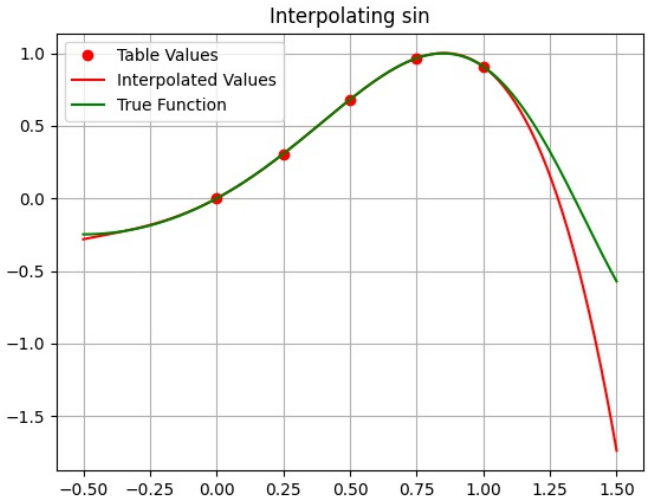
EE5011 Week1 Assignment

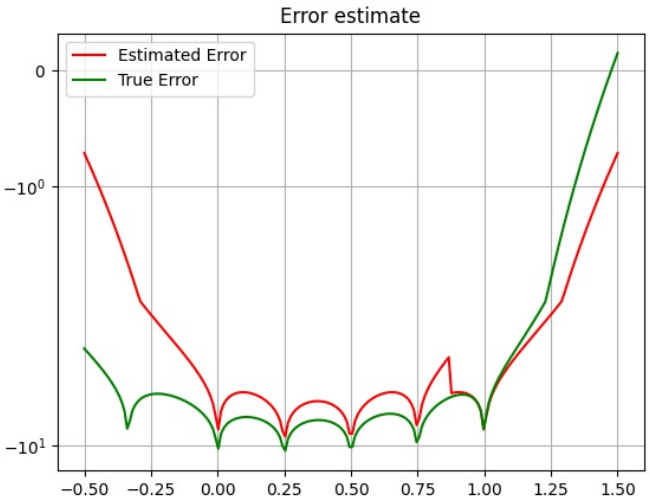
Arihant Jain ee21s061

September 15, 2021

1. Python Code for 4^{th} Order interpolation given 5 points in table

```
import polint
import math
import numpy as np
import matplotlib.pyplot as plt
plt.clf()
x=np.linspace(0,1,5)
sinx=np.sin((x+x**2))
xx=np.linspace(-0.5,1.5,200)
y=[polint.polint(x,sinx,w)[0] for w in xx]
dy=[polint.polint(x,sinx,w)[1] for w in xx]
plt.title("Interpolation Vs Sin function")
sinxx=np.sin((xx+xx**2))
plt.plot(x,sinx,'ro',xx,y,'r',xx,sinxx,'g')
plt.title("Interpolating sin")
plt.legend(["Table Values","Interpolated Values","True Function"])
plt.grid()
plt.savefig("sin_que1.jpg")
plt.clf()
plt.title("Error estimate")
np.seterr(all="ignore")
plt.yscale("symlog")
plt.plot(xx,np.log(np.absolute(dy)),color='red',label="Estimated Error")
plt.plot(xx,np.log(np.absolute(y-sinxx)),color='green',label="True Error")
plt.legend(numpoints=1)
plt.grid()
plt.savefig("error_que1.jpg")
```





Observations:

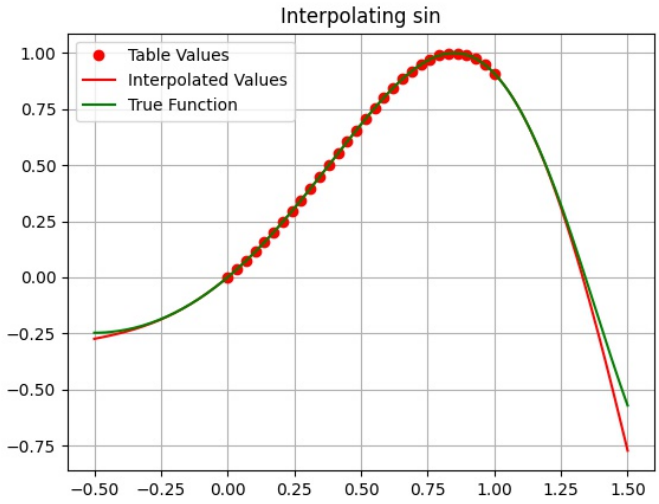
- We can see in the Error graph that the error grows exponentially once the x goes out of the sampled values (Below 0 and over 1). The Sin graph shows how the interpolation function follow true function closely between 0 to 1 but goes off outside of that range.
- $\sin(x+x^2)$ has 2 term and as we can see in the graph goes off course relatively more after 1 than below 0 this is primarily because x^2 term dominates the function over 1 and this is not part of sampled value.

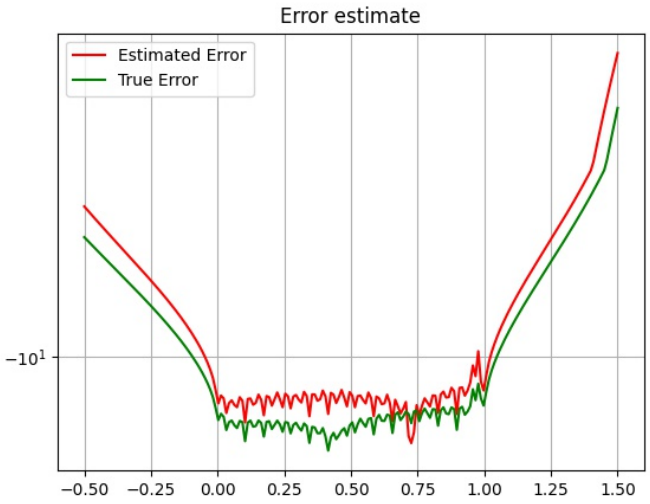
Q2: We sample the function at 30 points from 0 to 1 and add a search algorithm to find 5 nearest point to do a 4^{th} order interpolation.

Program to locate the nearest point index:

```
import polint
import math
import numpy as np
import matplotlib.pyplot as plt
def locate(x,q,n):
    x = np.asarray(x)
    idx = (np.abs(x - q)).argmin()
    if idx-n//2<1:
        return x[0:n]
    elif idx+n//2>len(x):
```

```
        return x[-n:]
    return x[idx-n//2:idx+n//2+1]
def sinlocate(x,q,n):
    x = np.asarray(x)
    idx = (np.abs(x - q)).argmin()
    if idx-n//2<1:
        return sinx[0:n]
    elif idx+n//2>len(x):
        return sinx[-n:]
    return sinx[idx-n//2:idx+n//2+1]
plt.clf()
n=5
x=np.linspace(0,1,30)
sinx=np.sin((x+x**2))
xx=np.linspace(-0.5,1.5,200)
y=[polint.polint(locate(x,w,n),sinlocate(x,w,n),w)[0] for w in xx]
dy=[polint.polint(locate(x,w,n),sinlocate(x,w,n),w)[1] for w in xx]
plt.title("Interpolation Vs Sin function")
sinxx=np.sin((xx+xx**2))
plt.plot(x,sinx,'ro',xx,y,'r',xx,sinxx,'g')
plt.title("Interpolating sin")
plt.legend(["Table Values","Interpolated Values","True Function"])
plt.grid()
plt.savefig("sin_que2.jpg")
plt.clf()
plt.title("Error estimate")
np.seterr(all="ignore")
plt.yscale("symlog")
plt.plot(xx,np.log(np.absolute(dy)),color='red',label="Estimated Error")
plt.plot(xx,np.log(np.absolute(y-sinxx)),color='green',label="True Error")
plt.legend(numpoints=1)
plt.grid()
plt.clf()
plt.savefig("error_que2.jpg")
```

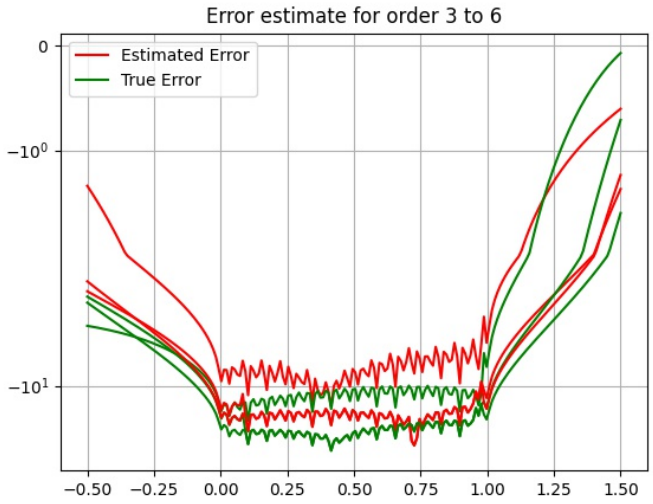


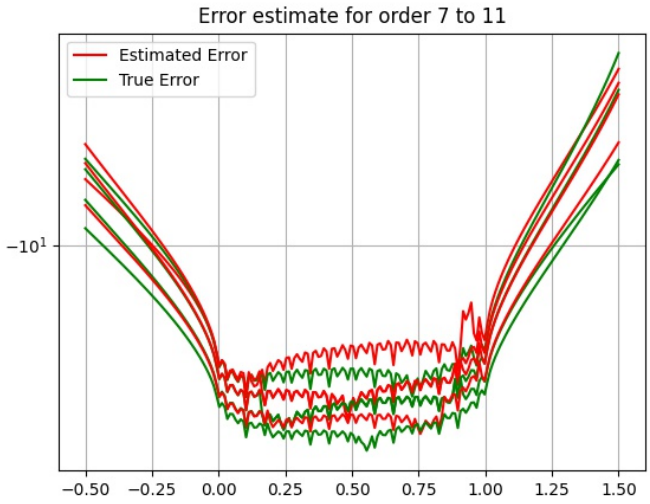


Observations

- The error in this case is significantly lower than the previous interpolation this is because we can now choose the closest 5 point for interpolation from 30 this significantly reduces the error and makes the interpolated function follow true function more closely.
- The Error Estimate graph shows that the error is now below 10^{-1} and also the estimated error follow the true error but still the true error is less than the estimated error this is because the estimated error comes from n-1 order for a n order interpolation.
- We see again that the error grows significantly before 0 and after 1 as the sampled value only have data from range 0 to 1 this is basically extrapolation.

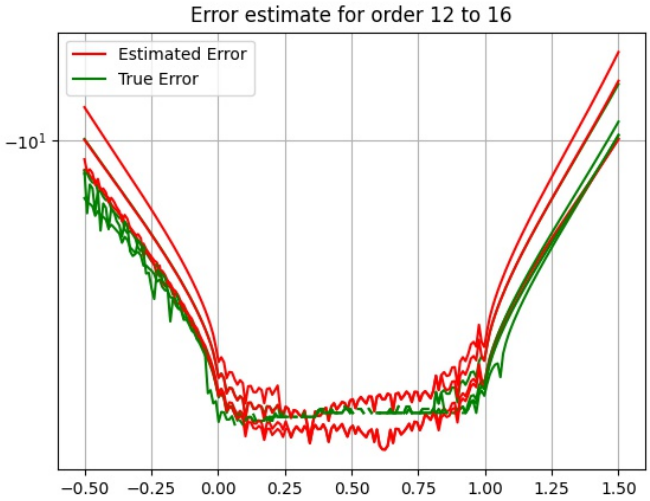
Q3: Plot of error as the order of interpolation vary from 3 to 16.





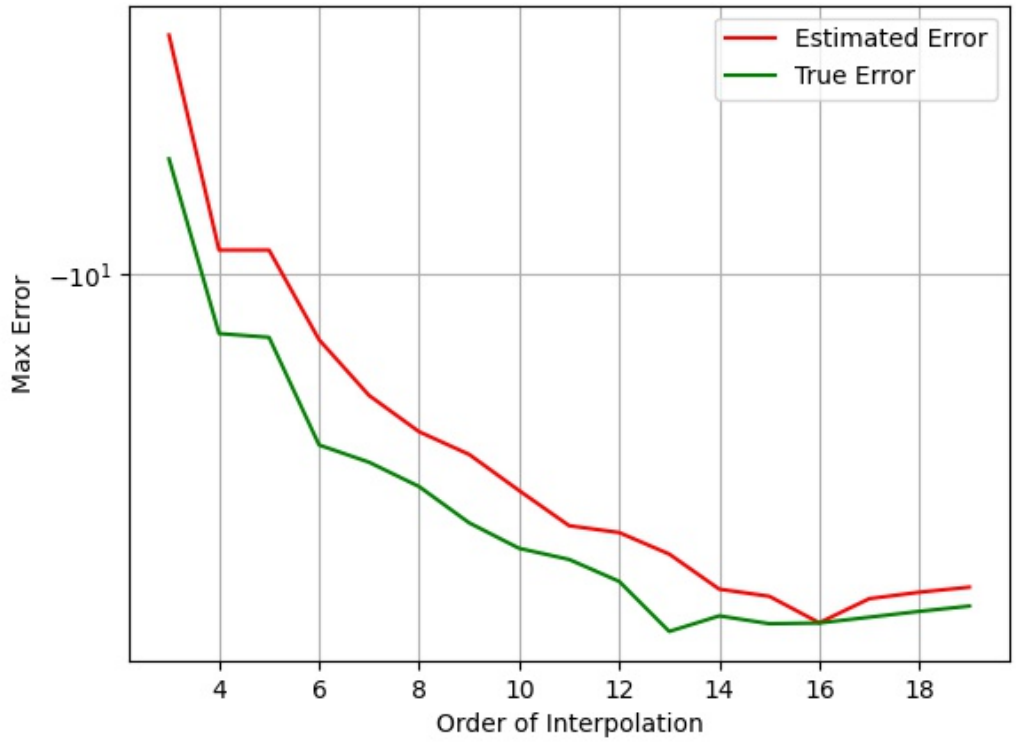
Observations

- The graph has two many data plot for us to make sense of so divided the graph over different order of interpolation.
- We can see that from the 2 image above plot shows a similar behaviour outside of range of 0 to 1 but between the range the error can be seen decreasing as the order increase.



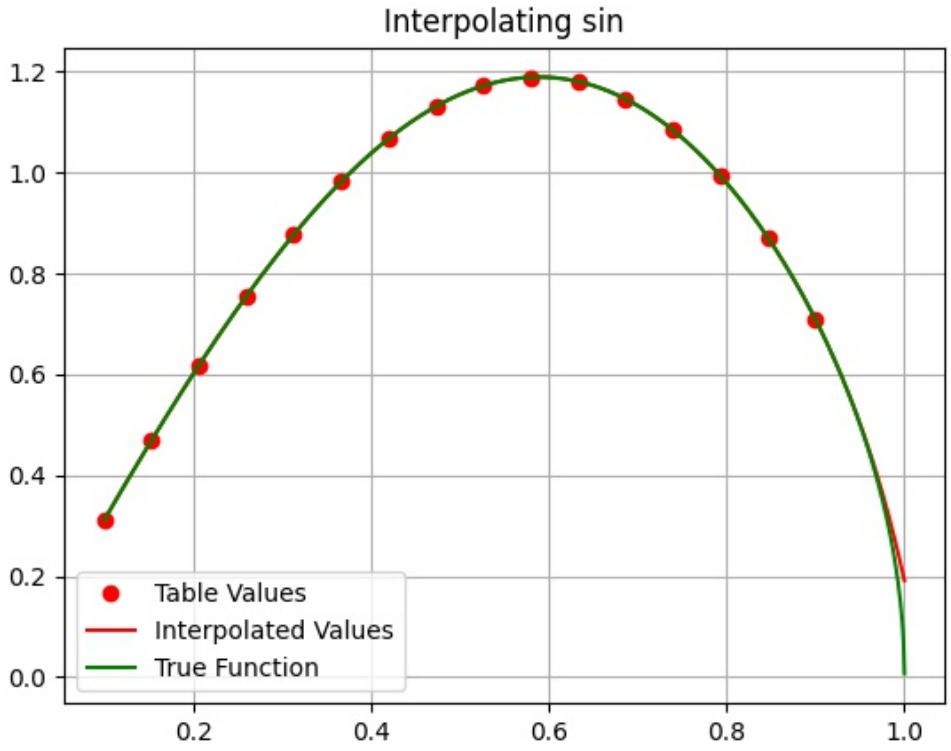
- For even higher order of interpolation we can see the error keeps on decreasing but now the peeks of graph between 0 to 1 are lower that is the frequency of the plot decrease as there are more points being used for interpolation.

Q4: Plot of Maxerror as the order of interpolation vary from 3 to 20.



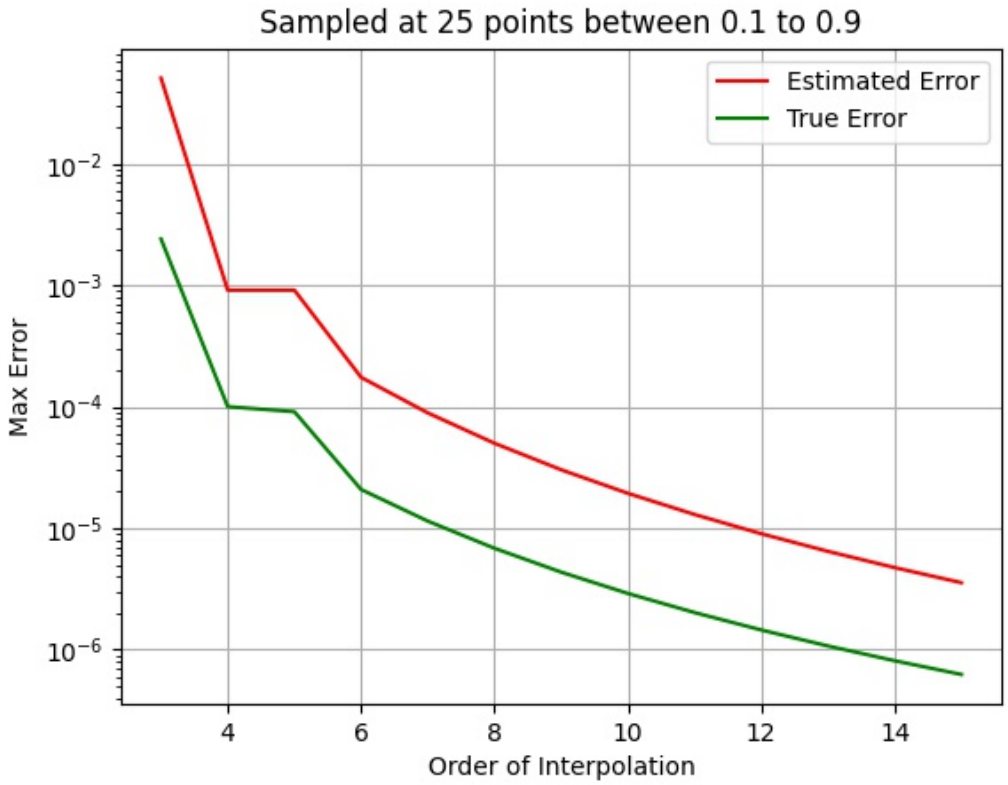
- The maxerror is calculated between the range of 0 to 1 and for order 3 to 20.
- We see the error drop sharply as order of interpolation goes 3 to 10 but after that the error drops more gradually and eventually plateau around 14 to 16.
- This means that the error can be dropped by increasing order of interpolation but there's limit after which we don't get significant gains and any more attempt to get more accurate by increasing order could harm the calculation and make the error go back up again.

Q5: plot for f(x) = $\sin(\pi x)/\sqrt{1-x^2}$

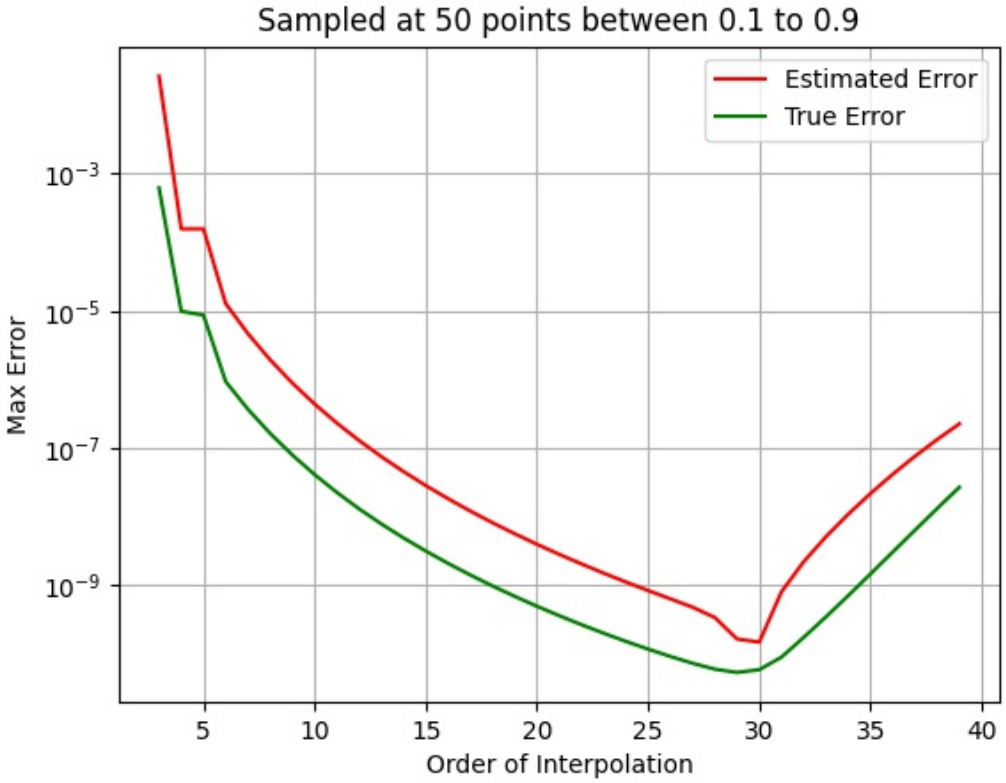


- The Function exist only in the open interval (-1,1) and at ±1 first derivative it goes to singularity.
- The function is analytic in the region 0.1 to 0.9 as it is smooth and continuous with no gaps and derivative of the function exist.
- The radius of convergence of the function would be $|\ x\ |<1$.

Q5(c):For 6 digit accuracy we vary the order from 3 to 16 and look at the max error graph



- The error drop sharply and then gradually after 6^{th} order interpolation and we get to 10^{-6} around 13^{th} order of interpolation. 10^{-6} error means we have the value of function accurate up-to 6 digit.
- The estimated error as usual is greater than the true error being derived from n-1 order of interpolation so looking at that we can see we reach 5 digit accuracy around 13^{th} order of interpolation. This accuracy can further improved using more points and a higher order of interpolation but there's limit to that.



- I sampled the function at 50 points and then varied the order of interpolation from 3 to 40 to see then the interpolation starts to fail.
- From the above graph we can see that the accuracy of interpolation reaches its peek around 30^{th} order of interpolation after which the error after accumulating and the interpolation starts experiencing more error.
- This seems to be because as we increase the order of polynomial it starts oscillating.