

Шнейвайс А.Б.

Азы GNUPLOTa

2016

Шнейвайс А.Б.

В пособии излагаются основные приёмы работы с утилитой `gnuplot`, обеспечивающей возможности графического отображения как результатов компьютерных вычислений, выведенных в файл, так и результатов расчётов, выполненных на языке самой утилиты. Пособие написано с целью упростить процесс её освоения. В пособии рассмотрены далеко не все возможности, предоставляемые ею. Однако, наличие полезной встроенной help-системы, обширная информация, имеющаяся и в Интернете, и в литературе, безусловно облегчат дальнейшее совершенствование студенческих `gnuplot`-скриптов

Рекомендовано учебно-методической комиссией
математико-механического факультета СПбГУ
для студентов, обучающихся по специальности
«АСТРОНОМИЯ».

Содержание

1	Начало освоения GNUPLOTa	3
1.1	График функции, заданной формулой	3
1.1.1	Включение рисунка в TEX-файл	6
1.1.2	Вывод на один чертеж графиков нескольких функций	7
1.1.3	Размещение команд GNUPLOTa в файле	8
1.2	График функции, заданной таблично	9
1.2.1	Уяснение ситуации	9
1.2.2	Первый опыт	9
1.2.3	Вывод графика табулированной функции в eps-файл	12
1.2.4	Графики нескольких табулированных функций. Опция USING.	15
1.3	Построение графика по данным, вводимым с клавиатуры	18
1.4	Модификатор (опция) INDEX	21
1.5	Вызов загрузочного модуля программы из GNUPLOTa	24
1.6	Организация циклоподобных процессов	25
1.6.1	Картинка образцов (стилей вычерчивания) линий и значков.	26
1.6.2	Моделирование движения (фигуры Лиссажу)	28
1.6.3	Изменение количества засечек осей	30
1.7	Пример демонстрации решения задачи двух тел.	32
1.7.1	GNUPLOT-скрипт	33
1.7.2	ФОРТРАН-программа расчета координат планеты	35
1.7.3	GNUPLOT-высветка результата. Опция EVERY	37
2	Представление значений функции.	40
2.1	Опция impulses.	41
2.2	Опция boxes.	42
2.3	Опции steps, fsteps и histeps.	43
3	Обрамляющая рамка вокруг рисунка.	45
4	О некоторых настройках plot.	53
5	Заключение	57

1 Начало освоения GNUPLOTa

gnuplot – утилита, предназначенная для представления результатов расчета в графической форме. Расчет может проводиться и самой утилитой по формулам, и независимо, например, **ФОРТРАН**- или **СИ**-программой. При этом **gnuplot** предоставляет возможность выборки данных и из файлов, и путем перенаправления их из стандартного вывода рабочей программы на стандартный ввод утилиты, и непосредственно, располагая данные, после вызова команды черчения. Утилита работает как в интерактивном режиме, так и под управлением командных файлов (**gnuplot**-скриптов). Описание на русском языке примеров работы с ней и соответствующего инструментария излагается в [1] (см. главу 14, пункты 4 – 12). Подробное руководство по утилите **gnuplot** на английском языке в **linux**-системах имеется в каталоге

`/usr/share/doc/gnuplot-doc/`

(см. файлы **gnuplot.html** и **tutorial.dvi**). Первый содержит полное описание всех команд утилиты и их опций, а также интернет-адреса, по которым при желании можно попытаться получить ответы на возникшие вопросы. Второй представляет краткое руководство по ориентации утилиты на выдачу результата в виде пригодном для использования в среде издательской системы **LaTeX**. Упомянутый каталог содержит также подкаталоги **demo** и **psdoc**. Из первого можно извлечь массу полезной информации, анализируя тексты демонстрационных файлов (около тридцати). Во втором имеется **gnu**-скрипт, демонстрирующий наборы образцов (стилей, шаблонов) линий и значков, используемых при создании рисунков.

Важное достоинство утилиты: gnuplot – свободно распространяемый программный продукт.

1.1 График функции, заданной формулой

1. В командной строке пишем команду вызова: **gnuplot** и нажимаем клавишу **enter**. В результате на экран высвечивается информация:

```
G N U P L O T
Version 3.7 patchlevel 3
last modified Thu Dec 12 13:00:00 GMT 2002
System: Linux 2.4.22-1.2115.nptl
Copyright(C) 1986 - 1993, 1998 - 2002
Thomas Williams, Colin Kelley and many others
Type 'help' to access the on-line reference manual
The gnuplot FAQ is available from
http://www.gnuplot.info/gnuplot-faq.html
Send comments and requests for help to <info-gnuplot@dartmouth.edu>
Send bugs, suggestions and mods to <bug-gnuplot@dartmouth.edu>
Terminal type set to 'x11'
gnuplot>
```

Помимо остальной справочной информации общего характера высвечена, в частности, строка **Terminal type set to 'x11'**, которая указывает, что в качестве **типа терминала** (устройства, через которое будет подан график) по умолчанию выбрано устройство вывода на графический экран. Вообще, тип терминала может оказаться и иным (например, 'qt'; возможны и другие).

2. Команда **plot cos(2*x)*exp(-x/7)** высветит график требуемой функции.
3. При необходимости отпечатать график его можно вывести в терминах языка **Postscript** в какой-нибудь файл. Переключение типа терминала с **x11** на **Postscript** обеспечивается командой **set terminal postscript**, а перенаправление вывода в файл с желаемым именем – командой **set output mycos.ps**. Обработка двух последних команд и повтор команды **plot cos(2*x)*exp(-x/7)** приведет к появлению в текущей директории файла с именем **mycos.ps**. Его обзор и печать реализуется одной из утилит: **gv**, **ghostview** или **GSView** (отпечатать файл можно и просто командой **lpr**, например, так: **lpr mycos.ps**). Пример вывода построенного графика дан на следующей странице.
4. Диапазоны изменения аргумента и функции выбраны утилитой из установок умолчания. Переназначить их при выводе одного конкретного рисунка можно непосредственно в команде чечения **plot**. Например,

```
plot [-3.5:4] [-2:2] cos(2*x)*exp(-x/7)}.
```

Для изменения рабочего диапазона только оси ординат при сохранении установки умолчания относительно оси абсцисс достаточно диапазон последней указать первыми квадратными скобками без содержимого, то есть

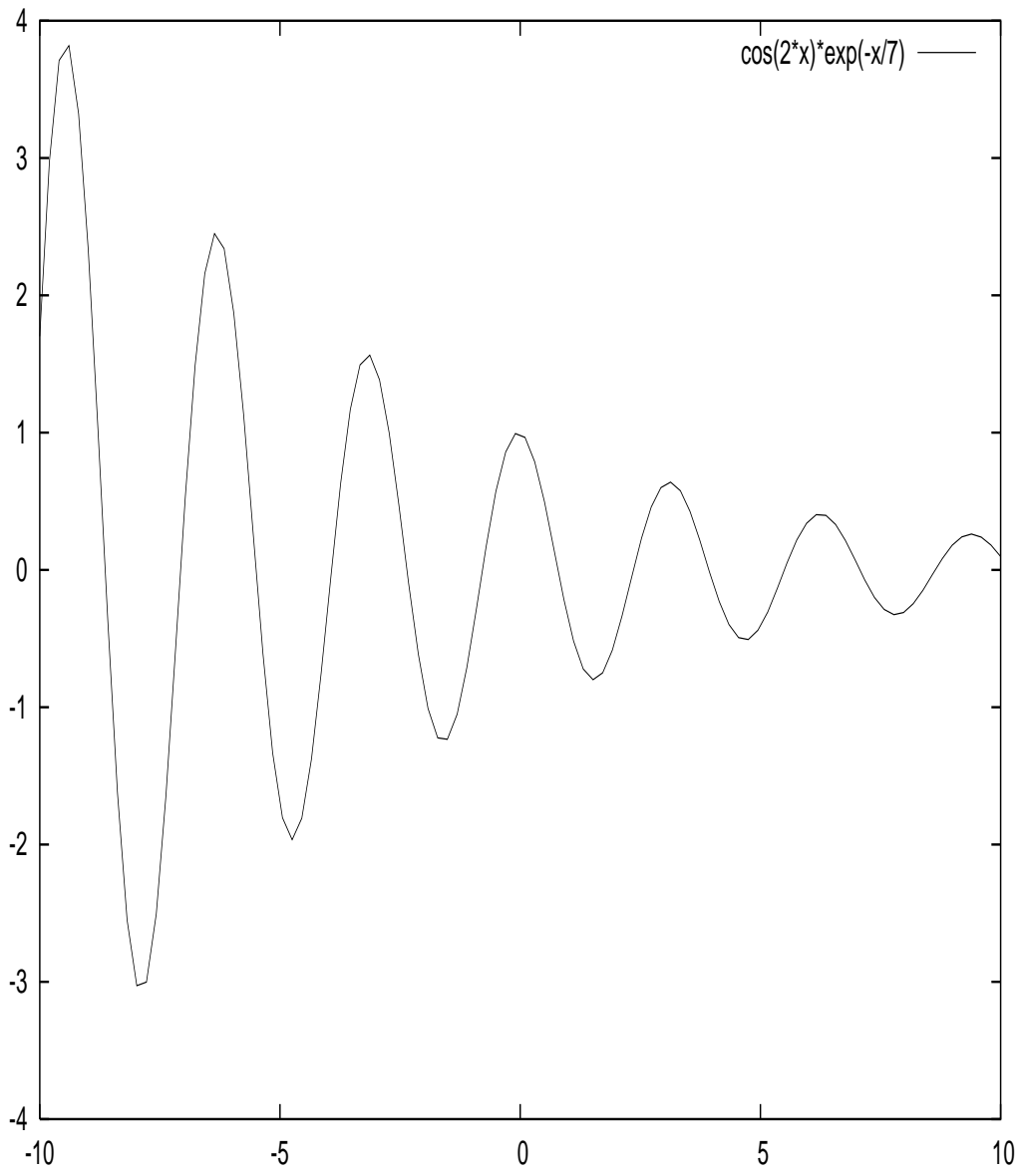
```
plot [] [-2:2] cos(2*x)*exp(-x/7)}.
```

5. Переназначение установок умолчания диапазонов при многих вызовах команды **plot** (если нужно, конечно; все решает человек) выгоднее провести один раз посредством команды **set xrange** и/или команды **set yrange**. Например,

```
set xrange [-3.5:4]; set yrange [-2:2]; plot cos(2*x)*exp(-x/7)}
```

Команда **set** предназначена для самых разнообразных установок (вспомним **set terminal** или **set output**). Слово после **set** – просто один из ее модификаторов.

6. Напомнить действующую установку по оси абсцисс можно командой **show xrange**, а узнать все действующие установки – командой **show all**.
7. При желании восстановить стандартные установки умолчания по всем используемым в **gnuplot** модификаторам команды **set** достаточно дать команду **reset**.
8. Выход из среды **gnuplot** возможен по любой из двух команд **quit** или **exit**. Заметим, что при употреблении первой для выхода достаточен набор лишь одной первой буквы **q**.



1.1.1 Включение рисунка в TEX-файл

Обычно рисунки помещаются в текст отчета, который часто составляется в среде издательской системы **LATEX**. Включаемые рисунки рекомендуется подавать в формате **eps** (Encapsulated postscript). Перевод файла с расширением **ps** в формат **eps** осуществим разными способами. Упомянем утилиты **ps2epsi**, **convert**:

```
ps2epsi myfile1.ps
convert myfile2.ps myfile2.eps
```

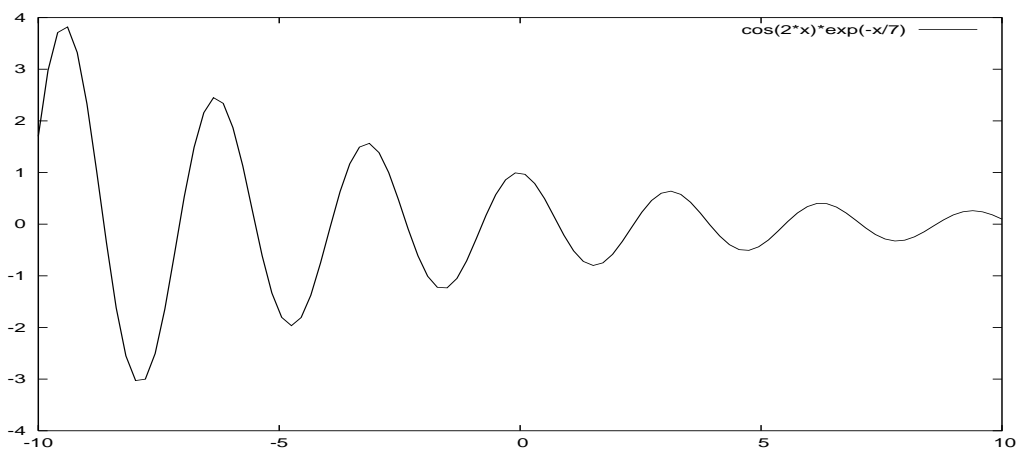
Графический редактор **gimp**, близкий по инструментарию к **photoshop**, позволяет читать графические файлы в самых разных форматах и преобразовывать их в случае необходимости к любому из имеющихся. Однако, работая в среде утилиты **gnuplot**, проще всего уточнить команду **set terminal postscript eps** опцией **eps**, то есть

```
gnuplot> set terminal postscript eps
gnuplot> set output 'mycos.eps'
gnuplot> plot cos(2*x)*exp(-x/7)
```

Кстати, и **lpr**, и **gv** понимают расширение **eps**, позволяя печатать рисунки из **eps**-файлов. При желании включить рисунок в **TEX**-файл достаточно поместить команду обращения к файлу с рисунком в нужную строку. Например,

```
\begin{figure}[ht]
\includegraphics [height=7cm,width=15cm,clip,
                  angle=0]{./figs/mycos.eps}
\end{figure}
```

В частности, видно, что команда **includegraphics** обеспечивает и изменение масштабов рисунка, и возможность поворота его на нужный угол.



1.1.2 Вывод на один чертеж графиков нескольких функций

Вывод на один чертеж графиков нескольких функций достигается расположением в строке команды **plot** соответствующих формул через запятую. Например, если после входа в **gnuplot** дать команду:

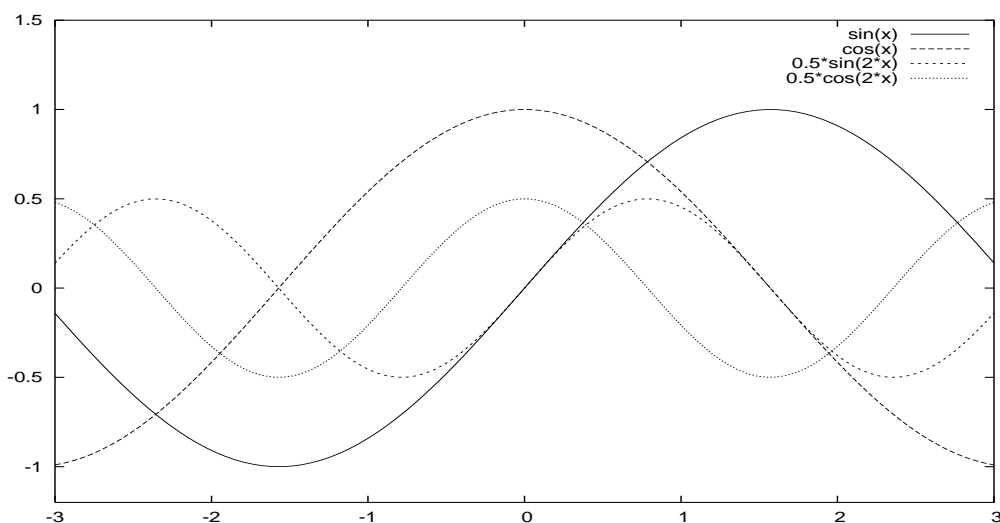
```
plot [-3:3] [-1.2:1.5] sin(x), cos(x), 0.5*sin(2*x), 0.5*cos(2*x)
```

то на экране высветятся требуемые графики. Для получения в текущей директории файла с именем **sincos4.eps**, который содержит графики тех же функций, достаточно дать команды:

```
set terminal postscript eps
set output 'sincos4.eps'
plot [-3:3] [-1.2:1.5] sin(x), cos(x), 0.5*sin(2*x), 0.5*cos(2*x)
```

Включение рисунка из файла **sincos4.eps** в TEX-файл реализуется в последнем командами

```
\begin{figure}[ht]
\includegraphics[height=7cm,width=14cm,clip,angle=0]{sincos4.eps}
\end{figure}
```



Для возвращения **gnuplot** в режиме выдачи графиков на экран можно дать команды **set terminal** и **set output** или команду **reset** (если желаем восстановить все установки стандартного режима умолчания).

1.1.3 Размещение команд GNUPLOТа в файле

Ясно, что каждый раз перебирать вручную тройку команд

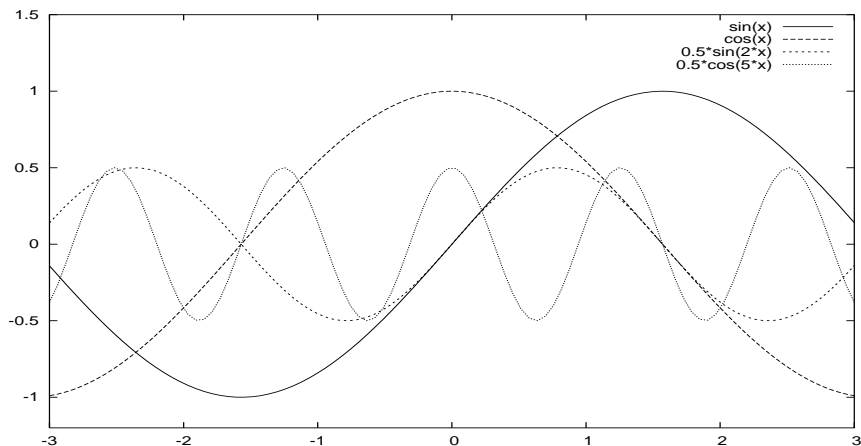
1. `set terminal postscript eps`
2. `set output имя файла`
3. `plot [-3:3] [-1.2:1.5] sin(x), cos(x), 0.5*sin(2*x), 0.5*cos(2*x)`

неудобно (можно случайно ошибиться, особенно, если команд много). Нужную последовательность команд **gnuplot**a можно разместить в текстовом файле с любым именем. Тогда при вызове команды **gnuplot** с указанием в качестве аргумента имени соответствующего файла будет выполнена вся записанная в нем последовательность **gnu**-команд. Например, сформируем файл с именем **sincos5.gnu** и запишем в него упомянутые команды. Расширение **.gnu** – чисто психологическая установка, напоминающая человеку о возможном наличии в файле команд **gnuplot**a.

```
# имя файла: sincos5.gnu
set terminal postscript eps
set output 'sincos5.eps'
plot [-3:3] [-1.2:1.5] sin(x), cos(x), 0.5*sin(2*x), 0.5*cos(5*x)
```

В тексте **gnu**-скрипта значок **#**, как и в **BASHe**, **PERLe**, или **RUBY**, – признак начала **оператора комментария**. Все, что следует за знаком диеза, до конца строки интерпретатором будет проигнорировано. Если теперь выполнить команду **gnuplot sincos5.gnu**, то в текущей директории появится файл с именем **sincos5.eps**, который можно обозревать через **gv sincos5.eps** или внести в качестве очередного рисунка в **TEX**-файл посредством команд

```
\begin{figure}[ht]
\includegraphics [height=6cm,width=14cm,clip,angle=0]{sincos5.eps}
\end{figure}
```



1.2 График функции, заданной таблично

1.2.1 Уяснение ситуации

Результаты счета какой-нибудь исследовательской **ФОРТРАН**- или **СИ**- программы часто выводятся не только (и не столько) на экран, но и в файлы на диске. Всегда выгодно сохранить результаты счета для их последующего внимательного анализа или ввода в качестве исходных данных другой программы. Примером такой другой программы может служить утилита **gnuplot**, если есть желание увидеть результаты в виде графиков. Не умаляя изобразительной мощи графических пакетов, встроенных в языки программирования, заметим, что исследователю абсолютно не интересно вникать в тонкости использования соответствующих графических процедур и функций (исключая, конечно, ситуацию, когда их освоение поставлено во главу угла). Обычно, после первых экспериментов с выводом графиков на экран хочется начать усовершенствование вывода (например, обеспечить изменение масштаба по осям или диапазонов изменения абсциссы и ординаты и т. д.). При этом быстро обнаруживается, что время, затрачиваемое на осмысливание, внесение изменений, их отладку и тестирование, оказывается больше времени, которое понадобилось для программирования проблемной содержательной части задачи. Ясно, что, если человек постоянно не работает с графикой на **ФОРТРАНе**, то вряд ли он сможет за обозримое время создать нечто, сравнимое по возможностям с утилитами **gnuplot**, **pgplot**, **plplot**, **grapher** или другими, которые создавались профессионалами в области компьютерного дизайна и графики. Упомянутые утилиты широко распространены, просты в освоении и обычно используются исследователями, пишущими свои программы на самых разных языках программирования. Выгода от умения применять **gnuplot** для графической интерпретации получаемых результатов несомненна.

1.2.2 Первый опыт

Пусть, например, в результате ручного набора или расчета в текущей директории появился файл с именем **gnuexam1.dat**, содержащий матрицу из двух столбцов: первый столбец – аргумент; второй – функция. Приведем текст простой программы на **ФОРТРАНе**, моделирующей ситуацию. Программа вычисляет значение функции $f(x) = x^3$ на промежутке изменения $x \in [-1, 1]$ с шагом по аргументу **0.1**

```
program gnuexam1
implicit none
integer i
real x, y
open (10,file='gnuexam1.dat')
do i=-10,10
  x=0.1*i
  y=x*x*x
  write(10,'(e15.7,e15.7)') x,y
enddo
end
```

Здесь оператор `open(unit=10,file='gnuexam1.dat')` требует, чтобы программа в текущей директории открыла файл с именем `gnuexam1.dat` и сопоставила ему логическое программное устройство вывода под номером `10`, на которое по мере работы будет выводить результат.

```
$ g77 gnuexam1.for           / получили загрузочный файл
$ ./a.out                   / инициировали его выполнение
$ cat gnuexam1.dat          / высветили на экран содержимое файла
-0.1000000E+01 -0.1000000E+01
-0.9000000E+00 -0.7290001E+00
-0.8000000E+00 -0.5120000E+00
-0.7000000E+00 -0.3430000E+00
-0.6000000E+00 -0.2160000E+00
-0.5000000E+00 -0.1250000E+00
-0.4000000E+00 -0.6400000E-01
-0.3000000E+00 -0.2700000E-01
-0.2000000E+00 -0.8000000E-02
-0.1000000E+00 -0.1000000E-02
 0.0000000E+00  0.0000000E+00
 0.1000000E+00  0.1000000E-02
 0.2000000E+00  0.8000000E-02
 0.3000000E+00  0.2700000E-01
 0.4000000E+00  0.6400000E-01
 0.5000000E+00  0.1250000E+00
 0.6000000E+00  0.2160000E+00
 0.7000000E+00  0.3430000E+00
 0.8000000E+00  0.5120000E+00
 0.9000000E+00  0.7290001E+00
 0.1000000E+01  0.1000000E+01
```

Теперь вызовем `gnuplot` и обратимся к команде `plot 'gnuexam1.dat'`. На экране появится график табулированной в файле функции. Правда, дан он будет в дискретной форме точками (точнее значками определенного по умолчанию вида: красные ромбики с точкой посерединке), которые фиксируют значения ординат. Сформируем командный `gnu`-файл, реализующий при запуске `gnuplot gnuexam1.gnu` выветку графика.

```
set terminal x11      # установка типа терминала;
set output           # установка вывода на экран;
plot 'gnuexam1.dat' # вычерчивание содержимого файла с данными;
```

При желании видеть непрерывную кривую достаточно дополнить команду `plot` опцией `with`, задающей стиль вычерчивания (форму представления изображения графика). Вот некоторые примеры (поэкспериментируйте с ними) в среде `gnuplot`):

Команда	График на экране
<code>plot 'gnuexam1.dat' with lines</code>	красная линия
<code>plot 'gnuexam1.dat' with l</code>	то же самое
<code>plot 'gnuexam1.dat' w l</code>	то же самое
<code>plot 'gnuexam1.dat' w l 1</code>	то же самое
<code>plot 'gnuexam1.dat' with lines 1</code>	то же самое
<code>plot 'gnuexam1.dat' w l 1</code>	то же самое
<code>plot 'gnuexam1.dat' w l lt 1</code>	то же самое
<code>plot 'gnuexam1.dat' w l lt 1 lw 1</code>	то же самое
<code>plot 'gnuexam1.dat' w l lt 1 lw 5</code>	линия впятеро толще
<code>plot 'gnuexam1.dat' w points pt 5 ps 1</code>	красные треугольники (размер по умолчанию)
<code>plot 'gnuexam1.dat' w points pt 1 ps 3</code>	красные ромбики
<code>plot 'gnuexam1.dat' w p pt 1 ps 3</code>	то же самое
<code>plot 'gnuexam1.dat' with p pt 1 ps 3</code>	то же самое
<code>plot 'gnuexam1.dat' with linespoint pt 1 ps 3</code>	то же самое, но вместе с линией
<code>plot 'gnuexam1.dat' with lp lt 2 lw 3 pt 1 ps 3</code>	то же, но изменили цвет и толщину линии

Формально после ключевого слова **with** пишется одно из имен, задающих образец (стиль черчения, шаблон) линии, выбранной для графика. Запись имени образца возможна и в сокращенной форме (например, достаточно одной буквы **w** вместо **with** или **l** вместо **lines**). Имена стилей, предоставляемых утилитой **gnuplot**, высвечиваются в ее среде командами **set data style** или **set function style**. Например,

1. **lines** означает линию, стиль и толщина которой задаются их номерами после опций **lt** (**linetype**) и **lw** (**linewidth**) соответственно.
2. **points** – дискретный символ (значок), вид и размер которого задаются аналогично после опций **pt** (**pointtype**) и **ps** (**pointsize**).
3. **linespoints** – линия с распределенными по ней значками; возможно совместное применение опций **lt, lw, pt** и **ps**.
4. **dots** – просто точки;
5. **impulses** – отрезки ординат (всплесков) от оси абсцисс (**y=0**);

Возможны и другие стили изображения графиков. Соответствие стиля линии и значка их номерам можно высветить в среде **gnuplot** командой **test** (попробуйте).

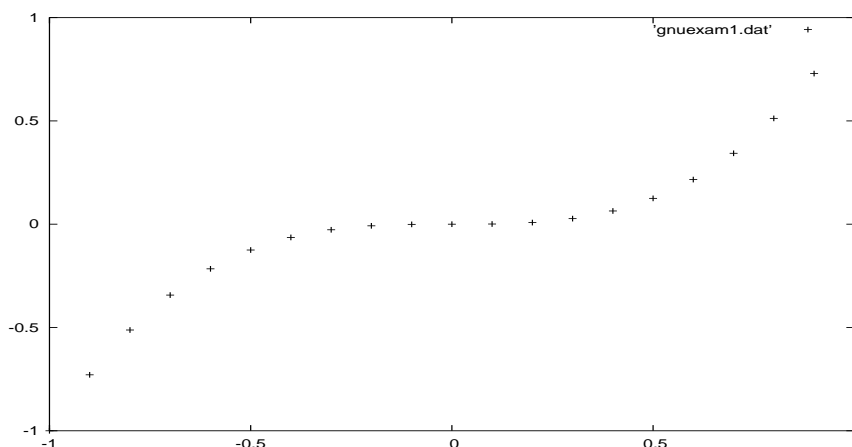
1.2.3 Вывод графика табулированной функции в eps-файл

Сформируем файл `gnuexam1.gnu`

```
set terminal postscript eps      # установка типа терминала;
set output 'gnuexam1.eps'      # установка вывода в eps-файл;
plot 'gnuexam1.dat'           # вычерчивание содержимого файла с данными;
```

Вызов `gnuplot gnuexam1.gnu` запишет график в файл `gnuexam1.eps` с тем, чтобы изображение можно было не только высветить и отпечатать посредством `gv gnuexam1.eps`, но и вставить в **TEX**-отчет, используя **TEX**-команды, например:

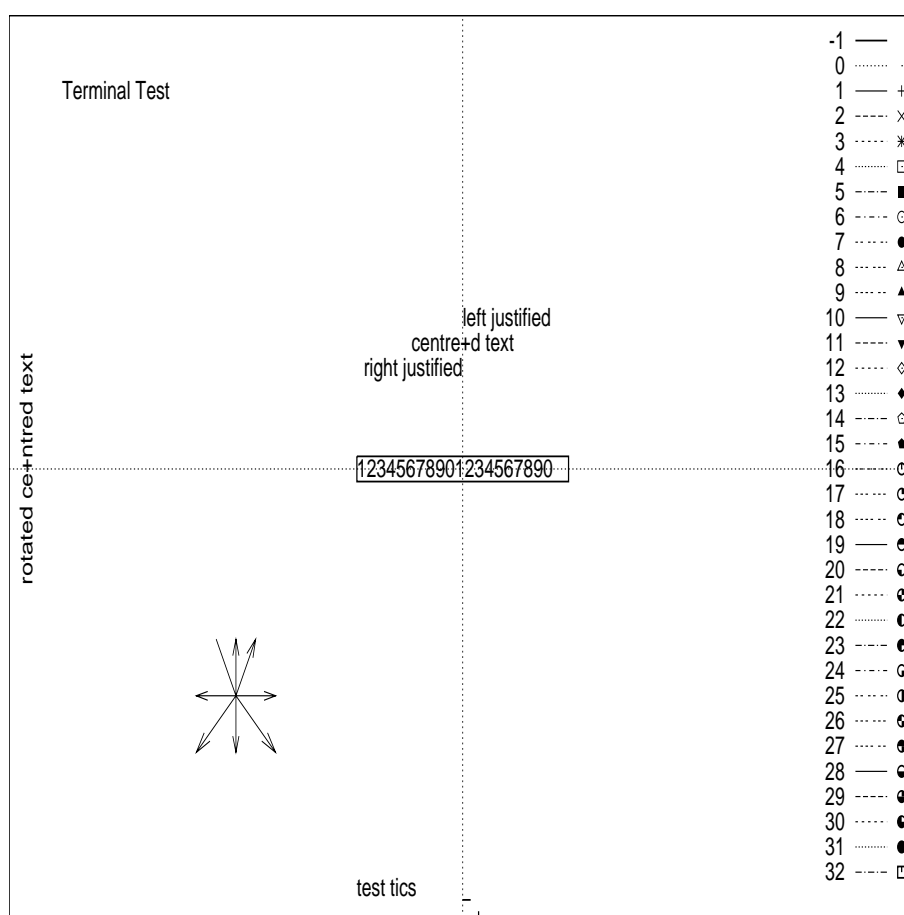
```
\begin{figure}[ht]
\includegraphics[height=6cm,width=12cm,clip,angle=0]{gnuexam1.eps}
\end{figure}
```



Обратим внимание, что в **eps**-формате значки, отображающие точки графика изменились по сравнению со значками, высвечиваемыми через терминал **x11**: красный ромбик превратился в серый знак плюс. Дело в том, что по умолчанию опция **eps** из команды `set terminal postscript eps` предполагает вычерчивание графика не в цветном режиме. Узнать о представлении линий и значков в режиме умолчания `postscript eps`, как и раньше, можно посредством команды `test`, предварительно позаботившись о перенаправлении вывода в файл (предположим, с именем `gnutest.eps`). Для достижения этой цели достаточно, войдя в среду `gnuplot`, дать команды `set terminal postscript eps`, `set output 'gnutest.eps'` и `test`; также полезно осмыслить сообщение, выводимое в процессе отработки первой команды:

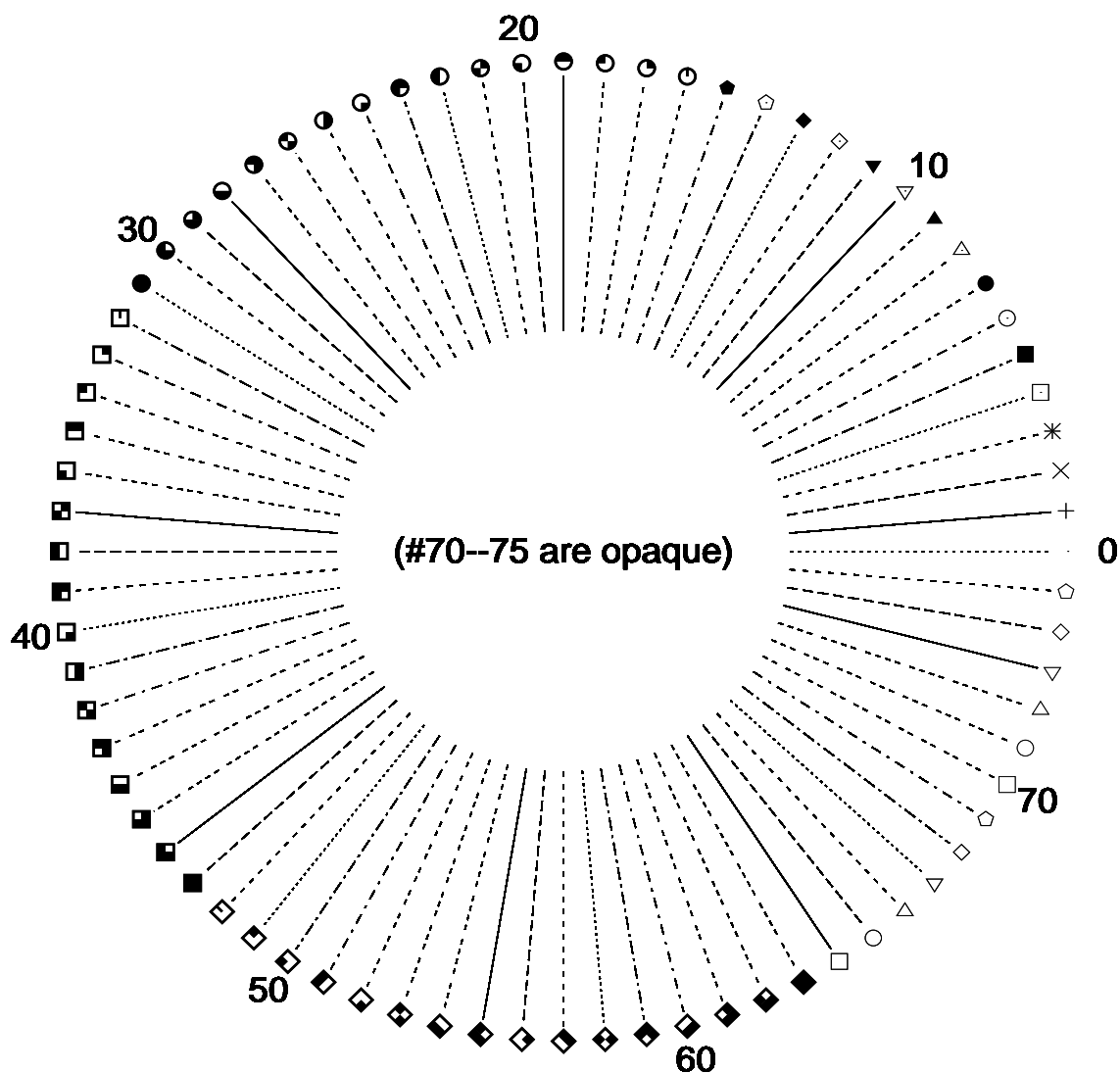
```
gnuplot> set terminal postscript eps
Terminal type set to 'postscript'
Options are 'eps noenhanced monochrome dashed defaultplex "Helvetica" 14'
gnuplot> set output 'gnutest.eps'
gnuplot> test
gnuplot>
```

Видны ключевые слова установленных опций: **noenhanced** – нельзя использовать верхние или нижние индексы; **monochrome** – чернобелый режим; **dashed** – допустимо пунктирное изображение линий; **defaultplex** – настройка печати по умолчанию (односторонний или двусторонний); **“Helvetica” 14** – название используемого шрифта и его размер. Нумерацию образцов линий и значков, соответствующих данной установке, можно видеть на рисунке, выводимом в файл командой `set output 'gnutest.eps'`:



Список установленных опций можно при желании переустановить, дополнив первую команду `set terminal postscript eps` соответствующими ключевыми словами. Например, `set terminal postscript eps “Helvetica” 7`, т.е. вдвое уменьшили размер шрифта **Helvetica**. Тогда рисунок, полученный командой `test`, будет содержать почти вдвое больше образцов, хотя восприятие его ввиду их малости будет менее удобным.

Наглядное изображение шаблонов значков и стилей линий запрограммировано на языке операторов утилиты **gnuplot** в файле **ps_symbols.gpi** (см., например, `/usr/share/doc/gnuplot-3.7.3/psdoc`):



Значки с номерами **70** – **75** названы *непрозрачными*, хотя единственным темным местом является их контур. Дело в том, что в чернобелом режиме белый цвет воспринимается как цвет фона, на котором рисуется контур значка. Если же выбрать цветной режим, то изображения упомянутых значков не изменятся, а темные значки при выборе соответствующего цвета полностью окажутся цветными.

1.2.4 Графики нескольких табулированных функций. Опция USING.

Часто функции, табулированные для одного набора узлов аргумента, выводятся в файл на диске в виде матрицы, первый столбец которой хранит значения аргумента, а каждый следующий – массив значений соответствующей функции. Ниже приведена **ФОРТРАН**-программа, которая выводит матрицу в файл **testx3p.res**:

```
implicit none
real x, y, p(-4:4) /-0.9,-0.5,-0.3,-0.1, 0.0, 0.1, 0.3, 0.5, 0.9/
integer i, j, nres / 10 /
open (unit=nres,file='testx3p.res')
write(nres,1000) (p(j),j=-4,4)
do i=0,20
  x=-1+i*0.1; write(nres,1001) x,((x+p(j))**3,j=-4,4)
enddo
1000 format(1x,'# p ', 9f7.2,/1x,'# x ')
1001 format(10f7.2)
end
```

Его содержимое – семейство графиков функции $(x + p)^3$ при $x \in [-1, 1]$ и $p = -0, 9; -0, 5; -0, 3; -0, 1; 0, 0; 0, 1; 0, 3; 0, 5; 0, 9$:

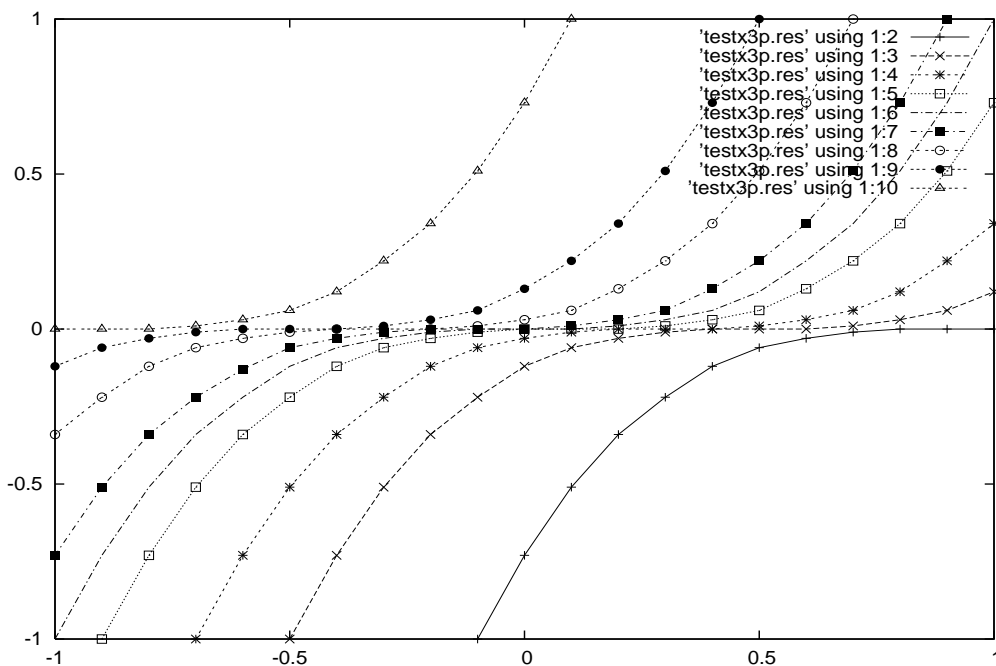
# p	-0.90	-0.50	-0.30	-0.10	0.00	0.10	0.30	0.50	0.90
# x									
-1.00	-6.86	-3.38	-2.20	-1.33	-1.00	-0.73	-0.34	-0.12	0.00
-0.90	-5.83	-2.74	-1.73	-1.00	-0.73	-0.51	-0.22	-0.06	0.00
-0.80	-4.91	-2.20	-1.33	-0.73	-0.51	-0.34	-0.12	-0.03	0.00
-0.70	-4.10	-1.73	-1.00	-0.51	-0.34	-0.22	-0.06	-0.01	0.01
-0.60	-3.38	-1.33	-0.73	-0.34	-0.22	-0.13	-0.03	0.00	0.03
-0.50	-2.74	-1.00	-0.51	-0.22	-0.12	-0.06	-0.01	0.00	0.06
-0.40	-2.20	-0.73	-0.34	-0.12	-0.06	-0.03	0.00	0.00	0.12
-0.30	-1.73	-0.51	-0.22	-0.06	-0.03	-0.01	0.00	0.01	0.22
-0.20	-1.33	-0.34	-0.12	-0.03	-0.01	0.00	0.00	0.03	0.34
-0.10	-1.00	-0.22	-0.06	-0.01	0.00	0.00	0.01	0.06	0.51
0.00	-0.73	-0.12	-0.03	0.00	0.00	0.00	0.03	0.13	0.73
0.10	-0.51	-0.06	-0.01	0.00	0.00	0.01	0.06	0.22	1.00
0.20	-0.34	-0.03	0.00	0.00	0.01	0.03	0.13	0.34	1.33
0.30	-0.22	-0.01	0.00	0.01	0.03	0.06	0.22	0.51	1.73
0.40	-0.12	0.00	0.00	0.03	0.06	0.13	0.34	0.73	2.20
0.50	-0.06	0.00	0.01	0.06	0.12	0.22	0.51	1.00	2.74
0.60	-0.03	0.00	0.03	0.13	0.22	0.34	0.73	1.33	3.38
0.70	-0.01	0.01	0.06	0.22	0.34	0.51	1.00	1.73	4.10
0.80	0.00	0.03	0.12	0.34	0.51	0.73	1.33	2.20	4.91
0.90	0.00	0.06	0.22	0.51	0.73	1.00	1.73	2.74	5.83
1.00	0.00	0.12	0.34	0.73	1.00	1.33	2.20	3.38	6.86

Здесь первый столбец – набор из равноотстоящих по промежутку $[-1, 1]$ значений аргумента x , а каждый очередной столбец соответствует своему значению параметра p . Простейший **gnu**-скрипт, реализующий вывод всех графиков в файл **testusi0.eps** и соответствующий ему рисунок:


```

set terminal postscript eps enhanced
set output 'testx3p0.eps'
plot [-1:1] [-1:1] 'testx3p.res' using 1:2 w lp ,\
'testx3p.res' using 1:3 w lp , 'testx3p.res' using 1:4 w lp ,\
'testx3p.res' using 1:5 w lp , 'testx3p.res' using 1:6 w lp ,\
'testx3p.res' using 1:7 w lp , 'testx3p.res' using 1:8 w lp ,\
'testx3p.res' using 1:9 w lp , 'testx3p.res' using 1:10 w lp

```



Здесь команда **plot** по сравнению с предыдущими вариантами ее использования содержит опцию **using**, после которой пишется номер столбца аргумента и через двоеточие номер столбца, соответствующего текущему значению параметра **p**. Отсутствие опции **using** по умолчанию эквивалентно **using 1:2**, а при наличии в исходном файле лишь одного столбца в качестве аргумента берется номер точки.

Замечания:

1. Заголовок таблицы, присутствующий в файле **testx3p.res**, **НЕ МЕШАЕТ** утилите правильно отображать кривые. Содержимое заголовка игнорируется утилитой, так как **ФОРТРАН**-программа каждую его строку начинает со знака диеза (**#**), который в **gnuplot** служит признаком начала комментария.
2. В правом верхнем углу рисунка размещена **легенда**. Налицо ее неудачное размещение на изображениях кривых (да и имя каждой кривой чересчур длинно). Указанные недостатки легко исправляются явным указанием места расположения **легенды** и имени очередной кривой.

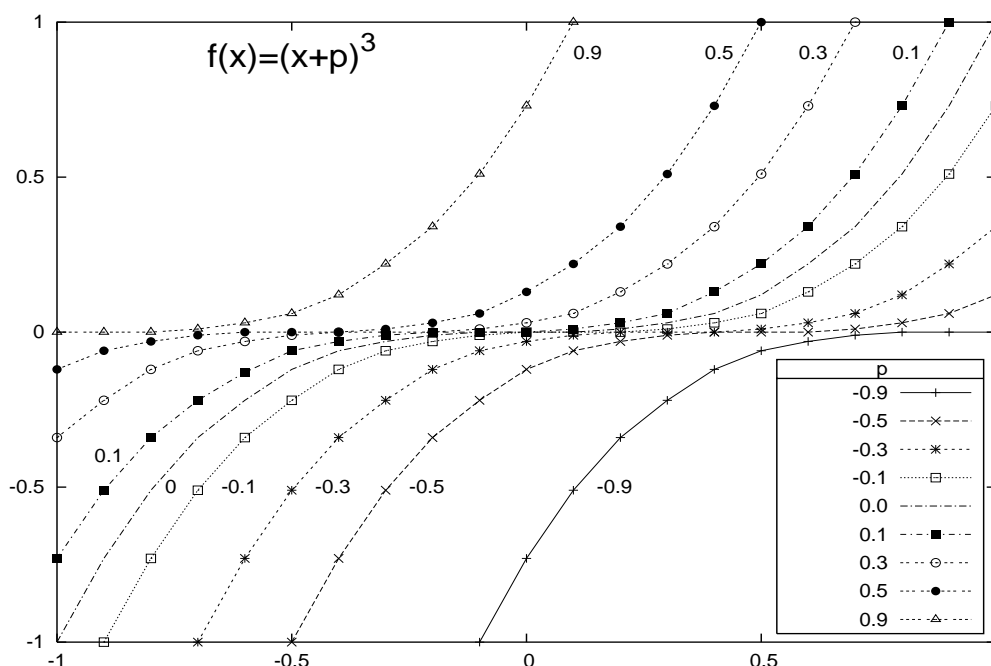
```

set terminal postscript eps enhanced # установка типа терминала и
set output 'testx3p1.eps'           # --"--"-- имени .eps-файла;
set key right bottom samplen 5 spacing 1.5 width 5 title "p" box
set label "f(x)=(x+p)^3" at -0.5,0.9 center font "Helvetica,24"
set label "-0.9" at 0.15,-0.5; set label "-0.5" at -0.25,-0.5
set label "-0.3" at -0.45,-0.5; set label "-0.1" at -0.65,-0.5
set label "0" at -0.77,-0.50; set label "0.1" at -0.92,-0.40
set label "0.1" at 0.78, 0.90; set label "0.3" at 0.58,0.90
set label "0.5" at 0.38, 0.90; set label "0.9" at 0.10,0.90
plot [-1:1] [-1:1] 'testx3p.res' using 1:2 title "-0.9" w lp,\
                  'testx3p.res' using 1:3 title "-0.5" w lp,\
                  'testx3p.res' using 1:4 title "-0.3" w lp,\
                  'testx3p.res' using 1:5 title "-0.1" w lp,\
                  'testx3p.res' using 1:6 title " 0.0" w l, \
                  'testx3p.res' using 1:7 title " 0.1" w lp,\
                  'testx3p.res' using 1:8 title " 0.3" w lp,\
                  'testx3p.res' using 1:9 title " 0.5" w lp,\
                  'testx3p.res' using 1:10 title " 0.9" w lp

```

- (a) Здесь *Третья строка* через оператора **set key** определяет вид **легенды**;
 - (b) Слова **right** и **bottom** информируют **gnuplot** о расположении **легенда** в правом нижнем углу рисунка;
 - (c) Число после **samplen** задает длину шаблона линии в легенде;
 - (d) Число после **spacing** – расстояние между строками легенды;
 - (e) Число после **width** приращение ширины легенды
 - (f) Строковая констата после **title** задает имя параметра **легенды**: в предыдущем варианте оно по умолчанию оказалось пустым; сейчас же определено буквой **p**;
3. *Четвертая строка* задает текст метки, которым хотим именовать рисунок; координаты точки рисунка, относительно которой текст центрируется (**center**); наименование и размер используемого шрифта. Опция **enhanced** в первой строке **gnuplot**-скрипта разрешает в режиме терминала **postscript eps** написание нижних и верхних индексов при формировке названия рисунка.
 4. *Строки с пятой по девятую* задают содержимое и места расположения меток, которые помечают на рисунке кривые соответственно значениям параметра **p**.
 5. *Десятая строка* – вызов команды черчения **plot**. Графиков несколько. Аргументы и опции для всех не помещаются в одной строке. Символ их продолжения на очередную – обратный слэш.
 6. Для каждой столбца указывается свое значение параметра, которое в качестве имени кривой автоматически занесется в соответствующее поле легенды.

Таким образом, получается следующий рисунок:



Конструирование надписей на рисунке посредством инструментария **gnuplot**, хотя и удобно, но не всегда выгодно. Шрифты **gnuplot**a несколько отличаются от шрифтов системы **latex**. Поэтому начертания букв на рисунке будут слегка отличаться от своих аналогов в тексте. Так что иногда выгоднее записать нужный текст в рамках **TEX**-команды **figure**, используя команды **TEX**a.

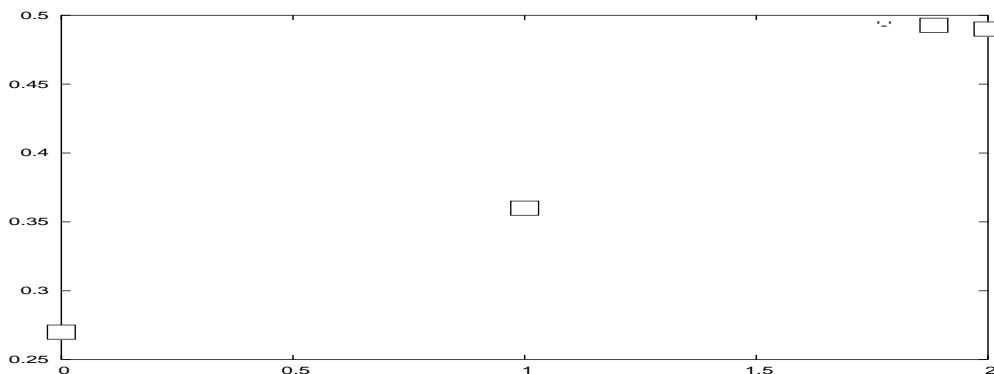
1.3 Построение графика по данным, вводимым с клавиатуры

Команда **plot** позволяет строить график и по данным, которые вводятся с клавиатуры. Указанием о включении именно такого ее режима работы служит использование в качестве имени файла, из которого читаются данные, **особого имени**, обозначаемого знаком **минус** ('-'). Например,

```
gnuplot> plot '-'  
input data ('e' ends) > 0.25  
input data ('e' ends) > 0.36  
input data ('e' ends) > 0.49  
input data ('e' ends) > e
```

нажатие клавиши **enter** после набора команды **plot** с модификатором **'-'** утилита выведет на экран консоли текст-подсказку **input data ('e' ends) >** (*введи данные*), напоминая, что признаком завершения вводимого набора данных служит ввод буквы **e**. Набираем нужное значение (только одно в строке) и снова нажимаем клавишу **enter**. Повторяем ввод данных до тех пор, пока не решим завершить набор,

введя литеру **e**. В результате на экране отобразятся точки графика функции, ординаты которых равны введенным числам, а за соответствующую абсциссу (по умолчанию) принят **номер** вводимого числа (**номер** первого числа полагается равным **нулю**). Конкретно для приведенного примера высветятся три точки с координатами **(0,0.25)**, **(1,0.36)**, **(2,0.49)**.



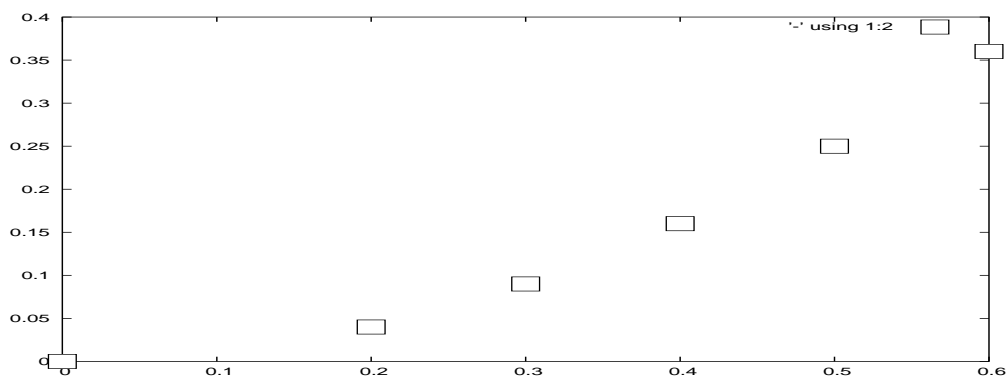
В правом верхнем углу видны две точки: **левая** из них – изображение **легенды**. Его можно передвинуть, например, посредством установки **set key left top** или вообще отказаться от легенды (**set nokey**). Приведенный рисунок в **eps**-формате получен пропуском следующего **gnu**-скрипта:

```
# Файл par1p3.gnu (демонстрация ввода данных с клавиатуры или
# из строк, расположенных непосредственно за командой plot '-')
#=====
set terminal postscript eps
set output 'par1p3.eps'
plot '-' w p pt 70 ps 3
0.27
0.36
0.49
e
```

При желании ввести и абсциссу и ординату достаточно дополнить рассматриваемую модификацию команды **plot '-'** опцией **using**, как делалось в пункте **1.2.4**. Так, например, ввод данных в форме

```
gnuplot> plot '-' using 1:2
input data ('e' ends) > 0.0 0.0
input data ('e' ends) > 0.2 0.04
input data ('e' ends) > 0.3 0.09
input data ('e' ends) > 0.4 0.16
input data ('e' ends) > 0.5 0.25
input data ('e' ends) > 0.6 0.36
input data ('e' ends) > e
gnuplot>
```

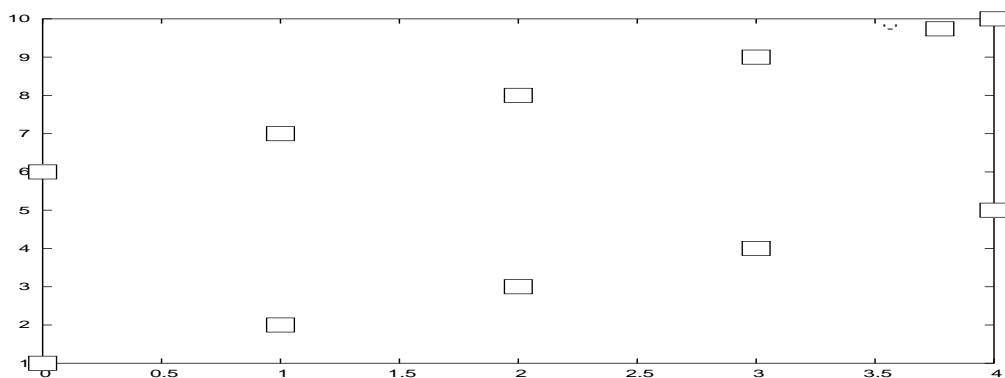
приведет к выводу точек соответствующей части параболы x^2 :



Два набора чисел разделенных двумя (или более) следующими непосредственно друг за другом пробельными строками отобразятся соответственно двумя последовательностями точек. Например, набор

```
gnuplot> plot '-'  
input data ('e' ends) > 1  
input data ('e' ends) > 2  
input data ('e' ends) > 3  
input data ('e' ends) > 4  
input data ('e' ends) > 5  
input data ('e' ends) >  
input data ('e' ends) >  
input data ('e' ends) > 6  
input data ('e' ends) > 7  
input data ('e' ends) > 8  
input data ('e' ends) > 9  
input data ('e' ends) > 10  
input data ('e' ends) > e  
gnuplot>
```

даст рисунок:



1.4 Модификатор (опция) INDEX

Один файл с данными может хранить несколько различных наборов чисел, каждый из которых отделен от соседнего не менее чем двумя строками пробелов. Количество чисел в каждом наборе может быть разное. Пусть по каким-то причинам подобное хранение данных кажется исследователю более предпочтительным нежели расфасовка разнородной информации по разным файлам. Например, меньше вероятность забыть что-либо при копировании – один единственный файл вряд ли забудешь; вероятно и объем программы, работающей с одним файлом, меньше; желание приучить пользователя к оригинальной авторской программе обработки; вероятно в старые добрые времена маломощных ЭВМ подобная организация данных вообще была единственно разумной.

Задача:

Нанести на один рисунок точки соответствующие первому, третьему и пятому наборам, а на второй рисунок второму и четвертому.

Пусть строки файла с первой по пятую хранят набор из пяти чисел: **1, 2, 3, 4, 5**; строки с восьмой по двенадцатую хранят элементы другого набора: **8, 9, 10, 11**; строки с четырнадцатой по двадцатую – третьего: **21, 22, 23, 24, 25, 26, 27**; строки с двадцать третьей по двадцать шестую – четвертого: **36, 37, 38, 39**; и, наконец, строки с двадцать девятой по тридцать третью хранят пятый набор: **43, 44, 45, 46, 33**.

Для выбора из заданных таким образом наборов нужного команда **plot** предоставляет модификатор **index** – служебное слово, числа после которого указывают номера (**индексы**) выбираемых наборов. Пусть **mysets.dat** имя файла, хранящего указанные наборы. Пусть помимо упомянутых наборов чисел в файле содержатся еще и комментарии (всегда полезно иметь под рукой текст, не мешающий работе утилиты). Не зависимо от наличия или отсутствия комментариев поставленная задача решается **gnuplot**-запуском скрипта **mysetsx11.gnu**:

```
#                               Демонстрация использования модификатора INDEX.
set terminal x11
set output
plot 'mysets.dat' index 0:4:2
print " Zero, second and fourth sets. Press ENTER"
pause -1
plot '' index 1:3:2
print " The first and three sets. Press ENTER"
pause -1
plot ''
print " All sets. Press ENTER"
pause -1
# Обратите внимание, что отсутствие имени файла между апострофами,
# команда PLOT воспринимает как синоним имени ранее явно указанного
# файла, позволяя нам сократить запись скрипта.
```

```

# Файл mysets.dat хранит набор из пяти порций чисел.
# Каждая порция отделена от соседней не менее чем двумя
# строками пробелов:
#
#           Первый набор:
1
2
3
4
5

#
#           Второй набор:
8
9
10
11

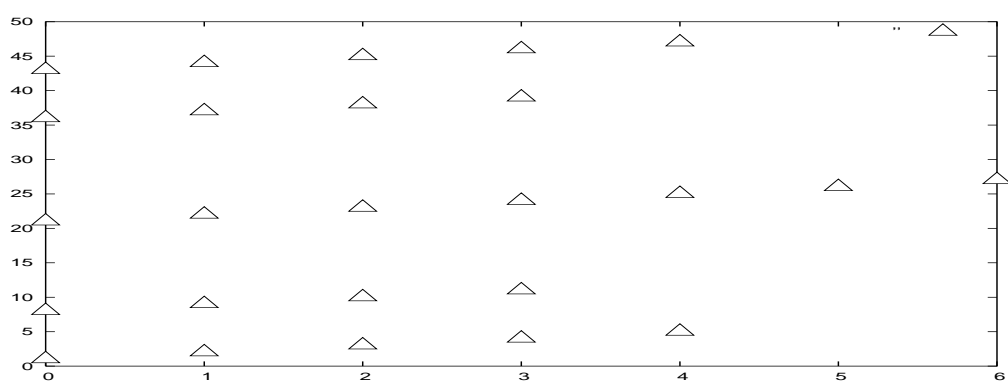
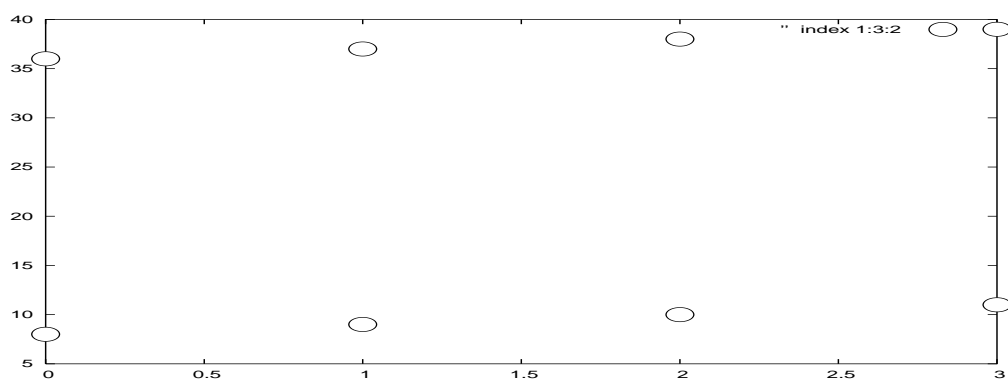
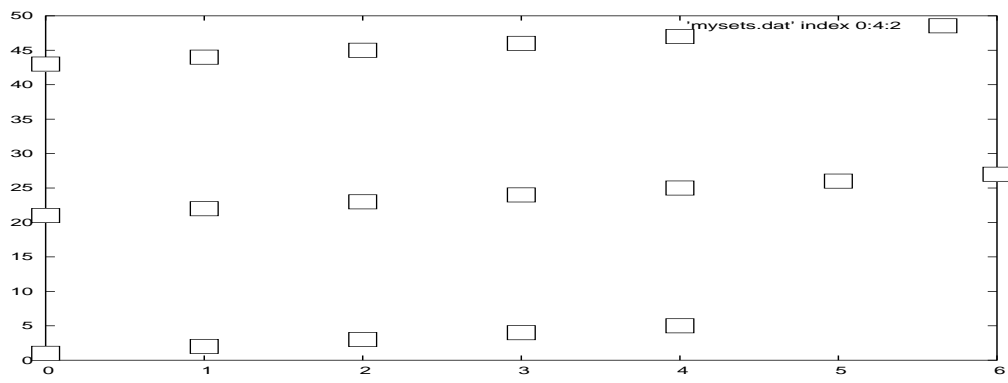
#
#           Третий набор:
21
22
23
 24
 25
  26
  27

#
#           Четвертый набор:
36
37
 38
  39

#
#           Пятый набор:
43
 44
  45
  46
    47

```

Обратим внимание, что каждое данное может быть расположено в любом месте строки. На следующей странице приведены соответствующие рисунки, полученные в **eps**-формате, и соответствующий им **gnuplot**-скрипт.



```

set terminal postscript eps
set output 'myset042.eps'
plot 'mysets.dat' index 0:4:2 w p pt 70 ps 3
set output 'myset132.eps'
plot '' index 1:3:2          w p pt 71 ps 3
set output 'mysets.eps'
plot ''                      w p pt 72 ps 3

```


1.5 Вызов загрузочного модуля программы из GNUPLOTa

Иногда нет желания запоминать результаты работы **ФОРТРАН**- или **СИ**- программы в отдельном файле, а график построить хочется. **gnuplot** предоставляет возможность перенаправить данные, идущие на стандартный вывод программы пользователя, непосредственно на вход к команде **plot**. Например, дана программа на **ФОРТРАНе**, которая вычисляет значения функции $f(x)=x**3$:

```
write(*,'(1x,3x,' '#   x''',11x,' 'y(x)=x**3''')')
do i=1, 10
  x=0.1*i
  y=x*x*x
  write(*,'(1x,e15.7,e15.7)') x, y
enddo
end
```

Заметим, что программа написана в стиле древних **ФОРТРАН**-программ и не имеет никаких особых операторов, нацеленных на использование утилиты **gnuplot**. Результат ее работы просто выводится на экран в виде таблички:

```
[aw@theor3 fourie]$ g77 mypipe.for
[aw@theor3 fourie]$ ./a.out
#   x           y(x)=x**3
0.1000000E+00  0.1000000E-02
0.2000000E+00  0.8000000E-02
0.3000000E+00  0.2700000E-01
0.4000000E+00  0.6400000E-01
0.5000000E+00  0.1250000E+00
0.6000000E+00  0.2160000E+00
0.7000000E+00  0.3430000E+00
0.8000000E+00  0.5120000E+00
0.9000000E+00  0.7290001E+00
0.1000000E+01  0.1000000E+01
[aw@theor3 fourie]$
```

Войдем в среду утилиты и вызовем команду **plot** в форме:

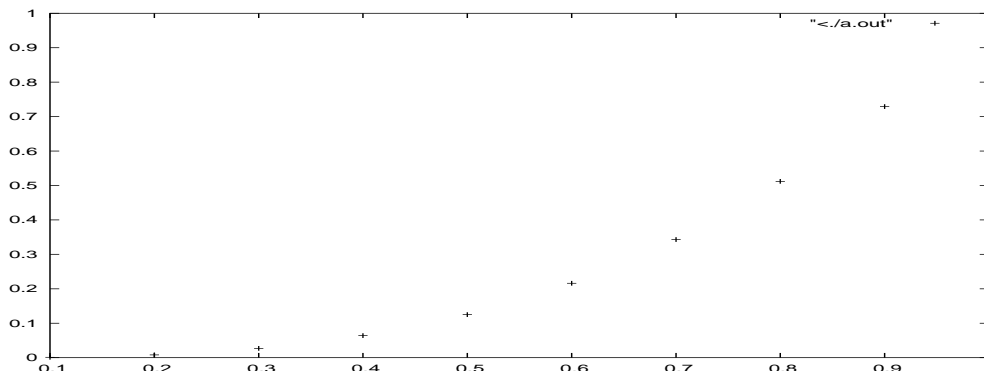
```
gnuplot> plot "<./a.out"
```

Здесь **./a.out** – загрузочный модуль, полученный в результате трансляции приведенного выше исходного. Значок **<** перед именем загрузочного модуля служит указанием о перенаправлении выходного потока программы **./a.out** на вход к команде **plot**. В результате на экране появится график функции, вычисленной программой пользователя. Безусловно, указанный вариант команды **plot** может быть помещен и в **gnuplot**-скрипт. Текст последнего для получения графика в **eps**-формате:

```

set terminal postscript eps
set output 'mypipe.eps'
plot "<./a.out"

```



1.6 Организация циклоподобных процессов

Утилита **gnuplot** не предоставляет в явном виде оператор цикла. Организация циклического процесса осуществляется на основе оператора загрузки **load**, условного оператора **if** и оператора **reread**, который перечитывает заново **gnuplot**-файл, в котором находится. Ниже приведен текст **gnuplot**-программы, которая после очередного нажатия клавиши **enter** высвечивает график функции $\sin(kx)$ для очередного значения $k=0,1,2,\dots,10$.

```

# главная gnu-программа; файл mainloop.gnu.
k=0 # инициализация частоты
load "testlooper.gnu" # загрузка тела цикла из файла testlooper.gnu

# Тело цикла. Файл testlooper.gnu
plot sin(x*k) # строим график.
print "k=",k # Печатаем на консоли частоту синусоиды.
k=k+1 # Увеличиваем частоту на единицу.
pause -1 "Press ENTER" # Задержка выполнения до нажатия клавиши ENTER
if (k<11) reread # Контроль условия продолжения цикла.

```

Иницилируем автоматическое выполнение файла **mainloop.gnu**, запустив в командной строке виртуальной графической консоли команду **gnuplot mainloop.gnu**, и следуя подсказкам, высвечиваемым программой.

1. График, высвеченный в первый раз, – прямая линия $y=0$, как и должно быть.
2. На экране консоли видны фразы: $k=0$ и в следующей строке **Press Enter**
3. Нажатие на клавишу **Enter** не приводит к ожидаемому продолжению, так как в данный момент активно окошко, в котором высвечен график, а не окно консоли. Активируем последнее левой клавишей мыши. При этом полезно его верхнюю границу совместить с верхней границей окна с графиком, а левую границу с правой границей последнего. И тогда каждое нажатие на клавишу **enter** будет приводить к смене графика.
4. Анализируя искажения графика в окрестности прямых $y=-1$ и $y=+1$, заключаем, что, видимо, количество точек дискретизации функции, назначаемое по умолчанию и равное **100**, недостаточно для высоких частот. Поэтому либо добавим в файл главной программы оператор соответствующего переназначения **set samples 1000** или просто вызовем **gnuplot** и осуществим указанное переназначение вручную, после чего иницилируем выполнение главной программы посредством **load mainloop.gnu**.

1.6.1 Картинка образцов (стилей вычерчивания) линий и значков.

Изображение картинки из пункта **1.2.3** со страницы **12** получается при запуске командой **gnuplot mysymbol.gpi** скрипта из файла **mysymbol.gpi**, который играет роль главной программы, и, в свою очередь, вызывает скрипты из файлов **myarrow** и **mypoint**. Первый из последних двух обеспечивает смену образца линии под номером **k** на следующий образец, а второй – смену образца значка. Тексты их предельно просты:

```
#                                     Файл myarrow
# =====
#
set arrow from f1(k),g1(k) to f2(k),g2(k) nohead lt k
k=k+1 ;          # изменяем номер образца линии
if (k<76) reread # перечитываем и исполняем данный файл пока нужно.

#                                     Файл mypoint
# =====
#
plot f(k),g(k) w p 0 k # чертим k-ый символ в нужном месте диаграммы;
k=k+1;                # увеличиваем номер символа;
if (k<76) reread     # перечитываем и перевыполняем данный файл пока нужно
```

```

#                               Файл mysymbol.gpi (главная программа)
#=====
set terminal postscript eps;      # режим postscript eps.
set output 'ps_symbols.eps'      # выходной файл.
set noborder; set nozeroaxis;    # Не нужны: ни бордюр рисунка, ни оси,
set noxtics; set noytics; set nokey # ни отсечки по ним, ни легенда.
set size square;                # Установка: равного масштаба по осям,
set xrange [-1.2:1.2]; set yrange [-1.2:1.2]; # их диапазонов изменения,
set parametric; set samples 2    # параметрического режима, числа узлов.

f(x)=cos(pi*x/38.0); g(x)=sin(pi*x/38.0) # Описание функций расчета

set label '(#70--75 are opaque)' at 0,0 center # координат оцифровки
set label '0' at 1.07*f(0), 1.07*g(0) center # круговой шкалы номеров
set label '10' at 1.07*f(10), 1.07*g(10) center # шаблонов линий и значков
set label '20' at 1.07*f(20), 1.07*g(20) center # с шагом 10 и установка
set label '30' at 1.07*f(30), 1.07*g(30) center # соответствующих меток.
set label '40' at 1.07*f(40), 1.07*g(40) center
set label '50' at 1.07*f(50), 1.07*g(50) center
set label '60' at 1.07*f(60), 1.07*g(60) center
set label '70' at 1.07*f(70), 1.07*g(70) center

f1(x)=0.45 *cos(pi*x/38.0); g1(x)=0.45 *sin(pi*x/38.0) # функции расчета
f2(x)=0.97*cos(pi*x/38.0); g2(x)=0.97*sin(pi*x/38.0) # концов стрелок

k=0; load 'myarrow' # настройка счетчика стрелок и загрузка файла их установки

set multiplot # режим черчения нескольких графиков на один рисунок;
k=0; load 'mypoint' # обнуление счетчика значков и загрузка файла их вывода;

```

Файл `ps_symbols.gpi` из каталога `/usr/share/doc/gnuplot-3.7.3/psdoc/` нацелен на выполнение той же работы, что и рассмотренные. Очевидно, что файлы `myarrow` и `mypoint` в 13 раз короче соответствующих частей `ps_symbols.gpi`. Последний реализует, вероятно, самый естественный и наиболее быстрый способ построения желаемого рисунка, хотя и не единственный. Желание сократить объем исходного `gnuplot`-текста и привело к предложенным циклическим алгоритмам расчета координат концевых точек стрелок и смене шаблона значка, используемого для изображения точки графика. Заметим, что последняя операция должна выполняться в режиме `multiplot` (иначе элементы рисунка, нанесенные ранее, в частности, и предыдущие значки) не будут сохраняться. Вывод рисунка происходит гораздо медленнее чем в случае `ps_symbols.gpi`, поскольку для вывода каждого значка требуется новый вызов команды `plot`. Так что минимизация объема исходного текста произошла за счет проигрыша во времени исполнения. Тем не менее данный пример полезен как демонстрация инструментария утилиты.

Замечание: При использовании режима `set multiplot` проявка всех изображений на экране часто требует после нанесения последнего рисунка прямого указания об отмене режима, то есть вызова команды `set nomultiplot`.

1.6.2 Моделирование движения (фигуры Лиссажу)

Начертить в декартовой системе координат траекторию точки, которая совершает два свободных гармонических колебания по осям абсцисс и ординат соответственно. Удобно параметрическое описание процесса:

$$x(t) = \sin(a * t + p) \quad , y(t) = \sin(b * t + q)$$

Здесь

1. **t** – параметр (именно такое имя **gnuplot** выбирает для параметра по умолчанию). В терминах утилиты ключевым словом, задающим имя параметра, служит слово **dummy**. В частности, в режиме обычной декартовой системы координат по умолчанию принимается установка **set dummy x, y**, а в параметрическом **set dummy t**. Если же по проблемным причинам хотим переопределить имя независимой переменной, то делаем это явно оператором **set dummy**. Команда **show dummy** высвечивает имя текущего параметра.
2. **a** и **b** – частоты первого и второго колебания;
3. **p** и **q** – соответствующие фазы;
4. **x** и **y** – абсцисса и ордината, зависящие от параметра **t**.

При фиксированных параметрах **a**, **b**, **p** и **q** требуемая кривая чертится просто

```
a=1; b=1; p=0; q=0;
set parametric; set size square
plot sin(a*t+p), sin(b*t+q)
```

Результат – диагональ **y=x**, поскольку и амплитуды, и частоты, и фазы **x**- и **y**-гармонических составляющих одинаковы. При **q=pi/2** смещение по ординате равно **cos(b*t)** так, что кривая окажется окружностью, а при иных значениях фазы – эллипсом. Интересно посмотреть в режиме мультипликации, как меняется форма кривой при изменении фазы одной гармоники. Посредством уже рассмотренных в двух предыдущих примерах операторов **load**, **if** и **reread** это делается элементарно. Ниже дан текст соответствующего **gnuplot**-скрипта, который

1. строит на экране траекторию движения точки при **a=2**, **b=1**, **p=0**, **q=0**;
2. перечерчивает ее для каждого из **50** значений фазы **p** из промежутка **[0, 2π]**;
3. выводит график последней кривой через **eps**-терминал в файл **lissagp.eps**;
4. моделирует траекторию набором дискретных точек при **p = 2π** и **q=0**;
5. демонстрирует изменение дискретной траектории с фазой **q**;
6. выводит последнюю в файл с именем **lissagq.eps** через **eps**-терминал;
7. восстанавливает настройки умолчания и завершает работу при нажатии **enter**.

```

#                               Файл lissagu.gnu (главная программа).
#=====
#
set parametric; set nokey;      # параметрический режим; отсутствие легенды;
set size square                 # установка одинакового масштаба по осям;
a=2; b=1; p=0; q=0;            # явное задание частот и фаз;
plot sin(a*t+p), sin(b*t+q)    # вычерчивание требуемой кривой;
pause 2 " 1) Activate run-window!" # "Активируйте окно пропуска!"
pause -1 " 2) for demonstration of the motion to press Enter!"

load 'figp.gnu'                 # загрузка и обработка цикла по p;

set terminal postscript eps     # Через терминал postscript eps выводится
set output 'lissagp.eps'       # в файл lissagu.eps непрерывная кривая
replot                         # траектории точки для  $p=2\pi$ 

set terminal x11                # возвращаем терминал в режим визуализации x11;
set output                      # устанавливаем вывод на экран;
set pointsize 3;               # размер символа втрое больше размера по умолчанию;
set samples 50;                # количество точек дискретизации кривой =50;
plot sin(a*t+p), sin(b*t+q) with points 0 70 # высветка точек траектории;
pause -1 " for demonstration of the motion to press Enter!"

load 'figq.gnu'                 # загрузка и обработка цикла по q;

set terminal postscript eps;    # Через терминал postscript eps выводится
set output 'lissagq.eps';      # в файл lissagu.eps непрерывная кривая
replot                         # траектории точки для  $p=2\pi$ 
pause -1 "To exit press enter!"
reset

#                               файл figp.gnu (организация цикла по p).
#                               =====
replot sin(a*t+p), sin(b*t+p)
p=p+2*pi/50; if (p<2*pi) reread

#                               файл figq (организация цикла по q).
#                               =====
replot sin(a*t+p), sin(b*t+q) with points 0 70
q=q+2*pi/100; if (q<2*pi) reread

```

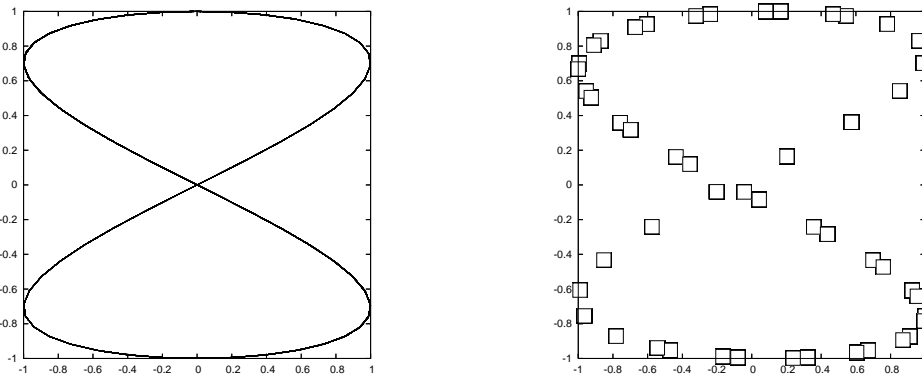
Предположим, что рисунки из файлов **lissagp.eps** и **lissagq.eps** надо разместить на странице рядом, а не один под другим. Наиболее просто достичь желаемого посредством подключения через оператор `\usespackage` пакета **epsf**, одна из команд которого `\epsfbox{filename.eps}` работает аналогично команде **mbox**, но по отношению к содержимому графического файла с расширением **eps**. Соответствующий **TEX**-фрагмент ее вызова может иметь вид

```

\begin{figure}[ht]
\hspace{0\textwidth}\epsfxsize=0.5\textwidth \epsfbox{lissagp.eps}
\hspace{0\textwidth}\epsfxsize=0.5\textwidth \epsfbox{lissagq.eps}
\end{figure}

```

Здесь `\textwidth` – название единицы измерения длины, которая равна ширине строки текста на странице. Команда `\hspace{0\textwidth}` означает, что отступ от начала строки равен нулю (сравни, например, `\hspace{3cm}`). Команда `\epsfxsize = 0.5\textwidth` означает, что размер рисунка из `eps`-файла должен быть равен половине ширины текстовой строки. Окончательно получается следующее изображение:



Видно, что оцифровка осей при выбранном размере рисунка кажется чересчур мелкой. Внесем в текст файла `lissagu.gnu` соответствующие исправления. Модифицированный файл назовем `lissagu1.gnu`

1.6.3 Изменение количества засечек осей

Установка засечек, например, по оси абсцисс, осуществляется командой `set xtics` (если засечки не нужны, то используем установку `set noxtics`). В случае данного рисунка для засечек по обеим осям более предпочтительным кажется шаг 0,4. Таким образом, в тексте `lissagu1.gnu`, в разделе установок вывода для непрерывной траектории через `eps`-терминал появились четыре оператора, настраивающих расположение и формат оцифровки засечек:

```

set xtics -0.8, 0.4, 0.8 # Отсечки оси абсцисс идут от -0.8 с шагом 0.4
set ytics -0.8, 0.4, 0.8 # до 0.8; аналогично и по оси ординат.
set format x "%4.1f"     # Формат оцифровки отсечек: четыре символа и
set format y "%4.1f"     # одна цифра после фиксированной запятой.

```

При указании формата оцифровки отсечек `gnuplot` использует синтаксис спецификаторов форматного вывода языка СИ. Заметим, что при выводе через тот же `eps`-терминал траектории в виде дискретного набора точек упомянутый выше набор можно не указывать: установки для данного типа терминала сохранились по умолчанию, несмотря на работу между двумя `eps`-выводами вывода через `x11`-терминал с иной настройкой оцифровок.

```

#                               Файл lissagu.gnu (главная программа).
#=====
#
set parametric; set nokey;      # параметрический режим; отсутствие легенды;
set size square                 # установка одинакового масштаба по осям;
a=2; b=1; p=0; q=0;           # явное задание частот и фаз;
plot sin(a*t+p), sin(b*t+q)    # вычерчивание требуемой кривой;
pause 2 " 1) Activate run-window!" # "Активируйте окно пропуска!"
pause -1 " 2) for demonstration of the motion to press Enter!"

load 'figp.gnu'                # загрузка и обработка цикла по p;

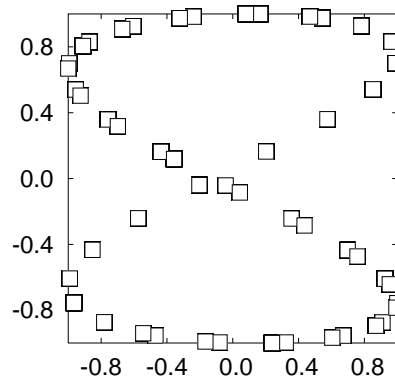
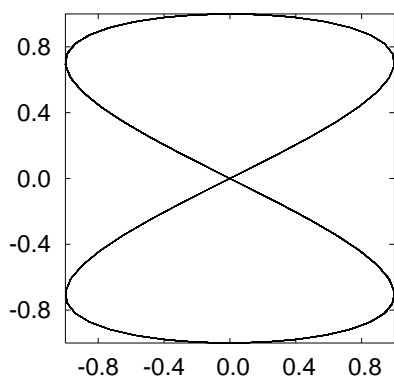
set terminal postscript eps     # Через терминал postscript eps выводится
set output 'lissagp.eps'       # в файл lissagu.eps непрерывная кривая
replot                         # траектории точки для  $p=2\pi$ 

set terminal x11                # возвращаем терминал в режим визуализации x11;
set output                      # устанавливаем вывод на экран;
set pointsize 3;               # размер символа втрое больше размера по умолчанию;
set samples 50;                # количество точек дискретизации кривой =50;
plot sin(a*t+p), sin(b*t+q) with points 0 70 # подсветка точек траектории;
pause -1 " for demonstration of the motion to press Enter!"

load 'figq.gnu'                # загрузка и обработка цикла по q;

set terminal postscript eps;    # Через терминал postscript eps выводится
set output 'lissagq.eps';      # в файл lissagu.eps непрерывная кривая
replot                         # траектории точки для  $p=2\pi$ 
pause -1 "To exit press enter!"
reset

```

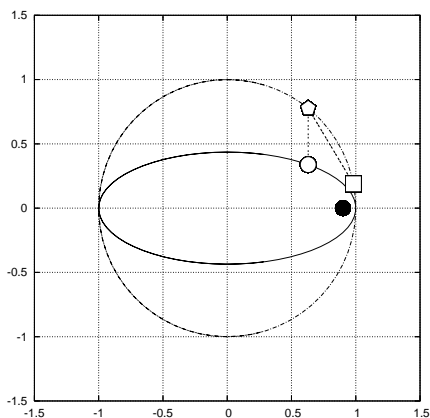


1.7 Пример демонстрации решения задачи двух тел.

Цель – смоделировать на экране движение планеты вокруг Солнца по эллиптической орбите. Алгоритм решения:

1. Задать \mathbf{a} (в а.е.) – размер большой полуоси орбиты.
2. Задать e эксцентриситет орбиты (определяется отношением расстояния фокуса эллипса от его центра к длине большой полуоси и характеризует меру отклонения эллипса от окружности, для которой $e=0$).
3. вычислить период оборота по третьему закону Кеплера $T = \sqrt{a^3}$
4. Задать количество точек дискретизации орбиты n .
5. Вычислить шаг по времени $ht=T/(n-1)$.
6. Поочередно для каждой i -ой точки орбиты вычислять
 - (a) текущий момент времени $t = i * ht$,
 - (b) значение средней аномалии по формуле $M = \frac{2\pi}{T}t$ (предполагается, что планета проходит через перигелий в момент $t_0 = 0$).
 - (c) эксцентрическую аномалию E , посредством решения уравнения Кеплера $E - e \sin E = M$, причем, если $M = k\pi$, то и $E = k\pi$
 - (d) истинную аномалию v по формуле $v = 2 * \arctg \sqrt{\frac{1+e}{1-e}} \operatorname{tg} \frac{E}{2}$
 - (e) радиус-вектор $\mathbf{r} = \mathbf{a}(1 - e \cos E)$
 - (f) выводить положение планеты на орбите

Результат работы программы, полностью написанной на языке утилиты **gnuplot**, можно увидеть на экране в цветном варианте, инициировав ее выполнение командой **gnuplot mainkepl.gnu**. Текст программы приведен на следующей странице. Здесь же дана картинка, получающаяся на экране в конечном итоге.



Черный кружок моделирует звезду;
квадратик – точку, соответствующую последнему значению текущей средней аномалии на круговой орбите, радиус которой равен большой полуоси;
пятиугольник дает положение на этой круговой орбите точки, соответствующей эксцентрической аномалии; а **кружок** – положение планеты на соответствующей эллиптической орбите.
Пунктирные линии образуют треугольник, образованный упомянутыми точками.

1.7.1 GNUPLOT-скрипт

```
#                               файл mainkepl (главная программа).
#=====
set parametric; set nokey;      # параметрический режим; отсутствие легенды;
set size square                 # установка одинакового масштаба по осям;
set xrange[-1.5:1.5]           # --"-- диапазона по оси абсцисс;
set yrange[-1.5:1.5]           # --"-- диапазона по оси ординат;
set pointsize 3                 # --"-- размера символа;
set grid                         # --"-- высветки координатной сетки;
set dummy w                     # имя параметра параметрического режима;
dmod(x,y)=x-floor(x/y)*y        # функция поиска остатка от деления x на y

load 'param.gnu'                # ввод параметров задачи из файла param.gnu

print "a=",a," e=",e, " n=", n, " eps=", eps
  b=a*sqrt(1-e*e)                # расчет малой полуоси;
tt=sqrt(a*a*a); print " tt=", tt # расчет периода оборота;
ht=tt/(n-1)                      # расчет шага по времени;
i=0                               # инициализация номера текущей точки;
load 'cyclen.gnu'               # обработка текущей точки орбиты;
load 'orbita.gnu'               # вывода последней картинкой в eps-файл;
pause -1                         # наслаждаемся последней картинкой;
reset                            # восстанавливаем установки умолчания.

#                               Файл param.gnu с параметрами задачи.
#=====
a=1      # большая полуось;
e=0.9    # эксцентриситет;
eps=1e-7 # требуемая погрешность решения уравнения Кеплера;
n=100    # количество точек дискретизации;

#                               Файл orbita.gnu (построение eps-картинки).
# =====
set terminal postscript eps
set output 'kepler.eps'
set arrow from xe, ye to xe, yv nohead lt 7
set arrow from xe, ye to xm, ym nohead lt 7
set arrow from xm, ym to xe, ye nohead lt 7
plot a*e, 0 w p 0 7, \
      a* cos(w), a*sin(w) with l lt 5 lw 2, \
      a* cos(w), b*sin(w) with l lt 1 lw 2, \
      x+a*e, y w p 0 71, \
      xm, ym w p 0 70 , \
      xe, ye w p 0 75
```

```

# Файл cyclen.gnu (расчет положения планеты на текущий момент времени)
#=====
t=i*ht # расчет текущего момента времени
dm=2*pi * t/tt # и текущей средней аномалии;
de=dm # нач. приближение для эксцентрической
if (dmod(dm,pi)!=0) load "iterat.gnu" # если средняя не кратна pi, то
# решаем уравнение Кеплера;
r=a*(1-e*cos(de)) # вычисляем радиус-вектор,
v=2*atan(sqrt( (1+e)/(1-e) ) *tan(de/2) ) # истинную аномалию,
x=r*cos(v); y=r*sin(v) # координаты планеты отн. фокуса;
xm=a*cos(dm); ym=a*sin(dm) # --"-- средней аномалии отн. центра;
xe=a*cos(de); ye=a*sin(de); # --"-- эксцентрической --"---"---";
yv=b*sin(de); # --"-- ордината планеты --"---"---";
set arrow from xe, ye to xe, yv nohead lt 7 # сторона эксцентрическая-истинная,
set arrow from xe, ye to xm, ym nohead lt 7 # --"---"---"---"---"---"--- -средняя,
set arrow from xm, ym to xe, yv nohead lt 7 # --"---"---"---"--- средняя-истинная;

plot a*e, 0 w p 0 7, \
a*cos(w), a*sin(w) with l lt 5 lw 2, \
a*cos(w), b*sin(w) with l lt 1 lw 2, \
x+a*e, y w p 0 71, \
xm, ym w p 0 70, \
xe, ye w p 0 75

set noarrow # убирание нарисованных сторон треугольника;
i=i+1 # номер очередного момента времени
if (i<=3*n) reread # если он не больше 3*n, то перевыполняем этот файл.

# Файл iterat.gnu (решение уравнения Кеплера методом итераций).
# =====
d1=de
de=dm+e*sin(de)
#print d1, '...', de
if (abs(d1-de)>eps) reread

```

Конечно, использование **gnuplot**a в качестве основного расчетного средства (что здесь продемонстрировано) вряд ли оправдано для проведения сложных расчетов, поскольку временные затраты утилиты вряд ли сопоставимы со скоростными характеристиками работы загрузочных модулей, сгенерированных компиляторами. Да и вычислительный инструментарий **gnuplot**a, несмотря на достаточно большой набор операций и встроенных математических функций не может сравниться с накопленным за десятилетия объемом **ФОРТРАН**- или **СИ**- пакетов прикладных программ. Предположим, что решенная нами задача – сложная настолько, что ее расчетную часть выгодно выполнить на **ФОРТРАНе**, результаты записать в файл, а **gnuplot** использовать по его прямому назначению для графической визуализации желаемых графиков и рисунков.

1.7.2 ФОРТРАН-программа расчета координат планеты

В качестве исходных данных намечаем вводить те же параметры, что и в предыдущей **gnuplot**-реализации, то есть **a** (большая полуось), **e** (эксцентриситет), **eps** (требуемая погрешность решения уравнения Кеплера методом итераций) и **n** (число точек дискретизации эллиптической орбиты).

В качестве результата надо для каждого момента времени получить значения **средней аномалии, эксцентрической аномалии, истинной аномалии**, длину радиуса-вектора, декартовы координаты интересующих нас точек.

Текст программы в стиле ФОРТРАНа-77

```
! Файл mainkepl.for (главная управляющая программа)
!=====
program mainkepl
implicit none          ! Отключение правила умолчания;

integer ninp / 5 /    ! программное имя файла с вх. параметрами
integer nres / 10 /   ! ---"---"---"---"---"--- с результатами
integer ngnu / 20 /   !

real*8 a, b           ! большая и малая полуоси;
real*8 e               ! эксцентриситет;
real*8 eps            ! требуемая погрешность;
integer n              ! количество точек дискретизации;

real*8 x,y, xm,ym, xe,ye,yv, s
real*8 dm, de, v, d1, r
real*8 pi, pi2
real*8 t, ht, tt
integer i

open(unit=ninp, file='mainkepl.inp') ! открытие файла ввода;
open(unit=nres, file='mainkepl.res') ! открытие файла вывода;
open(unit=ngnu, file='mainfpar.gnu') ! gnu-файл с параметрами ввода;

read (ninp, '(d10.3)') a, e, eps ! ввели параметры real*8;
read (ninp, '( i10 )') n         ! ввели параметры integer;
write(nres,1000) a, e, eps, n    ! контрольная печать параметров;

pi = 4*datan(1d0)               ! расчет математических констант
pi2 = 2*pi
b = a*dsqrt(1-e*e)              ! расчет малой полуоси
tt = dsqrt(a*a*a)               ! ---"--- периода оборота
ht = tt/(n-1)                   ! ---"--- шага по времени

write(ngnu,2000) a, b, e, tt, ht, eps, n, 3 ! параметры для gnuplot
write(*,*) ' tt=',tt,' ht=',ht
```

```

write(nres, 1010)
do i=0,n          ! Пока не обработаны все n точек
  t=i*ht          !..... расчет текущего момента,
  dm=pi2*t/tt    ! --"-- средней аномалии,
  de=dm           ! --"-- нач. пригл. эксцентр.
                  ! if средняя аномалия не кратна pi,
  if (dmod(dm,pi).gt.eps) then ! то эксцентр. ищем путем решения
10  continue      ! ур. Кеплера методом итерации:
      d1=de        !..... расчет нового грубого;
      de=dm+e*dsin(de) ! --"-- уточненного
      if (dabs(d1-de).gt.eps) goto 10 ! пока различие между ними
  endif          ! не окажется меньше eps;
  r=a*(1-e*dcos(de)) ! расчет рад.-вект.,
  v=2*datan(sqrt( (1+e)/(1-e) ) *tan(de/2) ) ! истинной аномалии,
  x=r*dcos(v); y=r*dsin(v) ! координат планеты,
  xm=a*dcos(dm); ym=a*dsin(dm) ! средней аномалии,
  xe=a*dcos(de); ye=a*dsin(de); ! эксцентрической,
  yv=b*dsin(de); ! ординаты планеты.
  write(nres,1011) i,t,dm,de,v,r,x,y,xm,ym,xe,ye,yv,s
enddo
close(nres)      ! закрытие файла вывода.
close(ngnu)
stop
1000 format(1x,'# a=',d15.7/ 1x,'# e=',d15.7/
> 1x,'# eps=',d15.7/ 1x,'# n=',i4)
2000 format(1x,'# ',30x,'Файл mainfpar.gnu'/
> 1x,'# ',60('=')/
>1x,' a=',d15.7,' # большая полуось'/
>1x,' b=',d15.7,' # малая полуось'/
>1x,' e=',d15.7,' # эксцентриситет '/
>1x,' tt=',d15.7,' # период '/
>1x,' ht=',d15.7,' # шаг по времени '/
>1x,' eps=',d15.7,' # погрешность '/
>1x,' n=',i15,' # число узлов дискретизации орбиты'/
>1x,' k=',i15,' # число узлов дискретизации орбиты')

1010 format(1x,'# i ',5x,' t ',10x,'dm',10x,'de',10x,' v',10x,' r',
> 10x,' x',10x,' y',
> 10x,'xm',10x,'ym',10x,'xe',10x,'ye',10x,'yv',10x,' s')
1011 format(1x,i5,e12.4,e12.4,e12.4,e12.4,e12.4,e12.4, e12.4,
> e12.4,e12.4,e12.4,e12.4,e12.4,e12.4)
end

```

1.7.3 GNUPLOT-высветка результата. Опция EVERY

```
#                               Файл mainfort.gnu
# =====
set parametric; set nokey;      # параметрический режим; отсутствие легенды;
set size square                 # установка одинакового масштаба по осям;
set xrange[-1.5:1.5]           # --"-- диапазона по оси абсцисс;
set yrange[-1.5:1.5]           # --"-- диапазона по оси ординат;
set pointsize 3                 # --"-- размера символа;
set grid                        # --"-- подсветки координатной сетки;
set dummy w                     # имя параметра параметрического режима;
load 'mainfpar.gnu'             # ввод параметров из файла mainfpar.gnu

print "a=",a," b=",b," e=",e," eps=", eps," tt=",tt," ht=",ht," n=", n
print "k=",k
i=6
load 'mainplt1.gnu'             # загрузка мультика на один оборот;
pause -1                        # наслаждаемся последней картинкой;
reset

#                               Файл mainplt1.gnu (рисует один оборот)
# =====
set samples 300                # число точек дискретизации формульных кривых
plot a*e, 0 w p 0 7,\
      -a*e, 0 w p 0 7,\
      a*cos(w), a*sin(w) with l lt 5 lw 2,\
      a*cos(w), b*sin(w) with l lt 1 lw 2,\
      'mainkepl.res' every 1::i::i:1 using ($7)+a*e:($8) w p 0 71,\
      'mainkepl.res' every 200::i::i:1 using 9:10 w p 0 70,\
      'mainkepl.res' every 200::i::i:1 using 11:12 w p 0 75
i=i+1; if (i<=n-1) reread
```

Последний **gnuplot**-скрипт:

1. ставит ромбик в правом фокусе орбиты (строка **4**);
2. ставит ромбик в левом фокусе орбиты (строка **5**);
3. чертит круговую орбиту (строка **6**);
4. чертит эллиптическую орбиту (строка **7**);
5. изображает треугольником точку истинного положения планеты, выбирая из текущей **i**-ой строки матрицы результата координаты точки относительно фокуса (абсцисса – **\$7**, ордината – **\$8**), и перевычисляя абсциссу относительно начала координат (строка **8**);
6. изображает крестиком точку пересечения луча средней аномалии, выбирая из текущей **i**-ой строки той же матрицы (столбцы 9 и 10) соответствующие координаты (строка **9**);

7. изображает квадратом точку пересечения луча эксцентрической аномалии с круговой орбитой, выбирая координаты из 11 и 12 столбцов той же i -ой текущей строки (строка **10**);
8. находит номер очередной строки результата (точки орбиты; $i=i+1$);
9. и, наконец, перезагружает сам себя с новым значением i , если последнее меньше числа точек дискретизации орбиты.

Замечания:

1. Номера столбцов матрицы естественно обозначать целыми числами без знака. Опция **using** такую возможность и использует, как основную. Однако встречаются ситуации, когда выбранное значение необходимо дополнительно как-то преобразовать. Восьмая строка **mainplt1.gnu**-скрипта специально нацелена на демонстрацию подобного случая: извлекается значение абсциссы относительно фокуса эллипса, хотя нужно относительно центра. Ясно, что **странно** было бы требовать от **gnuplot** *понимать* под выражением $7+a*e$ операцию сложения произведения $a*e$ не с числом **семь**, а с содержимым седьмого столбца (ведь при иных расчетах возможно и число **семь** придется увеличить на $a*e$). Согласно синтаксису команды **plot** опция **using**, чтобы не перепутать в арифметическом выражении обычную числовую константу с содержимым столбца, номер которого с ней совпадает, требует предварять номер столбца префиксом-указателем **§**.
2. Новый элемент команды **plot**, впервые встретившийся нам в данном **gnuplot**-скрипте, – **модификатор every**. В данном случае его назначение: сообщить команде-чертежнику номер строки, из которой следует выбирать данные, находящиеся в указанных модификаторам **using** столбцах. Если не использовать **every**, то на рисунок попадают все точки из выбранных столбцов. в **using** столбцах. Поскольку мы генерируем эффект движения, Поскольку генерируется эффект движения, то необходимо последовательно высвечивать только одну очередную точку, а затем затирать ее следующим рисунком. Поэтому и потребовалась опция **every**, которая является инструментом управления нужным нам процессом: *выборкой только одной i -ой строки матрицы*.
3. Как видели ранее, подаваемые на вход к команде **plot** в одном файле данные могут быть весьма разной структуры (вспомним о модификаторе **index**). Авторы **gnuplot** постарались предоставить возможность выбирать из файла данные так, как хочет пользователь. Они учли, что пользователь, возможно, поместит в файл не одну, например, матрицу, а много; что выбирать данные он захочет не из каждой (или из каждой), причем не все подряд, а с каким-то шагом по строкам, хотя возможно и все подряд. Хозяин – пользователь. Он главный – ему видней. В результате была реализована семантика и синтаксис модификатора **every** в некоем универсальном общем виде, нацеленном на периодическую выборку данных, именно

- (a) если **every** не задано, то при построении графика используются все строки (точки) указанного в **using** набора;
- (b) первое число после **every** означает шаг по строкам внутри выбранного блока (то есть по строкам данных, которые внутри одного файла разделяются не менее чем двумя строками пробелов);
- (c) затем идет разделитель двоеточие **;**;
- (d) второе число – означает шаг по блокам, если их много;
- (e) **:**
- (f) третье число – номер начальной строки выбранного блока;
- (g) **:**
- (h) четвертое число – номер первого выбранного блока;
- (i) **:**
- (j) номер конечной строки выбранного блока;
- (k) **:**
- (l) номер последнего из выбранных блоков.

В соответствие с изложенным приходим к формировке параметров опции **every** из восьмой, девятой и десятой строк **mainplt1.gnu**-скрипта. Поскольку блок только один, то посредством **every** указано, что номера первой выбираемой точки (третий параметр **i**) и последней (пятый параметр **i**) совпадают. Так что первое число после **every** в нашем скрипте вообще не играет никакой роли (коли из всего набора выбирается каждый раз только одна единственная точка).

2 Представление значений функции.

Наряду с обычно применяемыми способами изображения значения функции на графике в виде точки, обозначаемой каким-нибудь значком, а самого графика в виде ломаной, соединяющей соседние точки, утилита `gnuplot` предоставляет также целый набор неких стандартных шаблонов, которые могут оказаться полезными.

Рассмотрим некоторые из них на примере построения графика функции x^3 , табулированной по промежутку $[-1, 1]$ с шагом `0, 1` программой `exam`:

```
program exam
implicit none
integer i
real    x, y
open (10,file='exam.dat')
do i=-10,10
  x=0.1*i
  y=x*x*x
  write(10,'(e15.7,e15.7)') x,y
enddo
close(10)
end
```

Значения аргумента и функции выведены в файл `exam.dat`:

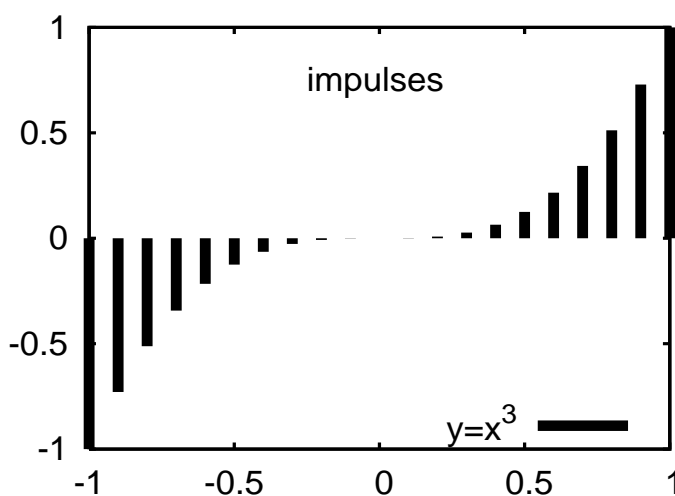
```
-0.1000000E+01 -0.1000000E+01
-0.9000000E+00 -0.7290001E+00
-0.8000000E+00 -0.5120000E+00
-0.7000000E+00 -0.3430000E+00
-0.6000000E+00 -0.2160000E+00
-0.5000000E+00 -0.1250000E+00
-0.4000000E+00 -0.6400000E-01
-0.3000000E+00 -0.2700000E-01
-0.2000000E+00 -0.8000000E-02
-0.1000000E+00 -0.1000000E-02
 0.0000000E+00  0.0000000E+00
 0.1000000E+00  0.1000000E-02
 0.2000000E+00  0.8000000E-02
 0.3000000E+00  0.2700000E-01
 0.4000000E+00  0.6400000E-01
 0.5000000E+00  0.1250000E+00
 0.6000000E+00  0.2160000E+00
 0.7000000E+00  0.3430000E+00
 0.8000000E+00  0.5120000E+00
 0.9000000E+00  0.7290001E+00
 0.1000000E+01  0.1000000E+01
```

2.1 Опция `impulses`.

Опция `impulses` команды `plot` из `gnuplot`-скрипта:

```
set term postscript eps enhanced 32
set output 'fig14.eps'
set key bot right
set border 15 lw 3
set label "impulses" at -0.25, 0.75
plot 'exam.dat' title "y=x^3" w impulses lw 20
```

выводит данные из файла `exam.dat` вертикальными отрезками:

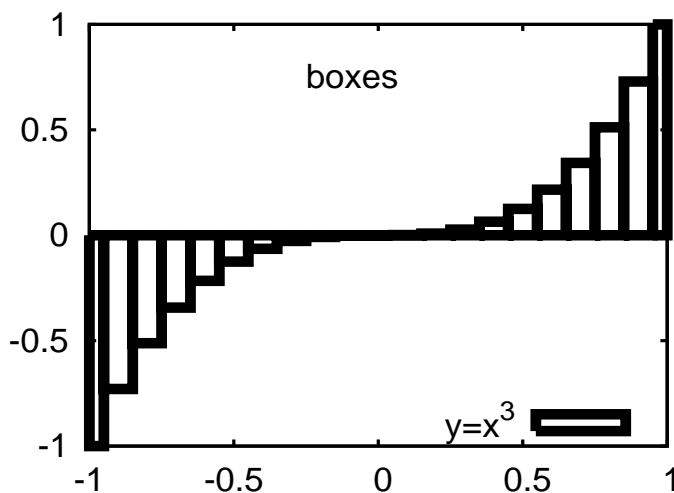


Опция `impulses` удобна для отображения δ -образных всплесков; наглядно выделяет пропуски во временных рядах, может оказаться полезной и при построении гистограмм. Напомним, что в приведённом скрипте

- 1) первая строка устанавливает режим вывода результата в терминах `eps` формата языка `postscript`;
- 2) опция `enhanced` при встрече в текстах на рисунке значков `_` и `^` трактует их в `TeX`-смысле, т.е. позволяет выводить верхние и нижние индексы.
- 3) вывод рисунка в требуемом формате осуществляется в файл `fig14.eps`.
- 4) место легенды (внизу, справа) задаётся оператором `set key bot rihgt`;
- 5) числом `15` в `set border 15 lw 3` определённым образом кодируется вид рамки (объяснение даётся в следующем разделе);
- 6) команда `set` впервые использована с опцией `label`, позволяющей размещать в нужном месте рисунка необходимые метки. По умолчанию текст метки (в данном случае это слово `impulses`) выравнивается влево на точку с графическими координатами, указываемыми после служебного слова `at`.

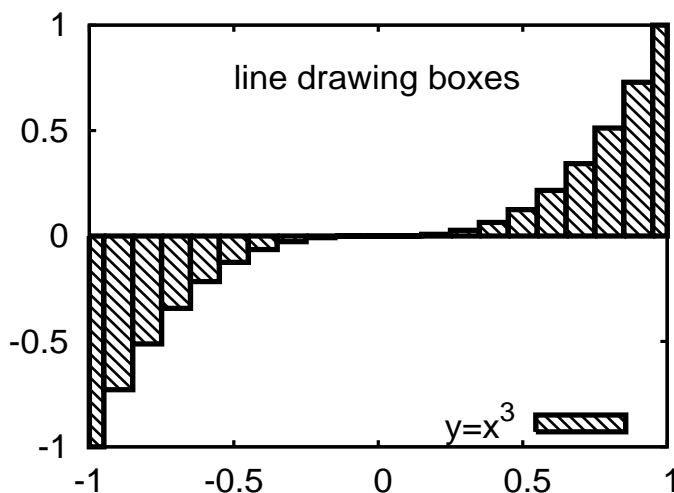
2.2 Опция boxes.

```
set term postscript eps enhanced 32      # Опция
set output 'fig15.eps'                   #          boxes
set key bot right; set border 15 lw 3     # трактует
set label "boxes" at -0.25, 0.75         # данные из файла
plot 'exam.dat' title "y=x^3" w boxes lw 20 # 'exam.dat' так:
```



Возможна заливка **boxes** назначаемым видом штриховки.

```
set term postscript eps enhanced 32      # Назначение штриховки
set output 'fig16.eps'                   # образца номер 4
set key bot right; set border 15 lw 3    #          дается
set style fill pattern 4                  # <--= этой командой.
set label "line drawing boxes" at -0.5, 0.75
plot 'exam.dat' title "y=x^3" w boxes lw 10
```

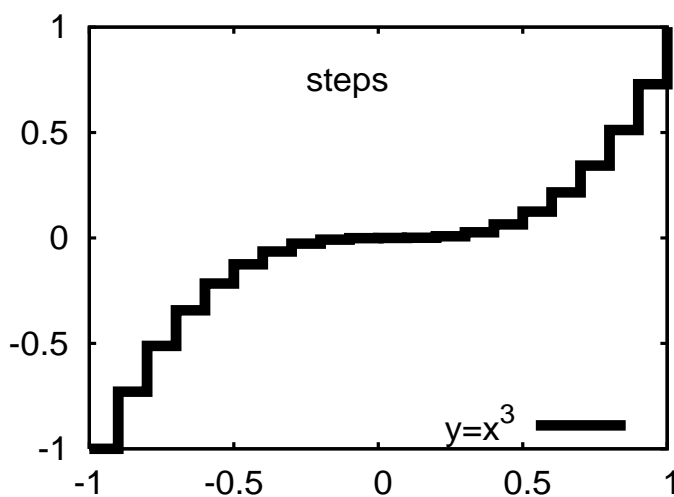


2.3 Опции steps, fsteps и histeps.

Опции **steps**, **fsteps** и **histeps** позволяют отобразить график ступенчатой фигурой с постоянным значением функции в пределах одного шага. Различие между опциями заключено в очередности и способе черчения горизонтальной и вертикальной компонент ступени. Так

1. **steps** чертит сначала горизонтальную часть k -ой ступеньки отрезком прямой $y = y_k$, ограниченной прямыми $x = x_k$ и $x = x_{k+1}$;
2. **fsteps** чертит сначала вертикальную часть k -ой ступеньки отрезком прямой $x = x_k$, ограниченный прямыми $y = y_k$ и $y = y_{k+1}$;
3. **histeps** работает в стиле **steps**, однако вертикальная компонента проводится не в узловой точке, а в срединной точке каждого шага по аргументу, т.е. в точке $\frac{x_k + x_{k+1}}{2}$

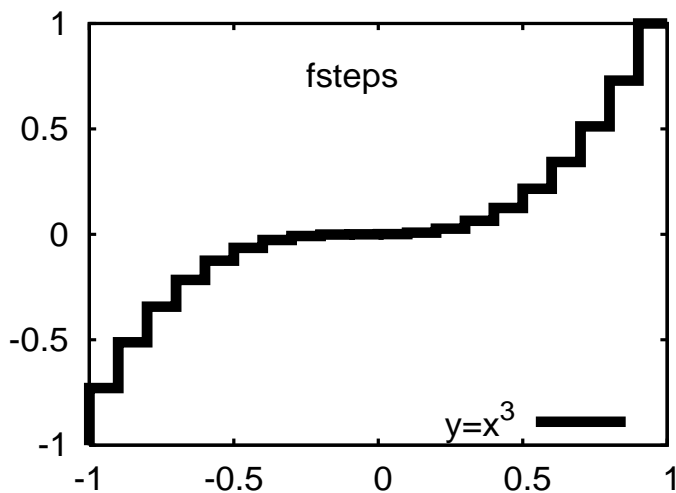
```
set term postscript eps enhanced 32      # Демонстрация
set output 'fig17.eps'                  # работы опции
set key bot right; set border 15 lw 3    # steps.
set label "steps" at -0.25, 0.75
plot 'exam.dat' title "y=x^3" w steps lw 20
```



```

set term postscript eps enhanced 32      # Демонстрация
set output 'fig18.eps'                  # работы опции
set key bot right; set border 15 lw 3   # fsteps.
set label "fsteps" at -0.25, 0.75
plot 'exam.dat' title "y=x^3" w fsteps lw 20

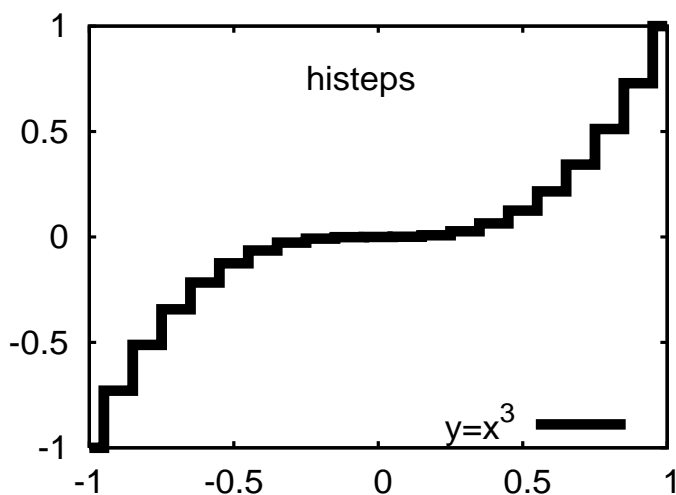
```



```

set term postscript eps enhanced 32      # Демонстрация
set output 'fig19.eps'                  # работы опции
set key bot right; set border 15 lw 3   # histeps.
set label "histeps" at -0.25, 0.75
plot 'exam.dat' title "y=x^3" w histeps lw 20

```



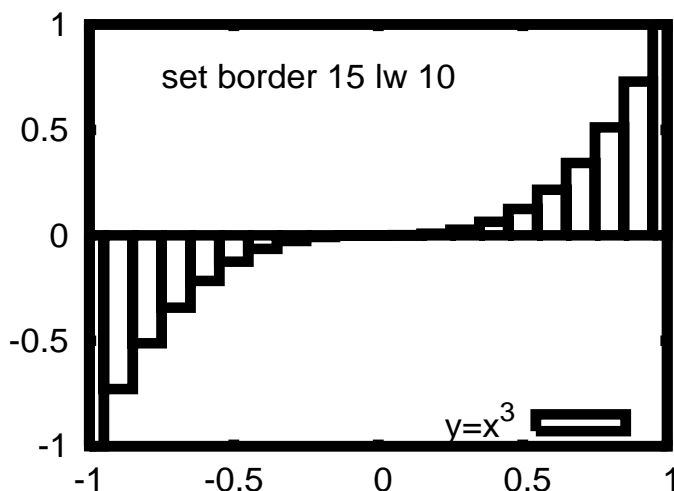
3 Обрамляющая рамка вокруг рисунка.

Известны различные стили оформления рисунка с графиком:

1. ЗаклЮчить рисунок в рамку, на каждой из сторон которой нанесены риски оцифровки осей при размещении самой оцифровки под нижней и слева от левой сторон рамки (этот стиль утилита использует по умолчанию).
2. То же, что и в случае 1, но без нанесения рисок на верхней и правой сторонах рамки.
3. Вообще не чертить верхнюю и правую стороны рамки (естественно, с отменой нанесения соответствующих им рисок).
4. Не только начертить всю рамку, но и установить по верхней стороне рамки иную нежели на нижней шкалу оцифровки.

Изменим толщину рамки, ограничивающей рисунок:

```
set term postscript eps enhanced 32
set output 'fig20.eps';
set border 15 lw 10      # Толщина линии бордюра = 10
set key bot right        # а толщина
set label "set border 15 lw 10" at -0.75,0.75 # линии
plot 'exam.dat' title "y=x^3" w boxes lw 20 # boxes = 20
```



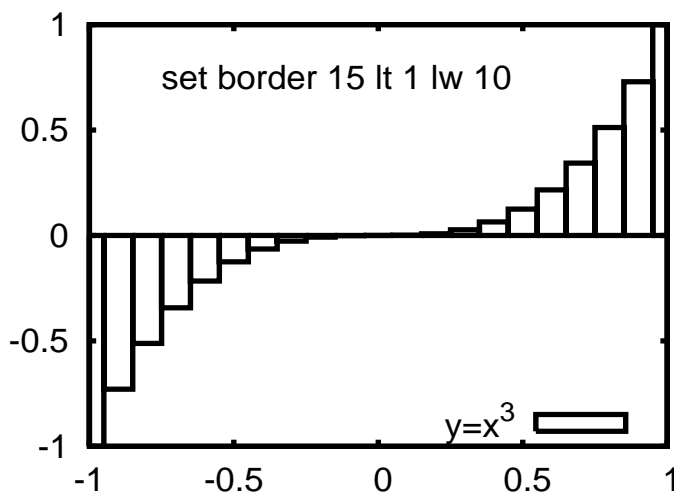
Толщины линий бордюра и шаблона **boxes** выглядят одинаковыми, хотя соответствующие значения опции **lw** различаются вдвое. Причина: для черчения разных объектов выбраны линии разных типов (в частности, разной толщины). Одинаковый тип линий потребует равных значений **lw**:

```

set term postscript eps enhanced 32
set output 'fig21.eps'; set key bot right
set border 15 lt 1 lw 10
set label "set border 15 lt 1 lw 10" at -0.75,0.75
plot 'exam.dat' title "y=x^3" w boxes lt 1 lw 10

```

и соответствующий рисунок будет иметь вид:



В команде установки образца рамки после служебного слова **border** указано некоторое целое число (в скриптах, определяющих этот и предыдущий рисунки, оно равно **15**), каждая из четырех единиц двоичного представления которого информирует утилиту о необходимости черчения соответствующей стороны рамки. В частности,

Число	Двоичный код	Смысловая нагрузка кода: наличие у бордюра
1	0001	нижней стороны
2	0010	левой стороны
4	0100	верхней стороны
8	1000	правой стороны

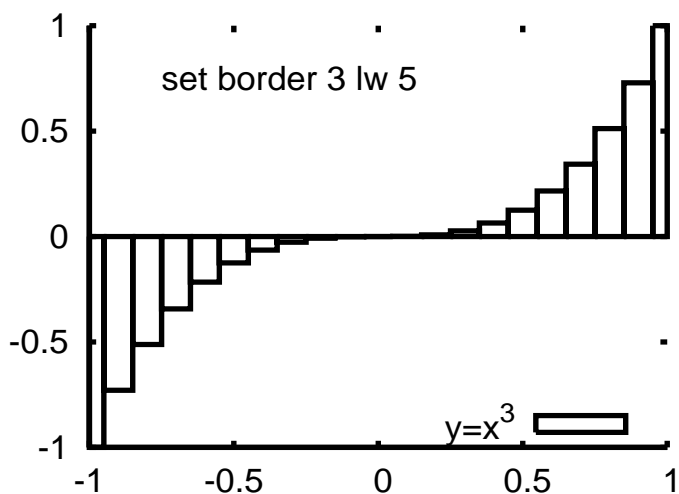
Так что число $15 = 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$ означает наличие у бордюра четырех сторон. Если верхняя и правая стороны рамки не нужны, то вместо числа **15** следует указать число **3** ($3 = 1 * 2^1 + 1 * 2^0$):

```

set term postscript eps enhanced 32          # Формировка
set output 'fig22.eps'; set key bot right    # бордюра
set border 3 lw 5                             # без верхней
set label "set border 3 lw 5" at -0.75,0.75 # и правой
plot 'exam.dat' title "y=x^3" w boxes lw 10 # сторон

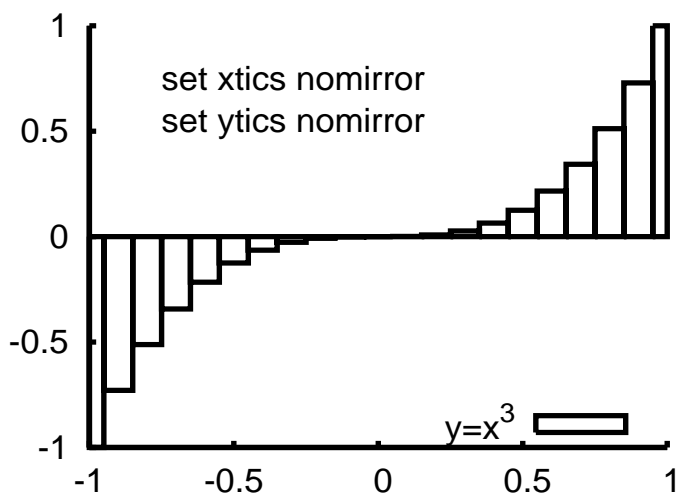
```

В результате возникнет рисунок:



на котором отсутствуют верхняя и правая стороны рамки, хотя наличие реперов оцифровки осей сохранилось. Устранение реперов достигается командами `set xtics nomirror` и `set ytics nomirror`, отменяющих зеркальное отражение реперов:

```
set term postscript eps enhanced 32
set output 'fig23.eps'; set key bot right
set border 3 lt 1 lw 10; set xtics nomirror; set ytics nomirror
set label "set xtics nomirror" at -0.75,0.75
set label "set ytics nomirror" at -0.75,0.55
plot 'exam.dat' title "y=x^3" w boxes lt 1 lw 10
```



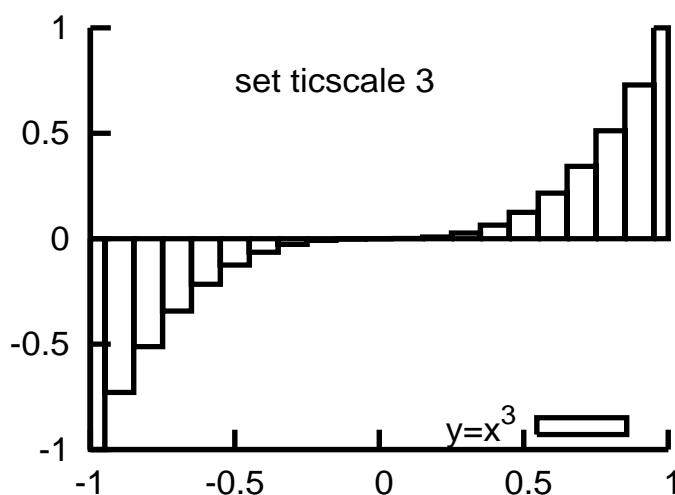
На данном рисунке реперы выглядят чересчур короткими. Изменяет их длину команда

```
set ticscale желаемая_длина_репера
```


В частности, скрипт

```
set term postscript eps enhanced 32
set output 'fig24.eps'
set key bot right; set ticscale 3
set xtics nomirror
set ytics nomirror; set border 3 lt 1 lw 10
set label "set ticscale 3" at -0.5, 0.75
plot 'exam.dat' title "y=x^3" w boxes lt 1 lw 10
```

произведет рисунок



Параметры, определяющие вид оцифровки и длину реперов по умолчанию, выводятся на консоль `gnuplot`-командой `show tics`.

По команде `show border` утилита сообщит:

```
border is drawn 31
Border drawn with linetype -1, linewidth 1.000
```

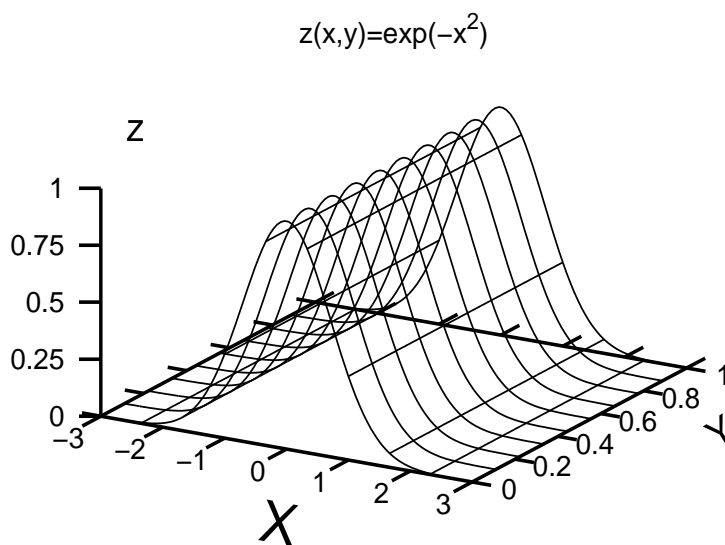
т.е. вид бордюра определяется числом $(31)_{10} = (11111)_2$, в котором есть и единица из разряда весом 2^4 . Разряды с номерами $k = 0(1)3$ (из младшей двоичной тетрады) определяют необходимость черчения ребер прямоугольника, ограничивающего плоскую фигуру. При выводе изображения трехмерной фигуры, который обеспечивается вызовом команды `splot`, может потребоваться бордюр не только для основания фигуры, но и для некоторых (или всех) граней параллелепипеда, содержащего её. Необходимость черчения его ребер при работе `splot` и определяется наличием соответствующей двоичной единицы в разрядах с номерами $k = 4(1)11$. Таким образом, число **31**, выведенное утилитой означает, что наряду с прямоугольником, лежащим в плоскости абсциссы и ординаты **XY**, будет выведен вертикальный отрезок с реперами, соответствующими отсчетам аппликаты. Например, скрипт

```

set term postscript eps enhanced 24
set output 'fig25.eps'; set nokey
set border 31 lw 3          # Шкала оси аппликат.
set encoding koi8r         # Используем кириллицу.
set title "z(x,y)=exp(-x^2)" # Заголовок рисунка.
set tics out               # Все реперы наружу.
set ticscale 3             # Длина репера = 3
set ztics 0,0.25,1        # Параметры оцифровки оси аппликат.
set ticslevel 0           # Уровень плоскости XY-реперов.
set label 1 "абсцисса X" at 0.5, -0.5, 0\
  center rotate by -10\
  font "Helvetica,48"      # подпись к оси абсцисс.
set label 2 "ордината Y" at 5, 0.4, 0\
  center rotate by 33\
  font "Helvetica,32"      # подпись к оси ординат.
set zlabel "аппликата Z"  # подпись к оси аппликат.
splot [-3:3][0:1] exp(-x*x) lw 3

```

получит файл **fig25.eps** с рисунком



Скрипт, получающий последний рисунок, помимо команды **splot** содержит ещё несколько команд, ранее не встречавшихся в данном пособии:

1. **set encoding koi8** – установка возможности использования в скрипте строковых констант, содержащих кириллицу.
2. **set title "z(x, y) = exp(-x²)"** – задание заголовка рисунка.
3. **set tics out** – требуем, чтобы реперы оцифровки были направлены наружу, а не внутрь области расположения графика (по умолчанию действует установка **set tics in**).
4. **set ticscale 3** – переопределение длины репера оцифровки (по умолчанию действует установка **set ticscale 1**).
5. **set ztics 0, 0.25, 1** – задание начального отсчета, шага и конечного отсчета оцифровки по оси аппликат (без этой установки шаг оцифровки аппликаты по умолчанию оказался бы равным **0,1**; в то же время явное указание диапазона оцифровки оси аппликат при вызове команды

splot [-3:3] [0:1] [0:1] exp(-x*x)

привело бы к шагу **0,2**).

6. **set xlabel** **аппликата Z** – желаем видеть на рисунке такое название оси аппликат. У команды **set** есть опции и для пометки названий осей абсцисс и ординат (**xlabel** и **ylabel** соответственно). Эти пометки, однако, в случае трехмерного изображения нельзя повернуть на желаемый угол. Поворот текста допускается при вызове команды **set** с опцией **label**.
7. **set label ...** – задание желаемого текста, места его расположения на рисунке, ориентации и шрифта с размером. Метки иногда удобно именовать (в качестве имени используются целые числа). В рассматриваемом скрипте для текста, помечающего ось абсцисс, выбрана метка с именем **1**, а для текста, помечающего ось ординат, метка с именем **2**. Правда, в данном примере наличие тэга (имени метки) не является необходимостью (подробнее см. **help set label**).
8. **set ticslevel параметр**, который характеризует аппликату плоскости с реперами оцифровки осей абсцисс и ординат. Параметр представляет собой отношение

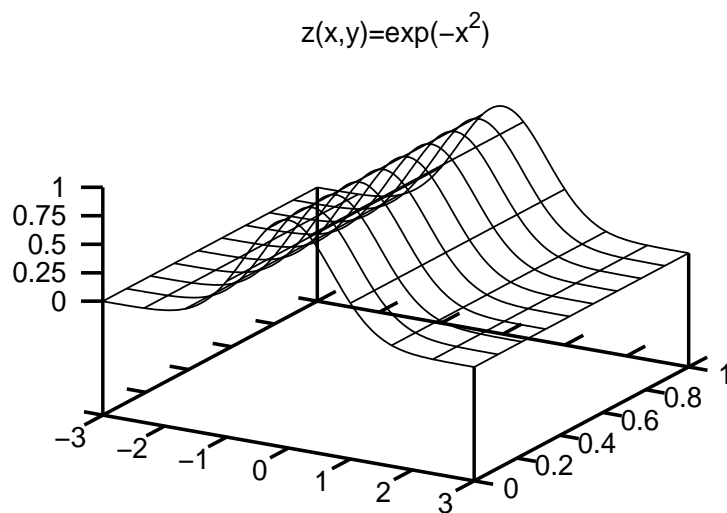
$$\frac{z_{xy} - z_{\min}}{z_{\min} - z_{\max}},$$

где z_{xy} – аппликата плоскости с **XY**- реперами, а z_{\min} и z_{\max} – концевые точки **оцифрованного** отрезка оси аппликат.

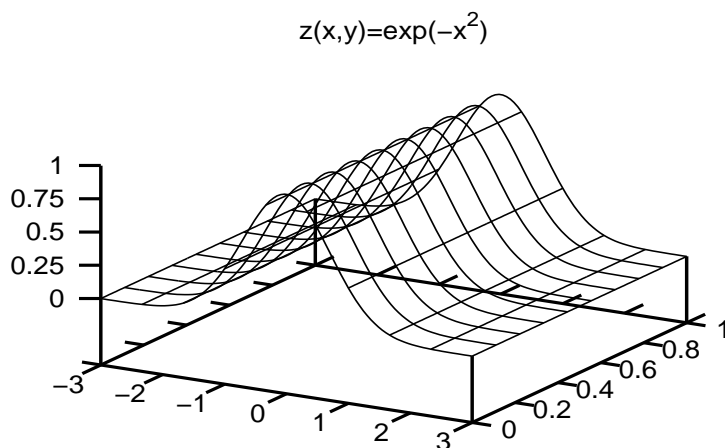
Так что задаваемый через **set ticslevel** параметр – мера расстояния между плоскостями z_{xy} и z_{\min} в долях отрезка $[z_{\min}, z_{\max}]$.

При нулевом параметре плоскости z_{xy} и z_{min} совпадают. При `set ticslevel 1` плоскость z_{xy} ниже плоскости z_{min} на величину оцифрованного отрезка $[z_{min}, z_{max}]$.

```
set term postscript eps enhanced 24
set output 'fig26.eps'
set encoding koi8r; set nokey
set title "z(x,y)=exp(-x^2)" ; set border 31 lw 3
set ztics 0,0.25,1; set tics out
set ticscale 3
set ticslevel 1          # Уровень плоскости XY-реперов
splot [-3:3][0:1]exp(-x*x) lw 3
```

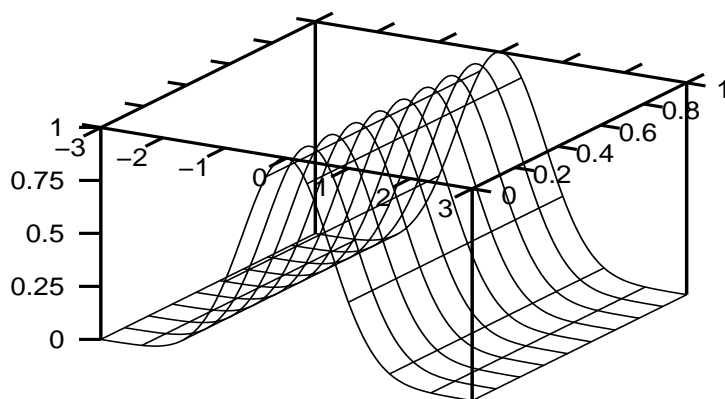


По умолчанию параметр `ticslevel` полагается равным **0.5**, т.е.



При отрицательных значениях опции `ticslevel` плоскость **XY**-оцифровки будет пересекать трехмерное изображение, а при `set ticslevel -1` совпадет с плоскостью $z = z_{\max}$. Например,

```
set term postscript eps enhanced 24
set output 'fig28.eps'; set nokey; set border 31 lw 3
set ztics 0,0.25,1; set tics out; set ticscale 3
set ticslevel -1      # Уровень плоскости XY-реперов
plot [-3:3][0:1]exp(-x*x) lw 3
```



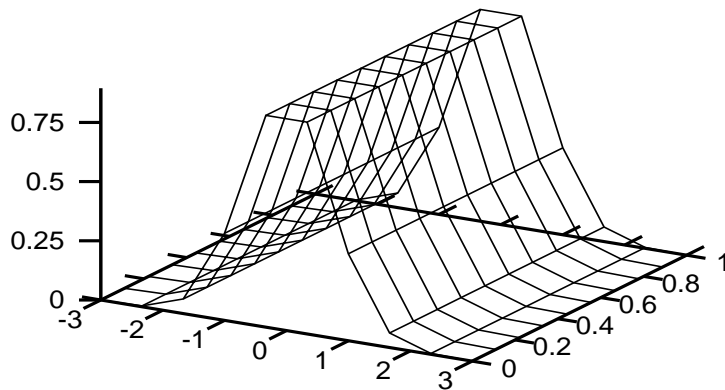
4 О некоторых настройках `splot`.

1. При **формульном** задании функции команды `plot` и `splot` аппроксимируют изображаемый объект ломаной, проходящей через точки кривой, соответствующие равномерному дроблению промежутка базовой независимой переменной. Количество точек дробления по умолчанию принимается равным **100**. Настройка выполняется командой `set samples 100`.

Если, например, явно укажем `set samples 10`:

```
set term postscript eps enhanced 24
set output 'fig29.eps'
set nokey; set border 31 lw 3; set ztics 0,0.25,1;
set tics out; set ticscale 3 ; set ticslevel 0
set samples 10
splot [-3:3][0:1] exp(-x*x) lw 3
```

то получим рисунок:



2. Кривые, которыми чертится изображение объемной фигуры являются изолиниями, образованными её сечением плоскостями постоянства ординаты. По умолчанию их количество равно **10**. Такое же количество и изолиний сечения поверхности плоскостями постоянства абсциссы (прямые линии). Желаемая настройка устанавливаются командой:

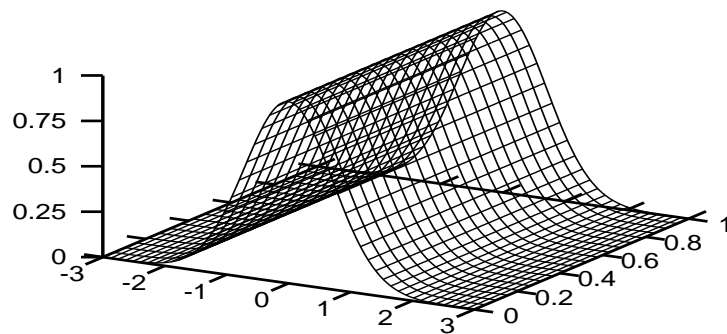
`set isosamples` число-Х-точек, число-У-точек

Например,

```

set term postscript eps enhanced 24
set output 'fig30.eps'
set nokey; set border 31 lw 3; set ztics 0,0.25,1;
set tics out; set ticscale 3 ; set ticslevel 0
set isosamples 50,20
splot [-3:3][0:1] exp(-x*x) lw 3

```



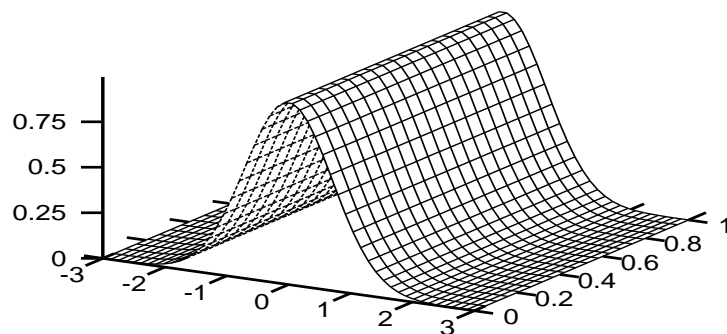
3. На последнем рисунке поверхность выглядит прозрачной. При установке `set hidden3d`

```

set term postscript eps enhanced 24
set output 'fig31.eps'
set nokey; set border 31 lw 3; set ztics 0,0.25,1;
set tics out; set ticscale 3 ; set ticslevel 0
set isosamples 50,20;          set hidden3d
splot [-3:3][0:1] exp(-x*x) lw 3

```

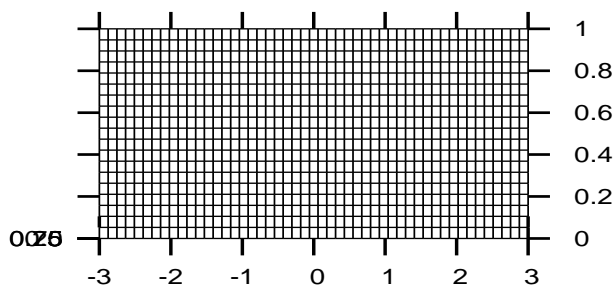
скроются детали, находящиеся вне зоны прямой видимости.



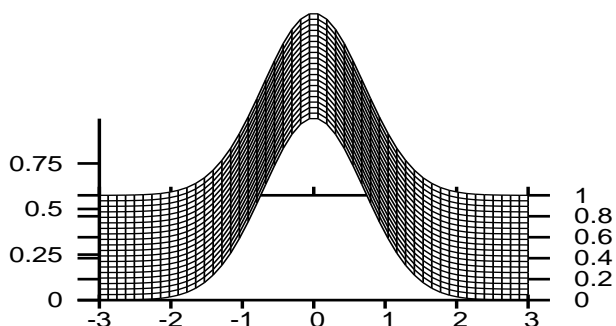
4. Ракурс обзора и масштабы изображения по осям устанавливаются командой `set view 60, 30, 1, 1`. Здесь **60** и **30** углы поворота в градусах вокруг оси абсцисс и оси аппликат соответственно. Считается, что с изображением связана система координат с горизонтальной осью абсцисс, вертикальной осью ординат и осью аппликат перепендикулярной плоскости экрана. Так что, если обнулить углы поворота:

```
set term postscript eps enhanced 24
set output 'fig32.eps'
set nokey; set border 31 lw 3; set ztics 0,0.25,1;
set tics out; set ticscale 3 ; set ticslevel 0
set isosamples 50,20; set hidden3d
set view 0,0
splot [-3:3][0:1] exp(-x*x) lw 3
```

то увидим на нашу поверхность сверху:



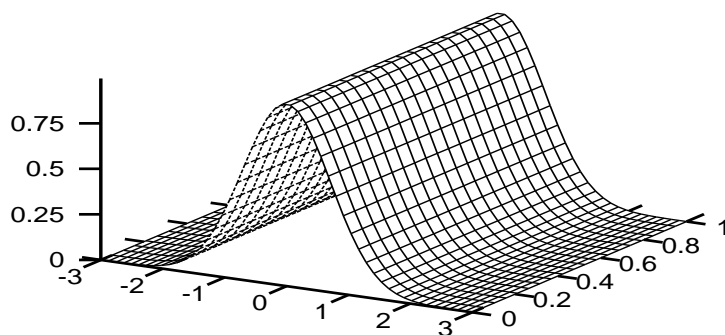
Установка `set view 60, 0` приоткроет изгиб поверхности



поворачивая изображение предыдущего рисунка на 60 градусов вокруг горизонтальной оси абсцисс, лежащей в плоскости экрана так, что станет видимой

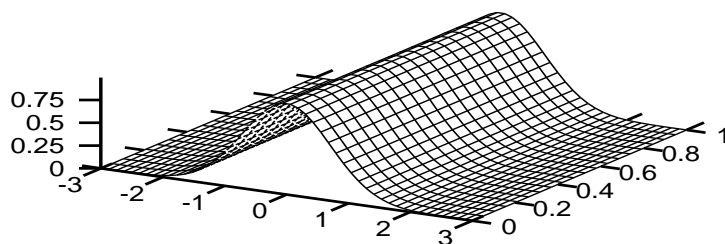
часть плоскости с **XY**-реперами, ранее закрываемая изображаемой поверхностью. Проекция оси аппликат на экран при таком повороте совпадет с сечением плоскости с **XY**-реперами плоскостью $x=-3$.

Установка же **set view 60, 30** повернет изображение фигуры на последнем рисунке вокруг оси аппликат на 30 градусов так, что последняя будет проецироваться на свободное место экрана, а не на левую сторону прямоугольника с **XY**-реперами:



Диапазон углов поворота вокруг оси абсцисс : **[0,180]**; вокруг оси аппликат – **[0,360]**.

5. **Первая** единица в команде **set view 60, 30, 1, 1** задает масштаб рисунка в целом, вторая – исключительно по оси аппликат. Например, установка **set view 60, 30, 0.5** приведёт к рисунку:



5 Заключение

В данном пособии рассмотрены далеко не все возможности, предоставляемые утилитой **gnuplot**. Напомним ещё раз о **help**-системе встроенной в утилиту, которая в любой момент работы позволяет пользователю прояснить забытое или узнать новое; о возможности распечатать подробное руководство из `/usr/share/doc/gnuplot`; о книге Ю.Н. Рыжикова [1], в четырнадцатой главе которой дан, в частности, подробный обзор инструментария **gnuplot** с массой примеров; об использовании утилиты для построения трехмерных изображений; о богатейшей информации по работе с **gnuplot**, имеющейся в Интернете. Тем не менее, для начинающего, вероятно, не бесполезным окажется и вышеизложенное. Составитель пособия искренне признателен В.Б. Титову за полезные обсуждения, ценные советы и консультации.

Список литературы

- [1] Рыжиков Ю.И. 2004. Современный ФОРТРАН: Учебник. – СПб.: КОРОНА принт, –288 с.