ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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

Project Report

Modeling International Negotiations with a Game Theoretical model of Trust

Jan-Aurel Pfister

Zurich
Fall semester 2017

Agreement for free-download

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

Jan-Aurel Pfister

Contents

1 Abstract **4**

2 Individual contributions **5**

3 Introduction and Motivations **6**

4 Description of the Model **7**

5 Implementation **10**

6 Simulation Results and Discussion **18**

7 Summary and Outlook **22**

8 References **23**

1 Abstract

International cooperation is crucial for keeping the peace in the world. Although there is no central international authority to enforce agreements, states seem to obey them by free will and not only due to a constant threat of war and retaliation.
Some states could be better off by not following some agreements but mostly do follow the agreements anyway, they do so, because they fear the loss of trust from the other states. Trust allows for cooperation since the perceived risk of betrayal, which leaves the cooperator worse off, is reduced. Trust is also dependent on specific actions a state, for example espionage. Both, specific actions and counteractions and international agreements can be modeled by game theory.

By implementing a game theoretical model for trust in matlab, I can identify certain crucial conditions critical for first interactions to allow for a continuously cooperative relationship.

2 Individual contributions

The Model and all equations have been developed and implemented by Jan-Aurel Pfister. The Report was written by Jan-Aurel Pfister as well. This Report will be complemented and completed end of December 2017 with a Policy Analysis Report about the real world application of the equations. With focus on negotiations and relationship between the United States of America and Germany.

3 Introduction and Motivations

Trust on intergovernmental level is crucial for emergence of cooperation if no authority is present to enforce a contract or agreement.
To understand the decision processes and why negotiations about contracts are successful or a failure, one needs to understand not only rational decisions that are considered by game theory, but also emotional influences on rational decisions. One of these emotional influences is trust, which will be modeled and compared to the most used and accepted strategies of game theoretical decision making.The Trust model has to interact with itself, the tit for tat strategy, the always cooperate strategy, the always defect strategy, the win stay lose shift strategy and a purely random strategy. Most strategies assume an initial cooperation, I will make the same assumption and will try to identify the initial conditions for a cooperative relationship to be successful.
Furthermore I will in a later stage compare the model to reality, by analyzing international contracts between countries or alliances. By analyzing the history of the relationships, I will try to assign payoffs to the different contracts and agreements and compare real world negotiation outcomes with the prediction of my model.

The aim of this report and model would be to reliably and quantitatively analyze international negotiations and understand why certain countries are more likely to cooperate whereas others are rather unlikely to cooperate.

4 Description of the Model

I decided to model trust in dependence of the payoff sum over all iterations, the payoff of the last iteration (n-1) and the value of trust in the last iteration beginning with trust=1. Since a country or a negotiator cannot negotiate with it- or himself, I used the ratios of the payoff sums and the payoffs of the two players to update the trust function. This allows me to fulfill certain conditions, that I believe to be crucial:

1. Gain and loss of trust is not only dependent on payoff but also on magnitude of difference between payoffs
2. As long as the payoff is positive, trust rises.
3. Trust rises faster when the other player sacrifices (Payoff 1>0, Payoff 2<0) himself for your gain
4. Betrayal (Payoff 1<0, Payoff 2>0) leads to faster decrease of trust than the rise due to sacrifice.

The equations are therefore underlying the following conditions.

If PlayerPayoff 1>0 and PlayerPayoff2>0 or PlayerPayoff1<0 and PlayerPayoff2<0

$$TP1(n) = TP1(n-1) + \frac{PP1 * \frac{|PP1|}{|PP2|} * \frac{|sumPP1|}{|sumPP2|}}{|Min/MaxPayoff|} \tag{1}$$

If PlayerPayoff 1>0 and PlayerPayoff2<0

$$TP1(n) = TP1(n-1) + \frac{(PP1 - PP2) * \frac{|PP1|}{|PP2|} * \frac{|sumPP1|}{|sumPP2|} * (1.5)}{|MaxPayoff|} \tag{2}$$

If PlayerPayoff 1<0 and PlayerPayoff2>0

$$TP1(n) = TP1(n-1) + \frac{(PP2 - PP1) * \frac{|PP1|}{|PP2|} * \frac{|sumPP2|}{|sumPP1|} * (-2)}{|MaxLoss|} \tag{3}$$

These equations change the value of trust. Trust lowers the perceived risk of betrayal, this updates the game theory grid. Since the Nash equilibrium assumes players to minimize loss, I decided to update the value of loss if player 1 cooperates and player 2 defects. That value will be divided by the calculated value of trust. If if this payoff becomes smaller than the payoff for the case where both players defect, the trust strategy player is willing to cooperate. This approach allows to forgive minor mistakes depending on the previous payoffs and the magnitude of loss due to the mistake.

Specific description of the equations

1. Equation one shows us the trust of player 1 if both players had a positive or negative payoff. PP1 determines the if trust grows or shrinks, since all other variables in numerator and denominator are absolute. Trust is a function of the payoff (PP1), the ratio of payoffs
(PP1(n-1)/PP2(n-1)) and the ratio of all payoffs (sumPP1/sumPP2) all in relation to the maximum or minimum payoff of the last round. If the sumPP1 it does not matter if the sumPP1 would be negative, because that would already be included in TP1(n-1).

2. Equation two shows us the trust of player 1 if player 2 sacrifices himself. Trust is a function of the difference of the payoffs (PP1-PP2) the ratio of the payoffs(PP1/PP2) and the ratio of all payoffs(sumPP1/sumPP2) all in relation to the maximum payoff of the last round. It is important to understand, that the ratio PP1/PP2 is dependent on the utility of both. Meaning that a very low payoff for player 2 and only a small payoff for player 1 lets trust rise slower. Since the action taken by player 2 which sacrifices himself for only a small gain for player 1 does make him not rational and his actions are therefore harder to predict. Therefore the trust is not rising as fast. However small sacrifices that lead to large payoff gains of player 1 increase trust drastically.

3. Equation three shows us the trust of player 1 if he is betrayed by player 2. This situation arises if the Payoff of player 1 negative if the payoff of player 2 is positive. Trust is a function of the difference of the payoffs (PP1-PP2) the ratio of the payoffs(PP1/PP2) and the ratio of all payoffs(sumPP2/sumPP1) all in relation to the maximum payoff of the last round. The ratio of the sum of payoffs has changed from (sumPP1/sumPP2) to (sumPP2/sumPP1) in regard to the equation two. Since the last term (-2) lowers the trust and makes the whole fraction negative, it is important to adjust for differences in the sum of all payoffs for the case that the sum of the payoffs for player 1 is higher than the sum of the payoffs for player 2. If we would calculate the value of trust with (sumPP1/sumPP2), a lower trust value would be the result since we are amplifying the negative event of a betrayal. However we should do the exact opposite, which means that due to our positive results in the past we are more likely to forgive a minor error or betrayal of the counter party. This leads us to (sumPP2/sumPP1).

Variable Legend:
PP1: Payoff of player 1 (per round)
PP2: Payoff of player 2 (per round)
sumPP1: Sum of payoffs for player 1
sumPP2: Sum of payoffs for player 2
TP1: Trust Player 1
n: This round
n-1: Last round
MaxPayoff: Maximum Payoff in the previous round
MaxLoss: Maximum Loss in the previous round

5 Implementation

Simulation Input Here the user can decide what conditions (fixed or random payoff matrix) he wants to use, furthermore the simulation can be chosen.

TrustMaster.m

```
function [SSP1, SSP2]= TrustMaster(s,s2);
%script input
 strs  = {'1. Trust strategy versus Trust strategy', '2. Trust strategy vs Tit for Tat st
 prmpt = 'Select a simulation you want to run:';
 [s,v] = listdlg('PromptString',prmpt,'SelectionMode','multi','ListString',strs);
input where the matrix type can be selected
 strs  = {'1. Random Payoff Matrix', '2. Fixed Payoff Matrix'};
 prmpt = 'Select a simulation you want to run:';
 [s2,v2] = listdlg('PromptString',prmpt,'SelectionMode','multi','ListString',strs);
```

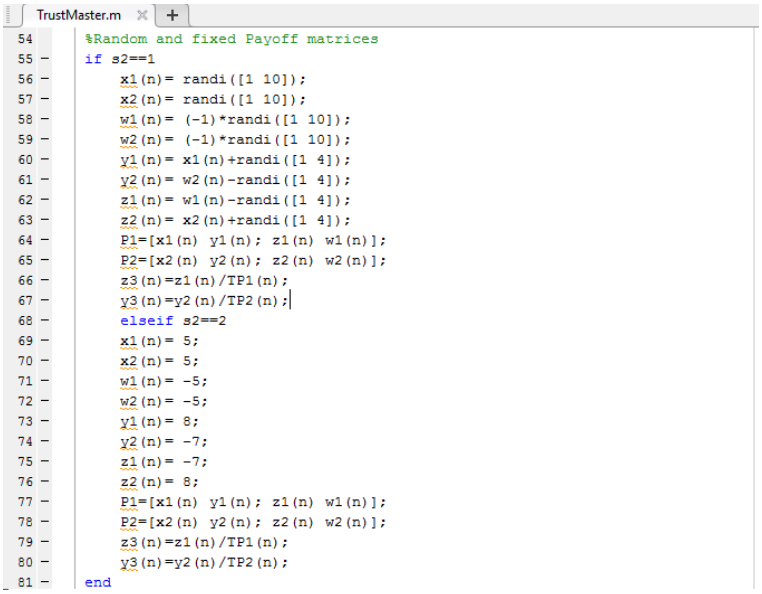
Trust equations
The equations for trust calculation are implemented as follows:

TrustMaster.m

```
%for loop for trust function update
for n=2:10
    %here the equations for the trust function are implementend for different conditions
    if PP1(n-1)>0 && PP2(n-1)>0
        TP1(n)=TP1(n-1)+(PP1(n-1)*abs(PP1(n-1))/abs(PP2(n-1))*sum(PP1)/sum(PP2))/y1(n-1)
        TP2(n)=TP2(n-1)+(PP2(n-1)*abs(PP2(n-1))/abs(PP1(n-1))*sum(PP2)/sum(PP1))/abs(z2(n-1))

    elseif PP1(n-1)<0 && PP2(n-1)<0
        TP1(n)=TP1(n-1)+(PP1(n-1)*abs(PP1(n-1))/abs(PP2(n-1))*sum(PP1)/sum(PP2))/abs(z1(n-1))
        TP2(n)=TP2(n-1)+(PP2(n-1)*abs(PP2(n-1))/abs(PP1(n-1))*sum(PP2)/sum(PP1))/abs(z2(n-1))

    elseif PP1(n-1)>0 && PP2(n-1)<0
       TP1(n)=TP1(n-1)+((PP1(n-1)-PP2(n-1))*abs(PP1(n-1))/abs(PP2(n-1))*sum(PP1)/sum(PP2)*(-1.5)/y1(n-1))
       TP2(n)=TP2(n-1)+((PP1(n-1)-PP2(n-1))*abs(PP2(n-1))/abs(PP1(n-1))*sum(PP1)/sum(PP2)*(-2)/abs(y2(n-1)))

    elseif PP1(n-1)<0 && PP2(n-1)>0
       TP1(n)=TP1(n-1)+((PP2(n-1)-PP1(n-1))*abs(PP1(n-1))/abs(PP2(n-1))*sum(PP2)/sum(PP1)*(-2)/abs(z1(n-1)))
       TP2(n)=TP2(n-1)+((PP2(n-1)-PP1(n-1))*abs(PP2(n-1))/abs(PP1(n-1))*sum(PP2)/sum(PP1)*(-1.5)/z2(n-1))
    end
```

Random vs Fixed Payoff Matrix

The script chooses between fixed and random matrix type dependent on the user input.



```
TrustMaster.m
54  %Random and fixed Payoff matrices
55  if s2==1
56      x1(n)= randi([1 10]);
57      x2(n)= randi([1 10]);
58      w1(n)= (-1)*randi([1 10]);
59      w2(n)= (-1)*randi([1 10]);
60      y1(n)= x1(n)+randi([1 4]);
61      y2(n)= w2(n)-randi([1 4]);
62      z1(n)= w1(n)-randi([1 4]);
63      z2(n)= x2(n)+randi([1 4]);
64      P1=[x1(n) y1(n); z1(n) w1(n)];
65      P2=[x2(n) y2(n); z2(n) w2(n)];
66      z3(n)=z1(n)/TP1(n);
67      y3(n)=y2(n)/TP2(n);
68      elseif s2==2
69      x1(n)= 5;
70      x2(n)= 5;
71      w1(n)= -5;
72      w2(n)= -5;
73      y1(n)= 8;
74      y2(n)= -7;
75      z1(n)= -7;
76      z2(n)= 8;
77      P1=[x1(n) y1(n); z1(n) w1(n)];
78      P2=[x2(n) y2(n); z2(n) w2(n)];
79      z3(n)=z1(n)/TP1(n);
80      y3(n)=y2(n)/TP2(n);
81  end
```

Decision Tree for Strategies based on Input

The user input selects the strategy from the decision tree and passes the strategy for each player to the payoff calculator.

TrustMaster.m

```
%SP1= Strategy Player 1; SP2= Strategy Player 2
%Cooperate=1, Defect=0
if s==1
SP1=TrustvTrust(z3,w1,y3,w2,n,SP1,SP2);
SP2=TrustvTrust(z3,w1,y3,w2,n,SP1,SP2);
    elseif s==2
    SP1=TitForTat(z3,w1,SP1,n,SP2);
    SP2=TitForTat(z3,w1,SP1,n,SP2);
            elseif s==3
            SP1=WSLS(z3,w1,SP2,PP2,SP1,n);
            SP2=WSLS(z3,w1,SP2,PP2,SP1,n);
                elseif s==4
                SP1=ADefect(z3,w1,n,SP1,SP2);
                SP2(n)=0;
                    elseif s==5
                    SP1=ACooperate(z3,w1,SP1,SP2,n);
                    SP2(n)=1;
                        elseif s==6
                        SP1=ACooperate(z3,w1,SP1,SP2,n);
                        SP2(n)=randi([0,1]);
                            elseif s==7
                            SP1=WSLS2(SP1,PP1,n);
                            SP2(n)= randi([0,1]);
                                elseif s==8
                                SP1(n)= randi([0,1]);
                                SP2=WSLS(z3,w1,SP2,PP2,SP1,n);
end
```

Trust versus Trust

The Trust versus Trust game is implemented as follows.

TrustvTrust.m

```
%SP1= Strategy Player 1; SP2= Strategy Player 2
%Trust versus Trust
%Cooperate=1, Defect=0
function [SP1, SP2]=TrustvTrust(z3,w1,y3,w2,n,SP1,SP2)
if z3(n) >= w1(n)
    SP1(n)= 1;
else
    SP1(n)= 0;
end

if y3(n) >= w2(n)
    SP2(n)= 1;
else
    SP2(n)= 0;
end
```

Trust versus Tit for Tat

The Trust versus Tit for Tat game is implemented as follows.

TitForTat.m

```
%SP1= Strategy Player 1; SP2= Strategy Player 2
%Trust versus Tit for Tat
%Cooperate=1, Defect=0
function [SP1, SP2]= TitForTat(z3,w1,SP1,n,SP2)
if z3(n)>w1(n)
    SP1(n)=1;
else SP1(n)=0;
end

if SP1(n-1)==1
    SP2(n)=1;
else SP2(n)=0;
end
```

Trust versus Win-stay Lose-shift

The Trust versus Win-stay Lose-shift game is implemented as follows.

WSLS.m

```
%SP1= Strategy Player 1; SP2= Strategy Player 2
%Trust versus Win-stay Lose-shift
%Cooperate=1, Defect=0
function [SP1, SP2]= WSLS(z3,w1,SP2,PP2,SP1,n)
if z3(n)>w1(n)
    SP1(n)=1;
else SP1(n)=0;
end

if PP2(n-1)<0 && SP2(n-1)==1
    SP2(n)=0;
elseif PP2(n-1)<0 && SP2(n-1)==0
    SP2(n)=1;
elseif PP2(n-1)>0 && SP2(n-1)==1
        SP2(n)=1;
elseif PP2(n-1)>0 && SP2(n-1)==0
    SP2(n)=0;
end
```

Trust versus Always Defect

The Trust versus always defect game is implemented as follows.

ADefect.m

```
%SP1= Strategy Player 1; SP2= Strategy Player 2
%Trust versus Always Defect
%Cooperate=1, Defect=0function [SP1, SP2]= ADefect(z3,w1,n,SP1,SP2)
if z3(n)>w1(n)
    SP1(n)=1;
else
    SP1(n)=0;
end
SP2(n)=0;
end
```

Trust versus Always Cooperate

The Trust versus always cooperate game is implemented as follows.

ACooperate.m

```
%SP1= Strategy Player 1; SP2= Strategy Player 2
%Trust versus Always Cooperate
%Cooperate=1, Defect=0
function [SP1, SP2]= ACooperate(z3,w1,SP1,SP2,n)
if z3(n)>w1(n)
    SP1(n)=1;
else SP1(n)=0;
end
SP2(n)=1;
```

Trust versus Random Strategy

The Trust versus random strategy game is implemented as follows.

zuf.m

```
%SP1= Strategy Player 1; SP2= Strategy Player 2
%Trust versus Random
%Cooperate=1, Defect=0
function [SP1, SP2]= zuf(z3,w1,n,SP1,SP2)
if z3(n)>=w1(n)
    SP1(n)=1;
else SP1(n)=0;
    SP2(n)=randi([0, 1]);
end
```

Tit for Tat versus Random Strategy

The Tit for Tat versus random strategy game is implemented as follows.

TitForTat2.m

```
%SP1= Strategy Player 1; SP2= Strategy Player 2
%Tit for Tat versus Random
%Cooperate=1, Defect=0
function [SP1]= TitForTat(SP1,n,SP2);
if SP2(n-1)==1
    SP1(n)=1;
else SP1(n)=0;
end
```

Win-stay Lose-shift versus Random Strategy

The Win-stay Lose-shift versus random strategy game is implemented as follows.

WSLS2.m

```
%SP1= Strategy Player 1; SP2= Strategy Player 2
%Win-stay Lose-Shift versus Random
%Cooperate=1, Defect=0
function [SP1]= WSLS2(SP1,PP1,n)
if PP1(n-1)<0 && SP1(n-1)==1
    SP1(n)=0;
elseif PP1(n-1)<0 && SP1(n-1)==0
    SP1(n)=1;
elseif PP1(n-1)>0 && SP1(n-1)==1
        SP1(n)=1;
elseif PP1(n-1)>0 && SP1(n-1)==0
    SP1(n)=0;
end
```

Payoff Calculation

The payoff calculation and plotting is executed as follows.

TrustMaster.m

```
%Finding Nash Equlibrium with updated values, 0=Defect, 1=Cooperate
[PP1, PP2]=Payoffs(x1,x2,w1,w2,y1,y2,z1,z2,SP1,SP2,n,PP1,PP2);
    end
%Finding Payoff Sum over the whole game.
d = [sum(PP1)  sum(PP2)];
figure(1)
b = bar(d)
set(gca, 'XTick', [1 2])
set(gca, 'XTickLabel', {'Sum Payoffs Player1' 'Sum Payoffs Player2'})
SSP1=sum(PP1)
SSP2=sum(PP2)
%Plotting Payoff fluctuations
Baseline(n)=0;
figure(2)
plot(1:1:n, PP1);
hold on;
plot(1:1:n, PP2);
hold on;
plot(1:1:n, Baseline);
hold on;
ylim([-15,15]);
```

Payoffs.m

```
%Finding Nash Equlibrium with updated values, 0=Defect, 1=Cooperate

function[PP1, PP2]=Payoffs(x1,x2,w1,w2,y1,y2,z1,z2,SP1,SP2,n,PP1,PP2)

if SP1(n)==1 && SP2(n)==1
    PP1(n)=x1(n);
    PP2(n)=x2(n);
    elseif SP1(n)==1 && SP2(n)==0
        PP1(n)=z1(n);
        PP2(n)=z2(n);
        elseif SP1(n)==0 && SP2(n)==1
            PP1(n)=y1(n);
            PP2(n)=y2(n);
            elseif SP1(n)==0 && SP2(n)==0
                PP1(n)=w1(n);
                PP2(n)=w2(n);
end
end

```

Result Plots

Plots from the Result section created as follows. Note that the Input and the Plots in the Trust Master script need to be set as comments in order to have an automatic calculation in the following script.

Payoff_many.m

```
% Negotiation outcome series of 10 iterated 100 times. This makes sure that
% differences in the initial buildup of of Trust is taken into account!
for i=1:100
    s=5
    s2=1
SSP1(i)= TrustMaster(s,s2);
SSP2(i)=TrustMaster(s,s2);

d2 = [sum(SSP1)  sum(SSP2)];
figure(1)
b = bar(d2)
set(gca, 'XTick', [1 2])
set(gca, 'XTickLabel', {'Trust' 'Random'})
title('Strategy1 vs Strategy2, Fixed or Random Payoff')
SSP1=sum(SSP1)
SSP2=sum(SSP2)
end

```

6 Simulation Results and Discussion

The simulation shows, that the Tit for Tat strategy is best to maximize the own payoff. However the Win-stay Lose-shift strategy and the Trust strategy both perform better when one player wants to maximize its payoff and minimize the payoff of the other player. Meaning that the difference in payoffs between the player using the trust or the win-stay lose-shift strategies and the payoffs of the random strategy player are higher. None of the Payoffs are significantly different from each other on the 95 percent level. To complete the analysis, the strategies were run against each other. Under stable cooperation conditions it makes sense that a continous cooperation is accomplished, furthermore it is also logical that under unstable trust conditions, the players switch to a continuous relationship of defection.

If we include the following stability condition $j[V(i,i) \geq V(j,i)]$ that is stated by Axelrod, which assumes a strategy i to be stable if the payoff for i playing against i is higher than the payoff for j playing against i. [3] It can be inferred from figure 2, that the Trust strategy is stable against Tit for Tat, Win-stay lose shift and Random strategy (in a fixed payoff scheme). This is expected from the design of the Trust model. Since the first interaction is set to be cooperative, it is dependent on the Payoff for the Trust strategy player of the first round. For the generic Payoff grid:

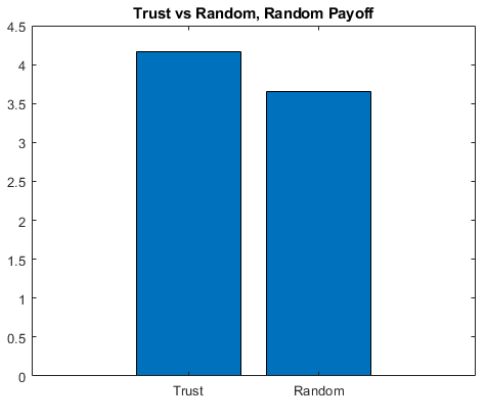
Player 1 /Player 2	Cooperate	Defect
Cooperate	A_1 , A_2	B_1 , B_2
Defect	C_1 , C_2	D_1 , D_2

the payoff for the trust strategy player (Player 1) has to follow the following condition to allow for continuous cooperation with a Tit for Tat or a Win-stay Lose-shift player.

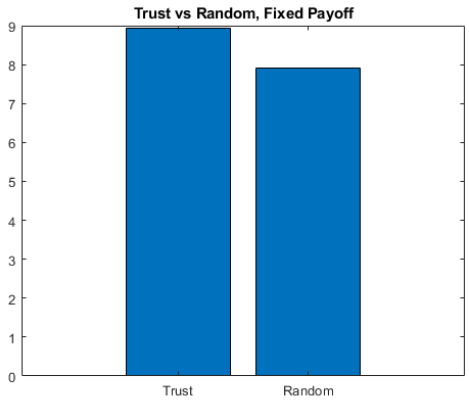
$$\frac{C1}{TP1(n-1)} \geq D1 \tag{4}$$

A continuous cooperation can be assumed if played against Tit for Tat and Win-stay Lose-shift strategy. For Trust vs Trust based games, an additional condition for Player 2 has to be formulated which is similar to the condition for Player 1:

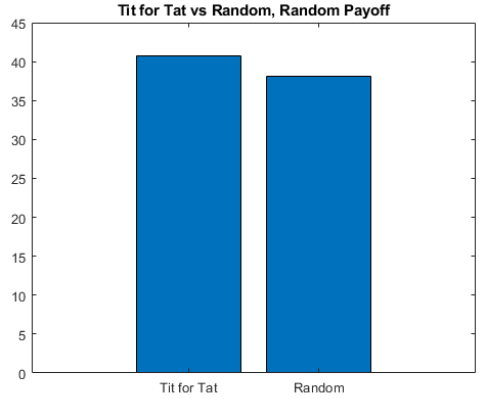
$$Player1 : \frac{C1}{TP1(n-1)} \geq D1 \quad Player2 : \frac{B1}{TP2(n-1)} \geq D2 \tag{5}$$



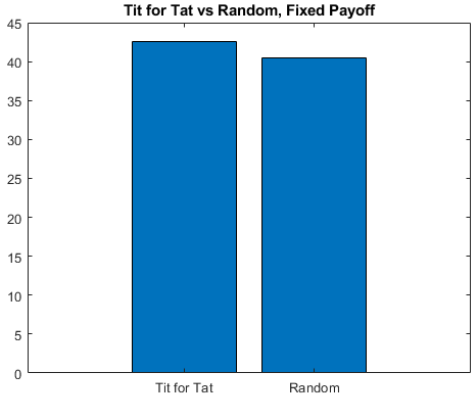
(a)



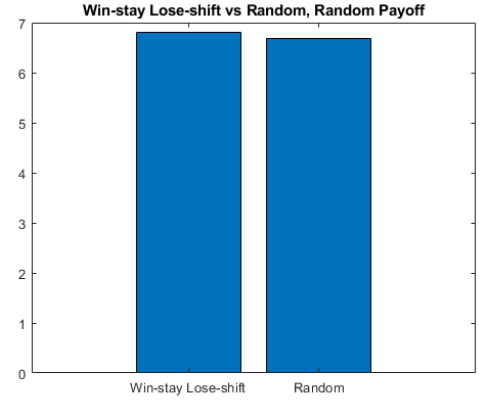
(b)



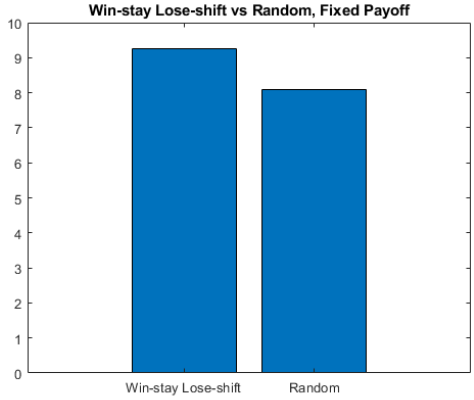
(c)



(d)

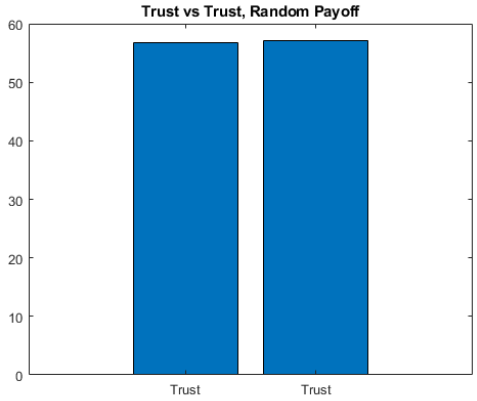


(e)

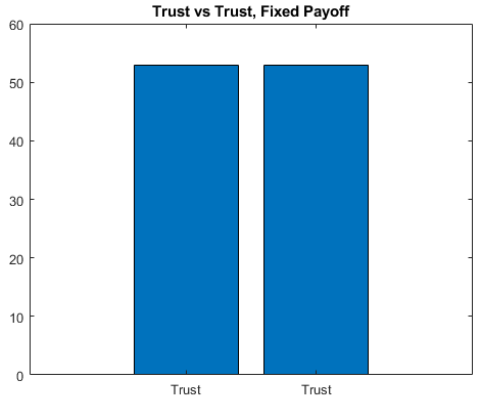


(f)

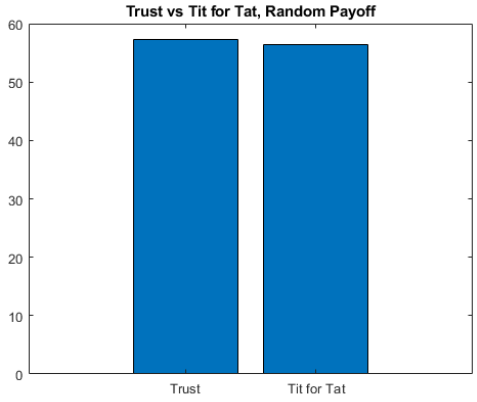
Figure 1: Payoff comparison for different strategies against a random strategy


Trust vs Trust, Random Payoff
60
50
40
30
20
10
0
Trust
Trust

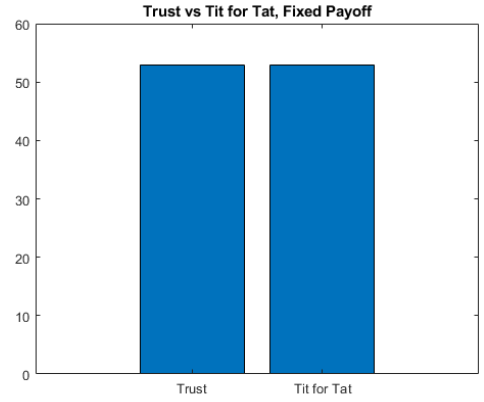
(a)


Trust vs Trust, Fixed Payoff
60
50
40
30
20
10
0
Trust
Trust

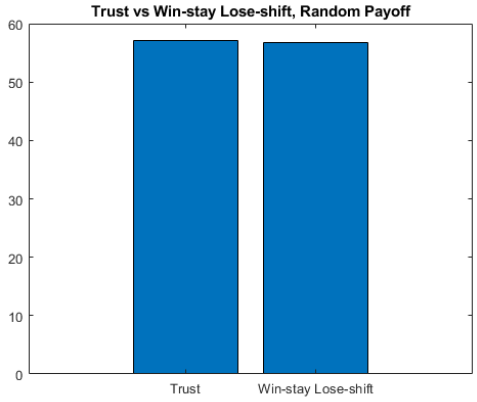
(b)


Trust vs Tit for Tat, Random Payoff
60
50
40
30
20
10
0
Trust
Tit for Tat

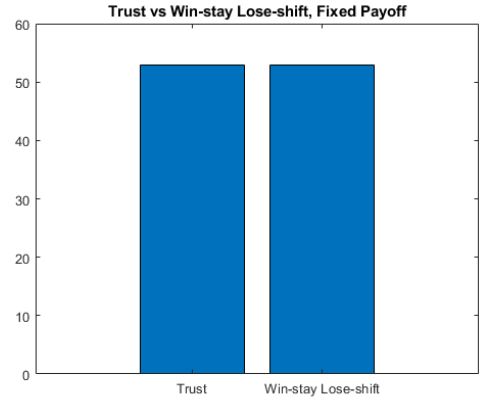
(c)


Trust vs Tit for Tat, Fixed Payoff
60
50
40
30
20
10
0
Trust
Tit for Tat

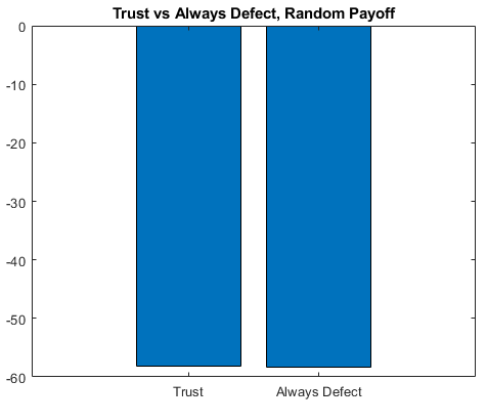
(d)


Trust vs Win-stay Lose-shift, Random Payoff
60
50
40
30
20
10
0
Trust
Win-stay Lose-shift

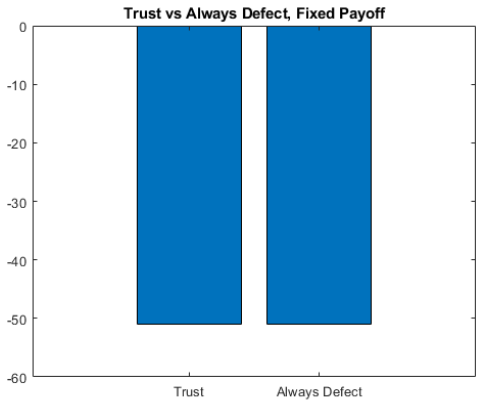
(e)


Trust vs Win-stay Lose-shift, Fixed Payoff
60
50
40
30
20
10
0
Trust
Win-stay Lose-shift

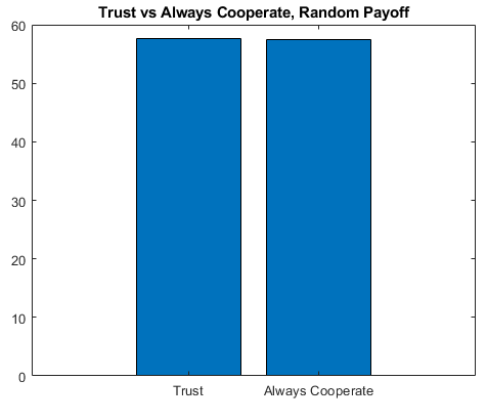
(f)



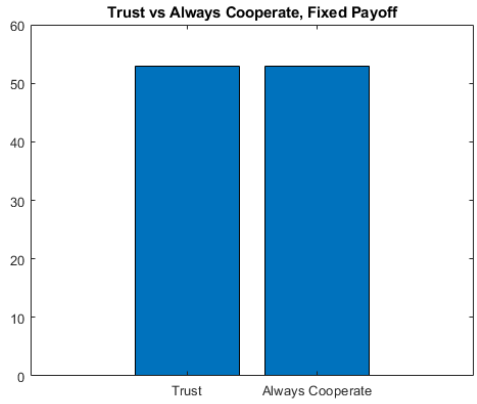
(g)



(h)



(i)



(j)

Figure 2: Payoff comparison for different strategies against each other.

7 Summary and Outlook

The simulation allows for a comparison of the trust strategy against various other strategies. Furthermore a stability condition for a stable cooperative relationship between trust and tit for tat, win stay lose shift or trust strategy has been identified. The trust model has proven to be a valid player in a game theoretical framework. The upcoming policy analysis report will further illuminate the suitability for real world application.

8 References

1. Axelrod, Robert. "The emergence of cooperation among egoists." American political science review 75.2 (1981): 306-318.

2. Nowak, Martin, and Karl Sigmund. "A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoner's Dilemma game." Nature 364.6432 (1993): 56-58.

3. https://plato.stanford.edu/entries/prisoner-dilemma/guide.html (last called 27.11.17)