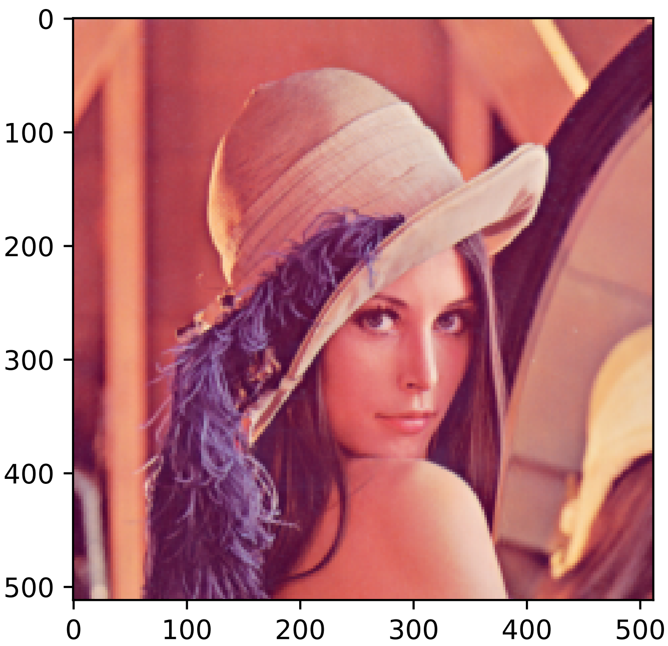
作业目的：理解图像灰度变换的基本原理，掌握灰度变换的实现方法

作业内容：选取一张灰度图像，用给定的变换函数对图像进行灰度变换，对比灰度变换效果

In [133…
```
#导入包
import numpy as np
import matplotlib.pyplot as plt
import math


#选择图像
lenna = plt.imread("images/lenna.jpg") # 加载当前文件夹的图片
plt.imshow(lenna)
```

Out[133…
```
<matplotlib.image.AxesImage at 0x126da47c0>
```



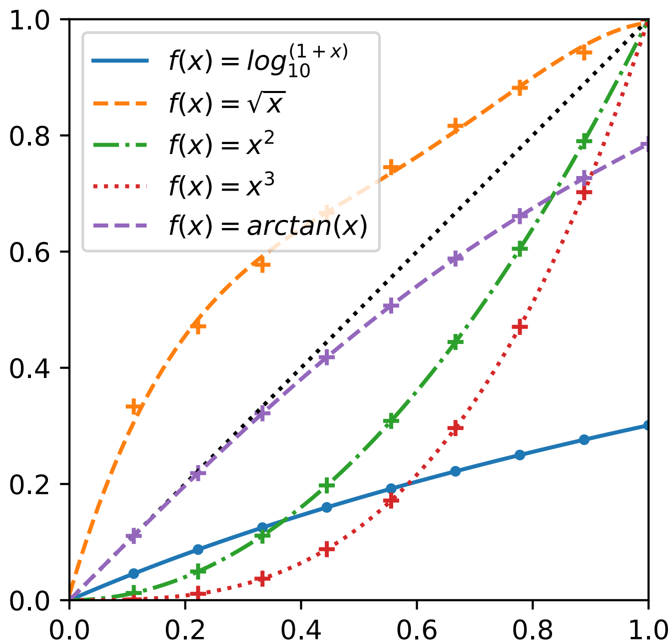
作业要求：（1）在（0，1）范围内随机设定5对(x,y)值（包含(0,0)和（1，1））,并利用其构建4阶拟合多项式 （2）利用拟合多项式曲线作为变换函数对灰度图像进行处理，给出处理结果

作业提交：（1）PDF格式 （2）绘制离散点及拟合的多项式曲线，同时给原始图像和用该曲线对应函数进行灰度变换的输出图像 （3）从数学角度推导多项式拟合参数计算方法，利用Numpy线性代数工具箱求解多项式系数，与Numpy的polyfit函数的计算结果进行对比，看看两者是否一致。 （4）给出算法实现的Python代码

In [134…
```
# （2）绘制离散点及拟合的多项式曲线，同时给原始图像和用该曲线对应函数进行灰度变换的输出图像

# 模拟生成10个离散的点
xp = np.linspace(0, 1, 10)
y1 = np.log10(1+xp)      #像素值取对数
y2 = xp **2              #像素值二次方
y3 = xp ** 3             #像素值三次方
y4 = np.sqrt(xp)         #像素值开方
y5 = np.arctan(xp)         #像素值
print(y1)



# 通过多项式拟合得到拟合曲线
z_1 = np.poly1d(np.polyfit(xp, y1, 4))
z_2 = np.poly1d(np.polyfit(xp, y2, 4))
z_3 = np.poly1d(np.polyfit(xp, y3, 4))
```

```python
z_4 = np.poly1d(np.polyfit(xp, y4, 4))
z_5 = np.poly1d(np.polyfit(xp, y5, 4))

# 绘制拟合曲线
# 这里也展示了显示单张图片的一种方法
fig = plt.figure()
ax = fig.add_subplot(111)
ax.set_aspect('equal')
x = np.linspace(0, 1, 100)
ax.plot(x,x,linestyle=':',color = 'black')
ax.scatter(xp, y1, marker='.')
ax.plot(x, z_1(x),linestyle='-', label='$f(x)=log_{10}^{(1+x)}$')
ax.scatter(xp,y4, marker='+')
ax.plot(x, z_4(x),linestyle='--',label='$f(x)=\sqrt{x}$')
ax.scatter(xp,y2, marker='+')
ax.plot(x, z_2(x),linestyle='-.',label='$f(x)=x^2$')
ax.scatter(xp,y3, marker='+')
ax.plot(x, z_3(x),linestyle=':',label='$f(x)=x^3$')
ax.scatter(xp,y5, marker='+')
ax.plot(x, z_5(x),linestyle='dashed',label='$f(x)=arctan(x)$')
plt.ylim(0,1)
plt.xlim(0,1)
plt.legend()
plt.show()
```

```
[0.         0.04575749 0.08715018 0.12493874 0.15970084 0.19188553
 0.22184875 0.24987747 0.27620641 0.30103    ]
```



In [135…

```python
# 根据拟合曲线构建映射表
map_1 = np.zeros(256)
map_2 = np.zeros(256)
map_3 = np.zeros(256)
map_4 = np.zeros(256)
map_5 = np.zeros(256)
x = np.linspace(0, 1, 256)
for i in np.arange(256):
    temp = z_1(x[i])
    if temp > 1:
        map_1[i] = 1
    elif temp < 0:
        map_1[i] = 0
    else:
        map_1[i] = temp
```

```
for i in np.arange(256):
    temp = z_2(x[i])
    if temp > 1:
        map_2[i] = 1
    elif z_2(x[i]) < 0:
        map_2[i] = 0
    else:
        map_2[i] = temp
for i in np.arange(256):
    temp = z_3(x[i])
    if temp > 1:
        map_3[i] = 1
    elif z_3(x[i]) < 0:
        map_3[i] = 0
    else:
        map_3[i] = temp
for i in np.arange(256):
    temp = z_4(x[i])
    if temp > 1:
        map_4[i] = 1
    elif z_3(x[i]) < 0:
        map_4[i] = 0
    else:
        map_4[i] = temp
for i in np.arange(256):
    temp = z_5(x[i])
    if temp > 1:
        map_5[i] = 1
    elif z_5(x[i]) < 0:
        map_5[i] = 0
    else:
        map_5[i] = temp
```

In [136…

```
def f_1(x):
    return map_1[x]
def f_2(x):
    return map_2[x]
def f_3(x):
    return map_3[x]
def f_4(x):
    return map_4[x]
def f_5(x):
    return map_5[x]
```

In [137…

```
# f_1仅对单个标量进行操作，frompyfunc函数让其支持对图像每个像素的处理

im_1 = np.frompyfunc(f_1,1,1)(lenna).astype(np.float)
im_2 = np.frompyfunc(f_2,1,1)(lenna).astype(np.float)
im_3 = np.frompyfunc(f_3,1,1)(lenna).astype(np.float)
im_4 = np.frompyfunc(f_4,1,1)(lenna).astype(np.float)
im_4 = np.frompyfunc(f_4,1,1)(lenna).astype(np.float)
```

In [138…

```
plt.subplot(161)
plt.imshow(lenna)
plt.axis('off')
plt.title('Lenna_Orig')
plt.subplot(162)
plt.imshow(im_1)
plt.axis('off')
plt.title('$log^{(1+x)}$')
plt.subplot(163)
plt.imshow(im_4)
plt.axis('off')
plt.title('$\sqrt{x}$')
```

```python
plt.subplot(164)
plt.imshow(im_2)
plt.axis('off')
plt.title('$x^2$')
plt.subplot(165)
plt.imshow(im_3)
plt.axis('off')
plt.title('$x^3$')
plt.subplot(166)
plt.imshow(im_5)
plt.axis('off')
plt.title('$arctan(x)$')
plt.show()
```

In [139…

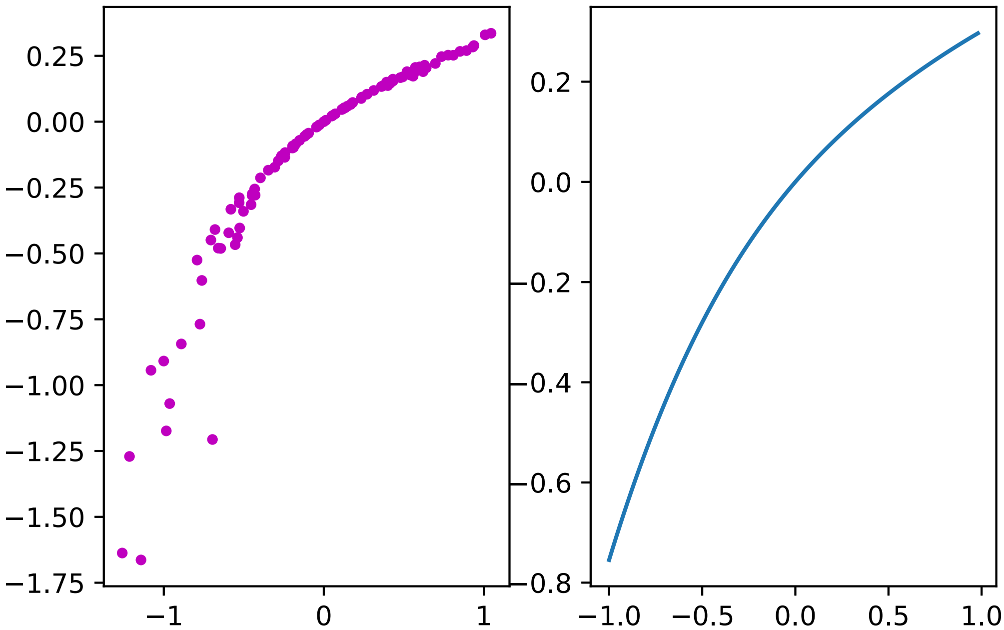
```python
# (3) 从数学角度推导多项式拟合参数计算方法，利用Numpy线性代数工具箱求解多项式系数，与Numpy

#Numpy的polyfit结果
print("z_1:\n",z_1,"\n")
print("z_2:\n",z_2,"\n")
print("z_3:\n",z_3,"\n")
print("z_4:\n",z_4,"\n")
print("z_5:\n",z_5,"\n")


# y1 = np.log10(1+xp)
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import random

fig = plt.figure()
ax = fig.add_subplot(121)

#阶数为9阶
order=9

#生成曲线上的各个点
x = np.arange(-1,1,0.02)
y = [np.log10(1+a) for a in x]

# 生成的曲线上的各个点偏移一下，并放入到xa,ya中去
i = 0
xa = []
ya = []
for xx in x:
    yy = y[i]
    d = float(random.randint(60, 140)) / 100
    # ax.plot([xx*d],[yy*d],color='m',linestyle='',marker='.')
    i += 1
    xa.append(xx * d)
    ya.append(yy * d)

'''''for i in range(0,5):
    xx=float(random.randint(-100,100))/100
    yy=float(random.randint(-60,60))/100
    xa.append(xx)
```

```
    ya.append(yy)'''

ax.plot(xa, ya, color='m', linestyle='', marker='.')

# 求出等式左边的矩阵A

matA=[]
for i in range(0,order+1):
    mat=[]
    for j in range(0+i,order+1+i):
        sumA=0
        for xx in xa:
            sumA=sumA+xx**j
        mat.append(sumA)
    matA.append(mat)
A=np.array(matA)

# 求出右边的等式B
matB=[]
for j in range(0,order+1):
    sumB=0
    for xx,yy in zip(xa,ya):
        sumB=sumB+xx**j*yy
    matB.append(sumB)
B=np.array(matB)
# 另外一种该方法求A
# 求出等式左边的矩阵A
A=[]
for xx in xa:
    matA = []
    for i in range(0,order+1):
        mat = []
        for j in range(0+i,order+1+i):
            mat.append(xx**j)
        matA.append(mat)
    A.append(matA)
# 求和
A=sum(np.array(A))

a=np.linalg.solve(A,B)
# 定义拟合函数
def fun_solve(x,a):
    y=0
    for i in range(len(a)):
        y+=a[i]*x**i
    return y

xxa= np.arange(-1,1.06,0.01)
yya=[]
for xxaa in xxa:
    yya.append(fun_solve(xxaa,a))


ax2 = fig.add_subplot(122)

ax2.plot(x, z_1(x),linestyle='-', label='$f(x)=log_{10}^{(1+x)}$')


ax.plot(xxa,yya,color='g',linestyle='-',marker='')
```

```
z_1:
            4          3          2
-0.02411 x + 0.0951 x - 0.2027 x + 0.4328 x + 1.436e-05

z_2:
              4                 3       2
```

```
-3.967e-15 x + 9.573e-15 x + 1 x + 2.575e-15 x - 3.774e-16

z_3:
           4      3              2
2.9e-15 x + 1 x + 2.324e-15 x - 6.511e-16 x - 1.931e-16

z_4:
         4          3          2
-3.33 x + 8.181 x - 7.241 x + 3.374 x + 0.009951

z_5:
          4           3            2
0.1401 x - 0.3425 x - 0.01506 x + 1.003 x - 4.859e-05
```

Out[139… [<matplotlib.lines.Line2D at 0x12890aee0>]



由图像观察发现在[-1,-0.5]部分拟合的不是很好但是[0,1]处则点比较密集,相对于Numpy的polyfit函数还是做的不够好，这方法使用的是最小二乘法多项式曲线拟合原理

作业提示：建设离散点数位N，拟合多项式阶次N-1，求解多项式系数过程就是解线性组问题