RISK_Mng_HW5_Duanhong Gao(dg2896)

October 13, 2016

In [17]:
```
########## HW5 Problem 2 differnet lambda values  ############
import numpy as np
import scipy.stats as ss
import pandas as pd
import math
import matplotlib.pyplot as plt
import datetime as dt
import plotly.graph_objs as go
import plotly.plotly as py

path = '~/Documents/Semester3/M5320/homework/HW4/AMD-yahoo.csv'
sto_AMD= pd.read_csv(path, header = 0)
sto_AMD.shape
#data1 = sto_AMD.values[1:252*20,:]
data1 = sto_AMD.values
#data1 = data1[::-1]
AMD_close = list(data1[:,6])

## exponential weighting paramater lambda
lambda1 = 0.998614
lambda2 = 0.998723
lambda3 = 0.999362
lambda4 = 0.94
lambda5 = 0.97

### List the lambda values

def list_lambdas(lambda_k):
    list_lambda = []
    for i in range(len(data1)):
        lambda_value = (lambda_k**i)
        list_lambda.append(lambda_value)
    return(list_lambda)

list_lambda1 = list_lambdas(lambda1)
list_lambda2 = list_lambdas(lambda2)
list_lambda3 = list_lambdas(lambda3)
list_lambda4 = list_lambdas(lambda4)
list_lambda5 = list_lambdas(lambda5)

def weigthed_vol_and_mu(list_lambda, years):
    wgt_A_vol_lambda = []
    wgt_A_mu_lambda = []
```

```
    wgt_log_rtn = []
    wgt_log_rtn_sq = []

    for i in range( len(data1)-1 ):
        log_return = math.log( AMD_close[i]/AMD_close[i+1] )
        wgt_log_return = log_return *  list_lambda[i]
        wgt_log_rtn.append(wgt_log_return)

        log_return_sq = log_return ** 2
        wgt_log_rtn_sq.append( log_return_sq * list_lambda[i] )

    for j in range(len(data1)-252*years):
        wgt_mu_lambda =  sum(wgt_log_rtn[j:j+252*years])/sum(list_lambda[j:j+252*years])
        wgt_vol_lambda = sum(wgt_log_rtn_sq[j:j+252*years])/sum(list_lambda[j:j+252*years])- w
        wgt_A_vol_lambda.append(wgt_vol_lambda)
        wgt_A_mu_lambda.append(wgt_mu_lambda)

    return(wgt_A_vol_lambda, wgt_A_mu_lambda )

years = 5
wgt_A_vol_lambda1 = weigthed_vol_and_mu(list_lambda1, years)[0][1: 252*20]
wgt_A_mu_lambda1 = weigthed_vol_and_mu(list_lambda1, years)[1][1: 252*20]

wgt_A_vol_lambda2 = weigthed_vol_and_mu(list_lambda2, years)[0][1: 252*20]
wgt_A_mu_lambda2 = weigthed_vol_and_mu(list_lambda2, years)[1][1: 252*20]

wgt_A_vol_lambda3 = weigthed_vol_and_mu(list_lambda3, years)[0][1: 252*20]
wgt_A_mu_lambda3 = weigthed_vol_and_mu(list_lambda3, years)[1][1: 252*20]

wgt_A_vol_lambda4 = weigthed_vol_and_mu(list_lambda4, years)[0][1: 252*20]
wgt_A_mu_lambda4 = weigthed_vol_and_mu(list_lambda4, years)[1][1: 252*20]

wgt_A_vol_lambda5 = weigthed_vol_and_mu(list_lambda5, years)[0][1: 252*20]
wgt_A_mu_lambda5 = weigthed_vol_and_mu(list_lambda5, years)[1][1: 252*20]




timeline = data1[1:252*20,0]
timeline = [dt.datetime.strptime(d,'%Y-%m-%d').date() for d in timeline]


fig, ax = plt.subplots()
ax.plot(timeline, wgt_A_vol_lambda1, 'r-', label='lambda1 = 0.998614')
ax.plot(timeline, wgt_A_vol_lambda2, 'g-', label='lambda2 = 0.998723')
ax.plot(timeline, wgt_A_vol_lambda3, 'y-', label='lambda3 = 0.999362')
ax.plot(timeline, wgt_A_vol_lambda4, 'b-', label='lambda4 = 0.94')
ax.plot(timeline, wgt_A_vol_lambda5, 'm-', label='lambda4 = 0.97')


legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('AMD vols')
```

```
fig, ax = plt.subplots()
ax.plot(timeline, wgt_A_mu_lambda1, 'r-', label='lambda1 = 0.998614')
ax.plot(timeline, wgt_A_mu_lambda2,'g-', label='lambda2 = 0.998723')
ax.plot(timeline, wgt_A_mu_lambda3,'y-', label='lambda3 = 0.999362')
ax.plot(timeline, wgt_A_mu_lambda3,'b-', label='lambda4 = 0.94')
ax.plot(timeline, wgt_A_mu_lambda3,'m-', label='lambda4 = 0.97')

legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('AMD mu')
plt.show()
```

In [1]:
```
##### HW5 Problem 3 ###### PART 1: unweighted mu and vols for VaR and ES #######
import numpy as np
import scipy.stats as ss
import pandas as pd
import math
import matplotlib.pyplot as plt
import datetime as dt


path = '~/Documents/Semester3/M5320/homework/HW4/INTC-yahoo.csv'
sto_AMD= pd.read_csv(path, header = 0)
sto_AMD.shape
#data1 = sto_AMD.values[1:252*20,:]
data1 = sto_AMD.values
#data1 = data1[::-1]
AMD_close = list(data1[:,6])


A_log_rtn = []
A_log_rtn_sq = []

for i in range(1 , len(data1)-1):
    log_return = math.log( AMD_close[i]/AMD_close[i+1] )
    A_log_rtn.append(log_return)
    A_log_rtn_sq.append(log_return**2)

def vol_and_mu(years):
    S0 = 10000
    T = 5/252
    p = 0.99

    A_vol_years = []
    A_mu_years = []
    A_VaR_years = []
    A_ES_years =[]


    for i in range(len(data1)-252*years):
        vol_years = np.std(A_log_rtn[i:i+252*years]) * np.sqrt(252)
        mu_years = np.mean(A_log_rtn[i:i+252*years])*252 + (vol_years**2)/2
        VaR_years = S0 - S0 * np.exp( vol_years * T**(0.5)* ss.norm.ppf(1-p) + (mu_years - pow(
        A_vol_years.append(vol_years)
        A_mu_years.append(mu_years)
```

```
            A_VaR_years.append(VaR_years)


            if i<6:
                ES_years = VaR_years
            else:
                sum_loss = []
                sum_loss_value = 0
                for k in range(4):
                    sum_loss_value = sum_loss_value + A_VaR_years[i-k]
                    sum_loss.append(sum_loss_value/(k+1))
                    s = pd.Series(sum_loss)
                    ES_years = s.quantile(.975)
            A_ES_years.append(ES_years)

        return(A_vol_years, A_mu_years, A_VaR_years, A_ES_years)


    A_VaR_2years = vol_and_mu(2)[2][1:252*20]
    A_ES_2years = vol_and_mu(2)[3][1:252*20]


    A_VaR_5years = vol_and_mu(5)[2][1:252*20]
    A_ES_5years = vol_and_mu(5)[3][1:252*20]

    A_VaR_10years = vol_and_mu(10)[2][1:252*20]
    A_ES_10years = vol_and_mu(10)[3][1:252*20]



    timeline = data1[1:252*20,0]
    timeline = [dt.datetime.strptime(d,'%Y-%m-%d').date() for d in timeline]

    fig, ax = plt.subplots()
    ax.plot(timeline, A_VaR_2years, 'y-', label='VaR 2 year')
    ax.plot(timeline, A_VaR_5years,'g-', label='VaR 5 year')
    ax.plot(timeline, A_VaR_10years,'m-', label='VaR 10 year')
    ax.plot(timeline, A_ES_2years,'r-', label='ES 2 year')
    ax.plot(timeline, A_ES_5years,'c-', label='ES 5 year')
    ax.plot(timeline, A_ES_10years,'b-', label='ES 10 year')


    legend = ax.legend(loc='upper right', shadow=True)
    ax.set_title('INTC VaRs and ESs Windowed')
    plt.show()
```

In [3]:
```
########## HW5 Problem3 ############ PART 2: weighted VaRs and ES
import numpy as np
import scipy.stats as ss
import pandas as pd
import math
import matplotlib.pyplot as plt
import datetime as dt
```

```python
path = '~/Documents/Semester3/M5320/homework/HW4/INTC-yahoo.csv'
sto_AMD= pd.read_csv(path, header = 0)
sto_AMD.shape
data1 = sto_AMD.values
AMD_close = list(data1[:,6])

## exponential weighting paramater lambda
lambda1 = 0.998614


### List the lambda values

def list_lambdas(lambda_k):
    list_lambda = []
    for i in range(len(data1)):
        lambda_value = (lambda_k**i)
        list_lambda.append(lambda_value)
    return(list_lambda)

list_lambda1 = list_lambdas(lambda1)


def weigthed_VaR_and_ES(list_lambda, years):
    wgt_A_vol_years = []
    wgt_A_mu_years = []

    wgt_log_rtn = []
    wgt_log_rtn_sq = []



    for i in range( len(data1)-1 ):
        log_return = math.log( AMD_close[i]/AMD_close[i+1] )
        wgt_log_return = log_return *  list_lambda[i]
        wgt_log_rtn.append(wgt_log_return)

        log_return_sq = log_return ** 2
        wgt_log_rtn_sq.append( log_return_sq * list_lambda[i] )


    S0 = 10000
    T = 5/252
    p = .99


    wgt_A_VaR_years = []
    wgt_A_ES_years =[]

    for j in range(len(data1)-252*years):
        wgt_mu_lambda = sum(wgt_log_rtn[j:j+252*years])/sum(list_lambda[j:j+252*years])
        wgt_vol_lambda = sum(wgt_log_rtn_sq[j:j+252*years])/sum(list_lambda[j:j+252*years])- wg
        wgt_VaR_years = S0 - S0 * np.exp( wgt_vol_lambda * T**(0.5)* ss.norm.ppf(1-p) + (wgt_mu
        wgt_A_vol_years.append(wgt_vol_lambda)
        wgt_A_mu_years.append(wgt_mu_lambda)
```

```
            wgt_A_VaR_years.append(wgt_VaR_years)

            if j<6:
                wgt_ES_years = wgt_VaR_years
            else:
                wgt_sum_loss = []
                wgt_sum_loss_value = 0
                for k in range(4):
                    wgt_sum_loss_value = wgt_sum_loss_value + wgt_A_VaR_years[j-k]
                    wgt_sum_loss.append(wgt_sum_loss_value/(k+1))
                    s = pd.Series(wgt_sum_loss)
                    wgt_ES_years = s.quantile(.975)
            wgt_A_ES_years.append(wgt_ES_years)

        return(wgt_A_VaR_years, wgt_A_ES_years)



    years_2 = 2
    wgt_A_VaR_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[0][1: 252*20]
    wgt_A_ES_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[1][1: 252*20]

    years_5 = 5
    wgt_A_VaR_5years = weigthed_VaR_and_ES(list_lambda1, years_5)[0][1: 252*20]
    wgt_A_ES_5years = weigthed_VaR_and_ES(list_lambda1, years_5)[1][1: 252*20]

    years_10 = 10
    wgt_A_VaR_10years = weigthed_VaR_and_ES(list_lambda1, years_10)[0][1: 252*20]
    wgt_A_ES_10years = weigthed_VaR_and_ES(list_lambda1, years_10)[1][1: 252*20]


    timeline = data1[1:252*20,0]
    timeline = [dt.datetime.strptime(d,'%Y-%m-%d').date() for d in timeline]


    fig, ax = plt.subplots()
    ax.plot(timeline, wgt_A_VaR_2years, 'r-', label='VaR 2 year')
    ax.plot(timeline, wgt_A_ES_2years, 'y-', label='ES 2 year')

    ax.plot(timeline, wgt_A_VaR_5years, 'b-', label='VaR 5 year')
    ax.plot(timeline, wgt_A_ES_5years, 'g-', label='ES 5 year')

    ax.plot(timeline, wgt_A_VaR_10years, 'm-', label='VaR 10 year')
    ax.plot(timeline, wgt_A_ES_10years, 'c-', label='ES 10 year')


    legend = ax.legend(loc='upper right', shadow=True)
    ax.set_title('INTC VaRs and ESs Exponential Windowed Equivalent')

    plt.show()
```

In [5]:
```
########## HW5 Problem3 ############ PART 3:  VaRs and ES exponential lamda =0.94 /0.97
import numpy as np
import scipy.stats as ss
```

```python
import pandas as pd
import math
import matplotlib.pyplot as plt
import datetime as dt

path = '~/Documents/Semester3/M5320/homework/HW4/INTC-yahoo.csv'
sto_AMD= pd.read_csv(path, header = 0)
sto_AMD.shape
data1 = sto_AMD.values
AMD_close = list(data1[:,6])

## exponential weighting paramater lambda
lambda1 = 0.94
lambda2 = 0.97

### List the lambda values

def list_lambdas(lambda_k):
    list_lambda = []
    for i in range(len(data1)):
        lambda_value = (lambda_k**i)
        list_lambda.append(lambda_value)
    return(list_lambda)

list_lambda1 = list_lambdas(lambda1)
list_lambda2 = list_lambdas(lambda2)

def weigthed_VaR_and_ES(list_lambda, years):
    wgt_A_vol_years = []
    wgt_A_mu_years = []

    wgt_log_rtn = []
    wgt_log_rtn_sq = []


    for i in range( len(data1)-1 ):
        log_return = math.log( AMD_close[i]/AMD_close[i+1] )
        wgt_log_return = log_return *  list_lambda[i]
        wgt_log_rtn.append(wgt_log_return)

        log_return_sq = log_return ** 2
        wgt_log_rtn_sq.append( log_return_sq * list_lambda[i] )

    S0 = 10000
    T = 5/252
    p = .99

    wgt_A_VaR_years = []
    wgt_A_ES_years =[]

    for j in range(len(data1)-252*years):
        wgt_mu_lambda = sum(wgt_log_rtn[j:j+252*years])/sum(list_lambda[j:j+252*years])
        wgt_vol_lambda = sum(wgt_log_rtn_sq[j:j+252*years])/sum(list_lambda[j:j+252*years])- wg
        wgt_VaR_years = S0 - S0 * np.exp( wgt_vol_lambda * T**(0.5)* ss.norm.ppf(1-p) + (wgt_mu_
```

```
        wgt_A_vol_years.append(wgt_vol_lambda)
        wgt_A_mu_years.append(wgt_mu_lambda)
        wgt_A_VaR_years.append(wgt_VaR_years)

        if j<6:
            wgt_ES_years = wgt_VaR_years
        else:
            wgt_sum_loss = []
            wgt_sum_loss_value = 0
            for k in range(4):
                wgt_sum_loss_value = wgt_sum_loss_value + wgt_A_VaR_years[j-k]
                wgt_sum_loss.append(wgt_sum_loss_value/(k+1))
                s = pd.Series(wgt_sum_loss)
                wgt_ES_years = s.quantile(.975)
        wgt_A_ES_years.append(wgt_ES_years)

    return(wgt_A_VaR_years, wgt_A_ES_years)



years_2 = 2
wgt_A_VaR_lambda1 = weigthed_VaR_and_ES(list_lambda1, years_2)[0][1: 252*20]
wgt_A_ES_lambda1 = weigthed_VaR_and_ES(list_lambda1, years_2)[1][1: 252*20]


wgt_A_VaR_lambda2 = weigthed_VaR_and_ES(list_lambda2, years_2)[0][1: 252*20]
wgt_A_ES_lambda2 = weigthed_VaR_and_ES(list_lambda2, years_2)[1][1: 252*20]


timeline = data1[1:252*20,0]
timeline = [dt.datetime.strptime(d,'%Y-%m-%d').date() for d in timeline]


fig, ax = plt.subplots()
ax.plot(timeline, wgt_A_VaR_lambda1, 'r-', label='VaR lambda=0.94')
ax.plot(timeline, wgt_A_ES_lambda1, 'y-', label='ES lambda=0.94')

ax.plot(timeline, wgt_A_VaR_lambda2, 'b-', label='VaR lambda=0.97')
ax.plot(timeline, wgt_A_ES_lambda2, 'g-', label='ES lambda=0.97')

legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('INTC VaRs and ESs Exponential')

plt.show()
```

In [6]:
```
########## HW5 Problem3 ############ PART 5:  VaR 2yr Window vs Equivalent Exponential
import numpy as np
import scipy.stats as ss
import pandas as pd
import math
import matplotlib.pyplot as plt
import datetime as dt

path = '~/Documents/Semester3/M5320/homework/HW4/INTC-yahoo.csv'
```

```
sto_AMD= pd.read_csv(path, header = 0)
sto_AMD.shape
data1 = sto_AMD.values
AMD_close = list(data1[:,6])


A_log_rtn = []
A_log_rtn_sq = []

for i in range(1 , len(data1)-1):
    log_return = math.log( AMD_close[i]/AMD_close[i+1] )
    A_log_rtn.append(log_return)
    A_log_rtn_sq.append(log_return**2)

def vol_and_mu(years):
    S0 = 10000
    T = 5/252
    p = 0.99

    A_vol_years = []
    A_mu_years = []
    A_VaR_years = []
    A_ES_years =[]


    for i in range(len(data1)-252*years):
        vol_years = np.std(A_log_rtn[i:i+252*years]) * np.sqrt(252)
        mu_years = np.mean(A_log_rtn[i:i+252*years])*252 + (vol_years**2)/2
        VaR_years = S0 - S0 * np.exp( vol_years * T**(0.5)* ss.norm.ppf(1-p) + (mu_years - pow(v
        A_vol_years.append(vol_years)
        A_mu_years.append(mu_years)
        A_VaR_years.append(VaR_years)


        if i<6:
            ES_years = VaR_years
        else:
            sum_loss = []
            sum_loss_value = 0
            for k in range(4):
                sum_loss_value = sum_loss_value + A_VaR_years[i-k]
                sum_loss.append(sum_loss_value/(k+1))
                s = pd.Series(sum_loss)
                ES_years = s.quantile(.975)
        A_ES_years.append(ES_years)

    return(A_VaR_years, A_ES_years, A_mu_years, A_vol_years)


A_VaR_2years = vol_and_mu(10)[0][1:252*20]

##################################################################################

## exponential weighting paramater lambda
```

```
lambda1 = 0.998614

### List the lambda values

def list_lambdas(lambda_k):
    list_lambda = []
    for i in range(len(data1)):
        lambda_value = (lambda_k**i)
        list_lambda.append(lambda_value)
    return(list_lambda)

list_lambda1 = list_lambdas(lambda1)


def weigthed_VaR_and_ES(list_lambda, years):
    wgt_A_vol_years = []
    wgt_A_mu_years = []

    wgt_log_rtn = []
    wgt_log_rtn_sq = []



    for i in range( len(data1)-1 ):
        log_return = math.log( AMD_close[i]/AMD_close[i+1] )
        wgt_log_return = log_return *  list_lambda[i]
        wgt_log_rtn.append(wgt_log_return)

        log_return_sq = log_return ** 2
        wgt_log_rtn_sq.append( log_return_sq * list_lambda[i] )


    S0 = 10000
    T = 5/252
    p = .99

    wgt_A_VaR_years = []
    wgt_A_ES_years =[]

    for j in range(len(data1)-252*years):
        wgt_mu_lambda = 100* sum(wgt_log_rtn[j:j+252*years])/sum(list_lambda[j:j+ 252*years])
        wgt_vol_lambda =((sum(wgt_log_rtn_sq[j:j+252*years])/sum(list_lambda[j:j+ 252*years]))-
        wgt_VaR_years = S0 - S0 * np.exp(wgt_vol_lambda * T**(0.5)* ss.norm.ppf(1-p) + (wgt_mu_
        wgt_A_vol_years.append(wgt_vol_lambda)
        wgt_A_mu_years.append(wgt_mu_lambda)
        wgt_A_VaR_years.append(wgt_VaR_years)

        if j<6:
            wgt_ES_years = wgt_VaR_years
        else:
            wgt_sum_loss = []
            wgt_sum_loss_value = 0
            for k in range(4):
                wgt_sum_loss_value = wgt_sum_loss_value + wgt_A_VaR_years[j-k]
```

```
                wgt_sum_loss.append(wgt_sum_loss_value/(k+1))
                s = pd.Series(wgt_sum_loss)
                wgt_ES_years = s.quantile(.975)
        wgt_A_ES_years.append(wgt_ES_years)

    return(wgt_A_VaR_years, wgt_A_ES_years)



years_2 = 10
wgt_A_VaR_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[0][1: 252*20]

timeline = data1[1:252*20,0]
timeline = [dt.datetime.strptime(d,'%Y-%m-%d').date() for d in timeline]


fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax2.plot(timeline, wgt_A_VaR_2years, 'r-', label='VaR 10yr Exp Equiv')
ax1.plot(timeline, A_VaR_2years, 'b-', label='VaR 10yr windowed')

legend1 = ax1.legend(loc='upper right', shadow=True)
legend2 = ax2.legend(loc='upper right', shadow=True)
ax1.set_title('INTC VaR 10yr Window vs Equivalent Exponential')
plt.show()
```

In [9]:
```
########## HW5 Problem 4 ########### PART 1: Phi AND D_Phi
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt
import pylab
def norm_func(x,h):
    pdf = norm.pdf(x)
    cdf = norm.cdf(x)
    D_Phi = (norm.cdf(x+h)-norm.cdf(x-h))/(2*h)
    DErr = norm.pdf(x) - D_Phi
    first_derivative = np.gradient(cdf)
    second_derivative = np.gradient(first_derivative)
    return(pdf, D_Phi, DErr, second_derivative)

x = np.linspace(-3,3, 100)
h1 = 1e-04
fun1 = norm_func(x, h1)
h2 = 1e-05
fun2 = norm_func(x, h2)
h3 = 1e-06
fun3 = norm_func(x, h3)

fig, ax1 = plt.subplots()
ax1.plot(x, fun1[0],'r-', lw=1, alpha=0.6, label='phi')

ax1.plot(x, fun1[1],'b-', lw=1, label='d(Phi,h=1e^-04)')

ax1.plot(x, fun2[1], 'g-',lw=1, label='d(Phi,h=1e^-05)')
```

```
ax1.plot(x, fun3[1], 'm-',lw=1, label='d(Phi,h=1e^-06)')

legend1 = ax1.legend(loc='upper right', shadow=True)
ax1.set_title('phi vs dPhi(h)')
plt.show()
```

In [3]:
```
########## HW5 Problem 4 ########### PART 2: phi - dPhi(h)
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt
import pylab
def norm_func(x,h):
    pdf = norm.pdf(x)
    cdf = norm.cdf(x)
    D_Phi = (norm.cdf(x+h)-norm.cdf(x-h))/(2*h)
    DErr = norm.pdf(x) - D_Phi
    first_derivative = np.gradient(cdf)
    second_derivative = np.gradient(first_derivative)
    return(pdf, D_Phi, DErr, second_derivative)

x = np.linspace(-3,3, 100)
h1 = 1e-04
fun1 = norm_func(x, h1)
h2 = 1e-05
fun2 = norm_func(x, h2)
h3 = 1e-06
fun3 = norm_func(x, h3)

fig, ax1 = plt.subplots()
ax1.plot(x, fun1[2],'r-', lw=10, label='DErr(h=1e^-04)')

#ax1.plot(x, fun1[1],'b-', lw=1, label='DErr(h=1e^-04)')

ax1.plot(x, fun2[2], 'g-',lw=2, label='DErr(h=1e^-05)')
ax1.plot(x, fun3[2], 'm-',lw=2, label='DErr(h=1e^-06)')

legend1 = ax1.legend(loc='upper right', shadow=True)
ax1.set_title('DErr = phi - dPhi(h)')
plt.show()
```

In [4]:
```
########## HW5 Problem 4 ########### PART 3: Second Derivatives
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt
import pylab
def norm_func(x,h):
    pdf = norm.pdf(x)
    cdf = norm.cdf(x)
    D_Phi = (norm.cdf(x+h)-norm.cdf(x-h))/(2*h)
    DErr = norm.pdf(x) - D_Phi
    first_derivative = np.gradient(cdf)
    second_derivative = np.gradient(first_derivative)
    return(pdf, D_Phi, DErr, second_derivative)

x = np.linspace(-3,3, 100)
```

```
h1 = 1e-05
fun1 = norm_func(x, h1)
h2 = 1e-06
fun2 = norm_func(x, h2)
h3 = 1e-07
fun3 = norm_func(x, h3)

fig, ax1 = plt.subplots()
ax1.plot(x, fun1[3],'m-', lw=1, label='ddPhi(x,1e-05)')
ax1.plot(x, fun2[3],'g-',lw=2, label='ddPhi(x,1e-06)')
ax1.plot(x, fun3[3], 'r-', label='ddPhi(x,1e-07)')
legend = ax1.legend(loc='upper right', shadow=True)
ax1.set_title('Second derivatives ')
plt.show()
```

```
In [5]: ########## HW5 Problem 5 ########### PART I : BS CALL And Delta
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt
import pylab
import numpy as np
import scipy.stats as ss
import time

class Solution():
    def __init__(self, type1, S0, K, r, sigma, T):
        self.type1= type1
        self.S0=S0
        self.K=K
        self.r=r
        self.sigma=sigma
        self.T=T

    def d1(self):
        return (np.log(self.S0/self.K) + (self.r + self.sigma**2 / 2) * T)/(self.sigma * np.sqr

    def d2(self):

        return (np.log(self.S0 / self.K) + (self.r - self.sigma**2 / 2) * self.T) / (self.sigma

    def BlackScholes(self):
        if self.type1=="C":
            return self.S0 * ss.norm.cdf(self.d1()) - self.K * np.exp(-self.r * self.T) * ss.no
        else:
            return self.K * np.exp(-self.r * self.T) * ss.norm.cdf(-d2()) - self.S0 * ss.norm.c

    def Delta(self):
        return np.exp(-self.r * self.T) * ss.norm.cdf(self.d1())


S0 = np.linspace(50, 150, 1000)
K = 100
r=0.05
sigma = 0.25
```

```
    T = 1
    type1='C'
    call = Solution(type1, S0, K, r, sigma, T)
    call.BlackScholes()
    call.Delta()

    fig, ax1 = plt.subplots()
    #ax.plot(x, fun[0],'r-', lw=1, alpha=0.6, label='norm pdf')
    ax2 = ax1.twinx()
    ax1.plot(S0, call.BlackScholes(),'b-', lw=3, label='BSCall')
    ax2.plot(S0, call.Delta(),'r-', lw=3, label='Delta')
    legend1 = ax1.legend(loc='upper right', shadow=True)
    ax1.set_title('BS Call(blue) and delta(red)')
    plt.show()
```

In [41]:
```
########## HW5 Problem 5 ########### PART 2: Difference derivative deltas
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt
import pylab
import numpy as np
import scipy.stats as ss
import time

class Solution():
    def __init__(self, type1, S0, K, r, sigma, T):
        self.type1= type1
        self.S0=S0
        self.K=K
        self.r=r
        self.sigma=sigma
        self.T=T

    def d1(self):
        return (np.log(self.S0/self.K) + (self.r + self.sigma**2 / 2) * T)/(self.sigma * np.sq

    def d2(self):

        return (np.log(self.S0 / self.K) + (self.r - self.sigma**2 / 2) * self.T) / (self.sigm

    def BlackScholes(self):
        if self.type1=="C":
            return self.S0 * ss.norm.cdf(self.d1()) - self.K * np.exp(-self.r * self.T) * ss.n
        else:
            return self.K * np.exp(-self.r * self.T) * ss.norm.cdf(-d2()) - self.S0 * ss.norm.

    def Delta(self):
        return np.exp(-self.r * self.T) * ss.norm.cdf(self.d1())


def Diff_BS(h):
    h =1
    S1 = np.linspace(50, 150, 1000)+h
    S2 = np.linspace(50, 150, 1000)-h
```

```
    K = 100
    r=0.05
    sigma = 0.25
    T = 1
    type1='C'

    call1 = Solution(type1, S1, K, r, sigma, T)
    call2 = Solution(type1, S2, K, r, sigma, T)

    diff_BS = (call1.BlackScholes()-call2.BlackScholes())/(2*h)
    return diff_BS


S0 = np.linspace(50, 150, 1000)
K = 100
r=0.05
sigma = 0.25
T = 1
type1='C'
call = Solution(type1, S0, K, r, sigma, T)
call.BlackScholes()
call.Delta()

h1=5*(1e-04)
diff_BS1 = Diff_BS(h1)

h2=3
diff_BS2 = Diff_BS(h2)

fig, ax1 = plt.subplots()
ax1.plot(S0, diff_BS1,'b-', lw=1, label='dBS(x,h=1e-04)')
ax1.plot(S0, diff_BS2,'r-', lw=1, label='dBS(x,h=1e-05)')
ax1.plot(S0, call.Delta(), 'm-', lw=1, label='delta')

legend1 = ax1.legend(loc='upper right', shadow=True)
ax1.set_title('Difference derivative deltas')
plt.show()
```

In [65]:
```
########## HW5 Problem3 ############ PART 2: weighted VaRs and ES
import numpy as np
import scipy.stats as ss
import pandas as pd
import math
import matplotlib.pyplot as plt
import datetime as dt

path = '~/Documents/Semester3/M5320/homework/HW4/AMD-yahoo.csv'
sto_AMD= pd.read_csv(path, header = 0)
sto_AMD.shape
data1 = sto_AMD.values
AMD_close = list(data1[:,6])


A_log_rtn = []
```

```python
A_log_rtn_sq = []

for i in range(1 , len(data1)-1):
    log_return = math.log( AMD_close[i]/AMD_close[i+1] )
    A_log_rtn.append(log_return)
    A_log_rtn_sq.append(log_return**2)

def vol_and_mu(years):
    S0 = 10000
    T = 5/252
    p = 0.99

    A_vol_years = []
    A_mu_years = []
    A_VaR_years = []
    A_ES_years =[]


    for i in range(len(data1)-252*years):
        vol_years = np.std(A_log_rtn[i:i+252*years]) * np.sqrt(252)
        mu_years = np.mean(A_log_rtn[i:i+252*years])*252 + (vol_years**2)/2
        VaR_years = S0 - S0 * np.exp( vol_years * T**(0.5)* ss.norm.ppf(1-p) + (mu_years - pow
        A_vol_years.append(vol_years)
        A_mu_years.append(mu_years)
        A_VaR_years.append(VaR_years)


        if i<6:
            ES_years = VaR_years
        else:
            sum_loss = []
            sum_loss_value = 0
            for k in range(4):
                sum_loss_value = sum_loss_value + A_VaR_years[i-k]
                sum_loss.append(sum_loss_value/(k+1))
                s = pd.Series(sum_loss)
                ES_years = s.quantile(.975)
        A_ES_years.append(ES_years)

    return(A_VaR_years, A_ES_years, A_mu_years, A_vol_years)


A_VaR_2years = vol_and_mu(2)[0][1:252*20]
A_ES_2years = vol_and_mu(2)[1][1:252*20]
####################################################################################




## exponential weighting paramater lambda
lambda1 = 0.998614

### List the lambda values

def list_lambdas(lambda_k):
```

```
    list_lambda = []
    for i in range(len(data1)):
        lambda_value = (lambda_k**i)
        list_lambda.append(lambda_value)
    return(list_lambda)

list_lambda1 = list_lambdas(lambda1)


def weigthed_VaR_and_ES(list_lambda, years):
    wgt_A_vol_years = []
    wgt_A_mu_years = []

    wgt_log_rtn = []
    wgt_log_rtn_sq = []



    for i in range( len(data1)-1 ):
        log_return = math.log( AMD_close[i]/AMD_close[i+1] )
        wgt_log_return = log_return *  list_lambda[i]
        wgt_log_rtn.append(wgt_log_return)

        log_return_sq = log_return ** 2
        wgt_log_rtn_sq.append( log_return_sq * list_lambda[i] )


    S0 = 10000
    T = 5/252
    p = .99

    wgt_A_VaR_years = []
    wgt_A_ES_years =[]

    for j in range(len(data1)-252*years):
        wgt_mu_lambda = 100* sum(wgt_log_rtn[j:j+252*years])/sum(list_lambda[j:j+ 252*years])
        wgt_vol_lambda =((sum(wgt_log_rtn_sq[j:j+252*years])/sum(list_lambda[j:j+ 252*years]))
        wgt_VaR_years = S0 - S0 * np.exp(wgt_vol_lambda * T**(0.5)* ss.norm.ppf(1-p) + (wgt_mu
        wgt_A_vol_years.append(wgt_vol_lambda)
        wgt_A_mu_years.append(wgt_mu_lambda)
        wgt_A_VaR_years.append(wgt_VaR_years)

        if j<6:
            wgt_ES_years = wgt_VaR_years
        else:
            wgt_sum_loss = []
            wgt_sum_loss_value = 0
            for k in range(4):
                wgt_sum_loss_value = wgt_sum_loss_value + wgt_A_VaR_years[j-k]
                wgt_sum_loss.append(wgt_sum_loss_value/(k+1))
                s = pd.Series(wgt_sum_loss)
                wgt_ES_years = s.quantile(.975)
        wgt_A_ES_years.append(wgt_ES_years)
```

```
    return(wgt_A_VaR_years, wgt_A_ES_years, wgt_A_mu_years, wgt_A_vol_years)



years_2 = 2
wgt_A_VaR_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[0][1: 252*20]
#wgt_A_VaR_2years  = [x*50 for x in wgt_A_VaR_2years ]

wgt_A_vol_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[3][1: 252*20]
A_vol_2years =  vol_and_mu(years_2)[3][1: 252*20]

wgt_A_mu_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[2][1: 252*20]
A_mu_years = vol_and_mu(years_2)[2][1: 252*20]

#print(wgt_A_vol_2years[1:10],A_vol_2years[1:10],wgt_A_mu_2years[1:10], A_mu_years[1:10])



timeline = data1[1:252*20,0]
timeline = [dt.datetime.strptime(d,'%Y-%m-%d').date() for d in timeline]


fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax2.plot(timeline, wgt_A_VaR_2years, 'r-', label='Wgted VaR 2 year')
ax1.plot(timeline, A_VaR_2years, 'b-', label='VaR 2 year')
plt.show()
```

In [ ]: