1.) a.)

Taylor Series of f(x) = cos($\frac{x}{3}$) about $x_0 = \pi$:

T(x) = cos($\frac{\pi}{3}$) - $\frac{1}{3}$· sin($\frac{\pi}{3}$)(x-π) - $\frac{1}{2}$ · $\frac{1}{9}$· cos($\frac{\pi}{3}$)(x-π)^2 + $\frac{1}{6}$ · $\frac{1}{27}$· sin($\frac{\pi}{3}$)(x-π)^3 + $\frac{1}{24}$ · $\frac{1}{81}$· cos($\frac{\pi}{3}$)(x-π)^4 - $\frac{1}{120}$ · $\frac{1}{243}$· sin($\frac{\pi}{3}$)(x-π)^5 - ...

T(x) = cos($\frac{\pi}{3}$) - $\frac{1}{3}$ · $\frac{\sqrt{3}}{2}$(x-π) - $\frac{1}{18}$ · $\frac{1}{2}$(x-π)^2 + $\frac{1}{162}$ · $\frac{\sqrt{3}}{2}$(x-π)^3 + $\frac{1}{1944}$ · $\frac{1}{2}$(x-π)^4 - $\frac{1}{29160}$ · $\frac{\sqrt{3}}{2}$(x-π)^5 - ...

T(x) = cos($\frac{\pi}{3}$) - $\frac{\sqrt{3}}{6}$(x-π) - $\frac{1}{36}$(x-π)^2 + $\frac{\sqrt{3}}{324}$(x-π)^3 + $\frac{1}{3888}$(x-π)^4 - $\frac{\sqrt{3}}{58320}$(x-π)^5 - ...

b.)

true = cos($\frac{1}{3}$) = 0.94495694631

est = 1 - $\frac{(\frac{1}{3})^2}{2!}$ + $\frac{(\frac{1}{3})^4}{4!}$ = 1 - $\frac{1}{18}$ + $\frac{1}{1944}$ = 0.94495884773

absolute error = $|true - est|$ = $|0.94495694631 - 0.94495884773|$ = 0.00000190142

relative error = $|\frac{true-est}{true}|$ = $|\frac{0.94495694631-0.94495884773}{0.94495694631}|$ = 0.00000201217

c.)

b's absolute error = 1.901 x 10^{-6}

b's relative error = 2.012 x 10^{-6}

if $\hat{x}$ = 0.999, $\hat{f}(\hat{x})$ = 0.94506784876

absolute error = $|0.94495884773 - 0.94506784876|$ = 0.00010900103

relative error = $|\frac{0.94495884773-0.94506784876}{0.94495884773}|$ = 0.00011535002

d.)

$\hat{f}(r)$ = cos($\frac{x}{3}$) - $\frac{1}{3}$ sin($\frac{x}{3}$)(r-x) - $\frac{1}{18}$ cos($\frac{x}{3}$)(r-x)^2

if r = x + h, using only the linear approximation gives:

$\hat{f}(x+h)$ = cos($\frac{x}{3}$) - $\frac{1}{3}$ sin($\frac{x}{3}$)(x + h - x) = cos($\frac{x}{3}$) - $\frac{1}{3}$ sin($\frac{x}{3}$)h$

absolute error: $|f(x+h) - \hat{f}(x+h)| = |cos(\frac{x+h}{3}) - cos(\frac{x}{3}) - \frac{1}{3}sin(\frac{x}{3})h|$

relative error: $|\frac{f(x+h)-\hat{f}(x+h)}{f(x+h)}| = |\frac{cos(\frac{x+h}{3})-cos(\frac{x}{3})-\frac{1}{3}sin(\frac{x}{3})h}{cos(\frac{x+h}{3})}|$

For h very small, I will approximate cos(x+h) = cos(x), so the approximated errors become:

absolute error: $|cos(\frac{x}{3}) - cos(\frac{x}{3}) - \frac{1}{3}sin(\frac{x}{3})h| = |-\frac{1}{3}sin(\frac{x}{3})h|$

relative error: $|\frac{cos(\frac{x}{3}) - cos(\frac{x}{3}) - \frac{1}{3}sin(\frac{x}{3})h}{cos(\frac{x+h}{3})}| = |-\frac{1}{3}tan(\frac{x}{3})h|$

e.)

Condition of f(x), $\kappa = |\frac{xf'(x)}{f(x)}| = |\frac{-xsin(\frac{x}{3})}{3cos(\frac{x}{3})}| = -\frac{x}{3}tan(\frac{x}{3})$

f(x) is ill conditioned for large magnitude x and for x close to $\pm\frac{\pi}{2}, \pm\frac{3\pi}{2}, \pm\frac{5\pi}{2}, \ldots$

2.) a.)

$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \ldots$

$f(x_0 - h) = f(x_0) - hf'(x_0) + \frac{h^2}{2}f''(x_0) - \frac{h^3}{3!}f'''(x_0) + \ldots$

$f(x_0 + h) - f(x_0 - h) = 2hf'(x_0) + 2\frac{h^3}{3!}f'''(x_0) + \ldots$

$\frac{f(x_0+h) - f(x_0-h)}{2h} = f'(x_0) + \frac{h^2}{3!}f'''(x_0) + \ldots$

So an approximation for $f'(x_0)\text{is}\frac{f(x_0+h)-f(x_0-h)}{2h}$.

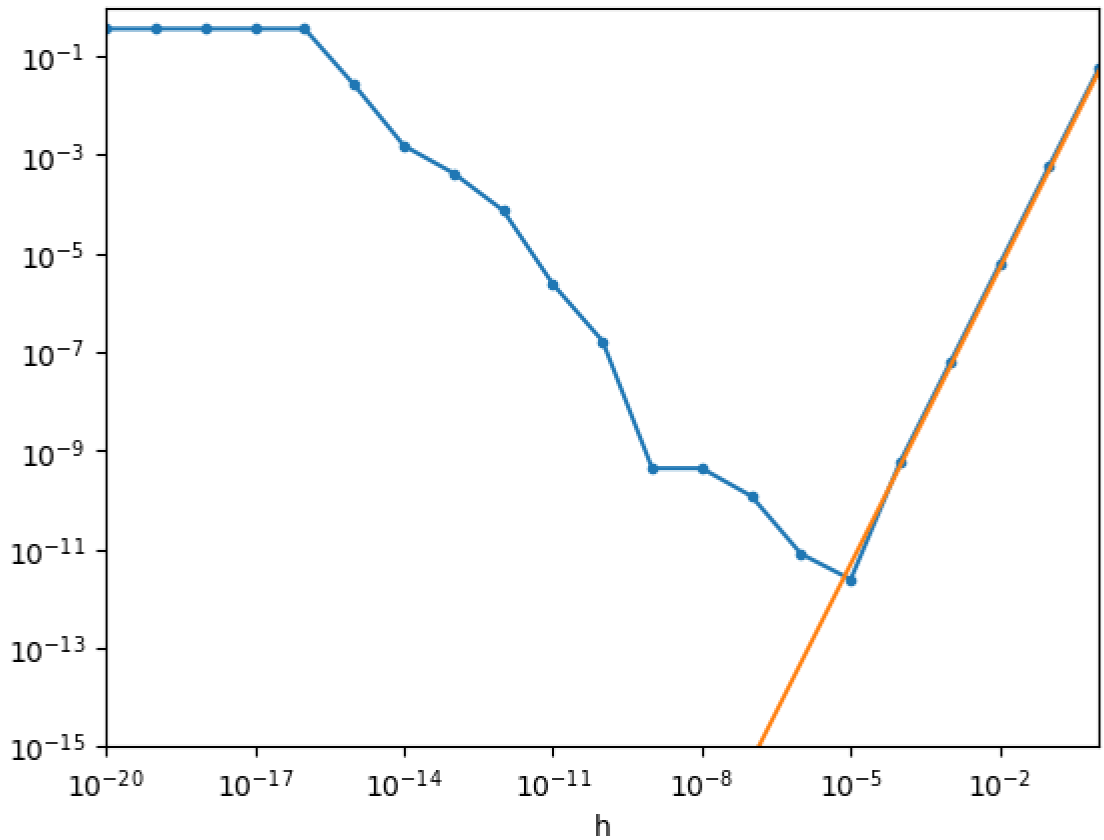
The associated in error in this approximation is $Error = \frac{h^2}{3!}f'''(x_0) + \frac{h^4}{5!}f^{(5)}(x_0) + \ldots$ which is O(h^2)

b.)

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return np.sin(x)

def derivative(x):
    return np.cos(x)

x0 = 1.2
f0 = f(x0)
fp = derivative(x0)
h = np.logspace(-20,0,21)
err = abs(fp - (f(x0+h)-f(x0-h))/(2*h))

plt.plot(h,err,'.-')
plt.plot(h,0.05*h**2)
plt.axis([1e-20,1,1e-15,1e0])
plt.xlabel('h')
#plt.ylabel('err')
```

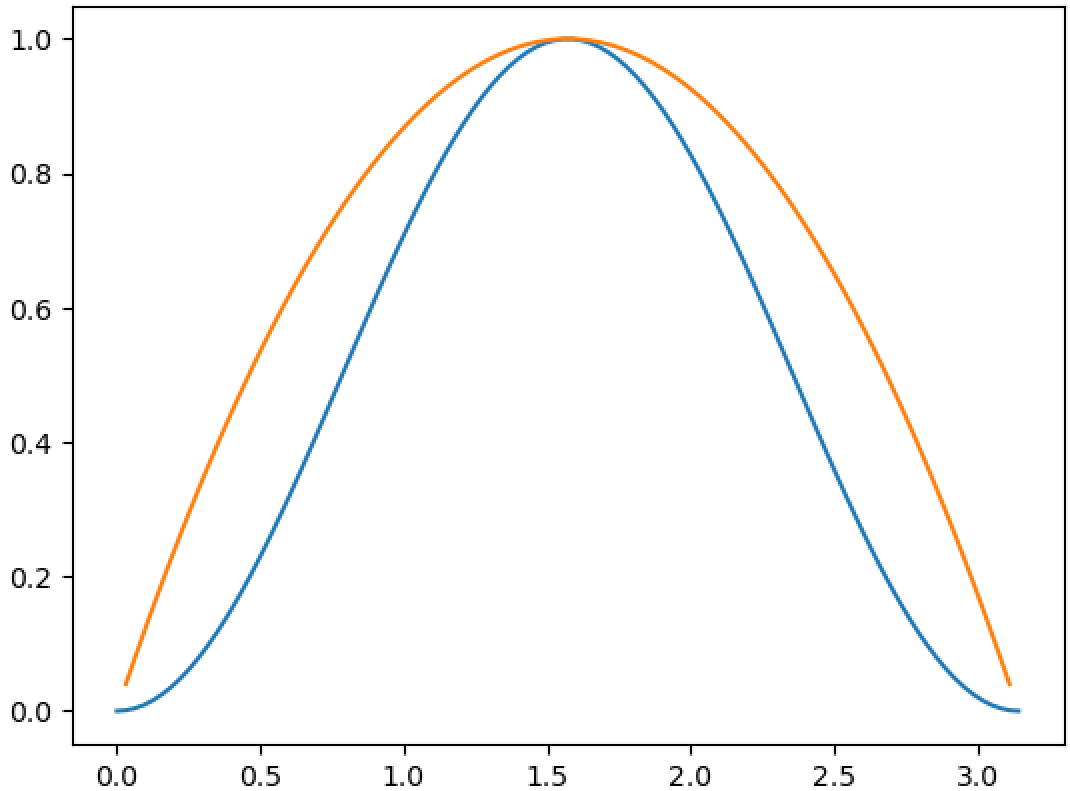
```
plt.yscale('log')
plt.xscale('log')
```



The graph of the central difference approximation is steeper against h as the error using this approximation is O(h^2), where as the error in the forward differences approximation is O(h). This algorithm begins to become inaccurate at h values of roughly 10 $^{-5}$. This is when the cancellation error of f(x+h) - f(x-h) begins to have a significant effect when multiplied by $\frac{1}{2h}$.

3.) a.)

In [ ]:
```
def f(x):
    return(np.sin(x)**2)

xs = [0, np.pi/2, np.pi]
ys = f(xs)

w0 = 2/(np.pi**2)
w1 = -4/(np.pi**2)
w2 = 2/(np.pi**2)

def p(x):
    return [x*(x-xs[1])*(x-xs[2])*(ys[0]*w0/x + ys[1]*w1/(x-xs[1]) + ys[2]*w2/(x-xs[2

xArr = np.linspace(0,np.pi,100)
plt.plot(xArr,f(xArr),'-')
xArr = [x for x in xArr if x not in xs]
plt.plot(xArr,p(xArr),'-')
plt.show()
```

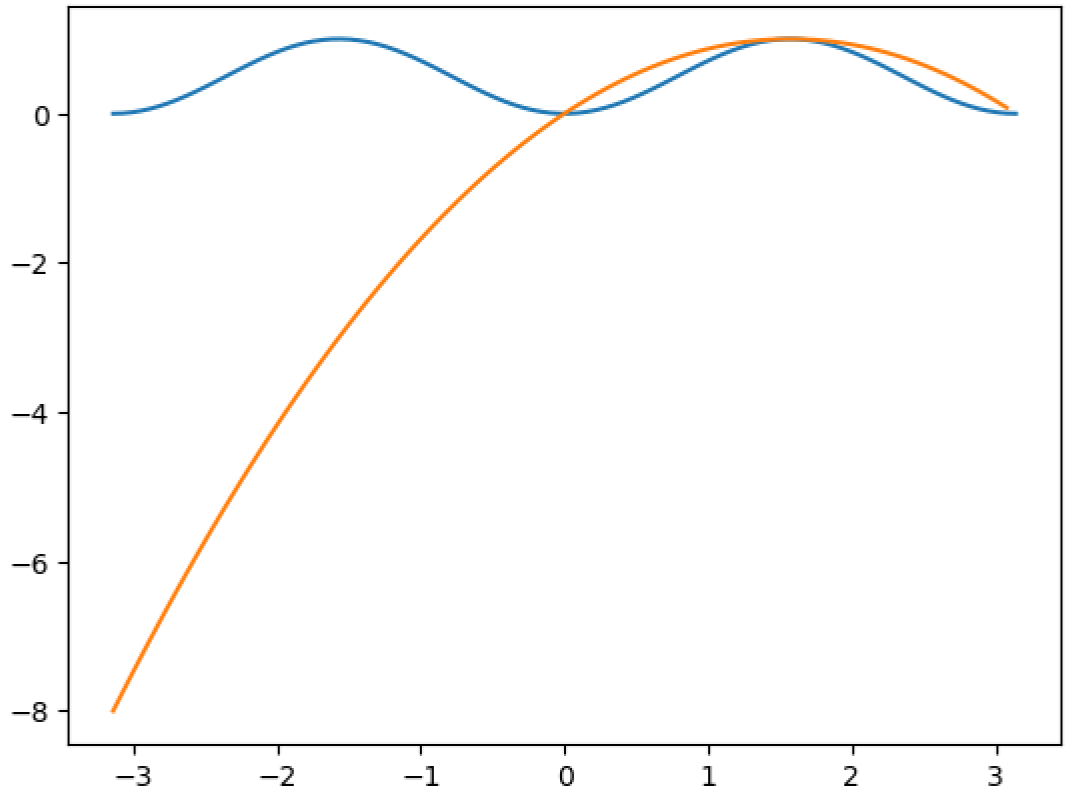


To improve this approximation with the fewest number of extra points, I would add points at x = -$\frac{\pi}{2}$ and $\frac{\pi}{2}$. These two points will significantly help the interpolation hold closely to the curve.

b.)

In [ ]:
```
xArrBigger = np.linspace(-np.pi,np.pi,100)
plt.plot(xArrBigger,f(xArrBigger),'-')
xArrBigger = [x for x in xArrBigger if x not in xs]
plt.plot(xArrBigger,p(xArrBigger),'-')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x1fd87d6c0d0>]
```



To improve this interpolation, I would add symmetric points, x = -x. This should help the curve come back up and down for increasingly negative xs.

In general, to improve an interpolation, add extra points where the interpolation is visually very different from the original function.

4.) a.)

Using Newton Bases,

$u(x) = (\gamma_0 + \gamma_1 x + \gamma_2 x^2)^{\frac{1}{2}}$

$v(x) = u(x)^2 = \gamma_0 + \gamma_1 x + \gamma_2 x^2$

$\phi_0 = 1, \phi_0 = (x - x_0), \phi_0 = (x - x_0)(x - x_1)$

$v(x) = y_0^2 + \frac{y_1^2 - y_0^2}{x_1 - x_0}(x - x_0) + \frac{\frac{y_2^2 - y_1^2}{x_2 - x_1} - \gamma_1}{x_2 - x_0}(x - x_0)(x - x_1)$

$u(x) = \sqrt{v(x)} = \sqrt{y_0^2 + \frac{y_1^2 - y_0^2}{x_1 - x_0}(x - x_0) + \frac{\frac{y_2^2 - y_1^2}{x_2 - x_1} - \gamma_1}{x_2 - x_0}(x - x_0)(x - x_1)}$

b.)

In [ ]:
```
pts = [(0.1,0.5), (1,1), (2,4)]

def u(x):
```

```
    gamma1 = (pts[1][1]**2 - pts[0][1]**2/(pts[1][0]-pts[0][0]))
    return(np.sqrt(pts[0][1]**2 + gamma1*(x-pts[0][0]) + (((pts[2][1]**2 - pts[1][1]**

xArr = np.linspace(0,2,100)
plt.plot(xArr, u(xArr),'-')
plt.show()
```

```
C:\Users\Cameron Holland\AppData\Local\Temp\ipykernel_19380\3740751931.py:5: RuntimeW
arning: invalid value encountered in sqrt
  return(np.sqrt(pts[0][1]**2 + gamma1*(x-pts[0][0]) + (((pts[2][1]**2 - pts[1][1]**
2)/(pts[2][0] - pts[1][0]) - gamma1)/(pts[2][0] - pts[0][0]))*(x-pts[0][0])*(x-pts[1]
[0])))
```