MFE 405; Project 2

Aliaksei Kanstantsinau

In [128…

```
import numpy as np
import matplotlib.pyplot as plt
```

Problem 1.

Compare the convergence rates of the four methods below by doing the following: Use the Binomial Method to price a 6-month American Put option with the following information: the risk-free interest rate is 5.5% per annum; the volatility is 25% per annum; the current stock price is $180; and the strike price is $170. Divide the time interval into n parts to estimate the price of this option. Use n = 20, 40, 80, 100, 200, 500, to estimate the price and draw all resulting prices in one graph, where the horizontal axis measures n, and the vertical one the price of the option.

(a) Use the binomial method in which: $$ u=\frac{1}{d} ; d = c - \sqrt{c^2 - 1}; c =\frac{1}{2}(e^{-rdt} + e^{(r+\sigma^2)dt}); p=\frac{e^{rdt-d}}{u-d}$$

In [129…

```
# define problem parameters
n_array = [20, 40, 80, 100, 200, 500]
T = .5
r = .055
sigma = .25
S0 = 180
K = 170
```

In [131…

```
# define problem (a) function
def binomial_method_a(S0, K, T, r, sigma, n_array):
    # initiate array to store values
    values = []
    # iterate through array
    for interval in n_array:
        # set dt, c, d, u, and p
        dt = T/interval
        c = 1/2*(np.exp(-r*dt) + np.exp((r+sigma**2)*dt))
        d = c - np.sqrt(c**2 - 1)
        u = 1/d
        p = (np.exp(r*dt) - d)/(u-d)
        # initiate matrices for stock and option values
        S_values = np.zeros((interval+1, interval+1))
        P_values = np.zeros((interval+1, interval+1))
        # set initial stock value to zero
        S_values[0, 0] = S0
        # generate stock prices, we go step by step in interval range
        for step in range(1, interval+1):
            S_values[step, 0] = S_values[step-1, 0] * u
            # make sure there are 2 states for each step
```

```
            for state in range(1, step+1):
                S_values[step, state] = S_values[step-1, state-1] * d
        # set option value to max between strike and stock value
        P_values[step, state] = max(K-S_values[step, state], 0)
        # work backwards, we need to see whether EV or HV is bigger
        for step in range(interval-1, -1, -1):
            for state in range(step+1):
                exercise_value = K - S_values[step, state]
                # calculate hold value
                hold_value = (p*P_values[step+1, state] + (1-p)*P_values[ste
                P_values[step, state] = max(exercise_value, hold_value)

        values.append(P_values[0, 0])
    return values
```

(b) Use the binomial method in which: $$u=e^{(r-.5\sigma^2)dt + \sigma \sqrt(dt)}; d=e^{(r-.5\sigma^2)dt - \sigma \sqrt(dt)}; p=\frac{1}{2}$$

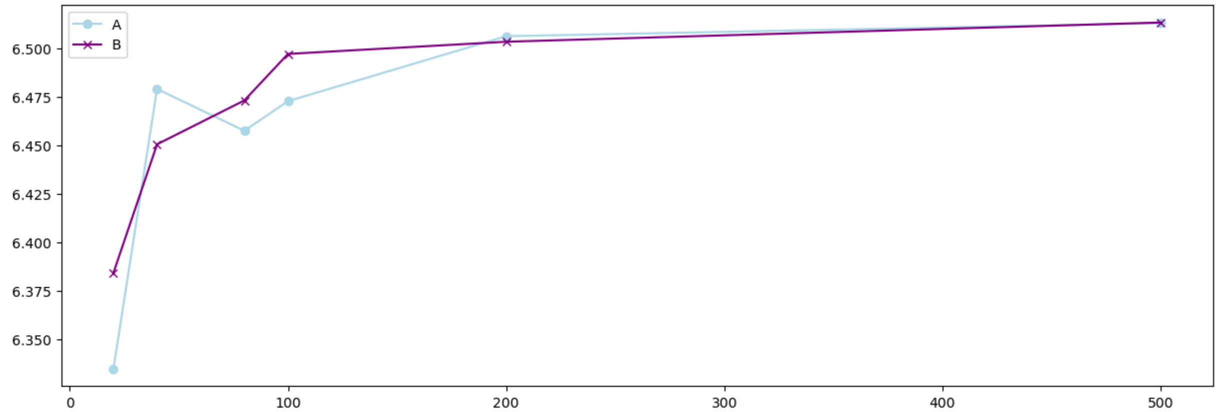
In [132…

```
# define problem (b) function. same as a the only difference is u, d, and p
def binomial_method_b(S0, K, T, r, sigma, n_array):
    # initiate array to store values
    values = []
    # iterate through array
    for interval in n_array:
        # set dt, c, d, u, and p
        dt = T/interval
        d = np.exp((r - .5*sigma**2)*dt - sigma*np.sqrt(dt))
        u = np.exp((r - .5*sigma**2)*dt + sigma*np.sqrt(dt))
        p = 1/2
        # initiate matrices for stock and option values
        S_values = np.zeros((interval+1, interval+1))
        P_values = np.zeros((interval+1, interval+1))
        # set initial stock value to zero
        S_values[0, 0] = S0
        # generate stock prices, we go step by step in interval range
        for step in range(1, interval+1):
            S_values[step, 0] = S_values[step-1, 0] * u
            # make sure there are 2 states for each step
            for state in range(1, step+1):
                S_values[step, state] = S_values[step-1, state-1] * d
        # set option value to max between strike and stock value
        P_values[step, state] = max(K-S_values[step, state], 0)
        # work backwards, we need to see whether EV or HV is bigger
        for step in range(interval-1, -1, -1):
            for state in range(step+1):
                exercise_value = K - S_values[step, state]
                # calculate hold value
                hold_value = (p*P_values[step+1, state] + (1-p)*P_values[ste
                P_values[step, state] = max(exercise_value, hold_value)

        values.append(P_values[0, 0])
    return values
```

Outputs: Graphs: Two plots in one graph.

In [133…

```
# get values
value_a = binomial_method_a(S0, K, T, r, sigma, n_array)
value_b = binomial_method_b(S0, K, T, r, sigma, n_array)

# plot the figure
plt.figure(figsize=(15, 5))
plt.plot(n_array, value_a, label='A', color='lightblue', marker='o')
plt.plot(n_array, value_b, label='B', color='purple', marker='x')
plt.legend()
plt.show()
```



Problem 2.

Consider the following information on the stock of a company and American put options on it: $$ S0 = \$180, X = \$170, r = 0.055, \sigma = 0.25, T = 6m, \mu = 0.15 $$ Using the CRR Binomial tree method, estimate the following and draw their graphs:

(i) Delta of the put option as a function of S0, for S0 ranging from $170 to $190, in increments of $2.

In [134…

```
# define crr model
def crr(range_start, range_end, K, T, r, sigma):
    deltas = []
    for S0 in range(range_start, range_end+1, 2):
        # assume 500 time steps
        n = 500
        dt = T/n
        d = np.exp(-sigma * np.sqrt(dt))
        u = np.exp(sigma * np.sqrt(dt))
        p = 1/2*(1 + ((r - .5*sigma**2)*dt)/sigma)
        # initiate matrices for stock and option values
        S_values = np.zeros((n+1, n+1))
        P_values = np.zeros((n+1, n+1))
        # set initial stock value to zero
        S_values[0, 0] = S0
        # generate stock prices, we go step by step in interval range
        for step in range(1, n+1):
            S_values[step, 0] = S_values[step-1, 0] * u
            # make sure there are 2 states for each step
            for state in range(1, step+1):
```

```
                S_values[step, state] = S_values[step-1, state-1] * d
        # set option value to max between strike and stock value
        P_values[step, state] = max(K-S_values[step, state], 0)
        # work backwards, we need to see whether EV or HV is bigger
        for step in range(n-1, -1, -1):
            for state in range(step+1):
                exercise_value = K - S_values[step, state]
                # calculate hold value
                hold_value = (p*P_values[step+1, state] + (1-p)*P_values[ste
                P_values[step, state] = max(exercise_value, hold_value)

        # in theory I only need two nodes but
        delta = (P_values[1, 0] - P_values[1, 1])/(S_values[1, 0] - S_values
        deltas.append(delta)
    return deltas
```

(ii) Delta of the put option, as a function of T (time to expiration), T ranging from 0 to 0.18 in increments of 0.003.

In [135…

```
# copy crr, make edits
def crr_time(S0, K, tstart, tend, tincrement, r, sigma):
    deltas = []
    T = tstart + tincrement
    while T <= tend:
        # assume 500 time steps
        n = 500
        dt = T/n
        d = np.exp(-sigma * np.sqrt(dt))
        u = np.exp(sigma * np.sqrt(dt))
        p = 1/2*(1 + ((r - .5*sigma**2)*dt)/sigma)
        # initiate matrices for stock and option values
        S_values = np.zeros((n+1, n+1))
        P_values = np.zeros((n+1, n+1))
        # set initial stock value to zero
        S_values[0, 0] = S0
        # generate stock prices, we go step by step in interval range
        for step in range(1, n+1):
            S_values[step, 0] = S_values[step-1, 0] * u
            # make sure there are 2 states for each step
            for state in range(1, step+1):
                S_values[step, state] = S_values[step-1, state-1] * d
        # set option value to max between strike and stock value
        P_values[step, state] = max(K-S_values[step, state], 0)
        # work backwards, we need to see whether EV or HV is bigger
        for step in range(n-1, -1, -1):
            for state in range(step+1):
                exercise_value = K - S_values[step, state]
                # calculate hold value
                hold_value = (p*P_values[step+1, state] + (1-p)*P_values[ste
                P_values[step, state] = max(exercise_value, hold_value)

        # in theory I only need two nodes but
        delta = (P_values[1, 0] - P_values[1, 1])/(S_values[1, 0] - S_values
        deltas.append(delta)
        T += tincrement
```

```
    return deltas
```

(iii) Theta of the put option, as a function of T (time to expiration), T ranging from 0 to 0.18 in increments of 0.003.

In [136…

```
# copy crr, make edits
def crr_theta(S0, K, tstart, tend, tincrement, r, sigma):
    thetas = []
    T = tstart + tincrement
    while T <= tend:
        # assume 500 time steps
        n = 500
        dt = T/n
        d = np.exp(-sigma * np.sqrt(dt))
        u = np.exp(sigma * np.sqrt(dt))
        p = 1/2*(1 + ((r - .5*sigma**2)*dt)/sigma)
        # initiate matrices for stock and option values
        S_values = np.zeros((n+1, n+1))
        P_values = np.zeros((n+1, n+1))
        # set initial stock value to zero
        S_values[0, 0] = S0
        # generate stock prices, we go step by step in interval range
        for step in range(1, n+1):
            S_values[step, 0] = S_values[step-1, 0] * u
            # make sure there are 2 states for each step
            for state in range(1, step+1):
                S_values[step, state] = S_values[step-1, state-1] * d
        # set option value to max between strike and stock value
        P_values[step, state] = max(K-S_values[step, state], 0)
        # work backwards, we need to see whether EV or HV is bigger
        for step in range(n-1, -1, -1):
            for state in range(step+1):
                exercise_value = K - S_values[step, state]
                # calculate hold value
                hold_value = (p*P_values[step+1, state] + (1-p)*P_values[ste
                P_values[step, state] = max(exercise_value, hold_value)

        # in theory I only need two nodes but
        theta = (P_values[2, 1] - P_values[0, 0])/2*dt
        thetas.append(theta)
        T += tincrement

    return thetas
```

(iv) Vega of the put option, as a function of S0, for S0 ranging from \$170 to \$190, in increments of \$2.

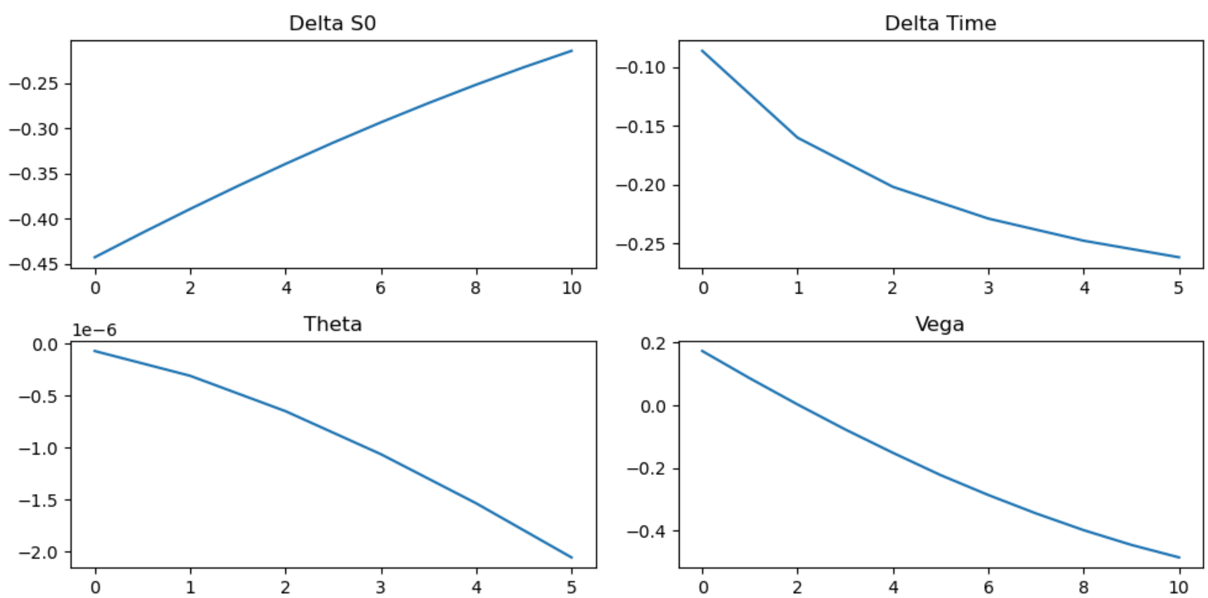
In [137…

```
def crr_vegas(range_start, range_end, K, T, r, sigma):
    x = crr(range_start, range_end, K, T, r, sigma)
    y = crr(range_start, range_end, K, T, r, sigma+.05)
    vegas = []
    for pricex, pricey in zip(x, y):
        vega = (pricey - pricex)/.05
```

```
        vegas.append(vega)
    return vegas
```

Outputs: Graphs: 4 separate graphs.

In [55]:
```
fig, ax = plt.subplots(2, 2, figsize=(10, 5))
ax[0, 0].plot(crr(170, 190, 170, 0.5, 0.055, 0.25))
ax[0, 0].set_title('Delta S0')

ax[0, 1].plot(crr_time(180, 170, 0, 0.18, 0.03, 0.055, 0.25))
ax[0, 1].set_title('Delta Time')

ax[1, 0].plot(crr_theta(180, 170, 0, 0.18, 0.03, 0.055, 0.25))
ax[1, 0].set_title('Theta')

ax[1, 1].plot(crr_vegas(170, 190, 170, 0.5, 0.055, 0.25))
ax[1, 1].set_title('Vega')

plt.tight_layout()
plt.show()
```



Problem 3.

Compare the convergence rates of the two methods, (a) and (b), described below, by doing the following: Use the Trinomial-tree method to price a 6-month American put option with the following information: the risk-free interest rate is 5.5% per annum, the volatility is 25% per annum, the current stock price is $180, and the strike price is $170. Divide the time interval into n equal parts to estimate the option price. Use n = 20, 40, 70, 80, 100, 200, 500; to estimate option prices and draw them all in one graph, where the horizontal axis measures n, and the vertical one measures option price. The two methods are in (a) and (b) below:

(a) Use the Trinomial-tree method applied to the stock price-process (S_t) in which: $$u=\frac{1}{d}; d=e^{-\sigma \sqrt{3dt}};p_u = \frac{rdt(1-u)+(rdt)^2 + \sigma^2dt}{(u-d)(1-d)} p_d = \frac{rdt(1-d)+(rdt)^2 + \sigma^2dt}{(u-d)(u-1)}; p_m = 1 - p_u - p_d$$

In [283…
```
# define problem (a) function
def trinomial_method_a(S0, K, T, r, sigma, n_array):
    # initiate array to store values
    values = []
    # iterate through array
    for n in n_array:
        dt = T / n
        d = np.exp(-sigma * np.sqrt(3 * dt))
        u = 1 / d
        p_u = ((r * dt * (1 - d) + (r * dt)**2 + sigma**2 * dt) / ((u - d) *
        p_d = ((r * dt * (1 - u) + (r * dt)**2 + sigma**2 * dt) / ((u - d) *
        p_m = 1 - p_u - p_d

        # stock, option prices
        S = np.zeros((2 * n + 1, n + 1))
        P = np.zeros((2 * n + 1, n + 1))

        # initial condition
        S[n, 0] = S0

        # generate stock prices
        for j in range(1, n + 1):
            for i in range(n - j + 1, n + j):
                S[i + 1, j] = S[i, j - 1] * u
                S[i, j] = S[i, j - 1]
                S[i - 1, j] = S[i, j - 1] * d

        for i in range(2 * n + 1):
            P[i, n] = max(K - S[i, n], 0)

        # backward induction
        for j in range(n - 1, -1, -1):
            for i in range(n - j, n + j + 1):
                P[i, j] = (p_u * P[i + 1, j + 1] + p_m * P[i, j + 1] + p_d *
                if j <= n:
                    P[i, j] = max(P[i, j], K - S[i, j])

        # store values
        values.append(P[n, 0])

    return values
```

(b) Use the Trinomial-tree method applied to the Log-stock price-process (X_T) in which: $$\Delta X_u = \sigma \sqrt{3dt}; \Delta X_d = -\sigma \sqrt{3dt} \\ p_d = \frac{1}{2}(\frac{\sigma^2dt + (r - 0.5\sigma^2)^2 dt^2}{\Delta X_u^2} - \frac{(r - 0.5\sigma^2)^2 dt}{\Delta X_u}); p_u = \frac{1}{2}(\frac{\sigma^2dt + (r - 0.5\sigma^2)^2 dt^2}{\Delta X_u^2} + \frac{(r - 0.5\sigma^2)^2 dt}{\Delta X_u}) \\ p_m = 1 - p_u - p_d$$

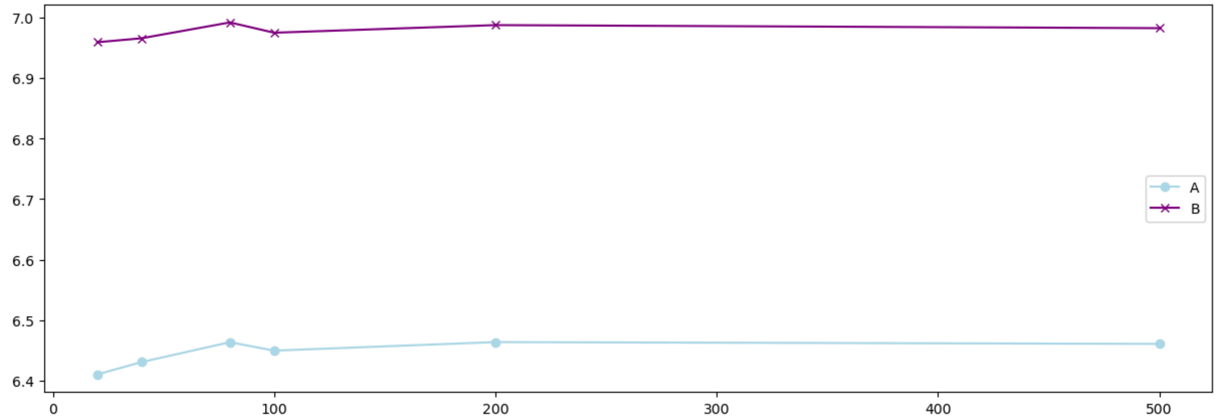
In [288…
```
# define problem (a) function
def trinomial_method_b(S0, K, T, r, sigma, n_array):
```

```
    # initiate array to store values
    values = []
    # iterate through array
    for n in n_array:
        # set dt, c, d, u, and p
        dt = T/n
        X_u = sigma * np.sqrt(3*dt)
        X_d = -sigma * np.sqrt(3*dt)
        p_u = .5 * (((sigma**2*dt + (r - .5*sigma**2)**2 * dt**2)/(X_u**2))+
        p_d = .5 * (((sigma**2*dt + (r - .5*sigma**2)**2 * dt**2)/(X_u**2))-
        p_m = 1 - p_u - p_d
        # stock, option prices
        S = np.zeros((2 * n + 1, n + 1))
        P = np.zeros((2 * n + 1, n + 1))

        # initial condition
        S[n, 0] = S0

        # generate stock prices
        for j in range(1, n + 1):
            for i in range(n - j + 1, n + j):
                S[i + 1, j] = S[i, j - 1] * np.exp(X_u)
                S[i, j] = S[i, j - 1]
                S[i - 1, j] = S[i, j - 1] * np.exp(X_d)

        for i in range(2 * n + 1):
            P[i, n] = max(K - S[i, n], 0)

        # backward induction
        for j in range(n - 1, -1, -1):
            for i in range(n - j, n + j + 1):
                P[i, j] = (p_u * P[i + 1, j + 1] + p_m * P[i, j + 1] + p_d *
                P[i, j] = max(P[i, j], K - S[i, j])

        # store values
        values.append(P[n, 0])

    return values
```

Outputs: Graphs: plot in a graph.

In [289…

```
# get values
value_a_trinomial = trinomial_method_a(S0, K, T, r, sigma, n_array)
value_b_trinomial = trinomial_method_b(S0, K, T, r, sigma, n_array)

# plot the figure
plt.figure(figsize=(15, 5))
plt.plot(n_array, value_a_trinomial, label='A', color='lightblue', marker='c
plt.plot(n_array, value_b_trinomial, label='B', color='purple', marker='x')
plt.legend()
plt.show()
```



Problem 4

Consider the following information on the stock of company XYZ: The current stock price is $180, and the volatility of the stock price is $\sigma = 25\%$ per annum. Assume the prevailing risk-free rate is r = 5.5% per annum. Use the following method to price the specified option:

(a) Use the LSMC method with N=100,000 paths simulations (50,000 plus 50,000 antithetic variates) and a time step of Δ= 1 $\sqrt{N}$ to price an American Put option with strike price of X = $170 and maturity of 0.5-years and 1.5-years. Use the first k of the Laguerre Polynomials for k = 2, 3, 4, 5. (That is, you will compute 8 prices here). Compare the prices for the 4 cases, k = 2, 3, 4, 5 and comment on the choice of k.

Define regression function that will solve for $\beta = (X^TX)^{-1}X^Ty$

In [141…
```
def regression(X, y):
    beta = np.dot((np.dot(np.linalg.inv(np.dot(X.T, X)), X.T)), y)
    return beta
```

In [143…
```
# define function that will generate Laguerre polynomials
def laguerre(x, k):
    # get zeros array
    L = np.zeros((len(x), k))
    # set first one at one
    L[:, 0] = 1
    # handle first k
    if k > 1:
        # set first one to 1-x
        L[:, 1] = 1 - x
    # handle other k's
    for n in range(2, k):
        L[:, n] = ((2 * n - 1 - x) * L[:, n - 1] - (n - 1) * L[:, n - 2]) /
    return L
```

In [144…
```
# to have a clear visual on the result
import pandas as pd
```

In [271…

```
def lsmc(S0, X, T, r, sigma, N, k):
    # set time increment and number of steps
    dt = 1/np.sqrt(N)
    steps = int(T/dt)
    df = np.exp(-r * dt)

    # simulate brownian motion
    W = np.random.normal(0, 1, 50000)
    # normal simulations
    S_normal = S0 * np.exp(np.cumsum(r - .5*sigma**2)*dt + sigma * np.sqrt(d
    # antithetic
    S_antithetic = S0 * np.exp(np.cumsum(r - .5* sigma**2)*dt - sigma * np.s
    # combine
    S = np.vstack((S_normal, S_antithetic))
    # get exercise values (payoffs)
    ev = np.maximum(X-S, 0)
    # store them for each step iterating backwards
    evs = ev[:, -1]

    for time in range(steps - 1, 0, -1):
        # for each time get paths that are in the money
        itm = S[:, time] < X
        # store them separately
        S_itm = S[itm, time]
        # discount values
        dcf = evs[itm] * df

        L = laguerre(S_itm, k)
        A = regression(L, dcf)

        ecv = L.dot(A)
        immediate_exercise = np.maximum(X - S_itm, 0)
        exercise = immediate_exercise > ecv
        evs[itm] = np.where(exercise, immediate_exercise, dcf)
    price = evs.mean() * df
    return price


p4_df = pd.DataFrame(index = ['.5', '1.5'],
                            columns = ['k=2', 'k=3', 'k=4', 'k=5'])
p4_df.at['.5', 'k=2'] = lsmc(180, 170, .5, .055, .25, .25, 2)
p4_df.at['.5', 'k=3'] = lsmc(180, 170, .5, .055, .25, .25, 3)
p4_df.at['.5', 'k=4'] = lsmc(180, 170, .5, .055, .25, .25, 4)
p4_df.at['.5', 'k=5'] = lsmc(180, 170, .5, .055, .25, .25, 5)
p4_df.at['1.5', 'k=2'] = lsmc(180, 170, 1.5, .055, .25, .25, 2)
p4_df.at['1.5', 'k=3'] = lsmc(180, 170, 1.5, .055, .25, .25, 3)
p4_df.at['1.5', 'k=4'] = lsmc(180, 170, 1.5, .055, .25, .25, 4)
p4_df.at['1.5', 'k=5'] = lsmc(180, 170, 1.5, .055, .25, .25, 5)
print(p4_df)


# calculate payoffs (ev) it will be maximum between X-S
# work backwards to figure out how many path cross exercise value
# need to get continuation value for each step which is option value at that
# based on that fact we know whether we exercise or not
```

```
           k=2        k=3        k=4       k=5
.5   16.702439  28.79679  12.131152       0.0
1.5  32.173035  19.597963  25.678233  2.376152
```

(b) Use the LSMC method with N=100,000 paths simulations (50,000 plus 50,000 antithetic variates) and a time step of Δ= 1 $\sqrt{N}$ to price an American Put option with strike price of X = \$170 and maturity of 0.5-years and 1.5-years. Use the first k of the Hermite Polynomials for k = 2, 3, 4, 5. (That is, you will compute 8 prices here). Compare the prices for the 4 cases, k = 2, 3, 4, 5 and comment on the choice of k.

In [146…

```python
# define hermite polynomial function generator
def hermite(x, k):
    # lengtha
    H = np.zeros(len(x), k)
    H[:, 0] = 1
    # handle first k
    for n in range(1, k):
        H[:, n] = (-1)**n * np.exp(.5*x**2) * np.exp(.5*(-x)**2)
    return H
```

In [272…

```python
def lsmc_h(S0, X, T, r, sigma, N, k):
    # set time increment and number of steps
    dt = 1/np.sqrt(N)
    steps = int(T/dt)
    df = np.exp(-r * dt)

    # simulate brownian motion
    W = np.random.normal(0, 1, 50000)
    # normal simulations
    S_normal = S0 * np.exp(np.cumsum(.5 * r * sigma**2)*dt + sigma * np.sqrt
    # antithetic
    S_antithetic = S0 * np.exp(np.cumsum(.5 * r * sigma**2)*dt - sigma * np.
    # combine
    S = np.vstack((S_normal, S_antithetic))
    # get exercise values (payoffs)
    ev = np.maximum(X-S, 0)
    # store them for each step iterating backwards
    evs = ev[:, -1]

    for time in range(steps - 1, 0, -1):
        # for each time get paths that are in the money
        itm = S[:, time] < X
        # store them separately
        S_itm = S[itm, time]
        # discount values
        dcf = evs[itm] * df

        H = hermite(S_itm, k)
        A = regression(H, dcf)

        ecv = H.dot(A)
        immediate_exercise = np.maximum(X - S_itm, 0)
        exercise = immediate_exercise > ecv
        evs[itm] = np.where(exercise, immediate_exercise, dcf)
```

```
    price = evs.mean() * df
    return price

p4_df_2 = pd.DataFrame(index = ['.5', '1.5'],
                       columns = ['k=2', 'k=3', 'k=4', 'k=5'])
p4_df_2.at['.5', 'k=2'] = lsmc_h(180, 170, .5, .055, .25, .25, 2)
p4_df_2.at['.5', 'k=3'] = lsmc_h(180, 170, .5, .055, .25, .25, 3)
p4_df_2.at['.5', 'k=4'] = lsmc_h(180, 170, .5, .055, .25, .25, 4)
p4_df_2.at['.5', 'k=5'] = lsmc_h(180, 170, .5, .055, .25, .25, 5)
p4_df_2.at['1.5', 'k=2'] = lsmc_h(180, 170, 1.5, .055, .25, .25, 2)
p4_df_2.at['1.5', 'k=3'] = lsmc_h(180, 170, 1.5, .055, .25, .25, 3)
p4_df_2.at['1.5', 'k=4'] = lsmc_h(180, 170, 1.5, .055, .25, .25, 4)
p4_df_2.at['1.5', 'k=5'] = lsmc_h(180, 170, 1.5, .055, .25, .25, 5)
print(p4_df_2)
```

```
          k=2        k=3       k=4       k=5
.5   0.877032  13.121164  10.45334  30.20704
1.5  9.711041  26.182219       0.0  10.04102
```

(c) Use the LSMC method with N=100,000 paths simulations (50,000 plus 50,000 antithetic variates) and a time step of Δ= $1\sqrt{N}$ to price an American Put option with strike price of X = \$170 and maturity of 0.5-years and 1.5-years. Use the first k of the Simple Monomials for k = 2, 3, 4, 5. (That is, you will compute 8 prices here). Compare the prices for the 4 cases, k = 2, 3, 4, 5 and comment on the choice of k.

In [148…
```
def monomials(x, k):
    return x**k
```

In [273…
```
def lsmc_m(S0, X, T, r, sigma, N, k):
    # set time increment and number of steps
    dt = 1/np.sqrt(N)
    steps = int(T/dt)
    df = np.exp(-r * dt)

    # simulate brownian motion
    W = np.random.normal(0, 1, 50000)
    # normal simulations
    S_normal = S0 * np.exp(np.cumsum(.5 * r * sigma**2)*dt + sigma * np.sqrt
    # antithetic
    S_antithetic = S0 * np.exp(np.cumsum(.5 * r * sigma**2)*dt - sigma * np.
    # combine
    S = np.vstack((S_normal, S_antithetic))
    # get exercise values (payoffs)
    ev = np.maximum(X-S, 0)
    # store them for each step iterating backwards
    evs = ev[:, -1]

    for time in range(steps - 1, 0, -1):
        # for each time get paths that are in the money
        itm = S[:, time] < X
        # store them separately
        S_itm = S[itm, time]
        # discount values
        dcf = evs[itm] * df
```

```
        M = monomials(S_itm, k)
        A = regression(M, dcf)

        ecv = M.dot(A)
        immediate_exercise = np.maximum(X - S_itm, 0)
        exercise = immediate_exercise > ecv
        evs[itm] = np.where(exercise, immediate_exercise, dcf)
    price = evs.mean() * df
    return price

p4_df_3 = pd.DataFrame(index = ['.5', '1.5'],
                       columns = ['k=2', 'k=3', 'k=4', 'k=5'])
p4_df_3.at['.5', 'k=2'] = lsmc_m(180, 170, .5, .055, .25, .25, 2)
p4_df_3.at['.5', 'k=3'] = lsmc_m(180, 170, .5, .055, .25, .25, 3)
p4_df_3.at['.5', 'k=4'] = lsmc_m(180, 170, .5, .055, .25, .25, 4)
p4_df_3.at['.5', 'k=5'] = lsmc_m(180, 170, .5, .055, .25, .25, 5)
p4_df_3.at['1.5', 'k=2'] = lsmc_m(180, 170, 1.5, .055, .25, .25, 2)
p4_df_3.at['1.5', 'k=3'] = lsmc_m(180, 170, 1.5, .055, .25, .25, 3)
p4_df_3.at['1.5', 'k=4'] = lsmc_m(180, 170, 1.5, .055, .25, .25, 4)
p4_df_3.at['1.5', 'k=5'] = lsmc_m(180, 170, 1.5, .055, .25, .25, 5)
print(p4_df_3)
```

```
          k=2        k=3        k=4       k=5
.5   23.672941  10.287958   4.769637  8.776895
1.5  32.545743   9.981765  21.547401  2.986268
```

(d) Compare all your findings above and comment. Note: You will need to use the weighted-polynomials as it was done by the authors of the method.

- 1 and 3 seem similar, 2 is off probably because I made a mistake in defining Hermite polynomials but I can't find it.

Problem 5.

Consider the following information on the stock of company XYZ: The volatility of the stock price is $\sigma = 25\%$ per annum. Assume the prevailing risk-free rate is $r = 5.5\%$ per annum. Use the $X = ln(S)$ transformation of the Black-Scholes PDE, and $\Delta t = 0.002$, with $\Delta X = \sigma\sqrt{\Delta t}$; then with $\Delta X = \sigma\sqrt{3\Delta t}$; then with $\Delta X = \sigma\sqrt{4\Delta t}$, and a uniform grid (on X) to price an American Put option with strike price of $X = \$170$, expiration of 6 months and current stock prices ranging from \$170 to \$190; using the specified methods below:

(a) Explicit Finite-Difference method

In [211…

```
# since according to the propt we need price for $1 increments:
from scipy.interpolate import interp1d
```

In [235…

```
# get explicit 1
def explicit_1(S0_start, S0_end, K, r, sigma, T, dt):
    dx = sigma * np.sqrt(dt)
    X_start = np.log(S0_start)
    X_end = np.log(S0_end)
    # define prices and increment
    N = int((X_end - X_start) / dx)
```

```
    log_grid = np.linspace(X_start, X_end, N + 1)
    S = np.exp(log_grid)  # Actual stock price grid
    # Time steps
    M = int(T / dt)

    # put option value matrix
    P = np.zeros((N + 1, M + 1))
    P[:, -1] = np.maximum(K - S, 0)

    # set probabilities
    P_u = dt*(((sigma**2)/(2*dx**2)) + ((r - .5 * sigma**2)/(2*dx)))
    P_m = 1 - dt*(sigma**2/dx**2) - r*dt
    P_d = dt*(((sigma**2)/(2*dx**2)) - ((r - .5 * sigma**2)/(2*dx)))

    # implement efd
    for j in range(M - 1, -1, -1):
        for i in range(1, N):
            P[i, j] = P_u * P[i+1, j+1] + P_m * P[i+1, j] + P_d * P[i+1, j-1
        # boundary conditions
        P[0, j] = max(K - S[0], P[0, j+1]*np.exp(-r*dt))
        P[N, j] = 0
        P[:, j] = np.maximum(P[:, j], K - S)
    # interpolate to get the increments
    target_S = np.arange(170, 191)
    interp_func = interp1d(S, P[:, 0], kind='linear', fill_value="extrapolat
    target_P = interp_func(target_S)

    return target_P, target_S
```

In [239…

```
# same as the function above with different dx
def explicit_2(S0_start, S0_end, K, r, sigma, T, dt):
    dx = sigma * np.sqrt(3*dt)
    X_start = np.log(S0_start)
    X_end = np.log(S0_end)
    # define prices and increment
    N = int((X_end - X_start) / dx)
    log_grid = np.linspace(X_start, X_end, N + 1)
    S = np.exp(log_grid)  # Actual stock price grid
    # Time steps
    M = int(T / dt)

    # put option value matrix
    P = np.zeros((N + 1, M + 1))
    P[:, -1] = np.maximum(K - S, 0)

    # set probabilities
    P_u = dt*(((sigma**2)/(2*dx**2)) + ((r - .5 * sigma**2)/(2*dx)))
    P_m = 1 - dt*(sigma**2/dx**2) - r*dt
    P_d = dt*(((sigma**2)/(2*dx**2)) - ((r - .5 * sigma**2)/(2*dx)))

    # implement efd
    for j in range(M - 1, -1, -1):
        for i in range(1, N):
            P[i, j] = P_u * P[i+1, j+1] + P_m * P[i+1, j] + P_d * P[i+1, j-1
        # boundary conditions
        P[0, j] = max(K - S[0], P[0, j+1]*np.exp(-r*dt))
```

```
            P[N, j] = 0
            P[:, j] = np.maximum(P[:, j], K - S)
    # interpolate to get the increments
    target_S = np.arange(170, 191)
    interp_func = interp1d(S, P[:, 0], kind='linear', fill_value="extrapolat
    target_P = interp_func(target_S)

    return target_P, target_S
```

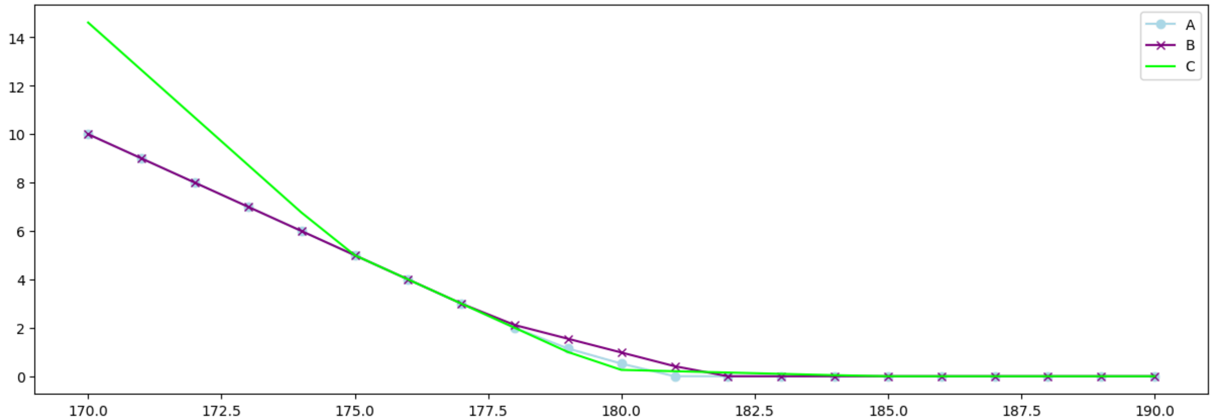
In [244…

```
# same as the function above with different dx
def explicit_3(S0_start, S0_end, K, r, sigma, T, dt):
    dx = sigma * np.sqrt(4*dt)
    X_start = np.log(S0_start)
    X_end = np.log(S0_end)
    # define prices and increment
    N = int((X_end - X_start) / dx)
    log_grid = np.linspace(X_start, X_end, N + 1)
    S = np.exp(log_grid)  # Actual stock price grid
    # Time steps
    M = int(T / dt)

    # put option value matrix
    P = np.zeros((N + 1, M + 1))
    P[:, -1] = np.maximum(K - S, 0)

    # set probabilities
    P_u = dt*(((sigma**2)/(2*dx**2)) + ((r - .5 * sigma**2)/(2*dx)))
    P_m = 1 - dt*(sigma**2/dx**2) - r*dt
    P_d = dt*(((sigma**2)/(2*dx**2)) - ((r - .5 * sigma**2)/(2*dx)))

    # implement efd
    for j in range(M - 1, -1, -1):
        for i in range(1, N):
            P[i, j] = P_u * P[i+1, j+1] + P_m * P[i+1, j] + P_d * P[i+1, j-1
        # boundary conditions
        P[0, j] = K - S[0] * np.exp(-r*(M-j)*dt)
        P[N, j] = 0
        P[:, j] = np.maximum(P[:, j], K - S)
    # interpolate to get the increments
    target_S = np.arange(170, 191)
    interp_func = interp1d(S, P[:, 0], kind='linear', fill_value="extrapolat
    target_P = interp_func(target_S)

    return target_P, target_S
```

In [245…

```
# get values
put_a, grid_a = explicit_1(170, 190, 180, .055, .25, .5, .002)
put_b, grid_b = explicit_2(170, 190, 180, .055, .25, .5, .002)
put_c, grid_c = explicit_3(170, 190, 180, .055, .25, .5, .002)

# plot the figure
plt.figure(figsize=(15, 5))
plt.plot(grid_a, put_a, label='A', color='lightblue', marker='o')
plt.plot(grid_b, put_b, label='B', color='purple', marker='x')
plt.plot(grid_c, put_c, label='C', color='lime')
```

```
plt.legend()
plt.show()
```



In [255…

```
# implicit
def implicit_1(S0_start, S0_end, K, r, sigma, T, dt):
    dx = sigma * np.sqrt(dt)
    X_start = np.log(S0_start)
    X_end = np.log(S0_end)
    # define prices and increment
    N = int((X_end - X_start) / dx)
    log_grid = np.linspace(X_start, X_end, N + 1)
    S = np.exp(log_grid)  # Actual stock price grid
    # Time steps
    M = int(T / dt)

    # put option value matrix
    P = np.zeros((N + 1, M + 1))
    P[:, -1] = np.maximum(K - S, 0)

    P_u = -.5*dt*(((sigma**2)/(dx**2)) + ((r - .5 * sigma**2)/dx))
    P_m = 1 + dt*(sigma**2/dx**2) + r*dt
    P_d = -.5*dt*(((sigma**2)/(dx**2)) - ((r - .5 * sigma**2)/dx))

    # implement efd
    for j in range(M - 1, -1, -1):
        for i in range(1, N):
            P[i, j] = P_u * P[i+1, j+1] + P_m * P[i+1, j] + P_d * P[i+1, j-1
        # boundary conditions
        P[0, j] = K - S[0] * np.exp(-r*(M-j)*dt)
        P[N, j] = 0
        P[:, j] = np.maximum(P[:, j], K - S)
    # interpolate to get the increments
    target_S = np.arange(170, 191)
    interp_func = interp1d(S, P[:, 0], kind='linear', fill_value="extrapolat
    target_P = interp_func(target_S)

    return target_P, target_S
```

In [256…

```
def implicit_2(S0_start, S0_end, K, r, sigma, T, dt):
    dx = sigma * np.sqrt(3*dt)
    X_start = np.log(S0_start)
    X_end = np.log(S0_end)
```

```
    # define prices and increment
    N = int((X_end - X_start) / dx)
    log_grid = np.linspace(X_start, X_end, N + 1)
    S = np.exp(log_grid)  # Actual stock price grid
    # Time steps
    M = int(T / dt)

    # put option value matrix
    P = np.zeros((N + 1, M + 1))
    P[:, -1] = np.maximum(K - S, 0)

    P_u = -.5*dt*(((sigma**2)/(dx**2)) + ((r - .5 * sigma**2)/dx))
    P_m = 1 + dt*(sigma**2/dx**2) + r*dt
    P_d = -.5*dt*(((sigma**2)/(dx**2)) - ((r - .5 * sigma**2)/dx))

    # implement efd
    for j in range(M - 1, -1, -1):
        for i in range(1, N):
            P[i, j] = P_u * P[i+1, j+1] + P_m * P[i+1, j] + P_d * P[i+1, j-1

        # boundary conditions
        P[0, j] = K - S[0] * np.exp(-r*(M-j)*dt)
        P[N, j] = 0
        P[:, j] = np.maximum(P[:, j], K - S)
    # interpolate to get the increments
    target_S = np.arange(170, 191)
    interp_func = interp1d(S, P[:, 0], kind='linear', fill_value="extrapolat
    target_P = interp_func(target_S)

    return target_P, target_S
```

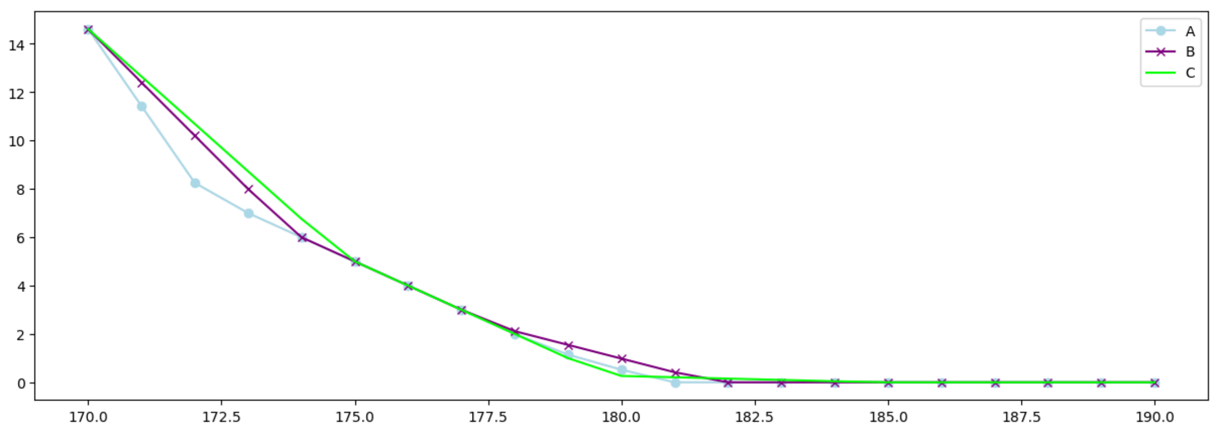
In [257…

```
def implicit_3(S0_start, S0_end, K, r, sigma, T, dt):
    dx = sigma * np.sqrt(4*dt)
    X_start = np.log(S0_start)
    X_end = np.log(S0_end)
    # define prices and increment
    N = int((X_end - X_start) / dx)
    log_grid = np.linspace(X_start, X_end, N + 1)
    S = np.exp(log_grid)  # Actual stock price grid
    # Time steps
    M = int(T / dt)

    # put option value matrix
    P = np.zeros((N + 1, M + 1))
    P[:, -1] = np.maximum(K - S, 0)

    P_u = -.5*dt*(((sigma**2)/(dx**2)) + ((r - .5 * sigma**2)/dx))
    P_m = 1 + dt*(sigma**2/dx**2) + r*dt
    P_d = -.5*dt*(((sigma**2)/(dx**2)) - ((r - .5 * sigma**2)/dx))

    # implement efd
    for j in range(M - 1, -1, -1):
        for i in range(1, N):
            P[i, j] = P_u * P[i+1, j+1] + P_m * P[i+1, j] + P_d * P[i+1, j-1

        # boundary conditions
```

```
        P[0, j] = K - S[0] * np.exp(-r*(M-j)*dt)
        P[N, j] = 0
        P[:, j] = np.maximum(P[:, j], K - S)
    # interpolate to get the increments
    target_S = np.arange(170, 191)
    interp_func = interp1d(S, P[:, 0], kind='linear', fill_value="extrapolat
    target_P = interp_func(target_S)

    return target_P, target_S
```

In [258…

```
# get values
put_2a, grid_2a = implicit_1(170, 190, 180, .055, .25, .5, .002)
put_2b, grid_2b = implicit_2(170, 190, 180, .055, .25, .5, .002)
put_2c, grid_2c = implicit_3(170, 190, 180, .055, .25, .5, .002)

# plot the figure
plt.figure(figsize=(15, 5))
plt.plot(grid_2a, put_2a, label='A', color='lightblue', marker='o')
plt.plot(grid_2b, put_2b, label='B', color='purple', marker='x')
plt.plot(grid_2c, put_2c, label='C', color='lime')
plt.legend()
plt.show()
```



In [298…

```
# crank-nikolson
def cn_1(S0_start, S0_end, K, r, sigma, T, dt):
    dx = sigma * np.sqrt(dt)
    X_start = np.log(S0_start)
    X_end = np.log(S0_end)
    # define prices and increment
    N = int((X_end - X_start) / dx)
    log_grid = np.linspace(X_start, X_end, N + 1)
    S = np.exp(log_grid)
    # time steps
    M = int(T / dt)

    # put option value matrix
    P = np.zeros((N + 1, M + 1))
    P[:, -1] = np.maximum(K - S, 0)

    P_u = -.25*dt*(((sigma**2)/(dx**2)) + ((r - .5 * sigma**2)/dx))
    P_m = 1 + dt*(sigma**2/dx**2) + r*dt
    P_d = -.25*dt*(((sigma**2)/(dx**2)) - ((r - .5 * sigma**2)/dx))
```

```
    # implement efd
    for j in range(M - 1, -1, -1):
        for i in range(1, N):
            P[i, j] = (-P_u * P[i+1, j+1] - (P_m - 1) * P[i+1, j] - P_d * P[

        # boundary conditions
        P[0, j] = K - S[0] * np.exp(-r*(M-j)*dt)
        P[N, j] = 0
        P[:, j] = np.maximum(P[:, j], K - S)
    # interpolate to get the increments
    target_S = np.arange(170, 191)
    interp_func = interp1d(S, P[:, 0], kind='linear', fill_value="extrapolat
    target_P = interp_func(target_S)

    return target_P, target_S
```

In [262…

```
# crank-nikolson
def cn_2(S0_start, S0_end, K, r, sigma, T, dt):
    dx = sigma * np.sqrt(3*dt)
    X_start = np.log(S0_start)
    X_end = np.log(S0_end)
    # define prices and increment
    N = int((X_end - X_start) / dx)
    log_grid = np.linspace(X_start, X_end, N + 1)
    S = np.exp(log_grid)  # Actual stock price grid
    # Time steps
    M = int(T / dt)

    # put option value matrix
    P = np.zeros((N + 1, M + 1))
    P[:, -1] = np.maximum(K - S, 0)

    P_u = -.25*dt*(((sigma**2)/(dx**2)) + ((r - .5 * sigma**2)/dx))
    P_m = 1 + dt*(sigma**2/dx**2) + r*dt
    P_d = -.25*dt*(((sigma**2)/(dx**2)) - ((r - .5 * sigma**2)/dx))

    # implement efd
    for j in range(M - 1, -1, -1):
        for i in range(1, N):
            P[i, j] = (-P_u * P[i+1, j+1] - (P_m - 1) * P[i+1, j] - P_d * P[

        # boundary conditions
        P[0, j] = K - S[0] * np.exp(-r*(M-j)*dt)
        P[N, j] = 0
        P[:, j] = np.maximum(P[:, j], K - S)
    # interpolate to get the increments
    target_S = np.arange(170, 191)
    interp_func = interp1d(S, P[:, 0], kind='linear', fill_value="extrapolat
    target_P = interp_func(target_S)

    return target_P, target_S
```

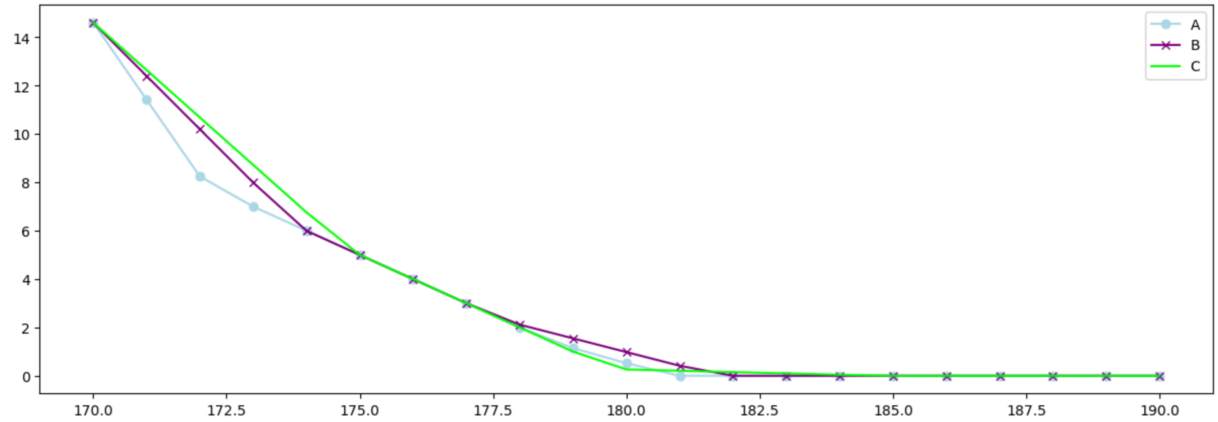
In [263…

```
# crank-nikolson
def cn_3(S0_start, S0_end, K, r, sigma, T, dt):
```

```
    dx = sigma * np.sqrt(4*dt)
    X_start = np.log(S0_start)
    X_end = np.log(S0_end)
    # define prices and increment
    N = int((X_end - X_start) / dx)
    log_grid = np.linspace(X_start, X_end, N + 1)
    S = np.exp(log_grid)  # Actual stock price grid
    # Time steps
    M = int(T / dt)

    # put option value matrix
    P = np.zeros((N + 1, M + 1))
    P[:, -1] = np.maximum(K - S, 0)

    P_u = -.25*dt*(((sigma**2)/(dx**2)) + ((r - .5 * sigma**2)/dx))
    P_m = 1 + dt*(sigma**2/dx**2) + r*dt
    P_d = -.25*dt*(((sigma**2)/(dx**2)) - ((r - .5 * sigma**2)/dx))

    # implement efd
    for j in range(M - 1, -1, -1):
        for i in range(1, N):
            P[i, j] = (-P_u * P[i+1, j+1] - (P_m - 1) * P[i+1, j] - P_d * P[

        # boundary conditions
        P[0, j] = K - S[0] * np.exp(-r*(M-j)*dt)
        P[N, j] = 0
        P[:, j] = np.maximum(P[:, j], K - S)
    # interpolate to get the increments
    target_S = np.arange(170, 191)
    interp_func = interp1d(S, P[:, 0], kind='linear', fill_value="extrapolat
    target_P = interp_func(target_S)

    return target_P, target_S
```

In [299…

```
# get values
put_3a, grid_3a = cn_1(170, 190, 180, .055, .25, .5, .002)
put_3b, grid_3b = cn_2(170, 190, 180, .055, .25, .5, .002)
put_3c, grid_3c = cn_3(170, 190, 180, .055, .25, .5, .002)

# plot the figure
plt.figure(figsize=(15, 5))
plt.plot(grid_3a, put_3a, label='A', color='lightblue', marker='o')
plt.plot(grid_3b, put_3b, label='B', color='purple', marker='x')
plt.plot(grid_3c, put_3c, label='C', color='lime')
plt.legend()
plt.show()
```



In [266…

```
# not working LOL
write_up = pd.DataFrame(columns = ['Explicit dx(a)', 'Explicit dx(b)', 'Expl
                                    'Crank-Nikolson dx(a)', 'Crank-Nikolson c
                          index= ['$170', '$171', '$172', '$173', '$174', '$1
                                  '$181', '$182', '$183', '$184', '$185', '$1
explicit_da, grid_da = explicit_1(170, 190, 180, .055, .25, .5, .002)
explicit_db, grid_db = explicit_2(170, 190, 180, .055, .25, .5, .002)
explicit_dc, grid_dc = explicit_3(170, 190, 180, .055, .25, .5, .002)
implicit_da, grid_2da = implicit_1(170, 190, 180, .055, .25, .5, .002)
implicit_db, grid_2db = implicit_2(170, 190, 180, .055, .25, .5, .002)
implicit_dc, grid_2dc = implicit_3(170, 190, 180, .055, .25, .5, .002)
crank_da, grid_3da = cn_1(170, 190, 180, .055, .25, .5, .002)
crank_db, grid_3db = cn_2(170, 190, 180, .055, .25, .5, .002)
crank_dc, grid_3dc = cn_3(170, 190, 180, .055, .25, .5, .002)
write_up['Explicit dx(a)'] = explicit_da
write_up['Explicit dx(b)'] = explicit_db
write_up['Explicit dx(c)'] = explicit_dc
write_up['Implicit dx(a)'] = implicit_da
write_up['Implicit dx(b)'] = implicit_db
write_up['Implicit dx(c)'] = implicit_dc
write_up['Crank-Nikolson dx(a)'] = crank_da
write_up['Crank-Nikolson dx(b)'] = crank_db
write_up['Crank-Nikolson dx(c)'] = crank_dc
write_up
```

Out[266…

	Explicit dx(a)	Explicit dx(b)	Explicit dx(c)	Implicit dx(a)	Implicit dx(b)	Implicit dx(c)	Crank-Nikolson dx(a)
$170	10.000000	10.000000	14.611304	14.611304	14.611304	14.611304	14.611304
$171	9.000000	9.000000	12.649298	11.429950	12.405434	12.649298	11.429950
$172	8.000000	8.000000	10.687292	8.248597	10.199564	10.687292	8.248597
$173	7.000000	7.000000	8.725286	7.000000	7.993694	8.725286	7.000000
$174	6.000000	6.000000	6.763279	6.000000	6.000000	6.763279	6.000000
$175	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000
$176	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000
$177	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000
$178	2.000000	2.115195	2.000000	2.000000	2.115195	2.000000	2.000000
$179	1.144948	1.548443	1.000000	1.144948	1.548443	1.000000	1.144948
$180	0.521333	0.981691	0.262743	0.521333	0.981691	0.262743	0.521333
$181	0.000000	0.414939	0.207885	0.000000	0.414939	0.207885	0.000000
$182	0.000000	0.000000	0.153028	0.000000	0.000000	0.153028	0.000000
$183	0.000000	0.000000	0.098170	0.000000	0.000000	0.098170	0.000000
$184	0.000000	0.000000	0.043313	0.000000	0.000000	0.043313	0.000000
$185	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
$186	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
$187	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
$188	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
$189	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
$190	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Problem 6.

Consider the following information on the stock of company XYZ: The volatility of the stock price is σ = 25% per annum. Assume the prevailing risk-free rate is r = 5.5% per annum. Use the Black-Scholes PDE (for S) to price American Put options with strike prices of K = $170, expiration of 6 months and current stock prices for a range from $170 to $190; using the specified methods below: (a) Explicit Finite-Difference method:

In [295…

```
```

```
Initial option values: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 1: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 2: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 3: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 4: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 5: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 6: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 7: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 8: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 9: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 10: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 11: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 12: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 13: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 14: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 15: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 16: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 17: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 18: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
```

```
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 19: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 20: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 21: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 22: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 23: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 24: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 25: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 26: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 27: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 28: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 29: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 30: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 31: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 32: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 33: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 34: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 35: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 36: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 37: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
```

```
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 38: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 39: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 40: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 41: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 42: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 43: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 44: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 45: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 46: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 47: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 48: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 49: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 50: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 51: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 52: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 53: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 54: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 55: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
Option values at step 56: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 57: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 58: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 59: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 60: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 61: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 62: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 63: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 64: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 65: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 66: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 67: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 68: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 69: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 70: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 71: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 72: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 73: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 74: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
```

```
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 75: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 76: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 77: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 78: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 79: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 80: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 81: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 82: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 83: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 84: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 85: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 86: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 87: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 88: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 89: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 90: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 91: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 92: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 93: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
```

```
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 94: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 95: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 96: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 97: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 98: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 99: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 100: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 101: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 102: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 103: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 104: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 105: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 106: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 107: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 108: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 109: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 110: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 111: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
Option values at step 112: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 113: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 114: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 115: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 116: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 117: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 118: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 119: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 120: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 121: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 122: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 123: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 124: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 125: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 126: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 127: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 128: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 129: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 130: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
```

```
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 131: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 132: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 133: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 134: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 135: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 136: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 137: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 138: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 139: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 140: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 141: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 142: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 143: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 144: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 145: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 146: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 147: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 148: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 149: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
```

```
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 150: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 151: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 152: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 153: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 154: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 155: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 156: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 157: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 158: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 159: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 160: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 161: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 162: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 163: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 164: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 165: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 166: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 167: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
Option values at step 168: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 169: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 170: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 171: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 172: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 173: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 174: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 175: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 176: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 177: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 178: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 179: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 180: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 181: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 182: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 183: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 184: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 185: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 186: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
```

```
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 187: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 188: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 189: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 190: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 191: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 192: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 193: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 194: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 195: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 196: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 197: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 198: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 199: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 200: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 201: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 202: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 203: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 204: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 205: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
```

```
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 206: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 207: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 208: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 209: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 210: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 211: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 212: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 213: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 214: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 215: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 216: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 217: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 218: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 219: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 220: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 221: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 222: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 223: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
Option values at step 224: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 225: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 226: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 227: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 228: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 229: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 230: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 231: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 232: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 233: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 234: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 235: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 236: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 237: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 238: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 239: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 240: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 241: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 242: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
```

```
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 243: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 244: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 245: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 246: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 247: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 248: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 249: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Option values at step 250: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Out[295…

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0.])
```