Problem 1.5

In [1]:
```
from scipy.optimize import brentq
from scipy.stats import norm
from math import sqrt, exp, log

def implied_volatility(market_price, S, K, T, r, option_type='call'):

    def objective_function(sigma):

        def bs_price(S, K, T, r, sigma, option_type, t=0):
            d1 = (log(S/K) + (r + 0.5 * sigma ** 2) * (T-t)) / (sigma * sqrt(T-t))
            d2 = d1 - sigma * sqrt(T-t)
            if option_type == 'call':
                return S * norm.cdf(d1) - K * exp(-r * (T-t)) * norm.cdf(d2)
            else:
                return K * exp(-r * (T-t)) * norm.cdf(-d2) - S * norm.cdf(-d1)

        return bs_price(S, K, T, r, sigma, option_type) - market_price

    return brentq(objective_function, 1e-12, 1)
```

```
/Users/ommehta/anaconda3/lib/python3.9/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.25.2
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [7]:
```
stock_prices = [322.09, 321.9, 321.57, 321.57, 321.57, 321.47, 321.47, 321.66, 321.66, 321.62]
strike_prices = [365, 270, 290, 300, 350, 310, 320, 330, 340, 330]
option_prices = [0.13, 0.18, 32.8, 1.11, 0.51, 2.62, 8.1, 3.75, 1.43, 5.5]
```

In [13]:
```
r = 0.0519
T = (14/365)
for i in range(0, 10):
    option_type = 'call'
    if i in [1, 3, 5]:
        option_type = 'put'
    print(round(implied_volatility(option_prices[i], stock_prices[i], strike_prices[i], T, r, option_type), 2))
```

```
0.3
0.44
0.36
0.31
0.28
0.29
0.28
0.27
0.27
0.35
```

Problem 2.2

This problem is relatively simple in that it can be modeled as a system of equations, where the variables a, b, c are the respective quantities of contracts of option 1, option 2, and shares of stock.

Simply solving for x in the problem:

$$A = \begin{bmatrix} 0.4 & -0.3 & 1 \\ 0.1 & 0.05 & 0 \\ 2 & 5 & 0 \end{bmatrix} \quad \vec{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad \vec{b} = \begin{bmatrix} 80 \\ -10 \\ -40 \end{bmatrix}$$

In [2]:
```
import numpy as np

# Define the matrix A and vector b
A = np.array([
    [0.4, -0.3, 1],
    [0.1, 0.05, 0],
    [2, 5, 0]
])

b = np.array([80, -10, -40])

# Solve for x
x = np.linalg.solve(A, b)
x
```

Out[2]:
```
array([-120.,   40.,  140.])
```

Based on the output we get the column vector

$$\vec{x} = \begin{bmatrix} -120 \\ 40 \\ 140 \end{bmatrix}$$

This suggests that we should short 120 contracts of option 1, long 40 contracts of option 2, and long 140 shares of stock in order to make our position neutral of all greeks.