第四周上机

一、问题叙述

考虑区间[0,1]上的函数：

$$f(x)=\frac{1}{(x-0.3)^2+0.01}+\frac{1}{(x-0.9)^2+0.04}-6$$

以 $x_i=\frac{i}{n}$， $i=0,1,2,\cdots,n$ 为插值节点，取不同的分点数目n=4、8、12，分别进行Newton插值、分段线性插值，并画出原函数$f(x)$及插值多项式函数在[0,1]上的图像。

二、Newton 插值

代码如下：

```
n = 4

def f(x_1):
    return 1 / ((x_1-0.3)**2 + 0.01) + 1 / ((x_1-0.9)**2 + 0.04) - 6

#draw the original figure
interval = 0.01
x_0 = np.arange(0, 1 + interval, interval)
y_0 = []
for i in x_0:
    y_0.append(f(i))

#Newton
table = np.zeros((n+1, n+1))
x_1 = np.arange(0, 1 + 1/n, 1/n)

#create difference quotient table
y = []
for i in x_1:
    y.append(f(i))

for i in range(n+1):
    table[i][0] = y[i]
for i in range(1, n+1):
    for j in range(i, n+1):
        table[j][i] = (table[j][i-1] - table[j-1][i-1]) / (x_1[j] - x_1[j-i])
```

```python
#store coefficients
a = np.diagonal(table)

#define Newton interpolation polynomial
def Newton(x, x_1, n, a):
    sum = a[0]
    for i in range(1, n+1):
        product = 1
        for j in range(0, i):
            product *= x - x_1[j]
        sum += a[i] * product
    return sum

y_1 = []
for i in x_0:
    y_1.append(Newton(i, x_1, n, a))

#piecewise linear interpolation
y_2 = []
for i in x_1:
    y_2.append(Newton(i, x_1, n, a))
```

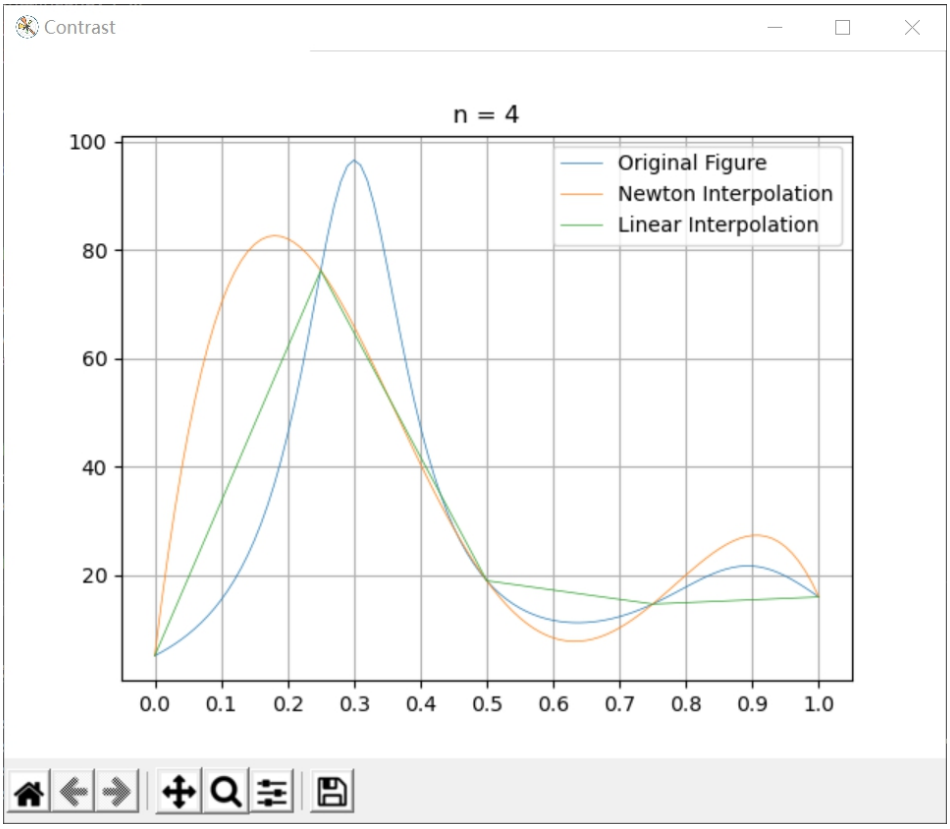
三、分段线性插值

由于该方法较为简单，故在已有的牛顿插值的代码基础上进行添加：

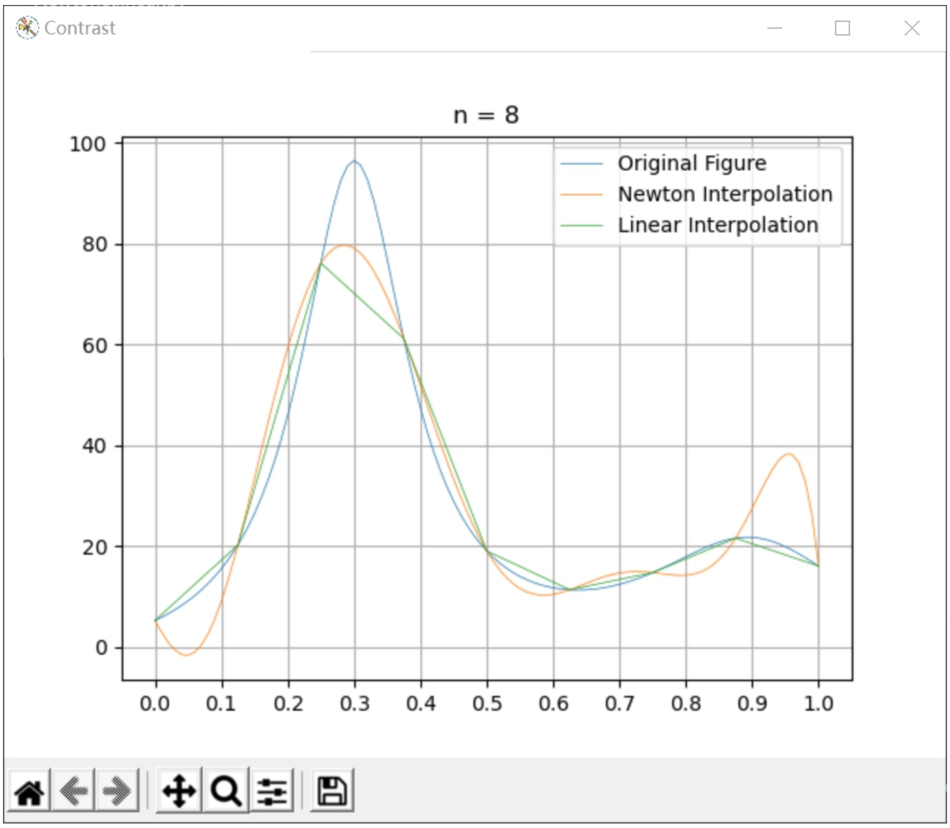
```python
#piecewise linear interpolation
y_2 = []
for i in x_1:
    y_2.append(f(i))
```

四、结果分析

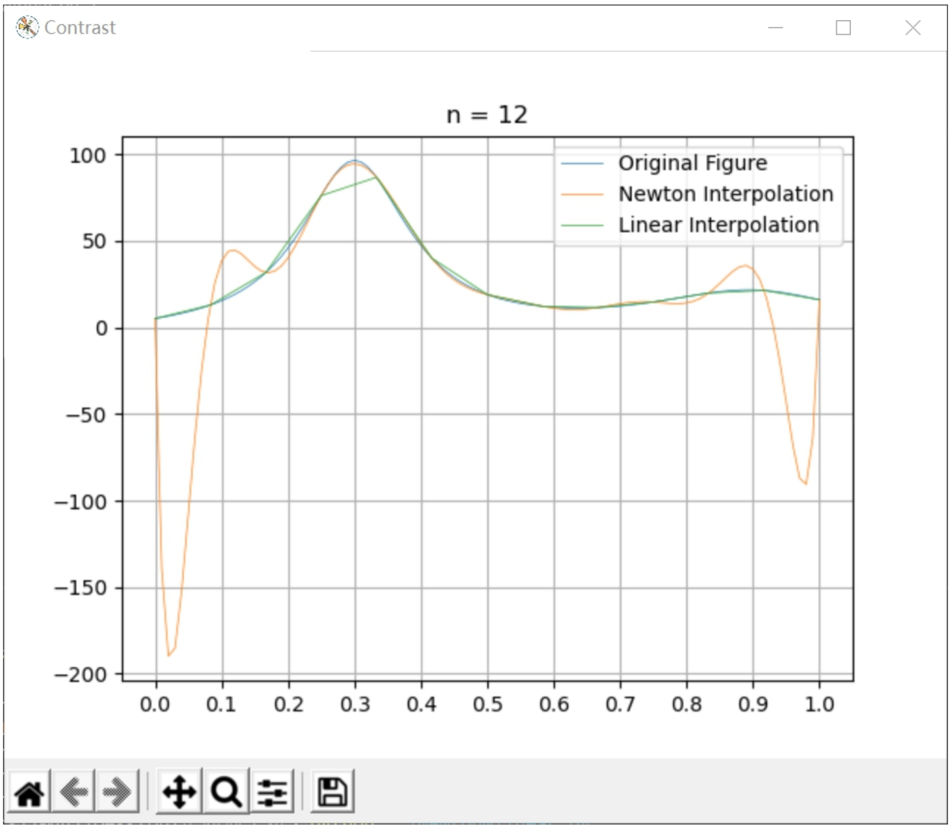
1、N = 4


Contrast
n = 4
Original Figure
Newton Interpolation
Linear Interpolation
100
80
60
40
20
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

2、N = 8


Contrast
n = 8
Original Figure
Newton Interpolation
Linear Interpolation
100
80
60
40
20
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

3、N = 12



随着 N 的增大，分段线性插值的拟合度越来越好，而 Newton 插值法在 N = 8 时效果最好，若 N 继续变大则会出现 Runge 现象，端点附近的抖动很大。