Experiment: changing a simulation system

- Now consider this updated game:
  1. There are N=2K persons in the game
  2. In each time, two person (K pairs) will trade with each other
  3. You and your counterparties both have three options:
     - Trade, or says, trust
     - Cheat, or says, betray
     - Reject, or says, refuse to trade

     Once both persons choose his/her options, calculate the points he/she get as the table

		A		
		Trust	Betray	Reject
B	Trust	A: +10; B: +10	A: +2X; B: -X	A&B: +0
	Betray	A: -X; B: +2X	A: -Y; B: -Y	A&B: +0
	Reject	A&B: +0	A&B: +0	A&B: +0

Experiment: changing a simulation system

- Now, you are not the participant only, you are the game designer!
- You need to consider two parts:
  - Technique part: how to change the system so that the ‘Reject’ can be added?
  - strategy part: how to design X and Y so that the ‘Reject’ is useful?
- The Technique part is easy:
  - In previous system, we design ‘0’ = ‘Trade’ and ‘not 0’ = ‘Betray’
  - Quick question: why not assign ‘0’ = ‘Trade’ and ‘1’ = ‘Betray’?
    - Because, in that case, one might give a strategy ‘2’, or ‘A’, and the system will fails.
  - Therefore, we must define at least one option of the system as ‘others’
  - For example:
    - ‘0’ = ‘Trade’ and ‘>0’ = ‘Betray’
    - ‘others’ = ‘Reject’

Experiment: changing a simulation system

‘0’ = ‘Trade’ and ‘>0’ = ‘Betray’
‘others’ = ‘Reject’

```
if Strategy_this==0
    if Strategy_counterparty==0
        Return_one_trade(person_id) = 10;   % both trust, add 10 points
    elseif Strategy_counterparty>0
        Return_one_trade(person_id) = -X_betray_trust_point;   % self trust, counterparty betray, -X = -
    else
        Return_one_trade(person_id) = 0;    % self trust, counterparty reject, 0 points
    end
elseif Strategy_this>0
    if Strategy_counterparty==0
        Return_one_trade(person_id) = 2*X_betray_trust_point;   % self betray, counterparty trust, + 2 >
    elseif Strategy_counterparty>0
        Return_one_trade(person_id) = -Y_betray_betray_point;    % self betray, counterparty betray, -Y
    else
        Return_one_trade(person_id) = 0;    % self betray, counterparty reject, 0 points
    end
else    % self reject, always 0 points, no matter what counterparty action is
    Return_one_trade(person_id) = 0;
end
```

Experiment: changing a simulation system

- Before we goes on, let's see the loopholes, or the unsatisfying parts of the previous system, and change them first.
- Let's take the previous default id21.m as an example

```
% Print your student ID and Name here, for example
% 000000    Weize Sun
%%
% your_strategy returns your strategy of the trade this time
% your_strategy = 0 means that you want to trust the counterparty this time
% your_strategy not equal to 0 means that you want to betray the
% counterparty this time
%%
% counterparty_id is the ID of the counterparty you are going to trade with
% this time
%% Now we begins
function [your_strategy] = id21(counterparty_id)
    load storage_id21.mat
    if mod(Trade_no,2) == 0
        your_strategy = 0;  % this means that you will trust this person
    else
        your_strategy = 1;  % this means that you will betray this person
    end
    Trade_no = Trade_no + 1;
    save('storage_id21', 'Trade_no', 'your_id')
    % your strategy will trush one person and then betray one and goes on
    % ONLY save your data in the file storage_id21.mat,
    % otherwise you will be treated as 'homework not submitted'
end
```

Here, it load the 'storage_id21.mat' at default, and use the 'Trade_no' to decide his strategy. It will leads to the following problems:

1. If there is no 'storage_id21.mat', the system will fails.
2. The 'storage_id21.mat' might give a default value of 'Trade_no' randomly, making the strategy very unstable

- For a game participant, he should care about the 2nd problem majorly (also, he should care about the 1st problem in order to avoid failure of his program);
- But, for the game designer, he should care about the 1st problem seriously!

Experiment: changing a simulation system

- Before we goes on, let's see the loopholes, or the unsatisfying parts of the previous system, and change them first.
- Let's take the previous default id21.m as an example

```
% Print your student ID and Name here, for example
% 000000    Weize Sun
%%
% your_strategy returns your strategy of the trade this time
% your_strategy = 0 means that you want to trust the counterparty this time
% your_strategy not equal to 0 means that you want to betray the
% counterparty this time
%%
% counterparty_id is the ID of the counterparty you are going to trade with
% this time
%% Now we begins
function [your_strategy] = id21(counterparty_id)
    load storage_id21.mat
    if mod(Trade_no,2) == 0
        your_strategy = 0;  % this means that you will trust this person
    else
        your_strategy = 1;  % this means that you will betray this person
    end
    Trade_no = Trade_no + 1;
    save('storage_id21', 'Trade_no', 'your_id')
    % your strategy will trush one person and then betray one and goes on
    % ONLY save your data in the file storage_id21.mat,
    % otherwise you will be treated as 'homework not submitted'
end
```

Therefore, in the main program, we should add some codes to generate a default 'storage_idXX.mat' with a parameter 'Trade_no', then the system will be of less possibility to fail.

Experiment: changing a simulation system

- That is:

```
creat_storage(N_persons)                    % create all the  'storage_idXX.mat'
```

```
文件    导航    编辑    断点
function creat_storage(N_persons)
    for your_id=1:N_persons
        filename=['storage_id',num2str(your_id)];
%         eval([filename,'=your_id']);
        Trade_no = 0;
        save(filename, 'your_id', 'Trade_no');
    end
end
```

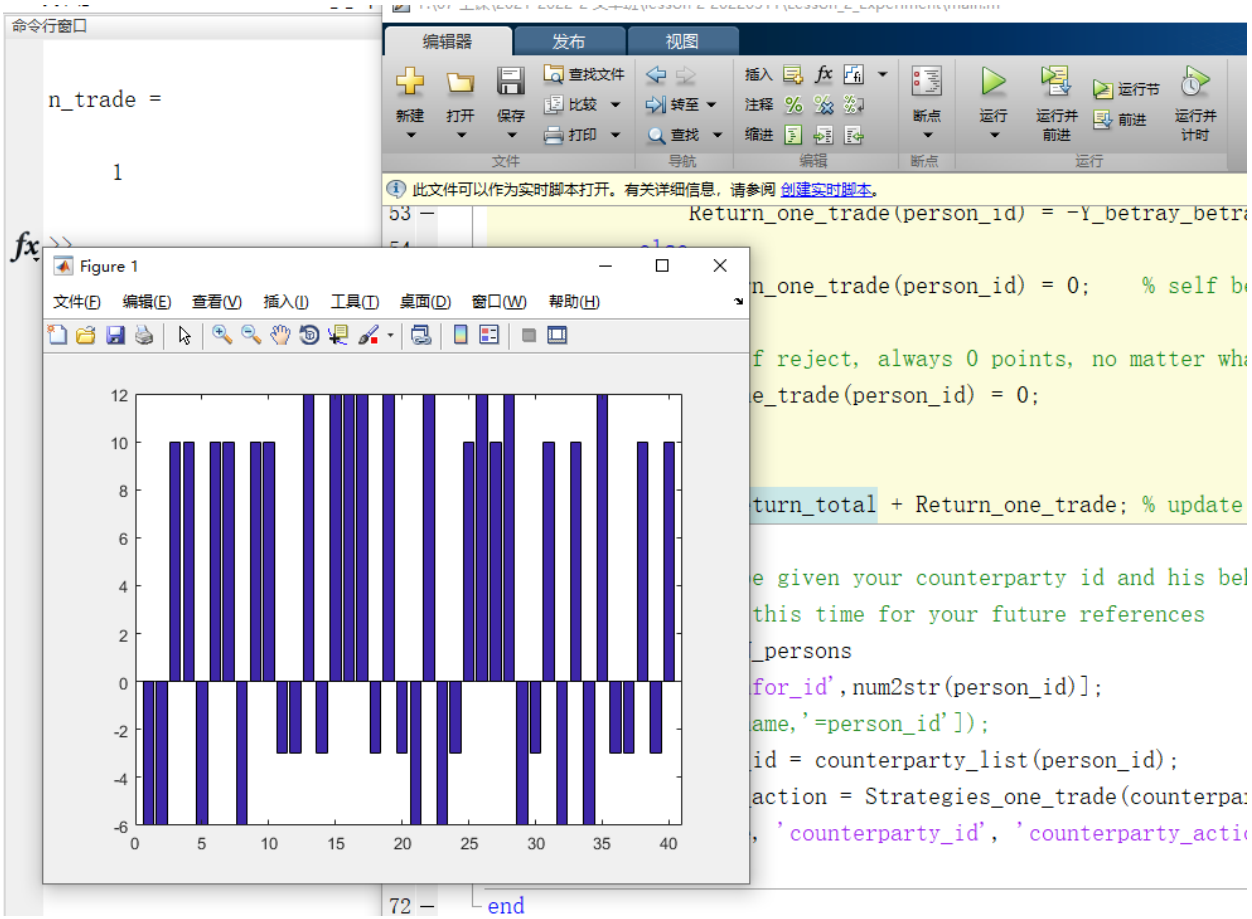
Experiment: changing a simulation system

- Similarly, you are also required to take care of the ‘infor_idXX.mat’, as its generation code is placed in the end of the main loop.
- If there are no ‘infor_idXX.mat’, for example, ‘infor_id31.mat’, in the very beginning, the system will fail!

- Try to modify this system by yourself

Experiment: changing a simulation system

- For example: there is ‘infor_idXX.mat’



Experiment: changing a simulation system

- For example: no ‘infor_idXX.mat’

命令行窗口

```
n_trade =

     1

错误使用 load
无法读取文件 'infor_id31.mat'。没有此类文件或目录。

出错 id31 (line 12)
    load infor_id31.mat

出错 Run_Strategies (line 32)
    Strategies_one_trade(31) = id31(counterparty_list(31));

出错 main (line 36)
    Strategies_one_trade = Run_Strategies(counterparty_list);

fx >>
```

工作区

F:\07 上课\2021-2022-2 文华班\lesson 2 20220311\Lesson_2_Experiment\main.m

编辑器 发布 视图

此文件可以作为实时脚本打开。有关详细信息，请参阅 创建实时脚本。

```
53 —                Return_one_trade(person_id) = -Y_betray_betray_point;
54 —            else
55 —                Return_one_trade(person_id) = 0;    % self betray, co
56 —            end
57 —        else    % self reject, always 0 points, no matter what counte
58 —            Return_one_trade(person_id) = 0;
59 —        end
60 —    end
61 —    Return_total = Return_total + Return_one_trade; % update the retu
62      %%
63      % here you will be given your counterparty id and his behaviour
64      % information of this time for your future references
65 —    for person_id=1:N_persons
66 —        filename=['infor_id',num2str(person_id)];
67  %         eval([filename,'=person_id']);
68 —        counterparty_id = counterparty_list(person_id);
69 —        counterparty_action = Strategies_one_trade(counterparty_id);
70 —        save(filename, 'counterparty_id', 'counterparty_action');
71 —    end
72 —
```

- Try to modify this system by yourself

Experiment: changing a simulation system

- Now we go back to the strategy problem: how to design X and Y so that the ‘Reject’ is useful?
- Generally speaking, it is a ‘strategy’ or ‘Sociology(社会学)’ problem
  - Different person can give different ideology, thus leading to different X and Y
- But, this is a ~~math~~ programming course!
- Therefore, I will simply introduce a ‘programming’ method to decide the X and Y.

Experiment: changing a simulation system – the programming method

- We begin with the following assumptions:
  - The previous default 30 strategies, i.e., id1 to id30, are appropriate
    - 10 always trade; 10 always betray; 10 trade once and then betray once and then go on
  - There are 10 more strategies for us to test the ‘reject’ option
    - Of course, you can use more, for example, you can set ‘N_persons = 300’ and test 270 strategies to see how ‘reject’ works, but that is another story.
  - For these 10 strategies, if the ‘reject’ option is not chosen, randomly trade or betray
    - Other strategies can be set, but this ‘randomly trade or betray’ can make the system more robust to the general case: there are always trade, and betray

Experiment: changing a simulation system – the programming method

- How to design?
  - 10 strategies, with ‘reject’ probability 10% to 100%
  - trade 1000 times
  - Test several pairs of X and Y values, and see which pair make the strategy that ‘reject’ with probability 30% wins
  - Choose this X and Y

- Note: this is just one idea, you can use your idea to do the design
- Your are encourage to try this test, but it is not a homework
- Here I will show one ‘reject after being betrayed’ strategy

Experiment: the reject after being betrayed strategy

```
function [your_strategy] = id36(counterparty_id)
    counterparty_now = counterparty_id;
    % as loading the infor_id36.mat will give a variable named
    % counterparty_id, therefore here, we store the counterparty_id from
    % the input to counterparty_now first!
    % In fact, this problem can be solved by a good design of the system,
    % you can modify the system by yourself, however, when submitting your
    % homework, your are suggestted to use this commend!
    load infor_id36.mat
    load storage_id36.mat
    if Trade_no==0
        list_betray = [];
    end
    if counterparty_action > 0
        list_betray = [list_betray; counterparty_id];
    end
    [m]=find(list_betray==counterparty_now);
    % if m is 'empty 0*0 double', then the counterparty_now had not been
    % betrayed you; otherwise, the counterparty_now had betrayed you at
    % least once
    if isempty(m)
        your_strategy = 0;      % if not been betrayed before, trade with this person
    else
        your_strategy = -1;     % otherwise, reject to trade
    end
    Trade_no = Trade_no + 1;
    save storage_id36.mat Trade_no your_id list_betray
end
```

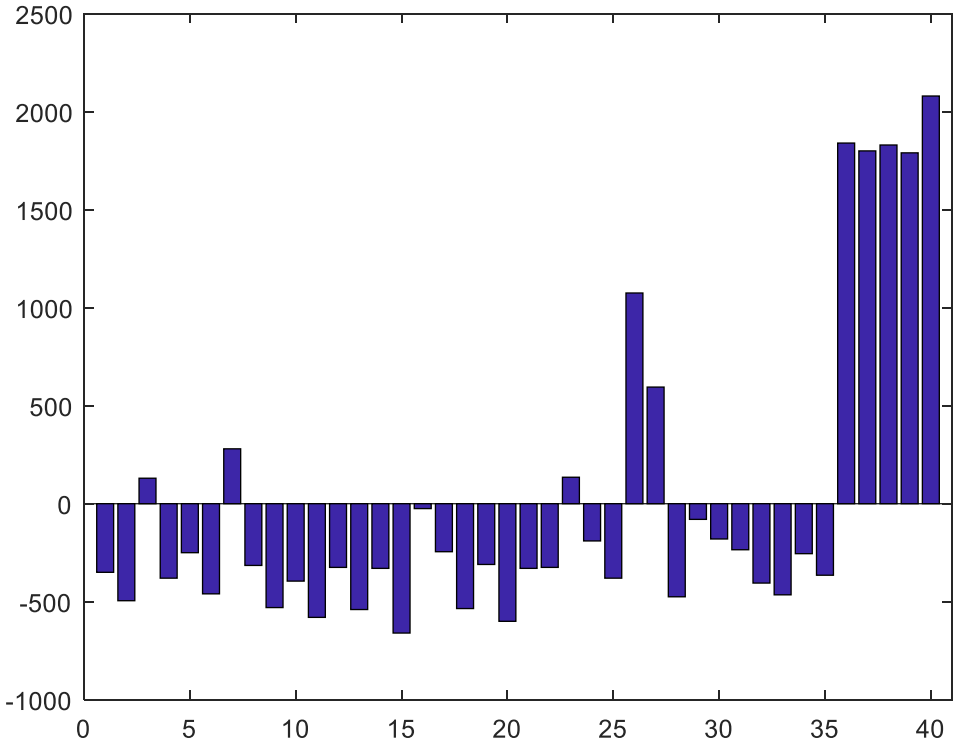
Experiment: the result

- The settings
  - The setting now: {X=5 and Y=5}, and {Repeated_trails=1, N_trades=1000}
  - 10 default strategies given: 5 'betray always' and 5 'reject after being betrayed', named as id31-35 and 36-40
  - Therefore, there are now 40 persons

```
Repeated_trails = 1;
N_trades = 1000;
N_persons = 40;
N_persons_pairs = round(N_persons/2);

X_betray_trust_point = 5;
Y_betray_betray_point = 5;
```

```
%% Now we begins
function [your_strategy] = id26(counterparty_id)
    your_strategy = 0;  % this means that you will trust this person
end
```



- It is not strange that id36-40 wins, as I design the 'reject after being betrayed' strategy based on the fact that many students betray a lot
- I hope that you can beat my 'reject after being betrayed' strategy under {X=5 and Y=5}, in this homework
- Note that, a strategy always trade will not as good as enough (but usable) this time, for example, id26