Scipy 作业

1、求解非线性方程组，cos(a) = 1 - d^2 / (2*r^2) ，L = a * r，d = 140，L = 156；导入参数雅克比矩阵，再次进行求解。

Code:

```
from scipy.optimize import fsolve
from math import cos, sin

d, L = 140, 156

def f(x):
    x0, x1 = x.tolist()  # x0 表示 a，x1 表示 r
    return [cos(x0) - 1 + d ** 2 / (2 * x1 ** 2), L - x0 * x1]

result1 = fsolve(f, [1, 1])
print(result1)
print(f(result1))

# 下面引入雅可比矩阵
def j(x):
    x0, x1 = x.tolist()
    return [[sin(x0), -19600 / (x1 ** 3)], [-x1, -x0]]

result2 = fsolve(f, [1, 1], fprime=j)
print(result2)
print(f(result2))
```

Result:

```
# 直接求解
[ 1.5940638   97.86308398]
# 误差
[4.596323321948148e-14, 2.76543232757831e-11]
# 加入 Jacobbian 矩阵
[ 1.5940638   97.86308398]
# 误差
[-1.4652723479002816e-12, -9.947598300641403e-13]
```

2、用 curve_fit()函数对高斯分布进行拟合，x∊[0,10],高斯分布函数为 y=a*np.exp(-(x-b)**2/(2*c**2)) ，其中真实值 a=1,b=5,c=2。试对 y 加入噪声之后进行拟合，并作图与真实数据进行比较。（参见课件 leastsq(),curve_fit()拟合）

Code:

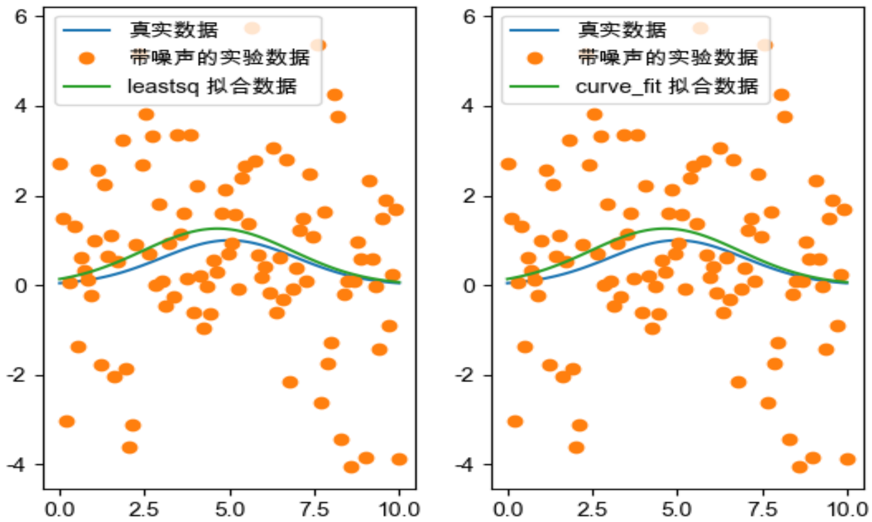
```
import numpy as np
```

```python
from scipy import optimize
import matplotlib.pyplot as plt

def func(x, p):
    a, b, c = p
    return a * np.exp(-(x - b) ** 2 / (2 * c ** 2))

def residuals(p, y, x):
    return y - func(x, p)

def func2(x, a, b, c):
    return a * np.exp(-(x - b) ** 2 / (2 * c ** 2))

x = np.linspace(0, 10, 100)
a, b, c = 1, 5, 2
y0 = func(x, [a, b, c])
np.random.seed(10)
y1 = y0 + 2 * np.random.randn(len(x))
p0 = [1, 1, 1]

# 使用 curve_fit
popt, pcov = optimize.curve_fit(func2, x, y1, p0=p0)
print("真实参数:", [a, b, c])
print("拟合参数", popt)

# 使用 leastsq
plsq = optimize.leastsq(residuals, p0, args=(y1, x))
print("真实参数:", [a, b, c])
print("拟合参数:", plsq[0])  # 实验数据拟合后的参数

plt.rcParams['font.sans-serif'] = ['Arial Unicode MS']
plt.rcParams['axes.unicode_minus'] = False

fig, ax = plt.subplots(1, 2)
ax[0].plot(x, y0)
ax[0].plot(x, y1, "o")
ax[0].plot(x, func(x, plsq[0]))
ax[0].legend(["真实数据", "带噪声的实验数据", "leastsq 拟合数据"])
ax[1].plot(x, y0)
ax[1].plot(x, y1, "o")
ax[1].plot(x, func(x, popt))
ax[1].legend(["真实数据", "带噪声的实验数据", "curve_fit 拟合数据"])
```

```
    plt.show()
```

Result:

```
#使用 curve_fit
真实参数: [1, 5, 2]
拟合参数 [1.26202731 4.6472208  2.2033595 ]
# 使用 leastsq
真实参数: [1, 5, 2]
拟合参数: [1.26202731 4.6472208  2.2033595 ]
```



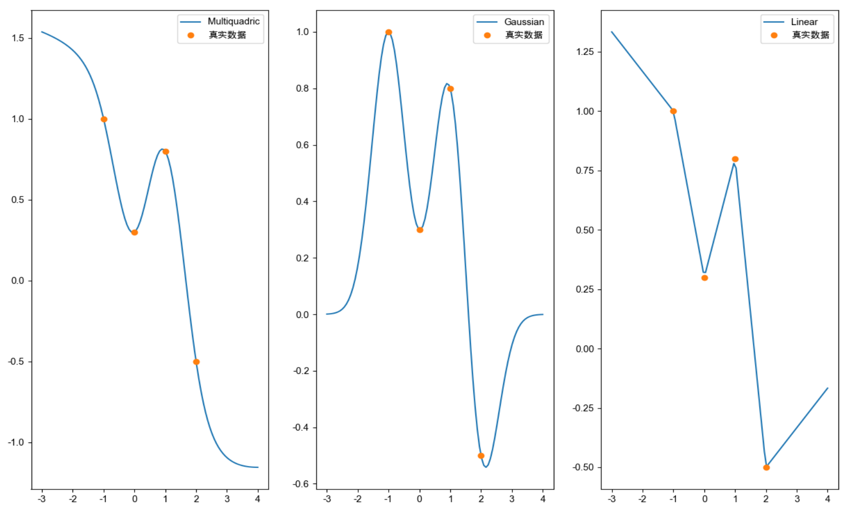
3、对 4 个数据点 x = [-1, 0, 2.0, 1.0]，y = [1.0, 0.3, -0.5, 0.8]进行 Rbf 插值，插值中使用三种插值方法分别是 multiquadric、gaussian、和 linear(参见课件 5,scipy_rbf.py),需要作点图(加密点)为 np.linspace(-3, 4, 100)。

Code:

```
    import numpy as np
    from scipy import interpolate
    import matplotlib.pyplot as plt
    x = [-1, 0, 2.0, 1.0]
    y = [1.0, 0.3, -0.5, 0.8]
    x_inter = np.linspace(-3, 4, 100)
    newfunc1 = interpolate.Rbf(x,y,function="multiquadric")
    y_multi = newfunc1(x_inter)

    newfunc2 = interpolate.Rbf(x, y, function='gaussian')
    y_gaussian = newfunc2(x_inter)

    newfunc3 = interpolate.Rbf(x, y, function='linear')
    y_linear = newfunc3(x_inter)
```

```
plt.rcParams['font.sans-serif'] = ['Arial Unicode MS']
plt.rcParams['axes.unicode_minus'] = False

fig, ax = plt.subplots(1,3,figsize=(15,10))
ax[0].plot(x_inter,y_multi)
ax[0].plot(x,y,'o')
ax[0].legend(['Multiquadric', '真实数据'])
ax[1].plot(x_inter,y_gaussian)
ax[1].plot(x,y,'o')
ax[1].legend(['Gaussian', '真实数据'])
ax[2].plot(x_inter,y_linear)
ax[2].plot(x,y,'o')
ax[2].legend(['Linear', '真实数据'])
plt.show()
```

Result:



4、分别用 optimize.fmin_bfgs、optimize.fminbound、optimize.brute 三种优化方法对函数 x**2 + 10 * np.sin(x)求最小值，并作图。x∈[-10, 10].

Code:

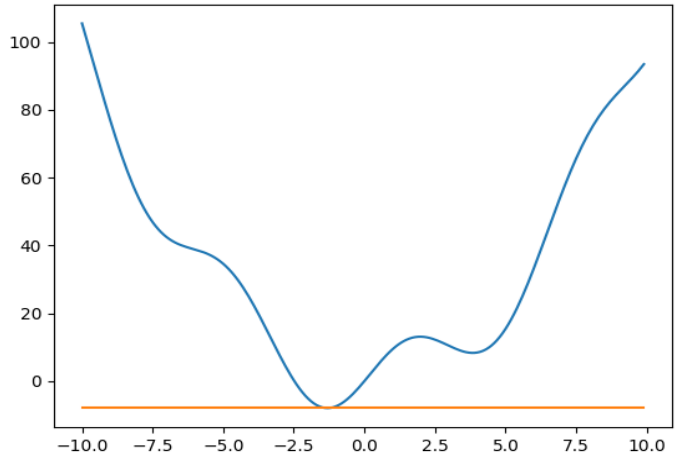
```
import numpy as np

from scipy import optimize

import matplotlib.pyplot as plt
```

```
def func(x):

    return x ** 2 + 10 * np.sin(x)



# fmin_bfgs 方法

y_bfgs = optimize.fmin_bfgs(func, [-1])

print(y_bfgs, func(y_bfgs))

# optimize.fminbound 方法

y_minbound = optimize.fminbound(func, -10, 10)

print(y_minbound, func(y_minbound))

# optimize.brute 方法

y_brute = optimize.brute(func, (slice(-10, 10, 0.01),))

print(y_brute, func(y_brute))

x = np.arange(-10, 10, 0.1)

y = func(x)

plt.plot(x, y)

plt.plot(x, func(y_bfgs) * np.ones(x.shape))

plt.show()
```

Result:

```
# 使用 fmin_bfgs 方法

Optimization terminated successfully.

         Current function value: -7.945823

         Iterations: 3

         Function evaluations: 10

         Gradient evaluations: 5
```

```
[-1.30643969] [-7.94582338]

# 使用 fminbound 方法

-1.306440096615395 -7.945823375615237

# 使用 brute 方法

[-1.30641797] [-7.94582337]
```



5、计算积分 a) $\int_0^3 \cos^2(e^x)dx$， b) $\int_0^{1/2} dy \int_0^{\sqrt{1-4y^2}} 16xydx$.

Code:

```
import numpy as np

from scipy import integrate



def func1(x):

    return (np.cos(np.exp(x))) ** 2



def func2(x, y):

    return 16 * x * y
```

```
result1 = integrate.quad(func1, 0, 3)

result2 = integrate.dblquad(func2, 0, 0.5, 0, lambda y: np.sqrt(1 - 4 *
y ** 2))

print(result1)

print(result2)
```

Result:

```
# 问题 a)的积分与误差
(1.296467785724373, 1.397797106902389e-09)
# 问题 b)的积分与误差
(0.49999999999999994, 1.7092350012594845e-14)
```

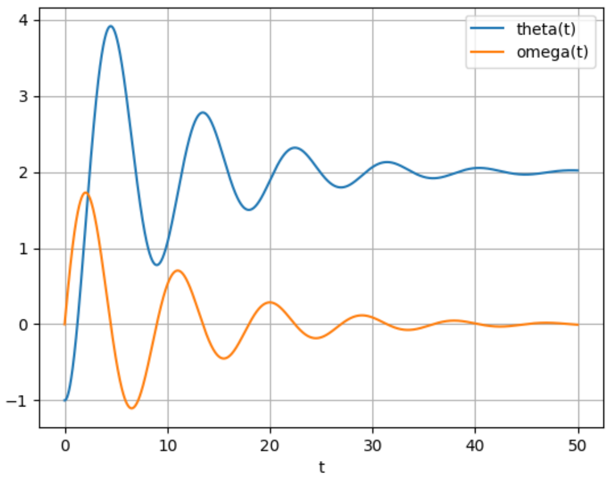
6、弹簧系统每隔 1ms 周期的系统状态 $Mx'' + bx' + kx = F$，试用 odeint()对该系统进行求解并作图，其中参数 M, k, b, F = 1.0, 0.5, 0.2, 1.0;初值 init_status = -1, 0.0; t = np.arange(0, 50, 0.02)。

Code:

```
import matplotlib.pyplot as plt



def func(x, t, M, k, b, F):

    theta, omega = x

    # theta'(t) = omega

    dxdt = [omega, (F - b * omega - k * theta) / M]

    return dxdt



M, k, b, F = 1.0, 0.5, 0.2, 1.0

init_status = -1, 0.0

t = np.arange(0, 50, 0.02)

sol = integrate.odeint(func, init_status, t, args=(M, k, b, F))

plt.plot(t, sol[:, 0])
```

```
plt.plot(t, sol[:, 1])
plt.legend(["theta(t)", "omega(t)"])
plt.xlabel('t')
plt.grid()
plt.show()
```

Result:



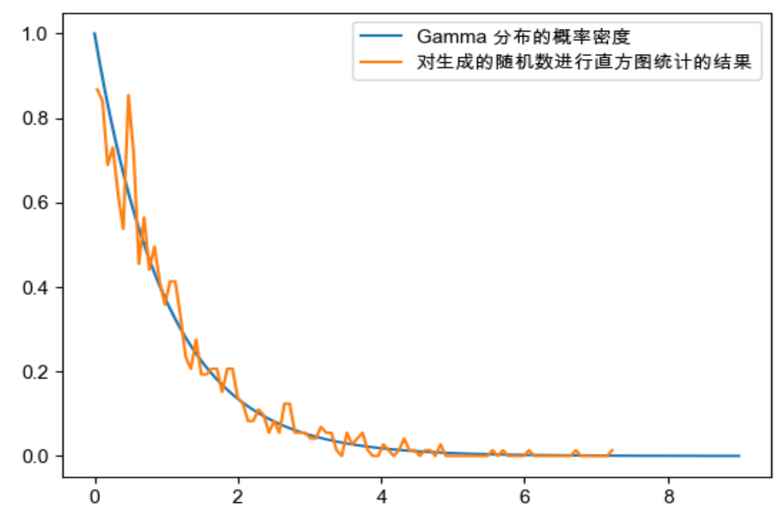
7、从参数为 1 的伽马分布生成 1000 个随机数,然后绘制这些样点的直方图。你能够在其上绘制此伽马分布的 pdf 吗(应该匹配)？(参见课件)

Code:

```
from scipy import stats
import numpy as np
import matplotlib.pyplot as plt


X = stats.gamma(1)
x0 = X.rvs(size=1000)
```

```
t = np.arange(0, 9, 0.01)


plt.rcParams['font.sans-serif'] = ['Arial Unicode MS']

plt.rcParams['axes.unicode_minus'] = False

fig, ax = plt.subplots(1, 1)

ax.plot(t, X.pdf(t))

p, t2 = np.histogram(x0, bins=100, normed=True)

t2 = (t2[:-1] + t2[1:]) / 2

ax.plot(t2, p)

ax.legend(["Gamma 分布的概率密度", "对生成的随机数进行直方图统计的结果"])

plt.show()
```

Result:



8、scipy.sparse 中提供了多种表示稀疏矩阵的格式，试用 dok_martix，lil_matrix 表示表示的矩阵[[3 0 8 0] [0 2 0 0] [0 0 0 0] [0 0 0 1]]，并与 sparse.coo_matrix 表示法进行比较。

Code:

```
from scipy import sparse
```

```
A_dok = sparse.dok_matrix([[3, 0, 8, 0], [0, 2, 0, 0], [0, 0, 0, 0], [0, 0, 0, 1]])

A_lil = sparse.lil_matrix([[3, 0, 8, 0], [0, 2, 0, 0], [0, 0, 0, 0], [0, 0, 0, 1]])

A_coo = sparse.coo_matrix([[3, 0, 8, 0], [0, 2, 0, 0], [0, 0, 0, 0], [0, 0, 0, 1]])



print("dok_matrix:\n", A_dok)

print("lil_matrix:\n", A_lil)

print("coo_matrix:\n", A_coo)
```

Result:

```
dok_matrix:

   (0, 0)	3

  (0, 2)	8

  (1, 1)	2

  (3, 3)	1

lil_matrix:

   (0, 0)	3

  (0, 2)	8

  (1, 1)	2

  (3, 3)	1

coo_matrix:

   (0, 0)	3

  (0, 2)	8
```

(1， 1)	2
(3， 3)	1