习题二

Author:孟群康
Student_number:2022202020095

一、题干

被插函数为$f(x)=\sqrt{x}$，已知其在$x=100,121,144$上的值，利用线性插值，二次插值求$\sqrt{115}$的值，并计算结果的误差限和有效数字

二、线性插值法

由于有三个已知点，因此线性插值可以取100，121；100，144；121，144三种排列组合的情况来进行

2.1 选取100，121的情况

In [1]:
```
Discrete_x_point=[100,121]
```

计算出对应点的函数值，作为已知离散点进行插值

In [2]:
```
def origin_function(x):
    return x**(0.5)
```

In [3]:
```
Discrete_y_point=[]
for item in Discrete_x_point:
    Discrete_y_point.append(origin_function(item))
for i in range(len(Discrete_x_point)):
    print("f(%.4f)=%.4f"%(Discrete_x_point[i],Discrete_y_point[i]),sep='')
```

```
f(100.0000)=10.0000
f(121.0000)=11.0000
```

利用2个函数值，可以计算出线性插值多项式，写出任意拉格朗日多项式插值公式函数

In [4]:
```
def Pnx(x,y,x_input):
    if(len(x)!=len(y)):
        raise Exception("len x and len y is NOT EQUAL!")
    Power=0
    for i in range(len(x)):
        multi_item=1
        div_item=1
        for j in range(len(x)):
            multi_item *= (1 if(j==i) else (x_input-x[j]))
            div_item *= (1 if(j==i) else(x[i]-x[j]))
        power = y[i]*multi_item/div_item
        Power+=power

    return Power
```

可以写出这两点确定的表达式的解析形式为

In [5]:
```
import sympy as sp
x = sp.symbols('x')
print("Lagrange interpolation of function is f(x)=")
sp.simplify(Pnx(Discrete_x_point,Discrete_y_point,x))
```

```
Lagrange interpolation of function is f(x)=
```

Out[5]: $0.0476190476190476x + 5.23809523809523$

因此$\sqrt{115}$的线性插值估计，误差限和有效数字分别为：

In [6]:
```
est = Pnx(Discrete_x_point,Discrete_y_point,115)
acc = origin_function(115)
error = est-acc if est>acc else acc-est
# valid_num =
print("sqrt{115}的线性插值估计值是",est)
print("sqrt{115}的精确值是",acc)
print("误差是",error)
print("误差限是0.01")
print("有效数字是4位")
```

```
sqrt{115}的线性插值估计值是 10.714285714285714
sqrt{115}的精确值是 10.723805294763608
误差是 0.009519580477894252
误差限是0.01
有效数字是4位
```

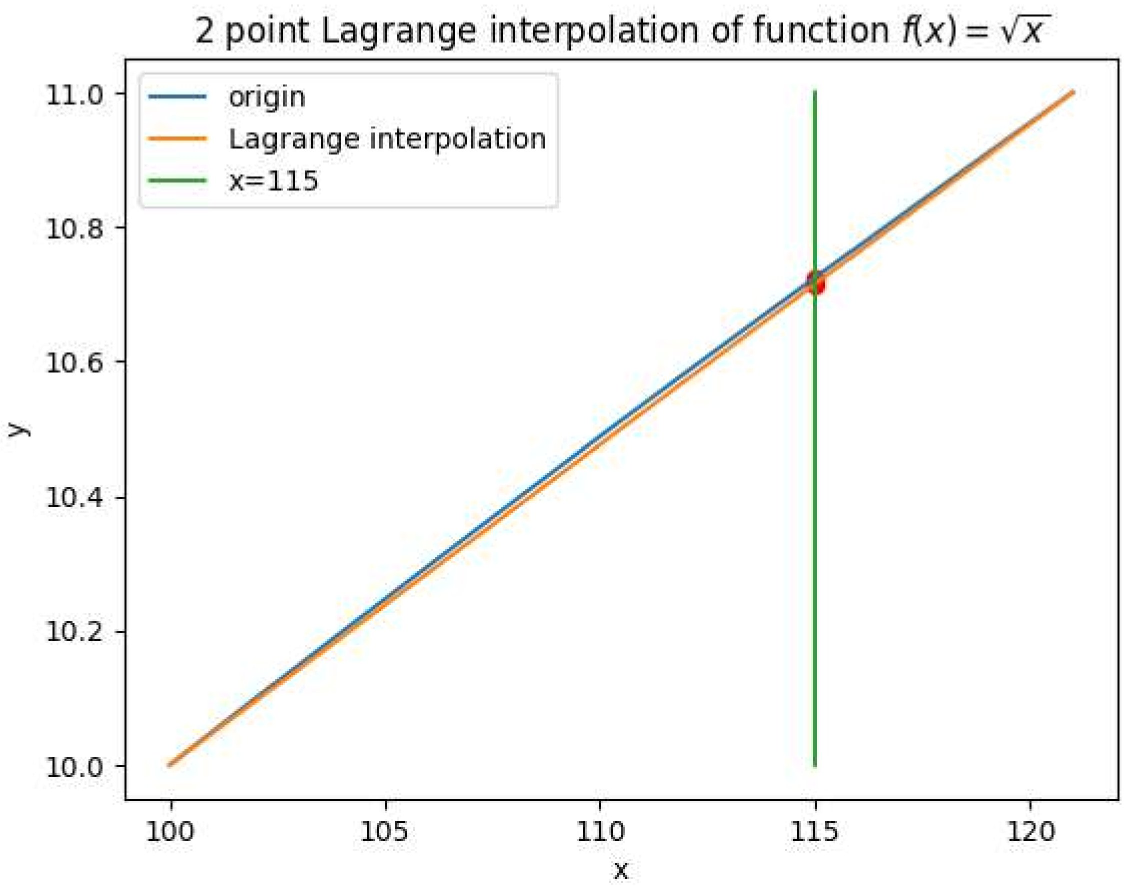
线性插值多项式示意图如下：

In [7]:
```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(100,121,100)
y = origin_function(x)
y_= Pnx(Discrete_x_point,Discrete_y_point,x)
x_disrete_point=115
y_est = est
y_acc = acc
x_line = np.linspace(115,115,100)
y_line = np.linspace(10,11,100)


plt.plot(x,y,label = 'origin')
plt.plot(x,y_,label = 'Lagrange interpolation')
plt.plot(x_line,y_line,label = 'x=115')

plt.scatter(x_disrete_point,y_est,color = 'red')
plt.scatter(x_disrete_point,y_acc,color = 'red')
plt.xlabel("x")
plt.ylabel("y")
plt.title(r"$2$ point Lagrange interpolation of function $f(x)=\sqrt{x}$")
plt.legend(loc = 'upper left')
plt.show()
```



2.2 选取100，144的情况

In [8]:
```
Discrete_x_point=[100,144]
```

计算出对应点的函数值，作为已知离散点进行插值

In [9]:
```
def origin_function(x):
    return x**(0.5)
```

In [10]:
```
Discrete_y_point=[]
for item in Discrete_x_point:
    Discrete_y_point.append(origin_function(item))
for i in range(len(Discrete_x_point)):
    print("f(%.4f)=%.4f"%(Discrete_x_point[i],Discrete_y_point[i]),sep='')
```

```
f(100.0000)=10.0000
f(144.0000)=12.0000
```

利用2个函数值，可以计算出线性插值多项式，写出任意拉格朗日多项式插值公式函数

In [11]:
```
def Pnx(x,y,x_input):
    if(len(x)!=len(y)):
        raise Exception("len x and len y is NOT EQUAL!")
    Power=0
    for i in range(len(x)):
        multi_item=1
        div_item=1
        for j in range(len(x)):
            multi_item *= (1 if(j==i) else (x_input-x[j]))
            div_item *= (1 if(j==i) else(x[i]-x[j]))
```

```
        power = y[i]*multi_item/div_item
        Power+=power

    return Power
```

可以写出这两点确定的表达式的解析形式为

In [12]:
```
import sympy as sp
x = sp.symbols('x')
print("Lagrange interpolation of function is f(x)=")
sp.simplify(Pnx(Discrete_x_point,Discrete_y_point,x))
```

```
Lagrange interpolation of function is f(x)=
```

Out[12]: $0.0454545454545454x + 5.45454545454545$

因此$\sqrt{115}$的线性插值估计，误差限和有效数字分别为：

In [13]:
```
est = Pnx(Discrete_x_point,Discrete_y_point,115)
acc = origin_function(115)
error = est-acc if est>acc else acc-est
# valid_num =
print("sqrt{115}的线性插值估计值是",est)
print("sqrt{115}的精确值是",acc)
print("误差是",error)
print("误差限是0.1")
print("有效数字是3位")
```

```
sqrt{115}的线性插值估计值是 10.681818181818182
sqrt{115}的精确值是 10.723805294763608
误差是 0.04198711294542612
误差限是0.1
有效数字是3位
```

线性插值多项式示意图如下：

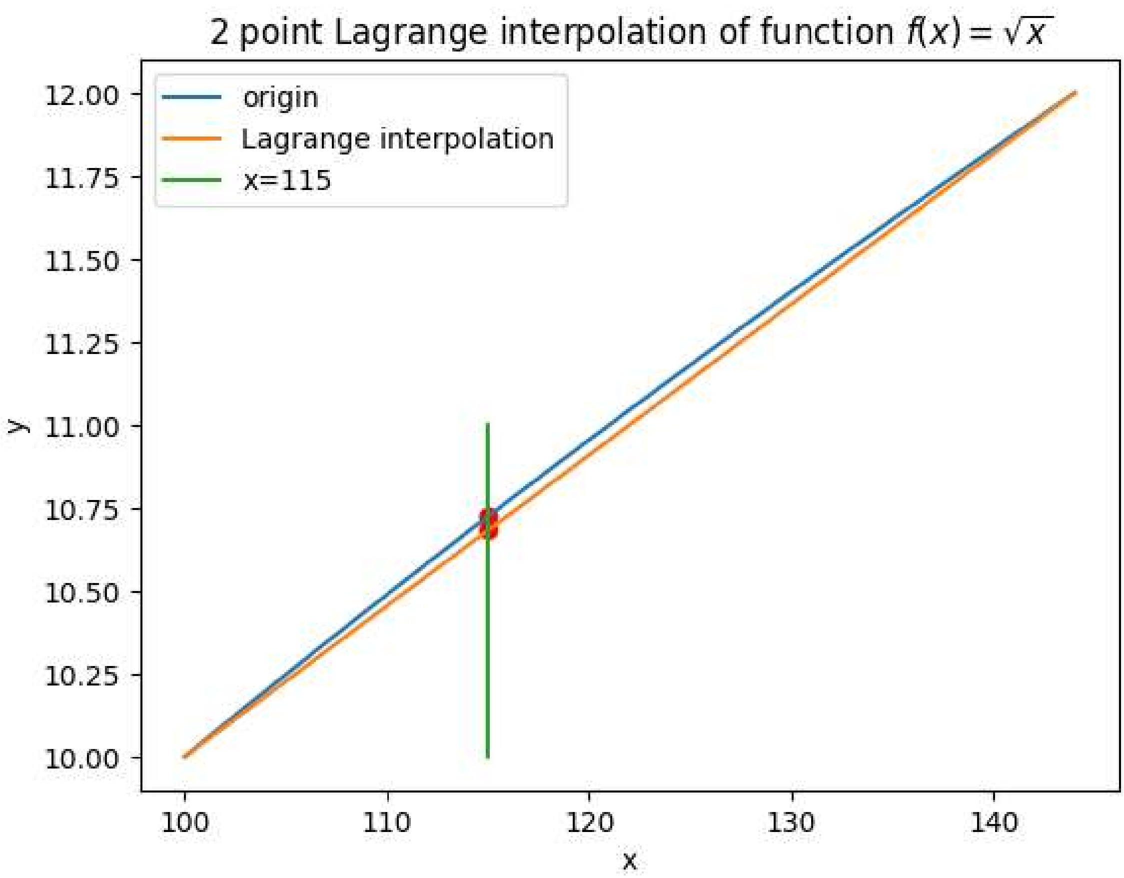
In [14]:
```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(100,144,100)
y = origin_function(x)
y_= Pnx(Discrete_x_point,Discrete_y_point,x)
x_disrete_point=115
y_est = est
y_acc = acc
x_line = np.linspace(115,115,100)
y_line = np.linspace(10,11,100)


plt.plot(x,y,label = 'origin')
plt.plot(x,y_,label = 'Lagrange interpolation')
plt.plot(x_line,y_line,label = 'x=115')

plt.scatter(x_disrete_point,y_est,color = 'red')
plt.scatter(x_disrete_point,y_acc,color = 'red')
plt.xlabel("x")
plt.ylabel("y")
plt.title(r"$2$ point Lagrange interpolation of function $f(x)=\sqrt{x}$")
```

```
plt.legend(loc = 'upper left')
plt.show()
```



2.3 选取121，144的情况

In [15]:
```
Discrete_x_point=[121,144]
```

计算出对应点的函数值，作为已知离散点进行插值

In [16]:
```
def origin_function(x):
    return x**(0.5)
```

In [17]:
```
Discrete_y_point=[]
for item in Discrete_x_point:
    Discrete_y_point.append(origin_function(item))
for i in range(len(Discrete_x_point)):
    print("f(%.4f)=%.4f"%(Discrete_x_point[i],Discrete_y_point[i]),sep='')
```

```
f(121.0000)=11.0000
f(144.0000)=12.0000
```

利用2个函数值，可以计算出线性插值多项式，写出任意拉格朗日多项式插值公式函数

In [18]:
```
def Pnx(x,y,x_input):
    if(len(x)!=len(y)):
        raise Exception("len x and len y is NOT EQUAL!")
    Power=0
    for i in range(len(x)):
        multi_item=1
        div_item=1
        for j in range(len(x)):
```

```
            multi_item *= (1 if(j==i) else (x_input-x[j]))
            div_item *= (1 if(j==i) else(x[i]-x[j]))
        power = y[i]*multi_item/div_item
        Power+=power

    return Power
```

可以写出这两点确定的表达式的解析形式为

In [19]:
```
import sympy as sp
x = sp.symbols('x')
print("Lagrange interpolation of function is f(x)=")
sp.simplify(Pnx(Discrete_x_point,Discrete_y_point,x))
```

```
Lagrange interpolation of function is f(x)=
```

Out[19]: $0.0434782608695652x + 5.7391304347826$

因此$\sqrt{115}$的线性插值估计，误差限和有效数字分别为：

In [20]:
```
est = Pnx(Discrete_x_point,Discrete_y_point,115)
acc = origin_function(115)
error = est-acc if est>acc else acc-est
# valid_num =
print("sqrt{115}的线性插值估计值是",est)
print("sqrt{115}的精确值是",acc)
print("误差是",error)
print("误差限是0.1")
print("有效数字是3位")
```

```
sqrt{115}的线性插值估计值是 10.73913043478261
sqrt{115}的精确值是 10.723805294763608
误差是 0.015325140019001537
误差限是0.1
有效数字是3位
```

线性插值多项式示意图如下：

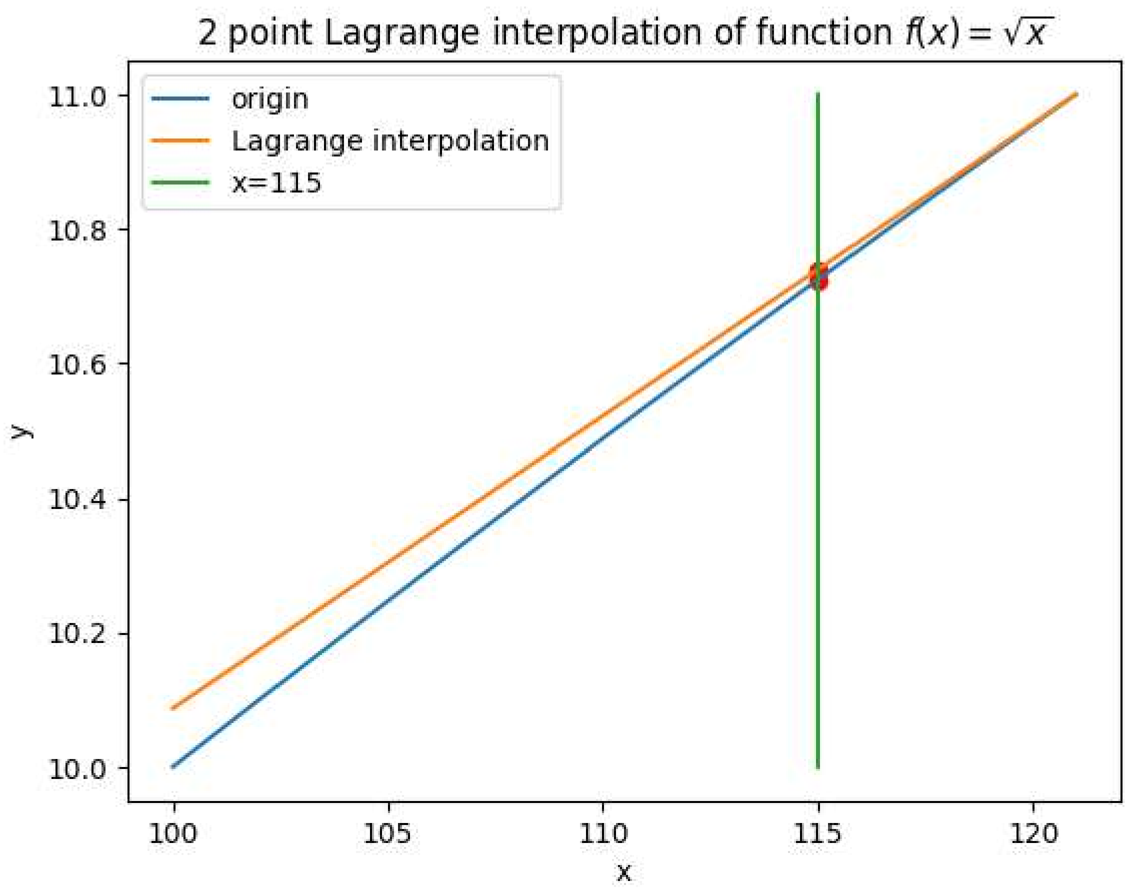
In [21]:
```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(100,121,100)
y = origin_function(x)
y_= Pnx(Discrete_x_point,Discrete_y_point,x)
x_disrete_point=115
y_est = est
y_acc = acc
x_line = np.linspace(115,115,100)
y_line = np.linspace(10,11,100)


plt.plot(x,y,label = 'origin')
plt.plot(x,y_,label = 'Lagrange interpolation')
plt.plot(x_line,y_line,label = 'x=115')

plt.scatter(x_disrete_point,y_est,color = 'red')
plt.scatter(x_disrete_point,y_acc,color = 'red')
plt.xlabel("x")
plt.ylabel("y")
plt.title(r"$2$ point Lagrange interpolation of function $f(x)=\sqrt{x}$")
```

```
plt.legend(loc = 'upper left')
plt.show()
```



三、二次函数插值法

100，121，144都要选取

In [22]:
```
Discrete_x_point=[100,121,144]
```

计算出对应点的函数值，作为已知离散点进行插值

In [23]:
```
def origin_function(x):
    return x**(0.5)
```

In [24]:
```
Discrete_y_point=[]
for item in Discrete_x_point:
    Discrete_y_point.append(origin_function(item))
for i in range(len(Discrete_x_point)):
    print("f(%.4f)=%.4f"%(Discrete_x_point[i],Discrete_y_point[i]),sep='')
```

```
f(100.0000)=10.0000
f(121.0000)=11.0000
f(144.0000)=12.0000
```

利用2个函数值，可以计算出线性插值多项式，写出任意拉格朗日多项式插值公式函数

In [25]:
```
def Pnx(x,y,x_input):
    if(len(x)!=len(y)):
        raise Exception("len x and len y is NOT EQUAL!")
    Power=0
```

```
    for i in range(len(x)):
        multi_item=1
        div_item=1
        for j in range(len(x)):
            multi_item *= (1 if(j==i) else (x_input-x[j]))
            div_item *= (1 if(j==i) else(x[i]-x[j]))
        power = y[i]*multi_item/div_item
        Power+=power

    return Power
```

可以写出这两点确定的表达式的解析形式为

In [26]:
```
import sympy as sp
x = sp.symbols('x')
print("Lagrange interpolation of function is f(x)=")
sp.simplify(Pnx(Discrete_x_point,Discrete_y_point,x))
```

```
Lagrange interpolation of function is f(x)=
```

Out[26]: $-9.41087897609674 \cdot 10^{-5}x^2 + 0.0684170901562213x + 4.09937888198755$

因此$\sqrt{115}$的线性插值估计，误差限和有效数字分别为:

In [27]:
```
est = Pnx(Discrete_x_point,Discrete_y_point,115)
acc = origin_function(115)
error = est-acc if est>acc else acc-est
# valid_num =
print("sqrt{115}的线性插值估计值是",est)
print("sqrt{115}的精确值是",acc)
print("误差是",error)
print("误差限是0.01")
print("有效数字是4位")
```

```
sqrt{115}的线性插值估计值是 10.722755505364201
sqrt{115}的精确值是 10.723805294763608
误差是 0.0010497893994063645
误差限是0.01
有效数字是4位
```

线性插值多项式示意图如下:

In [28]:
```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(100,121,100)
y = origin_function(x)
y_= Pnx(Discrete_x_point,Discrete_y_point,x)
x_disrete_point=115
y_est = est
y_acc = acc
x_line = np.linspace(115,115,100)
y_line = np.linspace(10,11,100)


plt.plot(x,y,label = 'origin')
plt.plot(x,y_,label = 'Lagrange interpolation')
plt.plot(x_line,y_line,label = 'x=115')

plt.scatter(x_disrete_point,y_est,color = 'red')
```

```
plt.scatter(x_disrete_point,y_acc,color = 'red')
plt.xlabel("x")
plt.ylabel("y")
plt.title(r"$3$ point Lagrange interpolation of function $f(x)=\sqrt{x}$")
plt.legend(loc = 'upper left')
plt.show()
```