Основы матанализа

(модули scipy.integrate и scipy.fftpack)

Ф.Я.Халили

МГУ, физический факультет

22 апреля 2008 г.

1. Дифференцирование
2. Интегрирование аналитически заданных функций
3. Интегрирование таблично заданных функций
4. Дискретное преобразование Фурье

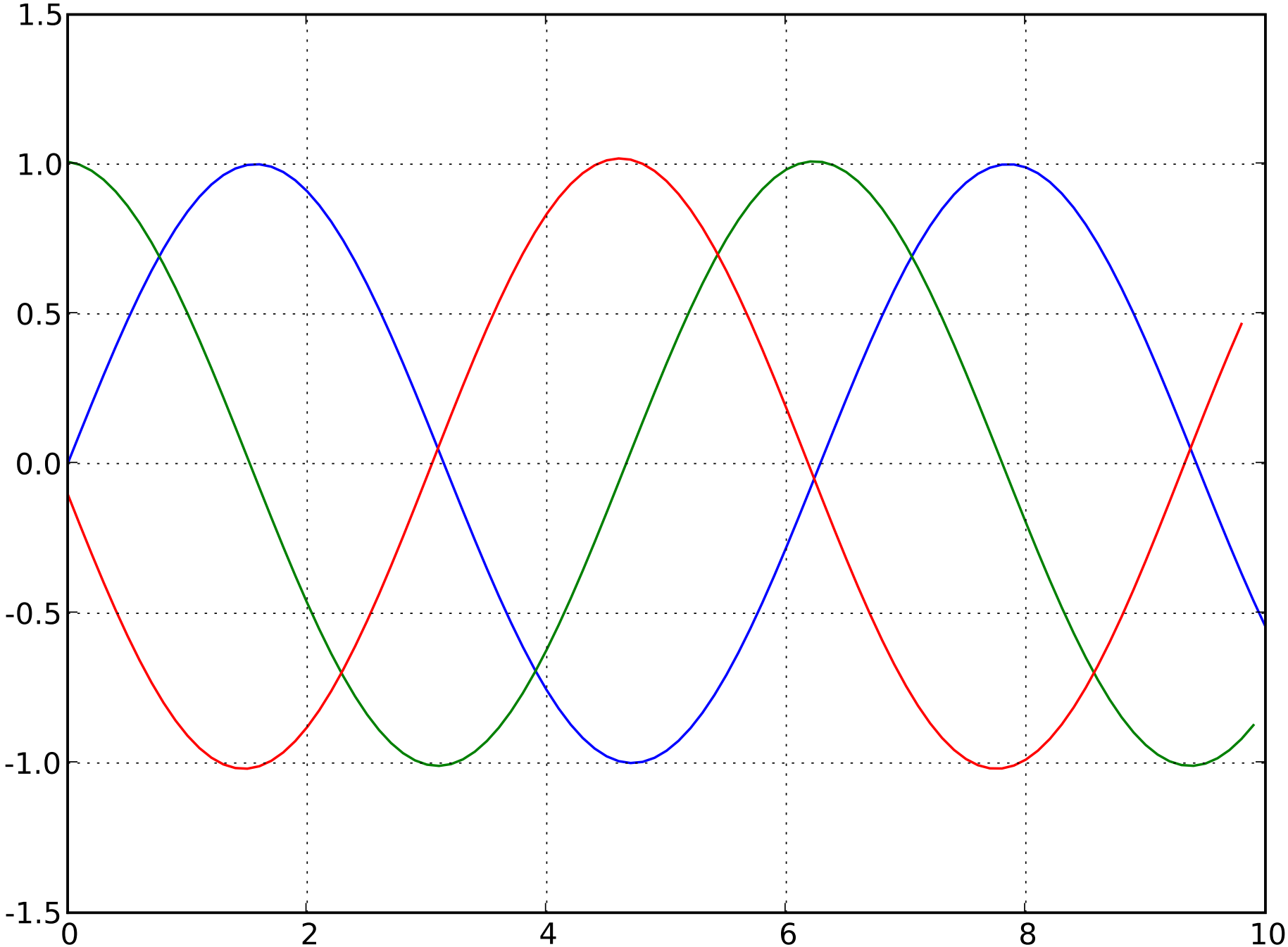
1. Дифференцирование
2. Интегрирование аналитически заданных функций
3. Интегрирование таблично заданных функций
4. Дискретное преобразование Фурье

Дифференцирование

`diff(<массив>,<порядок>=1,<ось>=-1)`

(эта команда входит в состав базового модуля numpy)

```
from pylab import *
xmax=10
N=101
x=linspace(0,xmax,N)
y=sin(x)
y1=(float(N)/xmax)*diff(y)
y2=(float(N)/xmax)**2*diff(y,2)
plot(x,y)
plot(x[:-1],y1)
plot(x[:-2],y2)
grid()
savefig('plot1.pdf')
```


1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
0
2
4
6
8
10

1. Дифференцирование
2. Интегрирование аналитически заданных функций
3. Интегрирование таблично заданных функций
4. Дискретное преобразование Фурье

Подключение

Субмодуль scipy.integrate обеспечивает стандартные методы численного интегрирования. Загружается он командами

```
from scipy import *
from scipy.integrate import *
```

либо

```
from pylab import *
from scipy.integrate import *
```

(второй вариант подключает также модуль matplotlib).

Одномерный интеграл

`[<интеграл>,<ошибка>] = quad(`$f(x)$`,`$x_{\min}$`,`$x_{\max}$`)`

`Inf` изображает бесконечный предел интегрирования

```
>>> quad(sin,0,pi)
(2.0, 2.2204460492503131e-14)
>>> quad(sin,0,pi)[0]
2.0
>>> quad(sin,pi,0)[0]
-2.0
>>> quad(lambda x:e**(-x**2),-Inf,Inf)/sqrt(pi)
array([  1.00000000e+00,   8.01297973e-09])
```

Более (но не совсем) полный синтаксис

```
quad(f(x), x_min, x_max, full_output=0, \
  epsabs=1.5·10^-8, epsrel=1.5·10^-8, \
  limit=50, points=None, ...)
```

full_output	выдача информации о процессе интегрирования
epsabs	абсолютная ошибка
epsrel	относительная ошибка
limit	верхний предел числа суб-интервалов
points	список точек, где могут возникать проблемы (разрывы, сингулярности)

Кратные интегралы

`dblquad(`$f(y,x)$`,`$x_{\min}$`,`$x_{\max}$`,`$y_{\min}(x)$`,`$y_{\max}(x)$`)`

`tplquad(`$f(z,y,x)$`,`$x_{\min}$`,`$x_{\max}$`,`$y_{\min}(x)$`,`$y_{\max}(x)$`,`
$z_{\min}(y,x)$`,`$z_{\max}(y,x)$`)`

$y_{\min,\max}$ и $z_{\min,\max}$ обязательно должны быть заданы как функции!

```
>>> dblquad(lambda y,x:sqrt(1-x**2-y**2),-1,1,\
... lambda x:0,lambda x:sqrt(1-x**2))[0]/pi
0.3333333333333331
```

1. Дифференцирование
2. Интегрирование аналитически заданных функций
3. Интегрирование таблично заданных функций
4. Дискретное преобразование Фурье

Метод Симпсона

`simps(y,dx=1,axis=-1)` `simps(y,x,axis=-1)`

```
>>> x=linspace(0,pi,10)
>>> simps(sin(x))
5.7282851768386545
>>> x=linspace(0,pi,11)
>>> simps(sin(x))
6.3665463281165549
>>> simps(sin(x),dx=pi/10)
2.0001095173150043
>>> simps(sin(x),x)
2.0001095173150043
```

Сравнение методов интегрирования

`trapz(y,dx=1,axis=-1)` `trapz(y,x,axis=-1)`

`romb(y,dx=1,axis=-1)`

```
>>> x=linspace(0,pi,5)
>>> trapz(sin(x),dx=pi/4)-2
-0.103881102062960 2
>>> simps(sin(x),dx=pi/4)-2
0.0045597549844207386
>>> romb(sin(x),dx=pi/4)-2
0.094395102393195263
>>> x=linspace(0,pi,33)
>>> trapz(sin(x),dx=pi/32)-2
-0.0016066390298556943
>>> simps(sin(x),dx=pi/32)-2
1.0333694127062643e-06
>>> romb(sin(x),dx=pi/32)-2
-5.4127098358947023e-09
```

Сразу несколько интегралов

```
>>> x=linspace(0,1,5)
>>> z=vstack((x,x**2,x**3))
>>> z
array([[ 0.   , 0.25    , 0.5    ,  0.75    ,  1.  ],
       [ 0.   , 0.0625  , 0.25   ,  0.5625  ,  1.  ],
       [ 0.   , 0.015625, 0.125  ,  0.421875,  1.  ]])
>>> simps(z,x)
array([ 0.5        ,  0.33333333,  0.25      ])
>>> simps(z,dx=1,axis=0)
array([ 0. , 0.171875, 0.54166667, 1.140625 ,  2. ])
```

Свертка

`convolve(<вектор1>,<вектор2>)`

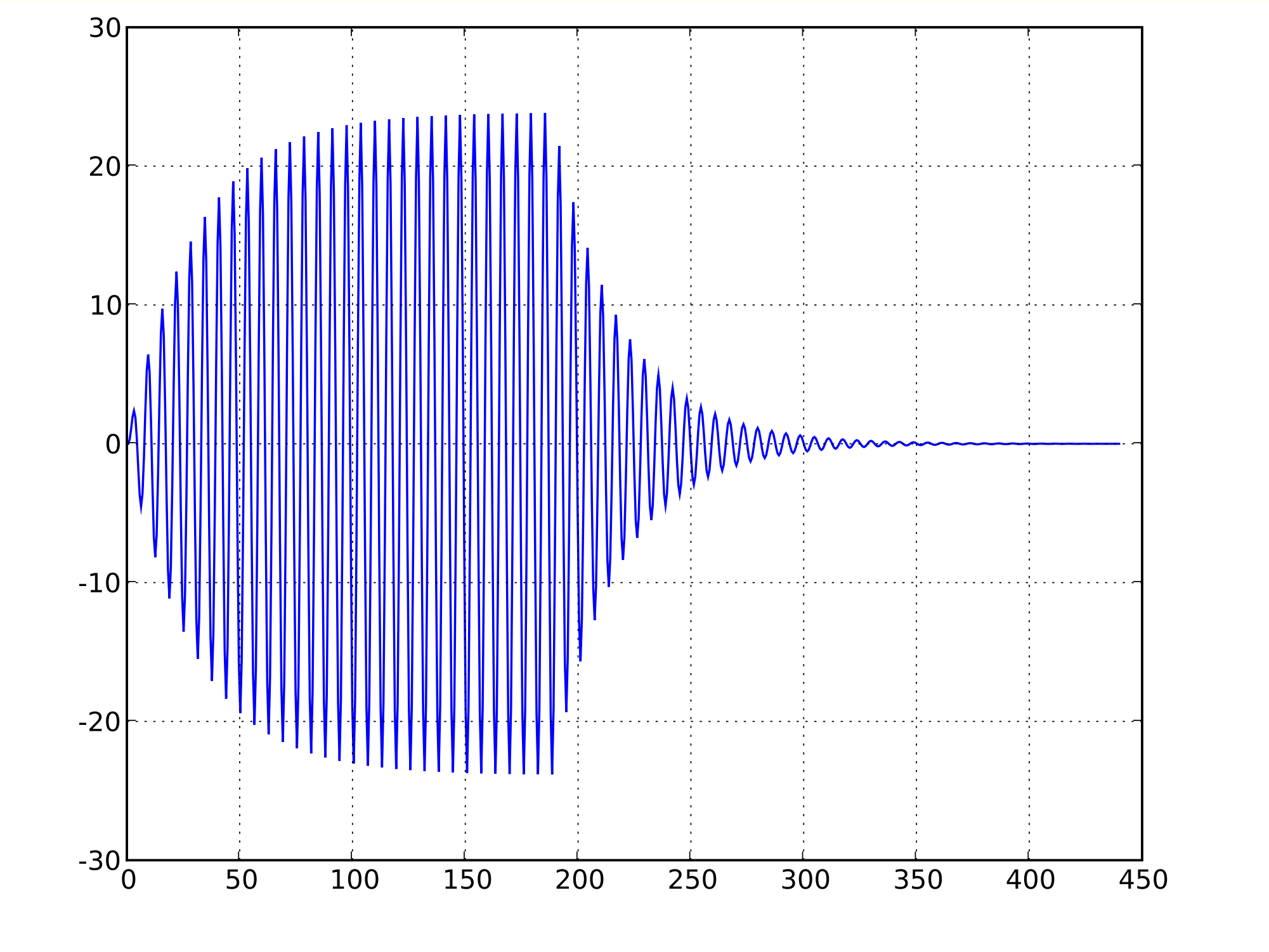
Эта команда входит в состав базового модуля numpy и вычисляет свертку

$$\text{convolve}(x, y)[j] = \sum_k x[j-k] \cdot y[k]$$

Размер полученного вектора равен сумме длин аргументов - 1.

Свертка

```
from pylab import *
T1=30
N1=10*T1+1
t1=linspace(0,2*pi*T1,N1)
f=sin(t1)
T2=40
N2=10*T2+1
t2=linspace(0,2*pi*T2,N2)
l=exp(-t2/30)*sin(t2)
x=convolve(l,f)
t=linspace(0,2*pi*(T1+T2),N1+N2-1)
plot(t,x)
grid()
savefig('plot7.pdf')
```


30
20
10
0
-10
-20
-30
0
50
100
150
200
250
300
350
400
450

1. Дифференцирование
2. Интегрирование аналитически заданных функций
3. Интегрирование таблично заданных функций
4. Дискретное преобразование Фурье

Подключение

Субмодуль scipy.fftpack обеспечивает стандартные функции быстрого преобразования Фурье. Загружается он командами

```
from scipy import *
from scipy.fftpack import *
```

либо

```
from pylab import *
from scipy.fftpack import *
```

(второй вариант подключает также модуль matplotlib).

Прямое преобразование Фурье:

```
fft(x,axis=-1)
```
```
fft(x,n=<размер>, axis=-1)
```

Обратное преобразование Фурье:

```
ifft(x,axis=-1)
```
```
fft(x,n=<размер>, axis=-1)
```

Двумерное прямое преобразование Фурье:

```
fft2(x,axes=(-2,-1))
```

```
ifft2(x,shape=<(размер,размер)>, axes=(-2,-1))
```

Двумерное обратное преобразование Фурье:

```
ifft2(x,axes=(-2,-1))
```

```
ifft2(x,shape=<(размер,размер)>, axes=(-2,-1))
```

n-мерное прямое преобразование Фурье:

```
fftn(x,axes=(...,-2,-1))
```

```
ifftn(x,shape=<(размер,...)>, axes=(...,-2,-1))
```

n-мерное обратное преобразование Фурье:

```
ifftn(x,axes=(...,-2,-1))
```

```
ifftn(x,shape=<(размер,...)>, axes=(...,-2,-1))
```

Справочные формулы

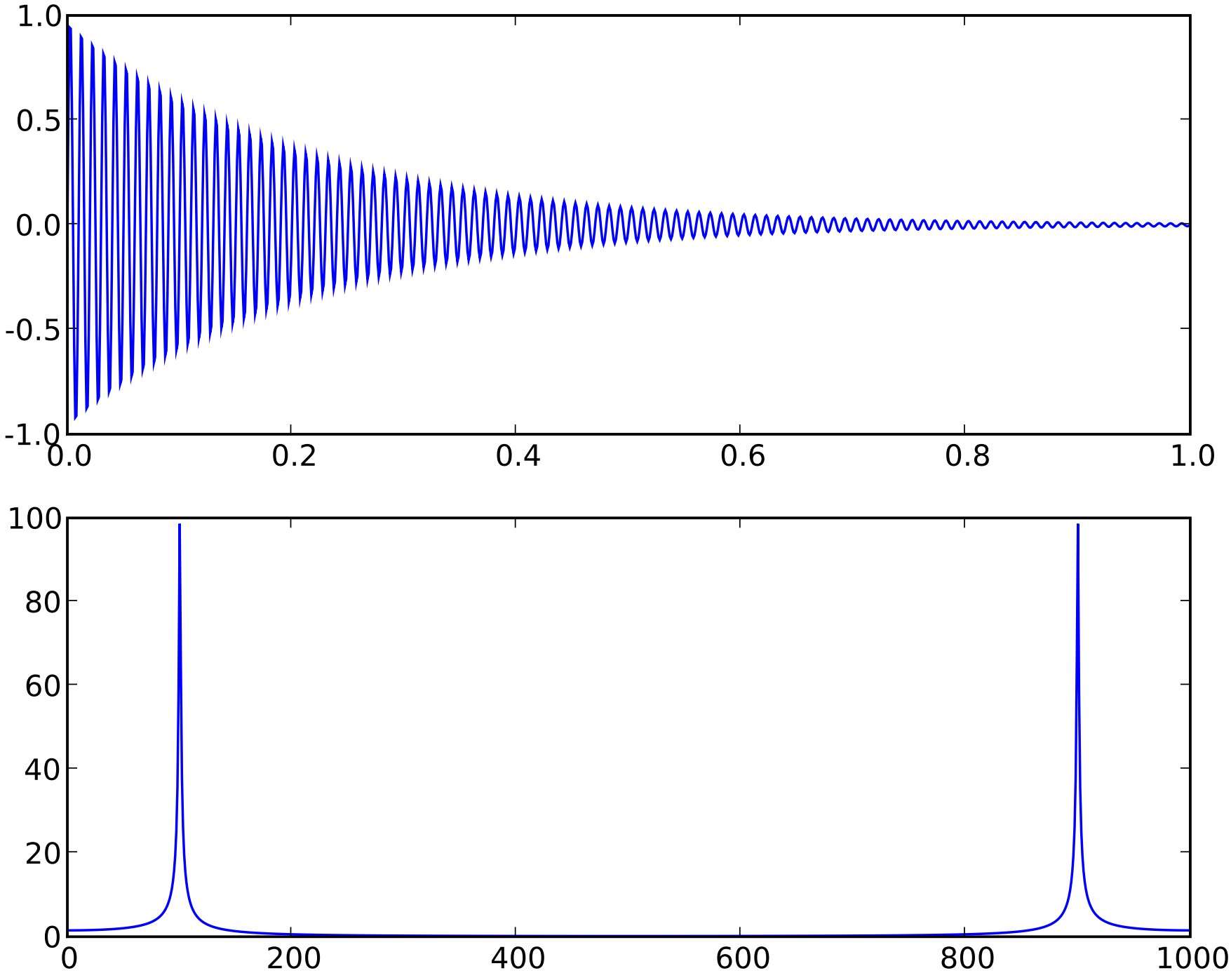
$$\text{fft}: \quad \tilde{x}[j] = \sum_{k=0}^{n-1} x[k] \exp\left(\frac{2\pi i}{n} jk\right)$$

$$\text{ifft}: \quad x[j] = \frac{1}{N} \sum_{k=0}^{n-1} \tilde{x}[k] \exp\left(-\frac{2\pi i}{n} jk\right)$$

$$\text{fft}(\text{ifft}(x)) = \text{ifft}(\text{fft}(x)) = x$$

Пример

```
from pylab import *
from scipy.fftpack import *
t=linspace(0,1,1000)
x=exp(-5*t)*sin(2*pi*100*t)
s=fft(x)
subplot(211)
plot(t,x)
subplot(212)
plot(abs(s))
savefig('plot8.pdf')
```


1.0
0.5
0.0
-0.5
-1.0
0.0
0.2
0.4
0.6
0.8
1.0
100
80
60
40
20
0
0
200
400
600
800
1000