MScFE 620 DERIVATIVE PRICING

Group Work Project # 1

By

Arijit Chatterjee

Alper Ülkü

STEP 1: Put Call Parity in Context of Binomial Tree (Q1-Q10)

1. Does put-call parity apply for European options? Why or why not?

put-call parity is applicable only when the underlying , expiration and strike price are the same. For European options the exercise only happens at the expiry date not before. So, for european options if the underlying and strike price and expiration are same then the put-call parity holds.

2. Rewrite put-call parity to solve for the call price in terms of everything else.

Call price = Put price - K * e^(-rT)+ S0

where, K = Strike, r = risk free rate, T = option maturity, S0 = spot price

3. Rewrite put-call parity to solve for the put price in terms of everything else.

Put price = Call price + K * e^(-rT)-S0

where, K = Strike, r = risk free rate, T = option maturity, S0 = spot price

4. Does put-call parity apply for American options? Why or why not?

put-call parity is applicable only when the underlying , expiration and strike price are the same. For American options the maturity is not fixed as it can be exercised at any time during the life of the option. Due to this the price of the American option considers the possibility of early exercise. Due to this the put call parity does not hold true for the American option.

5. Price an ATM European call and put using a binomial tree:

Let us assume that $S0$=100; r=5%; σ=20%; T=3$months$

```
import numpy as np
```

```
#European call option
def call_option_delta(S_ini, K, T, r, sigma, N):
    dt = T / N  # Define time step
    u = np.exp(sigma * np.sqrt(dt))  # Define u
    d = np.exp(-sigma * np.sqrt(dt))  # Define d
    p = (np.exp(r * dt) - d) / (u - d)  # risk neutral probs
    C = np.zeros([N + 1, N + 1])  # call prices
    S = np.zeros([N + 1, N + 1])  # underlying price
    Delta = np.zeros([N, N])  # delta
    for i in range(0, N + 1):
        C[N, i] = max(S_ini * (u ** (i)) * (d ** (N - i)) - K, 0)
        S[N, i] = S_ini * (u ** (i)) * (d ** (N - i))
    for j in range(N - 1, -1, -1):
        for i in range(0, j + 1):
            C[j, i] = np.exp(-r * dt) * (p * C[j + 1, i + 1] + (1 - p) * C[j + 1, i])
            S[j, i] = S_ini * (u ** (i)) * (d ** (j - i))
            Delta[j, i] = (C[j + 1, i + 1] - C[j + 1, i]) / (
                S[j + 1, i + 1] - S[j + 1, i]
            )
    return C[0, 0], C, S, Delta
```

```
#European call option
import numpy as np
price_array = []
for N in [1, 10,50, 100, 200, 300, 400, 500]:
    call_price, C, S, delta = call_option_delta(100, 100, 1/4,0.05, 0.2,N)
    price_array.append(call_price)
    print("With N = {:3d}, the European call price is {:.2f}".format(N, call_price))
```

```
With N =   1, the European call price is 5.59
With N =  10, the European call price is 4.52
With N =  50, the European call price is 4.60
With N = 100, the European call price is 4.61
With N = 200, the European call price is 4.61
With N = 300, the European call price is 4.61
With N = 400, the European call price is 4.61
With N = 500, the European call price is 4.61
```

```
#European put option
def put_option_delta(S_ini, K, T, r, sigma, N):
    dt = T / N  # Define time step
    u = np.exp(sigma * np.sqrt(dt))  # Define u
    d = np.exp(-sigma * np.sqrt(dt))  # Define d
    p = (np.exp(r * dt) - d) / (u - d)  # risk neutral probs
    P = np.zeros([N + 1, N + 1])  # call prices
    S = np.zeros([N + 1, N + 1])  # underlying price
    Delta = np.zeros([N, N])  # delta
    for i in range(0, N + 1):
        P[N, i] = max(-(S_ini * (u ** (i)) * (d ** (N - i))) + K, 0)
        S[N, i] = S_ini * (u ** (i)) * (d ** (N - i))
    for j in range(N - 1, -1, -1):
        for i in range(0, j + 1):
            P[j, i] = np.exp(-r * dt) * (p * P[j + 1, i + 1] + (1 - p) * P[j + 1, i])
            S[j, i] = S_ini * (u ** (i)) * (d ** (j - i))
            Delta[j, i] = (P[j + 1, i + 1] - P[j + 1, i]) / (
                S[j + 1, i + 1] - S[j + 1, i]
            )
    return P[0, 0], P, S, Delta
```

```
#European put option
import numpy as np
price_array = []
for N in [1, 10, 50,100, 200, 300, 400, 500]:
    put_price, P, S, delta = put_option_delta(100, 100, 1/4,0.05, 0.2,N)
    price_array.append(put_price)
    print("With·N·=·{:3d},·the·European·put·price·is·{:.2f}".format(N,·put_price))
```

```
With N =   1, the European put price is 4.34
With N =  10, the European put price is 3.27
With N =  50, the European put price is 3.35
With N = 100, the European put price is 3.36
With N = 200, the European put price is 3.37
With N = 300, the European put price is 3.37
With N = 400, the European put price is 3.37
With N = 500, the European put price is 3.37
```

5.a) & b) From the above call and put price we can see post the 100 steps the prices are converging. Here the 100 steps are the steps to cover the life of the option i.e. 3 months. As we know the accuracy of the price of option increases with the number of steps, however we also acknowledge that the more number of steps will increase the computational cost. However, for our calculation we can safely use N = 100. which gives us the call price as 4.61 and put price as 3.36. We have written the code such that it can price the option as well as delta for the later stage. The code takes input as Strike = 100 (as ATM), Spot = 100, volatility =20%, risk free rate = 5% and time to maturity as 3 months i.e. 1/4 years. The code take these inputs and calculate $u=e\sigma dt\sqrt{}$ and $d=e-\sigma dt\sqrt{}$ and provide the call and put option price as output

6.a & 6.b

```
#European call option delta
call_price, C, S, delta1 = call_option_delta(100, 100, 1/4,0.05, 0.2,100)
print(np.round(delta1,2))
```

```
[[0.57 0.   0.   ... 0.   0.   0.  ]
 [0.53 0.61 0.   ... 0.   0.   0.  ]
 [0.49 0.57 0.65 ... 0.   0.   0.  ]
 ...
 [0.   0.   0.   ... 1.   0.   0.  ]
 [0.   0.   0.   ... 1.   1.   0.  ]
 [0.   0.   0.   ... 1.   1.   1.  ]]
```

```
#European put option delta
put_price, P, S, delta1 = put_option_delta(100, 100, 1/4,0.05, 0.2,100)
print(np.round(delta1,2))
```

```
[[-0.43  0.    0.   ...  0.    0.    0.  ]
 [-0.47 -0.39  0.   ...  0.    0.    0.  ]
 [-0.51 -0.43 -0.35 ...  0.    0.    0.  ]
 ...
 [-1.   -1.   -1.   ...  0.    0.    0.  ]
```

```
       [-1.   -1.   -1.   ...  0.    0.    0.  ]
       [-1.   -1.   -1.   ...  0.    0.    0.  ]]
```

6.a. From the above outcome we can observe the delta of the european call option as +0.57 whereas for the european put option it is -0.43 at t=0. We can see although the S=K so this is an ITM and the ideal level for the delta should be +-0.5 but it is not the case due to volatility.

6.b For call option the delta is positive, the reason is call option gives the buyer right to buy at a predefined price so for the seller the risk is upside i.e. if price goes up which why positive delta imply that much underlying stock should be purchased by the option seller. Similarly for the put option the delta is negative as for the put seller the risk is downside i.e. if price goes below the strike. To hedge this risk the option seller need to sell the underlying and mitigate the risk.

```
#European call option
call_price, C, S, delta1 = call_option_delta(100, 100, 1/4,0.05, 0.25,100)
print("The European call price with 25% volatility is {:.2f}".format(call_price))
```

```
    The European call price with 25% volatility is 5.59
```

```
#European put option
put_price, P, S, delta1 = put_option_delta(100, 100, 1/4,0.05, 0.25,100)
print("The European put price with 25% volatility is {:.2f}".format(put_price))
```

```
    The European put price with 25% volatility is 4.34
```

7.a.The premium for both the options have increased now. for the european call option it is 5.59 and for the european put option it is 4.34.

7.b. We are pricing the options at ATM due to which the increase in volatility increases the chance of more price fluctuation which is why the premium is going up.

8. Repeat Q5, but this time consider options (call and put) of American style. (Answer sections a and b of Q5 as well)

8.a & 8.b

As we understand the number of steps N is used to determine the size of each time step. So, in our case it is (3/12 (years))/N. Ideally the smaller the N is the accuracy of the model increases. too large N also computationally expensive. So, we have run code with various Ns i.e. [10, 20,50,100,200,500] to check the price convergence or to understand after which N the option price is not changing. in the following code we have defined a function to price an american option. The input of the functions are Spot, Strike, risk free rate, volatility, time , number of steps[N] and a flag to define call or put. In our case as the price requirement at ATM the strike and spot will be the same = 100, risk free rate is 5% volatility (sigma) 20% time 3 month ~ 3/12 years ~ .25 years.

```
#American opton pricing using binomial tree

import numpy as np

def american_option_binomial(S, K, r, sigma, T, N, is_call=True):
    dt = T/N
    u = np.exp(sigma*np.sqrt(dt))
    d = 1/u
    p = (np.exp(r*dt) - d) / (u - d)
    price = np.zeros((N+1, N+1))
    price1 = np.zeros((N+1,N+1))

    for i in range(0,N+1):
      price1[N,i] = S*(u**(i))*(d**(N-i))
      if is_call:
        price[N,i] = np.maximum(0, price1[N,i] - K)
    else:
        price[N,i] = np.maximum(0, K - price1[N,i])
    for j in range(N-1, -1, -1):
        for i in range(0,j+1):
            price[j, i] = np.exp(-r*dt) * (p * price[j+1, i+1] + (1 - p) * price[j+1, i])
            price1[j,i] = S*(u**(i))*(d**(j-i))
            if is_call:
                price[j,i] = np.maximum(price[j,i], price1[j,i] - K)
            else:
                price[j,i] = np.maximum(price[j, i], K - price1[j,i])
    return price[0, 0]


#American call option price based on different N
for N in  [10, 20,50,100,200,500]:
  price = american_option_binomial(100, 100, .05, .2, 1/4, N,1)
  price = round(price,2)
  print("With N = {:3d}, the American call price is {:.2f}".format(N, price))
```

```
With N =  10, the American call price is 4.52
With N =  20, the American call price is 4.57
With N =  50, the American call price is 4.60
With N = 100, the American call price is 4.61
With N = 200, the American call price is 4.61
With N = 500, the American call price is 4.61
```

```
#American put option price based on different N
for N in  [10, 20,50,100,200,500]:
  price = american_option_binomial(100, 100, .05, .2, 1/4, N,0)
  price = round(price,2)
  print("With N = {:3d}, the American put price is {:.2f}".format(N, price))
```

```
With N =  10, the American put price is 3.43
With N =  20, the American put price is 3.45
With N =  50, the American put price is 3.47
With N = 100, the American put price is 3.47
With N = 200, the American put price is 3.48
With N = 500, the American put price is 3.48
```

From the above numbers we can see the price of the option is converging from 50 to 200 N. however the variance between the 50 and 200 is very low for the call and put option respectively 4.6 vs 4.61 and 3.47 vs 3.48.

For our model we will use N = 50 for a reliable outcome.

So the price of an American call option will be 4.6 and the put option will be 3.47

```
#American opton pricing using binomial tree

import numpy as np

def american_option_binomial1(S, K, r, sigma, T, N, call=True):
    dt = T/N
    u = np.exp(sigma*np.sqrt(dt))
    d = 1/u
    p = (np.exp(r*dt) - d) / (u - d)
    price = np.zeros([N+1, N+1])
    price1 = np.zeros([N+1,N+1])
    Delta = np.zeros([N,N])
    for i in range(0,N+1):
      price1[N,i] = S*(u**(i))*(d**(N-i))
      if call:
        price[N,i] = np.maximum(0, price1[N,i] - K)
    else:
        price[N,i] = np.maximum(0, K - price1[N,i])
    for j in range(N-1, -1, -1):
        for i in range(0,j+1):
            price[j, i] = np.exp(-r*dt) * (p * price[j+1, i+1] + (1 - p) * price[j+1, i])
            price1[j,i] = S*(u**(i))*(d**(j-i))
            Delta[j,i] = round((price[j+1,i+1]-price[j+1,i])/(price1[j+1,i+1]-price1[j+1,i]),2)
            if call:
                price[j,i] = np.maximum(price[j,i], price1[j,i] - K)
            else:
                price[j,i] = np.maximum(price[j, i], K - price1[j,i])

    return round(price[0, 0],2), Delta


print("The american call option delta")
american_option_binomial1(100,100,0.05,0.2,1/4,50,1)
```

```
The american call option delta
(4.6, array([[  0.57,   0.  ,   0.  , ...,   0.  ,   0.  ,   0.  ],
       [  0.51,   0.62,   0.  , ...,   0.  ,   0.  ,   0.  ],
       [  0.45,   0.57,   0.68, ...,   0.  ,   0.  ,   0.  ],
       ...,
       [  0.  ,   0.  ,   0.  , ...,   1.  ,   0.  ,   0.  ],
       [  0.  ,   0.  ,   0.  , ...,   1.  ,   1.  ,   0.  ],
       [  0.  ,   0.  ,   0.  , ...,   1.  ,   1.  , -17.18]]))
```

```
print("The american put option delta")
american_option_binomial1(100,100,0.05,0.2,1/4,50,0)
```

```
The american put option delta
(3.47, array([[-0.45,  0.  ,  0.  , ...,  0.  ,  0.  ,  0.  ],
       [-0.51, -0.39,  0.  , ...,  0.  ,  0.  ,  0.  ],
       [-0.58, -0.45, -0.33, ...,  0.  ,  0.  ,  0.  ],
       ...,
       [-1.  , -1.  , -1.  , ...,  0.  ,  0.  ,  0.  ],
       [-1.  , -1.  , -1.  , ...,  0.  ,  0.  ,  0.  ],
       [ 0.  ,  0.  ,  0.  , ...,  0.  ,  0.  ,  0.  ]]))
```

9. Repeat Q6, but considering American-style options. (Answer/comment on sections a and b of Q6 as well).

9.a. As we can see the delta of the american call is coming 0.57 at time 0 and the american put is coming -0.45 at time 0. 9.b. Given that the option is an ATM the ideal level of delta should at time 0 +- .5. However we can observe little different delta for both the options. delta for the call option meaning the price of the option will increase along with the increase in underlying. In our case the call option delta at t=0 is 0.57. On other hand for the put option delta imply the downside change in the price leading to the change in the option price. In our case the put option delta at t =0 is -0.45. delta also can be used to assess the probability of an option to expire ITM. In our case it is approximately 50% for both the cases. Delta hedging is an important aspect of option trading. The explanations are as follows. For the American call option the delta is .57 which means for 100 shares (lot size) of the underlying we can sell 57 shares to hedge the option price to avoid loss due to the downside movement of the underlying. Similarly for the American put option the delta is -0.45 which means we have to buy 45 shares of the underlying to hedge the price movement of the underlying.

```
#Price of the American call option with 5% increased volatility
print("The american call option with 25% volatility")
round(american_option_binomial(100,100,0.05,0.25,1/4,50,1),2)
```

```
The american call option with 25% volatility
5.57
```

```
#Price of the American put option with 5% increased volatility
print("The american put option with 25% volatility")
round(american_option_binomial(100,100,0.05,0.25,1/4,50,0),2)
```

```
The american put option with 25% volatility
4.45
```

10.Repeat Q7, but considering American-style options. (Answer/comment on sections a and b of Q7 as well).

10.a When we increase the volatility the price of both the option changes. As the increased volatility implied more risk which needed to be compensated by more expensive option price. Price of the American call option is 5.57 and price of the American put option is 4.45.

10.b. We can observe the American call option price have increased by 21% whereas the american put price increased by 28%. From this change we can see the put option become more expensive with the same strike and spot due to the increase in volatility.So, we can say that the increase in volatility leads to price uncertainty which mostly impacts on downside due to which the put option becomes more expensive.

STEP 2: Trinomial Trees (Q15-Q18)

15.

```
import math
import numpy as np
class SOption(object):
    def __init__(
        self, S0, K, r=0.05, T=1/4, N=100, pu=0, pd=0, sigma=0, is_put=False, is_american=False):
        self.S0 = S0
        self.K = K
        self.r = r
        self.T = T
        self.N = max(1, N)
        self.STs = []
        self.pu, self.pd = pu, pd
        self.sigma = sigma
        self.is_call = not is_put
        self.is_european = not is_american
    @property
    def dt(self):
        return self.T/float(self.N)
    @property
    def df(self):
        return math.exp(-(self.r)*self.dt)
class BTOption(SOption):
    def setup_parameters(self):
        self.u = 1+self.pu  # Expected value up
        self.d = 1-self.pd  # Expected value down
        self.qu = (math.exp((self.r)*self.dt)-self.d)/(self.u-self.d)
        self.qd = 1-self.qu
    def init_stock_price_tree(self):
        self.STs = [np.array([self.S0])]
        for i in range(self.N):
            prev_br = self.STs[-1]
```

```
            pre_bs = self.STs[-1]
            st = np.concatenate((pre_bs*self.u,[pre_bs[-1]*self.d]))
            self.STs.append(st)
    def init_payoffs_tree(self):
        if self.is_call:
            return np.maximum(0, self.STs[self.N]-self.K)
        else:
            return np.maximum(0, self.K-self.STs[self.N])
    def check_early_exercise(self, payoffs, node):
        if self.is_call:
            return np.maximum(payoffs, self.STs[node] - self.K)
        else:
            return np.maximum(payoffs, self.K - self.STs[node])
    def traverse_tree(self, payoffs):
        for i in reversed(range(self.N)):
            payoffs = (payoffs[:-1]*self.qu+payoffs[1:]*self.qd)*self.df
            if not self.is_european:
                payoffs = self.check_early_exercise(payoffs,i)
        return payoffs
    def begin_tree_traversal(self):
        payoffs = self.init_payoffs_tree()
        return self.traverse_tree(payoffs)
    def price(self):
        self.setup_parameters()
        self.init_stock_price_tree()
        payoffs = self.begin_tree_traversal()
        return payoffs[0]
class TTreeOption(BTOption):
    def setup_parameters(self):
        self.u = math.exp(self.sigma*math.sqrt(2.*self.dt))
        self.d = 1/self.u
        self.m = 1
        self.qu = ((math.exp((self.r) * self.dt/2.)-math.exp(-self.sigma * math.sqrt(self.dt/2.))) /(math.exp(self.sigma
                        math.sqrt(self.dt/2.)) - math.exp(-self.sigma * math.sqrt(self.dt/2.))))**2
        self.qd = ((math.exp(self.sigma * math.sqrt(self.dt/2.)) - math.exp((self.r) * self.dt/2.)) /
                   (math.exp(self.sigma *math.sqrt(self.dt/2.)) - math.exp(-self.sigma * math.sqrt(self.dt/2.))))**2.
        self.qm = 1 - self.qu - self.qd
    def init_stock_price_tree(self):
        self.STs = [np.array([self.S0])]
        for i in range(self.N):
            prev_nodes = self.STs[-1]
            self.ST = np.concatenate((prev_nodes*self.u, [prev_nodes[-1]*self.m, prev_nodes[-1]*self.d]))
            self.STs.append(self.ST)
    def traverse_tree(self, payoffs):
        for i in reversed(range(self.N)):
            payoffs = (payoffs[:-2] * self.qu +
                       payoffs[1:-1] * self.qm +
                       payoffs[2:] * self.qd) * self.df
            if not self.is_european:
                payoffs = self.check_early_exercise(payoffs,i)
        return payoffs


#15.a.Pricing of European Call option with 5 different strike
for MN in  [.90,.95,1,1.05,1.1]:
  european_call = TTreeOption(100, 100*MN, r=0.05, T=1/4, N=100, sigma=0.2, is_american = False,is_put=False)
  european_call1 = round(european_call.price(),2)
  print("Moneyness","{:.2f}".format(MN), "European Call option price","{:.2f}".format(european_call1))
```

```
Moneyness 0.90 European Call option price 11.67
Moneyness 0.95 European Call option price 7.72
Moneyness 1.00 European Call option price 4.61
Moneyness 1.05 European Call option price 2.48
Moneyness 1.10 European Call option price 1.19
```

15.b. We can observe the call option price increases with decrease in strike i.e. at 90% moneyness the strike is 90 and spot is 100 so the call option payoff is +10 which is deep in the money and option price is maximum for this. the call option price decreases based on the change in moneyness. We can observe the call option price is lowest for the strike at 110% i.e. 110 as in this case the call option is deep Out of the money so the call option is cheapest.

```
#16.a.Pricing of European put option with 5 different strike
for MN in  [.90,.95,1,1.05,1.1]:
  european_put = TTreeOption(100, 100*MN, r=0.05, T=1/4, N=100, sigma=0.2, is_put=True)
  european_put1 = round(european_put.price(),2)
  print("Moneyness","{:.2f}".format(MN), "European Put option price","{:.2f}".format(european_put1))
```

```
Moneyness 0.90 European Put option price 0.55
Moneyness 0.95 European Put option price 1.54
Moneyness 1.00 European Put option price 3.37
Moneyness 1.05 European Put option price 6.18
Moneyness 1.10 European Put option price 9.83
```

16.b. We can observe the put option price increases with increase in strike i.e. at 110% moneyness the strike is 110 and spot is 100 so the put option payoff is +10 which is deep in the money and put option price is maximum for this. the put option price decreases based on the change in moneyness. We can observe the put option price is lowest for the strike at 90% i.e. 90 as in this case the put option is deep Out of the money (the current payoff is 0 as the spot is at 100 and strike at 90, to make the option payoff positive in this case the spot need to be lower than 90, which is difficult as to achieve that the underlying price need to drop by 10%) so the put option is cheapest.

17.a

```
#17.a.Pricing of American Call option with 5 different strike
for MN in  [.90,.95,1,1.05,1.1]:
  American_call = TTreeOption(100, 100*MN, r=0.05, T=1/4, N=100, sigma=0.2, is_american = True, is_put=False)
  American_call1 = round(American_call.price(),2)
  print("Moneyness","{:.2f}".format(MN), "American Call option price","{:.2f}".format(American_call1))
```

```
Moneyness 0.90 American Call option price 11.67
Moneyness 0.95 American Call option price 7.72
Moneyness 1.00 American Call option price 4.61
Moneyness 1.05 American Call option price 2.48
Moneyness 1.10 American Call option price 1.19
```

17.b. We can observe the call option price increases with decrease in strike i.e. at 90% moneyness the strike is 90 and spot is 100 so the call option payoff is +10 which is deep in the money and option price is maximum for this. the call option price decreases based on the change in moneyness. We can observe the call option price is lowest for the strike at 110% i.e. 110 as in this case the call option is deep Out of the money so the call option is cheapest.

```
#18.a.Pricing of American Put option with 5 different strike
for MN in  [.90,.95,1,1.05,1.1]:
  American_put = TTreeOption(100, 100*MN, r=0.05, T=1/4, N=100, sigma=0.2, is_american = True, is_put=True)
  American_put1 = round(American_put.price(),2)
  print("Moneyness","{:.2f}".format(MN), "American Put option price","{:.2f}".format(American_put1))
```

```
Moneyness 0.90 American Put option price 0.56
Moneyness 0.95 American Put option price 1.58
Moneyness 1.00 American Put option price 3.48
Moneyness 1.05 American Put option price 6.43
Moneyness 1.10 American Put option price 10.33
```

18.b. We can observe the call option price increases with decrease in strike i.e. at 90% moneyness the strike is 90 and spot is 100 so the call option payoff is +10 which is deep in the money and option price is maximum for this. the call option price decreases based on the change in moneyness. We can observe the call option price is lowest for the strike at 110% i.e. 110 as in this case the call option is deep Out of the money so the call option is cheapest.

STEP 3: Real World Questions (Q25-Q27)

Dynamic Delta Hedging: 25.a. & 25.b will be in the main document

26.a

```
# American put option using 25 path
# K = 182
# S = 180
# r = 2%
# Sigma = 25%
# T =0.5 Year
# Step = 25
print("The binomial delta calculation with stated parameters and price of a put option")
american_option_binomial1(180,182,0.02,0.25,1/2,25,0)
```

```
[-1.  , -1.  , -0.98, -0.88, -0.71, -0.5 , -0.29, -0.14, -0.05,
 -0.01, -0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -0.95, -0.81, -0.61, -0.39, -0.2 , -0.08,
 -0.02, -0.  , -0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -0.99, -0.9 , -0.73, -0.5 , -0.28, -0.12,
 -0.04, -0.01, -0.  , -0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -1.  , -0.97, -0.84, -0.63, -0.38, -0.18,
 -0.06, -0.01, -0.  , -0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -1.  , -1.  , -0.92, -0.75, -0.51, -0.27,
 -0.1 , -0.03, -0.  , -0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -1.  , -1.  , -0.98, -0.87, -0.65, -0.38,
 -0.16, -0.04, -0.01, -0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -0.95, -0.79, -0.52,
 -0.25, -0.08, -0.01, -0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -0.9 , -0.67,
 -0.37, -0.13, -0.03, -0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -0.98, -0.83,
 -0.53, -0.22, -0.05, -0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -0.95,
 -0.71, -0.35, -0.09, -0.01,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  ,
 -0.9 , -0.54, -0.17, -0.02,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  ,
 -1.  , -0.78, -0.31, -0.04,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  ,
 -1.  , -1.  , -0.57, -0.07,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,
  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ,  0.  ],
[-1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  , -1.  ,
 -1.  , -1.  , -1.  , -0.15,  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.  ]]))
```

27 a. will be in the main document

✓ 0s completed at 03:03