实　验　报　告

实验地点		学生姓名	
实验日期	年　月　日　第　节	学院	
实验课程		学号	
实验项目		成绩	

一、实验目的或要求

1、编写牛顿插值算法程序

2、找出创新点

二、实验过程记录

1、编写牛顿插值程序

①主要代码

A、差商函数

```
def diff_quot(X,Y):
    '''
    n阶差商计算
    输入：n个差分节点X，Y
    输出：标量，差分值
    '''
    X = np.array(X)
    Y = np.array(Y)
    n = len(X)
    f = np.ones(n)
    for j in range(n):
        f[j] = f[j]*Y[j]
        for jj in range(n):
            if jj != j:
                f[j] = f[j]/(X[j]-X[jj])
    F = f.sum()
    return F
```

B、牛顿法

```python
def Newton_inter(x,X,Y):
    X = np.array(X)
    Y = np.array(Y)
    x = np.array(x)
    n = len(X)
    nx = len(x)
    y = np.ones(nx) # 保存x的插值
    for k in range(nx):

        N = np.ones(n)
        m = 0
        for i in range(n):
            for mm in range(m):
                N[i] = N[i]*(x[k]-X[mm])
            N[i] = N[i]*diff_quot(X[:m+1],Y[:m+1])
            m = m+1

        y[k] = N.sum()
    return y
```

C、可在原来基础上添加点的牛顿法函数

```python
def Append_Newton_inter(x,X,Y,y_old=False,old_num=0):
    X = np.array(X)
    Y = np.array(Y)
    x = np.array(x)
    n = len(X)
    nx = len(x)
    if type(y_old) == bool:
        y_old = np.zeros(nx)
    else:
        y_old = np.array(y_old)
    y = np.zeros(nx)
    for k in range(nx):
        m = old_num
        N = np.ones(n-old_num)
        for i in range(n-old_num):
            for mm in range(m):
                N[i] = N[i]*(x[k]-X[mm])
            N[i] = N[i]*diff_quot(X[:m+1],Y[:m+1])
            m = m+1

        y[k] = y_old[k]+N.sum()
    return y
```

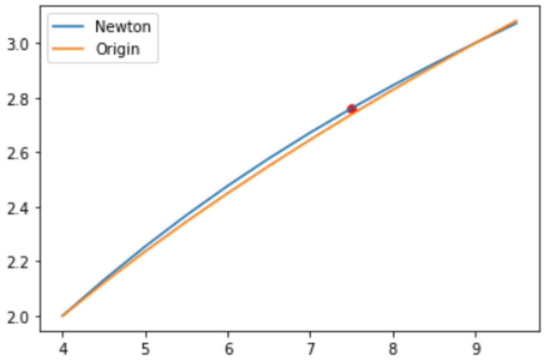
②实验结果

```
[5]: x = np.arange(4,10,step=0.5)
     y = Newton_inter(x,[1,4,9,16],[1,2,3,4])
```

```
[6]: y
```

```
[6]: array([2.        , 2.13125   , 2.25396825, 2.36875   , 2.47619048,
            2.57688492, 2.67142857, 2.76041667, 2.84444444, 2.92410714,
            3.        , 3.07271825])
```

```
[64]: plt.plot(x,y,label='Newton')
      plt.plot(x,np.sqrt(x),label='Origin')
      plt.scatter(x[7],y[7],s=30.,c='r')
      plt.legend(loc='best')
      plt.show()
```



2、找出创新点

本实验报告构思写出了一个可以在给定插值区间、待插值点不变的情况下，不断在插值区间增加插值点直到满足精度的条件时返回待插值点值与插值点数。

其中判断插值函数与原函数误差的误差值计算公式应为：

$$LOSS = \frac{(y_{real} - y)^2}{length(y)}$$

其中， y_{real} 是指真实的插值， y 是指插值函数计算出来的值， $length(y)$ 表示插值点的个数。

①代码

```
def mini_enough(x, loss, lb, ub,fun):
    N = 2 # 插值点数
    X = np.random.uniform(lb,ub,N)
    Y = fun(X)
    y_real = fun(x)
    y = Append_Newton_inter(x,X,Y)
    LOSS = np.sum((y_real-y)**2)/len(y)
    print(LOSS)
    K = loss
    while LOSS > K:
        N = N + 1
        X = np.append(X,np.random.uniform(lb,ub,1))
        Y = fun(X)
        y = Append_Newton_inter(x,X,Y,y,N-1)
        LOSS =  np.sum((y_real-y)**2)/len(y)
        print(LOSS)
    return {'y':y,'N':N}
```

②实验结果

```
[10]: def fun(x):
          return np.log(x)
      x = np.array([1,2,3,4,5,6])
      loss = 0.1
      lb = 1
      ub = 5

[11]: out = mini_enough(x,loss,lb,ub,fun)

      0.27544257186919324
      0.6976864649720308
      0.1626056236105244
      0.031096938681162298

[14]: out['y']

[14]: array([0.00821371, 0.69517992, 1.09698713, 1.38591417, 1.54435658,
             1.36482661])

[13]: out['N']

[13]: 5
```

如上图，定义函数为 $y=\ln(x)$，带入函数得到迭代到第 5 次时精度达到要求的 0.1，此时

$\ln(i), i=1,2,3,...,6$ 的预测值为 $\left[0.0082, 0.6951, 1.0970, 1.3859, 1.5444, 1.3648\right]$。

三、实验结果报告及总结

牛顿法便于在原来点基础上添加点，节省了大量的再插值时间。

牛顿法与拉格朗日插值法实质上是一个函数的不同表达形式，牛顿法的表达形式更便于添加节点，避免重复计算。

实验结果反思及讨论：

教师对报告的最终评价和意见：   年　　月　　日