Лабораторная работа № 1

Основы математического моделирования. Работа в среде моделирования SciLab

Аппроксимация и интерполяция табличных функций. Численное решение нелинейных уравнений

Цель работы: 1) ознакомиться со средой численного моделирования SciLab; 2) научиться исследовать простые математические модели. 3) изучить способы аппроксимации и интерполяции экспериментальных данных, представленных таблично; 4) научиться численно решать уравнения.

Теоретические сведения

Запуск среды моделирования SciLab

Для запуска среды моделирования SciLab необходимо щёлкнуть по соответствующему пункту меню оболочки или выполнить в терминале команду:

```
scilab
```

Основные компоненты среды моделирования SciLab

После запуска среды моделирования SciLab на экране появляется командное окно (рис.1):

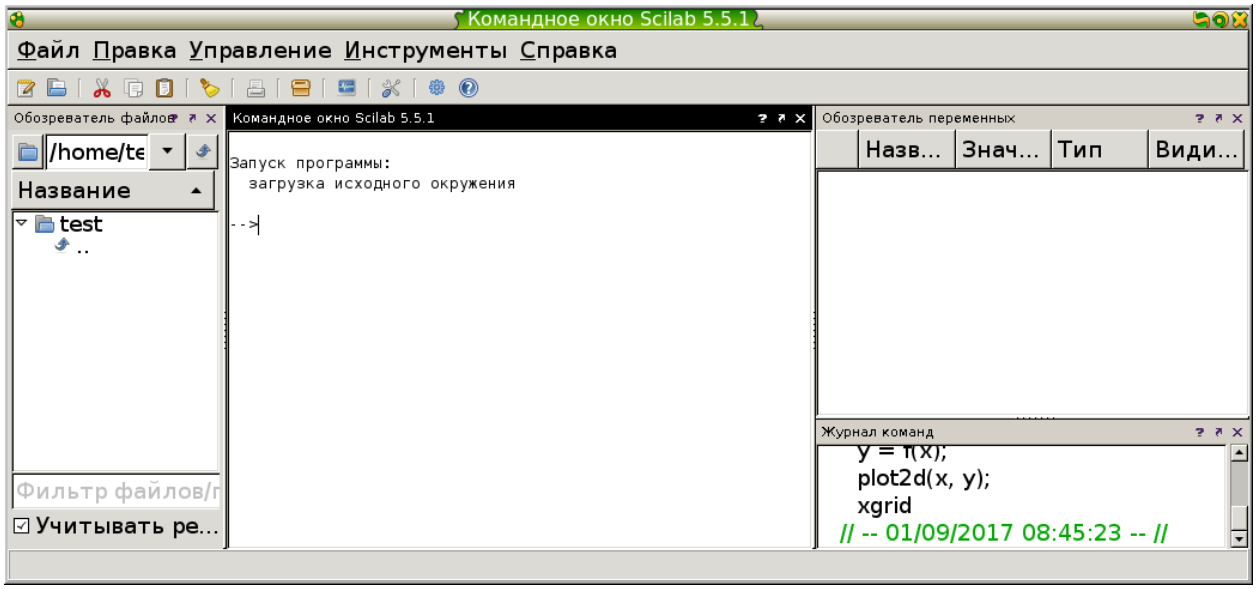

Рис.1. Командное окно SciLab

Его основной частью является рабочая область — консоль SciLab, где вводятся выполняемые команды и отображаются результаты уже выполненных.

Выполнить команду можно, если ввести её в командную строку после приглашения

```
-->
```

и нажать клавишу ENTER. В рабочей области появится результат.

Последовательность команд можно сохранить в текстовом файле с расширением .sce, который называется скриптом или сценарием. Такой файл может быть открыт в редакторе SciNotes через главное меню Файл Scilab.

Скрипт может быть выполнен функцией exec либо через меню редактора SciNotes. В первом случае, если путь не указан (exec("testscript.sce");), скрипт при вызове должен быть в текущем каталоге (выбирается в обозревателе файлов, по умолчанию слева от консоли).

Справка среды моделирования SciLab

Справка доступна пункт «Справка» через главное меню.

Сведения об использовании какой-либо функции можно получить, выполнив команду **help**. Например, команда

```
help rand
```

откроет описание простейшего генератора случайных чисел (в частности, внизу, среди ссылок на смежные и аналогичные функции можно найти ссылку на более качественный генератор **grand**).

Элементарные математические выражения. Использование встроенных переменных и функций

Для выполнения простейших арифметических операций в Scilab применяют следующие операторы: + сложение, — вычитание, * умножение, / деление слева направо, \ деление справа налево, ^ возведение в степень.

Вычислить значение арифметического выражения можно, если ввести его в командную строку и нажать клавишу ENTER.

Если символ точки с запятой «;» указан в конце выражения, то результат вычислений не выводится, а активизируется следующая командная строка.

В рабочей области Scilab можно определять переменные, а затем использовать их в выражениях.

Любая переменная до использования в формулах и выражениях должна быть определена.

Для определения переменной необходимо набрать имя переменной, символ «=» и значение переменной. Здесь знак равенства — это оператор присваивания, действие которого не отличается от аналогичных операторов языков программирования. То есть, если в общем виде оператор присваивания записать как

```
имя переменной = значение выражения
```

то в переменную, имя которой указано слева, будет записано значение выражения, указанного справа.

Scilab содержит некоторое количество уже определённых переменных. Все системные переменные в Scilab начинаются с символа %: мнимая единица (%i); число «пи» (%pi); число *e=2.7182818* (%e) и др.

Пакет Scilab снабжён достаточным количеством всевозможных встроенных функций.

Тригонометрические

sin(x)	синус числа x
cos(x)	косинус числа x
tan(x)	тангенс числа x
cotg(x)	котангенс числа x
asin(x)	арксинус числа x
acos(x)	арккосинус числа x
atan(x)	арктангенс числа x

Экспоненциальные

exp(x)	Экспонента числа x
log(x)	Натуральный логарифм числа x

Другие

sqrt(x)	корень квадратный из числа x
abs(x)	модуль числа x
log10(x)	десятичный логарифм от числа x
log2(x)	логарифм по основанию два от числа x

Построение графиков

Функция plot2d

В общем виде обращение к функции имеет вид:

```
plot2d([loglog],x,y,[key1=value1,key2=value2, ..., keyn=valuen]
```

logflag — строка из двух символов, каждый из которых определяет тип осей (n -—нормальная ось, l — логарифмическая ось), по умолчанию "nn";

x — массив абсцисс;

y — массив ординат (или матрица, каждый столбец которого содержит массив ординат очередного графика) (количество элементов в массиве x и y должно быть одинаковым), если x и y — являются матрицами одного размера, то в этом случае каждый столбец матрицы y отображается относительно соответствующего столбца матрицы x;

keyi=valuei — последовательность значений параметров графиков, возможны следующие значения параметров: style — определяет массив (mas) числовых значений цветов графика (id цвета), количество элементов массива совпадает с количеством изображаемых графиков, по умолчанию представляет собой массив masi=i, цвет i-й линии совпадает с номером i, для формирования id соответствующего цвета (кода цвета) можно воспользоваться функцией color, которая по названию (color("цвет")) или коду rgb (color(r,g,b)) цвета формирует нужный id (код) цвета. Если значение стиля отрицательное, то это будет точечный график без соединения точек между собой линиями. Пример построения нескольких графиков различного цвета приведён ниже

```
x=[-2*%pi:0.1:2*%pi];
y=[sin(x); cos(x)];
plot2d(x,y',style=[color("red"), color("blue")]);
```

rect — этот вектор [xmin, ymin, xmax, ymax] определяет размер окна вокруг графика.

```
x=[-2*%pi:0.1:2*%pi];
y=[sin(x); cos(x)];
plot2d(x,y',style=[color("red"),color("blue")],rect=[-8,-2,8,2]);
```

frameflag — параметр определяет окно в котором, будет изображаться график, он может принимать следующие значения: 0 — не вычислять размеры окна, использовать значения по умолчанию или значения из предыдущего графика, 1 — размер окна определяется параметром rect, 2 — размер окна определяется из соотношения между минимальным или максимальным значениями x и y, 3 — размер окна определяется параметром rect в в изометрическом масштабе, 4— размер окна определяется из соотношения между минимальным или максимальным значениями x и y в изометрическом масштабе,

axesflag — параметр, который определяет рамку вокруг графика, следует выделить следующие значения этого параметра: 0 — нет рамки вокруг графика; 1 изображение рамки, ось y слева; 3 — изображение рамки, ось y справа; 5 — изображение осей, проходящих через середину окна;

```
x=[-2*%pi:0.1:2*%pi];
y=[sin(x); cos(x)];
plot2d(x,y',style=[color("red"), color("blue")], axesflag=0);
```

nax — этот параметр используют, если параметр axesflag равен 1, nax представляет массив из четырёх значений [nx, Nx, ny, Ny] — где Nx (Ny) — число основных делений с подписями под осью X (Y), nx (ny) — число промежуточных делений;

leg — строка, определяющая легенды для каждого графика, структура строки такая:

"leg1@leg2@leg3@...@legn", где leg1 — легенда первого графика, ..., legn — легенда первого графика.

Пример построения графиков функций с использованием параметра nax при построении с помощью функции plot2d.

```
x=[-8:0.1:8];
y=[sin(x); cos(x)];
plot2d(x,y',style=[color("red"),color("blue")],axesflag=1,
nax=[4,9,3,6]);
```

Пример построения графиков функции с использованием легенд.

```
x=[-2*%pi:0.1:2*%pi];
y=[sin(x); cos(x)];
plot2d(x,y',style=[color("red"), color("blue")], axesflag=5,
leg="sin(x)@cos(x)");
```

В частности, если необходимо построить график функции y = f(x), необходимо создать массивы значений x и y. Если f(x) будет использована где-то ещё, имеет смысл описать её как функцию:

```
function y = f(x)
  y = 4*exp(-x).*cos(x);
endfunction;
```

после выполнения данного кода возможно использовать f(x) так же, как и встроенные функции Scilab (подробнее об описании функций можно узнать из встроенной справки, выполнив команду help functions).

Таким образом, график функции f(x) на отрезке от 0 до 4 с шагом 0.1 можно построить последовательностью команд:

```
x = 0:0.1:4;
y = f(x);
plot2d(x, y);
xgrid;
```

Функция **xgrid** расчерчивает оси сеткой (при необходимости можно задать цвет и стиль линий сетки, используя параметры).

В Scilab определена также функция **plot**, поддерживающая все основные возможности соответствующей функции .Matlab (полный список совместимых с Matlab функций, а также аналогов, можно увидеть в соответствующем разделе справки).

Гистограммы

Команды для построения гистограмм: **histplot** для двумерной и **hist3d** для трёхмерной гистограмм.

В частности, построить для выборки из 40 значений, равномерно распределённых в диапазоне [−sqrt(5), sqrt(5)], гистограмму с 8 столбцами (чтобы на каждый приходилось в среднем по 5 точек и при этом столбцы были не слишком узкими и не слишком широкими) можно следующим образом:

```
u = grand(40, 1, "unf", -sqrt(5), sqrt(5));
histplot(8,u);
```

Наложение графиков и новые окна

Графики любого вида строятся в новом окне только в случае, когда нет ни одного графического окна, иначе график накладывается на текущее окно. По умолчанию текущим является последнее открытое окно.

Открыть новое окно позволяет функция **scf()**, если её вызвать без параметров или передать дескриптор несуществующего окна. Если функции scf() передать дескриптор существующего окна, оно станет текущим. Возвращает scf() дескриптор открытого или активированного окна. Получить дескриптор текущего окна позволяет функция **gcf()**.

Изменение графика после построения

Граничные значения осей по умолчанию таковы, чтобы график поместился в осях целиком, границы были бы круглыми и при этом остался минимум свободного места. Иногда, если график круто уходит вверх или вниз, это неудобно.

Изменить оси после построения графика можно, получив дескриптор текущих осей функцией **gca()**:

```
-->a = gca();

-->old_data_bounds = a.data_bounds
 old_data_bounds  =
  - 6.2831853  - 0.9999902
    6.2168147    1.
```

Вызвав справку по свойствам осей командой help axes, можно узнать, что поле data_bounds имеет структуру [xmin,ymin; xmax,ymax].

Изменив поле data_bounds для валидной ссылки на оси, можно изменить граничные значения осей на соответствующем графике:

```
-->a.data_bounds = [-8 -2; +8 +2]
```

Если после вызова команды gca(), но до обращения к полученной ссылке окно с графиком было закрыто, то ссылка a уже не будет валидной.

Программирование в Scilab

Работа в Scilab может осуществляться в режиме командной строки, но и в так называемом программном режиме. Для создания программы (программу в Scilab иногда называют сценарием) необходимо:

1. Вызвать команду Editor из меню или запустить любой редактор неформатированного текста.

2. В окне редактора SciPad (или другого) набрать текст программы.

3. Сохранить текст программы с помощью команды File-Save в виде файла с расширением sce, например, lab1.sce.

4. После чего программу можно будет вызвать, набрав в командной строке exec, например, exec("lab1.sce") или вызвав команду меню File-Exec....

Решение нелинейных уравнений

Задача нахождения корня уравнения *f(x) = 0* итерационным методом состоит из двух этапов:

1) отделение корней — отыскание приближённого значения корня или содержащего его отрезка;

2) уточнение приближённых корней — доведение их до заданной точности.

Приближённые значения корней (начальные приближения) могут быть найдены, например, графическим способом.

Итерационный процесс состоит в последовательном уточнении начального приближения *x0*. Каждый такой шаг называется *итерацией*. В результате итераций находится последовательность приближённых значений корня *x1, x2, ..., xn*. Если эти значения с увеличением числа итераций *n* приближаются к истинному значению корня, то говорят, что итерационный процесс *сходится*.

Метод половинного деления

Для нахождения корня уравнения *f(x) = 0*, принадлежащего отрезку *[a, b]*, в котором график *f(x) пересекает* ось абсцисс, делим этот отрезок пополам:

$$c = \frac{a+b}{2} \qquad (8)$$

Если *f(c) = 0*, то *x = c* является корнем уравнения. Если *f(c)* не равно 0 (что, практически, наиболее вероятно), то выбираем ту из половин *[a, c]* или *[c, b]*, на концах которой функция *f(x)* имеет противоположные знаки. Новый отрезок *[a1, b1]* снова делим пополам и производим те же самые действия, пока длина отрезка не станет меньше заданной точности ε.

Метод половинного деления практически удобно применять для грубого нахождения корня данного уравнения, метод прост и надёжен, всегда сходится (но медленно).

Метод простых итераций

Для использования метода простых итераций исходное уравнение *f(x) = 0* заменяется равносильным уравнением

$$x = \varphi(x) \qquad (9)$$

таким, что для всех *x* в рассматриваемой окрестности корня

$$|\varphi'(x)| \le \gamma < 1 \qquad (10)$$

Пусть известно начальное приближение корня *x = x0*. Подставляя это значение в правую часть уравнения (9), получим новое приближение:

$x1 = \varphi(x0).$

Далее, подставляя каждый раз новое значение корня в (9), получаем последовательность значений:

$x_{i+1} = \varphi(x_i),\ i = 0, 1, ...$

Процесс итерирования останавливают, когда $|x_{i+1} - x_i| < \varepsilon$. Если в рассматриваемой окрестности корня $\varphi'(x) < 0$, в этом случае достигается точность ε, в противном случае корень может отстоять от полученного значения больше, чем на ε.

Скорость сходимости последовательности x_i к корню зависит от $|\varphi'(x)|$ в корне x^* и максимальна при $|\varphi'(x^*)|=0$. При $|\varphi'(x^*)|>1$ метод расходится.

Метод Ньютона

Метод Ньютона — развитие метода простых итераций, когда исходное уравнение $f(x) = 0$ заменяется равносильным уравнением $x - f(x)/f'(x) = x$. В этом случае из окрестности любого корня исходного уравнения итерации очень быстро сходятся к этому корню.

fsolve

В Scilab для численного решения нелинейных уравнений предназначена функция **fsolve**.

Функция fsolve в простейшем случае принимает начальное приближение и функцию, ноль которой необходимо отыскать, а возвращает найденное значение:

```
xr = fsolve(1.5, f)
```

Размерность начального приближения должна быть равна размерности предполагаемого корня. Так, в задании 2 данной лабораторной работы x одномерен, в отличие от примера из справки.

Задачи аппроксимации и интерполяции

Пусть $y=f(x)$ — некоторая функция, для которой известны лишь табличные значения, т.е. известно, что при значениях аргумента $x=x0, x1, ..., xn$ функция принимает значения $y =y0, y1, ..., yn$ (таблица 1)

Таблица 1

Экспериментальные данные

xi	x1	x2	x3	x4	x5	x6	x7	...	xn
f(xi)	y1	y2	y3	y4	y5	y6	y7	...	yn

Функция подобного вида может являться результатом проведения измерительного либо вычислительного эксперимента, проведённого с некоторым дискретным шагом.

Возникает задача нахождения промежуточных значений функции и отыскания простого аналитического представления функции.

Если полученная функция может проходить рядом с точками *(xi, yi)*, это задача *аппроксимации*. Если необходимо точное совпадение с заданными значениями в точках *xi*, называемых узловыми, то это — задача *интерполяции*.

Наиболее популярным методом аппроксимации является метод наименьших квадратов, который позволяет по экспериментальным данным подобрать аналитическую функцию заданного вида, которая проходит настолько близко к экспериментальным точкам, насколько это возможно.

Для интерполирования часто используется полиномиальное представление искомой функции — так называемая полиномиальная интерполяция.

Метод наименьших квадратов

Метод наименьших квадратов позволяет по экспериментальным данным подобрать такую аналитическую функцию, которая проходит максимально близко к экспериментальным точкам, причём мерой ошибки служит сумма квадратов отклонений функции от эксперимента в каждой точке.

Пусть в результате эксперимента были получены некоторые данные, отображённые в виде таблицы (табл. 1). Требуется построить аналитическую зависимость, наиболее точно описывающую результаты эксперимента.

Идея метода наименьших квадратов заключается в том, что функцию $Y = f(x, a_0, a_1, ..., a_k)$ необходимо подобрать таким образом, чтобы сумма квадратов отклонений измеренных значений y_i от расчётных Y_i была наименьшей:

$$S = \sum_{i=1}^{n} (y_i - f(x_i, a_0, a_1, \ldots, a_k))^2 \to \min \qquad (1)$$

Задача сводится к определению коэффициентов a_i из условия (1).

Подбор параметров a_i для произвольного случая

Для реализации этой задачи для произвольной функции *f* в Scilab предусмотрена функция

```
[a,S]=datafit(G,Z,a0)
```

где *G* — функция невязки, параметры которой необходимо подобрать; *Z* — матрица исходных данных; *a0* — вектор начальных приближений параметров; *a* — вектор параметров; *S* — сумма квадратов отклонений измеренных значений от 0. Функция *datafit* позволяет подобрать вектор параметров *a* так, чтобы приближённо

решить систему уравнений $G(a, z\_i)=0$, где z_i — единичное измерение — столбец матрицы Z. Если необходимо решить систему уравнений вида $y=f(a, x)$, то функция невязки будет иметь вид $G(a, [x; y]) = y – f(a, x)$, а матрица измерений Z — состоять из двух строк — $Z = [Y; X]$.

Для поиска минимума суммарной невязки используется итерационный квазиньютоновский метод (отличается от метода Ньютона тем, что вместо производной используется её численная оценка).

Подбор коэффициентов СЛАУ

Если функция f линейна относительно коэффициентов a_i, например, имеет вид полинома

$$f(a, x) = a_0 + a_1x + a_2x^2 + \ldots + a_kx^k \quad (2)$$

то система уравнений вида $y = f(a, x)$ — система линейных алгебраических уравнений относительно a:

$$\begin{pmatrix} 1 & x_0 & \ldots & x_0^k \\ \ldots & & \ldots & \\ 1 & x_n & \ldots & x_n^k \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ \ldots \\ a_k \end{pmatrix} = \begin{pmatrix} y_0 \\ \ldots \\ y_n \end{pmatrix} \quad (3)$$

то есть

$$W*A = Y$$

где W — матрица (если f — многочлен — матрица Вандермонда), рассчитанная по вектору X, A — искомый вектор-столбец коэффициентов, Y — вектор-столбец измеренных значений. В общем случае W не квадратна, в типичной задаче аппроксимации её строк (измерений) больше, чем столбцов (параметров).

Эту систему можно свести к системе с квадратной матрицей домножением обеих частей на транспонированную матрицу W: $(W'*W)*A = W'*Y$ и затем решить задачу, обратив $(W'*W)$. Полученное решение соответствует условию (1).

В Scilab для решения переопределённых систем линейных алгебраических уравнений методом наименьших квадратов предназначен оператор левого деления:

$$A = W\backslash Y$$

Решение переопределённой СЛАУ быстрее и точнее, чем использование функции datafit.

Полиномиальная интерполяция

В случае полиномиальной интерполяции (необходимо точное совпадение с заданными значениями в точках *xi*), а интерполирующая функция ищется в виде полинома степени *n*

$$u(x) = c0 + c1\,x + c2\,x^2 + c3\,x^3 + \ldots + cn\,x^n \quad (1)$$

Коэффициенты *c0, ..., cn* определяются из условия

$$u(xi) = yi \quad (2)$$

Существует единственный многочлен степени *n*, принимающий в заданных *n +1* точках заданные значения.

Для вычисления значений интерполяционного полинома не обязательно решать систему (2). Существует множество различных формул для полиномиальной интерполяции.

Наиболее общей из них является формула Лагранжа

$$L_n(x) = \sum_{j=0}^{n} y_j l_j(x) \quad (3)$$

где функции $l_j(x)$ конструируются так, чтобы принимать значение 1 в точке *xj* и значение 0 во всех остальных точках *xi*:

$$l_j(x) = \prod_{i=0, j\neq i}^{n} \frac{x - x_i}{x_j - x_i} = \frac{x - x_0}{x_j - x_0} \cdots \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdots \frac{x - x_n}{x_j - x_n} \quad (4)$$

Интерполяционный полином в форме Ньютона *Pn(x)* записывается в виде:

$$P_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \ldots + c_n(x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_{n-1}) \quad (5)$$

Очевидно,

$$c0 = Pn(x0) = y0 \quad (6)$$

последующие коэффициенты можно рассчитать по формуле

$$c_i = \frac{y_i - \left(c_0 + \ldots + c_{i-1} \cdot (x_i - x_0) \cdot \ldots \cdot (x_i - x_{i-2})\right)}{(x_i - x_0) \cdot \ldots \cdot (x_i - x_{i-2}) \cdot (x_i - x_{i-1})} \quad (7)$$

Так как при заданных *(xi, yi)* существует единственный интерполяционный полином, эти формулы приведут к одному и тому же результату.

Контрольные вопросы

1. Каково назначение рабочей области SciLab?

2. Что такое сценарии SciLab?

3. Какова основная идея итерационных методов решения уравнений?

4. Перечислите известные методы численного решения уравнений.

5. Каковы задачи аппроксимации и интерполяции таблично заданных функций?

6. Какова основная идея метода наименьших квадратов?

Задание на лабораторную работу

Внимание!

Данная лабораторная работа выполняется парой студентов. По результатам работы оформляется один отчёт. Если л/р сдаётся в день написания, отчёт не оформляется.

1. Построить график правой части уравнения, соответствующего номеру варианта. Локализовать корень (корни) уравнения. Выбрать начальное приближение(я) для дальнейшего итерационного поиска.

2. Найти с заданной пользователем точностью корень заданного уравнения, используя стандартные функции Scilab.

3. Найти параметры нормального и равномерного распределений такие, чтобы матожидание было равно 0, а дисперсия 1.

Написать скрипт, по заданному N рассчитывающий:

— N случайных значений u_i, распределённых нормально с нулевым матожиданием и дисперсией 1;

— N случайных значений g_i, распределённых равномерно с нулевым матожиданием и дисперсией 1.

и строящий гистограммы распределения u_i и g_i.

Выполнить расчёт и построение гистограмм для N = 10, 100, 1000, 10000 (для каждого N повторить не менее 4 раз).

При каких значениях N можно отличить нормальное распределение от равномерного по гистограмме более чем в половине случаев?

4. Написать скрипт для аппроксимации таблично заданной функции прямой, параболой, многочленом третьей степени и функцией, соответствующей варианту (таблица 2). Для полученных аппроксимирующих функций необходимо построить гладкие графики; исходные данные (x,y) должны отображаться на графике в виде отдельных точек.

Выбрать произвольные 4 точки *(x, y)* и аппроксимировать полученную зависимость.

Какой из графиков проходит ближе всего к исходным четырём точкам?

Почему?

Для подбора каких параметров возможна аппроксимация при помощи решения СЛАУ (оператор «матричного деления»), а где нужна итерационная подгонка (функция datafit в Scilab)?

5. Аппроксимировать зависимость, соответствующую варианту (таблица 3).

Какой из графиков проходит ближе всего к исходным восьми точкам?

Какой выглядит наиболее правдоподобно?

Какой функцией вы бы предложили аппроксимировать данную зависимость?

Таблица 1

Варианты заданий

Вариант	**Задание**	
1, 24	$0 = 1.2x^2 - \sin(10x)$	все положительные корни
2, 25	$0 = 2^x - 2x^2 - 1$	положительные корни
3, 26	$0 = 2 \ln x - 1/x$	все корни
4, 27	$0 = 2 \lg x - x/2 + 1$	положительные корни
5, 28	$0 = \lg x - 7/(2x + 6)$	все корни
6, 29	$0 = x \lg x - 1/2$	все корни
7, 30	$0 = \lg(3x - 1) + \exp(2x - 1)$	все корни
8, 31	$0 = \exp(-x) - 2(x - 1)^2$	все корни
9, 32	$0 = 2 - x \exp(x)$	все корни
10	$0 = 1/x - \pi \cos(\pi x)$	минимальный положительный корень
11	$0 = \sec(x) - x^2 - 1$	минимальный положительный корень
12	$0 = \mathrm{ctg}(1, 05x) - x^2$	минимальный положительный корень

13	$0 = 2x - \lg(x) - 7$	все положительные корни
14	$0 = \exp(-x) + x^2 - 2$	отрицательные корни
15	$0 = 0,5x^2 - \cos(2x)$	все корни
16	$0 = \ln(0,5x) - 0,5\cos(x)$	все корни
17	$0 = \ln(2x) - \exp(2x)$	все корни
18	$0 = \exp(-x) + x^3 - 3$	все положительные корни
19	$0 = 2x^2 - \cos(2x)$	все корни
20	$0 = x^2 - 20\sin x$	все корни
21	$0 = x^2 - \sin 5x$	все корни
22	$0 = \ln x + (x+1)^3$	все корни
23	$0 = 2.2x - 2^x$	наименьший корень

Таблица 2

Варианты функций

(Вариант–1)%4+1	Задание
1	$\alpha \cdot \exp(\beta x) + \gamma$
2	$\alpha \cdot \cos(\beta x) + \gamma$
3	$b / (c \cdot x + d)$
4	$\alpha \cdot \sin(\beta \cdot x) / x + \gamma$

Таблица 3

Варианты исходных данных (точки)

(Вариант–1)%7+1	Задание
1	x = [1.28 2.00 2.91 4.07 5.00 6.00 7.01 7.89] y = [2.36 3.15 3.55 4.00 4.41 5.08 5.49 6.12]
2	x = [0.78 1.85 3.05 3.84 4.91 5.78 6.99 7.77] y = [3.55 4.48 4.72 4.46 3.55 1.95 -0.02 -2.71]
3	x = [0.80 2.09 3.08 3.91 5.10 5.89 6.99 7.70] y = [-20.59 -2.97 3.36 4.93 5.13 4.95 3.33 -3.28]
4	x = [0.94 1.96 3.10 4.18 5.23 6.07 6.87 8.02] y = [0.03 0.35 0.32 0.90 2.98 3.28 0.80 0.30]
5	x = [-1.10 -0.33 -0.21 1.01 1.36 2.45 2.77 3.26] y = [3.73 5.08 2.37 1.57 5.74 2.25 5.12 3.68]
6	x = [1.03 1.94 2.84 4.06 5.00 5.94 6.88 7.90] y = [1.03 0.74 2.51 1.38 0.52 2.15 1.72 0.58]
7	x = [0.93 2.04 2.98 4.00 4.85 6.04 7.02 7.95] y = [0.01 -0.84 -0.50 0.62 1.61 1.43 0.35 -0.63]

Требования к отчёту

Отчёт должен содержать:

- заголовок, тему и цель лабораторной работы;
- группу и фамилию авторов;
- задания и варианты заданий (полностью, в частности, для заданий 1-2 необходимо привести не только уравнение, но и множество искомых корней);
- листинги сценариев или история команд;
- результаты работы (графики, значения корней и гистограммы);

- ответы (полностью, в частности, для задания 1 необходимо привести не только график левой части уравнения, но и количество корней, а также начальные приближения для каждого корня; для задания 2 необходимо проверить правильность полученного результата; все значения должны быть прокомментированы);
- пояснения;
- выводы.