Matplotlib

Основные возможности

Ф.Я.Халили

МГУ, физический факультет

14 апреля 2009 г.

1. Просто график
2. Несколько графиков
3. Подписи, заголовки, сетка
4. Интеграция с TEX'ом
5. Специальные графики
6. Двумерные массивы
7. 3D

Подключение matplotlib

Пакет matplotlib обеспечивает matlab-подобный набор команд для построения графиков. Подключается он следующей инструкцией:

```
from pylab import *
```

Отметим, что при этом загружается также пакет numpy.

Создание графика

Для создания простого графика достаточно следующих команд:

```
>>> from pylab import *
>>> y=sin(linspace(0,10,100))
>>> plot(y)
[<matplotlib.lines.Line2D instance at 0x8899e0c>]
```

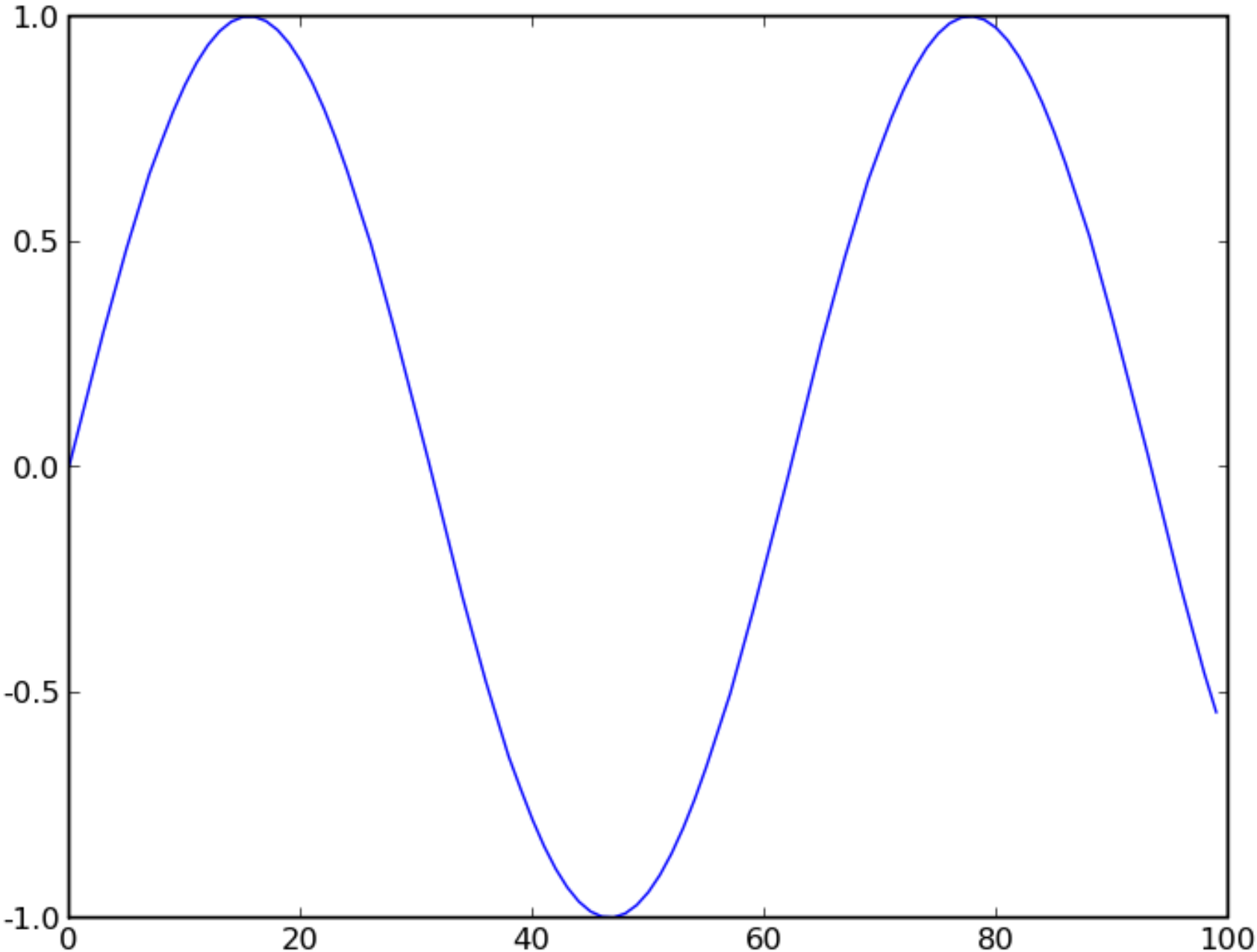
График, однако, именно *создается*, о чем говорит ответ системы (в квадратных скобках), но никуда не выводится. Чтобы увидеть его, необходимо добавить команду `show()`, что приведет к выводу на экран окна с графиком и несколькими кнопками управления, позволяющими, в частности, сохранить график в файле.

Сохранение графика

Сохранить график можно также командой

```
savefig(<имя файла>)
```

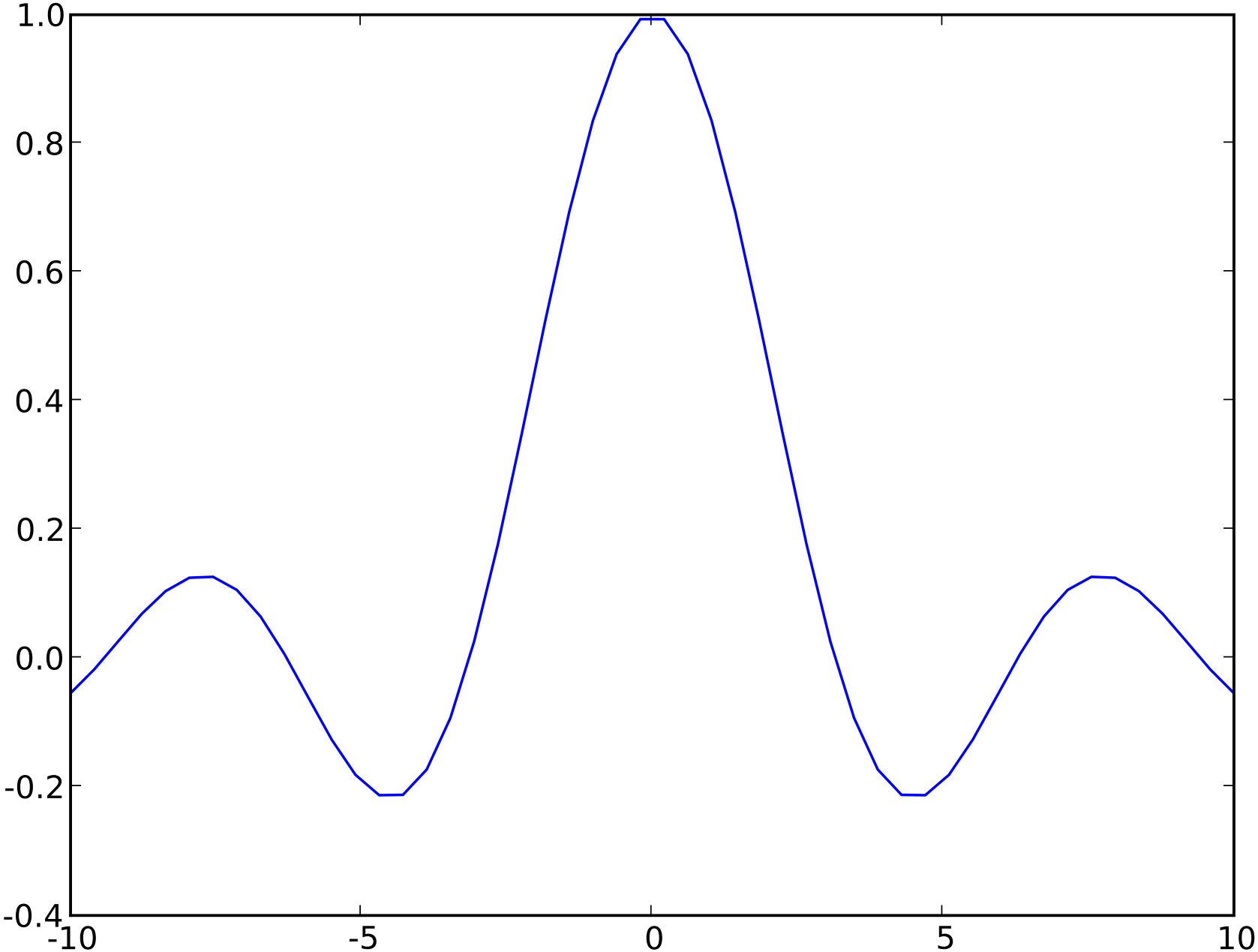
Расширение имени файла задает его формат, возможные варианты: eps, jpeg, pdf, png (по умолчанию), ps, svg. Результат показан на следующем слайде.


1.0
0.5
0.0
-0.5
-1.0
0
20
40
60
80
100

Независимая переменная

В качестве независимой переменной в этом примере использовался просто индекс массива, $0 : 100$. Однако можно задать ее и явно:

```
from pylab import *
x=linspace(-10,10,50)
y=sin(x)/x
plot(x,y)
savefig('plot2.pdf')
```


1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-10
-5
0
5
10

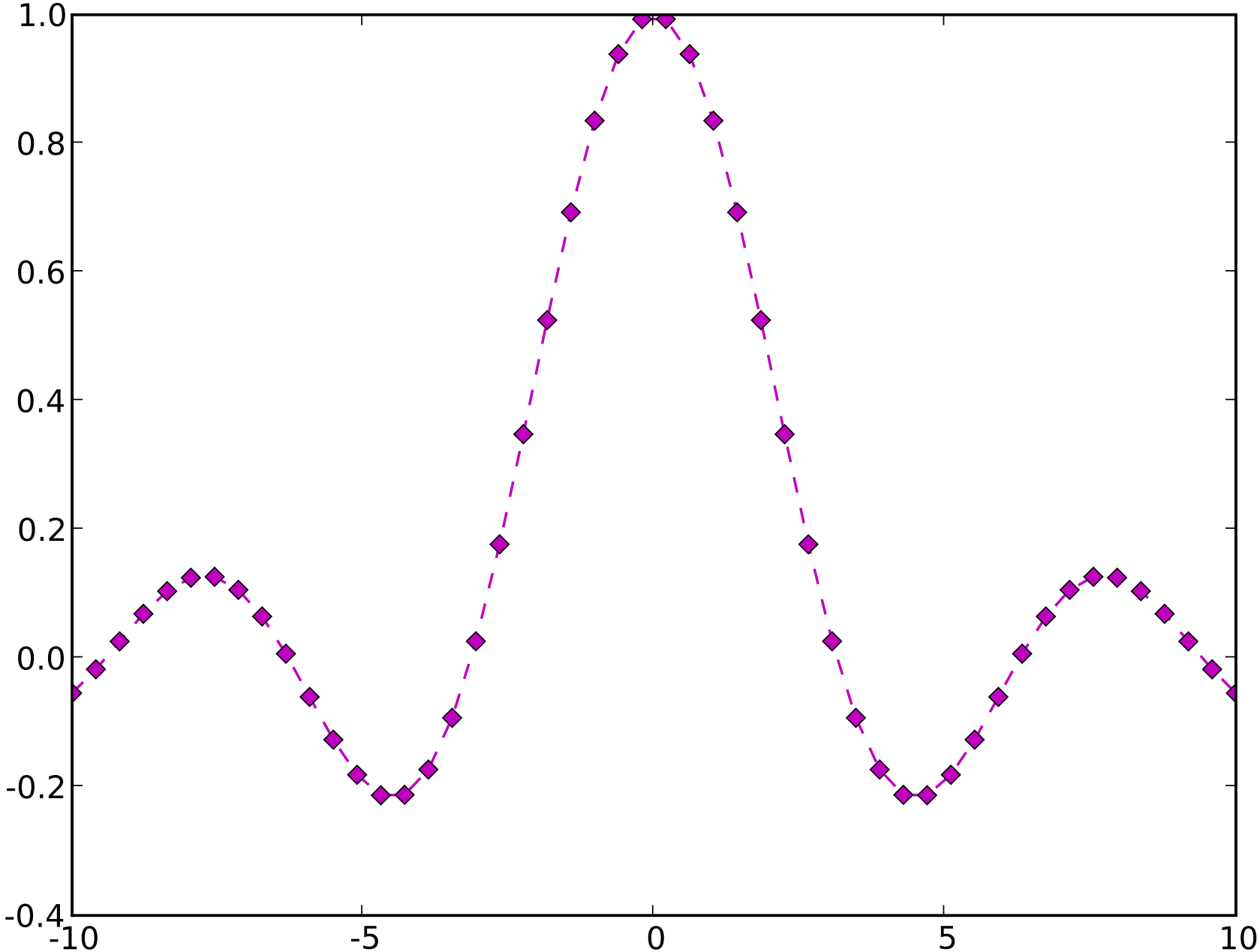
Задание вида графика, короткая форма

Вид и цвет графика можно задавать в старом матлабовском стиле единым параметром — строкой, которая может содержать указание:

- цвета (одна из букв rgbymcwk)
- вида линии (- -- -. :)
- маркера (один из символов .,oû<>s+xDd1234hHp|_)

Например:

```
from pylab import *
x=linspace(-10,10,50)
y=sin(x)/x
plot(x,y,'mD--')
savefig('plot2a.pdf')
```


1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-10
-5
0
5
10

Задание вида графика, подробная форма

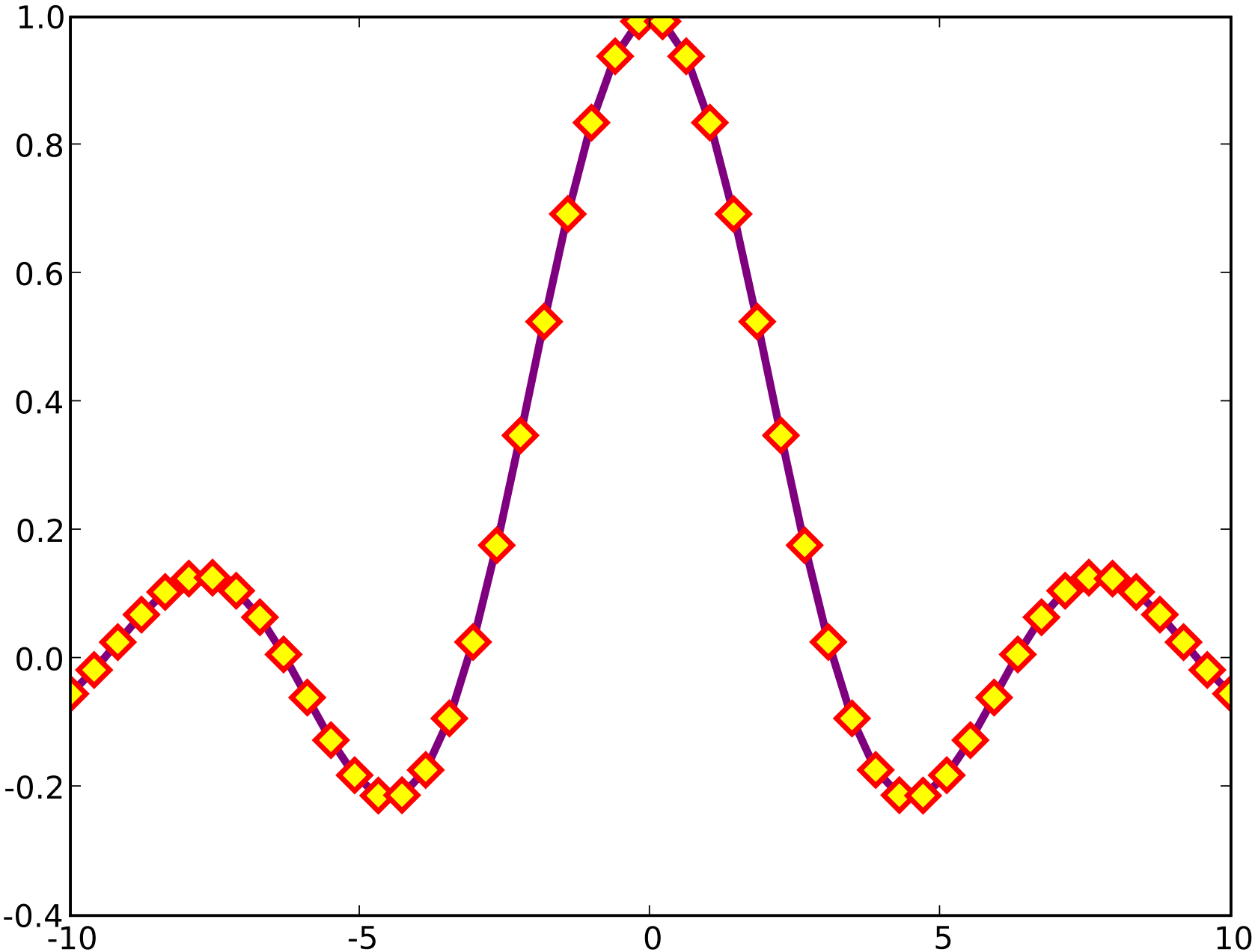
Более детальное управление обеспечивается специальными аргументами

- c=<цвет линии>
- ls='<тип линии>' (те же - -- -. :)
- lw=<толщина линии>
- marker='<тип маркера>' (те же .,ov^<>s+xDd1234hHp|_)
- mec=<цвет границы маркера>
- mfc=<цвет маркера>
- mew=<толщина границы маркера>
- ms=<размер маркера>

Задание вида графика, подробная форма

Например:

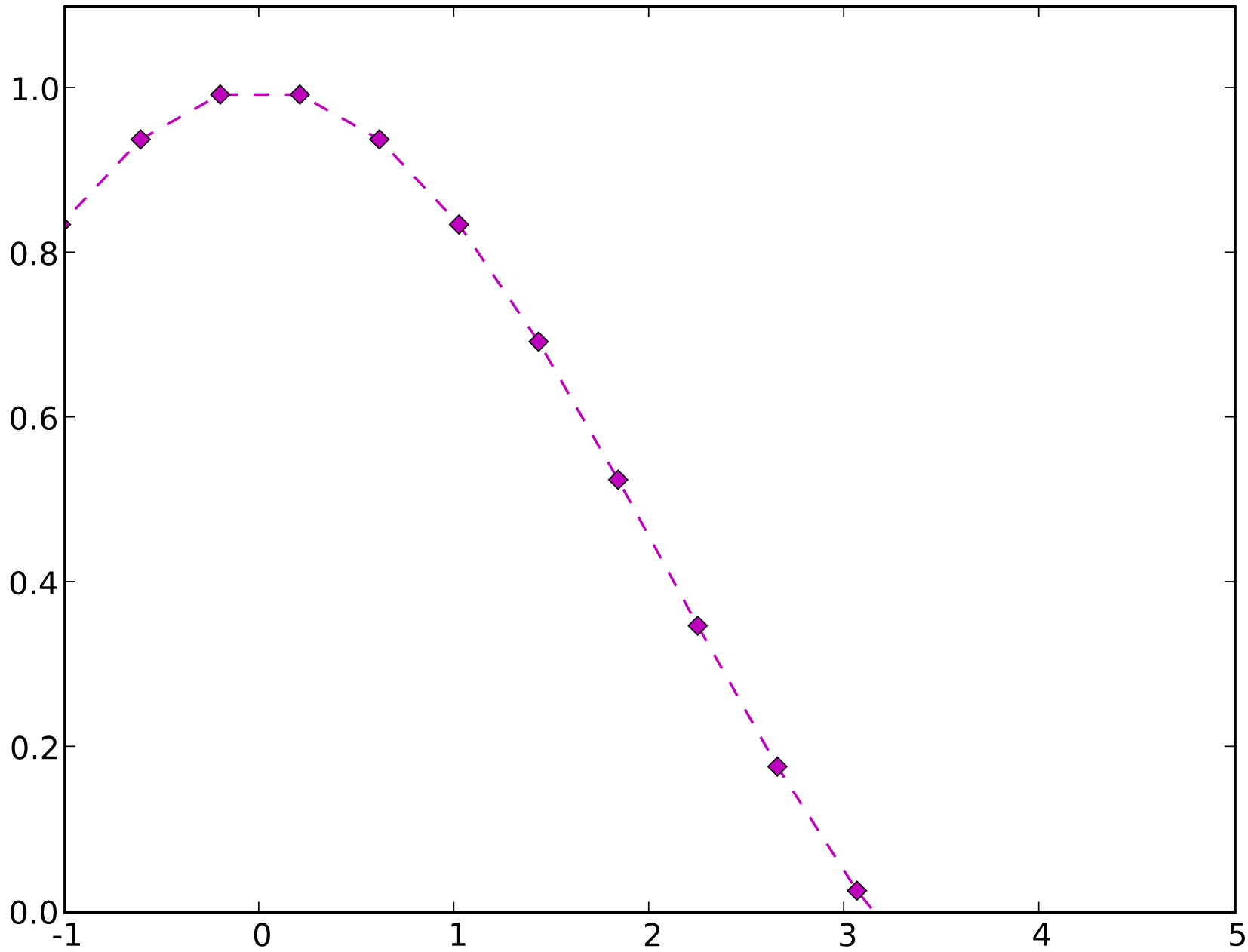
```
from pylab import *
x=linspace(-10,10,50)
y=sin(x)/x
plot(x,y,c='#7F007F',lw=3,marker='D',mec='#FF0000',\
  mfc='#FFFF00',mew=2,ms=10)
savefig('plot2b.pdf')
```


1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-10
-5
0
5
10

Масштаб по осям

Команды
`xlim(xmin,xmax)` и `ylim(ymin,ymax)`
задают масштаб по осям x и y:

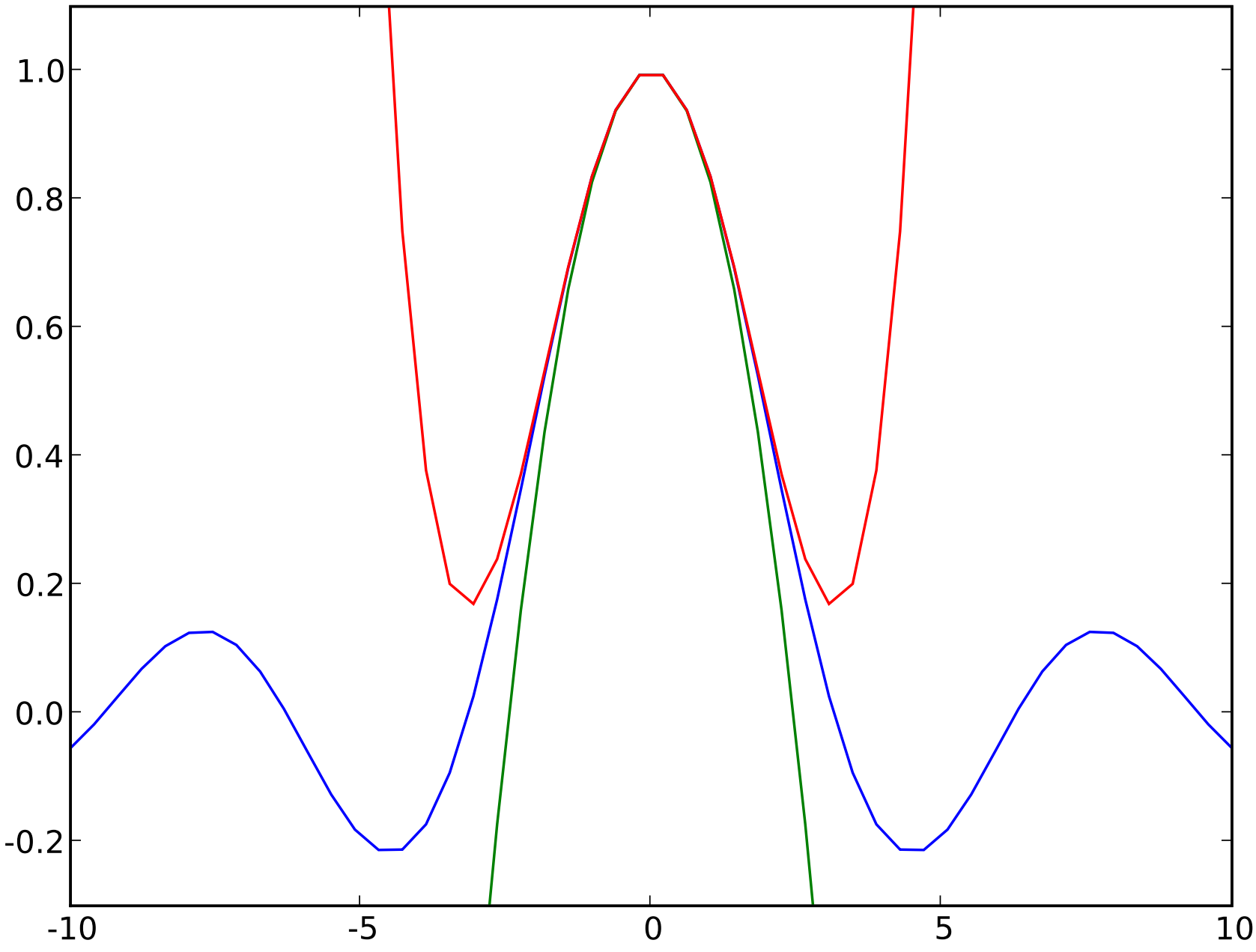
```
from pylab import *
x=linspace(-10,10,50)
y=sin(x)/x
plot(x,y,'mD--')
xlim(-1,5)
ylim(0,1.1)
savefig('plot2c.pdf')
```


1.0
0.8
0.6
0.4
0.2
0.0
-1
0
1
2
3
4
5

Несколько кривых на одном графике

Каждая команда `plot` добавляет свою кривую:

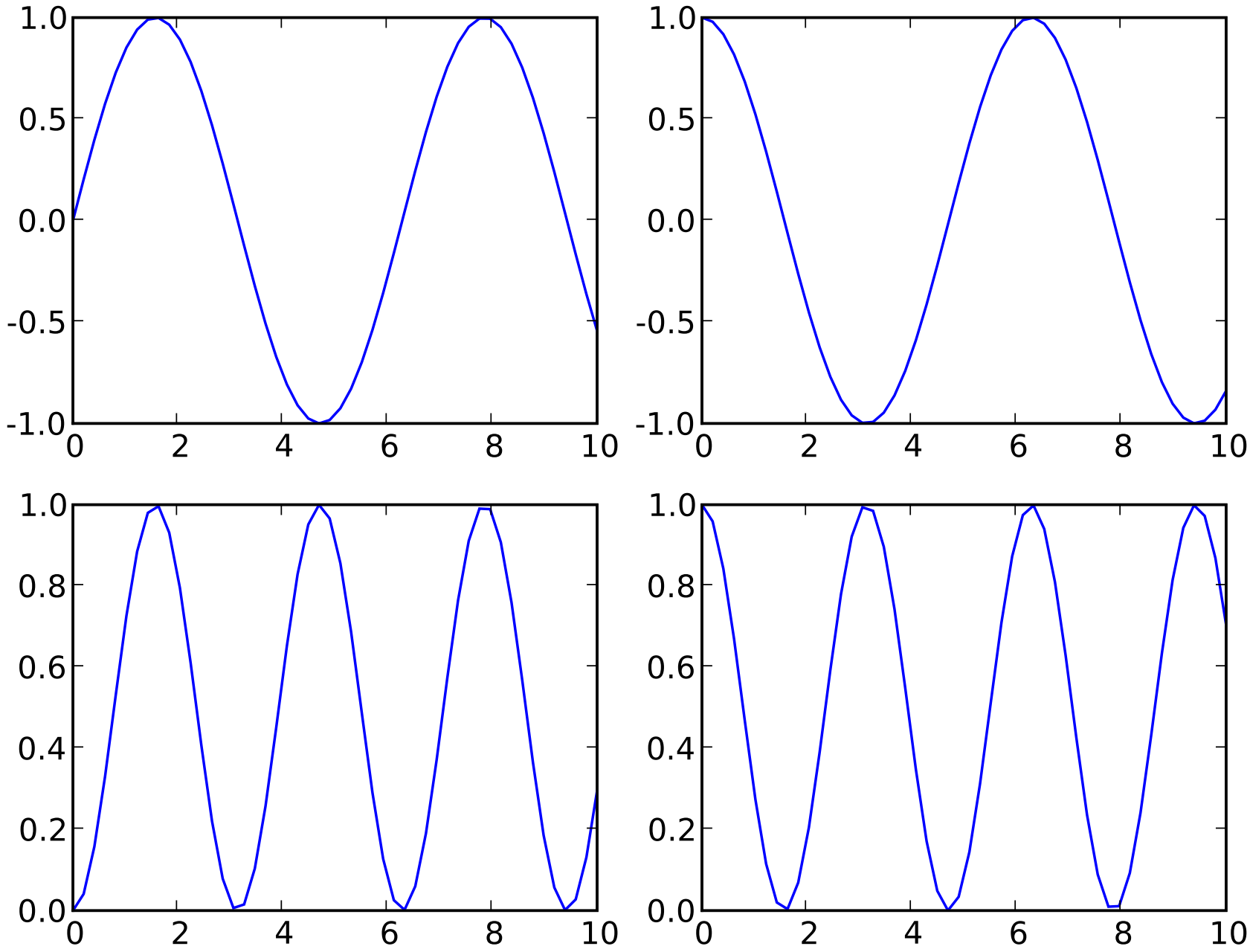
```
from pylab import *
x=linspace(-10,10,50)
plot(x,sin(x)/x)
plot(x,1-x**2/6)
plot(x,1-x**2/6+x**4/120)
ylim(-0.3,1.1)
savefig('plot3.pdf')
```


1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-10
-5
0
5
10

Несколько графиков в одной картинке

Команда `subplot` безо всяких изменений позаимствована из matlab'а:

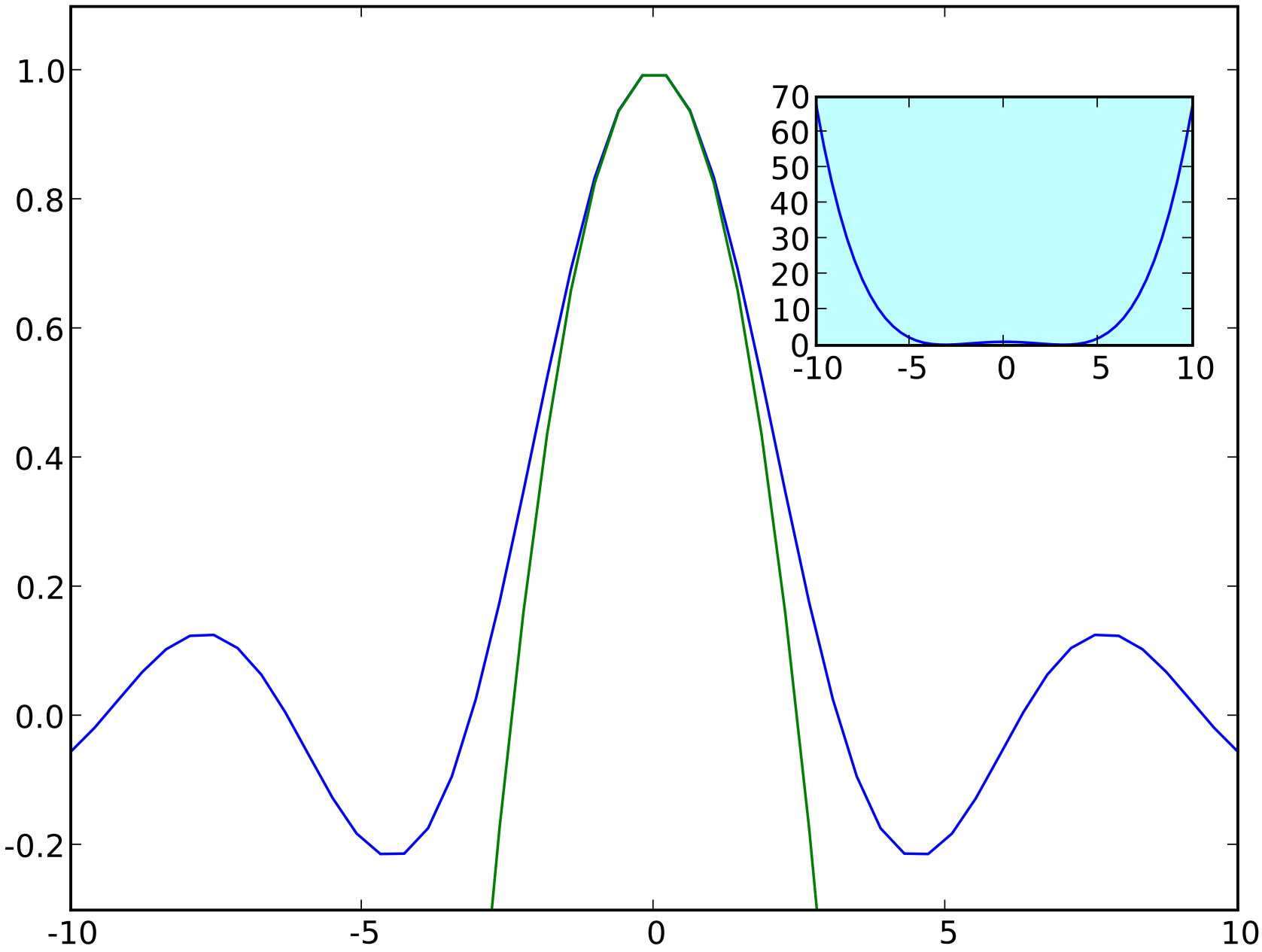
```
from pylab import *
x=linspace(0,10,50)
subplot(221)
plot(x,sin(x))
subplot(222)
plot(x,cos(x))
subplot(223)
plot(x,sin(x)**2)
subplot(224)
plot(x,cos(x)**2)
savefig('plot3a.pdf')
```


1.0
0.5
0.0
-0.5
-1.0
0
2
4
6
8
10
1.0
0.5
0.0
-0.5
-1.0
0
2
4
6
8
10
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10

Вложенные графики

Команда
`axes(<координаты>,axisbg=<цвет фона>)`
(второй аргумент необязателен) создает вложенный график:

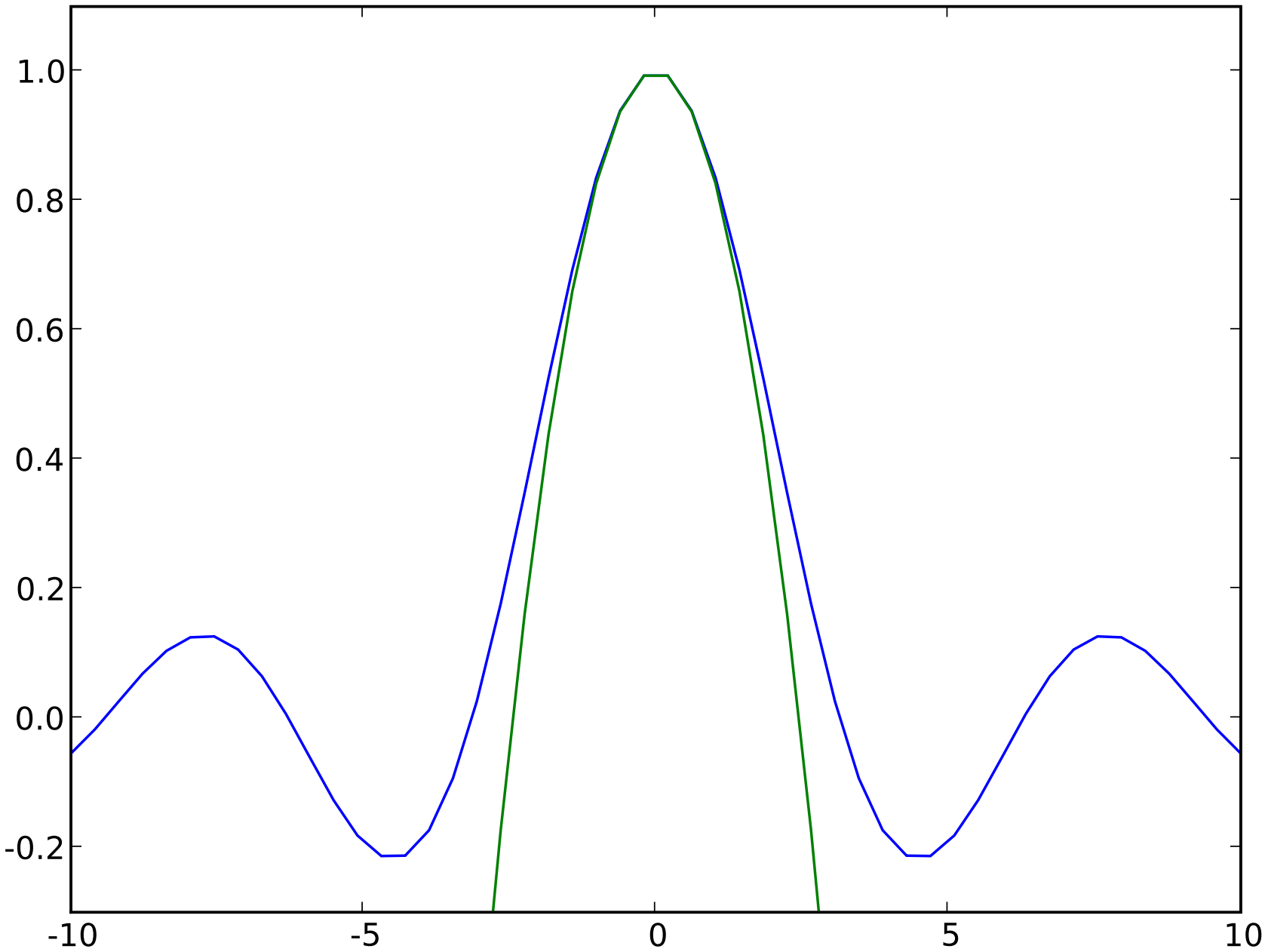
```
from pylab import *
x=linspace(-10,10,50)
plot(x,sin(x)/x)
plot(x,1-x**2/6)
ylim(-0.3,1.1)
sp=axes([0.62,0.6,0.25,0.22],axisbg='#BFFFFF')
plot(x,1-x**2/6+x**4/120)
savefig('plot3b.pdf')
```

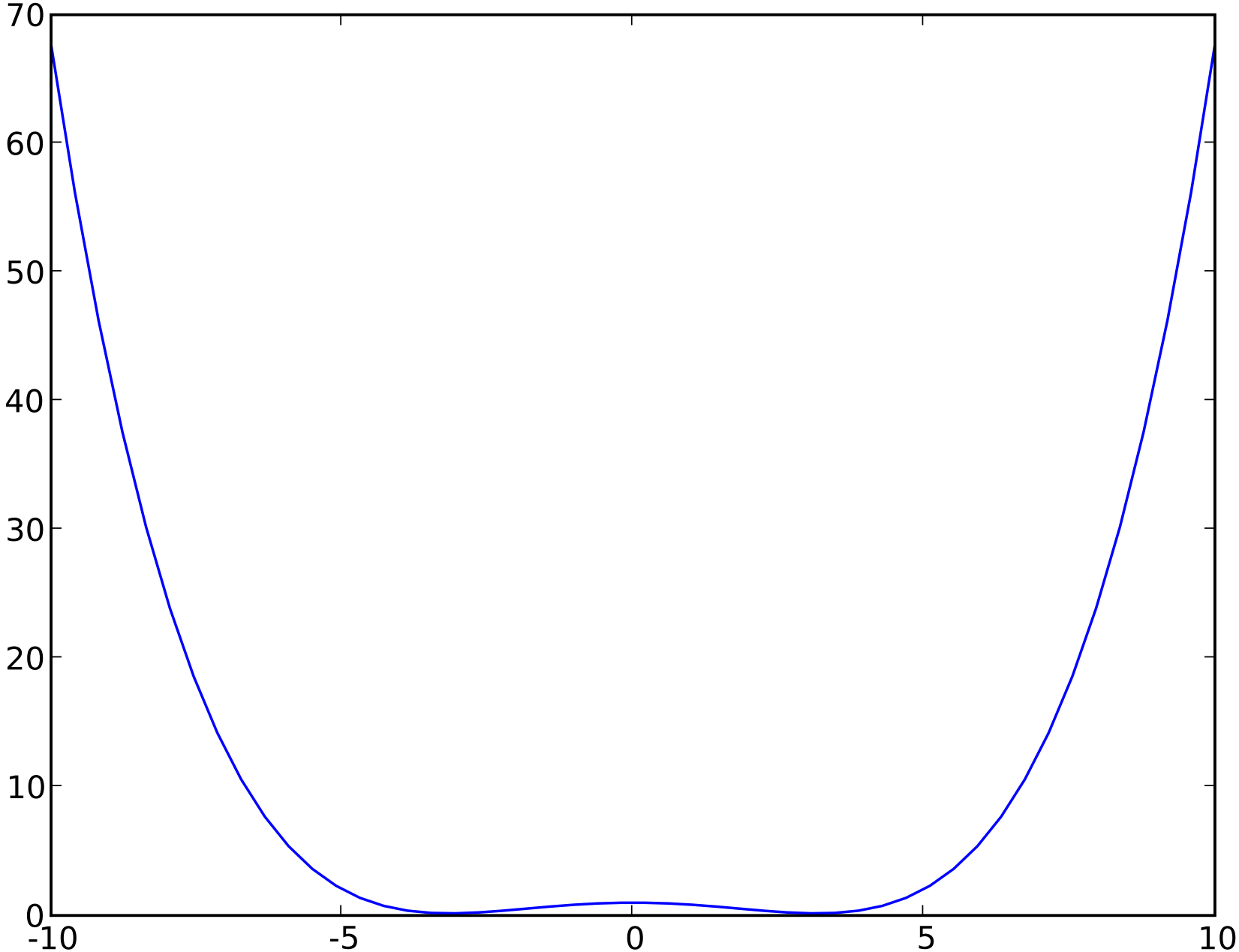

1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-10
-5
0
5
10
70
60
50
40
30
20
10
0
-10
-5
0
5
10

Несколько картинок в одном скрипте

Как правило, все графики к данной статье или другому документу удобно генерировать одним скриптом. При этом для перехода от одного графика к другому используется команда `figure()`:

```
from pylab import *
x=linspace(-10,10,50)
plot(x,sin(x)/x)
plot(x,1-x**2/6)
ylim(-0.3,1.1)
savefig('plot4a.pdf')
figure()
plot(x,1-x**2/6+x**4/120)
savefig('plot4b.pdf')
```

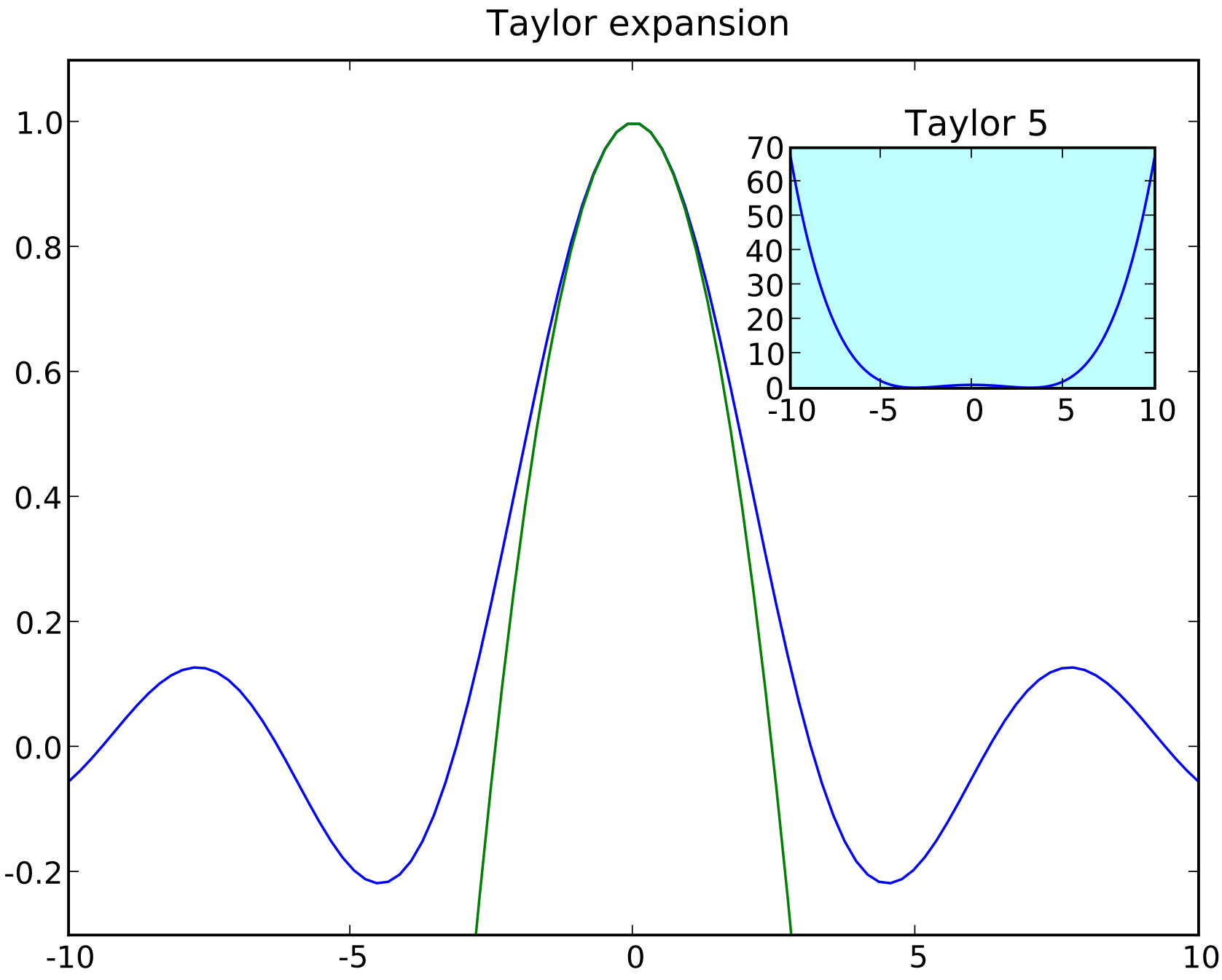

1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-10
-5
0
5
10


70
60
50
40
30
20
10
0
-10
-5
0
5
10

Заголовок

Команда `title` создает общие заголовки к графику и подграфикам (если таковые есть):

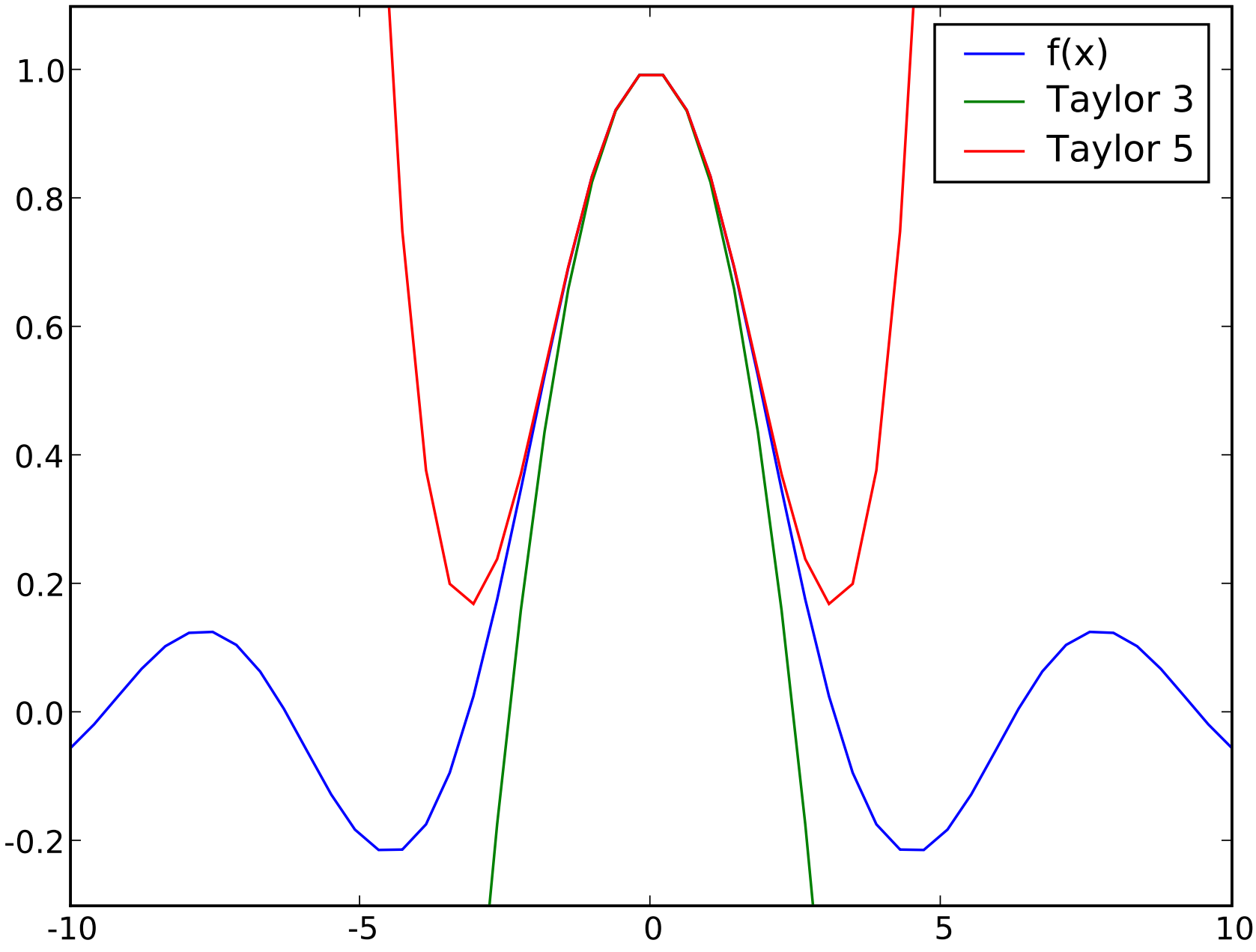
```
from pylab import *
x=linspace(-10,10,50)
plot(x,sin(x)/x)
plot(x,1-x**2/6)
ylim(-0.3,1.1)
title('Taylor expansion')
sp=axes([0.62,0.6,0.25,0.22],axisbg='#BFFFFF')
plot(x,1-x**2/6+x**4/120)
title('Taylor 5')
savefig('plot5.pdf')
```


Taylor expansion
Taylor 5
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-10
-5
0
5
10
70
60
50
40
30
20
10
0
-10
-5
0
5
10

Легенда

Команда `legend` создает описания к графикам. Ее второй (необязательный) аргумент может быть цифрой от 0 до 9, кодирующей расположение легенды как в matlab'e.

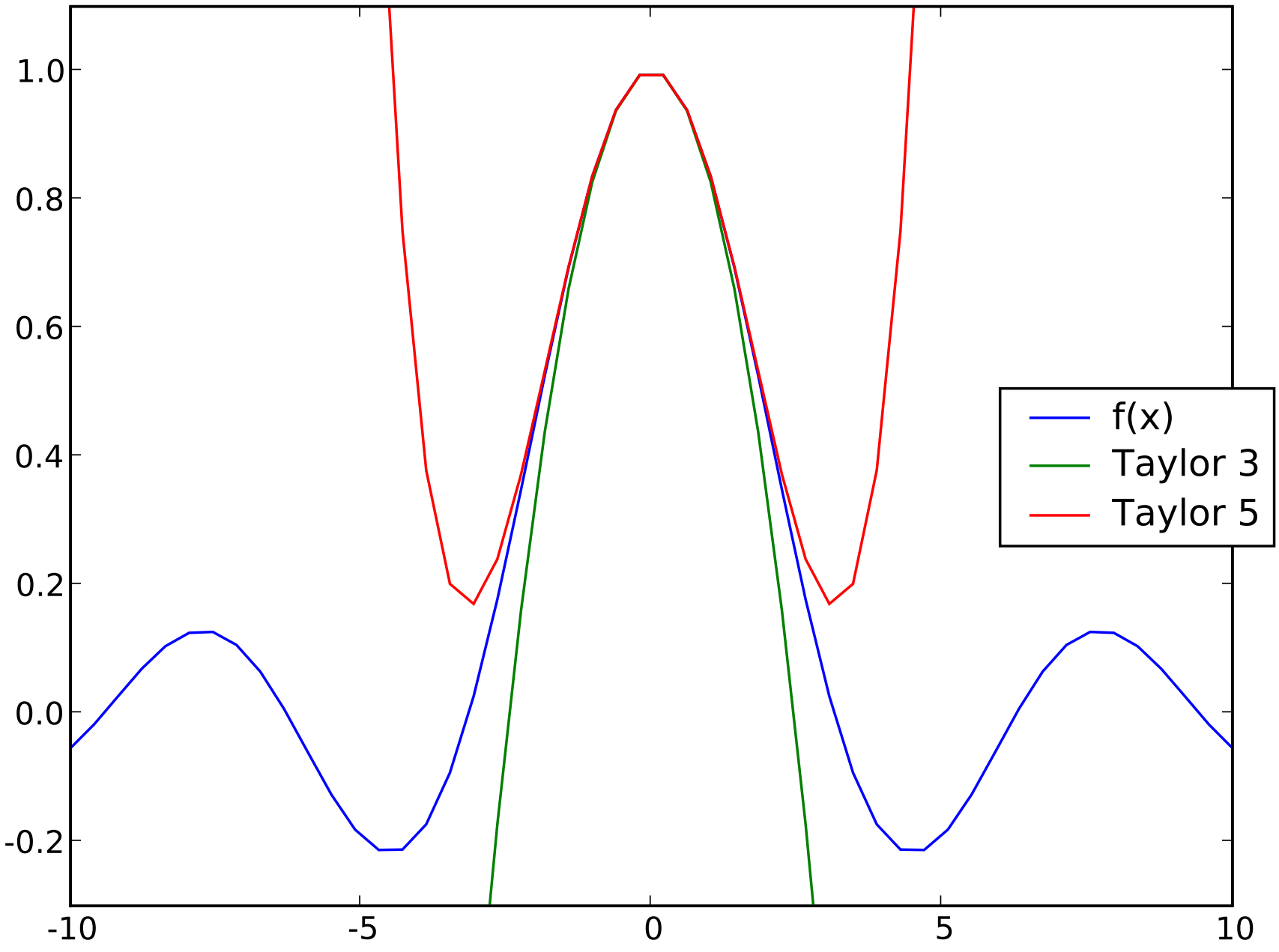
```
from pylab import *
x=linspace(-10,10,50)
plot(x,sin(x)/x)
plot(x,1-x**2/6)
plot(x,1-x**2/6+x**4/120)
ylim(-0.3,1.1)
legend(('f(x)','Taylor 3','Taylor 5'),1)
savefig('plot5a.pdf')
```


f(x)
Taylor 3
Taylor 5
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-10
-5
0
5
10

Легенда

Расположение легенды можно также явно задать парой числе от 0 до 1 (в этом случае необходимо явно указать ключевое слово loc):

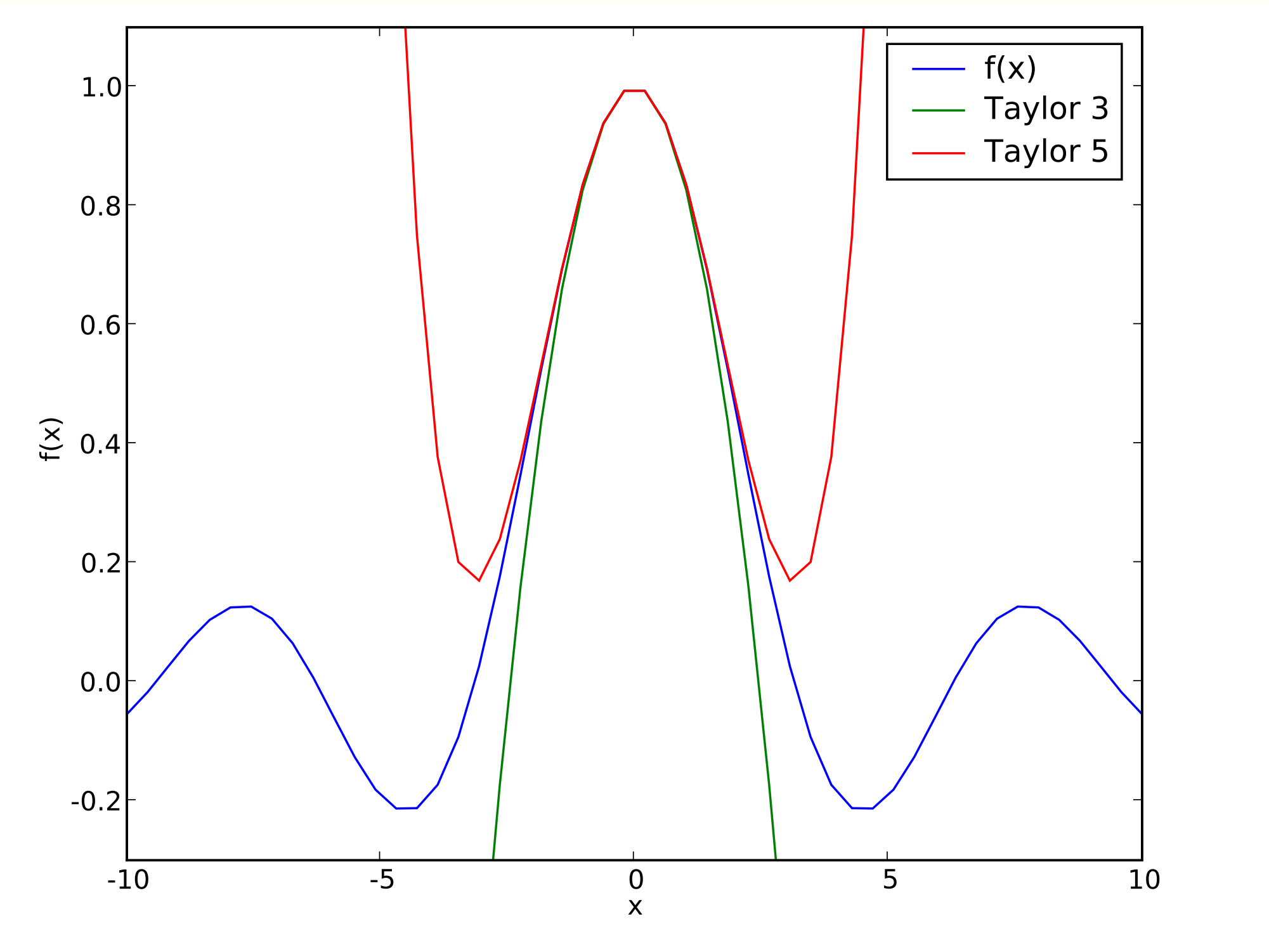
```
from pylab import *
x=linspace(-10,10,50)
plot(x,sin(x)/x,label='1')
plot(x,1-x**2/6,label='2')
plot(x,1-x**2/6+x**4/120,label='3')
ylim(-0.3,1.1)
legend(('f(x)','Taylor 3','Taylor 5'),loc=(0.8,0.4))
savefig('plot5b.pdf')
```


1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-10
-5
0
5
10
f(x)
Taylor 3
Taylor 5

Оси

Команды `xlabel` и `ylabel` создают подписи к осям:

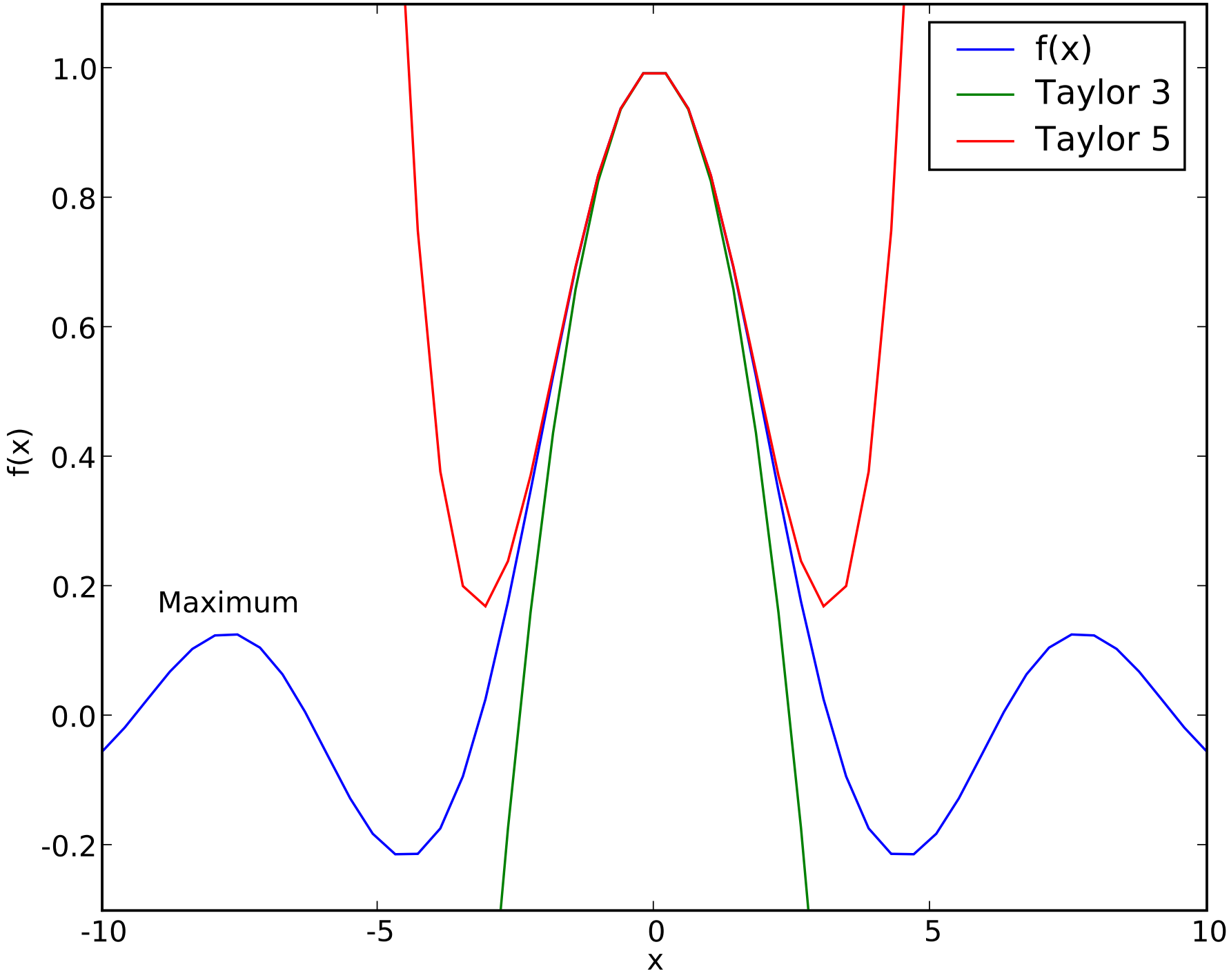
```
from pylab import *
x=linspace(-10,10,50)
plot(x,sin(x)/x)
plot(x,1-x**2/6)
plot(x,1-x**2/6+x**4/120)
ylim(-0.3,1.1)
legend(('f(x)','Taylor 3','Taylor 5'),1)
xlabel('x')
ylabel('f(x)')
savefig('plot6.pdf')
```


f(x)
Taylor 3
Taylor 5
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
f(x)
-10
-5
0
5
10
x

Оси

Команда `text(x,y,'<текст>',<размер шрифта>=12)`, в соответствии с нзванием, выводит текст в заданной точке:

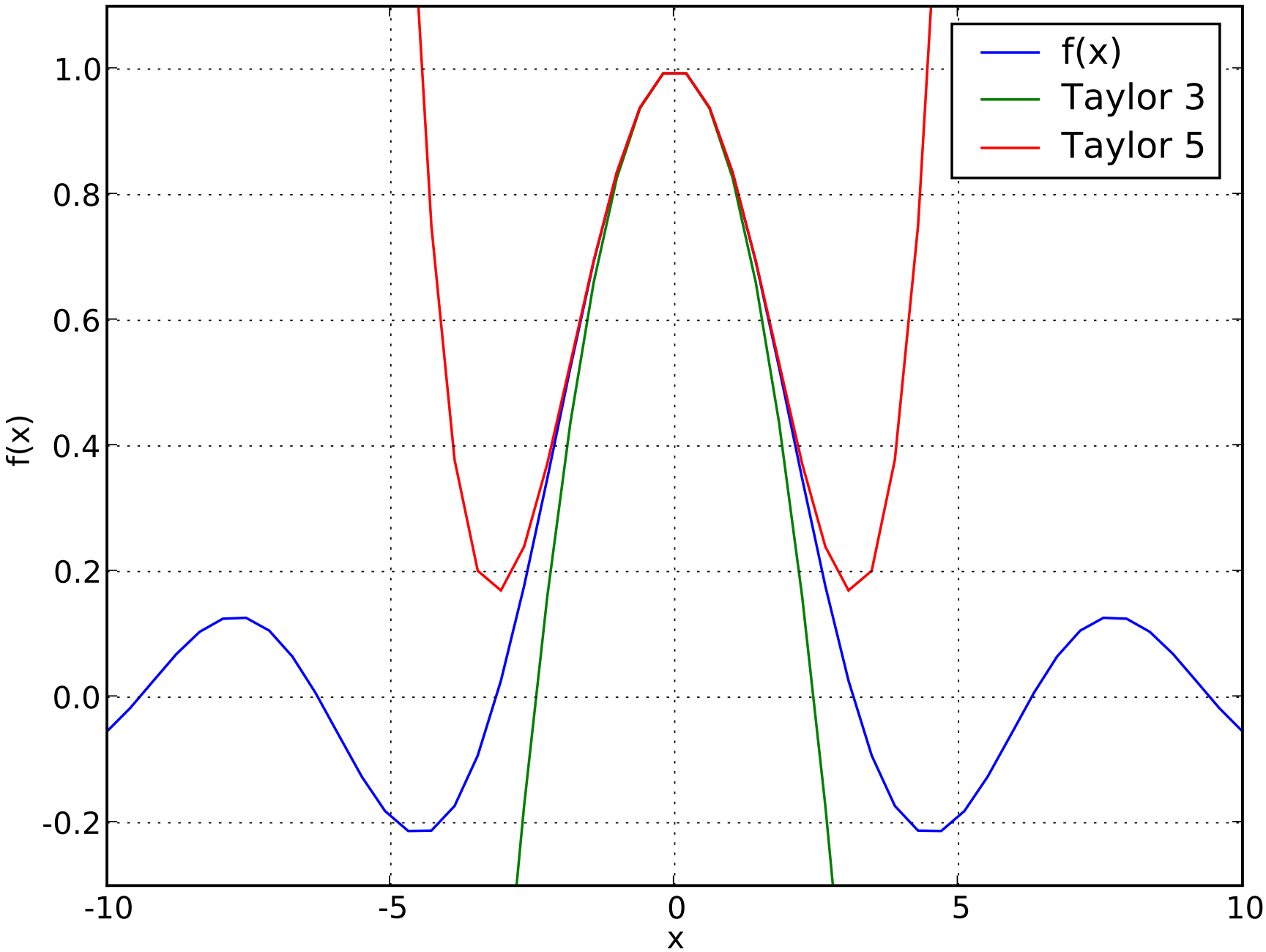
```
from pylab import *
x=linspace(-10,10,50)
plot(x,sin(x)/x)
plot(x,1-x**2/6)
plot(x,1-x**2/6+x**4/120)
ylim(-0.3,1.1)
legend(('f(x)','Taylor 3','Taylor 5'),1)
xlabel('x')
ylabel('f(x)')
text(-9,0.16,'Maximum')
savefig('plot6a.pdf')
```


f(x)
Taylor 3
Taylor 5
Maximum
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
f(x)
-10
-5
0
5
10
x

Сетка

Команда `grid` рисует сетку:

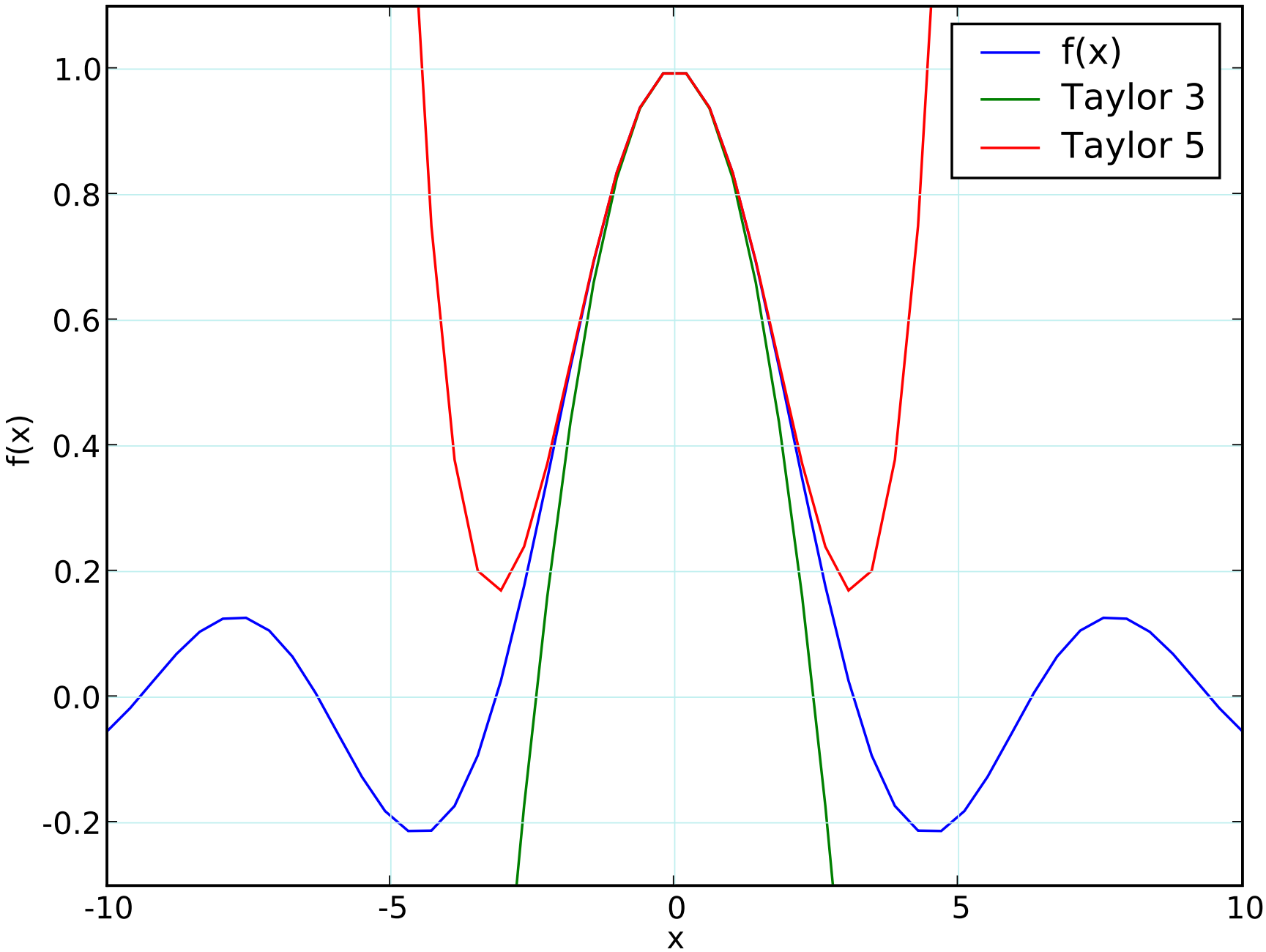
```
from pylab import *
x=linspace(-10,10,50)
plot(x,sin(x)/x)
plot(x,1-x**2/6)
plot(x,1-x**2/6+x**4/120)
ylim(-0.3,1.1)
legend(('f(x)','Taylor 3','Taylor 5'),1)
xlabel('x')
ylabel('f(x)')
grid()
savefig('plot7.pdf')
```


f(x)
Taylor 3
Taylor 5
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
f(x)
-10
-5
0
5
10
x

Сетка

Как и при рисовании самого графика, можно указывать цвет и вид линий сетки:

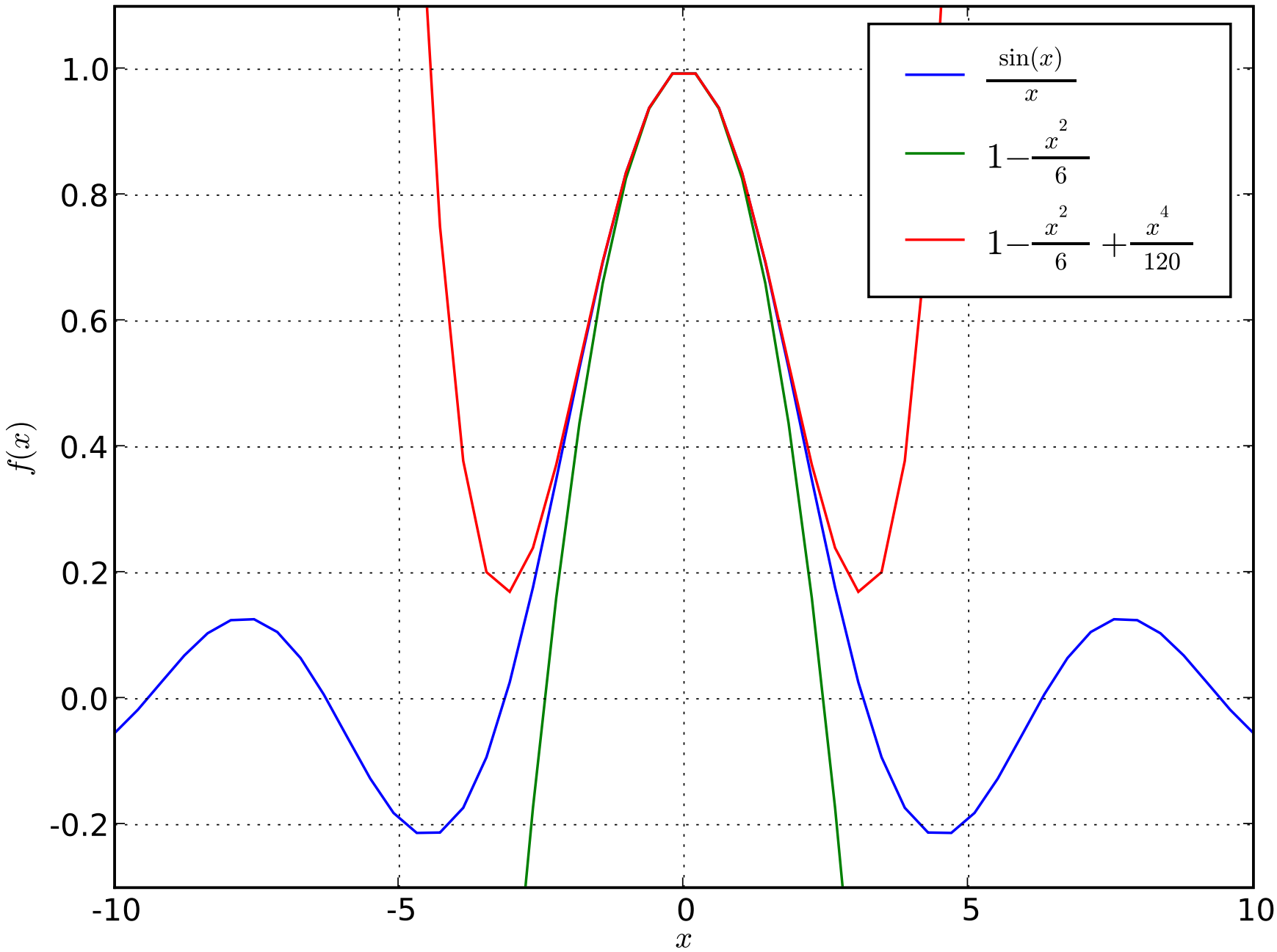
```
from pylab import *
x=linspace(-10,10,50)
plot(x,sin(x)/x)
plot(x,1-x**2/6)
plot(x,1-x**2/6+x**4/120)
ylim(-0.3,1.1)
legend(('f(x)','Taylor 3','Taylor 5'),1)
xlabel('x')
ylabel('f(x)')
grid(c='#BFEFEF',ls='-',lw=0.5)
savefig('plot7a.pdf')
```


f(x)
Taylor 3
Taylor 5
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
f(x)
-10
-5
0
5
10
x

LaTeX

В состав matplotlib входит урезанный интерпретатор LaTeX'а, обрабатывающий строчки, заключенные в $$ (желательно также ставить перед строчкой префикс `r`):

```
from pylab import *
x=linspace(-10,10,50)
plot(x,sin(x)/x,label='1')
plot(x,1-x**2/6,label='2')
plot(x,1-x**2/6+x**4/120,label='3')
ylim(-0.3,1.1)
legend((r'$\frac{\sin(x)}{x}$',r'$1-\frac{x^2}{6}$',\
  r'$1-\frac{x^2}{6}+\frac{x^4}{120}$'),1)
xlabel(r'$x$')
ylabel(r'$f(x)$')
grid()
savefig('plot8.pdf')
```


1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-10
-5
0
5
10
$f(x)$
x
$\frac{\sin(x)}{x}$
$1-\frac{x^2}{6}$
$1-\frac{x^2}{6}+\frac{x^4}{120}$

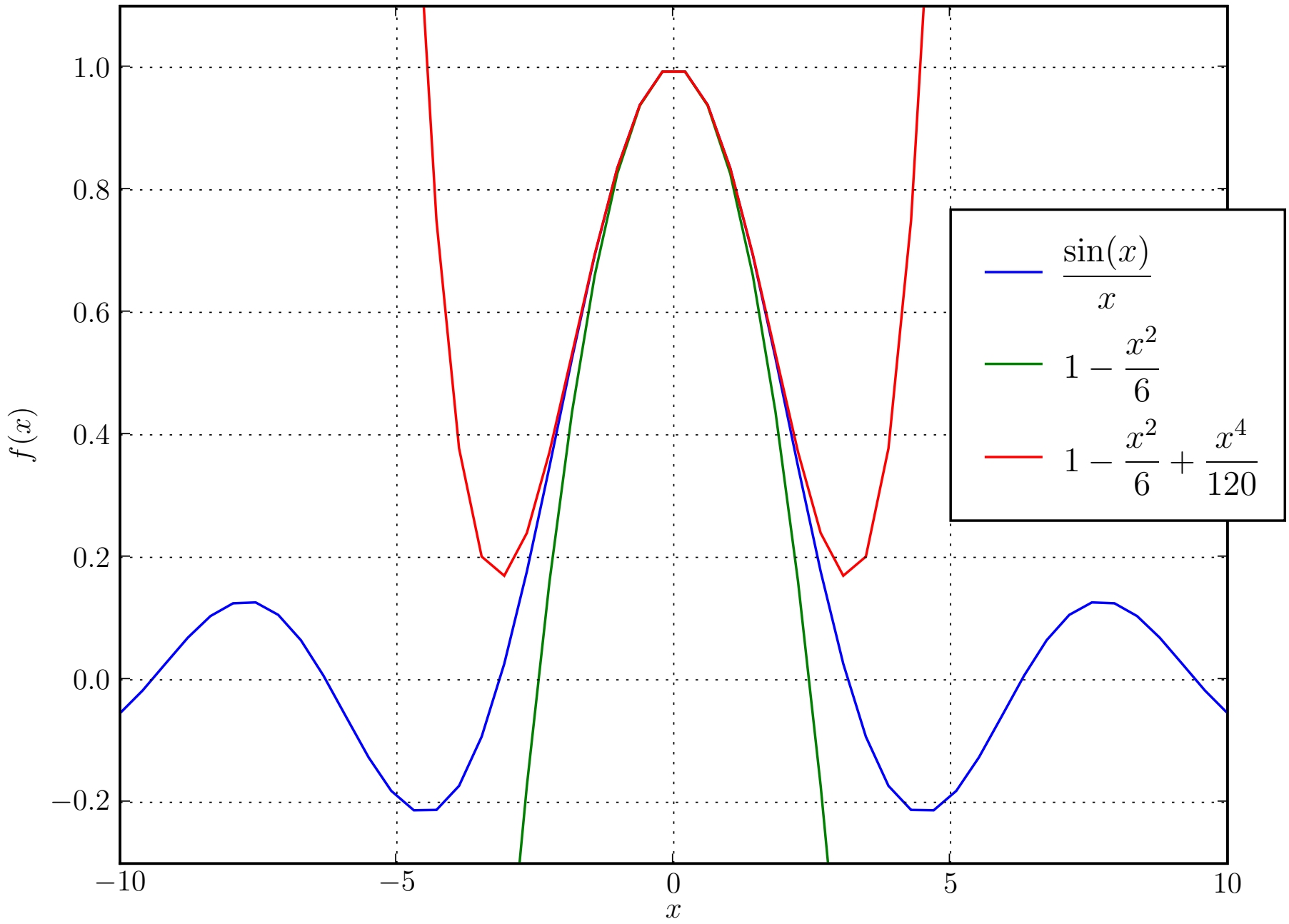
Если возможностей встроенной версии недостаточно, то можно подключить и внешнюю с помощью команды
`rc('text',usetex=True)`
Вообще команды вида `rc(...,...)` временно "на лету" модифицируют файл настроек matplotlibrc, живущий в домашнем каталоге пользователя либо в
???
Он позволяет очень гибко настраивать внешний вид графиков; крайне желательно, в частности, проигнорировать грозное предупреждение и заменить строчку

```
#text.latex.preamble : # IMPROPER USE ...
```

на

```
text.latex.preamble : \usepackage{amsmath}
```

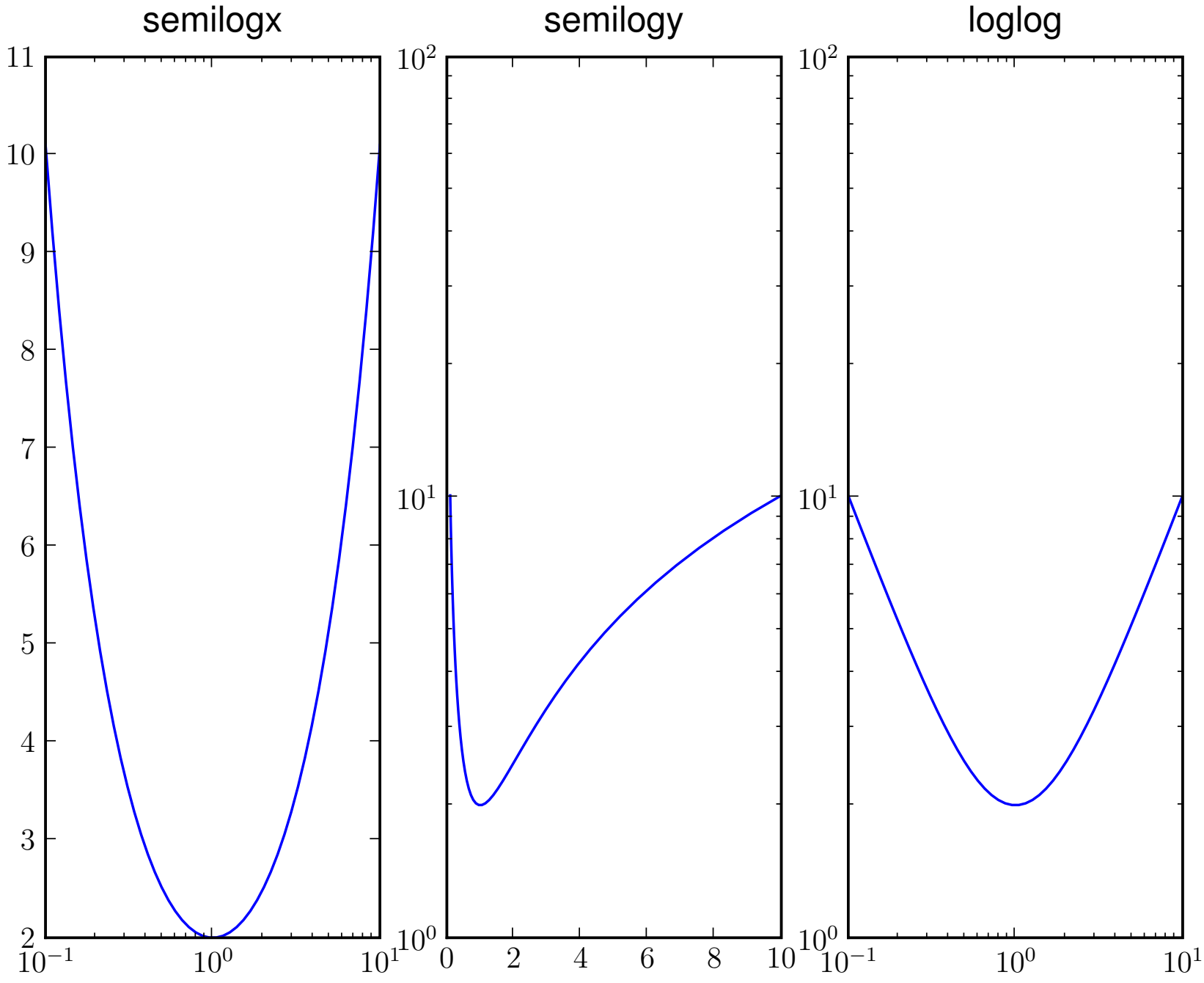
```
from pylab import *
rc('text',usetex=True)
x=linspace(-10,10,50)
plot(x,sin(x)/x,label='1')
plot(x,1-x**2/6,label='2')
plot(x,1-x**2/6+x**4/120,label='3')
ylim(-0.3,1.1)
legend((r'$\dfrac{\sin(x)}{x}$',\
  r'$1-\dfrac{x^2}{6}$',\
  r'$1-\dfrac{x^2}{6}+\dfrac{x^4}{120}$'),\
  loc=(0.75,0.4))
xlabel(r'$x$')
ylabel(r'$f(x)$')
grid()
savefig('plot9.pdf')
```


1.0
0.8
0.6
0.4
0.2
0.0
−0.2
$f(x)$
−10
−5
0
5
10
x
$\frac{\sin(x)}{x}$
$1 - \frac{x^2}{6}$
$1 - \frac{x^2}{6} + \frac{x^4}{120}$

Логарифмические координаты

Команды semilogx, semilogy и loglog позаимствованы из matlab'а и понимают такой же набор параметров, как и plot:

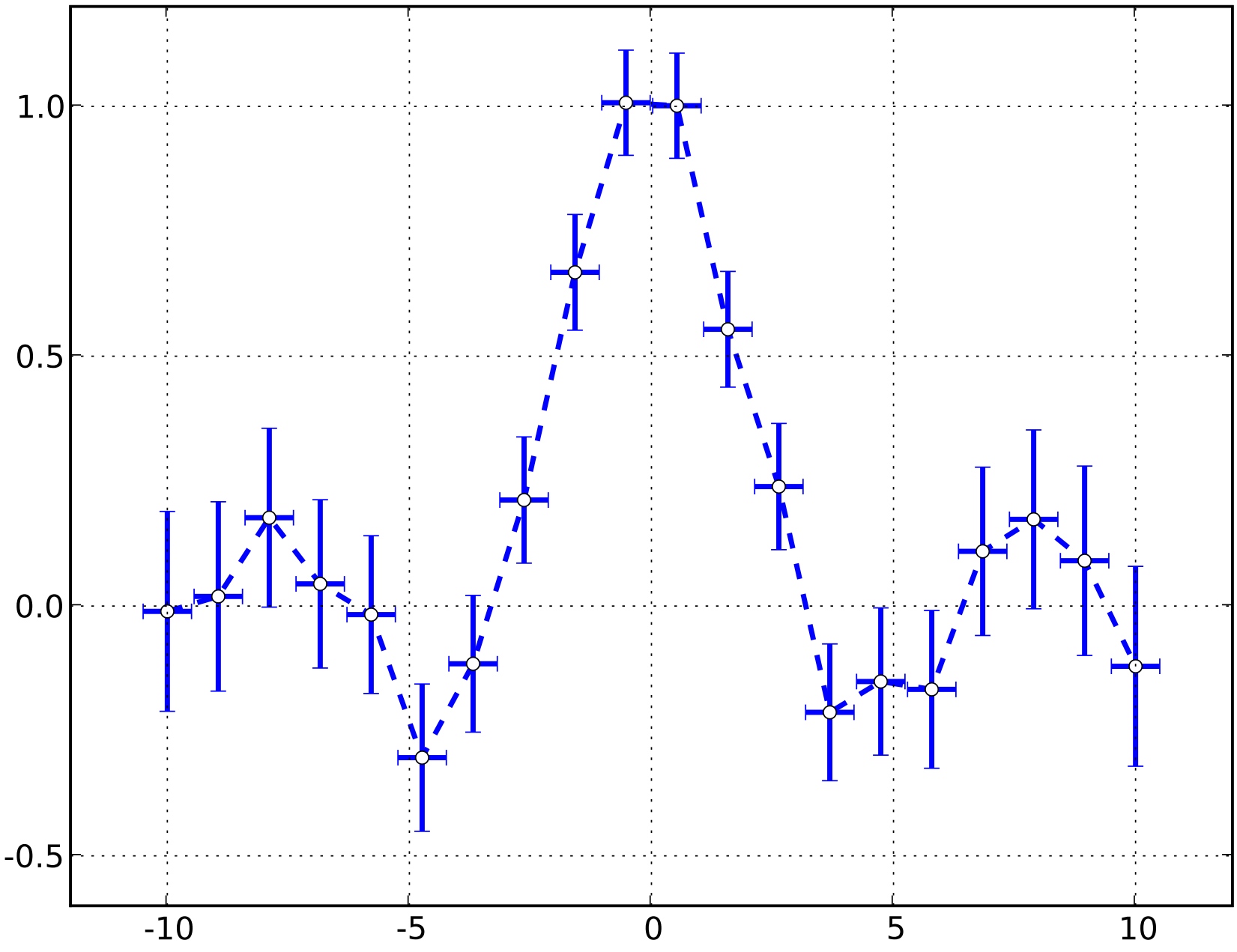
```
from pylab import *
rc('text',usetex=True)
x=logspace(-1,1,50)
subplot(131)
title('semilogx')
semilogx(x,x+1.0/x)
subplot(132)
title('semilogy')
semilogy(x,x+1.0/x)
subplot(133)
loglog(x,x+1.0/x)
title('loglog')
savefig('plot10.pdf')
```


semilogx
semilogy
loglog
11
10
9
8
7
6
5
4
3
2
10^{-1}
10^{0}
10^{1}
10^{2}
10^{1}
10^{0}
0
2
4
6
8
10
10^{2}
10^{1}
10^{0}
10^{-1}
10^{0}
10^{1}

Ошибки измерения

Команда errorbar(x,y,xerr=<данные>,yerr=<данные>) рисует график с указанием ошибок измерения. Параметры xerr, yerr могут быть массивами такой же длины, как x, y, скалярами либо отсутствовать. Остальные необязательные параметры такие же, как у команды plot:

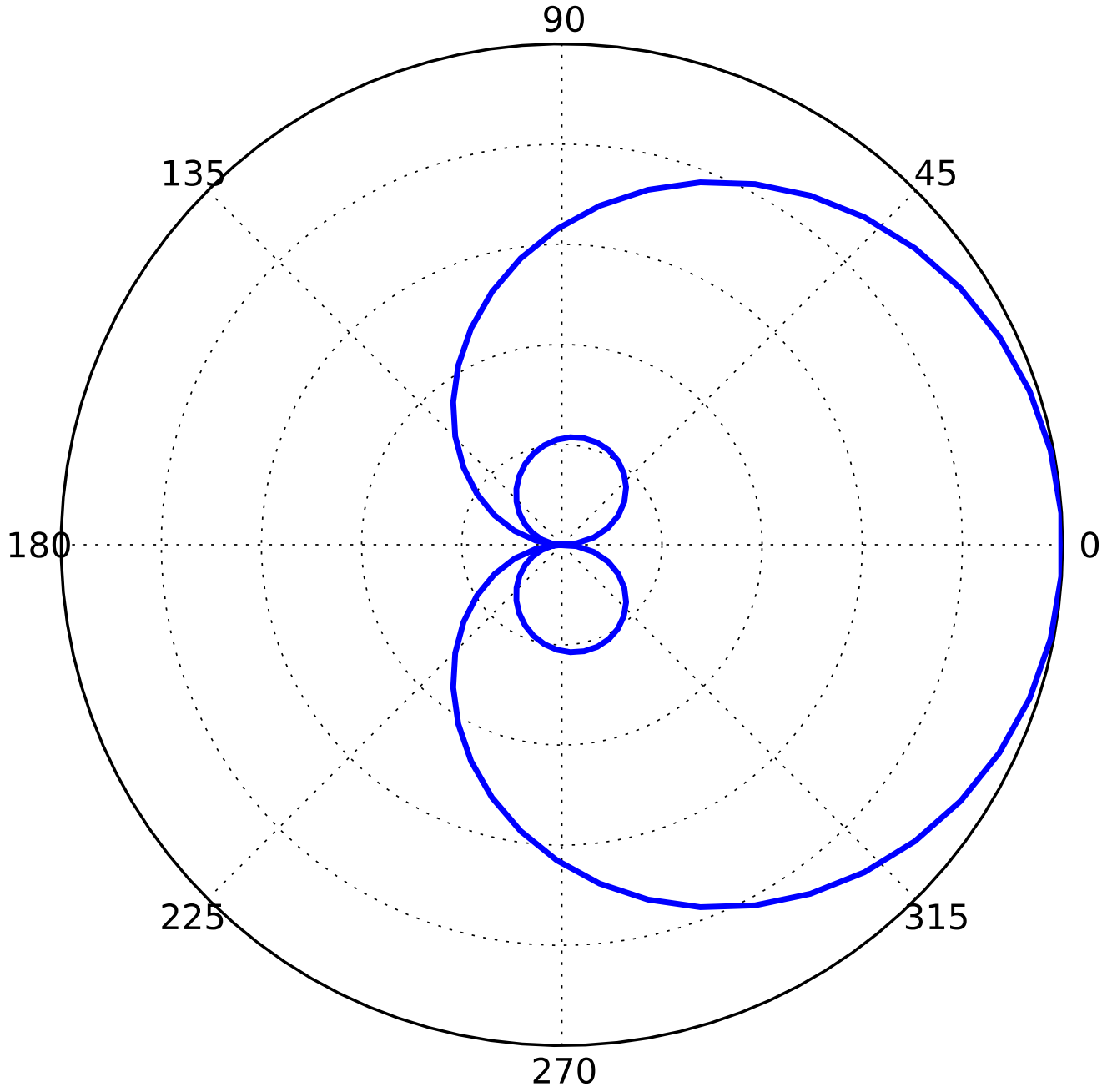
```
from pylab import *
x=linspace(-10,10,20)
y=sin(x)/x + 0.2*rand(20)-0.1
dx =0.5
dy = 0.1+abs(x)/100
errorbar(x, y, xerr=dx, yerr=dy, c='blue', ls='--',\
  lw=2, marker='o', mfc='white', ms=5)
xlim(-12,12)
grid()
savefig('plot11.pdf')
```


1.0
0.5
0.0
-0.5
-10
-5
0
5
10

Полярные координаты

Команда polar(theta,r) рисует график в полярных координатах. Остальные необязательные параметры такие же, как у команды plot:

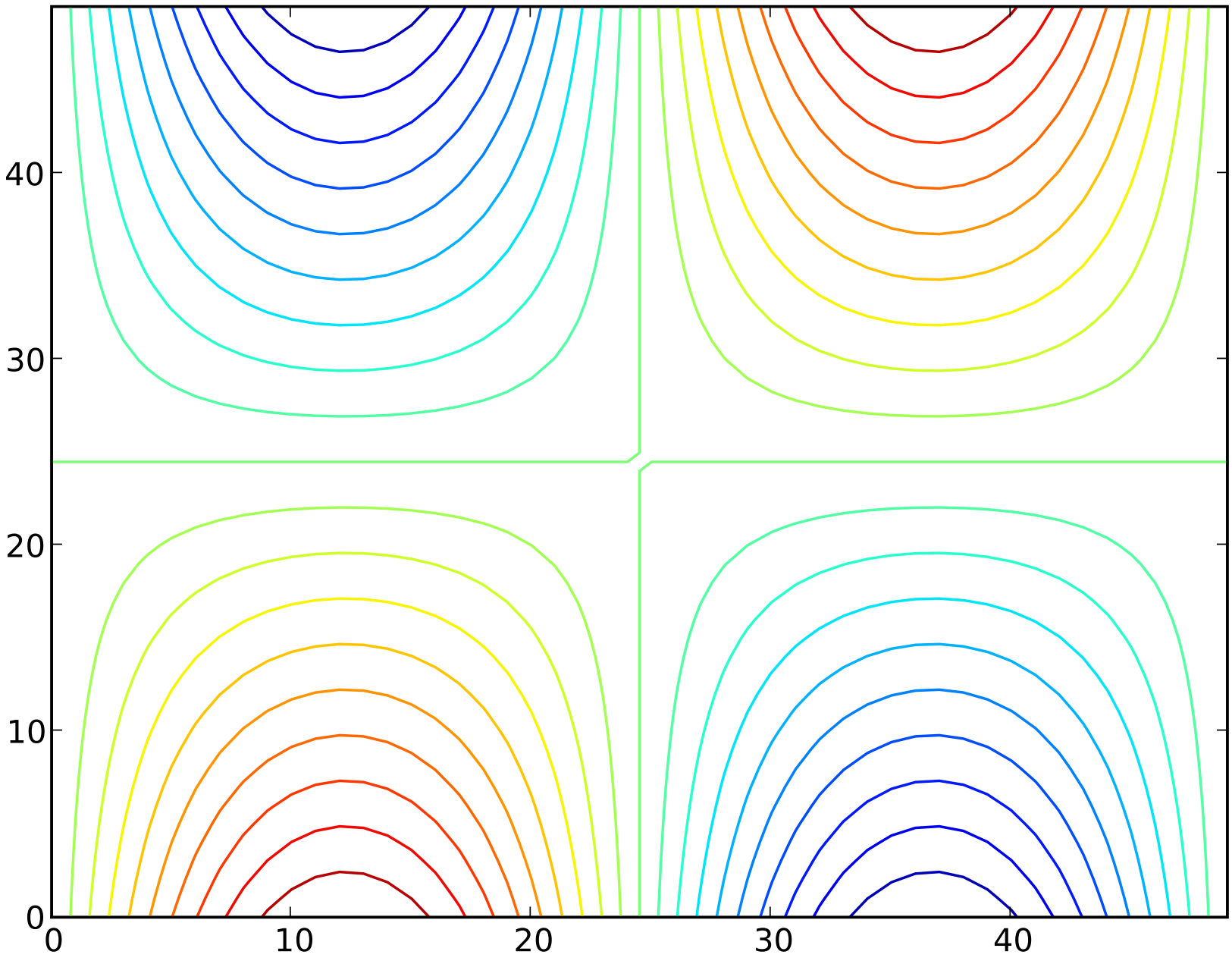
```
from pylab import *
x=linspace(-2*pi,2*pi,100)
y=sin(x)/x
polar(x, y, c='blue', lw=2)
#grid()
savefig('plot12.pdf')
```


90
135
45
180
0
225
315
270

Контурный график

Команда contour(z,levels) рисует контурный график двумерного массива z. Параметр levels – одномерный массив, задающий изоуровни:

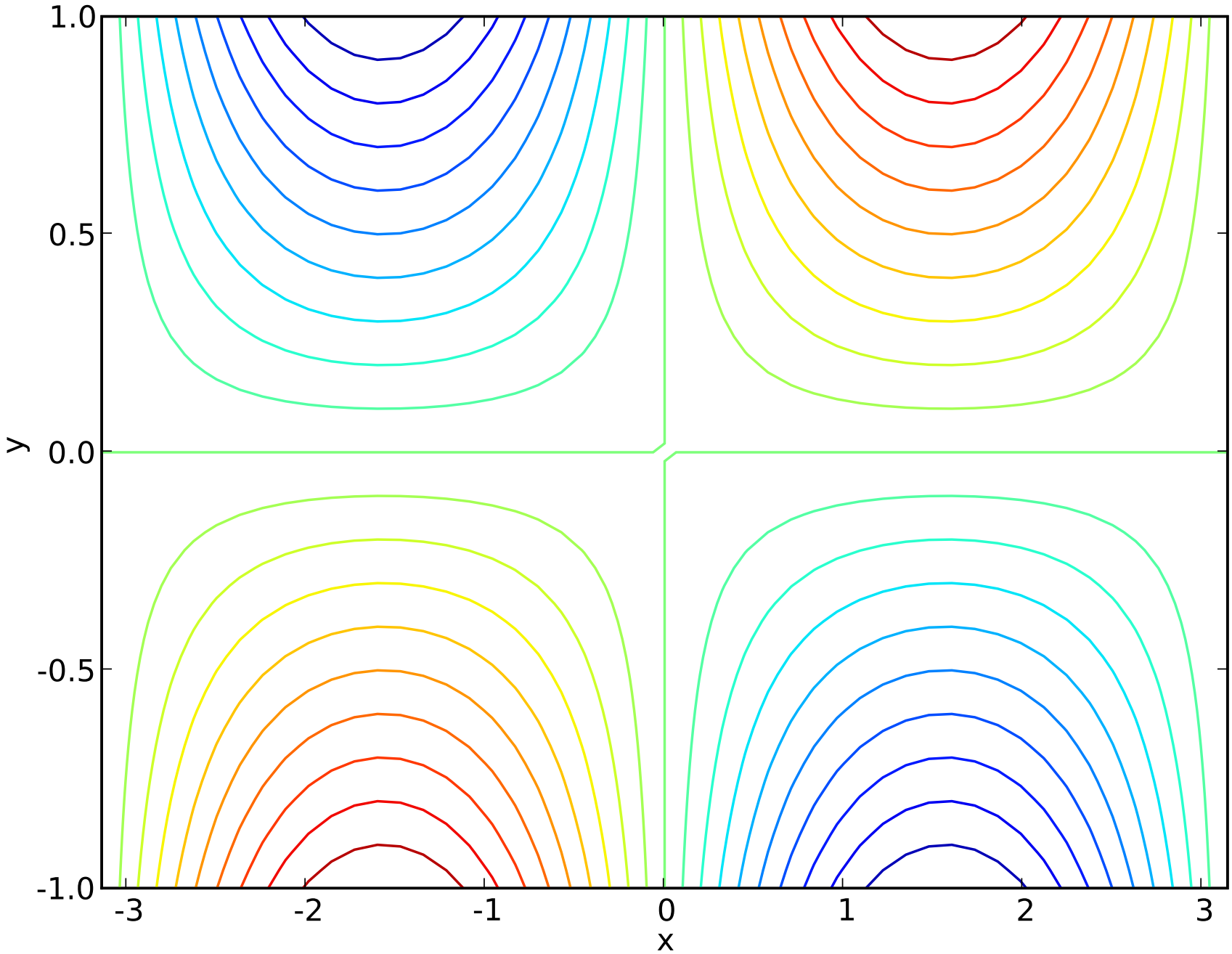
```
from pylab import *
x=linspace(-pi,pi,50)
y=linspace(-1,1,50)
z=matrix(y).T*sin(x)
contour(z,linspace(-1,1,21))
savefig('plot13.pdf')
```


0
10
20
30
40
0
10
20
30
40

Независимые переменные

В качестве независимых переменных x, y использовались индексы массива. Можно задать их и явно:

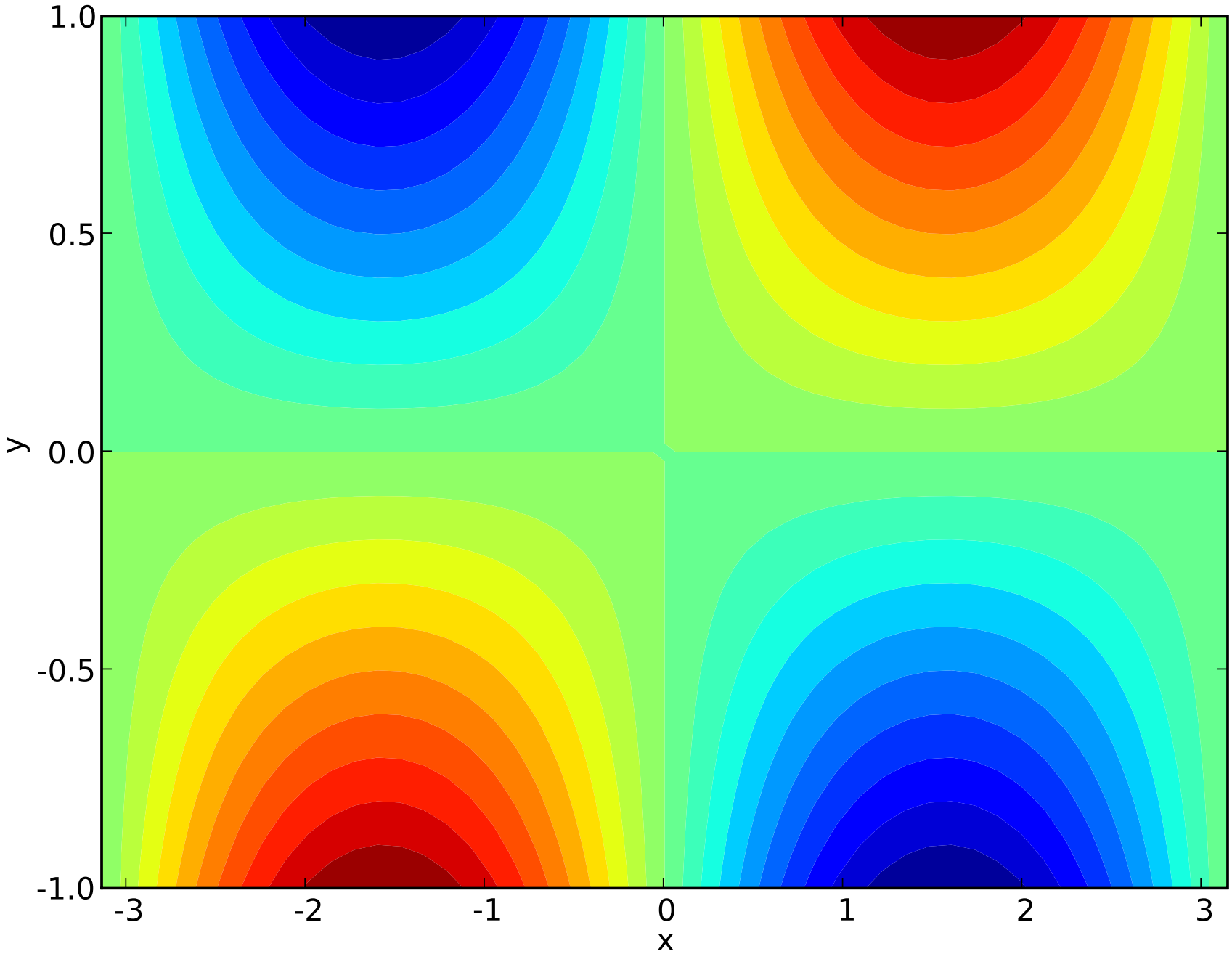
```
from pylab import *
x=linspace(-pi,pi,50)
y=linspace(-1,1,50)
z=matrix(y).T*sin(x)
contour(x,y,z,linspace(-1,1,21))
xlabel('x')
ylabel('y')
savefig('plot14.pdf')
```


1.0
0.5
0.0
-0.5
-1.0
y
-3
-2
-1
0
1
2
3
x

Заполнение цветом

Команда contourf, с тем же синтаксисом, рисует “цветную” версию того же графика:

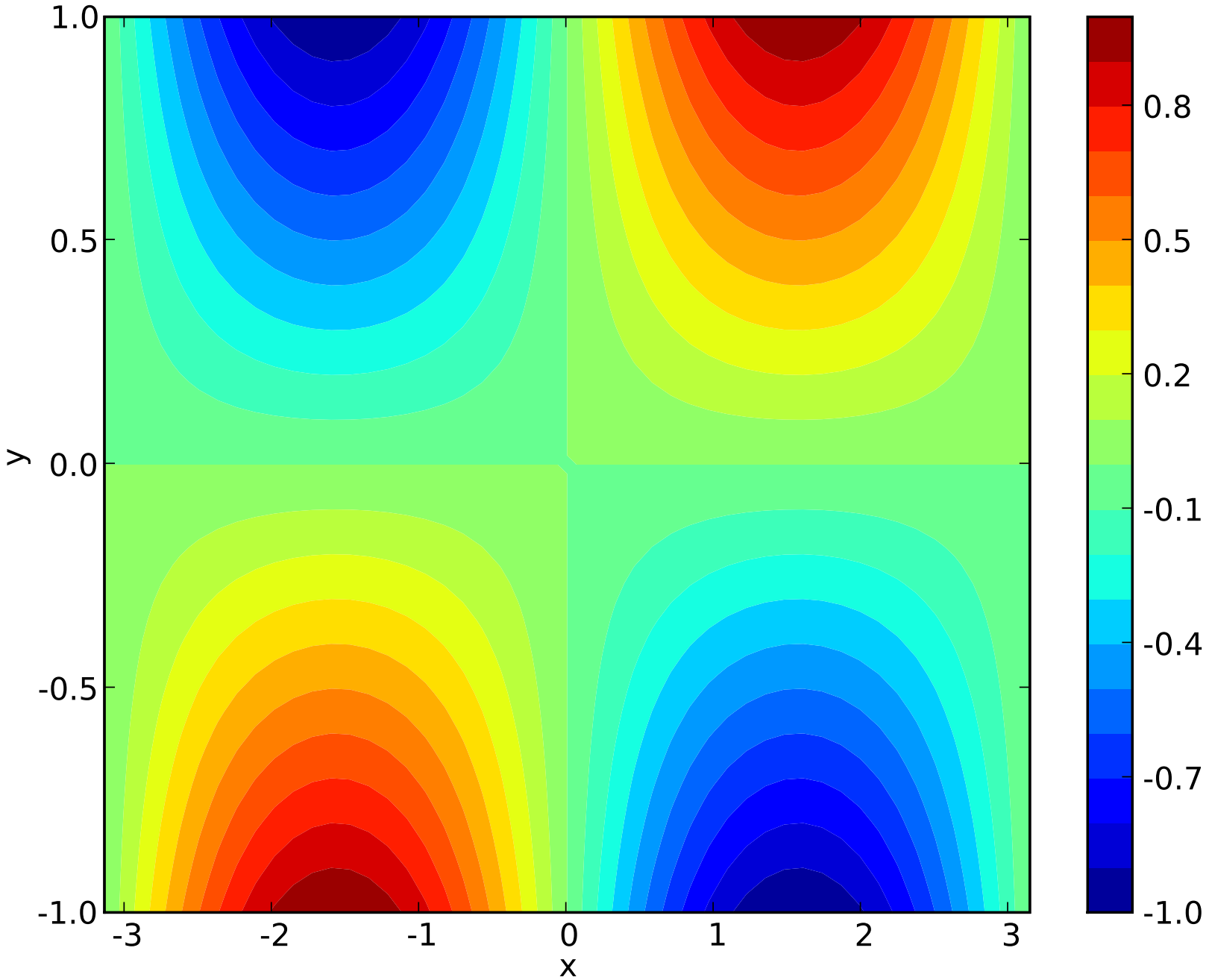
```
from pylab import *
x=linspace(-pi,pi,50)
y=linspace(-1,1,50)
z=matrix(y).T*sin(x)
contourf(x,y,z,linspace(-1,1,21))
xlabel('x')
ylabel('y')
savefig('plot15.pdf')
```


1.0
0.5
0.0
-0.5
-1.0
y
-3
-2
-1
0
1
2
3
x

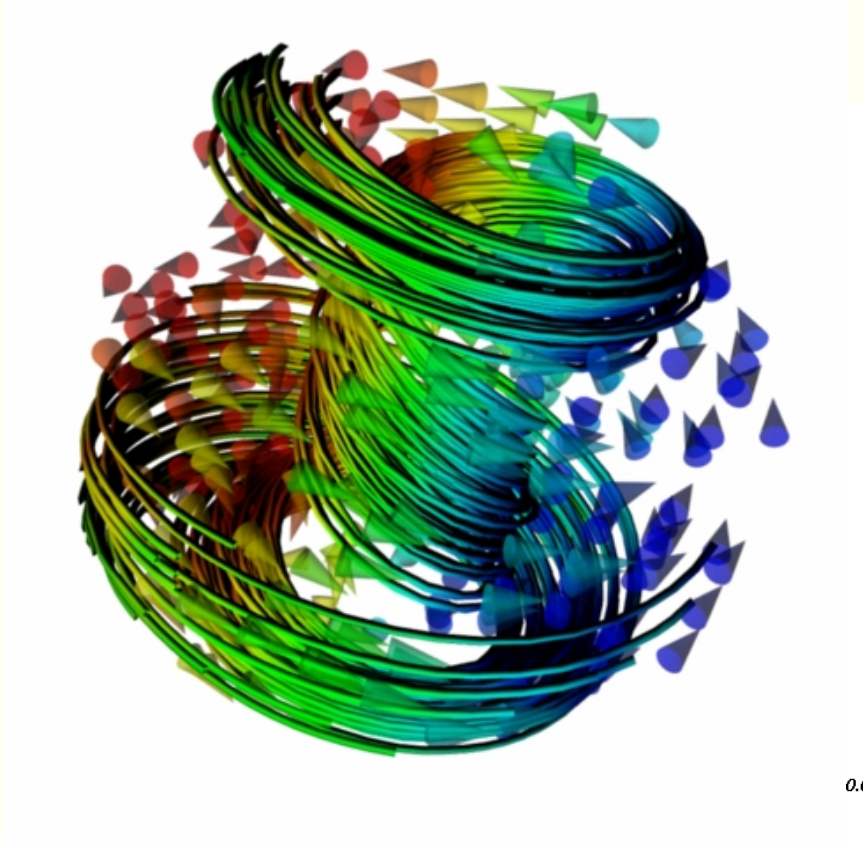
Colorbar

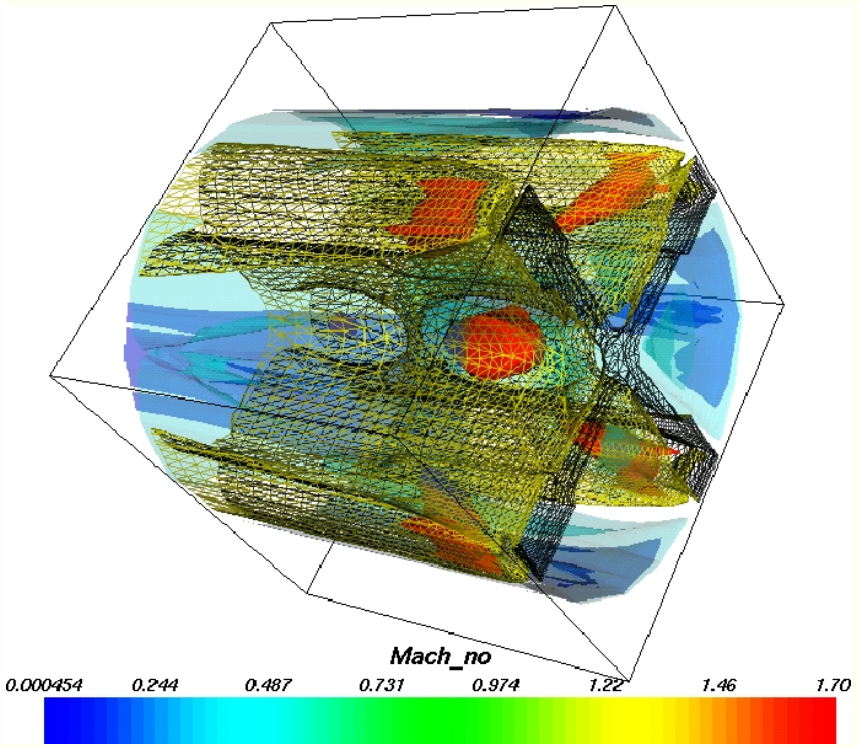
Обычно к такому графику прилагается colorbar:

```
from pylab import *
x=linspace(-pi,pi,50)
y=linspace(-1,1,50)
z=matrix(y).T*sin(x)
cf=contourf(x,y,z,linspace(-1,1,21))
colorbar(cf)
xlabel('x')
ylabel('y')
savefig('plot16.pdf')
```


1.0
0.5
0.0
-0.5
-1.0
y
-3
-2
-1
0
1
2
3
x
0.8
0.5
0.2
-0.1
-0.4
-0.7
-1.0

Mayavi



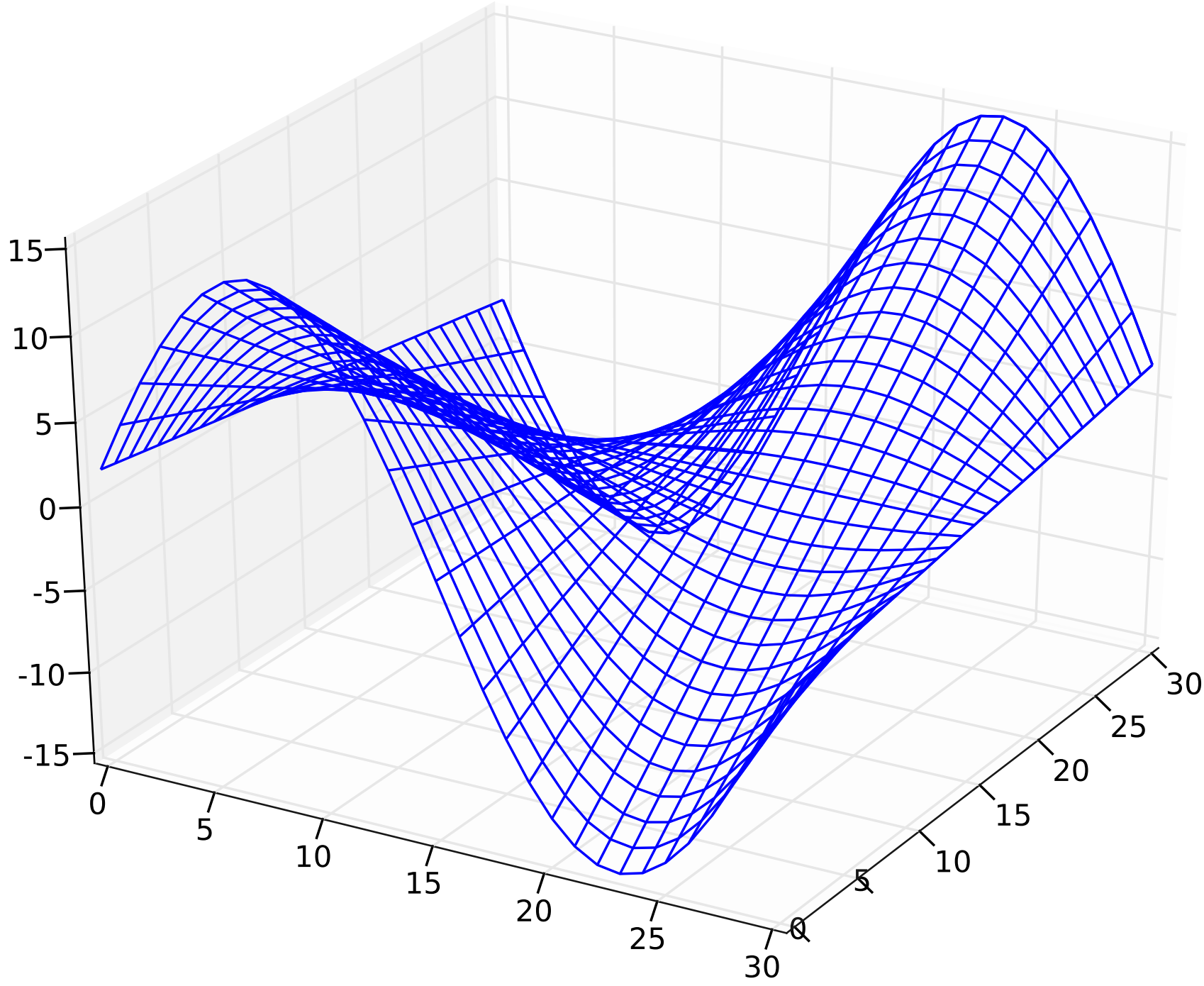


Для построения 3D графиков рекомендуется использовать отдельный пакет mayavi, см. http://code.enthought.com/projects/mayavi

3D

В предыдущих версиях natplotlib имелись рудиментарные возможности 3D. Они до сих пор поддерживаются в т.н. “maintenance” версиях 0.91.x. В частности, команда plot_wireframe рисует “проволочный график”:

```
from pylab import *
import matplotlib.axes3d as axes3d
ax=axes3d.Axes3D(gcf())
N=31
x=linspace(0,N-1,N)
y=linspace(0,N-1,N)
X,Y=meshgrid(x,y)
Z=fromfunction(lambda y,x:sin(0.2*(x-N/2))*(y-N/2),\
  (N,N))
ax.plot_wireframe(X,Y,Z)
savefig('plot17.pdf')
```

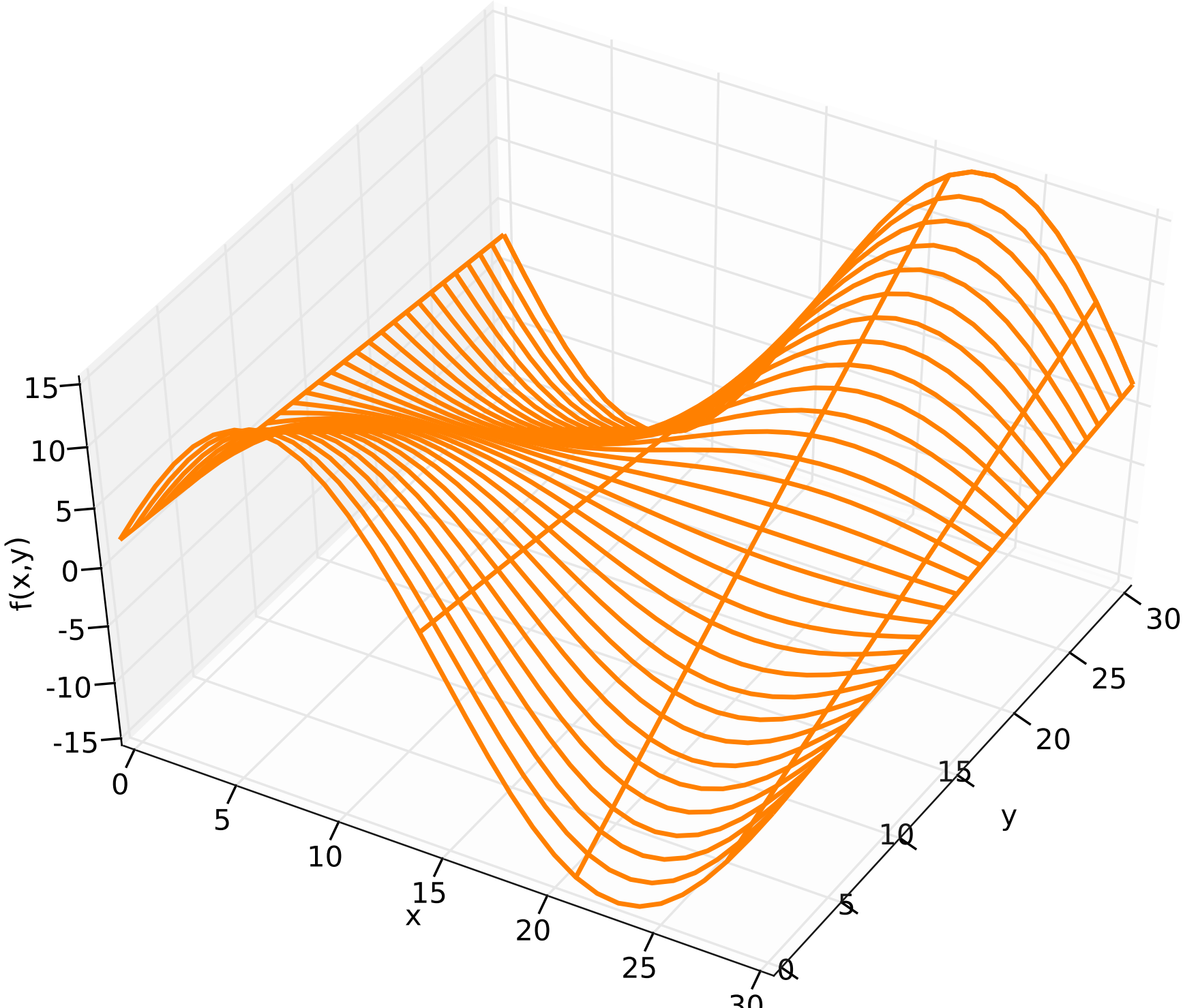

15
10
5
0
-5
-10
-15
0
5
10
15
20
25
30
0
5
10
15
20
25
30

3D

Команда plot_wireframe имеет два необязательных аргумента cstride и rstride, задающие шаг сетки по рядам и колонкам. Иногда они позволяют улучшить вид графика.
Толщину и цвет линии можно задавать через аргументы lw и color.
Начальный угол зрения задается дополнительной командой view_init(<угол места>,<азимут>).
Обратите также внимание, что подписи по осям задаются несколько иначе, чем в двумерных графиках.

3D

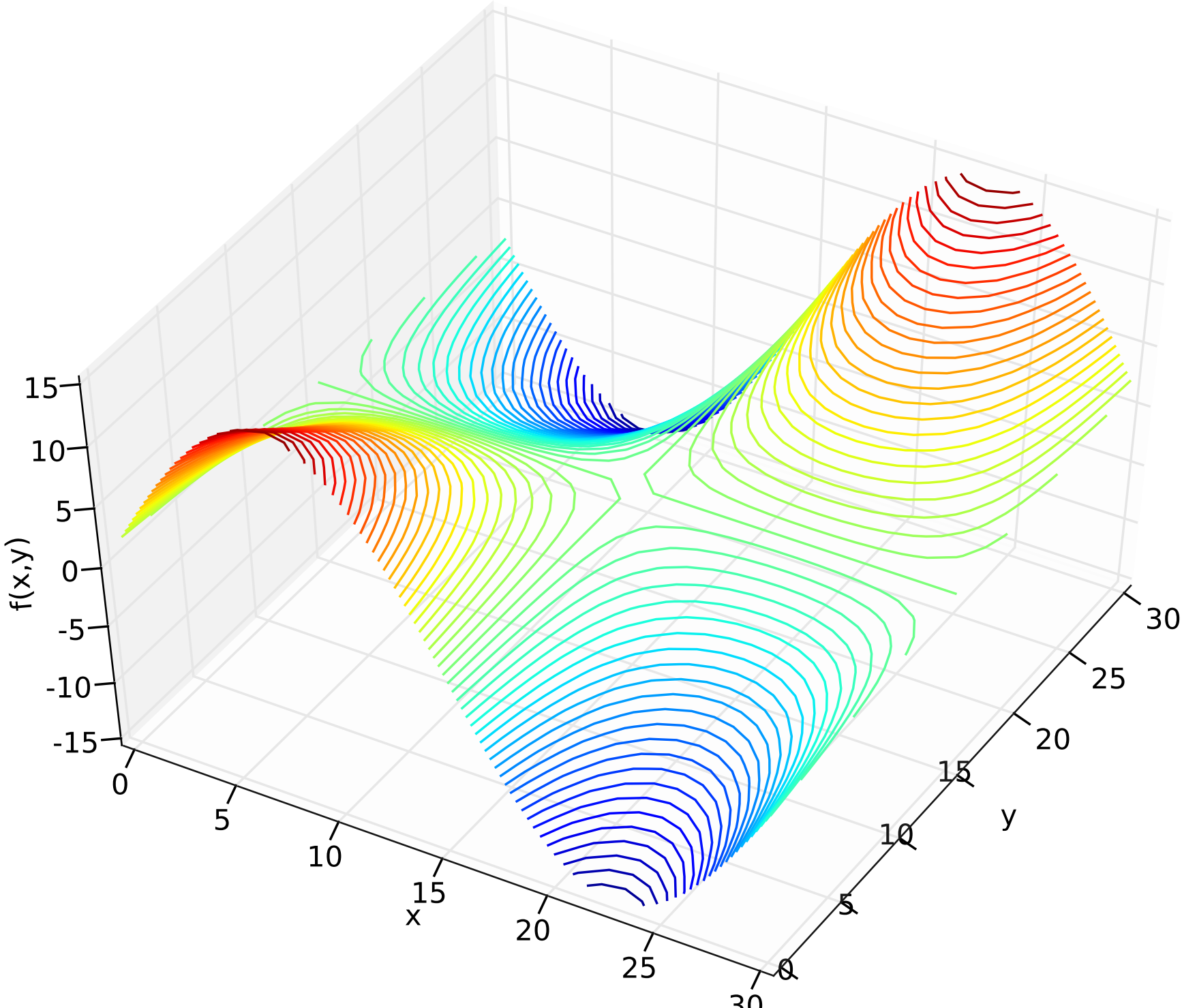
```
from pylab import *
import matplotlib.axes3d as axes3d
ax=axes3d.Axes3D(gcf())
N=31
x=linspace(0,N-1,N)
y=linspace(0,N-1,N)
X,Y=meshgrid(x,y)
Z=fromfunction(lambda y,x:sin(0.2*(x-N/2))*(y-N/2),\
  (N,N))
ax.view_init(50,-60)
ax.plot_wireframe(X,Y,Z,color='#FF8000',lw='2',\
  cstride=7,rstride=1)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('f(x,y)')
savefig('plot18.pdf')
```


15
10
5
0
-5
-10
-15
f(x,y)
0
5
10
15
20
25
30
x
0
5
10
15
20
25
30
y

3D

Другой вариант – это команда contour3D:

```
from pylab import *
import matplotlib.axes3d as axes3d
ax=axes3d.Axes3D(gcf())
N=31
x=linspace(0,N-1,N)
y=linspace(0,N-1,N)
X,Y=meshgrid(x,y)
Z=fromfunction(lambda y,x:sin(0.2*(x-N/2))*(y-N/2),\
  (N,N))
ax.view_init(50,-60)
ax.contour3D(X,Y,Z,linspace(-15,15,51))
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('f(x,y)')
savefig('plot19.pdf')
```


15
10
5
0
-5
-10
-15
f(x,y)
0
5
10
15
20
25
30
x
0
5
10
15
20
25
30
y