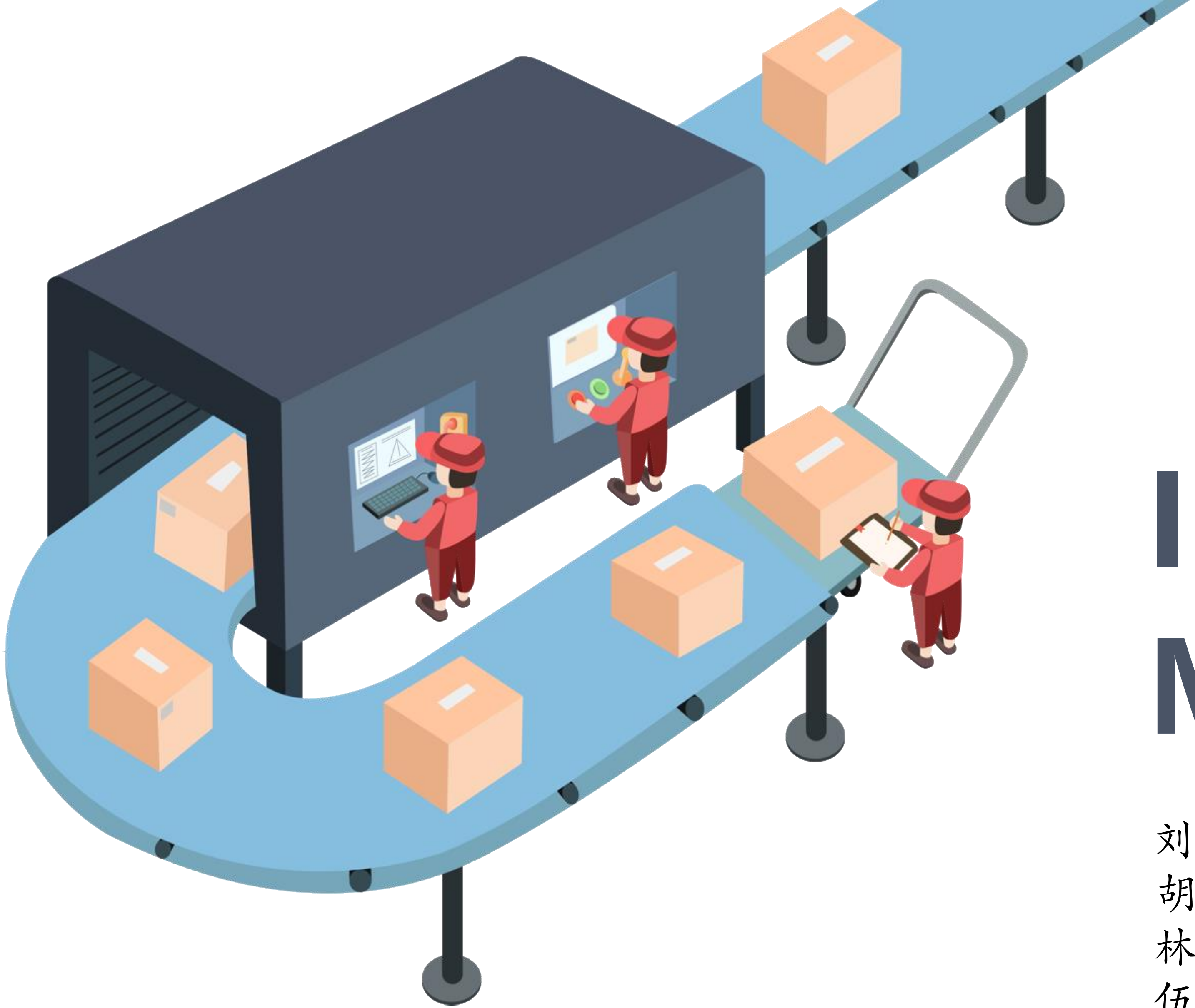
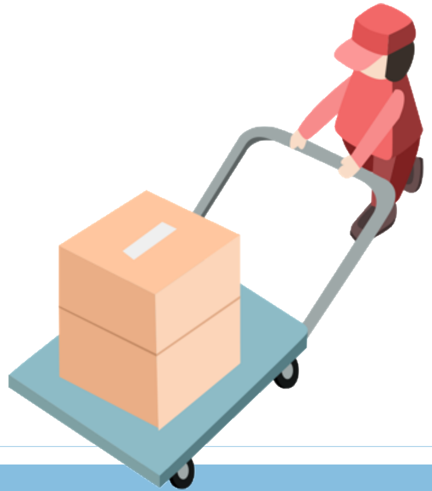
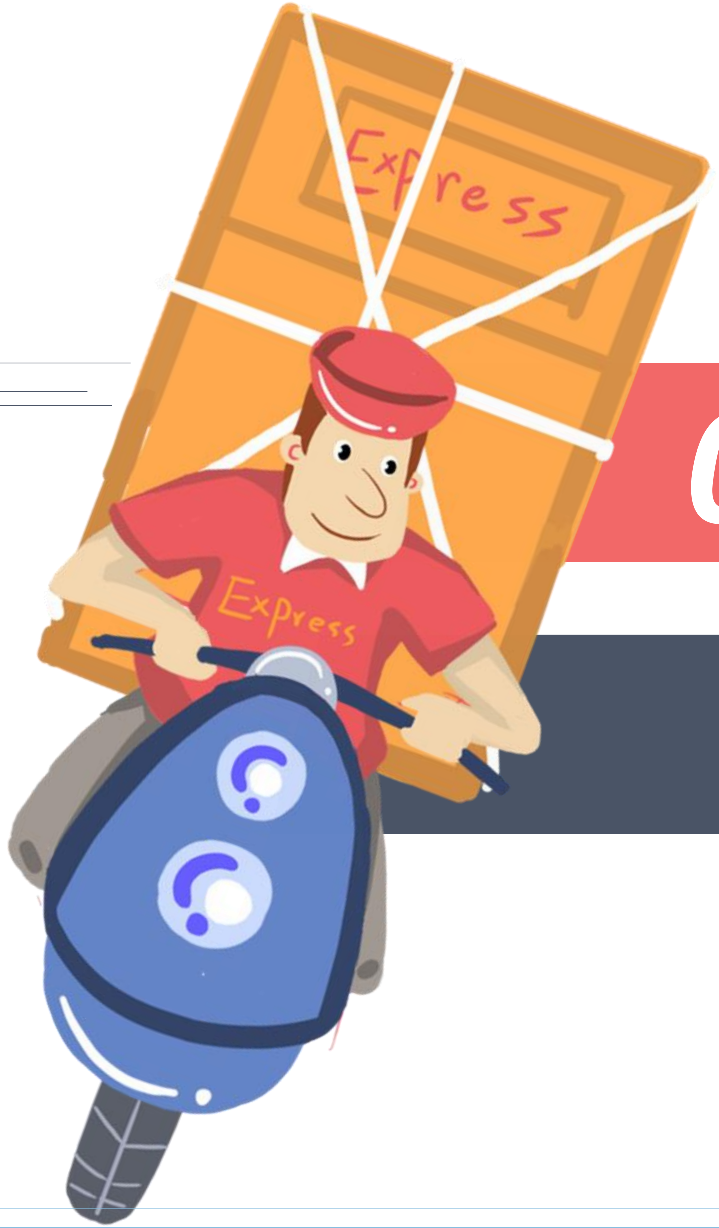


INVENTORY MODEL

刘凯阳 1830004016
胡慧妍 1830005009
林斯弘 1830005020
伍小雨 1830005033
钟睿婕 1830005056



CONTENTS



01

Part A

Problem 1 Statement

Inventory

- ✓ Customer Appearance: Poisson Process with rate 8
- ✓ Demanding: Discrete Random Variable (D=1,2,3,4)
- ✓ If inventory level < lower limit → Replenishment

Delivery:2h

Target:

- ✓ Trace of model

  (Graphs [10 Days])
  - ✓ Event lists
  - ✓ Demanding
  - ✓ Inventory level
  - ✓ Revenue
  - ✓ Profit
- ✓ Whether make profit in a month? & Average Net Profit

Nomenclatures

Time & Counter

L: Delivery time

t :Current time

T :Limit time

d :Counter of delivery

i :Counter of events

Event_type: Type of events

Quantity

s :Minimum inventory level

S :Maximum inventory level

y :Quantity of goods ordered but not arrived

w :Sales volume

Cost & Loss

C :Cost price c :Cost price/unit

H :Storing cost h :Store price/unit

Dv :Delivery cost

Loss :What should have been earned is not earned

Events list

t_0 :The arrival time of one customer

t_1 :The arrival time of product

Target

D :Demanding/ customer

x :Current inventory level

R :Revenue r :sell price/unit

Profit :Net Profit

Algorithm

Assumption

The depot start to replenish once the store opens.

Inventory Model 1

t0<t1 — People come before products

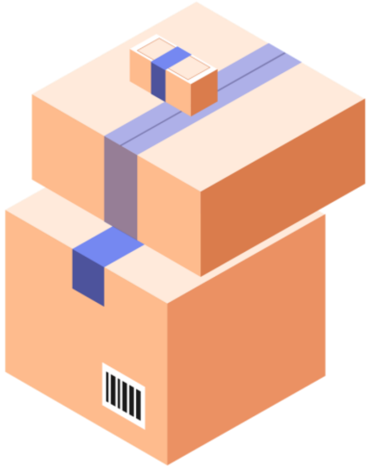
- Update storing cost (H) &current time (t=t0)
- Generate Demanding (D) (Discrete Distribution)
- Check whether any stock
  - Compare Demanding & Inventory level
    - D<x — *Buy it! Deduct x*
    - D>x — *Pity! Loss it*
- Update Revenue (R)
- Check whether needs replenishment
  - *Rules: x< min level & no order (y)*
- Generate next customer's arrival

t0>t1 — People come after products

- Update storing cost (H) &current time (t=t1)
- Update order amount/cost (C) & Inventory level (X)
- No goods on the way (y=0 & t1=∞)
- No Demanding (D)

Update Profit per Event

Matlab Code



1st . Set initial values

```
%Time:
T=30;                        %limit time
t=[];t(1)=0;t(2)=0;          %current time
L=2/24;                      %Delivery time
t0=-(1/8)*log(rand(1));      %The arrival time of one customer(initial value)
t1=t0+2/24;                  %The arrival time of product(initial value)
```

```
%Cost series
H=[];H(1)=0;                 %Storing cost
C=[];C(1)=0;                 %Cost price
Loss=[];                     %Money that should have been earned is not earned
R=[];R(1)=0;                 %Revenue
Profit=[];Profit(1)=0;       %Net Profit
```

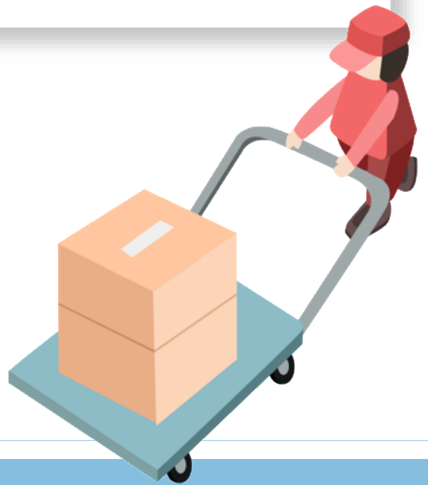
```
%Quantity:
x=[];x(1)=0;x(2)=0;          %Current inventory level
D=[];D(1)=0;                 %Demanding/ customer
s=10;S=120;                  %Minimum & Maximum inventory level
y=0;                         %Quantity of goods ordered but not arrived
```

```
%Counters
i=2;                         %Counter of event
d=0;                         %Counter of delivery
```

```
%Cost price
r=12;                        %sell price/unit
h=0.5;                       %store price/unit
c=5;                         %cost price/unit
Dv=10;                       %Delivery cost/ride
```

Assumption

The store start to replenish once the first customer comes.



Matlab Code

2nd. Build the structure

```
while t(i)<=T
```

```
if t0<t1 %Customer arrives earlier than product
```

```
t(i+1)=t0;
```

Ask for product

```
u1=rand(1);
if u1<0.7
    D(i)=1;
else if u1<0.9
    D(i)=2;
    else if u1<0.98
            D(i)=3;
        else D(i)=4;
        end
    end
end
```

Sell product

```
if D(i)<x(i)
    w=D(i);
    x(i+1)=x(i)-w;
    Loss(i)=0;
else
    w=x(i);
    Loss(i)=(D(i)-x(i))*2;
    x(i+1)=0;
end
```

Supplement

```
if x(i+1)<s&&y==0
    y=S-x(i+1);
    t1=t(i+1)+L;
    d=d+1;
end
```

```
t0=-(1/8)*log(u)+t(i+1);
```

```
else %if t0>t1:Product arrives earlier than customer
```

```
t(i+1)=t1;
```

```
x(i+1)=x(i)+y;
y=0;
```

```
t1=Inf;
```

Matlab Code

```
if t0<t1 %Customer arrives earlier than product
```

```
t(i+1)=t0;
```

```
u1=rand(1);
if u1<0.7
    D(i)=1;
else if u1<0.9
    D(i)=2;
    else if u1<0.98
            D(i)=3;
        else D(i)=4;
        end
    end
end
```

- **Generate demanding:**
  discrete random variable

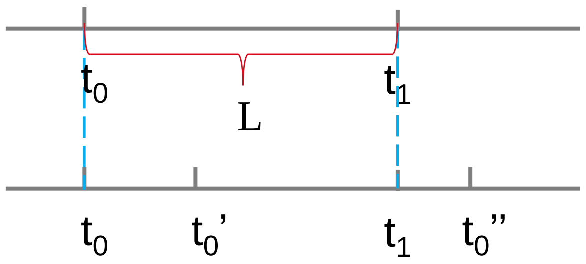
N	1	2	3	4
p	0.7	0.2	0.08	0.02

```
if D(i)<x(i)
    w=D(i);
    x(i+1)=x(i)-w;
    Loss(i)=0;
else
    w=x(i);
    Loss(i)=(D(i)-x(i))*2;
    x(i+1)=0;
end
```

- **Compare demanding with current inventory:**
  update state of inventory
  calculate the loss

Timeline:

```
t0=-(1/8)*log(rand(1));
t1=t0+2/24;
```




```
if x(i+1)<s&&y==0
    y=S-x(i+1);
    t1=t(i+1)+L;
    d=d+1;
end
```

Check if it is time to supplement:
define ordered quantity ***y***
define the arrival time of order t_1
count deliver frequency ***d***

```
u=rand(1);
t0=-(1/8)*log(u)+t(i+1);
```

- **Generate new t_0**

Matlab Code

```
else %if t0>t1:Product arrives earlier than customer
```

```
t(i+1)=t1;
```

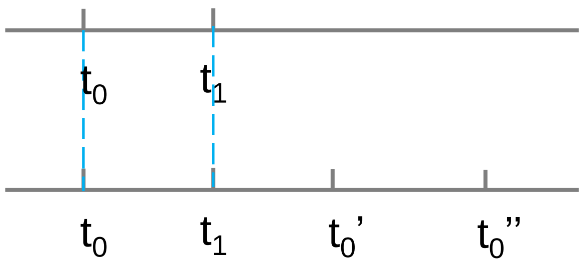
```
x(i+1)=x(i)+y;
y=0;
```

```
t1=Inf;
```

- Already supplement the products
- No waiting order

- Ensure that next time is t0<t1

Timeline:



Matlab Code

3rd. Calculate the money

```
if t0<t1 %Customer arrives earlier than product
```

```
H(i)=H(i-1)+(t0-t(i))*x(i)*h;
```

```
t(i+1)=t0;
```

```
u1=rand(1);
if u1<0.7
    D(i)=1;
else if u1<0.9
    D(i)=2;
    else if u1<0.98
            D(i)=3;
        else D(i)=4;
        end
    end
end
```

```
if D(i)<x(i)
    w=D(i);
    x(i+1)=x(i)-w;
    Loss(i)=0;
else
    w=x(i);
    Loss(i)=(D(i)-x(i))*2;
    x(i+1)=0;
end
```

```
R(i)=R(i-1)+w*r-Loss(i);
```

```
if x(i+1)<s&&y==0
    y=S-x(i+1);
    t1=t(i+1)+L;
    d=d+1;
end
```

```
C(i)=C(i-1);
```

```
t0=-(1/8)*log(u)+t(i+1);
```

```
else %if t0>t1:Product arrives earlier than customer
```

```
t(i+1)=t1;
```

```
x(i+1)=x(i)+y;
y=0;
```

```
H(i)=H(i-1)+(t1-t(i))*x(i-1)*h;
R(i)=R(i-1);
C(i)=C(i-1)+c*y;
```

```
t1=Inf;
```

```
Profit(i)=R(i)-C(i)-H(i)-d*Dv;
```

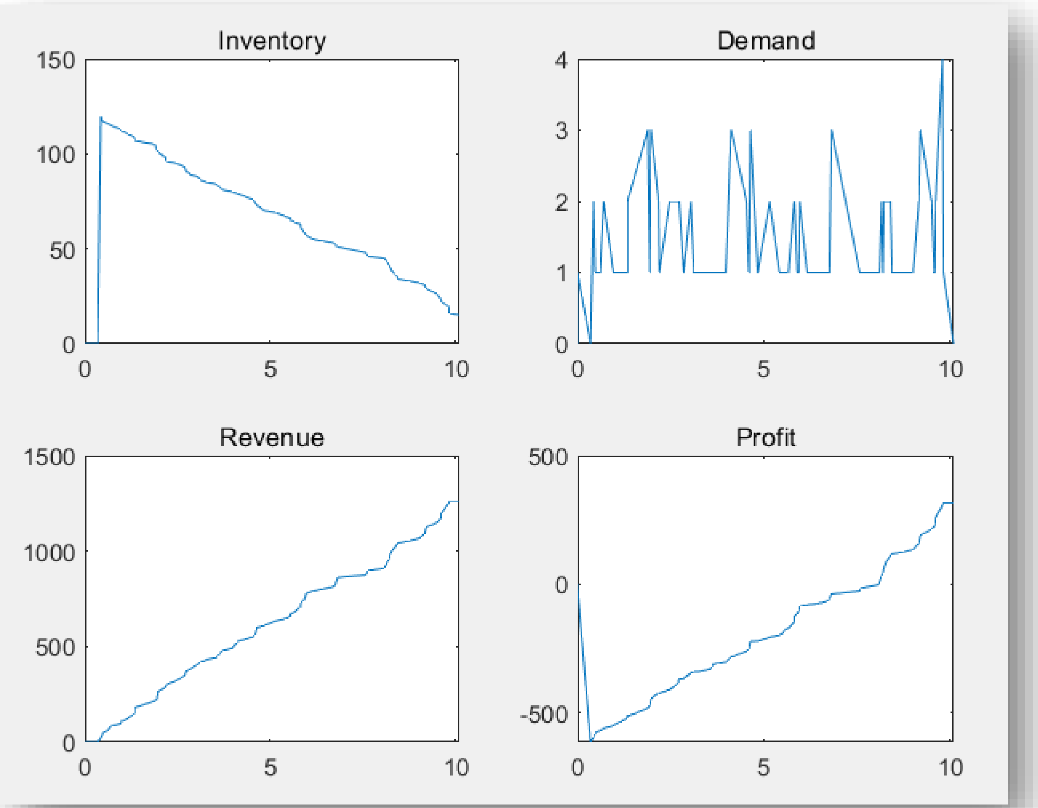
02

Result of Part A

Matlab Code

Output

10 days:



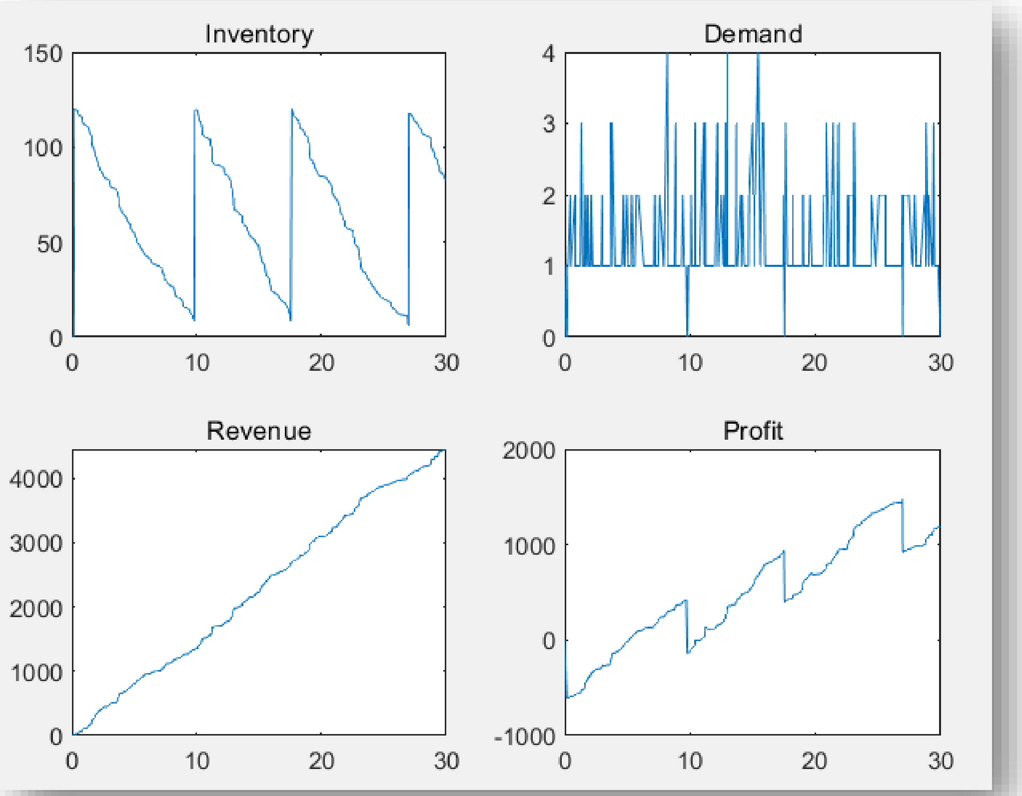
	1 Time	2 Demand	3 Inventory_Level	4 Revenue	5 Profit	6 Loss	7 Event_type
1	0	0	0	0	0	0	t0
2	0	1	0	-2	-12	2	t0
3	0.0098	0	0	-2	-612	0	t1
4	0.0931	1	120	10	-602.2688	0	t0
5	0.1309	2	119	34	-580.3434	0	t0
6	0.1658	3	117	70	-551.3949	0	t0
7	0.2863	2	114	94	-539.1152	0	t0
8	0.4920	1	112	106	-536.5940	0	t0
9	0.6612	1	111	118	-564.2773	0	t0
10	1.3762	1	110	130	-559.5936	0	t0
11	1.5093	1	109	142	-548.0720	0	t0
12	1.5180	2	108	166	-533.1491	0	t0
13	1.6861	1	106	178	-548.0477	0	t0
14	2.1937	1	105	190	-538.2322	0	t0
15	2.2353	1	104	202	-528.3945	0	t0
16	2.2768	1	103	214	-522.0897	0	t0
17	2.3874	1	102	226	-522.1154	0	t0
18	2.6232	1	101	238	-512.9977	0	t0
19	2.6803	1	100	250	-503.9436	0	t0
20	2.7392	1	99	262	-497.6589	0	t0
21	2.8547	1	98	274	-487.0382	0	t0
22	2.8828	3	97	310	-456.8662	0	t0
23	3.0030	1	94	322	-457.1360	0	t0
24	3.2641	1	93	334	-447.2962	0	t0
25	3.3105	1	92	346	-435.3160	0	t0

```
Net Profit = 244.635283
Average Net Profit = 24.703882
```

Matlab Code

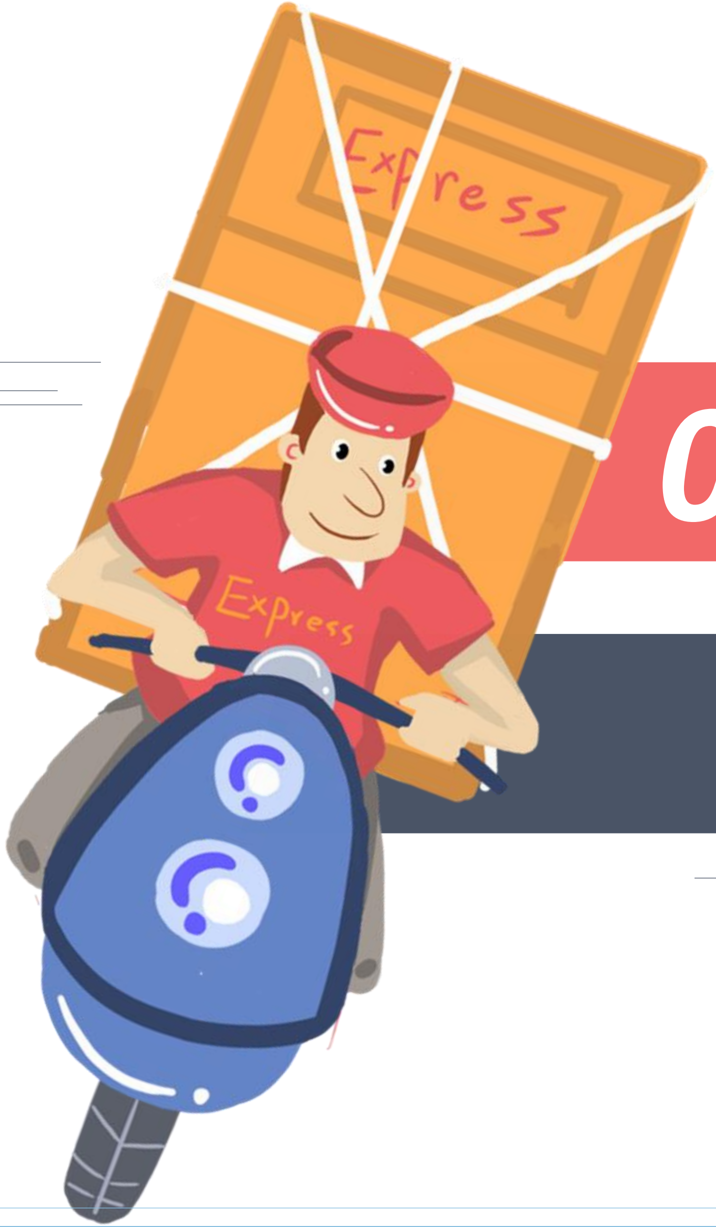
Output

30 days:



	1 Time	2 Demand	3 Inventory_Level	4 Revenue	5 Profit	6 Loss	7 Event_type
1	0	0	0	0	0	0	t0
2	0	1	0	-2	-12	2	t0
3	0.1818	0	0	-2	-612	0	t1
4	0.2651	1	120	10	-609.5232	0	t0
5	0.4238	1	119	22	-604.2679	0	t0
6	0.5372	1	118	34	-599.7146	0	t0
7	0.6634	1	117	46	-591.3295	0	t0
8	0.7252	1	116	58	-583.2704	0	t0
9	0.7931	2	115	82	-561.3718	0	t0
10	0.8297	1	113	94	-564.3973	0	t0
11	1.0956	1	112	106	-565.8978	0	t0
12	1.3367	1	111	118	-558.5194	0	t0
13	1.4200	1	110	130	-548.7276	0	t0
14	1.4601	2	109	154	-536.5920	0	t0
15	1.6778	1	107	166	-525.8388	0	t0
16	1.7011	1	106	178	-518.5983	0	t0
17	1.7909	3	105	214	-493.3095	0	t0
18	1.9949	3	102	250	-465.3618	0	t0
19	2.1528	1	99	262	-455.7193	0	t0
20	2.2005	2	98	286	-435.3952	0	t0
21	2.2755	1	96	298	-432.1305	0	t0
22	2.4575	1	95	310	-426.3060	0	t0
23	2.5875	3	94	346	-395.9776	0	t0
24	2.7081	1	91	358	-386.7507	0	t0
25	2.7691	1	90	370	-379.2459	0	t0
26	2.8690	2	89	394	-360.2402	0	t0

```
Net Profit = 1267.126943
Average Net Profit = 42.359962
```

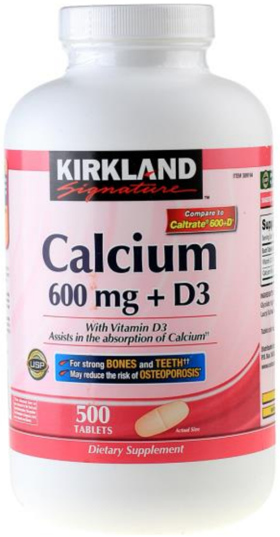


03

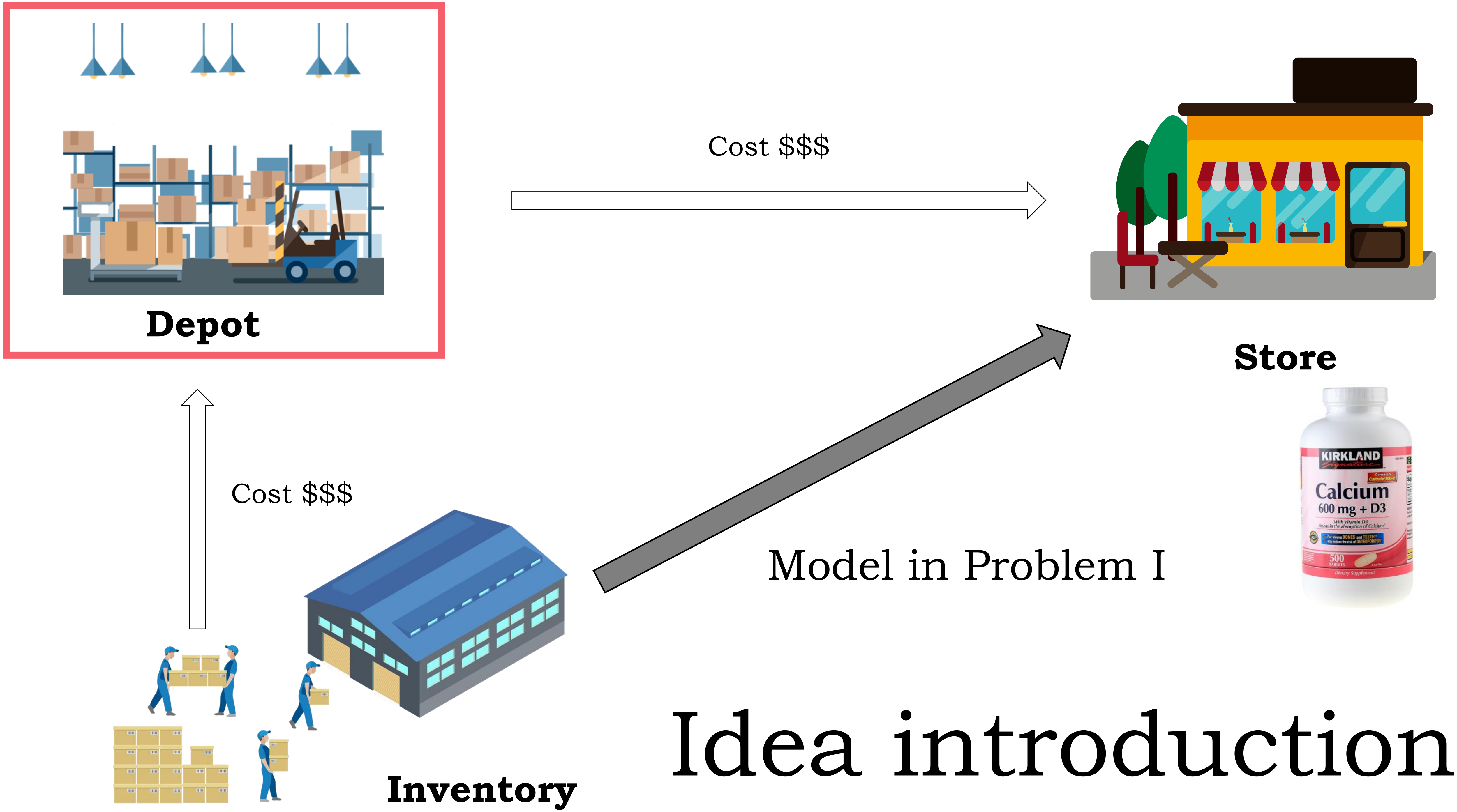
Part B

A more complicated inventory system

Calcium tablet



Idea introduction



Idea Explanation

Depot

Ordering policy

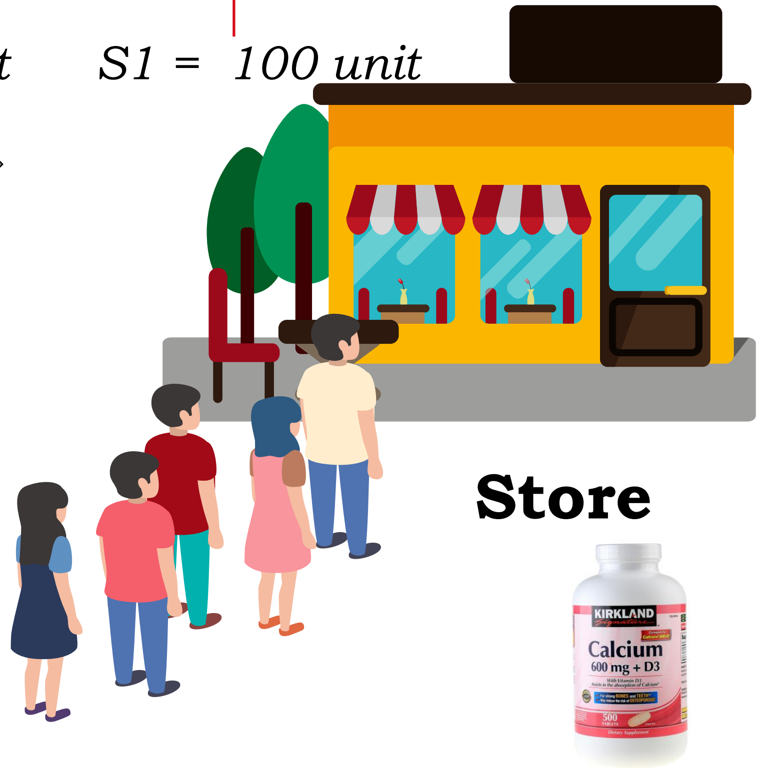
$s2=50$ unit $S2=300$ unit

Ordering policy

$s1 = 20$ unit $S1 = 100$ unit

frequently orders, small amount

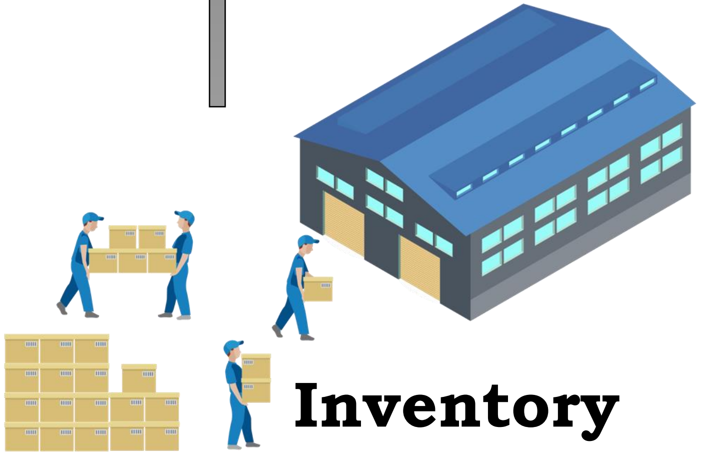
Delivery cost K1=5 ¥
Delivery time L1=0.1 Day



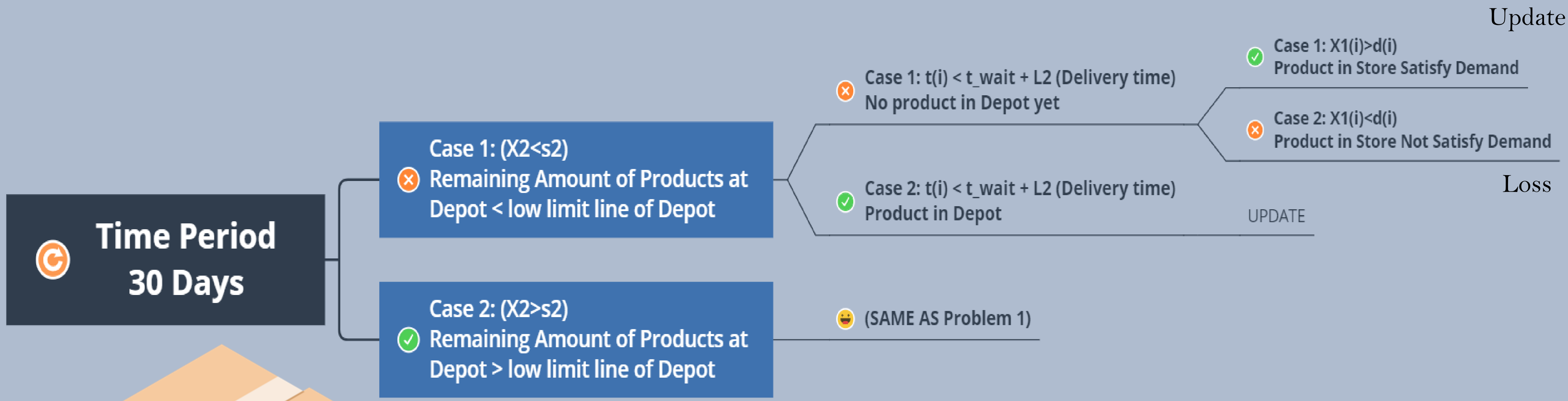
Store

Delivery cost K2=5 ¥
Delivery time L2=2 Day
Holding cost per item per unit time :h2=0.5 ¥
Not frequently orders, large amount

Demand: D
Sell price per unit : r = 200 ¥
Cost price per unit : c=50 ¥
Loss due to shortage : L (D, x)
Holding cost per item per unit time :h1=2 ¥



Inventory



Assumption

The store can supplement their storage until the depot have sufficient products

Details Explanation

MATLAB Code – Conditions Clarification

```
x1=0; s1=20; S1=100; r=200; L1=0.1; K1=5; h1=2; T=30;x2=0; s2=50; S2=300; L2=2; K2=5;
h2=0.5; c=50;
X1 = [x1]; % Remaining Amount of Products in Store
X2 = [x2]; % Remaining Amount of Products at Depot
d = [0]; % Demand of Each Customer
R = [0]; % Profit
t = [];% current time
t(1)=0;
i=1;
Loss=[];
cost=0;
y = 0; % the quantity of products have been ordered but not yet delivered
j=0;
t0 = -(1/8)*log(rand(1));
t1=t0+L1;
```

While t < T

- Case 1: X2 < s2 (Depot Lacks Products)

- Case 2: X2 > s2 (Depot doesn’t Lack Products)

Case 1: X2 < s2 (Depot Lacks Products)

```
• %Record the time
• t_wait = t(i);

• %Case 1.1: If there is a demand from customers when the products
  heading for depot are still on the go
• if t(i) < t_wait+L2
• %Update the current time and storage at depot
•             t(i+1) = t0;
•             X2(i+1)=X2(i);
• %Customers Demand
•             d(i) = randsample(4,1,true,[.7,.2,.08,.02]);
• %Start considering the storage in the store
```

Case 1.1: Customer Purchases when Products Heading for Depot are Still on the Go

```
%Check if the store has enough storage for customers
if X1(i)>d(i) %If yes
%Update the storage and the profit, and no need to consider the loss
        X1(i+1) = X1(i) - d(i);
        R(i+1) = R(i) + d(i) * r - (X1(i) * h1 - X2(i) * h2)*(t(i+1)-
t0);
else %If no
%Update the storage and the profit, and consider the loss
        X1(i+1)=0;
        Loss(i) = (d(i)-X1(i))*2;
        R(i+1) = R(i) + X1(i) * r - (X1(i) * h1 - X2(i) * h2)*(t(i+1)-
t0)-Loss(i);
end
```

Case 1.2: Products Arrive at Depot

```
• %Case 1.2: If the products arrive at depot
• else
• %Update current time
•             t(i+1)=t(i);
•             t1=t(i)+L1;
• %Update the storage at depot and in the store
•             X2(i+1) = S2;
•             X1(i+1)=X1(i);
• %Update the profit
•             R(i+1) = R(i)-K2-c*(S2-X2(i));
• end
```

Case 2: X2 > s2 (Depot doesn’t Lack Products)

Case 2.1: Customer arrives earlier than products

1 Check demand and inventory

2 Check if store needs replenishment

```
if X2(i)> s2 % Depot has enough inventory to supply the store
      if t0<t1
         t(i+1)=t0; %update current time
         d(i) = randsample(4,1,true,[.7,.2,.08,.02]); %demand of customer
         X2(i+1)=X2(i);
   1     if d(i)<X1(i) %inventory level of store > customer demand, no loss
             w=d(i); %number of products sold to the customer
             X1(i+1)=X1(i)-w; %update inventory
             Loss(i)=0;
         else     %store can’t provide the demanded products, Loss
             w=X1(i);
             Loss(i)=(d(i)-X1(i))*2;
             X1(i+1)=0;
         end
         R(i+1) = R(i) + w * r - (X1(i) * h1 - X2(i) * h2)*(t(i+1)-t(i)); %profit

   2     if X1(i)<s1 && y==0
         y=S1-X1(i); %the quantity of products need to replenish
         X2(i+1) = X2(i)-y; %update depot’s inventory level
         t1=t(i+1)+L1; %arrival time of the replenished products
         end
```

Case 2: X2 > s2 (Depot doesn't Lack Products)

Case 2.2: Products arrive earlier than customer

1 Check if store needs replenishment

2 Update inventory and profit

```
else %if products arrive earlier than customers(t0>t1)
     1  if X1(i)<s1 && y==0
            y=S1-X1(i); %the quantity of products need to replenish
            X2(i+1) = X2(i)-y; %update depot's inventory level
        else
            X2(i+1)=X2(i);
        end
        t(i+1)=t0+L1;
        R(i+1)=R(i);
     2  X1(i+1)=S1;
        cost = c*y;
        y=0;
        t1=Inf;
        d(i)=0;
        R(i+1) = R(i) - (X1(i) * h1 - X2(i) * h2)*(t(i+1)-t(i))-K1-cost;
end
```

04

Result of Part B

A more complicated inventory system

Calcium tablet

Output

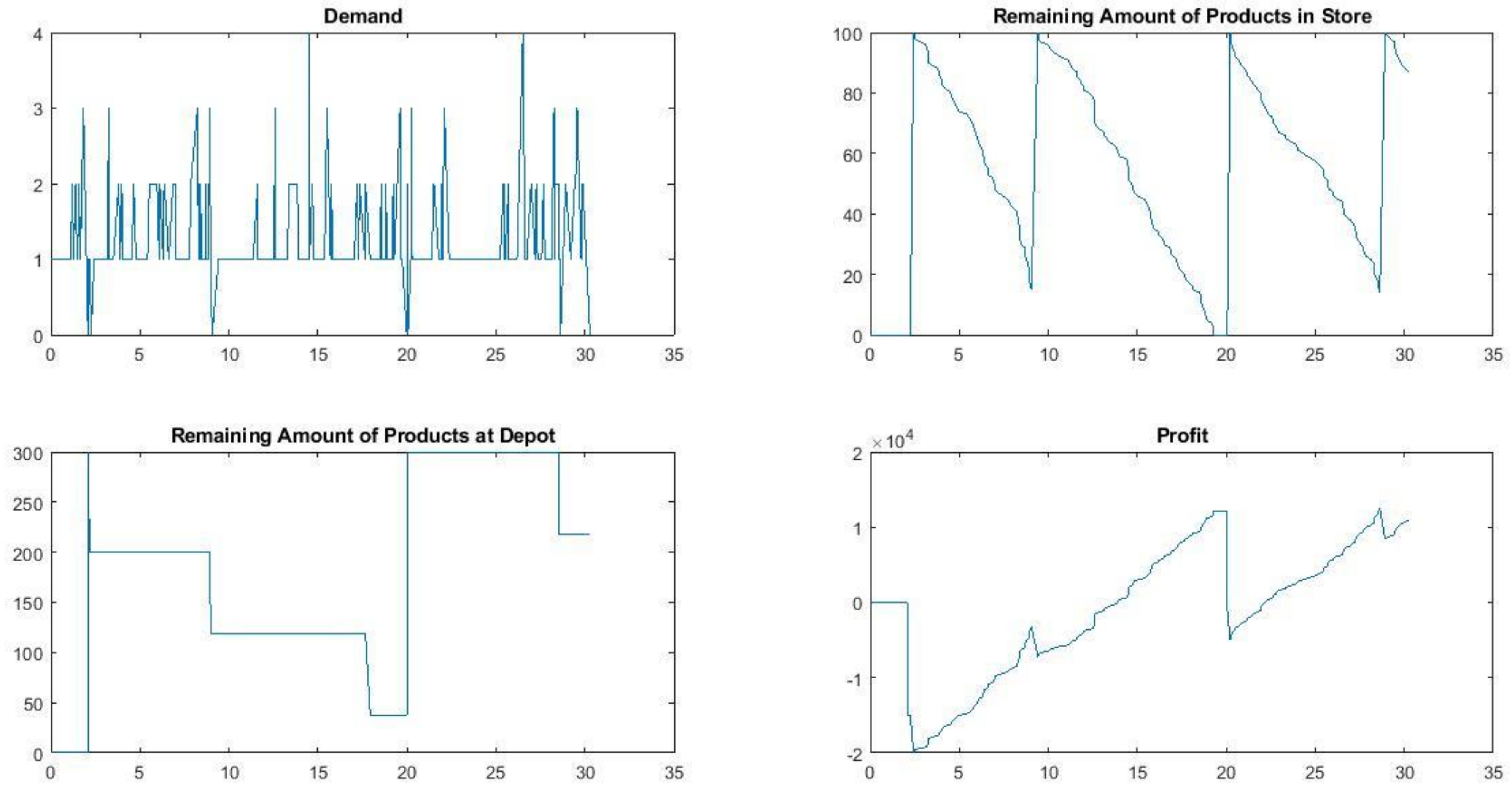
T =

229×7 table

Time	Demand	Event_type	Inventory_Store	Inventory_Depot	Profit	Loss
0	1	t0	0	0	0	2
0.01072	1	t0	0	0	-2	2
0.32687	1	t0	0	0	-4	2
0.34989	2	t0	0	0	-6	4
0.38097	1	t0	0	0	-10	2
0.44672	1	t0	0	0	-12	2
0.56321	1	t0	0	0	-14	2
0.67075	1	t0	0	0	-16	2
0.7582	1	t0	0	0	-18	2
0.82961	2	t0	0	0	-20	4
0.84669	3	t0	0	0	-24	6
0.98218	4	t0	0	0	-30	8
0.98797	1	t0	0	0	-38	2
1.2188	1	t0	0	0	-40	2
1.393	2	t0	0	0	-42	4
1.5749	1	t0	0	0	-46	2

Plots for Demand, Storage, and Profit

30 Days



05

Conclusion

Conclusion

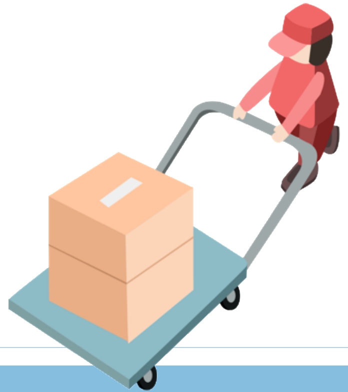
Comparison of the results of both questions

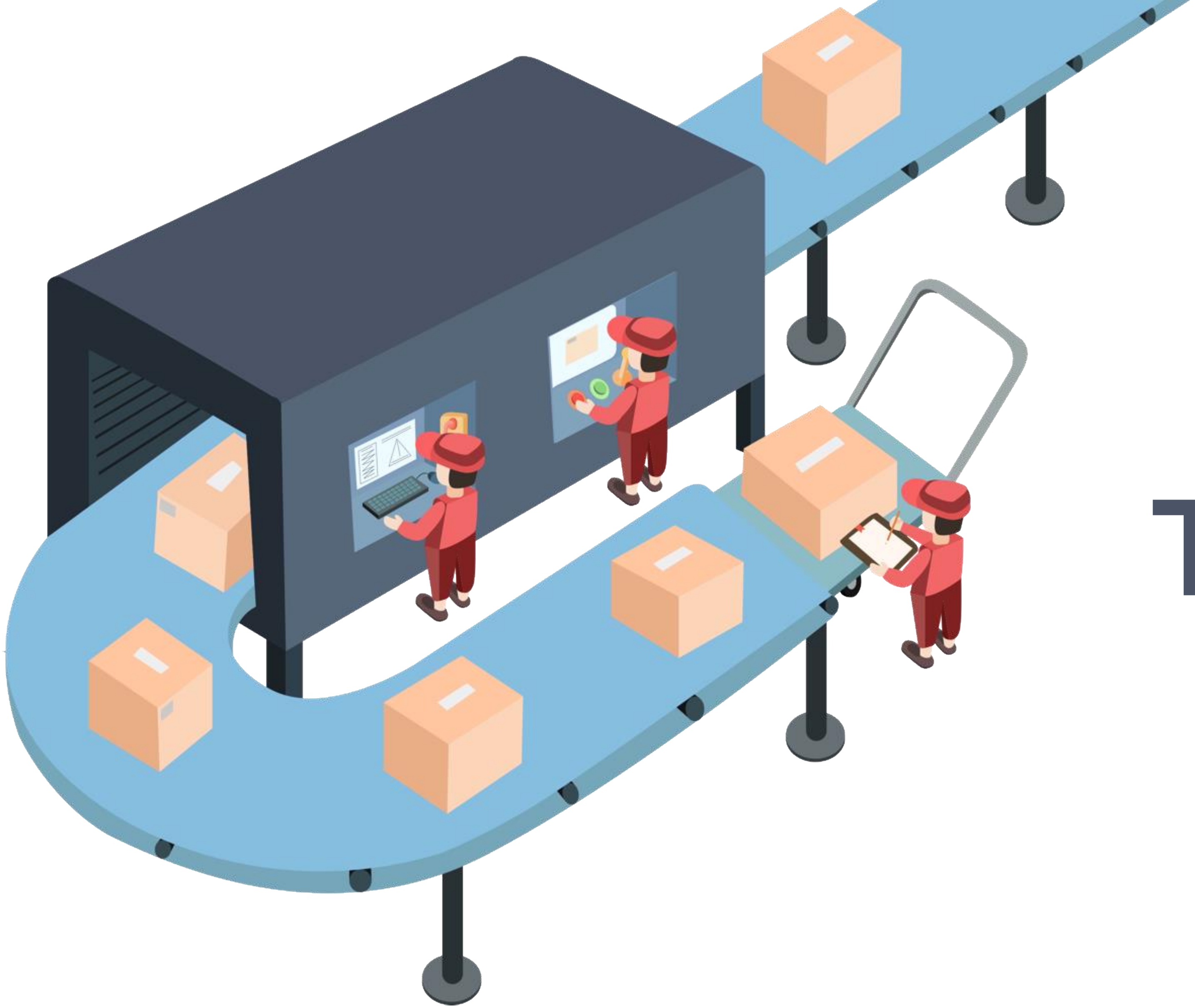
By setting the depot location,

- The holding cost increase significantly
- Have to pay more delivery cost
- More likely to have loss

- Reduce the frequency of replenishment in depot
- Minimize the distance from depot to store and to inventory

Better logistics strategy by simulating the inventory changing process!





Thank you!