# Algorítmica y Programación

Enero - Mayo 2020

Dr. Iván S. Razo Zapata
(ivan.razo@itam.mx)

ITAM

# Programación Orientada a Objetos

# Temas para esta sesión

- **Ejercicio**

# Otra forma de modelar sistemas

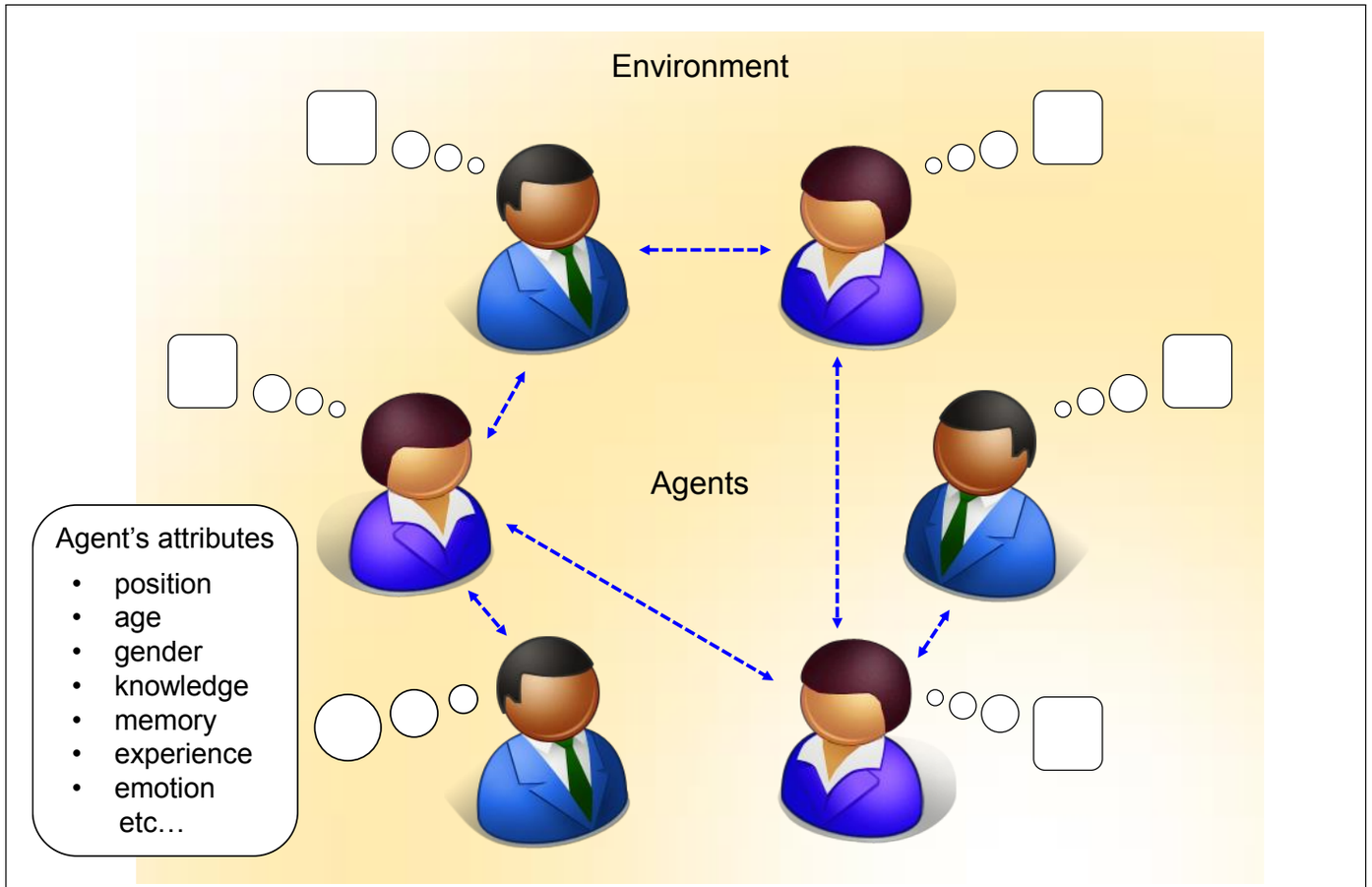



Figure 19.1: Schematic illustration of what an agent-based model (ABM) looks like.

*Agent-based models* are computational simulation models that involve many discrete *agents.*

# Agent-based models (ABMs)

- ABMs are widely used in a variety of disciplines to simulate dynamical behaviors of systems made of a large number of entities
- Traders' behaviors in a market (in **economics**)
- Migration of people (in **social sciences**)
- Interaction among employees and their performance improvement (in **organizational science**)
- Flocking/schooling behavior of birds/fish (in **behavioral ecology**)
- Cell growth and morphogenesis (in d**evelopmental biology**), and
- Collective behavior of granular materials (in **physics**)

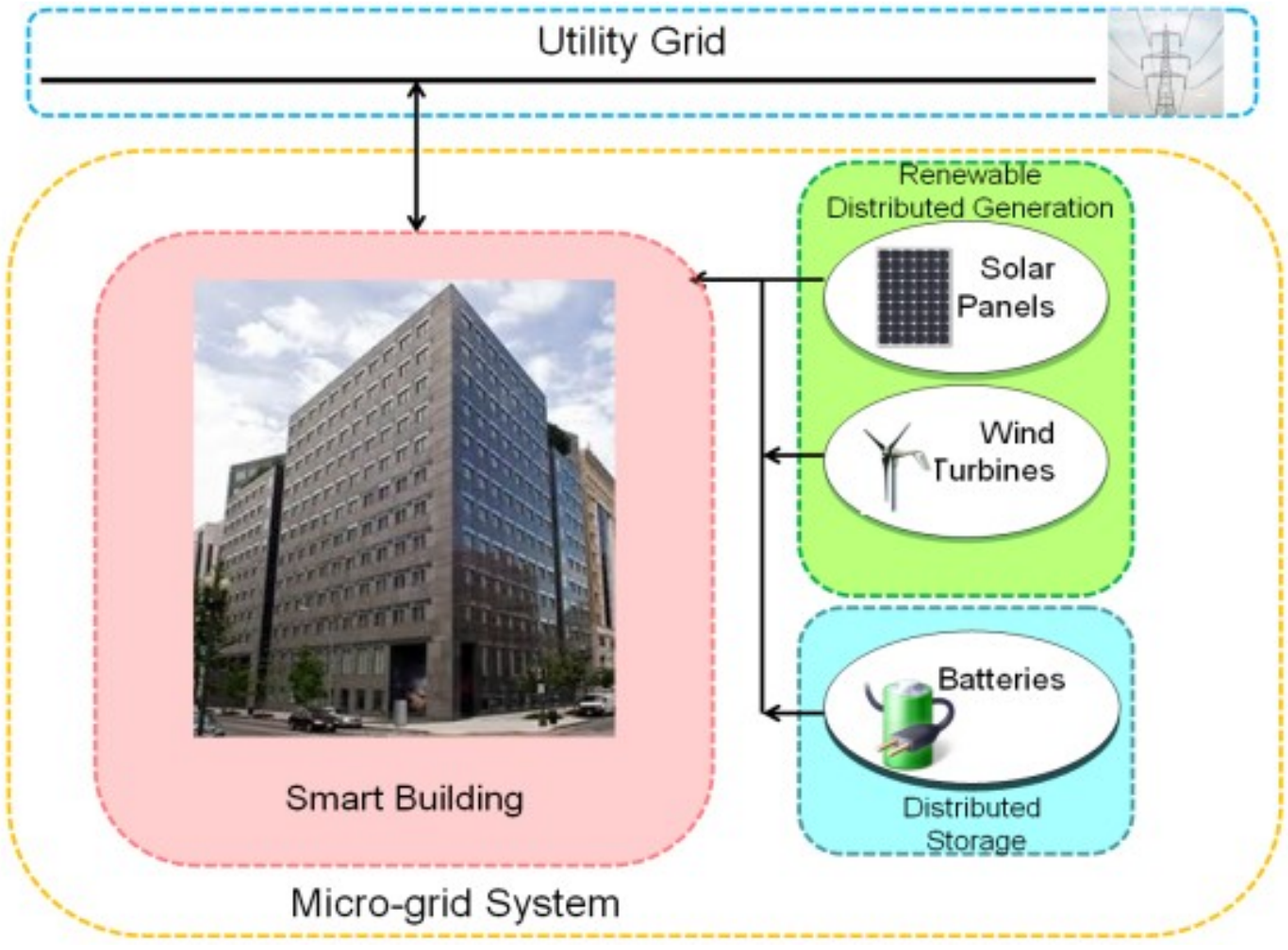
ITAM

# Sistemas basados en agentes - Propiedades

- Agents are **discrete** entities
- Agents may have **internal states**
- Agents may **perceive** and **interact** with the **environment**
- Agents may behave based on **predefined rules**
- Agents may be able to **learn** and **adapt**
- Agents may **interact** with other **agents**
- ABMs often **lack central supervisors/controllers**
- ABMs may produce nontrivial **“collective behavior”** as a whole

ITAM

# Ejercicio




Journals & Magazines > IEEE Transactions on Smart Grid > Volume: 4 Issue: 2

**Adaptive Negotiation Agent for Facilitating Bi-Directional Energy Trading Between Smart Building and Utility Grid**

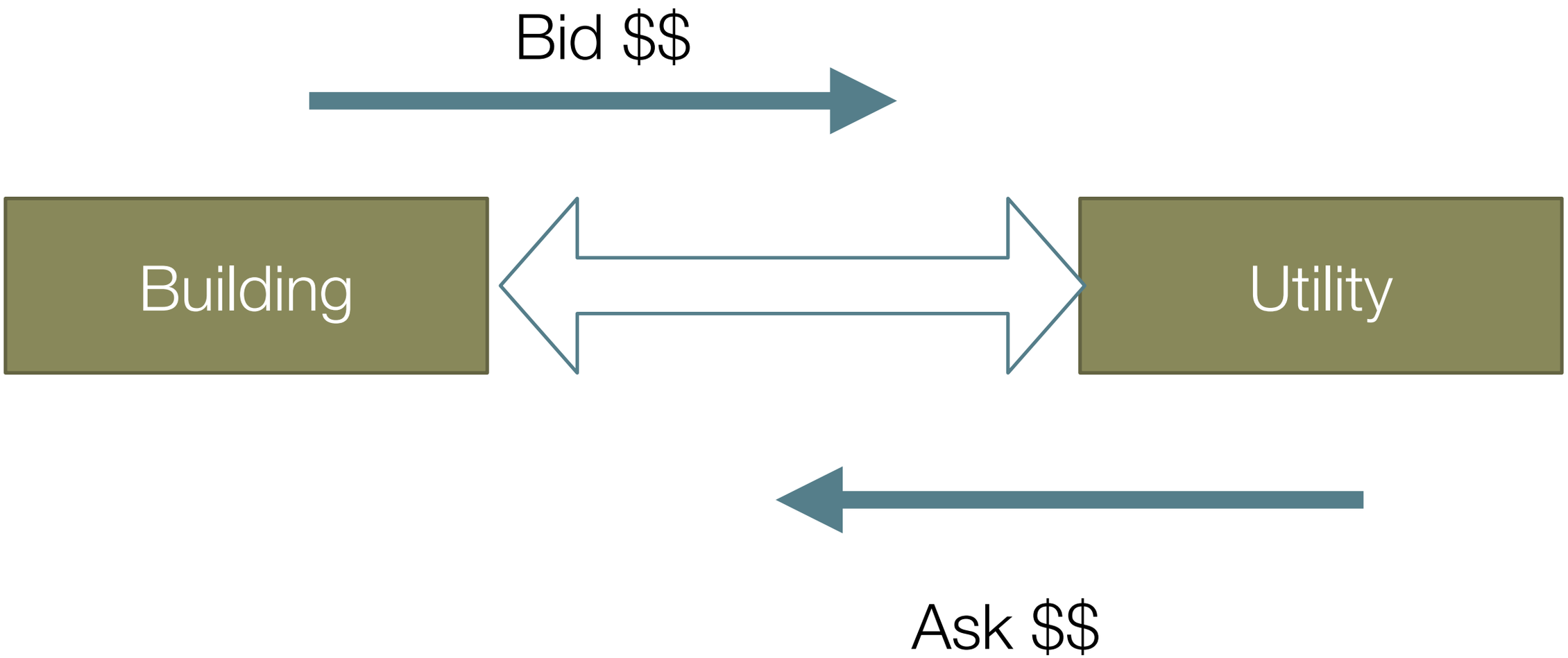
**Publisher: IEEE** Cite This PDF

**2 Author(s)** Zhu Wang ; Lingfeng Wang **All Authors**

32 Paper Citations

1300 Full Text Views

# Ejercicio



# Ejercicio - ¿Cómo calcular Bid y Ask?

## Reglas del mercado

*OA* is the outstanding ask, which is the lowest ask in the current market;
*OB* is the outstanding bid, which is the highest bid in the current market;
*Phl* is the high limit, which is the highest acceptable price in the market, which can be obtained from the historical data;
*Pll* is the low limit, which is the lowest acceptable price in the market, which can be obtained from the historic data;

# Ejercicio - ¿Cómo calcular Bid y Ask?

## Atributos de los agentes

*C* is the reservation price of the seller calculated from the utility function;
*D* is the reservation price of the buyer calculated from the utility function;
*Pb* is the basic price, which gives a starting profit to the trader to start from;
*Pt* is the target price, which gives a target profit to the trader to move towards;
$\gamma$ is a small positive number to adjust the size of the negotiation step.

# Ejercicio - Calculando Ask

On-Peak hour: $$OA = C + \lambda_1^1 \times (Phl - C); \lambda_1^1 \in rand[0.85,1] \quad (9)$$

Off-Peak hour: $$OA = C + \lambda_1^2 \times (Phl - C); \lambda_1^2 \in rand[0.5,0.85] \quad (10)$$

The seller needs to compute their basic price $Pb$ and target price $Pt$. According to the reservation price, the basic price will give the seller some profit instinctively. The current bid price from the buyer serves as the target price and the equations are described as follows:

$$Pb = C * \lambda_2, \lambda_2 \in rand[1,1.5] \quad (11)$$

$$Pt = OB_{current} \quad (12)$$

The basic price and the target price are the start profit and the destination for the seller, respectively. The seller calculates the ask price according to the time pressure and the eagerness value to maximize its profit. The size of the step derived from [21] is calculated as follows:

$$Step = \begin{cases} (Pt - Pb) * \gamma * T, if \quad Pt > Pb \\ (\max(Pt, C) - Pb) * \gamma * (1 - T), if \quad Pt < Pb \end{cases} \quad (13)$$

The ask price for the next round can be obtained from the following equation:

$$OA_{next} = Pb + Step \quad (14)$$

# Ejercicio - Calculando Ask

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

class Agente:
    __ID    = 0
    __gamma = 0
    __EA    = 0 # np.random.uniform(low=0.1, high=2.0)
    __Ask   = 0
    __Bid   = 0

    def computeAsk(self):
        return self.__Ask

    def computeBid(self):
        return self.__Bid
```

# Ejercicio - Calculando Ask

```
class Utility(Agente):

    def __init__(self,v1):
        self.__ID = 1
        self.__EA = np.random.uniform(low=0.1, high=2.0)
        self.__Ask = 0
        self.__lambda1 = np.random.uniform(low=0.85, high=1.0) # rand[0.85,1] - On-peak
        self.__lambda2 = np.random.uniform(low=0.75, high=1.25)
        self.__gamma   = np.random.uniform(low=0.1, high=0.25)
        self.__C   = v1
```

# Ejercicio - Calculando Ask

```
def computeAsk(self,OBid,Phl):
    # Reglas para calcular Ask
    Pb = 0
    Pt = 0
    Step = 0

    if self.__EA < 2.0:
        self.__EA += self.__EA * (self.__EA / 10)
    else:
        self.__EA = np.random.uniform(low=0.1, high=2.0)

if (OBid == 0):
    self.__Ask = self.__C + (self.__lambda1 * (Phl - self.__C))
else:
    Pb = self.__C * self.__lambda2
    Pt = OBid

    if (Pt > Pb):
        Step = (Pt -Pb) * self.__gamma * self.__EA
    else:
        Step = ( max(Pt, self.__C) - Pb) * (1 - self.__EA)

    self.__Ask = Pb + Step


return self.__Ask, Pb, Pt, Step
```

# Ejercicio - Calculando Bid

*2. Bidding strategy for the buyer*

Similar to the bidding strategy for the seller, the buyer submits its original bid price according to its reservation price and the price range. The equations (15) and (16) illustrate the original prices for on-peak hour and off-peak hour, respectively.

On-peak hour: $OB = D - \lambda_3 \times (D - Pll); \lambda_3 \in rand[0.5,0.85]$ (15)

Off-peak hour: $OB = D - \lambda_3 \times (D - Pll); \lambda_3 \in rand[0.85,1]$ (16)

The basic price, the target price and the size of step are described as follows [21]:

$$Pb = D * \lambda_4, \lambda_4 \in rand[0.5,1] \quad (17)$$

$$Pt = OA_{current} \quad (18)$$

$$Step = \begin{cases} (Pt - Pb) * \gamma * T, if \quad Pt \le Pb \\ (\min(Pt, D) - Pb) * \gamma * (1 - T), if \quad Pt > Pb \end{cases} \quad (19)$$

So the bid price for the next round can be calculated by (20):

$$OB_{next} = Pb + Step \quad (20)$$

# Ejercicio - Calculando Bid

```python
class Building(Agente):

    def __init__(self,v1):
        self.__ID = 2
        self.__EA      = np.random.uniform(low=0.1, high=1.0)
        self.__Bid     = 0
        self.__lambda3 = np.random.uniform(low=0.5, high=0.85) # rand[0.5,0.85] - On-peak
        self.__lambda4 = np.random.uniform(low=0.5, high=1.0)
        self.__gamma   = np.random.uniform(low=0.1, high=0.25)
        self.__D = v1
```

# Ejercicio - Calculando Bid

```
def computeBid(self,OAsk,Pll):
    # Reglas para calcular Bid
    Pb = 0
    Pt = 0
    Step = 0


    if self.__EA < 2.0:
        self.__EA += self.__EA * (self.__EA / 10)
    else:
        self.__EA = np.random.uniform(low=0.1, high=2.0)

if (OAsk == 0):
    self.__Bid = self.__D - (self.__lambda3 * (self.__D - Pll))
else:
    Pb = self.__D * self.__lambda4
    Pt = OAsk

    if (Pt <= Pb):
        Step =  ((Pt - Pb) * self.__gamma) * self.__EA
    else:
        Step = (min(Pt, self.__D) - Pb) * self.__gamma * (1 - self.__EA)

    self.__Bid = Pb + Step

return self.__Bid, Pb, Pt, Step
```

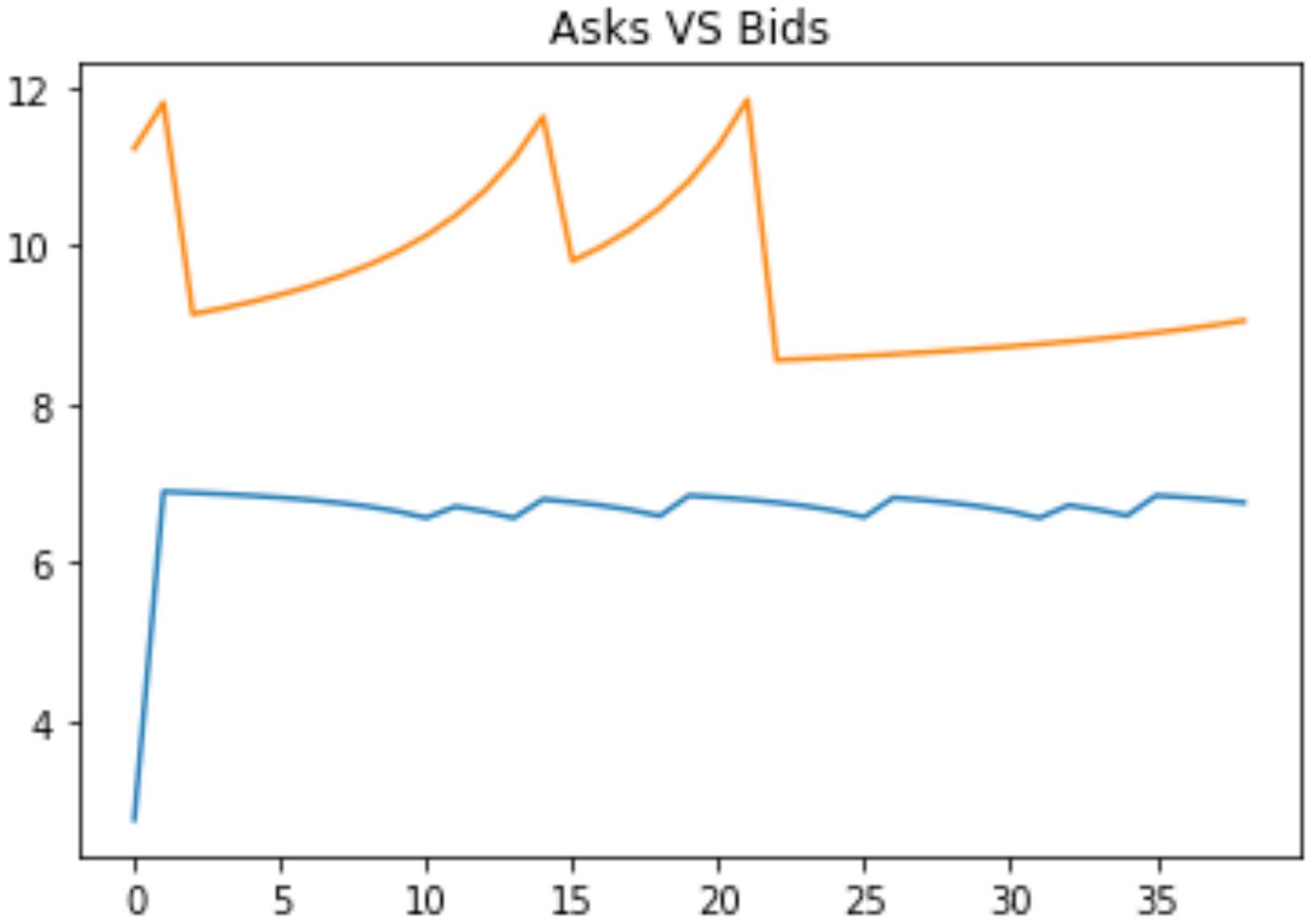
# Ejercicio - Simulando el mercado

```
#### Mercado ####
#### Valores estaticos
Phl = 10 #Phl is the high limit, which is the highest acceptable price in the market
Pll = 1 #Pll is the low limit, which is the lowest acceptable price in the market

### Valores dinámicos
OA = 0 #OA is the outstanding ask, which is the lowest ask in the current market;
OB = 0 #OB is the outstanding bid, which is the highest bid in the current market;

rondas = 40

a1 = Utility(8)
a2 = Building(8)

lAB = []
log = []

for rondaActual in range(1,rondas):#
    print("Ronda ", rondaActual)

    OB, PbB, PtB, StepB = a2.computeBid(OA,Pll)
    OA, PbU, PtU, StepU = a1.computeAsk(OB,Phl)

    log.append([OB, PbB, PtB, StepB,OA, PbU, PtU, StepU])
    lAB.append([OB,OA])
    if(OB >= OA):
        print("Matching done")
        break
```

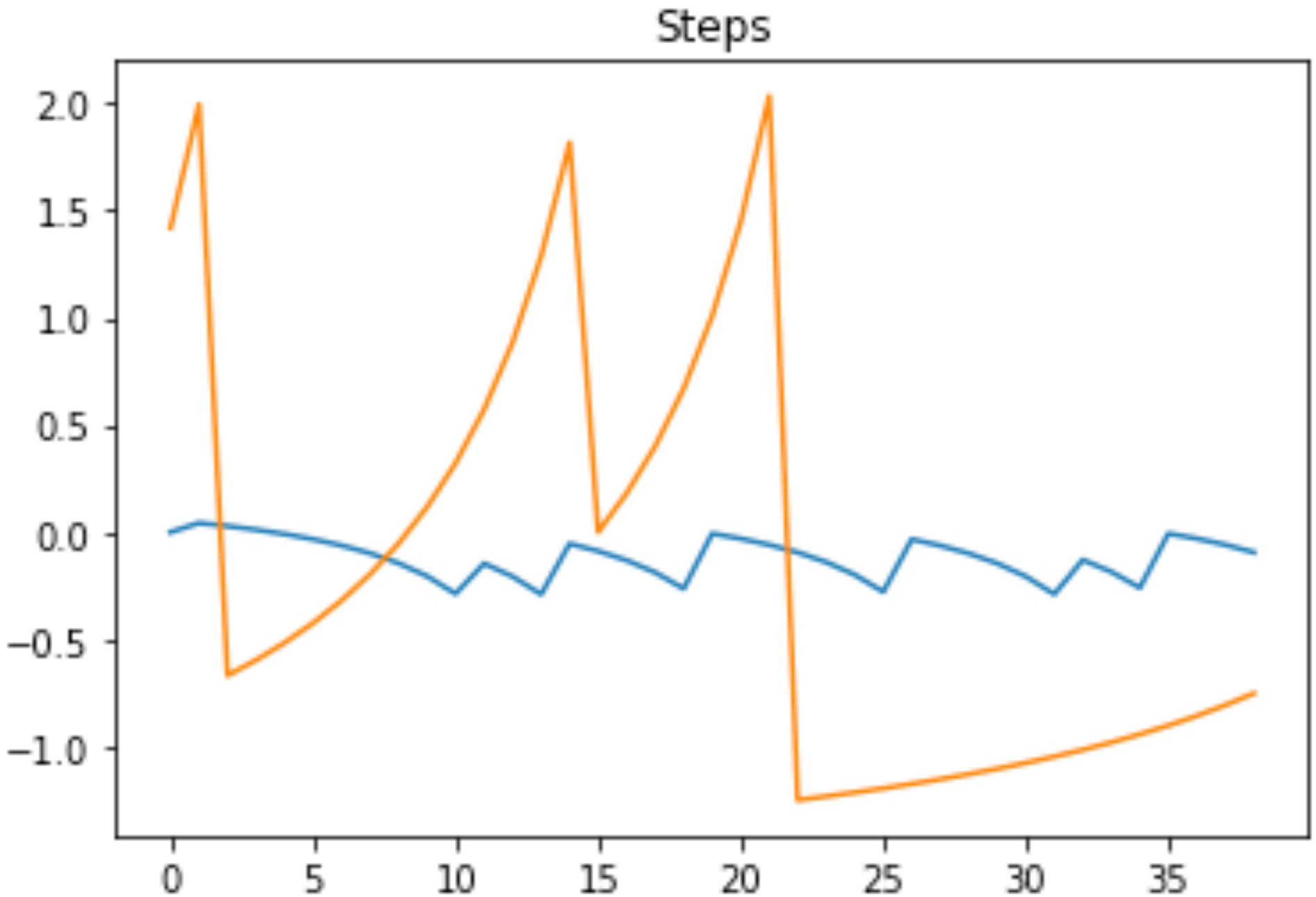
# Ejercicio - Simulando el mercado

```
arrayAB = np.asarray(lAB)
arrayLog = np.asarray(log)

plt.plot(arrayAB)
plt.title("Asks VS Bids")
plt.show()

plt.plot(arrayLog[:,3])
plt.plot(arrayLog[:,7])
plt.title("Steps")
plt.show()
```

# Ejercicio - Simulando el mercado



# Ejercicio - Simulando el mercado



# Ejercicio - Zero Intelligence

```
def computeBid(self,Phl):
    # Zero intelligence Bid
    self.__Bid = np.random.uniform(low=0.1, high=Phl)
    return self.__Bid, 0, 0, 0
```

# Ejercicio