Strategy Determination from Payoffs

01 March 2021 | Rohith Krishna

Lectures on Computational Finance by Rakesh Nigam

Using ideas from linear algebra we construct a calculator that computes payoff from any given strategy. We also solve the inverse problem: given a set of payoffs we compute the strategy adopted.

Options, strategies and payoffs

Defintion. A *financial derivative* is a contract between two parties detailing payments given the realization of an underlying variable.

The underlying variable is usually an asset such as a stock and the randomness element in the derivative's payoff stems from this underlying variable. For instance, the toss of a coin could determine the payoff: Win $1 when heads and loss of $1 when tails. We adopt the following sign convention:

- Positive payoff = buyer receives payoff from seller.
- Negative payoff = buyer gives payoff to the seller.

Definition. A *stock* is a financial asset that takes S_0 amount to buy today, and has a price of S_T at time T. The payoff of a long stock is therefore S_T and is offset by its price today S_0. A Zero Coupon Bond (ZCB) is a riskless asset that pays a fixed payoff irrespective of the realization of stock price S_T.

Definition. A forward contract is an agreement to exchange a traded asset at a future time for a price fixed today. No initial payment is made on the forward today - it is simply a contract that bets on the future price of an asset. In the long position, at time T one exchanges the traded asset (S_T) against the fixed price $F_t(T)$. The payoff of a long forward is $S_T - F_t(T)$ and that of a short forward is $F_t(T) - S_T$.

Definition. An option gives the buyer of the contract a right to buy (call) or sell (put) an underlying asset at a fixed price (strike) on/before a given date (maturity). The fixed price called strike or exercise price (K) is decided on the date of writing the contract. Exercising an option means to buy/sell the underlying asset using the option contract rather than from the market. We denote maturity period by T, underlying asset (stock) price at $t = 0$ by S_0 and that at maturity by S_T.

There exists an asymmetry between the buyer and seller of any option. The buyer of any option would pay a fee called the premium, which is the price of buying the contract. This is the price paid to have an option to exercise a right. If the right is to buy an underlying asset then the contract is called a call. If the right is to sell an underlying asset, it is called a put. The seller of an option, also referred to as the writer of the contract always has an obligation. As a cost of this obligation, they charge the premium on the buyer of the contract. Long and short positions on an option refers to the buying and selling of the contract respectively, and has nothing to do with the buying or selling of the underlying asset. This is listed below:

- Long Call - the buyer of the call has the right to buy the underlying asset at K on T.
- Short Call - the seller of the call has the obligation to sell the underlying asset at K on T.
- Long Put - the buyer of the put has the right to sell the underlying asset at K on T.
- Short Put - the seller of the put has the obligation to buy the underlying asset at K on T.

Position	Payoff	Profit
Long Call	$C_L(T) = [S_T - K]^+$	$C_L(T) - \text{call price}$
Short Call	$C_S(T) = -[S_T - K]^+$	$C_S(T) + \text{call price}$
Long Put	$P_L(T) = [K - S_T]^+$	$P_L(T) - \text{put price}$
Short Put	$P_S(T) = -[K - S_T]^+$	$P_S(T) + \text{put price}$

Table. The payoffs and profits from different positions on calls and puts.

Payoff Calculator

We construct an algorithm to calculate the total payoff arising from different strategies. A strategy refers to the positions taken on various assets and derivatives over a certain period of time. A strategy begins with the purchase or sale of certain assets or derivatives. At the end period, the strategy is closed by taking a counter position.

An agent takes risk by placing bets on the future outcome of the asset. The price of the underlying asset (stock) S_T is the random variable. Depending on the positions taken in the strategy, the agent gets a cumulative payoff once S_T has been realized. For a given strategy, every realization of S_T offers a different payoff. The plot of S_T realizations versus the payoff obtained for a given strategy is called the *payoff diagram*. We assume that no risk-less profits, called *arbitrage*, can be made.

Algorithm

1. **Input position and quantity.** The position on various assets or derivatives is taken as input. The quantity of each item bought is a positive real number (Long). For a short position, the quantity sold is input as a negative real number. For instance, long 3 call options is input as +3, while short 2 stocks is input as -2.
2. **Input strike prices.** For calls and puts, the strike price K is to be specified to compute the payoff. For a stock, the initial price at $t = 0$, S_0 is input. Likewise the fixed price $F_t(T)$ for forward and payoff from risk-less ZCB are also input.
3. **Set random vector S_T.** The various possible realizations of random variable S_T for a given time T is input as a vector.
4. **Compute individual payoff vector.** Using the payoff scheme for each instrument compute the individual payoff vector. For instance, a long call has a payoff vector given by $[S_T - K]^+$. For each outcome of S_T, there exists a corresponding element in $[S_T - K]^+$ vector.
5. **Linear combination.** In order to obtain total payoff vector, multiply each instruments' individual payoff with the corresponding units purchased or sold. Incorporate the positions taken using the sign convention: long is in positive number of units, and short is in negative number of units. A linear combination of these individual position payoff vectors gives the total payoff of the strategy.

Example. Consider a reverse covered call strategy, where one goes long on 3 calls and short on 2 stock. Let current stock price be $S_0 = 4$ and possible values of S_T be 3, 4, 5, 6 or 7 at $T = 0$. Expressed as a vector, $\mathcal{S}_T = [3, 4, 5, 6, 7]^T$. The strike price of call is $K = 5$.

The payoff from a long call is $[S_T - K]^+$. The short stock pays $S_T - S_0$ at T, where S_0 is used to offset the stock payoff. The individual instruments' payoffs are:

$$\text{Long Call} : \mathcal{P}_c = [0, 0, 0, 1, 2]^T \tag{1}$$
$$\text{Long stock} : \mathcal{P}_s = [-1, 0, +1, +2, +3]^T \tag{2}$$

A linear combination of the individual payoffs with corresponding positions and units gives the total payoff $\mathcal{P}$. Thus we have,

$$\mathcal{P} = (+3\mathcal{P}_c) + (-2\mathcal{P}_s) \tag{3}$$
$$\mathcal{P} = \left[2, 0, -2, -1, 0\right]^T \tag{4}$$

Strategy Calculator

The inverse problem is to identify the strategy adopted on the financial instruments (calls, puts, stocks, forwards, ZCBs) given the total payoffs. The stock price S_T is the random variables and its possible outcomes are given to us. We are also provided data on S_0, $F_t(T)$, K and $B_T(T)$.

Algorithm

1. **Input strike price.** The various derivatives and assets available for investment and the corresponding strike prices for options, fixed price for forwards, bond price at maturity and initial stock price are input.
2. **Input total payoffs and S_T as vectors.** The possible outcomes of $\{S_T\}$ is input as a vector $\mathcal{S}_T$. The total payoffs resulting from the unknown strategy is also input and is denoted as vector $\boldsymbol{b}$.
3. **Form a matrix with individual payoffs as columns.** For each instrument i, form a matrix A with i columns such that each column A_i contains the payoff vector corresponding to instrument i. Typically, $\mathcal{S}_T$ has several possible outcomes and the number of financial instruments in existence is fairly limited; this implies that the number of rows in A exceed the number of columns. Thus A is a tall and thin matrix.
4. **Solve $A\boldsymbol{x} = \boldsymbol{b}$.** The unknown strategy is specified by the vector $\boldsymbol{x}$, with number of components equal to the number of instruments whose details are provided. $A\boldsymbol{x}$ denotes the linear combination of columns of A, which should yield the given payoff $\boldsymbol{b}$. Thus to obtain the positions on the individual instruments we solve for $\boldsymbol{x}$ in $A\boldsymbol{x} = \boldsymbol{b}$.

Sign convention dictates that positive values for x_i indicates a long position, that is buy x quantity of instrument i and a negative x_i indicates a short postition, which is to sell x quantity of instrument i.

Note. For a tall and thin matrix A, the solving $A\boldsymbol{x} = \boldsymbol{b}$ could lead to (1) *No solution.* Or (2) *Unique Solution.* Recall that for such a matrix A:

- All columns of A are pivot columns. A has a full column rank, $(r = n, r < m)$ case.
- If $\boldsymbol{b} \notin \mathcal{C}_A$ and $\mathcal{N}_A = \{\}$, then $A\boldsymbol{x} = \boldsymbol{b}$ has *no solution*. Else, if $\boldsymbol{b} \in \mathcal{C}_A$ and $\mathcal{N}_A = \{\boldsymbol{0}\}$, then $A\boldsymbol{x} = \boldsymbol{b}$ has a *unique solution*.

Example. Consider the previous example. Let current stock price be $S_0 = 4$ and possible values of S_T be 3, 4, 5, 6 or 7 at $T = 0$. Expressed as a vector, $\mathcal{S}_T = [3, 4, 5, 6, 7]^T$. The strike price of call is $K = 5$. Let the total payoff vector $\boldsymbol{b} = [2, 0, -2, -1, 0]^T$. Let $\boldsymbol{x} = [x_c, x_s]^T$ represent the positions taken in calls and stocks respectively.

The payoff from a long call is $[S_T - K]^+$. The short stock pays $S_T - S_0$ at T, where S_0 is used to offset the stock payoff. The payoffs vectors calculated for calls and stocks are as before.

$$\text{Long Call} : A_c = [0, 0, 0, 1, 2]^T \tag{5}$$
$$\text{Long stock} : A_s = [-1, 0, +1, +2, +3]^T \tag{6}$$

Thus we have the form $A\boldsymbol{x} = \boldsymbol{b}$,

$$x_c \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 2 \end{bmatrix} + x_s \begin{bmatrix} -1 \\ 0 \\ 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ -2 \\ -1 \\ 0 \end{bmatrix} \implies \underbrace{\begin{bmatrix} 0 & -1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x_c \\ x_s \end{bmatrix}}_{A\boldsymbol{x}} = \underbrace{\begin{bmatrix} 2 \\ 0 \\ -2 \\ -1 \\ 0 \end{bmatrix}}_{\boldsymbol{b}} \tag{7}$$

This can be solved using Gaussian Elimination. If indeed the payoffs can arise from positions on given instruments, $\boldsymbol{x}$ can be uniquely determined.

$$
\begin{aligned}
[A|\boldsymbol{b}] &= \left[\begin{array}{ccc|c} 0 & -1 & 2 & \\ 0 & 0 & 0 & \\ 0 & 1 & -2 & \\ 1 & 2 & -1 & \\ 2 & 3 & 0 & \end{array}\right] \longrightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & \\ 0 & 1 & -2 & \\ 2 & 3 & 0 & \\ 0 & 0 & 0 & \\ 0 & 0 & 0 & \end{array}\right] \longrightarrow \\
&\longrightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & \\ 0 & 1 & -2 & \\ 0 & -1 & 2 & \\ 0 & 0 & 0 & \\ 0 & 0 & 0 & \end{array}\right] \longrightarrow \left[\begin{array}{cc|c} 1 & 0 & 3 \\ 0 & 1 & -2 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right] \longrightarrow \left[\begin{array}{c|c} I & \boldsymbol{x} \\ \hline O & O \end{array}\right] \\
\text{Thus, } \boldsymbol{x} &= \begin{bmatrix} x_c \\ x_s \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \end{bmatrix} = \begin{bmatrix} \text{Long 3 calls} \\ \text{Short 2 puts} \end{bmatrix}
\end{aligned} \tag{8}
$$

We have $\boldsymbol{x} = [3, -2]^T$, that is, the strategy obtained from the given payoffs is to take the positions: long 3 calls and short 2 stocks.

Programs on Python

The code for both the Payoff calculator and the Strategy Calculator in python is given in this section.

Payoff Calculator

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(rc={'figure.figsize':(12,8)}, style = 'whitegrid')


def payoffCalculator(c=1,p=0,s=1,f=0,z=0):
    # c = number of unique calls
    # p = number of unique puts
    # s = number of unique stocks
    # f = number of unique forwards
    # z = number of unique zero coupon bonds, cash or any risk-free asset

    # columns of the payoff matrix
```

```
    cols = []
    for i in range(c):
        cols.append(str('c')+str(i+1))
    for i in range(p):
        cols.append(str('p')+str(i+1))
    for i in range(s):
        cols.append(str('s')+str(i+1))
    for i in range(f):
        cols.append(str('f')+str(i+1))
    for i in range(z):
        cols.append(str('z')+str(i+1))

    # units contains the units and position in individual stocks
    units = {}
    for i in range(c):
        string = 'c'+str(i+1)
        units['c'+str(i+1)] = int(input("Enter Units of Call %d (%s) = " %
(i+1,string)))
    for i in range(p):
        string = 'p'+str(i+1)
        units['p'+str(i+1)] = int(input("Enter Units of Put %d (%s) = " %
(i+1,string)))
    for i in range(s):
        string = 's'+str(i+1)
        units['s'+str(i+1)] = int(input("Enter Units of Stock %d (%s) = " %
(i+1,string)))
    for i in range(f):
        string = 'f'+str(i+1)
        units['f'+str(i+1)] = int(input("Enter Units of Forward %d (%s) =
" % (i+1,string)))
    for i in range(z):
        string = 'z'+str(i+1)
        units['z'+str(i+1)] = int(input("Enter Units of ZCB %d (%s) = " %
(i+1,string)))

    # strike contains the strike prices and stock prices
    strike = {}
    for i in range(c):
        string = 'K(c'+str(i+1)+')'
        strike['c'+str(i+1)] = int(input("Enter Strike Price of Call %d =
%s = " % (i+1,string)))
    for i in range(p):
```

```
        string = 'K(p'+str(i+1)+')'
        strike['p'+str(i+1)] = int(input("Enter Strike Price of Put %d = %s
= " % (i+1,string)))
    for i in range(s):
        string = 'K(s'+str(i+1)+')'
        strike['s'+str(i+1)] = int(input("Enter the Initial Stock Price =
%s = " % (string)))
    for i in range(f):
        string = 'K(f'+str(i+1)+')'
        strike['f'+str(i+1)] = int(input("Enter Fixed Price of Forward %d =
%s = " % (i+1,string)))
    for i in range(z):
        string = 'K(z'+str(i+1)+')'
        strike['z'+str(i+1)] = int(input("Enter Payoff of ZCB %d (at T) =
%s = " % (i+1,string)))

# create payoff dataframe
    df = pd.DataFrame()
    frac = strike['s1']/1000
    df['ST'] =  np.arange(0,2*strike['s1'] +0.0001,frac)
    #ST = np.linspace(1,10,10)
    #df = pd.DataFrame()
    #df['ST'] = ST
    for i in cols:
        df[i] = 0.0
    for j in strike:
        if (j[:1] == 'c'):  # call payoffs
            for i in range(len(df)):
                if (units[j]>0):
                    df[j][i] = max((units[j]*(df['ST'][i]-strike[j])),0)
                else:
                    df[j][i] = min((units[j]*(df['ST'][i]-strike[j])),0)
        elif (j[:1] == 'p'): # put payoffs
            for i in range(len(df)):
                if (units[j]>0):
                    df[j][i] = max((units[j]*(strike[j]-df['ST'][i])),0)
                else:
                    df[j][i] = min((units[j]*(strike[j]-df['ST'][i])),0)
        elif (j[:1] == 's'): # stock payoffs
            for i in range(len(df)):
                if (units[j]!=0):
                    df[j][i] = units[j]*(df['ST'][i]-strike[j])
```

```
        elif (j[:1] == 'f'): # forward contract payoffs
            for i in range(len(df)):
                if (units[j]>0):
                    df[j][i] = units[j]*(df['ST'][i]-strike[j])
                else:
                    df[j][i] = units[j]*(strike[j] - df['ST'][i])
        elif (j[:1] == 'z'): # riskless ZCB payoffs
            for i in range(len(df)):
                if (units[j]!=0):
                    df[j][i] = strike[j]

    df['TP'] = 0.0
    for j in strike:
        df['TP'] = df['TP'] + df[j]

    # plotting the payoffs
    sns.lineplot(x='ST', y='TP', data=df )
    return(df)
```

Example

```
TP = payoffCalculator(c=1,p=0,s=1,f=0,z=0)['TP']
print('TP')
```

```
Enter Units of Call 1 (c1) = 3
Enter Units of Stock 1 (s1) = -2
Enter Strike Price of Call 1 = K(c1) = 5
Enter the Initial Stock Price = K(s1) = 4

[Output]
3     2.0
4     0.0
5    -2.0
6    -1.0
7     0.0
Name: TP, dtype: float64
```

Strategy Calculator

```
def getStrategy(b,c =1,p=0,s=1,f=0,z=0):
    # c = number of unique calls
    # p = number of unique puts
    # s = number of unique stocks
    # f = number of unique forwards
    # z = number of unique zero coupon bonds, cash or any risk-free asset

    # columns of the payoff matrix
    cols = []
    for i in range(c):
        cols.append(str('c')+str(i+1))
    for i in range(p):
        cols.append(str('p')+str(i+1))
    for i in range(s):
        cols.append(str('s')+str(i+1))
    for i in range(f):
        cols.append(str('f')+str(i+1))
    for i in range(z):
        cols.append(str('z')+str(i+1))

    # strike contains the strike prices and stock prices
    strike = {}
    for i in range(c):
        string = 'K(c'+str(i+1)+')'
        strike['c'+str(i+1)] = int(input("Enter Strike Price of Call %d =
%s = " % (i+1,string)))
    for i in range(p):
        string = 'K(p'+str(i+1)+')'
        strike['p'+str(i+1)] = int(input("Enter Strike Price of Put %d = %s
= " % (i+1,string)))
    for i in range(s):
        string = 'K(s'+str(i+1)+')'
        strike['s'+str(i+1)] = int(input("Enter the Initial Stock Price =
%s = " % (string)))
    for i in range(f):
        string = 'K(f'+str(i+1)+')'
        strike['f'+str(i+1)] = int(input("Enter Fixed Price of Forward %d =
%s = " % (i+1,string)))
    for i in range(z):
        string = 'K(z'+str(i+1)+')'
```

```
        strike['z'+str(i+1)] = int(input("Enter Payoff of ZCB %d (at T) =
%s = " % (i+1,string)))

    # create payoff dataframe
    df = pd.DataFrame()
    frac = strike['s1']/1000
    df['ST'] =  np.arange(0,2*strike['s1'] +0.0001,frac)
    #ST = np.linspace(1,10,10)
    #df = pd.DataFrame()
    #df['ST'] = ST
    for i in cols:
        df[i] = 0.0
    for j in strike:
        if (j[:1] == 'c'):
            for i in range(len(df)):
                df[j][i] = max(((df['ST'][i]-strike[j])),0)
        elif (j[:1] == 'p'):
            for i in range(len(df)):
                df[j][i] = max(((strike[j]-df['ST'][i])),0)
        elif (j[:1] == 's'):
            for i in range(len(df)):
                df[j][i] = (df['ST'][i]-strike[j])
        elif (j[:1] == 'f'):
            for i in range(len(df)):
                df[j][i] = (df['ST'][i]-strike[j])
        elif (j[:1] == 'z'):
            for i in range(len(df)):
                df[j][i] = strike[j]

    # Solve Ax = b to get strategy
    # b = TP
    A = df.copy()
    A.drop(['ST'], axis = 1, inplace = True)
    A = A.to_numpy()
    b = b.to_numpy()
    x = np.linalg.lstsq(A,b, rcond = -1)[0]
    print("\n====== STRATEGY ======")
    for i,j in zip(strike,x):
        if j>0:
            print("Long",round(j),"units of",i)
        elif j<0:
            print("Short",round(-j),"units of",i)
```

```
    return(x)
```

Example.

```
b = np.array([2,0,-2,-1,0])
x = getStrategy(b,c=1,p=0,s=1,f=0,z=0)
print(x)p
```

```
Enter Strike Price of Call 1 = K(c1) = 5
Enter the Initial Stock Price = K(s1) = 4

[Output]
====== STRATEGY ======
Long 3.0 units of c1
Short 2.0 units of s1
array([ 3., -2.])
```

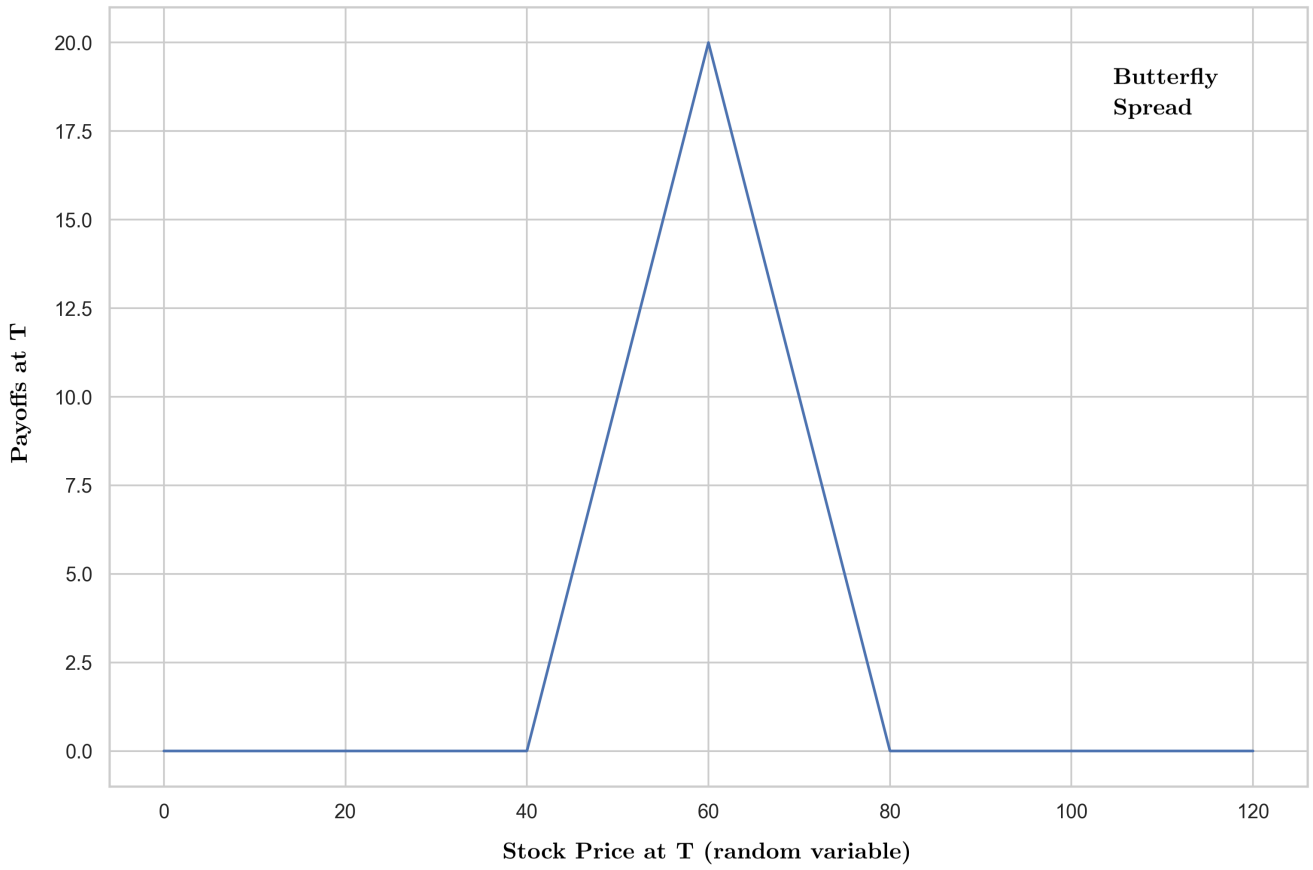
Example: Butterfly Spread.

A butterfly spread is a strategy where one take long on 2 calls with strike prices K_1 and K_2 and simultaneous goes short on two calls of a third kind with $K_3 = 0.5(K_1 + K_2)$. For the given parameters we have,

Payoff calculator

```
TP = payoffCalculator(c=3,p=0,s=1,f=0,z=0)['TP']
print(TP)
```

```
Enter Units of Call 1 (c1) = 1
Enter Units of Call 2 (c2) = -2
Enter Units of Call 3 (c3) = 1
Enter Units of Stock 1 (s1) = 0
Enter Strike Price of Call 1 = K(c1) = 40
Enter Strike Price of Call 2 = K(c2) = 60
Enter Strike Price of Call 3 = K(c3) = 80
Enter the Initial Stock Price = K(s1) = 60
```



Strategy Calculator

```
x = getStrategy(TP,c=3,p=0,s=1,f=0,z=0)
print(x)
```

```
Enter Strike Price of Call 1 = K(c1) = 40
Enter Strike Price of Call 2 = K(c2) = 60
Enter Strike Price of Call 3 = K(c3) = 80
Enter the Initial Stock Price = K(s1) = 60

====== STRATEGY ======
Long 1.0 units of c1
Short 2.0 units of c2
Long 1.0 units of c3
Short 0.0 units of s1
```

As expected, the strategy calculator solves $A\boldsymbol{x} = \boldsymbol{b}$ to obtain the exact positions taken in corresponding instruments.