.    6.
-->2*(A+B')^2+5*A*(B-A')
 ans  =
   106.    3.      536.
   309.    52.   - 171.
   794.    257.    524.
```

Пример 8. Решить матричные уравнения вида ***A·x=B***, ***x·A=B***, где ***A*** и ***B*** из задачи 7. Проверить полученное решение.

```
-->x=A\B
 x  =
   3.3469388  - 0.6530612    1.2653061
   2.622449     0.1224490  - 0.1122449
 - 0.4693878    1.5306122    0.3469388
-->A*x-B
 ans  =
   0.         - 8.882D-16    0.
   0.           0.           0.
   1.776D-15    0.         - 1.776D-15
-->x=B/A
 x  =
   1.25         1.4285714    0.5357143
 - 1.5          0.2857143    0.3571429
   3.3214286    0.3673469    2.2806122
```

Пример 9. Сформировать матрицу единиц, нулевую матрицу, единичную матрицу, многомерную матрицу.

```
->a=ones(2,3)
 a  =
   1.    1.    1.
   1.    1.    1.
-->z=zeros(2,3)
```

```
 z  =
   0.    0.    0.
   0.    0.    0.
-->o=eye(2,3)
 o  =
   1.    0.    0.
   0.    1.    0.

-->H=hypermat([2 3 3]);
-->H(:,:,1)=a;
-->H(:,:,2)=z;
-->H(:,:,3)=o;
-->H
 H  =
(:,:,1)
   1.    1.    1.
   1.    1.    1.
(:,:,2)
   0.    0.    0.
   0.    0.    0.
(:,:,3)
   1.    0.    0.
   0.    1.    0.
```

Пример 10. Определить размер для матрицы A из задачи 7, её максимальный и минимальный элемент, длину, сумму элементов.

```
-->A=[1 2 -3;5 -6 0;3 2 7];
-->max(A)
 ans  =
   7.
-->min(A)
 ans  =
 - 6.
-->size(A)
 ans  =
   3.    3.
-->length(A)
 ans  =
   9.
-->sum(A)
 ans  =
   11.
```

***Задачи*.**

1. Сформировать вектор **a** с 10 элементами из натуральных чисел от 1до 10 и вектор **b** с 6 элементами из натуральных чисел от 1до 6. Сформировать из векторов **a** и **b** вектор **c** из 16 элементов.

2. Сформировать вектор **a**=**a**(1:10) со значениями **a**=x-10, где 0≤x≤20. Определить его максимальный и минимальный элементы, вычислить сумму элементов.

3. Образовать вектор **c** -1,2,0,5,5,10,11,-12,9,3 и упорядочить его по возрастанию и убыванию.

4. Сформировать вектор **a** со значениями -1,2,0,5,5,10,11,-12,9,3 и переставить элементы этого вектора в обратном порядке. Результат записать в новый вектор.

5. Используя функцию **mean()** вычислить среднее арифметическое векторов: **x**(-1,2,0,5,5,10,11,-12,9,3) и **y**(4.4, 3.3, 2.2, 1.1).

6. У вектора **a**=sin(1:10) вычислить среднее арифметическое элементов.
Заменить минимальный элемент вектора на максимальный.

7. Вычислить произведение элементов двух векторов **x**=sin(1:10), **y**=cos(1:10). У полученного вектора найти сумму элементов.

8. Заменить мнимой единицей минимальный элемент вектора **x** и максимальный элемент вектора **y** из задания 7.

9. По заданному вектору -1,2,0,5,5,10,11,-12,9,3 определить номер элемента с наименьшим отклонением от среднего арифметического всех элементов векторов.

10. Задать вектор с 10 элементами, состоящий из случайных значений командой **rand().** Найти среднее арифметическое элементов заданного вектора и заменить последний элемент этим значением.

11. Задать матрицу размерности (3x3), состоящую из случайных значений командой **rand()**. Вычислить максимальное значение среди элементов главной диагонали заданной матрицы, используя команду **diag().**

12. Переставить первый столбец квадратной матрицы из задания 11 с последним. Заменить первую строку элементами главной диагонали.

13. Вычислить значение суммы всех элементов матрицы **x**=**rand**(5,5) кроме элементов главной диагонали.

14. Заменить минимальный элемент матрицы **x**=**rand**(10,10,'normal') средним значением всех его элементов.

15. Сформировать матрицу из векторов x=sin(1:5), y=cos(1:5). Заменить элемент с индексами 2,1 произведением всех элементов матрицы.

16. Сформировать матрицу **x=rand**(5,5). Используя команду **Matplot()**, графически отобразить матрицу. Обратить внимание, что значения элементов матрицы должны превышать единицу.

17.Сформировать матрицу размером (5x10), используя команду **meshgrid()**. Найти значение суммы всех элементов.

18. Для матрицы $a = \begin{pmatrix} 1 & 2 & 6 \\ 4 & 2 & 3 \\ -5 & -1 & 0 \end{pmatrix}$ найти обратную матрицу, используя команду **inv()**. Проверить правильность вычисления обратной матрицы.

19. Для матрицы $a = \begin{pmatrix} 1 & 3 & -8 \\ 4 & 0 & 7 \\ -3 & -1 & 10 \\ 6 & -8 & 9 \end{pmatrix}$ найти элементы главной диагонали, на первой диагонали выше главной и ниже главной диагонали. Используйте команду **diag()**.

20. Элементы матрицы $a = \begin{pmatrix} 1 & 3 & -8 \\ 4 & 0 & 7 \\ -3 & -1 & 10 \\ 6 & -8 & 9 \end{pmatrix}$ отсортировать по возрастанию и по убыванию, используя команду **gsort()**.

2.4 Операции с полиномами.

Пример 1. Создать полином, имеющий корни 3 и 2 с символьной переменной x.

1-й способ:

```
-->p1 = poly([3 2], 'x') // по умолчанию задаются корни полинома
 p1  =
                2
   6 - 5x + x
-->p1 = poly([3 2], 'x','r') //явно указано, что заданы корни
 p1  =
                2
   6 - 5x + x
->type(p1) // тип переменной p1
 ans  =
   2.
```

2-й способ:

```
-->x=%s; // %s системная переменная, имеющая тип полином
-->p1=6-5*x+x^2
 p1  =
                2
   6 - 5s + s
-->type(p1)
```

```
 ans  =
    2.
```

3-й способ:

```
-->x=poly([0],'x') // задание типа для переменной x
 x  =
    x
-->type(x)
 ans  =
    2.
-->p1=6-5*x+x^2
 p1  =
                2
    6 - 5x + x
```

Пример 2. Создать полиномы, имеющие коэффициенты $p_1=16-10x+3x^2$, $p_2=1+3x+x^3-5x^5$ с символьной переменной x.

```
->p1 = poly([16 -10 3], 'x', 'c')
 p1  =
              2
    16 - 10x + 3x
-->p2 = poly([1 3 0 1 0 -5], 'x', 'c')
 p2  =
                3    5
    1 + 3x + x - 5x
```

Пример 3. Создать два полинома $p_1=1-x+5x^2$, $p_2=1+3x+x^2$. Найти сумму, разность, произведение и частное этих полиномов.

```
-->p1 = poly([1 -1 5], 'x', 'c');
-->p2 = poly([1 3 1], 'x', 'c');
-->p3=p1+p2
 p3  =
                 2
    2 + 2x + 6x
-->p4=p1-p2
 p4  =
              2
  - 4x + 4x
->p5=p1*p2
 p5  =
                 2        3     4
    1 + 2x + 3x + 14x + 5x
-->typeof(p5)
```

```
ans  =
polynomial
 -> p6=p1/p2
p6  =
                2
   1 - x + 5x
   ----------
                2
   1 + 3x + x
-->typeof(p6)
ans  =
 rational
```

Пример 4. Создать два полинома $p_1=1-x+5x^2$, $p_2=1+3x+x^2$. Проверить их равенство, найти коэффициенты p_1 и его производную.

```
->p1==p2 // операция сравнения полиномов. Ответ не равны
 ans  =
  F
-->coeff(p1) //коэффициенты полинома
 ans  =
    1.  - 1.    5.
-->derivat(p1) // производная полинома
 ans  =
  - 1 + 10x
```

Пример 5. Задать полином y с корнями x1= 2, x2= -3. Вычислить значения полинома на отрезке [0≤x≤10] и построить график зависимости y(x).

```
-->y=poly([2 -3],'x')
 y  =
                2
  - 6 + x + x
-->x=0:10;
-->y=horner(y,x)
-->plot(x,y,'dk')
-->plot(x,y)
```

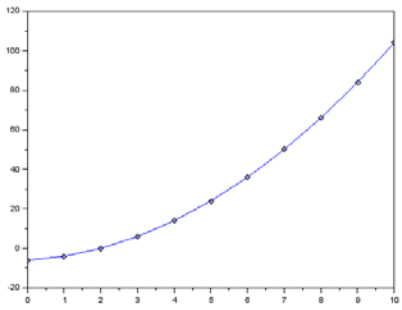


Рис.41 График полинома $y= -6 + x + x^2$

Пример 6. Задать полином с числителем в виде $1+x$ и знаменателем вида $1+x^2$. Определить числитель и знаменатель. Найти производную от этого полинома.

```
-->x=%s;
-->p=(1+x)/(1+x^2)
 p  =
       1 + s
       -----
            2
      1 + s
->numer(p) //числитель
 ans  =
   1 + s
-->denom(p) //знаменатель
 ans  =
          2
   1 + s
-->derivat(p) // производная
 ans  =
              2
   1 - 2s - s
   ----------
           2    4
   1 + 2s + s
```

Пример 7. Дан полином вида $p = \frac{2+3x+x^2}{-2-x+x^2}$.Упростить вид полинома.

```
-->x=poly(0,'x');
-->[n, d]=simp(2+3*x+x^2,-2-x+x^2)
 d  =
 - 2 + x
 n  =
   2 + x
```

Пример 8. Найти корни полинома вида $p = 6 - 5 \cdot x + x^2$.

```
-->x=poly(0,'x');
-->p = 6-5*x + x^2;
-->roots(p)
 ans  =
   3.
   2.
```

Пример 9. Сформировать рациональную матрицу из полиномов вида $p = \begin{pmatrix} 1 & \frac{1}{s} \\ 1 & 1 \end{pmatrix}$, найти её обратную, числитель и знаменатель обратной матрицы.

```
-->r=1/%s;
-->a=[1,r;1,1]
 a  =
   !1     1!
   !-     - !
   !1     s !
   !        !
   !1     1 !
   ! -    - !
   !1     1!

-->b=inv(a) // обратная матрица
 b  =

!    s      - 1      !
!  -----     -----   !
! - 1 + s   - 1 + s !
!                    !
! - s         s      !
!  -----     -----   !
! - 1 + s   - 1 + s !
-->b.num      // числитель
 ans  =
   s   - 1
 - s     s
-->b.den   // знаменатель
 ans  =
 - 1 + s   - 1 + s
 - 1 + s   - 1 + s
```

Пример 10. Задать полином вида $p = 1 - 5x + 3x^3 + 2x^4$. Найти его корни.

```
-->p=poly([1 -5 0 3 2],'x','c')
p  =
                 3    4
   1 - 5x + 3x + 2x
-->roots(p)
 ans  =
 - 1.3030302 + 0.9995932i
```

```
- 1.3030302 - 0.9995932i
  0.9000985
  0.2059620
```

Задачи.

1. Создать полином с символьной переменной x и корнями 2 и 4.

2. Создать полином $6 + 15x + 2x^2$ с символьной переменной x.

3. Найти корни полинома $6 – 5x + x^2$.

4. Создать полином $\frac{2x+4x^2+x^3}{6-20x+21x^2} + x^3 + 5x^4$. Возможно ли его дальнейшее упрощение?

5. Создать полином $6 + 15x + 2x^2+x^4$, вычислить его значения в диапазоне [-5≤x≤5]. Построить график зависимости *p(x).*

6. Создать полином $\frac{1-5x+3x^3+2x^4}{1-5x+2x^3}$, вычислить его значения в диапазоне [5≤x≤20]. Построить график зависимости p(x).

7. Задать два полинома p1 и p2 с корнями соответственно [1 -3 0 7 8 -9], [-10 -1.3 -7.5 0 0.9] Найти их сумму, разность, частное и произведение.

8. Создать полином вида: $- 120 + 274x - 225x^2 + 85x^3 - 15x^4 + x^5$. Вычислить значения этого полинома в 5 точках (0, -4, 5, 7.5, 10).

9. Пусть значения переменной x заданы в виде матрицы $x = \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix}$. Необходимо вычислить значения полинома вида $1 + 2x + 3x^2$ при этих значениях переменной x.

10. Задать два полинома p1 и p2 с корнями соответственно [10 -13 6 7 8], [-10 -1 -5 0.9]. Упростить вид полученного полинома p1/p2. Найти корни этого полинома.

11. Если в качестве первого аргумента функции poly задать квадратную матрицу, то она вернёт характеристический многочлен, связанный с матрицей. Пусть A - вещественная квадратная матрица размером n × n. Тогда применение функции poly даст многочлен det(A − xI), где I - единичная матрица, x символьная переменная типа полином. Пусть задана матрица $A = \begin{pmatrix} 2 & 3 \\ 6 & 7 \end{pmatrix}$, найти характеристический многочлен, его корни и используя функцию **spec()** найти собственные значения матрицы A.

12. Пусть задана матрица $A = \begin{pmatrix} 2 & 3 \\ 6 & 7 \end{pmatrix}$. Задать символьную переменную типа полином, и используя определение собственного значения матрицы, найти характеристический многочлен (используя при этом функцию **det()**).

13. Задать полином вида $p = 10 - 15x + 3x^5 + 2x^7$. Найти его корни.

14. Создать полином : $-1.2 + 2.7x + 22.5x^2 - 8.5x^4 - 7x^5$. Вычислить значения этого полинома в 3 точках (0, -4, 7.5).

15. Задать два полинома p1 и p2 с корнями соответственно [1 -3 0 7], [-10 -1 -7 0.9] Найти их сумму, разность, частное и произведение. Вычислить значения этого произведения полиномов в 3 точках (0, -4, 7.5).

16. Создать полином $\frac{1-5x+3x^3}{1-x+2x^3}$, вычислить его значения в диапазоне [3≤x≤10]. Построить график зависимости p(x).

17. Задан вектор коэффициентов полинома степени n=5 $a = (1\ \ 2\ 3\ 4\ 5)$. Задать полином и вычислить его производную.

18. Задан вектор коэффициентов полинома степени n=4 a=(-1 3 -4 5). Задать полином, вычислить его значения в диапазоне [0≤x≤10]. Построить график полученного полинома.

19. Задан полином $p = \frac{x}{x^2-3x-4}$. Вычислить его значение при x=1. Найти производную.

20. Сформировать рациональную матрицу из полиномов вида $p = \begin{pmatrix} 2s & \frac{1}{s} \\ 1+s & s \end{pmatrix}$, найти её обратную, числитель и знаменатель обратной матрицы.

Решение нелинейных уравнений

3

Задача нахождения корней нелинейных уравнения с одной неизвестной величиной вида $f(x) = 0$ (где - $f(x)$ –дифференцируемая функция) может быть как отдельной задачей, так и частью другой задачи. Такие уравнения обычно разбивают на два класса – алгебраические и трансцендентные. Напомним, что алгебраическим уравнением называется уравнение, содержащие только алгебраические функции переменных (целых, рациональных, иррациональных). Те же уравнения, которые содержат тригонометрические, показательные или логарифмические функции, называются трансцендентными. Необходимо отметить, что не всегда существует точное аналитическое решение уравнения. Поэтому на практике достаточно найти корни уравнения с заданной степенью точности и для этого используются численные методы. Задача нахождения корней заключается в определении точек пересечения графика функции $f(x)$ с осью абцисс x. *Основная идея, которая используется для поиска корней уравнения численными методами, заключается в том, что процесс поиска разбивается на два этапа:*

1) отделение корней, т. е. разбиение области определения функции f(x) на отрезки, в каждом из которых содержится один и только один корень уравнения;

2) уточнение приближенных корней, т. е. получение заданной степени точности.

В процессе отделения корней на заданной области используется свойство непрерывности функции f (x), когда выполняется условие $f(a) \cdot f(b) < 0$, где a и b концы отрезка на котором ищутся корни. В случае выполнения этого условия, на отрезке a и b есть по крайней мере одна точка, в которой $f(x) = 0$. При этом, если выполняется условие, что функция $f(x)$ имеет первую производную, не изменяющую знака (монотонна) на этом отрезке, то этот корень единственный.

На втором этапе происходит последовательное уточнение значения корня, полученного на первом этапе, до получения значения с заданной точностью. Каждый шаг поиска называется итерацией. В результате каждой итерации находится приближенное значение корня $x_0, x_1, x_2, \ldots, x_n$, где n- номер итерации. Если с ростом количества итераций n приближенные значения корня стремятся к истинному значению корня, то говорят, что итерационный процесс сходится.

Локализацию (отделение) корней можно проводить *графическим или аналитическим методом.* В графическом методе строится график функции $f(x)$ и ищется точка пересечения кривой с осью абцисс x, что позволяет определить корни уравнения. В случае аналитического метода отделения корней уравнения применяется метод сканирования с постоянным или переменным шагом. В этом методе, заданную область определения функции [a; b] разбивают на n отрезков, точками x_i, расположенными на расстоянии $h=(b-a)/n$ одна от другой. После вычисления значений $f(x_i)$, происходит проверка выполнения условия $f(x_{i-1}) \cdot f(x_i) < 0$ на отрезке $[x_{i-1}; x_i]$. Если данное условие выполняется, то координаты этих отрезков фиксируются и в дальнейшем используются на этапе уточнения корней уравнения. Модификацией этого метода является метод сканирования с переменным шагом, в котором проводится сужение интервала, на котором находится корень. Для уточнения корня на выделенном участке, где этот корень является единственным, применяются следующие методы: метод проб, метод половинного деления (частный случай метода проб), метод хорд, метод касательных (Ньютона), метод итераций. С сущностью этих методов можно ознакомиться по литературе [1-4].

В Scilab вычисление корней нелинейных уравнений с одним неизвестным можно осуществить с использованием функции **roots()**, которая позволяет вычислить корни алгебраических уравнений, и более мощной функции **fsolve()**, предназначенной для решения уравнений с различными видами функций. Кроме этого, используя язык программирования Scilab, возможна реализация перечисленных методов нахождения корней в виде программ, написанных пользователем.

3.1 Способы отделения корней уравнения

Пример 1. Графическим методом отделить корни уравнения: $100 + 2x - 9x^2 = 0$

```
->x=-8:8;
-->y=100+2.*x-9.*x.^2;
-->plot(x,y)
-->a=gca();
-->a.y_location="origin";//положение оси y
-->a.x_location="origin";//положение оси x
-->roots(100+2*%s-9*%s^2)//вычисление
//корней
ans  =
    3.4462958
  - 3.2240736
```

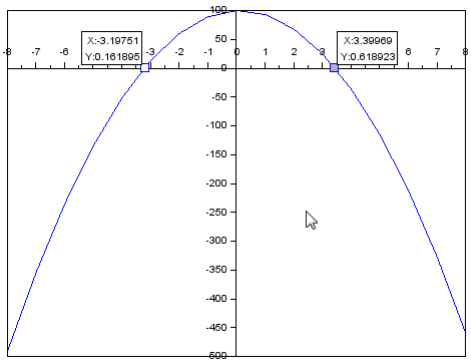


Рис.42 График полинома $y= 100 +2x - 9x^2$ с его корнями

Значения любой точки графика в графическом окне можно получить, включив пункт "Включить управление подсказками данных" в меню "Правка". Необходимо также заметить, что точность отделения корней графическим методом можно увеличить, уменьшив шаг между соседними значениями по оси x.

Пример 2. Графическим методом отделить корни уравнения: *1+2x-sin(x)=0*

```
-->x=-8:0.01:8;
-->y=1+2.*x-sin(x);
-->a=gca();
-->a.x_location="origin";
-->a.y_location="origin";
-->plot(x,y)
```

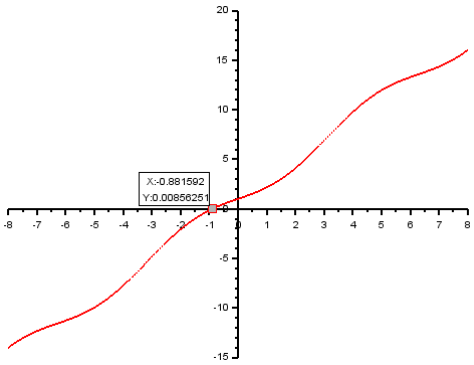


Рис.43 График функции *y= 1 +2 x – sin(x)*

Следует обратить внимание на последовательность выполнения команд, приведшую к отсутствию рамки у графика и на частоту разбивки оси x и связанную с этим точность определения точки пересечения функции с осью x

Пример 3. Графическим методом отделить корни уравнения: sin(x)=0.5 на отрезке [-4π,4π].

```
-->x=-4*%pi:0.1*%pi:4*%pi;
-->y=sin(x)-0.5;
-->plot(x,y)
-->a=gca();
-->a.y_location="origin";
-->a.x_location="origin";
```

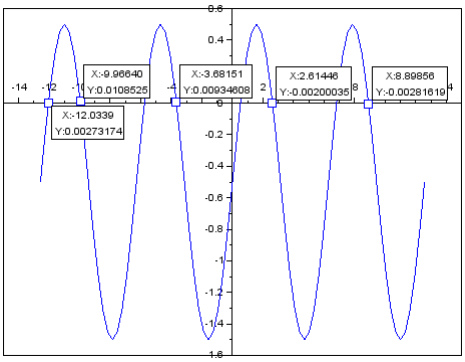


Рис.44 График функции *y= sin(x) - 0.5*

Пример 4. Графическим методом отделить корни уравнения: *sin(x)=0.5·cos(2x)-1* на отрезке [-4π,4π].

```
-->x=-4*%pi:0.1*%pi:4*%pi;
-->y1=sin(x);
-->y2=0.5*cos(x)-1;
-->plot(x,y1)
-->plot(x,y2,'-r')
-->hl=legend(['sin(x)'],%f)//легенда
```

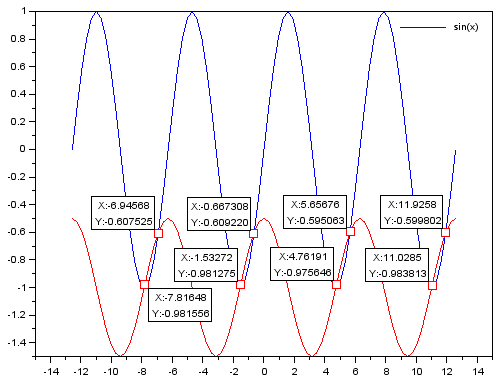


Рис.45 Графики функций *y1= sin(x), y2=0.5·cos(2x)-1*

Точки пересечения графиков y1 и y2 являются корнями заданного уравнения.

Пример 5. Методом сканирования с постоянным шагом отделить корни в уравнении $x^2-4=0$ на отрезке [-5,5].

Предварительные замечания. На этом простом примере продемонстрируем, что работа алгоритма может зависеть от шага разбиения или задания отрезка. Проблема заключается в том, что точки разбиения отрезка могут совпасть с корнями уравнения. Построим график функции для наглядности. И напишем небольшую программу в SciNotes, реализующую алгоритм отделения корней уравнения.

```
deff('y=fun(x)','y=x.^2-4')//определение функции
a=-5.;b=5.;n=10;//отрезок, число разбиений
dx=(b-a)/n;x=a:dx:b;//шаг разбиения, точки разб.
[m ,k]=size(x);//размерность вектора x
f=fun(x);//значения функции в точках x
for i=1:k-1
   p=f(i).*f(i+1);
if p<0 then disp(i,i+1)//проверка условия f_i*f_i+1<0
else disp('корней нет')
end
end
```

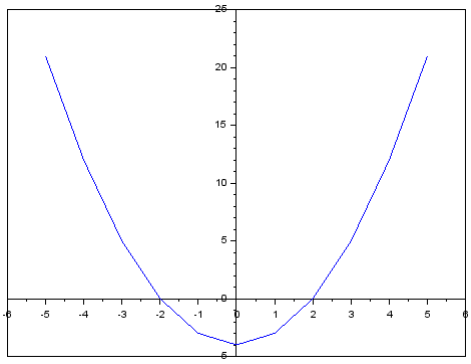


Рис.46 График функции $y=x^2-4$

Запустим программу на выполнение и получим ответ. Как видно, в каждом цикле происходит проверка условия f(x)f(x+dx)<0, и оно не выполняется. Посмотрим, почему это происходит. Выведем значения x, x+dx и f(x)f(x+dx).

```
корней нет
корней нет
корней нет
корней нет
корней нет
корней нет
корней нет
корней нет
корней нет
корней нет
```

```
-->x
 x  =
 - 5.  - 4.  - 3.  - 2.  - 1.   0.   1.   2.   3.   4.   5.
-->x+dx
 - 4.  - 3.  - 2.  - 1.   0.   1.   2.   3.   4.   5.   6.
-->fun(x).*fun(x+dx)
 ans  =
   252.   60.   0.   0.   12.   12.   0.   0.   60.   252.   672.
```

Как видно, проблема возникла из-за того, что некоторые точки разбиения оси x совпали с корнями уравнения. Чтобы избежать такой ситуации, необходимо включить предварительное условие проверки наличия корней в точках разбиения. Модифицируем нашу программу. После этого, изменим количество разбиений на n=11, и получим интервалы, в которых находятся корни.

```
deff('y=fun(x)','y=x.^2-4')
a=-5.;b=5.;n=10;
dx=(b-a)/n;x=a:dx:b;disp(n, 'n=')
[m ,k]=size(x);
f=fun(x);
for i=1:k-1
    p=f(i).*f(i+1);
    if f(i)==0 //проверка корней
       disp(x(i))
       end
if p<0 then disp(x(i+1),x(i))
    else disp('корней нет')
  end
end
```

```
n=10
корней нет
 корней нет
 корней нет
 корней нет
  - 2.
 корней нет
 корней нет
 корней нет
 корней нет
   2.
 корней нет
 корней нет
 корней нет
```

```
n=  11.
 корней нет
 корней нет
 корней нет
  - 2.2727273
 - 1.3636364
корней нет
 корней нет
 корней нет
1.3636364
2.2727273
 корней нет
 корней нет
```

Пример 6. Методом сканирования с постоянным шагом отделить корни в уравнении *sin(x)-0.5=0* на отрезке [-4π,π].

Выберем интервал разбиения n=20. Построим график, чтобы проверить работу программы, которую рассматривали в задаче 5, и сравним полученные результаты.

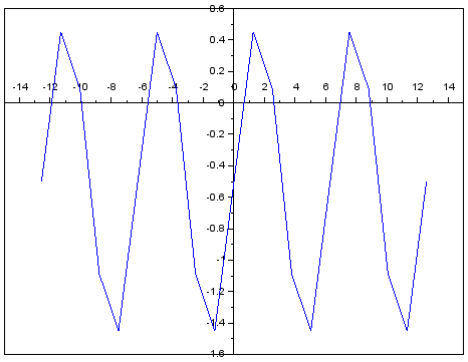


Рис.47 График функции *y= sin(x) - 0.5*

```
deff('y=fun(x)','y=sin(x)-0.5')
a=-4*%pi;b=4*%pi; n=20;
dx=(b-a)/n;x=a:dx:b;disp(n, 'n=' )
[m ,k]=size(x);f=fun(x);
for i=1:k-1
p=f(i).*f(i+1);
   if f(i)==0
disp(x(i))
      end
if p<0 then disp(x(i),x(i+1))
    else disp('корней нет')
 end
end
```

```
n=  20.
 - 11.309734   - 12.566371
корней нет
 - 8.7964594   - 10.053096
корней нет
корней нет
- 5.0265482    - 6.2831853
корней нет
 - 2.5132741    - 3.7699112
корней нет
корней нет
   1.2566371   0.
корней нет
 3.7699112     2.5132741
корней нет
корней нет
7.5398224     6.2831853
корней нет
10.053096     8.7964594
корней нет
```

Пример 7. Методом сканирования с постоянным шагом отделить корни уравнения:

1+2x-sin(x)=0 на отрезке [-8,8]. Сравнить с результатами, полученными в задаче 2.

```
n=   10.
 корней нет
 корней нет
 корней нет
 корней нет
  0.
- 1.6
 корней нет
 корней нет
 корней нет
 корней нет
 корней нет
```

```
deff('y=fun(x)','y=1+2.*x-sin(x)')
a=-8;b=8; n=10;
dx=(b-a)/n;x=a:dx:b;disp(n, 'n=' )
[m ,k]=size(x);
f=fun(x);
for i=1:k-1
    p=f(i).*f(i+1);
    if f(i)==0
       disp(x(i))
       end
if p<0 then disp(x(i),x(i+1))
    else disp('корней нет')
 end
end
```

Пример 8. Методом сканирования с постоянным шагом отделить корни уравнения:

$100 + 2x - 9x^2 = 0$. Задать количество шагов разбиения h=20, 200, 2000. Сравнить с результатами задачи 1.

```
deff('y=fun(x)','y=100+2*x-9*x.^2')
a=-8.;b=8.;n=2000;
dx=(b-a)/n;x=a:dx:b;
[m ,k]=size(x);
f=fun(x);
[i]=(find(f==0));
t=isempty(i);
if t==%F then xx=x(i)
    disp( xx,'xx=')
    end
p=fun(x).*fun(x+dx);
[j]=find(p<=0);
xxa=x(j);
 [nn,kk]=size(xxa);
 xxb=xxa+dx;
```

```
->xxa //начало отрезка
 xxa  =
  - 4.    3.2                 n=20
-->xxb //конец отрезка
xxb  =
  - 3.2    4.

-->xxa
 xxa  =
  - 3.28   3.44              n=200
-->xxb
xxb  =
  - 3.2     3.52

-->xxa
 xxa  =
 - 3.232   3.44           n=2000
 -->xxb
 xxb  =
  - 3.224    3.448
```

Пример 9. Методом сканирования с постоянным шагом отделить корни уравнения $100 + 2x - 9x^2 = 0$. Уточнить отрезки нахождения корней, модифицировав алгоритм таким образом, чтобы найденный отрезок, в котором находится корень, можно было бы разбить с более мелким шагом.

```
deff('y=fun(x)','y=100+2*x-9*x.^2')
a=-8.;b=8.;n=11;eps=0.1;ne=1/eps;//eps относит.
точность
dx=(b-a)/n;x=a:dx:b;
[m ,k]=size(x);//размерность вектора x
f=fun(x);
[i]=(find(f==0));//поиск нулевых значений функции
t=isempty(i);//поиск пустого элемента вектора
if t==%F then xx=x(i)// корни уравнения, если функция
//равна 0
   disp( xx,'xx=')
   end
p=fun(x).*fun(x+dx);
[j]=find(p<=0);//поиск знакопеременного отрезка
xxa=x(j);//значение начала отрезка
xxb=xxa+dx;//значение конца отрезка
 [nn,kk]=size(xxa);
  ddx=dx/100;//шаг разбиения отрезка
 for j=1:kk
 for i=1:100
   xn(i,j)=xxa(j)+i*ddx //интервал вычисления функции
 end
end
p1=fun(xn(:,1)).*fun(xn(:,1)+ddx);
p2=fun(xn(:,2)).*fun(xn(:,2)+ddx);
[j1]=find(p1<=0);
[j2]=find(p2<=0);
xxa1=xn(j1,1);xxa2=xn(j2,2);
```

Немодифицированный алгоритм

```
-->xxa
 xxa  =
  - 4.0909091    3.1818182
-->xxb
 xxb  =
  - 3.1818182    4.0909091
```

Модифицированный алгоритм

```
->xxa1
 xxa1  =
  - 3.2272727
-->xxa2
 xxa2  =
    3.4454545
-->xxa1+ddx
 ans  =
  - 3.2181818
-->xxa2+ddx
 ans  =
    3.4545455
```

Пример 10. Отделить корни уравнения sin(x)+cos(x)-0.5=0 методом сканирования с постоянным шагом на отрезке [-4π,4π].

```
-->xxa
 xxa  =
  - 11.309734  - 7.5398224  - 5.0265482  - 1.2566371    1.2566371    5.0265482
7.5398224    11.309734
-->xxb
 xxb  =
  - 10.053096  - 6.2831853  - 3.7699112    0.    2.5132741    6.2831853
8.7964594    12.566371
```

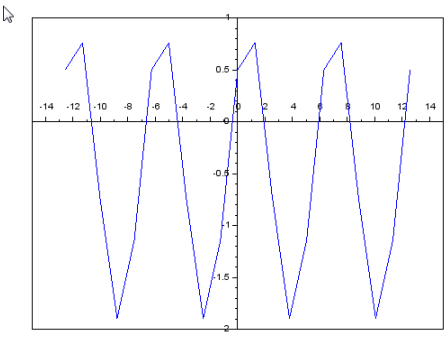


Рис.48 График функции *y= sin(x)+cos(x)-0.5*

Задачи.

1. Отделить корни уравнения $e^x-10x=0$ графическим методом.

2. Графическим методом отделить корни уравнения $sin(x)=0.5\cdot cos(x)-0.1\cdot e^{-x}$ на отрезке [-5,5].

3. Графическим методом отделить корни уравнения $e^x=cos(x)-sin(x)$ на отрезке $[-4\pi,4\pi]$.

4. Графическим методом отделить корни уравнения $-3+9x-6x^2+x^3=0$. Провести отделение с шагом разбиения отрезка x на 10, 100, 1000 частей.

5. Графическим методом отделить корни уравнения $cos(x)-x\cdot e^x=0$, где $0\le x\le 2$.

6. Графическим методом отделить корни уравнения $0.3\cdot x^2+cos(x)-|x|=0$ на интервале [0, 5].

7. Графическим методом отделить корни уравнения $(x-5)^2\cdot log_2 x-1.2=0$ на интервале [1, 4].Провести отделение с шагом разбиения отрезка x на 10 и 100 частей.

8. Графическим методом отделить корни уравнения $x^2+6\cdot e^{0.15x}=0$

9. Графическим методом отделить корни уравнения $x\cdot lg(x)-1.2=0$.

10. Графическим методом отделить корни уравнения $lg(2+x)+2x-3=0$.

11. Методом сканирования с постоянным шагом отделить корни уравнения $cos\,x-x+1=0$ на отрезке $[-2\pi,2\pi]$.

12. Методом сканирования с постоянным шагом отделить корни уравнения $x^3-0.1x^2+1.5x-1.5=0$

13. Методом сканирования с постоянным шагом отделить корни уравнения $0.5^x-3+(x+1)^2=0$ на отрезке [-2,2].

14. Методом сканирования с постоянным шагом отделить корни уравнения $x^2-ln(6+x)=0$ на отрезке [-2,2]. Уточнить отрезки нахождения корней, модифицировав алгоритм таким образом, чтобы найденный отрезок, в котором находится корень, можно было бы разбить с более мелким шагом.

15. Методом сканирования с постоянным шагом отделить корни уравнения $x+lg\,x-0.5=0$.

16. Методом сканирования с постоянным шагом отделить корни уравнения $x^3-8\cdot x^2+3=0$ на отрезке [-2,2].

17. Методом сканирования с постоянным шагом отделить корни уравнения $e^x-e^{-x}-2=0$ Задать количество шагов разбиения n=20, 200. Сравнить результаты.

18. Графическим методом и методом сканирования с постоянным шагом, используя один и тот же шаг разбиения для оси x, отделить корни уравнения $x^3-sin\,x=0$ на отрезке $[-2\pi,2\pi]$. Сравнить полученные отрезки.

19. Методом сканирования с постоянным шагом отделить корни уравнения $x^3 - 0.1x^2 + 0.4x - 1.5 = 0$.

20. Методом сканирования с постоянным шагом отделить корни уравнения $x^2 - sin5x = 0$ на отрезке $[-2\pi, 2\pi]$.

3.2 Уточнение корней нелинейных уравнений.

Пример 1. Отделить графически и найти методом половинного деления корни уравнения $x^3-2x^2-2x-1=0$. Выяснить, зависит ли точность определения корня от длины отрезка отделения корня.

```
deff('y=f(x)','y=x^3-2*x^2-2*x-1')
a=2;b=3; N = 100; eps = 1.e-5; //
if(abs(f(a)) < eps) then error ('корень в точке a')
abort
end
if(abs(f(b)) < eps) then error ('корень в точке b')
abort
end;
while (N > 0)
 c = (a+b)/2
if(abs(f(c)) < eps) then x = c;
return
end
if(f(a)*f(c)<0) then b=c;
else
a = c;
end
N = N - 1;
end
->x// значение корня
 x  =
      2.8311768
```

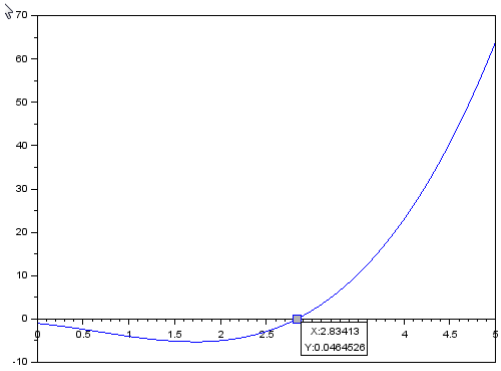


Рис.49 График функции $y= x^3-2x^2-2x-1$

Рассмотрим следующие отрезки, в которых находится корень уравнения: [-5,5], [2.5, 3], [2.7,3]. Найдём корень для всех отрезков.

```
->format(25)
-->x//  отрезок  [2,3]
 x  =
   2.831176757812500000000
-->x// отрезок [-5,5]
 x  =
 2.8311777114868164062500
 -->x// отрезок [2.5,3]
 x  =
   2.831176757812500000000
-->x// отрезок [2.7,3]
 x  =
            2.831176757812500000000
```

Как видно, точность определения корня слабо зависит от длины отрезка.

Пример 2. Отделить графически и найти методом половинного деления корни уравнения $x^2 - \sin 3x = 0$ на отрезке.

Вначале графически отделим отрезки, в которых находятся корни уравнения. Очевидно, можно сразу сказать, что x=0 тривиальный корень. Но для проверки работы алгоритма зададим оба отрезка [-0.5,0.5] и [0.6,1].

```
deff('y=f(x)','y=x^2-sin(3*x)')
a=-0.5;b=0.5; N = 100; eps = 1.e-5; //
if(abs(f(a)) < eps) then error ('корень в точке a')
abort
end
if(abs(f(b)) < eps) then error ('корень в точке b')
abort
end;
while (N > 0)
 c = (a+b)/2
if(abs(f(c)) < eps) then x = c;
return
end
if(f(a)*f(c)<0) then b=c;
else
a = c;
end
N = N - 1;
end
```

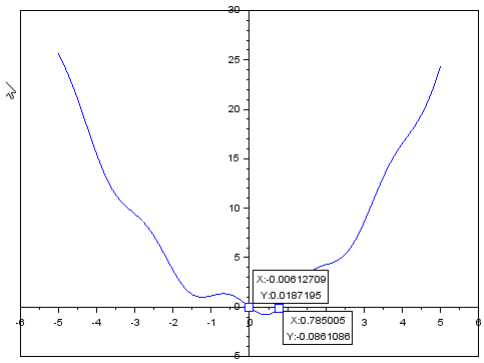


Рис.50 График функции $y= x^2-sin(3x)$

```
-->x //  отрезок  [-0.5,0.5]
 x  =
   0.
-->x//  отрезок  [0.6,1]
x  =
 0.8092132568359373667732
```

Пример 3. Отделить графически и найти методом хорд корни уравнения $x^3-2x^2+1=0$.

```
deff('[y]=f(x)','y=x^3-2*x^2+1')
N = 100; eps = 1.e-5; // максим.кол-во итераций и
относит. точность
a=-1;b=0;
x1 = a; x2 = b;
while (N>0)
gp = (f(x2)-f(x1))/(x2-x1);
xn = x1-f(x1)/gp;
if(abs(f(xn))<eps) then
x=xn
return(x);
end;
N = N - 1;
x1 = x2;
x2 = xn;
end;
```

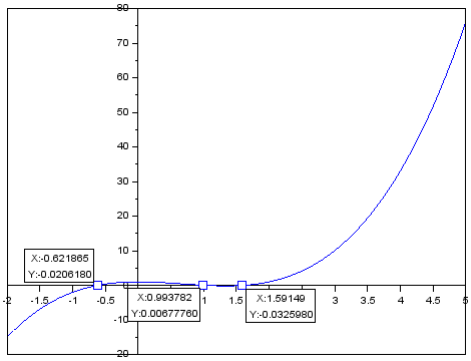


Рис.51 График функции $y= x^3-2x^2+1$

```
-->x// отрезок  [-1,0]
 x  =
  - 0.6180340
 -->x// отрезок  [0.5,1.1]
 x  =
 1.0000003
-->x //отрезок  [1.4,2]
 x  =
    1.618036
```

Корни уравнения $x^3-2x^2+1=0$, полученные методом хорд.

Пример 4. Отделить графически и найти методом хорд корни уравнения x^2-*sin(5x)-x+1=0*.

```
-->x// отрезок  [0,0.3]
 x  =
   0.1995552
-->x// отрезок  [0.3,0.6]
 x  =
  0.4581768
```

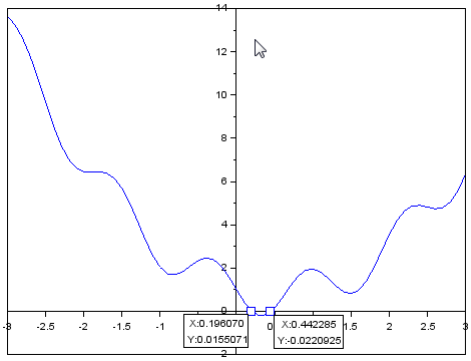


Рис.52 График функции $y= x^2$-*sin(5x)-x+1*

Пример 5. Отделить графически и найти методом Ньютона (касательных) корни уравнения $2x^2-1=0$.

```
clear;clc;
deff('[y]=f(x)','y=2*x^2-1');
deff('[y]=fp(x)','y=4*x');//производная
N = 100; eps = 1.e-5;
a=-1; //начальное значение корня
xx = a;
while (N>0)
xn = xx-f(xx)/fp(xx);
if(abs(f(xn))<eps) then
x=xn
return(x);
end
N = N - 1;
xx = xn;
end
->x//начальное значение -1
 x  =
  - 0.7071078
->x//начальное значение 1
 x  =
  0.7071078
```

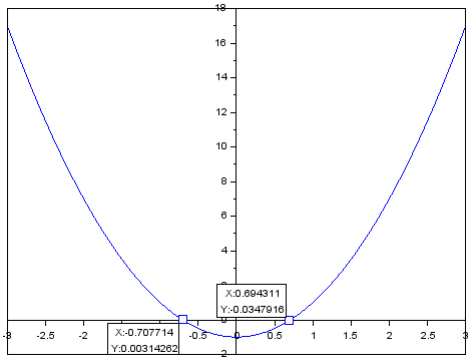


Рис.53 График функции $y= 2x^2-1$

Пример 6. Отделить графически и найти методом Ньютона (касательных) корни уравнения e^x-e^{-x}-*2=0*.

```
clear;clc;
deff('[y]=f(x)','y=%e^x-%e^(-x)-2');
deff('[y]=fp(x)','y=%e^x+%e^(-x)');
N = 100; eps = 1.e-5;a=0.5;
xx = a;
while (N>0)
xn = xx-f(xx)/fp(xx);
if(abs(f(xn))<eps) then
x=xn
return(x);
end
N = N - 1;
xx = xn;
end
-->x
 x  =
    0.8813737
```

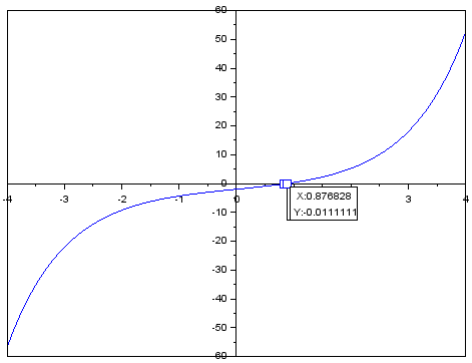


Рис.54 График функции $y=e^{x}-e^{-x}-2$

Пример 7. Отделить графически и найти методом простой итерации корни уравнения $3x - e^{x} + 2 = 0$.

Построим график нашей функции. На графике видно, что существуют два корня.

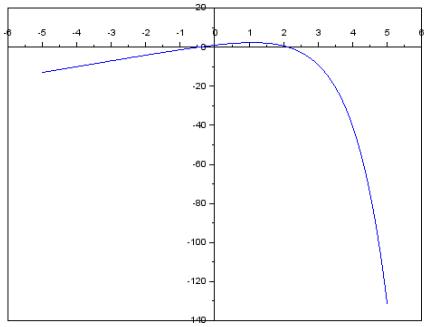


Рис.55 График функции $y=3x - e^{x} + 2$

Найдём корни уравнения методом простой итерации. Основная идея которого заключается в том, что исходное уравнение f(x)=0 переписывается в рекурсивной форме x = φ(x). Имея начальное значение x_0, и подставляя его в это уравнение, можно получить через несколько итераций решение исходного уравнения $x_{n+1} = \varphi(x_n)$. Итерационный процесс заканчивается, если $|x_{n+1}-x_n|<\varepsilon$. Необходимо отметить, что исходное уравнение в эквивалентном виде x=φ(x) может быть представлено различными способами. Выбирается такой способ, чтобы получить сходящуюся к корню последовательность вычислений. Представим исходное уравнение в виде $x=ln(3x+2)$и найдём все его корни.

```
deff('[y]=f(x)','y=log(3*x+2)')
N = 100; eps = 1.e-5; // максим.число итераций, макс. погрешность
x0=-1;
xx = x0;
while (N>0)
xn = f(xx);
if(abs(xn-xx)<eps) then
x=xn
disp(100-N);
return(x);
end;
N = N - 1;
xx = xn;
end;
error('Не сходится');
abort;
```

```
-->x// нач. точка x=-1,число итерац.13
 x  =
   2.1253935 + 0.0000049i
->x
 x  =// нач. точка x=0,число итерац.13
   2.1253872
 ->x
 x  =// нач. точка x=1,число итерац.12
     2.1253885
```

Видно, что при всех наших начальных значениях x, все значения корня сходятся ко второму значению. Представим исходное уравнение в виде $x=(e^x-2)/3$ и найдём его корни.

```
-->x .//нач. точка x=-1,число итерац.7
 x  =
  - 0.4552349
 -->x//нач. точка x=0,число итерац.7
 x  =
  - 0.4552309
 -->x//нач. точка x=1,число итерац.9
 x  =
  - 0.4552324
```

Видно, что при всех наших начальных значениях x, все значения корня сходятся к первому значению.

Исходя из вышеизложенного, делаем вывод, что при использовании метода итераций очень важно подобрать эквивалентные преобразования исходного уравнения, которые бы не повлияли на полноту получаемого решения.

Рассмотрим примеры использования встроенных функций Scilab **roots()** и **fsolve()** для решения нелинейных уравнений.

Пример 8. Найти корни уравнения полиномиального уравнения $1 - 4x + 5x^2 + 8x^3=0$.

Для нахождения корней полиномиальных уравнений в Scilab предусмотрена функция **roots().**

```
->p=poly([1 -4 5 8],'x','c')// определение полинома
 p  =
                2     3
   1 - 4x + 5x + 8x //нахождение корней
 -->roots(p)
 ans  =
  - 1.1527909
    0.2638954 + 0.1969561i
    0.2638954 - 0.1969561i
```

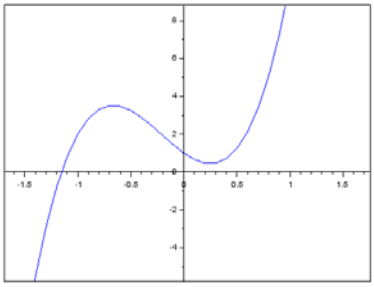


Рис.56 График функции $y=1 - 4x + 5x^2 + 8x^3$

Рассмотрим использование функции ***fsolve*** для решения нелинейных уравнений. Синтаксис для вызова функции **fsolve** :

[x [,v [,info]]]=**fsolve**(x0,fct [,fjac] [,tol]),

где x0 - начальное приближение для решения уравнения;

fct - внешняя (функция или список), описывающая уравнение fct(x) = 0;

fjac - внешняя (функция или список), описывающих якобиан или производные,

tol - скаляр задающий условия при проверке сходимости. По умолчанию tol = 1.0×10^{-10}.

Более подробное описание возможностей функции **fsolve()** можно найти в руководстве.

Пример 9. Найти корни уравнения $sin(x)\cdot cos(x)-x^2=0$, используя функцию **fsolve().**

Вначале отделим графически корни уравнения.

```
-->x=[-5:0.5:5];
-->plot(x,cos(x)*sin(x))
-->plot(x,x^2)
```

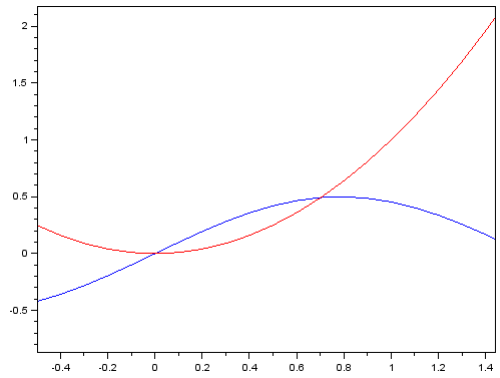


Рис.57 Графики функций $y=sin(x)cos(x)$ *и* $y=x^2$

Из графика видно, что у уравнения имеется два корня, которые находятся на отрезке [-0.1,0.8]. Рассмотрим использование функции **fsolve(x0,f)** с использованием производной и без неё.

```
-->deff('[y]=f(x)',['y=cos(x)*sin(x)-x^2'])
-->deff('[y1]=f1(x)',['y1=-sin(x)*cos(x)-2*x'])
-->x0=-0.1
-->x1=fsolve(x0,f) //без производной
x1  =
      0.  //первый корень
x0=0.5
-->x1=fsolve(x0,f) //без производной
x1  =
   0.7022074 // второй корень
```

```
-->deff('[y]=f(x)',['y=cos(x)*sin(x)-x^2'])
-->deff('[y1]=f1(x)',['y1=-sin(x)*cos(x)-2*x'])
-->x0=-0.1
-->x2=fsolve(x0,f,f1) //с производной
x2  =
      0.  //первый корень
x0=0.5
-->x2=fsolve(x0,f,f1) // с производной
x2  =
   0.7022074 // второй корень
```

Пример 10. Найти корни уравнения $x^{0.5} - 3+(x+1)^2=0$ на отрезке [-2,2], используя функцию **fsolve()**.

```
->deff('[y]=f(x)',['y=x^0.5-3+(x+1).^2'])
->x0=0.3;
-->x1=fsolve(0.3,f)
 x1  =
   0.5115471
```

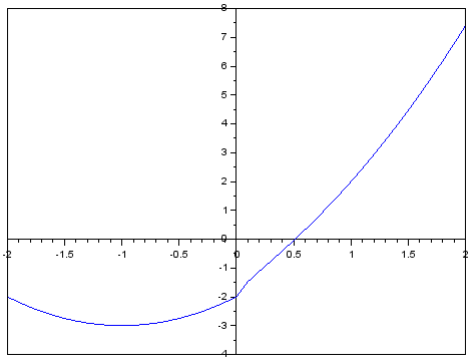


Рис.58 График функции $y=x^{0.5}-3+(x+1)^2$

Задачи.

1. Используя метод половинного деления, найти корни уравнения $1.8x^4 - sin10x = 0$ с точностью 0.01.

2. Используя метод половинного деления, найти корни уравнения $x + lg\ x - 0.5 = 0$ с точностью 0.01.

3. Используя метод половинного деления, найти корни уравнения $2ln\ x - x^{-1} + 0.5=0$ с точностью 0.01

4. Используя метод половинного деления, найти корни уравнения $cos\ x - x + 1 = 0$ с точностью 0.01

5. Отделить графически и найти методом хорд корни уравнения $\lg(2 + x) + 2x - 3 = 0$ с точностью 0.001.

6. Отделить графически и найти методом хорд корни уравнения $x^2-2sin(5x)=0$ с точностью 0.001

7. Отделить графически и найти методом хорд корни уравнения $2x \cdot lg(x) - 2.4 = 0$ с точностью 0.001

8. Определить количество действительных корней уравнения $x^3 + 4x - 6 = 0$, отделить эти корни и, применяя метод хорд, найти их приближённое значение с точностью 0.001.

9. Определить количество действительных корней уравнения $x^3-2x^2-4x-7=0$, отделить эти корни и, применяя метод касательных, найти их приближённое значение. Приближённые значения вычислить с двумя знаками после запятой.

10. Определить количество действительных корней уравнения $(x-1)^2e^x-7=0$. Отделить и найти методом касательных.

11. Отделить графически и найти методом касательных корни уравнения $\sqrt{|x-4|}-x+1=0$ с точностью до 0.01.

12. Отделить графически и найти методом касательных корни уравнения $x^4-2x-4=0$ с точностью до 0.001.

13. Отделить графически и найти методом простых итераций корни уравнения $x^3-2x^2-4x-7=0$ с точностью до 0.001.

14. Отделить графически и найти методом простых итераций корни уравнения $2-\lg(x)-x=0$ с точностью до 0.001.

15. Отделить графически и найти методом простых итераций корни уравнения $x^5-2x^4+15x^2-32x+1=0$ с точностью до 0.01.

16. Отделить графически и найти методом простых итераций корни уравнения $x^4-3x^3-7x^2+15x+18=0$ с точностью до 0.01.

17. Отделить графически и найти корни уравнения $4-2x-\sin(x)=0$, используя функцию **fsolve()**.

18. Отделить графически и найти корни уравнения $x\cdot\ln(x)-x-6=0$, используя функцию **fsolve()**.

19. Отделить графически и найти корни уравнения $x^4+(x-2)^2-8=0$, используя функцию **fsolve()**.

20. Отделить графически и найти корни уравнения $2x-\cos x=0$, используя функцию **fsolve()**.

Системы линейных алгебраических и нелинейных уравнений

4

Более 70% всех вычислительных задач сводятся к решению систем линейных алгебраических уравнений. Поэтому необходимо наличие эффективных алгоритмов и пакетов программ для их решения. Система Scilab изначально создавалась, как программа, позволяющая эффективно работать с матрицами и матричными уравнениями. В ней предусмотрен большой набор встроенных функций, предназначенных для решения систем уравнений. Вначале рассмотрим наиболее часто используемые алгоритмы, а затем встроенные функции Scilab для выполнения этих задач.

Кратко рассмотрим понятия и определения, которыми будем оперировать в дальнейшем. Система из n уравнений с n неизвестными называется линейной, если неизвестные входят в нее только в первой степени:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 ;$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 ;$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n$$

Где a_{ij}– коэффициенты уравнений; x_i – неизвестные; b_i– свободные члены уравнений, $i, j = 1, 2, \ldots, n$. Представив коэффициенты этой системы a_{ij} в виде прямоугольной таблицы, а неизвестные x_i и свободные члены b_i – в виде столбцов

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}; \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}; \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

представим систему уравнений в матричном виде:

$$Ax = b .$$

Такая система имеет единственное решение, если входящие в нее n уравнений линейно независимы. Необходимым и достаточным условием этого является условие неравенства нулю определителя данной системы. В случае, если детерминант равен нулю, то система либо не имеет решения, либо имеет бесчисленное множество решений.

Алгоритмы решения задач такого типа подразделяются на прямые (приводят к решению за конечное число арифметических операций) и итерационные (точное решение получается за бесконечное число арифметических операций). Один из используемых прямых методов - это метод Гаусса. Модификацией метода Гаусса является метод Гаусса-

Жордана, предложенный Жорданом. В случае применения итерационных алгоритмов решения систем линейных уравнений, используются Методы Якоби и Гаусса-Зейделя. Для решения систем нелинейных уравнений используются методы простых итераций, Ньютона, его модификаций и т.д. Более полный обзор существующих методов можно найти в соответствующей литературе, например [7]. Как отмечалось выше, в пакете Scilab существует достаточно богатый набор встроенных специальных матричных функций, с помощью которых можно реализовать решение системы уравнений: **rref(), linsolve(), inv(A), fsolve()**.

Пример 1. Найти методом Гаусса решение для системы линейных уравнений из трёх неизвестных.

$$\begin{cases} 5x_1 + 10x_2 + x_3 = 28 \\ x_1 + x_2 + x_3 = 6 \\ 4x_1 + 8x_2 + 3x_3 = 29 \end{cases}$$

```
clc ; clear ;
A =[5 ,10 ,1 ,28;1 ,1 ,1 ,6;4 ,8 ,3 ,29]; // расширенная матрица
for i =1:4
C(1,i)=A(1,i)
C(2,i)=A(2,i) -(A(2 ,1)/A(1 ,1))*A(1,i)// приведение к треугольному
C(3,i)=A(3,i) -(A(3 ,1)/A(1 ,1))*A(1,i)// виду
end
x (3) =C(3 ,4)/C(3 ,3); //решение для x(3)
for i =2: -1:1
   k=0
for j=i +1:3
k=k+C(i,j)*x(j)
end
x(i) =(1/ C(i,i))*(C(i ,4) -k);//решение для x(3)
end
```

```
->x
 x  =
   1.
   2.
   3.
 -->A
 A  =
   5.   10.   1.   28.
   1.   1.    1.   6.
   4.   8.    3.   29.
-->B
 C  =
   5.   10.   1.    28.
   0.  - 1.   0.8   0.4
   0.   0.    2.2   6.6
```

Пример 2. Создать сценарий-функцию (скрипт) для решения методом Гаусса системы линейных уравнений

и сохранить его.

```
function [x] = lingauss(A,b) //эта функция позволяет получить решение
системы линейных уравнений A*x = b,
    //где A матрица коэффициентов и  b вектор свободных членов, матрицы A
должна быть квадратная
[nA,mA] = size(A)// определение размерности матрицы A
[nb,mb] = size(b)// определение размерности b вектор свободных членов
a = [A b]; //Расширенная матрица
//прямой ход
n = nA;
//Циклы, реализующие алгоритм Гаусса
for k=1:n-1
for i=k+1:n
for j=k+1:n+1
a(i,j)=a(i,j)-a(k,j)*a(i,k)/a(k,k);
end;
end;
end;
//обратная подстановка
x(n) = a(n,n+1)/a(n,n);
for i = n-1:-1:1
sumk=0
for k=i+1:n
sumk=sumk+a(i,k)*x(k);
end;
x(i)=(a(i,n+1)-sumk)/a(i,i);  //вычисление корней системы
end:
```

Пример 3. Решить систему линейных уравнений, методом Гаусса, используя результаты примера 2.

$$\begin{cases} -7x_1 + x_2 + 3x_3 = 3 \\ 2x_1 - 7x_2 + 3x_3 = 2 \\ -x_1 - 2x_2 + 7x_3 = -4 \end{cases}$$

Сделаем рабочей директорией директорию, где расположен файл lingauss.sci или укажем полный путь до этого файла и запустим его командой exec('lingauss.sci').

```
->A = [-7,1,3;2,-7,3;-1,-2,7]; b = [3;2;-4];

-->lingauss(A,b)
 ans  =

  - 1.
  - 1.
  - 1.
```

Пример 4. Решить систему линейных уравнений, методом Гаусса-Жордана.

$$\begin{cases} 5x_1 + 10x_2 + x_3 = 28 \\ x_1 + x_2 + x_3 = 6 \\ 4x_1 + 8x_2 + 3x_3 = 29 \end{cases}$$

```
clc ; clear ; close ;
A =[5 ,10 ,1 ,28;4 ,8 ,3 ,29;1 ,1 ,1 ,6];
disp(A)
for i =1:3
j=i
while (A(i,i) ==0 & j <=3)
for k =1:4
B(1,k)=A(j+1,k)
A(j+1,k)=A(i,k)
A(i,k)=B(1,k)
end
j=j+1
end
for k =4: -1: i
A(i,k)=A(i,k)/A(i,i)
end
for k =1:3
if(k~=i) then
l=A(k,i)/A(i,i)
for m=i:4
A(k,m)=A(k,m)-l*A(i,m)
end;end;end;end;
disp (A)
```

```
A= //исходная матрица
   5.   10.   1.   28.
   4.   8.    3.   29.
   1.   1.    1.   6.

A= // приведённая, красным выделен ответ
   1.   0.   0.   1.
   0.   1.   0.   2.
   0.   0.   1.   3.
```

Пример 5. Решить систему линейных уравнений итерационным способом, используя метод Якоби с точностью 10^{-6}.

$$\begin{cases} 8x_1 - 3x_2 + 2x_3 = 20 \\ 4x_1 + 11x_2 - x_3 = 33 \\ 6x_1 + 3x_2 + 12x_3 = 36 \end{cases}$$

Запишем вычисления по методу Якоби в матричном виде. Дана система уравнений, которая в матричном виде выглядит следующим образом: **A·X=B**. Тогда её решение имеет вид: **X=C*X0+D**, где **C=-F^{-1}*(A-F)** и **D=F^{-1}*B**, **F** - квадратная диагональная матрица, элементы которой на главной диагонали совпадают с диагональными элементами матрицы **A**, **X0**-начальное приближение.

```
A=[8 , -3 ,2;4 ,11 , -1;6 ,3 ,12];
B =[20;33;36]
// формирование матрицы F
R=diag(A);
F=diag(R);
// формирование матриц C и D
C=-inv(F)*(A-F);
D=inv(F)*B;
// первое приближение
X0=[0;0;0];
X=C*X0+D;
//следующие итерации
while max(abs(X-X0))>=0.0000001,
X0=X; X=C*X0+D;
end;
```

```
Ответ
X  =
   3.
   2.
   1.
```

Пример 6. Решить систему линейных уравнений итерационным способом, используя метод Гаусса-Зейделя с точностью 10^{-6}.

$$\begin{cases} 8x_1 - 3x_2 + 2x_3 = 20 \\ 4x_1 + 11x_2 - x_3 = 33 \\ 6x_1 + 3x_2 + 12x_3 = 36 \end{cases}$$

Входные параметры: A - матрица коэффициентов, B - вектор правой части, e - критерий окончания процесса (погрешность).

```
Ответ:
->X
 X  =
    2.9999997
    2.0000001
    1.0000001
```

```
A=[8 , -3 ,2;4 ,11 , -1;6 ,3 ,12]; B =[20;33;36];
e=10-^6;
N=length(B); //размер вектора
X=zeros(N,1); // начальное приближение
n=0; // число итераций
it=1; // признак итераций
while it
 n=n+1;
 if n>100
  error('Число итераций >100')
  it=0;
  break
 end
 delta=0; // первоначальное  приращение решения
 for j=1:N
  XK=X(j); // запоминание старого значения решения
  X(j)=(B(j)-A(j,[1:j-1,j+1:N])*X([1:j-1,j+1:N]))/A(j,j);
  if abs(X(j)-XK)>delta
     delta=abs(X(j)-XK); // поправка решения
  end;end
if delta/norm(X,%inf) <=e;// критерий завершения
   it=0;
    end
end
```

Пример 7. Решить систему линейных уравнений используя функцию **linsolve().**

$$\begin{cases} -5x_1 + 3x_2 + 2x_3 = -1 \\ 3x_1 - 5x_2 + x_3 = 0 \\ 2x_1 + x_2 + 5x_3 = 4 \end{cases}$$

Особенностью этой функции является то, что надо изменить знак элементов вектора свободных членов на обратный.

```
-->A=[-5 3 2;3 -5 1;2 1 5];
-->B=[-1;0;4];
-->B=-B;
-->X=linsolve(A,B)
 X  =
   0.6666667
   0.4871795
   0.4358974
```

Пример 8. Решить систему линейных уравнений используя функцию **linsolve().**

$$\begin{cases} 2x_1 + 3x_2 - 5x_3 = -10 \\ x_1 - 3x_2 + 8x_3 = 85 \end{cases}$$

Рассмотрим решение системы уравнений с числом переменных большим, чем число уравнений. Каждое уравнение определяет плоскость в пространстве $x_1;x_2;x_3$. Поэтому решения системы образуют линию пересечения этих плоскостей, а не отдельные значения. Найдём эти решения, используя функцию **linsolve().** В этом случае **linsolve()** возвращает

частное решение и ядро матрицы (нуль-пространство) (множество решений системы однородных линейных уравнений). Тогда решение системы можно записать в виде:

*x = x0 + nspA*y* , где *y* любое число. В примере *y=5*.

```
-->A = [2, 3, -5; 1, -3, 8], b = [10; -85]
-->[x0,nspA] = linsolve(A,b)
 nspA  =
  - 0.3665083
    0.8551861
    0.3665083
 x0  =
    15.373134
    2.4626866
    9.6268657
-->x1 = x0 + nspA*5
 x1  =
    13.540593
    6.7386171
    11.459407
-->A*x1+b//проверка
 ans  =
 10^(-13) *
  - 0.2842171
    0.2842171
```

Пример 9. Решить систему линейных уравнений, используя обратную матрицу.

$$\begin{cases} -5x_1 + 3x_2 + 2x_3 = 1 \\ 3x_1 + 7x_2 + x_3 = 1 \\ 2x_1 - x_2 + 4x_3 = 1 \end{cases}$$

Решим систему линейных алгебраических уравнений матричным методом, т.е. умножим обе части уравнения на обратную матрицу $\mathbf{(A^{-1}A)X = A^{-1}B}$, где $\mathbf{A^{-1}}$ – матрица, обратная матрице **A**, $\mathbf{A^{-1}A = E}$, **E** – единичная матрица. Решение будет: $\mathbf{X = A^{-1}B}$. Критерием точности решения является число обусловленности **cond(A)** матрицы **A**. Чем оно больше, тем менее точно полученное решение. Как правило, **cond(A)** < 10.

```
->A=[-5 3 2;3 7 1;2 -1 4];
-->B=[1;1;1];
-->X=inv(A)*B
 X  =
  - 0.0191388
    0.1100478
    0.2870813
-->cond(A)
 ans  =
    1.8037769
```

Пример 10. Решить систему линейных уравнений, используя оператор левого деления.

$$\begin{cases} 3x_1 - 2x_2 = -1 \\ 3x_1 - 5x_2 + x_3 = 1 \\ x_2 + x_3 = 0 \end{cases}$$

Если система линейных уравнений A·x=b, имеет квадратную матрицу коэффициентов A, то можно получить решения этой системы, воспользовавшись обратным оператором деления x = A\b.

```
-->A=[3 -2 0;3 -5 1;0 1 1];
-->B=[-1;1;0];
-->x=A\B
 x  =
  - 0.6666667
  - 0.5
    0.5
```

Пример 11. Решить систему линейных уравнений, используя функцию **rref()**.

$$\begin{cases} -4x_1 - x_2 + 2x_3 = 1 \\ 2x_1 - 3x_2 + x_3 = 1 \\ 2x_1 + x_2 - 5x_3 = 1 \end{cases}$$

Данная функция позволяет привести матрицу A к треугольной форме, используя при этом метод исключения Гаусса. Четвёртый столбец полученной матрицы C содержит значение неизвестных переменных.

```
->A=[-4 -1 2;2 -3 1;2 1 -5];
-->B=[1;1;1];
-->AA=[A B]
 AA  =
  - 4.  - 1.    2.    1.
    2.  - 3.    1.    1.
    2.    1.  - 5.    1.
 -->C=rref(AA)
 C  =
    1.    0.    0.  - 0.3076923
    0.    1.    0.  - 0.6923077
    0.    0.    1.  - 0.4615385
```

Пример 12. Решить систему нелинейных уравнений, используя функцию **fsolve()** и начальное значение вектора x=(0.5 0.5 0.5).

$$\begin{cases} y_1 = {x_1}^3 + {x_2}^2 + x_3 - 5 \\ y_2 = {x_1}^2 + x_2 - 5x_3 \\ y_3 = 3{x_1}^3 - 5x_2 + x_3 \end{cases}$$

```
function [y]=fun(x)
y(1)=x(1)^3+x(2)^2+x(3)-5 //  первое уравнение
y(2)=x(1)^2+x(2)-5*x(3) //   второе  уравнение
y(3)=3*x(1)^3-5*x(2)+x(3) //  третье уравнение
endfunction
-->fsolve([0.5 0.5  0.5],fun)
 ans  =

   1.3058388    1.4627607    0.6335951
```

Задачи.

1. Решить систему линейных уравнений, методом Гаусса.

$$\begin{cases} -9x_1 + 2x_2 + 3x_3 = 4 \\ x_1 - 5x_2 + 3x_3 = 1 \\ -4x_1 - 2x_2 + 9x_3 = -3 \end{cases}$$

2. Решить систему линейных уравнений, методом Гаусса.

$$\begin{cases} 6x_1 - 2x_2 - x_3 = -3 \\ 3x_1 - 9x_2 + 2x_3 = 4 \\ x_1 - 2x_2 + 5x_3 = -4 \end{cases}$$

3. Решить систему линейных уравнений, методом Гаусса-Жордана

$$\begin{cases} 8x_1 - 3x_2 - 2x_3 = -3 \\ -x_1 + 5x_2 - 2x_3 = -2 \\ -3x_1 - 2x_2 + 8x_3 = -3 \end{cases}$$

4. Решить систему линейных уравнений, методом Гаусса-Жордана

$$\begin{cases} -7x_1 + x_2 + 3x_3 = 3 \\ 2x_1 - 7x_2 + 3x_3 = 2 \\ -x_1 - 2x_2 + 7x_3 = -4 \end{cases}$$

5. Решить систему линейных уравнений итерационным способом, используя метод Якоби с точностью 10^{-6}.

$$\begin{cases} 5x_1 - x_2 - 3x_3 = -1 \\ 4x_1 - 8x_2 + x_3 = 3 \\ -1x_1 - 3x_2 + 6x_3 = -2 \end{cases}$$

6. Решить систему линейных уравнений итерационным способом, используя метод Якоби с точностью 10^{-6}.

$$\begin{cases} -5x_1 + 2x_2 + 2x_3 = 1 \\ -x_1 + 8x_2 - 4x_3 = -3 \\ -2x_1 + x_2 + 5x_3 = -4 \end{cases}$$

7. Решить систему линейных уравнений итерационным способом, используя метод Гаусса-Зейделя с точностью 10^{-6}.

$$\begin{cases} -9x_1 + 5x_2 + 2x_3 = 2 \\ -2x_1 + 6x_2 + x_3 = -5 \\ -3x_1 - 4x_2 + 9x_3 = -2 \end{cases}$$

8. Решить систему линейных уравнений итерационным способом, используя метод Гаусса-Зейделя с точностью 10^{-6}.

$$\begin{cases} 5x_1 - 2x_2 + x_3 = -4 \\ 3x_1 + 8x_2 - 4x_3 = -7 \\ -x_1 - 2x_2 + 5x_3 = -2 \end{cases}$$

9. Решить систему линейных уравнений используя функцию **linsolve()**.

$$\begin{cases} 8x_1 - 3x_2 - 2x_3 = -3 \\ -x_1 + 5x_2 - 2x_3 = -2 \\ -3x_1 - 2x_2 + 8x_3 = -3 \end{cases}$$

10. Решить систему линейных уравнений используя функцию **linsolve()**.

$$\begin{cases} 5x_1 + x_2 + 2x_3 = 2 \\ 3x_1 - 9x_2 + 4x_3 = 2 \\ -x_1 + 2x_2 + 7x_3 = -8 \end{cases}$$

11. Решить систему линейных уравнений, используя оператор левого деления.

$$\begin{cases} -5x_1 + 3x_2 + x_3 = 1 \\ 3x_1 + 7x_2 + x_3 = 1 \\ 2x_1 - x_2 + 4x_3 = 1 \end{cases}$$

12. Решить систему линейных уравнений, используя оператор левого деления.

$$\begin{cases} 3x_1 + x_2 + x_3 = -1 \\ 3x_1 - x_2 + x_3 = 2 \\ 2x_1 + x_2 = 2 \end{cases}$$

13. Решить систему линейных уравнений, используя обратную матрицу

$$\begin{cases} 7x_1 - 3x_2 - x_3 = 3 \\ 2x_1 - 7x_2 + 2x_3 = -3 \\ -2x_1 - 3x_2 + 7x_3 = 2 \end{cases}$$

14. Решить систему линейных уравнений, используя обратную матрицу

$$\begin{cases} -8x_1 + 3x_2 + 4x_3 = 1 \\ 2x_1 - 6x_2 + 3x_3 = 1 \\ -3x_1 - 2x_2 + 6x_3 = -1 \end{cases}$$

15. Решить систему линейных уравнений, используя функцию **rref()**.

$$\begin{cases} 5.4x_1 + 1.8x_2 - 3x_3 = 7 \\ 4.5x_1 - 2.8x_2 + 6.7x_3 = 2.6 \\ 5.1x_1 + 3.7x_2 - 1.4x_3 = -0.14 \end{cases}$$

16. Решить систему линейных уравнений, используя функцию **rref()**.

$$\begin{cases} 4x_1 - 3x_2 + 2x_3 = 9 \\ 2x_1 + 5x_2 - 3x_3 = 4 \\ 5x_1 + 6x_2 - 2x_3 = 18 \end{cases}$$

17. Решить систему нелинейных уравнений, используя функцию fsolve() . Начальные значения корней локализовать графически.

$$\sin(x_1 + x_2) - x_2 - 1.2 = 0$$
$$2x_1 + \cos x_2 - 2 = 0$$

18. Решить систему нелинейных уравнений, используя функцию fsolve() . Начальные значения корней локализовать графически.

$$\sin(x_1 + x_2) - 1.3x_1 - 1 = 0$$
$$x_1^2 + 0.2x_2^2 - 1 = 0$$

19. Решить систему нелинейных уравнений, используя функцию fsolve() . Начальные значения корней локализовать графически.

$$\tan(x_1 x_2 + 0.2) - x_1^2 = 0$$
$$0.6x_1^2 + 2x_2^2 - 1 = 0$$

20. Решить систему нелинейных уравнений, используя функцию fsolve() . Начальные значения корней локализовать графически.

$$\sin(x_1 + x_2) - 1.6x_1 - 1 = 0$$
$$x_1^2 + x_2^2 - 1 = 0$$

Аналитическое приближение табличных функций

Если на некотором отрезке [a≤x≤ b] задан дискретный ряд значений y_i функции $y=f(x)$ в зависимости от значений аргумента x_i в виде числовой таблицы, то такая функция называется табличной. Тогда шагом таблицы является разница между соседними значениями аргумента $x_{i+1} - x_i = h$. И если эта разница постоянна, то говорят, что шаг таблицы равномерный, если нет, то шаг - неравномерный. При этом подразумевается, что аргументы таблицы являются точными числами.

Табличную функцию можно сформировать по результатам измерений или путем вычисления значений аналитически заданной функции. Аналитическим приближением табличной функции $f(x_i)$ является аналитически заданная и достаточно просто вычисляемая функция $p(x)$, совпадающая или близкая по своим значениям с табличными значениями функции $f(x_i)$ на всем отрезке области определения или на некоторой ее части

$$f(x_i) \approx p(x), \quad [a \le x_i \le b]$$

Процедура замены табличной функции $f(x_i)$ на аналитически заданную функцию $p(x)$, называется аппроксимацией, а сама функция $p(x)$ - аппроксимирующей.

Аппроксимация табличной функции называется интерполированием, или интерполяцией, если её значения y_i совпадают со значениями аппроксимирующей функции $p(x_i)$ в точках, соответствующих табличным аргументам x_i . В задачах интерполяции функцию $p(x)$ называют интерполирующей функцией, а табличные аргументы $x_0, x_i..., x_n$, в которых её значения должны совпадать со значениями табличной функции $f(x_i)$ - узлами интерполирования.

Способ интерполяции предполагает точное совпадение значений табличной и интерполирующей функций в узлах интерполирования. Если же табличная функция получена вычислениями с погрешностями или в результате выполнения измерений, то условие обязательного совпадения значений функций означало бы повторение имеющихся в расчетных или экспериментальных данных ошибок. В этих случаях, процедура аппроксимации должна решать задачу сглаживания табличных значений, правильно отражая описываемую зависимость. Способ сглаживающей аппроксимации, при котором график аппроксимирующей функции проходит по возможности близко между значениями табличной функции, называется фитированием, а сама аппроксимирующая функция – функцией фитирования.

Общая постановка задачи фитирования: для таблично заданной функции $f(x_i)$ требуется построить аналитическую функцию $\varphi(x,c_0,c_1,\ldots,c_m)$,так чтобы минимизировать значение

функционала Q, сформированного совокупностью отклонений, имеющих место при фитировании значений табличной функции: min $Q\{ \varphi(x,c_0,c_1,\ldots,c_m)-y_i\}$, где x_i , y_i - табличные значения, $c_0,c_1,\ldots,c_m$ - числовые коэффициенты.

Существует достаточно много способов интерполяции: *линейная*, *метод конечных разностей*, интерполяционная формула *Ньютона*, многочлен *Лагранжа*, *сплайнами*. Объединяет их то, что интерполирующую функцию ищут в виде многочлена (полинома). При этом интерполирование называют полиномиальным, а соответствующий многочлен – интерполяционным. Наиболее общей формулой для интерполирования является интерполяционный многочлен *Лагранжа*. Если узлы интерполирования x_o, …, x_n - равноотстоящие, то используются более простые для вычислений интерполяционные формулы *Ньютона*. В случае использования большого числа узлов интерполяции на всем отрезке [a, b], интерполяционные формулы *Лагранжа*, *Ньютона* могут привести к плохому приближению из-за накопления погрешностей в процессе вычислений. Кроме того, из-за расходимости процесса интерполяции увеличение числа узлов не обязательно приводит к повышению точности. Для снижения погрешностей при большом числе узлов интерполирования выполняется кусочно-полиномиальная интерполяция функции $f(x_i)$ с использованием сплайнов.

В случае, если стоит задача фитирования табличной функции, чаще всего используется метод наименьших квадратов.

В Scilab методы интерполяции реализованы с помощью большого набора встроенных функций. Перечислим некоторые из них:

interp - интерполирование функции, заданной дискретно, с помощью кубического сплайна

interp1 - одномерная функция

interpln - линейная интерполяция

linear_interpn - n размерная линейная интерполяция

smooth - сглаживание с помощью сплайнов

splin - кубическая сплайн интерполяция

Для задач фитирования функций можно использовать встроенную функцию **datafit()**, либо - **leastsq()**.

Пример 1. Даны значения табличной функции в точках (x,y): (3.4, 0.294118), (3.5, 0.285714), (3.6, 0.277778). Интерполировать данную функцию полиномом и найти её значение в точке x=3.55.

Представим нашу функцию в виде полинома: $y = P_2(x) = a_0 + a_1x + a_2x^2$, поскольку заданы только три точки. Чтобы найти коэффициенты полинома, необходимо решить систему из трёх уравнений:

$$\begin{cases} a_0 + a_1x_0 + a_2x_0^2 = y_0 \\ a_0 + a_1x_1 + a_2x_1^2 = y_1 \\ a_0 + a_1x_2 + a_2x_2^2 = y_2 \end{cases}$$

Решение выглядит следующим образом:

```
x = [3.4 3.5 3.6]; y = [0.294118, 0.285714, 0.277778];
[m1,n1]=size(x);
m = n1;
n = m - 1; //порядок полинома
B = ones(m,m); //приравнять коэфф. матрицы 1.0
//вычислить значения оставшихся коэфф. матрицы
for i = 1:m
for j = 2:m
B(i,j) = x(i)^(j-1);
end;
end;
a = linsolve(B,-y'); //решение системы
a=a'; //показать решение как вектор-строку
 a  =
   0.858314  - 0.2455    0.0234
 -->p = poly(a,'x','coeff')// полученный полином
 p  =
                          2
   0.858314 - 0.2455x + 0.0234x
-->horner(p,x)
 ans  =
   0.294118    0.285714    0.277778 //значения в точках x
-->x1=3:0.1:5;
-->y1=horner(p,x1);
-->plot(x1,y1)
-->plot(x,y,'dk')
-->plot(3.55,horner(p,3.55),'or')
```

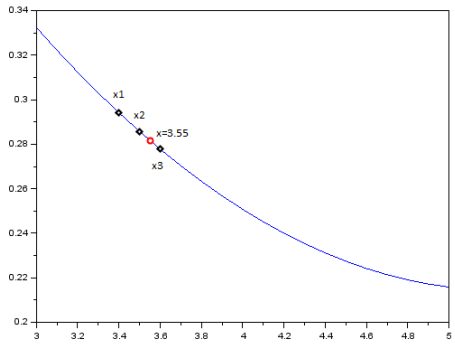


Рис.59 График интерполяционного полинома с узлами интерполяции

Пример 2. Написать файл-сценарий для интерполяции табличной функции полиномом Лагранжа порядка n. Вычислить значения табличной функции в точке x=3.44 в случае линейной и квадратичной интерполяции для значений табличной функции в точках (x,y): (3.4, 0.294118), (3.5, 0.285714), (3.6, 0.277778).

```
n=1;// степень полинома
x = [3.4 3.5 3.55 3.65]; yy = [0.294118, 0.285714, 0.281690, 0.273973];
x0=3.44
[m1,n1]=size(x);
m = n1;
if(n>m-1) then
error('используйте меньшую степень полинома n');
abort
end
m = n + 1;
N = ones(1,m);
D = N;
L = N;
y = 0.0;
for j = 1:m
for k = 1:m
if(k <> j) then
N(j) = N(j)*(x0-x(k));
D(j) = D(j)*(x(j)-x(k));
end;
end;
L(j) = N(j)/D(j);
y = y + L(j)*yy(j);
end;
 y  =
    0.2907564  //линейная n=1
 y  =
 0.2906994 // квадратичная n=2
```

Пример 3. Задана табличная функция в точках (x,y): (-1, 1), (1, 3), (2, -1),(3.5,1.9),(5,4). Линейно интерполировать функцию и построить её график на интервале [-4; 45], найти ее значение в точке x = 1.5.

```
x=[-1 1 2 3.5 5]; y=[-1 3 -1 1.9 4];
xy=[x;y];
x_new=(-4:8);
yi=interpln(xy,x_new);
plot2d(x',y',-2);
plot2d((-4:8)',yi');
-->z=interpln(xy,1.5)//значение в точке x=1.5
 z  =
    1.
```

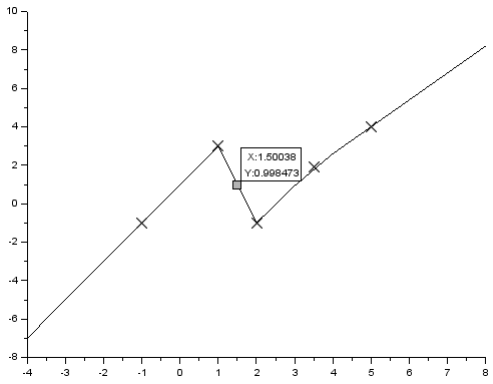


Рис.60 График табличной функции зад.3

Пример 4. Задана табличная функция (значения tg(x)) в точках (x,y): (-1.5, -14.1014), (-0.75,-0.931596), (0, 0), (0.75 ,0.931596),(1.5,14.1014). Интерполировать функцию линейно, кубическим сплайном, по соседним точкам и построить её график на интервале [-2; 2], найти ее значение в точке x = 1.5. Использовать функцию **interp1()**.

```
x=[-1.5 -0.75 0 0.75 1.5];
y=[-14.10114 -0.931596 0 0.931596 14.1014];
xx=linspace(-2,2,100);
yy1=interp1(x,y,xx,'linear');
yy2=interp1(x,y,xx,'spline');
yy3=interp1(x,y,xx,'nearest');
plot(xx',[yy1;yy2;yy3]',x,y,'*')
xtitle('interpolation of square function')
legend(['линейная','сплайн','соседняя'],a=2)
-->yy2=interp1(x,y,1.5,'spline')//значение в x=1.5
 yy2  =
   14.1014
```

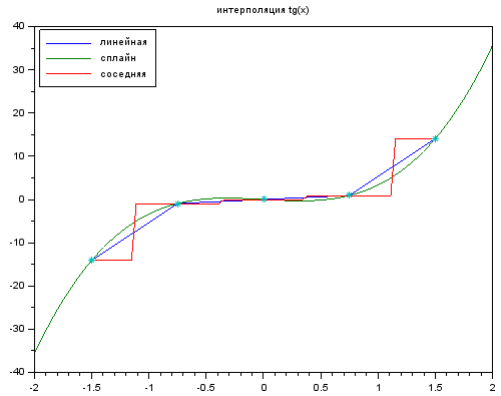


Рис.61 График y=tg(x)

Пример 5. Задана табличная функция в точках (x,y): (0.0, 15), (1.2, 29), (3.5, 13.3), (4.2 ,-6.41),(6.2,2.9),(8.1,17.1),(11.2,-8) Интерполировать её кубическими сплайнами и построить график, полученной функции.

Чтобы интерполировать функцию кубическими сплайнами в Scilab, необходимо вначале вычислить коэффициенты сплайна с помощью функции **d=splin(x,y)**, а затем вычислить значения интерполяционного полинома в нужной точке, используя функцию **y=interp(t,x,y,d)**.

```
y = [15.0,29.0,13.3,-6.4,2.9,17.1,-8.0];
x = [0.0,1.2,3.5,4.2,6.2,8.1,11.2];
d = splin(x,y);
xx = [0:0.1:11.2];
[y0,y1,y2,y3] = interp(xx,x,y,d);
plot(xx,y0)
plot(x,y,'ok')
```

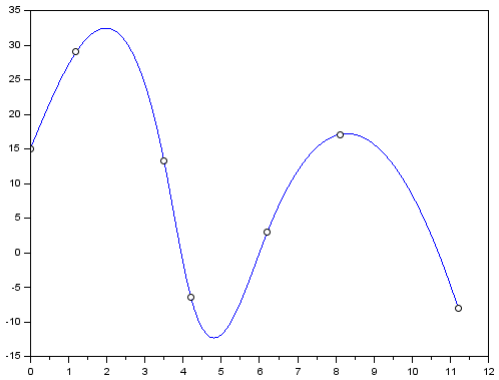


Рис.62 График функции

Пример 6. Задана табличная функция в точках (x,y): (0.0, 15.5), (1.25, 29.1), (3.75, 13), (4.32 ,-6.9),(6.2,2.99),(8.1,15.1),(11.25,-5) Интерполировать её кубическими сплайнами, вычислить значения функции с шагом Δx=0.1, и построить график, полученной функции.

Для решения этой задачи используем функцию Scilab **smooth()**.

```
y = [15.5,29.1,13.0,-6.9,2.99,15.1,-5.0];
x = [0.0,1.25,3.75,4.32,6.2,8.1,11.25];
xy_o=[x;y];
xy_f=smooth(xy_o,0.1);
xx=xy_f(1,:); yy = xy_f(2,:);
plot(xx,yy)
plot(x,y,'ok')
```

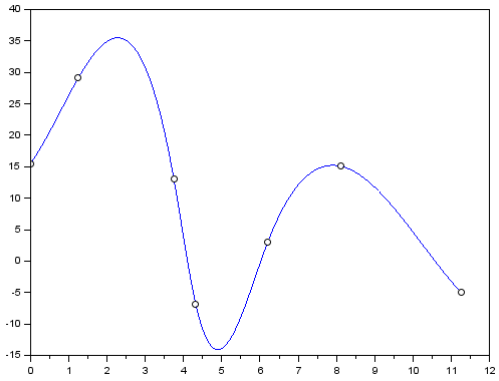


Рис.63 График функции

Пример 7. Задана функция $z = 2 \cdot sin(x) \cdot cos(y+x)$ где x и y меняются в пределах от 0 до 2π. Заданы узлы интерполирования- оси x и y равномерно разбиты на 10 отрезков. Используя линейную интерполяцию заданной функции, вычислить её значения в заданных точках (количество точек по оси x и y равно 50). Построить график интерполированной функции и отметить узлы интерполирования.

```
n = 10;
x = linspace(0,2*%pi,n); y = x;
z = 2*sin(x')*cos(y+x);
xx = linspace(0,2*%pi, 50);
[xp,yp] = ndgrid(xx,xx);
zp = linear_interpn(xp,yp, x, y, z);
clf()
plot3d(xx, xx, zp, flag=[2 6 4])
[xg,yg] = ndgrid(x,x);
param3d1(xg,yg, list(z,-5*ones(1,n)), flag=[0 0])
```

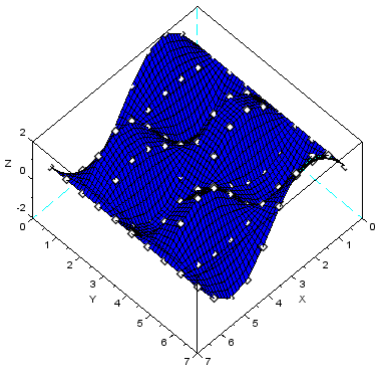


Рис.64 График функции $z = 2 \cdot sin(x) \cdot cos(y+x)$

Пример 8. Задан набор экспериментальных данных y, зависящих от x. Известно, что экспериментальные значения y_i содержат ошибки. Считая, что реальная зависимость (x_i,y_i) линейная $y_i = b+ax_i$, найти параметры a_1 и a_2. Значения x: 0; 0.15; 0.31; 0.48; 0.61; 0.77; 0.92; 1.01; 1.16; 1.32; 1.47. Значения y: 0; 0.16; 0.33; 0.42 ;0.59; 0.72 ;0.75; 0.88; 0.95; 0.96; 0.99.

Такая задача называется парной линейной регрессией или фитированием данных линейной зависимостью. Регрессионные коэффициенты a_1 и a_2 найдём с помощью функции Scilab [**a b sig]= reglin (x,y)**, где a,b –регрессионные коэффициенты, x и y – экспериментальные данные, sig-стандартное отклонение разности между данными и вычисленными точками.

```
x=[0 0.15 0.31 0.48 0.61 0.77 0.92 1.01 1.16 1.32 1.47];
y=[0 0.16 0.33 0.42 0.59 0.72 0.75 0.88 0.95 0.96 0.99];
plot2d(x,y,-5);
[a b sig]=reglin(x,y)
t=0:0.01:1.5;
yt=b+a*t;
plot2d(t,yt);
-->sig
 sig  =
   0.0690468
```

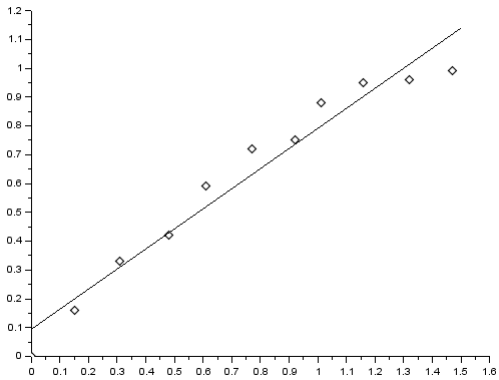


Рис.65 Линейная регрессия

Пример 9. Задан набор экспериментальных данных y, зависящих от t. Известно, что экспериментальные значения y_i содержат ошибки. Считая, что реальная зависимость (t_i,y_i) имеет вид $y = a \cdot exp(-b \cdot t)$, найти параметры a и b. Значения t: 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5. Значения y: 0.79, 0.59, 0.47, 0.36, 0.29, 0.23, 0.17, 0.15, 0.12, 0.08.

Данная задача отличается от предыдущей, тем, что зависимость y от t описывается нелинейной функцией. Поэтому, необходимо воспользоваться другой встроенной функцией - **leastsq()**. Эта функция минимизирует в задаче сумму квадратов разностей между экспериментальным и вычисленным значением (здесь wm – вес измерения).

$$f(x) = \sum_i wm^2(yth(tm(i), x) - ym(i))^2$$

```
function y=yth(t, x)
   y  = x(1)*exp(-x(2)*t)
endfunction
m = 10;
tm = [0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5]';
ym = [0.79, 0.59, 0.47, 0.36, 0.29, 0.23, 0.17, 0.15, 0.12, 0.08]';
// веса измерений, здесь все равны 1
wm = ones(m,1);
//начальные значения параметров фита
x0 = [1.5 ; 0.8];
function e=myfun(x, tm, ym, wm)
   e = wm.*( yth(tm, x) - ym )
endfunction
[f,xopt, gopt] = leastsq(list(myfun,tm,ym,wm),x0)
tt = linspace(0,1.1*max(tm),100)';
yy = yth(tt, xopt);
plot(tm, ym, 'kx')
plot(tt, yy, 'b-')
legend(['заданные точки', 'фит']);
xtitle('фит функцией leastsq')
```

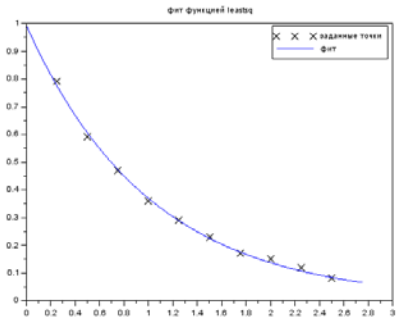


Рис.66 График фита функции с измеренными точками

Пример 10. Задан набор экспериментальных данных y, зависящих от x. Известно, что экспериментальные значения yi содержат ошибки. Считая, что реальная зависимость (x_i,y_i) имеет вид $y = Ax^2 + Bx + C$, найти параметры A, B и C. Значения x :72, 73,77, 81, 85, 92, 97. Значения y : 183, 172, 163, 153, 147, 141, 136.

Для решения этой задачи воспользуемся встроенной функцией - **datafit()**. Эта функция фитирует данные методом наименьших квадратов, минимизируя функцию ошибок e = G(p,z), где p – вектор –столбец из m рядов, представляющий параметры модели, а z – вектор-столбец из n рядов, представляющий переменные. Функция datafit ищет решение системы из k уравнений

$e_i = G(p,z_i) = 0$, минимизируя функционал

$$G^T(p,z_1)\cdot W\cdot G(p,z_1) + G^T(p,z_2)\cdot W\cdot G(p,z_2) + \ldots + G^T(p,z_n)\cdot W\cdot G(p,z_n),$$

где -W вес измерений. Простой вызов функции datafit выглядит следующим образом:

[p,err]=datafit(G,Z,p0)

Здесь G имя функции ошибок G(p,z), Z – матрица, ряды которой состоят из векторов переменных, p0 вектор-столбец, представляющий начальные значения параметров модели.

```
function [zr]=G(c,z)
zr=z(2)-c(1)-c(2)*z(1)-c(3)*z(1)^2
endfunction
x= [72 73 77 81 85 92  97];
y = [183 172 163  153 147 141 136];
z=[x;y];
c=[0;0;0];
[a,err]=datafit(G,z,c)
-->a
 a  =
    784.13675
  - 13.409302
    0.0695040
тогда  A = 0.0695040, B =  - 13.409302, C = 784.13675
dd2=0.0695040*x.^2-13.409302*x+784.13675;
  plot2d(x,dd2,-4);plot2d(x,dd2);
```

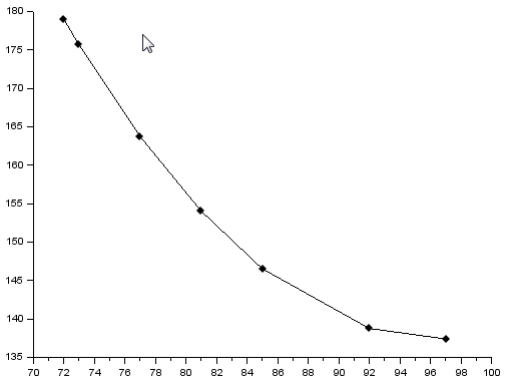


Рис.67 График фита функции с измеренными точками

Задачи.

1. Интерполировать таблично заданную функцию полиномами 1 степени.

x_1	x_2	x_3	y_1	y_2	y_3
-5	-2	0	-2	0	-1

2. Интерполировать таблично заданную функцию полиномом Лагранжа 2 степени.

x_1	x_2	x_3	y_1	y_2	y_3

-4 -1 1 3 1 2

3. Для функции *y=f(x)*, заданной значениями *f(1)=0.70, f(2)= 3.31, f(3)=0.181* найти коэффициенты интерполирующего многочлена $P_2(x)=a_0+a_1x+a_2x^2$.

4. Дана таблица значений функции *y=lg(x)*;

x1 x2 x3 x4 x5 y1 y2 y3 y4 y5

11 12 13 14 15 1.0414 1.0792 1.1139 1.1461 1.1761

С помощью линейной интерполяционной формулой Лагранжа вычислить значение *lg(11.5),* оценить погрешность вычисления.

5. Аппроксимировать таблично заданную функцию полиномами 1 степени.

x_1	x_2	x_3	y_1	y_2	y_3
-4	*-3*	*-1*	*-2*	*0*	*-1*

6. Интерполировать таблично заданную функцию с помощью команды **interpln**.

x_1	x_2	x_3	y_1	y_2	y_3
-3	-1	1	0	3	-2

7. Аппроксимировать таблично заданную функцию полиномом 2 степени.

x_1	x_2	x_3	y_1	y_2	y_3
-3	*-1*	*0*	*1*	*-1*	*2*

8. Восстановить многочлен по его значениям *y=P3(x)*:

x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4
-1	*0*	*1*	*2*	*1*	*1*	*-1*	*7*

9. Найти многочлен P(x), данные о котором представлены следующей таблицей:

x	P(x)	P'(x)	P''(x)
-1	15	-14	-2
0	4	-7	
2	18		

10. Интерполировать таблично заданную функцию функцией **interpln().**

x_1	x_2	x_3	y_1	y_2	y_3
-2	*0*	*1*	*-1*	*1*	*-2*

11. Задана табличная функция

x_1	x_2	x_3	y_1	y_2	y_3
-5	*-3*	*-1*	*0*	*3*	*-2*

Интерполировать функцию линейно, кубическим сплайном, по соседним точкам и построить её график на интервале [-6; 0], найти ее значение в точке x = -1.5. Использовать функцию **interp1()**.

12. Задана табличная функция в точках (x,y):

x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4
-1	*0*	*1*	*2*	*1*	*1*	*-1*	*7*

Интерполировать её кубическими сплайнами и построить график, полученной функции.

13. Задана функция $z = 4 \cdot \sin^2(x+y) \cdot \cos(y+x)$ где x и y меняются в пределах от 0 до 2π. Заданы узлы интерполирования- оси x и y равномерно разбиты на 10 отрезков. Используя линейную интерполяцию заданной функции, вычислить её значения в заданных точках (количество точек по оси x и y равно 50). Построить график интерполированной функции и отметить узлы интерполирования.

14. Результаты измерения величин x и y даны в таблице:

x_i	-2	0	1	2	4
y_i	0.5	1	1.5	2	3

Предполагая, что между x и y существует линейная зависимость $y=a+bx$, определить коэффициенты a и b, использовать функцию [**a b sig]= reglin (x,y)**.

15. Задан набор экспериментальных данных y, зависящих от x. Известно, что экспериментальные значения y_i содержат ошибки. Считая, что реальная зависимость (x_i,y_i) имеет вид $y = C_1 x \cos(C_2 x) + C_3$;, найти параметры C_1, C_2, C_3.

x = 0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6

y= 5.02 6.08 3.33 -0.93 -0.22 7.83 16.52 15.55 2.67 -11.42 -11.78 5.09 25.25

Использовать функцию **leastsq()**.

16. Для табличной функции

x_i	0.5	1	1.5	2	2.5
y_i	10.5	1.6	0.55	0.26	0.15

подобрать подходящий вид аппроксимирующей её нелинейной зависимости из ряда:

а) $y=ax^b$ б) $y=ae^{bx}$ в) $y=a+b/x$ г) $y=1/(a+bx)$

Использовать при этом метод наименьших квадратов (функцию **datafit()**). Сравнивать между собой среднеквадратические погрешности.

17. Дана табличная функция

x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.8	0.9	1.0
y	0	0.095	0.182	0.262	0.337	0.406	0.470	0.588	0.642	0.693

Используя функцию **reglin ()**, аппроксимировать данную функцию функцией вида $y=a_0+a_1x$ по трём точкам (x=0, x=0.5, x=1.0) и по всем точкам. Сравнить результаты с результатом применения функции **datafit()**. Построить график функции $y=ln(1+x)$ и график аппроксимирующей функции.

18. Заданы результаты измерений:

x = 1.30; 1.40; 1.55; 1.65; 1.74; 1.80; 1.95; 2.05; 2.13; 2.19; 2.35; 2.39; 2.58

y = 3.35; 3.45; 3.75; 4.30; 4.53; 4.86; 5.44; 6.04; 6.55; 7.32; 9.24; 10.21; 13.54

Считая, что реальная зависимость (x_i,y_i) имеет вид : $y = a_1 + a_2 x + a_3 x^2 + a_4 x^3$, найти значения параметров данной зависимости и построить графики экспериментальных данных и функции фитирования.

19. Построить уравнение аппроксимирующей параболы для следующих данных

x: 0, 1, 2, 3; y: -1.2, -0.1, 1.3, 1.8.

20. Найти параметры следующей функции $y=x/(Ax+B)$, если заданы её значения в 10 точках:

x 3 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9

y 0.61 0.6 0.592 0.58 0.585 0.583 0.582 0.57 0.572 0.571

Численное дифференцирование и интегрирование 6

Задачи численного дифференцирования и интегрирования возникают, когда необходимо вычислить производные или определённые интегралы от функций, заданных таблично или, когда непосредственное дифференцирование и интегрирование затруднено из-за сложного аналитического вида функций.

Основные формулы численного дифференцирования вытекают непосредственно из определения производной функции *f(x)*. Но, как хорошо известно, численное дифференцирование функций, исходя из определения производной, может привести к большим погрешностям результатов, в свою очередь, для производных высших порядков возможны ещё большие погрешности. Более общий подход к задаче численного дифференцирования табличной функции основан на использовании интерполяционных полиномов. В Scilab существует четыре встроенные функции для вычисления производных: **diff()**, **derivat()**, **numdiff()**, **numderivative()**.

Что касается вычисления значений определённых интегралов, то в основе большинства приближенных методов вычисления лежит разбиение интервала интегрирования *[a,b]* на *n* элементарных отрезков и замена на каждом частичном отрезке интегрирования подынтегральной функции *f(x)* интерполяционным полиномом определенной степени или сплайнами разных типов. Точки разбиения $a=x_0,x_1,\ldots x_n=b$ называются узлами интегрирования, а расстояния между узлами $h_i=x_i-x_{i-1}$ – шагами интегрирования. Шаг интегрирования может быть как постоянным, так и переменным. Приближенное значение интеграла определяется суммой частичных интегралов от функций $\varphi_i(x)$, взятых в пределах от x_{i-1} до x_i:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} \int_{x-1}^{x_i} \varphi_i(x)dx$$

Перечислим некоторые методы численного интегрирования: *метод прямоугольников, метод средних прямоугольников, метод трапеций, метод парабол*.

В **Scilab** существует достаточно большое количество встроенных функций, реализующих разные методы численного интегрирования: **inttrap(), integrate(), intg()**, **intsplin(), intc(), intl(), int2d(), int3d().** Рассмотрим задачи, в которых необходимо численно продифференцировать или проинтегрировать заданную функцию.

Пример 1. Вычислить значение производной функции $f(x)=(x+5)^4+2x$ в точке x=0.5.

Если функция задана таблицей своих значений, то для нахождения первого дифференциала используют функцию dy1=diff(y,1), а для нахождения n-го дифференциала

dyn=diff(y,n). После нахождения дифференциалов рассчитывают производную по формуле:

$$y' = \frac{1}{h}\left(dy1 - \frac{dy2}{2} + \frac{dy3}{3} - \frac{dy4}{4} + \frac{dy5}{5} - \cdots\right)$$

```
function y=f(x)
    y=(x+5)^4+2.*x
   endfunction
   h=0.1
x=0:h:1 // вектор x
y=f(x);
dy1=diff(y,1);
dy2=diff(y,2);
dy3=diff(y,3);
dy4=diff(y,4);
for k=1:8, z(k)=(dy1(k)-dy2(k)/2+dy3(k)/3)/0.1, end
n=find(x==0.5) //индекс точки x=0.5 в векторе x
n  =
   6.
-->z(6) //значение производной в точке x=0.5
 ans  =
   667.5
```

Пример 2. Вычислить значение производной функции $f(x)=x^3+2sin(x)$ в точке x=π/2.

Воспользуемся встроенной функцией **numdiff(fun,x)**.

```
function y=f(x)
    y=x.^3+2.*sin(x)
   endfunction
z=numdiff(f,%pi/2);
-->z
 z  =
   7.4022033
```

Пример 3. Найти первую и вторую производную для следующего полинома:

$p(s)=s^6+5s^2-1$.

Для вычисления производных от полиномов используется функция **derivat().**

```
->s=poly(0,'s');
-->p=s^6+5*s^2-1
 p  =
            2    6
  - 1 + 5s + s
-->derivat(p)   ans  =
                5
    10s + 6s

-->derivat(ans)
 ans  =
                4
   10 + 30s
```

Пример 4. Вычислить значение производной функции $f(x_1,x_2,x_3)=sin(x_1x_2) + exp(x_2x_3+x_1)$ в точке $x_1=1$, $x_2=2$, $x_3=3$.

Для этой цели воспользуемся функцией **numderivative().**

```
function y=fun(x)
  f1 = sin(x(1)*x(2)) + exp(x(2)*x(3)+x(1))
  y = f1;
endfunction
x = [1; 2; 3];
J = numderivative(fun, x)
-->J
 J  =
    1095.8009    3289.4833    2193.2663
```

Пример 5. Вычислить значение интеграла $\int_0^1 (e^{2x} - 1)dx$, используя функции **inttrap()** – (метод трапеций) и **integrate()** (в функции реализованы квадратурные формулы Ньютона–Котеса высших порядков).

```
//способ – метод трапеций:
h=0.001;     x=0:h:1;
y=exp(2*x)-1;
int=inttrap(x,y)
int  =
    2.1945291
//второй способ
-->integrate('exp(2*x)-1','x',0,1)
 ans  =
    2.194528
```

Пример 6. Вычислить интеграл $\int_0^1 (e^{2x} - 1)dx$, используя функцию **intg()**.

Данная функция является в Scilab наиболее универсальной, поскольку подынтегральное выражение может быть задано в виде набора дискретных точек (как таблица) или с помощью внешней функции. Кроме этого могут быть заданы абсолютная и относительная точность вычислений, которые являются необязательными параметрами.

```
-->deff('y=G(x)','y=exp(2*x)-1'); intg(0,1,G)
 ans  =
   2.194528
```

Пример 7. Вычислить интеграл $\int_0^1 (e^{2x} - 1)dx$, используя функцию **intsplin().**

С помощью функции **intsplin** производится интегрирование экспериментальных данных с помощью сплайн-интерполяции.

```
x=0:0.1:1;
y=exp(2*x)-1;
v=intsplin(x,y)
v  =
    2.1945385
```

Пример 8. Выполнить численное интегрирование дискретно заданной функции $y(x)=x^2$ в интервале от1 до 6. Аналитически вычисленное значение интеграла I =63/3-1/3=71.666667 .

```
-->x=[1.,2.,3.,4.,5.,6.];
-->y=[1,4,9,16,25,36];
-->v=intsplin(x,y)
Результат:
-->v =
71.66667
```

В Scilab предусмотрена функция для вычисления определённого двойного интеграла от функции z = f(x,y) определённой на области, где область разбивается на n треугольников. Функцию можно вызвать следующим образом: **[Int,er] = int2d(X,Y,f)**,

где: Int –значение интеграла, er-оценка общей ошибки вычислений, X и Y массив 3 на n, содержащий абсциссы и ординаты вершин n треугольников, на которые поделена область интегрирования.

Пример 9. Вычислить интеграл от функции *f(x,y) = cos(x+y)* в области R= {0 <x<5, 0<y< 5}. Разобьём область интегрирования на треугольники. Её можно разбить двумя способами, которые указаны на рисунке ниже.

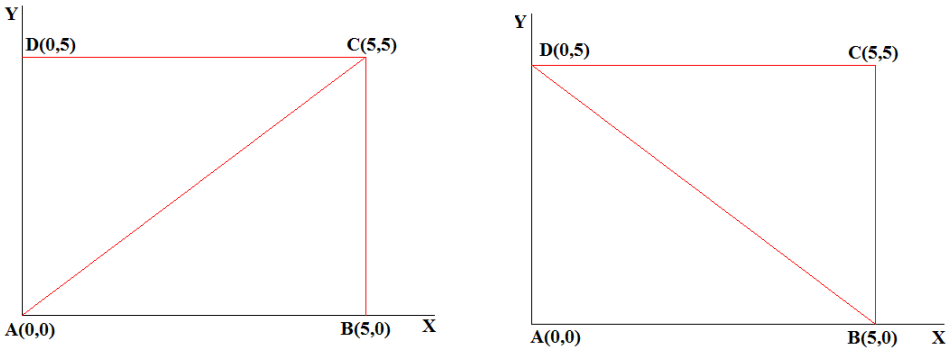


Рис.68 Разбиение области интегрирования

Определим матрицы X и Y для первого и второго случая.

1.) $$X=\begin{bmatrix}0 & 0\\ 5 & 5\\ 5 & 0\end{bmatrix}, Y=\begin{bmatrix}0 & 0\\ 0 & 5\\ 5 & 5\end{bmatrix}$$

2.) $$X=\begin{bmatrix}0 & 5\\ 5 & 5\\ 0 & 0\end{bmatrix}, Y=\begin{bmatrix}0 & 0\\ 0 & 5\\ 5 & 5\end{bmatrix}$$

Вычислим значение интеграла для обоих случаев.

```
deff('z=f(x,y)','z=cos(x+y)')
// Треугольники ABC-ACD:
X = [0,0;5,5;5,0]; Y = [0,0;0,5;5,5];
[Int,er] =int2d(X,Y,f)
-->[Int,er]
 ans  =
    0.4063959    1.059D-09
// Треугольники ABD-BCD:
X = [0,5;5,5;0,0]; Y = [0,0;0,5;5,5];
[Int,er] =int2d(X,Y,f)
-->[Int,er]
 ans  =
    0.4063959    1.059D-09
```

Как было отмечено выше, в Scilab возможно вычисление интегралов от комплексных функций с помощью функций **intc()** и **intl()** . **Функция intc()** позволяет вычислить интеграл от любой аналитической комплексной функции f(z) между комплексными числами z1 и z2 вдоль прямой линии, связывающей эти числа. Функцию можно вызвать следующим образом: **[y]=intc(z1,z2,f)**, где f является комплексной функции f(z).

Пример 10. Вычислить интеграл от комплексной функции *f(z)=1/z* между точками $z_1=2+3i$ и $z_2= -5+4i$ вдоль прямой линии, объединяющей эти точки.

```
-->deff('[w]=f(z)','w=1/z');

-->intc(2+3*%i,-5+4*%i,f)
 ans  =

   0.5743114 + 1.484058i
```

Функция intl(a,b,z0,r,f) позволяет вычислить интеграл от комплекснозначной функции f(z) по dz вдоль кривой на комплексной плоскости, определённой как z0 + r·exp(it) для a ≤t ≤b (часть окружности с центром z0 и радиусом r с фазой между a= θ_1и b= θ_2).

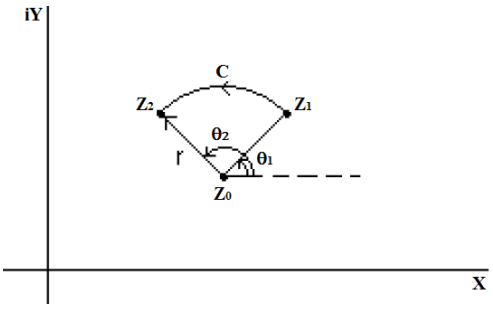


Рис.69 Область интегрирования

Пример 11. Вычислить интеграл от комплексной функции *f(z)=1/z+z* вдоль кривой, определяемой параметрами $\theta_1=\pi/6$ и $\theta_2=\pi/4$ для центра $z_0=2$ с радиусом r=1.

```
deff('w=f(z)','w=1/z+z')
intl(%pi/6,%pi/4,2,1,f)
ans  =
 - 0.6068734 + 0.5639769i
```

Задачи.

1. Вычислить значение производной функции $f(x)=(x+5)^7+2x^2+5x$ в точке $x=0.8$, используя функцию **diff()**.

2. Вычислить значение производной функции $f(x)=sin(x^2+5)+2xcos(x)$ в точке $x=45°$, используя функцию **diff()**.

3. Вычислить значение производной функции $f(x)=x^2+2sin(x)cos(x)+e^x$ в точке $x=\pi/4$, используя функцию **numdiff().**

4. Вычислить значение производной функции $f(x)=tg(x^2-1)+lg(x+1)$ в точке $x=\pi/4$, используя функцию numdiff().

5. Найти первую и вторую производную для следующего полинома $p(x)=x^3+5x^2-x+5$, используя функцию **derivat()**.

6. Найти первую и вторую производную для следующего полинома $p(x)=x^7-5x^4-x^2+5x$, используя функцию **derivat()**. Построить график функции и второй производной.

7. Вычислить значение производной функции $f(x_1,x_2,x_3)=2cos(x_1x_2) + exp(x_2+x_1)+x_3x_1$ в точке $x_1=1$, $x_2=2$, $x_3=3$, используя функцию **numderivative().**

8. Вычислить значение производной функции $f(x_1,x_2,x_3,x_4)=2x_1x_2 + x_1+x_2+x_3+x_4+x_1x_2x_3x_4$ в точке $x_1=1$, $x_2=2$, $x_3=2$, $x_4=1$, используя функцию **numderivative().**

9. Вычислить значение интеграла $\int_0^1(e^{4x}+x)dx$, используя функцию **inttrap()** (метод трапеций).

10. Вычислить значение интеграла $\int_0^5(3e^x-\sin(x))dx$, используя функцию **integrate().**

11. Вычислить интеграл $\int_0^1\left(tg(x^{2+x})\frac{x}{x+1}+x\right)dx$, используя функцию **intg()**.

12. Вычислить интеграл $\int_0^{10}\left((x^{2+x})\frac{x}{x+2}+x^x\right)dx$, используя функцию **intg()**.

13. Вычислить интеграл $\int_0^1(e^{2x}-e^{-2x})dx$, используя функцию **intsplin().**

14. Вычислить интеграл $\int_0^1 \left(e^{\sin(x)} + x\right)dx$, используя функцию **intsplin().**

15. Выполнить численное интегрирование дискретно заданной функции

$y(x)=tg(x^x)\frac{x}{x+1}+x$ в интервале от1 до 6.

16. Выполнить численное интегрирование дискретно заданной функции

$y(x)=\frac{2}{3}(20-x^2)^{\frac{5}{2}}$ в интервале от 0 до 6.

17. Вычислить интеграл от функции $f(x,y)=sin(x^2+y^2)$ в прямоугольной области R={0 <x<1, 0<y< 1}.

18. Вычислить интеграл от функции $f(x,y)=x^2+y^2$ в треугольной области R={0 <x<2, 0<y< 1}.

19. Вычислить интеграл от комплексной функции $f(z)=log(z)$ между точками $z_1=2$ и $z_2= 3$ вдоль прямой линии, объединяющей эти точки.

20. Вычислить интеграл от комплексной функции $f(z)=z+2/z^2$ между точками $z_1=2-4i$ и $z_2= 3-i$ вдоль прямой линии, объединяющей эти точки.

Численное решение обыкновенных дифференциальных уравнений

7

Любую ситуацию, где рассматривается степень изменения одной переменной по отношению к другой переменной, можно попытаться описать с помощью дифференциального уравнения. Дифференциальными называются уравнения, содержащие одну или несколько производных. В зависимости от числа независимых переменных в дифференциальных уравнениях различают обыкновенные дифференциальные уравнения, содержащие одну независимую переменную, и уравнения в частных производных, содержащие несколько независимых переменных.

В общем случае, обыкновенными дифференциальными уравнениями (ОДУ) называют такие уравнения, которые содержат одну или несколько производных от искомой функции $y = y(x)$. Их можно записать в виде

$$F(x,y,y',y'',\ldots y^{(n)})=0 \ ,$$

где x – независимая переменная.

Лишь небольшое число ОДУ удаётся решить без помощи компьютеров. Поэтому, численные методы решения дифференциальных уравнений играют важную роль в практике расчетов.

Если n раз дифференцируемая функция $y = \varphi(x)$ после подстановки в ОДУ n-го порядка обращает его в тождество, то она называется его решением. А график функции $y = \varphi(x)$ называют интегральной кривой. Как известно, общее решение ОДУ $y = \varphi(x, c_1,c_2,..c_n)$ содержит n произвольных постоянных $c_1,c_2,..c_n$, которые могут принимать любые значения, т.е. исходное уравнение имеет бесконечное множество решений. Единственные (частные) решения получают с помощью дополнительных условий, которым должны удовлетворять искомые решения. Тогда постоянные $c_1, c_2, .. c_n$, получают конкретные значения.

Способ задания дополнительных условий для получения частного решения ОДУ определяет тип постановки задачи. Рассматриваются три типа задач: задача Коши, краевая задача и задача на собственные значения.

Когда дополнительные условия задаются *в одной точке*, то такая задача называется *задачей Коши*. Дополнительные условия в задаче Коши называются начальными условиями, а точка $x = x_0$, в которой они задаются, – начальной точкой. Начальные условия задаются в следующем виде:

$$y(x_0) = y_0, y'(x_0) = y_{1,0},\ldots y^{(n-1)}(x_0)=y_{n-1,0}$$

Когда дополнительные условия задаются *более чем в одной точке*, т.е. при разных значениях независимой переменной, то такая задача называется *краевой*. Дополнительные условия в таких случаях называются граничными (или краевыми) условиями. Обычно граничные условия задаются в двух точках, являющихся границами отрезка, на котором рассматривается решение ОДУ. Минимальный порядок ОДУ, для которого может быть сформулирована краевая задача, равен двум.

Третий тип задач – это *задачи на собственные значения*. Такие задачи отличаются тем, что кроме искомых функций $y(x)$ и их производных в уравнения входят дополнительно m неизвестных параметров $\gamma_1, \gamma_2, \ldots \gamma_m$, которые называются *собственными значениями*. При этом для нахождения единственного (частного) решения на интервале [a,b] необходимо задать $n+m$ граничных условий.

Используются следующие методы решения ОДУ : аналитические, приближенные и численные. В аналитических методах решение получают в виде формул путем аналитических преобразований. Такие методы используются для уравнений первого порядка (с разделяющимися переменными, однородных, линейных), и для некоторых уравнений высших порядков (например, линейных с постоянными коэффициентами).

В приближенных методах используются различные упрощения самих уравнений путем обоснованного пренебрежения некоторыми их членами.

Численные методы решения дифференциальных уравнений используются, когда невозможно получить аналитическое или приближённое решение.

Для численного решения ОДУ разработано достаточное количество методов. Довольно часто используют метод Эйлера и его модификации, которые являются частными случаями однопараметрического семейства схем Рунге-Кутты различного порядка точности. Необходимо отметить, что метод Рунге-Кутты требует существенно большего объема вычислений по сравнению с методом Эйлера и его модификациями, но это позволяет получать решения с большей точностью при большем шаге. Т.е. для получения результатов с одинаковой точностью в методе Эйлера потребуется значительно меньший шаг, чем в методе Рунге-Кутты. На практике широкое распространение получил метод Рунге-Кутты четвертого порядка точности.

В Scilab предусмотрен широкий набор функций для получения решения не только ОДУ, но и дифференциальных алгебраических уравнений и их систем. Перечислим некоторые из них: **ode()**, **odedc()**, **dassl()**, **impl().**

Необходимо отметить, что функция **ode()** является программой решения обыкновенных дифференциальных уравнений и обеспечивает интерфейс для различных программ решения, в частности для ODEPACK.

Исходя из типа решаемой задачи и используемого метода, выбирается программа решения и задаётся в первом необязательном аргументе type при вызове функции **ode().** Значения аргумента **type**:

Если аргумент не задан, то вызывается по умолчанию программа решения lsoda из пакета ODEPACK. Она автоматически выбирает между нежёстким методом прогноза-исправления Адамса (predictor-corrector Adams method) и жёстким методом обратной дифференциальной формулой (ОДФ) (Backward Differentiation Formula (BDF) method). Изначально она применяет нежёсткий метод и динамически проверяет данные для того, чтобы решить какой метод использовать.

"adams":

Используется для нежёстких задач. Вызывается программа решения lsode из пакета ODEPACK, и при этом используется метод Адамса.

"stiff":

Предназначена для решения жёстких задач. Вызывается программа решения lsode из пакета ODEPACK, используется метод ОДФ.

"rk":

Адаптивный метод Рунге-Кутты 4-го порядка (RK4).

"rkf":

Используется программа Шампайна и Уотса (Shampine и Watts), основанная на методе Рунге-Кутты-Фелберга пары 4 и 5-го порядка (RKF45). Она для нежёстких и среднежёстких задач. Этот метод обычно не используют, если требуется высокая точность.

"fix":

Та же программа решения, что и "rkf", с простым пользовательским интерфейсом, т. е. программе решения могут быть переданы только параметры rtol и atol. Это самый простой метод для пробных вычислений.

"root":

Программа решения ОДУ с возможностью нахождения корней. Используется программа решения lsodar из пакета ODEPACK. Является вариантом программы решения lsoda, где о ищутся корни заданной векторной функции (см. ode_root).

"discrete":

Моделирование дискретного времени (см. ode_discrete).

Как видно из выше изложенного, функция **ode()** может иметь очень много разнообразных параметров. Поэтому, часто бывает удобно, устанавливать режим солвера ОДУ (**ode**) с помощью функции **odeoptions()**, меняя параметры переменной **%ODEOPTIONS**.

Необходимо обратить внимание, что функция **odeoptions()** должна быть набрана обязательно строчными буквами, а не заглавными, в то время как название переменной **%ODEOPTIONS** состоит из заглавных букв.

После выполнения функции **odeoptions()** появится окно, где в диалоговом режиме будет предложена последовательность форм. Заполнив которые, можно ввести параметры вычисления для решения ОДУ. Содержание формы показывает предварительно установленные параметры.

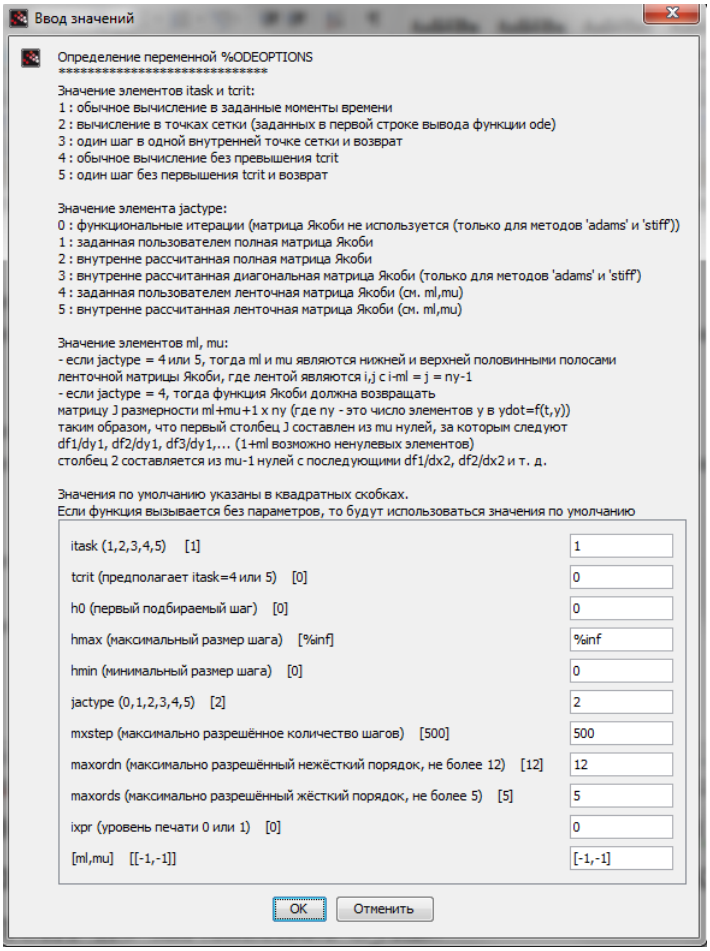


Рис.70 Форма для изменения параметров при решении ОДУ.

Функция **ode** проверяет существование переменной **%ODEOPTIONS** и, если она она существует, то использует её. Для использования значений по умолчанию, необходимо очистить эту переменную командой **clear %ODEOPTIONS**. Для создания этой

переменной необходимо выполнить инструкцию **%ODEOPTIONS=odeoptions()**. Значение %ODEOPTIONS по умолчанию:

%ODEOPTIONS =[1,0,0,%inf,0,2,500,12,5,0,-1,-1]

Пример 1. Найти методом Эйлера решение для ОДУ: $dy/dx = g(x,y) = x^{0.5} + y^2$ с начальными условиями: $y_0 = 0.0$, $x_0 = 0.0$. Переменная x задана на отрезке $x_0 = 0$, $x_n = 2.0$ с шагом $Dx = 0.1$. Построить график интегральной функции.

```
deff('[Df]=g(x,y)','Df=x^0.5+y^2')
x0 = 0; y0 = 0; Dx = 0.1; xn = 2.0;
x=[x0:Dx:xn]; y = zeros(x); n = length(y);
y(1)=y0+Dx*g(x0,y0);
for j = 1:n-1
y(j+1) = y(j) + Dx*g(x(j),y(j));
end;
plot(x,y,'-r')
plot(x,y,'d')
```

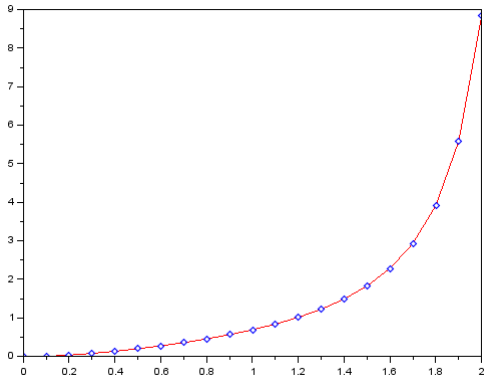


Рис.70 Интегральная кривая

Пример 2. Найти решение ОДУ $dy/dx = g(x,y) = x^{0.5} + y^{0.5}$ с начальными условиями: $y_0 = 0.0$, $x_0 = 0.0$, используя функцию **ode().** Переменная x задана на отрезке $x_0 = 0$, $x_n = 2.0$ с шагом $Dx = 0.1$. Сравнить с решением, полученным методом Эйлера. Построить графики интегральной функции для обоих методов.

```
//использование ode
deff('[z]=f(x,y)','z=x^0.5+y^0.5')
x0 =0; y0 = 0; Dx =0.1; xn =2; x=[x0:Dx:xn];
y = ode(y0,x0,x,f);
plot(x,y)
//метод Эйлера
deff('[Df]=g(x,y)','Df=x^0.5+y^0.5')
x0 = 0; y0 = 0; Dx = 0.1; xn = 2.0;
x=[x0:Dx:xn]; y = zeros(x); n = length(y);
y(1)=y0+Dx*g(x0,y0);
for j = 1:n-1
y(j+1) = y(j) + Dx*g(x(j),y(j));
end;
plot(x,y,'-r')
plot(x,y,'d')
```

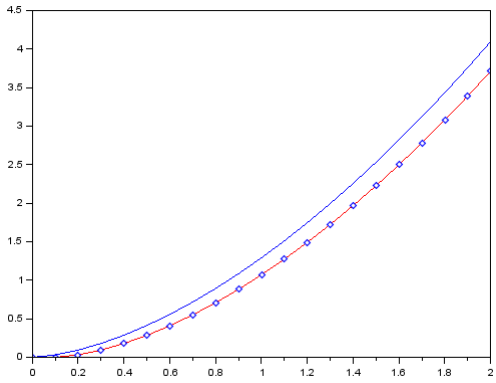


Рис.71 Интегральные кривые для двух методов

Пример 3. Получить решение системы дифференциальных уравнений 1-го порядка

$$\begin{cases} \frac{dy_1}{dx} = y_2 + 2x^2 \\ \frac{dy_2}{dx} = -y_1 + x^2 \end{cases}$$

при начальных условиях: y_1=1, y_2 = 2, для x_0 = 0 и построить графики интегральных кривых.

```
deff('[w]=f(x,y)',['y1=y(2)+2*x^2','y2=-y(1)+x^2','w = [y1;y2]'])
x0 = 0; Dx = 0.1; xn = 2; y0 = [1;2];
x = [x0:Dx:xn]; y = ode(y0,x0,x,f);
plot2d([x',x'],[y(1,:)',y(2,:)'],[1,-1],'111','y1@y2',[0 -3 2 6])
```

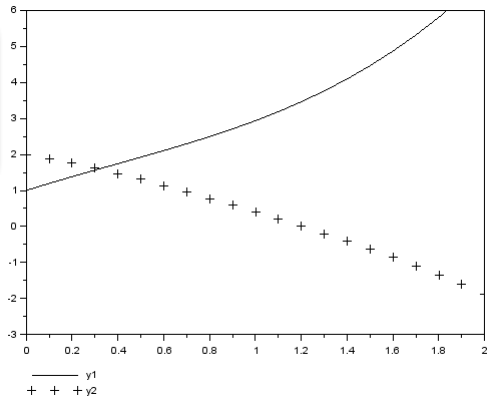


Рис.72 Интегральные кривые для системы

. Решить дифференциальное уравнение второго порядка:

Пример 4 $$\frac{d^2y}{dx^2} + \frac{2dy}{dx} - 3y = 2x$$

в диапазоне 0<x<2 с начальными условиями y(0) = 1, y'(0)=1, и построить график интегральной кривой.

Преобразуем уравнение в систему из двух линейных ОДУ первого порядка

$$\begin{cases} \frac{dy}{dx} = u \\ \frac{du}{dx} = 2x + 3y - 2u \end{cases}$$

```
deff('[w]=f(x,y)',['y1=y(2)','y2=3*y(1)+2*x-2*y(2)','w = [y1;y2]'])
x0 = 0; Dx = 0.1; xn = 2; y0 = [1;1];
x = [x0:Dx:xn]; y = ode(y0,x0,x,f);
plot2d([x',x'],[y(1,:)',y(2,:)'],[1,-1],'111','y1@y2',[0 -3 2 6]
```

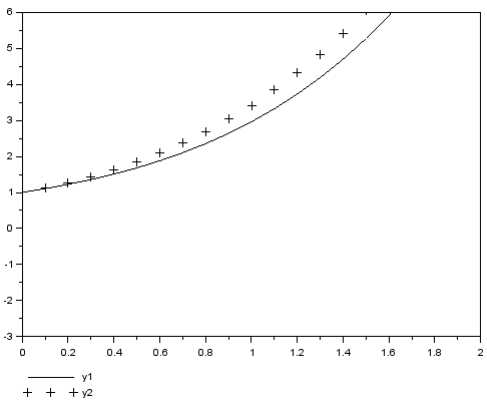


Рис.73 Интегральные кривые для уравнения

Пример 5. Решить дифференциальное уравнение второго порядка:

$$\frac{d^2y}{dt^2} = \frac{1}{1+t} + \sin(t)\sqrt{t}$$

с начальными условиями: y(0)=0, $\frac{dy(0)}{dt} = -2$.

Преобразуем уравнение в систему из двух линейных ОДУ первого порядка.

$$\begin{cases} \frac{dy}{dt} = z \\ \frac{dz}{dt} = \frac{1}{1+t} + \sin(t)\sqrt{t} \end{cases}$$

У нас есть два вектор столбца:

$$dx = \begin{bmatrix} \frac{dy}{dt} \\ \frac{dz}{dt} \end{bmatrix}; x = \begin{bmatrix} z \\ \frac{1}{1+t} + \sin(t)\sqrt{t} \end{bmatrix}$$

Приведём код в Scilab и интегральную кривую.

```
//зададим дифф. уравнение как функцию
function dx = f(t, x)
 dx(1) = x(2);
 dx(2) = 1/(t+1) + cos(t)*sqrt(t);
endfunction
// зададим область интегрирования t, начальные условия y0,t0
t = 0:0.01:10*%pi;
t0 = min(t);
y0 = [0; -1];
//вызов ode
y = ode(y0, t0, t, f);
//график интегральной кривой
plot(t,y(1,:),'LineWidth',2)
plot(t,y(2,:),'r','LineWidth',2)
xgrid();
xlabel('$t \quad [c]$','FontSize',3)
ylabel('$f(t,x)$','FontSize',3)
title(['Интегрирование' '$\frac{d^2 x}{dt^2} = \frac{1}{t+1}..
+ cos(t)\sqrt{t}$'],'FontSize',3)
legend(['$\Large{x}$' '$\Large{dx/dt}$'],2)
```

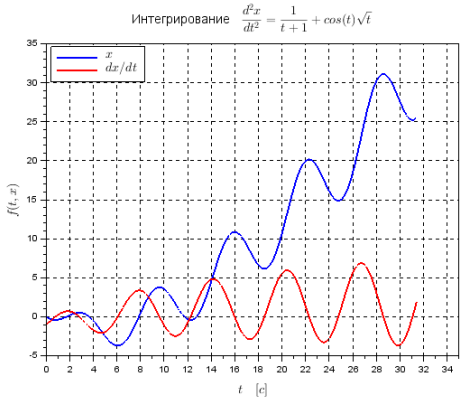


Рис.74 Интегральные кривые для уравнения

Пример 6. Решить краевую задачу для дифференциального уравнение второго порядка $y'' = 10y^3 + 3y + t^2$, $0 \le t \le 1$, с граничными условиями y(0) = 0, y(1) = 1. Построить интегральную кривую.

Для решения этой задачи будем использовать метод стрельбы. Сущность метода стрельбы заключается в том, чтобы краевую задачу свести к многократному решению задачи Коши для того же уравнения. Необходимо подобрать недостающее начальное условие y'(0)=zz так, чтобы решение задачи Коши в точке t=1 совпадало бы с заданным граничным условием y(1) = 1 с заданной точностью eps.

```
function dy=fun1(x,y)
dy(1)=y(2);
dy(2)=10*y(1)^3+3*y(1)+x^2;
endfunction
//описание функции, решающей задачу Коши
function er=fun2(z)
y1z=1;
y0=[0;z];
x0=0; x=0:0.05:1.; y=ode(y0,x0,x,fun1);
n=length(x);
y1k=y(1,n);
er=abs(y1k-y1z);
endfunction
// решение с помощью функции fsolve
zz=fsolve(0,fun2)
//построить интегральную кривую
y01=[0;zz]
x0=0; x=0:0.05:1.;y=ode(y01,x0,x,syst);
plot(x,y(1,:))
za=[x' y']
```

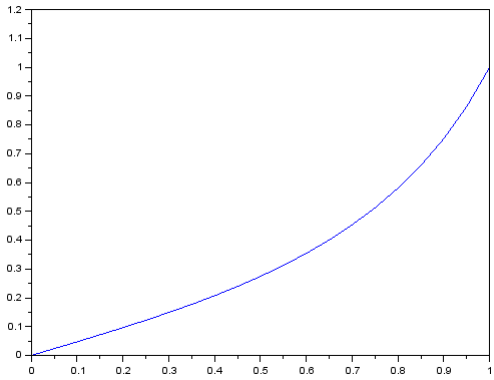


Рис.75 Интегральная кривая для уравнения

Пример 7. Задано дифференциальное уравнение *dy/dt = y*, с начальным условием y(0) = 1. найти минимальное значение t, при котором выполняется условие y(t)= 4. Проверить полученное значение, найдя решение ОДУ.

Воспользуемся опцией "roots" для функции **ode().**

```
deff('[ydot]=f(t,y)','ydot=y')
deff('[z]=g(t,y)','z=y-4')
//t=0:0.5:2;
y0=1;ng=1;t0=0;
[y,rd]=ode("roots",y0,t0,4,f,ng,g)
-->y
 y  = 4.
-->rd
       rd  = 1.3862941    1.
->exp(rd(1,1)
 ans  = 3.9999991
```

Пример 8. Задана линейная непрерывная система. Найти отклик данной системы на входной импульс в виде ступенчатой и импульсной функции.

$$\begin{cases} \dfrac{dx_1}{dt} = -2x_1 + x_2 + u \\ \dfrac{dx_2}{dt} = -8x_1 - x_2 \\ y = x_1 + x_2 \end{cases}$$

С помощью функции **csim()** можно изучить временной отклик линейной системы на входную функцию (при этом используется функция **ode()**). Вызов функции **csim()** выглядит следующим образом:

[y [,x]]=csim(u,t,sl,[x0 [,tol]])

u –входная функция, t –время симуляции отклика, причём t(1) –начальное время, y-матрица величин y(i)=[y(t(i)] для i = 1, . . . , n. tol –вектор из двух компонент [atol rtol], задающий абсолютную и относительную ошибку **ode**. Для u предусмотрены две специальные опции, такие как : "impuls" и "step" . Если задана опция "impuls" , то

изучается отклик системы на дельта-функцию, если "step" –то отклик на ступенчатую функцию при t = 0. Если u задана как "impuls",то предполагается D = 0, x0=0.

Если u задана как "step", то u = 1 при t > 0.

```
A=[-2 1;-8 -1];B=[1;0];C=[1 1 ];D=0;
[sl]=syslin("c",A,B,C);
t=0:0.01:5;
[ys,xs]=csim("step",t,sl);
plot2d(t,ys);
[yi,xi]=csim("impuls",t,sl);
f1=scf(1);
plot2d(t,yi)
```

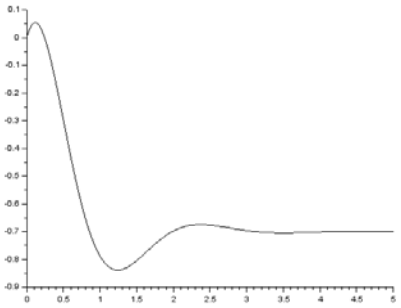


Рис.76 Отклик на ступенчатую функцию

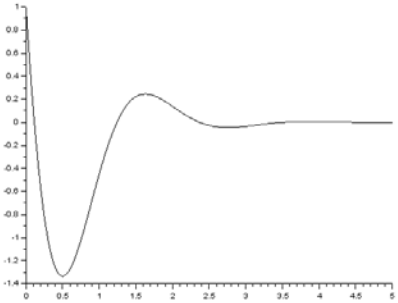


Рис.77 Отклик на импульсную функцию

Пример 9. Задана дискретная система вида y(k + 1) = f(k, y(k)), y(k0) = y0.Используя функцию ode_discrete, решить данное уравнение, если f(k, y(k))= y + cos(t/10)

Для решения воспользуемся функцией **y=ode("discrete",y0,k0,kvect,f).**

Необходимо обратить внимание, что начальное время k0 является целым числом и kvect, содержащий время при котором необходимо решения, является вектором из целых чисел. kvect(1) должен быть больше или равен k0.

```
function z=f(t,y)
z=y + cos(t/10)
endfunction
kvect=0:20;
y=ode("discrete",1,0,kvect,f);
plot2d2(kvect,y)
```

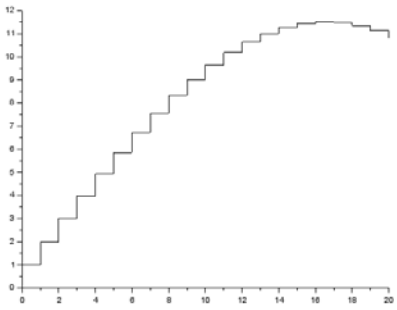


Рис.78 Интегральная кривая

Пример 10. Дано дифференциальное алгебраическое уравнение с начальными условиями. Получить решение и построить интегральную кривую.

$$tan\left(\frac{dy}{dt}\right) = -y + 30tcos(t); y(0) = 0, \frac{dy(0)}{dt} = 0$$

Подобные уравнения вида *F(x,y,y')=0*, где $F-$ непрерывная функция, называются ещё *уравнением первого порядка, не разрешенным относительно производной*. В отличие от обыкновенных дифференциальных уравнений для получения решения этого уравнения необходимо задать не только начальное значение y(t0), но и значение производной в этой точке *dy(t0)/dt*. Для получения решения таких уравнений в Scilab предусмотрена функция **dassl().**

```
x0=[0 0];
t0=0;
tt=0:0.1:10;
function [r,ires]=res(t,y,ydot)
r=tan(ydot)+y-30*t*cos(t)
ires=0
endfunction
r0=dassl(x0,t0,tt,res);
plot2d(r0(1,:),r0(2,:),style=-1);
```

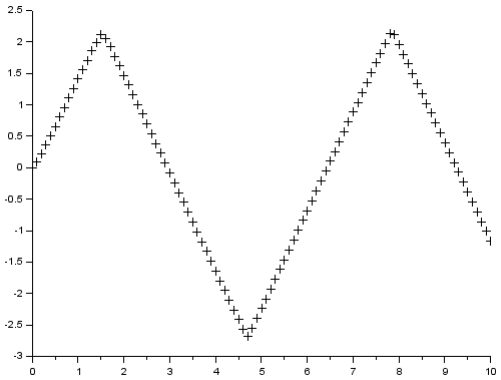


Рис.79 Интегральная кривая

Задачи.

1. Найти методом Эйлера решение для ОДУ: *dy/dx = g(x,y) = x + y* с начальными условиями: y0 = 0.0, x0 = 0.0. Переменная x задана на отрезке x0 = 0, xn = 2.0 с шагом Dx = 0.1. Построить график интегральной функции.

2. Найти методом Эйлера решение для ОДУ: *dy/dx = g(x,y) = e^x + y* с начальными условиями: y0 = 1, x0 = 0.0. Переменная x задана на отрезке x0 = 0, xn = 2.0 с шагом Dx = 0.1. Построить график интегральной функции.

3. Найти решение ОДУ *dy/dx = g(x,y) = sin(0.5x) + $y^{0.5}$* с начальными условиями: y0 = 0.0, x0 = 0.0, используя функцию ode(). Переменная x задана на отрезке x0 = 0, xn = 4.0 с шагом Dx = 0.1. Построить график интегральной функции.

4. Найти решение дифференциального уравнения, используя решатель ode.

$$(1+x^2)y'+1=0 \qquad y(0)=1 \qquad x(0)=0 \qquad [0\ ;\ 0.5]$$

5. Получить решение системы дифференциальных уравнений 1-го порядка

$$\begin{cases} \frac{dy_1}{dx} = 5y_2 + 2(x+1)^2 \\ \frac{dy_2}{dx} = -2y_1 + x^2 \end{cases}$$

при начальных условиях: y_1=1, y_2 = 2, для x_0 = 0 и построить графики интегральных кривых.

6. Получить решение системы дифференциальных уравнений 1-го порядка

$$\begin{cases} \frac{dy_1}{dx} = y_2 + 2y_1 + x \\ \frac{dy_2}{dx} = -y_1 + x \end{cases}$$

при начальных условиях: y_1=1, y_2 = 2, для x_0 = 0 и построить графики интегральных кривых.

7. Найти решение дифференциального уравнения, используя решатель ode.

$$8 \cdot y'' + 2 \cdot y' - 3 \cdot y = x + 5 \qquad y(0) = \frac{1}{9} \qquad y'(0) = -\frac{7}{2} \qquad [0\ ;\ 1]$$

8. Решить задачу Коши: y'' + 2y' +10y = sin x; y(0) = 1, y'(0) = 0, x=[0;10]; шаг h=0.01.

9. Решить краевую задачу для дифференциального уравнение второго порядка

y'' = $5y^2$ + y + t, 0 ≤ t ≤ 1, с граничными условиями y(0) = 0, y(1) = 1. Построить интегральную кривую.

10. Решить краевую задачу для дифференциального уравнение второго порядка

y'' = sin(3y) + $2t^2$, 0 ≤ t ≤ 1,с граничными условиями y(0) = 0, y(1) = 1. Построить интегральную кривую.

11. Задано дифференциальное уравнение $dy/dt = y^2+t$, с начальным условием $y(0) = 1$. Найти минимальное значение *t*, при котором выполняется условие $y(t)= 2$. Проверить полученное значение, найдя решение ОДУ.

12. Задано дифференциальное уравнение $dy/dt = sin(y)+t^2$, с начальным условием y(0) = 1. найти минимальное значение *t*, при котором выполняется условие y(t)= 0.5.

13. Задана линейная непрерывная система. Найти отклик данной системы на входной импульс в виде ступенчатой и импульсной функции.

$$\begin{cases} \dfrac{dx_1}{dt} = -2x_1 + u \\ \dfrac{dx_2}{dt} = -8x_1 - 2x_2 \\ y = x_1 + 2x_2 \end{cases}$$

14. Задана линейная непрерывная система. Найти отклик данной системы на входной импульс в виде ступенчатой и импульсной функции.

$$\begin{cases} \dfrac{dx_1}{dt} = -2 + x_2 + u \\ \dfrac{dx_2}{dt} = -x_2 \\ y = x_1 + x_2 \end{cases}$$

15. Задана дискретная система вида *y(k + 1) = f(k, y(k)), y(k0) = y0*. Получить решение данного уравнения, если *f(k, y(k))= y^2 + sin(t/2).*

16. Задана дискретная система вида *y(k + 1) = f(k, y(k)), y(k0) = y0*. Получить решение данного уравнения, если *f(k, y(k))= sin(y) + cos(t/2).*

17. Дано дифференциальное алгебраическое уравнение *$y=ln[25+(y')^2]$*, с начальными условиями: *y(0)=3.2188758, y'(0)=0*. Получить решение и построить интегральную кривую.

18. Дано дифференциальное алгебраическое уравнение *$9(y')^2-4x=0$*, с начальными условиями: *y(0)=0, y'(0)=0*. Получить решение и построить интегральную кривую.

19. Дано дифференциальное алгебраическое уравнение $2y=2x^2+4xy'+(y')^2$, с начальными условиями: x(0)=0, y'(0)=0. Получить решение и построить интегральную кривую.

20. Дано дифференциальное алгебраическое уравнение *$2y=2x^2+4xy'+(y')^2$*, с начальными условиями: y(0)=0.5, y'(0)=1. Получить решение и построить интегральную кривую.

Статистическая обработка эмпирических распределений 8

Математическая обработка выборки значений $x_1, x_2, \ldots, x_n$ случайной величины X встречается в любой работе, связанной с измерениями. По результатам обработки выборки определяют точечные и интервальные оценки параметров продукции, анализируют технологические процессы, определяют надёжность элементов и систем, оценивают истинные значения физических величин и т.д. Для получения как можно более точных оценок параметров необходимо оперировать с большим массивом данных. Scilab спроектирован для работы с матрицами, поэтому использует эффективные алгоритмы для работами с такими массивами и обладает большим набором функций для вычисления оценок, генерирования числовых последовательностей с заданными распределениями вероятности, операций манипулирования данными. Рассмотрим эти возможности.

8.1 Вычисление выборочных статистик

Для построения вариационного ряда из числового массива эмпирических данных используется функция **gsort()**:

Пример 1. Упорядочить массив данных: x = 1.5, 3.4, 7, 4.6, 11 y = 5.0, 30, 7.0, 40, 11.0. по возрастанию.

```
-->x=[1.5 3.4 7 4.6 11];
-->y=[5.0 30 7.0 40 11.0];
-->v=[x;y];
-->[vv,k]=gsort(v,'g','i');// по возрастанию
-->vv
 vv  =
   1.5   4.6   7.   11.   30.
   3.4   5.    7.   11.   40.
```

Пример 2. Упорядочить массив данных: x = 1.5, 3.4, 7, 4.6, 11 y = 5.0, 30, 7.0, 40, 11.0. по убыванию.

```
-->x=[1.5 3.4 7 4.6 11];
-->y=[5.0 30 7.0 40 11.0];
-->v=[x;y];
-->[vv,k]=gsort(v,'g','d');// по убыванию
-->vv
   40.   11.   7.   5.    3.4
   30.   11.   7.   4.6   1.5
```

Пример 3. Вычислить выборочное среднее числовых данных, заданных матрицей

$$M = \begin{pmatrix} 1 & 2 & 100 \\ 70 & 17.1 & 7.64 \end{pmatrix}$$

```
-->M=[1,2,100;70,17.1,7.64];
-->C=mean(M) // среднее всех
элементов
 C  =
   32.956667
-->C=mean(M,'r')// среднее по столбцам
 C  =
   35.5    9.55    53.82
-->C=mean(M,'c')) // среднее по рядам
 C  =
   34.333333
   31.58
```

Пример 4. Даны результаты двух серий измерений: x_1 = 0.311 0.0020 0.6601;

x_2 = 0.8560 0.5303 0.0286

и матрица весов $\mathbf{f}=\begin{pmatrix} 12 & 15 & 12 \\ 2 & 2 & 2 \end{pmatrix}$. Найти средневзвешенное выборочных средних.

```
-->x1= [0.3111  0.0020  0.6601];
-->x2= [0.8560   0.5303   0.0286];
-->x=[x1;x2] //матрица измерений
 x  =
   0.3111    0.002     0.6601
   0.856     0.5303    0.0286
-->f=[12 15 12;2 2 2] //матрица весов
 f  =
   12.    15.   12.
    2.     2.    2.
-->meanf(x,f) // средневзвешенное
 ans  =
   0.3225378
```

Пример 5. Пусть имеется матрица x из примера 4. Найти медиану для этих данных и построить гистограмму, чтобы визуально сравнить расчёты.

```
-->x1= [0.3111  0.0020  0.6601];
-->x2= [0.8560   0.5303   0.0286];
-->x=[x1;x2] //матрица измерений
 x  =
   0.3111    0.002     0.6601
   0.856     0.5303    0.0286
-->y=median(x)
 y  =
   0.4207
-->histplot(8,x);
```

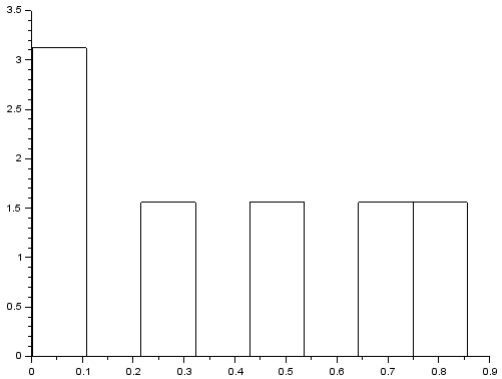


Рис.80 Гистограмма

Из рисунка 80 видно, что площадь двух левых столбцов равна площади трёх правых столбцов, что совпадает с полученным результатом y = 0.4207

Пример 6. Вычислить (среднее квадратичное отклонение) стандартное отклонение выборочных значений 0.211 0.122 0.667 0.766 0.335 0.648.

С помощью функции **stdev()** вычислим стандартное отклонение.

```
-->x=[0.211  0.122  0.667  0.766  0.335  0.648]
 x  =
   0.211    0.122    0.667    0.766    0.335    0.648
-->s=stdev(x)
 s  =
   0.2696957
```

Пример 7. Вычислить средневзвешенное стандартное отклонение для результатов измерений 0.211 0.221 0.381 0.439 0.254 0.918 с весами 1 2 1 4 4 3.

Средневзвешенное стандартное отклонение вычислим, используя функцию **stdevf ().**

```
-->x=[0.211  0.221  0.381  0.439  0.254  0.918 ]
 x  =
   0.211    0.221    0.381    0.439    0.254    0.918
-->fre=[1 2 1 4 4 3];
-->s=stdevf(x,fre)
 s  =
    0.2639775
```

Пример 8. Вычислить дисперсию для следующих результатов измерений: 2.1, 1., 6.1, 1.7, 1.4, 1.6.

Дисперсия значений вектора или матрицы вычисляется с помощью функции **variance()**.

```
-->x=[2.1 1.0 6.1 1.7 1.4 1.6]
 x  =
   2.1    1.    6.1    1.7    1.4    1.6
-->d=variance(x)
 d  =
   3.5656667
```

Пример 9. Определить частоты появления значений 2, 3, 2, 3, 2, 5, 4, 4, 4, 5.

В Scilab для этой цели существует специальная функция **tabul()**.

```
-->x=[2 3 2 3 2 5 4 4 4 5]
 x  =
   2.    3.    2.    3.    2.    5.    4.    4.    4.    5.
-->t=tabul(x)
 t  =
   5.    2.
   4.    3.
   3.    2.
   2.    3.
```

Пример 10. Получены данные: 1., 2., 2., 3.,1., 4., 3., 3., 4., 4. Необходимо сгруппировать по интервалам и определить частоты появления значений.

Для этого спользуем формулу Стерджесса для нахождения количества интервалов и найдём количество попаданий в каждый интервал, используя функцию

[cf, ind] = histc(n, data [,normalization]) (введена начиная с версии Scilab 5.5.0).

```
-->x=[1., 2., 2., 3.,1., 4., 3., 3., 4., 4];
-->n=length(x)
-->m=1+3.32*log10(n) //формула Стерджесса, количество интервалов
 m  =
   4.32
-->[cf, ind] = histc(4, x, normalization=%f)
 ind  =
   1.   2.   2.   3.   1.   4.   3.   3.   4.   4.
 cf  =          //частота значений в каждом из интервалов
   2.   2.   3.   3.
```

Задачи.

1.Упорядочить массив данных: x = 1.5, 3.5, 9, 5.6, 10; y = 50, 30, 17, 140, 110.;

z = 1., 3., 7.5, 5, 13 по возрастанию.

2.Упорядочить массив данных: x = 1.5, 3.5, 9, 5.6, 10; y = 50, 30, 17, 140, 110.;

z = 1., 3., 7.5, 5, 13; t=2, 4, 6, 1, 0.1 по возрастанию. Определить размерность матрицы.

3.Упорядочить массив данных: x = 1.5, 3.5, 9, 5.6, 10; y = 50, 30, 17, 140, 110.;

z = 1., 3., 7.5, 5, 13 по убыванию.

4.Упорядочить массив данных: x = 0.2113249 0.7560439 0.0002211 0.3303271 0.6653811 по убыванию.

5. Вычислить выборочное среднее числовых данных, заданных матрицей

$$M=\begin{bmatrix} 0.62 & 0.87 & 0.66 & 0.54 & 0.21 \\ 0.84 & 0.06 & 0.72 & 0.23 & 0.88 \\ 0.68 & 0.56 & 0.19 & 0.23 & 0.65 \end{bmatrix}$$

6. Вычислить выборочное среднее числовых данных, заданных матрицей

$$\begin{bmatrix} 2.72 & -1.70 & 0.06 \\ -0.71 & 0.95 & -2.87 \end{bmatrix}$$

7. Даны результаты двух серий измерений: x_1 = - 1.37, - 0.17, - 0.60;

x_2 = 0.79, 0.76, - 0.02

и матрица весов $\mathbf{f}=\begin{pmatrix} 1 & 5 & 1 \\ 1 & 3 & 2 \end{pmatrix}$. Найти средневзвешенное выборочных средних.

8. Даны результаты серии измерений:

x: -1.56, -0.56, -0.38, -0.65, 0.65, -0.67, -0.70, 0.33, -0.82, 0.46 и вектор весов f=1,1,2,2,3,2,1,4,3,1. Найти средневзвешенное выборочных средних.

9. Даны результаты серии измерений 1.13,-0.31,-0.23, 0.97,-0.23,-0.18. Найти медиану для этих данных и построить гистограмму, чтобы визуально сравнить расчёты.

10. Пусть имеется матрица x из примера 5. Найти медиану для этих данных и построить гистограмму, чтобы визуально сравнить расчёты.

11. Вычислить стандартное отклонение выборочных значений 1.1,-0.3,-0.2,0.9,-0.2,-0.1.

12. Вычислить стандартное отклонение выборочных значений в двух сериях измерений. x_1: 0.34, 0.92, 0.34, 0.73, 0.49; x_2: 0.38, 0.94, 0.37, 0.26, 0.26

13. Вычислить средневзвешенное стандартное отклонение для результатов измерений 0.52, 0.53, 0.11, 0.22, 0.62, 0.76 с весами 1 2 1 1 2 1.

14. Вычислить средневзвешенное стандартное отклонение для результатов измерений 0.52, 0.53, 0.11, 0.22, 0.62, 0.76 с весами 1 1 1 1 1 1. Сравнить результат с результатом, вычисленным для стандартного отклонения без использования весов.

15. Вычислить дисперсию для следующих результатов измерений: 0.04, 0.67, 0.20, 0.39, 0.83, 0.58, 0.48, 0.22, 0.84, 0.12.

16. Вычислить дисперсию для следующих результатов измерений: - 0.42, 0.71,1.18, 0.08, 1.50, - 0.54, 0.32,1.04.

17. Определить частоты появления значений 0.75, 0.11, 0.28, 0.28,- 0.66, 0.08, - 0.66, 0.37.

18. Определить частоты появления значений: 0.77, 0.35, 0.08, 0.77, 0.08, 0.59, 0.30, 0.35, 0.62, 0.11.

19. Определить частоты появления значений: 0.77, 0.35, 0.08, 0.77, 0.08, 0.59, 0.30, 0.35, 0.62, 0.11. Построить гистограмму, найти частоты появления в каждом интервале.

20. Определить частоты появления значений. Построить гистограмму, найти частоты появления в каждом интервале. Значения: 0.99,1.58, 0.39, - 2.43,-1.62, 0.04, - 0.61,- 1.68, - 0.75, 0.22.

8.2 Генерирование последовательности значений с заданным распределением вероятности.

В Scilab имеются генераторы псевдослучайных чисел, с помощью которых можно получать случайные числа с широким набором вида функций распределения. Для этих целей можно использовать следующие функции : **grand()** и **rand()**.

Пример 1. Сгенерировать последовательность из 10 независимых одинаково распределённых чисел, равновероятных на интервале (0,1). Построить гистограмму этой последовательности. Вычислить среднее и среднее квадратичное отклонение.

```
-->x=rand(1,10)
 x  =
    0.3454984    0.7064868    0.5211472    0.2870401
0.6502795    0.0881335  0.4498763    0.7227253    0.8976796
0.2427822
-->histplot(4,x)
 ans  =
   1.482312    0.9882080    0.9882080    1.482312
-->mean(x)
 ans  =
   0.4911649
-->stdev(x)
 ans  =
 0.2538970
```

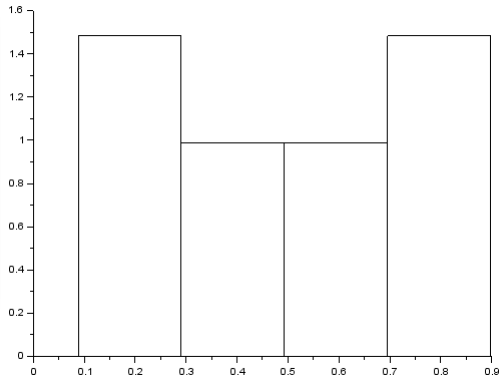


Рис.81 Гистограмма

Пример 2. Сгенерировать матрицу размером 10x10 из независимых нормально распределённых случайных чисел, с математическим ожиданием 0 и дисперсией 1. Построить гистограмму этой последовательности. Вычислить среднее и дисперсию.

```
-->x=rand(10,10,'normal');
-->mean(x)
 ans  =
  - 0.0385853
-->variance(x)
 ans  =
    0.9642236
-->histplot(12,x);
```

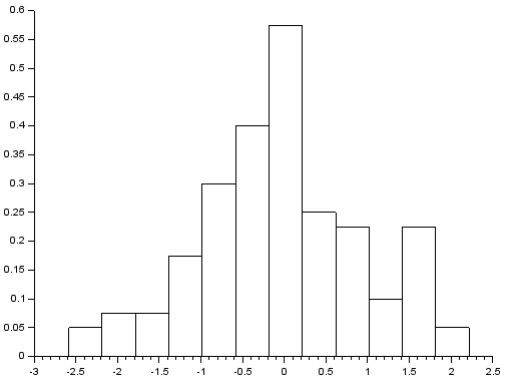


Рис.82 Гистограмма нормального распределения

Пример 3. Сгенерировать матрицу размером 400x500 из независимых нормально распределённых случайных чисел со средним Av =0 и стандартным отклонением Sd = 1. Построить гистограмму этой последовательности. Вычислить среднее и дисперсию.

```
-->x =grand(400,500,'nor',0,1);
-->histplot(10,x);
-->mean(x)
 ans  =
  - 0.0010875
-->variance(x)
 ans  =
   1.0039454
```

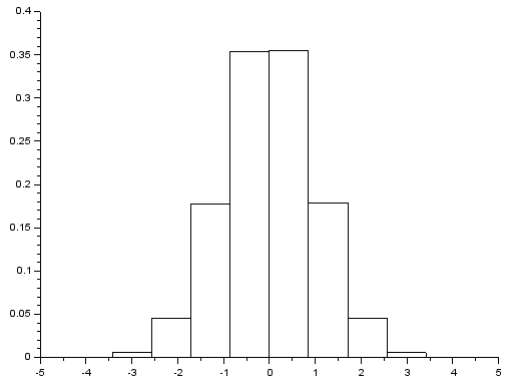


Рис.83 Гистограмма нормального распределения

Пример 4. Сгенерировать последовательность из 1000 независимых одинаково распределённых чисел, равновероятных на интервале (-1,1).

```
-->R = grand(1,1000,'unf',-1,1);
-->histplot(15,R);
-->mean(R)
 ans  =
  - 0.0101266
-->variance(R)
 ans  =
   0.3071423
```

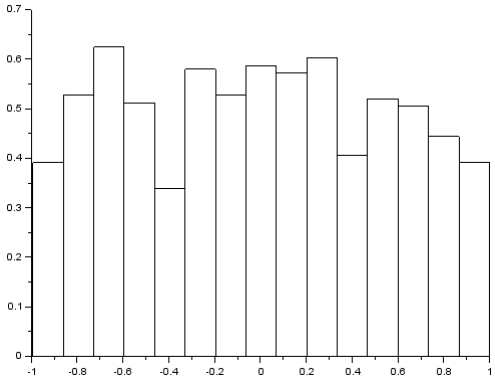


Рис.84 Гистограмма однородного распределения

Пример 5. Сгенерировать экспоненциальное распределение. Построить теоретическую кривую и гистограмму.

```
lambda=3;
N=10000;
X = grand(1,N,"exp",lambda);
scf();
classes = linspace(0,12,25);
histplot(classes,X)
x=linspace(0,12,25);
y = (1/lambda)*exp(-(1/lambda)*x);
plot(x,y,"rd-");
legend(["Эксперимент" "Теория"]);
```

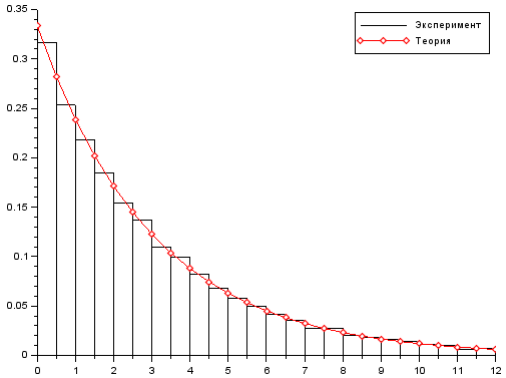


Рис.85 Гистограмма экспоненциального распределения

Пример 6. Сгенерировать распределение Пирсона χ^2 с десятью степенями свободы.

```
-->R=grand(1,100000,'chi', 10);
-->histplot(20,R);
```

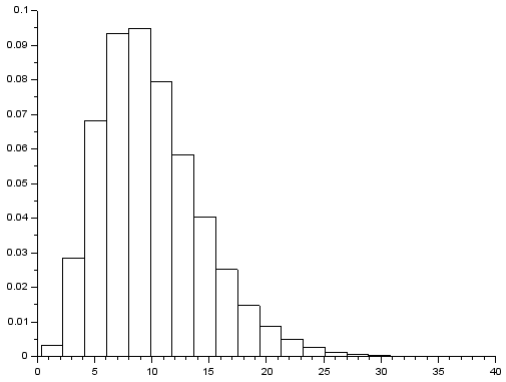


Рис.86 Гистограмма распределения Пирсона χ^2

Пример 7. Сгенерировать распределение хи-квадрат с 1, 5, 20 степенями свободы. Построить графики плотности распределений.

```
-->R1=grand(1,100000,'chi', 1);
-->R2=grand(1,100000,'chi', 5);
-->R3=grand(1,100000,'chi', 20);
-->histplot(20,R1);
-->histplot(20,R2,style=10);
-->histplot(20,R3,style=5);
-->legend('n=1','n=5','n=20')
```

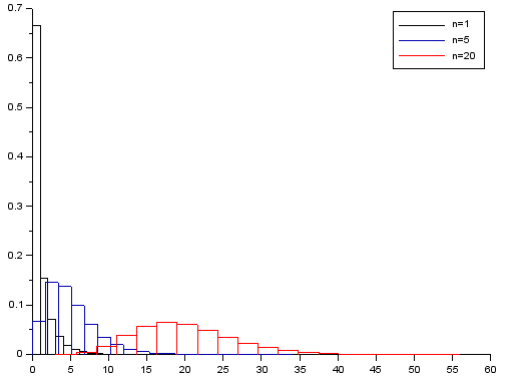


Рис.87 Гистограмма распределения Пирсона χ^2 с 1,5,20 степенями свободы

Пример 8. Сгенерировать нормальное распределение. Построить график плотности распределения и полигон.

```
->n=20;
-->data=rand(1,10000,"normal");
-->histplot(n, data, style=10, polygon=%t);
-->legend(["гистограмма распределения" "Полигон"]);
```

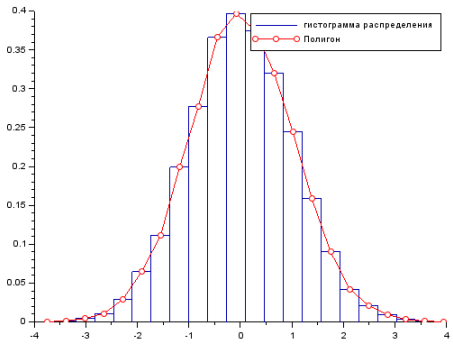


Рис.88 Гистограмма нормального распределения и частотный полигон

Пример 9. Сгенерировать распределение Пуассона с мат. ожиданием 2. Построить график плотности распределения и полигон.

```
-->R = grand(1,100000,"poi",2);
-->histplot(10,R,polygon=%t);
-->legend(["гистограмма распределения" "Полигон"]);
```

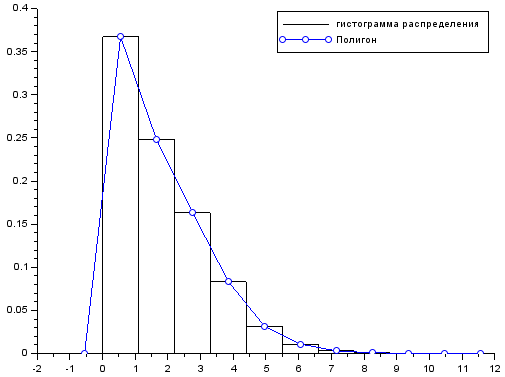


Рис.89 Гистограмма распределения Пуассона и частотный полигон

Пример 10. Сгенерировать однородное распределение для 10000 целых чисел в интервале (-10,10). Построить гистограмму распределения и полигон.

```
-->R = grand(1,10000,"uin",-10,10);
-->histplot(12,R,polygon=%t);
```

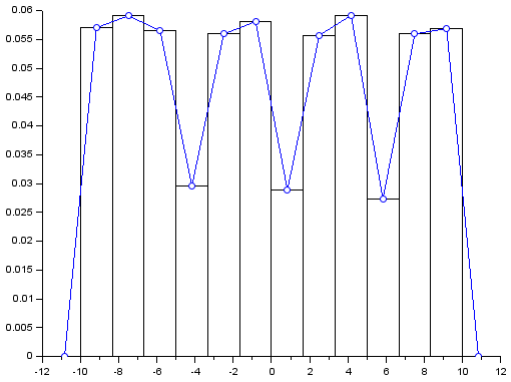


Рис.90 Гистограмма распределения и частотный полигон

Задачи.

1. Сгенерировать последовательность из 100 независимых одинаково распределённых чисел, равновероятных на интервале (0,1). Построить гистограмму этой последовательности. Вычислить среднее и среднее квадратичное отклонение

2. Сгенерировать матрицу (10,10) из независимых одинаково распределённых чисел, равновероятных на интервале (0,1). Построить гистограмму этой последовательности. Вычислить среднее и среднее квадратичное отклонение.

3. Сгенерировать матрицу размером 100x100 из независимых нормально распределённых случайных чисел, с математическим ожиданием 0 и дисперсией 1. Построить гистограмму этой последовательности. Вычислить среднее и дисперсию.

4. Сгенерировать матрицу размером 1x100 из независимых нормально распределённых случайных чисел, с математическим ожиданием 0 и дисперсией 1. Построить гистограмму этой последовательности. Вычислить среднее, дисперсию и среднее квадратичное отклонение.

5. Сгенерировать матрицу размером 40x200 из независимых нормально распределённых случайных чисел со средним Av =5 и стандартным отклонением Sd = 0.5. Построить гистограмму этой последовательности. Вычислить среднее, дисперсию и стандартное отклонение.

6. Сгенерировать матрицу размером 400x1 из независимых нормально распределённых случайных чисел со средним Av =1 и стандартным отклонением Sd = 2. Построить гистограмму этой последовательности. Вычислить среднее, дисперсию и стандартное отклонение.

7. Сгенерировать последовательность из 1000 независимых одинаково распределённых чисел, равновероятных на интервале (-5,5). Построить гистограмму этой последовательности. Вычислить среднее, дисперсию и стандартное отклонение.

8. Сгенерировать матрицу из 10x100 независимых одинаково распределённых чисел, равновероятных на интервале (-1,10). Построить гистограмму этой последовательности. Вычислить среднее, дисперсию и стандартное отклонение.

9. Сгенерировать экспоненциальное распределение с параметром λ=5. Построить теоретическую кривую и гистограмму. Вычислить среднее, дисперсию.

10. Сгенерировать матрицу 1x1000 из независимых экспоненциально распределённых случайных чисел с параметром λ=10. Построить теоретическую кривую и гистограмму. Вычислить среднее, дисперсию.

11. Сгенерировать последовательность из 500 случайных чисел с распределением Пирсона χ^2 с пятью степенями свободы. Вычислить среднее, дисперсию.

12. Сгенерировать матрицу 10x1000 из независимых случайных чисел с распределением Пирсона $\chi 2$ с двадцатью степенями свободы. Вычислить среднее, дисперсию.

13. Сгенерировать распределение хи-квадрат с 1, 10, 30 степенями свободы. Построить графики плотности распределений.

14. Сгенерировать распределение хи-квадрат с 1, 5, 20 степенями свободы. Построить графики плотности распределений. Вычислить среднее, дисперсию и медиану.

15. Сгенерировать матрицу чисел 10000x1 с нормальным распределением. Построить график плотности распределения и полигон.

16. Сгенерировать последовательность из 10000 чисел с нормальным распределением. Построить график плотности распределения и полигон. Вычислить среднее, дисперсию и медиану.

17. Сгенерировать распределение Пуассона с мат. ожиданием 2. Объём выборки N=100, 10000, 100000. Построить график плотности распределения и полигон для выборки N=100000. Вычислить среднее, дисперсию для каждого значения выборки.

18. Сгенерировать распределение Пуассона с мат. ожиданием 10. Объём выборки N=10000. Построить график плотности распределения и полигон. Вычислить среднее, дисперсию.

19. Сгенерировать однородное распределение для 10000x1 целых чисел в интервале

(-1,10). Построить гистограмму распределения и полигон. Вычислить среднее, дисперсию.

20. Сгенерировать однородное распределение для 100x100 целых чисел в интервале (1,100). Построить гистограмму распределения и полигон. Вычислить среднее, дисперсию, среднее квадратичное отклонение и медиану.

8.3 Идентификация эмпирических распределений. Вычисление интервальных оценок.

К числу теоретических распределений, которые наиболее часто используются при идентификации эмпирических распределений, относятся: распределение Гаусса или нормальное распределение, а также равномерное распределение. Как известно, существует достаточно большое количество статистических критериев для идентификации эмпирических распределений как параметрических, так и непараметрических. Один из наиболее часто используемых критериев – критерий Пирсона хи-квадрат. Поэтому рассмотрим примеры с его использованием.

Вычисление интервальных оценок для истинных значений параметров теоретического распределения по выборочным статистикам можно осуществить с помощью функций Scilab. Перечислим некоторые такие функции:

cdfchi()- функция интегрального распределения χ^2;

cdff() - функция интегрального F- распределения;

cdfgam()- функция интегрального гамма- распределения;

cdfnor()- функция интегрального нормального распределения;

cdft()- функция интегрального распределения Стьюдента.

Для примера рассмотрим синтаксис и применение функции **cdfnor()**.

[p,q] = cdfnor('PQ',x,mu,sigma)

[x] = cdfnor('X',mu,sigma,p,q)

[mu] = cdfnor('Mean',sigma,p,q,x)

[sigma] = cdfnor('Std',p,q,x,mu)

где - **mu** выборочное среднее; **sigma** - стандартное отклонение;

p = P(X<x) – вероятность того, что случайная величина находится в интервале $[-\infty, x]$;

q = 1 - p = P(X>x).

Первый аргумент в различных вызовах **cdfnor** является строковой переменной, которая указывает на тип ожидаемого результата:

'**PQ**' – вычисление значений вероятностей p и q при заданном значении нормированной переменной z;

'**X**' - вычисление значений нормированной переменной $z = \frac{x - M_x}{\sigma}$ с $M_z = 0$ и $\sigma_z = 1$ в зависимости от вероятности p;'**Mean**' – вычисление теоретического среднего значения распределения случайной величины X; '**Std**' - вычисление теоретического среднего квадратичного отклонения распределения.

Как видно из вышеперечисленных команд, можно вычислить любой из недостающих параметров по значениям всех остальных параметров.

Пример 1. Проверить гипотезу о нормальности распределения при уровне значимости α=0.05, используя критерий Пирсона для набора данных, которые получены с помощью функции Scilab **grand()**.

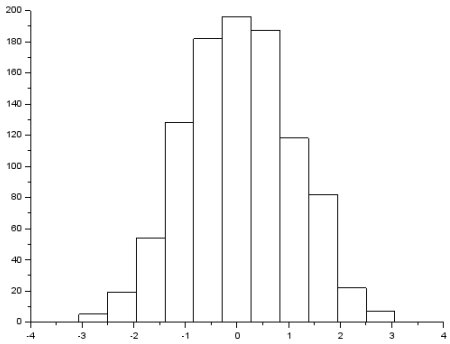


Рис.91 Гистограмма распределения

```
v=1000;//кол-во экс. точек
y=grand(1,v,'nor',0,1)//нормальное распределение со средним 0 и с.к.о равным 1
n=1+3.3*log10(v) //кол-во интервалов разбиения
n=ceil(n); //округлим до целого
ymax=max(y); //максимальное значение y
ymin=min(y);  //минимальное значение y
h=(ymax-ymin)/n; //шаг разбиения
[c1,in]=histplot(n,y, normalization=%f) // определение частот и построение гистограммы
c1 // частоты
yi=ymin+h*(0:n); // определение границ интервалов
xmean = mean(y) //среднее значение
sx = stdev(y) // среднее квадратическое отклонение
for i=1:n+1
[P(i),Q(i)]=cdfnor('PQ',yi(i),xmean,sx) //вычисление значений нормальной функции на границах интервалов
end
for j=1:n
   d(j)=v*(P(j+1)-P(j)); //вычисление теоретической частоты попадания в интервал
end
chi_p=(c1-d').^2/d'
chi2=sum(chi_p) //вычисление значения хи-квадрат
nu=n-1; //число степеней свободы
alpha=0.05; //уровень значимости
chi_t=cdfchi('X',nu,1-alpha,alpha); //табличное значение χ2
dif=chi2 - chi_t  //если dif<0, то , гипотеза о нормальности эмпирического распределения подтверждается.
```

Пример 2. Получить набор случайных чисел, распределённых по Пуассону со средним равным 10. Проверить гипотезу о соответствии экспериментального распределения распределению Пуассону. Использовать критерий Пирсона. Вычислить среднее и дисперсию.

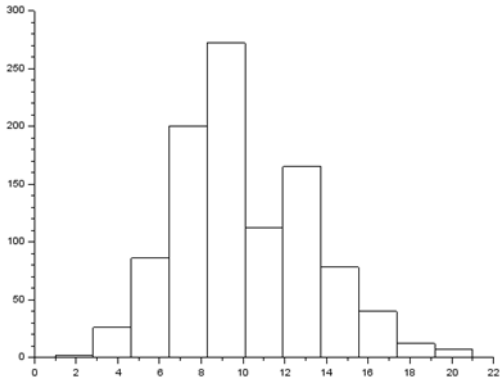


Рис.92 Гистограмма распределения

```
v=1000;//кол-во экс. точек
y=grand(1,v,'poi',10)// распределение Пуассона со средним  10
n=1+3.3*log10(v) //кол-во интервалов разбиения
n=ceil(n); //округлим до целого
ymax=max(y); //максимальное значение y
ymin=min(y);  //минимальное значение y
h=floor((ymax-ymin)/n); //шаг разбиения
[c1,in]=histplot(n,y, normalization=%f) // определение частот c1 и построение гистограммы
yi=ymin+h*(0:n); // определение границ интервалов
xmean = ceil(mean(y)) //среднее значение
sx = variance(y) // среднее дисперсия
for i=1:n+1
[P(i),Q(i)]=cdfpoi('PQ',yi(i),xmean) //вычисление значений функции Пуассона на границах
интервалов
end
for j=1:n
   d(j)=v*(P(j+1)-P(j)); //вычисление теоретической частоты попадания в интервал
end
chi_p=(c1-d').^2/d'
chi2=sum(chi_p) //вычисление значения хи-квадрат
nu=n-1; //число степеней свободы
alpha=0.05; //уровень значимости
chi_t=cdfchi('X',nu,1-alpha,alpha); //табличное значение χ2
dif=chi2 - chi_t //если dif<0, то , гипотеза об эмпирическом распределении подтверждается.
->dif
 dif  =
  - 17.516798
-->xmean
 xmean  =
   10.
-->sx
 sx  =
   9.6661772
```

Пример 3. Выборка из 100 резисторов используется для определения сопротивления партии резисторов. Среднее значение сопротивления для этой выборки оказалось равной 580 Ом. Если техническими условиями задано, что стандартное отклонение сопротивления равно 10 Ом, то определить доверительный интервал для среднего значения сопротивления с доверительной вероятностью 0,95.

Данные известные из условия: $n = 100$, $\bar{x} = 580$, $\sigma = 10$, $\alpha = 0.05$. Чтобы вычислить доверительный интервал, необходимо найти $z_{\alpha}/2$ из условия $P\ (Z > z_{\alpha}/2) = \alpha/2$, или $P\ (Z > z_{\alpha}/2) = 1 - \alpha/2$.

```
n=100;xm=580;sigma=10;al=0.05;// исходные данные
z_al2=cdfnor('X',0,1,1-al/2,al/2) // вычисление квантильного множителя
L=xm-z_al2*sigma/sqrt(n) // левая граница доверительного //интервала
R=xm+z_al2*sigma/sqrt(n) // правая граница доверительного интервала
Res=[L,R]// доверительный интервал
-->Res
 Res  =
   578.04004    581.95996
```

Пример 4. При проведении 20 измерений толщины бруска было получено значение средней толщины 10.7 мм со средним квадратичным отклонением 0.1мм. Определить доверительный интервал для истинной толщины бруска с 95% вероятностью.

Из условия задачи имеем следующие данные: $n = 20$, $\bar{x} = 10.7$, $S_x = 0.1$, $\alpha = 0.05$. Чтобы вычислить доверительный интервал, необходимо найти значение $t_{n-1,\alpha}/2$ из условия $P(T > t_\alpha/2) = \alpha/2$.

```
n=20;xm=10.7;s=0.1;alpha=0.05;
t_alpha_2 = cdft('T',n-1,1-alpha/2,alpha/2)
R=xm+t_alpha_2*s/sqrt(n)
L=xm-t_alpha_2*s/sqrt(n)
Interval=[L R]//доверительный интервал
->Interval
 Interval  =
   10.653199    10.746801
```

Пример 5. Сгенерировать набор из 1000 случайных чисел, имеющих распределение Вейбулла с параметрами α=3, β=4. Построить гистограмму сгенерированных данных и график плотности распределения.

В Scilab нет функции, с помощью которой можно было бы генерировать набор случайных чисел с распределением Вейбулла. Посмотрим, как это можно сделать, используя интегральную функцию распределения вероятностей (этот способ называется методом обратных функций). Воспользуемся функцией **rand()** для генерации однородно распределённых чисел p в диапазоне от 0 до 1. Пусть p является вероятностью того, что $p = F_X(x) = P(X<x)$. Интегральная функция этого распределения выглядит следующим образом:

$$F(x) = 1 - \exp(-\alpha \cdot x^{\beta}), \text{ для } x > 0, \alpha > 0, \beta > 0$$

Решим его относительно x.

$$x = \left[-\frac{\ln(1-p)}{\propto}\right]^{1/\beta}$$

функция плотности распределения Вейбулла имеет вид:

$$f(x) = \alpha\beta x^{\beta-1} e^{-\alpha x^{\beta}}$$

```
p=rand(1,1000); //равномерное распределение
a=3; b=4; //параметры распределения Вейбулла
x = (-log(1-p)/a)^(1/b); //генерация данных из расп. Вейбулла
histplot(11,x,normalization=%t)// гистограмма
x=gsort(x,'g','i');//сортировка чисел
f=a*b*x.^(b-1).*exp(-a*x.^b);//плотность расп. Вейбулла
plot(x,f)// график
```

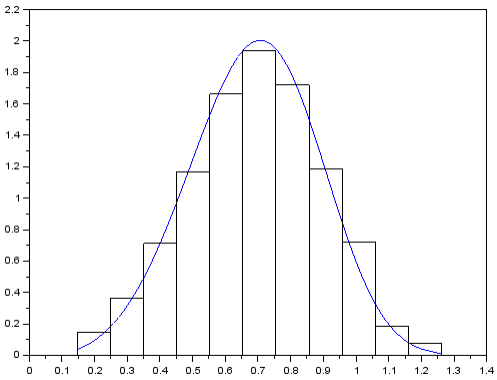


Рис.93 Гистограмма и график распределения Вейбулла

Пример 6. Сгенерировать набор из 1000 случайных чисел, имеющих распределение Стьюдента с 10 степенями свободы. Построить гистограмму сгенерированных данных и график плотности распределения.

В опциях функции **grand()** нет возможности генерировать данные с распределением Стьюдента. Зато есть интегральная функция распределения **cdft( )**. Воспользуемся опять методом обратных функций, имея в виду, что явно получить выражение для x затруднительно, и поэтому используем возможности функции **cdft()**.

```
nu=10;//число степеней свободы
pp = rand(1,1000); //раномерное распр.
x = []; //
for j =1:1000 //величина x
val=cdft('T',nu,pp(j),1-pp(j));
 x = [x val];
end;
histplot(11,x,style=10)//гистограмма
deff('[f]=fT(t,nu)',..
'f=gamma((nu+1)./2).*(1+t.^2./nu).^..
(-(nu+1)/2)/(sqrt(%pi*nu)*gamma(nu/2))')//аналит. Стьюдента
x=gsort(x,'g','i');//сортировка x
tt=x;ff=fT(tt,nu);
plot(tt,ff,'-k') //график
```

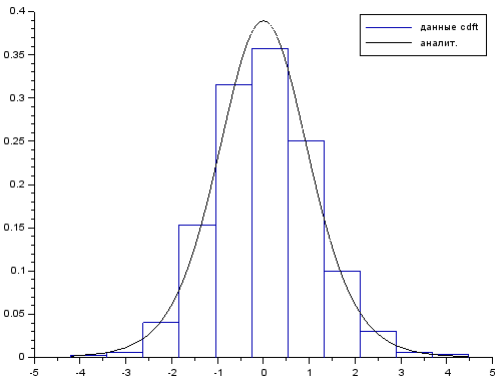


Рис.94 Гистограмма и график распределения Стьюдента

Пример 7. Найти значение x для нормального распределения со средним Mean = 0, Std = 1 при значениях: p = 0.2, q = 0.8.

```
-->Mean = 0;
-->Std = 1;
-->p = 0.2;
-->q = 0.8;
-->x = cdfnor('X',Mean,Std,p,q)
x  =
 - 0.8416212
```

Пример 8. Вычислить значение вероятности для нормального распределения при x = 0.7 и параметрах m = 2 и sigma = 4.

```
x=0.5;
 std =4
m=2;
[P,Q]=cdfnor('PQ',x,m,std)
->Q
 Q  =
    0.6461698
-->P
 P  =
    0.3538302
```

Пример 9. Получить выборку объёмом n=100 случайных чисел для нормального распределения с мат. ожиданием m=2 и среднеквадратичным отклонением σ=3. Вычислить оценку среднего, среднеквадратичного отклонения, дисперсии. Получить с 95% вероятностью доверительный интервал для оценки дисперсии.

Несмещённая оценка дисперсии σ^2

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Тогда величина

$$(n-1)\frac{s^2}{\sigma^2} = \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

описывается распределением хи-квадрат с *n-1* степенями свободы.

Тогда для σ^2 двусторонний доверительный интервал с доверительной вероятностью (1-α)·100% находится из выражения:

$$P\left(\chi^2_{n-1,1-\alpha/2} \le (n-1)\frac{s^2}{\sigma^2} \le \chi^2_{n-1,\alpha/2}\right) = 1-\alpha$$

поэтому доверительный интервал для σ^2 :

$$\left(\frac{(n-1)s^2}{\chi^2_{n-1,\alpha/2}}, \frac{(n-1)s^2}{\chi^2_{n-1,1-\alpha/2}}\right)$$

```
n=100;
x=grand(1,100,'nor',2,3);
s=stdev(x);al=0.05;// al уровень значимости
Chi_al_2 = cdfchi('X',n-1,1-al/2,al/2);
Chi_1_al_2=cdfchi('X',n-1,al/2,1-al/2);
CL=(n-1)*s^2/Chi_al_2//нижний предел интервала
CU=(n-1)*s^2/Chi_1_al_2//верхний предел
-->[CL CU]
 ans  =
   8.6045813    15.06272
-->mean(x)
 ans  =
   2.0031304
-->stdev(x)
 ans  =
   3.3409269
-->variance(x)
 ans  =
   11.161792
```

Пример 10. Используя метод обратной функции, получить формулу для формирования выборки случайных чисел с равномерным распределением в заданном интервале. Сформируйте 100 случайных чисел с равномерным распределением из интервала [-2; 5] с использованием функции **rand().**

Равномерно распределенная случайная величина в интервале [a; b] имеет функцию плотности

$$f(x) = \frac{1}{b-a},$$

интегральная функция будет:

$$F(x) = r = \int_a^x \frac{dt}{b-a} = \frac{x}{b-a} - \frac{a}{b-a}; x = a + (b-a)r,$$

где r-равномерно распределённое число в интервале от 0 до 1.

```
a = -2; b = 5; N = 100;
t = a + (b - a)*rand(1,N)
```

Задачи.

1. Проверить гипотезу о нормальности распределения при уровне значимости α=0.01, используя критерий Пирсона для 100 случайных чисел, которые получены с помощью функции Scilab **grand**(1,v,'nor',0,1).

2. Проверить гипотезу о нормальности распределения при уровне значимости α=0.05, используя критерий Пирсона для 20 случайных чисел, которые получены с помощью функции Scilab **grand**(1,v,'nor',0,1).Вычислить оценку среднего и дисперсии для полученной выборки. Построить гистограмму.

3. Получить набор из 100 случайных чисел, распределённых по Пуассону со средним равным 5. Проверить гипотезу о соответствии экспериментального распределения распределению Пуассону при уровне значимости α=0.05. Использовать критерий Пирсона. Вычислить среднее и дисперсию.

4. Получить набор случайных чисел, распределённых по Пуассону со средним равным 2. Проверить гипотезу о соответствии экспериментального распределения распределению Пуассону при уровне значимости α=0.1. Использовать критерий Пирсона. Вычислить среднее и дисперсию.

5. Выборка из 10 валов используется для определения диаметра партии валов. Среднее значение диаметра для этой выборки оказалось равной 10.5 мм. Если техническими условиями задано, что стандартное отклонение диаметра равно 0.10 мм, то определить

доверительный интервал для среднего значения диаметра с доверительной вероятностью 0.95.

6. Выборка из 100 резисторов используется для определения сопротивления партии резисторов. Среднее значение сопротивления для этой выборки оказалось равной 300 Ом. Если техническими условиями задано, что стандартное отклонение сопротивления равно 5 Ом, то определить доверительный интервал для среднего значения сопротивления с доверительной вероятностью 0,99.

7. При проведении 20 измерений толщины бруска было получено значение средней толщины 50.7 мм со средним квадратичным отклонением 0.1мм. Определить доверительный интервал для истинной толщины бруска с 95% вероятностью.

8. Найти доверительный интервал для оценки с вероятностью P = 0,99 математического ожидания M_x нормально распределенной величины X, если даны - среднее квадратичное отклонение теоретического распределения $\sigma = 3$, выборочное среднее $\bar{x} = 20$, объем выборки N = 100.

9. Сгенерировать набор из 100 случайных чисел, имеющих распределение Вейбулла с параметрами α=2, β=5, используя метод обратных функций. Построить гистограмму сгенерированных данных и график плотности распределения.

10. Сгенерировать набор из 50 случайных чисел, имеющих распределение *f(x)=x* на интервале (0;1), используя метод обратных функций. Построить гистограмму сгенерированных данных и график плотности распределения.

11. Сгенерировать набор из 100 случайных чисел, имеющих распределение Стьюдента с 5 степенями свободы. Построить гистограмму сгенерированных данных и график плотности распределения.

12. Сгенерировать набор из 50 случайных чисел, имеющих распределение Стьюдента с 20 степенями свободы. Построить гистограмму сгенерированных данных и график плотности распределения.

13. Найти значение x для нормального распределения с математическим ожиданием Mean = 0, среднее квадратичное отклонение $\sigma = 1$ при значениях: p = 0.01, q = 0.99.

14. Найти значение x для нормального распределения со математическим ожиданием Mean = 5, среднее квадратичное отклонение $\sigma = 2$ при значениях: p = 0.5, q = 0.5.

15. Вычислить значение вероятности для нормального распределения при x = 0.3 и параметрах: математическим ожиданием m = 1, среднее квадратичное отклонение $\sigma = 1$.

16. Вычислить значение вероятности для нормального распределения при x = 0, и параметрах: математическим ожиданием m = 0, среднее квадратичное отклонение $\sigma = 1$.

17. Получить выборку объёмом n=50 случайных чисел для нормального распределения с мат. ожиданием m=4 и среднеквадратичным отклонением σ=5. Вычислить оценку среднего, среднеквадратичного отклонения, дисперсии. Получить с 95% вероятностью доверительный интервал для оценки дисперсии.

18. Получить выборку объёмом n=10 случайных чисел для нормального распределения с мат. ожиданием m=0 и среднеквадратичным отклонением σ=1. Вычислить оценку среднего, среднеквадратичного отклонения, дисперсии. Получить с 95% вероятностью доверительный интервал для оценки дисперсии. Построить гистограмму для полученных данных и график теоретической функции плотности вероятности.

19. По данным N = 20 независимых равноточных измерений некоторой физической величины найдены среднее арифметическое результатов измерений $\bar{x}$=10 и исправленное среднее квадратичное отклонение S = 2. Оценить истинное значение измеряемой величины при помощи доверительного интервала с надежностью γ = 0.95.

20. Используя метод обратной функции, получить формулу для формирования выборки случайных чисел с плотностью распределения, заданной функцией $f(x)=x^2+1$ в интервале [-5; 5] . Сформируйте 100 случайных чисел с этим распределением.

Спектральный анализ

Многие физические величины, которые можно представить в виде суммы синусоидальных компонент, удобно описывать с помощью рядов Фурье. Чтобы разложить функцию в ряд Фурье исходят из предположения, что её можно выразить в виде сходящихся тригонометрических рядов на определённом интервале. Для описания непериодических сигналов с конечной энергией используют преобразования Фурье. Математически сигнал можно представить с помощью некоторой функции f(t), где t - чаще всего время, но может иметь и другой смысл, например - пространственной координаты.

В Scilab имеется большое количество функций, предназначенных для обработки сигналов. Рассмотрим некоторые из них, а именно отображение сигнала, оценка спектра сигнала.

Пример 1. Разложить в ряд Фурье по синусам функцию $f(x) = x - x^4$ заданную на отрезке [-2;3] с периодом равным 2. В разложении ограничиться 30 членами ряда.

```
N=30;L=2;
x=[-2:0.05:3];
deff('[y]=f(x)','y=x-x.^4')
y=f(x);
xset('window',0)
plot(x,y)
deff('[w]=s(x)','w=f(x).*sin(n*%pi*x/L)')
deff('[bn]=b(n)','bn=(2/L)*intg(0,L,s)')
c = [];
for j=1:N
c =[c b(j-1)];
end;
m = length(c); n = length(x);
y1 = [];
for i = 1:n
yy = 0;
for j = 1:m
yy = yy + c(j)*sin((j-1)*%pi*x(i)/L);
end;
y1 = [y1 yy];
end
xset('window',1)
plot2d([x' x'],[y' y1'],[1,-5],'011',' ',[-2 -35 3 20])
```

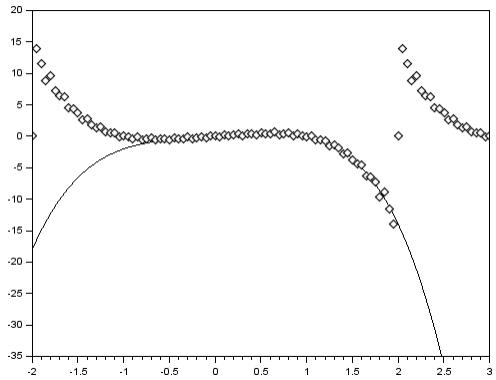


Рис.95 График функции $f(x)=x-x^4$ и график функции из 30 членов ряда Фурье

Пример 2. Разложить в ряд Фурье по косинусам функцию

f(x) = (1-x+x^2)(x^2-1)(x+2), заданную на отрезке [-0.5;2] с периодом равным 1. В разложении ограничиться 40 членами ряда.

```
L=1;n=40;x = [-0.50:0.01:2];
deff('[y]=f(x)','y=(1-x+x.^2).*(x.^2-1).*(x+2)')
deff('[y]=g(x)','y=f(x)*cos(n*%pi*x/L)')
deff('[an]=a(n)',..
['if n==0 then';'an=intg(0,L,f)/L';'else';'an=(2/L)*intg(0,L,g)/L';'end'])
c=[];for j=1:n, c=[c a(j-1)]; end;
m = length(c); n = length(x);
y = [];
for i = 1:n
yy = c(1);
for j = 2:m
yy = yy + c(j)*cos((j-1)*%pi*x(i)/L);
end;
y = [y yy];
end;
y1 = f(x);
plot2d([x' x'],[y' y1'],[1,-1],'011',' ',[-0.5 -2.5 2 0.5])
```

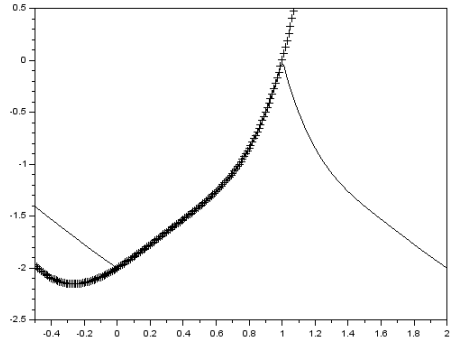


Рис.96 График функции
f(x) = (1-x+x^2)(x^2-1)(x+2)
и график функции из 40 членов ряда Фурье

Пример 3. Разложить в ряд Фурье функцию *f(x)= (1+x^2+2x^3)*, заданную на отрезке [-1;1] с периодом равным 2. В разложении ограничиться 50 членами ряда.

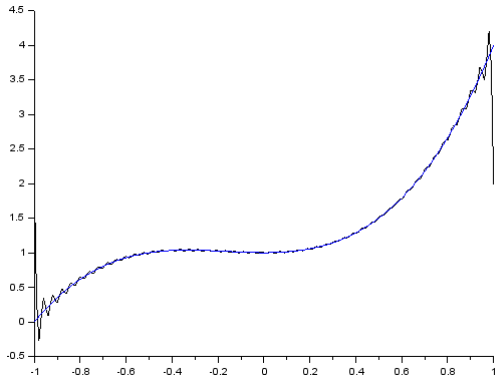


Рис.97 График функции *f(x) = (1+x^2+2x^3)* и график функции из 50 членов ряда Фурье

```
deff('[y]=f(x)','y=(1+x^2+2*x^3)')
L = 2; x=[-1:0.01:1]; y1 = f(x);
c0=-L/2;n=50;tol=10^-5;
deff('[gg1]=g1(x)','gg1=f(x)*cos(2*nn*%pi*x/L)');
deff('[gg2]=g2(x)','gg2=f(x)*sin(2*nn*%pi*x/L)');
deff('[aaa]=a1(nn)','aaa=(2/L)*intg(c0,c0+L,g1,tol)');
deff('[bbb]=b1(nn)','bbb=(2/L)*intg(c0,c0+L,g2,tol)');
a0 = (2/L)*intg(c0,c0+L,f,tol);
nmax = max(n);a = []; b = [];
for j = 1:nmax
a = [a a1(j)]; b = [b b1(j)];
end;
y=[];
nn = length(a); mm = length(x);yy = zeros(1,mm);
for j = 1:mm
yy = a0/2;
for k = 1:nn
yy = yy + a(k)*cos(2*k*%pi*x/L) + b(k)*sin(2*k*%pi*x/L);
end;
y=yy;
end;
plot2d([x' x'],[y' y1'])
```

Пример 4. Дана функция *h(x) = 3, -1/2 < x < 1/2* и *h(x)* = 0 вне интервала. Произвести фитирование кусочно-непрерывнной функции (прямоугольного импульса) рядом Фурье. В разложении ограничиться 10 членами ряда.

```
deff('[y]=f(x)',['if x>-1/2 & x<1/2 then';'y=3';'else';'y=0';'end'])
L = 2; x=[-1:0.01:1];
y1=[]; for j =1:length(x), y1=[y1 f(x(j))]; end;
c0=-L/2;n=10;tol=10^-5;
deff('[gg1]=g1(x)','gg1=f(x)*cos(2*nn*%pi*x/L)');
deff('[gg2]=g2(x)','gg2=f(x)*sin(2*nn*%pi*x/L)');
deff('[aaa]=a1(nn)','aaa=(2/L)*intg(c0,c0+L,g1,tol)');
deff('[bbb]=b1(nn)','bbb=(2/L)*intg(c0,c0+L,g2,tol)');
a0 = (2/L)*intg(c0,c0+L,f,tol);
nmax = max(n);a = []; b = [];
for j = 1:nmax
a = [a a1(j)]; b = [b b1(j)];
end;
y=[];
nn = length(a); mm = length(x);yy = zeros(1,mm);
for j = 1:mm
yy = a0/2;
for k = 1:nn
yy = yy + a(k)*cos(2*k*%pi*x/L) + b(k)*sin(2*k*%pi*x/L);
end;
y=yy;
end;
plot2d([x' x'],[y' y1'])
```

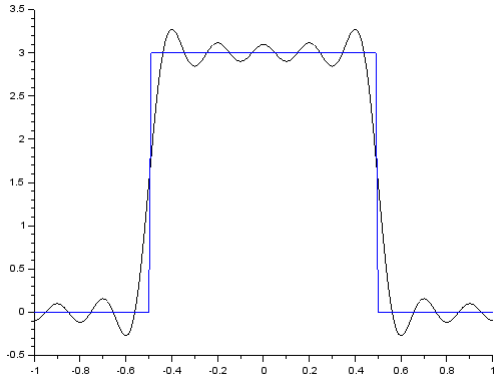


Рис.98 График функции и график функции из 10 членов ряда Фурье

Пример 5. Дана функция h(x) = 3, -1/2 < x < 1/2 и h(x) = 0 вне интервала. Произвести фитирование кусочно-непрерывнной функции (прямоугольного импульса) рядом Фурье. В разложении ограничиться 10 членами ряда. Период повторения равен двум, интервал, на котором необходимо провести разложение, [-4;4].

Для решения этой задачи воспользуемся текстом сценария, приведённого в примере 4. Изменим только интервал. В результате получим следующий график.

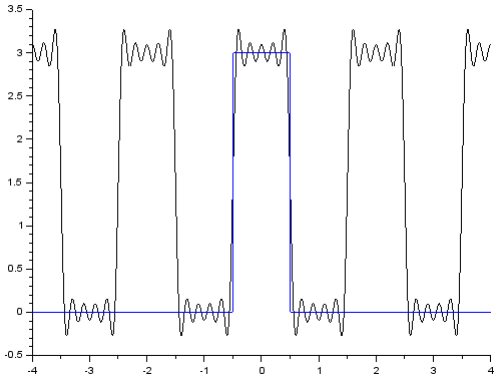


Рис.99 График функции и график функции, состоящей из 10 членов ряда Фурье на интервале [-4;4]

Пример 6. Найти Фурье преобразование квазипериодического сигнала, $y(t) = \sin(100\pi t) + \sin(200\pi t)$, имеющего две гармоники 50 и 100 Гц, с наложенным гауссовым шумовым сигналом. Частоту дискретизации выбрать равной 1000 Гц и длительность сигнала 1с. Построить амплитудный спектр сигнала.

Частота дискретизации (или частота семплирования, англ. sample rate) это - частота взятия отсчетов непрерывного во времени сигнала при его дискретизации. Измеряется в герцах. Длительность сигнала t связана с частотой дискретизации Fs, следующим соотношением: t=N/Fs; где N-общее количество отсчётов аргумента (пропорционально длительности нашего сигнала). Частота дискретизации (оцифровки) сигнала должна быть как минимум в два раза больше максимальной частоты входного сигнала (теорема Котельникова-Найквиста). Если человеческая речь, например, занимает полосу частот до 3-4 кГц, то для ее оцифровки потребуется частота 8 кГц и т.п.

```
sample_rate=1000;
t = 0:1/sample_rate:1;
N=size(t,'*'); //число отсчётов сигнала
s=sin(2*%pi*50*t)+sin(2*%pi*100*t)+grand(1,N,'nor',0,1);
y=fft(s);
f=sample_rate*(0:(N/2))/N; //частота
n=size(f,'*')
xf=abs(y(1:n))//
clf()
plot(f,xf)
```

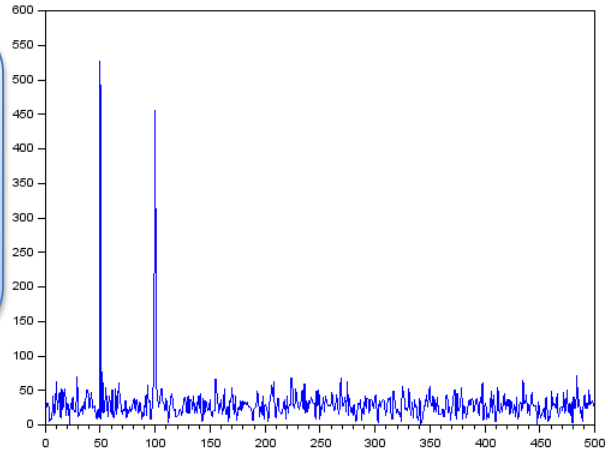


Рис.100 График амплитудного спектра

Пример 7. Найти энергетический спектр сигнала по дискретной выборке на основе преобразования Фурье. Дан квазипериодический сигнал, представляющий собой сумму гармоник с двумя частотами: $y(t) = 1.5\sin(2t) + 0.5\cos(2\pi^{-1}t)$. Дан интервал дискретности $T_0 = 0.1$ и число точек отсчетов $N = 2^{10} = 1024$.

```
dt=0.1;
N=2^10;
t=0:dt:(N-1)*dt;
y=1.5*sin(2*t)+0.5*cos(2/%pi*t);
subplot(2,1,1)
plot2d(t,y)
Y=fft(y,-1);
PY=abs(Y).^2/N;
wn2=2*%pi/dt;
dw=wn2/N;
w=0:dw:5;
lw=length(w);
subplot(2,1,2)
plot2d(w,PY(1:lw))
```

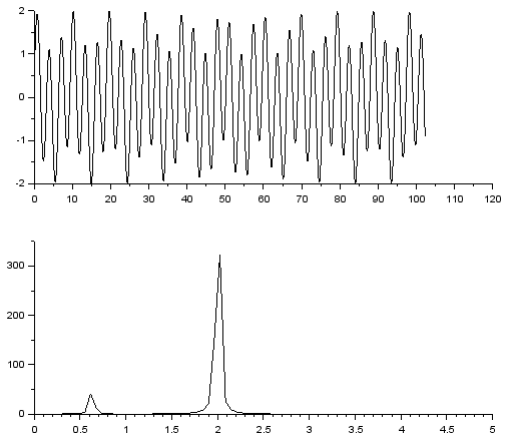


Рис.101 График сигнала и энергетического спектра в диапазоне от 0 до 5 с.

Пример 8. Задан сигнал с помощью двадцати комплексных коэффициентов преобразования Фурье, которые определены выражением $A = 2\cos(2\pi x)+5i\cdot\sin(-x)$, где x изменяется от 1до 20. Получить амплитуду сигнала с помощью обратного преобразования Фурье.

```
x=1:20;
Xf2 = 2*cos(2*%pi*x)+%i*5*sin(-x);
Xf2A = abs(Xf2);
subplot(2,1,1)
xset('mark',-9,1); plot2d([x]',Xf2A',-9)
plot2d3('onn',[x]',Xf2A')
xs2=fft(Xf2,1);
xs2A = abs(xs2);nn=[1:1:20];
subplot(2,1,2)
plot2d(nn',xs2A,-9,'011',' ',[0 -3 21 13])
plot2d(nn',xs2A,1,'011',' ',[0 -3 21 13])
```

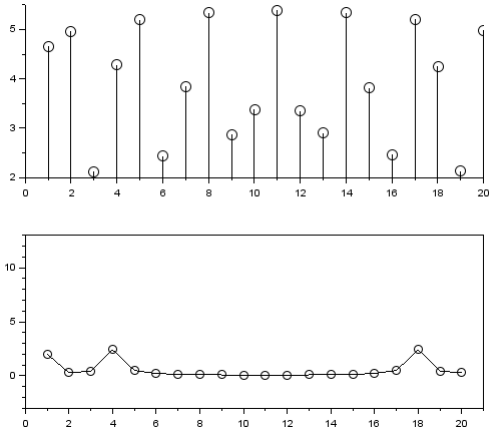


Рис.102 График коэффициентов и амплитуды сигнала

Пример 9. Получить дискретное преобразование Фурье для двухмерного сигнала *z=sin(2x)cos(3y),* в области *x,y* от -10 до 10. Сигнал включает в себя случайную компоненту - белый шум (моделировать функцией rand()).

```
deff('[w]=f(x,y)','w=sin(2*x)*cos(3*y) +
rand()')
x=[-10:0.5:10];y=x;z=feval(x,y,f);
subplot(2,1,1)
plot3d(x,y,z)
Z = fft(z,-1);
Za=abs(Z);
subplot(2,1,2)
hist3d(Za)
```

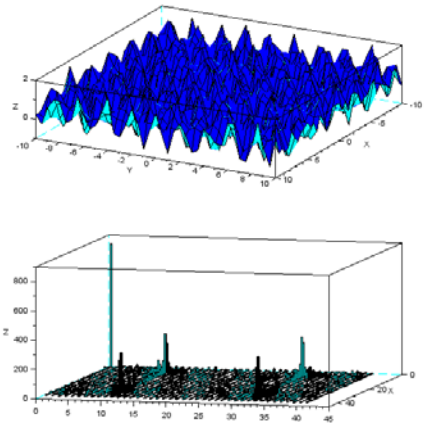


Рис.103 График функции и амплитуда коэффициентов Фурье

Пример 10. Дан сигнал y=3sin(t)+10sin(3t)+0.5sin(15t)+13rand(t) в диапазоне t от 0 до 100. Построить график спектральной плотности, разместив нулевую частоту в середине спектра. Использовать функцию **fftshift().**

```
t=0:0.1:100;
x=3*sin(t)+10*sin(3*t)+0.5*sin(15*t)+13*rand(t);
y=fft(x,-1);
subplot(2,1,1);plot2d(abs(y))
subplot(2,1,2);plot2d(fftshift(abs(y)))
```

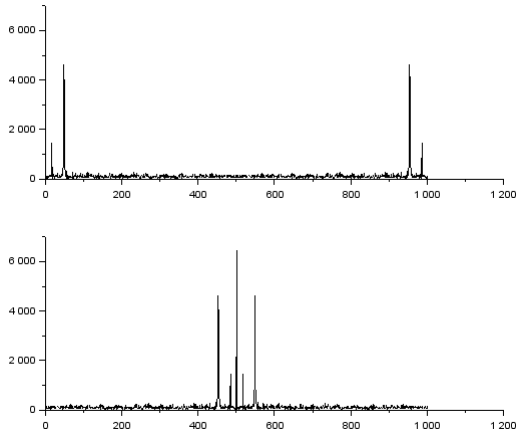


Рис.104 Графики амплитуды коэффициентов Фурье

Задачи.

1. Разложить в ряд Фурье по синусам функцию $f(x) = 2x - 3x^2$ заданную на отрезке [-2;3] с периодом равным 2. В разложении ограничиться 20 членами ряда.

2. Разложить в ряд Фурье по синусам функцию $f(x) = x$ заданную на отрезке [-1;1] с периодом равным 2. В разложении ограничиться 30 членами ряда.

3. Разложить в ряд Фурье по косинусам функцию $f(x) = (1+x^2)(x^2-1)$, заданную на отрезке [-2;2] с периодом равным 1. В разложении ограничиться 40 членами ряда.

4. Разложить в ряд Фурье по косинусам функцию $f(x) = x^2(x-1)(x+1)$, заданную на отрезке [-0.5;2] с периодом равным 1. В разложении ограничиться 20 членами ряда.

5. Разложить в ряд Фурье функцию $f(x) = (x^2+2x^3+x)$, заданную на отрезке [-1;1] с периодом равным 3. В разложении ограничиться 50 членами ряда.

6. Разложить в ряд Фурье функцию $f(x) = (1+x^2+e^{-x})$, заданную на отрезке [-1;1] с периодом равным 3. В разложении ограничиться 50 членами ряда.

7. Дана функция $h(x) = 2$, $-1/3 < x < 1/4$ и $h(x) = 0$ вне интервала. Произвести фитирование кусочно-непрерывнной функции (прямоугольного импульса) рядом Фурье. В разложении ограничиться 20 членами ряда.

8. Дана функция $h(x) = 3$, $-1/2 < x \leq 0$ и $h(x) = -3$, $0 \leq x < 1/2$ и $h(x) = 0$ вне интервала. Произвести фитирование кусочно-непрерывнной функции рядом Фурье. В разложении ограничиться 10 членами ряда.

9. Дана функция $h(x) = 2$, $-1/2 < x < 1/2$ и $h(x) = 0$ вне интервала. Произвести фитирование кусочно-непрерывнной функции (прямоугольного импульса) рядом Фурье. В разложении ограничиться 10 членами ряда. Период повторения равен двум, интервал, на котором необходимо провести разложение, [-5;5].

10. Произвести фитирование кусочно-непрерывнной функции из задачи 8 рядом Фурье. В разложении ограничиться 10 членами ряда. Период повторения равен двум, интервал, на котором необходимо провести разложение, [-5;5].

11. Найти Фурье преобразование квазипериодического сигнала, $y(t) = \sin(10\pi t) + \sin(20\pi t)$, имеющего две гармоники 5 и 10 Гц, с наложенным белым шумовым сигналом. Частоту дискретизации выбрать равной 1000 Гц и длительность сигнала 1с. Построить амплитудный спектр сигнала.

12. Найти Фурье преобразование квазипериодического сигнала $y(t) = \sin(10\pi t) + \cos(20\pi t)$, с наложенным гауссовым шумовым сигналом. Частоту дискретизации выбрать равной 500 Гц и длительность сигнала 1с. Построить амплитудный спектр сигнала.

13. Найти энергетический спектр сигнала по дискретной выборке на основе преобразования Фурье. Дан квазипериодический сигнал, представляющий собой сумму гармоник с двумя частотами: $y(t) = 3\sin(2t) + 1.5\sin(2\pi^{-1}t+1)$. Дан интервал дискретности $T_0 = 0.1$ и число точек отсчетов $N = 2^{10} = 1024$.

14. Найти энергетический спектр сигнала по дискретной выборке на основе преобразования Фурье. Дан квазипериодический сигнал, представляющий собой сумму

гармоник с тремя частотами: $y(t) = sin(2.5t) + cos(2\pi^{-1}t)+sin(3t)$. Дан интервал дискретности $T_0 = 0.1$ и число точек отсчетов $N = 2^{12}$.

15. Задан сигнал с помощью двадцати комплексных коэффициентов преобразования Фурье, которые определены выражением $A = cos(2\pi x)+i\cdot sin(-x+1)$, где x изменяется от 1до 20. Получить амплитуду сигнала с помощью обратного преобразования Фурье.

16. Задан сигнал с помощью тридцати комплексных коэффициентов преобразования Фурье, которые определены выражением $A = sin(\pi x)+5i\cdot cos(2x)$, где x изменяется от 1до 30. Получить амплитуду сигнала с помощью обратного преобразования Фурье.

17.Получить дискретное преобразование Фурье для двухмерного сигнала z=sin(x)cos(y), в области x,y от -10 до 10. Сигнал включает в себя случайную компоненту - белый шум (моделировать функцией **rand()**).

18. Получить дискретное преобразование Фурье для двухмерного сигнала $z=\cos(x^2)\sin(2y)$, в области x,y от -10 до 10. Сигнал включает в себя случайную компоненту - гауссов шум.

19. Дан сигнал y=sin(2t)+10sin(3t)+0.5sin(10t)+15rand(t) в диапазоне t от 0 до 100. Построить график спектральной плотности, разместив нулевую частоту в середине спектра. Использовать функцию **fftshift().**

20. Пример 10. Дан сигнал y=cos(t)+10sin(3t)+10rand(t) в диапазоне t от 0 до 50. Построить график спектральной плотности, поместив нулевую частоту в середине спектра. Использовать функцию **fftshift()**.

Список литературы

[1] С.В. Ерин, Ю.Л. Николаев Автоматизация инженерных расчётов с использованием пакета Scilab: Москва: Издательство «Русайнс» 2015

[2] Е. Р. Алексеев, О. В. Чеснокова, Е. А. Рудченко Scilab Решение инженерных и математических задач Москва ALT Linux; БИНОМ. Лаборатория знаний 2008

[3] Е.Р. Алексеев, О.В. Чеснокова Scilab теория и практика Донецк 2007

[4] Андриевский А.Б., Андриевский Б.Р., Капитонов А.А., Фрадков А.Л. Решение инженерных задач в SCILAB - Санкт-Петербург: НИУ ИТМО, 2013. - 97 с. - экз.

[5] Р.С. Гутер, Б.В. Овчинников Элементы численного анализа и математическая обработка результатов опыта М., Мир, 1975

[6] С.В. Брановицкая, Р. Б.Медведев, Ю. Я. Фиалков Вычислительная математика в химии и химической технологии Киев: «Вища школа», 1986

[7] В.Б. Исаков Элементы численных методов: Учебное пособие. М.: Академия, 2003

[8] Т. Шуп Решение инженерных задач на ЭВМ Москва «Мир» 1982

[9] C. Gomez , C. Bunks, J. Ph. Chancelier, F. Delebecque, M. Goursat, R. Nikoukhah, S. Steer Engineering and Scientific Computing with Scilab Springer-Science+Business Media, LLC 1999

[10] Вержбицкий В.М. Основы численных методов. Учебник для вузов М.: Высшая школа , 2002 -840 стр.

Scilab – примеры и задачи

Практическое пособие

В авторской редакции