In [23]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from IPython.display import display, display_html
from itertools import chain, cycle
```

In [24]:

```
#display function from https://stackoverflow.com/a/44923103
def side_by_side(*args,titles=cycle(['']),header_size=2):
    html_str=''
    for df,title in zip(args, chain(titles,cycle(['</br>'])) ):
        html_str+='<th style="text-align:center;"><td style="vertical-align:top">'
        html_str+=f'<h{header_size}>{title}</h{header_size}>'
        html_str+=df.to_html().replace('table','table style="display:inline"')
        html_str+='</td></th>'
    display_html(html_str,raw=True)
```

Let $\Delta_h[f](x)$ be the Forward Difference of f on x with step size h

Let $\nabla_h[f](x)$ be the Backwad Difference of f on x with step size h

Let $\delta_h[f](x)$ be the Central Difference of f on x with step size h

Using the following Finite Difference Equations:

	$O(h)$	$O(h^2)$	$O(h^4)$
$\Delta_h[f](x)$	$\frac{f(x+h)-f(x)}{h}$	$\frac{-f(x+2h)+4f(x+h)-3f(x)}{2h}$	
$\nabla_h[f](x)$	$\frac{f(x)-f(x-h)}{h}$	$\frac{3f(x)-4f(x-h)-f(x-2h)}{2h}$	
$\delta_h[f](x)$		$\frac{f(x+h)-f(x-h)}{2h}$	$\frac{-f(x+2h)+8f(x+h)-8f(x-h)+f(x-2h)}{12h}$

In [25]:

```
#implement the Finite Difference Equations as lambdas
fd_h = lambda f,x,h: (f(x+h)-f(x))/h
fd_h2 = lambda f,x,h: (-f(x+2*h)+4*f(x+h)-3*f(x))/(2*h)

bd_h = lambda f,x,h: (f(x)-f(x-h))/h
bd_h2 = lambda f,x,h: (3*f(x)-4*f(x-h)+f(x-2*h))/(2*h)

cd_h2 = lambda f,x,h: (f(x+h)-f(x-h))/(2*h)
cd_h4 = lambda f,x,h: (-f(x+2*h) +8*f(x+h)-8*f(x-h)+f(x-2*h))/(12*h)
```

23.1 Compute forward and backward difference approximations of $O(h)$ and $O(h^2)$, and central difference approximations of $O(h^2)$ and $O(h^4)$ for the first derivative of $y = \cos x$ at $x = \pi/4$ using a value of $h = \pi/12$. Estimate the true percent relative error ε_t for each approximation.

In [29]:

```
#define f,x,h and true value(tv)
f = lambda x: np.cos(x)
x = np.pi/4
h = np.pi/12
tv = -0.707107

#calculate Forward Difference of O(h) and O(h^2)
fd = [fd_h(f,x,h),fd_h2(f,x,h)]
#calculate Backward Difference of O(h) and O(h^2)
bd = [bd_h(f,x,h),bd_h2(f,x,h)]
#calculate Central Difference of O(h^2) and O(h^4)
cd = [cd_h2(f,x,h),cd_h4(f,x,h)]

#add the resualts into a dataframe df with the notation for each Finite Difference
df = pd.DataFrame({"$$\Delta_{h} [cos](x)$$":fd,r"$$\nabla_{h} [cos](x)$$":bd,"$$\de

#calcuate the %error %Et for each Finite Difference approximation
df["$$\epsilon_{t,\Delta}$$"] = df["$$\Delta_{h} [cos](x)$$"].apply(lambda x:f"{(tv-

df[r"$$\epsilon_{t,\nabla}$$"] = df[r"$$\nabla_{h} [cos](x)$$"].apply(lambda x:f"{(t

df["$$\epsilon_{t,\delta}$$"] = df["$$\delta_{h} [cos](x)$$"].apply(lambda x:f"{(tv-
#extract the resualts and error from df to a new dataframe for each Finite Differenc

fd_df = df[["$$\Delta_{h} [cos](x)$$","$$\epsilon_{t,\Delta}$$"]]
fd_df=fd_df.rename(index={0: "$O(h)$", 1: "$O(h^2)$"})

bd_df = df[[r"$$\nabla_{h} [cos](x)$$",r"$$\epsilon_{t,\nabla}$$"]]
bd_df=bd_df.rename(index={0: "$O(h)$", 1: "$O(h^2)$"})

cd_df = df[["$$\delta_{h} [cos](x)$$","$$\epsilon_{t,\delta}$$"]]
cd_df=cd_df.rename(index={0: "$O(h^2)$", 1: "$O(h^4)$"})

#display side by side
side_by_side(fd_df,bd_df,cd_df,titles=["Forward Difference","Backward Difference","C
```

Forward Difference

	$\Delta_h[cos](x)$	$\epsilon_{t,\Delta}$
$O(h)$	-0.791090	-11.8769%
$O(h^2)$	-0.726013	-2.67368%

Backward Difference

	$\nabla_h[cos](x)$	$\epsilon_{t,\nabla}$
$O(h)$	-0.607024	14.153809%
$O(h^2)$	-0.719741	-1.786700%

Central Difference

	$\delta_h[cos](x)$	$\epsilon_{t,\delta}$
$O(h^2)$	-0.699057	1.138438%
$O(h^4)$	-0.706997	0.015562%

23.4 Use Richardson extrapolation to estimate the first derivative of $y = \cos x$ at $x = \pi/4$ using step sizes of $h_1 = \pi/3$ and $h_2 = \pi/6$. Employ centered differences of $O(h^2)$ for the initial estimates.

In [27]:

```
#define f,x,h1,h2 and true value(tv)
f = lambda x: np.cos(x)
x = np.pi/4
h1 = np.pi/3
h2 = np.pi/6
tv= -0.707107

#calculate Central Difference of O(h^2) and O(h^4)
cd = [cd_h2(f,x,h1),cd_h2(f,x,h2)]

#put the resualt into a dataframe cd_df and calcuate the error Et for each approxima
cd_df = pd.DataFrame({"$$\delta_{h} [cos](x)$$":cd})
cd_df["$$\epsilon_{t,\delta}$$"] = cd_df["$$\delta_{h} [cos](x)$$"].apply(lambda x:(

#rename the index according to h
cd_df= cd_df.rename(index={0: r"$h =\frac{\pi}{3}$",1: r"$h =\frac{\pi}{6}$"})

#calcualte Richardson Extrapolation(re) and re error Et and put them into a datafram
re_df = pd.DataFrame({"$$R(\delta_{h_1},\delta_{h_2})$$":[(4*cd[1]-cd[0])/3]})
re_df["$$\epsilon_{t,R}$$"] = re_df["$$R(\delta_{h_1},\delta_{h_2})$$"].apply(lambda

#hide the index by renameing since it is not needed.
re_df= re_df.rename(index={0:""})

#display side by side
side_by_side(cd_df,re_df,titles=["Central Difference","Richardson Extrapolation"],he
```

Central Difference

	$\delta_h[cos](x)$	$\epsilon_{t,\delta}$
$h = \frac{\pi}{3}$	-0.584773	0.173007
$h = \frac{\pi}{6}$	-0.675237	0.045071

Richardson Extrapolation

$R(\delta_{h_1}, \delta_{h_2})$	$\epsilon_{t,R}$
-0.705392	0.002425

23.8 Compute the first-order central difference approximations of $O(h^4)$ for each of the following functions at the specified location and for the specified step size:

(a) $y = x^3 + 4x - 15$ at $x = 0$, $h = 0.25$
(b) $y = x^2 \cos x$ at $x = 0.4$, $h = 0.1$
(c) $y = \tan(x/3)$ at $x = 3$, $h = 0.5$
(d) $y = \sin(0.5\sqrt{x})/x$ at $x = 1$, $h = 0.2$
(e) $y = e^x + x$ at $x = 2$, $h = 0.2$

Compare your results with the analytical solutions.

In [30]:

```
"""
F:  functions expressions as lambda for calculation
Y:  functions expressions as latex for display
X:  specified x values for function
H:  specified h values for function
TV: true value for each function at specified x
"""
#define F,Y,X,H,TV lists
F = [lambda x: x**3 +4*x-15,
     lambda x: x**2 * np.cos(x),
     lambda x: np.tan(x/3),
     lambda x:np.sin(0.5*np.sqrt(x))/x,
     lambda x:np.exp(x)+x]

Y = [r"$$x^3+4x-15$$",
     r"$x^2cosx$",
     r"$tan(\frac {x}{3})$",
     r"$\frac{sin(0.5\sqrt{x})}{x}$",
     r"$e^x + x$"]

X  = [0,0.4,3,1,2]
H  = [0.25,0.1,0.5,0.2,0.2]
TV = [4,0.6745,1.1418,-0.2600,8.3891]

#put them in a dataframe df
df = pd.DataFrame({"y":Y,"f":F,"x":X,"h":H,"True Value":TV})

#calculate Central Difference of O(h^4) for each f according to the specified x and
df["$$\delta_h [y](x)$$"] = df.apply(lambda d: cd_h4(d["f"],d["x"],d["h"]),axis=1)
#calculate Central Difference error Et for each approximations for comparison
df["$$\epsilon_{t,\delta_h}$$"]= df.apply(lambda d:(d["True Value"]-d["$$\delta_h [y

#drop the lambda functions expressions column before display
df.drop(['f'], axis=1, inplace=True)
#hide the df table index column for better looking table
df = df.style.hide_index()
#display df
display(df)
```

y	x	h	True Value	$\delta_h[y](x)$	ϵ_{t,δ_h}
$x^3 + 4x - 15$	0.000000	0.250000	4.000000	4.000000	0.000000
$x^2 cosx$	0.400000	0.100000	0.674500	0.674504	-0.000006
$tan(\frac{x}{3})$	3.000000	0.500000	1.141800	1.092486	0.043190
$\frac{sin(0.5\sqrt{x})}{x}$	1.000000	0.200000	-0.260000	-0.259081	0.003535
$e^x + x$	2.000000	0.200000	8.389100	8.388660	0.000052