Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Axelrod's Tournament with Noise

Andermatt Samuel   Bösser Jonathan   Meier David

Zurich
Dec 2011

Agreement for free download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Andermatt Samuel

Bösser Jonathan

Meier David

Contents

Matlabcode

List of Figures

List of Tables

1 Abstract

2 Individual contributions

2.1 Andermatt Samuel

- Make the program object orientated
- Implement the "Players from Literature"

2.2 Bösser Jonathan

- Explore and explain GitHub [www.github.com]

2.3 Meier David

- Write the first version of the programm
- Develop and implement "Tit for Average tat"

3 Introduction and Motivations

3.1 The Prisoner's Dilema

The prisoner's dilema is a model from game theory. 2 people are suspected to have done a crime together. Now they are examined seperatly in differnt rooms. In this situation, they can either whistle-blowing the other person to protect oneself or keep silent. Over all, it is of advantage, if both keep silent. But for the single person it is better to betray the other person. The risk of betraying is the following: if both accaused people betray the other, the penalty for both is the highest. This problem is in gametheory called "Prisoner's dilema" [Quelle: http://plato.stanford.edu/entries/prisoner-dilemma].

3.2 The Axelrod Experiment

In the year 1981, Robert Axelrod invited for a competition to the iterated prisoner's dilemma. People from different fields like mathematics, politics, economy or psychology have been asked to develop a winning strategy for this competition. All the different strategies were playing against another to find the most sucsessive strategy. Interesstingly, the very simple strategy "Tit for Tat" (TFT) won the tournament. During the first round, TFT keeps silent (cooperation) and during the rest of the game, just does, what its counterplayer did the round before. This sort of experiment is very interessting, because the results can be applied in many different fields in real life. Just one out of many examples: 2 countries make an agreement on their amount of weapons. For the single country it is of advantage to haves more wmilitary strenth than the other nation. But as in the prisoner's dilema, if both nations rise their military strenth, for both it is just a loss money and an increase in danger. [Quelle: Buch: Axelrod R. "Die Evolution der Kooperation" etc... (Google-Books)]

3.3 Introduction of Noise

A further development in the Axelrod Experiment is the introduction of noise. This means, cooperation is wrongly understood as defection and vise versa. The introduction of noise to the axelrod experiment ist nothing new[Quelle!!].

4 Description of the Model and Players

4.1 Simple Players

4.1.1 Cooperative Player

The player 1 is a very simple player: He always cooperates. This "decision" does not depend on any circumstances like the decisions of its antagonist.

4.1.2 Defective Player

Also the player 2 is a very simple player: He always defects.

4.1.3 Random Player

Like all players from this subsection, the decision of the random player does not depend on the results of the previous tournaments. The decision is randomly distributed and no decision is preferred.

4.2 Players from Literatur

All players in this subsection are taken from the first Axelrod's Turnament and implemented by us. Source: Lecture "Game Theory" [Quelle, Zitierung?!?]

4.2.1 Tit for Tat

The Player 4 during to the Axelrod Turnament the most successive player of all[Quelle]. The decision is the decision of the counterplayer from the last tournament. In the first round, the decision is cooperation. If the counter player cooperated during the last round, this player will cooperate in the current round. www.socio.ethz.ch/vlib/pesb/pesb9.pdf

4.2.2 Friedmann

The Player "Friedmann" cooperates until its counter player defects once. After that, Friedmann now deflects for the rest of the game. This corresponds to "everlasting death".

4.2.3 Pavlov

Pavlov changes its decision every time when the counter player defects. But if the counter player cooperates, Pavlov gives the same decision as in the round before. The first decision is cooperation.

4.2.4 Tit for two Tat

The first decision is cooperation. If the counter player cooperates, Tit for 2Tat" cooperates as well. Tit for 2 Tat only defects, if the counter player defected the last 2 rounds.

4.2.5 Joss

This is basically the same player like the player "Tit for Tat". The only difference: 10% of the cooperative decisions are randomly defected. www.socio.ethz.ch/vlib/pesb/pesb9.pdf

4.2.6 Diekmann

The player "Diekmann" plays basically Tit for Tat. The difference is, that every 10th move, he playes cooperative twice. www.socio.ethz.ch/vlib/pesb/pesb9.pdf

4.3 Own Players

4.3.1 Tit for Average Tat

Based on the idea "Tit for Tat", we developed a player who averages the decisions of its opppnent over the last 5 Rounds. The first 5 rounds he plays Tit for Tat. To get a more forgiving player, he starts playing Tit for tat for 5 rounds.

	Player B cooperates	Player B defects
Player A cooperates	A:3 B:3	A:0 B:5
Player A defects	A:5 B:0	A:1 B:1

Table 1: Reward Matrix

4.3.2

5 Implementation

In the following table 1 is shown the payoff matrix applied in our program.

Payoff matrix Spielablauf Informationen, die die Spieler sehen knnten Art des Noises

6 Simulation Results and Discussion

Alle erfolgreichen Spieler spielen irgend auf eine art und weise tit for tat. Naja, trozdem evt. erwhnen, drauf eingehen

7 Summary and Outlook

8 References

References

[1] Huynen, M. A. and Bork, P. 1998. Measuring genome evolution. *Proceedings of the National Academy of Sciences USA* 95:5849–5856.

[2] Caprara, A. 1997. Sorting by reversals is difficult. In: *Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB 97),* New York: ACM. pp. 75-83.

[3] McLysaght, A., Seoighe, C. and Wolfe, K. H. 2000. High frequency of inversions during eukaryote gene order evolution. In Sankoff, D. and Nadeau, J. H., editors, *Comparative Genomics*, Dordrecht, NL: Kluwer Academic Press. pp. 47–58.

[4] Reinelt, G. 1991. *The Traveling Salesman - Computational Solutions for TSP Applications.* Berlin: Springer Verlag.
Buch: Axelrod R. "Die Evolution der Kooperation", deutsche Fassung 2000

A Matlabcode

A.1 show_data.m

Listing 1: show_data.m

```
%#################################################################################
% Content of file:
% 1.cell: initialization
% 2.cell: plot reward of all players with given noiselevels
% 3.cell: plot cooperation of all players with given noiselevels
% 4.cell: plot statistics for a given player (reward vs given noiselevels)
% 5.cell: plot statistics for a given player (cooperation vs given noiselevels)
% 6.cell: reward vs noise with name of the best player in plot
% 7.cell: total cooperation/reward normed
% 8.cell: 2 given Players against each other

% Use of file:
% 1. set filename of the simulation file in the 1. cell
% 2. set desired noiselevel in the 2.cell
% 3. set desired noiselevel in the 3.cell
% 4. set desired positions of players and desired noiselevels in the 4.cell
% 5. set desired positions of players and desired noiselevels in the 5.cell
% 6. set playersInRange true in the 6.cell to write the players in range
% in a textfile
% 7. set the filename for the players in range in the 6.cell
% 8. set the range in the 6.cell
% 9. set the filename in the 7.cell for the file with the 2 matrices
%10. set the desired players to face each other
%11. run the whole file

% hint: just run one cell, if only this result is desired
% warning: the more things you want to plot, the more plots you got

%##########################################################################

%% Initialize and get data of the simulation file

% Clear used Variables:
clear filename vars rewardMatrix numberOfPlayers numberOfTurns listOfPlayers ...
    noise averageCoop lengthOfNoise i j k


% Inputs:
filename = 'simulation.mat';               % name of the simulation-file

% Calc:
```

```
vars=load(filename);                       % load variables of the simulation-file
figureCounter = 1;                         % open new figure for each plot

rewardMatrix = vars.Rewardmatrix;          % store the rewardmatrix
numberOfTurns = vars.N;                    % store the numbers of turns
listOfPlayers = vars.Names;                % store the list of players
noise = vars.Noise;                        % store the noisematrix
averageCoop = vars.AverageCoop;            % store average cooperation
short = vars.Shorts;                       % store short names of players

numberOfPlayers = length(listOfPlayers);   % get the numbers of players
lengthOfNoise = size(noise,2);

% Convert rewardmatrix
for i = 1:numberOfPlayers               % generate empty matrices for every player
    eval(['R' int2str(i) '=zeros(lengthOfNoise,lengthOfNoise,numberOfPlayers);']);
    eval(['C' int2str(i) '=zeros(lengthOfNoise,lengthOfNoise,numberOfPlayers);']);
end

for i = 1:numberOfPlayers                % rewardematrix Ri(noise1,noise2,oponent)
    for k = 1:lengthOfNoise
        for j = 1:lengthOfNoise
            eval(['R' int2str(i) '(' int2str(k) ',' int2str(j) ',:)='...
                'rewardMatrix(' int2str(i) ',:,' int2str(k) ',' int2str(j) ');']);
        end
    end

end

for i = 1:numberOfPlayers                   % coopmatrix Ci(noise1,noise2,oponent)
    for k = 1:lengthOfNoise
        for j = 1:lengthOfNoise
           eval(['C' int2str(i) '(' int2str(k) ',' int2str(j) ',:)=averageCoop'...
                '(' int2str(k) ',' int2str(j) ',' int2str(i) ',:);']);
        end
    end

end


%% Plot Reward of all players with given noiselevels

% Clear used Variables:
clear noiseLevel tempRewardMatrix rewardVectors i k h lengthN

% Inputs:
```

```
noiseLevel = [1;...                         % Noise Level 1 (player --> opponent)
              1];                           % Noise Level 2 (opponent --> player)


% Calc:
tempRewardMatrix = zeros(numberOfPlayers);  % temporary rewardmatrix
lengthN = size(noiseLevel,2);        % number of diffrent noise level constellations
rewardVectors = zeros(lengthN,numberOfPlayers);% the reward vector for each
                                             % noise level constellation is saved


h = figure(figureCounter);                                  % initialize figure
set(h,'NumberTitle','off')
set(h,'Position',[10 500 1200 800])          % position and size of figure
set(h,'Name','Reward of all Players at given Noiselevels')  % set title of figure

for i = 1:lengthN                        % iterate over all noise lever constellations
    for k = 1:numberOfPlayers                   % iterate over all players
        eval(['tempRewardMatrix(' int2str(k) ',:)=R' int2str(k) ...
        '(noiseLevel(1,i),noiseLevel(2,i),:);']); % save reward of each player in
                                                  % temporary rewardmatrix
    end
    rewardVectors(i,:)=sum(tempRewardMatrix')/(numberOfTurns*numberOfPlayers);
                         % rewardvector of each noise level constellation is saved

    subplot(lengthN,1,i)                       % plotting options
    bar(rewardVectors(i,:));
    grid on;
    set(gca,'XTick',1:1:numberOfPlayers)
    set(gca,'XTickLabel',short,'FontSize',8)
    set(gca,'XLim',[0 numberOfPlayers+1])
    set(gca,'YLim',[max((min(rewardVectors(i,:))-0.25),0) min((max(...
        rewardVectors(i,:))+0.25),5)])
    title(['Noiseplot with Noiselevel 1: ',num2str(noise(1,noiseLevel(1,i))),...
        ', and Noiselevel 2: ', num2str(noise(2,noiseLevel(2,i)))],'FontWeight'...
        ,'bold','FontSize',12);
    xlabel('Players','FontWeight','bold','FontSize',10)
    ylabel('Average profit of Player','FontWeight','bold','FontSize',10)
end

figureCounter = figureCounter + 1;  % update figurecounter

%% Plot Cooperation of all players with given noiselevels

% Clear used Variables:
clear noiseLevel tempCoopMatrix coopectors i k h lengthN
```

```
% Inputs:
noiseLevel = [1;...                   % Noise Level 1 (player --> opponent)
              1];                     % Noise Level 2 (opponent −−> player)


% Calc:
tempCoopMatrix = zeros(numberOfPlayers);  % temporary rewardmatrix
lengthN = size(noiseLevel,2);               % number of diffrent noise level constellations
coopVectors = zeros(lengthN,numberOfPlayers); % the cooperation vector for each noise level constellation is saved


h = figure(figureCounter);                                % initialize figure
set(h,'NumberTitle','off')
set(h,'Position',[10 500 1200 800])         % position and size of figure
set(h,'Name','Cooperation of all Players at given Noiselevels')  % set title of figure

for i = 1:lengthN                           % iterate over all noise lever constellations
    for k = 1:numberOfPlayers               % iterate over all players
        eval(['tempCoopMatrix(' int2str(k) ',:)=C' int2str(k) '(noiseLevel(1,i),noiseLevel(2,i),:);']); % save cooperation of each player in temporary cooperation matrix
    end
    coopVectors(i,:)=mean(tempCoopMatrix,2); % coopvector of each noise level constellation is saved


    subplot(lengthN,1,i)                    % plotting options
    bar(coopVectors(i,:));
    grid on;
    set(gca,'XTick',1:1:numberOfPlayers)
    set(gca,'XTickLabel',short,'FontSize',8)
    set(gca,'XLim',[0 numberOfPlayers+1])
    set(gca,'YLim',[max((min(coopVectors(i,:))-0.05),0) min((max(coopVectors(i,:))+0.05),1)])
    title(['Noiseplot with Noiselevel 1: ',num2str(noise(1,noiseLevel(1,i))),', and Noiselevel 2: ', num2str(noise(2,noiseLevel(2,i)))],'FontWeight','bold','FontSize',12);
    xlabel('Players','FontWeight','bold','FontSize',10)
    ylabel('Cooperation of Player in %','FontWeight','bold','FontSize',10)
end
```

```
figureCounter = figureCounter + 1;  % update figurecounter

%% Plot statistics for a given player (Reward vs given Noiselevels)

% Clear used Variables:
clear position noiseLevel lengthN givenPlayers tempRewardMatrix k tempRewardVector i h

% Inputs:
position = [1];                 % Numbers of the players (hint: type listOfPlayers to see which player has which number)
noiseLevel = [1  ;...           % Noise Level 1
              1];               % Noise Level 2

% Calc:
givenPlayers = length(position);    % number of given players
lengthN = size(noiseLevel,2);       % number of given noise level constellations

tempRewardMatrix = zeros(givenPlayers,numberOfPlayers,lengthN); % temporary reward matrix
tempRewardVector = zeros(1,numberOfPlayers);

for i = 1:givenPlayers              % fill tempRewardMatrix(given Player, all opponents, given noise level)
    for k = 1:numberOfPlayers
        for l = 1:lengthN
            tempRewardMatrix(i,k,l) = eval(['R' int2str(position(i)) '(noiseLevel(1,l),noiseLevel(2,l),k);']);
        end
    end
end

for i = 1:givenPlayers              % iterate over all given players
    h = figure(i+figureCounter);                              % initialize figure
    set(h,'NumberTitle','off')
    set(h,'Position',[10 500 1000 900])         % position and size of figure
    set(h,'Name',['Reward of Player "' listOfPlayers{position(i)} '" against all Players at given Noiselevels'])  % set title of figure
    for k = 1:lengthN               % iterate over each noiselevel constellation
        tempRewardVector = tempRewardMatrix(i,:,k)/numberOfTurns; % take right vector out of the tempRewardMatrix

        subplot(lengthN,1,k)        % plotting options
        bar(tempRewardVector);
        grid ON;
        set(gca,'XTick',1:1:numberOfPlayers)
        set(gca,'XTickLabel',short,'FontSize',8)
        set(gca,'XLim',[0 numberOfPlayers+1])
        set(gca,'YLim',[max((min(tempRewardVector)-0.25),0) min((max(
          tempRewardVector)+0.25),5)])
        title(['Noiseplot with Noiselevel 1: ',num2str(noise(1,noiseLevel(1,k))),'
          and Noiselevel 2: ', num2str(noise(2,noiseLevel(2,k))),' for Player "'
          listOfPlayers{position(i)}, '"'],'FontWeight','bold','FontSize',12);
        xlabel('Opponents','FontWeight','bold','FontSize',10)
        ylabel(['Average profit of Player "' listOfPlayers{position(i)} '"'],'
          FontWeight','bold','FontSize',8)
    end
end

figureCounter = figureCounter + givenPlayers; % update figurecounter

%% Plot statistics for a given player (Cooperation vs given Noiselevels)

% Clear used Variables:
clear position noiseLevel lengthN givenPlayers tempCoopMatrix k tempCoopVector i h

% Inputs:
position = [1];                  % Numbers of the players (hint: type listOfPlayers
  to see which player has which number)
noiseLevel = [1  ;...           % Noise Level 1
              1];              % Noise Level 2

% Calc:
givenPlayers = length(position);    % number of given players
lengthN = size(noiseLevel,2);       % number of given noise level constellations

tempCoopMatrix = zeros(givenPlayers,numberOfPlayers,lengthN); % temporary
  cooperation matrix
tempCoopVector = zeros(1,numberOfPlayers);

for i = 1:givenPlayers              % fill tempCoopMatrix(given Player, all
  opponents, given noise level)
    for k = 1:numberOfPlayers
        for l = 1:lengthN
            tempCoopMatrix(i,k,l) = eval(['C' int2str(position(i)) '(noiseLevel(1,l
              ),noiseLevel(2,l),k);']);
        end
    end
end

for i = 1:givenPlayers              % iterate over all given players
    h = figure(i+figureCounter);                              % initialize figure
    set(h,'NumberTitle','off')
    set(h,'Position',[10 500 1000 900])         % position and size of figure
    set(h,'Name',['Cooperation of Player "' listOfPlayers{position(i)} '" against all Players at given Noiselevels'])  % set title of figure
    for k = 1:lengthN                   % iterate over each noiselevel constellation
        tempCoopVector = tempCoopMatrix(i,:,k); % take right vector out of the tempCoopMatrix

        subplot(lengthN,1,k)            % plotting options
        bar(tempCoopVector);
        grid ON;
        set(gca,'XTick',1:1:numberOfPlayers)
        set(gca,'XTickLabel',short,'FontSize',8)
        set(gca,'XLim',[0 numberOfPlayers+1])
        set(gca,'YLim',[max((min(tempCoopVector)-0.05),0) min((max(tempCoopVector)+0.05),1)])
        title(['Noiseplot with Noiselevel 1: ',num2str(noise(1,noiseLevel(1,k))),' and Noiselevel 2: ', num2str(noise(2,noiseLevel(2,k))),' for Player "' listOfPlayers{position(i)}, '"'],'FontWeight','bold','FontSize',12);
        xlabel('Opponents','FontWeight','bold','FontSize',10)
        ylabel(['Cooperation of Player "' listOfPlayers{position(i)} '"'],'FontWeight','bold','FontSize',8)
    end
end

figureCounter = figureCounter + givenPlayers; % update figurecounter

%% Reward vs Noise with name of the best player

% Clear used Variables:
clear positions h tempRewardMatrix value position player noiseLevel tempPositions endPositions endReward playersInRange range filename file

% Inputs:
playersInRange = true;  % true: calculate players in range, false, don't calculate players in range
range = 0.05;             % how close have other players be, to be mentioned
filename = 'range.txt'; % file, where players in range are saved

% Calc:
positions = zeros(lengthOfNoise^2,numberOfPlayers);                           % vector for player with maximum reward for given noise
tempRewardMatrix = zeros(lengthOfNoise^2,numberOfPlayers);    % create new temporary reward matrix

for i = 1:lengthOfNoise                          % iterate over all noise combinations
```

```
    for k = 1:lengthOfNoise
        for l = 1:numberOfPlayers                       % iterate over all players
            for m = 1:numberOfPlayers                   % iterate over all opponents
                tempRewardMatrix(k+(i-1)*lengthOfNoise,l) = tempRewardMatrix(k+(i-1)*lengthOfNoise,l) + eval(['R' int2str(l) '(' int2str(i) ',' int2str(k) ',' int2str(m)   ');']); % add temporary rewardmatrix (1,player)
            end
        end
    end
end

[value, position] = max(tempRewardMatrix'/(numberOfTurns*numberOfPlayers));     % take maximas

for i = 1:lengthOfNoise^2                       % fill positionmatrix
    positions(i,position(i)) =  value(i);
end

[noiseLevel ,player] = find(positions);
tempPositions = sortrows([noiseLevel player],1);

for i = 1:lengthOfNoise                         % get positions matrix and reward matrix ready for plotting
    for k = 1:lengthOfNoise
        endPositions(i,k) = tempPositions(k+(i-1)*lengthOfNoise,2);
        endReward(i,k) = positions(k+(i-1)*lengthOfNoise,tempPositions(k+(i-1)*lengthOfNoise,2));
    end
end

h = figure(figureCounter+1);                               % initialize figure
set(h,'NumberTitle','off')
set(h,'Position',[10 500 800 800])         % position and size of figure
set(h,'Name','Reward vs Noise with best Player named')  % set title of figure

colormap(winter)
imagesc(0:1:lengthOfNoise-1,0:1:lengthOfNoise-1,endReward)
set(gca,'XTick',0:1:lengthOfNoise)
set(gca,'YTick',0:1:lengthOfNoise)
set(gca,'XTickLabel',noise(1,:))
set(gca,'YTickLabel',noise(2,:))

for i = 1:lengthOfNoise
    for k = 1:lengthOfNoise
        text(k-1,i-1,...
        [listOfPlayers{endPositions(i,k)}],...
        'HorizontalAlignment','center','VerticalAlignment','bottom','FontWeight','
          bold','FontSize',12);
        text(k-1,i-1,...
        [num2str(endReward(i,k))],...
        'HorizontalAlignment','center','VerticalAlignment','top');


   end
end



if(playersInRange)                                                           %
  caluclate players in range:
    result=zeros(lengthOfNoise^2,numberOfPlayers);                           % empty
       matrix for position of players
    tempRewardMatrix = tempRewardMatrix./(numberOfPlayers*numberOfTurns);    % norm
      tempRewardMatrix
    lowerValue = endReward .* (1-range);                                     %
      calculate lower value
    for i = 1:lengthOfNoise                                                  %
      iterate over all noise levels
        for k = 1:lengthOfNoise
            clear tempResult
            tempResult = find(tempRewardMatrix(k+(i-1)*lengthOfNoise,:)>=lowerValue
              (i,k));  % find players in range
            result(k+(i-1)*lengthOfNoise,1:length(tempResult)) = tempResult;
        end
    end


file = fopen(filename,'w');                                                  % open
  file with given filename
fprintf(file, 'Players in a %1.2f range for each noise level \n\n',range);   % print
   header

for i = 1:lengthOfNoise                                                      % print
   file
    for k = 1:lengthOfNoise
        fprintf(file, 'Noise level 1: %1.2f, Noise level 2: %1.2f, highest reward:
          %1.4f, in range (>%1.4f):\n',noise(1,i),noise(2,k),endReward(i,k),
          lowerValue(i,k));
        for l=find(result(k+(i-1)*lengthOfNoise,:))
            fprintf(file, '%s (%1.4f)\n',listOfPlayers{result(k+(i-1)*lengthOfNoise
              ,l)},tempRewardMatrix(k+(i-1)*lengthOfNoise,result(k+(i-1)*
              lengthOfNoise,l)));
        end
        fprintf(file, '\n');
    end
end

fclose(file);                                                                    % close file

end
figureCounter = figureCounter + 2;                                               % update figurecounter


%% Total Cooperation/Reward normed

% Clear used Variables:
clear i k totalReward totalCoop tempTotalCoop filename file

% Inputs:
filename = 'totalresult.txt';                                              % filename of file for total results

% Calc:

totalReward = zeros(lengthOfNoise);                                       % create total reward matrix
totalCoop = zeros(lengthOfNoise);                                         % create total cooperation matrix

for k=1:numberOfPlayers                                                   % iterate over all players
    for i=1:numberOfPlayers                                               % calculate total reward matrix
        totalReward(:,:)=totalReward(:,:)+eval(['R' int2str(k) '(:,:,' int2str(i) ')'  ';'])/(numberOfPlayers*numberOfTurns*numberOfPlayers);

    end
    for i=1:lengthOfNoise                                                 % calculate temporary total cooperation matrix
        for j=1:lengthOfNoise
            tempTotalCoop(i,j,k)=mean(eval(['C' int2str(k) '(i,j,:)'  ';']));
        end
    end
end
```

```
for l=1:lengthOfNoise                                        % calculate total
  cooperation matrix
    for j=1:lengthOfNoise
        totalCoop(l,j)=mean(tempTotalCoop(l,j,:));
    end
end

file = fopen(filename,'w');                                  % open file with given
  filename
fprintf(file, 'Total Rewardmatrix: \n\nNoise ',range);       % print header for
  rewardmatrix

fprintf(file, '| %1.2f ',noise(1,:));                        % print reward matrix
fprintf(file, '\n -----|');
for k=1:lengthOfNoise
    for i = 1:lengthOfNoise
        fprintf(file, '-------');
    end
    fprintf(file, '\n %1.2f ',noise(2,k));
    fprintf(file, '| %1.2f ',totalReward(k,:));
    fprintf(file, '\n -----|');
end
for i = 1:lengthOfNoise
    fprintf(file, '-------');
end

fprintf(file, '\n\nTotal Cooperatiomatrix: \n\nNoise ',range);  % print header for
  coopmatrix

fprintf(file, '| %1.2f   ',noise(1,:));                      % print coopmatrix
fprintf(file, '\n -----|');
for k=1:lengthOfNoise
    for i = 1:lengthOfNoise
        fprintf(file, '---------');
    end
    fprintf(file, '\n %1.2f ',noise(2,k));
    fprintf(file, '| %1.4f ',totalCoop(k,:));
    fprintf(file, '\n -----|');
end
for i = 1:lengthOfNoise
    fprintf(file, '---------');
end

fclose(file);                                                % close file

%% 2 given Players against each other
```

```
% Clear used Variables:
clear players shortTemp tempRewardMatrix l k i

% Inputs:
player = [1 ;...                % player 1
          1 ] ;                 % player 2


% Calc:
players = size(player,2);           % number of faceoffs
tempRewardMatrix = zeros(lengthOfNoise^2,2,players);     % create temporary rewardmatrix(noiselevel,2,faceoff)


for l = 1:players                   % create rewardmatrix
    for i = 1:lengthOfNoise
        for k = 1:lengthOfNoise
            tempRewardMatrix(k+(i-1)*lengthOfNoise,1,l) = eval(['R’ int2str(player(1,l)) ’(’ int2str(i) ’,’ int2str(k) ’,’ int2str(player(2,l)) ’);’])/numberOfTurns;
            tempRewardMatrix(k+(i-1)*lengthOfNoise,2,l) = eval(['R’ int2str(player(2,l)) ’(’ int2str(i) ’,’ int2str(k) ’,’ int2str(player(1,l)) ’);’])/numberOfTurns;
        end
    end
end

for l = 1:players                  % iterate over faceoffs
    h = figure(l+figureCounter);                              % initialize figure
    set(h,’NumberTitle’,’off’)
    set(h,’Position’,[10 500 1600 900])          % position and size of figure
    set(h,’Name’,[’"’ listOfPlayers{player(1,l)} ’" against "’ listOfPlayers{player(2,l)} ’ and vice versa’])  % set title of figure
    for i = 1:lengthOfNoise
        for k = 1:lengthOfNoise
            subplot(lengthOfNoise, lengthOfNoise, k+(i-1)*lengthOfNoise)
            bar(tempRewardMatrix(k+(i-1)*lengthOfNoise,:,l))
            grid ON;
            set(gca,’XTick’,1:1:2)
            shortTemp{1} = short{player(1,l)};
            shortTemp{2} = short{player(2,l)};
            set(gca,’XTickLabel’,shortTemp,’FontSize’,8)
            set(gca,’XLim’,[0 3])
            set(gca,’YLim’,[max((min(tempRewardMatrix(k+(i-1)*lengthOfNoise,:,l))-0.25),0) min((max(tempRewardMatrix(k+(i-1)*lengthOfNoise,:,l))+0.25)
              ,5)])
            title(['Noiselv 1: ',num2str(noise(1,k)),' and Noiselv 2: ', num2str(
              noise(1,i)),],'FontWeight','bold','FontSize',12);
            xlabel('Opponents','FontWeight','bold','FontSize',10)
            ylabel(['Reward'],'FontWeight','bold','FontSize',8)
        end
    end
end
```