厦門大學

# Management System Simulation Report

## An Inventory System

Group Members： Zhemian Zhao 17420182200850
Yifei Zhu 21620152203091

Instructors: Dr. Di Xu
Dr. Wei Zhang

2021.06.06

# Catalogue

# 1.Problem Description

## 1.1 Problem Background

The purpose of this simulation is to help a company which sells a single product to decide how many items it should have in inventory for each of the next n months (n is a fixed input parameter. Here, n=120 months, so this is a terminating simulation.). By controlling the same stream of random numbers, we can compare the average total cost per month of the 9 inventory policies and get the satisfactory solution.

## 1.2 The Simulation Variables

**Table 1.1** Input variables

| **Input** | | |
|---|---|---|
| **Data Type** | **Name** | **Description** |
| int | *m_initial_inv_level* | The initial inventory level |
| int | *m_simulation_length* | The End-simulation which n=120 months |
| int | *m_policy_number* | The number of the inventory policies chosen |
| int | *m_replication* | The times of replication |
| int | *m_interval* | The interval of each inventory policies |
| int | *m_num_values_demand* | The amount of a demand occurs |
| int | *m_seed* | The seed of the RNG |
| int | *m_min_inventory* | The lower limit of the inventory policies |
| int | *m_max_inventory* | The upper limit of the inventory policies |
| float | *m_maxlag* | The maximum delivery lag |
| float | *m_minlag* | The minimum delivery lag |
| float | *m_mean_interdemand_time* | The mean time between demands’ |
| float | *m_holding_cost* | Holding cost per item per month held |
| float | *m_shortage_cost* | Shortage cost per item per month occurs |
| float | *m_incremental_cost* | Incremental cost per item ordered |
| float | *m_setup_cost* | The setup cost of an order |

**Table 1.2** Output variables

| Output | | |
|---|---|---|
| **Data Type** | **Name** | **Description** |
| float | *Average total cost* | The sum of the average ordering cost per month, the average holding cost per month and the average backlog cost per month |
| float | *Average ordering cost* | The average ordering cost per month |
| float | *Average holding cost* | The average holding cost per month |
| float | *Average shortage cost* | The average backlog cost per month |
| float | *Expected proportion of backlog* | The expected proportion of time there is a backlog |
| int | *Expected number of express orders* | the expected number of express orders placed |
| float | *Proportion of spoiled* | the proportion of items taken out of the inventory that are discarded |

# 2.Model

## 2.1 The Probability Distribution

①The time between demands are IID exponential random variables with a mean of 0.1 month.

②The size of demands *D* are IID random variables(independent of when the demands occur)，with

$$D = \begin{cases} 1 \quad w.p.\dfrac{1}{6} \\ 2 \quad w.p.\dfrac{1}{3} \\ 3 \quad w.p.\dfrac{1}{3} \\ 4 \quad w.p.\dfrac{1}{6} \end{cases}$$

Where *w.p.* read “with probability”.

③The delivery lag (When an order is placed, the time required for it to arrive ) is a random variable that is distributed uniformly between 0.5 and 1 month. If an express order is placed, the deliver lag is uniformly distributed [0.25 , 0.5] month.

## 2.2 The State variables

The state variables for a simulation model of this inventory system are

- the inventory level *I(t)*,
- the amount of an outstanding order from the company to the supplier,
- and the time of the last event, which is needed to compute the areas under the $I^{+}(t)$ and $I^{-}(t)$ functions.

## 2.3 The Events of the Inventory System

**Table 2.1** Four events

| **Event description** | **Event type** |
|---|---|
| Arrival of an order to the company from the supplier | 1 |
| Demand for the product from a customer | 2 |
| End of the simulation after *n* months | 3 |
| Inventory evaluation (and possible ordering) at the beginning of a month | 4 |

Here we choose to make the end of the simulation event type 3 rather than type 4, since at time 120 both “end-simulation” and “inventory evaluation” events will eventually be scheduled and we would like to execute the former event first at this time.

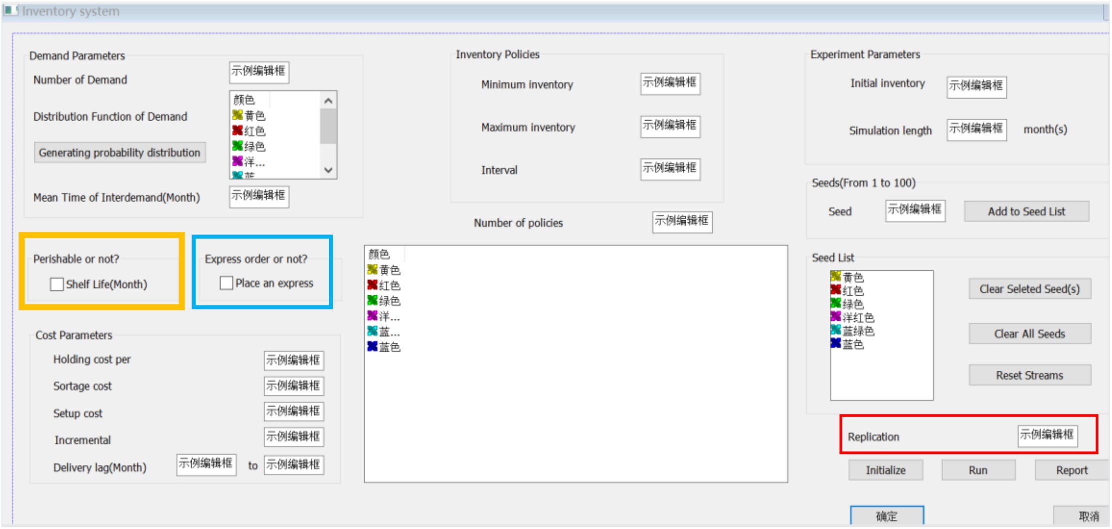
## 2.4 The Methods for computing the additional performance measures

2.4.1 Make five replication of each policy

In Dialog 4, add an edit control (Replication, area of the red box), at the same time, add a for loop to the main function `void Inventory::OnBnClickedButton2()` in *inventory.cpp*:

```
for (int i = 1; i <= replication; ++i)
{          ......
}
```

**Figure 2.1** Four events



2.4.3 Place an express order

In Dialog 4, add an check box (“Place an express”, area of the blue box), at the same time, add an if judgment statement to `void evaluate(void)` in *inventory.cpp*:

```
if (check_express == TRUE)
{
        ......
}
```

2.4.3 Place an express order

In Dialog 4, add an check box (“Shelf Life (Month)”, area of the yellow box), at the same time, add an if judgment statement to `void order_arrival(void)` and `void demand(int seed)` in *inventory.cpp*:

```
if (check_perishable == TRUE)
{
        ......
}
```

All above, the Newly added variables need to be initialized in `void inv_initialize(int seed)`.

# 3.Flow Chart

## 3.1 Flow Chart of the original problem

**Figure 3.1** Flow Chart for order-arrival routine

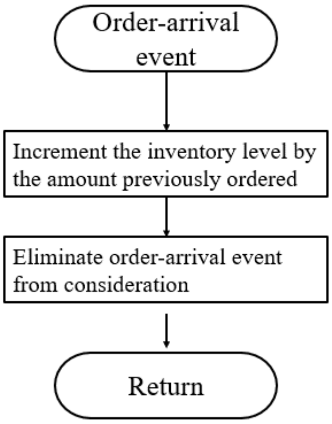

Order-arrival event
Increment the inventory level by the amount previously ordered
Eliminate order-arrival event from consideration
Return

**Figure 3.2** Flow Chart for demand routine

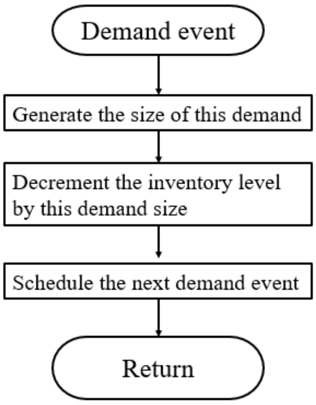

Demand event
Generate the size of this demand
Decrement the inventory level by this demand size
Schedule the next demand event
Return

**Figure 3.3** Flow Chart for inventory-evaluation routine

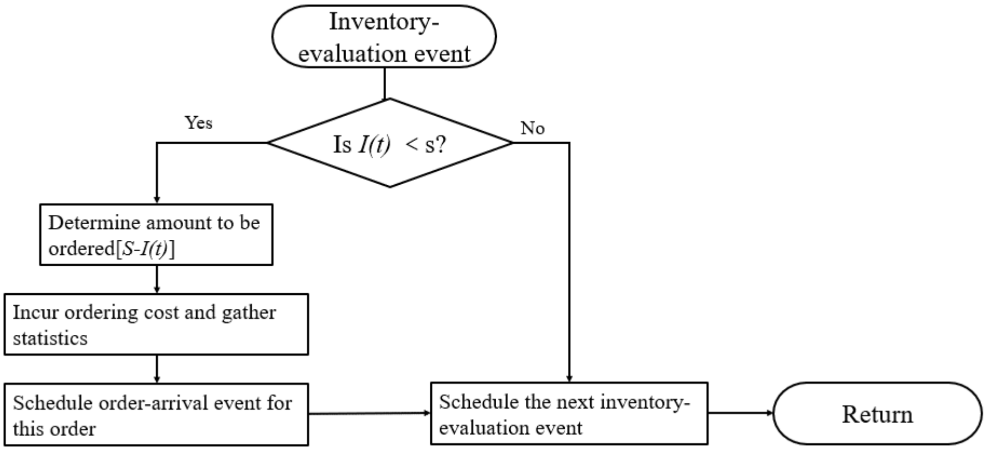
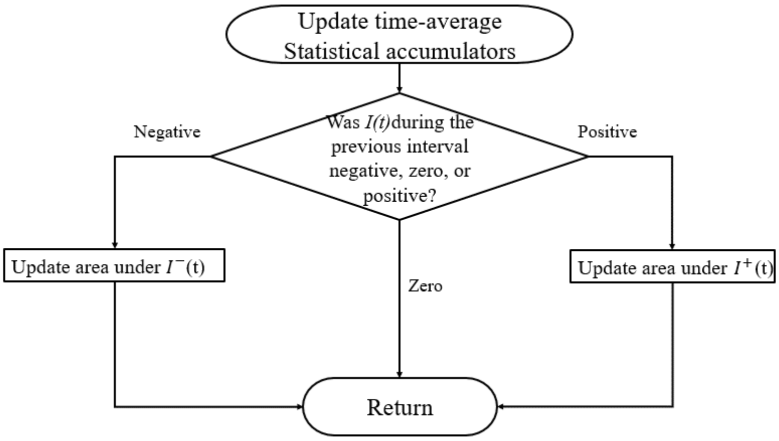

Inventory-evaluation event
Is I(t) < s?
Yes
No
Determine amount to be ordered[S-I(t)]
Incur ordering cost and gather statistics
Schedule order-arrival event for this order
Schedule the next inventory-evaluation event
Return

**Figure 3.4** Flow Chart for routine to update the continuous-time statistical accumulators


Update time-average Statistical accumulators
Was $I(t)$during the previous interval negative, zero, or positive?
Negative
Positive
Update area under $I^-$(t)
Zero
Update area under $I^+$(t)
Return

## 3.2 Flow Chart of the Additional performance measures

**Figure 3.5** Flow Chart for inventory-evaluation routine if placing an express order

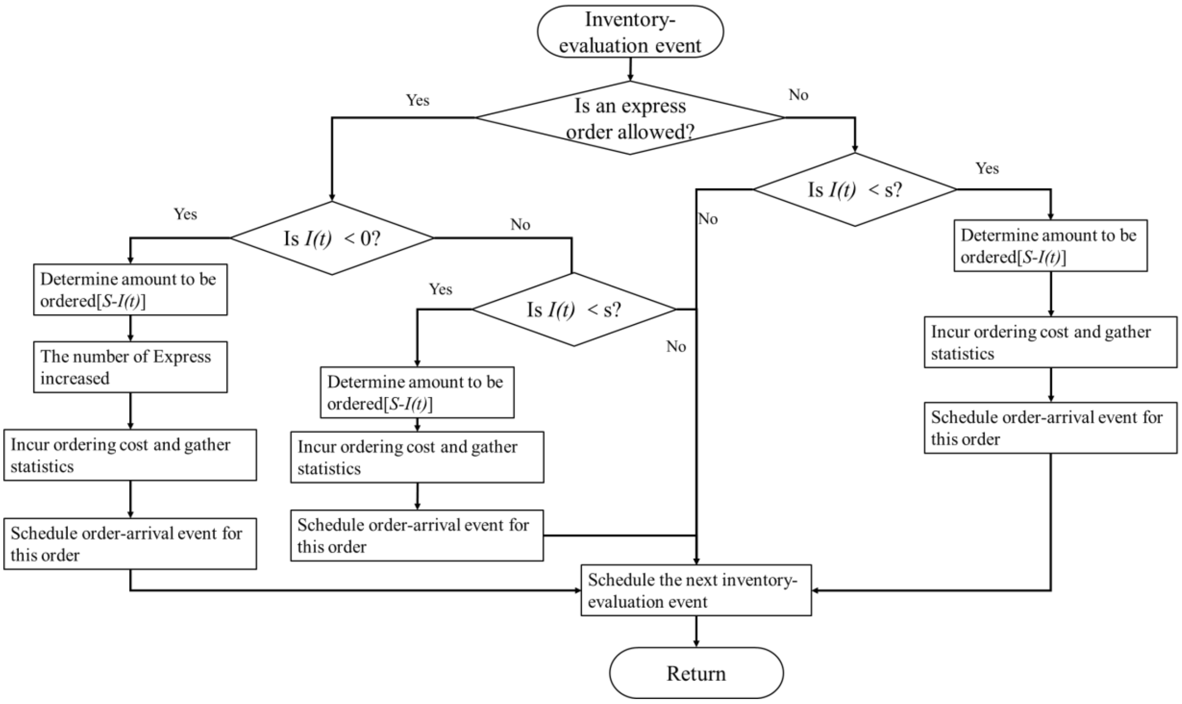

Inventory-evaluation event
Yes
Is an express order allowed?
No
Is $I(t)$ < s?
Yes
No
Yes
Is $I(t)$ < 0?
No
Determine amount to be ordered[$S$-$I(t)$]
Determine amount to be ordered[$S$-$I(t)$]
Yes
Is $I(t)$ < s?
No
Incur ordering cost and gather statistics
The number of Express increased
Determine amount to be ordered[$S$-$I(t)$]
Schedule order-arrival event for this order
Incur ordering cost and gather statistics
Incur ordering cost and gather statistics
Schedule order-arrival event for this order
Schedule order-arrival event for this order
Schedule the next inventory-evaluation event
Return

**Figure 3.6** Flow Chart for order-arrival routine if the inventory is perishable

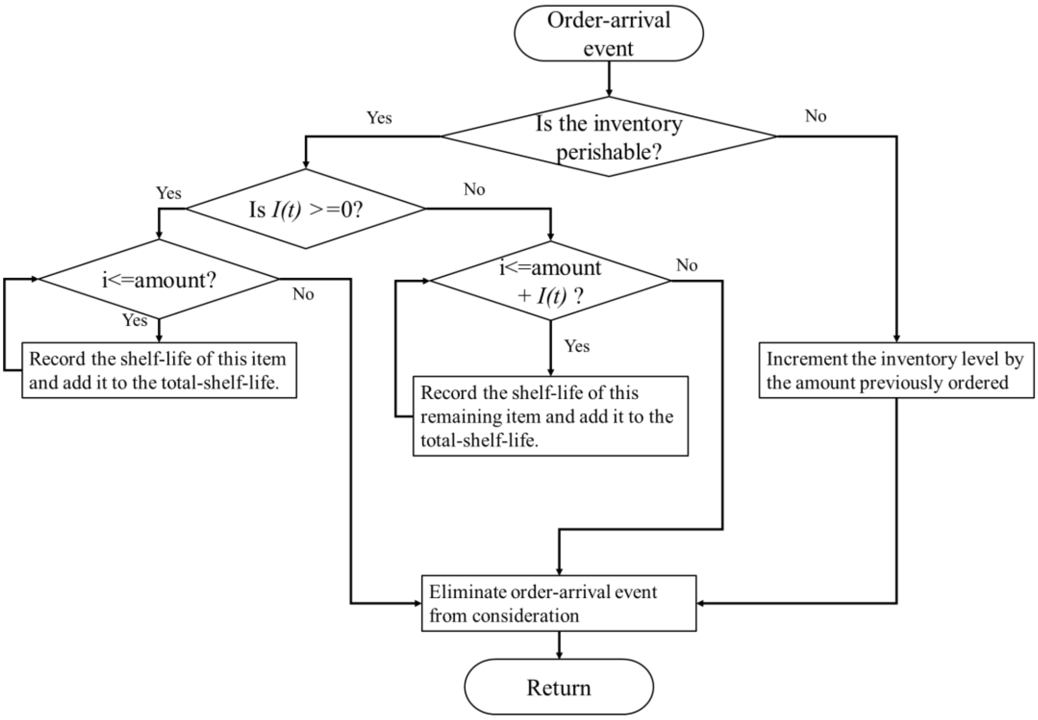
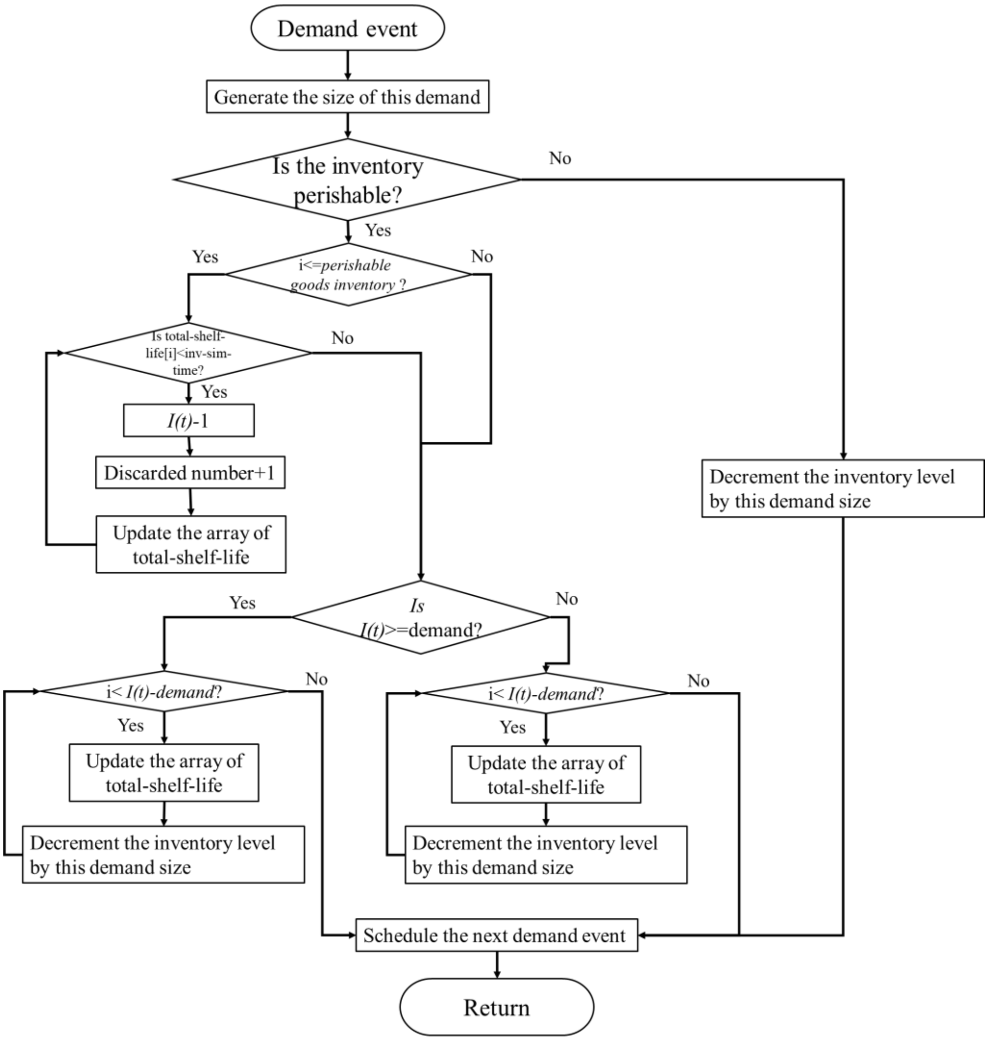

Order-arrival event
Is the inventory perishable?
Yes
No
Is I(t) >=0?
Yes
No
i<=amount?
No
Yes
Record the shelf-life of this item and add it to the total-shelf-life.
i<=amount + I(t) ?
No
Yes
Record the shelf-life of this remaining item and add it to the total-shelf-life.
Increment the inventory level by the amount previously ordered
Eliminate order-arrival event from consideration
Return

**Figure 3.7** Flow Chart for demand routine if the inventory is perishable

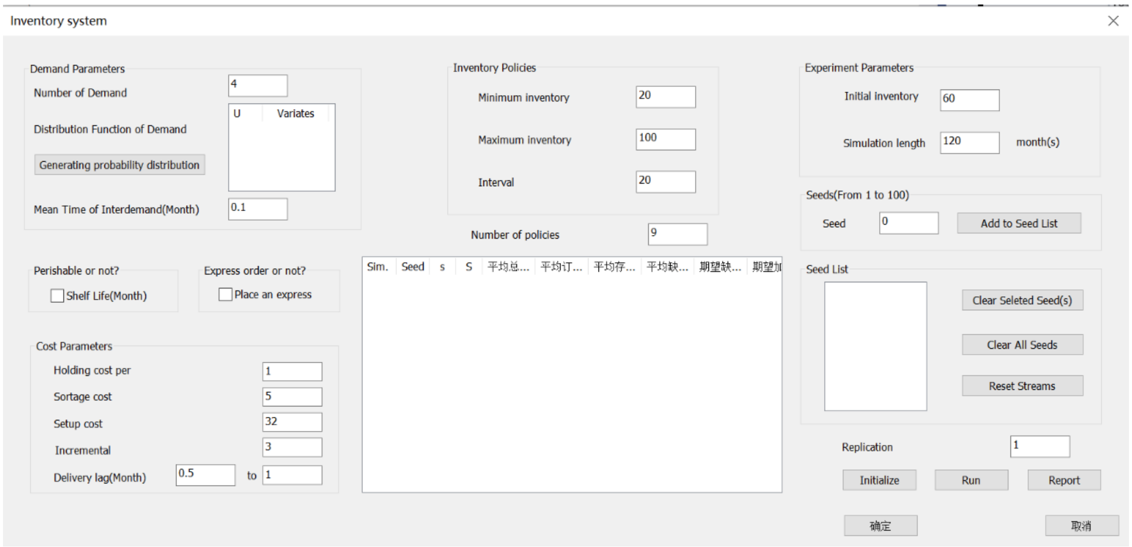

Demand event
Generate the size of this demand
Is the inventory perishable?
No
Yes
Yes
i<=perishable goods inventory ?
No
Is total-shelf-life[i]<inv-sim-time?
No
Yes
I(t)-1
Discarded number+1
Update the array of total-shelf-life
Decrement the inventory level by this demand size
Yes
Is I(t)>=demand?
No
i< I(t)-demand?
No
Yes
Update the array of total-shelf-life
Decrement the inventory level by this demand size
i< I(t)-demand?
No
Yes
Update the array of total-shelf-life
Decrement the inventory level by this demand size
Schedule the next demand event
Return

# 4.Output Analysis

## 4.1 Output of the original problem

Figure 4.1 shows the initial screen of the Inventory System. Most of the variables are initialized.

**Figure 4.1** The initial screen for the inventory system



Inventory system

Demand Parameters
Number of Demand: 4
Distribution Function of Demand: U | Variates
Mean Time of Interdemand(Month): 0.1

Perishable or not? □ Shelf Life(Month)

Express order or not? □ Place an express

Cost Parameters
Holding cost per: 1
Sortage cost: 5
Setup cost: 32
Incremental: 3
Delivery lag(Month): 0.5 to 1

Inventory Policies
Minimum inventory: 20
Maximum inventory: 100
Interval: 20

Number of policies: 9

| Sim. | Seed | s | S | 平均总... | 平均订... | 平均存... | 平均缺... | 期望缺... | 期望加 |
|---|---|---|---|---|---|---|---|---|---|

Experiment Parameters
Initial Inventory: 60
Simulation length: 120 month(s)

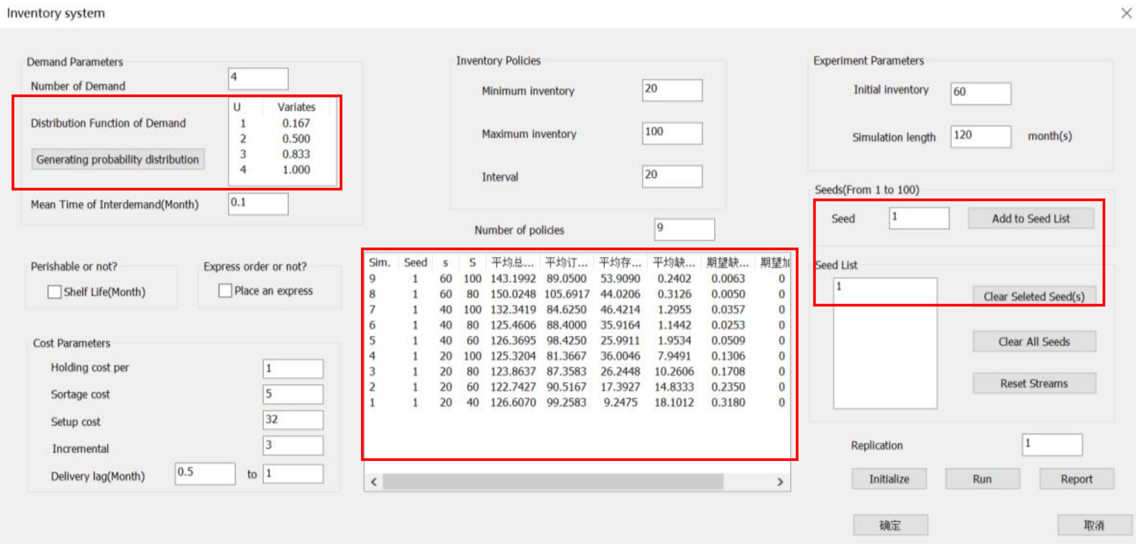
Seeds(From 1 to 100)
Seed: 0

Seed List

Replication: 1

When the button of "Generating probability distribution" is clicked, the distribution function of demand is shown. Then, add seed=1 into the SeedList, click the button of "Run", the results is shown below:

**Figure 4.2** The results of the original problem



Inventory system

Demand Parameters
Number of Demand: 4
Distribution Function of Demand:

| U | Variates |
|---|---|
| 1 | 0.167 |
| 2 | 0.500 |
| 3 | 0.833 |
| 4 | 1.000 |

Mean Time of Interdemand(Month): 0.1

Perishable or not? □ Shelf Life(Month)

Express order or not? □ Place an express

Cost Parameters
Holding cost per: 1
Sortage cost: 5
Setup cost: 32
Incremental: 3
Delivery lag(Month): 0.5 to 1

Inventory Policies
Minimum inventory: 20
Maximum inventory: 100
Interval: 20

Number of policies: 9

| Sim. | Seed | s | S | 平均总... | 平均订... | 平均存... | 平均缺... | 期望缺... | 期望加 |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 1 | 60 | 100 | 143.1992 | 89.0500 | 53.9090 | 0.2402 | 0.0063 | 0 |
| 8 | 1 | 60 | 80 | 150.0248 | 105.6917 | 44.0206 | 0.3126 | 0.0050 | 0 |
| 7 | 1 | 40 | 100 | 132.3419 | 84.6250 | 46.4214 | 1.2955 | 0.0357 | 0 |
| 6 | 1 | 40 | 80 | 125.4606 | 88.4000 | 35.9164 | 1.1442 | 0.0253 | 0 |
| 5 | 1 | 40 | 60 | 126.3695 | 98.4250 | 25.9911 | 1.9534 | 0.0509 | 0 |
| 4 | 1 | 20 | 100 | 125.3204 | 81.3667 | 36.0046 | 7.9491 | 0.1306 | 0 |
| 3 | 1 | 20 | 80 | 123.8637 | 87.3583 | 26.2448 | 10.2606 | 0.1708 | 0 |
| 2 | 1 | 20 | 60 | 122.7427 | 90.5167 | 17.3927 | 14.8333 | 0.2350 | 0 |
| 1 | 1 | 20 | 40 | 126.6070 | 99.2583 | 9.2475 | 18.1012 | 0.3180 | 0 |

Experiment Parameters
Initial Inventory: 60
Simulation length: 120 month(s)

Seeds(From 1 to 100)
Seed: 1

Seed List
1

Replication: 1

Compared with the results from the textbook, our experiment results is similar, which the difference may come from the different RNGs.

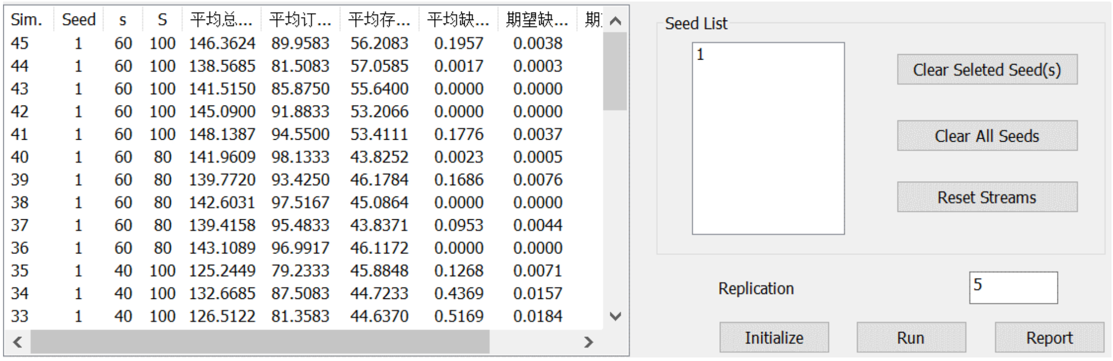
Compared with the 9 inventory policies, the satisfactory solution is (20,60),which is result of one test.

## 4.2 Output of five replication of each policy

### 4.2.1 Output

Click the button of “Reset Streams”, and then Enter 5 into the replication edit box, Run the program and get the results:

**Figure 4.3** Make five replication of each policy



| Sim. | Seed | s | S | 平均总... | 平均订... | 平均存... | 平均缺... | 期望缺... | 期 |
|---|---|---|---|---|---|---|---|---|---|
| 45 | 1 | 60 | 100 | 146.3624 | 89.9583 | 56.2083 | 0.1957 | 0.0038 | |
| 44 | 1 | 60 | 100 | 138.5685 | 81.5083 | 57.0585 | 0.0017 | 0.0003 | |
| 43 | 1 | 60 | 100 | 141.5150 | 85.8750 | 55.6400 | 0.0000 | 0.0000 | |
| 42 | 1 | 60 | 100 | 145.0900 | 91.8833 | 53.2066 | 0.0000 | 0.0000 | |
| 41 | 1 | 60 | 100 | 148.1387 | 94.5500 | 53.4111 | 0.1776 | 0.0037 | |
| 40 | 1 | 60 | 80 | 141.9609 | 98.1333 | 43.8252 | 0.0023 | 0.0005 | |
| 39 | 1 | 60 | 80 | 139.7720 | 93.4250 | 46.1784 | 0.1686 | 0.0076 | |
| 38 | 1 | 60 | 80 | 142.6031 | 97.5167 | 45.0864 | 0.0000 | 0.0000 | |
| 37 | 1 | 60 | 80 | 139.4158 | 95.4833 | 43.8371 | 0.0953 | 0.0044 | |
| 36 | 1 | 60 | 80 | 143.1089 | 96.9917 | 46.1172 | 0.0000 | 0.0000 | |
| 35 | 1 | 40 | 100 | 125.2449 | 79.2333 | 45.8848 | 0.1268 | 0.0071 | |
| 34 | 1 | 40 | 100 | 132.6685 | 87.5083 | 44.7233 | 0.4369 | 0.0157 | |
| 33 | 1 | 40 | 100 | 126.5122 | 81.3583 | 44.6370 | 0.5169 | 0.0184 | |

Seed List: 1

Replication: 5

### 4.2.2 Analysis

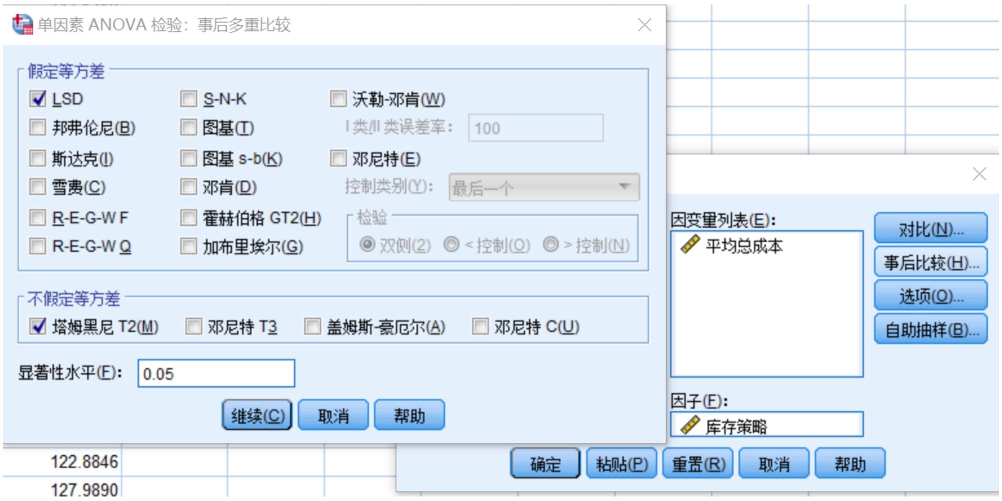
**Table 4.1** The results of 5 replication of each policy

| 库存策略(s,S) | 平均总成本 | 平均订货成本 | 平均存贮成本 | 平均缺货成本 |
|---|---|---|---|---|
| 60，100 | 146.3624 | 89.9583 | 55.2083 | 0.1957 |
| 60，100 | 138.5685 | 81.5083 | 57.0585 | 0.0017 |
| 60，100 | 141.5150 | 85.8750 | 55.6400 | 0.0000 |
| 60，100 | 145.0900 | 91.8833 | 53.2055 | 0.0000 |
| 60，100 | 148.1387 | 94.5500 | 53.4111 | 0.1775 |
| 60，80 | 141.9609 | 98.1333 | 43.8252 | 0.0023 |
| 60，80 | 139.7720 | 93.4250 | 45.1784 | 0.1685 |
| 60，80 | 142.6031 | 97.5167 | 45.0854 | 0.0000 |
| 60，80 | 139.4158 | 95.4833 | 43.8371 | 0.0953 |
| 60，80 | 143.1089 | 96.9917 | 45.1172 | 0.0000 |
| 40，100 | 125.2449 | 79.2333 | 45.8848 | 0.1258 |
| 40，100 | 132.6585 | 87.5083 | 44.7233 | 0.4369 |
| 40，100 | 126.5122 | 81.3583 | 44.6370 | 0.5169 |
| 40，100 | 130.3879 | 83.6417 | 45.8926 | 0.8537 |
| 40，100 | 124.7765 | 78.6167 | 45.5176 | 0.6423 |
| 40，80 | 127.5626 | 91.1917 | 34.4838 | 1.8872 |
| 40，80 | 130.4789 | 95.4500 | 33.8331 | 1.1958 |
| 40，80 | 124.9466 | 85.7000 | 36.3928 | 1.8537 |

| | | | | |
|---|---|---|---|---|
| 40，80 | 122.8846 | 87.0833 | 34.0853 | 1.7160 |
| 40，80 | 127.9890 | 91.7083 | 35.1725 | 1.1081 |
| 40，60 | 125.1134 | 98.0917 | 24.7010 | 2.3207 |
| 40，60 | 124.2856 | 97.1000 | 25.7525 | 1.4231 |
| 40，60 | 121.4633 | 93.4557 | 25.6160 | 1.3807 |
| 40，60 | 121.3767 | 93.3417 | 25.8571 | 1.1780 |
| 40，60 | 125.8853 | 100.6667 | 24.8114 | 1.4072 |
| 20，100 | 123.1172 | 78.4083 | 34.7198 | 9.9891 |
| 20，100 | 128.5315 | 84.1583 | 37.2588 | 7.1144 |
| 20，100 | 124.6411 | 81.8583 | 35.5752 | 7.2076 |
| 20，100 | 124.6774 | 80.6667 | 38. 3110 | 5.6997 |
| 20，100 | 131.2344 | 85.8750 | 36.1758 | 8.1836 |
| 20，80 | 117.0778 | 82.2250 | 27.6055 | 7.2452 |
| 20，80 | 116.9802 | 82.0750 | 25.8501 | 8.0550 |
| 20，80 | 111.4298 | 76.1917 | 29.3307 | 5.9075 |
| 20，80 | 121.7448 | 85.3833 | 25.1785 | 9.18 29 |
| 20，80 | 119.2427 | 83.0500 | 25.9139 | 9.2788 |
| 20，60 | 114.7740 | 85.0917 | 18.4501 | 11.2222 |
| 20，60 | 118.8412 | 89.7917 | 16.8877 | 12.1618 |
| 20，60 | 123.9769 | 93.2000 | 16.6448 | 14.1321 |
| 20，60 | 118.4904 | 88.1000 | 17.4303 | 12.9600 |
| 20，60 | 115.1206 | 85.6000 | 17.5250 | 10.9956 |
| 20，40 | 124.0061 | 95.9083 | 9.8085 | 18. 2892 |
| 20，40 | 128.1760 | 98.4250 | 9.2379 | 20.5131 |
| 20，40 | 126.7038 | 98.1917 | 9.0778 | 19.4344 |
| 20，40 | 125.6431 | 99.0917 | 9.1450 | 17.4064 |
| 20，40 | 126.6070 | 99.2583 | 9.2475 | 18.1012 |

Get the results into a table,and use SPSS to analysis. Let the inventory policies (20,40) ~ (60,100) be 1~9.

**Figure 4.3** The operation steps of SPSS



单因素 ANOVA 检验：事后多重比较

假定等方差

☑ LSD

不假定等方差

☑ 塔姆黑尼 T2(M)

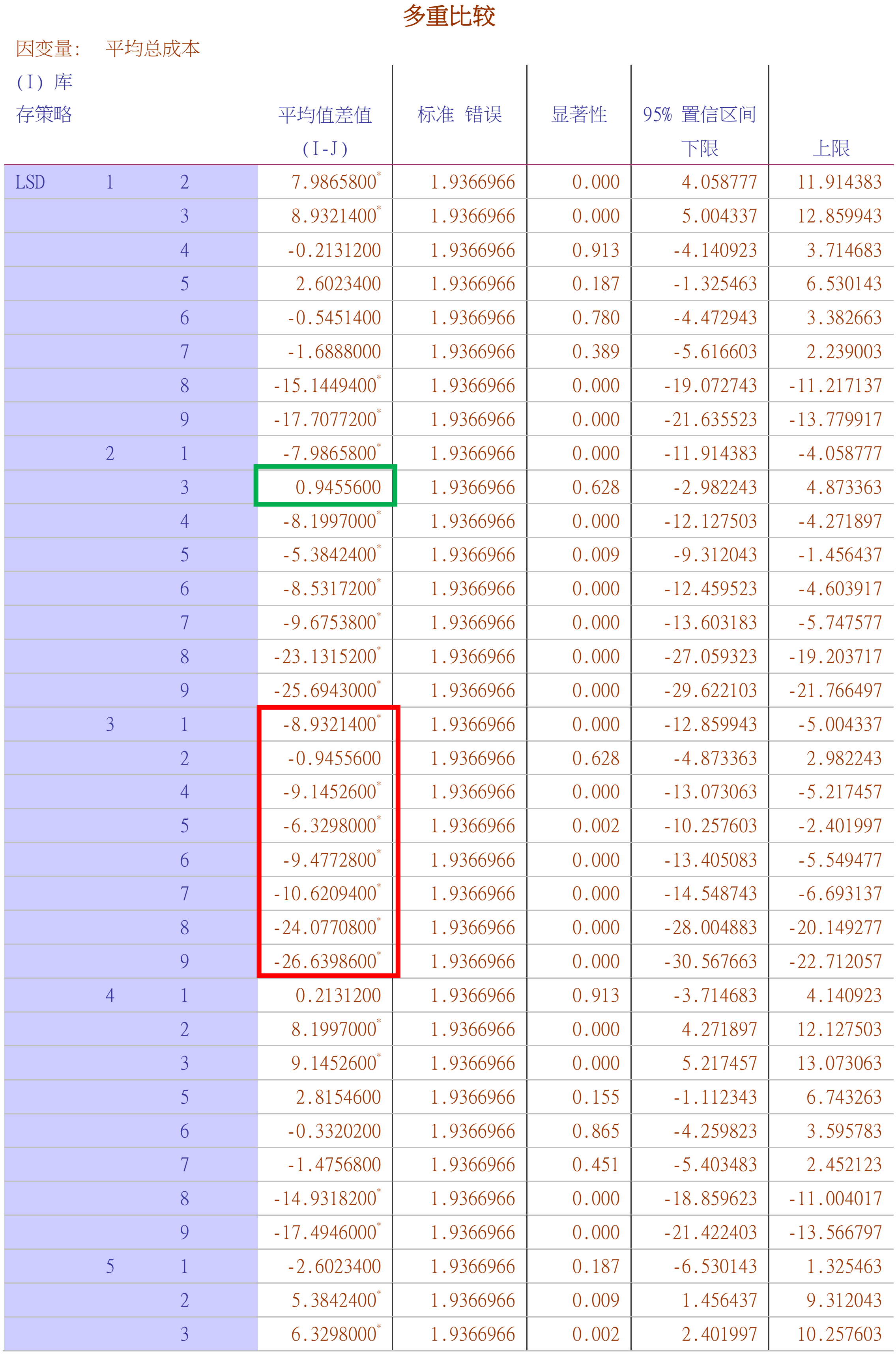
显著性水平(F)：0.05

因变量列表(E)：平均总成本

因子(F)：库存策略

Then ,get the following Table:

**Table 4.2** SPSS analysis



多重比较

因变量: 平均总成本

| | (I) 库存策略 | | 平均值差值 (I-J) | 标准 错误 | 显著性 | 95% 置信区间 | |
|---|---|---|---|---|---|---|---|
| | | | | | | 下限 | 上限 |
| LSD | 1 | 2 | 7.9865800* | 1.9366966 | 0.000 | 4.058777 | 11.914383 |
| | | 3 | 8.9321400* | 1.9366966 | 0.000 | 5.004337 | 12.859943 |
| | | 4 | -0.2131200 | 1.9366966 | 0.913 | -4.140923 | 3.714683 |
| | | 5 | 2.6023400 | 1.9366966 | 0.187 | -1.325463 | 6.530143 |
| | | 6 | -0.5451400 | 1.9366966 | 0.780 | -4.472943 | 3.382663 |
| | | 7 | -1.6888000 | 1.9366966 | 0.389 | -5.616603 | 2.239003 |
| | | 8 | -15.1449400* | 1.9366966 | 0.000 | -19.072743 | -11.217137 |
| | | 9 | -17.7077200* | 1.9366966 | 0.000 | -21.635523 | -13.779917 |
| | 2 | 1 | -7.9865800* | 1.9366966 | 0.000 | -11.914383 | -4.058777 |
| | | 3 | 0.9455600 | 1.9366966 | 0.628 | -2.982243 | 4.873363 |
| | | 4 | -8.1997000* | 1.9366966 | 0.000 | -12.127503 | -4.271897 |
| | | 5 | -5.3842400* | 1.9366966 | 0.009 | -9.312043 | -1.456437 |
| | | 6 | -8.5317200* | 1.9366966 | 0.000 | -12.459523 | -4.603917 |
| | | 7 | -9.6753800* | 1.9366966 | 0.000 | -13.603183 | -5.747577 |
| | | 8 | -23.1315200* | 1.9366966 | 0.000 | -27.059323 | -19.203717 |
| | | 9 | -25.6943000* | 1.9366966 | 0.000 | -29.622103 | -21.766497 |
| | 3 | 1 | -8.9321400* | 1.9366966 | 0.000 | -12.859943 | -5.004337 |
| | | 2 | -0.9455600 | 1.9366966 | 0.628 | -4.873363 | 2.982243 |
| | | 4 | -9.1452600* | 1.9366966 | 0.000 | -13.073063 | -5.217457 |
| | | 5 | -6.3298000* | 1.9366966 | 0.002 | -10.257603 | -2.401997 |
| | | 6 | -9.4772800* | 1.9366966 | 0.000 | -13.405083 | -5.549477 |
| | | 7 | -10.6209400* | 1.9366966 | 0.000 | -14.548743 | -6.693137 |
| | | 8 | -24.0770800* | 1.9366966 | 0.000 | -28.004883 | -20.149277 |
| | | 9 | -26.6398600* | 1.9366966 | 0.000 | -30.567663 | -22.712057 |
| | 4 | 1 | 0.2131200 | 1.9366966 | 0.913 | -3.714683 | 4.140923 |
| | | 2 | 8.1997000* | 1.9366966 | 0.000 | 4.271897 | 12.127503 |
| | | 3 | 9.1452600* | 1.9366966 | 0.000 | 5.217457 | 13.073063 |
| | | 5 | 2.8154600 | 1.9366966 | 0.155 | -1.112343 | 6.743263 |
| | | 6 | -0.3320200 | 1.9366966 | 0.865 | -4.259823 | 3.595783 |
| | | 7 | -1.4756800 | 1.9366966 | 0.451 | -5.403483 | 2.452123 |
| | | 8 | -14.9318200* | 1.9366966 | 0.000 | -18.859623 | -11.004017 |
| | | 9 | -17.4946000* | 1.9366966 | 0.000 | -21.422403 | -13.566797 |
| | 5 | 1 | -2.6023400 | 1.9366966 | 0.187 | -6.530143 | 1.325463 |
| | | 2 | 5.3842400* | 1.9366966 | 0.009 | 1.456437 | 9.312043 |
| | | 3 | 6.3298000* | 1.9366966 | 0.002 | 2.401997 | 10.257603 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 4 | -2.8154600 | 1.9366966 | 0.155 | -6.743263 | 1.112343 |
| | 6 | -3.1474800 | 1.9366966 | 0.113 | -7.075283 | 0.780323 |
| | 7 | -4.2911400* | 1.9366966 | 0.033 | -8.218943 | -0.363337 |
| | 8 | -17.7472800* | 1.9366966 | 0.000 | -21.675083 | -13.819477 |
| | 9 | -20.3100600* | 1.9366966 | 0.000 | -24.237863 | -16.382257 |
| 6 | 1 | 0.5451400 | 1.9366966 | 0.780 | -3.382663 | 4.472943 |
| | 2 | 8.5317200* | 1.9366966 | 0.000 | 4.603917 | 12.459523 |
| | 3 | 9.4772800* | 1.9366966 | 0.000 | 5.549477 | 13.405083 |
| | 4 | 0.3320200 | 1.9366966 | 0.865 | -3.595783 | 4.259823 |
| | 5 | 3.1474800 | 1.9366966 | 0.113 | -0.780323 | 7.075283 |
| | 7 | -1.1436600 | 1.9366966 | 0.559 | -5.071463 | 2.784143 |
| | 8 | -14.5998000* | 1.9366966 | 0.000 | -18.527603 | -10.671997 |
| | 9 | -17.1625800* | 1.9366966 | 0.000 | -21.090383 | -13.234777 |
| 7 | 1 | 1.6888000 | 1.9366966 | 0.389 | -2.239003 | 5.616603 |
| | 2 | 9.6753800* | 1.9366966 | 0.000 | 5.747577 | 13.603183 |
| | 3 | 10.6209400* | 1.9366966 | 0.000 | 6.693137 | 14.548743 |
| | 4 | 1.4756800 | 1.9366966 | 0.451 | -2.452123 | 5.403483 |
| | 5 | 4.2911400* | 1.9366966 | 0.033 | 0.363337 | 8.218943 |
| | 6 | 1.1436600 | 1.9366966 | 0.559 | -2.784143 | 5.071463 |
| | 8 | -13.4561400* | 1.9366966 | 0.000 | -17.383943 | -9.528337 |
| | 9 | -16.0189200* | 1.9366966 | 0.000 | -19.946723 | -12.091117 |
| 8 | 1 | 15.1449400* | 1.9366966 | 0.000 | 11.217137 | 19.072743 |
| | 2 | 23.1315200* | 1.9366966 | 0.000 | 19.203717 | 27.059323 |
| | 3 | 24.0770800* | 1.9366966 | 0.000 | 20.149277 | 28.004883 |
| | 4 | 14.9318200* | 1.9366966 | 0.000 | 11.004017 | 18.859623 |
| | 5 | 17.7472800* | 1.9366966 | 0.000 | 13.819477 | 21.675083 |
| | 6 | 14.5998000* | 1.9366966 | 0.000 | 10.671997 | 18.527603 |
| | 7 | 13.4561400* | 1.9366966 | 0.000 | 9.528337 | 17.383943 |
| | 9 | -2.5627800 | 1.9366966 | 0.194 | -6.490583 | 1.365023 |
| 9 | 1 | 17.7077200* | 1.9366966 | 0.000 | 13.779917 | 21.635523 |
| | 2 | 25.6943000* | 1.9366966 | 0.000 | 21.766497 | 29.622103 |
| | 3 | 26.6398600* | 1.9366966 | 0.000 | 22.712057 | 30.567663 |
| | 4 | 17.4946000* | 1.9366966 | 0.000 | 13.566797 | 21.422403 |
| | 5 | 20.3100600* | 1.9366966 | 0.000 | 16.382257 | 24.237863 |
| | 6 | 17.1625800* | 1.9366966 | 0.000 | 13.234777 | 21.090383 |
| | 7 | 16.0189200* | 1.9366966 | 0.000 | 12.091117 | 19.946723 |
| | 8 | 2.5627800 | 1.9366966 | 0.194 | -1.365023 | 6.490583 |

*. 平均值差值的显著性水平为 0.05。

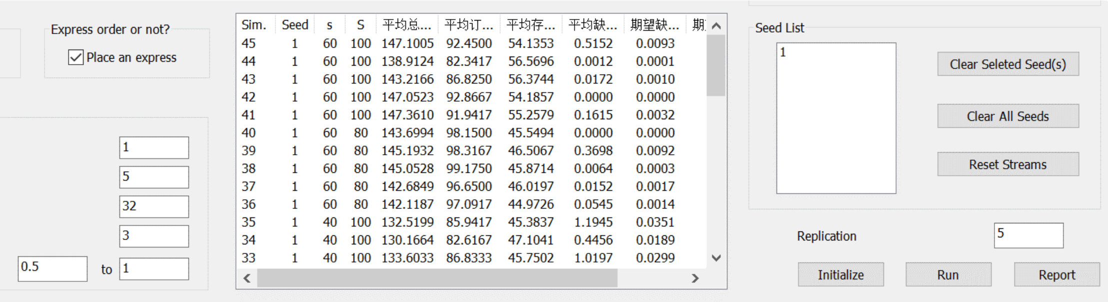
Form the table 4.2, we can find that making 5 replication will get another satisfactory solution, which is (20,60) at a 95% confidence level(area of the red box). This indicates that the results obtained by only one test are not credible, so it is of great significance to carry out repeated tests through simulation.

## 4.3 Output of express orders

### 4.2.1 Output

Click the button of “Reset Streams”, and then select the check of “Place an express”, Run the program and get the results:

**Figure 4.4** The results of express orders



Express order or not?

☑ Place an express

1

5

32

3

0.5 to 1

| Sim. | Seed | s | S | 平均总... | 平均订... | 平均存... | 平均缺... | 期望缺... | 期 |
|---|---|---|---|---|---|---|---|---|---|
| 45 | 1 | 60 | 100 | 147.1005 | 92.4500 | 54.1353 | 0.5152 | 0.0093 | |
| 44 | 1 | 60 | 100 | 138.9124 | 82.3417 | 56.5696 | 0.0012 | 0.0001 | |
| 43 | 1 | 60 | 100 | 143.2166 | 86.8250 | 56.3744 | 0.0172 | 0.0010 | |
| 42 | 1 | 60 | 100 | 147.0523 | 92.8667 | 54.1857 | 0.0000 | 0.0000 | |
| 41 | 1 | 60 | 100 | 147.3610 | 91.9417 | 55.2579 | 0.1615 | 0.0032 | |
| 40 | 1 | 60 | 80 | 143.6994 | 98.1500 | 45.5494 | 0.0000 | 0.0000 | |
| 39 | 1 | 60 | 80 | 145.1932 | 98.3167 | 46.5067 | 0.3698 | 0.0092 | |
| 38 | 1 | 60 | 80 | 145.0528 | 99.1750 | 45.8714 | 0.0064 | 0.0003 | |
| 37 | 1 | 60 | 80 | 142.6849 | 96.6500 | 46.0197 | 0.0152 | 0.0017 | |
| 36 | 1 | 60 | 80 | 142.1187 | 97.0917 | 44.9726 | 0.0545 | 0.0014 | |
| 35 | 1 | 40 | 100 | 132.5199 | 85.9417 | 45.3837 | 1.1945 | 0.0351 | |
| 34 | 1 | 40 | 100 | 130.1664 | 82.6167 | 47.1041 | 0.4456 | 0.0189 | |
| 33 | 1 | 40 | 100 | 133.6033 | 86.8333 | 45.7502 | 1.0197 | 0.0299 | |

Seed List: 1

Replication: 5

### 4.2.2 Analysis

We collect the average total cost of each policy that is expressed with 5 replication. To compare the express orders with the ordinary orders, we can also use SPSS. Let the inventory policies (20,40) ~ (60,100) with express orders be 11~19.

多重比较

因变量： 平均总成本

| (I) 库存策略 | | | 平均值差值 (I-J) | 标准 错误 | 显著性 | 95% 置信区间 | |
|---|---|---|---|---|---|---|---|
| | | | | | | 下限 | 上限 |
| LSD | 1 | | | | | | |
| | | 11 | -3.6192800 | 2.1026091 | 0.089 | -7.810755 | 0.572195 |
| | 2 | | | | | | |
| | | 12 | -7.1374600* | 2.1026091 | 0.001 | -11.328935 | -2.945985 |
| | 3 | | | | | | |
| | | 13 | -7.9823600* | 2.1026091 | 0.000 | -12.173835 | -3.790885 |
| | 4 | | | | | | |
| | | 14 | -3.5455400 | 2.1026091 | 0.096 | -7.737015 | 0.645935 |
| | 5 | | | | | | |
| | | 15 | -1.9466800 | 2.1026091 | 0.358 | -6.138155 | 2.244795 |
| | 6 | | | | | | |
| | | 16 | -1.4800000 | 2.1026091 | 0.484 | -5.671475 | 2.711475 |
| | 7 | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 17 | -0.2978200 | 2.1026091 | 0.888 | -4.489295 | 3.893655 |
| 8 | | | | | | |
| | 18 | -2.3796400 | 2.1026091 | 0.261 | -6.571115 | 1.811835 |
| 9 | | | | | | |
| | 18 | -0.7936400 | 2.1026091 | 0.707 | -4.985115 | 3.397835 |

*. 平均值差值的显著性水平为 0.05。

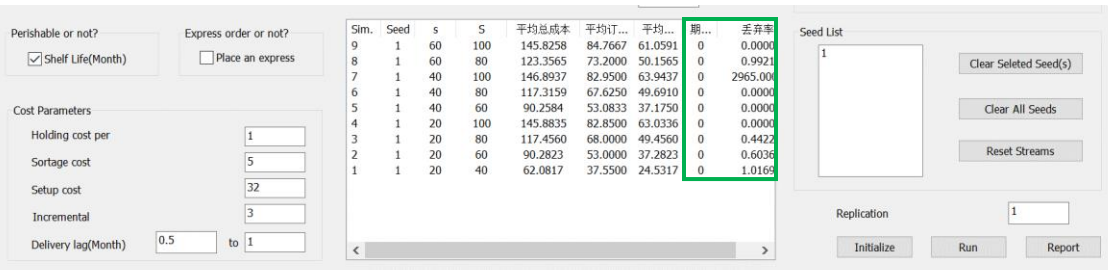
All of the mean difference(ordinary orders - express orders) is below zero, so we can make a conclusion that there is no worth to expressing the order.

## 4.4 Output of perishable inventory

4.2.1 Output

Click the button of "Reset Streams", and then select the check of "", Run the program and get the results:

**Figure 4.5** The results of perishable



Perishable or not?

☑ Shelf Life(Month)

Express order or not?

□ Place an express

Cost Parameters

| | |
|---|---|
| Holding cost per | 1 |
| Sortage cost | 5 |
| Setup cost | 32 |
| Incremental | 3 |
| Delivery lag(Month) | 0.5 to 1 |

| Sim. | Seed | s | S | 平均总成本 | 平均订... | 平均... | 期... | 丢弃率 |
|---|---|---|---|---|---|---|---|---|
| 9 | 1 | 60 | 100 | 145.8258 | 84.7667 | 61.0591 | 0 | 0.0000 |
| 8 | 1 | 60 | 80 | 123.3565 | 73.2000 | 50.1565 | 0 | 0.9921 |
| 7 | 1 | 40 | 100 | 146.8937 | 82.9500 | 63.9437 | 0 | 2965.00 |
| 6 | 1 | 40 | 80 | 117.3159 | 67.6250 | 49.6910 | 0 | 0.0000 |
| 5 | 1 | 40 | 60 | 90.2584 | 53.0833 | 37.1750 | 0 | 0.0000 |
| 4 | 1 | 20 | 100 | 145.8835 | 82.8500 | 63.0336 | 0 | 0.0000 |
| 3 | 1 | 20 | 80 | 117.4560 | 68.0000 | 49.4560 | 0 | 0.4422 |
| 2 | 1 | 20 | 60 | 90.2823 | 53.0000 | 37.2823 | 0 | 0.6036 |
| 1 | 1 | 20 | 40 | 62.0817 | 37.5500 | 24.5317 | 0 | 1.0169 |

Seed List

1

Replication: 1

We can find that there are some problems with the results. There must be some wrong with the code, but so far we haven't got it right.

We will try our best to make further improvements in the future.

## 5. Summary

5.1 Do everything with a sense of advance, both of us should learn to overcome procrastination.

5.2 Communication between team members is very important. We can learn from each other.

5.3 Experimental operation is just a skill. Whether the program can run or whether the result is correct, depends mostly on whether the logic of this problem is fully understood.