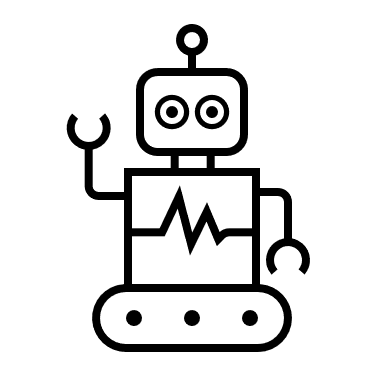
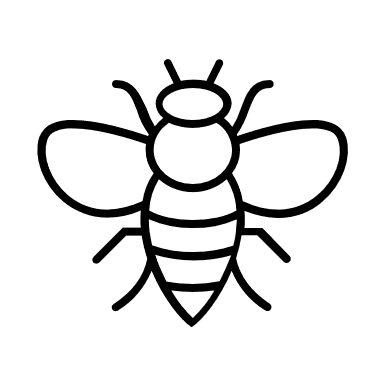
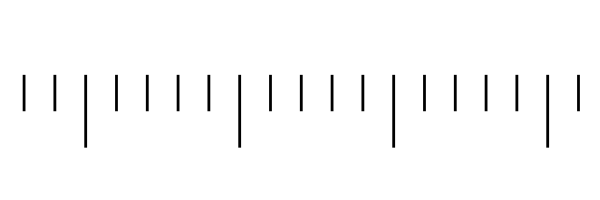
EAS-502 Project 1

Team Members:

1. Nabeel Khan 56
2. Vijaya Rana 106



Robot Object

d

The **experiment** **starts at t = 0 (d>0)**. Let ***T*** be the time at which the distance between Robot and object is zero (d=0). Experiment ends when the **robot and the object land on the same spot, or when they cross each other**.

We define:

# Strategy 1: At each unit of time, the robot will move 1 unit toward the object regardless of the object movement.

## Analytical Solution

For the initial distance d there are 3 cases which are mentioned below considering robot moves 1 unit towards object:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Case** | **Object movement** | **Robot movement** | **Distance** | **Probability** |
| 1 | Left | Right | d-2 | p |
| 2 | Right | Right | d | p |
| 3 | Stationary | Right | d-1 | 1-2p |

Where p = Probability of movement of Object

Breaking by using Total expectation theorem for above 3 cases:

= à Equation 1

For simplification considering: = *Ed* & =p

Simplifying further Equation 1

Ed = p(1+Ed) + (1-2p)(1+Ed-1) + p(1+Ed-2)

­­Rearranging for Ed

Ed = (p + (1-2p)(1+Ed-1) + p(1+Ed-2))/(1-p)

* This is *General term for the Strategy 1*

We calculate E[T|] using the *General term for the Strategy 1* and then we used this result to compute E[T|]. We’ll use MATLAB to calculate E[T|] for the relevant values of d.

E1 = 1/(1-p)

E2­ =(2-3p)/((1-p)^2)

Now using this we can calculate E3 to E19

## MATLAB Code for Strategy 1:

function [final\_expected\_vals] = strategy\_1()

% Function to calculate expected values of time for P\_om from .1 to .5

final\_expected\_vals = [];

% Using P\_om from .1 to .5

for p\_om = (1:5)/10

% Starting with the E1 and E2 calculated manually using general term

exp\_vals = [1/(1-p\_om), (2-3\*p\_om)/((1-p\_om)\*(1-p\_om))];

% Calculating E3 to E19 using general equation

for d = 3:19

E\_d = (p\_om + (1-2\*p\_om)\*(1+ exp\_vals(d-1)) + p\_om\*(1+ exp\_vals(d-2)))/(1-p\_om);

exp\_vals(d) = E\_d;

end

% Now calculating the expected value using conditional expectation

% Using pmf of d (uniform distribution with Pd = 0.1)

expected\_total = 0;

for d = 10:19

expected\_total = expected\_total + 0.1\*exp\_vals(d);

end

disp(expected\_total);

final\_expected\_vals(end + 1) = expected\_total;

end

end

## Analytical Result for Strategy 1:

|  |  |
| --- | --- |
| **Probability of Object Movement (POM** or **p)** | **Expected Value of Time (ET)** |
| 0.1 | 14.60 |
| 0.2 | 14.70 |
| 0.3 | 14.80 |
| 0.4 | 14.90 |
| 0.5 | 15.00 |

# Strategy 3: At each unit of time, the robot will move 1 unit toward the object if the object moves to either left or right, and the robot will stop if the object stops.

## Analytical Solution:

For the initial distance d there are 3 cases which are mentioned below considering robot moves 1 unit towards object:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Case** | **Object movement** | **Robot movement** | **Distance** | **Probability** |
| 1 | Left | Right | d-2 | p |
| 2 | Right | Right | d | p |
| 3 | Stationary | Stationary | d | 1-2p |

Where p = Probability of movement of Object

Breaking by using Total expectation theorem for above 3 cases:

=

For simplification considering = Ed& =p

Simplifying further:

Ed = p (1+ Ed-2) + p(1+Ed) + (1-2p) (1+Ed)

Now after solving for Ed, we get

Ed = (1+pEd-2)/p

* This is *General term for the Strategy 3*

We calculate E[T|] using the *General term for the Strategy 3* and then we used this result to compute E[T|]. We’ll use MATLAB to calculate E[T|] for the relevant values of d.

E1 = 1/p

E2 = 1/p

Now using this we can calculate E3 to E19

## MATLAB Code for Strategy 3:

function [final\_expected\_vals] = strategy\_3()

% Function to calculate expected values of time for P\_om from .1 to .5

final\_expected\_vals = [];

% Using P\_om from .1 to .5

for p\_om = (1:5)/10

% Starting with the E1 and E2 calculated manually

exp\_vals = [1/(p\_om), 1/(p\_om)];

% Calculating E3 to E19 using general term

for d = 3:19

E\_d = (1 + (p\_om)\*(exp\_vals(d-2)))/(p\_om);

exp\_vals(d) = E\_d;

end

% Now calculating the expected value using conditional expectation

% Using pmf of d (uniform distribution with Pd = 0.1)

expected\_total = 0;

for d = 10:19

expected\_total = expected\_total + 0.1\*exp\_vals(d);

end

disp(expected\_total);

final\_expected\_vals(end + 1) = expected\_total;

end

end

## Analytical Result for Strategy 3:

|  |  |
| --- | --- |
| **Probability of Object Movement (POM** or **p)** | **Expected Value of Time (ET)** |
| 0.10 | 75.00 |
| 0.20 | 37.50 |
| 0.30 | 25.00 |
| 0.40 | 18.75 |
| 0.50 | 15.00 |

# Strategy 2\*: At each unit of time, the robot will move 1 unit toward the object (with the probability of 1.5 or will stop (with the probability of 1-1.5) regardless of the object movement.

For the initial distance d there are 6 cases which are mentioned below considering robot moves 1 unit towards object:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Case** | **Robot movement** | **Fly movement** | **Distance** | **Probability** |
| 1 | Still | Left | d-1 | (1-q)p |
| 2 | Still | Right | d+1 | (1-q)p |
| 3 | Still | Still | d | (1-q)(1-2p) |
| 4 | Right | Left | d-2 | qp |
| 5 | Right | Right | d | qp |
| 6 | Right | Still | d-1 | q(1-2p) |

Where p = Probability of movement of Object, and q = Probability of movement of robot

=

­­Simplifying above equation considering = Ed

1+(p+q-3pq)+(3 pq-2p-q)+(pq) )/p(q-1)

Where q=1.5p, solving we get

* This is *General term for the Strategy 2*

Using *General term for the Strategy 2* we calculateE1, E2, E3 for respective values of p shown in table

|  |  |  |  |
| --- | --- | --- | --- |
|  | E1 | E2 | E3 |
| p=0.1 | 7.109830201 | 13.74703778 | 20.41566273 |
| p=0.2 | 3.832148359 | 7.090836764 | 10.4353403 |
| p=0.3 | 2.796251249 | 4.870194275 | 7.130718951 |
| p=0.4 | 2.349754291 | 3.736455736 | 5.51786687 |
| p=0.5 | 2.194335081 | 2.971675407 | 4.664041954 |

## MATLAB Code for Strategy 2:

function [final\_expected\_vals] = strategy\_2()

% Function to calculate expected values of time for P\_om from .1 to .5

initial\_E­ = [7.109830200957157 13.747037779905098 20.415662725586326

3.8321483594592816 7.090836763705904 10.435340300715005

2.7962512487710653 4.870194275499013 7.130718950732399

2.3497542908541056 3.736455736129949 5.5178668695576

2.1943350814194544 2.9716754070972726 4.664041954066908];

final\_expected\_vals = [];

% Using P\_om from .1 to .5

for iter\_count = (1:5)

% Using P\_om from .1 to .5

p\_om = iter\_count/10;

% Starting with the E1, E2, and E3 calculated manually

E = initial\_E(iter\_count,:);

% Calculating E4 to E19 using general equation

for d = 4:19

E(d)=(1+(2.5\*p\_om-4.5\*p\_om^2)\*E(d-2)+((4.5\*p\_om^2)-3.5\*p\_om)\*E(d-1) +(1.5\*(p\_om\*^2)\*E(d-3)))/(p\_om\*(1.5\*p\_om - 1));

end

% Now calculating the expected value using conditional expectation

% Using pmf of d (uniform distribution with Pd = 0.1)

expected\_total = 0;

for d = 10:19

expected\_total = expected\_total + 0.1\*E(d);

end

disp(expected\_total);

final\_expected\_vals(end + 1) = expected\_total;

end

end

## Analytical Result for Strategy 2:

|  |  |
| --- | --- |
| **Probability of Object Movement (POM** or **p)** | **Expected Value of Time (ET)** |
| 0.10 | 97.082 |
| 0.20 | 48.767 |
| 0.30 | 32.678 |
| 0.40 | 24.651 |
| 0.50 | 19.856 |

# Conclusion

We used simulation for 10000 iterations to **arrive at the same results** as concluded in the table below

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Strategy** | **P\_om** | **Analytical Expected Value** | **Simulation** | | |
| **Iterations** | **Total Time** | **Simulated Expected Value** |
| Stg\_1 | 0.10 | 14.600 | 10000 | 146042 | 14.6042 |
| Stg\_1 | 0.20 | 14.700 | 10000 | 146760 | 14.676 |
| Stg\_1 | 0.30 | 14.800 | 10000 | 147276 | 14.7276 |
| Stg\_1 | 0.40 | 14.900 | 10000 | 149199 | 14.9199 |
| Stg\_1 | 0.50 | 15.000 | 10000 | 149503 | 14.9503 |
| Stg\_2 | 0.10 | 97.082 | 10000 | 967419 | 96.7419 |
| Stg\_2 | 0.20 | 48.767 | 10000 | 489432 | 48.9432 |
| Stg\_2 | 0.30 | 32.678 | 10000 | 327241 | 32.7241 |
| Stg\_2 | 0.40 | 24.651 | 10000 | 245996 | 24.5996 |
| Stg\_2 | 0.50 | 19.856 | 10000 | 198819 | 19.8819 |
| Stg\_3 | 0.10 | 75.000 | 10000 | 749603 | 74.9603 |
| Stg\_3 | 0.20 | 37.500 | 10000 | 375690 | 37.569 |
| Stg\_3 | 0.30 | 25.000 | 10000 | 251017 | 25.1017 |
| Stg\_3 | 0.40 | 18.750 | 10000 | 188280 | 18.828 |
| Stg\_3 | 0.50 | 15.000 | 10000 | 150856 | 15.0856 |

## Python Code for simulation

import numpy as np

import random

import pandas as pd

table\_results = []

for p\_om in [.1, .2, .3, .4, .5]:

robot\_position = 0

obj\_position = np.random.choice(range(10,20))

robot\_mvmts = np.array([0,1])

obj\_mvmts = np.array([-1,0,1])

obj\_mvmt\_probs = np.array([p\_om, 1-2\*p\_om, p\_om])

robot\_mvmt\_probs = np.array([1-1.5\*p\_om, 1.5\*p\_om])

# strategy 1

stg\_1\_times = []

for sim\_count in range(10000):

time\_count = 0

while robot\_position < obj\_position:

robot\_position += 1

obj\_position += np.random.choice(obj\_mvmts, 1,p=obj\_mvmt\_probs)[0]

time\_count += 1

robot\_position = 0

obj\_position = np.random.choice(range(10,20))

stg\_1\_times.append(time\_count)

print(sum(stg\_1\_times)/len(stg\_1\_times))

table\_results.append(['Stg\_1', p\_om,sum(stg\_1\_times) ])

# strategy 2

stg\_2\_times = []

for sim\_count in range(10000):

time\_count = 0

while robot\_position < obj\_position:

robot\_position += np.random.choice(robot\_mvmts, 1,p=robot\_mvmt\_probs)[0]

obj\_position += np.random.choice(obj\_mvmts, 1,p=[p\_om, 1-2\*p\_om, p\_om])[0]

time\_count += 1

robot\_position = 0

obj\_position = np.random.choice(range(10,20))

stg\_2\_times.append(time\_count)

print(sum(stg\_2\_times)/len(stg\_2\_times))

table\_results.append(['Stg\_2', p\_om,sum(stg\_2\_times) ])

# strategy 3

stg\_3\_times = []

for sim\_count in range(10000):

time\_count = 0

while robot\_position < obj\_position:

obj\_movement = np.random.choice([-1,0,1], 1,p=[p\_om, 1-2\*p\_om, p\_om])[0]

obj\_position += obj\_movement

robot\_position += abs(obj\_movement)

time\_count += 1

robot\_position = 0

obj\_position = np.random.choice(range(10,20))

stg\_3\_times.append(time\_count)

print(sum(stg\_3\_times)/len(stg\_3\_times))

table\_results.append(['Stg\_3', p\_om,sum(stg\_3\_times) ])