```
import matplotlib.pyplot as plt
import math as m
from matplotlib import colors
import numpy as np

import polint as p

def locate(x,var,n):
    jl=jm=jl=diff=j=n1=0

    ju=len(x)

    d=[0]*(n+1)

    while((ju-jl)>1):
        jm= (ju+jl) >>1
        if(var>=x[jm]):
            jl=jm
        else:
            ju=jm
    if (var==x[0]):
        j=0
    if(var==x[-1]):
        j=len(x)
    else:
        j=jl
    n1=n//2
    diff=j-n1
    if(diff<=0):
        d=x[0:(n+1)]
    elif(diff>0 and j<(30-n1-1)):
        d=x[diff:(n+1+diff)]
    else:
        d=x[(len(x)-n-1):-1]


    y=[m.sin(w+w**2) for w in d]
    return d,y



x=[0.0,0.0,0.0,0.0,0.0]
y=[0.0,0.0,0.0,0.0,0.0]

for i in range(5):
    if (i!=0):
        x[i]=x[i-1]+0.25
    y[i]=m.sin(x[i]+(x[i])**2)

xx=np.linspace(-0.5,1.5,200)
yy0=np.sin(xx+xx**2)
yy=[p.polint(x,y,w)[0] for w in xx]
err=[p.polint(x,y,w)[1] for w in xx]
err1=np.absolute(np.array(err))


plt.clf()
plt.figure(1)
plt.plot(x,y,'ro')
plt.title("Table values")
plt.xlabel("Sample Values")
plt.ylabel("Function values")
plt.grid()
plt.savefig("week1_0.png")

plt.plot(xx,yy,'r')
```

```
plt.title("Interpolated values")
plt.xlabel("Sample Values")
plt.ylabel("Function values from interpolation")
plt.grid()
plt.savefig("week1_1.png")

plt.figure(2)

plt.plot(xx,yy0,'g')
plt.title("True values")
plt.xlabel("Sample Values")
plt.ylabel("True Function values")
plt.grid()
plt.savefig("week1_2.png")

plt.figure(3)

plt.plot(xx,err1,'r')
plt.title("Error values")
plt.xlabel("Sample Values")
plt.ylabel("Error values from interpolation")
plt.grid()
plt.savefig("week1_3")

plt.figure(4)

plt.plot(xx,np.log10(err1),'g')
plt.title("Logarithmic error")
plt.xlabel("Sample Values")
plt.ylabel("Error values for interpolation")
plt.grid()
plt.savefig("week1_4")

x=[0]*5
y=[0]*5

x1=list(np.linspace(0,1,30))
y1=[0]*30
for i in range(30):
    y1[i]=m.sin(x1[i]+(x1[i])**2)


xx1=list(xx)
yy1=[0]*200
err_30_1=[0]*200
err_est=[0]*200
max_err=0
max_err1=[0]*18



for i in range(200):
    x,y=locate(x1,xx[i],7)
    yy1[i]=p.polint(x,y,xx1[i])[0]
    err_30_1[i]=p.polint(x,y,xx1[i])[1]
    err_est[i]=(np.sin(xx1[i]+(xx1[i])**2))-yy1[i]




plt.figure(5)
plt.plot(xx1,np.log(np.absolute(err_30_1)),color="red",label='Estimated error')
plt.legend(loc='lower right')
plt.title(" Error values for 30 points for nth order")
plt.plot(xx1,np.log(np.absolute(err_est)),color="green",label='True error')
plt.legend(loc='lower right')
```

```
plt.xlabel("Values")
plt.ylabel("Error values")
plt.grid()
plt.savefig("week1_30_11.png")
```

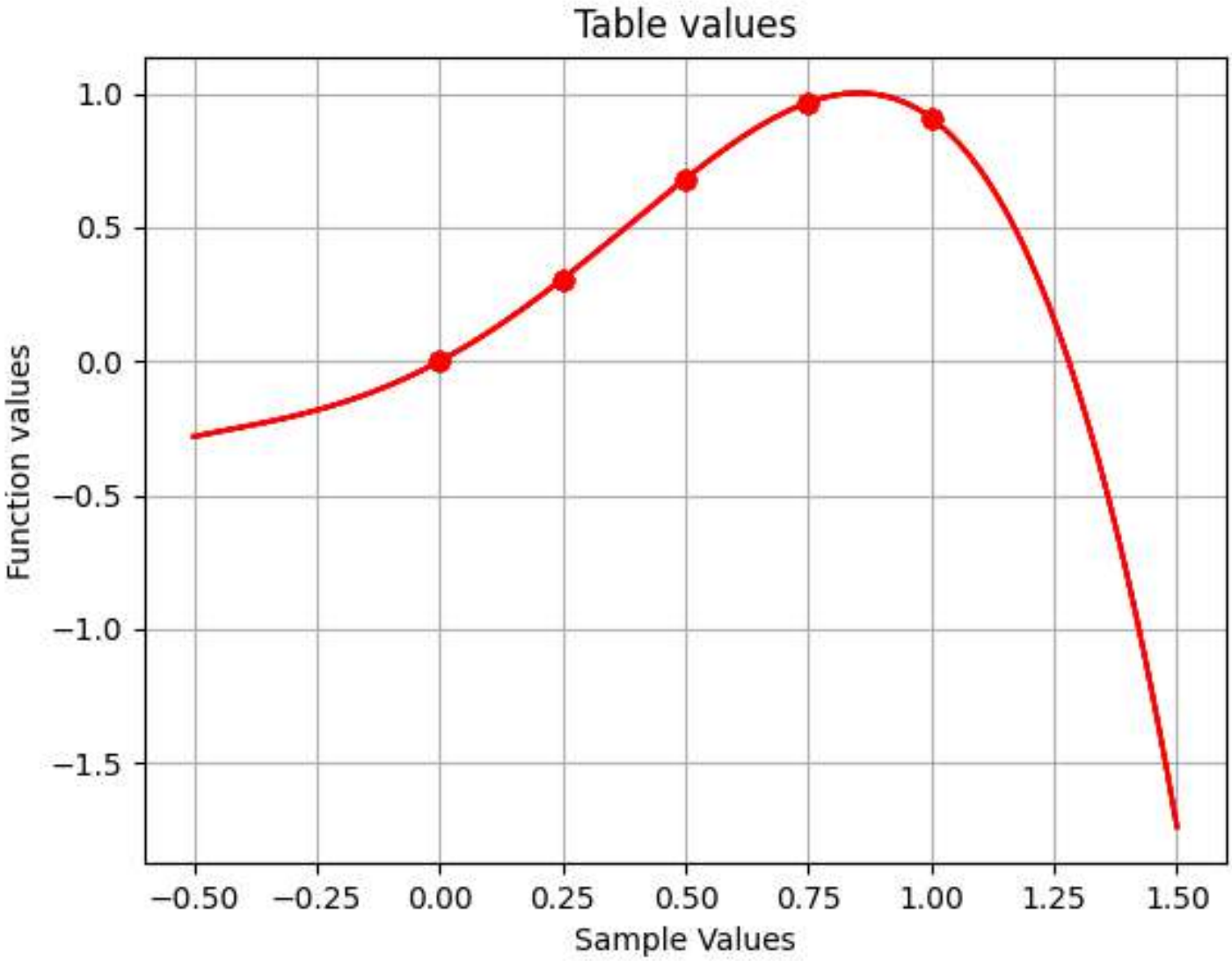
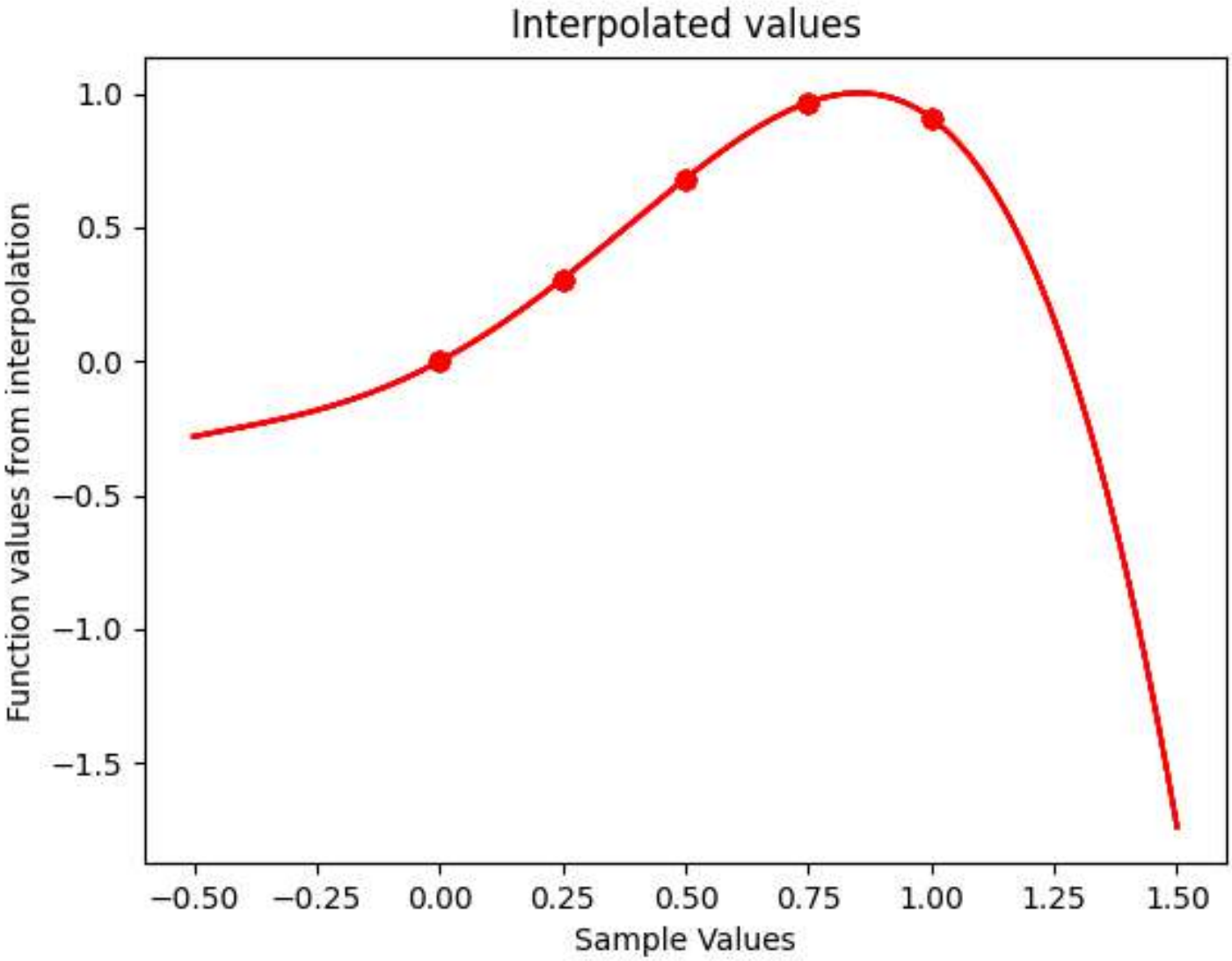
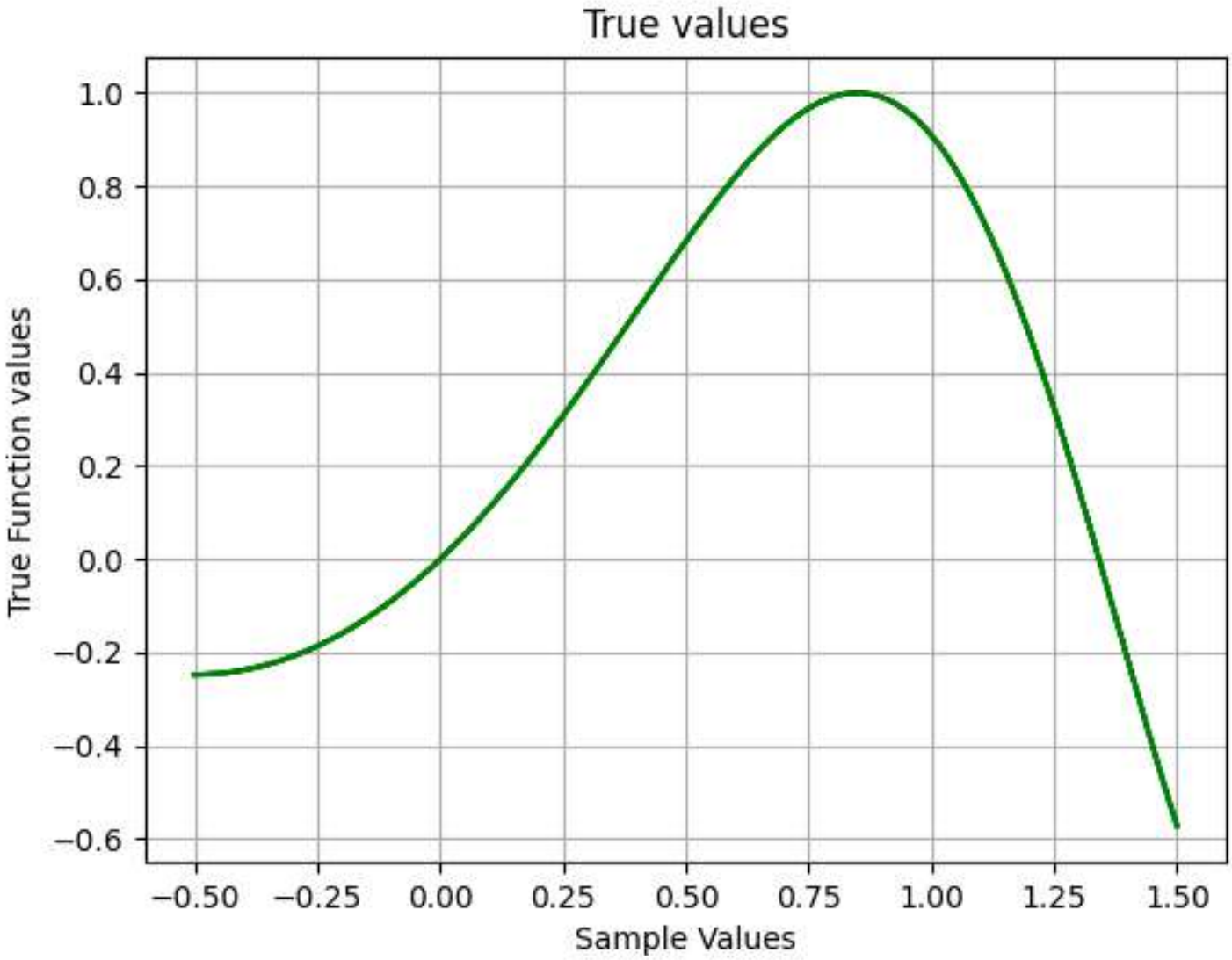
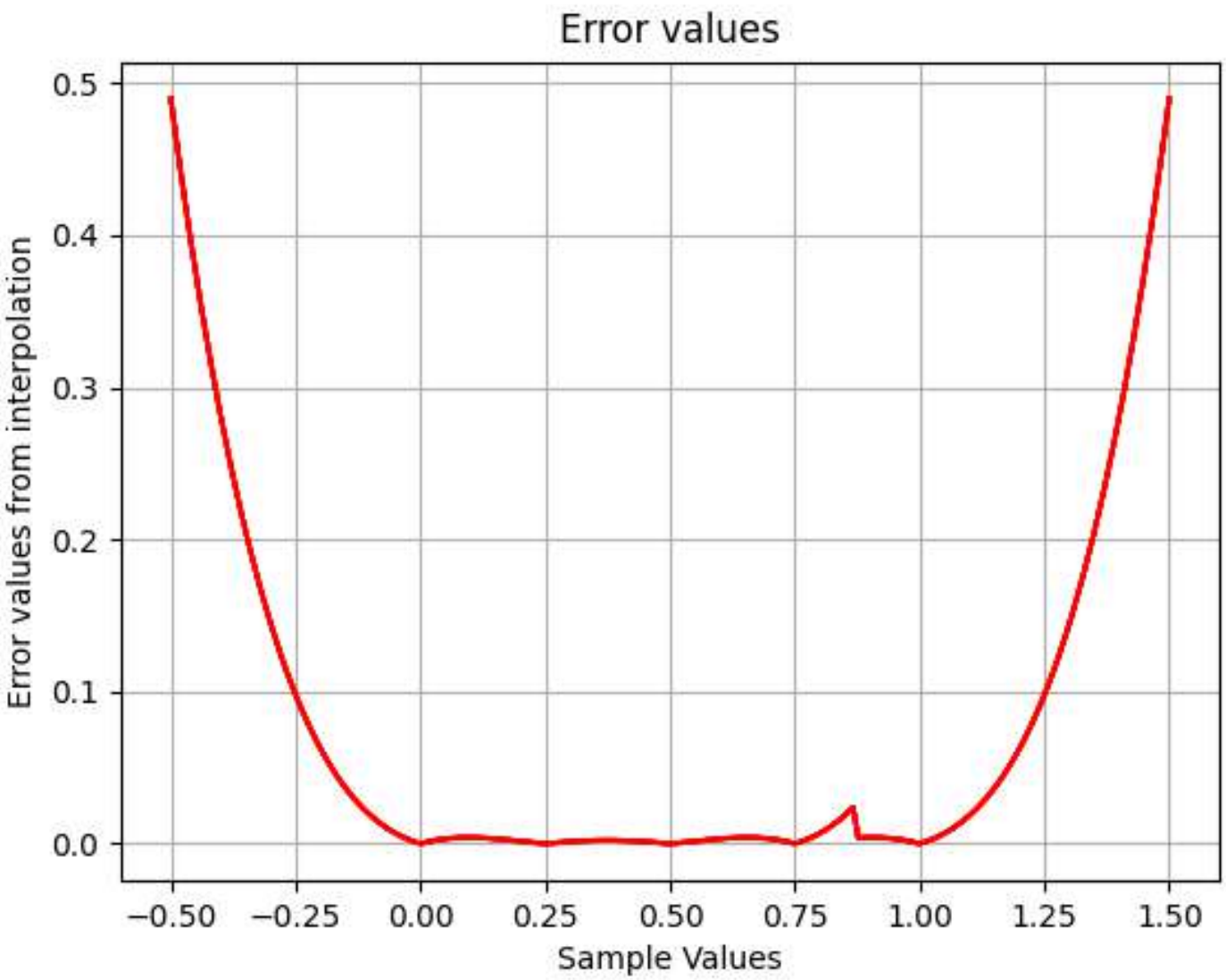
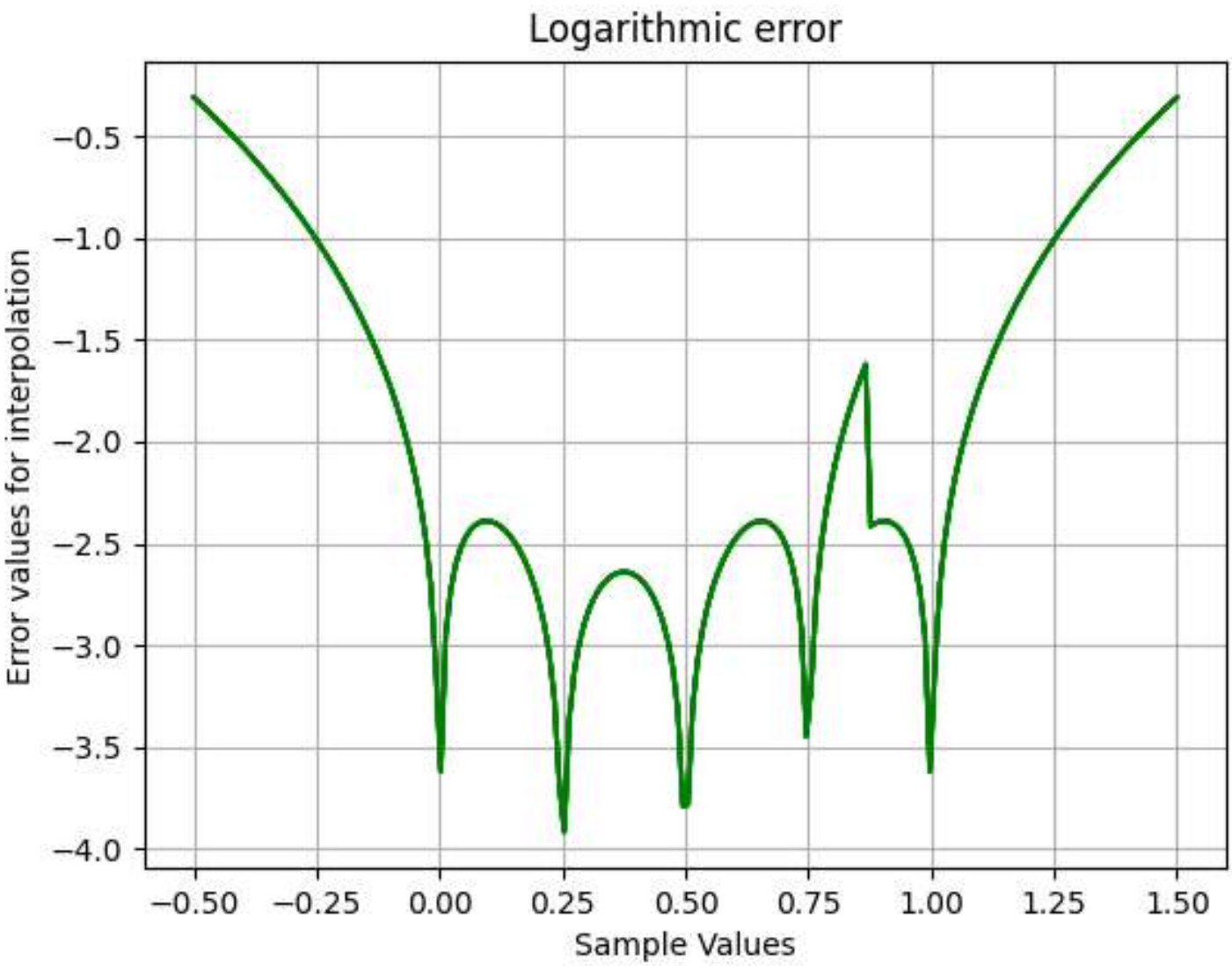


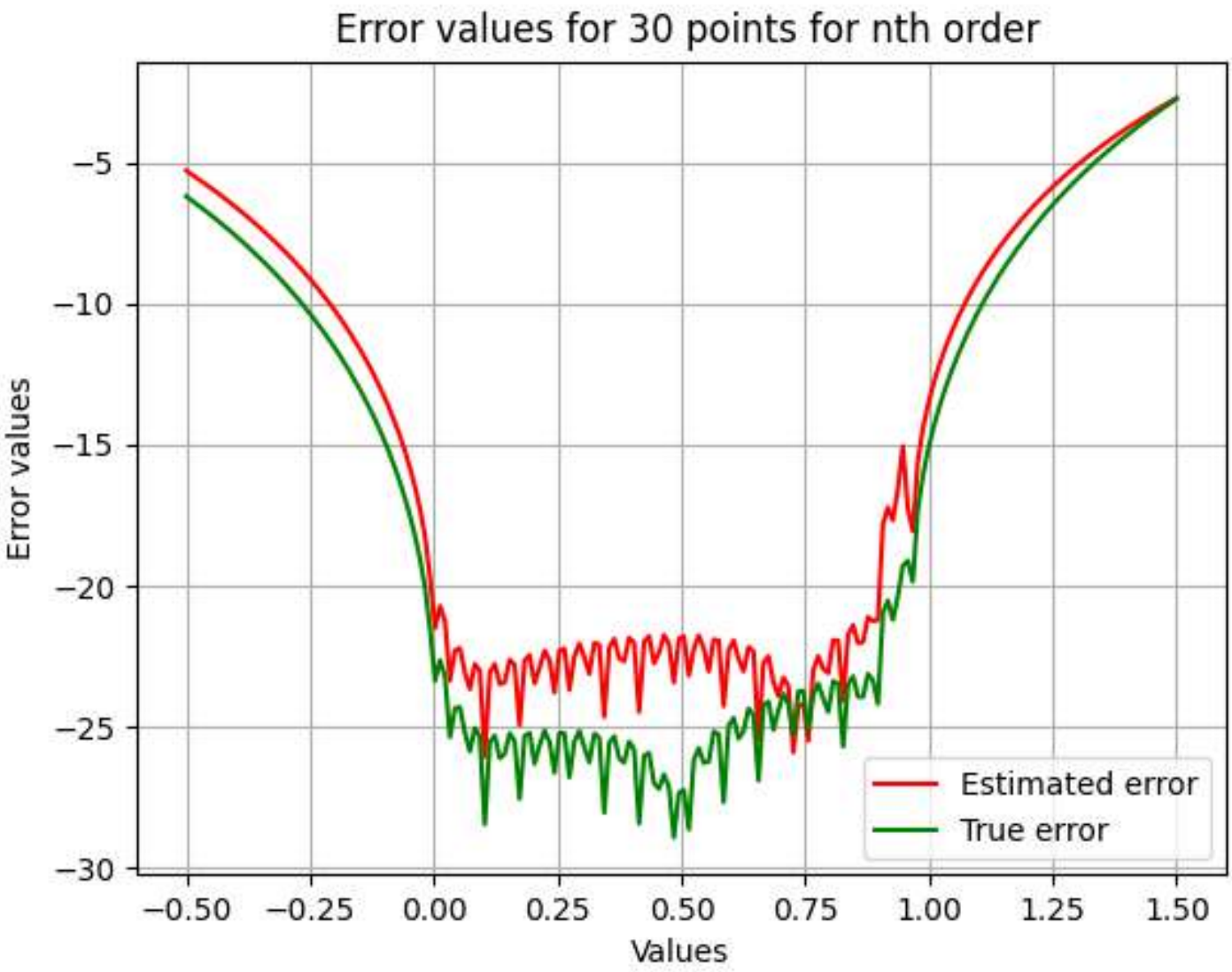
Table values
Function values
Sample Values
1.0
0.5
0.0
−0.5
−1.0
−1.5
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50


Interpolated values
Function values from interpolation
1.0
0.5
0.0
−0.5
−1.0
−1.5
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
Sample Values


True values
True Function values
Sample Values
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50

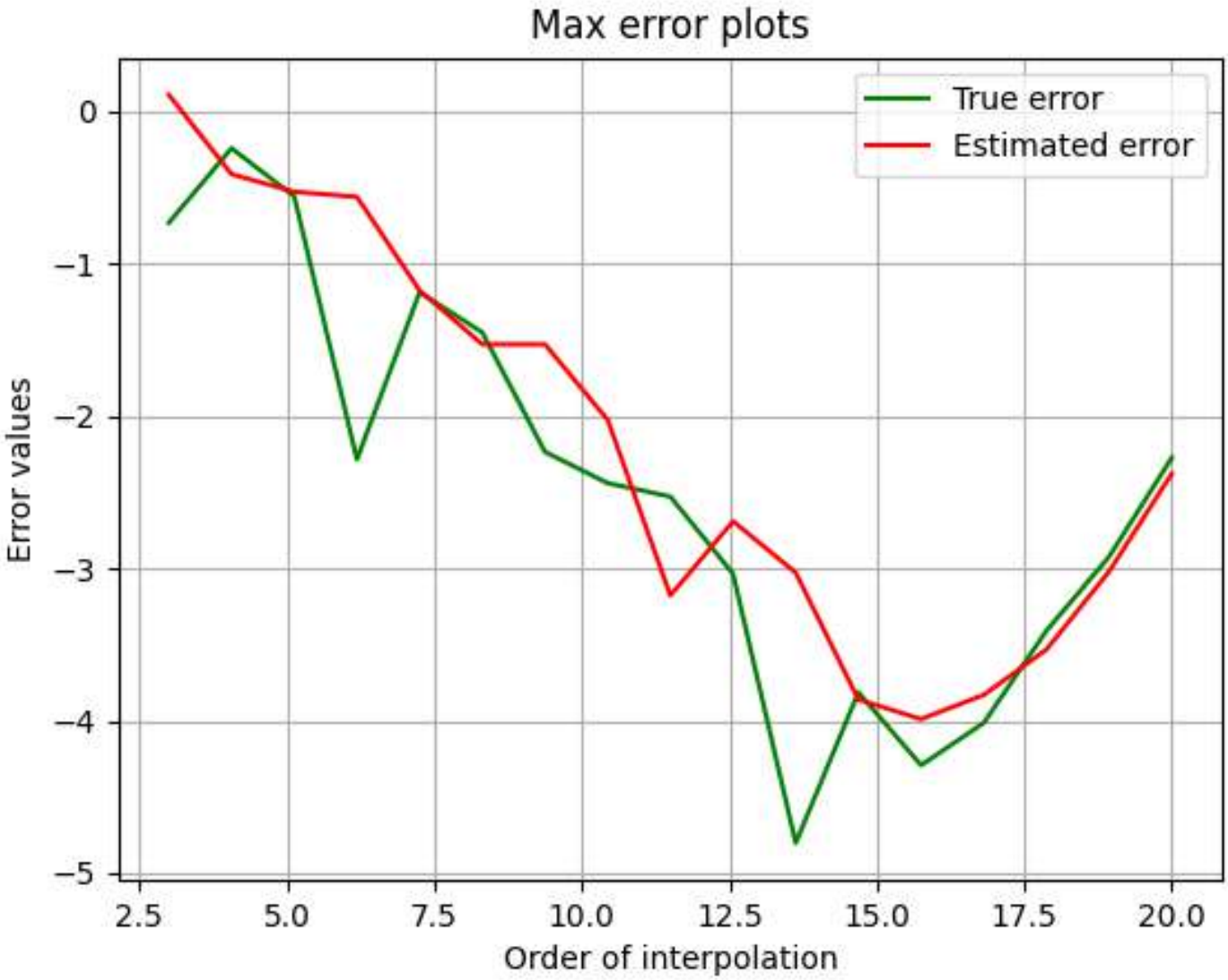

Error values
Error values from interpolation
0.5
0.4
0.3
0.2
0.1
0.0
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
Sample Values


Logarithmic error
Error values for interpolation
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
Sample Values


Error values for 30 points for nth order
Error values
−5
−10
−15
−20
−25
−30
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
Values
Estimated error
True error

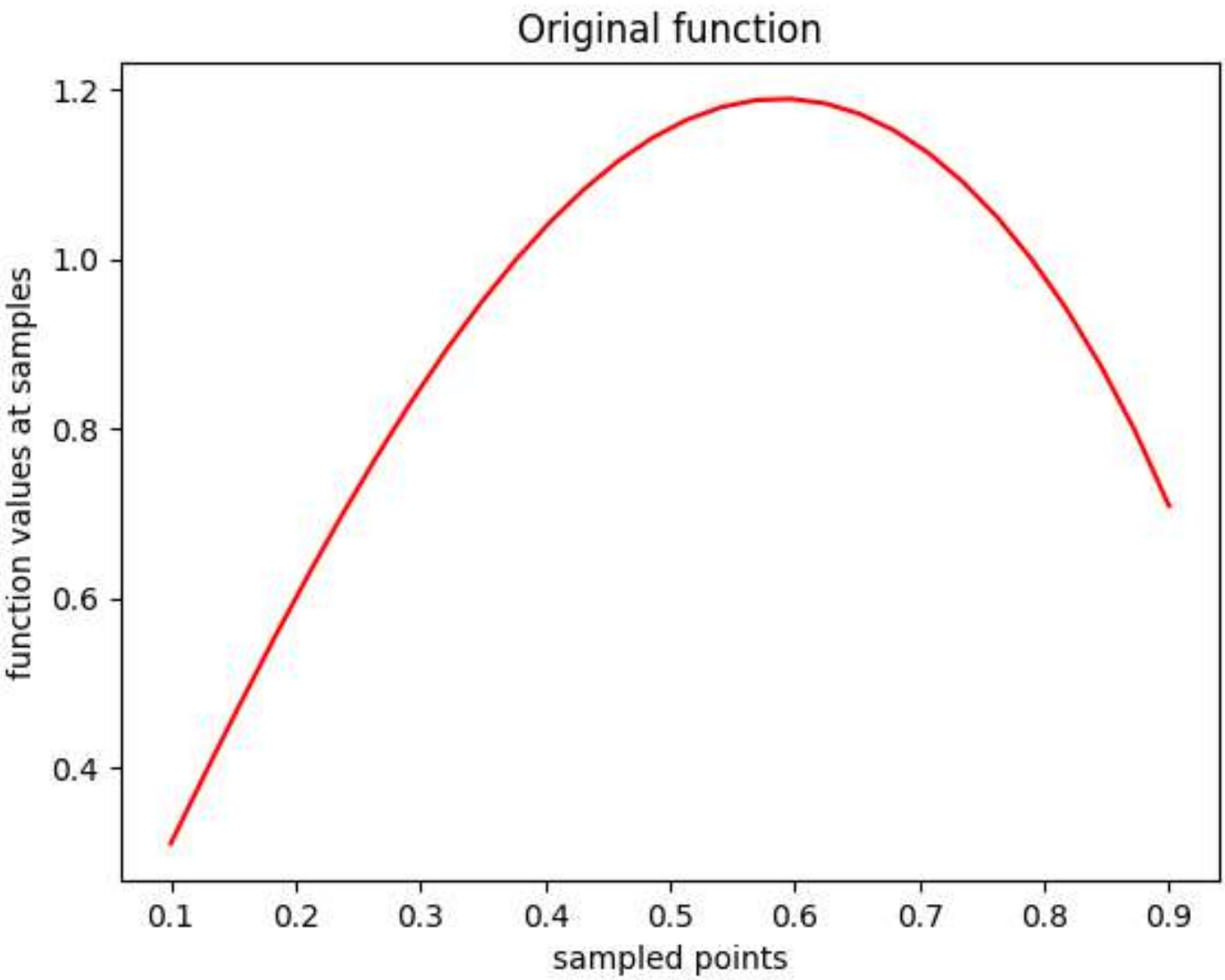
```
import matplotlib.pyplot as plt
import math as m
from matplotlib import colors
import numpy as np

import polint as p


def locate(x,var,n):
    jl=jm=jl=diff=j=n1=0

    ju=len(x)

    d=[0]*(n+1)

    while((ju-jl)>1):
        jm= (ju+jl) >>1
        if(var>=x[jm]):
            jl=jm
        else:
            ju=jm
    if (var==x[0]):
        j=0
    if(var==x[-1]):
        j=len(x)
    else:
        j=jl
    n1=n//2
    diff=j-n1
    if(diff<=0):
        d=x[0:(n+1)]
    elif(diff>0 and j<(30-n1-1)):
        d=x[diff:(n+1+diff)]
    else:
        d=x[(len(x)-n-1):-1]


    y=[m.sin(w+w**2) for w in d]
    return d,y


xx=np.linspace(-0.5,1.5,200)

x1=list(np.linspace(0,1,30))


xx1=list(xx)
yy1=[0]*200
err_30_1=[0]*200
err_est=[0]*200
max_err=0
max_err1=[0]*18
max_err_est=[0]*18



for j in range(3,20,1):
    x=y=[0]*(j+1)

    for i in range(200):


        x,y=locate(x1,xx[i],j)
        yy1[i]=p.polint(x,y,xx1[i])[0]
        err_30_1[i]=np.absolute(p.polint(x,y,xx1[i])[1])
        err_est[i]=np.absolute((np.sin(xx1[i]+(xx1[i])**2))-yy1[i])
```

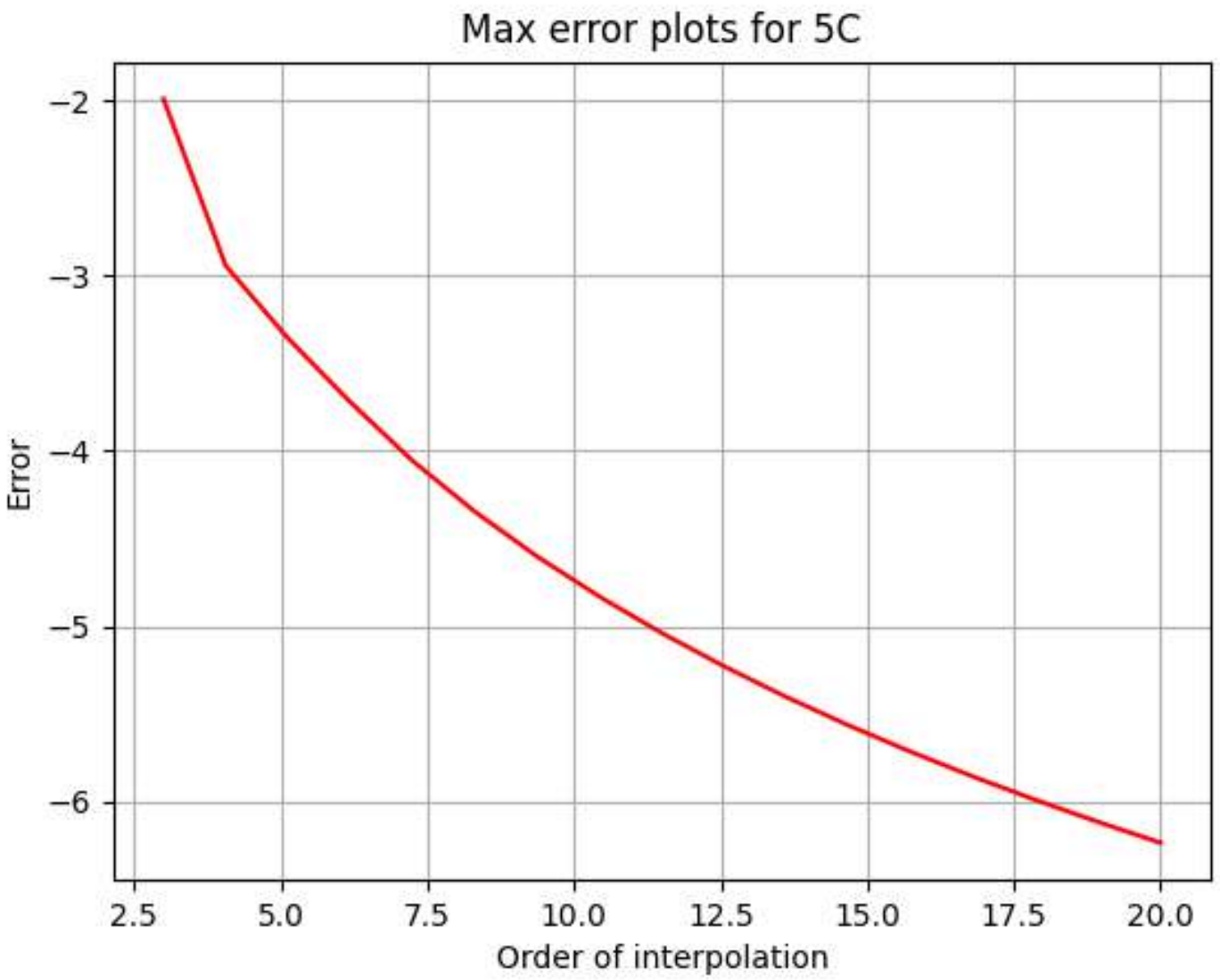
```
    max_err1[j-3]=max((err_30_1))
    max_err_est[j-3]=max((err_est))


del max_err1[-1]
del max_err_est[-1]


xval=list(np.linspace(3,20,17))
yval1=list(np.log10(max_err_est))
yval2=list(np.log10(max_err1))
plt.plot(xval,yval1,'g',label='True error')
plt.legend(loc='upper right')
plt.plot(xval,yval2,'r',label="Estimated error")
plt.legend(loc='upper right')
plt.title("Max error plots")
plt.xlabel("Order of interpolation")
plt.ylabel("Error values")
plt.grid()
plt.savefig("Max errorQ41.png")
```

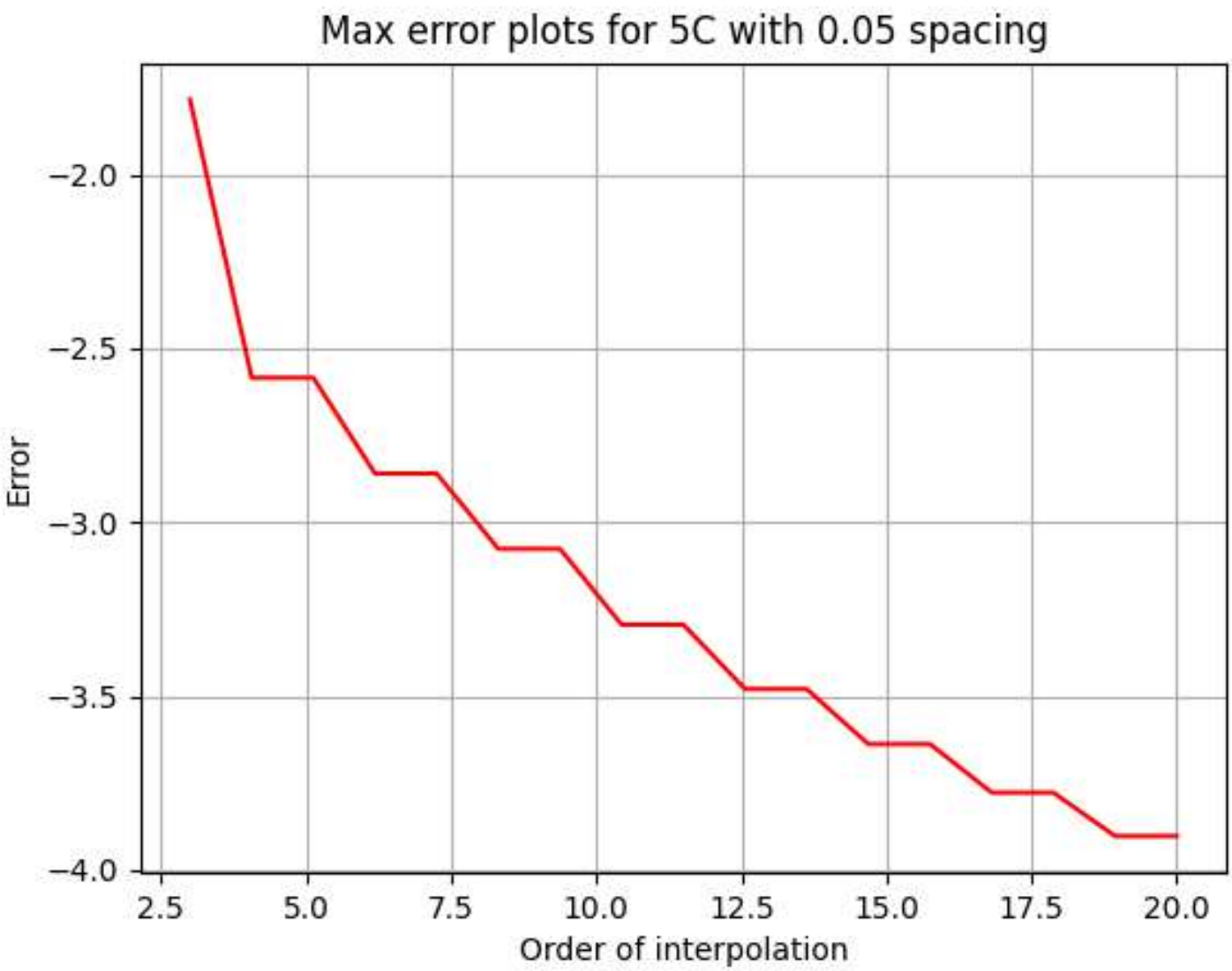

Max error plots
True error
Estimated error
0
−1
−2
−3
−4
−5
Error values
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Order of interpolation

```
import matplotlib.pyplot as plt
import math as m
from matplotlib import colors
import numpy as np

import polint as p


def locate(x,var,n):
    jl=jm=jl=diff=j=n1=0

    ju=len(x)

    d=[0]*(n+1)

    while((ju-jl)>1):
        jm= (ju+jl) >>1
        if(var>=x[jm]):
            jl=jm
        else:
            ju=jm
    if (var==x[0]):
        j=0
    if(var==x[-1]):
        j=len(x)
    else:
        j=jl
    n1=n//2
    diff=j-n1
    if(diff<=0):
        d=x[0:(n+1)]
    elif(diff>0 and j<(30-n1-1)):
        d=x[diff:(n+1+diff)]
    else:
        d=x[(len(x)-n-1):-1]
    y=[m.sin(m.pi*w)/(m.sqrt(1-w**2)) for w in d]
    return d,y

xx=np.linspace(0.1,0.9,30)
#upon increasing beyond 0.9 and plotting for 1.1 or 1.2 or 1.1 shows runtime warning. The plots dont change for the values mentioned.
#The funstion itself converges to a value mathematically, but the error from interpolation blows up.


f=np.sin(np.pi*xx)/(np.sqrt(1-xx**2))
plt.figure()
plt.plot(list(xx),list(f),'r')
plt.title("Original function")
plt.ylabel("function values at samples")
plt.xlabel("sampled points")
plt.savefig("Q5_b 30pts.png")


xx1=list(np.linspace(0.1,0.9,1000))
yy=[0]*1000
err_30_1=[0]*1000
err_true=[0]*1000
max_err=[0]*18
max_err_true=[0]*18

for j in range(3,20,1):
    x=y=[0]*(j+1)

    for i in range(1000):
```

```
        x,y=locate(xx,xx1[i],j)
        yy[i]=p.polint(x,y,xx1[i])[0]
        err_30_1[i]=np.absolute(p.polint(x,y,xx1[i])[1])
        err_true[i]=np.absolute((np.sin(np.pi*xx1[i])/(np.sqrt(1-xx1[i]**2)))-yy[i])
    max_err[j-3]=max((err_30_1))
    max_err_true[j-3]=max((err_true))

del max_err[-1]
del max_err_true[-1]
plt.figure()
xval=list(np.linspace(3,20,17))
plt.plot(xval,np.log10(max_err),'r')
plt.title("Max error plots for 5C")
plt.xlabel("Order of interpolation")
plt.ylabel("Error")
plt.grid()
plt.savefig("Max error_Q5 30pts.png")
# Order 0f 16 above gives 6 digit accuracy
```


Original function
1.2
1.0
0.8
0.6
0.4
function values at samples
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
sampled points


Max error plots for 5C
−2
−3
−4
−5
−6
Error
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Order of interpolation


Max error plots for 5C with 0.05 spacing
−2.0
−2.5
−3.0
−3.5
−4.0
Error
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Order of interpolation