# Evolutionary Games Toolkit
# v. 2025.06.26
# Documentation

The GameTheory2025 Collective

June 26, 2025

## 0 Introduction

The *Evolutionary Games Toolkit* (EGT) is a Matlab toolbox aimed at the study of (obviously) evolutionary games. Our definition of what *is* an "evolutionary game" is given in Section 2. EGT provides functions for three types of operations:

1. Simulation
2. Markovian analysis
3. Mean Field Dynamics (MFD) analysis

The exact meaning of the above terms is explained in the following sections.

## 1 Quick Start

EGT has been created with Matlab 2023b; while we have not tested it, it probably works with every Matlab version after 2020. It can be obtained from `https://github.com/thanasiskehagias/EvolutionaryGamesToolkit` and also from the Matlab File Exchange. It is provided under the MIT license.

To start using EGT, unzip `EGT.zip` and you will obtain a root folder `EGT` with several subfolders. Open Matlab, go to `EGT` and first run the script `AddMyPath`. Next go to subfolder `Examples` and then to any of its subfolders `Ex0501`, ..., `Ex0504` and run any of the scripts included in these; each script produces some plots which show the evolution of strategy usage over several generations of an evolutionary game.

## 2 Evolutionary Games and Markov Chains

Classical descriptions of evolutionary games appear in [2, 5, 6]. Our main starting point is [6], but we use a simplified version of the model presented therein. We find that our results closely approximate those of [6].

In our formulation, the ingredients of an evolutionary game are the following. We start with a *symmetric* two-player game described by an $M \times M$ bimatrix $\left(B, B^T\right)$; i.e., each player can use a pure strategy from the set $\Sigma = \{\sigma_1, ..., \sigma_M\}$ and we have

$$B_{m_1 m_2} = \text{"the payoff received by row player when he uses } \sigma_{m_1} \text{ against column player using } \sigma_{m_2}\text{"}.$$

Now assume the existence of $N$ *players* and define the following:

1. A *population* vector $\mathbf{z} = (z_1, ..., z_N) \in \mathbf{Z}$ indicates the strategy used by each player, i.e., $z_n = m$ means the $n$-th player is using the $m$-th strategy; we will often call such a player an "*m-user*". It takes values in the

$$\text{Population Space: } \mathbf{Z} = \{(z_1, ..., z_N) : \forall n : z_n \in \{1, .., M\}\}.$$

2. A *strategy* or *state* vector $\mathbf{s} = (s_1, ..., s_M) \in \mathbf{S}$ indicates how many players are using each strategy, i.e., $s_m$ is the number of $m$-users. It takes values in the

$$\text{State Space: } \mathbf{S} = \left\{ (s_1, ..., s_M) \in \{0, 1, .., N\} \text{ such that } \sum_{m=1}^{M} s_m = N \right\}.$$

For given payoff matrix $B$, and number of players $N$, the evolutionary game consists of the following procedure.

**Evolutionary Game Protocol**

1. First we choose an initial population vector $\mathbf{z}(0) = (z_1(0), ..., z_N(0)) \in \mathbf{Z}$, i.e., an initial distribution of strategies to players.
2. Then we repeat for $j \in \{0, 1, ..., J-1\}$ the following iteration.
   (a) We compute the state vector $\mathbf{s}(j) = (s_1(j), ..., s_M(j)) \in \mathbf{S}$ and the *frequency vector* $\mathbf{x}(j) = (x_1(j), ..., x_M(j))$ as follows:

   $$\forall m : s_m(j) = |\{z_n(j) : z_n(j) = m\}|, \qquad x_m(j) = \frac{s_m(j)}{\sum_{m=1}^{M} s_m(j)}.$$

   (b) The players play all possible matches between each other, i.e., for all the pairs $(n_1, n_2)$ with $n_1, n_2 \in \{1, ..., N\}$ and $n_1 \neq n_2$. In each match they play one round of the $B$ game.

   (c) The $n$-th player (for $n \in \{1, ..., N\}$) collects a total payoff

   $$q_n(j) = \sum_{n \neq n'} B_{z_n(j) z_{n'(j)}},$$

   i.e., the $n$-th player collects his total payoff by playing strategy $z_n$ against the $n'$-th player's strategy $z_{n'}$, for every $n' \neq n$.

   (d) The $m$-th strategy collects a total payoff

   $$Q_m(j) = \sum_{n : z_n = m} q_n(j),$$

   i.e., the total payoff of the $m$-th strategy is the sum of the payoffs of the $m$-users.

   (e) We obtain the next population vector $\mathbf{z}(j+1)$ (also called a *generation*) by letting some players update their strategy. This is achieved by a *revision protocol* $\mathcal{R}$; we will presently discuss several such protocols.

So an evolutionary game is a pair $(B, \mathcal{R})$ (with auxiliary parameters $N$ and $J$) and produces sequences $(\mathbf{z}(j))_{j=0}^{\infty}$, $(\mathbf{s}(j))_{j=0}^{\infty}$ and $(\mathbf{x}(j))_{j=0}^{\infty}$. It is clear that the population process $(\mathbf{z}(j))_{j=0}^{\infty}$ is a Markov chain and a little thought shows that the same is true of the state process $(\mathbf{s}(j))_{j=0}^{\infty}$; the *frequency process* $(\mathbf{x}(j))_{j=0}^{\infty}$ is simply a function of $(\mathbf{s}(j))_{j=0}^{\infty}$.

Given a specific evolutionary game $(B, \mathcal{R})$, we want to study the long term behavior of $(\mathbf{s}(j))_{j=1}^{\infty}$. By choosing revision protocols which favor better performing strategies (i.e., those with higher total payoff), we hope that $(\mathbf{s}(j))_{j=0}^{\infty}$ will converge into a state $\overline{\mathbf{s}}$ in which all players are using the best performing strategy (more accurately: *one* the best performing strategies). There are two ways to obtain information regarding long term behavior:

1. by performing a large number of simulations, or
2. by computing the state transition probability matrix $P$ of $(\mathbf{s}(j))_{j=0}^{\infty}$ and studying it by standard Markov chain analysis techniques.

We can classify revision protocols by the number of players who may change their strategy (in one generation). A *$K$-revision protocol* is one in which $K$ players can change. In EGT we have only implemented 1- and $N$-revision protocols.[1] Let us now present these.

We will use the following notations. We write the player total payoff vector as $\mathbf{q} = (q_1, ..., q_N)$ and the strategy total payoff vector as $\mathbf{Q} = (Q_1, ..., Q_M)$. Note that both $\mathbf{q}$ and $\mathbf{Q}$ are functions of the state vector $\mathbf{s}$ (or, equivalently, the frequency vector $\mathbf{x}$), the payoff matrix $B$ etc. However, for the sake of brevity, we will usually omit this dependence from our notation. We let $\overline{Q}$ be the average strategy payoff and $\widehat{Q}$ be the maximum strategy payoff

$$\overline{Q} = \sum_{m=1}^{M} x_m Q_m, \quad \widehat{Q} = \max_m Q_m.$$

Also we use the notation $[y]_+ = \max(y, 0)$, i.e., $[y]_+$ is the nonnegative part of $y$. Finally, $\mathbf{1_U}(u)$ is the indicator function of set $\mathbf{U}$, i.e., it equals 1 when the $u \in \mathbf{U}$ and 0 when $u \notin \mathbf{U}$.

EGT contains a single $N$-revision protocol, the **Simultaneous Proportional Revision**, which works as follows. In each generation *all* players update their strategies simultaneously; the $n$-th player adopts the $m$-th strategy with probability:

$$\forall m \in \{1, ..., M\}) : \pi_m = \frac{\sum_{n:s_n=m} q_n(j)}{\sum_{n=1}^{N} q_n(j)}.$$

EGT contains several 1-revision protocols, all of which follow the same general pattern:

1. A player is selected equiprobably from the population.
2. If the player is an $m_1$-user, then he adopts the $m_2$-th strategy with probability $R_{m_1 m_2}(\mathbf{s})$, where $R(\mathbf{s})$ is an $M \times M$ *rate matrix* depending on the current state $\mathbf{s}$. By specifying $R(\mathbf{s})$ we obtain a specific 1-revision protocol.

EGT contains the following rate matrix implementations (for brevity of notation, we omit the dependence on $\mathbf{s}$).

1. **Best Response**. Strategy $m_1$ switches equiprobably to the strategy of one of the best performing players. I.e., letting

$$\text{the set of the players who achieve max payoff: } \widehat{\mathbf{N}} = \left\{ n : q_n = \max_i q_i \right\},$$
$$\text{the set of the strategies of these players: } \widehat{\mathbf{M}} = \left\{ z_n : n \in \widehat{\mathbf{N}} \right\},$$

   we set

$$R_{m_1 m_2} = \frac{\mathbf{1}_{\widehat{\mathbf{M}}}(m_2)}{\left|\widehat{\mathbf{M}}\right|}.$$

2. **Pairwise Proportional Comparison**. Strategy $m_1$ switches to $m_2$ with probability proportional to the positive part of the surplus of $Q_{m_2}$ over $Q_{m_1}$, multiplied by the frequency of $m_2$:

$$R_{m_1 m_2} = \frac{x_{m_2} [Q_{m_2} - Q_{m_1}]_+}{\sum_{m=1}^{M} x_m [Q_m - Q_{m_1}]_+}.$$

3. **Pairwise Comparison**. Strategy $m_1$ switches to $m_2$ with probability proportional to the positive part of the surplus of $Q_{m_2}$ over $Q_{m_1}$:

$$R_{m_1 m_2} = \frac{[Q_{m_2} - Q_{m_1}]_+}{\sum_{m=1}^{M} [Q_m - Q_{m_1}]_+}.$$

[1] It should be fairly easy for the user to modify our code to produce $K$-revision protocols.

4. **Comparison to the Average Payoff**. Strategy $m_1$ switches to $m_2$ with probability proportional to positive part of the $Q_{m_2}$ surplus over $\overline{Q}$:

$$R_{m_1 m_2} = \frac{\left[Q_{m_2} - \overline{Q}\right]_+}{\sum_{m=1}^{M} \left[Q_m - \overline{Q}\right]_+}.$$

5. **Logit**. Strategy $m_1$ switches to $m_2$ with the following probability:

$$R_{m_1 m_2} = \frac{\exp\left(Q_{m_2}/\eta\right)}{\sum_{m=1}^{M} \exp\left(Q_m/\eta\right)}$$

   where $\eta$ is a revision parameter.

It is worth pointing out that all of the above rate matrices assign zero probability to the adoption of a strategy which is not represented in the population (why?). In other words, once a strategy becomes extinct in an evolutionary game, it will never reappear. Having selected a rate matrix $R$, we can compute the transition probability matrix $P$ as follows

$$\forall \mathbf{s}, \mathbf{s}' : P_{\mathbf{s}\mathbf{s}'} = \Pr\left(\mathbf{s}\left(j+1\right) = \mathbf{s}' | \mathbf{s}\left(j\right) = \mathbf{s}\right) = \begin{cases} x\left(\mathbf{s}\right) R_{m_1 m_2}\left(\mathbf{s}\right) & \text{if } s'_{m_1} = s_{m_1} - 1 \text{ and } s'_{m_2} = s_{m_2} + 1 \\ 0 & \text{else} \end{cases}$$

Actually, the above formula holds only for "interior" states, i.e., those with $s_m \in \{2, ..., N-1\}$ for all $m$. The modifications for "boundary" states are straightforward; for example, with $N = 3$: the only possible transition from $(3,0,0)$ is to itself, the possible transitions from $(2,1,0)$ are to $(3,0,0)$ and to $(1,2,0)$ etc.

## 3 Mean Field Dynamics

It is well known [5, 6] that, for certain revision protocols, the evolution of the frequency process $\left(\mathbf{x}\left(j\right)\right)_{j=1}^{\infty}$ can be approximated, for large $N$, by a *mean field dynamic* (MFD) expressed as a system of ordinary differential equations. The following table summarizes some of these correspondences.

| **Revision Protocol** | **Mean Field Dynamic** |
|---|---|
| Simultaneous Proportional Revision | Type 2 Replicator Dynamics |
| Pairwise Proportional Comparison | Type 1 Replicator Dynamics |
| Pairwise Comparison | Smith Dynamics |
| Comparison to Average Payoff | Brown-von Neumann-Nash Dynamics |
| Logit | Logit |

Table 1: Correspondence between revision protocols and mean field dynamics.

We will use the above correspondence to refer to both mean field dynamics *and* revision protocols; for example we will say "the Smith protocol" rather than "the pairwise comparison protocol".

Obviously, discrete time provides a more accurate description of the evolution of discrete generations. Consequently, in our version of evolutionary games we express the dynamics by *difference*, rather than differential equations. For each of the dynamics of Table 1, we give below a system of difference equations, which describe the evolution of the frequency vector $\mathbf{x}\left(j\right)$; from this and $N$ we can compute $\mathbf{s}\left(j\right)$ and a "representative" $\mathbf{z}\left(j\right)$. In what follows we will use the auxiliary quantities:

$$\text{Average Payoff: } \overline{Q}\left(\mathbf{x}\right) = \sum x_i Q_i\left(\mathbf{x}\right)$$

$$m\text{-th Strategy Excess Payoff: } \widehat{Q}_m\left(\mathbf{x}\right) = Q_m\left(\mathbf{x}\right) - \overline{Q}\left(\mathbf{x}\right)$$

Now we can present the list of dynamics.

1. **Replicator Dynamics Type 2**.

$$\forall j, m : x_m (j+1) = \frac{\sum_{n:s_n=m} q_n (j)}{\sum_{n=1}^{N} q_n (j)}.$$

2. **Replicator Dynamics Type 1**.

$$\forall j, m : x_m (j+1) = x_m (j) + h \cdot x_m (j) \cdot \widehat{Q}_m (x (j))$$

3. **Smith Dynamics**.

$$\forall j, m : x_m (j+1) = x_m (j) + h \cdot \left( \sum_{i=1}^{M} x_i \left[ Q_m (x (j)) - Q_i (x (j)) \right]_+ - x_m (j) \sum_{i=1}^{M} \left[ Q_i (x (j)) - Q_m (x (j)) \right]_+ \right).$$

4. **Brown-von Neumann-Nash Dynamics**.

$$\forall j, m : x_m (j+1) = x_m (j) + h \cdot \left( \left[ \widehat{Q}_m (x (j)) \right]_+ - x_m (j) \left( \sum_{i=1}^{M} \left[ \widehat{Q}_i (x (j)) \right]_+ \right) \right).$$

5. **Logit Dynamics**.

$$\forall j, m : x_m (j+1) = x_m (j) + h \cdot \left( \frac{\exp (Q_m (x (j)) / \eta)}{\sum_{i=1}^{M} \exp (Q_i (x (j)) / \eta)} \right).$$

In the above equations $h$ is an "integration parameter" which influences the convergence rate of the solution; and $\eta$ is a "noise parameter" used in the logit dynamics, with small values resulting in faster convergence. For each equation, *the user must select appropriate values of these parameters by trial and error.*

## 4 The Basic EGT Functions

EGT contains four basic functions:

1. `[P,S]=EGTMarkP(B,Dynamic,M,N,eta)` computes the transition probability matrix of an evolutionary game.
2. `[s,x]=EGTMarkSim(P,S,s0,J)` simulates an evolutionary game using the transition probability matrix.
3. `[s,x]=EGTMFDyn(B,Dynamic,s0,J,h,eta)` computes the mean field dynamics of an evolutionary game.
4. `[s,x]=EGTSim(B,Dynamic,s0,J,eta)` simulates an evolutionary game.

The input variables are as follows.

| | | |
|---|---|---|
| `B` | : | Payoff matrix |
| `Dynamic` | : | Dynamic or Protocol |
| `P` | : | Transition probability matrix |
| `S` | : | Srategy state space matrix |
| `s0` | : | Initial state |
| `M` | : | Number of strategies |
| `N` | : | Number of players |
| `J` | : | Number of generations |
| `h` | : | Integration step |
| `eta` | : | Noise parameter (only used by Logit functions) |

The output variables are as follows.

| | | |
|---|---|---|
| `P` | : | Transition probability matrix |
| `S` | : | Srategy state space matrix |
| `s` | : | Strategy state sequence (across generations) |
| `x` | : | Frequency sequence (across generations) |

A full list of the `EGT` functions, with some explanation of their use, can be found in Section A of the Appendix. More details are provided as comments in the corresponding function files.

## 5 Examples

### 5.1 Prisoner's Dilemma with Three Strategies

We are particularly interested in $B$ games obtained from an iterated two-player *base game*, described by a bimatrix $A$. From such an $A$ we construct an $M \times M$ matrix $B$ where

$$B_{m_1 m_2} = \text{"the payoff received by playing } T \text{ rounds of } A\text{, using strategy } \sigma_{m_1} \text{ against strategy } \sigma_{m_2}\text{".}$$

In this example we will suppose that $A$ is the Prisoner's Dilemma [1] with bimatrix

$$A = \begin{bmatrix} 3,3 & 1,4 \\ 4,1 & 2,2 \end{bmatrix},$$

and we will use three strategies: $\sigma_1 =$"AllC" (always cooperate), $\sigma_2 =$"AllD" (always defect) and $\sigma_3 =$"TitForTat". Further, we assume that the iterated game is played for $T = 1000$ rounds. Consider $B_{11}$, the payoff of a player who uses "AllC" for 1000 rounds, against another player who also uses "AllC"; hence the first player will receive

$$B_{11} = 1000 \cdot 3 = 3000.$$

similarly, $B_{12}$ is the payoff of a player who uses "AllC" for 1000 rounds, against another player who uses "AllD"; hence the first player will receive

$$B_{11} = 1000 \cdot 1 = 1000.$$

Continuing in this manner we obtain

$$B = \begin{bmatrix} 3000 & 1000 & 3000 \\ 4000 & 2000 & 2002 \\ 3000 & 1999 & 3000 \end{bmatrix}$$

To complete the description of the evolutionary game we select the best response revision protocol. We now have an evolutionary game, which we simulate using the script `x0501ASim.m` (contained in folder `Examples/Ex0501`) listed below.

```
clc; clear all
ExpName='x0501ASimBR';
GrName='Best Response Simulation';

A=[3 1; 4 2];
T=1000;
Strategies = ["AllC", "AllD", "Grim"];
s0=[5 5 5];
J=200;

B = MakePayoff(A,Strategies,T);
[s,x]=EGTSim(B,'BR',s0,J,[]);          save([ExpName 'S']);
GrSim(s,x,Strategies,ExpName,GrName);
```

Lines 1-9 perform the initialization. We use an initial population $\mathbf{s}(1) = (5,5,5)$, and run the game for 200 generations. The main action takes place in lines 11-13.

1. In line 11 we compute the payoff matrix $B$.
2. In line 12 we simulate the evolutionary game and obtain the state sequence $\mathbf{s}(1), ..., \mathbf{s}(200)$ and the frequency sequence $\mathbf{x}(1), ..., \mathbf{x}(200)$.
3. In line 13 we plot the state sequence. This appears in Figure 1: as a function of time (generations) on the left and as a trajectory in the 3-simplex on the right.

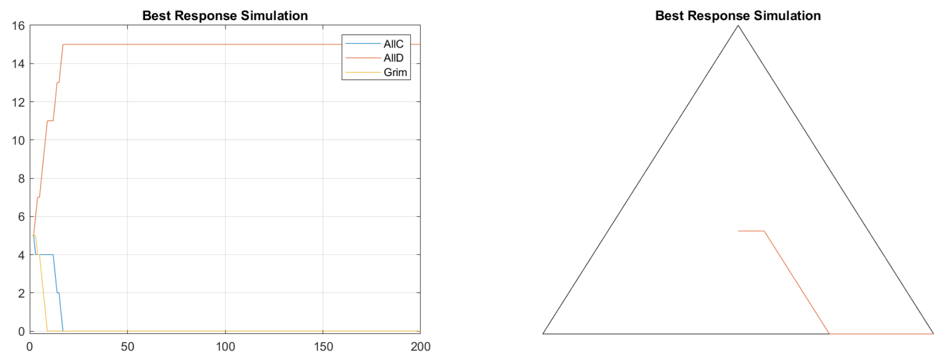


Figure 1: Simulation of an IPD evolutionary game. Plots of the numbers of $m$-users (for $m \in \{1,2,3\}$) as a function of generation number (left) and as a trajectory in the 3-simplex (right).

We see that the game settles at state $(0,15,0)$, in which all players use the AllD strategy. To get a better understanding of the system, we run a Markovian analysis, with the script `x0501AMark.m` (also contained in folder `Examples/Ex0501`) listed below.

```
1  ExpName='x0501AMrkBR';
2  GrName='Best Response';
3
4  A=[3 1; 4 2];
5  T=1000;
6  Strategies = ["AllC", "AllD", "Grim"];
7  s0=[5 5 5];
8  J=200;
9
10 M=length(s0);
11 N=sum(s0);
12 B = MakePayoff(A,Strategies,T);
13 [P,S]=EGTMarkP(B,'BR',M,N,[]);  save([ExpName 'P']);
14 [s,x]=EGTMarkSim(P,S,s0,J);     save([ExpName 'S']);
15 GrMrk(Strategies,P,S,s,ExpName,GrName);
```

1. The main action takes place in lines 13 (where the transition probability matrix $P$ is computed) and 14 (where the state vector sequence $\mathbf{s}(1), ..., \mathbf{s}(200)$ is produced directly from $P$).
2. In line 15 we plot the *state transition diagram* (STG) obtained from $P$ and the state vector sequence $\mathbf{s}(1), ..., \mathbf{s}(200)$. These plots appear in Figure 2.

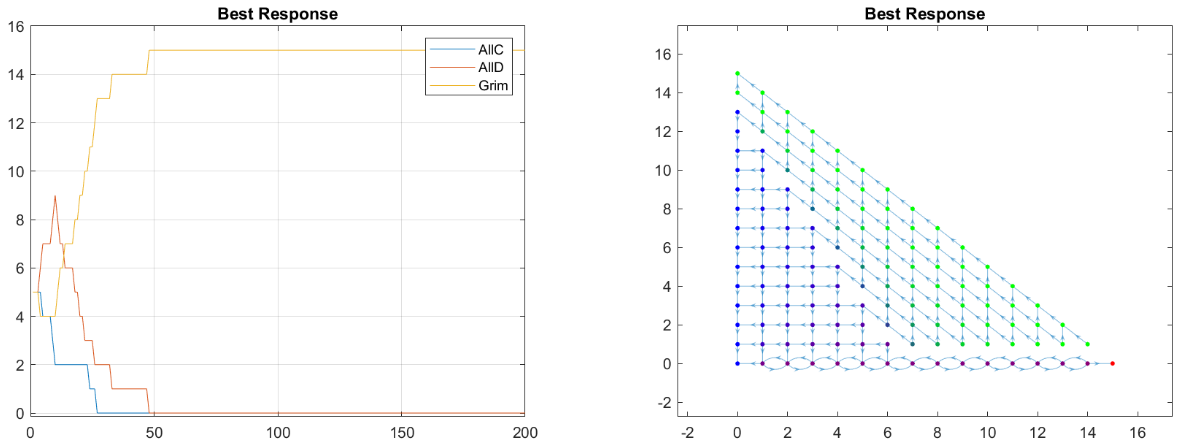


Figure 2: Markovian analysis of an IPD evolutionary game. Left: plots of the numbers of $m$-users (for $m \in \{1,2,3\}$) as a function of generation number; this is a realization of the Markov chain defined by the evolutionary game. Right: STG of the Markov chain.

The following remarks can be made regarding Figure 2.

1. Figure 2, left is similar to Figure 1, left. This is expected, since these are plots of two realizations of the same Markov chain. However we see that in Figure 2 the state process settles at state $(0,0,15)$, i.e., the "pure Grim" state. As we will see next, this can be expected from the structure of the corresponding transition probability matrix $P$.

2. Figure 2, right is the plot of the STG, obtained from $P$ and interpreted as follows.

   (a) Every state $\mathbf{s} = (s_1, s_2, s_3)$ corresponds to a dot of the diagram; the coordinates of the dot are $(s_1, s_2)$.

   (b) The diagram shows three absorbing states, the ones which have no outgoing arrows. They have coordinates $(0,0)$, $(15,0)$ and $(0,15)$ so they correspond to the "pure" states $(0,0,15)$, $(15,0,0)$ and $(0,15,0)$. No other absorbing states exist, so we know from Markov chain theory [3] that, with probability one, the population will eventually settle to one of the pure states.

   (c) Consequently, with every state $\mathbf{s} = (s_1, s_2, s_3)$ of the STG is associated a vector $(p_1(\mathbf{s}), p_2(\mathbf{s}), p_3(\mathbf{s}))$, where $p_1(\mathbf{s})$ is the probability that the process, when starting at $\mathbf{s}$, will be absorbed by $(15,0,0)$; similarly $p_2(\mathbf{s})$ is the probability of absorption by $(0,15,0)$ and $p_3(\mathbf{s})$ is the probability of absorption by $(0,0,15)$.

   (d) The dot of the STG associated with state $\mathbf{s}$ is color coded as follows: we take $(p_1(\mathbf{s}), p_2(\mathbf{s}), p_3(\mathbf{s}))$ to be an RGB color vector and color the corresponding dot accordingly. For example $(15,0,0)$ is absorbed into itself with probability one, hence it is painted red; similarly $(0,15,0)$ is painted green and $(0,0,15)$ blue. The color of dots close to $(0,0)$ (corresponding to states close to $(0,0,15)$) is close to blue, because the probability of being absorbed by $(0,0,15)$ is equal to or very close to one, as seen by the STG arrows. Similarly dots close to $(15,0)$ (corresponding to states close to $(15,0,0)$) are close to red, because the probability of being absorbed by $(15,0,0)$ is equal to or very close to one. On the other hand, dot $(5,5)$ (i.e., state $(5,5,5)$) is colored "green-blue", because (as seen by following the STG arrows) has about equal probability of being absorbed by either $(0,0,15)$ or $(0,15,0)$. This also explains why in Figure 1 the process settles at $(0,15,0)$, while in Figure 2 it settles at $(0,0,15)$.

We repeat the above experiment with larger populations, namely we take $\mathbf{s}(1) = (20,20,20)$. The experiment is implemented in files `x0501BSim.m` and `x0501BMrk.m` (contained in folder `Examples/Ex0501`). We plot the obtained results in Figures 3 and 4, which parallel closely Figures 1 and 2, respectively.

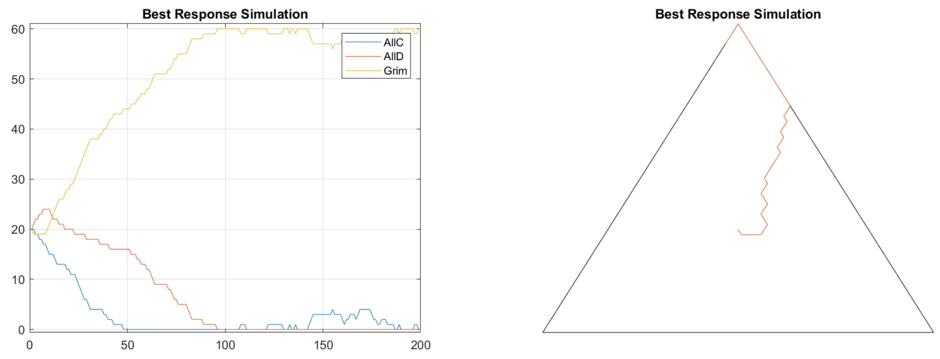


Figure 3: Simulation of an IPD evolutionary game. Plots of the numbers of $m$-users (for $m \in \{1,2,3\}$) as a function of generation number (left) and as a trajectory in the 3-simplex (right).

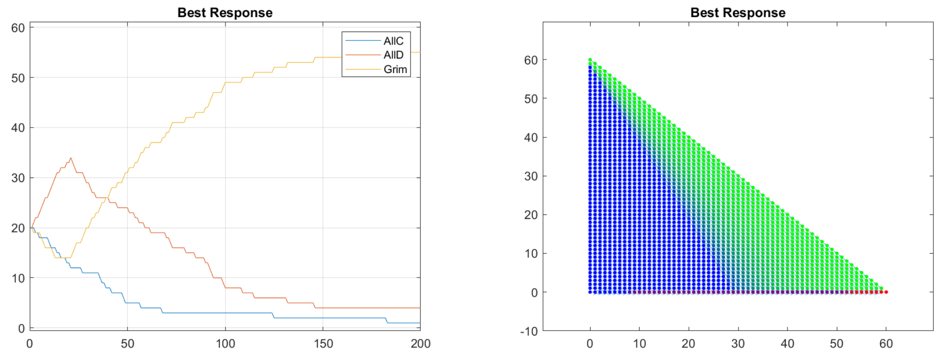


Figure 4: Markovian analysis of an IPD evolutionary game. Left: plots of the numbers of $m$-users (for $m \in \{1,2,3\}$) as a function of generation number; this is a realization of the Markov chain defined by the evolutionary game. Right: STG of the Markov chain.

Next we repeat the analysis with with $\mathbf{s}(1) = (100, 100, 100)$ (an population of 300 players) using the files `x0501CSim.m` and `x0501CMrk.m`. The "plain"simulation runs without problems, but Markovian analysis cannot be done in reasonable time on our computer. So we run the simulation analysis on the same evolutionary game, with $\mathbf{s}(1) = (100, 100, 100)$ and using the following revision protocols / dynamics: Replicator Type 1 and Type 2 and BNN. The experiments are coded in the file `x0501DSim.m` listed below.

```
clear; clc;
A=[3 1; 4 2];
T=1000;
Strategies = ["AllC", "AllD", "Grim"];
s0=[100 100 100];
J=1000;
eta=0.05;

B = MakePayoff(A,Strategies,T);

ExpName='x0501DSimRep1';
GrName='Replicator Type 1';
B = MakePayoff(A,Strategies,T);
[s,x]=EGTSim(B,'Rep1',s0,J,[]);          save([ExpName 'S']);
GrSim(s,x,Strategies,ExpName,GrName);

ExpName='x0501DSimRep2';
GrName='Replicator Type 2';
```

```
B = MakePayoff(A,Strategies,T);
[s,x]=EGTSim(B,'Rep2',s0,J,[]);          save([ExpName 'S']);
GrSim(s,x,Strategies,ExpName,GrName);

ExpName='x0501DSimBNN';
GrName='Brown-Von Neumann-Nash Simulation';
B = MakePayoff(A,Strategies,T);
[s,x]=EGTSim(B,'BNN',s0,J,[]);           save([ExpName 'S']);
GrSim(s,x,Strategies,ExpName,GrName);
```

The results are plotted in Figures 5-7.

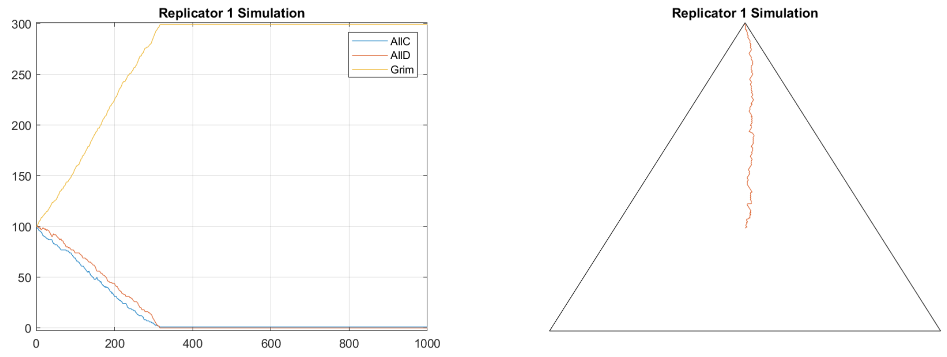


Figure 5: Simulation analysis of IPD evolutionary game with Type 1 Replicator dynamics.

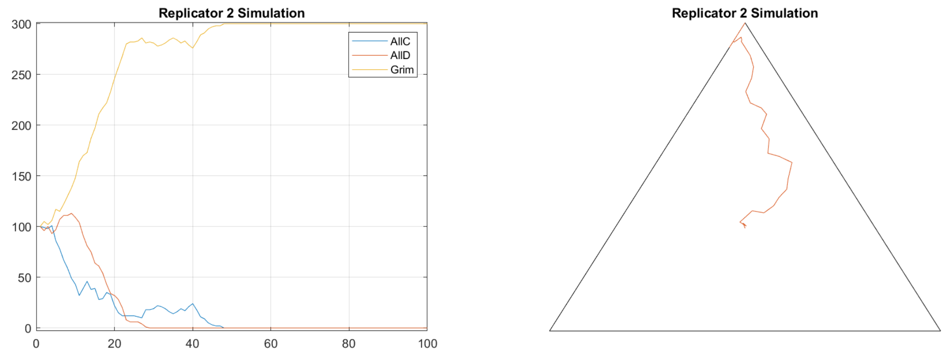


Figure 6: Simulation analysis of IPD evolutionary game with Type 2 Replicator dynamics.

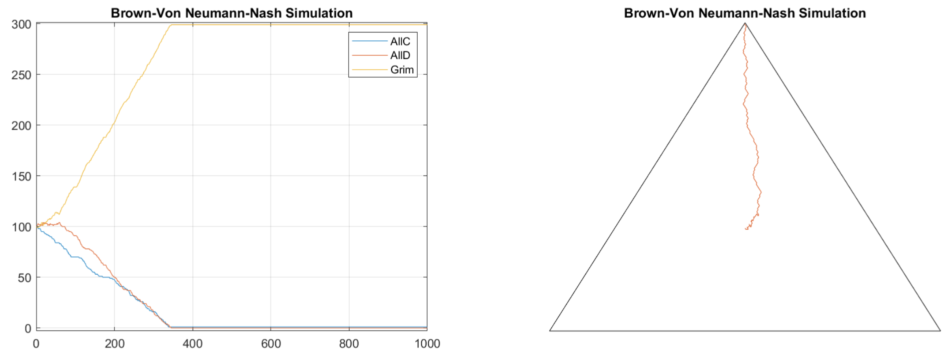


Figure 7: Simulation analysis of IPD evolutionary game with Brown-von Neumann-Nash dynamics.

Finally we use the same $\mathbf{s}(1) = (100, 100, 100)$ and the same revision protocols as previously, to run the mean field dynamics analysis. The experiments are coded in the files `x0501DMFD.m` listed below.

```
clear; clc;
A=[3 1; 4 2];
T=1000;
Strategies = ["AllC", "AllD", "Grim"];
s0=[100 100 100];
J=100;

B = MakePayoff(A,Strategies,T);

h=1/T;
ExpName='x0501DMFDRep1'; GrName='Replicator 1 Dynamics';
[s,x]= EGTMFDyn(B,'Rep1',s0,J,h,[]);     save([ExpName 'S']);
GrDyn(s,x,Strategies,ExpName,GrName);

h=1/T;
ExpName='x0501CMFDRep2'; GrName='Replicator 2 Dynamics';
[s,x]= EGTMFDyn(B,'Rep2',s0,J,h,[]);     save([ExpName 'S']);
GrDyn(s,x,Strategies,ExpName,GrName);

h=1/T;
ExpName='x0501CMFDBNN'; GrName='Brown-Von Neumann-Nash Dynamics';
[s,x]= EGTMFDyn(B,'BNN',s0,J,h,[]); save([ExpName 'S']);
GrDyn(s,x,Strategies,ExpName,GrName);
```

The results are plotted in Figures 8-10.

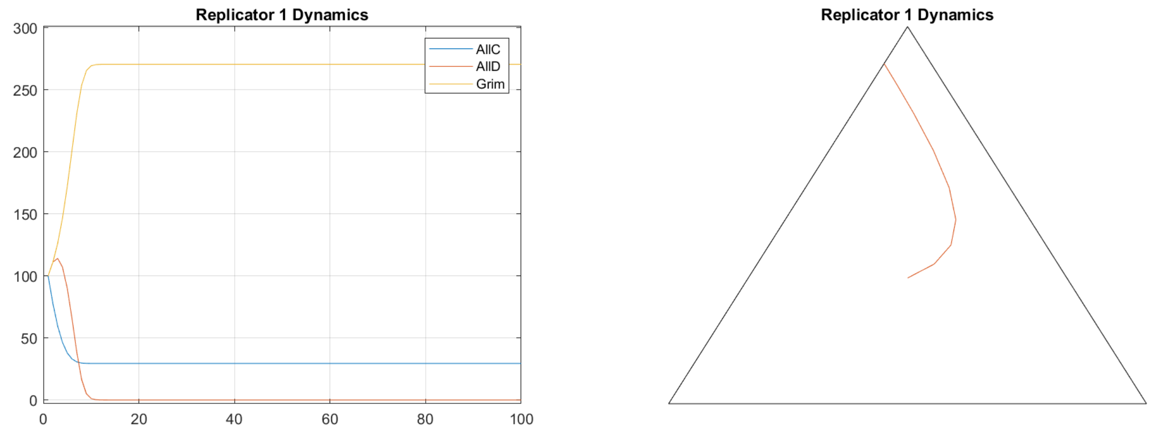


Figure 8: MFD analysis of IPD evolutionary game with Type 1 Replicator dynamics.

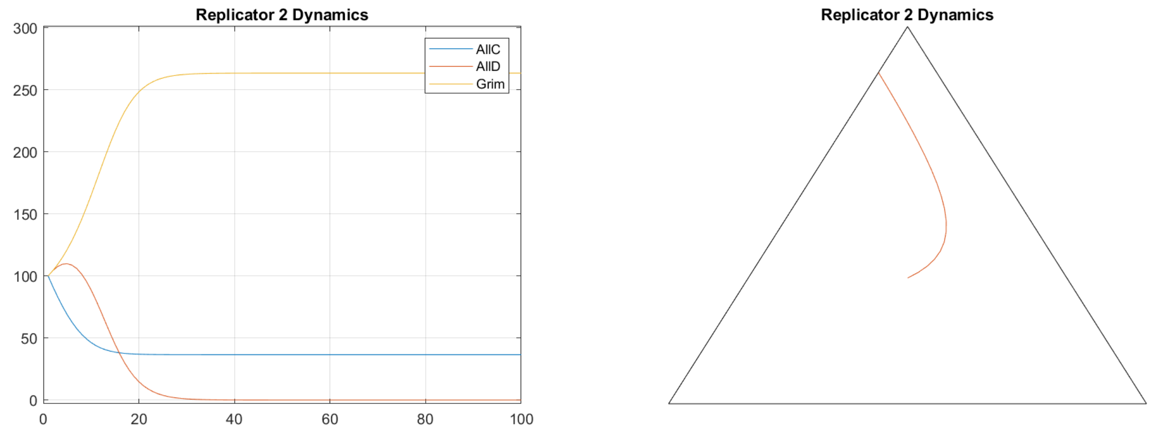


Figure 9: MFD analysis of IPD evolutionary game with Type 2 Replicator dynamics.

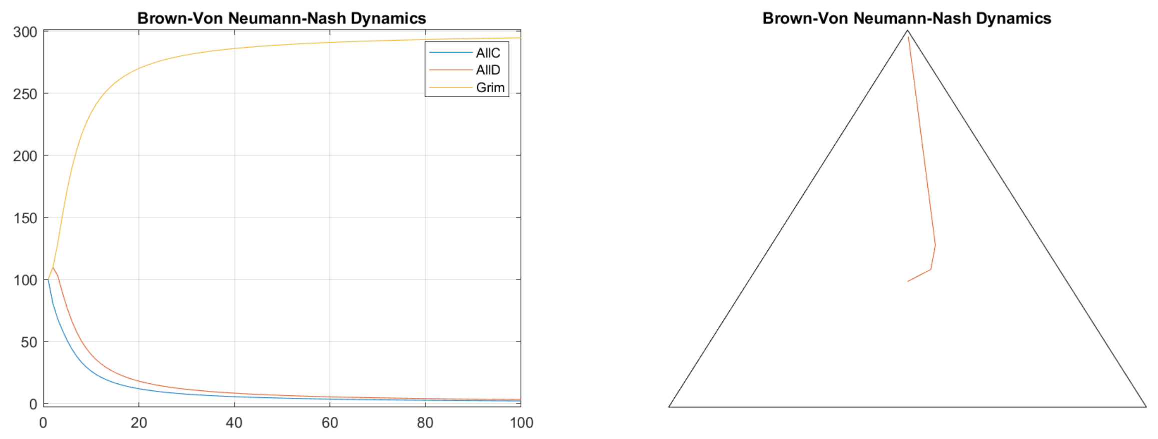


Figure 10: MFD analysis of IPD evolutionary game with Brown-von Neumann-Nash dynamics.

### 5.2 Prisoner's Dilemma with Ten Strategies

We now present an experiment in which we use the iterated PD with *ten* strategies, namely: AllC, AllD, Grim, Pavlov, Prober, TitForTat, PerCCD, PerDDC, Gradual, PerNasty. Our Markovian analysis cannot handle ten strategies, but the simulation and the mean field dynamics can do so without any problems.

So we implement Markovian simulation in the file `x0502ASim.m`, found in `Examples/x0502` and listed below. In all cases we use an initial population with 100 users of each strategy and run Replicator Type 1, Replicator Type 2 and Logit Dynamics.

```
clear; clc;
A=[3 1; 4 2];
T=1000;
Strategies = ...
    ["AllC","AllD","Grim","Pavlov","Prober","TitForTat","PerCCD","PerDDC","Gradual","PerNasty"];
s0=[100 100 100 100 100 100 100 100 100 100];
J=5000;

B = MakePayoff(A,Strategies,T);

ExpName='x0502ASimRep1';
GrName='Replicator Type 1';
B = MakePayoff(A,Strategies,T);
[s,x]=EGTSim(B,'Rep1',s0,J,[]);          save([ExpName 'S']);
GrSim(s,x,Strategies,ExpName,GrName);

ExpName='x0502ASimRep2';
GrName='Replicator Type 2';
B = MakePayoff(A,Strategies,T);
[s,x]=EGTSim(B,'Rep2',s0,J,[]);          save([ExpName 'S']);
GrSim(s,x,Strategies,ExpName,GrName);

ExpName='x0502ASimLogit';
GrName='Logit';
eta=0.01;
B = MakePayoff(A,Strategies,T);
[s,x]=EGTSim(B,'Logit',s0,J,eta);            save([ExpName 'S']);
GrSim(s,x,Strategies,ExpName,GrName);
```

Similarly, we implement the mean field dynamics in file `x0502AMFD.m`, found in `Examples/x0502` and listed below.

```
clear; clc;
A=[3 1; 4 2];
T=1000;
Strategies = ...
     ["AllC","AllD","Grim","Pavlov","Prober","TitForTat","PerCCD","PerDDC","Gradual","PerNasty"];
s0=[100 100 100 100 100 100 100 100 100 100];
J=100;

B = MakePayoff(A,Strategies,T);

ExpName='x0502AMFDLogit';
GrName='Logit';
h=100/T; eta=100;
B = MakePayoff(A,Strategies,T);
[s,x]= EGTMFDyn(B,'Logit',s0,J,h,eta);      save([ExpName 'S']);
GrSim(s,x,Strategies,ExpName,GrName);

ExpName='x0502AMFDRep1';
GrName='Replicator Type 1';
h=1/T;
B = MakePayoff(A,Strategies,T);
[s,x]= EGTMFDyn(B,'Rep1',s0,J,h,[]);        save([ExpName 'S']);
GrSim(s,x,Strategies,ExpName,GrName);

ExpName='x0502AMFDRep2';
GrName='Replicator Type 2';
h=1/T;
B = MakePayoff(A,Strategies,T);
[s,x]= EGTMFDyn(B,'Rep2',s0,J,h,[]);        save([ExpName 'S']);
GrSim(s,x,Strategies,ExpName,GrName);
```

The results are plotted in Figures 11-13.

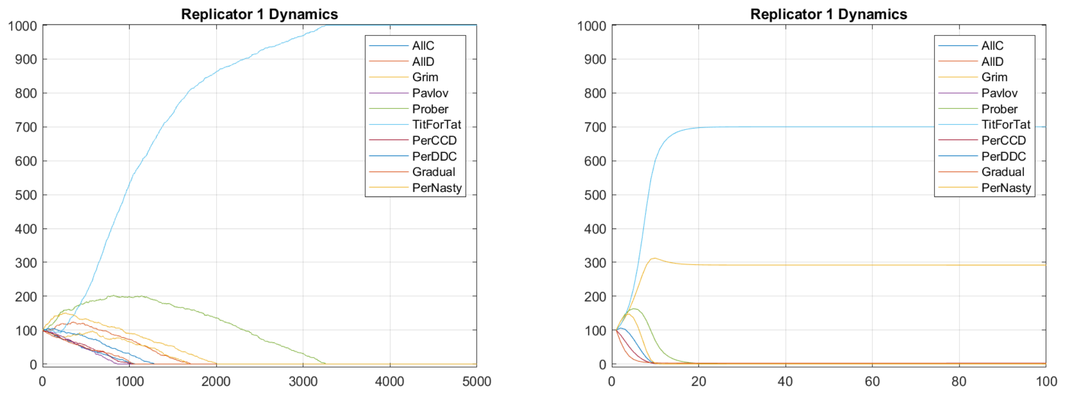


Figure 11: Prisoner's dilemma evolutionary game with ten strategies, starting with 100 users of each strategy. The Replicator Type 1 revision protocol is used. Left: simulation, right: mean field dynamics.

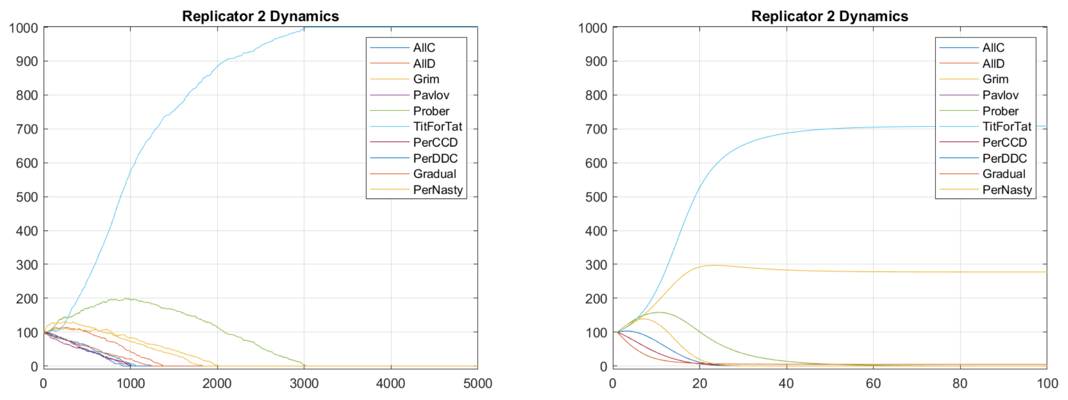


Figure 12: Prisoner's dilemma evolutionary game with ten strategies, starting with 100 users of each strategy. The Replicator Type 2 revision protocol is used. Left: simulation, right: mean field dynamics.

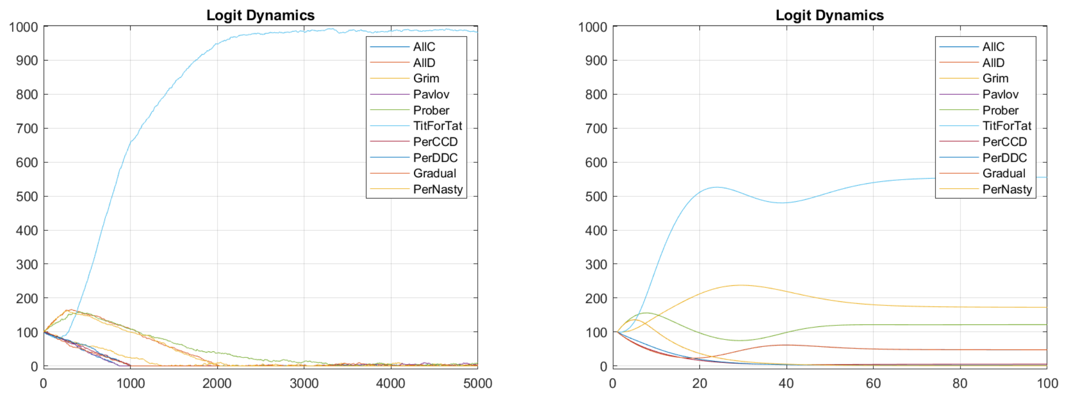


Figure 13: Prisoner's dilemma evolutionary game with ten strategies, starting with 100 users of each strategy. The Logit revision protocol is used. Left: simulation, right: mean field dynamics.

## 5.3 Stag Hunt with Three Strategies

We now present an experiment in which the base game is the Stag Hunt $\left(A, A^T\right)$ with payoff matrix

$$A = \begin{bmatrix} 10 & 1 \\ 8 & 5 \end{bmatrix}.$$

We set the strategy set to be {AllC, AllD, Grim} and we compute the matrix $B$. The experiments are implemented in the files `x0503AMrk.m` and `x0503AMFD.m` (in `Examples/Ex0503.m`).

```
clear; clc;
A=[10 1;8 5];
T=1000;
Strategies = ["AllC", "AllD", "Grim"];
s0=[5 5 5];
J=100;
eta=0.005;

M=length(s0);
N=sum(s0);
B = MakePayoff(A,Strategies,T);
```

```

ExpName='x0503AMrkBR';
GrName='Best Response';
[P,S]=EGTMarkP(B,'BR',M,N,[]);  save([ExpName 'P']);
[s,x]=EGTMarkSim(P,S,s0,J);     save([ExpName 'S']);
GrMrk(Strategies,P,S,s,ExpName,GrName);

ExpName='x0503AMrkRep1';
GrName='Replicator Type 1';
[P,S]=EGTMarkP(B,'Rep1',M,N,[]);    save([ExpName 'P']);
[s,x]=EGTMarkSim(P,S,s0,J);     save([ExpName 'S']);ExpName='x0503AMrkLogit';
GrMrk(Strategies,P,S,s,ExpName,GrName);

ExpName='x0503AMrkLogit';
GrName='Logit';
eta=0.1;
[P,S]=EGTMarkP(B,'Logit',M,N,eta);  save([ExpName 'P']);
[s,x]=EGTMarkSim(P,S,s0,J);     save([ExpName 'S']);
GrMrk(Strategies,P,S,s,ExpName,GrName);
```

```
A=[10 1;8 5];
T=1000;
Strategies = ["AllC","AllD","Grim"];
s0=[5 5 5];
J=100;
eta=0.05;

B = MakePayoff(A,Strategies,T);

h=0.1/T;
ExpName='x0503AMFDRep1'; GrName='Replicator 1 Dynamics';
[s,x]= EGTMFDyn(B,'Rep1',s0,J,h,[]);    save([ExpName 'S']);
GrDyn(s,x,Strategies,ExpName,GrName);

h=100/T;eta=1e2;
ExpName='x0503AMFDLogit'; GrName='Logit Dynamics';
[s,x]= EGTMFDyn(B,'Logit',s0,J,h,eta);  save([ExpName 'S']);
GrDyn(s,x,Strategies,ExpName,GrName);
```

We present the results in the following Figures 14 - 16, according to the revision protocol used. In all cases we use an initial population with 5 users of each strategy.

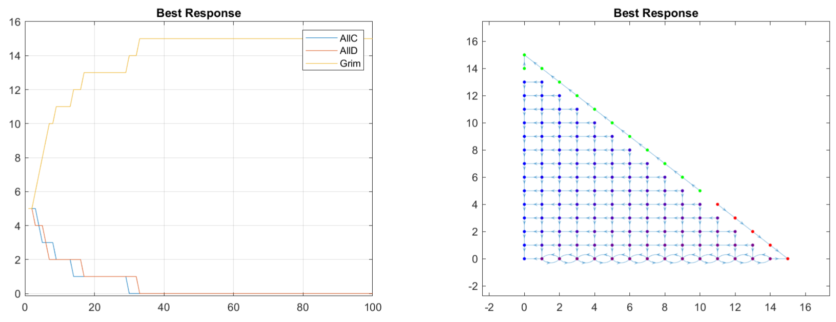


Figure 14: Stag Hunt evolutionary game with three strategies, starting with 5 users of each strategy. The Best Response revision protocol is used. Left: Markovian simulation, Right: STG obtained from $P$.

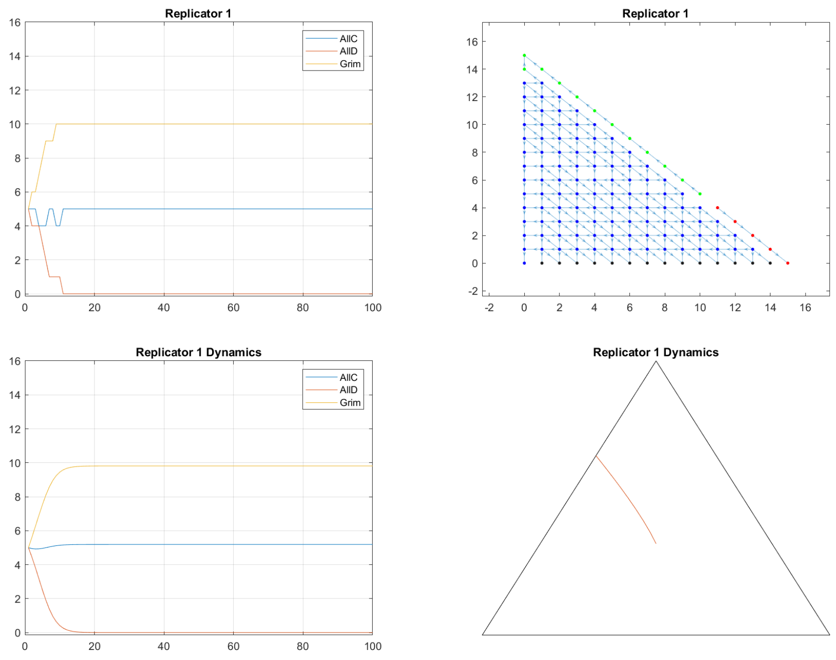


Figure 15: Stag Hunt evolutionary game with three strategies, starting with 5 users of each strategy. The Replicator Type 1 revision protocol is used. Top left: Markovian simulation, top right: STG obtained from $P$. Bottom left: Mean Field Dynamics simulation, bottom right: Mean Field Dynamics simplex trajectory.

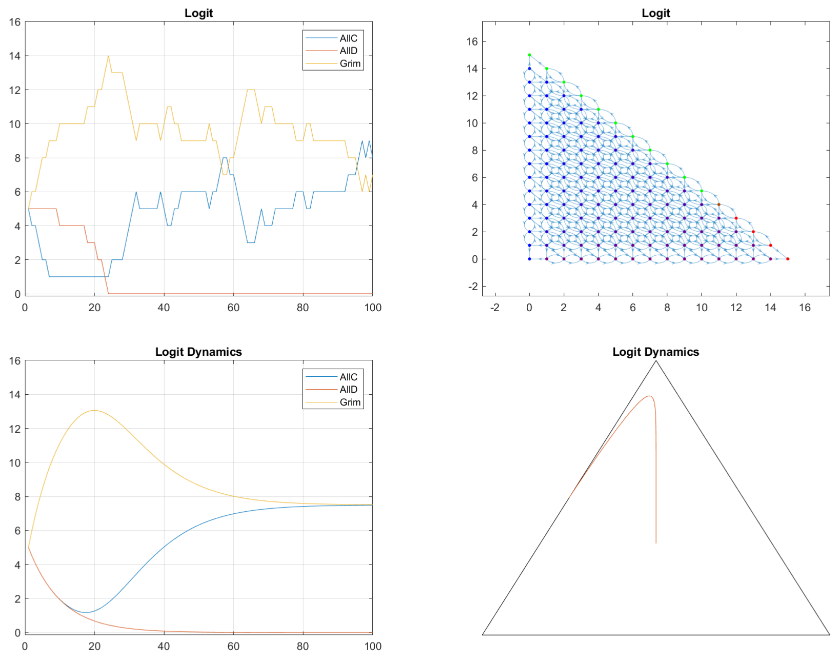


Figure 16: Stag Hunt evolutionary game with three strategies, starting with 5 users of each strategy. The Logit revision protocol is used. Top left: Markovian simulation, top right: STG obtained from $P$. Bottom left: Mean Field Dynamics simulation, bottom right: Mean Field Dynamics simplex trajectory.

We repeat the above experiments with the same setting, but initial population of 20 users of each strategy. We present the results in the following Figures 17 - 19, according to the revision protocol used. In all cases we use an initial population with 20 users of each strategy.

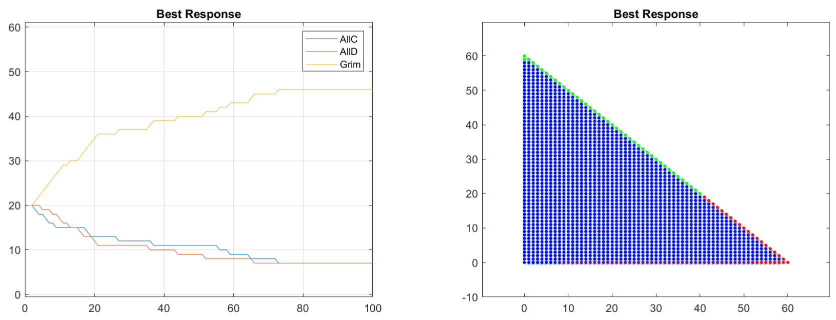


Figure 17: Stag Hunt evolutionary game with three strategies, starting with 20 users of each strategy. The Best Response revision protocol is used. Left: Markovian simulation. Right: STG obtained from $P$.

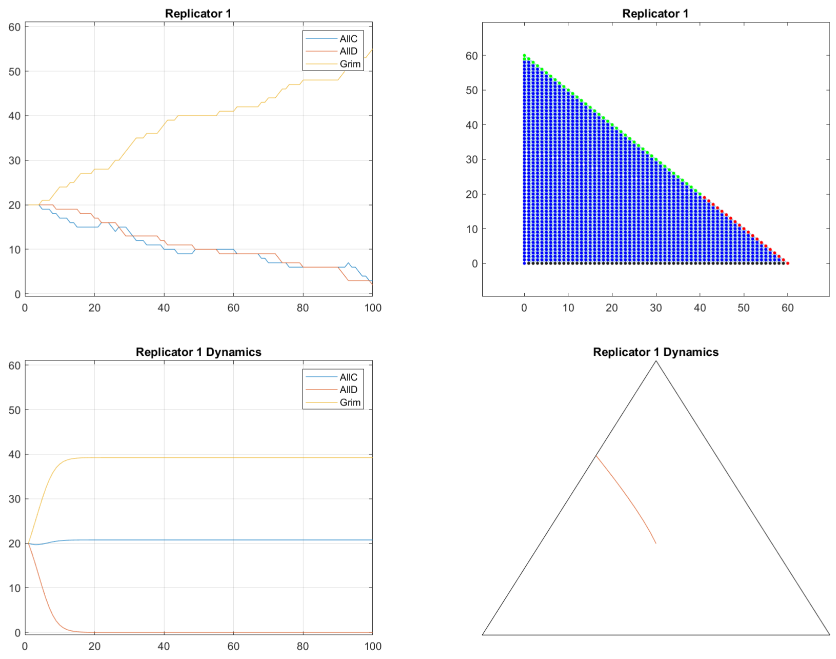


Figure 18: Stag Hunt evolutionary game with three strategies, starting with 20 users of each strategy. The Replicator Type 1 revision protocol is used. Top left: Markovian simulation, top right: STG obtained from $P$. Bottom left: Mean Field Dynamics simulation, bottom right: Mean Field Dynamics simplex trajectory.

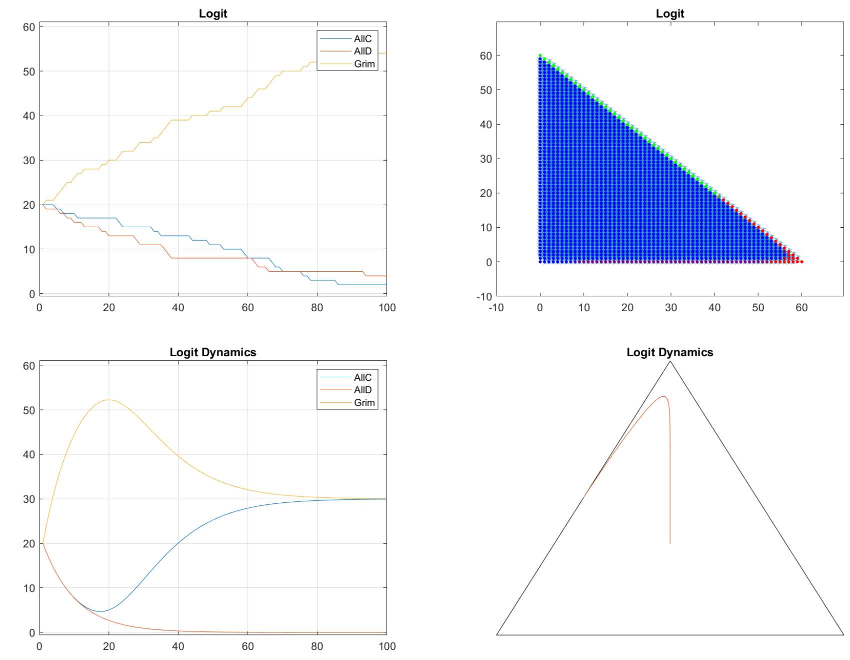


Figure 19: Stag Hunt evolutionary game with three strategies, starting with 20 users of each strategy. The Logit revision protocol is used. Top left: Markovian simulation, top right: STG obtained from $P$. Bottom left: Mean Field Dynamics simulation, bottom right: Mean Field Dynamics simplex trajectory.

## 5.4 A Noniterated Two-Player Game: Rock-Paper-Scissors

In the previous examples we have studied evolutionary games in which the $B$ matrix is obtained from an iterated game (iterated Prisoner's Dilemma, iterated Stag Hunt). In this example we start with a one-round game, namely the Rock-Paper-Scissor game which has payoff matrix.

$$B = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix}.$$

There are three pure strategies available to both players, labeled as Rock, Paper and Scissors.

### 5.4.1 Markovian vs. Mean Field Simulation

We first present a comparison of results from Markovian vs. Mean Field simulation, implemented in `x0504JSim.m` and `x0504JMFD.m`, for $N = 150$ players. In Figures 20-23 we present the results grouped by the revision protocol. As expected, in the Rock-Paper-Scissors game the population undergoes sustained oscillations.

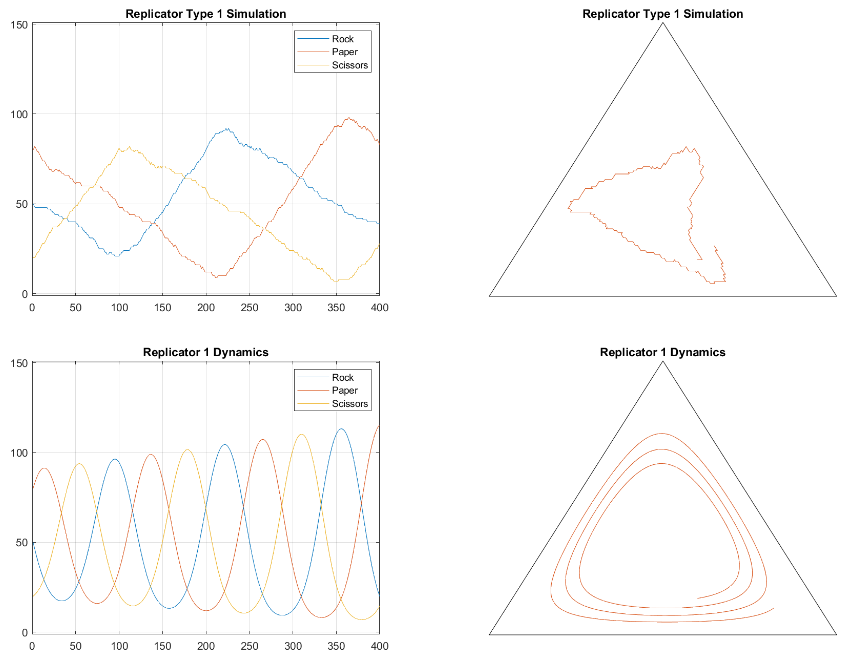


Figure 20: Rock-Scissor-Paper evolutionary game with three strategies, starting with 150 users of each strategy. The Type 1 Replicator revision protocol is used. Top left: Markovian simulation, top right: Markovian simulation simplex trajectory. Bottom left: Mean Field Dynamics simulation, bottom right: Mean Field Dynamics simplex trajectory.

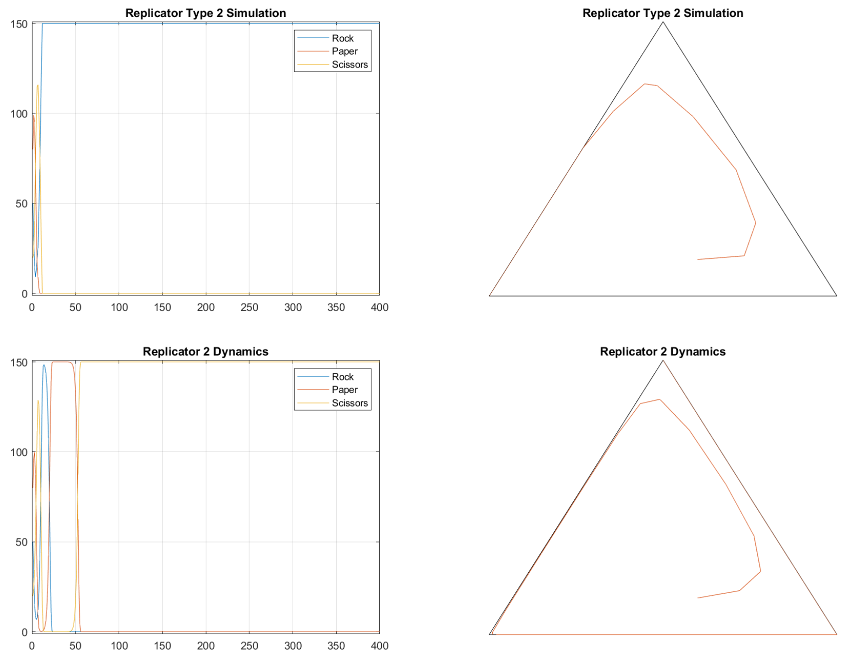


Figure 21: Rock-Scissor-Paper evolutionary game with three strategies, starting with 150 users of each strategy. The Type 2 Replicator revision protocol is used. Top left: Markovian simulation, top right: Markovian simulation simplex trajectory. Bottom left: Mean Field Dynamics simulation, bottom right: Mean Field Dynamics simplex trajectory.

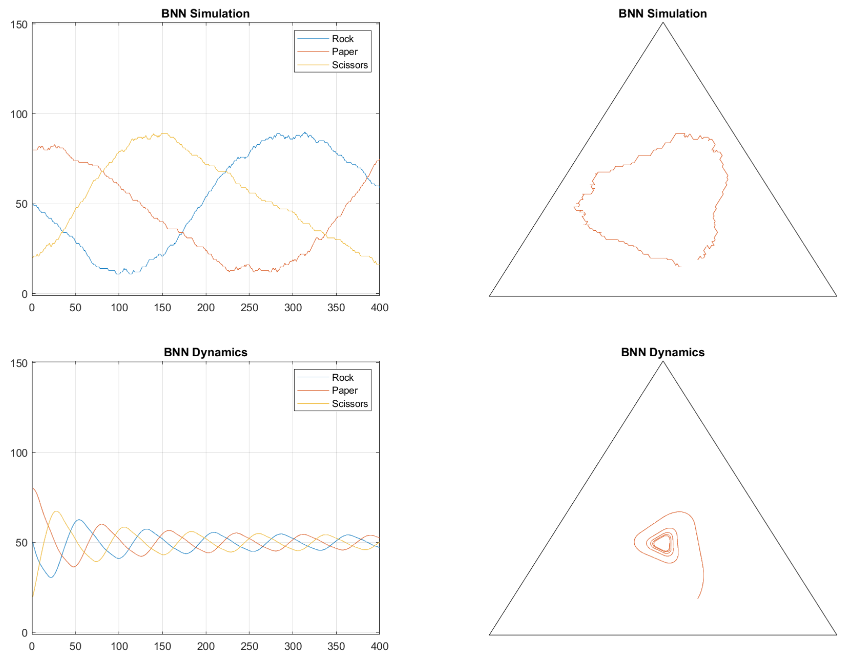


Figure 22: Rock-Scissor-Paper evolutionary game with three strategies, starting with 150 users of each strategy. The Brown-von Neumann-Nash revision protocol is used. Top left: Markovian simulation, top right: Markovian simulation simplex trajectory. Bottom left: Mean Field Dynamics simulation, bottom right: Mean Field Dynamics simplex trajectory.

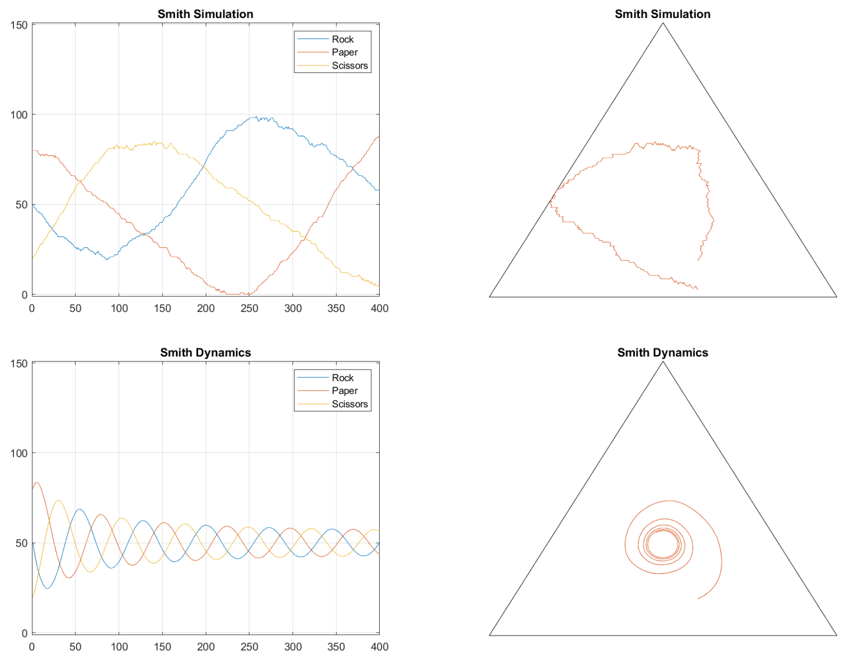
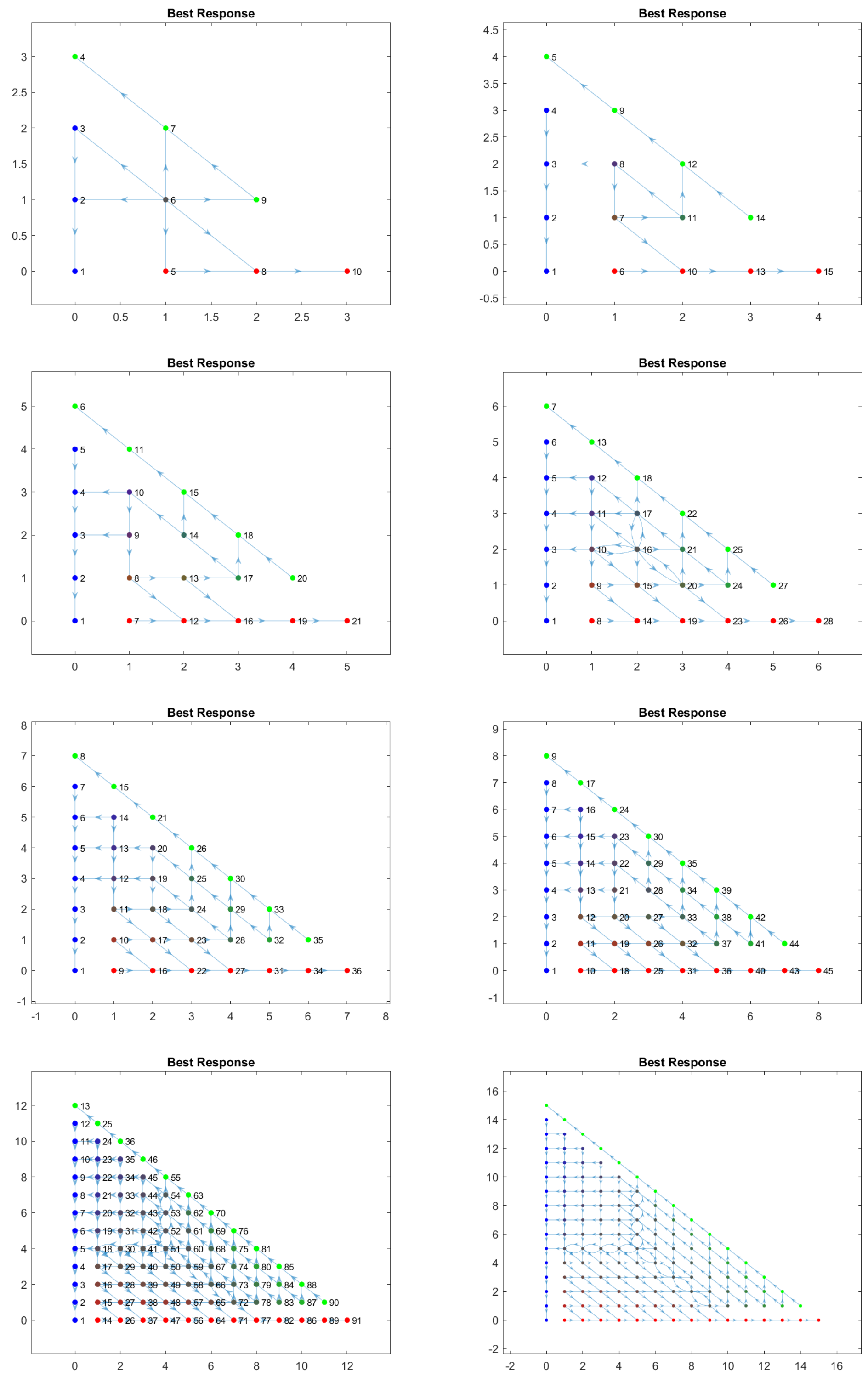


Figure 23: Rock-Scissor-Paper evolutionary game with three strategies, starting with 150 users of each strategy. The Smith revision protocol is used. Top left: Markovian simulation, top right: Markovian simulation simplex trajectory. Bottom left: Mean Field Dynamics simulation, bottom right: Mean Field Dynamics simplex trajectory.

### 5.4.2 Structure of the Transition Probability Matrix

The transition probability matrix $P$ corresponding to Rock-Paper-Scissors has an interesting structure, which is easiest to understand by looking at the STG's portrayed in Figure 24; these are the STG's obtained from the Best Response protocol, for $N \in \{3, 4, 5, 6, 7, 8, 12, 15, 60, 75\}$ players. Note the symmetric nature of the STG for all $N$ values. Also note that, for all $N$, the only absorbing states are the pure ones; this implies that in every realization the entire population will eventually adopt a single strategy.



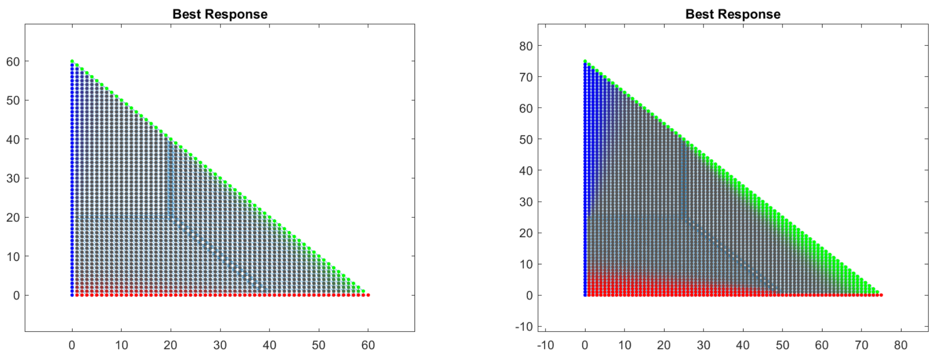


Figure 24: In this figure we see the STG's obtained from the Best Response protocol, for $N \in \{3, 4, 5, 6, 7, 8, 12, 15, 60, 75\}$ players. Note the symmetric nature of the STG for all $N$ values. Also note that, for all $N$, the only absorbing states are the pure ones; this implies that in every realization the entire population will eventually adopt a single strategy.

## References

[1] R. Axelrod and W. D. Hamilton. "The evolution of cooperation." *Science*, vol. 211, pp. 1390-1396, 1981.

[2] J. Hofbauer and K. Sigmund. *Evolutionary games and population dynamics.* Cambridge University Press, 1998.

[3] J.G. Kemeny and J. L. Snell. *Finite markov chains.* Princeton, NJ: van Nostrand, 1969.

[4] P. Mathieu, B. Beaufils and J.P. Delahaye. "Studies on Dynamics in the Classical Iterated Prisoner's Dilemma with Few Strategies". In: Fonlupt, C., Hao, JK., Lutton, E., Schoenauer, M., Ronald, E. (eds) *Artificial Evolution. AE 1999. Lecture Notes in Computer Science*, vol. 1829. Springer, Berlin, Heidelberg, 1999.

[5] W.H. Sandholm, *Population Games and Evolutionary Dynamics.* MIT Press, 2010.

[6] W.H. Sandholm. "Local stability under evolutionary game dynamics". *Theoretical Economics*, vol. 5, pp. 27-50, 2010.

## A Function Reference

The following is a list of the EGT functions and their intended use. Syntax details (input and output variables etc.) is only given for the main functions; for the rest of them, look at the corresponding Matlab files of the same name, which contain full description of the syntax.

### A.1 Main Functions

#### A.1.1 Basic

EGT contains four basic functions:

1. `[P,S]=EGTMarkP(B,Dynamic,M,N,eta)` computes the transition probability matrix of an evolutionary game.
2. `[s,x]=EGTMarkSim(P,S,s0,J)` simulates an evolutionary game using the transition probability matrix.
3. `[s,x]=EGTMFDyn(B,Dynamic,s0,J,h,eta)` computes the mean field dynamics of an evolutionary game.
4. `[s,x]=EGTSim(B,Dynamic,s0,J,eta)` simulates an evolutionary game.

The input variables are as follows.

| | | |
|---|---|---|
| `B` | : | Payoff matrix |
| `Dynamic` | : | Dynamic |
| `P` | : | Transition probability matrix |
| `S` | : | Srategy state space matrix |
| `s0` | : | Initial state |
| `M` | : | Number of strategies |
| `N` | : | Number of players |
| `J` | : | Number of generations |
| `h` | : | Integration step |
| `eta` | : | Noise parameter |

The output variables are as follows.

| | | |
|---|---|---|
| `P` | : | Transition probability matrix |
| `S` | : | Srategy state space matrix |
| `s` | : | Strategy state sequence (across generations) |
| `x` | : | Frequency sequence (across generations) |

### A.1.2 Dynamics

These functions are used by `EGTMFDyn`. They all work similarly to provide the update term for the various dynamics. For details of their syntax look inside the corresponding `*.m` files.

| **Function** | **Dynamics** |
|---|---|
| `DynBNN` | Brown-von Neumann-Nash |
| `DynLogit` | Logit |
| `DynRep1` | Replicator Type 1 |
| `DynRep2` | Replicator Type 2 |
| `DynSmith` | Smith |

### A.1.3 Rate Matrices

These functions are used by `EGTMarkP`. They all work similarly to provide the the rate matrix $R$ for the various protocols. For details of their syntax look inside the corresponding `*.m` files.

| **Function** | **Protocol** |
|---|---|
| `MakeRateBNN` | Brown-von Neumann-Nash |
| `MakeRateBR` | Logit |
| `MakeRateLogit` | Replicator Type 1 |
| `MakeRateRep1` | Replicator Type 2 |
| `MakeRateSmith` | Smith |

### A.1.4 Simulation

These functions are used by `EGTSim`. They all work similarly to to simulate a generation update for the various dynamics. For details of their syntax look inside the corresponding `*.m` files.

| **Function** | **Protocol** |
|---|---|
| `SimBNN` | Brown-von Neumann-Nash |
| `SimBR` | Best Response |
| `SimLogit` | Logit |
| `SimRep1` | Replicator Type 1 |
| `SimRep2` | Replicator Type 2 |
| `SimSmith` | Smith |

## A.2 Auxiliary Functions

### A.2.1 Plotting

These functions are used for plotting. For details of their syntax look inside the corresponding `*.m` files.

| | |
|---|---|
| `GrDyn` | Plots a sequence of frequency vectors $\mathbf{x}(1)$, ..., $\mathbf{x}(J)$, generated by `EGTMFDyn`. |
| `GrSim` | Plots a sequence of generation vectors $\mathbf{s}(1)$, ..., $\mathbf{s}(J)$, generated by `EGTSim`. |
| `GrMrk` | Plots the State Transition Graph (STG) of a Markov chain with transition matrix P as well as the plot of a strategy state sequence produced by the same P. Only works for three strategies. |
| `GrSimplex` | Plots a sequence of frequency vectors $\mathbf{x}(1)$, ..., $\mathbf{x}(J)$, generated by `EGTMFDyn`, on the 3-simplex. Only works for three strategies. |

### A.2.2 Creation Auxiliary Quantities

These create various quantities which are required by the main functions. For details of their syntax look inside the corresponding `*.m` files.

| | |
|---|---|
| `MakeNumState` | Given a state $\mathbf{s}$, it computes a standardized number. Only works for three strategies. |
| `MakePayoff` | Computes the payoff matrix $B$. |
| `MakeQ` | Computes the strategy payoff vector $\mathbf{Q}(\mathbf{s})$, for all $\mathbf{s} \in S$. |
| `MakeS` | Creates the state space $S$ as a matrix. Only works for three strategies. |
| `MakeSfromZ` | Computes a state vector $\mathbf{s}$ from a population vector $\mathbf{z}$. Only works for three strategies. |
| `MakeZfromS` | Computes a population vector $\mathbf{z}$ from a state vector $\mathbf{s}$. Only works for three strategies. |

## A.3 Strategy Functions

Given an iterated game history, these functions compute a player's next move when the he uses the strategy implied by the function name (and explained in the respective files). The strategies have been copied from [4]; some are well known, others not. We only give here a partial list, of the better known strategies.

| | |
|---|---|
| `AllC` | Always cooperate |
| `AllD` | Always defect |
| `Grim` | Cooperate in every round, until the opponent defects and then defect in all subsequent rounds |
| `Pavlov` | When the player and his opponent play the same (resp. different) move the player defects (resp. cooperates) in the next round |
| `TitForTat` | Cooperates in the first round, then always plays the opponent's previous move |

It is relatively easy to write your own strategies. All strategy files follow the same input / output convention; just open one of them, rename it and change the logic, to implement your own strategy.

# B The GameTheory2025 Collective

The origin of this toolbox was a team project assigned to the students of the Game Theory class taught in Spring 2025, at the Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Greece. Eight teams contributed their versions of the toolbox. Their members form the "*GameTheory2025 Collective*" and they are the following:

| Team No. | Members |
|---|---|
| 0 | Ath. Kehagias (Professor) |
| 1 | I. Lazaridis, I. Vardakis, K. Vlahakos |
| 2 | E. Delopoulos, D. Vaporis, V. Vlasios |
| 3 | G. Daios, G. Mousses, S. Nazlidis, K. Papadakis |
| 4 | T. Badas, G. Binos, G. Hatzilyras |
| 5 | N. Gaitanidis, D. Moutaftsidis, N. Stamatis |
| 6 | T. Billas, D. Kokkinakos, H. Marinopoulos |
| 7 | G. Gioulekas, E. Koumparidou, N. Kousta |
| 8 | I. Konstantakis, V. Kordis, I. Matsrogiannis |

The Evolutionary Game Toolboxes created by each team and submitted in the class can be found on GitHub. They can be considered as preliminary versions of the current toolbox. The original contributions were reworked by Ath. Kehagias and evolved (pun intended) to the current version of EGT.

## C EGT-similar Toolboxes

Several toolboxes / libraries comparable to EGT are available; each has its own strengths. Here is a partial list.

1. ABED. A NetLogo application. "ABED is free and open-source software for *simulating* evolutionary game dynamics in finite populations."

   Link: `https://luis-r-izquierdo.github.io/abed-1pop/`

2. Axelrod. A Python library. "A Python library with the goals of enabling the reproduction of previous Iterated Prisoner's Dilemma research as easily as possible."

   Link: `https://github.com/Axelrod-Python/Axelrod`

3. EGT Tools. A Python library. "EGTtools provides a centralized repository with analytical and numerical methods to study/model game theoretical problems under the Evolutionary Game Theory (EGT) framework."

   Link: `https://egttools.readthedocs.io/en/latest/`

4. Evolutionary Games. An R package. " A comprehensive set of tools to illustrate the core concepts of evolutionary game theory, such as evolutionary stability or various evolutionary dynamics."

   Link: `https://cran.r-project.org/web/packages/EvolutionaryGames/index.html`

5. ipd. It is a modern Python version of Prison. "It contains Python code and Jupyter notebooks to understand easily how to experiment computational game theory, build and evaluate strategies at the iterated prisoner's dilemma (IPD) and gives many tools for sophisticated experiments. It is also useful to reproduce most of the experiences of our research articles cited in the bibliography."

   Link: `https://github.com/cristal-smac/ipd/blob/master/README.md`

6. NashPy. A Python library. "A python library for 2 player games."

   Link: `https://nashpy.readthedocs.io/en/stable/`

7. PDToolbox. A Matlab toolbox. "PDToolbox is a matlab implementation of some evolutionary dynamics from game theory."

   Link: `https://github.com/carlobar/PDToolbox_matlab`

8. WinPri and Prison. These provided the original motivation for writing EGT. They are two forgotten gems, created in the 1990's by B. Beaufils, J.P. Delahaye and P. Mathieu. WinPri is a Windows 3.1 program and can only be run on an emulator (e.g., DosBox). Prison is written in ANSI C and can be compiled to run on modern Windows systems. I have not been able to find a live link for these programs.