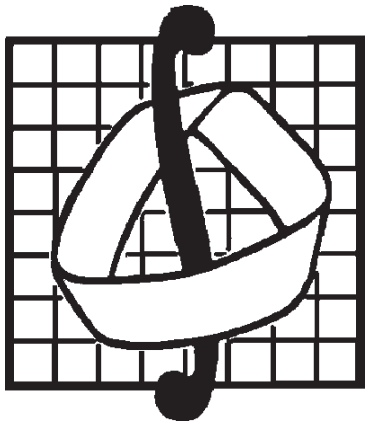
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ
имени М.В.ЛОМОНОСОВА



Механико-математический факультет
Кафедра вычислительной математики

А.В. Попов

GNUPLOT
и его приложения

Москва 2015 год

ББК 32.97
П 58
УДК 004.42+004.91+004.92

Попов А.В. GNUPLOT и его приложения. — М.: Издательство попечительского совета механико-математического факультета МГУ, 2015, – 240 с.

Книга посвящена популярному, свободно распространяемому графическому пакету Gnuplot, предназначенному для создания графических иллюстраций с использованием различных источников данных. С помощью Gnuplot могут быть созданы самые разнообразные графические иллюстрации, начиная от простейших графиков и диаграмм до очень сложных трехмерных объектов. Пакет может быть установлен на компьютеры, работающие под управлением всех популярных операционных систем. Наборы команд почти без изменений переносятся с одной системы на другую. Команды GNUPLOT имеют естественный синтаксис и легко запоминаются.

Создаваемые графики и анимационные фильмы допускают различные форматы сохранения. В книге описаны способы вставки полученных графических иллюстраций и анимаций в документы, набранные в системе LaTeX, для получения профессионально выглядящих книг, статей в научных журналах, отчетов, докладов и презентаций.

Книга предназначена всем тем, кому требуется качественно иллюстрировать свои результаты. Она может также быть интересна школьникам старших классов, интересующимся способами наглядного представления информации.

Рецензенты:
д.ф.-м.н., доц. К.Ю.Богачев,
д.ф.-м.н., проф. А.А.Корнев,
д.ф.-м.н., проф. Е.В.Чижонков.

Содержание

Предисловие

С понятием графика человек знакомится еще в школе задолго до выпускного класса. Это неслучайно, поскольку графическая интерпретация результатов человеческой мысли играет огромную роль в передаче новых знаний и в объяснении другим людям своих идей. Причем графическая информация не является прерогативой лишь естественных наук. Графический способ передачи информации не меньше, а под час и больше, распространен в гуманитарных дисциплинах, бизнесе, инженерных дисциплинах и других сферах интелектуальной деятельности человека. Тем самым легко объяснимо огромное количество платных и бесплатных программных пакетов, которые созданы для выполнения задачи о построении графических иллюстраций.

Как правило, в графическом виде требуется представить либо зависимости, заданные математическими формулами, либо полученные в результате измерений или каким либо другим образом дискретные данные, хранящиеся в файлах в виде таблиц. Часть данных из этих таблиц может интерпретироваться как координаты точек графиков, но таблицы могут содержать и другие полезные сведения. Для этой работы можно использовать любой доступный графический пакет. Одним из таких пакетов является бесплатно распространяемая программа Gnuplot. Ее можно скачать в интернете с сайта **www.gnuplot.info**. Далее будем считать, что нужная версия Gnuplot стоит на компьютере читателя и доступна для запуска из любой директории. Содержание этого пособия не претендует на полноту, но может послужить отправной точкой для изучения возможностей Gnuplot.

Помимо бесплатности Gnuplot обладает еще несомненными преимуществами. Приведем главные.

Во-первых, он прост в использовании. Его команды, особенно для создания не перегруженных информацией графиков, естественны и легко запоминаются.

Во-вторых, его инструменты достаточно разнообразны. Они могут удовлетворить запросы как новичков, так и умудренных опытом профессионалов.

В-третьих, это универсальность пакета, который работает под управлением всех распространенных популярных операционных систем, и как следствие, переносимость программных файлов между компьютерами независимо от установленных на них систем.

Предлагаемая книга знакомит читателя с используемой терминологией, основными командами и их ключами, применяемыми для построения различных графических иллюстраций. В ней описаны все необходимые первичные навыки. При необходимости можно дополнить свои знания, обратившись к документации Gnuplot, которая прилагается к пакету и может быть вызвана командой **help**, а также используя многочисленные сайты, которые доступны в интернете.

Книга написана так, чтобы она могла быть полезна для читателей разных категорий подготовленности: как для начинающих свое знакомство с Gnuplot специалистов различных областей, так и для людей, уже знакомых с этим пакетом. Последние могут использовать эту книгу как справочник.

Вся книга состоит из тринадцати глав, которые в свою очередь делятся на параграфы. Для удобства пользования книгой каждый параграф разбит на озаглавленные части. Заголовки кратко характеризуют то, чему посвящена соответствующая часть параграфа. Все эти заголовки вынесены в оглавление, из которого тем самым можно быстро понять какое место нужно прочитать, чтобы решить конкретный вопрос.

Первая глава "Для нетерпеливых читателей" написана именно для пользователей, имеющих опыт работы и желающих быстро разобраться в том, как можно, применяя Gnuplot, сделать их изображения более наглядными и информативными. Поэтому тем, кто лишь начинает работать с Gnuplot, имеет смысл читать книгу со второй главы, ознакомившись предварительно лишь со "Введением", где описаны необходимые сведения по запуску и управлению работой в пакете.

Вторая глава "Первые шаги по знакомству с Gnuplot" знакомит читателя с минимальными знаниями, необходимыми для построения простейших дву- и трехмерных графиков и контуров поверхностей. В этой главе приведены примеры построения раз-

нообразных графиков, даны краткие пояснения какой ключ за что отвечает и даны ссылки на другие разделы книги, где можно ознакомиться с их возможностями. Главы с третьей по десятую содержат параграфы, каждый из которых разъясняет возможности и особенности применения конкретных команд и их опций. Одиннадцатая глава посвящена созданию командных файлов, которые просто необходимо создавать для автоматизации обработки различных экспериментальных данных. В двенадцатой главе рассмотрен вопрос о вариантах сохранения графических изображений в различных форматах. Особое место уделено их вставке в тексты, набранные в системе LaTeX. В завершающей тринадцатой главе описаны возможности Gnuplot для создания анимационных иллюстраций и вставки их в электронные тексты докладов, презентаций и т.п.

Материал каждой главы и параграфа расположен по возможности в следующем порядке: сначала идет описание формального синтаксиса описываемой команды или опции; затем следуют примеры их использования и в конце описываются параметры, необходимые для тонкой настройки работы пакета. Учитывая еще то, что большинство ключей независимы друг от друга, книгу можно изучать не в порядке изложения материала, а по мере необходимости обращаться к различным ее разделам. Для удобства поиска нужного места в тексте в конце книги находится предметный указатель основных терминов.

Большая часть описанных возможностей проиллюстрирована в виде примеров команд или их наборов. Часть этих примеров содержит подробные комментарии, для чего та или иная команда была в них использована, и дается ссылка на место в книге, где можно ознакомиться с назначением команды, ее опциями и синтаксисом. Ссылки оформлены в виде номеров раздела и первой его страницы. Например, (см. 5.1, с. 110) означает, что для получения подробной информации следует читать раздел 5.1, начинающийся на странице 110. Результат работы бóльшей части из приведенных наборов команд показан на рисунках.

Автор благодарит за плодотворные обсуждения, конструктивную критику и многочисленные советы своих рецензентов д.ф.м.н. Богачева К.Ю., д.ф.м.н. Корнева А.А. и д.ф.м.н. Чи-

жонкова Е.В., а также коллег к.ф.м.н. Григорьева И.С., Самохина А.С. и к.ф.м.н. Соколова А.Г., поделившихся своим личным опытом по использованию Gnuplot и его приложений. Большая помощь была оказана моим другом начальником управления "Автоматизации" московского АЛЬТА-Банка к.ф.м.н. Чевелевым К.В. и моей женой к.ф.м.н. Поповой О.П. Автор благодарен за внимание к работе и помощь при опубликовании руководству кафедры Вычислительной математики механико-математического факультета МГУ профессору Кобелькову Г.М. и доценту Лапшину Е.А.

Все замечания и пожелания просьба отправлять автору по адресу

popov.gnuplot@yandex.ru

Введение

Пакет Gnuplot предназначен для построения всевозможных графиков и гистограмм. Режим работы с ним интерактивный. Запуск осуществляется командой **gnuplot**. Выход из интерактивного режима — **quit** или **exit**. В интерактивном режиме управление процессом построения графиков осуществляется с помощью достаточно простого внутреннего языка. Последовательность команд можно записывать в файлы, которые в интерактивном режиме запускаются командой **load 'file.name'**. Перенос части команды на следующую строку допускается, если в конце строки, где записано начало команды, поставить символ \. В строке файла может содержаться несколько команд. Для этого их необходимо разделять точкой с запятой. Запуск командных файлов может быть осуществлен также из командной строки командой **gnuplot file.name** без запуска интерактивного режима.

Содержимое строк после символа # игнорируется. В частности, это можно использовать для включения комментариев в командный файл.

В интерактивном режиме часто возникает потребность запустить уже использованную команду, либо аналогичную ей. В таких случаях полезным инструментом является возможность отобразить в командой строке предыдущую команду. Это легко сделать, нажав на клавиатуре клавишу "стрелка вверх". Повторное нажатие этой клавиши приводит к появлению очередной предыдущей команды из списка использованных. Клавиша "стрелка вниз" приводит к появлению команды из списка, следующей за текущей. Вызванные таким образом команды перед повторным исполнением можно редактировать обычным образом. В разделах 'HISTORY' и 'SET HISTORY' [1] описаны другие возможности, позволяющие оперативно и удобно возвращаться к определенной использованной ранее команде.

Большая часть команд Gnuplot предназначена для задания различных опций команд **plot** и **splot**, выполняющих непосредственное построение графиков функций одной и двух переменных соответственно. Многие опции могут быть заданы в строках

самих команд **plot** и **splot**. В этом случае заданные значения опций действуют лишь для тех команд, в которых они были указаны. В случае, когда какая-то опция явно не определена, ее значение определяется по умолчанию.

Значения опций, определенные по умолчанию, можно изменить специальной командой

set option_name parametr_1 . . . parametr_N

Широкий спектр значений параметров обеспечивает разнообразие возможностей, предоставляемых Gnuplot. Отметим, что командой **set** можно задать и такие опции, которые не входят в число ключей команд **plot** и **splot**, но также влияющие на те или иные свойства получаемых изображений.

Для возвращения опции в состояние, устанавливаемое по умолчанию, и для вывода параметров, задающих текущее состояние опции, используются команды

unset option_name
show option_name

Все заданные значения опций можно отменить командой **reset**, а полные возможности конкретной команды узнать, набрав в интерактивном режиме команду

help name_command

Отметим, что команда **reset** не отменяет настроек, заданных командами

'set term' 'set output' 'set loadpath'
'set fontpath' 'set linetype' 'set encoding'
'set decimalsign' 'set locate' 'set psdir'

По умолчанию для изображения графиков Gnuplot создает окно, в котором отображаются текущие построения. В интерактивном режиме есть набор "горячих клавиш", по нажатию которых в окне меняются те или иные параметры построенного графика. Например, очень удобно подбирать таким образом нужный ракурс обзора (см. 2.3, с. 46) или диапазоны изменения

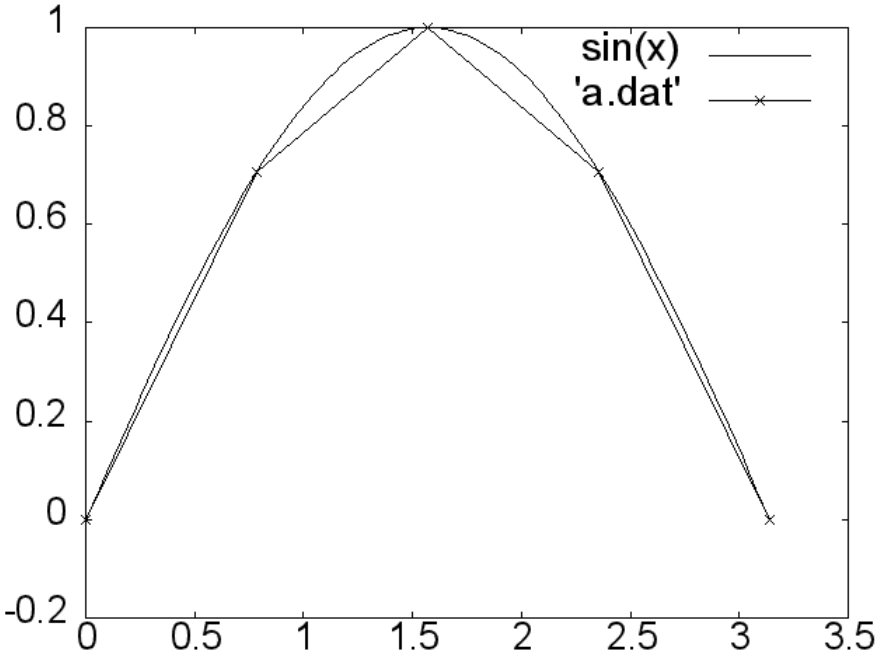


Рис. 1: Простейшие графики

независимых переменных. Полный список таких команд можно вывести на экран, нажав клавишу 'h'. Для дальнейшего использования результаты построений нужно сохранять. Способы сохранения описаны в главе 12.

Приведем пример, иллюстрирующий работу Gnuplot. Построим графики функции **y=sin(x)** на отрезке $[0; \pi]$ и функции, заданной с помощью файла данных **a.dat**. Файл **a.dat** содержит следующие пять строк:

```
0.          0.
0.7854      0.7071
1.5708      1.
2.3562      0.7071
3.1416      0.
```

Пара чисел, стоящих в каждой строке этого файла, задает координаты точек графика функции **y=sin(x)** при x равном 0, $\pi/4$, $\pi/2$, $3\pi/4$ и π с точностью 10^{-4}. Команда

plot sin(x), 'a.dat' with linespoints

создает изображение, приведенное на рисунке 1.

Никакого сглаживания для второй функции, заданной при помощи дискретных данных, задано не было. Поэтому построение ее графика свелось к тому, что были нанесены заданные точки на координатную плоскость и соседние точки соединили отрезками. Этот способ построения непрерывной функции по дискретным данным называется линейной интерполяцией [10]. Точки находятся друг от друга на достаточно большом расстоянии, поэтому график функции синус визуально отличается от графика, построенного по дискретным данным. Заметим, что если вместо линейной интерполяции при соединении соседних точек использовать кривые, описываемые кубическими многочленами (такой способ называется построением кубического сплайна), то два графика (графики самого синуса и его приближения, построенного по дискретным точкам) будут визуально практически неразличимы (4.3, с. 105 и см. 4.4, с. 107):

plot sin(x), 'a.dat' w lp smooth csplines

Названия опций можно сокращать до нескольких букв (часто даже до одной). Например, вместо **using** допускается писать просто **u**. В приведенной выше команде это правило было продемонстрировано: от **with** оставили лишь **w**, а от **linespoints** — **lp**. Допустимость конкретного сокращения проще всего проверить экспериментально.

Важную роль в любой иллюстрации играет ее внешний вид. Надписи не следует делать слишком мелкими, т.к. все элементы рисунка должны быть четкими и легко читаемыми. По умолчанию Gnuplot не всегда выбирает размер шрифта достаточно крупный, поэтому лучше подбирать его самим, тем более что это достаточно легко сделать (см. 10.4.2, с. 183). Это замечание особенно актуально при подготовке статей в журналы. Дело в том, что при верстке номеров редакторы часто используют сжатие рисунков, что под час делает некоторые детали в них слишком мелкими. Далее в примерах мы не всегда будем указывать команды, устанавливающие размер шрифта.

В приведенном выше примере размер шрифта у подписей делений на осях и в легенде рисунка был задан следующими командами:
set xtics font ",20"
set ytics font ",20"
set key title font ",22"
Разумеется они должны быть выполнены до команды **plot** или **splot**.

Отметим, что в двумерном случае прямоугольник, где рисуются графики (область изображения), окаймляется прямыми (отрезками), которые играют роль осей координат (см. 5.2, с. 113). Левая сторона прямоугольника считается основной осью y, а нижняя — основной осью x. По умолчанию на них наносятся деления с подписями. На двух других сторонах, которые называются осями координат x2 и y2, наносятся лишь деления. Легенда рисунка была создана самим Gnuplot без каких либо дополнительных команд.

Отрезок $[0; \pi]$ — диапазон именения независимой переменной, был выбран автоматически исходя из дискретных данных. Диапазон изменения зависимой переменной в нашем случае тоже был выбран автоматически исходя из областей значений изображаемых функций.

1 Для нетерпеливых читателей

В этой главе будут приведены примеры наборов команд, позволяющих создать графики функции одной переменной $\mathbf{y} = \mathbf{f}(\mathbf{x})$ и двух переменных $\mathbf{z} = \mathbf{f}(\mathbf{x}, \mathbf{y})$, а также другие полезные графические изображения.

В первом параграфе наборы команд в примерах минимальные. Они строго предназначены для решения конкретной задачи. При желании читатель может взять соответствующий набор за основу командного файла для решения своих задач.

Второй параграф содержит примеры наборов команд, которые предназначены для демонстрации возможностей Gnuplot по созданию изображений, отвечающих запросам требовательного пользователя. Подчеркнем то, что почти все команды в этих наборах не являются необходимыми для построения изображаемых графиков. В приведенных примерах они используются для иллюстрации того, что можно дополнительно сделать с изображением, используя Gnuplot. Например, в случаях необходимости вывода дополнительной информации, привлечения внимания к отдельным фрагментам графика и т.д. Каждая команда снабжена коротким комментарием того, для чего она использована и ссылкой на раздел пособия, где можно прочитать о ней.

Читателям, не планирующим подробно изучать пособие, можно посоветовать дополнительно обратить внимание на следующие примеры 2.9, 3.4, 4.19, 5.2, 5.3, 6.1, 7.1, 7.2, 9.1, 9.2 и 9.3.

1.1 Примеры простых графиков

1.1.1 Функции одной переменной

В качестве первого примера рассмотрим случай, когда требуется на одной иллюстрации привести графики нескольких функций одной переменной, у которых совпадают области определения, а области значений несильно различаются. Gnuplot позволяет рисовать графики функций, заданных по формуле (см. 11.1, с. 193

и 11.2, с. 194) или дискретными данными (см. 2.1, с. 39).

В приведенном ниже примере изображаются графики трех функций. Одна из них определена по формуле, а две другие задаются с помощью файла 'rezult.dat' [1], содержащего три колонки чисел. В первой колонке расположены значения аргумента, а в двух других — значения функций. Каждая колонка задает значения своей функции. В этом случае требуется явно указать способ использования данных с помощью опции **using** (см. 3.2, с. 69). В команде **plot**, непосредственно осуществляющей построение графиков, также использованы опции **with** для задания шаблона (способа изображения) кривых и **title** для создания нужного комментария в легенде.

Приведем команды Gnuplot, с помощью которых было получено изображение, которое можно увидеть на рисунке 2.

Пример 1.1

```
# Задать местоположения легенды слева в нижней части
# изображения (см. 8.2, с. 151).
set key left bottom font ',16'
# Задать подписи к осям в виде надписей 'X' и 'U'
# (см. 5.3, с. 114).
set xlabel "X" font ',16'
set ylabel "U" font ',16'
set xtics font ',16'
set ytics font ',16'
# Задать функцию u формулой, использующей тернарный
# оператор (см. 11.1, с. 193).
u(x)=(x<=1)?1:0
# Задать число используемых для построения графика узлов
# равным 1000 (см. 4.3, с. 105)
set samples 1000
# Нарисовать графики функций (см. 2.2, с. 40). Отметим,
```

[1] Первая из изображенных на рисунке 2 функций является численным решением уравнения переноса по разностной схеме "тренога", вторая — по разностной схеме первого порядка с разностью против потока и третья — точным решением этого уравнения [10]. Графики иллюстрируют точность численных решений на фоне точного.

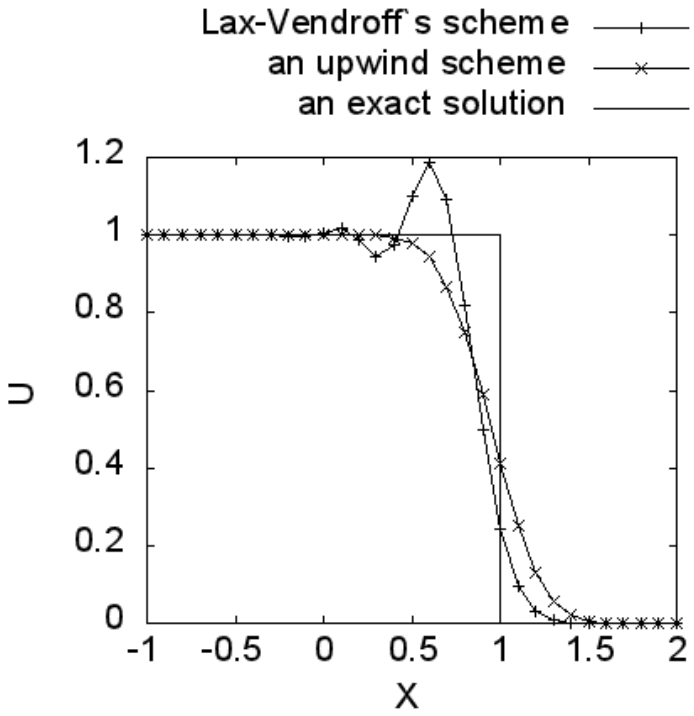


Рис. 2: Графики функций одной переменной

```
# что при повторном использовании в одной команде plot
# одного и того же файла данных его имя можно не указывать,
# но кавычки при этом нужно оставить.
plot 'rezult.dat' \
using 1:2 with linespoints title 'Lax-Vendroff's scheme', \
'' using 1:3 with linespoints title 'an upwind scheme', \
u(x) title 'an exact solution'
```

В случае, если кривые имеют существенно отличающиеся области определения или значений и в то же время их требуется изобразить на одном рисунке, в Gnuplot есть возможность задать две пары осей координат (см. 5.2, с. 113 и 5.7, с. 121) или воспользоваться режимом **multiplot** (см. 9, с. 165).

Gnuplot автоматически старается изображать графики разных функций, нарисованных на одной картинке, меняя цвет и шаблон используемых линий. В случае, если выбор Gnuplot Вас не устраивает, можно задавать способ изображения той или

иной линии явным образом (см. 4, с. 86).

Особенно хочется выделить опцию задания числа используемых узлов (**samples**). Эта опция действует при построении графиков функций, заданных по формуле. По умолчанию это число невелико: всего 100 точек. Узлы — это точки на отрезке, на котором строится график (заданном диапазоне изменения независимой переменной). В этих узлах Gnuplot вычисляет значения функции, заданной по формуле. В результате получаются точки на плоскости, через которые затем проводится кривая, которая принимается за график функции.

В случае функции из рассмотренного примера при опущенной команде **set samples 1000** разрыв у функции **u(x)** смещается из точки **x=1** в точку **x=1+3/100**, что визуально заметно. Другим примером важности этой команды является пример построения графика функции **tg(x)**, что можно увидеть на рисунке 33. Безусловно важным критерием выбора числа узлов является время построения графика: чем больше число узлов, тем дольше Gnuplot тратит время для создания изображения. Особенно это актуально при создании кадров в анимации.

1.1.2 Гистограмма

Приведем пример набора команд, позволяющий построить несложную гистограмму, используя данные из следующего файла с именем 'diag.dat'.

"year"	"number of students"	"2"	"3"	"4"	"5"	"average mark"
2012	30	3	8	15	4	3.67
2013	35	4	10	10	11	3.8
2014	28	2	8	12	6	3.79
2014	31	1	4	15	11	4.06

Эта таблица содержит сведения о результатах сдачи экзамена по некоторой дисциплине студентами в 2012-2014 г.г. Первая строка и первый столбец содержат заголовки данных соответствующих столбцов и строк таблицы соответственно.

Замечание. Элементы первой строки взяты в двойные кавычки для того, чтобы они в дальнейшем при обработке Gnuplot рассматривались как один элемент.

Команды из следующего примера строят изображенную на рисунке 3 гистограмму (типа **columnstacked** (см. 4.2.5, с. 96)), которая наглядно показывает распределение по годам каждого типа оценок.

Пример 1.2

Увеличить верхнее и нижнее поля окна, чтобы
поместились необходимые подписи (см. 10.3.5, с. 180).
set tmargin 2
set bmargin 4
Задать шрифт у подписей осей размера 20.
set xtics font ",20"
set ytics font ",20"
Задать диапазон оси Y.
set yrange [0:55]
Задать название оси X и
напечатать его шрифтом размера 20.
set xlabel "type of marks" font ",20"
Задать ширину прямоугольников гистограммы.
(см. 4.2.5, с. 96).
set boxwidth 0.9 relative
Задать построение гистограммы типа **columnstacked**.
set style histogram columnstacked
Данные считать данными для построения гистограммы.
set style data histograms
Определить тип заливки (см. 10.4.3, с. 184).
set style fill solid 1.0 pattern border lt -1
Задать общий заголовок.
set label 1 "Results of the examinations"
at -0.4,56 font ",26"
Легенду печатать шрифтом размера 16.
set key title left font ",16"
Построить гистограмму типа **columnstacked**, используя
первый столбец файла данных для создания подписей

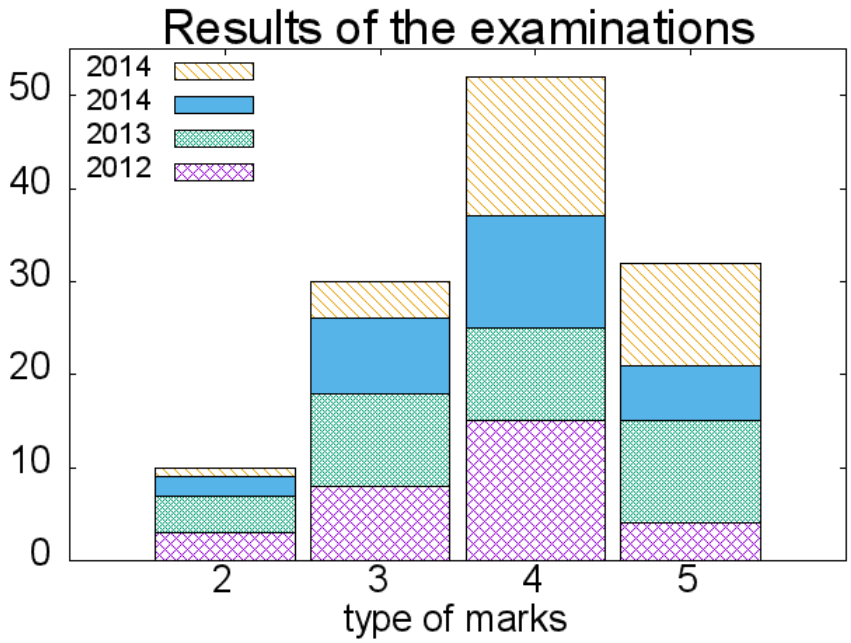


Рис. 3: Гистограмма оценок за 2012-2014 г.г.

```
# легенды, а заголовки столбцов в качестве подписей
# делений горизонтальной оси. В качестве данных использовать
# числа из столбцов с номерами 3, 4, 5 и 6.
plot 'diag.dat' using 3:key(1) t columnhead,\
' ' u 4 t columnhead, ' ' u 5 t columnhead,\
' ' u 6 t columnhead
```

Шаблоны построения гистограмм подробно рассмотрены в разделе 4.2.5. Там же можно найти другие простые примеры. Более сложный пример содержится в разделе 1.2.3.

1.1.3 Функции двух переменных

Гораздо более разнообразным является графическое изображение функций двух переменных. Их описание начнем с примера,

в котором функция задается формулой [2]

$$u(t,x) = \begin{cases} \begin{cases} 1, & x \le t,\ t \in [0,\ 1]; \\ \dfrac{1-x}{1-t}, & t < x < 1,\ t \in [0,\ 1]; \\ 0, & x \ge 1,\ t \in [0,\ 1]; \end{cases} \\ \begin{cases} 1, & x \le \dfrac{t+1}{2},\ \ 2 \ge t > 1; \\ 0, & x > \dfrac{t+1}{2},\ 2\ \ge t > 1. \end{cases} \end{cases}$$

Приведем команды Gnuplot, с помощью которых было получено изображение, которое можно увидеть на рисунке 4.

Пример 1.3

```
# Задать общий заголовок изображению и его местоположение
# (см. 8.2, с. 151).
set key title 'an exact solution of Hopf's equation'
set key at graph 1.5, 0.5, 2.35 font ',16'
# Задать диапазоны изменения независимых переменных
# (см. 5.1, с. 110).
set yrange [0:2]
set xrange [-0.5:2]
# Задать подписи к осям (см. 5.3, с. 114)
set xlabel "x" offset 0,-0.5 font ',16'
set ylabel "t" font ',16'
set zlabel "U" offset 1.1,1,1.5 font ',16'
set xtics offset 0,-0.5 font ',16'
set ytics offset 1,-0.25 font ',16'
set ztics font ',16'
# Задать формулу функции.
u1(x,y)=(1.-x)/(1.-y)
v0(x,y)=(x<=y)?1:((x>=1)?0.:u1(x,y))
v1(x,y)=(x<=(y+1)/2)?1.:0.
u(x,y)=(y<1)?v0(x,y):v1(x,y)
# Задать число используемых для построения графика узлов
```

[2] Эта функция является точным решением уравнения Хопфа с разрывным начальным условием $u_0 = u(0,x)$ при $t \in [0,\ 2]$.

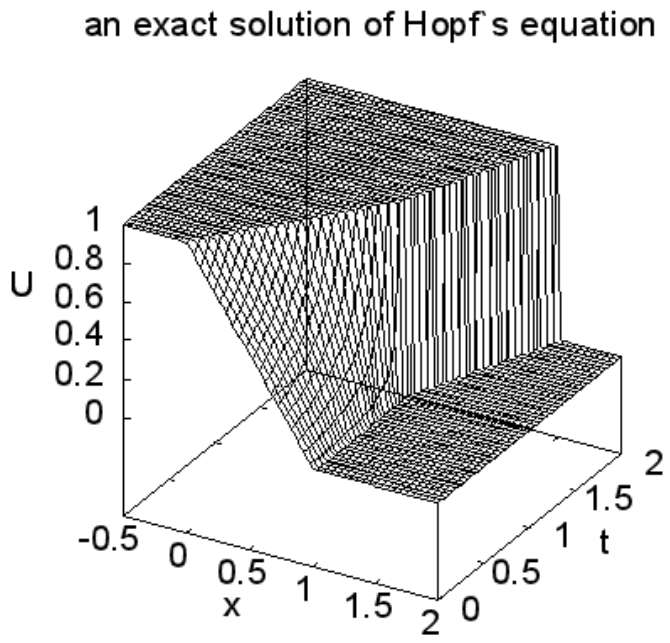


Рис. 4: График функции двух переменных

```
# равным 100 (см. 4.3, с. 105).
set samples 100
# Задать число рисуемых изолиний по каждому направлению
# равным 50 (см. 6.2, с. 129).
set isosamples 50
# Нарисовать поверхность шаблоном lines без легенды
# (см. 2.3.1, с. 46).
splot u(x,y) with lines title ' '
```

Замечание. Смещения надписей были указаны для того, чтобы избежать наложений различных элементов изображения друг на друга.

Также как и в случае функций одной переменной Gnuplot автоматически выбирает все опции для трехмерных изображений. Однако эта задача более сложная из-за возрастающего числа параметров и для создания изображения, устраивающего автора,

требуется часть опций задавать явно. К уже описанным выше добавляются еще задание ракурса обзора, положение плоскости XY и режим удаления скрытых линий, которые важны именно в трехмерном случае (см. 6, с. 128).

Особую роль играет возможность рисовать цветные поверхности с помощью шаблона **pm3d** (см. 7, с. 135). Приведем пример такого изображения поверхности, заданной с помощью файла с данными с именем 'u.dat' [3]. Файл 'u.dat' содержит четыре колонки чисел. Первые две — это значения независимых переменных, а значение функции вычисляется как корень из суммы квадратов значений из третьего и четвертого столбцов [4].

Приведем команды Gnuplot, с помощью которых было получено изображение, которое можно увидеть на цветной иллюстрации I (с. 233).

Пример 1.4

```
set xlabel "X"
set ylabel "Y"
set zlabel "U"
# Использованы сокращения:
#using – u, with – w, title – t.
splot 'u.dat' u 1:2:(sqrt($3*$3+$4*$4)) w pm3d t ' '
```

В команде **splot** для задания функции **u(x)** использована опция **using** (см. 3.2, с. 69), явно определяющая, что значения независимых переменных считываются из первых двух столбцов, а значение самой функции вычисляется по формуле $\sqrt{\$3^2 + \$4^2}$, где \$N — значение из столбца с номером N текущей строки (см. 11.1, с. 193).

Важно заметить, что в случае двух независимых переменных дискретные данные с построчным форматом записи (см. 3.1.1, с. 63) должны быть записаны поблочно. Блоком называют последовательность строк, не содержащую пустой строки. Блоки

[3] В файле 'u.dat' хранятся результаты расчета функции скорости двумерной задачи о каверне. Каждая строка файла содержит координаты одной точки и вектора скорости в этой точке.

[4] Таким образом, изображаемая функция является модулем вектора скорости в точках сетки в области каверны.

разделяются пустыми строками. В одном блоке записываются данные о функции в узлах, в которых значение одной из независимых переменных фиксировано.

При описании важных опций команды **splot** нельзя не упомянуть о задании числа используемых изолиний **isosamples** (см. 6.2, с. 129). Эта опция также как и опция **samples** играет роль лишь для функций, задаваемых по формуле. От задания этого параметра зависит насколько часто будут проводится изолинии, которыми изображается поверхность (см. рис. 36), а в случае шаблона **pm3d** — "пятнистость" изображения (см. рис. XI и XIII). По умолчанию число изолиний по каждому направлению равно 10.

1.1.4 Поле скоростей

В Gnuplot есть шаблон изображения векторов (см. 4.2.6, с. 101). С его помощью можно, например, изображать векторные поля, которые часто используют для иллюстрации течений жидкости и газа. Для примера возьмем файл 'u.dat', использованный в предыдущем примере, и построим векторное поле, заданное в этом файле, с помощью следующих команд.

Пример 1.5

```
# Задать одинаковый масштаб по вертикали и горизонтали.
set size ratio 1
# Задать название изображения и разместить его по центру
# верхнего поля изображения.
set key title 'a field of velocity'
set key center tm font ',18'
# Задать название осей.
set xlabel "X" font ',16'
set ylabel "Y" offset 1,0 font ',16'
set xtics 0.2 offset 1,0 font ',16'
set ytics offset 1,0 font ',16'
# Нарисовать поле скоростей.
plot 'u.dat' every 4:4:4:4:60:60 using 1:2:($3/4):($4/4)\
with vectors title ' '
```

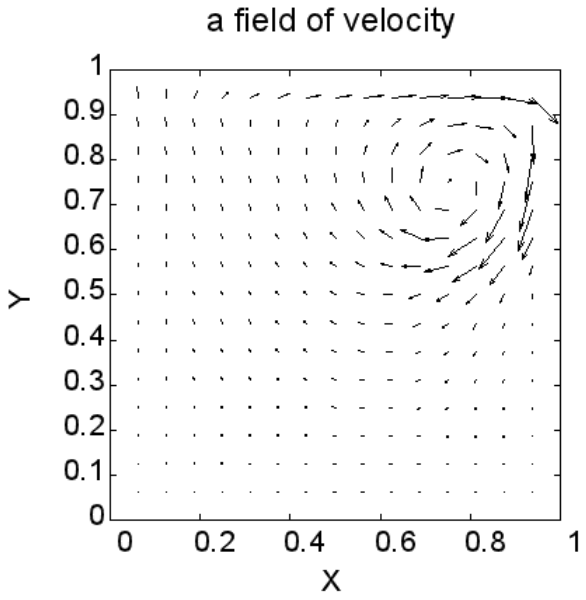


Рис. 5: Поле скоростей

Замечание 1. Для того, чтобы вектора на рисунке не изображались слишком часто и не пересекались, в команде **plot** были использованы опция **every** (см. 3.4, с. 77), задающая изображение вектора скорости лишь в каждой четвертой точке, и уменьшение всех длин векторов в четыре раза.

Замечание 2. Шаг расположения делений на оси X был задан 0.2 для того, чтобы отделить подписи различных делений друг от друга. Можно было пойти и другим путем. Оставить шаг расположения 0.1, выбираемый Gnuplot по умолчанию, но повернуть подписи на 90^o, использовав ключ **rotate**.

1.1.5 Линии уровня

Еще одним важным приемом, используемым при визуализации данных, является построение линий уровня (контуров поверхности) (см. 2.4, с. 56). В качестве примера приведем команды, создающие рисунок с линиями уровня поля скоростей из файла 'u.dat'.

Пример 1.6

```
set key title 'field of velocity
set key at 0.8, 1.15 font ',18'
set xlabel "X" offset 0,2 font ',16'
set ylabel "Y" offset -2,0 font ',16'
set xtics font ',16'
set ytics font ',16'
# Задать ракурс обзора сверху на плокость XY.
set view 0,0
# Задать равный масштаб по осям X и Y.
set view equal xy
# Рисовать контуры на нижней границе области изображения.
set contour base
# Задать значения рисуемых линий уровня.
set cntrparam levels discrete \
0.001, 0.005, 0.01, 0.025, 0.05, 0.075, 0.1, 0.5
# Не изображать поверхность и подписи к делениям оси Z.
unset surface
unset ztics
# Задать изображение всех контуров одним цветом.
set cntrlabel onecolor
# Задать начальное положение подписи к контуру и шаг
# вывода подписей. Отрицательный шаг означает, что
# подпись будет одна.
set cntrlabel start 5 interval -1
# Задать тип шрифта и его размер для подписей.
set cntrlabel font "Times, 10"
# Нарисовать линии уровня поля скоростей,
# записанного в файл 'u.dat'.
splot [0.:1.] [0.:1.] 'u.dat'\
using 1:2:(sqrt($3*$3+$4*$4)) with lines title ",\
' ' using 1:2:(sqrt($3*$3+$4*$4)) with labels title ' '
```

Линии уровня могут рисоваться как на самой поверхности, так и на верхней или нижней границах области изображения (см. 2.4, с. 56). До версии 5.0 Gnuplot контуры могли раскрашиваться в разные цвета, а из легенды изображения можно было

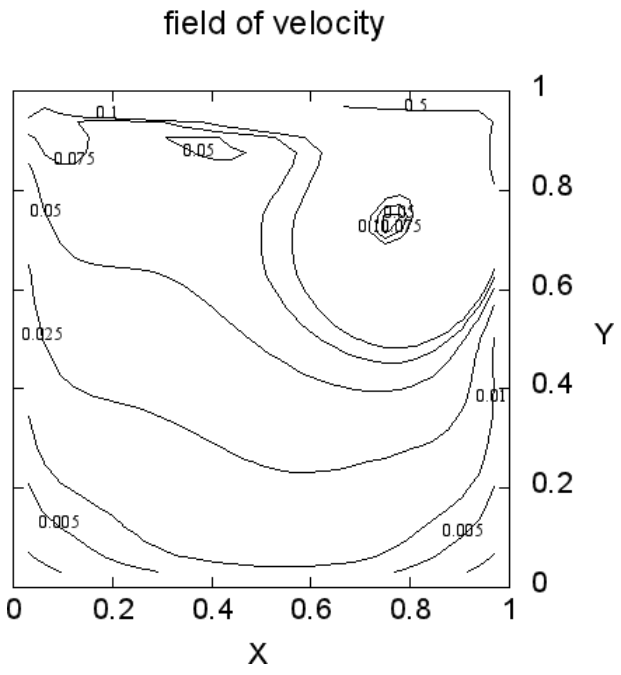


Рис. 6: Линии уровня поля скоростей

понять по цвету контура какое значение его задает. В версии 5.0 появилась возможность подписывать числовые значения контуров прямо на изображении, что безусловно очень удобно в случае черно-белой графики.

1.1.6 Цветная проекция

Возможность рисовать в цвете делает доступным еще один прием изображения разного рода данных. Рассмотрим его опять же на примере поля скоростей, записанного в файле 'u.dat'. Приведем цветную проекцию на нижнюю границу области изображения поверхности, определяемой модулем скорости. Следующий набор команд создает изображение, которое можно увидеть на цветной иллюстрации II (с. 233).

Пример 1.7

```
set key title "a color picture of velocity"
set xlabel "X"
```

```
set ylabel "Y"
unset colorbox
set view equal xy
set view 0, 0
unset ztics
# В следующей команде названия ключей using, with
# и title были сокращены до одной буквы
# согласно правилу, описанному во введении.
splot 'u.dat' u 1:2:(sqrt($3*$3+$4*$4)) w pm3d at b t ' '
```

Безусловно, что при наличии лишь черно-белой палитры картинка теряет многие свои полезные свойства, но даже в черно-белых тонах изображение может донести смысл иллюстрируемого эффекта.

Различные комбинации описанных способов создания графических иллюстраций позволяют полноценно представлять разного рода данные в статьях, книгах, докладах, презентациях и т.д. Возможность объединять стационарные изображения в анимационные фильмы предоставляет еще больше возможностей (см. 13, с. 216).

1.2 Примеры усложненных графиков

1.2.1 Кубическая парабола

Рассмотрим процесс построения графика кубической параболы, заданной формулой

$$\mathbf{y(x) = x^3 - x.}$$

На данном примере покажем как, используя возможности Gnuplot, можно:

1) изображать участки графика одной функции различными стилями, т.е. менять цвет, толщину, символы для узлов и т.д. рисуемой линии в зависимости от заданных условий;

2) задавать основные и дополнительные деления по осям, менять их размер, частоту, вставлять дополнительные деления и т.д.;

3) задавать различные форматы подписей к делениям и осям;

4) оформлять легенду изображения и задавать ее местоположение;

5) выводить дополнительные надписи, содержащие пояснения к графикам, задавать их местоположение, стиль надписи, ее цвет, размер и т.д.;

6) рисовать дополнительные отрезки разными стилями (пунктирной линией, со стрелочками на концах), отрезки могут начинаться и заканчиваться в любых точках изображения;

7) вывод по формату в пределах надписи значений различных переменных, вычисляемых Gnuplot в процессе построения графика;

8) изображение дполнительных объектов (в нашем случае это закрашенные круги), их размеров, цвета и т.д.

Набор команд, приведенных ниже, содержит также ряд других полезных предписаний: русификация Gnuplot, выбор типа терминала, размера окна, имени файла для сохранения результатов.

Приведем команды, позволяющие построить изображение, приведенное на цветной иллюстрации III (с. 234). Хотелось бы еще раз подчеркнуть, что большая часть использованных команд применялась для иллюстрации дополнительных возможностей Gnuplot, а не для изображения самого графика.

Пример 1.8

```
# Команда reset (см. с. 13) устанавливает настройки в
# состояние, принятое по умолчанию. В данном случае она
# применяется для подстраховки на случай, если какие-то
# параметры были заданы и не отменены до использования
# приведенного ниже набора команд.
reset
# Таблица русификации (см. 10.1, с. 174).
set encoding cp1251
# Формат, в котором будет сохранено изображение
# (см. 12, с. 205), и размер окна изображения
# (см. 10.3.1, с. 175).
```

```
set terminal png size 640, 800
# Имя файла, где будет сохранено созданное изображение
# (см. 12.1, с. 205).
set output 'xxx.png'
# Положение легенды графика (см. 8.2, с. 151) в координатах
# системы graph (см. 10.2, с. 174) и выбор шрифта
# для надписи легенды.
set key at graph 0.6, 0.9 width -3.9 height 0.5 \
font "times, 14" box
# Стили для изображения различных участков графика
# (см. 4.2.2, с. 87).
set style line 1 lw 2 lc "green" ps 1 pi 7 pt 6
set style line 2 lw 2 lc "red" ps 1 pi 7 pt4
# Вычисление значений, требующихся ниже
# (см. 11.1, с. 193 и 11.2, с. 194).
min=1/sqrt(3)
miny=min**3-min
# Функции, определяющие различные участки графика.
# Выражение '1/0' используется в тех случаях, когда
# требуется задать неопределенное значение переменной.
# Если точка (x, y) имеет хотя бы одну неопределенную
# координату, то такая точка на изображение не выводится.
f(x)=(abs(x)<min)?x*(x-1)*(x+1):1/0
g(x)=(abs(x)>min)?x*(x-1)*(x+1):1/0
# Параметры раскрашивания добавляемых объектов
# (см. 8.6.1, с. 161).
set style fill solid 1
# Добавление двух специально выделяемых точек: мининума и
# максимума функции (см. 8.6.3, с. 163).
set object 1 circle at min,miny size 0.015 front fc "blue"
set object 2 circle at -min, -miny size .015 front fc "yellow"
set style fill solid 0.5 border lc rgb "blue"
set object 3 rectangle from -1.2, 2.2 to 1.5, 3.15 \
behind fc "yellow"
# Заголовок графика (см. 8.2, с. 151).
set title 'cubic polynomial' font "times,20"
# Дополнительные надписи (см. 8.3, с. 156).
```

```
set label 'min' at 0.7,-1 font "times,24"
set label 'max' at -0.4,1 font "times,24"
set label 'y = x^3 - x' at 1, 0.5 rotate by 66 font "Times,30"
# Дополнительные надписи, в которые автоматически
# выводятся значения переменных по заданным форматам
# (см. 10.5.1, с. 184 и 10.5.4, с. 190).
set label sprintf("min=(1/sqrt(3);-2/3sqrt(3))=\
(%4.3f,%4.3f)",min,miny) at graph 0.1,0.6 font "times,16"
set label sprintf("max=(-1/sqrt(3);2/3sqrt(3))=\
(%4.3f,%4.3f)",-min,-miny) at graph 0.1,0.55 font "times,16"
# Подписи к дополнительным делениям на осях.
set label 'min_{x}' at 0.6,-1.7 font "times,12"
set label 'max_{x}' at -0.8,-1.7 font "times,12"
# Стрелочки от надписей к описываемым ими объектам
# (см. 8.4).
set arrow 1 from 0.7,-1 to min+0.01, miny-0.05 \
head size 0.05, 30
set arrow 2 from -0.4,1 to -min+0.01, -miny+0.05 \
head size 0.05, 30
# Пунктирные линии от точек графика к осям координат
# (см. 8.4, с. 158).
set arrow 3 from min, -2 to min, miny nohead lt 0 lw 1
set arrow 4 from -1.5, miny to min, miny nohead lt 0 lw 1
set arrow 5 from -min, -2 to -min, -miny nohead lt 0 lw 1
set arrow 6 from -1.5, -miny to -min, -miny \
nohead lt 0 lw 1
# Подписи к осям (см. 5.3, с. 114).
set xlabel "X" offset graph .48,0.02 font "times,18"
set ylabel "Y" offset graph 0.08, 0.5 \
norotate font "times,18"
# Размер делений на осях (см. 5.5, с. 117).
set xtics scale 2.0, 1.
set ytics scale 3.0, 2.
# Формат подписей у делений на осях
# (см. 5.9, с. 126 и 10.5.1, с. 184).
set format x "%5.3f"
set format y "%.1e"
```

```
# Положение больших делений на осях и шрифт подписей
# (см. 5.5, с. 117).
set xtics -1.5, 0.5, 1.5 font "times,14"
set ytics -2, 1, 5 font "times,12"
# Дополнительные деления на осях.
set ytics add nomirror \
("min_{y}" miny,"max_{y}" -miny)
set xtics add nomirror (" " -min," " min)
# Частота маленьких делений на осях (см. 5.6, с. 120).
set mxtics 4
set mytics 5
# Изображение графика: задается диапазон изменения
# независимой переменной и график рисуется в виде двух
# частей. Каждая часть изображается своим стилем и в
# легенде указываются какая часть как нарисована. Для
# этого были определены две функции, каждая из которых
# задает свою часть графика.
plot [-1.5:2.] f(x) with linespoints ls 1 \
title "segment of decreasing",\
g(x) with linespoints ls 2 title "segments of increasing"
```

Замечание. Создать символ квадратного корня средствами Gnuplot пока невозможно (включая все версии до 5.0 включительно). Поэтому приходится писать аббревиатуру '**sqrt**'. Если для каких-либо целей требуется создавать надписи, используя все возможности системы LaTeX, то нужно использовать возможность вызова команд Gnuplot из этой системы (см. 12.2.6, с. 212).

1.2.2 Функция двух переменных

Рассмотрим процесс построения графика функции двух переменных, заданной формулой

$$\mathbf{z(x,y)} = \mathbf{5} * \mathbf{cos(x)} * \mathbf{cos(y)} * \mathbf{exp}(-\mathbf{0.1} * (|\mathbf{x}| + |\mathbf{y}|)).$$

На его примере покажем возможности Gnuplot для построения трехмерных изображений. Перечислим те возможности,

которые были использованы при создании цветной иллюстрации IV (с. 235):

1) задание углов обзора;

2) задание точки пересечения плоскости xy и оси z;

3) задание палитры по rgb-формуле;

4) задание размера и местоположения цветовой схемы;

5) задание местоположения и стиля легенды;

6) задания режима стирания невидимых линий;

7) задание линий уровня на поверхности;

8) задание дополнительных надписей и стрелочек от них к нужным местам графика;

9) задание числа изолиний;

10) задание изображаемой границы изображения;

11) подписи к делениям оси y в формате чисел кратных числу π;

12) смена автоматической частоты делений по оси y.

Цветная иллюстрация IV была создана с помощью следующих команд.

Пример 1.9

```
reset
set terminal png size 640, 800
set output 'yyy.png'
# Углы обзора установить 37^o и 24^o,
# масштаб всего графика — 1, по оси z — 1.5 (см. 6.1, с. 128).
set view 37,24,1,1.5
# Полоскость xy провести при z=-4.5 (см. 6.4, с. 132).
set xyplane at -4.5
# Палитра зелено-красно-фиолетовая (см. 7.5.2, с. 144).
set palette rgbformulae 3, 11, 6
# Цветовая схема (см. 7.3, с. 137).
set colorbox vert user origin 0.92, 0.05 size 0.04, 0.9
# Положение легенды (см. 8.2, с. 151).
set key top center font "times, 14"
# Режим стирания невидимых линий (см. 6.3, с. 130).
set hidden3d
```

```
# Линии уровня (см. 2.4, с. 56).
set contour surface
set cntrparam levels discrete 2
unset clabel
# Надпись для изолинии (см. 8.3, с. 156) и
# ее стрелки (см. 8.4, с. 158).
set label 2 'contour z=2' at 3.5,0,6.5 font "times,14"
set arrow 2 from 6.5,0.,6.7 to 1,6.,2.2 head size 0.2, 30
set arrow 3 front from 6.5,0.,6.7 to -2.2,3.5,2. \
head size 0.2, 30
set arrow 4 front from 6.5,0.,6.7 to 3.,2.7,2. \
head size 0.2, 30
# Функция, задающая поверхность (см. 11.2, с. 194).
f(x,y)=5*cos(x)*cos(y)*exp(-0.1*(abs(x)+abs(y)))
# Число изолиний по каждой из независимых переменных
# (см. 6.2, с. 129).
set isosamples 200, 200
# Изображаемые границы (см. 10.3.2, с. 176).
set border 1+2+4+8+16
# Название графика (см. 8.2, с. 151).
set title 'surface' font 'times,20'
# Название осей (см. 5.3, с. 114).
set xlabel "X"
set ylabel "Y"
set zlabel "Z"
# Формат по оси y (см. 10.5.1, с. 184 и 5.5, с. 117).
set format y "%.1P pi"
set ytics 1.57077962
# Надпись "max" (см. 8.3, с. 156) и ее стрелка (см. 8.4, с. 158).
set label 1 'max=(0,0,5)' at -7,-2.4,6.2 font "times,14"
set arrow 1 from -2.8,-2.6,6.5 to 0,0,5 head size 0.2, 30
# Изобразить поверхность с помощью шаблона pm3d,
# задав легенду.
splot [-10:10] [-10:10] f(x,y) with pm3d\
title "f(x,y)=5*cos(x)*cos(y)*exp(-0.1*(abs(x)+abs(y)))"
```

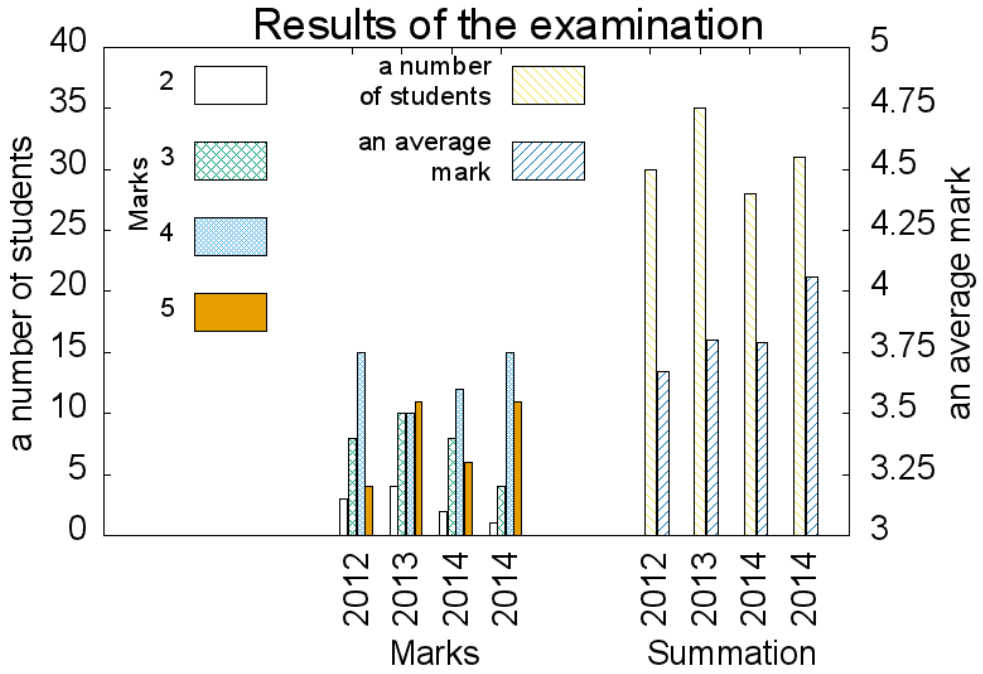


Рис. 7: Две гистограммы в одних осях

1.2.3 Две гистограммы на одном графике

Приведем пример команд, создающих две гистограммы на одном графике. Поскольку многие команды были использованы и прокомментированы в примере 1.2, то ниже будем объяснять назначение только не использовавшихся ранее операторов.

Пример 1.10

```
set tmargin 2
set bmargin 6
# Опция rotate без указания угла означает
# поворот подписей к оси на 90^o.
set xtics rotate offset 1,0 font ",20"
set ytics font ",20"
set xrange [-4:]
set yrange [0:40]
# В строящейся диаграмме вводятся две оси ординат: левая Y
# и правая Y2. По левой оси откладывается количество
```

```
# студентов, а по правой — средний бал.
set y2range [3:5]
set y2tics 3, 0.25, 5 font ",20"
# Комментарии к осям Y и Y2.
set ylabel "a number of students" font ",20"
set y2label "an average mark" offset 1.5,0 font ",20"
set boxwidth 0.9 relative
set style histogram cluster
set style data histograms
set style fill solid 1.0 pattern border lt -1
set key spacing 2.5
set label 1 "Results of the examination" at -1,41 font ",26"
set label 2 "Marks" at -3.2,25 rotate font ",16"
set key title at 6,40 font ",16"
# Шаблон newhistogram используется при построении двух и
# более гистограмм на одном графике.
plot newhistogram "Marks" offset 2.75,-2.5 font ",20",\
'diag.dat' using 3 t "2",' ' using 4 t "3", \
' ' using 5 t "4",' ' using 6:xtic(1) t "5",\
newhistogram "Summation" offset 2.75,-2.5 font ",20",\
'diag.dat' using 2:xtic(1) t "a number\n of students",\
' ' using 7 axes x1y2 t "an average\n mark"
```

Результат выполнения команд примера 1.10 изображен на рисунке 7. Черно-белый вариант гистограммы и размеры рисунка делают диаграмму не совсем качественной: многие детали недостаточно четко различимы. Эти недостатки исчезают либо при увеличении размера рисунка, либо при использовании цветов при раскраске.

Безусловно, с помощью Gnuplot можно создавать гораздо более сложные и интересные изображения. Их можно найти на домашней странице Gnuplot по адресу

www.gnuplot.info

.

2 Первые шаги знакомства с Gnuplot

2.1 Файлы с дискретными данными

Дискретные данные, требующиеся для построения графиков, можно хранить в файлах. Опишем такой способ записи данных в файл, чтобы Gnuplot без дополнительных ключей смог по ним построить график.

Начнем с функции одной переменной. В этом случае каждая строка файла с данными должна содержить информацию о координатах одной точки. По умолчанию числа в строке разделены пробелами и порядковый номер числа определяет номер столбца, которому это число принадлежит. Номера столбцов начинаются с единицы.

Если столбец один, то число, стоящее в каждой строке, задает значение функции, а номер строки определяет ее аргумент. Если количество столбцов больше или равно двум, то координаты точек графика определяются по правилу: число из первого столбца каждой строки является абсциссой точки, а число из второго столбца задает ординату точки. Это правило действует, когда явно не указано какое число из строки считать аргументом функции, а какое значением. Явное задание правила использования чисел строки из файла данных описано в разделе 3.2 (с. 69).

В случае построения графика функции двух переменных

$$\mathbf{z} = \mathbf{f}(\mathbf{x}, \mathbf{y})$$

простейший файл данных содержит три столбца чисел. Первое число каждой строки трактуется как значение $\mathbf{x}$, второе как $\mathbf{y}$, а третье как $\mathbf{z}$, если явно не указано другое правило. Все строки разделены на группы. Группы отделяются друг от друга пустой строкой. В каждой группе координата $\mathbf{x}$ должна быть постоянна.

Комментарии в Gnuplot возможны не только в командных строках, но и в файлах с данными. По умолчанию комментарии в строке начинаются после символа #. Комментарии могут

быть использованы для создания различных подписей в легенде к графику.

Другие способы хранения данных в файле и различные нестандартные ситуации описаны в разделе 3.1 (с. 63).

2.2 Изображение двумерных графиков

2.2.1 Синтаксис команды

Изображение графиков функций на плоскости выполняется командой **plot**. Ее формат имеет вид

plot {<ranges>} <plot-element> {, <plot-element>, <plot-element>}

Каждый ключ <**plot-element**> содержит в себе информацию об одной функции и способе ее изображения

plot-element:
{<iteration>} <definition> |
{sampling-range} <function> | <data source>
{axes <axes>} {<title-spec>} {with <style>}

Задание различных ключей этой команды позволяет получать всевозможные картинки с изображениями графиков. Коротко опишем предназначение наиболее часто используемых ключей.

Ключ <**ranges**> задает диапазоны изменения переменных по осям (см. 5.1, с. 110). Заметим, что выбор по умолчанию не всегда позволяет получить желаемое изображение.

Ключ <**iteration**> чаще всего используется в командных файлах для организации циклического перебора входных данных (см. 11.5.2, с. 199).

Ключ <**function**> | **{sampling-range}** <**function**> | <**data source**> задает либо аналитическую формулу функции, либо файл с данными, которые применяются для построения.

При задании функции дискретными данными, которые считываются из файла, часто требуются дополнительные ключи

для фильтрации данных. Эти ключи определяют правило использования чисел, содержащихся в файле. Дело в том, что часто возникает ситуация, когда источником данных служит файл, содержащий помимо координат точек графика еще другую информацию. Gnuplot предоставляет ряд возможностей указать каким образом следует фильтровать считываемые данные. Для этой цели у команды **plot** предусмотрены следующие опции

plot '<file_name>' {binary <binary list>}
{{nonuniform} matrix}
{index <index_list> | index "<name>"}
{every <every_list>}
{skip <number-of-lines>}
{using <using_list>}
{smooth <option>}
{volatile} {noautoscale}

Порядок использования этих ключей описан в отдельных параграфах главы 3.

Ключ <**axes**> дает возможность задавать оси координат. О том, как можно подписывать и размечать их, смотри главу 5.

Ключ <**title-spec**> определяет содержание комментариев к графикам, которые печатаются в легенде (см. 8.1, с. 151). О задании всевозможных параметров самой легенды написано в разделе 8.2 (с. 151).

По умолчанию функция, определенная формулой, рисуется в виде непрерывной кривой, а графиком функции, заданной с помощью дискретных данных, будет просто набор точек на координатной плоскости. Ключ <**with**> определяет каким образом будет изображена кривая графика, другими словами задает ее шаблон (см. 4.2, с. 86).

2.2.2 Декартовы координаты

Наиболее простым примером является изображение функции, заданной аналитической формулой, в декартовых координатах

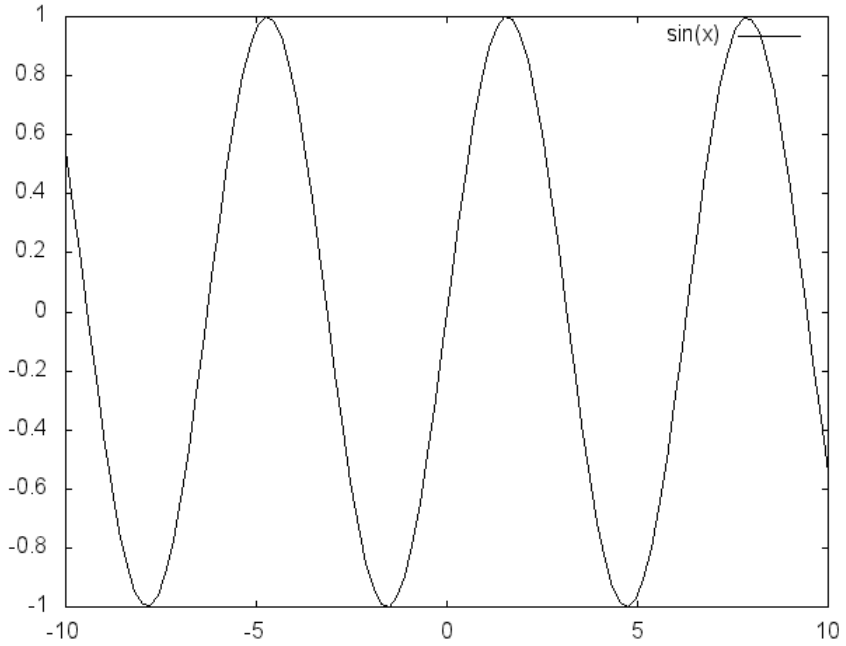


Рис. 8: график синуса

без каких либо дополнительных условий на получаемый график. В этом случае Gnuplon сам определяет в каких диапазонах переменных имеет смысл рисовать график, какие надписи делать и т.д.

Пример 2.1

plot sin(x)

Результатом исполнения этой команды является график, изображенный на рисунке 8. Обратим внимание на то, что все элементы оформления изображения были созданы Gnuplot автоматически. При данном масштабе картинки все надписи читаемы, однако легенда графика, располагаемая по умолчанию в правом верхнем углу, пересекается с изображением графика. При необходимости уменьшить размер картинки размер шрифта подписей и легенды окажется уже слишком маленьким. Для устранения отмеченных недостатков в Gnuplot есть опции для задания места вывода легенды и других элементов рисунка

(см. 10.2, с. 174) и выбора типа и размера шрифта (см. 10.4.2, с. 183).

Gnuplot может рисовать несколько графиков в одних осях. Для этого аналитические формулы функций, графики которых требуется изобразить, следует перечислить через запятую. Заметим, что различные графики по умолчанию рисуются разными цветами. Примером такой команды может служить

plot sin(x), cos(x), tan(x)

Графическое изображение, созданное этой командой, содержит три графика: sin(x) изображен красным цветом, cos(x) — зеленым и tg(x) — синим. Заметим, что график функции tg(x) будет неудачным при использовании настроек по умолчанию (см. 4.3, с. 105).

Иногда возникает необходимость нарисовать несколько графиков одной и той же функции. В этом случае удобно сначала определить эту функцию, а уже потом рисовать ее графики. Например,

p(x) = sin(x) * exp(−x * *2)
plot p(x), p(x + 1)

2.2.3 Дискретные данные

Другим важным способом задания функции служит файл, в котором хранятся координаты точек графика. Например, пусть требуется нарисовать график функции, информация о которой хранится в файле ro.dat в виде двух колонок чисел. В каждой строчке записана абсцисса и ордината точки графика. В этом случае график функции рисуется командой **plot "ro.dat"**. Имя файла с данными может заключаться как в простые (одинарные), так и в двойные кавычки.

Пример 2.2

```
set key font ',18'
set xtics font ',16'
set ytics font ',16'
plot sin(x), "ro.dat"
```

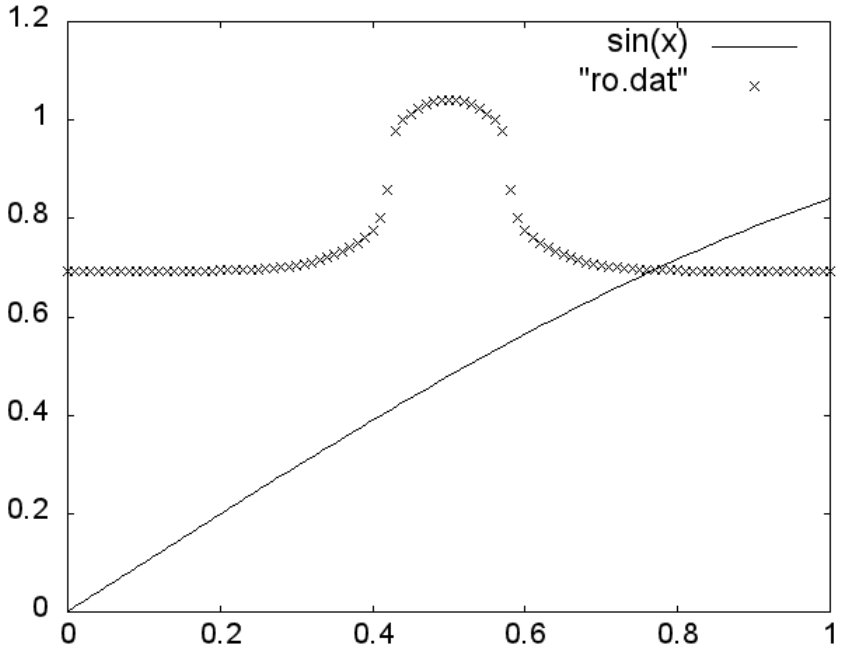


Рис. 9: графики синуса и функции дискретного аргумента

На рисунке 9 изображены графики, построенные командами из примера 2.2. Отметим, что без указания способа изображения (шаблона) графика Gnuplot изображает график дискретной функции набором точек, координаты которых он берет из файла данных. Размер шрифта для надписей был задан более крупным чем при автоматическом выборе.

Границы изменения абсциссы графиков были по умолчанию определены по данным, находящимся в файле ro.dat.

2.2.4 Параметрическое задание

Замкнутые кривые проще всего изображать, используя их параметрическое представление. Классическим примером такой кривой является окружность. Приведем команды, позволяющие нарисовать единичную окружность с центром в начале координат.

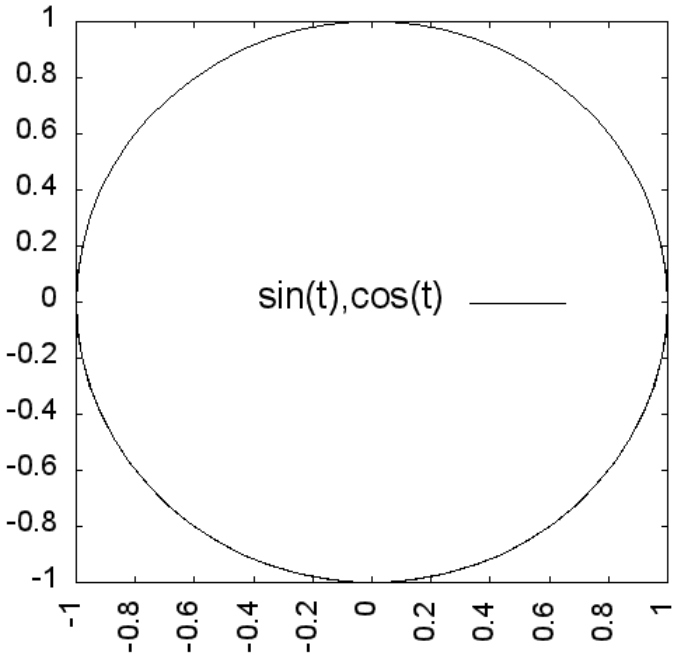


Рис. 10: окружность

Пример 2.3

```
# Установить одинаковый масштаб по осям графика.
set size ratio 1
# Установить параметрический способ задания функции.
set parametric
set key center font ',20' # легенду разместить по центру
set xtics rotate font ',16'
set ytics font ',16'
plot sin(t),cos(t)
```

Результат работы этих команд изображен на рисунке 10. Заметим, что при отсутствии первой команды оружность на получаемом изображении принимает форму эллипса. Это происходит вследствие того, что по умолчанию на осях выбираются разные масштабы. Подписи к делениям оси X были повернуты на 90^o включением ключа **rotate** в команду **set xtics**.

2.2.5 Полярные координаты

Полярные координаты достаточно часто используются для изображения различных кривых, заданных выражением $\mathbf{r} = \mathbf{f}(\mathbf{t})$, на плоскости. Режим изображения в полярных координатах устанавливается и отменяется следующими командами

set polar
unset polar

Независимой переменной в этом случае является угол, для задания которого по умолчанию используется переменная t. Пределы ее изменения устанавливаются от 0 до 2π или от 0^o до 360^o, если для углов была выбрана градусная мера измерения (см. 10.6, с. 191). Пределы изменения полярного угла можно установить явно с помощью опции **trange**, а зависимой переменной (по умолчанию r) — **rrange** (см. 5.1, с. 110). Диапазон изменения угла задает область определения функции, а диапазон по r определяет какую часть графика требуется изобразить.

Приведем команды, позволяющие нарисовать кривую

$$\mathbf{f(t)} = \mathbf{t} \cdot \mathbf{sin(t)}.$$

Пример 2.4

```
set polar
set trange [-2*pi:2*pi]
set rrange [0:3]
set top center font ',20'
set xtics font ',16'
set ytics font ',16'
set rtics font ',16'
plot t*sin(t)
```

Результат работы этих команд изображен на рисунке 11.

2.3 Изображение трехмерных графиков

2.3.1 Синтаксис команды

Изображение графиков функций с двумя независимыми переменными выполняется командой **splot**. Эта команда очень по-

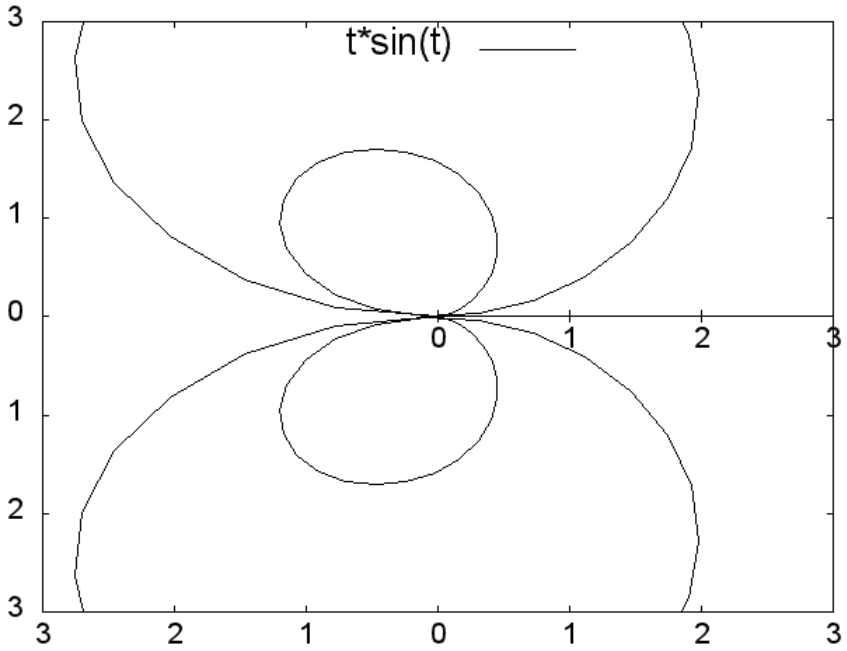


Рис. 11: кривая $r = t \cdot sin(t)$

хожа на команду **plot**. Ее формат имеет вид

splot {<ranges>} {<iteration>}
{<function>|{{<file name>|<data block name>}
{datafile-modifiers}}{<title-spec>}{with <style>}
{, {definitions{,}} <function> ...}

Ее опции во многом совпадают с опциями команды **plot**.

Команда **splot** строит графические изображения, используя либо аналитическую формулу, задающую функцию двух переменных, либо данные о точках графика, хранящихся в файле данных или в блоке данных. Синтаксис опции **ranges** зависит от способа задания функции. В случае непараметрического задания составляющие ее ключи указываются с следующем порядке

splot [<xrange>] [<yrange>] [<zrange>] ...

В случае параметрического задания порядок ключей такой

splot [<urange>] [<vrange>] [<yrange>] [<zrange>] ...

По умолчанию **splot** рисует координатную плоскость **XY** ниже всех точек графика. Ее положение можно задать явно командой **set xyplane** (см. 6.4, с. 132).

Ориентация графика устанавливается командой **set view** (см. 6.1, с. 128). Эта команда позволяет задавать углы поворота осей **X** и **Z**, а также масштабы, используемые для всего графика и по отдельно оси **Z**. Эти параметры выводятся в окне, где рисуется график поверхности. В интерактивном режиме можно поворачивать изображение графика стрелками **"вверх", "вниз"** и **"вправо", "влево"**, добиваясь самого удобного ракурса. Выбрав таким образом нужный ракурс, можно понять с какими значениями ключа **view** нужно создавать изображение.

Шаблоны графиков могут быть выбраны из стилей **lines, points, linespoints, dots** и **impulses** (см. 4, с. 86).

Режим стирания невидимых линий устанавливается с помощью команды **set hidden3d** (см. 6.3, с. 130).

2.3.2 Декартовы координаты

Разбор вариантов изображения поверхностей начнем с примера изображения эллиптического параболоида в декартовых координатах.

Пример 2.5

```
set key font ',20'
# задать подпись к оси абсцисс в виде X
set xlabel "X" font ',16'
# задать подпись к оси ординат в виде Y
set ylabel "Y" font ',16'
# задать подпись к третьей оси в виде Z
set zlabel "Z" font ',16'
# задать деления на оси Z
set ztics (10,20,30) font ',16'
set xtics offset 0,-0.5 font ',16'
set ytics offset 0.5, -0.25 font ',16'
p=2; q=4
splot x*x/(2*p)+y*y/(2*q) with lines
```

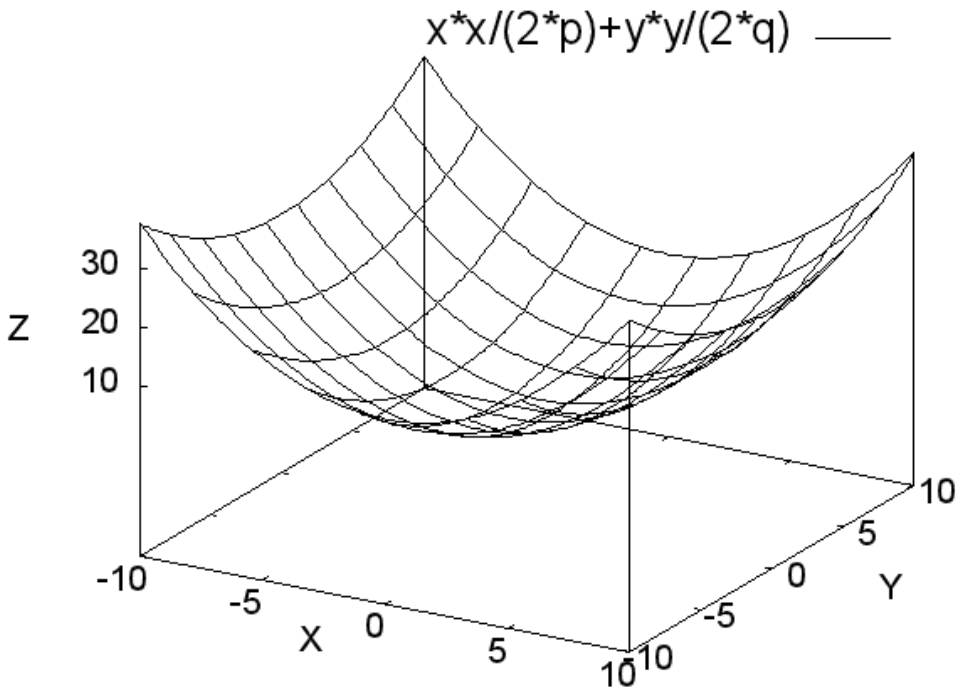


Рис. 12: Эллиптический параболоид

Результат выполнения команд примера 2.5 приведен на рисунке 12.

Замечание. В данном примере деления по оси **Z** были установлены явно, т.к. по умолчанию они были бы нанесены слишком часто. Результат задания меток по оси **Z** по умолчанию можно увидеть на рисунке 36.

2.3.3 Сферические координаты

Часто поверхности в трехмерном пространстве задаются параметрически. Одним из наиболее традиционных и часто встречающихся способов является случай сферических координат. Приведем пример их использования, построив сферу.

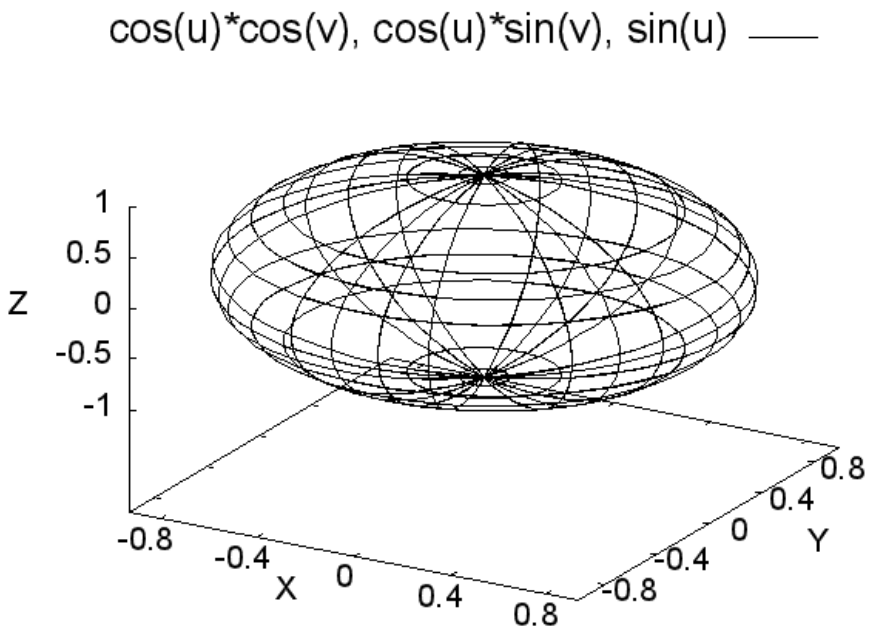


Рис. 13: Сфера

Пример 2.6

set mapping spherical # см. 6.5, с. 133
set parametric # задать параметрический режим
set key font ',20'
set xlabel "X" font ',16'
set ylabel "Y" font ',16'
set zlabel "Z" font ',16'
set ztics font ',16'
set xtics offset 0,-0.5 font ',16'
set ytics offset 1, -0.25 font ',16'
set xtics (-0.8,-0.4,0,0.4,0.8) # задать деления на оси X
set ytics (-0.8,-0.4,0,0.4,0.8) # задать деления на оси Y
set ztics (-1,-0.5,0,0.5,1) # задать деления на оси Z
splot cos(u)*cos(v), cos(u)*sin(v), sin(u) with lines

Результат команд из примера 2.6 приведен на рисунке 13.

2.3.4 Цилиндрические координаты

Еще одним часто используемым параметрическим способом задания функции двух переменных является случай цилиндрических координат. Приведем пример их использования, построив цилиндр.

Пример 2.7

```
set mapping cylindrical # см. 6.5, с. 133
set parametric # параметрический режим задания функции
set key font ',20'
set ylabel "X" 4font ',16'
set ylabel "Y" font ',16'
set ylabel "Z" font ',16'
set xtics 0.5 offset 0,-0.5 font ',16'
set ytics 0.5 offset 1, -0.25 font ',16'
set ztics font ',16'
splot cos(u), sin(u), v with lines
```

Результат выполнения команд из примера 2.7 приведен на рисунке 14.

2.3.5 Непрямоугольные области

Графики функций двух переменных, заданных аналитической формулой, рисуются над прямоугольной областью изменения независимых переменных. Эта область выбирается либо явным заданием параметров **xrange** и **yrange**, либо по умолчанию определяется Gnuplot. В случае, когда необходимо нарисовать поверхность над областью более сложной формы, можно поступить следующим образом. Для примера нарисуем функцию, равную 1 на единичном круге и неопределенную вне его. Для этого следует сначала задать саму функцию, воспользовавшись тернарным оператором (см. 11.1, с. 193).

Пример 2.8

```
set key font ',20'
set xlabel "X" font ',16'
```

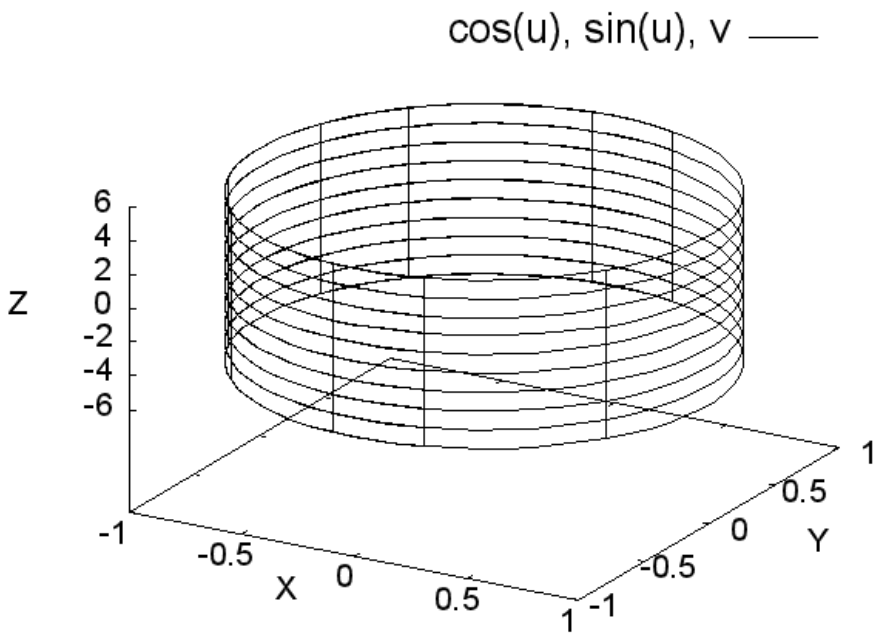


Рис. 14: Цилиндр

```
set ylabel "Y" font ',16'
set zlabel "Z" font ',16'
set xtics offset 0,-0.5 font ',16'
set ytics offset 1, -0.25 font ',16'
set ztics font ',16'
f(x,y)=(x*x+y*y<=1.)?(1.):(1/0)
splot [-1.:1.] [-1.:1.] f(x,y)
```

Результат выполнения команд из примера 2.8 приведен на рисунке 15.

В приведеной выше команде splot явно указаны границы изменения независимых переменных, поскольку по умолчанию Gnuplot определяет их слишком большими. Это приводит к тому, что сетка, по которой строится поверхность, выбирается слишком редкой и ни одного узла не попадает в область (единичный круг), где значения функции определены.

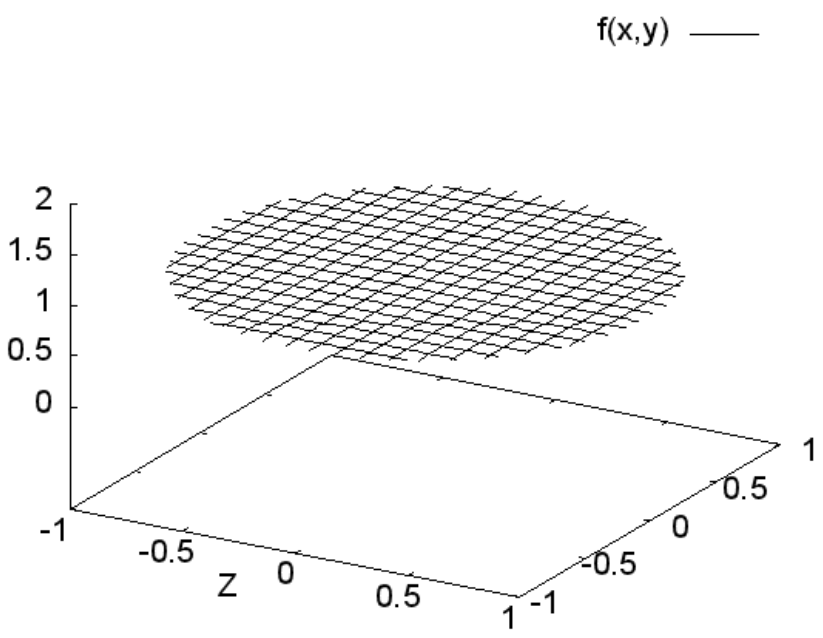


Рис. 15: график функции, заданной только на единичном круге

2.3.6 Дискретные данные

Графики функций двух переменных сразу рисуются лишь над областью, где находятся точки, координаты которых были заданы в файле с данными. По умолчанию, как и в двумерном случае, наносятся только точки, у которых определены все три координаты. Для того, чтобы график был больше похож на поверхность, применяют ключ **with** со значениями **lines** или **linespoints**.

Особенность изображения, на которую следует обратить внимание, состоит в том, что при наличии равного количества данных во всех блоках, поверхность рисуется в виде сетки. Сами такие данные тоже называются сеткой: **grid data**. Например, результатом обработки командой

splot 'spl_dis.dat' with lines title 'grid'

файла spl_dis.dat, содержащего следующие строки

```
#    x    y    z
```

```
0    0    0
0    1    1
0    2    4
0    3    9
0    4    16
0    5    25

1    0    1
1    1    2
1    2    5
1    3    10
1    4    17
1    5    26      # !!!!!

2    0    4
2    1    5
2    2    8
2    3    13
2    4    20       # !!!!!
2    5    29      # !!!!!

3    0    9
3    1    10
3    2    13
3    3    18      # !!!!!
3    4    25      # !!!!!
3    5    34      # !!!!!
```

будет рисунок 16.

Если же из файла spl_dis.dat удалить строки, отмеченные комментарием "!!!!!" (т.е. сделать количество точек в блоках разным), то вместо сетки будут нарисованы 3D линии. Результат приведен на рисунке 17.

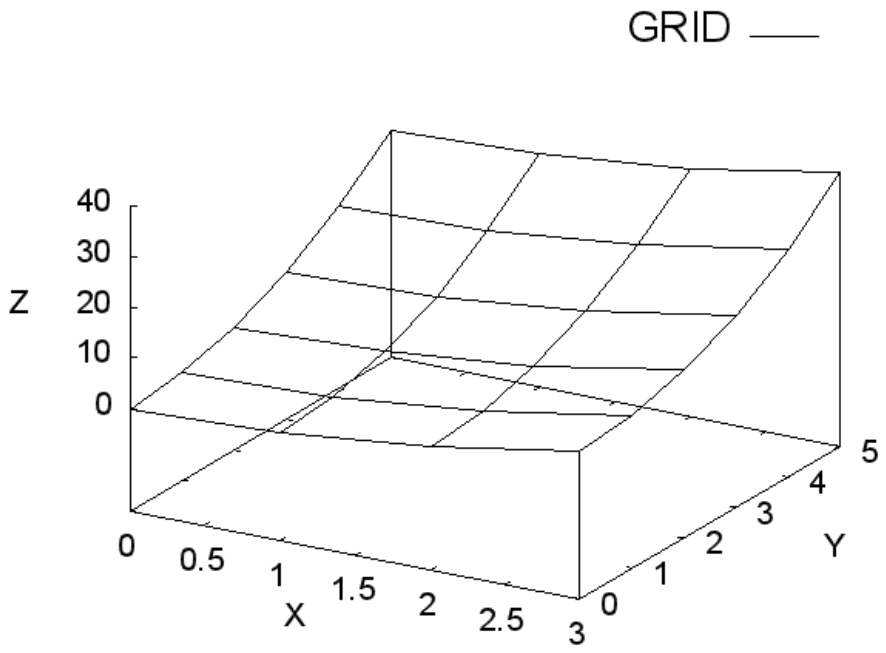


Рис. 16: поверхность, заданная сеткой данных

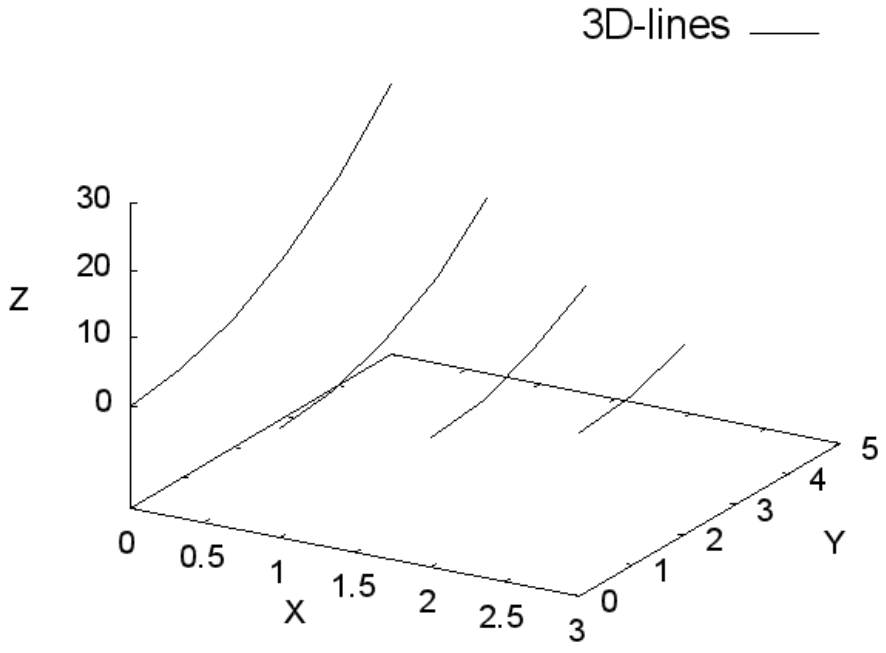


Рис. 17: 3D линии поверхности

2.4 Изображение контуров

2.4.1 Способ изображения

Контурами называют линии уровня поверхности $\mathbf{z} = \mathbf{f}(\mathbf{x}, \mathbf{y})$, задаваемые в трехмерном пространстве уравнениями $\mathbf{z_k} = \mathbf{f}(\mathbf{x}, \mathbf{y})$, где $\mathbf{z_k}$ — набор констант. Для изображения контуров поверхности нужно предварительно выполнить команду

set contour {base|surface|both}

Указание параметра **base** означает, что будут нарисованы проекции контуров на плоскости XY, параметра **surface** — на самой поверхности, а **both** — и то, и другое одновременно.

Для отмены режима рисования контуров предусмотрена команда

unset contour

Выбор величин $\mathbf{z_k}$ и аппроксимационных методов построения кривых, изображающих контуры, осуществляется командой

set cntrparam { { linear | cubicspline | bspline |
points <n> | order <n> levels { auto {<n>} | <n>
| discrete <z1> {,<z2>{,<z3> . . .} }
incremental <start>, <incr> {,<end> } } } }

Задание ключа **linear** означает, что контур будет изображен в виде ломаной. Ключ **cubicspline** позволяет строить контуры в виде более гладких кривых, но этот способ аппроксимации может давать излишнюю волнистость этим кривым. Волнистости можно избежать, задав ключ **bspline**. Однако в этом случае кривые контуров будут лишь приблизительно проходить через сетку точек, по которым они строятся. Ключ **points <n>** задает количество точек сетки (см. 4.3, с. 105). Ключ **order <n>** определяет степень используемых многочленов, если применяется метод **bspline**. Чем выше степень, тем более гладкой получается кривая, но тем больше погрешность совпадения в узлах сетки. Выбор значений параметра **<n>** допукается из множества целых чисел от 2 (линейный случай) до 10.

Число контуров определяется опцией **levels**. Автоматическое задание задается ключом **auto**. Этот ключ действует по умолчанию. Если задан параметр **n**, то число создаваемых контуров приблизительно равно этому значению. Значения $\mathbf{z_k}$ перечисляются явно, если выбран ключ **discrete**. При выборе такого ключа все задания параметра <**n**> игнорируются. Задание ключа **incremental** означает построение контуров, определяемых значениями $\mathbf{z_k}$ от **start** до **end** с шагом **increment**. В случае, если ось **Z** имеет логарифмический масштаб, то под шагом понимается степень основания логарифма.

Если команда **set cntrparam** выполнена без указания опций, то их значения по умолчанию полагаются равными

linear, 5 points, order 4, 5 auto levels

Для просмотра заданных параметров изображения контуров предусмотрена команда

show contour

Примеры.
1. **set cntrparam bspline** # задать способ построения контуров **bspline**.
2. **set cntrparam points 7** # задать число точек сетки равным 7.
3. **set cntrparam order 10** # задать степень многочленов в **bspline** равным 10.
4. **set cntrparam levels discrete .1,1/exp(1),.9** # задать три линии уровня, соответствующие константам 0.1; 0.37; 0.9.
5. **set cntrparam levels 0,1,4** # задать линии уровня с константами от 0 до 4 с шагом 1.
6. **set cntrparam levels 10** # задать число контуров равным 10.
7. **set cntrparam incremental 100, 50** # задать контуры, начиная с константы 100 и шагом выбора констант равным 50.

2.4.2 Подписи к контурам

До версии 5.0 в Gnuplot для создания подписей для контуров использовалась команда

set clabel { '<format>' }

По умолчанию формат задается **%8.3**, что означает три знака после десятичной запятой при общем числе знаков равном 8.

Эта команда всю информацию об изображенных контурах выводит в виде легенды, где напротив каждой линии ставится ее параметр $\mathbf{z_k}$. Поскольку все линии рисуются разными цветами, то на цветных изображениях такая легенда позволяет определить какие линии на графике какому параметру соответствуют. Но, если изображение черно-белое, то такая легенда не помогает ответить на этот вопрос. Гораздо удобнее, когда на самом графике подписаны все линии. Эта задача решена, начиная с версии 5.0. Для этого в команду **splot**, рисующую контуры, добавляется вторая часть: задание на изображение подписи к каждой линии, содержащей величину $\mathbf{z_k}$. Например,

splot f(x,y), f(x,y) with labels boxed title " "

Для задания типа и размера шрифта, которым выводятся подписи к контурам, используется команда

set cntrlabel {format "format"} {font "font"}

Можно задавать размещение подписей на контурах. Как любая линия контур рисуется, используя определенное число узлов (см. 2.4.1, с. 56). Это число определяет количество частей, из которых состоит линия контура. Параметр <**int_s**> задает номер части линии контура, на котором будет размещена первая подпись, а <**int_i**> — шаг цикла по номерам, определяющий на каких частях будут также размещены подписи

set cntrlabel {start <int_s>} {interval <int_i>}

Задание отрицательного значения <**int**> означает, что на каждой линии будет выведена лишь одна подпись. Однако, это правило нарушается, если числа, определяемые командами **set samples** или **set isosamples**, являются достаточно большими или длина изолинии велика.

В версии 5.0 появилась команда, после выполнения которой все линии уровня рисуются одним шаблоном (в частности, цветом)

set cntrlabel onecolor

2.4.3 Контуры параболоида

Приведем в качестве простейшего примера набор команд, который изображает проекции контуров парабалоида $\mathbf{z = x^2 + y^2}$ на плоскость **XY**.

Пример 2.9

```
set xlabel "x" font ',16'
set ylabel "y" font ',16'
set xtics font ',16'
set ytics font ',16'
# Легенду разметить на полях изображения.
set key outside font ',16'
set view 0, 0 # Задать углы обзора 0, 0.
set view equal xy # Задать равный масштаб по осям X и Y.
set contour base # Контуры изображать на полоскости XY.
# Рисовать следующие линии уровня.
set cntrparam levels discrete 0.8, 0.6, 0.4, 0.2, 0.1
unset surface # Не рисовать поверхность.
unset ztics # Не выводить подписи к делениям оси Z.
# Задать шрифт для подписей контуров.
set cntrlabel font "Times, 8"
# Задать порядок вывода подписей.
set cntrlabel start 5 interval -1
f(x,y)=x*x+y*y # Формула изображаемой функции.
splot [-1.:1.] [-1.:1.] f(x,y), f(x,y) with labels boxed title " "
```

Замечание Саму поверхность Gnuplot не рисует, если выполнить команду

unset surface

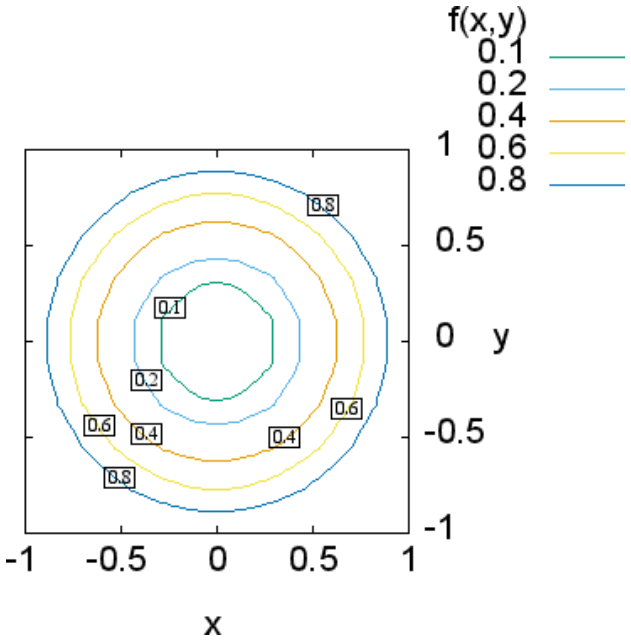


Рис. 18: контуры парабалоида

После выполнения этих команд на экране получаем изображение, которое приведено на рисунке 18. Заметим, что команда **unset ztics** была применена для того, чтобы убрать на рисунке деления на оси **Z**, которые автоматически не исчезают после выполения команды **unset surface**.

2.5 Интерактивный режим

Для любителей работать в интерактивном режиме безусловно полезными являются команды **replot** и **refresh**. Обе они выполняют последнюю из использованных команд **plot** или **splot**.

Команда **replot** без аргументов повторяет соответствующую команду, используя **все!!!** изменения и дополнения в настройках (опциях), которые были сделаны пользователем после предыдущего ее применения. Например, пусть изначально был создан файл a.dat. После чего в файле были сделаны изменения:

a.dat		new a.dat	
1	1	1	1

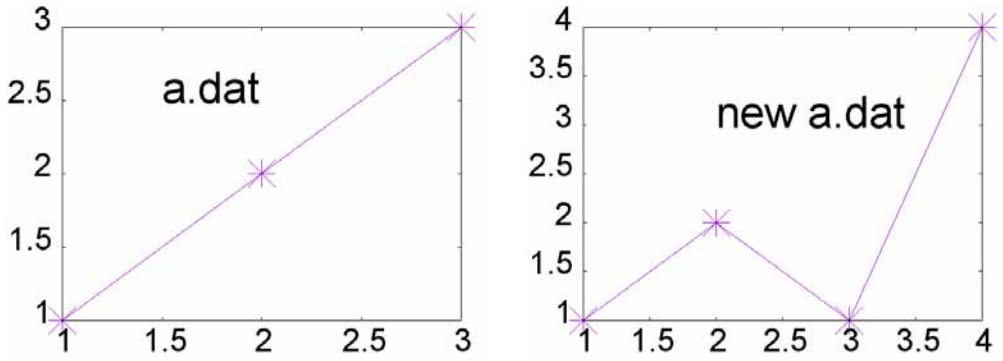


Рис. 19: результат работы команды replot

2	2	2	2
3	3	3	1
		4	4

Пример 2.10

```
set lmargin 7.5
set tmargin 1.5
set bmargin 3.
set xtics offset 0,-1 font ",32"
set ytics font ",32"
set label 1 "a.dat" at 1.5,2.5 font ",44"
plot 'a.dat' w lp pt 3 ps 6 notitle
# меняем текст и положение надписи и меняем
# содержимое файла a.dat
set label 1 "new a.dat" at 2,3
replot
```

В результате выполнения команд примера 2.10 на экране последовательно в одном окне будут нарисованы два изображения. Первой возникнет картинка, изображенная слева на рисунке 19, а затем ее сменит та, что справа.

Обратим внимание, что все настройки на втором изображении остались прежние за исключением тех, что были изменены между использованием команд. Данные, которые также были изменены, были использованы в новой версии. Этими свойствами команды **replot** удобно пользоваться при подборе наиболее

устраивающих автора параметров изображения (например, размер шрифтов, положение надписей и стрелок и т.д.) или же когда удалось добиться в процессе интерактивной работы нужного изображения и захотелось его сохранить. В этом случае достаточно изменить вывод на сохранение в файл и выполнить команду **replot**.

Другой очевидный способ использования **replot** — создание анимации. Предположим есть программа, позволяющая достаточно быстро вычислять некоторую функцию, меняющуюся со временем. Результаты работы программы записываются в файл, который используется для построения графического изображения этой функции. Если согласовать запись в файл новых данных и работу Gnuplot, то можно получить меняющуюся со временем картинку без хранения на компьютере данных в различные моменты времени.

Есть случай, когда использовать **replot** неудобно: если последняя команда **plot** использовала данные вводимые с экрана

plot '-' ; ... ;

то следующая за ней команда **replot** будет снова ждать ввода этих данных.

Для того, чтобы этого избежать, в арсенале Gnuplot есть команда **refresh**. Она строит график по данным, которые были использованы при последнем вызове команд **plot** или **splot**. Поэтому, если в примере 2.10 поменять команду **replot** на **refresh**, мы снова получим левый рисунок, но надпись все же будет 'new a.dat' и она будет расположена в новом месте. Команда **refresh** "откатывает" назад все преобразования графика, выполенные с помощью мыши после его построения.

По умолчанию Gnuplot обрабатывает файлы той директории, в которой он был запущен. Для смены директории предусмотрена команда

cd '<directory-name>'

Название директории должно быть заключено в одинарные кавычки.

3 Использование дискретных данных

3.1 Организация файлов с данными

3.1.1 Построчный формат

По умолчанию Gnuplot считает, что данные в файлах хранятся построчно, т.е. каждая строка содержит информацию про одну точку графика. Каждое число или набор символов из строки образуют поле данных. Различные поля по умолчанию разделяются пробелами. Порядковый номер поля (нумерация начинается с единицы) соответствует номеру колонки данных. Таким образом, все данные делятся на колонки (столбцы), номера которых используются опцией **using** для определения того, какие поля что означают (см. 3.2, с. 69).

В дополнение к фактическим колонкам файла данных Gnuplot допускает использование данных из трех псевдостолбцов (**"pseudocolumns"**). Эти столбцы имеют соответственно номера 0, -1 и -2. Первый (с номером 0) содержит последовательные номера всех строчек файла. Номера начинаются с нуля. Второй (с номером -1) также последовательно нумерует строчки, начиная отсчет с нуля, но сбрасывает счетчик на ноль, встречая пустую строку. Это требуется для трехмерных графиков с двумя независимыми переменными, где все точки делятся на группы (блоки). Каждая группа отделяется от другой пустой строкой. Номер точки в группе, которой она принадлежит, как раз и задает второй "псевдостолбец". Третий (с номером -2) является номером группы, которой принадлежит данная точка. Обращение к текущему значению этих "псевдостолбцов" осуществляется также как и к значениям обычных столбцов: **column(0)**, **column(-1)** или **column(-2)**. К нулевому столбцу есть возможность обращаться **$0**.

Обратим внимание на то, как реагирует Gnuplot на две пустые строки между данными. Встретив их в файле, задающем функцию одной переменной, он строит три графика вместо одного. Например, пусть в файле str.dat записаны следующие строки:

```
#   x   y   Yerror
    1.0 1.2 0.1
    2.0 1.8 0.1
    3.0 1.6 0.1


    1.1 0.8 0.2
    2.1 0.3 0.2
    3.1 1.0 0.2


    1.2 1.5 0.3
    2.2 2.3 0.3
    3.2 3.1 0.3
```

В результате после выполнения команд

Пример 3.1

```
set key font ',20'
set xtics font ',16'
set ytics font ',16'
plot 'str.dat' using 1:2 with lines
```

получится рисунок 20. Заметим, что в этом случае все три графика рисуются одним цветом.

3.1.2 Матрицы данных

В дополнение к **построчному хранению данных** Gnuplot допускает **матричное хранение**, которое применяется для определения значений функции двух переменных. Матричное хранение бывает двух видов: **однородное (uniform)** и **неоднородное**.

Опишем вначале **однородный** матричный способ хранения. В этом случае все данные хранятся в виде таблицы чисел

```
z11 z12 z13 z14 ...
z21 z22 z23 z24 ...
```

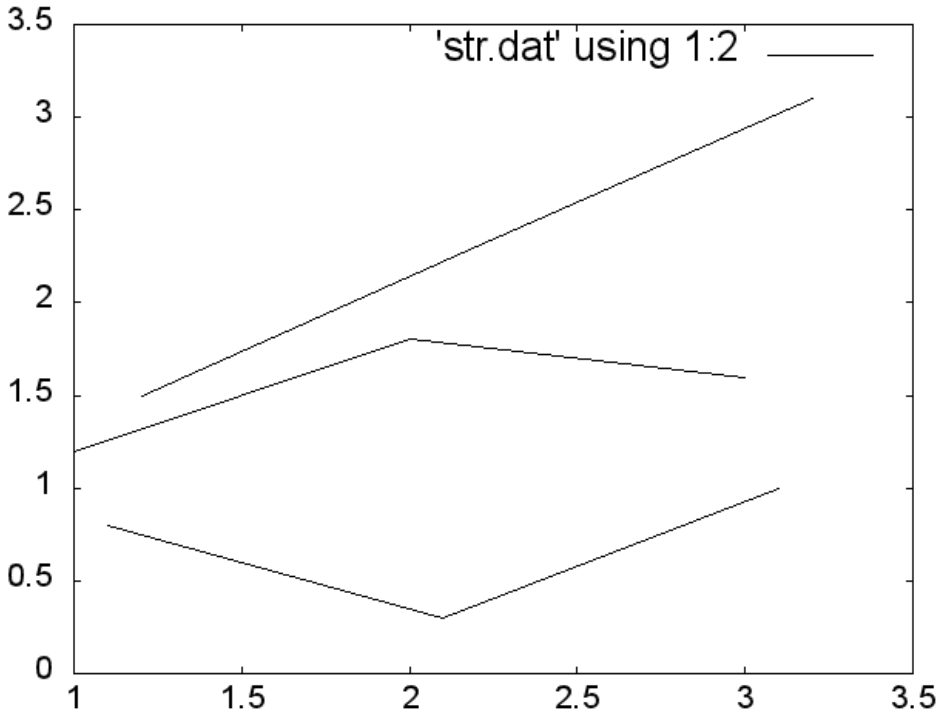


Рис. 20: эффект парных пустых строк

z31 z32 z33 z34 ...
. . . . . .

Каждый элемент этой таблицы задает значение функции $\mathbf{z} = \mathbf{f}(\mathbf{x}, \mathbf{y})$ по правилу

$$\mathbf{znm} = \mathbf{f}(\mathbf{m}, \mathbf{n}).$$

Пусть в файле matrix.dat задана матрица

0 1 4 9
1 2 5 10
4 5 8 13
9 10 13 18
16 17 20 25
25 26 29 34

По умолчанию при изображении поверхности форматом **matrix** координаты **x** и **y** это индексы строк и столбцов соответственно, т.е. диапазоны [0:3] и [0:5]. Следующий пример

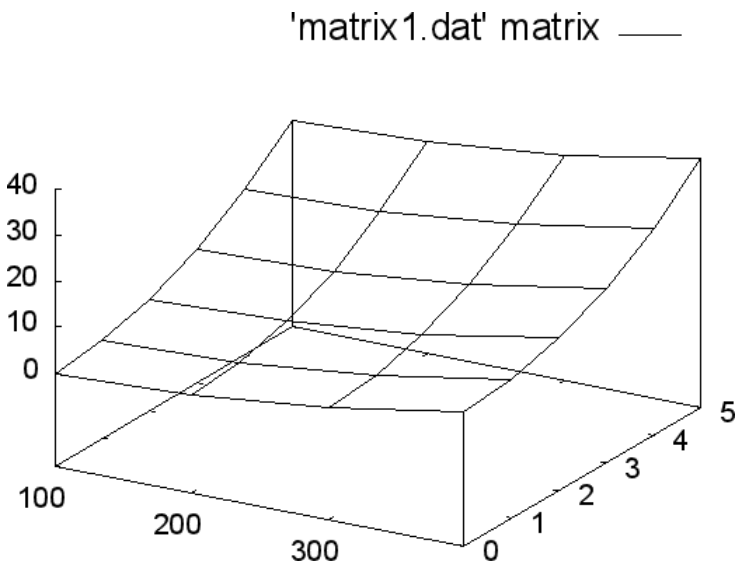


Рис. 21: Пример использования данных при однородном матричном способе хранения

показывает, как изменить диапазон по переменной **x** с [0:2] на [100:300].

Пример 3.2

```
set key font ',20'
set xtics offset 0,-1 font ',16'
set ytics offset 1,-0.25 font ',16'
set ztics 10 font ',16'
set xtics ("100" 0, "200" 1, "300" 2)
splot "matrix.dat" matrix with lines
```

Результат выполнения этих команд приведен на рисунке 21.

В случае **неоднородного** матричного способа хранения координаты точек в файле данных расположены по следующему принципу

N+1	y0	y1	y2	. . .	yN
x0	z00	z01	z02	. . .	z0N
x1	z10	z11	z12	. . .	z1N

. . . . . .

Эта таблица в процессе построения графика преобразуется в тройки чисел, задающие координаты точек поверхности

```
x0   y0   z00
x0   y1   z01
x0   y2   z02
. . . . . .
x0   yN   z0N
x1   y0   z10
x1   y1   z11
. . . . . .
```

Команда, строящая по таким данным поверхность, имеет следующий синтаксис

splot 'file.dat' nonuniform matrix

3.1.3 Комментарии и разделители

По умолчанию комментарии в строке файла с данными начинаются после символа #, но есть возможность задать набор символов, при встрече которого следующая за ним часть строки будет пропущена

datafile commentschars {"<string>"}

В интерактивном режиме может оказаться полезной команда, выводящая на экран ранее установленный командой **set datafile commentschars** набор символов

show datafile commentschars

Для отмены раннее установленного набора символов существует команда

unset commentschars

По умолчанию данные в строке разделяются пробелами. Однако на практике встречаются случаи, когда разделение полей осуществляется запятой или другим символом. Для этого в Gnuplot есть команда

set datafile separator {"<char>"|whitespase}

С ее помощью можно задать любой символ в качестве разделителя полей. Например, команды

set datafile separator ″,″
set datafile separator ″\t″

задают в качестве разделителя данных запятую или символ табуляции соответственно.

3.1.4 Форматы записи чисел

Для правильного считывания разных форм записи числовых данных из файлов в Gnuplot предусмотрены команды, задающие нужный формат.

Для чтения чисел из файла с данными, хранящихся в двоичном формате, обязательно следует указывать ключ **binary** после имени 'datafile'. Для задания по умолчанию чтения чисел в двоичном формате можно использовать команду

set datafile binary

Более подробно про чтение файлов, содержащих двоичный формат записи чисел, можно прочитать в разделе **binary** [1].

В языке FORTRAN используются символы 'd', 'D', 'q' и 'Q' при записи чисел в экспоненциальной форме. Для успешного считывания данных, записанных по такому формату, следует применить команду

set datafile fortran

Для возвращения обычного формата считывания чисел нужно использовать команды

unset datafile binary
unset datafile fortran

3.2 Фильтрация данных (опция USING)

Синтаксис опции **using** следующий

using <entry> {:<entry> {:<entry> ...}} {'format'}

Объяснение использования этой опции начнем с простейшего примера. Часто файлы данных содержат в каждой строке больше информации, чем требуется для изображения конкретного графика. Примером такой ситуации служит файл 'antrop.dat', содержащий следующие три колонки данных

Height	Weight	Age
0.3	3.2	0.1
0.4	4	1
0.8	6	3
1	15	8
1.2	30	11

Используя этот файл, можно построить зависимости величины роста от возраста и веса от возраста. Эту задачу можно решить многими способами. Все следующие команды выполняют это действие:

plot $'$**antrop.dat**$'$ **using 3 : 1**, $''$ **using 3 : 2**

plot $'$**antrop.dat**$'$ **u** (**column**($''$**Age**$''$)) : (**column**($''$**Height**$''$)), \
$''$ **using** (**column**($''$**Age**$''$)) : (**column**($''$**Weight**$''$))

plot $'$**antrop.dat**$'$ **using** (**column**($''$**Age**$''$)) : (**column**(**1**)), \
$''$ **using** (**column**($''$**Age**$''$)) : (**column**(**2**))

Результат изображен на рисунке 22.

Замечание 1. Если в одной команде **plot** требуется построить несколько графиков, данные для которых берутся из одного и того же файла, то имя файла требуется указывать один раз. Дальнейшие обращения к этому файлу будут происходить, если в нужном месте ставить просто две одинарные кавычки,

Замечание 2. Комментарии о том какой график какую зависимость отражает в данном конкретном случае были автоматически созданы благодаря использованию команды

set key autotitle columnheader

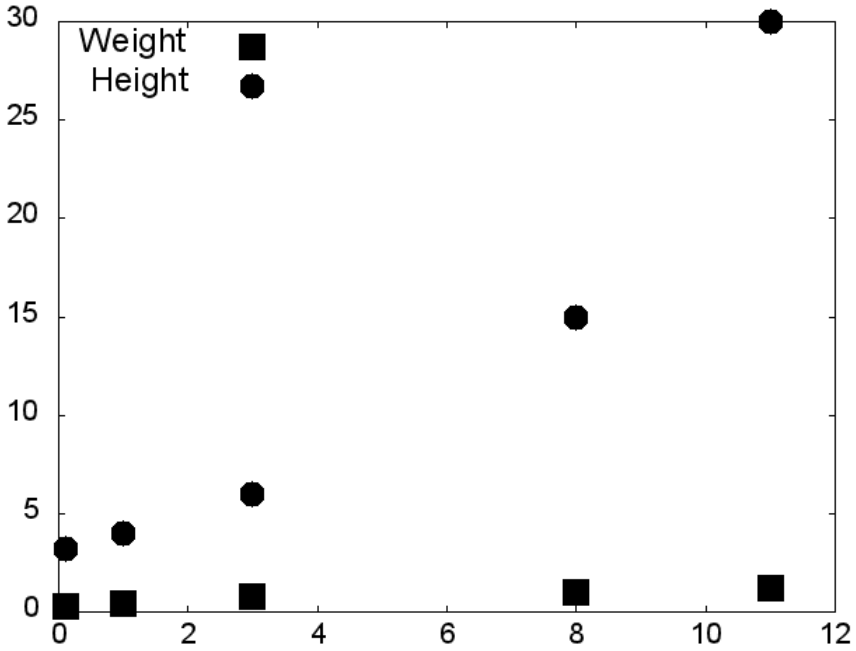


Рис. 22: Зависимость роста и веса от возраста

Пустая запись в перечислении используемых столбцов по умолчанию будет равна ее номеру в списке. Например, такое перечисление ::4 интепретируется, как 1:2:4. Если в перечислении указан только один столбец, то он используется для значений функции, а аргументами служат значения $0. Иначе говоря, команда

plot 'file' using 1

эквивалентна команде

plot 'file' using 0:1

Опция **using** предоставляет возможность указать формат, по которому нужно считать данные из файла. Все числа Gnuplot рассматривает, как записанные с двойной точностью в форме с плавающей точкой. Поэтому единственным допустимым числовым форматом при чтении является '%lf'. Формат строки для чтения данных из файла должен содержать минимум одну спецификацию числового формата и максимум семь. В строке формата могут присутствовать символы ' ' (пробела), '\t'

(табуляции), '\n' (новой строки) и '\f' (formfeed). Все остальные символы пропускаются. Использование символов пробела, табуляции, новой строки и formfeed в строке формата требует использования для ее обрамления двойных кавычек, а не одинарных.

Начнем с простейшего примера, когда файл 'file.dat' содержит минимум три колонки чисел. Иначе говоря, каждая строка файла данных состоит не меньше, чем из трех чисел, разделенных пробелами: $\mathbf{x_k\ y_k\ z_k}$ ..., где $\mathbf{k}$ - это номер строки. Функция, график которой требуется нарисовать, задается этими данными с помощью формулы

$$\mathbf{f(x)\ =\ y + z}.$$

Команда, создающая изображение графика этой функции, выглядит следующим образом

plot 'file.dat' using 1:($2+$3) '%lf,%lf,%lf'

Примером использования гораздо более сложного формата может служить следующая команда

$$\mathbf{plot\ 'file.dat'\ using\ ''\%*lf,\%lf,\%*20[\hat{}\backslash n]\%lf''}$$

Отдельные части этого составного формата означают

$\%*\mathbf{lf}$ – пропуск числа,
$\%\mathbf{lf}$ – чтение числа, записанного в формате десятичного с фиксированной точкой (по умолчанию для переменной x),
$\%*\mathbf{20}[\hat{}\backslash\mathbf{n}]$ – пропуск 20 символов перехода на новую строку ,
$\%\mathbf{lf}$ – чтение числа, записанного в формате десятичного с фиксированной точкой (по умолчанию для переменной y).

Опишем еще один трюк, позволяющий фильтровать данные. Он использует **тернарный оператор** (см. 11.1, с. 193). Приведем пример:

plot 'file' using 1:($3 > 10 ? $2 : 1/0)

Этот оператор для точки с координатой x, равной значению первого столбца, определяет координату y, равной значению второго столбца, если значение из третьего столбца больше

10, в противном случае точка не изображается, поскольку ее ордината полагается равной 1/0, а значение этого выражения не определено. Получающийся график совпадает с тем, который был бы изображен, если бы в нужных строчках во втором столбце стояло значение NaN.

Среди команд Gnuplot есть команда, позволяющая задать набор символов, при встрече которого в файле с данными соответствующая строка (точка графика) будет пропущена

set datafile missing {"<string>"}

Может оказаться полезной команда, выводящая на экран ранее установленный командой **set datafile missing** набор символов

show datafile missing

Для отмены раннее установленного набора символов существует команда

unset datafile

Разберем пример выполнения команды **plot** в четырех различных ситуациях. Пусть в интерактивном режиме были введены следующие команды:

```
# шаблон графика — ломаная, соединяющая вводимые точки
set style data linespoints
set key at 1.,50
# ключ '-' означает, что данные будут вводиться с экрана
plot '-' title 'a'
1 10
2 20
3 ?
4 40
5 50
e # символ e означает окончание ввода данных с экрана
set datafile missing {"?"}
plot '-' title 'b'
1 10
```

```
2 20
3 ?
4 40
5 50
e
plot '-' using 1:2 title 'c'
1 10
2 20
3 NaN
4 40
5 50
e
plot '-' using 1:($2) title 'd'
1 10
2 20
3 NaN
4 40
5 50
e
```

До версии Gnuplot 5.0 результат обработки этих данных был следующий. При первом вызове команды **plot** из третьей строки читается только одно число 3. По правилам gnuplot такая строка содержит одно число и его значение присваивается ординате соответствующей точки. Абсцисса этой точки по умолчанию равна номеру строки, т.е. 2 (напомним, что нумерация строк начинается с нуля). Таким образом, на графике будет изображена точка (2,3), что, естественно, неверно.

Второй и третий вызовы команды **plot** верно игнорируют третью строку вводимых данных и на графике точки с координатами (2,20) и (4,40) соединяются отрезком.

Четвертый вызов команды **plot** также проигнорировал третью строку, но на графике точки с координатами (2,20) и (4,40) не соединяются.

Начиная с версии 5.0, результат в первых трех случаях равен результату предудыдущих версий во втором и третьем случаях, а в четвертом для всех версий получается одинаковый результат (см. рис. 23).

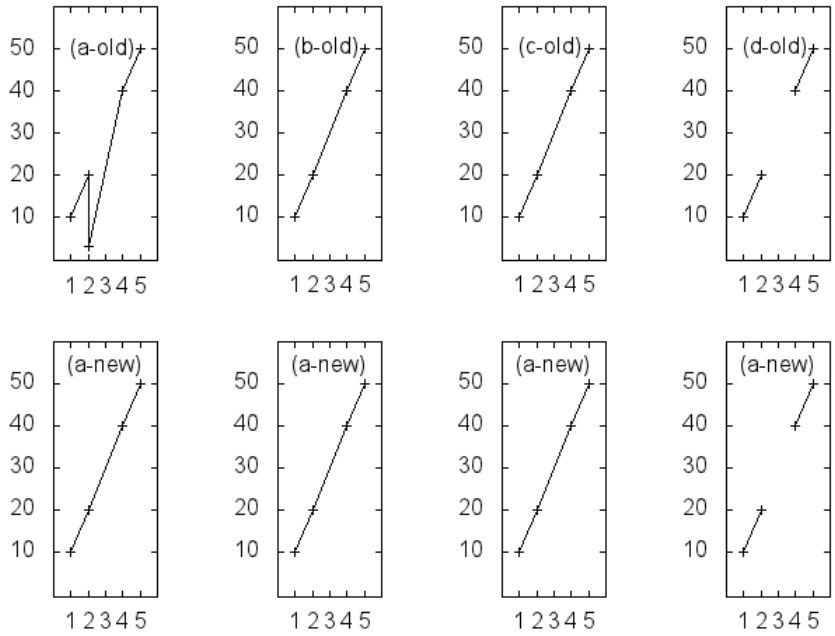


Рис. 23: Изменения в фильтрации данных

3.3 Фильтрация данных (опция INDEX)

По умолчанию считается, что при построении изображения требуется использовать данные из всех строчек файла. Строки нужно отфильтровывать, если не все они нужны для строящегося графика. Опция **{index<index_list>}** позволяет задать какие строки следует использовать.

Разделителями между частями файла служат две пустые строки. Ниже в этом параграфе будем называть части файла, разделенные двумя пустыми строками, группами. Общий синтаксис команды

plot 'file.dat' index {<m>{:<n>{:<p>}}|"<name>"}

Если указать только одно число (например, **index 1**), то при построении графика будут использованы данные группы с этим номером. Номера групп начинаются с нуля. Указание двух чисел через двоеточие говорит о том, что при построении изображения нужно использовать данные всех групп, номера которых больше или равны первого числа и меньше или равны второго. Например, **index 1:3** означает использование групп 1, 2,

3. Наконец, при указании трех чисел, разделенных двоеточием, означает циклический выбор групп с номерами большими или равными первому числу, меньшими и равными второму числу и шаг перебора будет равен третьему числу. Например, **index 0:10:2** означает использование групп 0, 2, 4, 6, 8 и 10.

Задание опции **index "<name>"** означает выбор данных, отмеченных меткой "<**name**>". Метку ставят в строке комментариев, при этом в саму метку знак комментария и пробелы не входят. Если строка комментариев начинается с <**name**>, то следующая за строкой группа данных подлежит использованию. Например, по команде

plot 'discr.dat' index '=plot='

будут нарисованы только точки (5,5), (6,6), (7,7) и (8,8), если файл discr.dat содержит следующие строки:

```
1 1
2 2
3 3
4 4


# =plot= !!!!!!!
5 5
6 6
7 7
8 8


9 9
10 10
```

Еще один пример использования опции **index** — изображение данных из файла str.dat из раздела 3.1.1.

Пример 3.3

```
set key at 3, 3.4 font ',20'
set xtics font ',16'
set ytics font ',16'
```

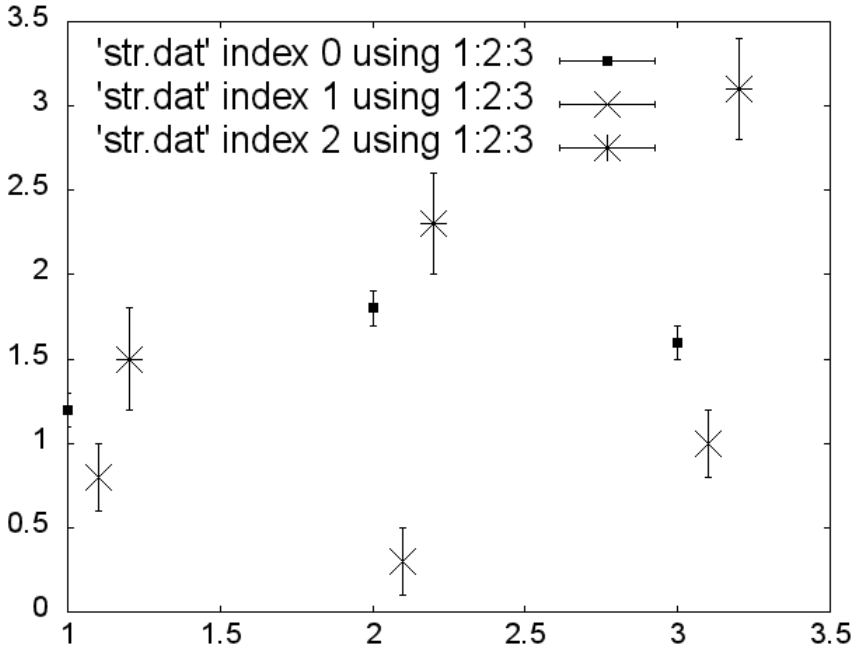


Рис. 24: использование опции index

```
plot "str.dat" index 0 u 1:2:3 w yerrorbars pt 5,\
     "str.dat" index 1 u 1:2:3 w yerrorbars ps 3,\
     "str.dat" index 2 u 1:2:3 w yerrorbars ps 3
```

Результат этих команд изображен на рисунке 24. В отличие от рисунка 20 все три графика изображаются разным цветом, если картинка не черно-белая.

Замечание. Команда сместить легенду была выполнена для того, чтобы она не закрывала точки графика (см. 8.2, с. 151).

Отметим еще одну особенность шаблонов, рисующих графики в виде дискретных точек (к ним относится стиль **yerrorbars** (см. 4.2, с. 86)). Эти шаблоны не делают различий между группами, разделенными двойными пустыми строками, т.е. команда построения графика

```
plot "str.dat" using 1:2:3 with yerrorbars
```

построит график одной функции, не делая различий между точками из разных групп. В то время как на рисунке 24 точки

разных групп изображены по-разному не только цветом, но и символьно.

3.4 Фильтрация данных (опция EVERY)

Другой опцией, позволяющей отфильтровывать данные, является **{every<every_list>}**. Синтаксис команды **plot** при использовании этой опции следующий

plot 'file.dat' every{<point_incr>}
{:{<block_incr>}
{:{<start_point>}
{:{<start_block>}
{:{<end_point>}
{:<end_block>}}}}}

Опция **every** позволяет работать с данными, записанными в файл периодическим образом. Данные выбираются, начиная от строки с номером <**start_point**> и кончая <**end_point**>, с шагом выборки <**point_incr**> и блоков от <**start_block**> до <**end_block**> с шагом выборки <**block_incr**>. Под блоком понимается множество строк, отделенных от другого блока пустой строкой. Все счетчики, как строк, так и блоков, начинаются с нуля. Важно помнить, что строки, содержащие "нерисуемую" информацию (например, комментарии), тоже нумеруются.

Любой из перечисленных параметров может быть пропущен. Знак ':' после каждого пропущенного параметра сохраняется. Шаги выборки по умолчанию равны единице.

Примеры.
1) **every :::3::3**— выбирает только четвертый блок (нулевой - это первый).
2) **every :::::9** — выбирает первые десять блоков.
3) **every 2:2** — выбирает каждую вторую точку в каждом втором блоке.
4) **every ::5:15** — выбирает строки с 5-ой по 15-ую в каждом блоке.

Ключ **skip N** означает, что нужно пропустить N строк в начале файла. Причем эти строки не учитываются при применении ключа **every**.

3.5 Фильтрация данных (опция THRU)

Опция **{thru<thru expression>}** введена для совместимости с ранними версиями Gnuplot. Например, следующая команда

plot 'file.dat' thru f(x)

полностью эквивалентна команде

plot 'file.dat' using 1:(f($2))

3.6 Метод наименьших квадратов

3.6.1 Команда FIT

В Gnuplot есть возможность построить методом наименьших квадратов приближение для заданных дискретных данных. В основу положен алгоритм Левенберга-Марквардa (Marquardt-Levenberg algorithm), сформулированный независимо Левенбергом (1944) и Марквардом (1963). Синтаксис команды следующий:

fit {<ranges>} <expression>
'<datafile>' {datafile-modifiers}{{unitweights}|
{y|xy|z} error | errors <var1>{,<var2>,...}}
via '<parameter file>' | <var1>{,<var2>,...}

Ключ **<ranges>** (см. 5.1, с. 110) служит фильтром для используемых данных: все точки, не попадающие в указанный диапазон, не учитываются при построении приближения. Диапазоны могут быть определены как для всех независимых переменных, так и для некоторых из них. Также можно задавать диапазон изменения для зависимой переменной, который тоже будет являться фильтром для используемых данных.

Выражение <**expression**> может быть любой допустимой в Gnuplot формулой, использующей действительные переменные. Обычно это заданная пользователем функция $\mathbf{f}(\mathbf{x})$ или $\mathbf{f}(\mathbf{x}, \mathbf{y})$. Имена независимых переменных задаются командой **set dummy** (см. 10.7, с. 191) или в ключе <**ranges**>. По умолчанию выражение должно зависеть от одной переменной **x** или двух — **x** и **y**, которые будут использованы как независимые. Также выражение должно зависеть от одной или более величин <**var**>, значения которых и выбираются по методу наименьших квадратов.

Имя <**datafile**> используется аналогично использованию в команде **plot**, а в качестве **datafile-modifiers** могут быть применены любые фильтры за исключением **smooth**. Использование данных из файла можно организовать всеми доступными в Gnuplot способами. Например, независимую переменную можно определить как сумму чисел из второго и третьего столбцов, а значение из шестого столбца использовать в качестве веса

fit ... using ($2+$3):6

В случае недостатка используемых спецификаций в опции **using** (**using** формирует лишь одно значение для каждой строки данных) считается, что в выражении есть всего одна независимая переменная. Такой же вывод делается, если файл содержит лишь одну колонку или из него читается лишь одна колонка (в этом случае по умолчанию она содержит значения функции). Вообще же при наличии опции **using** может быть до 12 независимых переменных.

Опция <**unitweights**> задает веса точек в методе наименьших квадратов. По умолчанию все точки считаются равноправными. Значения весов могут быть заданы, если использовать ключевое слово **errors**, чтобы считать одну или более точность задания для одной или более переменной из файла с данными. Эти точности интерпретируются как простые весовые множители вида $\frac{\mathbf{1}}{\mathbf{s^2}}$ при учете считанных данных, где s — длина интервала ошибки.

За опцией <**unitweights**> должен следовать список перечисленных через запятую имен переменных, для которых будут считаны точности задания. В этот список должна обязательно входить зависимая переменная **z**. Включение независимых переменных необязательно. Для каждой такой переменной файл с данными должен содержать дополнительную колонку, из которой будет считываться ошибка. Допускается произвольное использование данных строки для определения ошибки задания данных. Заметим, что число независимых переменных определяется по правилу: число спецификаций в опции **using** минус 1 (зависимая переменная) и минус число переменных, для которых заданы ошибки.

Например, при задании 2-х независимых переменных, ошибки для первой из них и ошибки для зависимой переменной (использована опция **errors x,z**) нужно использовать опцию **using** со следующими спецификациями

x : y : z : sx : sz

Здесь **x** и **y** — способы задания независимых переменных, **z** — зависимой переменной, **sx** и **sz** — ошибок.

В случае, когда ошибки считываются лишь для зависимой переменной, используют короткую запись опции **errors z**: **yerrors** в случае одной независимой переменной и **zerrors** в случае нескольких независимых переменных. Задание таких опций сообщают Gnuplot о том, что требуется столбец данных, содержащий ошибки зависимой переменной. Аналогично, при задании опции **xyerrors** в случае одной независимой переменной означает, что требуется два дополнительных столбца в файле с данными с ошибками для независимой и зависимой переменных.

На рисунке 25 приведен результат поиска методом наименьших квадратов приближения данных файла "fit.dat"

#	**x**	**y**	$\mathbf{y_{min}}$	$\mathbf{y_{max}}$
	-2	**5.5**	**3.9**	**6**
	-1	**0.0**	**-0.5**	**1.5**
	0	**-0.5**	**-1**	**0.1**
	1	**2**	**0.7**	**2.1**

2	**3**	**2.9**	**4.2**
3	**10**	**8.5**	**10.2**
4	**15**	**14.8**	**16.4**

Изображение было получено после выполнения следующих команд.

Пример 3.4

```
# Подключить режим сохранения статистических параметров
# в виде значений стандартных переменных.
set fit errorvariables
set key left font ',16' # Сместить легенду влево.
set xtics font ',16'
set ytics font ',16'
# Заголовок для списка выводимых параметров.
set label 'FINAL PARAMETRS' at graph 0.2, 0.8 \
font "Times,20"
f(x)=a*x*x+b*x+c # Задать вид приближения.
# Построить приближение.
fit f(x) 'fit.dat' using 1:2:(-$3+$4) yerror via a,b,c
# Вывод коэффициентов a, b и c.
set label sprintf("a=%6.4f",a) \
at graph 0.2, 0.7 font "Times,14"
set label sprintf("b=%6.4f",b) \
at graph 0.2, 0.65 font "Times,14"
set label sprintf("c=%6.4f",c) \
at graph 0.2, 0.6 font "Times,14"
# Вывод статистических параметров.
set label gprintf("+/- %6.3s*10^{%S}",a_err) \
at graph 0.4, 0.7 font "Times,14"
set label gprintf("+/- %6.3s*10^{%S}",b_err) \
at graph 0.4, 0.65 font "Times,14"
set label gprintf("+/- %6.3s*10^{%S}",c_err) \
at graph 0.4, 0.6 font "Times,14"
set label sprintf("FIT NDF=%.0f", FIT_NDF) \
at graph 0.2, 0.55 font "Times,14"
set label sprintf("FIT WSSR=%e", FIT_WSSR) \
at graph 0.2, 0.5 font "Times,14"
```

```
set label sprintf("FIT STDFIT=%e", FIT_STDFIT) \
at graph 0.2, 0.45 font "Times,14"
# Построение графиков данных и полученного приближения
# в одних осях.
plot [-2.5:4.5] 'fit.dat' u 1:2:3:4 with yerrorbars, f(x)
```

Приближение искалось в виде квадратичной функции

$$\mathbf{y = a\,x^2 + b\,x + c}$$

. Найденные значения коэффициентов параболы и полученные статистические параметры были сохранены в соответствующих переменных. В частности, для этого была выполнена команда

set fit errorvariables

После чего часть этих величин были выведена в виде дополнительных надписей на поле изображения.

3.6.2 Вывод параметров алгоритма

Выполняя команду **fit**, Gnuplot определяет значения параметров, заданных в опции **via**, оценивает ошибки в определении этих параметров и другую полезную статистическую информацию. В частности,

FIT_NDF — число степеней свободы;
FIT_WSSR — взвешенная сумма квадратов остатков;
FIT_STDFIT — $\dfrac{\mathbf{WSSR}}{\mathbf{NDF}}$;
FIT_P — достигаемый уровень значимости.
Вывод этих параметров осуществляется командой

```
set fit {logfile {"<filename>"|default}}
        {{no}quet|results|brief|verbose}
        {{no}errorvariables}
        {{no}covariancevariables}
        {{no}errorscaling}
        {{no}prescale}
        {maxiter <value>|default}
```

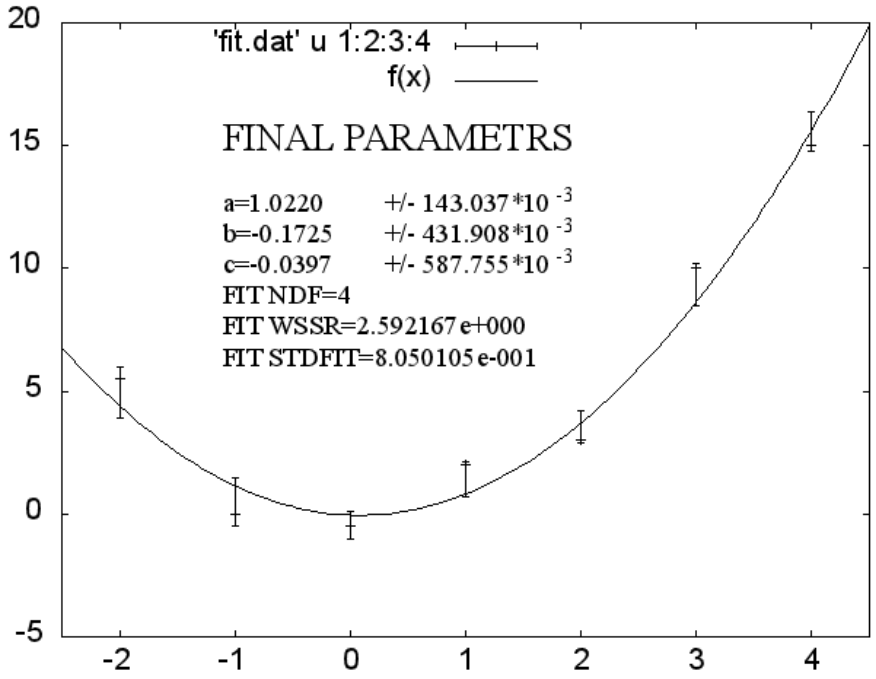


Рис. 25: Метод наименьших квадратов

{limit <epsilon>|default}
{limit_abs <epsilon_abs>}
{start-lambda <value>|default}
{lambda-factor <value>|default}
{script {"<command>"|default}}
{v4 | v5}

Опция **logfile** задает место, куда команда **fit** выводит свои результаты. По умолчанию вывод осуществляется в файл **fit.log**. Если имя <**filename**> заканчивается символами '/' или '\', то это считается путем к директории, где следует создать файл **fit.log**. Вся информация, записываемая в **fit.log**, также выводится в окно, где была запущена Gnuplot. Отключить вывод на экран можно командой

set fit quiet

Если указан ключ **rezults**, то выводятся лишь конечные результаты. Ключ **brief** добавляет к конечным результатам краткую информацию о ходе уточнения параметров (ее размер — одна строка на каждую итерацию). Ключ **verbose** означает, что требуется вывести подробную информацию об итерационном процессе.

При заданной опции **errorvariables** ошибки в определении каждого из параметров присваиваются специальным переменным. Имена этих переменных состоят из имени самого параметра и суффикса '**_err**'. Это позволяет выводить информацию об их значениях в виде надписей на поле изображения (см. рис. 25).

Ключ **errorscaling** (он задается по умолчанию) означает масштабирование ошибок: деление их на коэффициент **FIT_STDFIT**. Это масштабирование можно отменить, указав ключ **noerrorscaling**. Заметим, что если веса у данных не используются, то ошибки масштабируются всегда.

При указанном ключе **prescale** параметры масштабируются с помощью своих стартовых значений еще до процедуры Левенберга-Маркварда. Это очень помогает в случае, когда значения параметров очень сильно различаются по порядку их величин. В случае, когда все параметры близки к нулю, масштабирование не используется.

Максимальное количество итераций ограничивается параметром **maxiter**. Указание значения '0' означает, что ограничения нет. По умолчанию также считается, что нет ограничения на максимальное количество итераций.

Параметр **limit** задает критерий окончания итерационного процесса: итерации заканчиваются, когда взвешенная сумма квадратов остатков (**WSSR**) становится меньше значения этого параметра. По умолчанию он равен **1e-5**. Параметр **limit_abs** налагает дополнительное условие на то, чтобы сумма абсолютных значений изменений компонент **WSSR** при прерывании итераций была меньше указанного значения.

Для более тонкого задания параметров алгоритма Левенберга-Маркварда используется параметр **lambda**. Его начальное значение вычисляется автоматически, используя **ML-**

матрицу, но можно явно задать это значение через параметр **start_lambda**. Выбрав ключ **default**, можно вернуться к автоматическому выбору начального значения. Параметр **lambda_factor** задает коэффициент изменения **lambda** в зависимости от изменения значения целевой функции. По умолчанию этот коэффициент равен 10.

Опция **script** определяет команду Gnuplot, которая выполняется при аварийном завершении команды **fit**. Она нужна для предотвращения в этом случае автоматической перерисовки графика с неверными параметрами.

При указании опции **covariancevariables** ковариационная матрица между окончательными значениями параметров сохраняется в виде значений переменных. Имена этих переменных состоят из общего префикса **'FIT_COV_'**, далее идет имя первой переменной и через знак **'_'** — имя второй переменной. Например, ковариация двух параметров **a** и **b** запишется в переменную с именем **'FIT_COV_a_b'**.

В пятой версии Gnuplot поменялась задаваемая по умолчанию настройка при остутствии ключа **error**: теперь в этом случае автоматически выбирается опция **unitweights**. Для возвращения настроек четвертой версии следует указать ключ **v4**.

4 Шаблоны графиков

4.1 Команда TEST

Ниже мы будем называть шаблоном способ изображения кривой или поверхности. Команда **test** является инструментом, позволяющим получить информацию о значениях параметров, используемых при создании различных шаблонов

test {**terminal** | **palette** [**rgb**|**rbg**|**grb**|**gbr**|**brg**|**bgr**]} (4.1)

Результат исполнения команды **test terminal** приведен на цветной иллюстрации V (с. 236). В частности, из него можно узнать какие значения параметров соответствуют тем или иным линиям и точкам.

Задание опции **palette** позволяет получить информацию о возможных оттенках цветов для данного терминала. На цветной иллюстрации VI (с. 236) приведен результат выполнения команды **test palette rgb**. На этом рисунке можно увидеть цветовую схему (**colorbox**) и примеры графиков, изображенных тремя основными цветами (красным, зеленым, синим) и черным цветом.

4.2 Шаблоны графиков (опция WITH)

4.2.1 Синтаксис ключа задания шаблона

Графики на рисунках могут быть изображены множеством способов. Выбор шаблона осуществляется явным заданием опции **with** . Приведем синтаксис этого ключа:

with <**style**> { {**linestyle** | **ls** <**line_style**>}
| {{**linetype** | **lt** <**line_type**>}
{**linewidth** | **lw** <**line_width**>}
{**linecolor** | **lc** <**colorspec**>}
{**pointtype** | **pt** <**point_type**>}
{**pointsize** | **ps** <**point_size**>}
{**fill** | **fs** <**fillstyle**>} {**nohidden3d**}
{**nocontours**}{**nosurface**}{**palette**}}}

4.2.2 Функций непрерывного аргумента

Для графиков функций, заданных аналитическим выражением, подходят два шаблона **lines (l)** и **linespoints (lp)**. В **lines** график рисуется в виде линии, для которой можно задать цвет и толщину, указав ключи **linecolor (lc)** и **linewidh (lw)**. В **linespoints** на графике отмечаются еще точки цвета линии. Форму знаков, которыми изображаются точки, их размер и интервал между ними можно задавать, указав ключи **pointtype (pt)**, **pointsize (ps)** и **pointinterval (pi)**. Все задаваемые параметры имеют набор возможных числовых значений, задать которые можно либо явно, либо указав тип (**linetype (lt)**) или стиль (**linestyle (ls)**) линии (см. 10.4.1, с. 181).

Графики функций, заданных аналитическим выражением, допустимо изображать, выбрав шаблоны **points** или **dots**. В этом случае графики изображаются набором точек, расстояния между которыми по оси абсцисс визуально малы.

Приведем пример использования команд **set style line** и **plot** с опцией **with**. Для этого нарисуем, используя различные шаблоны, линейные функции. Пусть командный файл содержит следующие команды:

Пример 4.1

```
set key spacing 2. # Задать межстрочный интервал в легенде.
set style line 1 lc "brown" lw 2 pt 7 ps 1 pi 30
set style line 2 lc 2 lw 3 pt 3 ps 5 pi 20
set style line 3 lc 3 lw 1 pt 8 ps 3 pi 10
set style line 4 lc 0 lw 4 pt 1 ps 2 pi 5
plot [-1:1] [0:7] \
-x+5. with lines lt 1 title "lines",\
-x+4. with points ls 1 title "points",\
-x+3. with linespoints ls 2 title "linespoints",\
-x+2. with linespoints ls 3 title "linespoints",\
-x+1. with dots ls 4 title "dots"
```

Результатом работы этих команд является графическое изображение, представленное на цветной иллюстрации VII

(с. 237). Кратко прокомментируем свойства линий графиков, изображенных на рисунке, перебирая их сверху вниз.

При построении самого верхнего графика был использован шаблон **lines**, тип линии 1, остальные параметры заданы по умолчанию. У остальных графиков используются специально заданные стили. У второго графика задан стиль с индексом 1, а сам график рисуется по шаблону **points**. Шаблон нужно задавать явно, т.к. ключ **ls** определяет лишь свойства линии. Шаблон линии **points** не допускает выделенных точек на графике. Линия этого стиля изображается набором точек, поэтому из набора свойств с индексом 1 используются лишь свойства, заданные ключами **lc** и **lw**. Для построения третьего и четвертого графиков использован шаблон **linespoints** и стили с индексами 2 и 3 соответственно. Шаблон **linespoints** использует все возможные свойства линии. Шаблоном **dots** изображена самая нижняя прямая. При его использовании график всегда рисуется набором очень маленьких дискретных кружочков без выделенных точек. Размер кружочков всегда один и тот же независимо от размера, указанного при задании параметра **lw**.

Разнообразить шаблоны непрерывных линий можно путем задания опции (**dashtype**)(см. 10.4.1, с. 181). Приведем пример ее использования.

Пример 4.2

```
set key samplen 5 c tm font ',24'
set xtics font ',20'
set ytics font ',20'
set dashtype 1 "– ... _ _ _ _ "
plot sin(x) dt 1, cos(x) dt (10,5,4,8,1,5)
```

Графики, построенные с помощью команд примера 4.2, изображены на рисунке 26.

Цвет линии графика можно задать в файле с данными в виде отдельной колонки. В этом случае синтаксис команды **plot** следующий

```
plot 'data' using n:m:p with points lc variable
```

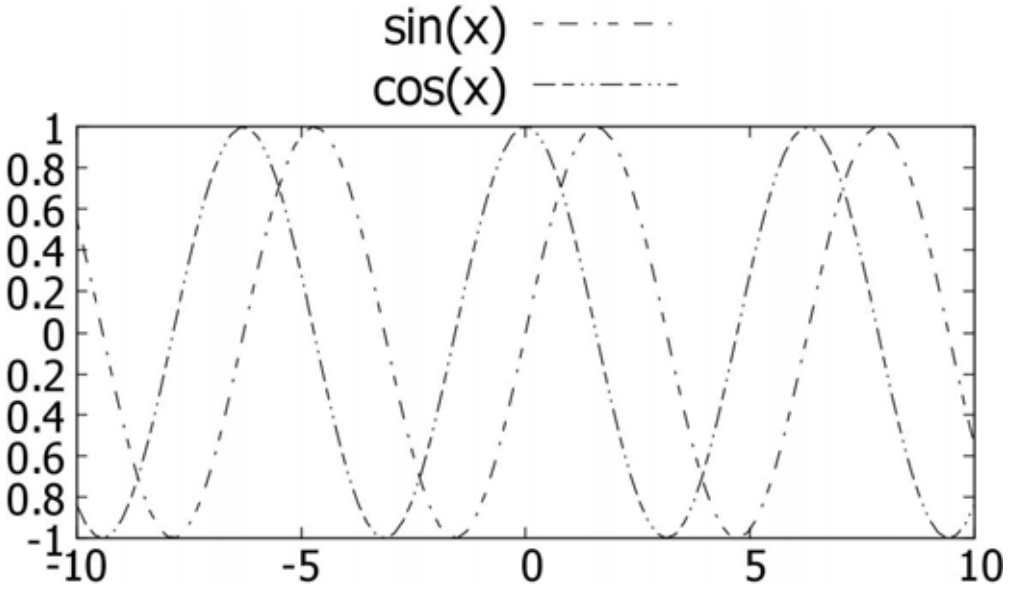


Рис. 26: пример различных 'dashtype'

Таким образом, при задании ключа **lc variable** требуется указывать номер третьей колонки, в которой должен быть записан тип линии (см. 4.1, с. 86). По этому номеру для каждой точки определяется ее цвет.

Можно разнообразить (или промаркировать) линии графиков, выбрав произвольный символ для обозначения точек. Например, выполнив следующую команду

plot sin(x) with linespoints pt 'a'

получим график функции **sin(x)**, на котором будут точки отмечены буквой **'a'**. Размер символов и частота их изображения могут быть заданы опциями **ps** и **pi** соответственно.

4.2.3 Функций дискретного аргумента

Приведем несколько примеров использования различных шаблонов для функций дискретного аргумента. В качестве файла данных возьмем файл with2.dat, содержащий следующие строки

1.0 2.0
2.0 3.0
3.0 1.0
4.0 2.5
5.0 1.5

Начнем с **impulses**. При использовании этого шаблона для каждой точки графика проводится вертикальная линия от точки до оси X. На рисунке 27 приводится пример графика, созданного по команде.

Пример 4.3

plot [0:6] [0:4] "with2.dat" with impulses title "impulses"

Замечание. В легенде графика была бы надпись **"with2.dat"**, которую Gnuplot создал по умолчанию. если не указать значение ключа **title "impulses"**.

Шаблон **impulses** может быть эффективно использован при изображении графиков функций в полярных координатах. Приведем пример картинки, нарисованной с помощью команд.

Пример 4.4

```
set xrange [-30:60]
set yrange [-40:20]
set polar
plot t**t+2*t+3 with impulses
```

Результат выполнения этих команд изображен на рисунке 27. Поскольку явного задания легенды в команде нет, то название графика будет состоять из формулы, задающей функцию.

Если задать шаблон **steps**, график будет ступенчатой функцией, причем сначала от предыдущей точки проводится горизонтальная линия, а затем уже вертикальная. На рисунке 27 приведен пример графика, созданного командой.

Пример 4.5

plot [0:6] [0:4] "with2.dat" with steps title "steps"

Шаблон **fsteps** очень похож на предыдущий. Разница заключается в порядке построения "ступеньки". В этом случае сначала

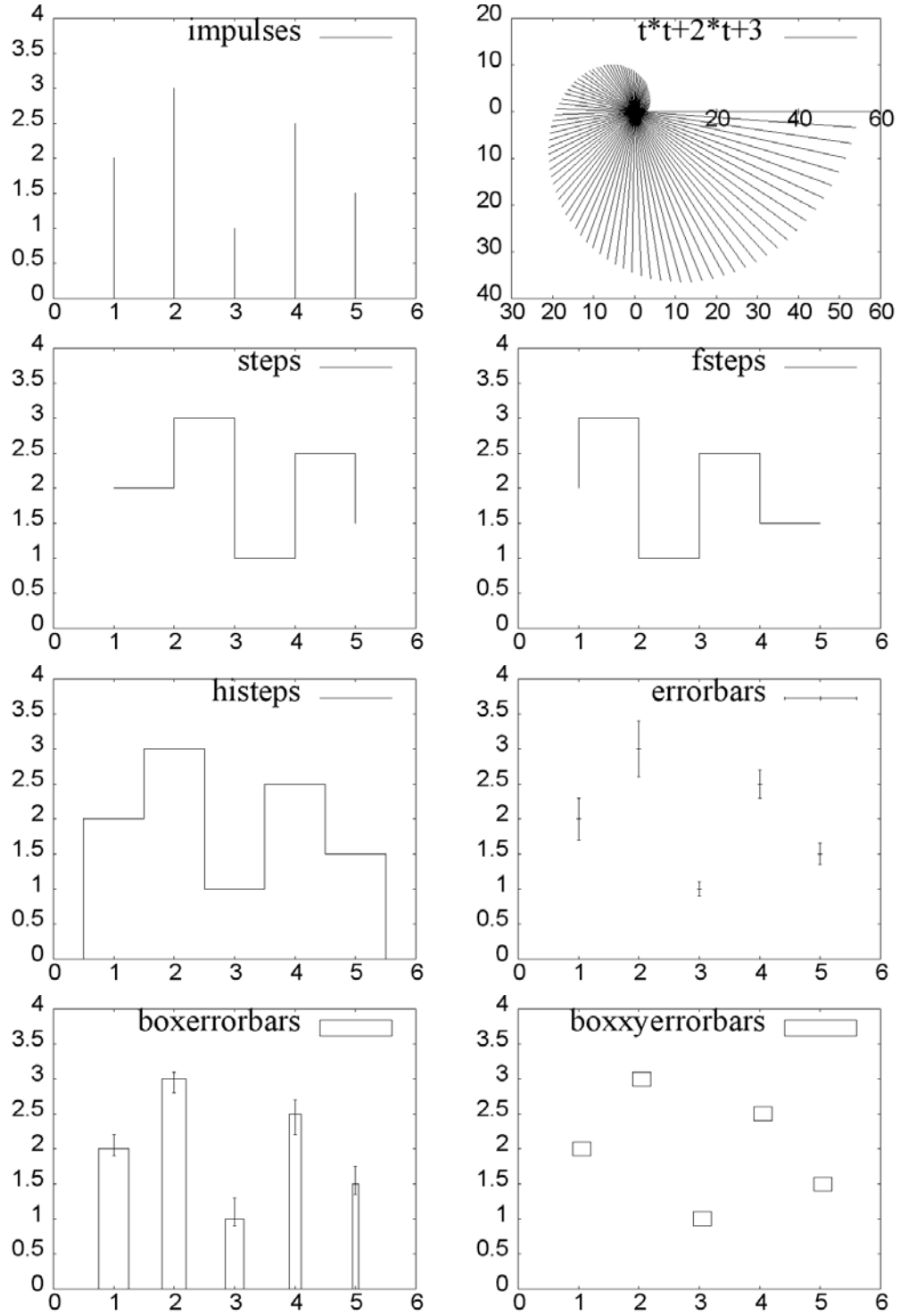


Рис. 27: Шаблоны функций дискретного аргумента

строится вертикальная, а уж потом горизонтальная линия (см. рис. 27).

Пример 4.6

plot [0:6] [0:4] "with2.dat" with fsteps title "fsteps"

Шаблон **histeps** предназначен для построения гистограмм. Процесс построения изображения выглядит следующим образом. Пусть (x_0, y_0), (x_1, y_1) и (x_2, y_2) три последовательные точки. Gnuplot проводит горизонтальную линию, соединяя точки с координатами $((x_0 + x_1)/2, y_1)$ и $((x_1 + x_2)/2, y_1)$. Затем проводятся вертикальные линии из начала и конца построенной горизонтали до значений $y = y_0$ и $y = y_2$ соответственно. Для граничных точек горизонтальные отрезки проводятся таким образом, чтобы точки оказались в середине соответствующих интервалов. На рисунке 27 приведен пример гистограммы, построенной по команде.

Пример 4.7

plot [0:6] [0:4] "with2.dat" with histeps title "histeps"

Шаблон **errorbars** (другое название **yerrorbars**) рисует картинку по следующему правилу. На месте каждой точки рисуется маленький кружочек, а затем проводится вертикальный отрезок. Для задания концов отрезка используются два типа данных: можно задать либо длину отрезка, либо его концевые точки. В первом случае отрезок соединяет две точки $(x, y - \delta)$ и $(x, y + \delta)$, а во втором – точки с координатами (x, y_{low}) и (x, y_{high}). Вид определяется автоматически по количеству заданных данных: если заданы три колонки, то задана длина интервала, а если четыре, то известны величины y_{low} и y_{high}. Шаблон используется для изображения изменения некоторой величины, про значения которой известно, что они заданы с определенной точностью. Например, изображается величина $y = y(x)$, значения которой измерены с некоторой точностью. Для иллюстрации сказанного возьмем файл данных with3.dat, содержащий следующие строчки

1.0 2.0 0.3

2.0 3.0 0.4
3.0 1.0 0.1
4.0 2.5 0.2
5.0 1.5 0.15

На рисунке 27 приведен график, построенный командой.

Пример 4.8

plot [0:6] [0:4] "with3.dat" w errorbars t "errorbars"

Шаблон **xerrorbars** используется для изображения графика зависимости $y = y(x)$, в которой значения координат точек графика по оси X считаются определенными с некоторой погрешностью. В этом шаблоне около каждой точки графика рисуются горизонтальные отрезки, характеризующие ошибки задания координат по оси X. Файл данных, как и в **yerrorbars**, должен содержать три или четыре колонки чисел.

Для зависимостей $y = y(x)$, в которых обе координаты точек заданы с некоторой погрешностью, используется шаблон **xyerrorbars**. При задании этого шаблона вокруг каждой точки строится два отрезка — как вертикальный, характеризующий ошибку по y, так и горизонтальный, характеризующий ошибку по x.

Шаблон **boxes** на месте каждой точки зависимости $y = y(x)$ рисует прямоугольник, нижняя сторона которого расположена на оси X, а верхняя на прямой, имеющую заданную ординату. Для определения ширины прямоугольника используется три различных алгоритма. Если в файле данных содержатся три столбца, то в качестве ширины прямоугольника берется значение из третьего столбца. Если заданы только два столбца, то используется значение, установленное с помощью команды **set boxwidth**. Если же выбор ширины невозможно сделать, используя два первых способа, то ширина столбцов уставливается такая, что два соседних прямоугольника склеиваются между собой.

Шаблон **boxerrorbars** представляет собой комбинацию шаблонов **boxes** и **yerrorbars**. Для графического изображения данных этим стилем требуется пять колонок данных. Первые две

колонки задают координаты точки графика, третья и четвертая – границы ошибки, а пятая – ширину прямоугольника. Зададим файл with4.dat, содержащий следующие строчки

1.0 2.0 1.9 2.2 0.3
2.0 3.0 2.8 3.1 0.4
3.0 1.0 0.9 1.3 0.2
4.0 2.5 2.2 2.7 0.1
5.0 1.5 1.35 1.75 0.15

На рисунке 27 приведен график, построенный по команде.

Пример 4.9

```
plot [0:6] [0:4] "with4.dat" \
with boxerrorbars title "boxerrorbars"
```

Шаблон **boxxyerrorbars** является комбинацией **boxes** и **xyerrorbars**. При его использовании вместо точки рисуется прямоугольник, длина и ширина которого определяются величинами ошибок. Центр прямоугольника имеет координаты соответствующей точки. Файл данных может содержать для каждой точки четыре или шесть параметров. В первых двух колонках содержатся координаты точек зависимости $y = y(x)$. В случае файла с четырьмя параметрами третья и четвертая колонки содержат размеры прямоугольников по x и y соответственно. Когда же колонок шесть, то числа в третьей и четвертой колонках задают абсциссы сторон прямоугольников, параллельных оси y, а в пятой и шестой – ординаты сторон, параллельных оси X. В качестве файла с данными можно использовать файл с двумя колонками. Для примера используем графическое представление (рис. 27) данных из файла with2.dat с двумя колонками (см. пример шаблона **impulses**), полученное с помощью команды.

Пример 4.10

```
plot [0:6] [0:4] "with2.dat"\
using 1:2:($1-0.1):($1+0.2):($2-0.1):($2+0.1)\
with boxxyerrorbars title "boxxyerrorbars"
```

Ключ **using**, использованный в этой команде, задает для

шаблона **boxxyerrorbars** в качестве первого и второго параметра колонки с номерами 1 и 2 файла with2.dat, а оставшиеся величины вычисляются по заданным в команде формулам. Так, например, абсцисса левой стороны прямоугольника является значением выражения \$1−0.1, где вместо \$1 берется значение, находящееся в первой колонке текущей строки.

4.2.4 Финансовые данные

Шаблон **financebars** используется для представления двумерных финансовых данных (например, результатов торгов на бирже). Требуется пять колонок данных. Первая колонка определяет значение абсциссы (обычно это дата), затем указываются значение открытия, минимальное, максимальное и закрывающее значения цен. При рисовании используется отрезок вертикальной линии, соединяющий максимальное и минимальное значение цен. Открывающее значение рисуется маленьким штрихом, направленным влево, а закрывающее – таким же штрихом, но направленным вправо. Длина штриха определяется параметром **bar**, который можно задать командой **set bar**. Пусть файл with6.dat содержит следующие пять колонок чисел

1 56 55 68 57
2 57 54 66 60
3 60 53 67 54
4 54 50 56 52
5 52 49 57 50

Приведем команды, создающие графическое представление этих данных (рис. 28).

Пример 4.11

```
set bar 5
plot [0:6] [47:69] "with6.dat"\
with fifancebars title "fifancebars"
```

Шаблон **candlesticks** используется для представления двумерных финансовых данных, представленных в том же формате, что и в случае стиля **fifancebars**. Для графического пред-

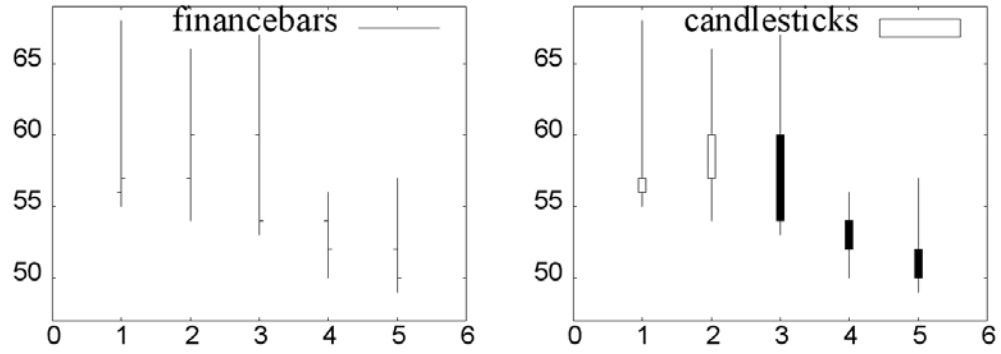


Рис. 28: шаблоны финансовых данных

ставления используется прямоугольник, центрированный относительно указанного значения x и ограниченный открывающей и закрывающей ценами. Между максимальными и минимальными значениями цен рисуется вертикальный отрезок. Ширина прямоугольника определяется значением параметра **bar**. Приведем пример графического представления (рис. 28) этим стилем данных, находящихся в файле with6.dat, с параметром **bar** равным 5.

Пример 4.12

**plot [0:6] [47:69] "with6.dat" with candlesticks **
title "candlesticks"

Заметим, что прямоугольник по умолчанию закрашивается, если закрывающая цена ниже открывающей, и остается без внутренней окраски в противном случае.

4.2.5 Гистограммы

Гистограммы можно рисовать лишь в двумерном случае. Для создания гистограммы требуется файл, в котором данные о каждой изображаемой величине хранятся в отдельном столбце (колонке). Перед построением нужно объявить, что данные будут использованы для построения гистограммы

set style data histogram

В Gnuplot 5.0 реализовано четыре различных вида гистограмм. Опишем их.

Первый тип гистограмм — это **кластерные диаграммы**

set style histogram clustered {gap <gapsize>}

По умолчанию устанавливается именно этот режим с **gap=2**.

Диаграмма строится следующим образом. Данные из k-ой строки используются для изображения кластера прямоугольников, который центрируется относительно точки с координатой равной k по горизонтальной оси. Высота прямоугольников задается числами, записанными в соответствующих колонках строки. Количество прямоугольников определяется в команде **plot** неявно — это число выбираемых столбцов. Ширина всех прямоугольников одинаковая. Промежуток между кластерами задается параметром **gap**. **Gap**, равный двум, означает, что это расстояние равно ширине двух прямоугольников. Все прямоугольники, относящиеся к одному типу данных, закрашиваются одним цветом или наполнителем (для черно-белых диаграмм).

Каждый тип данных определяется одним столбцом, а данные кластера считываются из одной строки. Часто в таких файлах данных первый элемент каждой строки является названием, которое может быть использовано для подписи соответствующего кластера через задание **xticlabels** в опции **using**.

Для дальнейших примеров будем использовать следующий файл с данными с именем 'his.dat'

month	A	A1	B	B1
may	1.	2.	7.	3.
june	3.	5.	4.	2.
july	4.	1.	2.	3.5
august	3.5	4.	3.	3.

Пример 4.13

```
set tmargin 1.5
set xtics offset 0,-0.4 font ",26"
set ytics font ",26"
set xrange [-1:]
```

```
set yrange [0:]
set boxwidth 0.9 relative
set style data histograms
set style fill solid 1.0 pattern border lt -1
set key title 'clustered' font ",32"
# Для выбора данных использована опция for
# (см. 11.5.1, с. 198)
plot for [COL=2:4:2] 'his.dat'\
using COL:xticlabels(1) title columnhead
```

Используя команды примера 4.13, получаем диаграмму, изображенную на рисунке 29.

Замечание. Команды не снабжены комментариями поскольку их применение уже было пояснено в примерах 1.2 и 1.10.

Второй тип гистограмм — это **errorbars**

set style histogram errorbars {gap <gapsize>}{<linewidth>}

Диаграммы в этом стиле очень похожи на кластерные диаграммы. Отличие заключается в дополнительно отображаемой информации об ошибке задания величин изображаемых данных. Для этого нужно задавать колонки со значениями ошибки. Как и в стилях, используемых для дискретных данных, дополнительных колонок требуется либо одна, если величина ошибки симметрична, либо две в противном случае. Внешний вид дополнительных элементов рисунка, предназначенных для изображения ошибок, можно менять при помощи комады **set bars**, задающей длину отрезка, и параметра <**linewidth**>, отвечающего за толщину линии.

Еще два типа гистограмм могут быть заданы командой

set style histogram {rowstacked | columnstacked}

В этих типах данные отображаются в виде прямоугольников, составленных из частей, доли которых определяются заданными значениями изображаемых величин. В случае типа **rowstacked** число прямоугольников равно количеству строк, а

число частей каждого прямоугольника определяется числом величин. В случае **columnstacked** число прямоугольников равно числу величин, а число частей — количеству строк.

Чтобы использовать первый столбец файла данных для генерации надписей в легенде в случае шаблона **columnstacked**, нужно выполнить команду

plot 'file.dat' using n:key(1)

Для задания подписей к делениям горизонтальной оси в виде элементов первой строки можно использовать команду

plot 'file.dat' using n title columnhead

Пример 4.14

```
set tmargin 1.5
set xtics offset 0,-0.4 font ",26"
set ytics font ",26"
set xrange [-1:]
set yrange [0:10]
set boxwidth 0.9 relative
set style histogram rowstacked
set style data histograms
set style fill solid 1.0 pattern border lt -1
set key title 'rowstacked' font ",32"
plot 'his.dat' using 2:xticlabels(1) title columhead,\
' ' using 4:xticlabels(1) title columhead
```

Пример 4.15

```
set label 1 "A" at 0,-1.8 font ",26"
set label 2 "B" at 1,-1.8 font ",26"
set bmargin 2
set tmargin 1.5
set xrange [0:]
set yrange [0:24]
set boxwidth 0.9 relative
set style data histograms
set style fill solid 1.0 pattern border lt -1
```

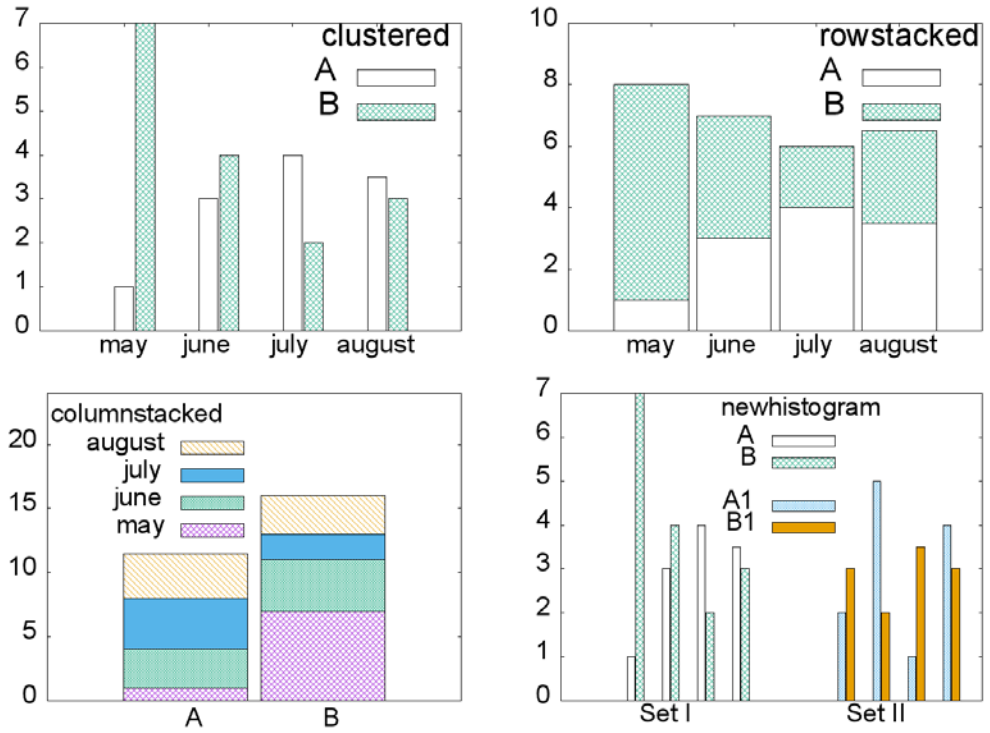


Рис. 29: шаблоны гистограмм

```
set key title 'columnstacked' at 0.53, 23 font ",26"
unset xtics
set ytics font ',26'
set style histogram columnstacked
plot 'his.dat' u 2:key(1) t columnheader, ' ' u 4
```

Используя команды примеров 4.14 и 4.15, получаем диаграммы, изображенные на рисунке 29.

В Gnuplot есть возможность нарисовать несколько гистограмм на одном графике. Для этого предусмотрен специальный шаблон **newhistogram**, имеющий следующий синтаксис:

```
newhistogram {"<title>" {font "name, size"}
              {tc <colorspec>}} {lt <linetype>}
              {fs <fillstyle>} {at <x-coord>}
```

Каждой гистограмме можно присвоить свое название, задаваемое ключом <**title**>. Это название выводится под горизонтальной осью на уровне диаграммы в виде подписи, местораспо-

ложение которой задается ключом **at**. Тип шрифта у названия, его размер и цвет определяются параметрами опции **font**. Ключи **lt** и **fs** задают тип линий и окрашивание у самой гистограммы. Следующий пример иллюстрирует использование шаблона **newhistogram**.

Пример 4.16

```
set bmargin 2
set tmargin 1.5
set spacing 0.8
unset xtics
set ytics font ",26"
set xrange [-1:]
set yrange [0:]
set boxwidth 0.9 relative
set style histogram cluster
set style data histograms
set style fill solid 1.0 pattern border lt -1
set xtic 1 offset character 0, 0.3
set key title 'newhistogram' at 8, 7 font ",26"
plot newhistogram "Set I" offset 0,-1.7 font ",26",\
'his.dat' using 2 title 'A', ' ' using 4 title 'B',\
newhistogram "Set II" offset 0,-1.7 font ",26",\
'his.dat' using 3 title 'A1', ' ' using 5 title 'B1'
```

Используя команды примера 4.16, получаем диаграмму, изображенную на рисунке 29.

4.2.6 Векторные поля

Шаблон **vectors** может быть полезен для графического представления двумерных (трехмерных) векторных полей (например, поля скоростей). При его использовании требуются четыре (шесть) колонки данных. В первых двух (трех) находятся координаты начала каждого вектора, а в оставшихся записана разница по каждой из координат между концевой и начальной точками. В обоих случаях можно добавить по одной колонке для задания цвета данного вектора. В результате рисуется вектор с началом в точке ($\mathbf{x}$, $\mathbf{y}$) и концом в ($\mathbf{x} + \delta\mathbf{x}$, $\mathbf{y} + \delta\mathbf{y}$). На конце

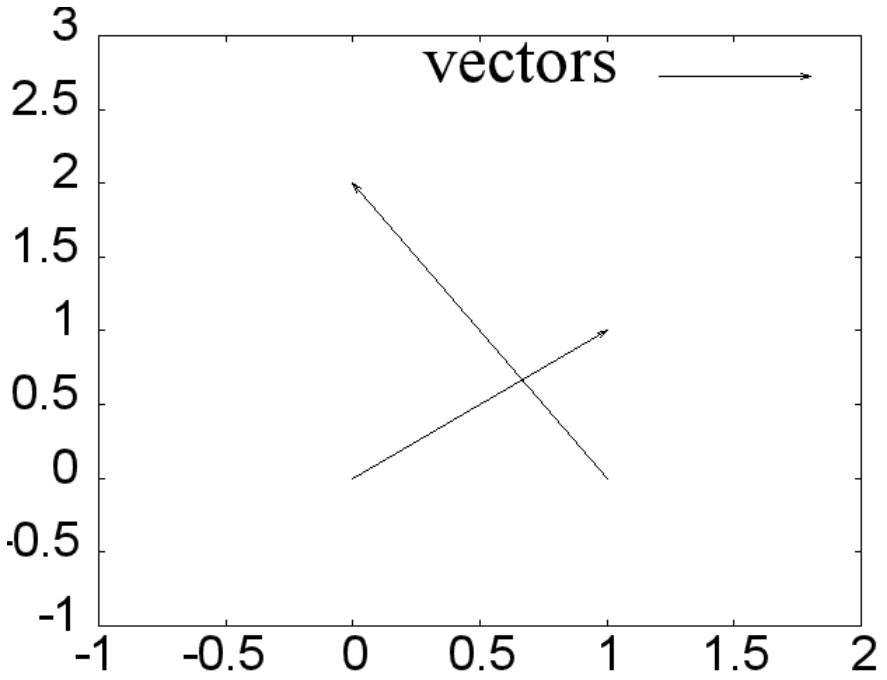


Рис. 30: шаблон vectors

вектора рисуется маленькая стрелка. Приведем на рисунке 30 графическое представление шаблоном **vectors** файла with7.dat, содержащего две строчки

0 0 1 1
1 0 -1 2

Это изображение было создано по команде.

Пример 4.17

plot [-1:2] [-1:3] ”with7.dat” with vectors title ”vectors”

4.2.7 Шаблон FILLEDCURVES

Шаблон **filledcurves** используется только для изображения графиков функций с одной независимой переменной. Возможны три варианта картинок. Первые два из них требуют задания одной функции либо с помощью формулы, либо через файл, содержащий минимум две колонки данных. В третьем случае требуется задание двух функций.

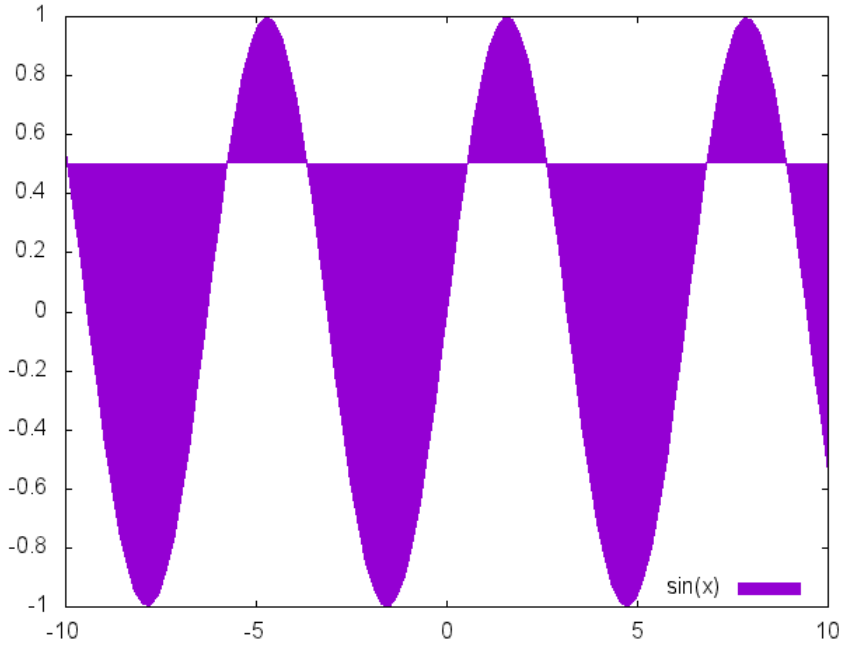


Рис. 31: шаблон filledcurves

Синтаксис команды **plot** при использовании этого шаблона следующий:

plot . . . with filledcurves [option]

Опции могут задаваться двумя возможными способами

[closed | { above | below }
{ x1 | x2 | y1 | y2 | r } | xy = <x>, <y>]

При заданном первом варианте ключа (**closed**) подразумевается, что функция ограничивает некоторую фигуру на плоскости (задает область), которая должна быть закрашена. Этот ключ считается заданным по умолчанию при указании только одной функции.

Во втором варианте закрашивается область между графиком и заданной либо осью координат, либо точкой. Координатная ось может быть заменена любой другой горизонтальной или вертикальной прямой.

Например.
1. **filledcurves closed** — закрашивается заданная область.
2. **filledcurves x1** — закрашивается область между кривой и осью x1.
3. **filledcurves y2=42** — закрашивается область между кривой и прямой, параллельной оси x и имеющей координаты по оси y2, равные 42.
4. **filledcurves xy = 10, 20** — заданная точка лежит в области, которую нужно раскрасить.
5. **filledcurves above r=1.5** — закрашивается заданная область, точки которой в полярных координатах имеют значения r большие 1.5.

Третий вариант, как уже было сказано, требует задания двух функций. При этом закрашивается область между кривыми, являющимися графиками заданных функций. При явном задании двух функций этот вариант шаблона используется по умолчанию.

Пример 4.18

plot sin(x) with filledcurves y1=0.5

График, созданный по команде примера 4.18, изображен на рисунке 31.

4.2.8 Априорный выбор шаблона

В Gnuplot есть возможность заранее задать шаблон, с помощью которого будут изображаться данные. Для функций, заданных формулой, синтаксис этой команды следующий

set style function <plotting-style>

Для дискретных данных эта команда имеет вид

set style data <plotting-style>

Например,

set style function lines
set style data linespoints

Если выполнена команда **set style ...**, то в командах **plot** и **splot** нет необходимости указывать ключ **with**.

Посмотреть каким шаблоном по умолчанию изображаются данные можно с помощью команд

show style function
show style data

4.3 Число используемых узлов (SAMPLES)

Узлом будем называть точку, для координат которой точно выполняется заданная функциональная связь. Графики по умолчанию строятся с использованием 100 узлов по каждой переменной. Число узлов может быть и меньше, если, например, файл с дискретными данными содержит меньшее число строк. От числа узлов зависит время построения графического изображения: чем больше число узлов, тем дольше будет строиться график. Большее число узлов обеспечивает большее совпадение построенного изображения графика с его точной формой.

Число узлов можно задать явно командой

set samples $<$ **samples_1** $>$ $\{, <$ **samples_2** $>\}$

Для одномерных графиков используется команда с одним аргументом, в двумерном случае задают либо два аргумента, задающие количество узлов по каждой из независимых переменных, либо один. Тогда по каждой переменной берется одинаковое число узлов, равное значению аргумента.

Для иллюстрации сказанного приведем график функции **sin(x)** (см. рис. 32), при построении которого было использовано всего 10 точек.

Пример 4.19

```
set xtics font 'times,18'
set ytics font 'times,18'
set samples 10
set key title font 'times,26
```

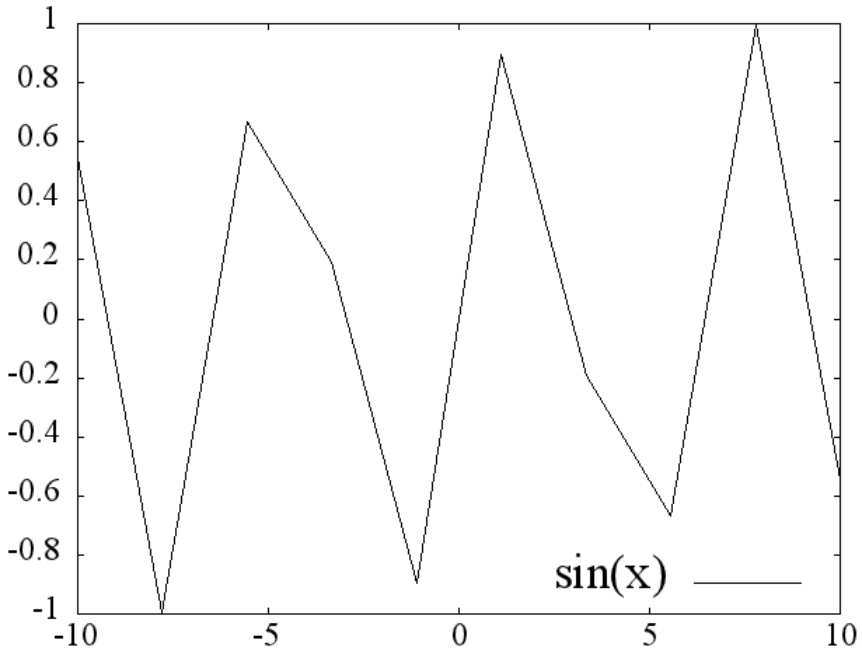


Рис. 32: график синуса, построенный по 10 узлам

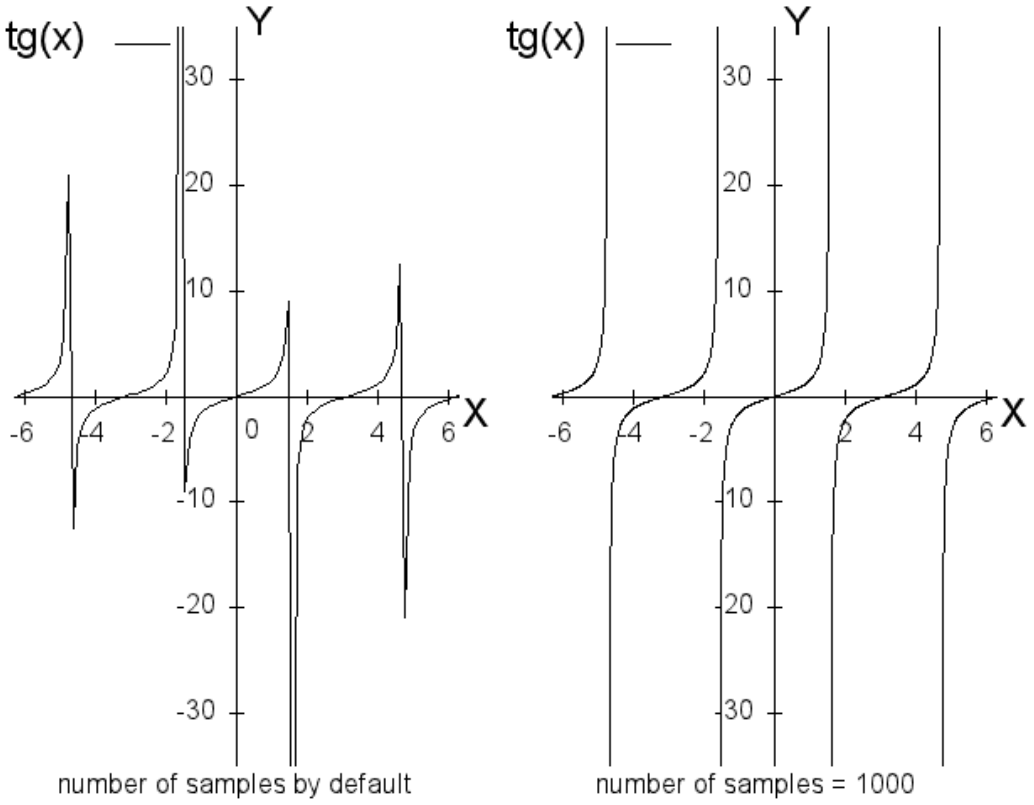


Рис. 33: графики **tg(x)** с числом узлов "по умолчанию" и 1000

set key right bottom
plot sin(x)

Заметим, что легенда графика была помещена в нижний правый угол изображения. Поскольку все остальные параметры были заданы по умолчанию, то на этом рисунке видно, как сказывается относительно небольшое количестово узлов на точности передачи особенностей поведения изображаемой функции (сравнить с рис. 8).

Заметим, что число заданных узлов игнорируется при использовании опции **smooth** в команде **plot**.

Вывести на экран заданное число узлов можно командой

show samples

Для иллюстрации того, к чему может привести игнорирование задания числа узлов, приведем два графика функции $\mathbf{y = tg(x)}$ на рисунке 33. Из левого графика видно, что недостаточное число узлов может привести к существенному искажению полученного изображения по сравнению с правильным.

4.4 Способ интерполяции (опция SMOOTH)

В Gnuplot реализовано несколько методов интерполяции и сглаживания обрабатываемых дискретных данных. Их применение обеспечивается заданием опции

smooth {unique | frequency | cumulative |
cnormal | kdensity | unwrap | csplines |
acsplines | mcsplines | bezier | sbezier }

Ниже будем называть независимую переменную x, а зависимую – y. Процесс построения графика начинается с упорядочивания всех имеющихся точек по значению независимой переменной. Дальнейшие действия зависят от выбранной опции.

Unique. При наличии нескольких точек с одинаковой координатой x: (x, y_1), $\ldots (x, y_k)$, они заменяются одной точкой с координатами (x, y), где $y = (y_1 + \ldots + y_k)/k$. После чего все полученные точки наносятся на координатную плоскость и соседние соединяются отрезками.

Frequency. При наличии нескольких точек с одинаковой координатой x: (x, y_1), $\ldots (x, y_k)$, они заменяются одной точкой с координатами (x, y), где $y = y_1 + \ldots + y_k$. После чего все полученные точки наносятся на координатную плоскость и соседние соединяются отрезками.

Cumulative. Для каждого значения x определяется одна точка. Ордината этой точки полагается равной сумме значений ординат всех точек из файла с данными с абциссой, не превышающей x. После чего все полученные точки наносятся на координатную плоскость и соседние соединяются отрезками. Эта опция может быть использована для построения комулятивного распределения, определяемого по заданным данным.

Cnormal. Для каждого значения x определяется одна точка. Ордината этой точки полагается равной сумме значений ординат всех точек из файла с данными с абциссой, не превышающей x, деленной на сумму ординат всех данных. После чего все полученные точки наносятся на координатную плоскость и соседние соединяются отрезками. Эта опция может быть использована для построения нормализованного комулятивного распределения, определяемого по заданным данным.

Kdensity. При применении этой опции данные, подлежащие обработке, считаются значениями случайной величины, для которой строится оценка ядерной плотности в виде гладкой гистограммы (kernal density estimate — KDE), использующая Гауссово ядро.

Unwrap. Входные данные модифицируются таким образом, чтобы у любых двух соседних точек значения координаты y не отличались больше, чем на π. Если для очередной точки это условие не выполняется, то ее координата по y уменьшается или увеличивается на 2π, пока условие не будет выполнено. Эта опция используется для фазовых и спектральных измерений.

Csplines. В отличие от опции unique получаемые точки графика соединяются с помощью натурального кубического сплайна.

Acsplines. Данная опция строит натуральный кубический "гладкий" сплайн. После упорядочивания точек, полученных из файла с данными, кривая составляется из кусков кубических

парабол, коэффициенты которых ищутся с учетом веса, присвоенного каждой точке. Вес может быть задан в третьем столбце файла с данными, либо в виде аналитической формулы. Например,

plot 'data.file' using 1:2:(1.0) smooth acsplines

Абсолютная величина веса является критерием использования каждого отдельного сегмента в итоговой кривой. Если величина веса точек на рассматриваемом сегменте велика, то итоговая кривая на нем будет являться натуральным кубическим сплайном. Если же веса точек сегментов маленькие, то кривая составляется из меньшего числа сегментов из условия увеличения гладкости. Предельным случаем является построение методом наименьших квадратов кубической параболы на всем отрезке. Сглаживающий вес может быть выражен в терминах ошибок: статистический вес точки, деленный на сглаживающий коэффициент для кривой. Поэтому величины стандартных ошибок значений, заданных в файле с данными, могут быть использованы как сглаживающие веса. Например,

$\mathbf{sw(x,S) = 1/(x * x * S)}$
plot 'data.file' using 1 : 2 : (sw($3, 100)) smooth acsplines

Mcsplines. Mcsplines — это кубические сплайны, которые хорошо передают свойства монотонности и выпуклости множества точек интерполяции [11].

Bezier и Sbezier. Эти опции позволяют соединять получаемые точки графика с помощью сплайнов Безье.

5 Оси координат и их оформление

5.1 Диапазоны изменения переменных

Опция **ranges** стоит первой в списке команды **plot**. Ее синтаксис допускает две формы

$$[\{<\mathbf{dummy-var}>=\}\{\{\mathbf{min}\}:\{\mathbf{max}\}\}]$$
$$[\{\{\mathbf{min}\}:\{\mathbf{max}\}\}]$$

Первая форма задает имя и диапазон изменения независимой переменной или параметра в случае параметрического задания функции; вторая — диапазон изменения зависимой переменной.

По умолчанию независимая переменная называется x, а зависимая имеет имя либо функции, либо формулы, задающей изображаемую зависимость, либо имя файла с использованными для построения данными. Диапазоны их изменения в случае аналитического задания функции Gnuplot выбирает по умолчанию сам, а для дискретных данных использует весь диапазон считываемых значений.

Имя независимой переменной можно задать по своему желанию. Например, в команде

$$\mathbf{plot}\ [\mathbf{t}=-\mathbf{pi}:\mathbf{pi}]\ [-\mathbf{1.3}:\mathbf{1.3}]\ \mathbf{sin}(\mathbf{t}),\ \mathbf{t}**\mathbf{2}$$

независимая переменная называется t, диапазон ее изменения $[-\pi;\ \pi]$, а ось ординат содержит значения $[-1.3;\ 1.3]$.

Диапазон изменения переменной по оси **X** можно явно изменять с помощью команды

$$\begin{aligned}&\mathbf{set\ xrange}\ [\{\{\mathbf{min}\}:\{\mathbf{max}\}\}]\{\{\mathbf{no}\}\mathbf{reverse}\}\\&\{\{\mathbf{no}\}\mathbf{writeback}\}\{\{\mathbf{no}\}\mathbf{extend}\}|\mathbf{restore}\}\end{aligned}\qquad(5.1)$$

Команды **yrange**, **zrange**, **x2range**, **y2range**, **cbrange**, **rrange**, **trange**, **urange** и **vrange** используются для задания диапазонов по соответствующим осям.

Параметры **{min}** и **{max}** необязательные. Они определяют нижнюю и верхнюю границы диапазона. В качестве значений для них может выступать число, переменная или же символ *.

В этом случае соответствующая граница будет вычисляться автоматически. Любое из значений **{min}** и **{max}** может быть опущено. Тогда используется значение, которое было установлено ранее, а если этого сделано не было, то берется значение, задаваемое автомасштабированием.

По умолчанию режим автомасштабирования установлен для всех осей. Отменить его можно командой

unset autoscale {<axes>}

Включается этот режим командой

set autoscale {<axes>{|min|max|fixmin|fixmax|fix} fix | keepfix }

Опция <axes> может принимать значения **x**, **y**, **z**, **cb**, **x2**, **y2** или **xy**.

При задании опций **{min}** и **{max}** диапазоны осей должны включать ближайшие деления оси такие, что интервал между этими делениями включает все координаты изображаемых точек графика. Добавление опций **fixmin**, **fixmax**, **fix** и **noextend** налагает запрет на выход границы диапазона за пределы используемых значений.

Примеры.
1. **set autoscale y** # включить режим автомасштабирования по оси **Y**.
2. **set autoscale ymin** # включить режим автомасштабирования по оси **Y** для выбора нижней границы диапазона. Верхняя граница диапазона оси Y и границы диапазонов остальных осей изменены не будут.
3. **set autoscale yfixmax** # включить режим автомасштабирования по оси **Y** для выбора верхней границы диапазона строго до максимального значения координаты 'y' среди использующихся точек графика.

Опция **reverse** меняет направление у осей, для которых включен режим автомасштабирования. При выключенном режиме автомасштабирования эта опция игнорируется.

Режим автомасштабирования включается в случае задания границы диапазона символом '*'. Можно ограничить выбираемое значение границы, задав условие

{<lb> < } * {< <ub>}

В этом случае определяемая граница будет задана из отрезка [lb, ub]. Его концы тоже могут служить значениями границы несмотря на строгие неравенства в условии! Эта опция может быть полезна для отсечения заведомо неверных данных, полученных, например, в результате ошибок при измерениях.

Примеры.

1. **set xrange [-10:10]** # Установить диапазон переменной по оси **X** в виде отрезка [-10; 10].
2. **set xrange [10:-10]** # Установить диапазон переменной по оси **X** в виде отрезка [-10; 10], задав обратное направление оси.
3. **set xrange [:20]** # Поменять верхнюю границу диапазона переменной по оси **X** на 20.
4. **set xrange [0<*:]** # Задать автоматическое определение нижней границы диапазона переменной по оси **X**, ограничив выбор неотрицательными значениями.
5. **set xrange [*<10:50<*]** # Установить диапазон переменной по оси **X** в виде отрезка, содержащего отрезок [10; 50].
6. **set xrange [-20<*<10:]** # Установить диапазон переменной по оси **X** с нижней границей от -20 до 10.

Опция **writeback** сохраняет в памяти компьютера найденный диапазон значений при автомасштабировании. Она может быть полезна, если требуется изобразить графики нескольких функций, использовав при этом диапазон одной из них. Опция **restore** позволяет использовать сохраненный диапазон.

Пример.

```
set xrange [-10:10]
set yrange [] writeback
plot sin(x)
set yrange restore
replot x/2
```

В результате диапазон по оси **Y** после изображения графика

sin(x) будет определен в виде отрезка [-1, 1] и сохранен в памяти компьютера. Перед изображением графика **x/2** этот диапазон будет считан из памяти и использован при выполнении команды **replot**. В результате на втором графике будут изображены лишь точки, у которых вторая координата по модулю не превосходит единицу.

Задание диапазонов изменения переменных в командах **plot** и **splot** действительно только для графиков, рисуемых по этой команде. Для изменения значений, выбираемых по умолчанию, нужно использовать команды типа (5.1). Диапазоны для осей x2 и y2 (см. 5.2, с. 113) могут быть заданы только командами

set x2range [{**min**} : {**max**}]
set y2range [{**min**} : {**max**}]

5.2 Расположение осей (опция AXES)

У команды **plot** есть опция **axes**. С ее помощью можно задавать расположение осей на рисунке. Возможны четыре значения опции:

x1y1 — задает стандартное расположение осей (ось абсцисс внизу, ординат слева);
x2y2 — ось абсцисс сверху, ось ординат справа;
x1y2 — ось абсцисс снизу, ось ординат справа;
x2y1 — ось абсцисс сверху, ось ординат слева.

При этом все равно рисуются все четыре оси, ограничивающие изображение с четырех сторон. Большие деления (см. 5.5, с. 117) наносятся на все четыре оси, а малые (см. 5.6, с. 120) и все подписи указываются лишь на заданных опцией осях.

Для удаления осей координат используется опция (см. 10.3.2, с. 176)

set border n

Каждой оси присваивается целое значение: нижней — 1, левой — 2, верхней — 4 и правой — 8. Значение **n** — это сумма значений изображаемых осей. Например, при n=1 показывается

только ось X1, при n=3 — оси X1 и Y1, все оси выводятся при n=31. Команда **set border** отвечает только за линии границ, так что деления остаются даже при n=0. Поэтому, чтобы стереть следы от осей, необходимо использовать команду (см. 5.5, с. 117)

set no{x|y}tics

В случае, когда дублеры делений на осях не нужны, можно использовать команду

set {x|y}tics nomirror

В Gnuplot есть команда, которая позволяет нарисовать любую ось более традиционно, а именно проходящей через начало координат. Синтаксис этой команды следующий:

set { x|x2|y|y2|z } zeroaxis { {linestyle | ls <line_style> }
| { linetype | lt <line_type> }
| { linewidh | lt <line_widh> } }

Смысл всех опций этой команды такой же, как и в разделе 4.2.2.

Для удаления соответствующих осей и просмотра заданных ранее опций предусмотрены две команды

unset {x|x2|y|y2|z} zeroaxis
show {x|y|z} zeroaxis

Значение опции **axes** может быть задано для каждой функции, но опция **ranges** в команде **plot** задает пределы изменения переменных только для первых осей.

По умолчанию значение опции **axes** задается **x1y1**.

5.3 Названия осей (опция XLABEL)

Подпись к оси абсцисс можно задать с помощью команды

set xlabel {"<label>"}
{offset <offset>} {font "<font>{,<size>}"}
{textcolor <colorspec>} {{no}enhanced}
{rotate by <degrees> | rotate parallel | norotate}

В интерактивном режиме может быть полезна команда, выводящая параметры заданной подписи

show xlabel

У команды **set** с ключами **x2label, ylabel, y2label, zlabel** и **cblabel** синтаксис аналогичный.

Параметр <**offset**> определяет смещение подписи. Его значение задается в зависимости от размерности графика двумя или тремя координатами. Координатная система может быть **first, second, graph, screen** или **character** (см. 10.2, с. 174). По умолчанию используется **character**. Например, команда

set xlabel offset -1, 0

задаст смещение подписи только по оси x влево на ширину одного символа. Ширина символа зависит от используемого шрифта и терминала.

Опция **font** определяет тип шрифта и его размер, которыми будет выполнена подпись (см. 10.4.2, с. 183).

Ключ **noenhanced** не меняет изображение подписи оси даже при включенном **enhanced text mode**.

Убрать подпись оси можно, выполнив команду **set xlabel** без указания опций.

По умолчанию заданы следующие значения параметров:

xlabel – подпись оси x находится снизу и центрирована;
ylabel – подпись оси y находится слева и центрирована, характер ориентации надписи (вертикальная или горизонтальная) зависит от типа терминала;
zlabel – подпись оси z центрирована относительно оси z и расположена сверху;
cblabel – название оси цветовой схемы центрируется относительно направления оси и располагается либо снизу, если ось горизонтальная, либо справа, если ось вертикальная;
y2label – подпись "правой" оси y находится справа, центрирована, ориентация зависит от типа терминала;
x2label – подпись "верхней" оси x находится по центру сверху, но ниже легенды.

Ориентация подписей осей **x, y, x2** и **y2** в случае двумерных графиков задается параметром **rotate by** <**degrees**>. Ориентация подписей осей **x** и **y** в случае трехмерных графиков по умолчанию горизонтальная, но при желании ее можно задать параллельно осям, задав специальный ключ **rotate parallel**.

5.4 Логарифмический масштаб (LOGSCALE)

По умолчанию Gnuplot задает линейный масштаб по каждой из осей координат. При необходимости его можно изменить на логарифмический. Для этого используется команда

set logscale <**axes**> {<**base**>}

Параметр <**axes**> может быть комбинацией ключей **x, x2, y, y2, z, cb** и **r** в любом порядке. Параметр <**base**> задает основание логарифмической шкалы. По умолчанию оно равно 10. Логарифмический масштаб после команды **set logscale** применяется на всех осях кроме **r**, если параметр <**axes**> опущен.

Например, команда

set logscale xz 2

устанавливает логарифмический масштаб по осям X и Z с основанием 2.

Отмена логарифмического масштаба осуществляется командой

unset logscale <**axes**>

В случае линейного масштаба все величины, характеризующие сдвиг относительно какой-либо точки, равны разности между конечным и начальным значением. Если же на оси выбран логарифмический масштаб, то эти величины равны отношению конечного и начального значений. Например, выполнены следующие команды

set logscale x
set arrow 100, 5 rto 10, 2 # Задать стрелку (см. 8.4, с. 158).

В результате будет задана стрелка из точки с координатами (100, 5) в точку (1000, 7).

Такое же правило определено и для шагов смещения, задаваемых в циклах.

5.5 Деления на осях (опция TICS)

Деления на оси x можно задать, воспользовавшись командой **set xtics**. Ее формат имеет вид

set xtics { axis | border} { {no}mirror } { in | out }
{ scale {default | <major> {,<minor>}}}
{{no}rotate {by <ang>}}
{ offset <offset> | nooffset}
{ left | right | center | autojustify } { add }
{ autofreq| <incr> | <start>, <incr> {,<end>} |
({"<label>"} <pos> {<level>} {,{"<label>"}. . .) }
{ format "formatstring" } { font "name{,<size>}" }
{ rangelimited } { textcolor <colorspec> }

Команды

unset xtics
show xtics

позволяют убирать деления на осях и показывать деления, установленные в момент запроса.

Аналогичные команды есть для задания делений на осях **y, z, x2, y2** и **r** (в полярных и сферических координатах).

Ключи **axis** и **border** размещают деления и подписи к ним вдоль осей или границы изображения соответственно. Отметим, что оси не обязательно совпадают с границами даже в случае двумерных графиков (см. 5.2, с. 113). Если ось координат находится вблизи границы, то нанесение заданных делений вдоль оси может проходить с ошибками.

Ключ **mirror** создает дубликаты основных (больших) делений заданной оси координат на ее дублере на противоположной границе окна. При этом подписи к этим делениям не создаются.

Ключи **in** и **out** определяют направленность делений: внутрь или наружу.

Ключ **scale** устанавливает размер делений. Деления бывают основные (большие) — major и маленькие — minor. Если параметр <**minor**> не задан, то он полагается равным **0.5***<**major**>. По умолчанию размер делений устанавливается равным 1.0 для больших делений и 0.5 для маленьких делений.

Ключ **rotate** определяет поворот текста на 90^o, **norotate** отменяет поворот и **rotate by angle** задает поворот на определенное число градусов.

По умолчанию заданы ключи **border mirror norotate** для осей X и Y, а для осей X2 и Y2 — **border nomirror norotate**. Для оси Z задание ключа **{axis | border}** невозможно и по умолчанию всегда **nomirror**. Для делений на оси Z возможно задание **mirror**, но после создания для них двоичного пространства с помощью команды **set border** (см. 10.3.2, с. 176).

Значения параметра <**offset**> (смещение) зависят от размерности системы координат и выбора координатной системы на экране (см. 10.2, с. 174). Задание <**offset**> смещает текст делений с их позиций по умолчанию. Задание <**nooffset**> отменяет установленный ранее <**offset**>. Например, команда

set xtics offset 0, graph 0.05

смещает метки делений ближе к графику.

По умолчанию подписи к делениям выравниваются автоматически в зависимости от выбранных осей и углов поворота. Если результат не устраивает, то коррекция их положения может быть выполнена за счет выбора значений **left**, **right** или **center**. Выбор значения **autojustify** возвращает к выбору по умолчанию.

Выполнение команды **set xtics** без опций восстанавливает настройки для изображения делений по умолчанию. Обратного действия эта команда не имеет.

Ключ **autofreq** позволяет расположить деления, используя для этого конструкцию цикла. Задание параметров <**start**>, <**end**>, <**incr**> устанавливает положения начального, конечного делений и расстояние между делениями. Если параметр **end** не задан, то считается, что он равен бесконечности. Параметр <**incr**> может быть отрицательным. Если параметр

<**start**> не задан, то он равен минус бесконечности. Если на оси был выбран логарифмический масштаб, то параметр <**incr**> используется как множитель (см. 5.4, с. 116).

На расположение делений влияет также значение установленного значения **'front'**, **'back'** или **'layerdefault'** командой **set grid** (см. 10.8, с. 191).

Примеры.

1. Установить метки на позиции 0, 0.5, 1, 1.5, ... , 9.5, 10.

set xtics 0, 0.5, 10

2. Установить метки на позиции ... , -10, -5, 0, 5, 10, ...

set xtics 5

3. Установить метки на позиции 1, 100, le4, le6, le8

set logscale x
set xtics 1, 100, le8

Явное указание параметров ("<**label**>"<**pos**><**level**>, **...**) позволяет задавать делениям, стоящим на произвольных позициях, любые, необязательно числовые, подписи. В этом случае для каждого деления указывается его положение и подпись. Текст подписи должен быть обязательно заключен в двойные кавычки. Если параметр "<**label**>" равен " ", то подпись ничего не содержит. При отсутствии этого параметра используется его значение по умолчанию, т.е. соответствующее число.

Третий параметр <**level**> означает тип деления. По умолчанию он равен 0, т.е. деление является <**major**> (большим). Значение параметра <**level**>, равное 1, означает, что деление будет <**minor**> (маленькое). Подпись соответствует выбранному размеру деления.

Примеры.

set xtics ("low" 0, "medium" 50, "high" 100)
set xtics (1,2,4,8,16,32,64,128,256,512,1024)
set ytics ("bottom" 0, "" 10, "top" 20)
set ytics ("bottom" 0, "" 10 1, "top" 20)

Во втором примере все деления будут подписаны. В третьем подписаны будут лишь концевые деления (первое и последнее). В четвертом неподписанным делением будет опять среднее и оно будет маленьким.

Если явно задаются деления, то автоматическая генерация не используется. Вернуть автоматическую генерацию делений можно, выполнив команду

set xtics auto

Можно добавлять к автоматической генерации делений произвольные метки, используя опцию **add**

set xtics 0, **.5**, **10**
set xtics add (″**Pi**″ **3.14159**)

Первая команда расставит деления с шагом 0.5 на всей оси, а вторая добавит еще одно деление на позиции 3.14159, сделав подпись **pi**.

Деления наносятся только в диапазоне, установленном командой **range**. Формат подписей у делений регулируется командой **set format** (см. 5.9, с. 126), если не использовано явное задание параметра <**label**>. Тип и размер шрифта определяются опцией **font** (см. 10.4.2, с. 183).

Маленькие деления без подписи могут быть автоматически сгенерированы с помощью команды **set mxtics** (см. 5.6, с. 120) или с помощью явного задания командой

set xtics (″ ″ < **pos** > **1**, ...)

5.6 Маленькие деления на осях (опция MTICS)

Маленькие (**minor**) деления на оси x можно задать явно, воспользовавшись командой **set mxtics**. Ее формат имеет вид

set mxtics {< **freq** > | **default**}

Команды

unset mxtics
show mxtics

позволяют убирать маленькие деления на осях и показывать установленные на момент запроса деления.

Аналогичные команды есть для задания маленьких делений на осях Y, Z, X2, и Y2.

Опция **freq** задает число подинтервалов на одном интервале между большими делениями. Таким образом, число маленьких делений всегда на одно меньше, чем значение параметра **freq**. По умолчанию это число равно двум или пяти. Выбор Gnuplot этого числа зависит от задания больших делений. При задании опции **default** число подинтервалов возвращается к выбираемому по умолчанию.

Для задания маленького деления в произвольном месте следует воспользоваться возможностями команды **set xtics** с ключом (”<**label**>”<**pos**><**level**>, ...), где параметр **level** равен 1. Малые деления могут быть заданы только в том случае, когда основные деления не задавались в ручную. Напомним, что дополнительные большие и маленькие деления могут быть добавлены к уже существующим командой **set xtics** с ключом **add** (см. 5.5, с. 117).

Пример. Команды

```
set xtics 0, 5, 10
set xtics add (7.5)
set mxtics 5
```

расставляют основные деления в позиции 0, 5, 7.5, 10, а малые в позиции 1, 2, 3, 4, 6, 7, 8, 9.

По умолчанию малые деления не ставятся для осей с линейным масштабом и, наоборот, включены в случае логарифмического масштаба. Малые деления наследуют от основных опции **axis|border** и **nomirror**.

5.7 Примеры использования двух пар осей

Визуальный эффект от выбора двух осей ординат можно оценить, представив данные файла antrop.dat с помощью команд.

Пример 5.1

```
# Легенду разместить в точке с координатами системы
# graph 0.8, 0.3, указав в качестве названий функций
# заголовки соответствующих столбцов.
set key autotitle columnhead at graph 0.8, 0.3
set xlabel "Age" # установить подпись "Age"на оси X1
# установить подпись "Height" на оси Y1
set ylabel "Height"
# установить подпись "Weight" на оси Y2
set y2label "Weight"
set y2tics # нанести деления на ось Y2
# Не рисовать дубликаты делений оси Y на оси y2.
set ytics nomirror
plot 'antrop.dat' using 3:1 axes x1y1 with linespoints, \
' ' using 3:2 axes x1y2 with linespoints
```

Результат их выполнения изображен на рисунке 34.

Другим примером использования двух пар осей является случай, когда у функций различные не только области значений, но и области определения. Рассмотрим функции

$$\mathbf{y = \sin(x),\ x \in [-3\pi;\ 3\pi]}; \qquad \mathbf{y = e^x,\ x \in [-0.5;\ 9]}.$$

На цветной иллюстрации VIII (с. 238) приведено такое изображение, полученное с помощью следующих команд.

Пример 5.2

```
# Задать положение легенды.
set key at graph 0.9, 0.93
# Деления осей не наносить на противоположную ось и подписи
# у делений осей X и Y сделать красными,
# а X2 и Y2 — зелеными.
set xtics nomirror tc "red"
set ytics nomirror tc "red"
set x2tics nomirror tc "green"
set y2tics nomirror tc "green"
# Задать число узлов для построения графиков.
```

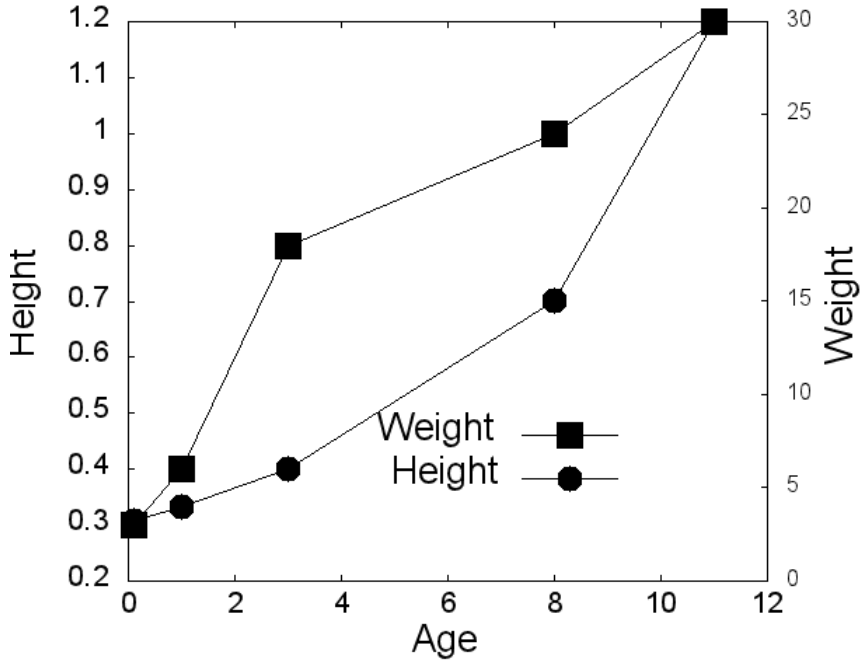


Рис. 34: выбор нескольких осей и задания их названий

```
set samples 1000
# Задать диапазоны изменения переменных.
set xrange [-3*pi:3*pi]
set yrange [-1.2:1.2]
set x2range [-0.5:9]
set y2range [0:9000]
# Задать деления на оси X2 от 0 до 8 с шагом 1.
set x2tics 0, 1 , 8
# Задать формат подписей на оси Y2.
set format y2 "%.0t*10^{%+03T}"
# Деления на оси Y2 задать с шагом 1000.
set y2tics 1000
# Задать подписи осей.
set xlabel "X (function sin(x))"
set ylabel "Y (function sin(x))"
set x2label "X (function e^x)"
set y2label "Y (function e^x)"
```

plot sin(x) with linespoints lt 3 lc "red" pi 30,
exp(x) axes x2y2 with linespoints lt 6 lc "green" pi 50

Замечание Можно использовать режим **multiplot** (см. 9, с. 165), чтобы нарисовать графики функций, имеющих разные области определения и значений. В этом случае можно сделать пары осей для этих функций разного цвета, но платой за это будет ручная подгонка совпадения областей изображений.

5.8 Временная ось

Часто для изображения данных требуется ось, по которой откладываются временные промежутки. Деления на временной оси создают командой

set {x|y|...}data time

Для работы с такими осями есть специальные переменные типа "время-дата" (см. 10.5.2, с. 187). Они имеют формат, задаваемый командой **set timefmt** (см. 10.5.2, с. 187).

Примеры.
1. Следующие команды создают временную ось с делениями "Dec 1", "Dec 3" и " Dec 5"

```
set x data time
set timefmt "%d/%m"
set xtics format "%b/%d"
set xrange ["01/12" : "06/12"]
set xtics "01/12", 172800, "05/12"
```

2. Заменив последнюю команду из примера 1 на

set xtics "01/12", "" "03/12", "05/12"

получим те же деления, но среднее, соответствующее позиции "Dec 3", будет неподписанным.

Приведем пример графика, где ось абсцисс является временной осью. Пусть в файле **"time.dat"** хранятся строки

03/27/15 10 : 00	**6.02e23**
03/28/15 11 : 00	**5.02e23**
03/29/15 12 : 00	**7.02e23**
03/30/15 13 : 00	**6.54e23**
03/31/15 10 : 30	**5.59e23**
04/01/15 11 : 30	**6.32e23**

График, изображенный на рисунке 35, получен командами.

Пример 5.3

```
# Задать поля окна для надписей.
set lmargin 14.0
set rmargin 4.0
set bmargin 4.0
# Задать размер шрифтов и положение легенды.
set key b font ',20'
set xtics font ',16'
set ytics font ',16'
# Стандартная надпись о времени создания изображения.
set timestamp offset 0,-0.5 font ',20'
set xdata time # Задать ось X, как временную.
set timefmt "%m/%d/%y" # Задать формат для времени.
# Диапазон временной оси.
set xrange ["03/27/15":"04/01/15"]
# Формат подписей у делений временной оси.
set format x "%m/%d"
set xtics 86400 # Расстояние между делениями временной оси.
# Формат чтения из файла значений времени.
set timefmt "%m/%d/%y %H:%M"
# В опции using указаны столбцы 1 и 3, поскольку значения
# времени, записанные в файле time.dat, расположены в первых
# двух столбцах: даты отделены от времени пробелами.
plot "time.dat" using 1:3 with lines
```

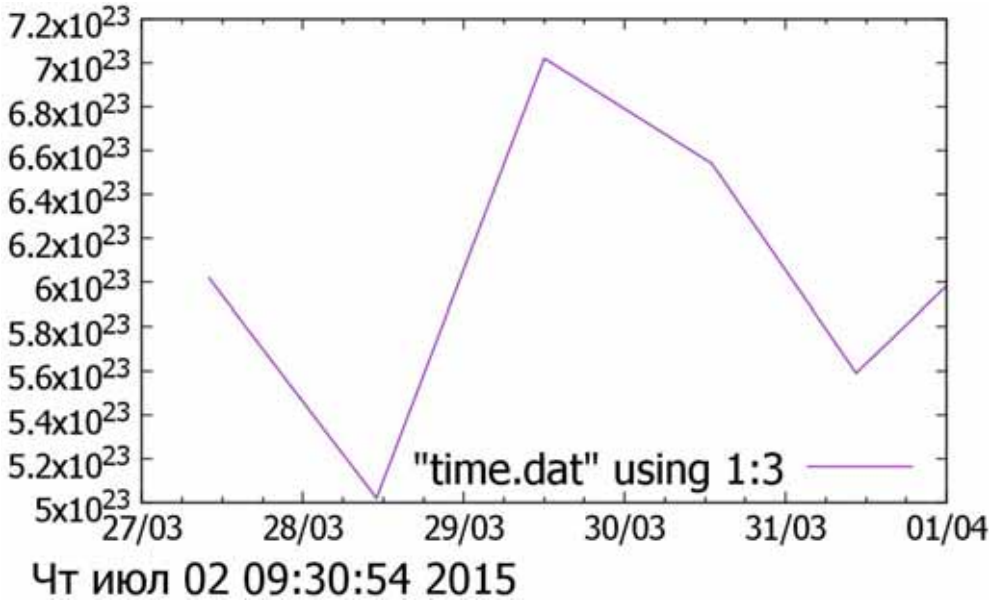


Рис. 35: пример использования временной оси

5.9 Формат подписей к делениям

Формат подписей к делениям на осях устанавливается командой

$$\textbf{set format } \{< \textbf{axes} >\} \ \{'< \textbf{format} - \textbf{string} >'\}$$

Допускается обрамление <**format-string**> двойными кавычками. Опция <**axes**> может принимать значения **x, y, xy, x2, y2, z, cb**. При опущенной опции <**axes**> задаваемый формат будет применен для всех осей.

Следующие две команды эквивалентны

set format y ″%.**2f**″
set ytics format ″%.**2f**″

Длина символьной строки, задающая подпись, ограничена 100 символами. Формат, задаваемый по умолчанию — ”%g”. Часто используются форматы ”%2f” и ”%3.0em” (см. 10.5.2, с. 187). Для системы LATEX, как правило, используется ”%g$”. Если в качестве ”<**format-string**>” будет задана пустая строка ” ”,

то деления будут иметь пустые подписи, которые оставят пустые места около делений. Для устранения этих "белых" пятен, можно использовать команды

unset xtics
set xtics scale 0

В формате допустим символ перехода на новую строку '\n'. В случае его использования следует применять двойные, а не одинарные кавычки, в которые заключается строка-формат (см. 11.3, с. 195). Символы, следующие за знаком '%', изображаются полностью. Например, при задании строки-формата "%g m" буква 'm' будет выводиться после каждого числа. Если вы хотите напечатать сам символ '%', то строка-формат должна быть следующей "%g%%" (см. 10.5.3, с. 189).

Посмотреть установленный формат можно с помощью команды

show format

6 Особенности 3D–изображений

6.1 Ракурс обзора (опция VIEW)

Опция **view** предназначена для задания ракурса обзора, наиболее удобного для нужного восприятия графиков функций двух переменных. Синтаксис команды, позволяющей задавать параметры этой опции, следующий:

set view
<rot_x>{,{<rot_z>}{,{<scale>}{,<scale_z>}}}

Параметры **<rot_x>** и **<rot_z>** задают угол поворота вокруг осей X и Z. Величины этих углов задаются в градусах. Значение **<rot_x>** выбирается в пределах от 0 до 180 (по умолчанию выбирается равным 60 градусам), а **<rot_z>** — от 0 до 360 (по умолчанию выбирается равным 30 градусам). Параметр **<scale>** определяет масштаб всего графика, в то время как **<scale_z>** масштабирует график только по оси Z. Оба параметра по умолчанию равны 1.

Примеры.

set view 60, 30, 1, 1
set view , , 0.5

Первая команда устанавливает все четыре параметра, равными их значениям по умолчанию. Вторая команда меняет масштаб изображения всего графика до 0,5.

Вывести на экран текущие значения параметров ключа **view** можно с помощью команды

show view

Для задания равных или неравных масштабов по осям применяется команда

set view {no}equal {xy|xyz}

Также для определения размеров изображения используется команда **set size** (см. 10.3.3, с. 178).

6.2 Число изолиний (опция ISOSAMPLES)

Напомним, что изолинией называется лежащая на поверхности кривая, для которой одна из независимых переменных фиксирована. Поверхности часто изображают в виде набора изолиний по обеим независимым переменным. По умолчанию число изолиний по каждой из переменных равно 10. Чем больше изолиний, изображающих поверхность, тем точнее передаются ее особенности.

Команда для явного задания числа изолиний выглядит следующим образом:

$$\textbf{set isosamples } < \textbf{iso\_1} > \{, < \textbf{iso\_2} >\}$$

Если число параметров 2, то каждый из них задает число изолиний по своей переменной. Когда же задан один параметр, то считается, что количество изолиний по каждой из переменных одинаково и равно значению заданного параметра.

Для иллюстрации сказанного приведем график параболоида (см. рис. 36)

$$\mathbf{z} = \frac{\mathbf{x^2}}{\mathbf{4}} + \frac{\mathbf{y^2}}{\mathbf{4}}$$

При построении этой поверхности было использовано 20 изолиний по переменной x и всего 5 по переменной y:

Пример 6.1

```
set key font ',20'
set xtics offset 0,-0.5 font ',16'
set ytics offset 1,-0.25 font ',16'
set ztics font ',16'
set isosamples 20, 5
splot x*x/4+y*y/4
```

На этом рисунке видно, как сказывается на изображении разное количество изолиний по независимым переменным (сравнить с рис. 12).

Заметим, что задание числа изолиний игнорируется при изображении поверхностей, задаваемых с помощью дискретных данных.

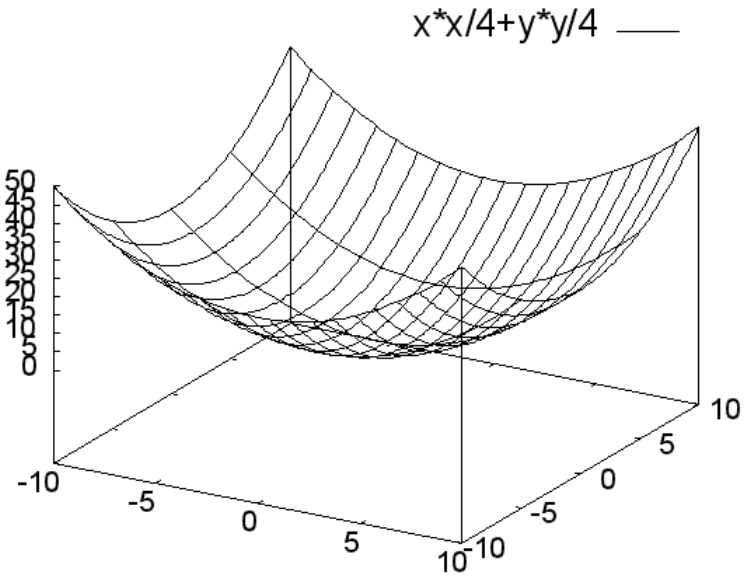


Рис. 36: график параболоида с разным числом изолиний

Вывести на экран заданное число изолиний можно командой

show isosamples

6.3 Скрытые линии (опция HIDDEN3D)

График поверхности обычно состоит из набора линий в пространстве, каждая из которых является ее плоским сечением. Не все участки этих кривых оказываются видны, если считать поверхность непрозрачной. Возможность удаления таких скрытых участков обеспечивает режим стирания невидимых линий, который задается с помощью команды

set hidden3d { defaults } |
{ {front|back}
{ {offset <offset>} | {nooffset } }
{ trianglepattern <bitpattern> }
{ { undefined <level> } | {noundefined } }
{ {no}altdiagonal }
{ {no}bentover }

Режим **hidden3d** до версии 4.6 работал лишь для поверхностей, заданных данными, которые можно считать сеткой (**grid data** (см. 2.3.6, с. 53)), и изображаются с помощью шаблонов **lines** или **linespoints**. Начиная с версии 4.6, под действие **hidden3d** стали попадать шаблоны **points**, **labels**, **vectors** и **impulses** даже в случаях, когда изображаемая поверхность не является традиционной, т.е. изображаемой сеткой изолиний. С помощью режима **hidden3d** можно нарисовать изображения одновременно двух и более поверхностей.

Одна и та же поверхность может рисоваться немного по разному в режимах **hidden3d** и **nohidden3d**, поскольку для изображения изолиний может быть задействовано разное число точек, задающих поверхность (**samples** (см. 4.3, с. 105)).

Изображения всех надписей (**labels**), стрелок (**arrows**) и содержимое легенды (**key**) никогда не пропадает из-за включенного режима **hidden3d**.

Ключ **offset** задает **linetype** линий для нижней стороны поверхности. По умолчанию он на единицу больше, чем для верхней стороны. Это делает более различимыми стороны поверхности. Можно задать **linetype** для нижней стороны в ручную, указав параметр <**offset**>. Обе стороны будут изображены одинаково, если задать <**offset**> равным нулю или указать ключ **nooffset**.

Ключ **trianglepattern** задает способ изображения различных элементов, на которые разбивается поверхность. Параметр <**bitpattern**> может принимать целые значения от 0 до 7. По умолчанию задается значение 3 (см. раздел hidden3d [1]).

Ключ **undefined** определяет, как поступать с точками, у которых какие-то координаты определены некорректно (например, пропущены или имеют значение nan) или не входят в диапазоны, определенные опцией **ranges**. В таких точках либо ничего не рисуется, либо они выбрасываются из множества данных и на поверхности возникает дырка. Задание параметра <**level**>, равного 3, не выбрасывает никаких точек. Это значение задается также выбором ключа **noundefined**. Параметр <**level**>, равный 2, выбрасывает только точки с некорректно определенными координатами, а при параметре <**level**>, рав-

ному 1 (это значение задается по умолчанию), удаляются точки, чьи координаты не попадают в заданные диапазоны.

Про остальные ключи опции **hidden3d** можно прочитать в одноименном разделе [1].

Отменить режим стирания скрытых линий и посмотреть установленные для этого режима ключи можно командами

unset hidden3d
show hidden3d

6.4 Положение плоскости XY (XYPLANE)

По умолчанию плоскость, где изображены оси XY, проводится ниже нижней границы допустимых значений координаты z. Напомним, что эта граница задается либо явно определением опции **zrange**, либо по умолчанию как минимальное значение координаты z по всем изображаемым точкам. Как правило, такое расположение не мешает правильно воспринимать изображенные графики и не создает лишних помех. Однако бывают случаи, когда требуется явно задать положение плоскости XY. Это можно сделать, использовав команду

set xyplane at <zvalue>

<**zvalue**> — точка на оси Z, в которой будут пересекаться оси X и Y.

Еще одна команда

set xyplane relative <frac>

задает расстояние от плоскости XY до z_{min} равным величине $z_{max} - z_{min}$, умноженной на величину <frac>. По умолчанию значение <frac> равно 0.5. Допускаются отрицательные значения <frac>, но при этом подписи делений на осях могут перекрываться.

Пример.
Провести плоскость XY через точку 0 на оси Z

set xyplane at 0

Показать установленную точку пересечения плоскости XY с осью Z можно, использовав команду

show xyplane

6.5 Соответствие между системами координат

При создании трехмерных изображений в цилиндрических или сферических координатах требуется задать инструкцию, поясняющую как интерпретировать дискретные данные

set mapping {cartesian|spherical|cylindrical}

По умолчанию задается **cartesian** — декартова система.

В сферическом случае данные должны содержать две или три колонки. Если колонок три, то первая — угол θ, вторая — угол ϕ, третья — радиус-вектор точки $\mathbf{r}$. В случае двух колонок углы берутся из имеющихся столбцов, а значение радиуса полагается равным единице для всех точек. Приведем формулы соответствия данных этих колонок декартовым координатам

$$\begin{aligned} \mathbf{x} &= \mathbf{r} * \mathbf{cos}\,(\phi) * \mathbf{cos}\,(\theta), \\ \mathbf{y} &= \mathbf{r} * \mathbf{sin}\,(\phi) * \mathbf{cos}\,(\theta), \\ \mathbf{z} &= \mathbf{r} * \mathbf{sin}\,(\theta). \end{aligned}$$

Требуется подчеркуть, что система координат $(\theta, \phi, \mathbf{r})$ является "географической", а не "сферической". Угол θ в ней задает угол между "экватором" и радиус-вектором точки: $\theta \in \left[-\frac{\pi}{\mathbf{2}}; \frac{\pi}{\mathbf{2}}\right]$. Полярный угол ϕ может принимать любые значения из отрезка $[\mathbf{0}; \mathbf{2}\pi]$.

Если данные задают поверхность в цилиндрических координатах, то столбцов с данными также может быть два или три. Первые два задают угол ϕ и координату $\mathbf{z}$. Третий столбец содержит значения радиус-вектор $\mathbf{r}$ точек, а при его отсутствии $\mathbf{r}$ полагается равным единице.

Формулы соответствия данных этих колонок декартовым координатам следующие

$$\begin{aligned} \mathbf{x} &= \mathbf{r} * \mathbf{cos}\,(\phi) \\ \mathbf{y} &= \mathbf{r} * \mathbf{sin}\,(\phi) \\ \mathbf{z} &= \mathbf{z}. \end{aligned}$$

Заметим, что опция **mapping** не влияет на работу команды **plot**.

7 Цветные 3D-изображения

7.1 Шаблон PM3D

Для создания цветных пространственных изображений предназначен шаблон **pm3d**. Этот шаблон используется только командой **splot**. Изображения могут быть как цветными, так и в черно-белых тонах. Используемый алгоритм допускает изображение данных как с одинаковым числом точек на срезе, так и с переменным без предварительной обработки.

На цветной иллюстрации XI (с. 240) приведен пример изображения параболоида с помощью шаблона **pm3d**

splot x * x/4 + y * y/8 with pm3d

Остальные команды, создающие это изображение, остались такими же, что и в примере 2.5, а опции шаблона **pm3d** были заданы по умолчанию.

7.2 Выбор цвета

Многие команды Gnuplot допускают явно задавать цвет изображаемых объектов. Синтаксис опций, позволяющих определить цвет, имеет вид:

... {**linecolor** | **lc**} {< **colorspec** > | < **n** >}
... {**textcolor** | **tc**} {< **colorspec** > | {**linetype** | **lt**} < **n** >}

Значения <**colorspec**> можно указать многими способами (см. раздел **colorspec** [1]). Приведем некоторые из них

rgbcolor "colorname" — например, "blue".
rgbcolor "#RRGGBB" — строка шестнадцатеричных констант, задающих цвет в формате X11.
rgbcolor variable — целочисленная переменная (ее значение считывается из файла с данными).
palette frac <val> — <val> из отрезка от 0 до 1 для действующей палитры.

palette cb <value> — <val> из диапазона **cbrange** действующей палитры.
<n> — цвета линии с номером n (см. цв.иллюстр. V).
bgnd — задание цвета фона.
black — задание черного цвета.

Значения **"#RRGGBB"** являются шестнадцатеричными константами с префиксом '#'. Каждая из констант задает три цвета — красный, зеленый и синий. Оттенки каждого из них задаются целым числом от 0 до 255.

Ключ **"colorname"** может принимать значения, которые можно узнать, применив команду

show colornames

Приведем несколько примеров таких имен и их кодировки

orange	**#ffa500 = 255 165 000**
brown4	**#801414 = 128 020 000**
gray	**#bebebe = 190 190 190**

Цветовая палитра ставит в соответствие каждому оттенку из предлагаемого набора индивидуальное числовое значение. В Gnuplot есть две цветовые палитры. Первая это **palette frac**. В ней каждому цвету ставится дробное число от 0 до 1 (см. цв.иллюстр. VI). Вторая — **palette cb**. В этой палитре устанавливается диапазон цветовых значений командой

set cbrange

Ключ **"palette z"** связывает диапазон координат по оси Z с диапазоном цветовых значений (**cbrange**). Это позволяет гладко менять цветовые оттенки в зависимости от величины координаты по оси Z. Этот параметр позволяет раскрашивать двумерные графики, используя для этого дополнительную колонку данных.

7.3 Цветовая схема

Для иллюстрации зависимости цвета элемента поверхности от координаты по оси Z при использовании шаблона **pm3d** служит цветовая схема. Цветовая схема это прямоугольник, в котором либо по горизонтали справа налево, либо по вертикали снизу вверх изменяются цвета по тому же закону, что и для поверхности от z_{min} до z_{max} (см. первый ключ команды задания цветовой схемы). Вывод цветовой схемы на поле изображения задается командой

set colorbox { { vertical | horizontal }
{ default | user } { origin x, y }
{ size x, y } { front | back }
{ noborder | bdefault | border [line style] } }

Размеры и положение прямоугольника цветовой схемы может быть задано либо пользователем, либо по умолчанию (ключи **user** и **default**). При выбранном ключе **user** можно задать положение левого нижнего угла прямоугольника (**origin x, y**, где x и y координаты этого угла) и длины его сторон (**size x, y**). В этих опциях параметры x и y по умолчанию задаются в системе координат **graph** (см. 10.2, с. 174). Например,

set colorbox horiz user origin .1, .02 size .8, .04

Ключи **back** и **front** определяют порядок изображения цветовой схемы относительно графика. В первом случае график будет накладываться на прямоугольник схемы, а во втором наоборот.

При выбранном ключе **noborder** прямоугольник цветовой схемы будет изображен без окантовки. Ключ **bdefault** означает, что граница прямоугольника будет изображена линией с параметрами, принятыми по умолчанию. Наличие ключа **border** без параметра также означает окантовку по умолчанию. Если после ключа **border** указано целое число, то окантовка будет типом линии, соответствующей этому числу (см. 10.4.1, с. 181).

Например,

set style line 2604 linetype −1 linewidth 2
set colorbox border 2604

В этом примере граница прямоугольника будет изображена линией типа (-1) шириной 2 (см. цв.иллюстр. V).

Цветовая схема может быть снабжена осью координат. Она называется **cb**. Для этой оси действителен такой же набор команд, как и для обычных осей: **cbrange, [m]cbtics, format cb, grid [m] cb, cblabel, cbdata, [no]cbtics, [no]cbmtics** (см. 5, с. 110).

Команда **set colorbox** без указания ключей возвращает параметры цветовой схемы к принятым по умолчанию.

Посмотреть на параметры цветовой схемы и отменить ее вывод можно с помощью команд **show colorbox** и **unset colorbox**.

7.4 Настройки шаблона PM3D

При применении шаблона **pm3d** можно использовать ряд опций, которые задаются с помощью команды

set pm3d { { at <position> }
{ interpolate <steps/points in scan, between scans> }
{scansautomatic | scansforward | scansbackward |
depthorder} { flush { begin | center | end } }
{ ftriangles | noftriangles } { clip1in | clip4in }
{ corners2color { mean | geomean | median
| min | max | c1 | c2 | c3 | c4 } }
{ hidden3d { <linestyle> } | nohidden3d }
{ implicit | explicit } { map } }

Задание опций допускается в любом порядке.

Показать заданные опции шаблона **pm3d** и вернуться к задаваемым по умолчанию настройкам можно, использовав команды **show pm3d** и **unset pm3d**.

График будет изображен в цвете, если при задании шаблона **pm3d** будет указана опция **implicit**. Например, оба графика

будут цветными в результате выполнения команд

set pm3d implicit
splot 'file1.dat' with lines, 'file2.dat' with lines

Если опция **implicit** не была упомянута при задании шаблона **pm3d** или была явно задана опция **explicit**, то данные изображаются непосредственно теми шаблонами, которые указаны в команде **splot**. Например, пусть выполнена одна команда

splot 'file1.dat' with lines, 'file2.dat' with pm3d

В результате первый график будет изображен шаблоном **lines** , а второй — шаблоном **pm3d**.

По умолчанию устанавливается опция **explicit**, но для совместимости с предыдущими версиями использование команды **set pm3d** без опций или же **set pm3d at X ...** переводит Gnuplot в режим **implicit**. Первая команда оставляет остальные настройки в состоянии "по умолчанию".

Еще одна возможность для изображения дискретных данных с помощью шаблона **pm3d** предоставляется командой **set style data pm3d** . После выполнения этой команды все графики изображаются шаблоном **pm3d** независимо от опций **implicit** или **explicit**.

Заметим, что при изображении нескольких графиков в одних осях координат каждый рисуется согласно ключам, указанным в командной строке. Поскольку первый рисуется сначала, то второй, если он закрашен, может закрывать части первого и т.д. Для того, чтобы избежать перекрытий, шаблон **pm3d** имеет возможность рисовать графики в трех разных позициях (ключ <**position**>): **top** — в виде цветной проекции на верхнюю грань прямоугольного параллелепипеда, содержащего изображение (см. 10.3.2, с. 176), **bottom** — то же, только на нижнюю грань, и **surface** — в виде поверхности. Сокращенная запись: **at {t}|{b}|{s}**. Например, на цветной иллюстрации XII (с. 240) изображен результат выполнения команд.

Пример 7.1

set border 4095

set pm3d at s
splot 10*x with pm3d at b,
x*x-y*y, x*x+y*y with pm3d at t

Поверхность, изображаемая шаблоном **pm3d**, разбивается на кусочки прямоугольной формы. Они рисуются в том порядке, который определяется сеткой узлов, задающей поверхность. От этого порядка зависит то, какие кусочки будут закрывать другие. Для изменения порядка используют ключи **scansforward** и **scansbackward**. По умолчанию используется ключ **scansautomatic**, при котором Gnuplot сам выбирает порядок изображения кусочков поверхности. Заметим, что эти ключи не совсем совместимы с алгоритмом удаления невидимых частей поверхности.

Если два последовательных среза не имеют равного числа узлов, то в этом случае требуется определить с какого конца следует начинать упорядочивать узлы: от начала срезов **flush begin**, от их конца **flush end** или от их центра **flush center**. Заметим, что опции **flush begin|end** несовместимы с ключом **scansautomatic**. Если они были выбраны, то ключ **scansautomatic** автоматически заменяется на **scansforward**.

Если два последовательных среза имеют разное число узлов, то опция **ftriangles** позволяет определить нужно ли на конце среза рисовать цветной треугольник. Это позволяет добиваться гладкости границы поверхности.

Для верного изображения цветных поверхностей шаблоном **pm3d** рекомендуется задавать следующий режим

set pm3d depthorder hidden3d

Здесь опция **depthorder** верно располагает непрозрачные кусочки поверхности, а **hidden3d** удаляет граничные линии этих кусочков (если они рисуются). Отметим, что команда **set hidden3d** не действует для **pm3d**-поверхностей.

При заданной опции **interpolate m, n** Gnuplot строит поверхность на более мелкой сетке, применяя интерполяцию. Для дискретных данных это приводит к более гладкому переходу

цветов и выделению острых выступов. В случае задания поверхности формулой это не имеет большого смысла, разве что позволяет более аккуратно расходовать память. В этом случае более целесообразно воспользоваться опциями **samples** и **isosamples**. Для положительных **m** и **n** каждый кусочек поверхности дробится на m и n частей по каждой координате соответственно. Для отрицательных значений **m** и **n** интерполяционная последовательность выбирается так, что будет использовано по меньшей мере |**m**| и |**n**| точек по каждому направлению. Команда

interpolate 0, 0

автоматически выберет оптимальное число разбиений.

В качестве подтверждения сказанного приведем изображение параболоида (цв.иллюстр. XIII), для которого было использовано большое количество изолиний (по 100 для каждой переменной). В результате пропадает "лоскутность" изображения, которая бросалась в глаза на цветной иллюстрации XI (с. 240).

Ключи **clip1in** и **clip4in** устанавливают способ обрезания края поверхности. При заданном **clip1in** граничный кусочек включается в изображение, если хотя бы одна из угловых точек попадает в заданную область изменения независимых переменных. При ключе **clip4in** условие включения — попадание всех четырех точек в область.

Часто для иллюстрации полученных численных результатов требуется нарисовать цветную проекцию на плоскость XY трехмерной поверхности. Приведем для примера команды, которые изображают такую проекцию в случае параболоида, заданного на единичном круге.

Пример 7.2

```
set view 360, 270  # Задание ракурса обзора.
# Установить одинаковый масштаб по осям X и Y.
set view equal xy
# Задание цветовой схемы.
set colorbox horizontal user origin .3, .2 size .4, .04
set cbtics 0, 0.2, 1  # Задать деления на цветовой схеме.
set isosamples 1000, 1000  # Задать число изолиний.
```

```
set border 15  # Задание границ изображения.
unset ztics  # Удалить подписи делений оси Z.
f(x,y)=(x*x+y*y<=1)?x*x+y*y:1/0
splot [-1.:1.] [-1.:1.] f(x,y) with pm3d at b title "z=x*x+y*y"
```

Результат их выполнения можно видеть на цветной иллюстрации XIV (с. 241).

7.5 Палитра

7.5.1 Команда задания

Палитра является хранилищем используемых цветов в шаблоне **pm3d**, который применяется для цветного изображения поверхностей, контуров, гистограмм и других объектов, где важны плавные переходы цвета. Задание палитры осуществляется командой

```
set palette { { gray | color } { gamma <gamma> }
    { rgbformulae <r>, <g>, <b> | defined
    { ( <gray1> <color1> {, <grayN> <colorN> }
    ...) } | file '<filename>' { datafile-modifiers }
    | functions <R>, <G>, <B> }
    { cubehelix { start <val> } {cycles <val> }
    { saturation <val> } }
    model { RGB | HSV | CMY | YIQ | XYZ } }
    {positive | negative } { nops_allcF | ps_allcF }
    { maxcolors <maxcolors> } }
```

Команда **set palette** без опций устанавливает цвета палитры, задаваемые по умолчанию. Команда **show palette** показывает параметры действующей палитры. Команда **show palette gradient** показывает градиент, определяющий палитру, когда он задан.

Команда **show palette palette <n>** выводит таблицу RGB-троек для заданной палитры, имеющей <n> дискретных цветов. По умолчанию таблица содержит 3 колонки чисел из отрезка

$[0,1]$ или целых чисел из диапазона $[0,255]$ в зависимости от того, какой ключ был задан **float** или **integer**.

Все изображамые объекты могут быть как цветные, так и черно-белые, где цвета отличаются оттенками серого. Опция **gray** переключает палитру в черно-белый режим. Команда

set palette color

возвращает режим применения последней из использовавшихся цветных палитр.

Если используется шаблон **pm3d**, то оттенок серого для каждого маленького прямоугольного кусочка поверхности получается осреднением его координат по оси Z и последующим определением оттенка, исходя из формулы перевода отрезка $[z_{min}, z_{max}]$ в отрезок $[0,1]$, состоящий из градаций серых тонов. Таким образом получаются карты оттенков серого. Для того, чтобы получить цветную карту, используется принятый закон, переводящий карту серых тонов в цветную

$$\mathbf{[0,1] \to ([0,1],\ [0,1],\ [0,1])}$$

Закон перевода задается либо аналитической формулой, либо дискретной таблицей соответствия с последующей интерполяцией.

Опции **rgbformulae**, **functions** и **cubehelix** определяют функциональный способ задания цветовой карты, а **defined** и **file** это делают через дискретные таблицы.

Многие терминалы поддерживают только дискретное число цветов (для большинства это число равно 256). После определения цветов по умолчанию лишние оттенки использоваться не будут. Поэтому в ряде случаев при использовании нескольких палитр возможны ошибки при изображении. В этом случае полезной командой оказывается

set palette mzxcolor <N>

Эта команда делит непрерывный цветовой спектр на N дискретных интервалов. После чего из каждого интервала выбирается свой цвет, который и будет использован для изображения. Если

деление на интервалы нужно сделать неравномерным, то следует использовать команду

set palette defined

В палитре допускается использование различных цветовых моделей.
1. **RGB** (red, green, blue) — красный, зеленый, синий — аддитивная цветовая модель, используемая в большинстве цветовых аппаратных решений, включая мониторы.
2. **HSV** (hue, saturation, value) — тон, насыщенность, значение — модель является нелинейным преобразованием RGB модели. Часто эту модель используют художники, потому что они считают, что устройство HSV ближе к человеческому восприятию цветов.
3. **CMY** (cyan, magenta, yellow) — голубой, пурпурный, желтый — субстрактная схема формирования цвета, используемая прежде всего в полиграфии для стандартной триадной печати.
4. **YIQ** — цветовая модель, в которой цвет представляется как три компоненты: яркость (Y) и два искусственных цветоразностных сигнала (I и Q). Сигнал I называется синфазным, Q - квадратурным. Модель применяется в телевещании по стандартам M-NTSC и M-PAL, где полоса частот видеосигнала заметно меньше, чем в других телевизионных стандартах.
5. **XYZ** — линейная трехкомпонентная цветовая модель, основанная на использовании трех основных цветов RGB, получена на основе усреднения статистических результатов цветовой чувствительности характеристик человеческого глаза.

При использовании цветовой модели, отличной от RGB, следует использовать команду **set palette defined** для определения таблиц цетовых оттенков для достижения лучшего результата. Все значения параметра **values** ограничены отрезком $[0, 1]$.

7.5.2 RGB-формула

При выборе опции **rgbformulae** требуется задать три формулы, которые будут определять для каждой точки значения трех цветов RGB по относительному положению ее координаты z на

диапазоне $[z_{min}, z_{max}]$. Каждая формула задается ее номером в списке доступных формул. Всего их 37. Номера начинаются с нуля и заканчиваются 36-ым. Если перед номером стоит знак '-', то это означает инвертирование цветовой компоненты. Тем самым в команде **set pm3d rgbformulae** параметры могут иметь значения от -36 до 36.

Команда

show palette rgbformulae

выводит список доступных формул перевода серой палитры в цветную:

0. $\mathbf{0}$.	1. $\mathbf{0.5}$	2. $\mathbf{1}$.
3. $\mathbf{x}$	4. $\mathbf{x^2}$	5. $\mathbf{x^3}$
6. $\mathbf{x^4}$	7. $\sqrt{\mathbf{x}}$	8. $\sqrt[4]{x}$
9. $\mathbf{sin(90x)}$	10. $\mathbf{cos(90x)}$	11. $\mathbf{\lvert x - 0.5\rvert}$
12. $\mathbf{(2x-1)^2}$	13. $\mathbf{sin(180x)}$	14. $\mathbf{\lvert cos(180x)\rvert}$
15. $\mathbf{sin(360x)}\rvert$	16. $\mathbf{cos(360x)}$	17. $\mathbf{\lvert sin(360x)\rvert}$
18. $\mathbf{\lvert cos(360x)\rvert}$	19. $\mathbf{\lvert sin(720x)\rvert}$	20. $\mathbf{\lvert cos(720x)\rvert}$
21. $\mathbf{3x}$	22. $\mathbf{3x-1}$	23. $\mathbf{3x-2}$
24. $\mathbf{\lvert 3x-1\rvert}$	25. $\mathbf{\lvert 3x-2\rvert}$	26. $\mathbf{(3x-1)/2}$
27. $\mathbf{(3x-2)/2}$	28. $\mathbf{\lvert (3x-1)/2\rvert}$	29. $\mathbf{\lvert (3x-2)/2\rvert}$
30. $\mathbf{x/0.32-0.78125}$	31. $\mathbf{2x-0.84}$	
32. $\mathbf{4x}$; $\mathbf{1}$; $\mathbf{-2x+1.84}$; $\mathbf{x/0.08-11.5}$		33. $\mathbf{\lvert 2x-0.5\rvert}$
34. $\mathbf{2x}$	35. $\mathbf{2x-0.5}$	36. $\mathbf{2x-1}$

Приведем примеры наиболее часто используемых схем опции **rgbformulae** (см. цв.иллюстр. IX):

7, 5, 15 — традиционная pm3d схема (черно-сине-красно-желтая),
3, 11, 6 — зелено-красно-фиолетовая,
23, 28, 3 — океаническая (зелено-сине-белая),
21, 22, 23 — тепловая (черно-красно-желто-белая),
30, 31, 32 — цветная на сером фоне (черно-сине-фиолетово-желто-белая),
33, 13, 10 — цветов радуги (сине-зелено-желто-красная),
34, 35, 36 — черно-красно-желто-белая.

Задание **rgbformulae** позволяет минимизировать размер получающегося файла, содержащего изображение. Поэтому с целью минимизации выходных файлов рекомендуется для особых пользовательских палитр подбирать подходящую RGB-формулу с помощью команды

set palette fit2rgbformulae

7.5.3 Функции, задающие палитру

Опция **functions <R>, <G>, <B>** позволяет задавать свои функции для определения трех цветов (красного, зеленого и синего) в зависимости от параметра **gray**. Областью определения и областью значений этих функций является отрезок [0, 1].

Заметим, что в случае выбора **HSV** цветовой модели, первая функция определяет тон палитры. Например,

set palette model HSV functions gray, 1, 1

Другим примером использования опции **functions** служит черно-золотистая **XYZ** цветовая модель

**set palette model XYZ functions **
gray0.35, gray**0.5, gray**0.8**

7.5.4 Цветовое колесо

Опция
cubehelix {start <val>}{cycles <val>}{saturation <val>}

позволяет задавать семейство палитр. Цвет (тон) этих палитр определяется циклически с помощью стандартного цветового колеса [7], в то время как интенсивность нарастает монотонно в зависимости от значения переменной **gray**, которое принадлежат отрезку [0, 1].

Ключ **start** задает начальную точку на цветовом колесе в радианах. Ключ **cycles** определяет сколько раз цветовое колесо укладывается диапазон палитры. Если значения ключа

saturation (насыщенность) больше единицы, то это может приводить к обрезанию отдельных RGB-компонент и делать рост интенсивности немонотонным. Свойства **cubehelix**-палитры зависят от параметра **gamma** (см. 7.5.5, с. 147). По умолчанию считаются заданными следующие значения: **start** = 0.5, **cycles** = -1.5, **saturation** = 1, **gamma** = 1.5.

7.5.5 γ-корректировка

Автоматическая γ-корректировка, задаваемая командой

set palette gamma <gamma>

может быть применена к черно-белым палитрам или **cubehelix**-палитре. Величина параметра <**gamma**> определяет закон перемены цветов по следующему правилу

color(gray) = gray **(1./gamma)
set palette model RGB
functions color(gray), color(gray), color(gray)

Например, если **gamma=1**, то в этом случае будет задан линейный закон изменения цвета в палитре. Примеры черно-белых палитр, созданных с различными **gamma**, приведены на рисунке 37.

7.5.6 Дискретное задание палитры

Возможно дискретное задание палитры. Для этого задается шкала градиента палитры, позволяющая по значениям переменной **gray** определять значения **RGB**-цветов:

**set palette defined **
{(< gray1 >< color1 > {, < grayN >< colorN >}...)}

Областями изменения независимой переменной и всех трех зависимых служит отрезок [0, 1]. В результате определяются узлы, между которыми для построения функций, задающих все три RGB-цвета, проводится линейная интерполяция. Параметр <**color**> может быть определен тремя разными способами

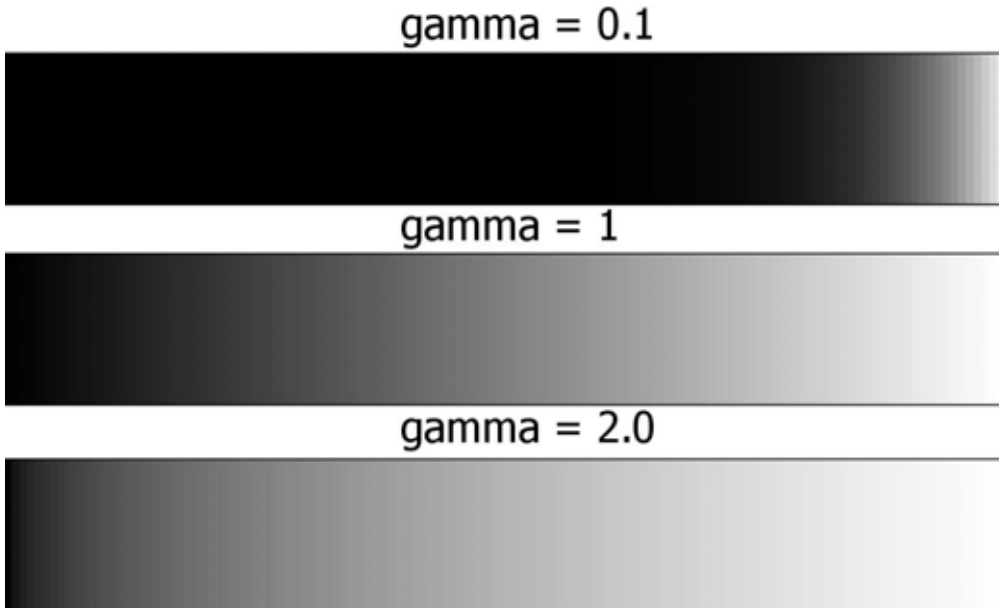


Рис. 37: примеры черно-белых палитр с различными **gamma**

<color> := { <r> <g> <b> | '<color-name>' | '#rrggbb' }

Если значения RGB-цветов определяются числами из отрезка [0, 1], то они разделяются пробелами. Если же используются названия цветов, то каждое из них заключают в кавычки. Номер RGB-цвета также заключают в кавычки. Значения параметра <**gray**> должны быть последовательностью действительных чисел, которые автоматически масштабируются на отрезок [0, 1].

Команда **set palette defined** без опций устанавливает стандартную RGB-палитру с полным спектром цветов.

Примеры (см. цв.иллюстр. X).

1. Простая черно-белая палитра эквивалентна следующим командам

set palette model RGB
set palette defined (**0** ″**black**″, **1** ″**white**″)

2. Для создания сине-желто-красной палитры следующие ко-

манды эквивалентны:

```
set palette defined ( 0 "blue", 1 "yellow", 2 "red" )
set palette defined ( 0 0 0 1, 1 1 1 0, 2 1 0 0 )
set palette defined ( 0 "#0000ff  1 "#ffff00  2 "#ff0000" )
```

3. Для создания палитры цветов радуги можно выполнить команду

```
set palette defined (0 "blue", 3 "green", 6 "yellow", 10 "red")
```

4. В случае **HSV** цветовой модели для создания спектральной палитры используются следующие команды:

```
set palette model HSV
set palette defined (0 0 1 1, 1 1 1 1 )
set palette defined \
(0 0 1 0, 1 0 1 1, 6 0.8333 1 1, 7 0.8333 0 1)
```

5. В пакете MATLAB по умолчанию используется следующая палитра:

```
set pal defined ( 1 '#00008f', 8 '#0000ff', 24 '#00ffff' \
                  40 '#ffff00', 56 '#ff0000', 64 '#800000' )
```

6. Палитра с равномерно-разнесенными цветами может быть создана командами:

```
set palette model RGB maxcolors 4
set palette defined ( 0 "yellow", 1 "red" )
```

7. Палитра "луч света" с точками преломления на относительных высотах 1/3 и 2/3 создается командами:

```
set palette model RGB
set palette defined (0 "dark-green", 1 "green", \
                     1 "yellow", 2 "dark-yellow", \
                     2 "red", 3 "dark-red")
```

7.5.7 Задание палитры с помощью файла

Задание палитры возможно посредством считывания из файла параметров, задающих градиент цветов. Файл должен содержать либо четыре колонки данных (gray, R, G, B), либо три колонки (R, G, B). Выбор колонок может быть осуществлен с помощью опции **using**. В случае, когда используется файл с тремя колонками, номер строки используется, как значение **gray**. Диапазон значений **gray** автоматически масштабируется на отрезок [0, 1]. Файл читается по правилам обычного файла с данными, поэтому все опции, используемые для фильтрации, действуют.

В качестве имени файла может быть задан символ '-', что означает, что параметры градиента будут вводиться с экрана. Конец ввода заканчивается набором символа 'e'.

Примеры.

1. Для задания палитры с помощью файла, содержащего в каждой строке тройки целых чисел из отрезка [0, 255], можно использовать команду:

```
set palette file 'some-palette' \
using ($1/255) : ($2/255) : ($3/255)
```

2. Задать палитру "радуга" можно, выполнив следующие команды:

```
set palette model RGB file "-"
0 0 1
0 1 0
1 1 0
1 0 0
e
```

3. Числа, задающие цвета, могут быть записаны в двоичном виде. Для задания палитры в этом случае следует использовать команду

```
set palette file 'palette.bin' binary record=64 using 1:2:3
```

8 Легенда, надписи и другие объекты

8.1 Название графика (опция TITLE)

Каждому графику присваивается свое название. По умолчанию это либо аналитическая формула, задающая функцию, либо имя файла, откуда считывались данные для построения. Название выводится в легенде изображения, где каждому графику выделяется отдельная строка. Название можно задать явным образом, включив в команду **plot** после задания функции опцию **title**. Ее синтаксис:

title $''$ < **text** >$''$
title columnheader

В первом случае в качестве названия будет фигурировать строка <**text**>, а во втором случае текст, содержащийся в первой строке файла, откуда считывались данные для построенного графика. Для задания специальных символов в строке заголовка нужно указывать их восьмеричный код.

Если в первой строке файла с данными в столбце с номером N содержится явное название функции, то используют следующую опцию:

title columnheader (N)

Иногда требуется отсутствие комментариев по поводу того, что изображает та или иная кривая. Этого можно добиться, определив опцию **title** следующим образом:

notitle [<ignored text>]

Отсутствия названия можно также достичь, установив "нулевую" легенду, т.е. применить опцию **title ' '**.

8.2 Легенда изображения (опция KEY)

Опция **key** отвечает за легенду графика, то есть надпись, которая содержит комментарии того, что изображено на рисунке. Ее задание осуществляется командой:

set key {on|off} {default}{{inside|outside}
|{lmargin|rmargin|tmargin|bmargin}|{at <position>}}
{left|right|center}{top|bottom|center}
{vertical|horizontal}{Left | Right }
{{no}opaque}
{{no}reverse} {{no}invert}
{samplen <sample_length>}
{spacing <vertical_spacing>}
{width {width_increment>}
{height <height_increment>}
{{no}autotitle {columnheader}}
{title "<text>"} {{no}enhanced}
{font "<face>,<size>"} {textcolor <colorspec>}
{{no}box {{linestyle | ls <line_style>}
| {linetype | lt <line_type>}
{linewidth | lw <line_width}}}
{maxcols {<max no. of columns> | auto }}
{maxrows {<max no. of rows> | auto }}

Легенда состоит из отдельных элементов. Особым элементом служит надпись, являющаяся общим названием изображенния. Остальные элементы - это подписи к каждому графику, которые в свою очередь состоят из текста названия функции (или другого комментария) и шаблона (образца) линии, которой график изображен. Каждый такой элемент описывает свой график. Вывод легенды может быть отключен одной из двух команд:

set key off
unset key

Вывод отдельных ключевых элементов отключается командами вида **notitle keyword**.

Элементы легенды упорядочены либо по вертикали, либо по горизонтали. Вертикальное расположение подразумевает, что каждый элемент легенды начинается в новой строке и находится под предыдущим. В случае горизонтального упорядочивания очередной элемент располается в той же строке, что и предыдущий. Вертикальный размер легенды может быть ограничен с помощью параметра **'maxrows'**, а горизонтальный —

'maxcol'. В случае, когда число строк колонки в случае вертикального упорядочивания или когда число элементов строки при горизонтальном упорядочивании не позволяют разместить все элементы легенды, добавляются новые колонки и строки соответственно. Таким образом, элементы легенды оказываются расположенными в виде таблицы. По умолчанию подписи к графикам выводятся в том порядке, в котором они указаны в команде **plot**. Опция **invert** меняет порядок размещения подписей.

При изображении контуров командой **splot** метки контуров перечисляются в легенде. В этом случае часто требуется для получения хорошо читаемой легенды задавать формат легенды не по умолчанию (см. 2.4.2, с. 57).

Элемент "Общий заголовок" является особым. Он всегда выводится в отдельной строке и эта строка не учитывается при оценке числа строк. "Общий заголовок" создается опцией **title** "<**text**>". Выравнивание этого заголовка происходит также, как выравнивание подписей к отдельным графикам.

Все элементы одной колонки выравниваются по вертикали относительно воображаемой линии, разграничивающей название функции и образец шаблона графика. Координаты верхней точки именно этой воображаемой линии задаются в качестве параметра <**position**> (см. 10.2, с. 174). Число передаваемых координат зависит от размерности изображения. При сохранении получаемого изображения в виде команд TeX или некоторых других возможных значений опции **terminal** gnuplot может лишь правильно оценить ширину строки для ее позиционирования.

Названия функций можно выровнять либо по левому краю колонки (**Left**), либо по линии разграничения (**Right**). По умолчанию в каждой надписи сначала идет текст, а потом шаблон линии. Этот порядок можно изменить, задав ключ **reverse**.

По умолчанию легенда размещается в правом верхнем углу изображения. Ключевые слова:

left — слева, **center** — по центру, **right** — справа,
top — сверху, **bottom** — снизу,

Рис. 38: положение легенды в зависимости от ключевых слов

inside — внутри, **outside** — снаружи,
lmargin — по левому краю, **tmargin** — по верхнему краю,
bmargin — по нижнему краю, **rmargin** — по правому краю

могут быть использованы для определения местоположения легенды (см. рисунок 38).

С помощью ключа at <**position**> можно задать точное местоположение легенды. В этом случае ключевые слова **left, right, top, bottom** и **center** служат для выравнивания надписи.

Вокруг легенды можно нарисовать рамку с помощью ключа **box**{...}. Параметры линии контура рамки задаются с помощью **linetype** и **linewidth** или через созданный пользователем **linestyle** (см. 4.2.2, с. 87). Опция <**height_increment**> является числовым параметром, на величину которого либо увеличивается, либо уменьшается высота окна легенды. Это удобно,

когда хочется подобрать окно под легенду по своему желанию.

По умолчанию подпись легенды к отдельному графику рисуется одновременно с соответствующим графиком. Поэтому графики, нарисованные позже, могут накладываться на уже существующие надписи. Задание опции легенды **opaque** позволяет нарисовать легенду позже всех графиков. В этом случае область размещения легенды окрашивается цветом фона и создается сама легенда. При таком порядке создания изображения могут быть частично скрыты участки некоторых графиков. Вернуться к первоначальному изображению можно с помощью опции **noopaque**

Все надписи легенды создаются по умолчанию заданием опции **autotitles**. Автоматическая генерация надписей может быть отменена заданием опции **noautotitles**. После чего необходимо явное задание легенды посредством задания опции **title** в команде **plot**.

Использовать первую запись в каждом столбце файла с дискретными данными для названия соответствующего графика можно с помощью опции **autotitle columnheader**. Если значения функции, график которой требуется построить, зависят от значений нескольких столбцов, то в этом случае можно выбрать один из заголовков этих столбцов. Например,

```
plot 'file.dat' using (($2+$3)/$4) \
title columnhead(3) with lines
```

Интервал между строками легенды задается командой (например, 2.0, как на рисунке VII)

```
set key spacing 2.
```

Аналогично можно задать в легенде длину образца шаблона линии

```
set key samplen <lengh>
```

По умолчанию для легенды определены следующие значения параметров: **on, right, top, vertical, Right, noreverse, noinvert, samplen 4, spacing 1.25, title " ", nobox**. Набор

установок <**linetype**> по умолчанию такой же, как установлен для изображаемых графиков. Команда

set key default

возвращает настройки легенды к состоянию "по умолчанию".

Если легенда располагается слева, то может быть удобным использовать команду

set key Left reverse

Примеры работы с опцией **key**:

set key at 2,3.5,2 – устанавливаются координаты легенды равными (2, 3.5, 2) в системе координат по умолчанию, либо в первой из заданных явно систем;
set key below – задает расположение легенды под графиком;
set key left bottom Left title 'Legend' box 3 – располагает легенду в нижнем левом углу, дает изображению название Legend и заключает легенду в рамку, нарисованную линией типа 3.

8.3 Дополнительные надписи (опция LABEL)

В Gnuplot есть возможность добавлять на изображение произвольные надписи. Команда, позволяющая это делать имеет формат:

set label {<tag>} {"<label text>"} {at <position>}
{left|center|right}
{norotate|rotate {by <degrees>}}
{font "<name>{,<size>}"}
{noenhanced}
{front|back}
{textcolor <colorspec>}
{point <pointstyle>|nopoint}
{offset <offset>}

Для удобства работы с надписями предусмотрены еще две команды:

unset label {< **tag** >}
show label

Первая отменяет надпись с именем <**tag**>, а вторая показывает все заведенные на момент запроса надписи.

Задание параметра <**position**> задает координаты надписи на изображении (см. 10.2, с. 174).

Значение параметра <**tag**> — целое число, используемое как идентификатор надписи. Параметр <**tag**> задается автоматически, если он опущен при задании **label**. Для изменения любого атрибута надписи достаточно использовать команду **set label** с указанием ее <**tag**> и новых значений меняемых параметров.

Текст надписи <**label_text**> может быть строкой символов, символьной переменной или символьной переменной с включением числовых выражений (см. 10.5.4, с. 190). По умолчанию текст надписи располагается слева от указанной точки, определяемой параметром <**position**>. Возможно явное задание положения надписи относительно <**position**> заданием значения опции **left**, **right** или **center**.

Надпись рисуется вертикально, если задана опция **rotate**. При задании параметра <**degrees**> надпись поворачивается на соответствующий угол.

Указание опции **font** задает тип шрифта и его размер, которыми выводится надпись. Опция **textcolor** служит для выбора цвета надписи (см. 10.4.2, с. 183).

Параметрами опции **pointstyle** могут служить **lt**, **pt** и **ps**. Эта опция задает точку у надписи. При заданном значении **nopoint** точка у надписи не рисуется.

Примеры задания надписей.

1. Задать надпись "y=x" в точке с координатами (1,2)

set label "y=x" at 1,2

2. Задать надпись в виде буквы Sigma размером 24 в центре окна

set label "S" at graph 0.5,0.5 center font "Symbol, 24"

3. Задать надпись "y=x*x", слева от точки (2,3,4) и номером 3

set label 3 "y=x*x" at 2,3,4 right

8.4 Изображение стрелок (опция ARROW)

В Gnuplot есть возможность добавлять на изображение различные стрелки, которые в частности позволяют визуально связать дополнительную надпись с точкой или участком кривой или поверхности. Команда, позволяющая это делать, имеет формат:

set arrow {<tag>} { from <position> }
{ to | rto <position>}
{ {arrowstyle | as <arrow_style>}
| { {nohead | head | backhead | heads }
{ size <length>, <angle> {,<backangle>}}
{filled | empty | nofilled }
{ front | back }
{ { linestyle | ls <line_style>}
| { linetype | lt <line_type> }
{ linewidth | lw <line_width> } } } }

Для удобства работы со стрелками предусмотрены еще две команды:

unset arrow {< **tag** >}
show arrow {< **tag** >}

Первая отменяет стрелку с именем <**tag**>, а вторая показывает параметры указанной стрелки. Использование этих команд без опции имени стрелки приводит к удалению всех используемых стрелок или к показу параметров всех стрелок.

Присвоение стрелке имени <**tag**> позволяет удалять или изменять ее параметры по ходу создания изображения. Редактирование стрелки осуществляется повторным вызовом команды **set arrow** с конкретным именем <**tag**> и указанием новых значений изменяемых параметров.

Задание значения параметров <**position**> позволяет задать координаты начала и конца стрелки на изображении (см. 10.2, с. 174).

Значение **nohead** означает, что стрелка будет без "наконечника" — просто отрезком. Это предоставляет еще одну возможность нарисовать нужный отрезок. По умолчанию стрелка имеет "наконечник". Задание значения **backhead** означает, что "на-

конечник" будет нарисован на конце, обозначенном как начало стрелки. "Наконечники" будут на обоих концах, если задано значение **heads**.

Параметры <**length**>, <**angle**> и <**backangle**> определяют размер "наконечника": <**length**> — длина сторон острия, а <**angle**> — угол, заданный в градусах и определяющий заостренность "наконечника". Параметр <**backangle**> задается только в случае задания опций **filled** или **empty**. Его значение это угол, заданный в градусах, у заднего "наконечника" стрелки. Есть пороговые значения этого угла. До 70^o "наконечник" имеет одно направление, а если больше 110^o, то другое. Если же задано значение между 70^o и 110^o, то получается просто отрезок без "наконечника".

Задание опции **filled** приводит к закрашиванию "наконечника". Значения параметров стиля линии задаются так, как это описано в разделе 4.2.2.

При задании опции **front** стрелка рисуется поверх графика, а задание **back** (задается по умолчанию) приводит к тому, что график накладывается на стрелку. В последнем случае график может серьезно затруднить понимание того, куда направлена стрелка. Особенно, если график это закрашенная поверхность.

Примеры.

1. Нарисовать стрелку из начала координат в точку (1,2) стилем, заданным пользователем, под номером 5

set arrow to 1,2 ls 5

2. Нарисовать стрелку с именем 3 из нижней левой точки окна в точку (-5, 5, 3)

set arrow 3 from graph 0,0 to -5, 5, 3

3. Изменить параметры стрелки с именем 3: задать новые координаты конца (1, 1, 1), отменить "наконечник" стрелки, установить ширину линии 2

set arrow 3 to 1,1,1 nohead lw 2

4. Нарисовать стрелку в виде латинской заглавной буквы I

set arrow 3 from 0,-5 to 0,5 heads size screen 0.1,90

8.5 Надпись "дата-время"

В изображение можно включить надпись, содержащую информацию о дате и времени его создания (см. рис. 35)

set timestamp {"<format>" {top|bottom} {{no}rotate}
{offset <xoff>{,<yoff>}} {font "<fontspec>"}
{textcolor <colorspec>}

Вместо **timestap** допускается использовать сокращение **time**. Отменить выведение и показать формат этой надписи можно с помощью команд

unset timestamp
show timestamp

По умолчанию формат вывода надписи "дата-время" следующий

%a %b %d %H:%M:%S %Y

В этом формате выводятся день недели, название месяца, число, час, минуты, секунды, год в четырехсимвольном формате. Для задания нужного формата можно использовать спецификации "время-дата" (см. 10.5.2, с. 187).

По умолчанию надпись выводится в левый нижний угол, но при желании ее можно разместить в верхнем левом углу, выбрав опцию **top**. Для возвращения ее обратно следует указать ключ **bottom**. Если ваш терминал допускает вертикальное размещение текста, можно воспользоваться опцией **rotate**.

Параметры <**xoff**> и <**yoff**> опции **offset** позволяют смещать позицию надписи в более удобное место (см. 10.2, с. 174).

Опции <**font**> и <**textcolor**> дают возможность задавать шрифт и цвет надписи (см. 10.4.2, с. 183).

Пример.
**set timestamp "%d/%m/%y %H:%M" offset 80, -2 **
font "Helvetica"

8.6 Изображение различных фигур

8.6.1 Общий синтаксис команды

В изображения, получаемые с помощью команды **plot**, можно добавлять изображения различных фигур, используя команду

set object <index>
<object-type> <object-properties>
{front|back|behind} {clip|noclip}
{fc|fillcolor <colorspec>} {fs <fillstyle>}
{default} {lw|linewidth <width>}
{dt|dashtype <dashtype>}

Отменить добавление фигуры можно, выполнив команду

unset object <index>

В версии 5.0 есть возможность добавлять прямоугольники (**rectangle**), круги (**circle**), эллипсы (**ellipse**) и многоугольники (**polygon**). Каждая из этих фигур описывается своим набором параметров, которые приводятся ниже в отдельных разделах.

Если задан ключ **front**, то Gnuplot рисует фигуру поверх всех графиков, но до надписей с таким же ключом. При задании ключа **back** фигура изображается до всех графиков и надписей. Ключ **behind** размещает фигуру под всеми объектами, включая оси и прямоугольники, отмеченные этим же ключом.

По умолчанию все фигуры обрезаются по границам изображения, но если указать ключ **noclip**, то фигура будет доходить до границы окна.

Далее перечислены опции, определяющие параметры заливки фигуры и изображения ее границы. Они описаны в разделах 10.4.3 с. 184 и 10.4.1 с. 181, поскольку эти ключи используются во многих командах.

Пример. Задание цветного фона для графика

set object rectangle from screen 0,0 to screen 1,1 behind

8.6.2 Изображение прямоугольника

Прямоугольник задается тремя возможными наборами параметров:

set object <index> rectangle
{from <position> {to|rto} <position> |
center <position> size <w>, <h> |
at <position> size <w>, <h> }

Первый вариант опций задает левый нижний и верхний правый углы прямоугольника, во втором и третьем вариантах задаются его центр, ширина и высота. Координаты **<position>** задаются в одной из возможных координатных систем (см. 10.2, с. 174).

Примеры.

1. Для задания голубого фона создается прямоугольник, совпадающий со всем окном, где рисуется график

```
set object 1 rect\
from graph 0, graph 0 to graph 1, graph 1 behind
set object 1 rect fc rgb ″cyan″, fillstyle solid 1.0
```

2. Создается красный прямоугольник с диагональю из точки (0,0) в точку (2,3)

```
set object 2 rect from 0, 0 to 2, 3 fc lt 1
```

3. Задается незакрашенный прямоугольник с синей границей

```
set object 3 rect from 0, 0 to 2, 3 \
fs empty border rgb ”blue”
```

4. Закрасить прямоугольник с номером 2, установленным по умолчанию цветом

```
set object 2 rect default
```

Заметим, что прямоугольник, задаваемый координатами **screen**, может выступать за пределы окна. Такие прямоугольники обрезаются по границе окна.

8.6.3 Изображение круга

Круг задается следующей командой:

set object <index> circle
{at|center} <position> size <radius>
{arc [<begin>:<end>]}
{<other-object-properties>}

Положение круга определятся координатами центра и радиусом. Ключевые слова **at** и **center** являются синонимами. Радиус задается в единицах длины горизонтальной оси.

Опция **arc** с параметрами <**begin**> и <**end**> предназначена для задания дуги. Отсчет углов осуществляется против часовой стрелки.

8.6.4 Изображение эллипса

Эллипс задается следующей командой:

set object <index> ellipse
{at|center} <position> size <w>, <h>
{angle <orientation>} {units xy|xx|yy}
{<other-object-properties>}

Положение и размеры эллипса задаются его центром, а также шириной и высотой (большой и малой осями симметрии). Ключевые слова **at** и **center** являются синонимами. Длины отрезков задаются в единицах длины соответствующих осей. Ориентация эллипса задается углом, заданным в градусах, между горизонтальной осью координат и большой осью симметрии. По умолчанию большая ось направлена вдоль горизонтальной оси координат. Если у опции **units** указан ключ **xy**, то размер большой оси задается в единицах оси **X**, а малой — оси **Y**. Если значение ключа — **xx**, то обе оси масштабируются в единицах оси **X**, а при значении **yy** — в единицах оси **Y**. По умолчанию это значение равно **xy**. Заметим, что различие в масштабах осей координат приводит к тому, что команды

set object ellipse size $< 2r >, < 2r >$
set object circle $< r >$

в случае **units xy** нарисуют разные кривые: в первом случае будет эллипс, а во втором всегда будет круг.

Дополнительные параметры для изображения эллипса можно задать командой

set style ellipse {units xx|xy|yy}
{size {graph|screen} <a>,
{{graph|screen} <b>}}
{angle <angle>}
{clip|noclip}

Эта команда задает параметры по умолчанию для всех изображаемых эллипсов. Опции **units, size** и **angle** имеют тот же смысл, что и в команде **set object ellipse**. По умолчанию параметры у опции **size** имеют значения **graph 0.05,0.03**.

При заданном ключе **clip** эллипс обрезается по границам графика, а если задан ключ **noclip** — нет.

8.6.5 Изображение многоугольника

Многоугольник задается координатами своих вершин

set object <index> polygon
from <position> to <position> ... {to <position>}

Пример. Задание треугольника голубого цвета с черной границей

set object 1 polygon from 0, 0 to1, 1 to 2, 0
set object 1 fc rgb ″cyan″ fillstyle solid 1.0 border lt − 1

9 Несколько изображений в одном окне

9.1 Описание режима MULTIPLOT

Для того, чтобы разместить несколько изображений в одном окне, используется специальный режим **multiplot**. Он задается командой

set multiplot
{ title <page title> { font <fontspec> }
{ enhanced | noenhanced } }
{ layout <rows>, <cols>
{ rowsfirst | columnsfirst } { downwards | upwards }
{ scale <xscale> {, <yscale> } }
{ offset <xoff> {, <yoff> } } }

Все надписи и стрелки, определенные ранее, будут изображены согласно определенным на текущий момент размеру и координатам положения изображения (за исключением тех, что были заданы в системе координат окна). То же самое касается всех других параметров, определяемых через команду **set ...**. Если Вы хотите, чтобы какая-то надпись появлялась только у одного из графиков, то необходимо до команды **plot**, **splot** или **replot** выполнить соответствующую команду **set ...**, а после **unset ...** внутри блока **multiplot**. Например, **set time**/**unset time**.

Заголовок окна независим от заголовков, входящих в него изображений. Место для него резервируется в верхней части окна.

Опция **layout** позволяет автоматически располагать изображения нескольких графиков в одном окне без использования команд **set size** (см. 10.3.3, с. 178) и **set origin** (см. 10.3.4, с. 179) перед каждым из них. При ее задании значения этих параметров определяются автоматически и не требуют переопределения каждый раз. Опция **layout** делит окно на строки и колонки. Их число задается параметрами **row** и **cols**. Соответствующие ключи в команде **set multiplot** определяют порядок заполнения

ячеек получающейся таблицы: **rowsfirst** — первыми заполняются строки, **columnsfirst** — первыми заполняются колонки, **downwards** и **upwards** — сверху вниз или снизу вверх соответственно. По умолчанию действуют **rowsfirst** и **downwards**.

Каждое изображение может быть отмасштабировано с помощью ключа **scale** с коэффициентами <**xscale**> и <**yscale**> по осям и сдвинуто при задании опции **offset**, где сдвиг по осям определяется параметрами <**xoff**> и <**yoff**>.

После завершения построения всех графиков следует выполнить команду

unset multiplot

Замечание. Некоторые типы терминалов не выводят окно с изображениями до того, как будет выполнена команда **unset multiplot**. Для других типов после каждой команды **plot** требуется обновить окно вывода, чтобы увидеть добавленный график.

Для того, чтобы очистить окно от предыдущих построений в режиме **multiplot** используется команда **clear**.

9.2 Несколько картинок в одном изображении

Режим **multiplot** можно использовать в различных целях. Наиболее простым является желание расположить в одном окне несколько изображений. Приведем пример расположения четырех графиков в одном окне (см. рис. 39), полученных с помощью следующих команд.

Пример 9.1

```
# Задать размер изображения и формат сохранения.
set terminal png size 600, 800
# Файл для вывода результатов
set output 'mult_tri.png'
# Задать режим multiplot с делением окна на четыре области.
set multiplot layout 2, 2
# Легенду выводить сверху изображения по центру.
set key c t o font ',20'
# Задать число узлов, используемых для построения.
```

```
set samples 1000
# Границу не рисовать.
set border 0
# Провести оси координат через точку (0,0).
set xzeroaxis lt -1
set yzeroaxis lt -1
# Задать деления на обеих осях.
set xtics axis
set ytics axis
# Задать подписи для следующих делений.
set xtics (-6,-4,-2,2,4,6)
set ytics (-1,-0.5,0.5,1)
# Название оси 'X' вывести в виде надписи.
set label 1 "X" at 6, 0.1 font ',20'
# Название оси 'Y' не поворачивать.
set ylabel "Y" norotate font ',20'
# Задать смещение названия оси 'Y'
# относительно положения по умолчанию.
set ylabel offset graph 0.55, 0.5
# Вывести подпись для точки '0' в виде надписи.
set label 2 "0" at 0.32, -0.1
# Задать диапазоны изменения переменных.
set xrange [-2*pi:2*pi]
set yrange [-1.1:1.1]
# Создать изображения графиков sin(x) и cos(x)
plot sin(x) with lines lc "red"
plot cos(x) with lines lc "green"
# Поменять параметры для создания графиков tg(x) и ctg(x).
set yrange [-20:20]
set label 1 "X"at 6, 2 font ',20'
set label 2 "0"at 0.32, -2
set ytics (-20,-10,10,20)
tg(x)=tan(x)
plot tg(x) with lines lc "brown"
ctg(x)=cos(x)/sin(x)
plot ctg(x) with lines lc "blue"
# Отключить режим multiplot.
```

unset multiplot

Замечание. Название оси 'X' было создано в виде надписи, поскольку задать смещение ее относительно положения по умолчанию сложнее, чем указать координаты этой надписи в используемой системе координат.

9.3 График и его увеличенный фрагмент

Режим **multiplot** может оказаться полезным в тех случаях, когда нужно изобразить график функции и его фрагмент на одном рисунке. Рассмотрим функцию, заданную параметрически

$$\mathbf{x = t^2 \cos(t)}, \qquad \mathbf{y = t^2 \sin(t)}.$$

Выбор большой длины спирали при ограниченном поле окна приводит к тому, что график превращается вблизи нуля в сплошное пятно. Для того, чтобы показать поведение функции вблизи нуля можно нарисовать ее при малых t на отдельном графике с большим масштабом. График фрагмента можно разместить, например, в правом нижнем углу окна основного изображнения. На рисунке 40 приведено такое изображение, полученное с помощью следующих команд.

Пример 9.2

```
set multiplot
# Задать диапазоны по осям X и Y.
set xrange [-160000:160000]
set yrange [-160000:160000]
# Задать положение легенды.
set key at graph 0.5, 0.9
# Использовать полярную систему координат.
set polar
# Деления по оси r нанести с шагом 50000.
set rtics 50000
# Задать число узлов.
set samples 10000
# Нарисовать основной график.
```

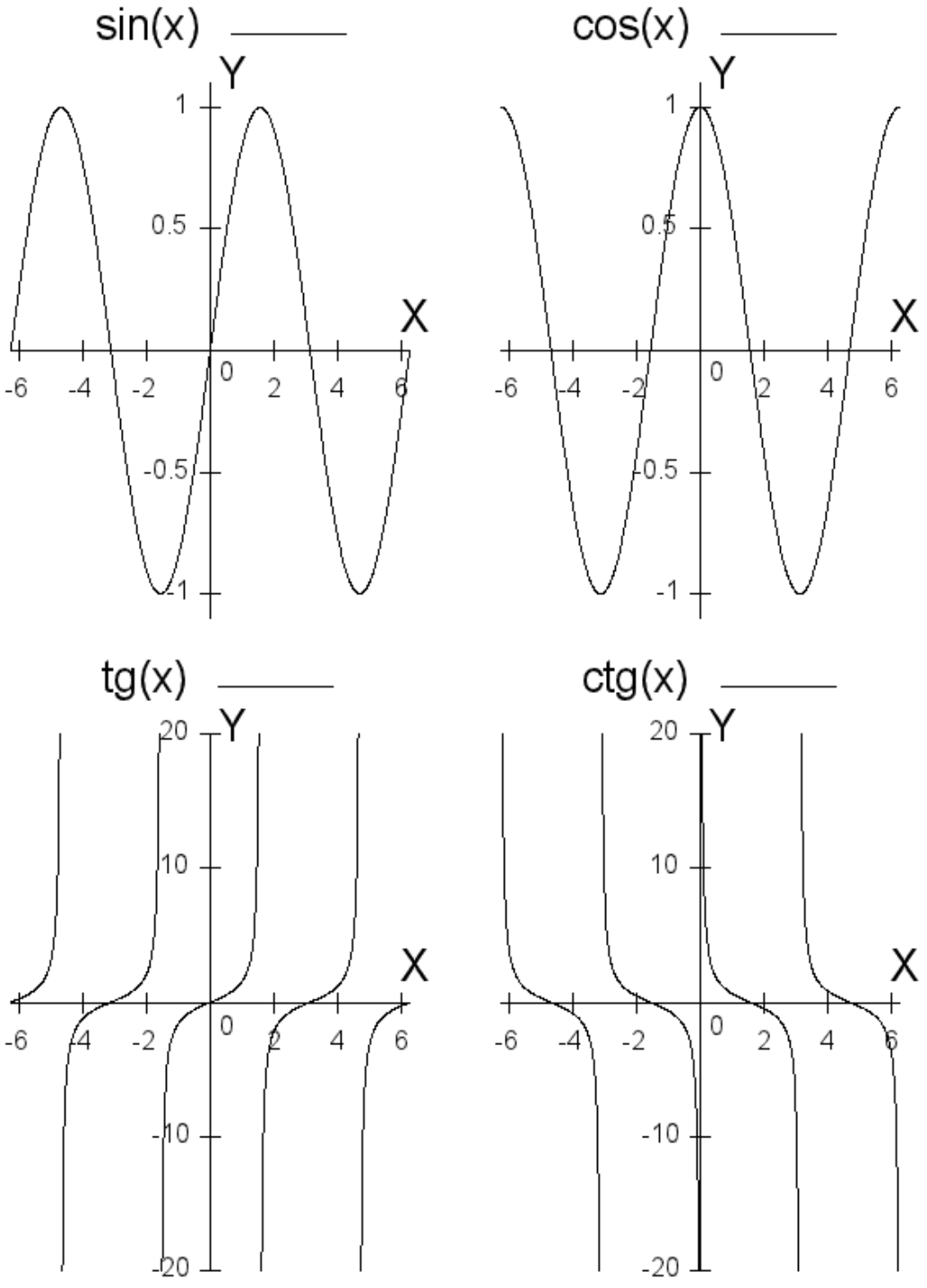


Рис. 39: графики тригонометрических функций

```
plot [0:300] t*t w lines t 'x=t^2 cos(t); y=t^2 sin(t)'
# Задать размер графика фрагмента.
set size 0.3, 0.3
# Задать положение графика фрагмента в окне.
set origin 0.675, 0.08
# Задать диапазоны по осям X и Y у графика фрагмента.
set xrange [-1600:1600]
set yrange [-1600:1600]
# Задать шаги между делениями по осям у графика фрагмента.
set xtics 1400
set ytics 700
# Задать величину делений.
set tics scale 0.5
# Удалить деления по оси r.
unset rtics
plot [0:40] t*t with lines title ' '
unset multiplot
```

9.4 Переменный масштаб оси

Сделать график функции более информативным вблизи ее особенности может помочь еще один прием, также использующий режим **multiplot**. Для его иллюстрации рассмотрим функцию

$$\mathbf{y = \sin\left(\frac{\pi}{x}\right)}.$$

Эта функция имеет особенность вблизи нуля: осцилляции становятся все чаще и чаще. Помочь понять ее поведение может введение при $\mathbf{x} \in (\mathbf{0}, \mathbf{2}]$ логарифмического масштаба по оси X. Для того, чтобы нарисовать график функции, используя два разных масштаба по одной оси, воспользуемся режимом **multiplot**. На рисунке 41 приведено такое изображение, полученное с помощью следующих команд.

Пример 9.3

```
set key t c r font ',30'
```

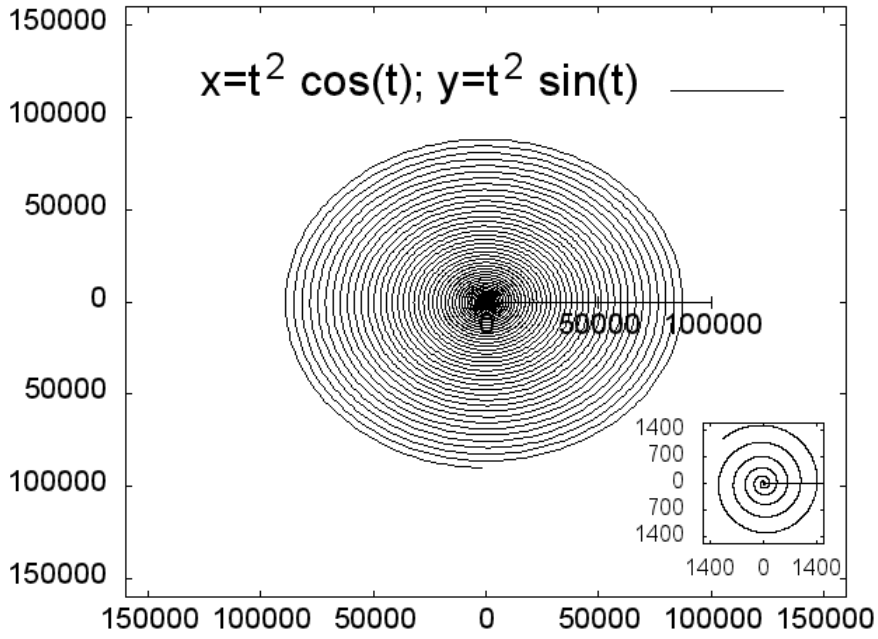


Рис. 40: график функции с увеличенным фрагментом

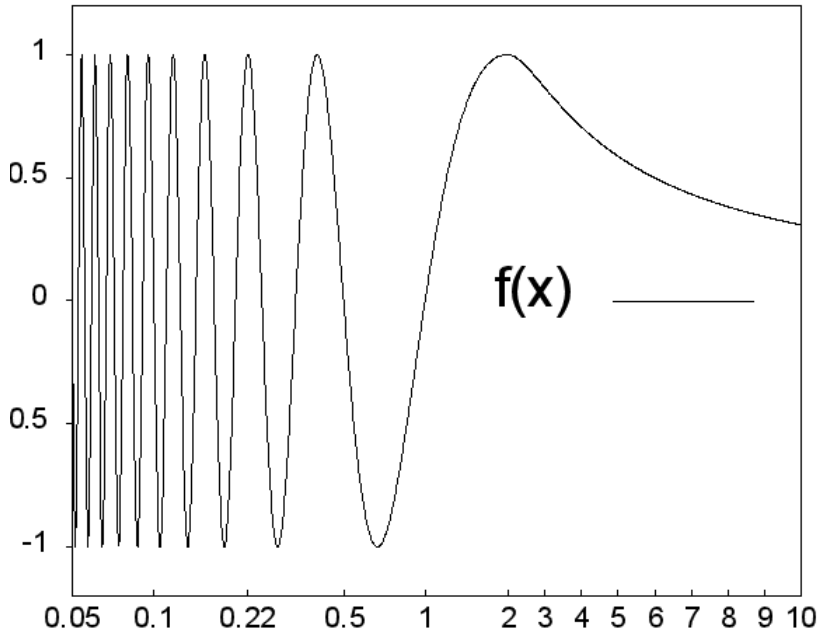


Рис. 41: график функции с переменным масштабом оси x

```
# Задать число узлов.
set samples 1000
set multiplot
# Задать диапазон по оси Y.
set yrange [-1.2:1.2]
# Задать логарифмический масштаб по оси X.
set logscale x
# Не наносить деления оси X на ось X2.
set xtics nomirror
# Наносить только левую, нижнюю и верхнюю границы.
set border 7
# Задать размер области первого изображения.
set size 0.5, 1.
# Задать положение первого изображения в окне.
set origin 0.07, 0.
# Задать левое и правое поля.
set lmargin 0
set rmargin 1
# Задать диапазон по оси X.
set xrange [0.05:2]
# Задать деления по оси X и Y.
set xtics (0.05,0.1,0.22,0.5,1) font ',16'
set ytics out (-1,-0.5,0.,0.5,1) font ',16'
# Не наносить промежуточных делений на ось X.
unset mxtics
# Не наносить дубликаты делений оси Y на ось Y2.
set ytics nomirror
f(x)=(abs(x)>0.00001)?sin(pi/x):0
plot f(x) with lines title ' '
# Отменить логарифмический масштаб по оси X.
unset logscale x
# Наносить только верхнюю, нижнюю и правую границы.
set border 13
# Задать размер области изображения.
set size 0.5, 1.
# Задать положение в окне изображения.
set origin 0.54, 0.
```

```
# Задать левое и правое поля.
set lmargin 1
set rmargin 10
# Задать диапазон изменения по оси X.
set xrange [2:10]
# Деления оси X не наносить на ось X2.
set xtics nomirror
# Деления по оси X нанести от 2 до 10 с шагом 1.
set xtics 2, 1, 10 font ',16'
# Не наносить деления по оси Y.
unset ytics
# Не подписывать деления по оси Y.
set format y " "
plot f(x) with lines
unset multiplot
```

10 Задание возможных режимов

10.1 Руссификация Gnuplot

Для создания надписей на русском языке следует задать соответствующую таблицу перекодировки

set encoding {< **value** >}

Для системы Unix используется таблица **koi8r**, а для системы Windows — **cp1251**.

Предусмотрена команда, делающая попытку автоматически подобрать нужную таблицу перекодировки

set encoding locale

Вывести на экран название используемой таблицы можно, применив команду

show encoding

10.2 Системы координат изображения

Команды **set arrow**, **set key**, **set label** и **set object** позволяют нарисовать стрелку, легенду, надпись или другой объект, привязанный к определенной точке изображения (**position**). Эта точка описывается двумя или тремя координатами

{<system>} <x>,{<system>} <y>{,{<system>} <z>}

Параметр <**system**> может принимать значения **first, second, graph, screen** или **character**.

Координаты точки x, y и z в системе **first** задаются координатной системой, в которой оси координат расположены по левой и нижней границам изображения. В **second** оси расположены по верхней и правой границам. В системе **graph** нижняя левая точка области, где изображается график, имеет координаты (0,0), а верхняя правая — (1,1). В случае использования команды **splot** точка (0,0,0) задает нижний левый угол области размещения графика. В системе **screen** координаты нижнего

левого угла окна, куда выводится изображение графика, полагаются равными (0,0), верхнего правого — (1,1). Отличие от системы **graph** заключается в том, что под окном понимается вся его площадь. Область размещения изображения может занимать лишь часть окна. В системе **character** положение точки задается в величинах кратных ширине и высоте символа, которые определяются исходя из заданного размера и типа шрифта. Левый нижний угол в этой системе имеет координаты (0,0).

Таким образом, задание значения параметра <**position**> зависит от размерности строящейся картинки и типа использованной координатной системы. По умолчанию считается заданной система **first**.

10.3 Размеры, связанные с изображением

У любого изображения есть размеры, которые могут быть заданы явно или определяются автоматически. К этим размерам относятся:
1. размер окна (canvas), где это изображение будет размещено;
2. размер области изображения и его положение в окне;
3. поля окна и изображения.

10.3.1 Размер окна

Общий размер окна (canvas), где рисуется график, может быть задан командой

set term <**terminal_type**> **size** <**XX**>, <**YY**>

Размер окна измеряется в пикселях, размер которых зависит от используемого терминала. От размера окна зависит размер получаемого на выходе файла. По умолчанию размер окна считается равным 640×480 и размер изображения совпадает с размером окна. Приведем набор команд, когда график будет занимать левую нижнюю четверть окна размером 600 пикселей

в ширину и 400 в высоту

```
set size 0.5, 0.5
set term png size 600, 400
set output "pic.png"
plot "data" with lines
```

10.3.2 Граница области изображения

В случае двумерных графиков область изображения является прямоугольником. Координаты точек (x,y) этого прямоугольника принадлежат диапазонам изменения переменных. В трехмерном случае областью изображения является прямоугольный параллелипипед. Точка с координатами (x,y,z) принадлежит этому параллелипипеду, если значения x и y лежат в диапазонах изменения независимых переменных, а значение z зависит от диапазона изменения зависимой переменной и координаты пересечения плоскости XY с осью Z (см. 6.4, с. 132).

Граница изображения может быть определена явно командой

```
set border {<integer>} {front | back}
           {linewidth | lw <line_width>}
           { {linestyle | ls <line_style>}
           | {linetype | <line_type>}}
```

Стороны прямоугольника плоской картинки естественно назвать "bottom" — нижняя, "left" — левая, "right" — правая и "top" — верхняя. В случае, если картинка трехмерная, то изображение плоскости XY всегда имеет четыре угла, которые можно назвать "front" — передний, "back" — задний, "left" — левый и "right" — правый. Эти названия определяют соответствующие углы, если задана ориентация осей с помощью команды set view 56,103. Такие же названия имеют углы плоскости, параллельной плоскости XY и имеющей максимальную координату по оси Z. Тем самым каждому ребру прямоугольного параллелипипеда, содержащего изображение, можно дать свое название. Например, "bottom right back" — нижнее, соединяющее правый и задний

углы, или "front vertical" — вертикальное, соединяющее два передних угла.

Эти названия позволяют связать целочисленный параметр с нужным ребром, используя следующую таблицу:

Bit	plot	splot
1	bottom	bottom left front
2	left	bottom left back
4	top	bottom right front
8	right	bottom right back
16	—	left vertical
32	—	back vertical
64	—	right vertical
128	—	front vertical
256	—	top left back
512	—	top right back
1024	—	top left front
2048	—	top right front

Для определения части границы, подлежащей определению, можно задавать любые значения таблицы или их суммы.

Примеры.

1. Нарисовать границы по умолчанию

set border

2. Нарисовать только левую и нижнюю границы в двумерном случае или переднее левое и заднее левое нижние ребра в трехмерном случае

set border 3

3. Нарисовать все ребра

set border 4095

4. Нарисовать все ребра кроме верхних и переднего вертикального

set border 127+256+512

или

set border 1023-128

5. Нарисовать верхнюю и правую границы двумерного изображения и подписать их как оси

unset xtics
unset ytics
set x2tics
set y2tics
set border 12

Вернуться к границе, определенной по умолчанию, и показать заданные параметры границы можно, использовав команды

unset border
show border

10.3.3 Размер области изображения

Изображение может занимать не все окно, куда оно выводится. Будем называть областью изображения ту часть окна, которая отводится под график, всевозможные подписи к нему и поля этой области. Размеры поля изображения задаются командой

set size {{no}square | ratio <r>} | noratio}
{<xscale>,<yscale>}

Параметры <**xscale**> и <**yscale**> являются масштабирующими коэффициентами для поля изображения относительно окна.

Ключ **ratio** создает изображение, в котором отрезки единичной длины по осям Y и X относятся с коэффициентом, определяемым параметром <**r**>, учитывая отношение, заданное параметрами <**xscale**> и <**yscale**>.

Результат применения отрицательных значений параметра <**r**> будет следующий. При **r=-1** эффект будет тот же, что и после использования команды

set view equal xy

Если **r=-2**, то единичный отрезок по оси Y будет в два раза длиннее единичного отрезка по оси X. И так далее.

Ключ **square** эквивалентен **r=1**. Использование двух ключей **noratio** и **nosquare** возвращает к настройкам по умолчанию для используемого терминала, но сохраняет заданные значения <**xscale**> и <**yscale**>. Ключи **ratio** и **square** не влияют на результаты 3D-графики, но применяются для 3D проекций, создаваемых с помощью режима

set view map

Примеры.
1. Создать изображение, имеющее одинаковые маштабы, и сжать его в два раза

set size square 0.5, 0.5

2. Создать изображение, имеющее размер единичного отрезка по оси ординат в два раза больше, чем по оси абсцисс

set size ratio 2

10.3.4 Положение изображения в окне

Положение области изображения в окне задается командой

set origin {<x-origin>,<y-origin>}

Параметры <**x-origin**> и <**y-origin**> являются координатами левого нижнего угла области изображения в координатной системе **screen** (см. 10.2, с. 174). По умолчанию <**x-origin**>=**0** и <**y-origin**>=**0**. Команда **set origin** без параметров возвращает координаты <**x-origin**> и <**y-origin**> к значениям $(0, 0)$.

Показать координаты <**x-origin**> и <**y-origin**> можно, использовав команду

show origin

10.3.5 Поля окна

Поля окна это расстояния между областью изображения и границами окна. Поля окна выбираются автоматически, но есть возможность задать их явно

set lmargin {{at screen} <margin>}
set rmargin {{at screen} <margin>}
set tmargin {{at screen} <margin>}
set bmargin {{at screen} <margin>}
set margins <left>, <right>, <top>, <bottom>

Задание отрицательного значения какого либо поля означает возврат к автоматическому определению этого значения. Величины полей зависят от размеров используемых символов. Для 3D-графиков можно задавать лишь величину левого поля.

Задание величин полей в координатной системе **screen** дает возможность определять ширину полей в относительных величинах к размеру окна. Автоматически значения полей рассчитываются исходя из размеров делений и подписей к ним а также подписей к осям, заголовка, легенды, штампа времени создания, если они находятся вне поля изображения.

Посмотреть заданные поля окна можно, использовав команду

show margin

10.3.6 Поля изображения

При автоматическом масштабировании области изображения есть возможность задавать поля самого изображения, т.е. расстояния от деталей изображения до его границ

set offsets <left>, <right>, <top>, <bottom>

Каждый параметр может быть либо константой, либо выражением. По умолчанию все они полагаются равными нулю. Единицы измерения левого и правого полей равны единице длины оси X, а верхнего и нижнего полей — единице длины оси Y. Другой способ выбора размера полей является относительным: задается

отношение ширины поля к длине соответствующей оси координат. Для этого нужно перед параметром указать ключевое слово **graph**. Положительные значения полей приводят к сжиманию поля изображения, а отрицательные к слабо прогнозируемым эффектам автомасштабирования и наложениям.

10.4 Детали изображения

10.4.1 Линии (опции LINETYPE и LINESTYLE)

Многие шаблоны графиков, а также оси координат, границы и другие элементы изображения изображаются линиями. Для изображения линии нужно задать ее параметры. Естественно, что типов линий может быть много. Поэтому им присваиваются номера, по которым определяются цвет (**linecolor (lc)**) и толщина (**linewidth (lw)**) линии. Помимо этих двух естественных параметров тип линии определяет символ (**pointtype (pt)**), которым изображаются точки на линии (если они предусмотрены шаблоном графика), их размер (**pointsize (ps)**) и частоту нанесения (**pointinterval (pi)**). Опция **linetype (lt)** с числовым параметром позволяет задавать нужный тип линии.

В Gnuplot есть набор стандарных типов линий. Выполнив команду **test** (см. 4.1, с. 86), можно увидеть их шаблоны. Они изображены в столбце правой части окна, появляющегося по этой команде (см. цвет. иллюстр. V). Каждому стандартному типу присвоен номер (от -1 до 21). Например, '1' это символ '+' и цвет "фиолетовый", '2' — 'x' и "зеленый", '3' — '*' и "голубой", '4' — квадратик и "оранжевый", '5' — закрашенный квадратик и "желтый" и т.д. Толщина линии задается целым числом от 1 до 6. Набор линий разной толщины приведены в левом нижнем углу цветной иллюстрации V (с. 236). У линий стандартных типов параметр **linewidth** имеет значение 1.

Можно переопределять стандарные типы линий, меняя в них отдельные параметры. Например,
set linetype 1 lw 2 lc rgb "blue" pt 6
set linetype 2 lw 2 lc rgb "green" pt 8

После выполнения приведенных команд все линии, изображаемые с опцией **lt** равной 1 и 2, будут иметь толщину 2. Цвет у первой линии будет "голубой", а у второй — "зеленый". Точки на этих линиях будут изображаться незакрашеными кружочками и треугольничками соответственно.

Особенно подчеркнем то, что команда **reset** не меняет настроек команды **set linetype**.

Можно задать набор свойств линии, который называется стиль линии (**linestyle (ls)**). Для этого используется следующая команда:

set style line <index>{{linetype | lt}
<line_type>|<colorspec>}
{{linecolor | lc} <colorspec>}
{{linewidth | lw} <line_width>}
{{pointtype | pt} <point_type>}
{{pointsize | ps} <point_size>}
{{pointinterval | pi} <interval>}
{palette}

Заданные свойства линии **ls** определяются по ее номеру **<index>**.

В версии Gnuplot 5.0 появилась возможность задавать набор дискретных символов, периодически повторяя которые, рисуется изображаемая линия:

set dashtype <index> "набор символов"

Допустимыми являются символы: "-" (минус), "_" (знак подчеркивания), "." (точка) и " " (пробел). Необязательный параметр **<index>** задает номер задаваемого набора для дальнейшего использования конкретного **dashtype** в команде **plot** и других командах.

Можно задавать **dashtype** еще одним способом, если он состоит только из точек цвета линии и точек цвета фона (идущие подряд точки образуют непрерывный участок линии):

set dashtype <index> (s1,e1,s2,e2,. . . ,sN,eN)

где **sk** — количество подряд идущих точек цвета линии, а **ek** — количество точек цвета фона. Максимальное число пар (**sk, ek**) — четыре.

Набор **dashtype** можно задавать непосредственно в исполняемой команде. В сочетании с параметрами **linecolor** и **linewidth** опция **dashtype** предоставляет большие возможности для создания разнообразных шаблонов линий. Особенно это актуально в чернобелом варианте.

Примеры задания различных **ls** и **dashtype** можно найти в 4.2.2.

10.4.2 Шрифт (опции FONT и TEXTCOLOR)

Заданием опции

font "<name>{,<size>}"

можно определить шрифт и его размер. Шрифт и его размер по умолчанию задаются равными тем, которые приняты для терминала.

Если задана опция **front**, то надпись будет изображена поверх графических элементов. При опции **back** , которая действует по умолчанию, надпись изображается на заднем плане.

Опция **textcolor<colorspec>** отвечает за цвет надписи. Значение параметра **<colorspec>** может быть выбрано из возможных цветов линий, либо как rgb-цвет, либо как цвет палитры (см. 7.2, с. 135).

Примеры задания **textcolor** (допускает сокращение **tc**).
1. **tc default** — задает цвет, принятый по умолчанию.
2. **tc lt <n>** — задает цвет такой же, как у линии с **lt**, равным **<n>**.
3. **tc ls <n>** — задает цвет такой же, как у линии с **ls**, равным **<n>**.
4. **tc palette cb <val>** — задает цвет палитры с параметром **<val>**.
5. **tc rgb "#RRGGBB"** — задает произвольный цвет 24 битной RGB палитры.

10.4.3 Заливка (опция FILL)

Параметры "заливки" задаются командой

set style fill {empty
| {transparent} solid {<density>}
| {transparent} pattern {<n>}}
{border {lt} {lc <colorspec>} | noboder}

По умолчанию устанавливается опция **empty**, т.е. область фигуры имеет цвет фона. Ключ **solid** означает сплошное закрашивание фигуры, а параметр <**density**> задает плотность заливки. При <**density**> равной 0.0 получается "пустая" заливка, при 1.0 — цвет будет таким же, как он указан в текущем **linetype**. Непрерывность или дискретность измененения плотности заливки в зависимости от значения <**density**> зависит от типа используемого терминала. По умолчанию значение параметра <**density**> равно 1.0.

Ключ **transparent** означает "прозрачность". При его задании через область фигуры видны элементы изображения, нарисованные до нее. Возможные значения опции **pattern** от 0 до 8 (см. цвет. иллюстр. V).

Ключ **pattern** означает придание блеска (покрытие лаком) заливки. Эта функция поддерживается далеко не всеми терминалами.

По умолчанию граница фигуры рисуется цветом, заданным в текущем <**linetype**>. Задание **border** <**colorspec**> позволяет выбрать для обозначения границы нужный цвет. При задании ключа **noborder** линия границы не рисуется.

10.5 Форматы ввода-вывода

10.5.1 Спецификации числовых форматов

В Gnuplot приняты следующие спецификации для числовых форматов:
%f — формат с плавающей точкой;
%e или **%E** — экспоненциальный формат с символом '**e**' или '**E**' перед степенью;

%g или **%G** — укороченный вариант формата **%e** (или **%E**) и **%f**;
%h или **%H** **%g** — формат, использующий '$\times 10^{\%S}$' или '$*10^{\%S}$' вместо '**e%S**;
%x или **%X** — формат для шестнадцатеричных чисел с маленькими или большими буквами ;
%o или **%O** — формат для восмеричных чисел;
%t — мантиса к основанию 10;
%l — мантиса к основанию текущего логарифмического масштаба оси;
%s — мантиса к основанию текущего логарифмического масштаба оси, научный формат записи степени;
%T — степень к основанию 10;
%L — степень к основанию текущего логарифмического масштаба оси;
%S — научный формат записи степени;
%c — формат, где используется научный формат записи степени;
%b — мантиса в ISO/IEC 80000 записи (ki, Mi, Gi, Ti, Pi, Ei, Zi, Yi);
%B — префикс в ISO/IEC 80000 записи (ki, Mi, Gi, Ti, Pi, Ei, Zi, Yi);
%P — кратный π.

Научный формат записи степени **%c** допускает степени в пределах от -18 до +18. Это в три раза больше обычного формата. Для записи чисел вне этих пределов используется формат степени в экспоненциальном виде.

Перечислим возможные модификаторы, которые можно использовать после знака '%' до спецификации формата.
1. Знак '-' указывает на левое выравнивание чисел, т.е. числа печатаются начиная с левого края отведенного для них места. Без знака '-' числа печатаются вплотную к правому краю.
2. Знак '+' означает обязательное включение знака числа в его запись.
3. Знак пробела ' ' означает, что для положительных чисел вместо знака '+' будет стоять пробел.
4. При указании знака '#' число записывается с мантисой в ви-

де $0.n_1 \dots n_m$.
5. '0' (цифра, а не буква) — заполняет все пустые позиции, отведенные под запись числа, нулями.
6. (m.n), где m и n целые числа — под число будет отведено минимум m позиций, из которых n знаков будет после десятичной точки. Для вывода целых чисел (без десятичной точки) следует указывать n=0. Если у числа дробная часть содержит знаков больше, чем n, то число округляется.

Примеры.
1. Метки к делениям вида '5.0' и '1.0' устанавливаются командами

set format y "%**2.1t**"
set ytics (**5**, **10**)

2. Метки к делениям вида '500' и '1.0' устанавливаются командами

set format y "%**4.1s**"
set ytics (**500**, **1000**)

3. Метка к делению вида '+12345.000 ' устанавливается командами

set format y "%+**−12.3f**"
set ytics (**12345**)

4. Метка к делению вида '$1.23 * 10^{+04}$' устанавливается командами

set format y "%.**2t** ∗ **10**^{%+**03T**}"
set ytics (**12345**)

5. Метка к делению вида '$12.345 * 10^{3}$' устанавливается командами

set format y "%**6.3s** ∗ **10**^{%**S**}"
set ytics (**12345**)

6. Метка к делению вида '12.345 kg' устанавливается командами

set format y "%**6.3s**%**cg**"
set ytics (**12345**)

7. Метка к делению вида '2 pi' устанавливается командами

set format y ″%.**0Ppi**″
set ytics (**6.283185**)

8. Метка к делению вида '50%' устанавливается командами

set format y ″%.**0f**%%″
set ytics (**50**)

10.5.2 Спецификации данных типа "время-дата"

Gnuplot позволяет использовать время и дату как вводимую информацию. Любые время и дата однозначно определяются количеством секунд, прошедших с момента условной точки отсчета или недостающих до нее. До пятой версии Gnuplot внутреннее время и даты начинали отсчет в секундах от 1 января 2000 года. Начиная с пятой версии точка отсчета поменялась на **Unix epoch**, которая равняется 1 января 1970 года. Разница между этими точками отсчета равна 946684800 секундам.

Формат переменных типа "время-дата" задается командой

set timefmt ″ < **formatstring** >″ (10.1)

Задаваемый в качестве параметра команды формат определяет способ чтения данных типа "время-дата". Допускаются следующие спецификации:
%d — день месяца, 1 – 31;
%m — месяц года, 1 – 12;
%y — год, 0 – 99;
%Y — год, 4-х значный;
%j — день года, 1 – 365;
%H — час, 0 – 24;
%M — минута, 0 – 60;
%s — секунда с момента 1970-01-01 00:00 UTC (Unix epoch);
%S — секунда, целое при выводе, двоичное или целое при вводе;
%b — сокращенное название месяца (3 символа);
%B — название месяца.

Полный список спецификаций форматов переменных типа "время-дата" приведен в разделе **time/data specifiers** [1].

Любые символы допускаются в строке, но все они должны строго соответствовать заданному формату. Символ табуляции — **\t** — считается допустимым. Бэк-слеш значения (\nnn) конвертируются в символы. При отсутствии разделяющих символов между элементами записи данных типа "время-дата" под форматы **%d, %m, %y, %H, %M** и **%S** отводится по два десятичных символа. Если десятичная точка следует за полем, прочитанным по формату **%S**, то следующие десятичные знаки считаются дробной частью. Под формат **%Y** отводится четыре десятичных знака, под **%j** — три, под **%b** — три символа, а под **%B** столько, сколько требуется.

Пробелы трактуются поразному. Пробел в строке символов ставится для правильной трактовки слитного или разделенного несколькими пробелами набора в файле. Например, формат **%H %M** может быть использован для чтения "1220" и "12 20" в одно выражение "12 20".

Каждое непустое множество символов в формате "время-дата" считается одним элементом для опции **using**. Так строка **11:11 25/12/76 21.0** состоит из трех элементов. Во избежании недоразумений, считается, что пользователь полностью соблюдает правила заполнения файлов при использовании опции **using**.

При чтении года с помощью формата **%y** величины 69-99 относятся к 20-му веку, величины 00-68 к 21-му веку.

Пример. Команда

set timefmt "%d/%m/%Y\t%H:%M"

читает дату и время, разделенные символом табуляции. В случае такого формата нужно внимательно следить за тем, что в файле действительно между датой и временем стоят не просто пробелы, а символ **\t**.

Команда **set timefmt** определяет формат для всех вводимых данных: файлов с данными, диапазонов, делений на осях, надписей — короче, для всего того, что может использовать формат "время-дата".

код	буква	код	буква	код	буква	код	буква
A	Alpha	N	Nu	a	alpha	n	nu
B	Beta	O	Omicron	b	beta	o	omicron
C	Chi	P	Pi	c	chi	p	pi
D	Delta	Q	Theta	d	delta	q	theta
E	Epsilon	R	Rho	e	epsilon	r	rho
F	Phi	S	Sigma	f	phi	s	sigma
G	Gamma	T	Tau	g	gamma	t	tau
H	Eta	U	Upsilon	h	eta	u	upsilon
I	Iota	W	Omega	i	iota	w	omega
K	Kappa	X	Xi	k	kappa	x	xi
L	Lambda	Y	Psi	l	lambda	y	psi
M	Mu	Z	Zeta	m	mu	z	zeta

Рис. 42: Таблица кодов греческих символов

10.5.3 Символьные строки

Символьная строка **string** состоит из последовательности символов из расширенной таблицы ASCII кодов, в которую могут быть включены числовые форматы и форматы переменных типа "дата-время". Символьные строки заключаются либо в одинарные, либо в двойные кавычки (см. 11.3).

Для включения символов греческого алфавита в символьную строку требуется вставить команду **{/Symbol <xxx>}**, где код <xxx> для конкретной буквы можно узнать из таблицы 42. Например, для вставки буквы α требуется набрать **{/Symbol a}**.

Для размещения изображения символьной строки в виде двух строк допускается использование символа перехода на новую строку '\n'. Для создания нескольких пробелов можно использовать символ табуляции '\t', а для вывода символа '%' его следует напечатать дважды.

Можно использовать специальные **postscript**-символы. Для этого следует установить пакет **gnuplot-doc**, найти в файле

ps\guide.ps код нужного символа и указать этот код в команде /**Symbol** <**xxx**>. Например, поставить знак процента в подписи к оси X можно командой

set xlabel "X{/Symbol \\045}"

Каждая символьная строка выводится одним стилем, который указан в команде, осуществляющий этот вывод. Можно часть символьной строки выделить, напечатав ее другим шрифтом. Для этого нужно заключить нужные символы в фигурные скобки и внутри этих скобок задать нужный шрифт

set xlabel "axe X ({/Times-Italic axe X})"

При указании опции **enhanced** вывод символьных строк осуществляется по правилу. При встрече знака '^' следующий знак (или группа знаков, заключенная в фигурные скобки) печатается как верхний индекс, а при встрече знака '_' — как нижний (см. раздел **enhanced text mode** [1]).

10.5.4 Вывод символьных строк

В Gnuplot есть две команды, позволяющие создавать символьные строки

gprint(″**format**″, **x**)
sprint(″**format**″, **var1**, **var2**, ...)

Первая команда в качестве значения создает символьную строку, используя формат, написанный по стандартам Gnuplot. В эту строку допускается включать значение одной переменной **x**. Заметим, что такой формат может отличаться от форматов языка Си потому, что число форм записи чисел в Gnuplot гораздо более разнообразное.

Вторая команда также создает символьную строку. В этой команде допустимо использовать несколько значений переменных, но формат, используемый для этого, должен быть написан по стандартам языка Си.

Обе команды применяются при создании символьных строк для разнообразных надписей, выводимых на поле изображения (см. примеры 1.8, 1.9, 3.4).

10.6 Способы измерения углов

Выбор способа измерения углов осуществляется командой

set angles { degrees | radians }

Можно сделать запрос о том, каким способом измеряются углы, использовав команду **show angles.**

Значения углов используются в виде параметра у опции **rotate** . Например, следующие три записи определяют один и тот же угол

rotate by 1.5708; **rotate by 0.5pi;** **rotate by 90deg**

10.7 Независимые переменные

Для определения того, какие переменные считать независимыми, используется следующая команда

set dummy {<dummy-var>} {,<dummy-var>}

По умолчанию команда **plot** считает независимой переменной **t** при параметрическом задании функции или задании функции в полярных координатах и **x** в остальных случаях. Аналогично, команда **splot** считает независимыми переменными **u** и **v** при параметрическом способе задания функции и **x** и **y** в остальных случаях.

Посмотреть, какие переменные объявлены независимыми, и вернуться к используемым по умолчанию переменным можно, использовав команды **show dummy** и **unset dummy**.

10.8 Сетка

Сеткой называется набор линий, для каждой из которых одна из координат постоянна. Значения этих констант определяются заранее выбранными большими и малыми делениями на осях координат (см. 5.5, с. 117 и 5.6, с. 120). Тип и ширина линий могут задаваться отдельно для этих двух типов делений. Линии сетки автоматически пропадают или наоборот появляются при

удалении или определении соответствующих делений на осях. В полярных координатах рисуются круги соответственно выбранным делениям по оси r и радиальные лучи с определенным шагом, заданным в градусной или радианной мере. По умолчанию шаг угла равен 30^o. Линии сетки задаются командой

set grid {{no}{m}xtics} {{no}{m}ytics} {{no}{m}ztics}
{{no}{m}x2tics} {{no}{m}y2tics}
{{no}{m}cbtics}
{polar {<angle>}}
{layerdefault | front | back}
{{linestyle <major_linestyle>}
| {linetype | lt <major_linetype>}
{linewidth | lw <major_linewidth>}
{ , {linestyle | ls <minor_linestyle>}
| {linetype | lt <minor_linetype>}
{linewidth | lw <minor_linewidth>} } }

При указанном ключе **front** сетка рисуется на переднем плане, а в случае **back** — на заднем плане. Задание ключа **front** позволяет избежать загораживания сетки непрозрачной поверхностью. По умолчанию для двумерных графиков задан ключ **back**. В трехмерном случае алгоритм нанесения линий сетки зависит от подключения режима **hidden3d** (см. 6.3, с. 130).

Линии сетки по оси Z рисуются снизу от поверхности. Они смотрятся лучше, если при этом границы области изменения переменных наносятся лишь частично (см. 10.3.2, с. 176).

Вернуться к определяемым по умолчанию параметрам сетки и узнать установленные на момент запроса параметры можно, использовав команды **unset grid** и **show grid**.

11 Командные файлы

11.1 Константы и переменные

Для задания функций с помощью формул требуется набор элементарных инструментов. В первую очередь это константы и переменные (**variable**). Они бывают либо вещественные, либо целые. Вещественная константа отличается от целой наличием десятичной точки. Переменная имеет тот тип, который был у константы, значение которой было присвоено переменной. Операция присвоения обозначается знаком '='. Если все операнды были целыми, то результат арифметической операции имеет целое значение. Поэтому операция деления целых констант или переменных на целочисленные производится по правилам целочисленной арифметики.

Допустимы еще комплексные переменные. Их значения записываются в виде двух действительных чисел, разделенных запятой и заключенных в общие фигурные скобки. Например, комплексное число **3+2i** представляется в виде **{3,2}**.

Две переменные имеют заранее определенные значения. Это **pi**, значение которой равно 3.14159... и **NaN** — бесконечность (Not a Number).

Правила записи арифметических выражений аналогичны правилам, принятым в языках программирования. Примерами **унарных (unary)** операций служат унарный минус '**-**', унарный плюс '+' и знак '**!**', который означает логическое отрицание, если стоит перед переменной, или вычисление факториала, если он стоит после переменной. Знак '**~**', стоящий перед переменной, означает взятие дополнения к ее значению. Важной унарной операцией служит '**$**'. Эта операция используется при работе с файлами. Например, **$n** означает, что вместо **$n** будет использовано число, стоящее в n-ом столбце текущей строки.

Перечислим основные **бинарные (binary)** операции, доступные в Gnuplot.

**** — возведение в степень.

+, -, *, / — арифметические операции (результат деления зависит от типа аргументов).

% — остаток от деления первого аргумента на второй (оба аргумента должны быть целыми).

==, != — сравнение двух чисел на равенство или неравенство.

<, <=, >, >= — знаки меньше, меньше или равно, больше, больше или равно.

&, ∧, | — побитные "и", "или" и "исключающее или".

&&, || — логические "и" и "или".

В Gnuplot есть единственный оператор с тремя аргументами (так называемый **тернарный (ternary)** оператор). Он имеет формат **a?b:c** и работает точно так же, как его аналог в Си. Сначала вычисляется значение функции, указанной в качестве первого аргумента (ее значение должно быть целым числом). Если оно ненулевое, то результатом работы будет вычисленное значение второго аргумента. В противном случае возвращается вычисленное значение третьего аргумента.

Описание операций можно найти в разделах документации по Gnuplot **Unary** , **Binary** и **Ternary**, доступ к которым открывается командой **help**. Правила работы с константами и трактовка различных выражений более полно описаны в разделах **Operators** и **String constants and string variables** [1].

11.2 Встроенные функции

Основные математические функции имеют имена, используя которые можно вычислить их значения. Приведем наиболее часто используемые.

$abs(x)$ – возвращает модуль аргумента. Для вещественных и комплексных аргументов результат будет вещественным, для целых – целым.

$acos(x)$, $asin(x)$, $atan(x)$, – вычисление значений арккосинуса, арксинуса и арктангенса.

$acosh(x)$, $asinh(x)$, $atanh(x)$ – вычисление значений гиперболических арккосинуса, арксинуса и арктангенса.

$arg(z)$ – возвращает аргумент комплексного числа. Результат выдается в радианах или градусах (формат устанавливается командой *set angles*).

$ceil(x)$ – возвращает наименьшее целое число, большее x. Для комплексных аргументов результатом будет наименьшее целое число, большее действительной части x.

$cos(x)$, $sin(x)$, $tan(x)$ – вычисление значений косинуса, синуса и тангенса.

$cosh(x)$, $sinh(x)$, $tanh(x)$ – вычисление значений гиперболических косинуса, синуса и тангенса.

$erf(x)$ – вычисление функции ошибок $Erf(x) = \int_{-\infty}^{x} e^{-t^2} dt$. Если x - комплексное, то функция вычисляется только для действительной части.

$erfc(x)$ – вычисление $1 - Erf(x)$.

$inverf(y)$ – вычисление функции, обратной к $Erf(x)$.

$exp(x)$ – вычисление экспоненты.

$floor(x)$ – возвращает целую часть x.

$log(x)$, $log10(x)$ – вычисление натурального и десятичного логарифма.

$sqrt(x)$ – вычисление квадратного корня.

$sgn(x)$ – вычисление знака аргумента.

Приведем еще несколько полезных функций Gnuplot.

column(n) – возвращает значение столбца с номером n.

columnhead(n) – возвращает первый элемент столбца с номером n в виде символьной строки.

time(x) – возвращает текущее системное время.

Полный набор доступных в Gnuplot математических функций содержится в разделе **Functions** [1].

11.3 Синтаксис командных строк

В командах между опциями и любыми сопровождающими параметрами вставляются пробелы, в то время как списки и координаты разделяются запятыми. Диапазоны изменения величин задаются минимальным и максимальным значением, разделенными двоеточием, и заключаются в квадратные скобки.

Gnuplot использует три типа кавычек (quotes): двойные (ascii 34 — "), одинарные (ascii 39 — ') и обратные (ascii 96 — ‘).

Символьные строки и имена файлов обрамляются в одинарные кавычки или двойные кавычки, что почти всегда приводит

к одному результату. Отметим лишь разницу в трактовке символа \n. Так строка

"Это первая строка текста. \n Это вторая строка текста."

будет воспринята как

Это первая строка текста.
Это вторая строка текста.

В случае, если обрамление будет выполнено из одинарных кавычек

'Это первая строка текста. \n Это вторая строка текста.'

то результат будет следующий

Это первая строка текста. \n Это вторая строка текста.

Обратные кавычки (**backquotes**) применяются при использовании команд системы в командной строке Gnuplot.

Например, создать надпись на графическом изображении, из которой будет следовать время его создания, можно, использовав следующие команды

set label "generated on `date +%Y-%m-%d` by `whoami`" at 1,1
set timestamp "generated on %Y-%m-%d by `whoami`"

Другим примером использования возможности вызова из командной строки Gnuplot команд системы служит вычисление значений функции f(x) с помощью программы **leastsq**. Следующая командная строка будет запускать эту программу и присваивать вычисленное ею значение функции f в точке x

f(x) = `leastsq`

11.4 Условный оператор

Полезной возможностью для написания командных файлов является команда **if**, которая означает обычное ветвление. Эта команда имеет два возможных формата.

if (< **condition** >) < **command** – **line** >
if (< **condition** >) < **command** – **line1** >; \
else < **command** – **line2** >

В первом случае выполнение условия < *condition* > влечет за собой выполнение команды < *command*−*line* >. Во-втором случае при выполнении условия выполняется < *command*−*line*1 >, а в противном случае – < *command* − *line*2 >. Важно помнить, что условие должно всегда быть заключено в круглые скобки. < *command* − *line* > может состоять из группы команд. Команды в группе должны разделяться точкой с запятой. Конец группы определяется либо словом *else*, либо концом строки без знака продолжения \.

При необходимости команду **if** можно расширить на использование трех и более условий. Для этого используется стандарный прием – соединение нескольких операторов в один.

if (< **condition1** >) < **command** – **line1** >; \
else if (< **condition2** >) < **command** – **line2** >; \
else if...
else < **command** – **line** >;

Выполнение конструкции начинается с проверки выполнения условия < *condition*1 >. Если оно оказывается истинным, то выполняется команда < *command* − *line*1 > и управление передается в конец всего оператора. В противном случае выполняется проверка второго условия < *condition*2 > и, если оно истинно, выполняется команда < *command* − *line*2 > и выполнение всего оператора завершается. Этот процесс продолжается до тех пор, пока не будет найдено истинное условие или же не окажется, что все условия ложны. В этом случае будет выполнена команда < *command* − *line* >.

В последних версиях Gnuplot реализован новый формат

условного оператора

if (condition) {<command>;<command>
<commands>
<commands>
} else {
<commands>
}

В ней реализован блочный формат выполняемых команд. Если после слов **if** или **else** следует открывающая скобка '{', то будут выполнены все команды до закрывающей скобки '}'.

11.5 Организация итераций

11.5.1 Опция FOR

Итерации можно организовать неспосредственно в командах **plot**, **splot**, **set** и **unset** с помощью опции **for**. Формат команды зависит от типа условия цикла. Первый тип условия имеет вид

for [<variable> = <start> : <end> {:<increment>}]

Параметр <**increment**>, задающий шаг изменения переменной <**variable**>, является необязательным. По умолчанию он равен 1.

Второй формат имеет вид

for [<variable> in "string of words"]

Параметр **"string of words"** является строкой слов. Например, если такая опция задана в команде **plot**, то содержимое **"string of words"** определяет имена файлов, в которых хранятся координаты точек функций, графики которых требуется нарисовать.

Тело цикла заканчивается либо ближайшей запятой после начала цикла, либо, если запятой нет, концом команды, в которой расположен цикл. Следующие две команды иллюстрируют последнее утверждение

plot for [i = 1 : 3] j = i, sin(j * x)
plot for [i = 1 : 3] j = i sin(j * x)

Результатом выполнения первой команды будет рисунок, содержащий один график функции **sin(3x)**, а второй команды — три графика в одних осях функций **sin(x)**, **sin(2x)** и **sin(3x)**.

Примером использования второго типа циклов может служить следующая команда

plot for [dataset in "v ro"] dataset.".dat" title dataset

При наличии двух файлов данных v.dat и ro.dat будут нарисованы в одних осях графики соответствующих функций. Причем в правом верхнем углу будет указано, какая кривая соответствует v, а какая ro. Это стало возможным за счет использования ключа title, который принимает значения переменной dataset. По умолчанию (при отсутствии явного задания **title**) названия кривых были бы одинаковые – **dataset.".dat"**.

Приведем примеры использования опции **for** в командах **set** и **unset**.

1. В стилях с номерами от 1 до 10 использовать синий цвет

set for [i=1:10] style line i lc rgb "blue"

2. Удалить надписи с номерами от 100 до 200

unset for [tag=100:200] label tag

Допускаются вложенные итерации. Например, создание надписей в точках с координатами (i, j) в виде чисел, вычисляемых по формуле 'i*10+j', происходит по команде

```
set for [i=1:9] [j=1:9] \
label i*10+j sprintf("%d",i*10+j) at i,j
```

Тэг надписи определяется значением выводимого числа.

11.5.2 Циклы

В gnuplot реализована возможность организации циклов.

do for <условие цикла> {
<commands>
<commands>
}

Заметим, что открывающая фигурная скобка тела цикла должна находиться в одной строке с ключевым словом **do**.

К уже описанным для опции **for** "условиям цикла" добавляются конструкции, использующие оператор **if**, а также условие типа **while**

while(<expr>)

Заметим, что конструкция **while** может сама быть использована для организации итераций

while(<expr>) {
<commands>
}

Выполнение конструкции **while** начинается с вычисления значения выражения <**expr**>. Если оно не равно нулю, то команды, заключенные в фигурные скобки, выполняются и снова вычисляется <**expr**>. Так происходит до тех пор, пока значение <**expr**> не окажется равным нулю. В этом случае начинают выполняться команды, следующие за закрывающей фигурной скобкой.

Замечание. В описанных конструкциях **do** и **while** допускается только новый формат оператора **if** с фигурными скобками.

Приведем пример размещения нескольких графиков с помощью режима **multiplot** и оператора **do**.

Пример 11.1

```
set multiplot layout 2, 2
fourier(k,x)=sin(3./2*k)/k*2./3*cos(k*x)
do for [power = 0:3] {
terms = 10**power
set title sprintf ("%g term Fourier series", terms)
```

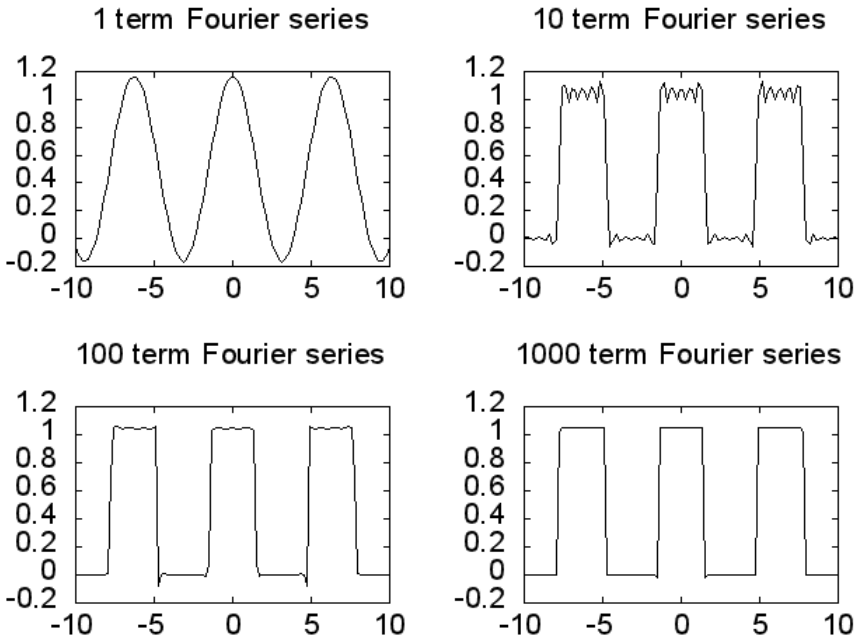


Рис. 43: вывод графиков с помощью оператора цикла

plot 0.5+sum [k=1:terms] fourier(k,x) lc "black" notitle
}
unset multiplot

Команды из примера 11.1 создают изображение, приведенное на рисунке 43.

В примере 11.1 была использована конструкция

sum [<var> = <start> : <end>] <expression>

В результате вычисляется сумма выражений **<expression>** при значениях переменной **<var>** от **<start>** до **<end>**. Совершенно необязательно, чтобы выражение **<expression>** зависело от переменной **<var>** . **<start>** и **<end>** могут быть выражениями, но их значения должны вычисляться до начала вычисления суммы. В случае, если **<start>** больше **<end>**, результат суммирования полагается равным нулю.

11.6 Команда PAUSE

Gnuplot выполняет последовательно все строки из командного файла. Время, затрачиваемое на обработку каждой из строк, очень мало. Иногда это оказывается удобно, но в других ситуациях мешает. "Затормозить" выполнение можно с помощью команды **pause**. Существует два варианта этой команды:

pause $<$ **time** $>$ $\{''<$ **string** $>''\}$
pause mouse $\{''<$ **string** $>''\}$

Первый вариант команды выводит на экран содержимое строки <**string**> и приостанавливает выполнение текущего командного файла на время <**time**>. Если в качестве аргумента <**time**> передано произвольное положительное число, то выполненение приостанавливается именно на такое количество секунд. В качестве значения параметра <**time**> может быть передано значение -1, то есть использован вариант

pause -1 'Please press any key'

В этом случае выполнение приостанавливается до нажатия произвольной клавиши на клавиатуре. Нулевое значение параметра **time** означает нулевую задержку. Можно сказать, что команда

pause 0 'A-A-A'

полностью эквивалентна команде **print**

print 'A-A-A'

Второй вариант команды **pause**, с использованием аргумента **mouse**, вызывает остановку выполнения программы до тех пор, пока не будет нажата кнопка мыши или комбинация 'Control-C'. При этом, правда, необходимо, чтобы существовало графическое окно Gnuplot (то есть был построен какой-либо график).

Другие полезные примеры использования команды **pause** можно найти в разделе **Pause** [1].

11.7 Команда BIND

Начиная с версии 4.0, в Gnuplot добавилась еще одна команда, существенно расширяющая возможности по написанию командных файлов – **bind**.

bind {allwindows} [<key-sequence>] [”<gnuplot commands>”]

Она позволяет сопоставлять определенные комбинации системных событий командам Gnuplot. Например, если в интерактивной сессии выполнить команду

bind a 'print 1'

и затем нарисовать некоторый график, то при нажатой клавише **a** на клавиатуре (при условии, что активным является окно с графиком!) в самой интерактивной сессии будет выводиться на экран единичка.

11.8 Макросы

В Gnuplot реализована возможность создания макросов. Их создание предваряется командой

set macros

После выполнения этой команды могут следовать несколько присвоений символьным переменным **stringvariablename** значений, которые позже можно использовать в качестве аргументов команд Gnuplot, предварив эти имена символом **@**.

Пример.

```
set macros
style1 = "lines lt 4 lw 2"
style2 = "points lt 3 pt 5 ps 2"
range1 = "using 1 : 3"
range2 = "using 1 : 5"
plot "dat1.dat" @range1 with @style1, "dat2.dat"\
@range2 with @style2
```

Эти строки эквивалентны следующей команде:

```
plot "dat1.dat" using 1:3 with lines lt 4 lw 2, \
     "dat2.dat" using 1:5 with points lt 3 pt 5 ps 2
```

Макросы можно использовать в качестве имен часто используемых команд. Например, присвоим переменной A следующее значение

```
set macros
A = "c = 1"
```

После чего команда **@A** будет выполнять присвоение **c=1**. Надо отметить, что создание макроса и его вызов не могут быть расположены в одной строке.

Среди команд Gnuplot есть **exists(stringvariablename)**. Значение этой функции равно 1, если значение аргумента **stringvariablename** было задано, и 0 в противном случае. Следующий пример показывает как можно использовать функцию **exists()** для проверки корректного использования макроса:

```
set macros
C = "pi"
if (exists(C)) print C,"=", @C
```

Имя макроса никогда не заключается в простые или двойные кавычки, но иногда его окаймляют обратными кавычками.

12 Возможности дальнейшего использования

12.1 Сохранение графических изображений

Опишем команды, предназначенные для управления выводом графических изображений. Опция **terminal** определяет, куда именно Gnuplot будет выводить построенный график. По умолчанию это значение устанавливается таким образом, что пользователь получает график на мониторе. Если же получаемый график планируется использовать в дальнейшем, то есть возможность сохранить данные в формате файла, имеющего нужный графический формат, или же в виде описания графика командами **LATEX**, создающими окружение **picture**. Об этом подробнее написано в разделе 12.2.

Примеры.

set terminal postscript – данные будут сохранены в формате PostScript.

set terminal X11 – X11 - это имя дисплея в операционной системе Linux, так что после этой команды изображения будут выводиться на монитор.

set terminal windows – то же самое, что и в предыдущем случае, но только в опрерационных системах Windows.

Опция **output** задает имя файла или устройства для вывода данных. По умолчанию она задана в виде дисплея компьютера, но если нет желания наблюдать текст PostScript на экране, то следует заранее выполнить команду

set output 'file.name'

Команду задания **output** надо выполнять перед каждым рисованием, иначе все графики будут записаны в один файл.

Полное описание всех возможностей по сохранению изображения можно найти в разделе **Complete list of terminals** [1].

12.2 Вставка графических изображений в тексты системы LaTeX

Графические интерпретации бывают нужны как отдельные изображения, предназначенные для анализа тех или иных результатов, так и в виде иллюстраций, включенных в текст доклада, статьи или книги. Одним из способов создания сложных по своей структуре и способам компановки материала является система **LaTeX**. Опишем основные возможности, существующие по включению графических изображений, созданных с помощью Gnuplot, в тексты, набранные в этой системе.

12.2.1 Окружение PICTURE

Gnuplot предоставляет возможность сохранить изображения в формате окружения **picture**, описанного непосредственно командами LaTeX. Для этого при создании изображения с именем **picture** нужно задать вывод изображения следующим образом:

```
set terminal latex
set output 'picture.tex'
```

В результате чего будет создан файл **picture.tex**, который можно вставить в нужное место текста, набранного в системе LaTeX, с помощью команды

```
\input picture.tex
```

Этот путь обладает существенными недостатками. Во-первых, цвет рисунка будет черно-белым даже на экране комьютера при просмотре видеообраза. Во-вторых, при этом теряется возможность сохранить ряд размеров, которые есть возможность задать в Gnuplot. Например, размер символов точек на графиках. Возможная толщина линий также будет определяться стандартами языка TeX, а не возможностями Gnuplot.

Если рисунок вставлять описанным выше способом, то его местоположение четко задано. Для того, чтобы не возникало

пустых строк на странице, в этом случае нужно в ручную подбирать удачное место вставки. Эта процедура может быть выполнена автоматически компилятором LaTeX, если воспользоваться окружением **figure**

```
\begin{figure}
\input picture.tex
\caption{название рисунка}
\label{picture_label}
\end{figure}
```

Команда **\caption{название рисунка}** создает подпись под вставленной картинкой в виде символьной строки **'название рисунка'**. Команда **\label{picture_label}** определяет метку рисунка **picture_label**. Используя эту метку, в любом месте текста можно сослаться на рисунок, написав **\ref{picture_label}**.

Заметим, что окружение **figure** является универсальным инструментом для автоматического подбора места вставки картинки и может быть использовано во всех случаях, описанных ниже.

12.2.2 Пакет GRAPHICX

Отмеченных недостатков при вставке рисунков с помощью окружения **picture** можно избежать, если сохранять их в виде графических файлов и вставлять, используя возможности дополнительного пакета **graphicx**. Для этого необходимо в начале основного файла включить команду

\usepackage{graphicx}

Далее в нужном месте следует вставить графический файл **file.name** с помощью команды

\includegraphics[keyval-list]{file.name}

Необязательный аргумент **[keyval-list]** может содержать целый список ключей. Значения ключей задаются в виде

key=value, а в списке они перечисляются через запятую. Команда **includegraphics** не заканчивает абзац, поэтому позволяет вставлять небольшие рисунки прямо внутрь текста.

Вставляемые рисунки можно поворачивать на определенный угол, задавая для этого необязательный параметр **angle**. Например, картинка, хранящаяся в файле picture.ext, будет повернута на 60^o, если использовать команду

includegraphics[angle=60]{picture.ext}

По умолчанию LaTeX ищет файлы с рисунками в каталоге, в котором находится входной файл. Декларация

includegraphics{dir-list}

позволяет расширить область поиска. В списке **{dir-list}** каждая из директорий заключается в фигурные скобки. Если указать, например,

includegraphics{{images/}{d:/images/eps/}

то LaTeX будет искать файлы с рисунками также в подкаталоге **images** текущего каталога, где находится входной файл, и директории **d:/images/eps**.

Замечание. Поскольку EPS-файл является текстовым файлом, его можно хранить в окружении **filecontents*** из стандартного LaTeX'a.

Gnuplot допускает сохранение создаваемых с его помощью изображений в виде графических файлов, использующих разнообразные форматы записи. Опишем некоторые из этих форматов и охарактеризуем основные различия между ними и особенности вставки графических файлов в зависимости от формата в тексты LaTeX.

Начнем с того, что изображения делятся на две группы. Первая группа это **растровые** изображения, вторая — **векторные**. **Растровое** изображение — изображение, представляющее собой матрицу пикселей (цветных точек) на мониторе, бумаге или другом отображающем устройстве. **Векторная** графика — способ представления объектов и изображений в компьютерной

графике, основанный на использовании элементарных геометрических объектов, таких как точки, линии, сплайны и многоугольники. Объекты векторной графики являются графическими изображениями совокупности математических функций. Растровые изображения плохо масштабируются, тогда как векторные изображения могут быть неограниченно увеличены или уменьшены без потери качества.

12.2.3 Формат PNG

Формат **PNG** (portable network graphics) — растровый формат, созданный в 1995 году и использующий сжатие без потерь по алгоритму **deflate**. Для сохранения картинки в этом формате требуется выполнить команды

```
set terminal png
set output 'picture.png'
```

В результате будет создан файл picture.png, который может быть вставлен в текст командой

```
\includegraphics{picture.png}
```

Необходимо отметить, что поскольку PNG-формат является растровым, то скомпилировать документ, в который включен такой файл, с помощью команды **latex** не удается. Это следует делать командой **pdflatex**.

12.2.4 Формат EPS

Формат **EPS** (encapsulated PostScript) — формат файлов, базирующийся на подмножестве языка PostScript и предназначенный для обмена графическими данными между различными приложениями. Формат EPS был создан компанией Adobe. В

своей минимальной конфигурации EPS-файл содержит информацию, описывающую размер изображения. Формат используется в профессиональной полиграфии и может содержать растровые изображения, векторные изображения, а также их комбинации. Для создания Gnuplot файла **picture.eps** нужно выполнить команды

set terminal epslatex
set output 'picture.eps'

Опции команды **includegraphics** можно опустить, если рисунок вставляется в "натуральную величину".

Размер рисунка задается для EPS-файла в строке

%%BoudingBox: llx lly urx ury

где целые числа **llx**, **lly**, **urx** и **ury** — это x- и y- координаты в больших пунктах (bp)[5] левого нижнего и правого верхнего углов области, внутри которой находится рисунок на "воображаемой" странице. Именно эта часть страницы импортируется в документ. LaTeX читает из EPS-файла только параметры Bounding Box для того, чтобы знать сколько места на странице надо отвести под рисунок. Рисунок из EPS-файла читает и вставляет в документ драйвер **dvips**, когда переводит **DVI** в **PS**.

Например, пусть в EPS-файле задано

%%BoudingBox: 12 18 84 162

Следовательно, точка отсчета рисунка (левый нижний угол) находится на расстоянии 12 bp и 18 bp от левой и нижней границ страницы, а сам рисунок имеет размеры 72 bp на 144 bp по горизонтали и вертикали соответственно.

При желании размер области, которую LaTeX отводит под размещение рисунка, определяется с помощью параметров **width** и **height**, которые можно задавать в любых TeX'овских единицах. Например, команда

\includegraphics[width=1in, height=10mm]{picture.eps}

[5] 72 bp равны 1 дюйму или 2.54 см.

определяет ширину области для изображения рисунка, равную 1 дюйму, а высоту 10 миллиметрам. При этом исходный рисунок будет соответствующим образом отмасштабирован. Если размеры области размещения были заданы непропорционально размерам рисунка, то изображение может существенно исказиться. Для того, чтобы этого избежать, можно указывать параметр **keepaspectratio**. Задание **keepaspectratio** приводит к равномерному сжатию или растяжению рисунка, а из параметров **width** и **height** выбирается тот, при котором данное преобразование осуществимо.

Есть еще один способ изменить размер рисунка при вставке — указывать параметр масштабирования (**scale**)

\includegraphics[scale=k]{picture.eps}

Если значение k больше единицы, то размер изображения увеличивается в k раз, а если k<1, то происходит сжатие в k раз.

Трансформировать рисунок к указанному размеру можно также командой

\resizebox{width}{height}{lr-text}

Эта команда помещает текст **lr-text** в бокс и затем сжимает или растягивает бокс вместе с содержимым так, что его ширина и высота стали равными **width** и **height**.

Сохранить отношение высоты к ширине бокса можно, указав в качестве **width** или **height** восклицательный знак '!'

\resizebox{3 cm}{!}{lr-text}

12.2.5 Несколько картинок рядом

Силами LaTeX'а можно разместить несколько графиков рядом. Для этого следует воспользоваться окружением **tabular**. В качестве примера приведем команды, с помощью которых был вставлен рисунок: 28

\begin{figure}
\begin{center}

```
\begin{tabular}{cc}
\resizebox{50mm}{!}{\includegraphics{with1.eps}}&
\resizebox{50mm}{!}{\includegraphics{with2.eps}}\\
\end{tabular}
\end{center}
\vspace{-0.5 cm}
\caption{шаблоны финансовых данных}
\label{fin}
\end{figure}
```

12.2.6 Интеграция Gnuplot в докуметы LaTeX

Команды Gnuplot можно использовать для создания графических иллюстраций непосредственно из документов системы LaTeX. Для этого необходимо в начале документа включить пакет **gnuplottex**, который входит в поставку **TeXlive**

```
\usepackage{gnuplottex}
```

Сборка документа должна выполняться с ключем **-shell-escape**, предоставляющим возможность выполнения команд оболочки. Например,

```
pdflatex -shell-escape document.tex
```

После чего в тело документа можно включать набор команд Gnuplot. Например [5],

```
\begin{figure}[h]
\centering
\begin{gnuplot}
plot sqrt(x) title '$y = \sqrt{x}$'
\end{gnuplot}
\end{figure}
```

Заметим, что при таком использовании Gnuplot для записи формул в легенде или надписях предоставляются возможности LaTeX.

12.3 Формат JPEG

Форматы **PNG** и **EPS** сохраняют изображение без потерь (lossless). Тем не менее борьба за объемы сохраняемой информации привела к появлению большого количества форматов, где часть информации об исходном изображении теряется. Одним из самых распространенных форматов является **JPEG**. В Gnuplot есть возможность сохранять результаты в этом формате

set terminal jpeg

Заметим, что формат **JPEG** (Joint Photographic Experts Group) является одним из популярных графических форматов, применяемых для хранения фотоизображений и подобных им. Алгоритм **JPEG** позволяет сжимать изображения как с потерями, так и без потерь. Этот алгоритм в наибольшей степени пригоден для сжатия фотографий и картин, содержащих реалистичные сцены с плавными переходами яркости и цвета. Наибольшее распространение **JPEG** получил в цифровой фотографии и передаче изображений с использованием интернета.

С другой стороны **JPEG** малопригоден для сжатия чертежей, текстовой и знаковой графики, где резкий контраст между соседними пикселами приводит к появлению заметных артефактов (искажений). Поэтому возможность сохранять результаты работы Gnuplot в этом формате является скорее признанием его распространенности, чем насущной необходимостью.

12.4 Формат PDF

Формат **PDF** (Portable Document Format) — формат файлов, разработанный фирмой Adobe Systems и опубликованный в 1993 г. Этот межплатформенный формат электронных документов в первую очередь предназначен для предоставления полиграфической продукции в электронном виде. Значительное количество современного профессионального печатного оборудования имеет аппаратную поддержку формата PDF, что позволяет производить печать документов в данном формате без использования какого-либо программного обеспечения. Чаще

всего PDF-файл является комбинацией текста с растровой и векторной графикой. Для задания PDF-формата получаемого рисунка используется команда

set terminal pdfcairo
{{no}enhanced} {mono|color} {solid|dashed}
{font<font>} {fontscale<scale>}
{linewidth<lw>} {rounded|butt} {dashlength<dl>}
{size <XX>{unit},<YY>{unit}}

Опция **enhanced** позволяет сохранять в PDF-формате текст, набранный с использованием верхних и нижних индексов и других элементов улучшенного набора (**enchanced text mode**).

Ширина всех линий на изображении преобразуется к одной, указанной в параметре <**lw**>. По умолчанию ширина линий равна 0.25 пиксела, где один пиксел "PostScript" равен 1/72 дюйма или 0.353 мм.

Опция **butt**, задаваемая по умолчанию, означает резкие окончания и соединения линий, а **rounded** — плавные.

По умолчанию размеры получаемого изображения равны 5 на 3 дюйма. Пользователь может указать свои размеры, задав параметры опции **size** в дюймах или сантиметрах.

Параметр <**font**> задается в формате "FontFace,FontSize". В качестве названия шрифта (FontFace) используются имена типа 'Arial'. Если название не указано, то применяется шрифт 'Sans'. Размер шрифта задается в пикселах. По умолчанию он равен 12. Однако, по умолчанию <**fontscale**> равен 0.5, поэтому в результате шрифт окажется меньше.

Примеры.

1. Использовать при создании pdf-формата шрифт Arial размером 12

set term pdfcairo font "Arial,12"

2. Изменить используемый шрифт на "Arial"

set term pdfcairo font "Arial"

3. Использовать шрифт 12-го размера

set term pdfcairo font ",12"

4. Использовать шрифт и размер на заданные

set term pdfcairo font " "

Заметим, что при сохранении в форматах **PNG**, **EPS** и **JPEG** также используются различные опции, описание которых можно прочесть в соответствующих разделах [1].

12.5 Общие замечания о сохранении графиков

Для окончательного выбора формата, в котором лучше сохранять ту или иную картинку, нужно определиться со следующими внешними условиями.

Первое. Если конечной целью является текст статьи, который будет отсылаться в определенный журнал, то прежде всего имеет смысл узнать условия, которые выдвигает редакция к присылаемым работам. Многие научные журналы принимают статьи с рисункам в формате EPS.

Второе. Настройки компьютера, на котором будут открывать Ваш файл, могут не соответствовать настройкам Вашего компьютера. Вследствие чего картинки могут неадекватно отображаться в отдельных форматах. Например, может не хватать шрифтов для EPS-формата. В этом случае можно попробовать сохранить файл в другом формате и, используя какую либо программу (например, Adobe Acrobat) для того, чтобы переписать его в нужном формате. Главное, чтобы окончательный вид рисунка Вас устраивал.

Важно использовать обязательно крупный шрифт в метках, подписях и легенде. При верстке журнала Ваши рисунки часто сильно масштабируются, поэтому мелкие подписи становятся трудно читаемыми.

Когда требуется включить несколько рисунков в один, лучше сделать их одного размера.

13 Анимация

По мере развития комьютерной техники у специалистов разных областей науки появляется все больше возможностей донести информацию до своих коллег. Одним из таких вариантов служат небольшие анимационные фильмы. Этот раздел посвящен описанию возможностей создания таких фильмов с помощью Gnuplot и вставке их в свои презентации, доклады и т.п. Удобное размещение видео в текстах является немаловажной составляющей успешного доклада.

Большая часть докладов создается в виде **pdf** или **ppt**-файлов и показывается, используя возможности техники организаторов выступления. Разнообразие применяемой техники и программного обеспечения, установленного на ней, приводит к тому, что включение анимаций в текст является рискованной операцией: фильм может просто не начать проигрываться должным образом. Самым простым и надежным способом показать видео является возможность быстро переключить показ текста выступления на просмотр фильма внешней программой, которая должна быть установлена на компьютере, через который происходит показ доклада. Наличие такой программы является необходимым условием реализации подобного варианта показа видеоинформации. На этом пути можно пойти дальше и вставить в **pdf**-файл ссылку на видеофайл, щелчок по которой автоматически заставляет включаться видео. Конечно, в таком случае многое зависит от настроек аппаратуры организаторов. Эти проблемы решаются безболезненно, если используется компьютер докладчика с заранее установленными необходимыми программами или есть возможность заранее протестировать оргтехнику и добиться от системного администратора полноценной ее работы в случае показа Ваших материалов.

Более универсальная возможность, но гораздо более требовательная к выделяемой памяти — создание анимации в самом **pdf**-файле, содержащем Ваш доклад. Это можно сделать средствами системы LaTeX, использовав специальный пакет **animate**. Для этого требуется набор статических изображений (кадров), из которых будет состоять видео. Все эти кадры в ито-

ге хранятся в **pdf**-файле доклада, что приводит к увеличению его объема, что в свою очередь может затруднить показ. Поэтому в настоящее время этот способ годится лишь для фильмов, состоящих из очень простых картинок малого объема, или для периодического показа небольшого числа изображений. Но назвать этот способ тупиковым нельзя, поскольку темпы развития возможностей компьютерной техники остаются высокими и то, что вчера еще казалось недоступным, сейчас широко используется.

13.1 Повторный запуск командных файлов

В Gnuplot существует возможность повторной загрузки командного файла. Это делается с использованием команды **reread**. Комбинируя **reread** с ветвлением, можно организовать циклы.

Например, требуется проследить изменение графика функции

$$f(x) = \frac{1}{1 + ax^2} \tag{13.1}$$

в зависимости от величины параметра a. Для этого создадим файл с именем **plotter.gn**, содержащий следующие строки:

```
# циклическое рисование
n=n+1
plot 1./(1.+n*x*x)
pause -1 'Press any key to continue'
if (n<10) reread
```

Этот файл будет играть роль тела цикла.

Теперь необходимо создать основной файл программы. Запишем его в файл **cycle.gn**:

```
# главная (стартовая) часть программы
n=1
set xrange [-3:3]
set yrange [0:1]
load 'plotter'
```

Выполнение поставленной задачи можно осуществить, набрав в

командной строке **gnuplot 'cycle.gn'**. Очередная картинка будет рисоваться после нажатия на любую клавишу в окне, где была выполнена эта команда.

Существует еще один вариант загрузки файлов, кроме **load** – команда **call**

call '<input-file>' <param-1>... <param-9>

Используя ее, можно передавать до девяти дополнительных параметров. Все параметры должны быть символьными строками. По этой команде начинается выполнение вызванного файла **<input-file>**. Встречая в командах этого файла имена **ARG0**, **ARG1**, ..., **ARG9**, Gnuplot заменяет их на соответствующие константы, переданные в команде **call**. Если требуется использовать передаваемый параметр не как символьную константу, а как аргумент команды, то нужно просто обратиться к этому имени как к макросу (см. 11.8, с. 203).

Пример 13.1

Пусть файл **b.gnu** содержит следующие команды
plot ARG1 with lines title ARG3
print ARG2 * 3.1, ARG2 * 3.1
print "This plot produced by script", ARG0
В интерактивном режиме присвоим значения двум переменным в виде символьных строк и выполним этот файл, набрав команду **call**
MYFILE = "b.gnu"
FUNC = "sin(x)"
call MYFILE FUNC 1.5 "This is a plot title"

В результате будет построен график функции **sin(x)** с легендой **"This is a plot title"**. После чего в окне, где запущен Gnuplot, будут напечатаны две строки сообщений. Первая строка содержит два одинаковых числа, равных 4,65. Это сделано для демонстрации того, что третий параметр функции, как один из операндов арифметического выражения, воспринимается как число в обоих случаях: когда к нему обращаются как к символьной строке и как к макросу. Для переменных это не проходит.

Наконец, вторая строка сообщения содержит надпись: **This plot produced by script b.gnu**.

До версии Gnuplot 5.0 действовала другая интепретация команды **call** (см. раздел Call (old-style) в [1]).

13.2 Простейшая анимация

С помощью команды reread легко можно сделать "движущуюся" картинку. Начнем с простого примера. Заменим в файле **plotter.gn** из раздела 13.1 команду

pause -1 'Press any key to continue'

на команду

pause 0.25

В результате после команды **gnuplot 'cycle.gn'** на экране возникнет короткая анимация, иллюстрирующая изменение графика функции (13.1) в зависимости от значения параметра a. От величины параметра команды **pause** зависит частота смены картинок.

Приведем командные файлы, позволяющие вращать трехмерный график вокруг оси z. Файл move.gnu содержит команды, рисующие поверхность $z = x^2 + y^2$ под циклически меняющимся углом обзора

```
a=a+1
zrot=(zrot+10)%360
set view xrot, zrot
splot x*x+y*y title ' '
pause 0.25
if (a<50) reread
```

Главная программа, запускающая анимацию, состоит из следующих команд:

```
a=0
set xtics (100,200)
xrot=60
```

zrot=0
load 'move.gnu'

Для удобства последующих ссылок будем считать, что эта программа записана в файле move1.gnu.

13.3 Создание анимационного файла

Рассмотрим способ создания файла, содержащего анимацию, с расширением **gif**. **Gif (Graphics Interchange Format)** — часто применяемый формат графических изображений, в котором можно хранить сжатые данные без потери качества в формате не более 256 цветов. Формат **Gif** был разработан в 1987 г. **(Gif87a)** фирмой **CompuServe** для передачи растровых изображений по сетям. Этот формат не зависит от аппаратного обеспечения. В 1989 г. была выпущена модификация **Gif89a**, в которую были добавлены поддержка "прозрачности" и "анимации".

Термин "прозрачность" означает следующее: один из цветов в палитре объявляется "прозрачным". Это означает, что сквозь пиксели, окрашенные прозрачным цветом, будет виден фон.

Термин "анимация" означает, что наряду с набором статичных кадров передается информация о том, сколько времени каждый кадр должен быть показан на экране. Анимацию можно сделать цикличной, когда вслед за последним кадром начнется воспроизведение первого кадра и т.д. **Gif**–анимация может использовать "прозрачность" для того, чтобы не сохранять очередной кадр целиком, а хранить только изменения относительно предыдущего.

Gif использует **LZW**-компрессию, что позволяет сжимать файлы, в которых много однородных заливок (логотипы, надписи, схемы). Этот метод, разработанный в 1978 г. Лемпелем (Lempel) и Зивом (Ziv) и доработанный Велчем (Welch), сжимает данные путем поиска одинаковых последовательностей (они называются "фразы") во всем файле. Выявленные последовательности сохраняются в таблице, им присваиваются более короткие маркеры (ключи). Метод **LZW** лучше действует на участках однородных, свободных от шума цветов.

Синтаксис команды, позволяющей сохранять анимацию в формате **gif**, следующий:

set terminal gif
{{no}enhanced}
{{no}transparent} {rounded | butt}
{linewidth <lw>} {dashlength <dl>}
{tiny | small | medium | large | giant}
{font "<face> {,<pointsize>}"} {fontscale <scale>}
{size <x>,<y>} {{no}crop}
{animate {delay <d>} {loop <n>} {{no}optimize}}
{background <rgb_color>}

Опция **transparent** делает один из цветов "прозрачным" (см. раздел **transparent** в [1]). По умолчанию ни один цвет не является "прозрачным".

Опции **linewidth** и **dashlength** задают масштабирующие множители в командах, рисующих линии.

Опция **butt** позволяет изображать линии, точно совпадающие с координатами заданных концевых точек. Эта опция применяется только для линий, ширина которых больше чем 1, и особенно полезна при изображении вертикальных и горизонтальных линий. По умолчанию выбирается опция **rounded**.

Выбор шрифта может быть сделан разными способами. Например, следующие две команды эквивалентны

set term gif font arial 11
set term gif font ″arial, 11″

Более подробно о выборе шрифтов можно прочитать в разделе **fonts** в [1].

Опция **animate** доступна только в случае, когда библиотека **gd** на Вашем компьютере позволяет создавать анимационные файлы. Частота смены кадров может быть задана в пределах от 1 до 100 секунд (по умолчанию выбирается значение 5). Реальная частота может зависеть еще и от программы, с помощью

которой организовывается просмотр анимационного файла. Количество повторений просмотров задается ключом **loop**, причем значение 0, задаваемое по умолчанию, означает бесконечное число повторов.

Опция **optimize** заключается в следующем.
1. Для всех статических изображений анимации используется однин цветовой набор. Это означает, что все цвета, которые используются в любом кадре, определены в самом первом кадре анимации.
2. По возможности, в анимации хранятся только части очередного кадра, отличающиеся от предыдущего. Это может быть несовместимо с использованием опции **transparency**.

Оба пункта оптимизации приводят к сокращению выходного файла, но это работает лишь в случае очень продолжительных анимаций, либо когда размер кадров относительно небольшой. Выбор ключа **nooptimize** отключает оба пункта оптимизации: каждый кадр сохраняется полностью со своим набором цветов. Отметим, что после создания анимации с ключом **nooptimize** возможно сжатие полученного файла внешними программами.

Ключ **size** имеет два параметра <**x**> и <**y**>. Эти параметры задают размеры в пикселях получающихся кадров (дополнительную информацию можно прочитать в разделе. 10.3.3). Чистые места по краям кадров можно обрезать, использовав опцию **crop**. Тем самым объем получающегося выходного файла становится меньше. По умолчанию задано значение **nocrop**.

Пример. Для создания анимационного фильма, где параболоид $z = x^2 + y^2$ вращается вокруг оси z, можно использовать следующие команды:

```
set term gif animate opt loop 10 size 200, 200
set output 'paraboloid.gif'
load 'move1.gnu'
```

В результате будет создан файл **paraboloid.gif**, при запуске которого цикл из 50 статических изображений, задаваемых в файле **move.gnu**, будет показан 10 раз. Величина окна при этом будет 200 на 200 пикселей, а размер файла **paraboloid.gif** будет уменьшен за счет применения опции **optimize**.

13.4 Создание анимации в пакете LaTeX

13.4.1 Синтаксис

Система LaTeXпозволяет создать из набора статических кадров динамическое изображение [6]. Для этого в начале документа необходимо указать, что дополнительно будут использоваться следующие пакеты:

usepackage{media9}
usepackage{animate}

После чего в теле документа для создания динамического изображения можно использовать команду

\animategraphics[<options>]{<frame rate>}
{<file basename>}{<first>}{<last>}

13.4.2 Последовательность графических файлов

Команда **\animategraphics** создает анимацию из последовательности графических файлов, задаваемой ее аргументами.

Тип используемых файлов должен быть согласован с драйвером, который будет использован для создания конечного документа. Если будет использоваться **LaTeX+ dvips**, то следует использовать формат **'eps'**, **'mps'** (METAPOST-generated Postscript) и **'ps'**. При использовании **pdfLaTeX** или **LuaLaTeX** возможные форматы **'pdf'**, **'mps'**, **'png'**, **'jpg'**, **'jpeg'**, **'jbig2'**, **'jb2'**, **'jp2'**, **'jk2'** и **'jpx'**. При использовании **LaTeX + dvipdfmx** — **'pdf'**, **'mps'**, **'eps'**, **'ps'**, **'png'**, **'jpg'**, **'jpeg'** и **'bmp'**. Выше перечислены расширения в том порядке, в котором они ищутся при компиляции.

Все файлы из последовательности, которая будет использоваться, должны быть пронумерованы. Исключение из этого правила лишь одно — использование опции **'every'**. Ключ **<file basename>** задает начальную часть имен файлов, которая общая для всех членов последовательности. Ключи **<first>**

и <**last**> задают первый и последний номера файлов, которые требуется использовать. Если первый номер больше последнего, то порядок использования будет иметь отрицательный шаг. Наиболее простая нумерация, например, от '0' до '99'. Если какие-то файлы имеют номера, начинающиеся с нулей, то для правильной интерпретации номеров нужно, чтобы они все имели одинаковое число цифр. Например, от '0000' до '0099'. При этом номера <**first**> и <**last**> должны также быть четырехзначными. Например, если из последовательности файлов с именами от 'frame_5.png' до 'frame_50.png' нужно создать анимацию с частотой 12 кадров в секунду, следует использовать команду

animategraphics{**12**}{**frame_**}{**5**}{**50**}

13.4.3 Опции

Перечислим возможные опции команды **animategraphics**.

label=<**label text**> — метка анимации.

type=[<**file ext**>] — тип используемых графических файлов.

poster=[**first** | <**num**> | **last** | **none**] — номер кадра, который следует изобразить в случае, если анимация не активирована.

every=<**num**> — использовать для анимации кадры, начиная с номера <num>. Первый кадр имеет номер '0'.

autopause — остановка анимации на текущем кадре при закрытии страницы с ней. В противном случае происходит возврат к ее началу.

autoplay — автоматический запуск анимации при открытии страницы, где она размещена.

autoresume — автоматическое возобновление показа остановленной анимации в случае повторного открытия страницы с ней.

loop — повторный запуск без задержки анимации после ее окончания.

palindrome — непрерывный показ анимации от первого до последнего кадра и наоборот.

step — смена кадров происходит по клику мыши. При задании этой опции параметр <**frame rate**> игнорируется.

width=<**h-size**>, **height**=<**v-size**> | **totalheight**=<**v-size**>, **keepaspectratio** — задание размеров окна, где отображается анимация. При отсутствии одного из параметров он задается так, чтобы сохранить пропорции первого кадра.

scale=<**factor**> — масшабирует изображение с заданным коэффициентом.

bb=<**llx**><**lly**><**urx**><**ury**> — четыре разделенных пробелами параметра задают ограничения размера изображения графических файлов. Первые два параметра задают смещение изображения в окне, а два последних — его размеры. Если единицы измерения опущены, то считается, что они равны 1 пикселу. Используется только в команде **animategraphics** и требует подключения пакета **'graphics'**.

viewport=<**llx**><**lly**><**urx**><**ury**> — эта опция использует, как и опция **bb**, четыре аргумента, но в этом случае их величины задают относительные размеры к начальным параметрам изображения графических файлов. Требует подключения пакета **'graphics'**.

trim=<**left**><**bottom**><**right**><**top**> — обрезает изображения по краям. Четыре аргумента задают сколько требуется обрезать с разных сторон. Если аргумент отрицательный, это означает добавление пустой полосы. Требует подключения пакета **'graphics'**.

controls — размещает клавиши управления снизу от окна анимации. Значения клавиш справа налево: стоп и показать первый кадр; сделать шаг назад; начать просмотр назад; начать просмотр вперед; сделать шаг вперед; стоп и показать последний кадр; уменьшить частоту кадров; установить частоту кадров, заданную по умолчанию; увеличить частоту кадров. Все клавиши, регулирующие просмотр во время показа анимации, заменяются одной большой клавишей <pause>.

buttonsize=<**size**> — размер клавиш. По умолчанию, 1.44em.

buttonbg=<**colour**>, **buttonfg**=<**colour**> — цвет фона и символов клавиш. Параметр <**colour**> — числа из отрезка

[0, 1], разделенные символом ':'. Число чисел определяет используемую цветовую модель. Для черно-белой модели параметр <**colour**> просто одно число. Например, '1'. Для модели RGB этих чисел уже три. Например, '1:0.5:0.2'. По умолчанию, символы на клавишах черные, а фон — прозрачный.

draft, final — при указанной опции <draft> анимация не включается, хотя место под нее резервируется согласно размерам кадров. При опции <final> анимация включается. Обе опции могут быть полезны на этапе отладки.

nomouse — анимация не откликается на клики мыши.

Параметр <**frame rate**> задает частоту кадров в секунду в создаваемой анимации.

Список литературы

[1] Thomas Williams, Colin Kelley Gnuplot 5.0. An Interactive Plotting Program. 2015. http://sourceforge.net/projects/gnuplot

[2] Пережогин А.С. Gnuplot. Не так часто задаваемые вопросы. www.conference2013.ikir.ru

[3] Конник М. Записки дебианщика. Строим научные графики с помощью Gnuplot. www.mydebianblog.blogspot.de/2006/08/gnuplot.html

[4] Нечаев А.Н. Краткое введение в Gnuplot. //old.tltsu.ru/archive/ graphics/gnuplot/docs/gdoc.pdf

[5] Притыкин Д. Оформление научных результатов: иртеграция LaTeXи Gnuplot. http://habrahabr.ru/250087/

[6] Grahn A. The animate Package. http://tug.ctan.org/macros/latex/contrib/animate/nimate.pdf, 2015.

[7] Green D.A. http://arxiv.org/abs/1108.5083, 2011.

[8] Львовский С.М. Набор и верстка в системе LaTeX. Изд-во МЦНМО, 2003.

[9] Сюткин В. Включение рисунков в LaTeX2$_\varepsilon$. sbras.ru>win/docs/TeX/LaTex2e/Graph_in_LaTeX.pdf, 2001.

[10] Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы М.: Лаборатория Базовых Знаний, 2000 г. - 624 с.

[11] FN Fritsch, RE Carlson Monotone Piecewise Cubic Interpolation. SIAM Jornal on Numerical Analysis 17, 1980, p. 238-246.

Предметный указатель

Попов Анатолий Вадимович

GNUPLOT

и его приложения.

М. Издательство попечительского совета
механико-математического факультета МГУ, 240 стр.

Оригинал макет автора.

Подписано в печать 30.06.2015 г.
Формат 60×90 1/16. Объем 15 п.л.
Заказ 3. Тираж 200 экз.

Издательство попечительского совета механико-
математического факультета МГУ
г. Москва, Воробьевы горы.

Отпечатано с оригинал–макета на типографском оборудовании механико-математического факультета

Цветные иллюстрации

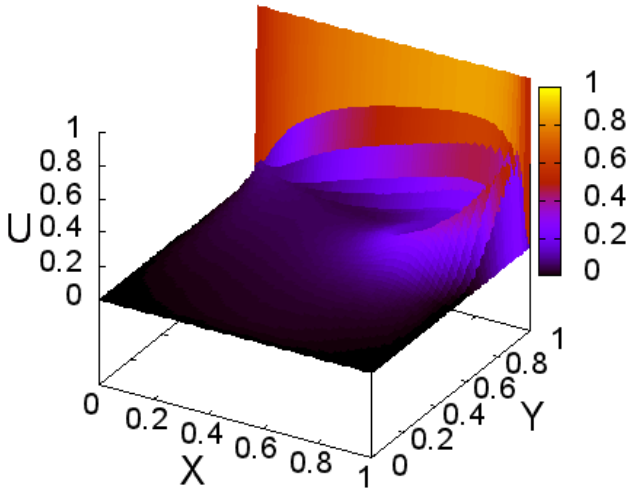


Рис. I. Модуль скорости течения в каверне

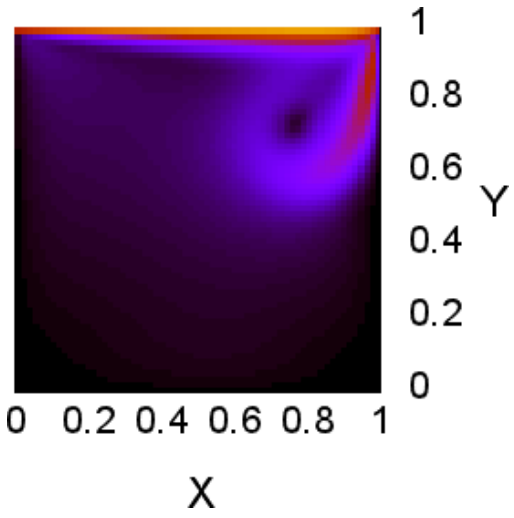


Рис. II. Изображение течения шаблоном pm3d

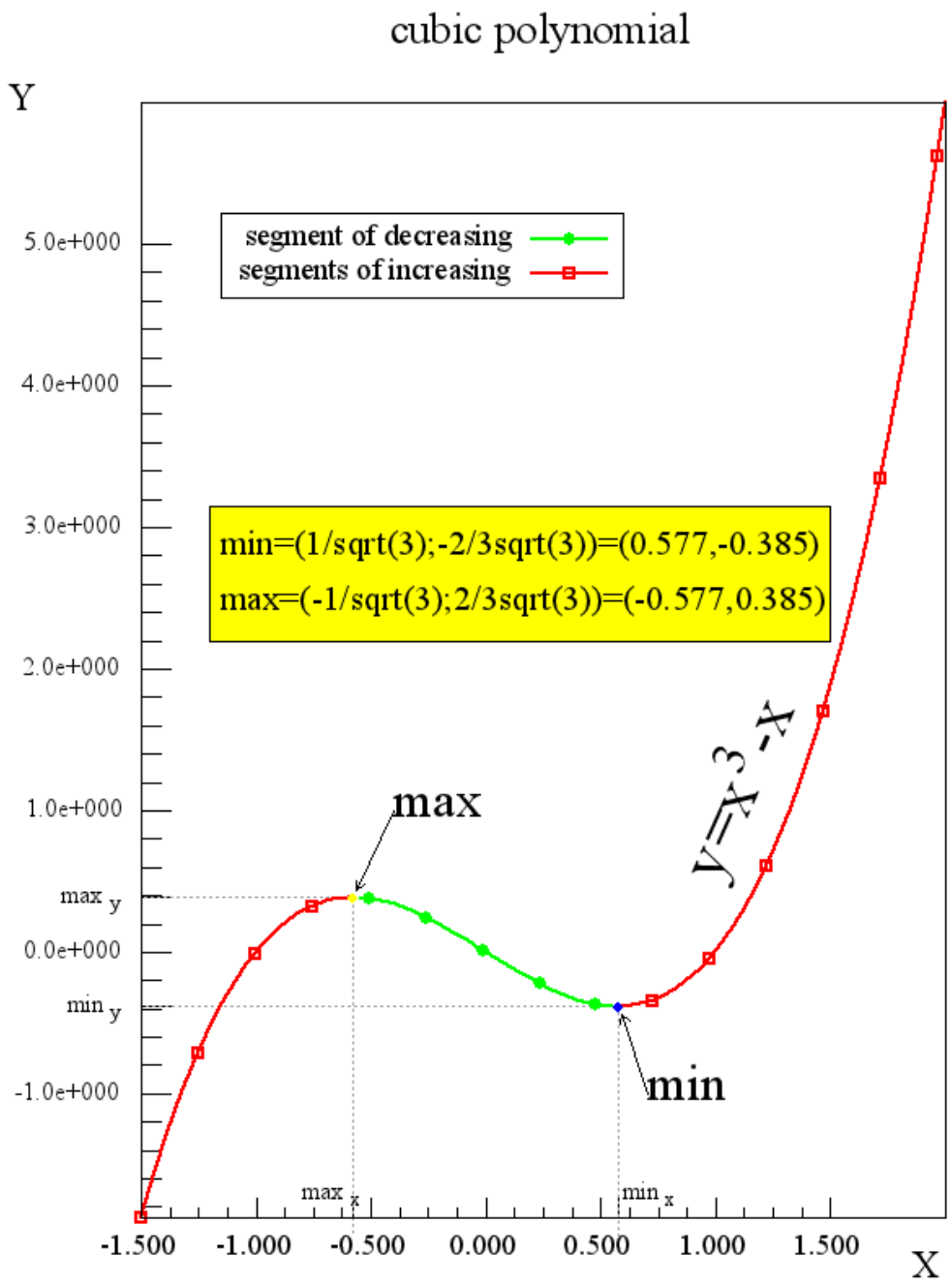


Рис. III. График кубической параболы

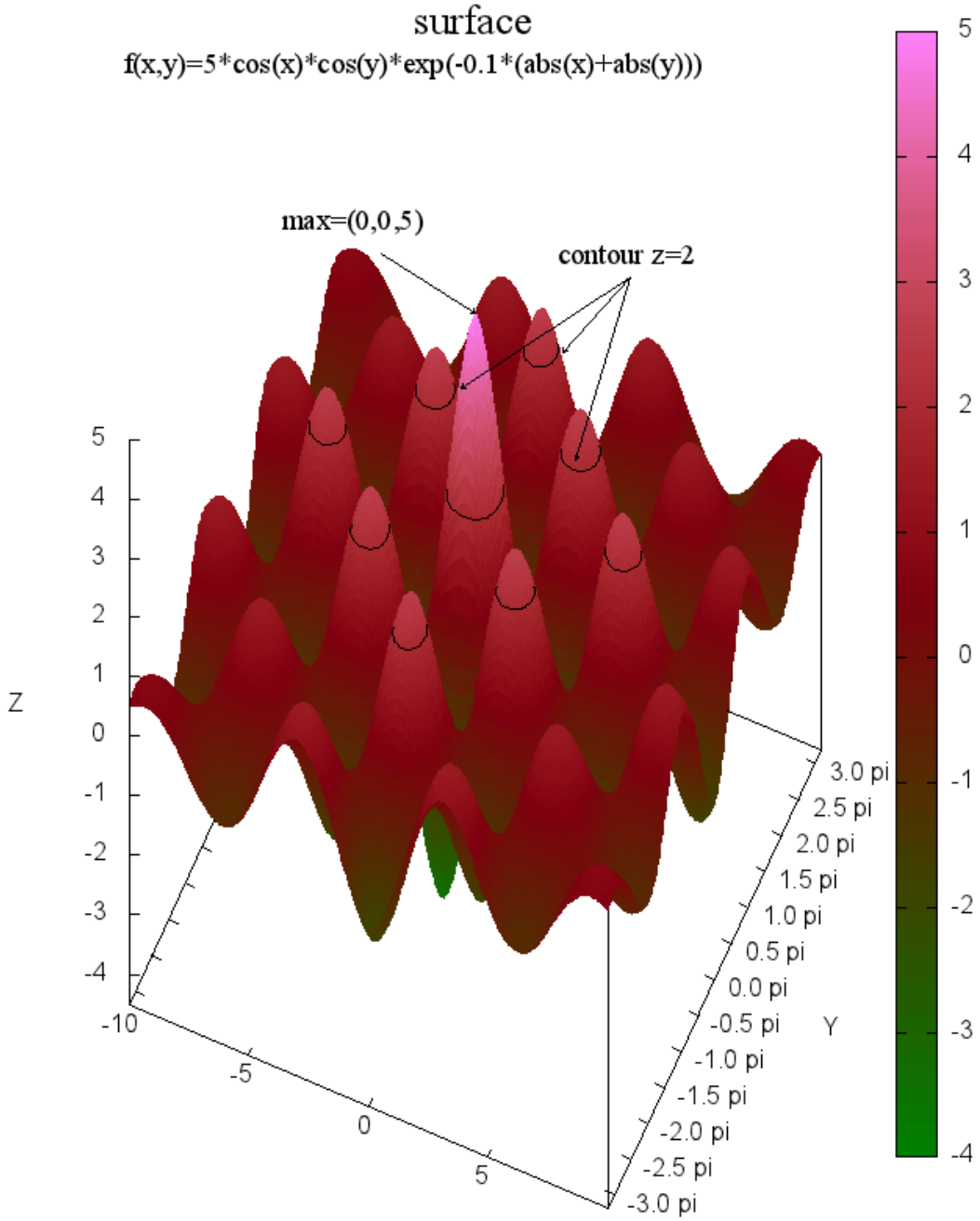


Рис. IV. Изображение поверхности шаблоном pm3d

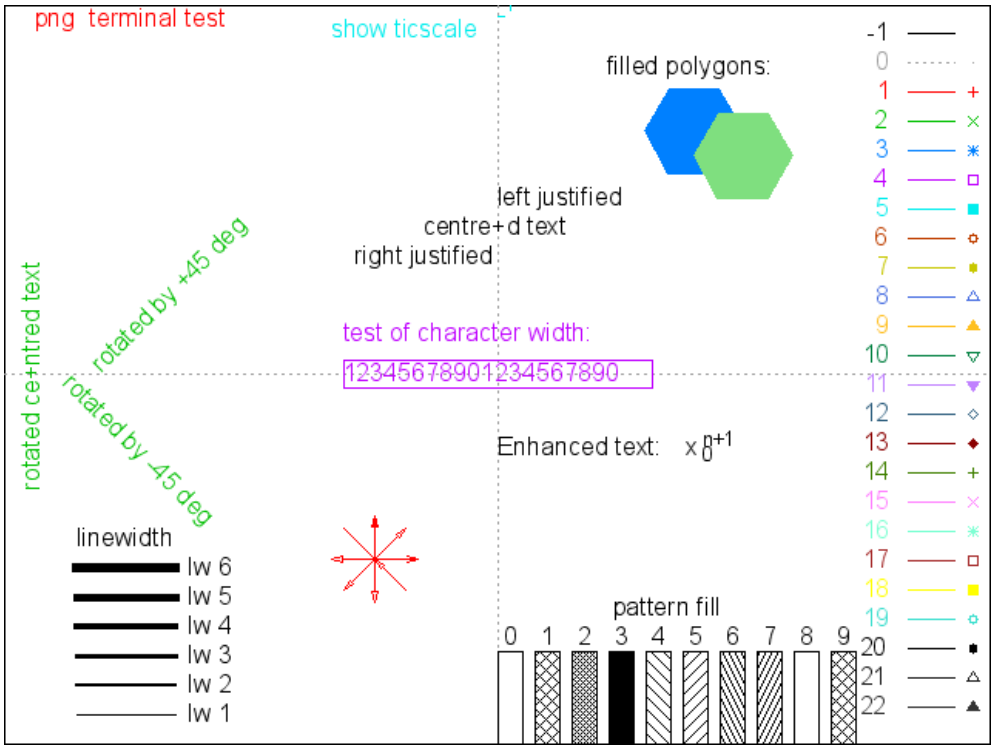


Рис. V. Графический тест

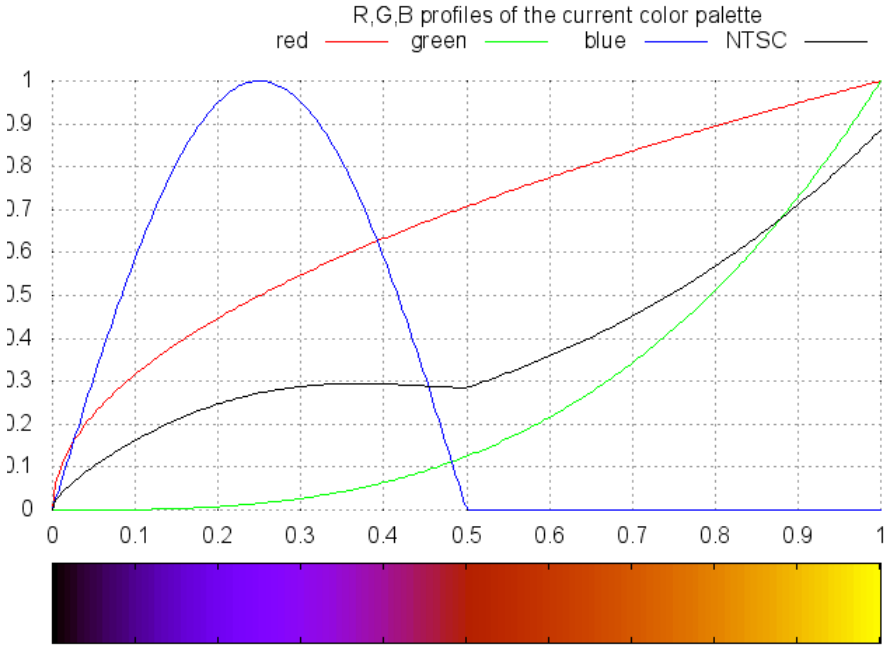


Рис. VI. Тест на цветовые оттенки

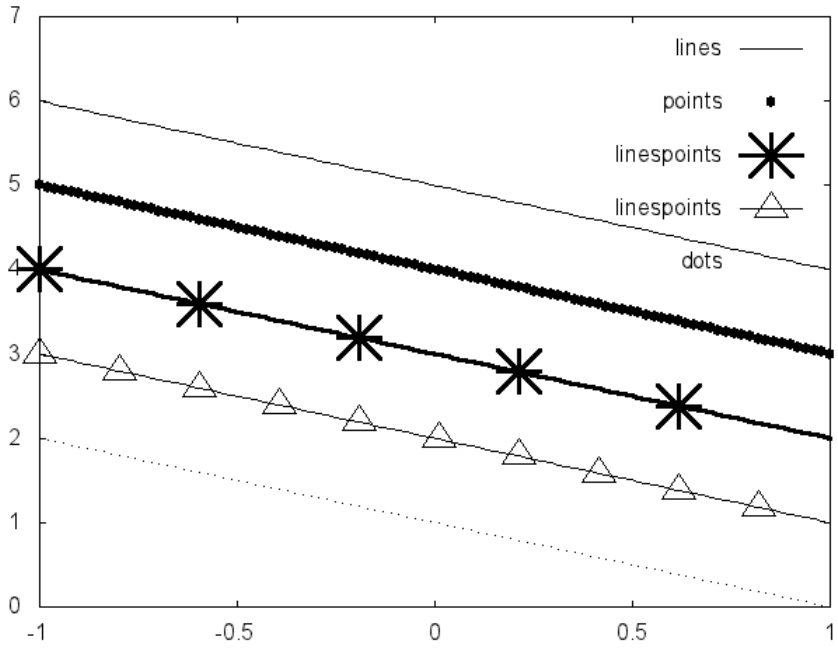


Рис. VII. Пример различных шаблонов графиков

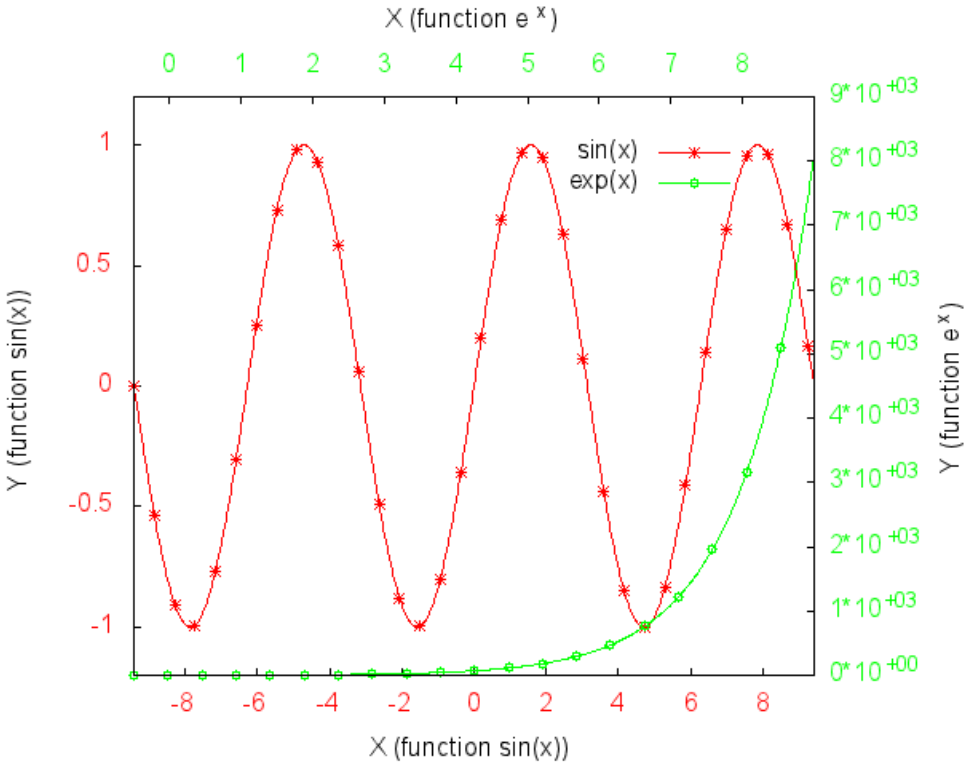


Рис. VIII. Графики двух функций с разными областями определения и значений

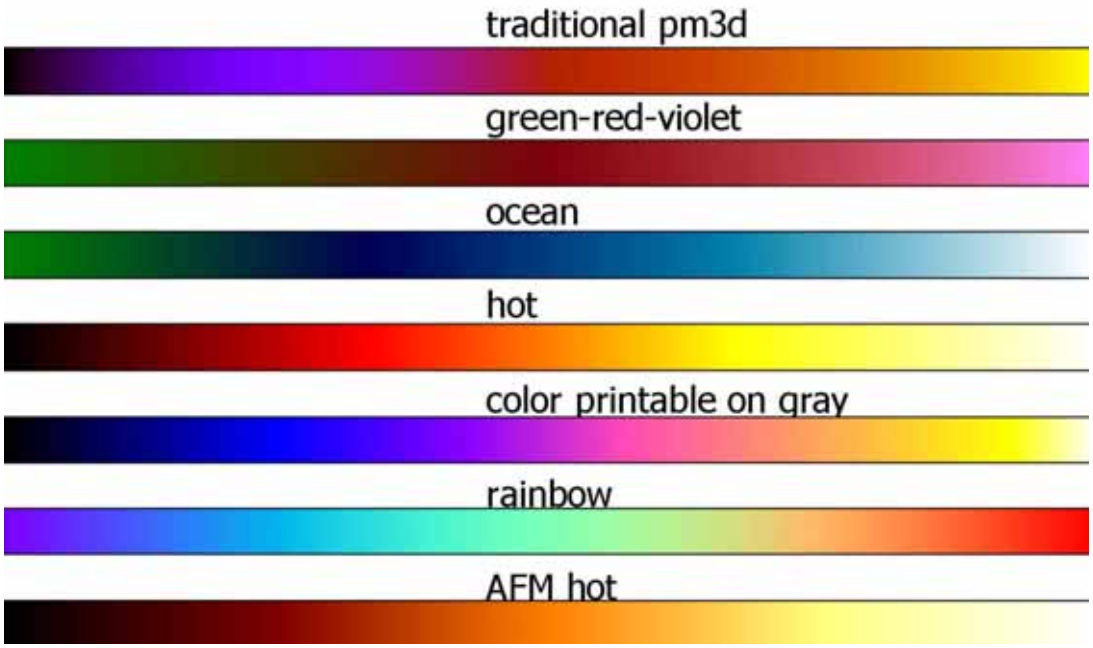


Рис. IX. Примеры палитр **rgbformulae**

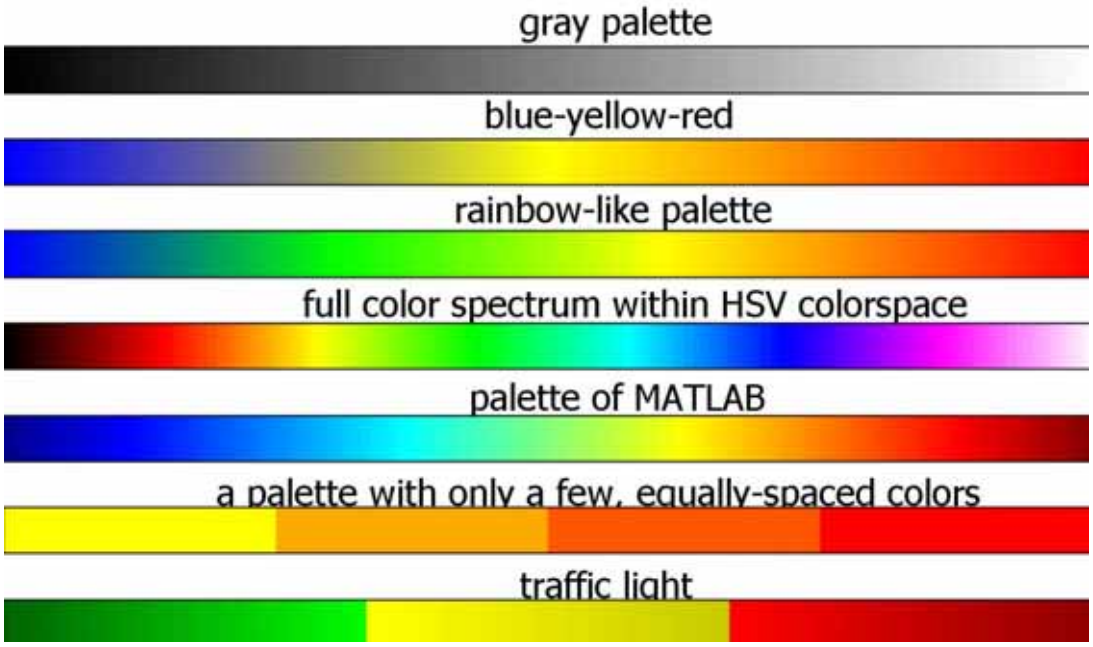


Рис. X. Примеры палитр, заданных ”вручную”

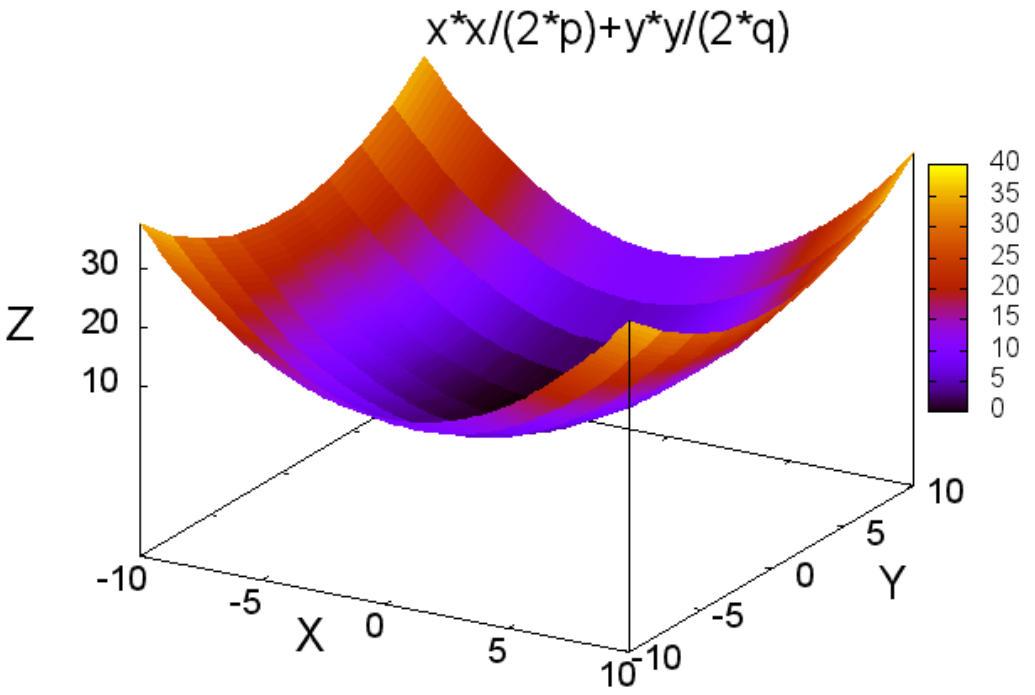


Рис. XI. Параболоид, изображенный шаблоном pm3d

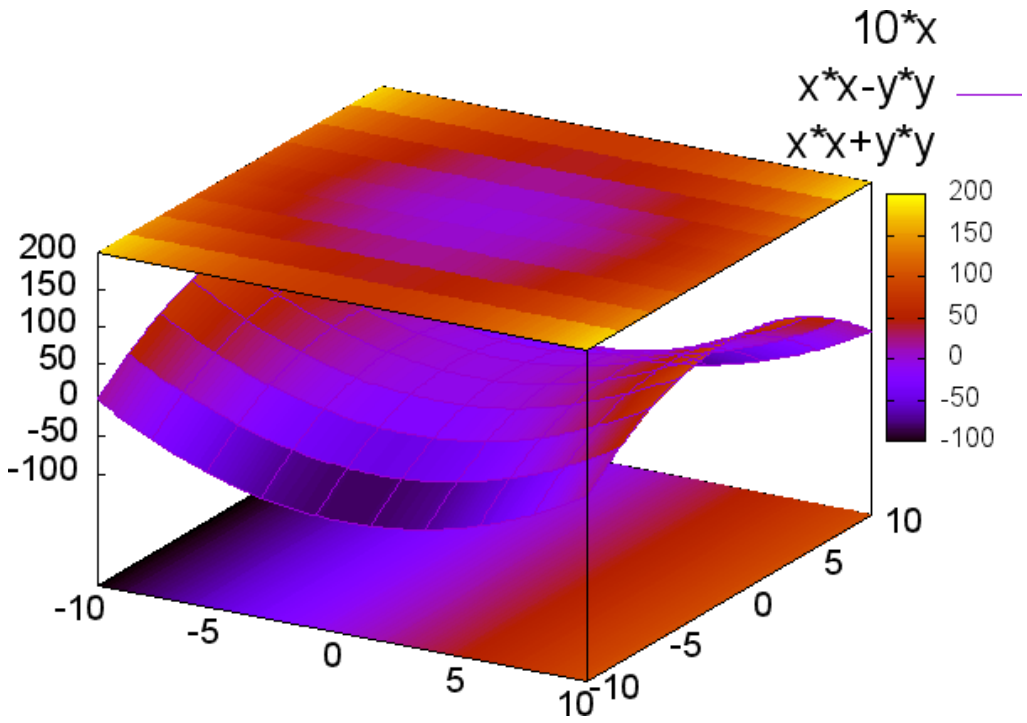


Рис. XII. Изображение данных 3d на разных уровнях

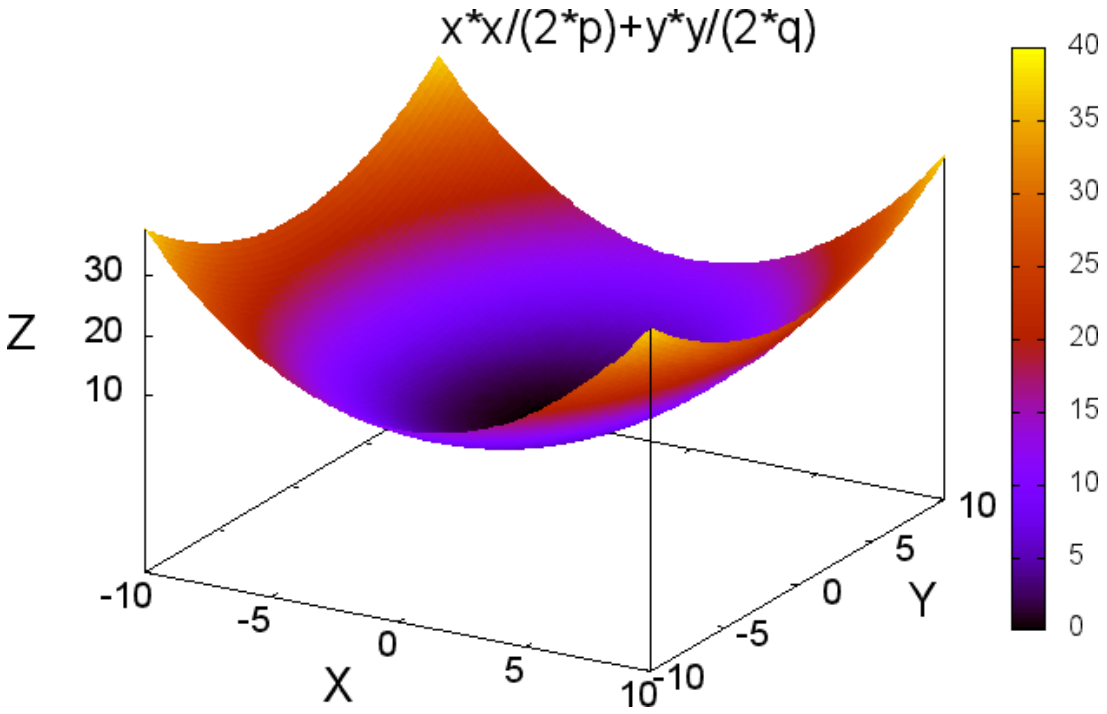


Рис. XIII. параболоид, изображенный шаблоном pm3d с больши́м числом изолиний

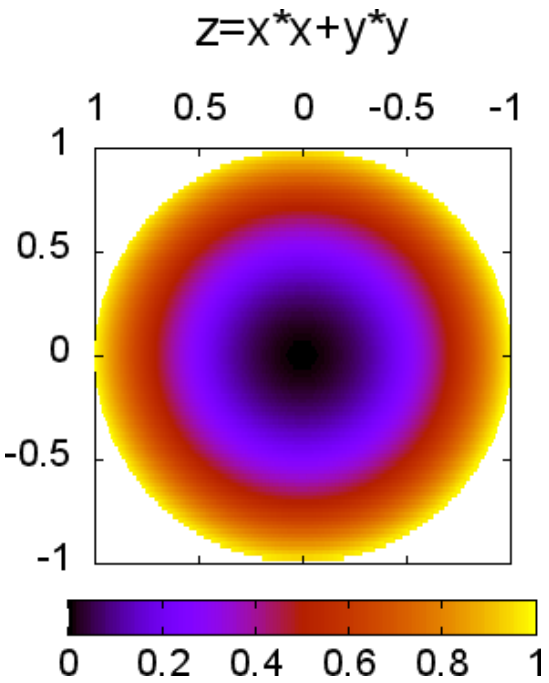


Рис. XIV. Цветная проекция параболоида