Создание иллюстраций в MetaPost. (из серии компьютерные TeXнологии)

© Балдин Е.М.

25 августа 2006 г.

Аннотация

Эта статья была опубликована в 9ом (майском) номере русскоязычного журнала Linux Format (`http://www.linuxformat.ru`) за 2006 год. На сайте `http://www.inp.nsk.su/~baldin/` статья размещена с разрешения редакции журнала и до ноября месяца все вопросы с размещением статьи в других местах следует решать с редакцией Linux Format.

Текст, представленный здесь, не является точной копией статьи в журнале. Текущий текст в отличии от журнального варианта корректор не просматривал. Все вопросы по содержанию, а так же замечания и предложения следует задавать мне по электронной почте `mailto:E.M.Baldin@inp.nsk.su`.

Содержание

4 Графики и диаграммы

Даже жизнь можно отобразить
в виде графика функции от времени.

Отображение данных на бумаге всегда было и останется не тривиальным процессом. Не смотря на то, что получение данных, как правило, занимает гораздо больше ресурсов, задача оформления графиков достойна автоматизации. Автоматизация, в частности позволяет, поняв как можно представить данные наилучшим образом, в дальнейшем не снижать уровень оформления.

По хорошему графики следует создавать в специализированных приложениях. Автоматизация этого процесса достаточно специфична, чтобы реализовывать его с помощью программ не предназначенных только для этого. Если Вам нужны двумерные графики, то никто не справится с этим лучше чем `gnuplot`. Если вас интересуют гистограммы, то это работа для пакетов анализа `paw/cernlib` или `root`. Но, если Вам хочется разукрасить график, то MetaPost будет для этого очень кстати.

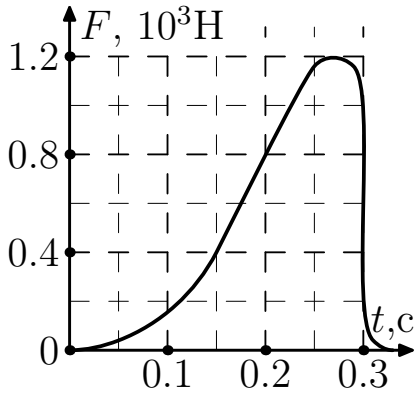

Рис. 23. График, созданный в ручную

Ничего не мешает нарисовать график самостоятельно. Всё что для этого надо:

- знать диапазон функции и её аргумента,
- нарисовать оси координат,
- сделать решётку и поставить метки на осях,

- изобразить функцию.

В этом смысле кодирование графика ничем не отличается от кодирования рисунка — всё что надо уметь это рисовать стрелки и линии. Однако, это годится только для простых графиков.

4.1 Графики в MetaPost

Для серьёзной работы с графиками в MetaPost есть довольно продвинутый пакет `graph.mp`. Этот пакет написан «отцом» MetaPost Джоном Хобби и к нему прилагается подробная документация, которую можно найти в стандартной поставке LaTeX в виде файла `mpgraph.pdf`. Если Вы планируете воспользоваться этим пакетом, то прежде изучите этот текст.

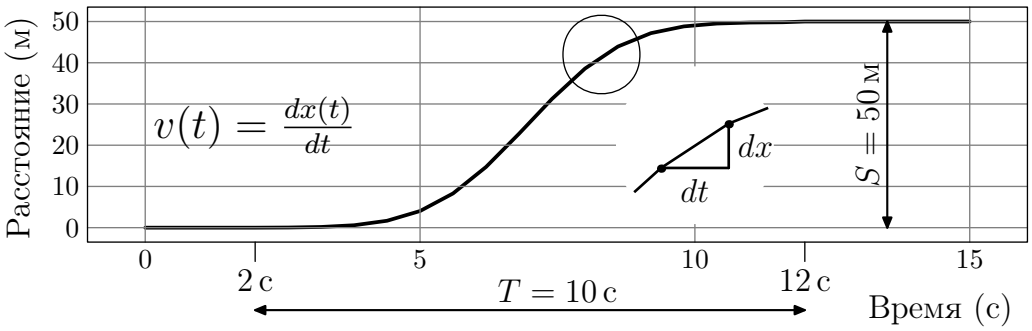

Рис. 24. Иллюстрация для объяснения понятия дифференцирования

```
%Файл graphics.mp
input graph;
%График переменной скорости - тема "дифференцирование"
beginfig(2) ;
  numeric u;
  u = 0.6mm;
  %начало графика
  draw begingraph(12cm,3cm);
    path r;
    numeric e,pi,A,sigma,n,scale,mean,y[],x[];
    e:=2.718;pi:=3.14159;A=50;
```

```
sigma=1.4;n:=100;scale=10;mean=5;
%создание функции - она использовалась не один раз
for j:=0 upto n:
  if (j=0):
    x[0]=0;y[0]=0;
  else:
    x[j]:=scale*j/n;
    y[j]:=y[j-1]+A*(e**(-(((scale*j/n-mean)/
    (1.41*sigma))**2)))*
    (x[j]-x[j-1])/(sqrt(2*pi)*sigma);
  fi
endfor;
%здесь нужна функция из отрезков, поэтому step 6
r:=(0,0)for j:=0 step 6 until (n-1):
   --((2,0)+(x[j],y[j])) endfor ..(12,50)--(15,50);
%отрисовка функции
gdraw r withpen pencircle scaled 0.8u;
%сетка
autogrid(grid.bot,grid.lft) withcolor .5white;
%дополнительные метки на оси абсцисс
otick.bot(btex 2\,\text{c} etex,2);
otick.bot(btex 12\,\text{c} etex,12);
%текст и стрелка за пределами графика
gdrawdblarrow (2,-20)--(12,-20)
                    withpen pencircle scaled 0.5u;
glabel.top(btex \(T=10\,\text{c}\) etex,(7.5,-20));
gdrawdblarrow (13.5,0)--(13.5,50)
                    withpen pencircle scaled 0.5u;
%текстовые метки внутри графика
glabel.lft(btex \(S=50\,\text{м}\) etex
              rotated 90, 1/2[(13.5,0),(13.5,50)]);
glabel.rt(btex \(v(t)=\frac{dx(t)}{dt}\) etex
                                  scaled 1.3,(0,25));
glabel(btex \(dx\) etex,(11.1,20)) ;
glabel(btex \(dt\) etex,(10,9)) ;
%подписи к осям
glabel.rt(btex Время (с) etex,(13,-20)) ;
glabel.lft(btex Расстояние (м) etex
```

```
                    rotated 90,OUT) shifted (0cm,0.5cm);
  endgraph;
endfig ;
```

В `graph.mp` определено специальное окружение. Между `begingraph` и `endgraph` действует своя система координат. Эта система координат привязана ни к геометрическим размерам картинки, а к диапазону осей графика. То есть точка (0,0) соответствует точке пересечения оси абсцисс и оси ординат графика. Размер создаваемого графика указывается сразу после `begingraph`.

Вместо стандартных команд `draw`, `fill`, `label` и `dotlabel` используются `gdraw`, `gfill`, `glabel` и `gdotlabel`, соответственно. Новые команды работают с учётом координат графика. С помощью них можно построить функцию именно по точкам, не заботясь о сдвигах и тому подобное. Для подписи осей существует специальная точка `OUT`, которая в зависимости от суффикса команды `glabel` выносит текст за пределы основной сетки.

Функция отрисовывается точно так же как любая из кривых: создаётся путь и выводится с помощью команды `gdraw`.

Для фиксации диапазона графика используется команда:

```
setrange("нижний левый угол","верхний правый угол");
```

При вызове этой инструкции все последующие функции работают в указанном диапазоне. Таким образом, используя одно и то же пространство можно совместить несколько графиков, имеющих различный диапазон аргументов и функций. Шкалу графика можно сделать логарифмической с помощью инструкции `setcoord`:

```
%x– линейная шкала, y – логарифмическая
setcoord(linear,log);
```

Для создания решётки и меток используется команда `autogrid`

```
%решётка по нижней и левой осям
autogrid(grid.bot,grid.lft) withcolor .5white;
%внешние метки по нижней оси и внутренние метки по правой оси
autogrid(otick.bot,itick.rt);
```

4.2 Работа с файлами данных

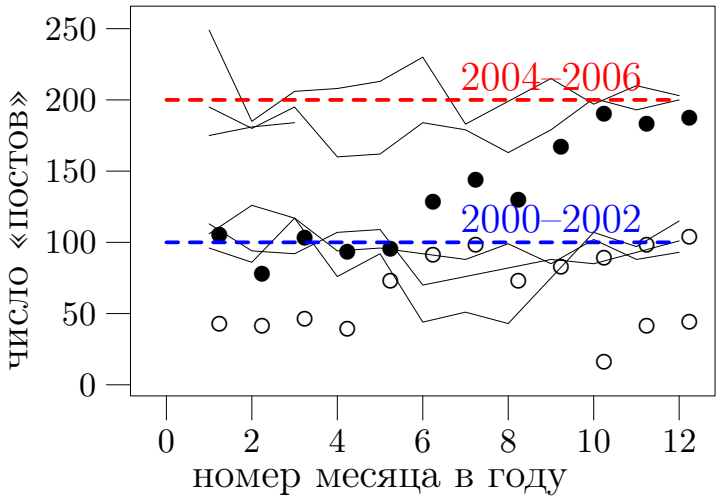

Рис. 25. Число постов на LOR

С помощью команды `gdraw` можно читать данные из файла. Если в качестве аргумента передаётся текстовая строка, то `gdraw` предполагает, что это имя файла.

```
%Файл graphics.mp
%Число постов на LOR от месяца года
beginfig(4);
  u := 0.4mm;
  draw begingraph(150u,100u);
    pickup pencircle scaled 0.2u;
    gdraw("lor-1998.dat") plot btex \(\circ\) etex;
    gdraw("lor-1999.dat") plot btex \(\circ\) etex;
    gdraw("lor-2000.dat");
    gdraw("lor-2001.dat");
    gdraw("lor-2002.dat");
    gdraw("lor-2003.dat") plot btex \(\bullet\) etex;
    gdraw("lor-2004.dat");
    gdraw("lor-2005.dat");
    gdraw("lor-2006.dat");
    gdraw (0,100)--(12,100) withpen pencircle scaled 1u
```

```
    dashed evenly scaled 1u withcolor blue;
    glabel.top(btex 2000--2002 etex,(9,100)) withcolor blue;
    gdraw (0,200)--(12,200) withpen pencircle scaled 1u
    dashed evenly scaled 1u withcolor red;
    glabel.top(btex 2004--2006 etex,(9,200)) withcolor red;
    glabel.lft(btex число <<постов>> etex rotated 90,OUT);
    glabel.bot(btex номер месяца в году etex,OUT);
  endgraph;
endfig;
```

`gdraw` так же как и команда `draw` «понимает» инструкции `withpen` (какое перо использовать), `withcolor` (цвет линии) и `dashed` (использовать пунктир). Дополнительно `gdraw` воспринимает команду `plot {picture}`. Указанная в инструкции `plot` картинка выбирается в качестве маркера. На рисунке чёрными маркерами отмечен 2003 год — год, когда произошло удвоение новостных постов в месяц на LOR (`http://www.linux.org.ru`), а белыми маркерами — первый год существования этого ресурса.

По умолчанию предполагается, что входной файл представляет из себя два столбца цифр, разделённые пробелами. Пример файла `lor-2006.dat`:

```
1 175
2 181
3 184
```

Данные взяты с сайта LOR.

Если данные имеют более сложное представление, чем в упомянутом выше файле, то используется команда `gdata`.

```
gdata("имя_файла",переменная куда считываются данные,действие);
```

Если на текстовый файл вида:

```
1 7.289 H
2 2.425 He
--вырезано--
26 -57.710 Mn
27 -60.604 Fe
```

воздействовать с помощью следующего кода:

```
%Файл graphics.mp
```

```
%дефект масс
beginfig(5) ;
  u := 0.8mm;
  draw begingraph(150u,100u);
    gdraw "mendeleev.dat"  dashed evenly scaled 1u
                                          withcolor 0.5white ;
    gdata("mendeleev.dat", s,
      glabel(s3,(scantokens s1,scantokens s2));
      )
    glabel.lft(btex Дефект массы (МэВ) etex rotated 90,OUT);
    glabel.bot(btex Порядковый номер элемента в таблице etex,OUT);
  endgraph;
endfig;
```

то получится простенькая картинка: Переменная `s` объявляется масси-

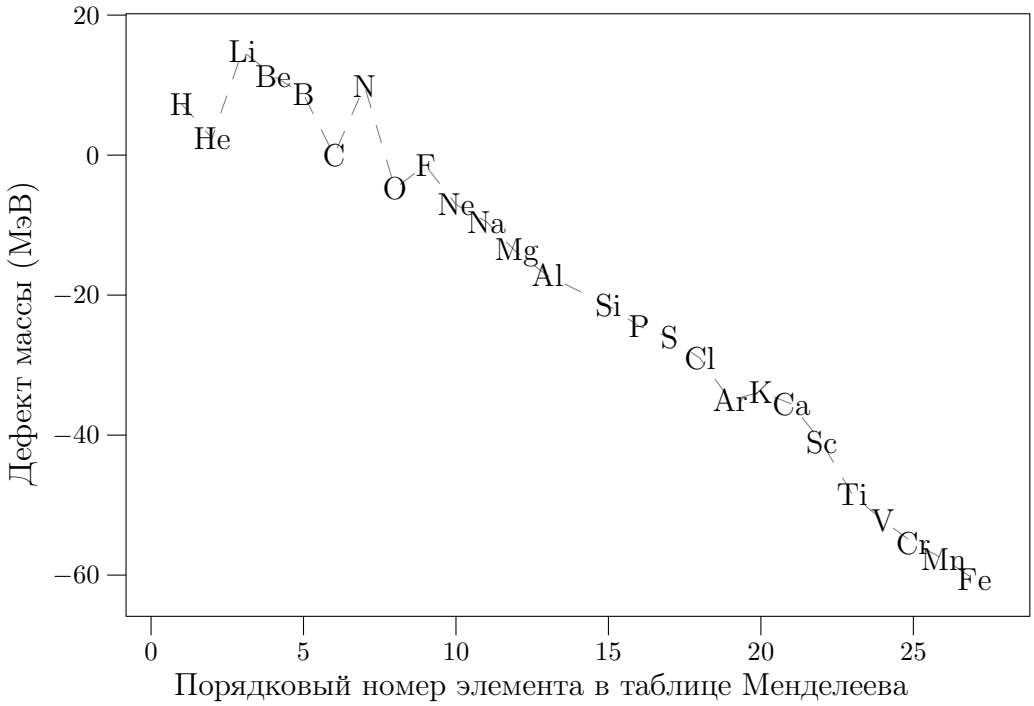

Рис. 26. Дефект массы химических элементов от порядкового номера в периодической таблице Меделеева. Приведены элементы от водорода до железа.

вом в который считывается строка. В качестве разделителя выступает пробел. Запись `s1` соответствует `s[1]` — первому элементу массива. Макрос `scantokens` «встраивает» значение аргумента в код. В данном случае происходит перевод строки в число. В качестве «действия» в команде `gdata` может быть набор из нескольких команд.

4.3 Гистограммы в MetaPost

Если Вы хотите нарисовать гистограмму, то можно воспользоваться следующими макросами:

```
%Файл graphics.mp
%делает профиль гистограммы из пути
def histpath(expr pairs) =
  for i=0 upto length pairs:
    if (i>0):--else:fi((point i of pairs)-
      ((xpart(point 1 of pairs)-
        xpart(point 0 of pairs))/2,0))--
    ((point i of pairs)+
      ((xpart(point 1 of pairs)-
        xpart(point 0 of pairs))/2,0))
  endfor
enddef;

%делает зацикленный профиль гистограммы
%для закрашивания
def histpathcycle(expr pairs) =
  ((xpart(point 0 of pairs),0)-
    ((xpart(point 1 of pairs)-
      xpart(point 0 of pairs))/2,0))
  --histpath(pairs)--
  ((xpart(point infinity of pairs),0)+
    ((xpart(point 1 of pairs)-
      xpart(point 0 of pairs))/2,0))--cycle
enddef;
```

Распределение представленное на картинке похожее на убывающую экспоненту. Вот так рисовались «пятёрки»:

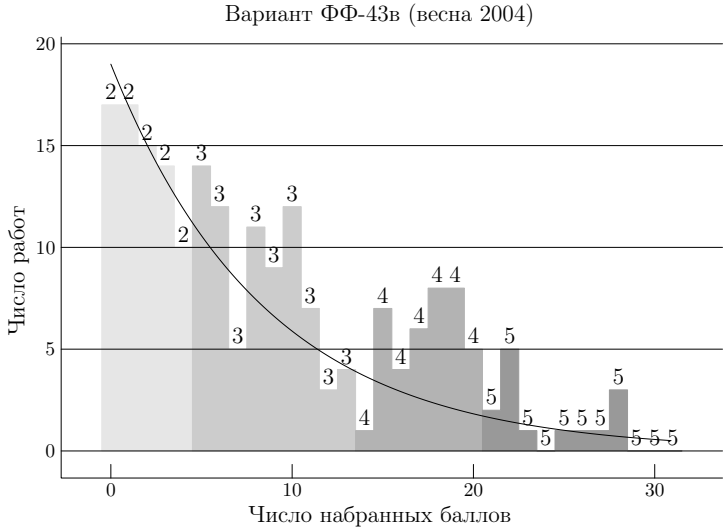

Рис. 27. Распределение по баллам, полученных на Открытой олимпиаде по физике в НГУ в 2004 году.

```
%Файл graphics.mp
%Оценки
path five;
five:=(21,2)--(22,5)--(23,1)--(24,0)--(25,1)--
      (26,1)--(27,1)--(28,3)--(29,0)--(30,0)--(31,0);
%Заполняем гистограмму
gfill histpathcycle(five) withcolor 0.6white;
```

Для закрашивания контура применяется инструкция `gfill` — аналог команды `fill`.

Данные по оценкам получались в результате обработки текстовых файлов со статистикой с помощью простого скрипта на `perl`. Время, которое потребовалось на автоматизацию составило примерно один человеко-день, что многократно уступает времени написания полного отчёта куда вошли подобные гистограммы и оформлению всех условий и решений Открытой олимпиады и последующих за ней летних вступительных экзаменов.

Создание своих макросов на MetaPost является стандартным действием. `META` не страдает избыточностью, поощряя «затачивать» окружения под свои нужды.

4.4 Круговые диаграммы

Пакет `piechartmp.mp` относительно «молодой» пакет. Замечательная, можно сказать красочная, документация поставляется в виде файла `piechartmp.pdf` и примеров.

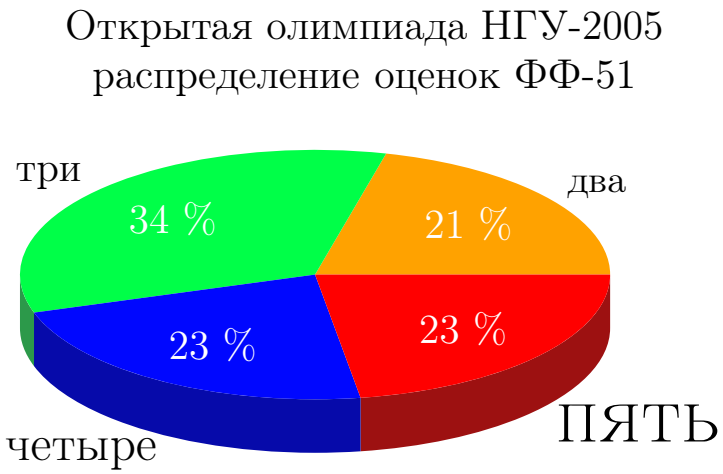

Рис. 28. Распределение оценок

Как обычно, за русским языком надо приглядывать. По умолчанию пакет работает только с латиницей.

```
%Файл pie.mp
input piechartmp;
%Круговые гистограммы
%Открытая Олимпиада "ФФ–51"
beginfig(1) ;
  numeric u; u:=1mm;
  %чтобы был русский
  SetupText(1, "\input{preheader–base}", "\begin{document}")
  label(btex Открытая олимпиада НГУ–2005 etex,(5u,25u));
  label(btex распределение оценок ФФ–51 etex,(5u,20u));
  %Чтобы отображался процент
  SetupPercent(this, "␣\%");
  %Опеределение сегментов
  Segment( 15,"\small␣два", auto) ;
  Segment( 24,"три", auto) ;
```

```
  Segment( 16,"\large четыре", auto) ;
  Segment( 16,"\huge пять", red ) ;
  %Создание круговой гистограммы
  PieChart(30u, 0.2, 65, 0, 0);
  %метки
  Label(0)(percent)(inwards,0) withcolor white;
  Label.auto(0)(name)(outwards,0) ;
endfig;
```

Алгоритм создания круговой или секторной диаграммы следующий:

- Если Вы хотите воспользоваться автоматической системой размещения меток, то с помощью команды `SetupText` следует настроить ввод текста. В файле `preheader-base.tex` должна быть минимальная шапка для документа LaTeX. Если Вы хотите использовать русский язык, то там обязательно должны быть строки вида `\usepackage[T2A]{fontenc}` и `\usepackage[koi8-r]{inputenc}`. Вместо `koi8-r` можно поставить свою кодировку.

- С помощью команды `Segment` определить сегменты. В качестве аргументов команде передаётся число (измеряемая величина), текстовая метка и цвет сегмента (можно указать значение по умолчанию `auto`).

  Можно также ввести четвёртый необязательный строковый параметр, который заменяет измеряемую величину при создании меток. Это решает проблему ограничения на диапазон чисел в MetaPost.

- Нарисовать гистограмму. В качестве параметров команде `PieChart` передаётся размер, высота диаграммы, угол под которым мы на неё смотрим (трёхмерия), угол поворота вокруг центральной оси и «смещение» сегментов относительно центра.

- Расставить метки.

Каждому сегменту при создании присваивается порядковый номер начиная с единицы. Все изменения в круговой диаграмме делаются глобально. Пакет написан так, чтобы было удобно работать в одном `mp`-файле ровно с одной диаграммой. Если при описании диаграммы внутри окружения `beginfig` что-то уже определили, то нет необходимости в последующих окружениях это повторять. Этот режим удобен если нужно

создать несколько модификаций одной диаграммы, например, для создания «оверлеев» в презентации. Минусом такого подхода является то, что если в этом же mp-файле хочется создать ещё одну гистограмму, то уже определённые сегменты необходимо спрятать.

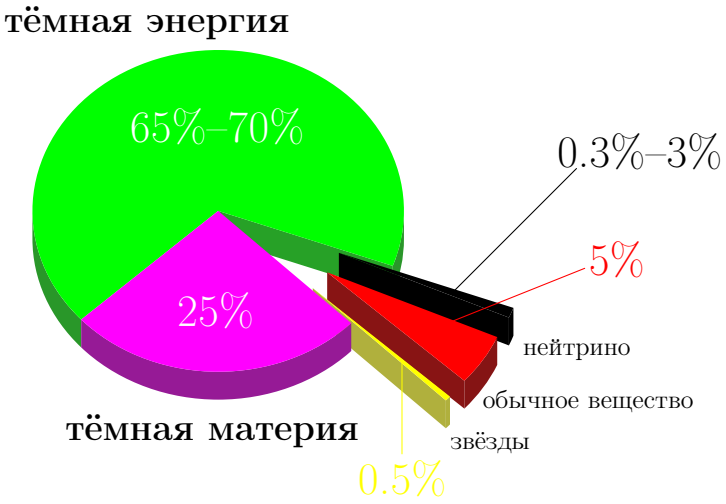

Рис. 29. Распределение «вещества» в современной вселенной.

```
%Круговые гистограммы
%распределение вещества во вселенной
beginfig(2) ;
  numeric u; u:=1mm;
  %прячем определённые ранее сегменты
  SegmentState(1,hidden,this);
  SegmentState(2,hidden,this);
  SegmentState(3,hidden,this);
  SegmentState(4,hidden,this);
  %Опеределение новых сегментов
  Segment( 65,"\Large\bf тёмная энергия",green,"\LARGE 65\%--70\%");
  Segment( 25,"\Large\bf тёмная материя", (1,0,1),"\LARGE 25\%");
  Segment( 0.5,"звёзды", (1,1,0),"\LARGE 0.5\%");
  Segment( 5,"обычное вещество", red,"\LARGE 5\%");
  Segment( 1,"нейтрино", black,"\LARGE 0.3\%--3\%");
  %Выдвигаем сегменты из диаграммы
  SegmentState(7,this,0.7);
```

```
  SegmentState(9,this,0.7);
  SegmentState(8,this,0.7);
  %Создание круговой гистограммы
  PieChart(30u, 0.3, 30, 340, 0);
  %метки
  Label.auto(0)(name)(outwards,0) ;
  Label.auto(5)(value)(inwards,0) shifted (7u,-7u) withcolor white;
  Label.auto(6)(value)(inwards,0) shifted (0u,5u) withcolor white;
  Label.auto(7)(value)(inwards,(0u,-15u)) withcolor (1,1,0);
  Label.auto(8)(value)(inwards,(25u,10u)) withcolor red;
  Label.auto(9)(value)(inwards,(20u,20u)) withcolor black;
endfig;
```

Состояние каждого из сегментов можно задавать с помощь команды `SegmentState`.

```
SegmentState(Порядковый номер сегмента,
    устанавливаемое состояние сегмента или this если оно не меняется,
    радиальный сдвиг сегмента или this если он не меняется);
```

Состояние сегмента может быть:

- `normal` — сегмент становится видимым,
- `invisible` — сегмент не рисуется при создании круговой диаграммы (в диаграмме остаётся пустое место),
- `hidden` — при создании диаграммы этот сегмент игнорируется.

Радиальный сдвиг сегмента указывается в процентах, где 1 — это 100%, то есть сегмент полностью «выдвинут».

Для установки меток используется команда `Label`:

```
Label(Порядковый номер сегмента)(метка)
     (базовая точка в системе отсчёта сегмента, сдвиг);
```

`Label` понимает те-же суффиксы, что и обычная команда `label`, плюс она понимает суффикс `auto`. При установки суффикса `auto` `Label` пытается сама угадать где лучше поставить метку.

В качестве порядкового номера сегмента можно передать 0. В этом случае команда `Label` применяется ко всем видимым сегментам. `Label`

может принимать в качестве аргумента список порядковых номеров сегментов, разделённых запятыми (например: `5,7,9`).

Метка может быть просто текстовой строкой или одним из стандартных значений:

- `value` — измеряемая величина, которая приписывается сегменту при его создании,
- `percent` — процент занимаемой площади,
- `name` — имя сегмента, которое приписывается ему при его создании.

Базовая точка представляется в виде пары чисел `(x,y)`, где `x` — расстояние от вершины сегмента (0 — это вершина, 1 — это противоположный от вершины край), а y — эквивалент полярному углу (0 — край сегмента по часовой стрелки, 1 — край сегмента против часовой стрелки). В пакете определены константы `inwards=(0.7,0.7)` и `outwards=(1.1,0.5)`. Сдвиг же представлен в терминах всего графика (например, `(1cm,0)` — означает сдвиг на 1 сантиметр вправо).

P.S. При желании тип заливки и порядок цветов по умолчанию можно определять самостоятельно. Подробней об этом рассказано в документации к пакету (`piechartmp.pdf`).

4.5 LaTeX рисует с помощью MetaPost

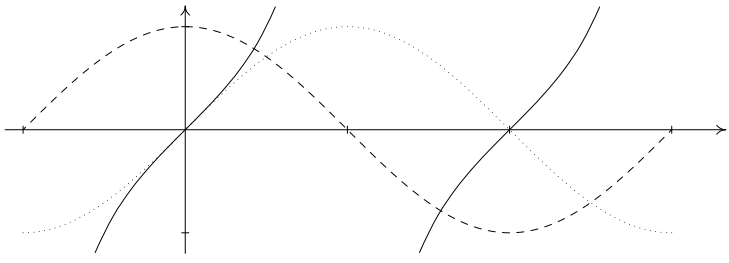

Рис. 30. График косинуса, синуса и тангенса.

Для рисования графиков можно воспользоваться и возможностями самого LaTeX. Например, с этой обязанностью прекрасно справляется стандартный пакет *mfpic*. Для «отрисовки» функций с помощью этого пакета достаточно задать саму функцию, а не рисовать по точкам:

```
%файл graphics-mfpic.tex
%Пример от Сергея В. Знаменского
\documentclass{article}
\usepackage[koi8-r]{inputenc}
\usepackage[russian]{babel}
\usepackage[metapost]{mfpic}
%указываем mp-файл куда будет «складываться» код mfpic
\opengraphsfile{graph-mfp}
\begin{document}
%не печатать номер страницы
\pagestyle{empty}
%начинаем создание графика
\mfpic[1][57.2]{-100}{300}{-1.2}{1.2}
%оси координат
\axes
\xmarks{-90,90,180,270}
\ymarks{-1,1}
%рисуем синус точками
\dotted\function{-90,270,4.5}{sind(x)}
%рисуем косинус пунктиром
\dashed\function{-90,270,4.5}{cosd(x)}
%рисуем тангенс сплошной линией
\function{-50,50,10}{tand(x)}
\function{130,230,10}{tand(x)}
%заканчиваем создание графика
\endmfpic
%закрываем mp-файл
\closegraphsfile
\end{document}
```

Для получения картинки необходимо выполнить следующие действия:

```
> latex graphics-mfpic.tex
> mpost graph-mfp.mp
> latex graphics-mfpic.tex
> dvips -E graphics-mfpic.dvi -o graphics-mfpic.eps
```

При компиляции файла `graphics-mfpic.tex` создаётся `graph-mfp.mp`, куда пишутся команды на языке `META`. Если посмотреть на код `graph-mfp.mp`,

то можно увидеть что там используется макрос `function`, который позволяет рисовать функции одной командой без всяких циклов. То, как это делается, можно подглядеть в библиотечке `grafbase.mp`, которая входит в состав пакета *mfpic*.

Более подробно про пакет `mfpic` можно узнать из документации которую можно найти в директории `$(TEXMF)/texmf-dist/doc/generic/mfpic/`, где `$(TEXMF)` — директория в которую установлен LaTeX (это заведомо верно для дистрибутива TeXLive).

4.6 gnuplot

В случае отображения больших объёмов данных лучше использовать `gnuplot`. Для того чтобы `gnuplot` работал с MetaPost, необходимо определить правильное устройство вывода, то есть `gnuplot` следует передать команду:

```
set terminal mp color latex
```

Рис. 31. Интегральная светимость от времени. Данные за полтора года (примерно 130 тысяч точек) обработаны с помощью программы `gnuplot`.

В этом случае `gnuplot` будет выводить графики в формате MetaPost. К сожалению, похоже, что вывод в MetaPost давно не поддерживался, поэтому установка кодировки:

```
set encoding koi8r
```

не оказывает должного эффекта. Получившийся mp-файл приходится «дотачивать» в ручную. Благо это не сложно (следует поправить только заголовок), но для автоматизации следует озадачиться исправлением этой неприятной ошибки в `gnuplot`.

В случае если будет желание изменить метки, то следует обратить внимание на переменную `textmag`, которая отвечает за размер текста.

4.7 Подведём итоги

С помощью MetaPost можно создавать двумерные графики и диаграммы любой сложности. Возможности языка `META` и стандартные пакеты позволяют сконцентрироваться на смысловой части и не отвлекаться на отдельные элементы оформления.