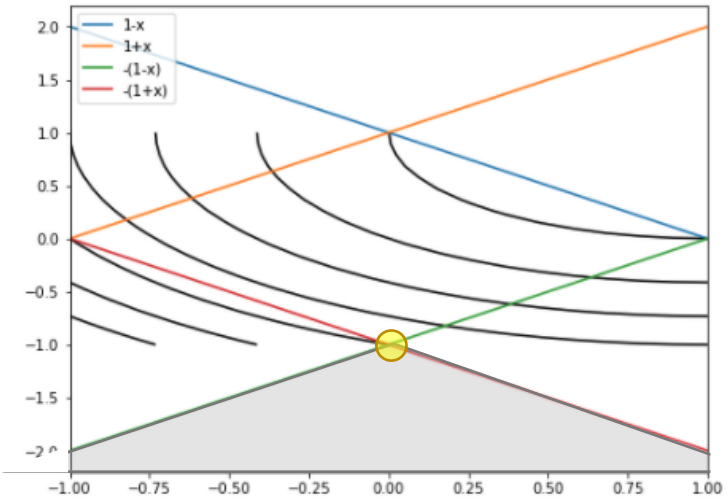
ISYE 6669: Homework 1

Question 1

The feasible region of this problem is shaded in grey in the figure below.

```
In [12]: from matplotlib.pyplot import figure

         fig, ax = plt.subplots()
         fig.set_size_inches(8, 6, forward=True)

         x = np.linspace(-1, 1, 20)
         y = np.linspace(-1, 1, 20)
         z = (x[:,np.newaxis] - 1)**2 + (y[np.newaxis,:]-1)**2

         xx, yy = np.meshgrid(x, y)

         ax.contour(xx, yy, z,  colors='k')
         ax.plot(x, 1-x, label = '1-x')
         ax.plot(x, 1+x, label = '1+x')
         ax.plot(x, x-1, label = '-(1-x)')
         ax.plot(x, -x-1, label = '-(1+x)')

         plt.legend(loc="upper left")
         plt.show()
```



Based on this drawing, the optimal solutions that maximize the objective function occur where $-(1-x)$ and $-(1+x)$ intersect (highlighted point in yellow). At this point, $x = 0$. Solving for y using the constraint function, we get $y = -1$ and $y = 1$ but the only feasible solution is $y = -1$. Given this, the optimal solution is $x = 0$ and $y = -1$ and the optimal objective value is 3.

Question 2

Take the derivative with respect to x:

$$\frac{d}{dx}x(1-2x)^2 = (1-2x)^2 - 4x(1-2x) = (1-2x)[(1-2x)-4x] = (1-2x)(1-6x)$$

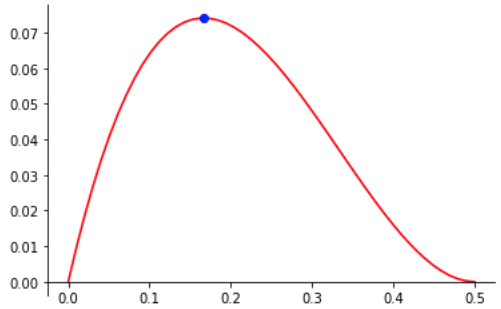
The two solutions are:

$$\left(\frac{1}{2},0\right),\left(\frac{1}{6},\frac{2}{27}\right)$$

Both solutions are feasible given the constraint of $0 \leq x \leq \frac{1}{2}$, but the solution that maximizes the function is $\left(\frac{1}{6},\frac{2}{27}\right)$.

Question 2

```
In [87]: x = np.linspace(0,0.5,50)
         y = x*(1-2*x)**2

         fig = plt.figure()
         ax = fig.add_subplot(1, 1, 1)
         ax.spines['bottom'].set_position('zero')
         ax.spines['right'].set_color('none')
         ax.spines['top'].set_color('none')
         ax.xaxis.set_ticks_position('bottom')
         ax.yaxis.set_ticks_position('left')

         plt.plot(x,y, 'r')
         plt.plot(1/6,2/27, 'bo')

Out[87]: [<matplotlib.lines.Line2D at 0x7fc055d07fa0>]
```



Question 3

a) (P) is a mixed integer quadratic optimization function because the constraint on the values of the variables can either be discrete (for x) or continuous (for z). It is quadratic because the objective function involves quadradic functions.

b) $(P)\max\{x(y^2-z^2): |y|+z \leq 1, x \in \{0,1\}, z \geq 0\}$
$\equiv -\min\{-x(y^2-z^2): |y|+z \leq 1, x \in \{0,1\}, z \geq 0\}$

c) First, we must consider what happens the function when $x = 0$ and when $x = 1$.

- $x = 0 \rightarrow x(y^2-z^2) = 0$
- $x = 1 \rightarrow x(y^2-z^2) = (y^2-z^2)$

Continuing with $x = 1$, we now must maximize $y^2 - z^2$. To maximize this function, we must maximimze the magnitude of y and minimize the magnitude of z.
Given the restraint of $z \geq 0$ and that we want to minimize z, z must equal 0.
Given the restraint of $|y| + z \leq 1$ and $z = 0$ (from the result above), this restraint becomes $|y| \leq 1$. Thus, the maximum values y can take is -1 or 1.

The resulting optimal solutions is as follows:

1. $x = 1, z = 0, y = 1$
2. $x = 1, z = 0, y = -1$

In both solutions, the value of $f(x, y, z) = x(y^2 - z^2) = 1$.

Question 4

Based on the data in the last 24 months, the maximum expected return is 3.43% driven by the Microsoft stock. This is approximately 1% higher than the maximum expected return in 2.3 of 2.46%, also driven by the Microsoft stock. The optimum portfolio with the new data is as follows:

Stock	% Allocation	Amount Invested	Last Price Per Share	Number of Shares
Microsoft	44.2%	$441.64	$284.37	1.55
VISA	0.0%	$0.00	$246.05	0
Walmart	55.8%	$558.37	$142.03	3.93
	Total	$1,000		

The expected monthly return on a $1000 investment is $22 and the standard deviation is $38. The end of month wealth has a mean of $1022 and a range of $908 and $1136. Compared to the results of the original portfolio optimization, the new mean is slightly higher by $2 and a lower standard deviation by $0.30.

The biggest difference between the original portfolio and this new one is that 0% is allocated to VISA. Comparing the old data to the new data, expected return is very close for all the stocks. Risk for VISA has almost doubled in the new data while the others remain relatively flat – this could be why the optimal solution doesn't allocate any percentage of the portfolio to VISA.

	Expected Return		Risk	
	Old Data	New Data	Old Data	New Data
Microsoft	2.5%	3.4%	5.8%	5.2%
VISA	1.8%	1.7%	4.3%	7.8%
Walmart	0.9%	1.2%	4.4%	4.7%

Question 4

In [32]:
```
# Import libraries
import yfinance as yf
from pandas_datareader import data as pdr
import pandas as pd

# Define variables
start = "2019-07-31"
end = "2021-07-30"
tickers_split = ['MSFT', 'V', 'WMT']
df = pd.DataFrame()


# Fetch 'Adj Close' monthly data from 2019-07-31 to 2021-07-30
for ticker in tickers_split:
    temp_df = pdr.get_data_yahoo(ticker, start=start, end=end, interval='m')
    df[ticker] = temp_df['Adj Close']

df.reset_index(inplace = True)
```

In [33]:
```
# read monthly_prices.csv
# mp = pd.read_csv("monthly_prices.csv",index_col=0)
mr = pd.DataFrame()

# compute monthly returns
for s in df.columns[1:]:
    date = df.index[0]
    pr0 = df[s][date]
    for t in range(1,len(df.index)):
        date = df.index[t]
        pr1 = df[s][date]
        ret = (pr1-pr0)/pr0
        #mr.set_value(date,s,ret)
        mr.at[date,s]=ret
        pr0 = pr1

# get symbol names
symbols = mr.columns

# convert monthly return data frame to a numpy matrix
#return_data = mr.as_matrix().T
return_data = mr.values.T

# compute mean return
r = np.asarray(np.mean(return_data, axis=1))

# covariance
C = np.asmatrix(np.cov(return_data))
```

In [39]:
```
print("Expected Return: " + str(r))
print("Covariance Matrix: ")
print(C)
```

In [39]:
```
print("Expected Return: " + str(r))
print("Covariance Matrix: ")
print(C)
```

```
Expected Return: [0.03428378 0.01694463 0.01220378]
Covariance Matrix:
[[0.00266443 0.0024785  0.00048071]
 [0.0024785  0.00601291 0.000909  ]
 [0.00048071 0.000909   0.00220791]]
```

In [34]:
```
# print out expected return and std deviation
print("----------------------")
for j in range(len(symbols)):
    print('%s: Exp ret = %f, Risk = %f' %(symbols[j],r[j], C[j,j]**0.5))

# set up optimization model
n = len(symbols)
x = Variable(n)
req_return = 0.02
ret = r.T*x
risk = quad_form(x, C)
prob = Problem(Minimize(risk),
               [sum(x) == 1, ret >= req_return, x >= 0])

# solve problem and write solution
try:
    prob.solve()
    print("----------------------")
    print("Optimal portfolio")
    print("----------------------")
    for s in range(len(symbols)):
        #print('x[%s] = %f'%(symbols[s],x.value[s,0]))
        print('x[%s] = %f'%(symbols[s],x.value[s]))
    print("----------------------")
    print('Exp ret = %f' %(ret.value))
    print('risk    = %f' %((risk.value)**0.5))
    print("----------------------")
except:
    print('Error')
```

```
----------------------
MSFT: Exp ret = 0.034284, Risk = 0.051618
V: Exp ret = 0.016945, Risk = 0.077543
WMT: Exp ret = 0.012204, Risk = 0.046988
----------------------
Optimal portfolio
----------------------
x[MSFT] = 0.441635
x[V] = 0.000000
x[WMT] = 0.558365
----------------------
Exp ret = 0.021955
risk    = 0.038015
----------------------
```