ARISTOTLE UNIVERSITY OF THESSALONIKI

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

DEPARTMENT OF COMPUTERS & ELECTRONICS

MAY 2025

# COURSEWORK

FOR THE COURSE

# GAME THEORY

8TH SEMESTER

(5TH DELIVERABLE)

GROUP 4:

( I ) ΜΠΙΝΟΣ ΓΕΩΡΓΙΟΣ
AEM: 9333
email: gampinos@ece.auth.gr

( II ) ΜΠΑΔΑΣ ΤΡΥΦΩΝ
AEM: 10680
email: tmpadasn@ece.auth.gr

( III ) ΧΑΤΖΗΛΥΡΑΣ ΓΙΩΡΓΟΣ
AEM: 10820
email: cgiorgos@ece.auth.gr

# (A) INTRODUCTION

## (A.1) Description of the problem

In this coursework, we attempt to create a functional toolbox in MATLAB, capable of running simulations of the Classic Iterated Prisoner's Dilemma (CIPD), as well as the Axelrod Tournament. This game, very simple to describe, covers a large scale of real-life situations and seems to catch the definition of conflicts of interests. Thus, a lot of different kinds of work have been done on it, involving not only mathematicians, but also social, zoological, biological as well as computer scientists. The game becomes the most used theoretical model for studying the cooperation and the evolution of cooperation in the population of agents.

The game, called the Prisoner's Dilemma, could be described very simply in the following way: let us meet two artificial agents having two choices (two strategies):

(a) COOPERATE, let us write C, and say to be nice.
(b) DEFECT, let us write D, and say to be naughty.

The payoff for each player depends on the moves played by the two agents. Tab. 1 names the score of each case. The dilemma comes when exploitation of one by the other (T) is better paid than cooperation between the two (R), which itself pays more than a case where the two tried two exploit each other (P), which finally is a better choice than to be exploited (S). This can be formalized as:

$$\boxed{S < P < R < T}$$

The dilemma stands on the fact that individual interest (Nash equilibrium) differs from collective one (Pareto issues). The one-shot game, involving rational agents and pure strategies, is solved in Game Theory by the Nash equilibrium, which is to always betray its partner: choosing the D strategy.

In an iterated version players meet each other more than one time. The payoff of an agent is then simply the sum of each of its meeting's payoff. The game is called the Classical Iterated Prisoner's Dilemma (CIPD).

## (A.2) Related work

The interest in analyzing the CIPD stems mainly from the task of mathematically proving why most species in nature choose to cooperate with each other, rather than defect, something that boosts their chances of survival, as we have observed. Several papers that tackle this exact task have been published, like our referenced paperworks [2] and [3].

# (B) QUICK START GUIDE

As stated in section (A), this project is a MATLAB toolbox that can simulate the Classic Iterated Prisoner's Dilemma (CIPD), as well as the Axelrod Tournament. The link to access the toolbox and the related documents, including this report is the following:

https://github.com/tmpadasn/EvolutionaryGamesToolbox

For your convenience, we have made some ready-to-run script files that you can run to reproduce the figures seen in referenced paperwork [1]. These scripts are located in the Code folder under the names FigX_TourType, where X is the number of the figure and TourType is the type of simulation being run (Fitness or Imitation).

# (C) FITNESS DYNAMICS

## (C.1) The process

Two kinds of experimentation are used in literature to evaluate strategies for the CIPD:

The basic one, is to make a two-by-two round robin tournament between strategies. The payoff of each one would be the total sum of each iterated game 1. A ranking could then be computed according to the score of each strategy. The higher a strategy is ranked, the better it is. As shown in previous work, [4], some cycles between strategies may be found (A better than B, which is better than C which is better than A), the order created by this method cannot be considered as total.

The second kind of experimentation is a kind of imitation of the natural selection process, and is closely related to population dynamics, but in a completely discrete context. Let us consider a population of N players, each one adopting a particular strategy. At the beginning we consider that each strategy is equally represented in the population. Then a tournament is made, and good strategies are favored, whereas bad ones are disadvantaged, by a proportional population redistribution. This redistribution process, also called a generation, is repeated until an eventual population stabilization, i.e. no changes between two generations. A good strategy is then a strategy which stays alive in the population for the longest possible time, and in the biggest possible proportion. This kind of evaluation quotes the robustness of strategies. This looks like prey-predator model, but is not. The number of species involved is not limited to two, interactions between, or into, species are much more complex, and global population is fixed. Once a population has disappeared it has no way to reappear: there is no stochastic perturbations nor n population distribution, nor in strategies description.

**Algorithm 1: TourTheFit**

**Input:** Payoff bimatrix $B$, strategies $\{\sigma_1, \ldots, \sigma_S\}$, initial population $\mathbf{POP}_0$, parameters $(T, J)$

Set $S = |\{\sigma_1, \ldots, \sigma_S\}|$.
Set total population $\Pi = \sum_{i=1}^{S} \mathbf{POP}_0(i)$.
Initialize population $\mathbf{POP}(1, :) = \mathbf{POP}_0$.
Compute payoff matrix $V$ :
  for $i, j \in \{1, \ldots, S\}$
    $V(i, j) =$ payoff to $\sigma_i$ vs $\sigma_j$ in $B$ (over $T$ rounds)
  end for
for $n \in \{1, \ldots, J\}$ do
  Let $W = \mathbf{POP}(n, :)$.
  for $i \in \{1, \ldots, S\}$
    Evaluate fitness:
    $g_i = W \cdot V(i, :)^T - V(i, i)$
  end for
  Compute total payoff: $\text{total} = \sum_{i=1}^{S} W(i) \cdot g_i$
  if total $= 0$ then
    $\mathbf{POP}(n + 1 : J, :) = W$
    break
  end if
  Update next population:
    $\mathbf{POP}(n + 1, :) = \left\lfloor \frac{\Pi \cdot W \cdot g}{\text{total}} \right\rfloor$
  Store best strategy:
    $\mathbf{BST}(n) = \arg\max_i g_i$
end for
return $\mathbf{POP}, \mathbf{BST}, g$ (fitness history)

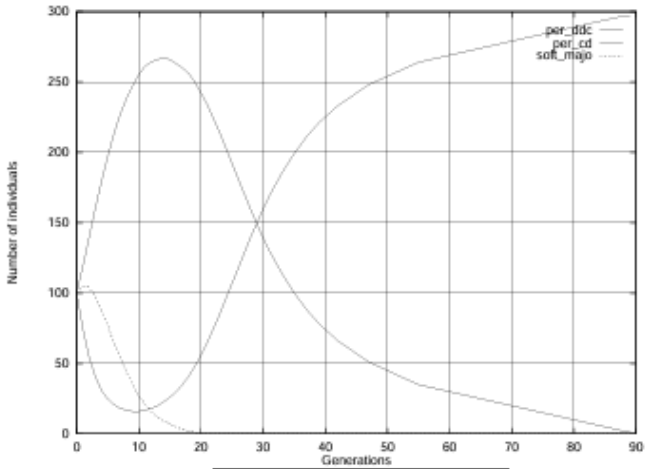
**Algorithm 2: TourSimFit**

**Input:** Payoff bimatrix $B$, strategies $\{\sigma_1, ..., \sigma_S\}$, initial population $\mathbf{POP}_0$, parameters $(T, J)$

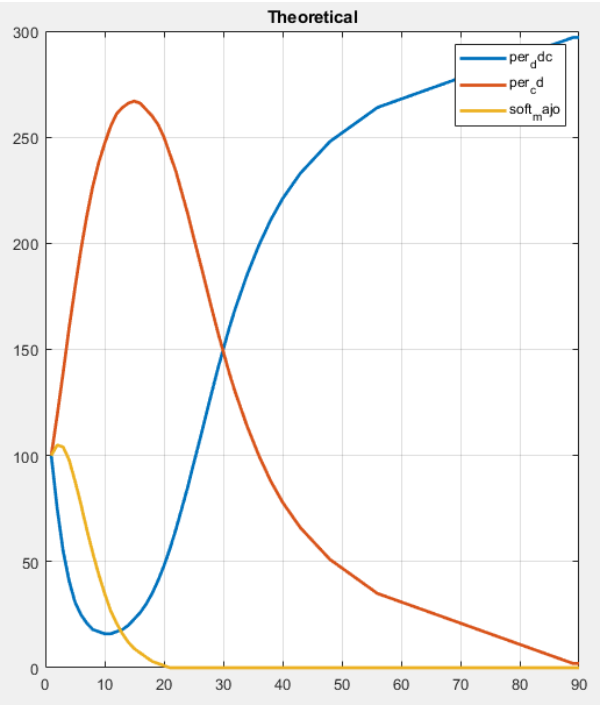
Set $S = |\{\sigma_1, ..., \sigma_S\}|$.
Initialize population $\mathbf{POP}(1, :) = \mathbf{POP}_0$.
for $n \in \{1, ..., J\}$ do
  Let $\mathbf{counts} = \mathbf{POP}(n, :)$.
  Initialize payoffs: $\mathbf{payoff} = \mathbf{0}_{1 \times S}$.
  for $i, j \in \{1, ..., S\}$ do
    if $\mathbf{counts}(i) > 0$ and $\mathbf{counts}(j) > 0$ then
      Play $T$ rounds between $\sigma_i$ and $\sigma_j$.
      $[p_1, p_2] = \text{MatchPayoff}(\sigma_i, \sigma_j, B, T)$
      if $i = j$ then
        $n_{\text{Games}} = \mathbf{counts}(i) \cdot (\mathbf{counts}(i) - 1)/2$
      else
        $n_{\text{Games}} = \mathbf{counts}(i) \cdot \mathbf{counts}(j)$
      end if
      $\mathbf{payoff}(i) = \mathbf{payoff}(i) + p_1 \cdot n_{\text{Games}}$
      $\mathbf{payoff}(j) = \mathbf{payoff}(j) + p_2 \cdot n_{\text{Games}}$
    end if
  end for
  Compute fitness: for $s \in \{1, ..., S\}$
    if $\mathbf{counts}(s) > 0$ then
      $\text{fitness}(s) = \mathbf{payoff}(s)/\mathbf{counts}(s)$
    end if
  $\mathbf{FIT}(n, :) = \text{fitness}$
  Store best strategy:
    $\mathbf{BST}(n) = \arg\max_s \text{fitness}(s)$
  if $n < J$ then
    Update next generation:
      $\mathbf{POP}(n + 1, :) = \text{round}\left(\frac{\text{fitness}}{\sum_{s=1}^{S} \text{fitness}(s)} \cdot \sum_{s=1}^{S} \mathbf{counts}(s)\right)$
  end if
end for
return $\mathbf{POP}, \mathbf{BST}, \mathbf{FIT}$

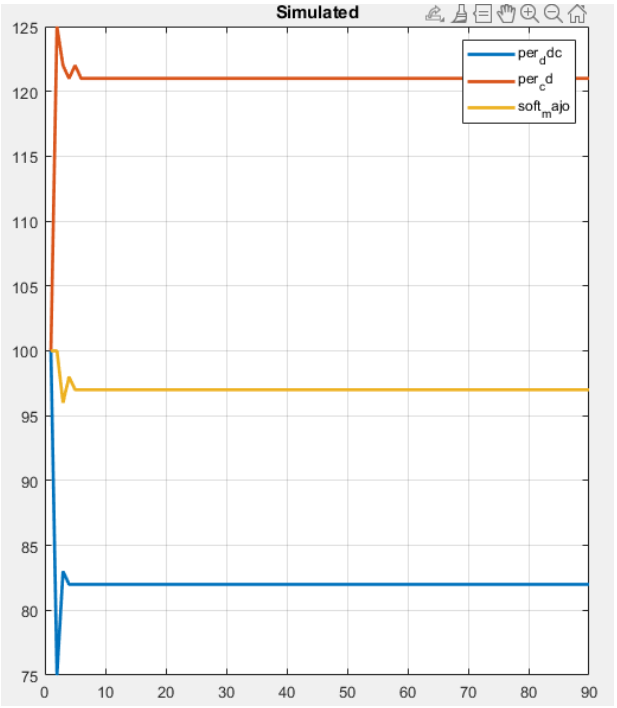
## (C.2) Experiments

Example 1




| | per-cd | per-ddc | soft-majo | Total | Init. Pop. |
|---|---|---|---|---|---|
| per_cd | 2000 | 1665 | 4000 | 7665 | 100 |
| per_ddc | 2500 | 1666 | 671 | 4837 | 100 |
| soft_majo | 1500 | 2331 | 3000 | 6831 | 100 |



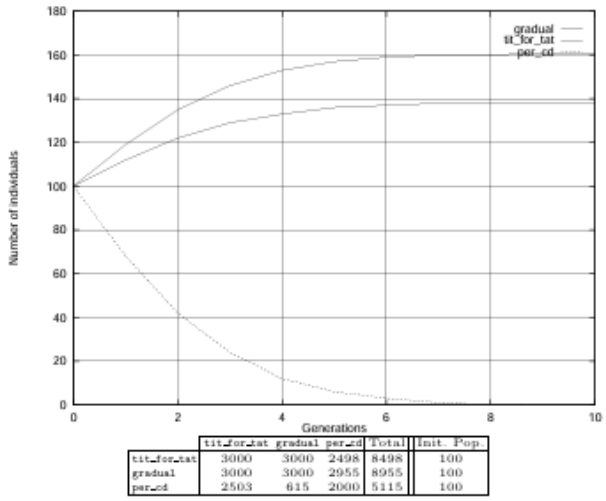

Script: Fig1_Fitness.m

Here we see the initial conclusion as to what the characteristics of a good strategy:

1. Nice (don't defect first)
2. Reactive
3. Forgiving
4. Not too clever (so the rest of the players understand their gameplan)

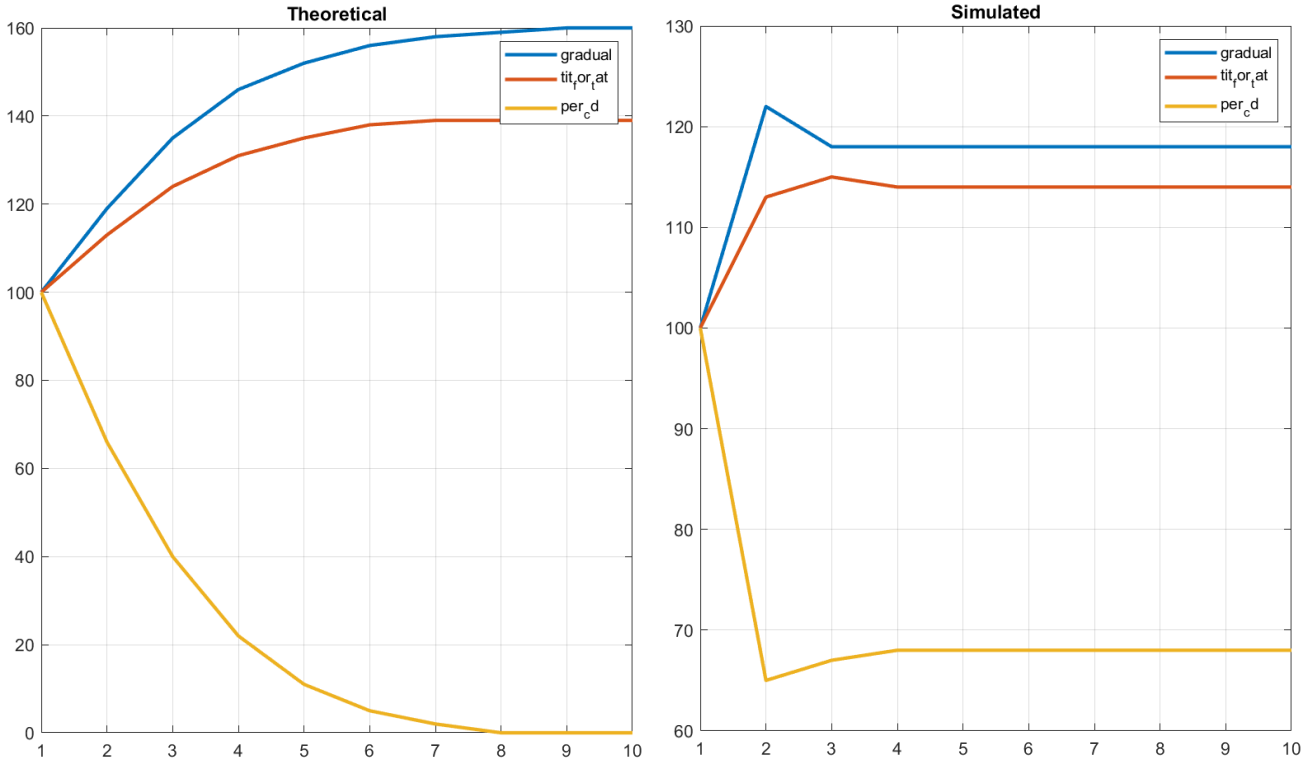
We have three strategies that all start with a population of 100. We see that while initially the defectors have an immediate payoff for defecting, later on the strategy with the characteristics above thrives. It should be noted that in certain scenarios, defectors can perform well, depending on what strategies they face up against and their populations (mainly if they're against cooperative strategies they can exploit).

Example 2




| | tit_for_tat | gradual | per_cd | Total | Init. Pop. |
|---|---|---|---|---|---|
| tit_for_tat | 3000 | 3000 | 2498 | 8498 | 100 |
| gradual | 3000 | 3000 | 2955 | 8955 | 100 |
| per_cd | 2503 | 615 | 2000 | 5115 | 100 |

**Fig. 2.** Monotonous convergence



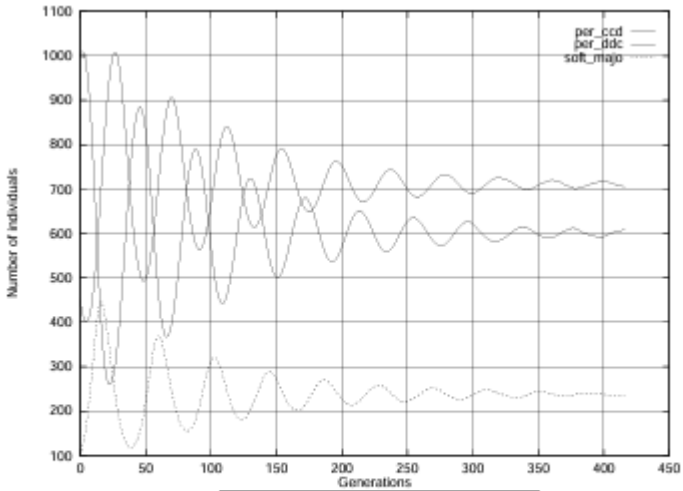

Script: Fig2_Fitness.m

Here we are observing the most common scenario; all the strategies show a *monotonous convergence* towards a stable population.

We have three different strategies that again all have an initial population of 100. We see that over time, the strategies' populations converge and stabilize, with little or no change.

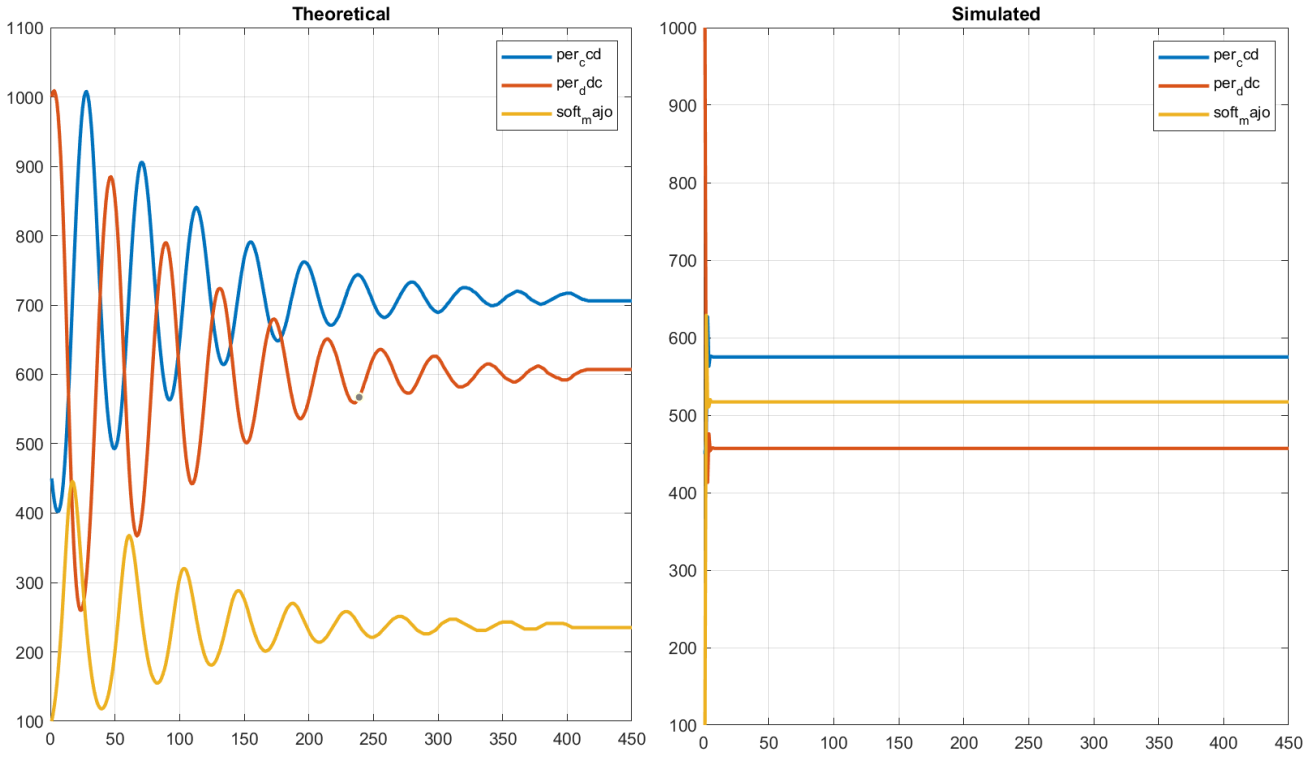
It's worth noting that we once again see the results of the previous example, possibly to a bigger scale. The defectors almost cease to exist, while the other two strategies reacted to the defectors and defected them back while cooperating with each other, resulting in their populations increasing.

Example 3




| | per_ccd | per_ddc | soft_majo | Total | Init. Pop. |
|---|---|---|---|---|---|
| per-ccd | 2334 | 1665 | 3666 | 7665 | 450 |
| per-ddc | 3335 | 1666 | 671 | 5672 | 1000 |
| soft-majo | 2001 | 2331 | 3000 | 7332 | 100 |

**Fig. 3.** Attenuated oscillatory movements




Script: Fig3_Fitness.m

Here we observe the scenario of *attenuated oscillations*, which means that the populations oscillate with a decreasing amplitude, resulting to their eventual convergence.

We see that the three strategies' are conflicting themselves with a lot of oscillations for the first 100 generations, and then they slowly find an equilibrium until generation 420, where they have completely stabilized.

Example 4

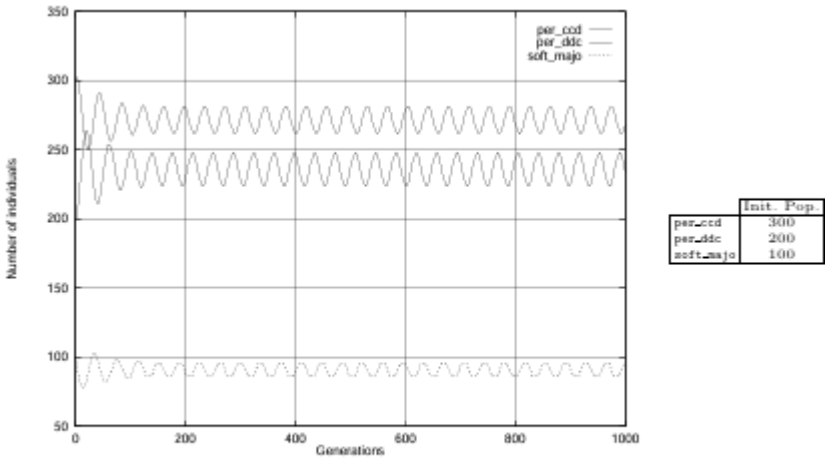
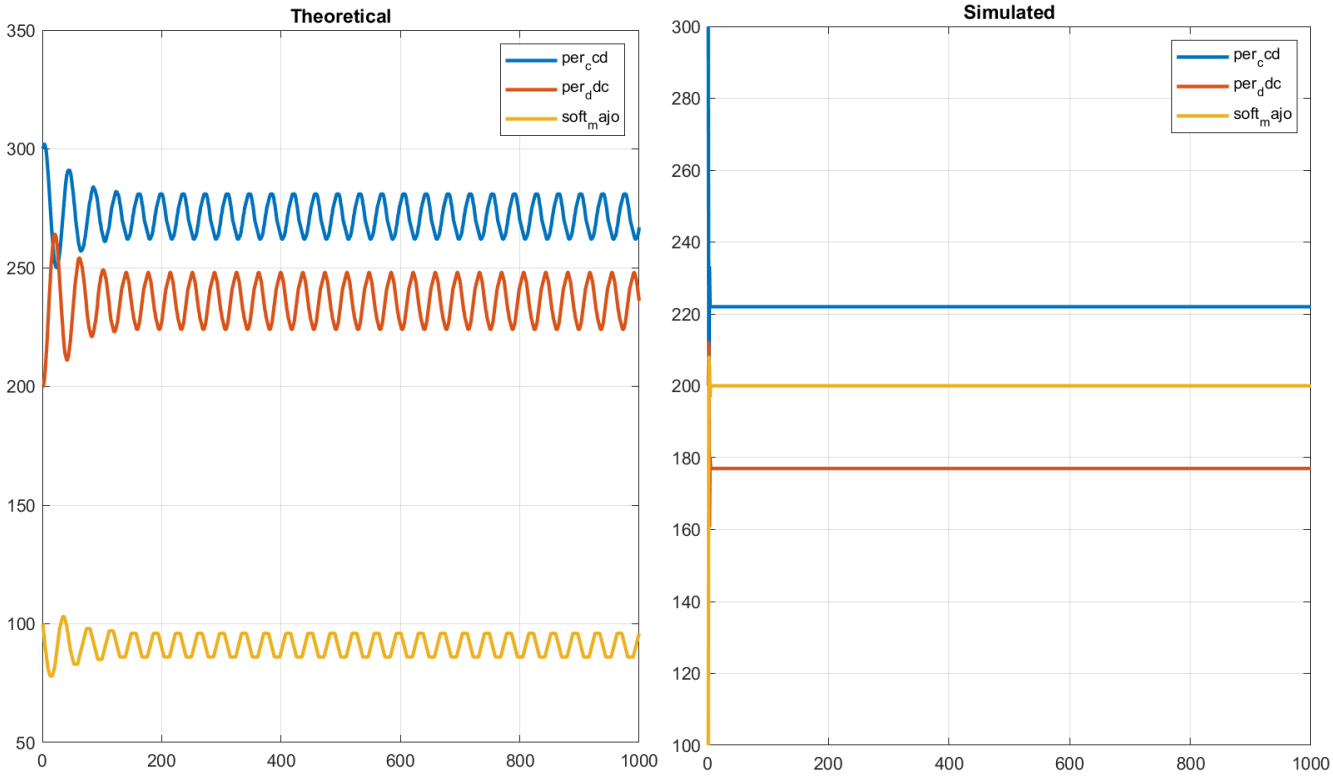



**Fig. 4.** Periodic movements




Script: Fig4_Fitness.m

We see the scenario where the populations oscillate, but rather than stabilize, they become *periodic*. This happens when the strategies make a cycle in the tournament:

A is better than B
B is better than C
C is better than A

In the example above, the strategies' populations constantly cycle, and specifically, we return to the same population numbers every 37 generations.

Example 5

| | Init. Pop. |
|---|---|
| per_cd | 300 |
| per_ddc | 400 |
| soft_majo | 200 |

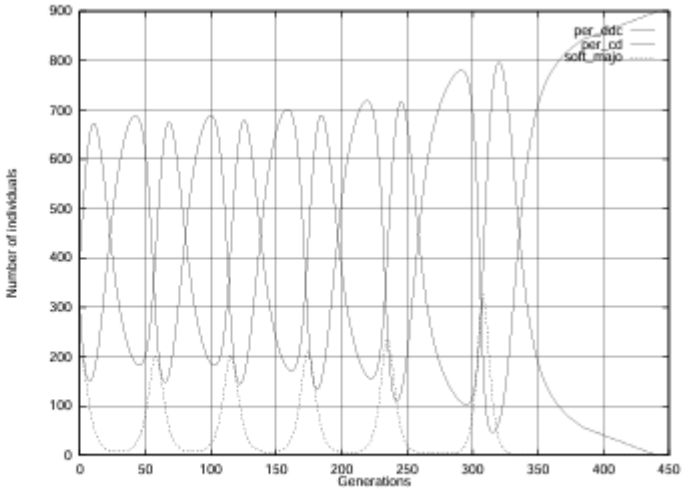
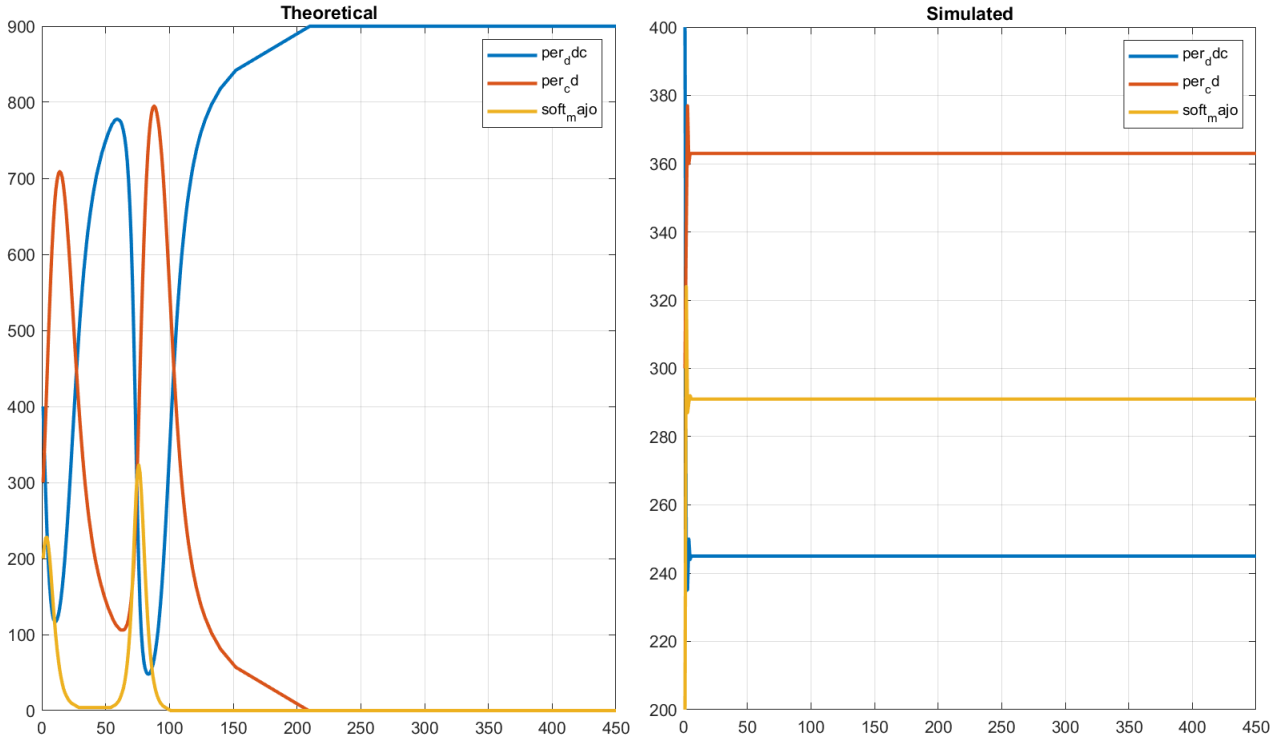



**Fig. 5.** Increasing oscillations

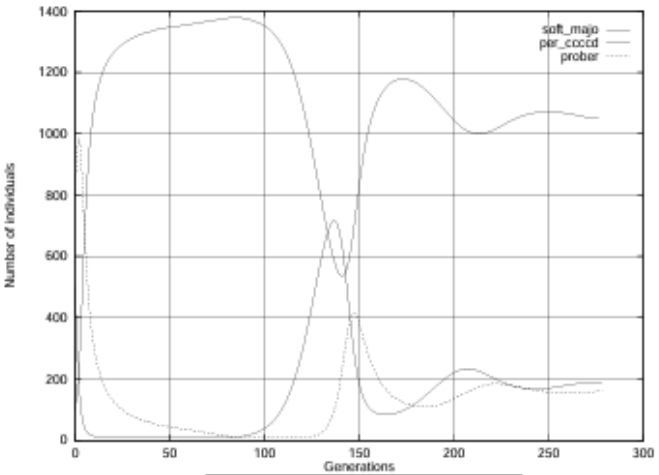



Script: Fig5_Fitness.m

Here we study *increasing oscillations*, which means that instead of the populations converging or cycling, the amplitude of their oscillations constantly increases, until one strategy reigns supreme.

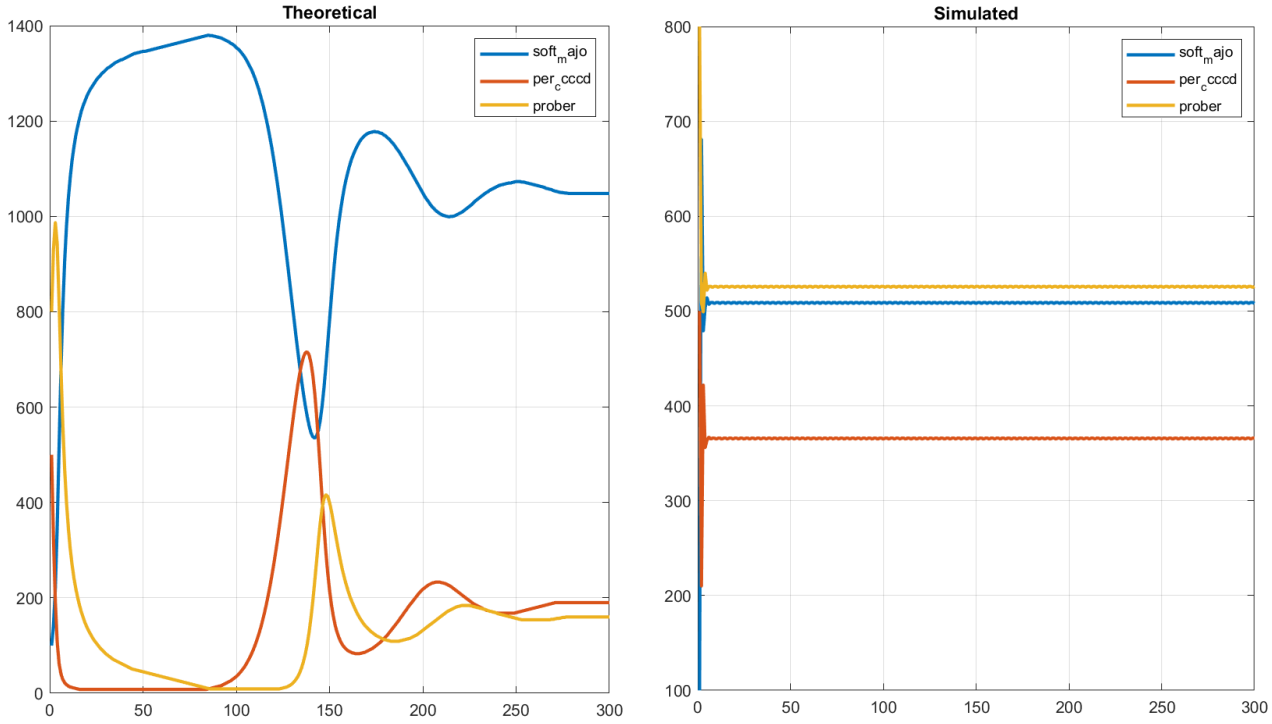
We see that these kinds of violent oscillations can be beneficial to non-cooperative strategies, as they thrive under the general disorder.

Example 6




| | soft_majo | per_cccd | prober | Total | Init. Pop. |
|---|---|---|---|---|---|
| soft_majo | 3000 | 2400 | 2999 | 8399 | 100 |
| per_cccd | 3400 | 2600 | 206 | 6206 | 500 |
| prober | 2999 | 4196 | 1004 | 8199 | 800 |

**Fig. 6.** Disordered oscillations




Script: Fig6_Fitness.m

Here we see the last type of oscillations possible during a tournament; disordered oscillations. The movements in these oscillations are disordered and they don't tend to last long.

In the example above, all three strategies come close to extinction during the first 250 generations, and afterwards they stabilize and an equilibrium is reached.

Example 7

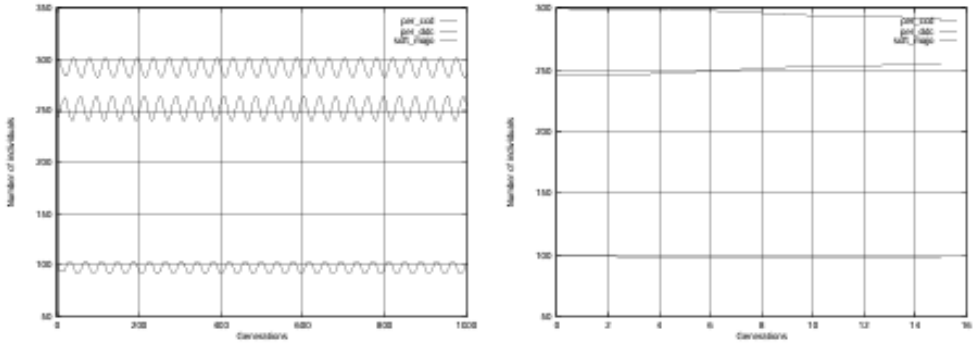
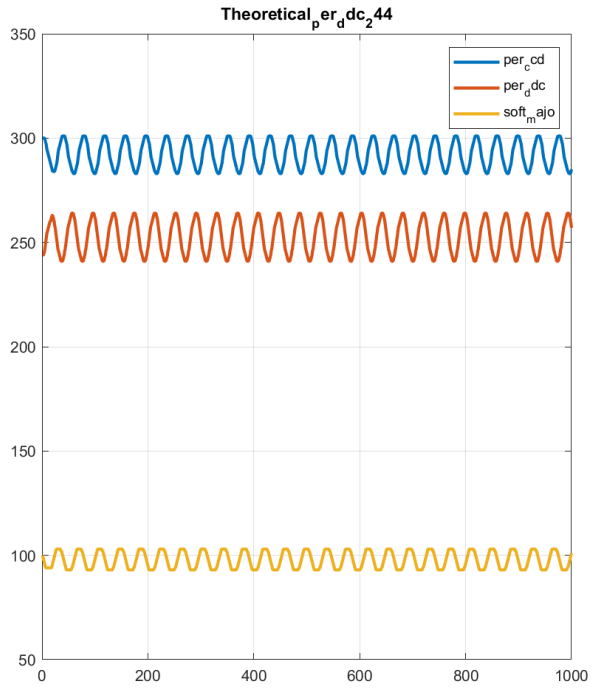
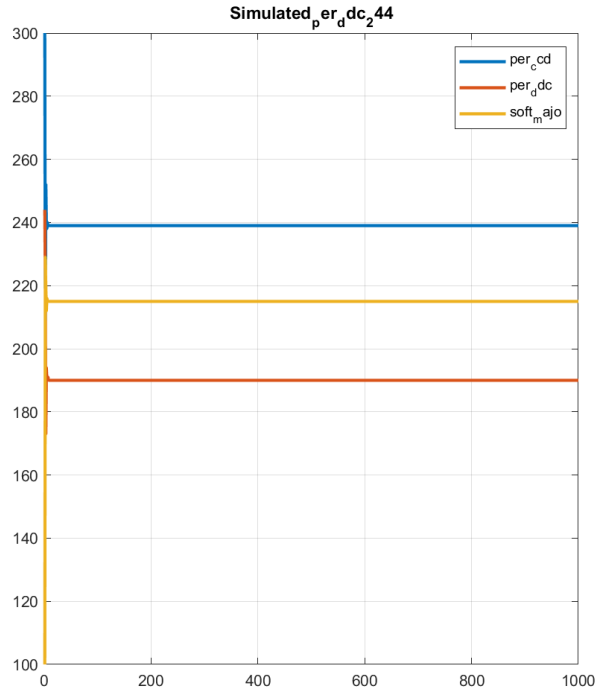
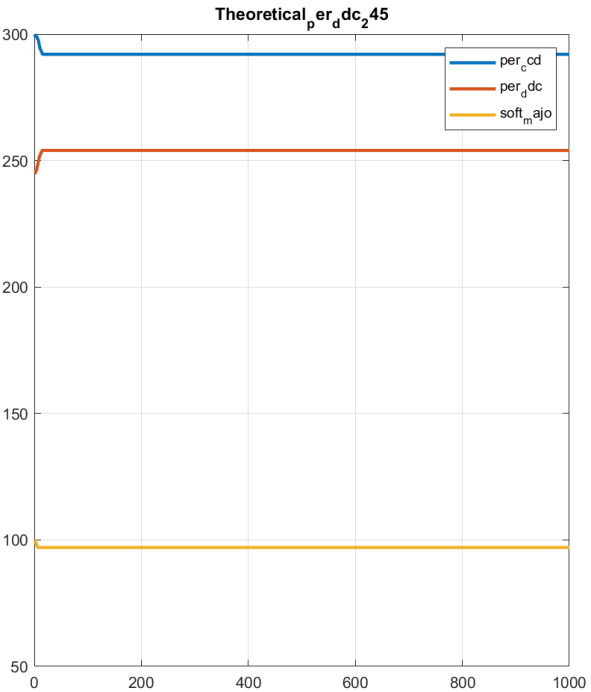
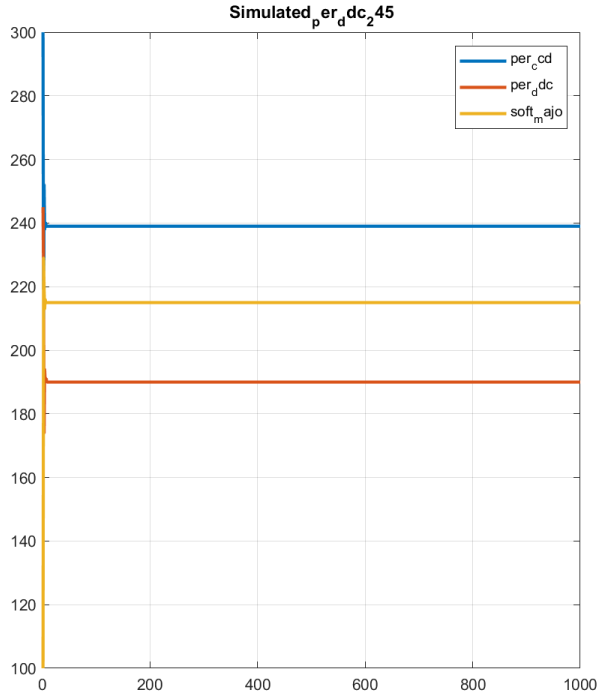


**Fig. 7.** Sensitivity of dynamics to population's size. All parameters are identical except for the initial size of `per_ddc` which is 244 on the left and 245 on the right



Script: Fig7_Fitness.m

In the following figures, we will see how the initial conditions of the tournament can wildly change its outcome. We first start by observing how the *population size* of the tournament change the end result.

In this scenario above, we first run the tournament with 300 individuals playing per_cdd, 100 playing soft_majo and 244 playing per_ddc (first pair of figures). Then, we run the exact same tournament again, with the only difference being that we add one more individual to play per_ddc (making them a *naughty* individual), making them 255 (second pair of figures).

We see that the addition of one individual changed the outcome of the tournament. From a periodic result where the populations constantly change, we end up with the populations instantly converging and stabilizing.

Example 8

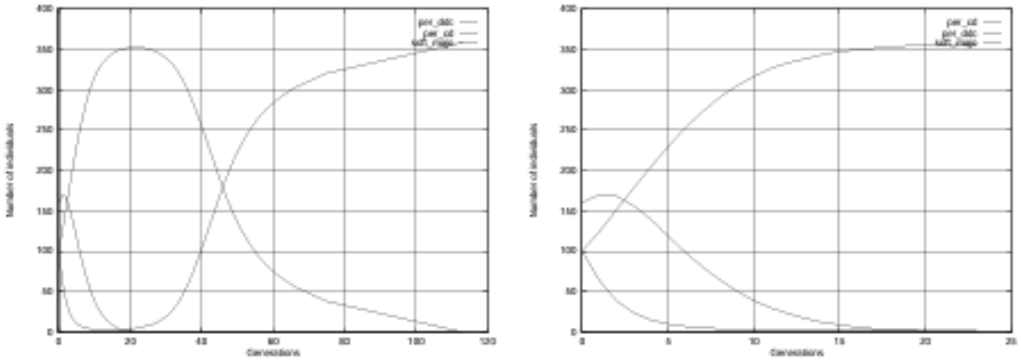
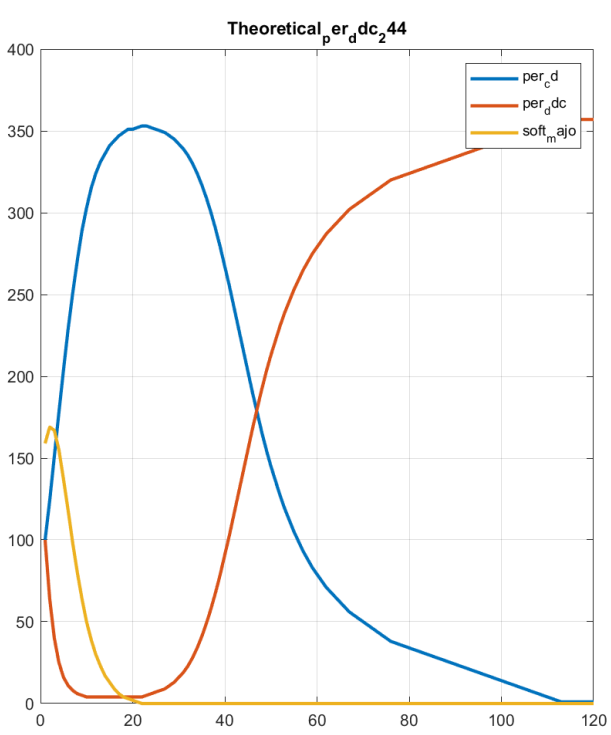
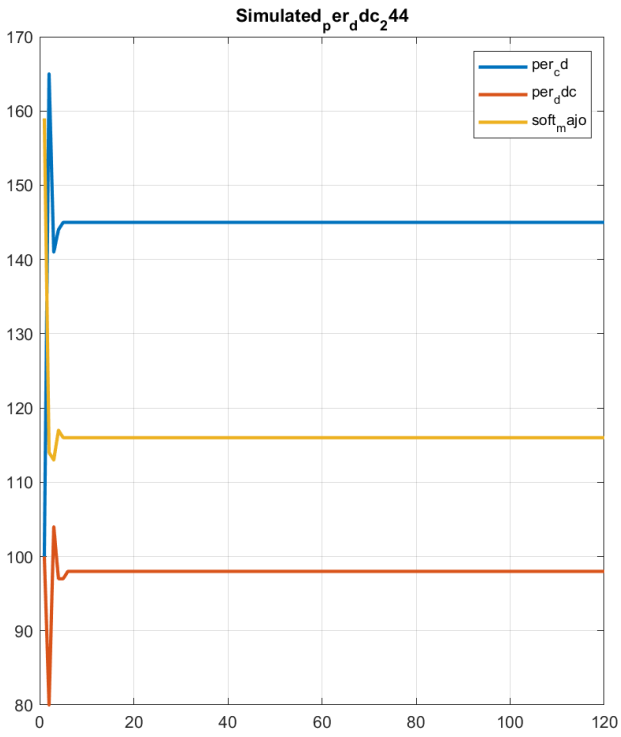
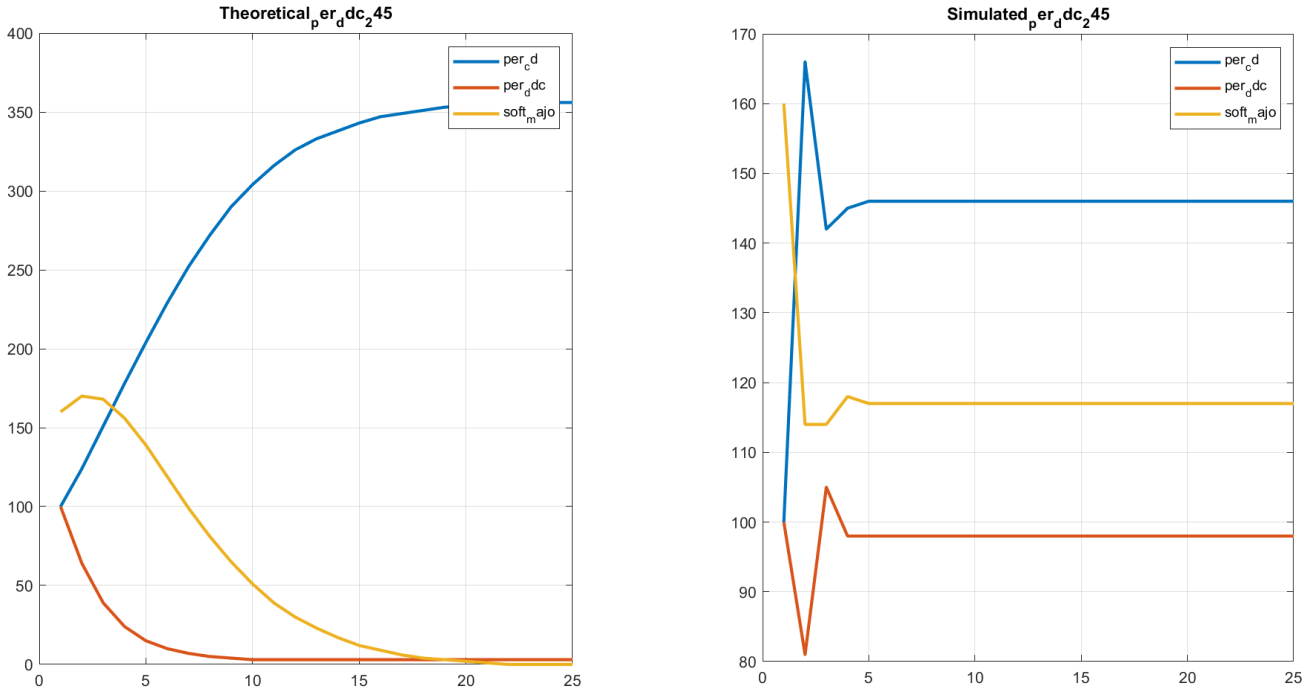


**Fig. 8.** Sensitivity of winner to population's size. All parameters are identical except for the initial size of `soft_majo` which is 159 on the left and 160 on the right.






Script: Fig8_Fitness.m

Here, we test again how the population size for each strategy influences how the tournament will play out. We start off with 100 per_ddc, 159 soft_majo and 100 per_cd and the scond time we run the tournament, we add one more soft_majo player.

We see that in the first tournament the winner is per_ddc, while in the second tournament, the winner changes, becoming per_cd. An interesting thing to note is that the strategy whose population changes is *never* the winner.

Example 9

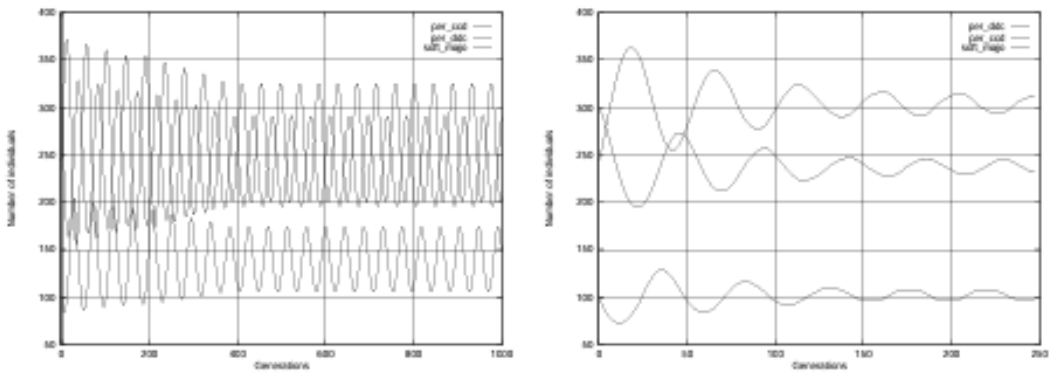
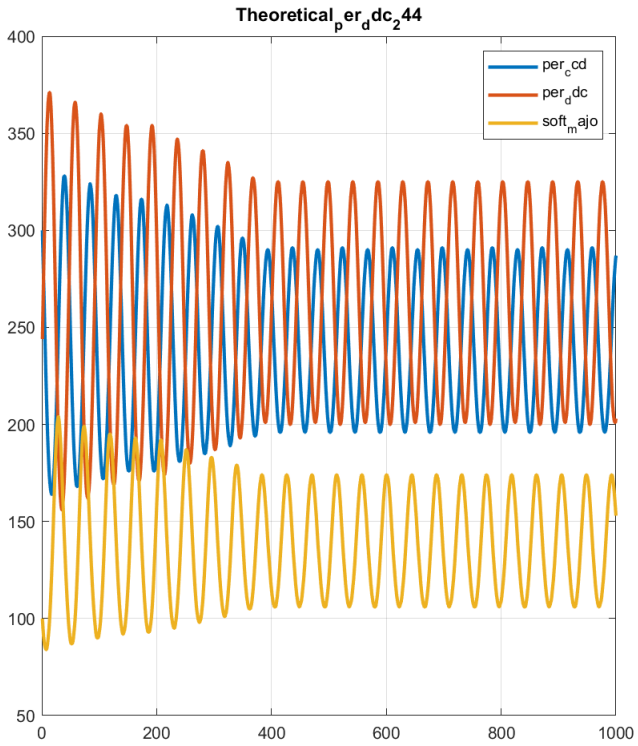
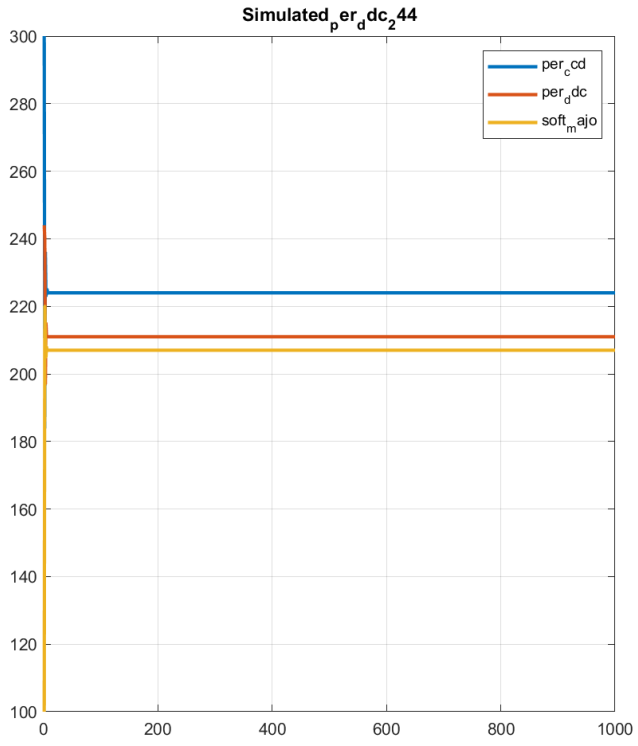
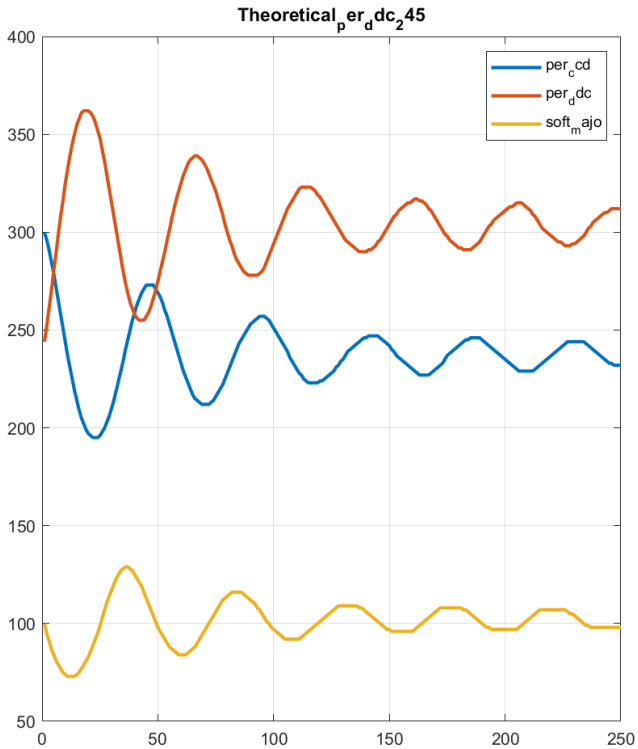
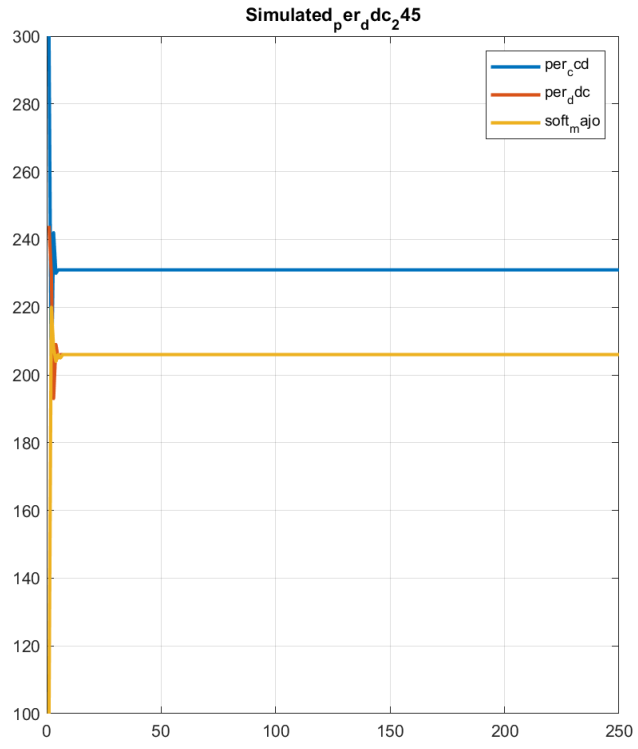


**Fig. 9.** Sensitivity to game length. All parameters are identical except for the game length which is 7 moves on the left and 6 on the right.




Script: Fig9_Fitness.m

In this example, we test how the total number of moves in the tournament affects the final outcome.

The first time we play the tournament, the game lasts for 7 moves while the second time we play it, it lasts for only six. We see that in the first scenario, the dynamic eventually ends up being periodic, while in the second scenario, it is an attenuated oscillation.

## Example 10

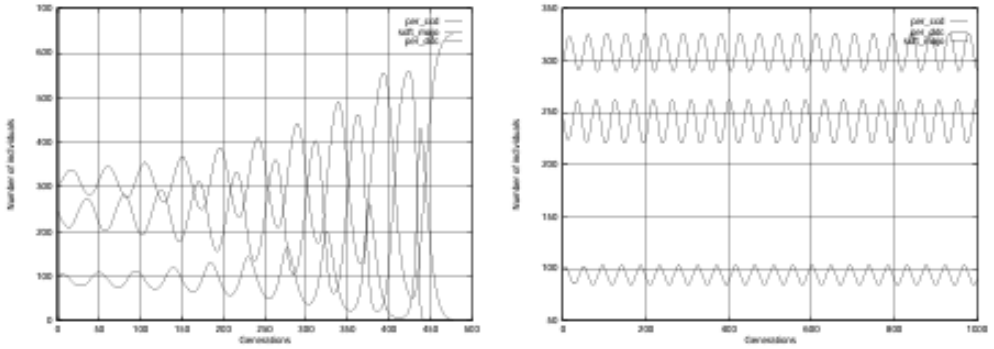
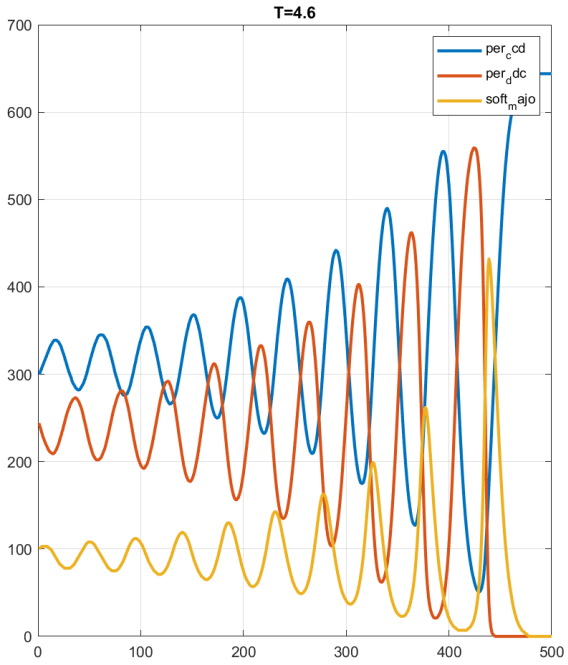
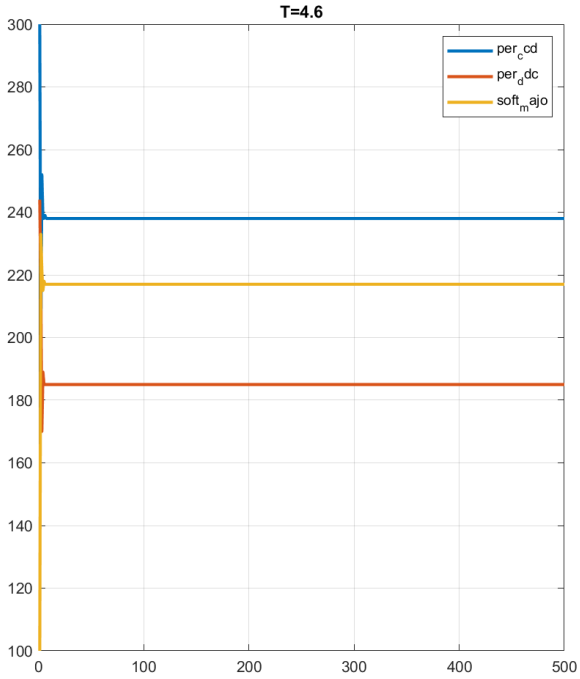
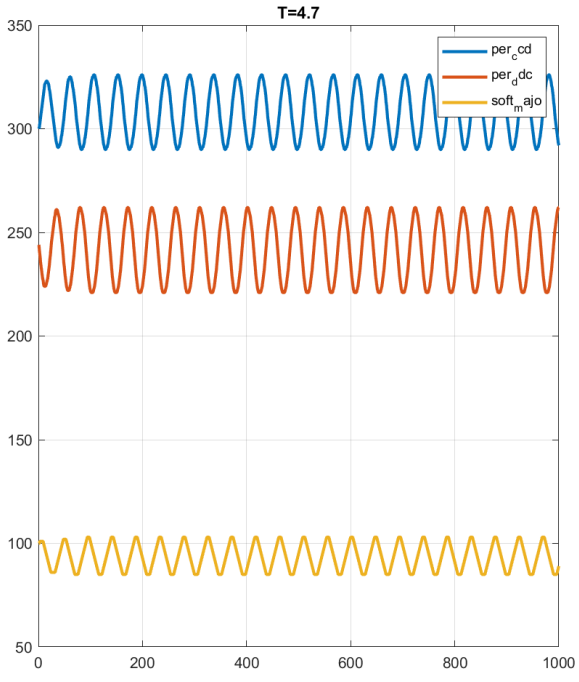
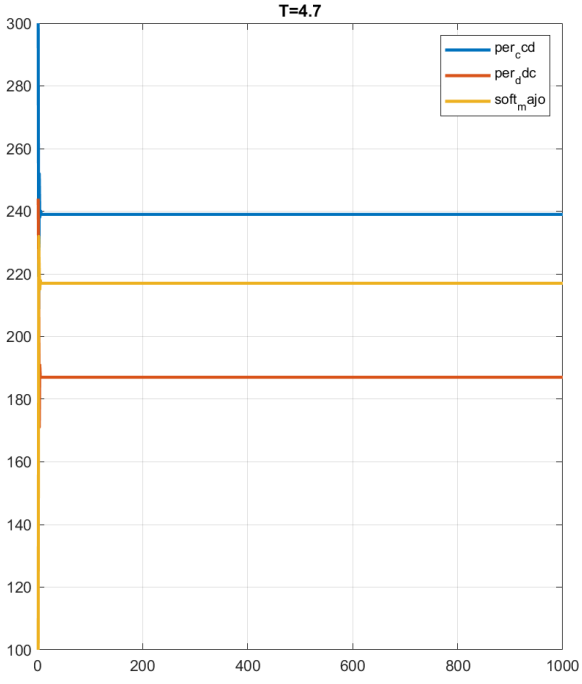


**Fig. 10.** Sensitivity to CIPD payoff. All parameters are identical except that T=4.6 on the left and T=4.7 on the right.



Script: Fig10_Fitness.m

This time, we change the payoff matrix in the classic Prisoner's Dilemma game to see the changes it causes.

The values R, P and S stay the same as in the classic Prisoner's Dilemma, but we change T's value; first to 4.6, and then to 4.7. When T=4.6, the dynamic becomes an increasing oscillation while when T=4.7, it becomes periodic.

Example 11

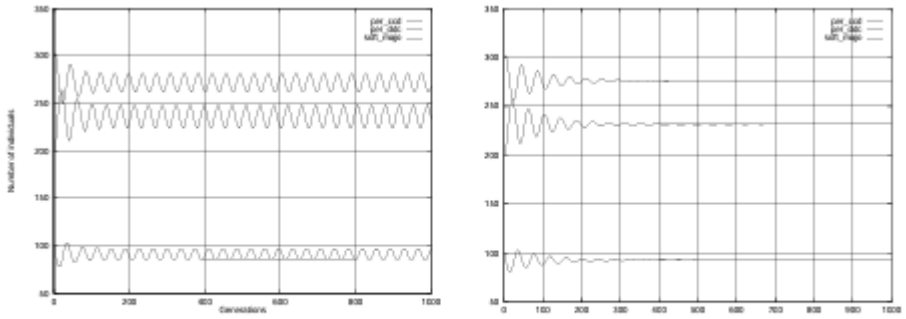
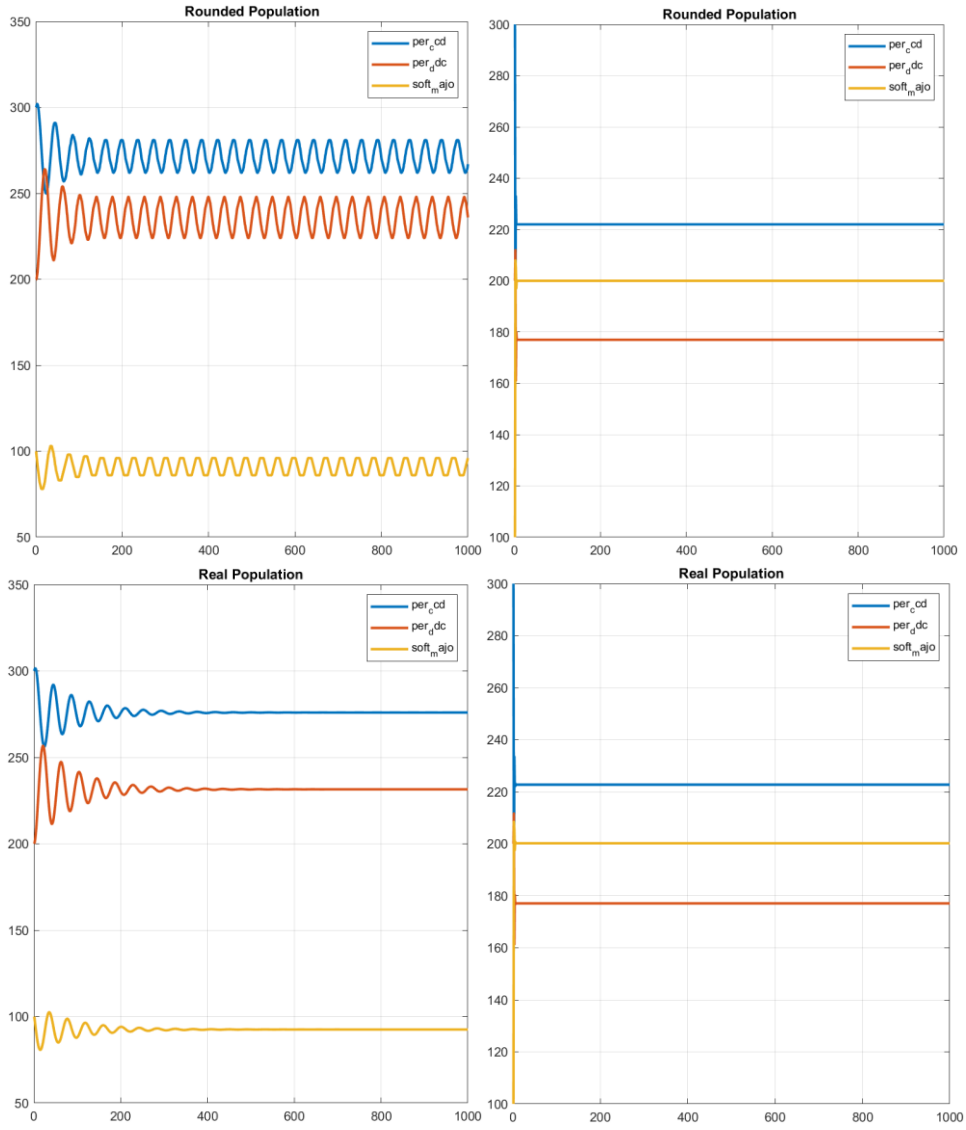


**Fig. 11.** Sensitivity to repartition computation method. All parameters are identical except that repartition on the left is done by rounding and uses real value on the right.



Script: Fig11_Fitness.m

Now we will see how the repartition computation method affects things, or in other words, how the results differ if we round down the populations to make them discrete, compared to if we use the real values.

We see that if we keep the populations discrete the dynamic is a periodic one, whereas if we keep the real values, it comes to an attenuated oscillated one.

Example 12

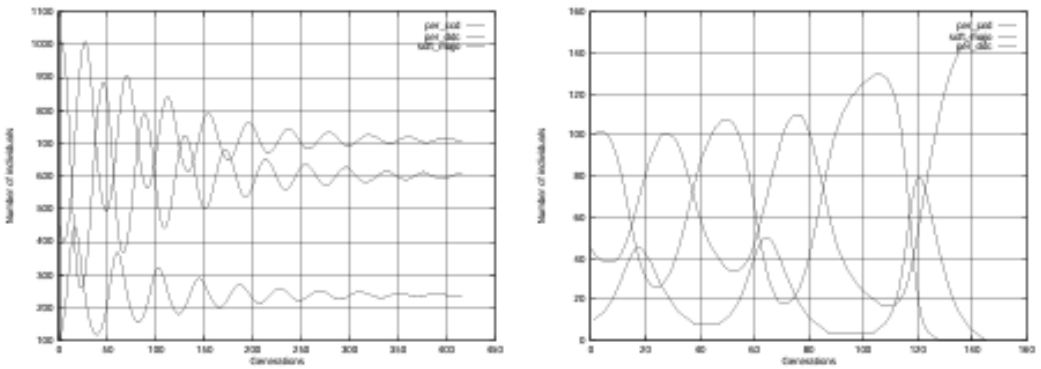
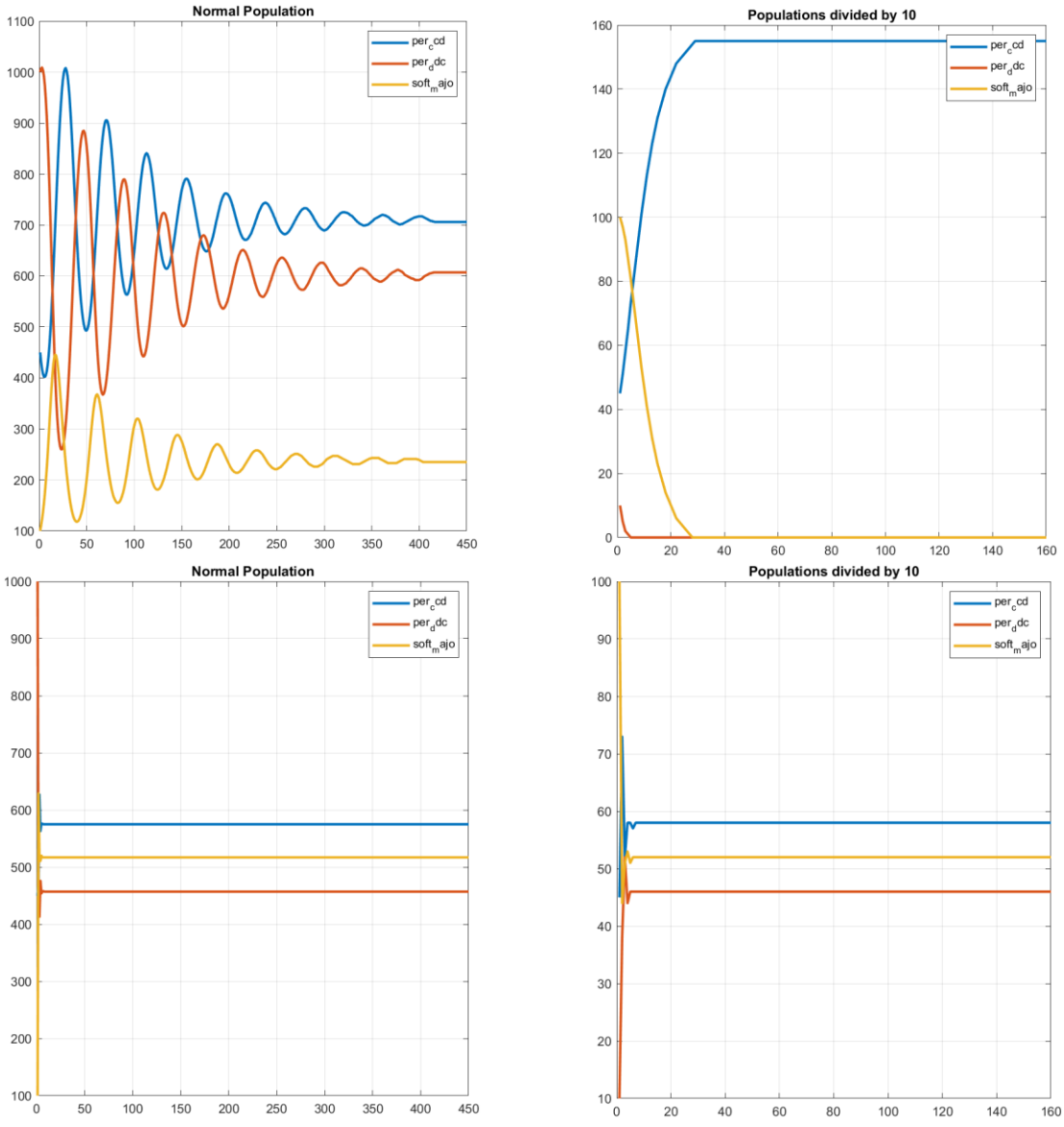


**Fig. 12.** Sensitivity to repartition computation method. All parameters are identical except that populations on the right are divided by 10.



Script: Fig12_Fitness.m

Finally, we once again see how the repartition computation influences the tournament, but in a different light. At first, we run the tournament with no changes to the computation, and the end result is an attenuated oscillation. But by dividing the population's by 10, the oscillation is now an increasing one.

## Discussion

So, in the end, we came to the following conclusions through the examples above:

1. The best strategies in most scenarios are:
    a. Nice
    b. Reactive
    c. Forgiving
    d. Simple

(Example 1)

2. The tournament can follow one of the following dynamics:
    a. Monotonous Convergence
    b. Attenuated Oscillation
    c. Periodic
    d. Increasing Oscillation
    e. Disordered Oscillation

(Examples 2-6)

3. The initial conditions can drastically change the way the tournament plays out, which includes:
    a. Initial Strategies' Population
    b. Number of Moves
    c. Payoff Matrix
    d. Repartition Computation Method

(Examples 7-12)

# (D) IMITATION DYNAMICS

## (D.1) The process

In Imitation Dynamics, things start off like they started in Fitness Dynamics; we assume strategies A, B and C that are represented by $W_n$ (A), $W_n$ (B) and $W_n$ (C) individuals in the $n_{th}$ generation and we calculate the score for each strategy.

Where Imitation Dynamics differs is that after generation n's tournament ends, K players from the non-best strategies (aka the strategies that didn't get the highest scores) change strategy, and they adopt one of the best strategies.

In the theoretical analysis, the goal is to calculate the transition probability matrix P, which shows us the probability of an individual from a non-best strategy to transition to one of the best ones for every generation n. This matrix shows us which strategies persist over time, which in turn helps us predict the equilibrium points of the evolutionary process.

Every row of the matrix P is a *Markovian chain*, a stochastic process $\{X(t)\}$:

$$X(t)=(\ x_1\,(t),\ x_2\,(t),\ldots,\ x_M\,(t))$$

where M the number of strategies and t current generation. This stochastic process has the following property:

$$\mathrm{P}(\ X(t+1)=\mathrm{j}|X(0)=k_0,X(1)=k_1,\ldots,X(t)=\mathrm{i})=\mathrm{P}(\ X_{t+1}=\mathrm{j}\mid X_t=\mathrm{i})\ \text{for t}=0,1,\ldots$$

which is called the *Markovian property.* In other words, in our application of Markov chains, the strategies that the non-best strategies will favor depends only on the results of the previous generation and not on the results of every generation prior to the current one. From the matrix, we create a diagram that shows us all the paths a player can take while switching strategies, as well as all possible *equilibrium points*, that are represented with red nodes.

**Algorithm 3: TourTheImi**

**Input:** Payoff bimatrix $B$, strategies $\{\sigma_1,\ldots,\sigma_M\}$, initial population $\mathbf{POP}_0$, parameters $(K,T,J)$

$N=\sum_{m=1}^{M}\mathbf{POP}_0(m)$
Compute payoff matrix $V(i,j)$ for all $i,j$ :
  $V(i,j)=$ payoff to $\sigma_i$ vs $\sigma_j$ in $B$ (over $T$ rounds)
Enumerate all possible population states:
  $S=\{s\in\mathbb{N}_0^M:\sum_{m=1}^{M}s(m)=N\},\ L=|S|$
Initialize transition matrix $P\in\mathbb{R}^{L\times L}$ as zeros
for each state $s\in S$ (index $l$)
  for each strategy $i$ with $s(i)>0$
    For each possible switch to $j\in\{1,\ldots,M\}$
      Construct $s'$ by moving one player from $i$ to $j$
      Compute payoff for adopting $j$ :
        $\Phi_j=\sum_{k=1}^{M}V(j,k)\cdot(s_k-\delta_{k,j})$
    end for
    Identify all $j$ maximizing $\Phi_j$
    For each such optimal $j$ :
      Update transition: find index $l_2$ of $s'$ in $S$
      $P(l,l_2)=P(l,l_2)+\frac{1}{|\mathcal{B}|}$
    end for
  end for
end for
Normalize each row of $P$ to sum to 1:
  If a row sums to 0, set diagonal to 1 (absorbing state)
return $P$

**Algorithm 4: TourSimImi**

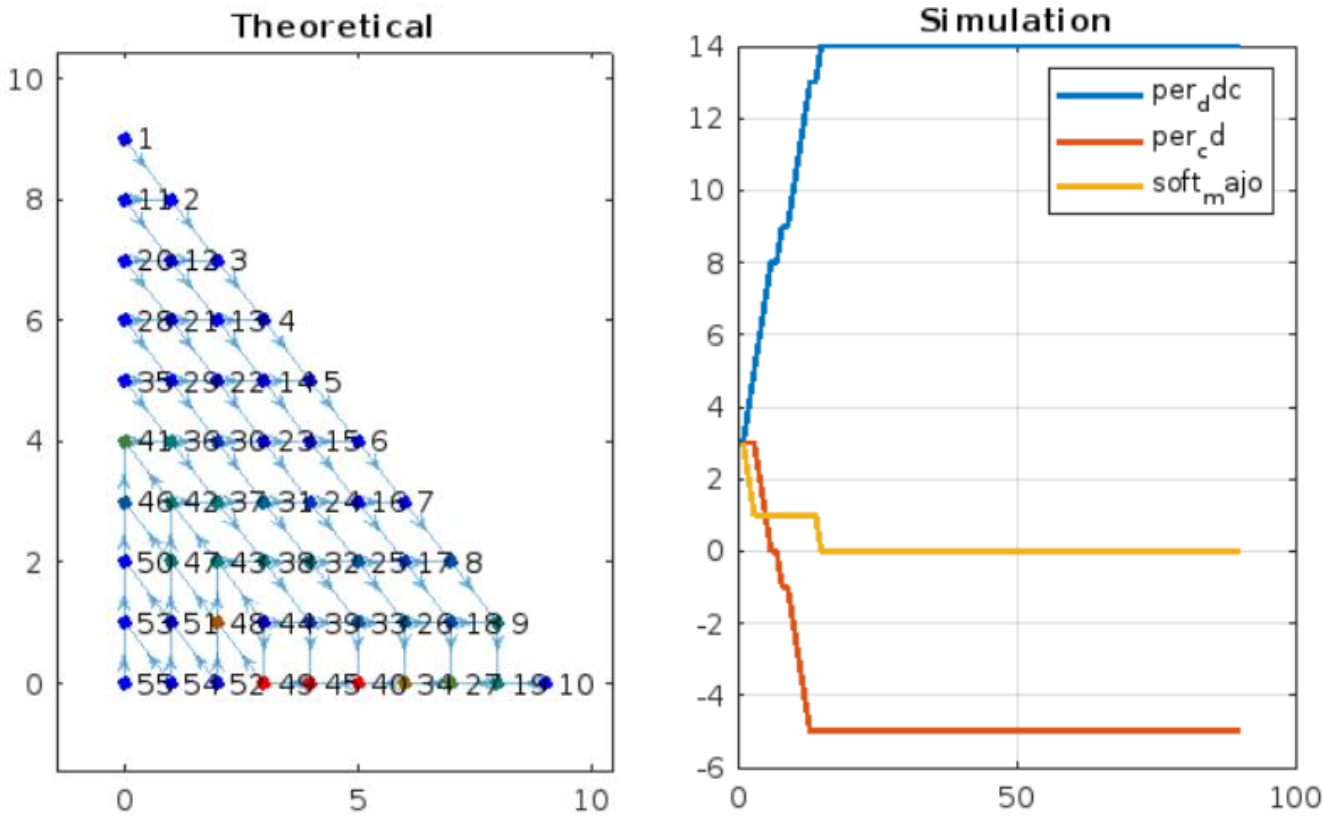
**Input:** Payoff bimatrix $B$, strategies $\{\sigma_1,\ldots,\sigma_S\}$, initial population $\mathbf{POP}_0$, parameters $(K,T,J)$

Set $S=|\{\sigma_1,\ldots,\sigma_S\}|$.
Initialize population $\mathbf{POP}(1,:)=\mathbf{POP}_0$.
for each generation $n\in\{1,\ldots,J\}$
  Let $\mathbf{counts}=\mathbf{POP}(n,:)$.
  Initialize payoffs: $\mathbf{payoff}=\mathbf{0}_{1\times S}$.
  for each pair $i,j\in\{1,\ldots,S\}$
    if $\mathbf{counts}(i)>0$ and $\mathbf{counts}(j)>0$ then
      $[p_1,p_2]=\mathrm{MatchPayoff}(\sigma_i,\sigma_j,B,T)$
      $n_{\text{Games}}=\begin{cases}\mathbf{counts}(i)\cdot(\mathbf{counts}(i)-1)/2, & i=j\\ \mathbf{counts}(i)\cdot\mathbf{counts}(j), & i\neq j\end{cases}$
      $\mathbf{payoff}(i)=\mathbf{payoff}(i)+p_1\cdot n_{\text{Games}}$
      $\mathbf{payoff}(j)=\mathbf{payoff}(j)+p_2\cdot n_{\text{Games}}$
    end if
  end for
  Compute fitness: for $s\in\{1,\ldots,S\}$,
    if $\mathbf{counts}(s)>0$ : fitness$(s)=\mathbf{payoff}(s)/\mathbf{counts}(s)$
  Identify best strategies: $\mathcal{B}=\{s:\mathbf{payoff}(s)=\max_j\mathbf{payoff}(j)\}$
  Identify candidates to switch: all players from $s\notin\mathcal{B}$
  $K_{\text{eff}}=\min(K,\text{total number of candidates})$
  Randomly select $K_{\text{eff}}$ candidates from non-best strategies to switch
  For each candidate, remove 1 player from their current strategy
  For each switcher, randomly assign to a strategy in $\mathcal{B}$
  Add one player for each assignment to the corresponding best strategy
  Update next generation: $\mathbf{POP}(n+1,:)=$ new counts
end for
For each generation, record the best strategy:
  $\mathbf{BST}(n)=\arg\max_s$ fitness$(s)$
return $\mathbf{POP},\mathbf{BST}$

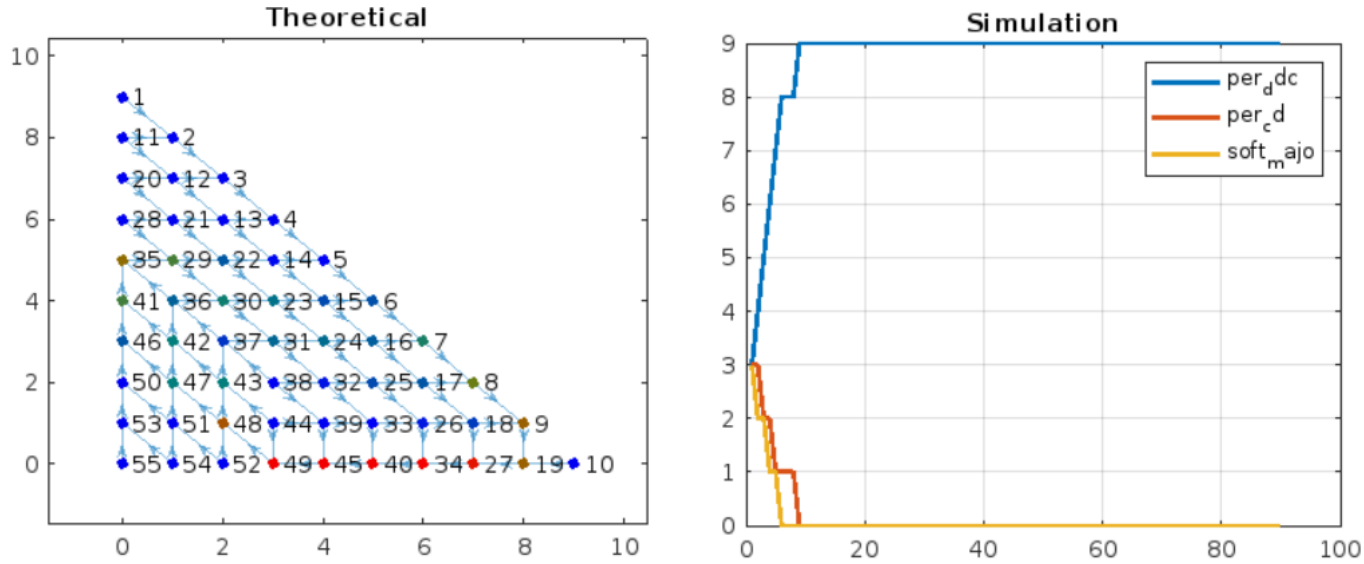
## (D.2) Experiments

Example 1

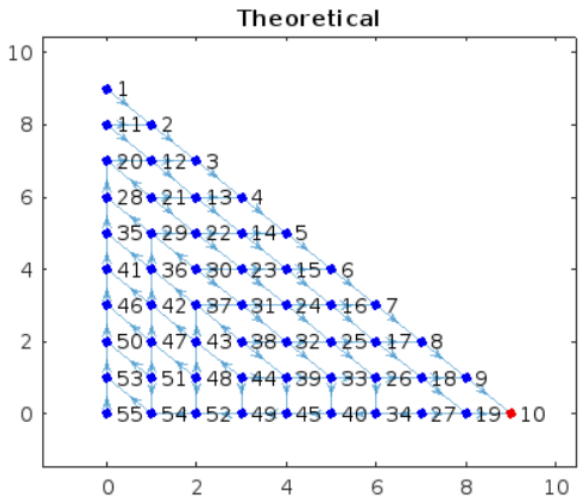
**a=3**

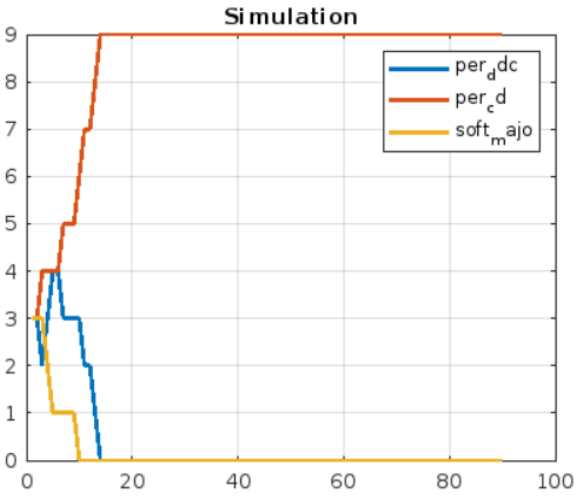



**a=3.4**



**a=3.8**



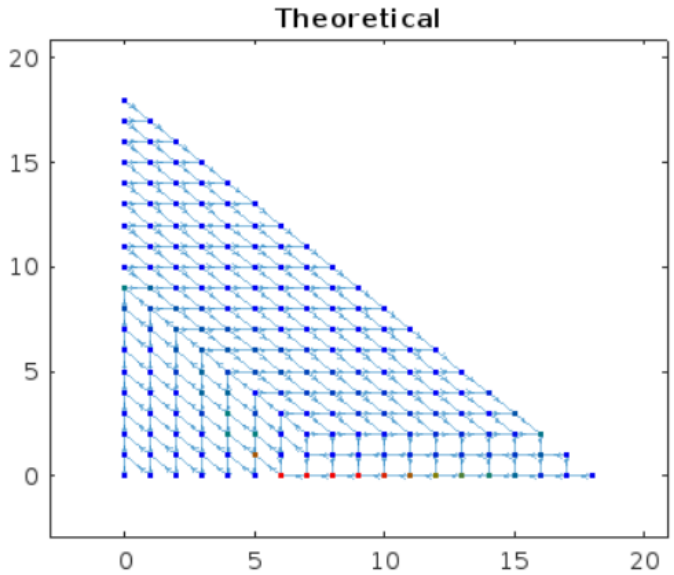



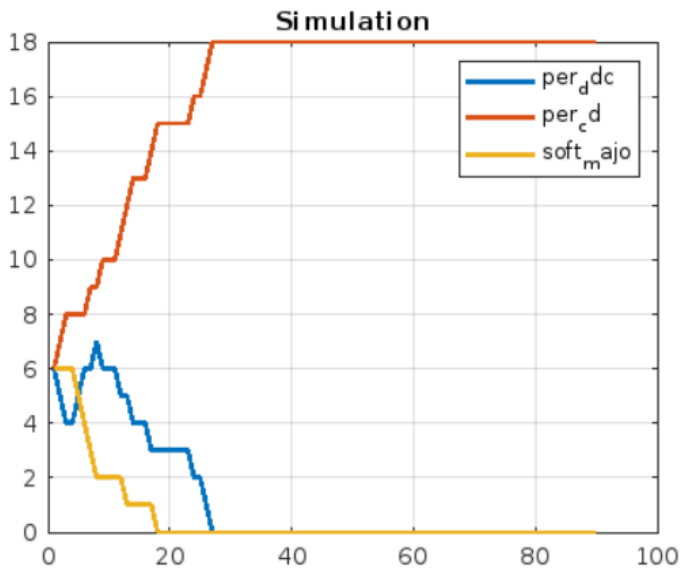

In the first example, we have 3 individuals playing per_ddc, 3 playing per_cd and 3 playing soft_majo. They play for 90 rounds and K=1, which means that between generations, one individual changes strategy.

We see that in the end, for a=3 and a=3.4, per_ddc is the winner, while for a=3.8, the winner changes, becoming per_cd. In all scenarios, all the losers cease to exist. Another thing to take note of is that the diagram for the transition probability matrix drastically changes (i.e. for a=3.8, node 10 is an equilibrium point, while not being one for a=3 and a=3.4)

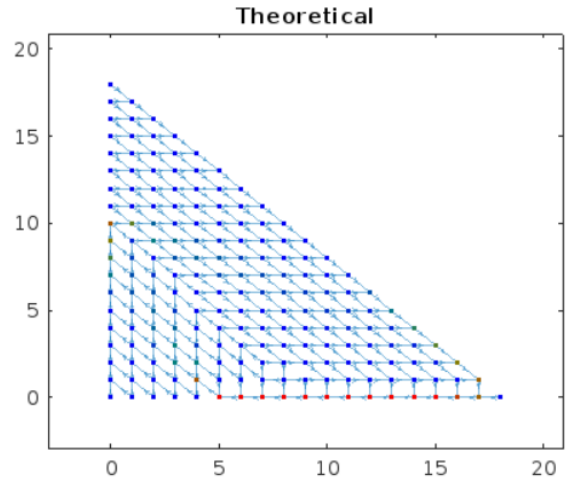
Example 2

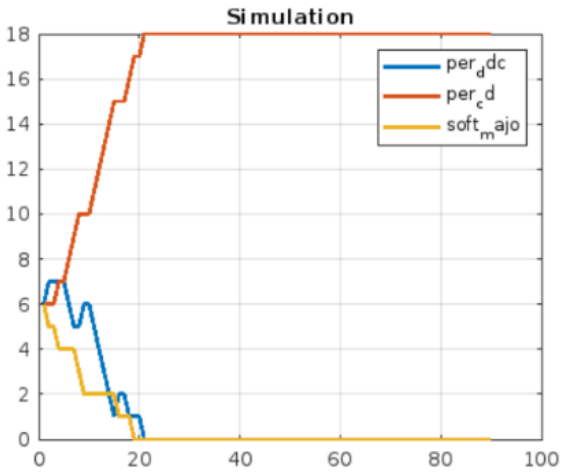
**a=3**



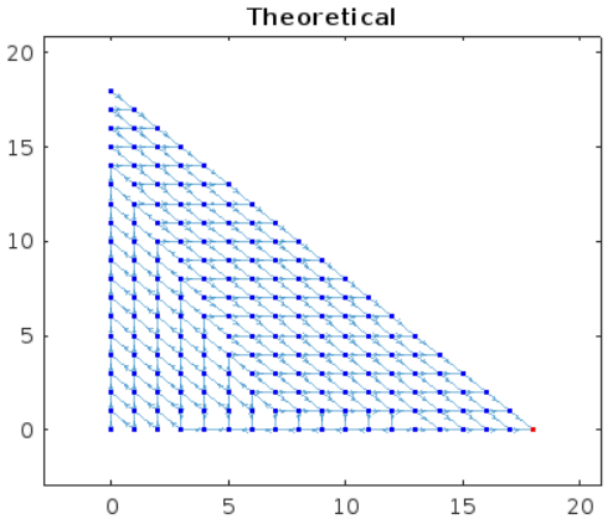



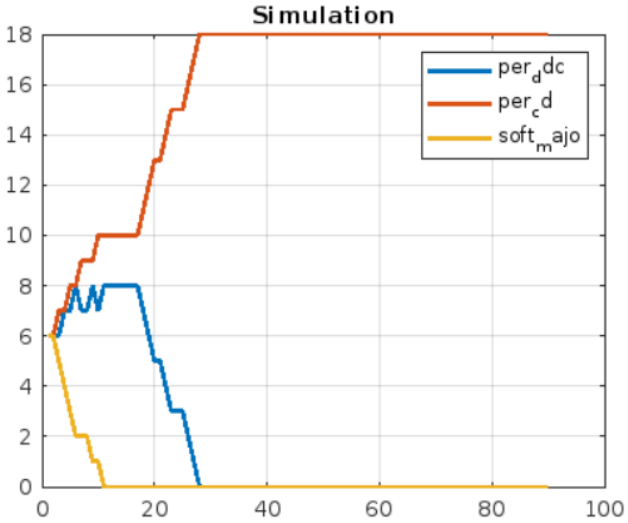
**a=3.4**




**a=3.8**



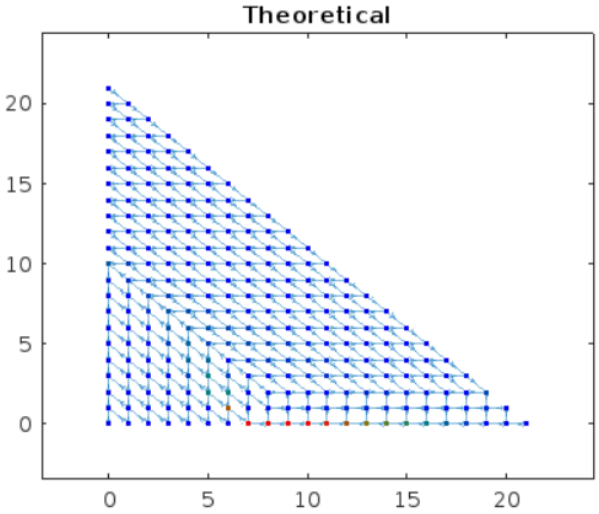



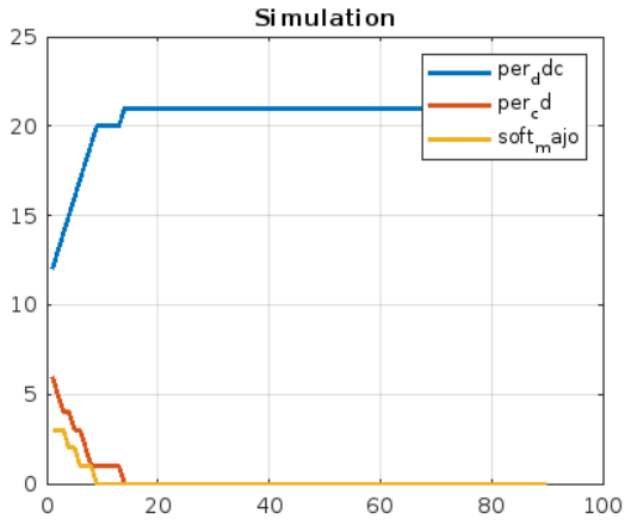

For Example 2, all initial parameters are the same as they were in Example 1, except there are 6 individuals representing each strategy instead of 3.

This time, we see that for all tested values of a, per_cd is the winning strategy, and the other two no linger exist by the end of the game. We also see that the diagrams for the transition matrix are similar to the ones in the previous examples, but bigger because of the population increase (equilibrium points in similar spots, most notable for a=3.8 where it is the most bottom right node).

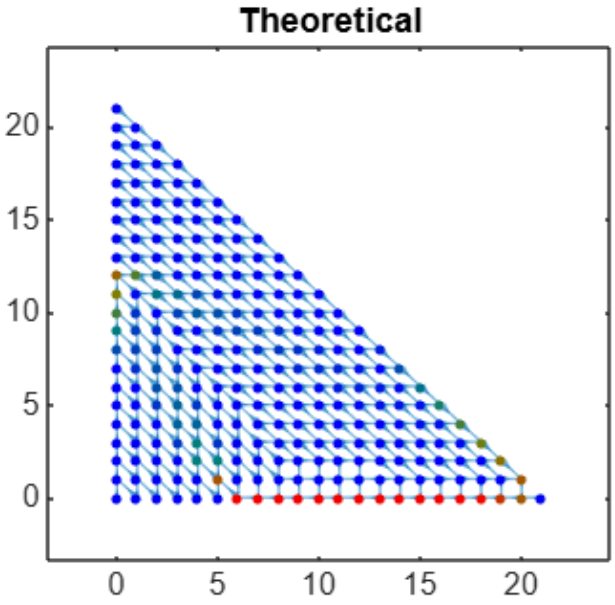
Example 3

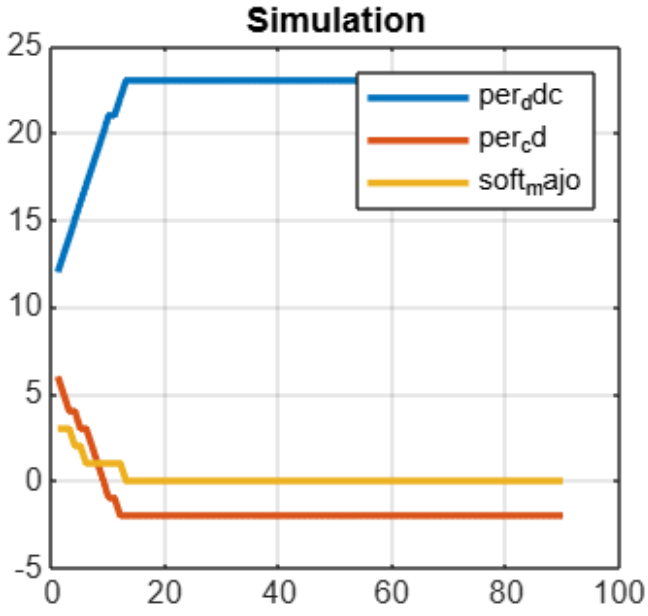
**a=3**


Theoretical
0
5
10
15
20

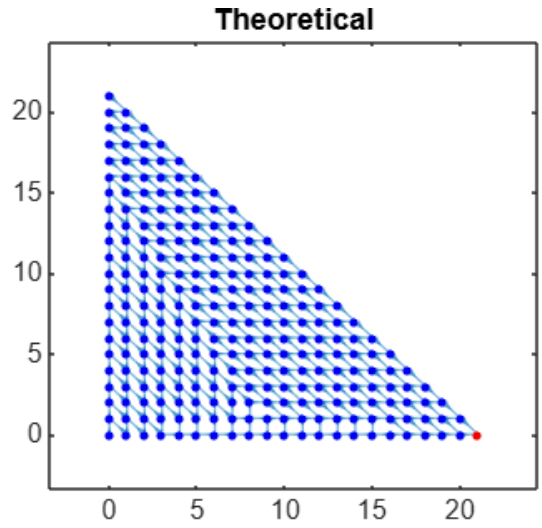

Simulation
per_ddc
per_cd
soft_majo
0
5
10
15
20
25
0
20
40
60
80
100

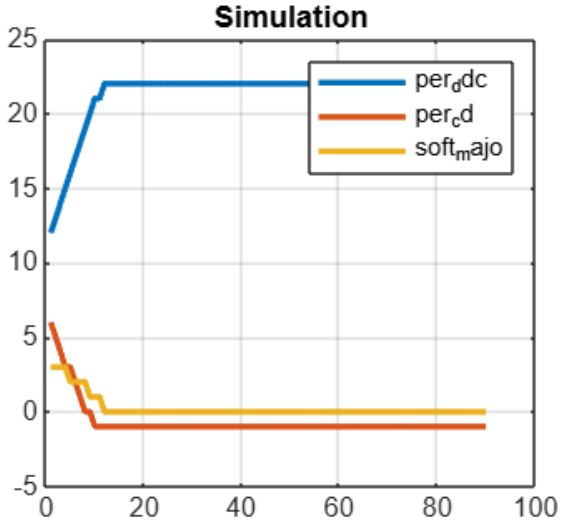
**a=3.4**


Theoretical
0
5
10
15
20


Simulation
per_ddc
per_cd
soft_majo
-5
0
5
10
15
20
25
0
20
40
60
80
100

a=3.8


Theoretical
0
5
10
15
20

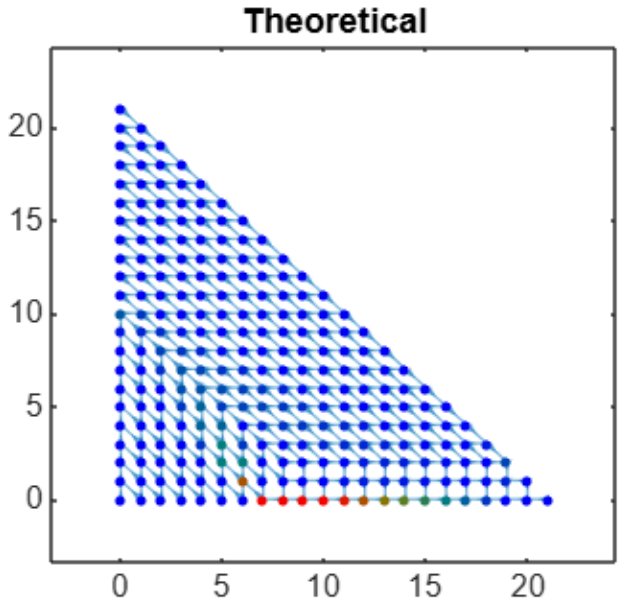

Simulation
per_ddc
per_cd
soft_majo
-5
0
5
10
15
20
25
0
20
40
60
80
100

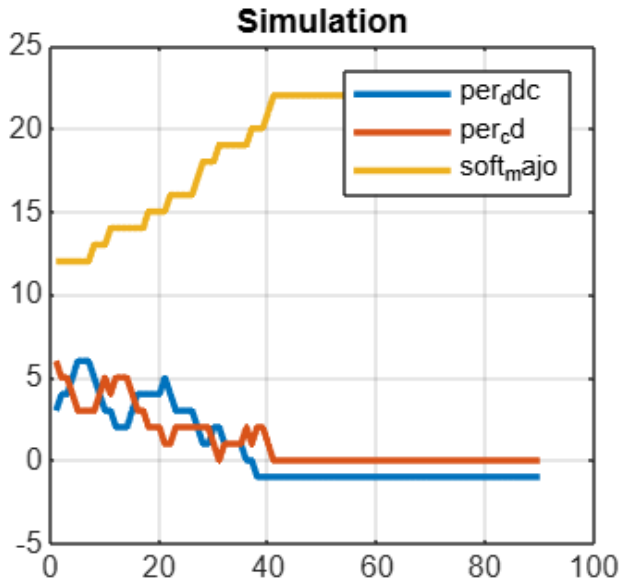
Like the pervious examples, we have the same initial parameters, and we only change the initial populations. This time, per_ddc is represented by 12 individuals, per_cd by 6 and soft_majo by 3.

Now, the winner is always per_dc, and we see the same similarity in the diagram for the transition matrix.

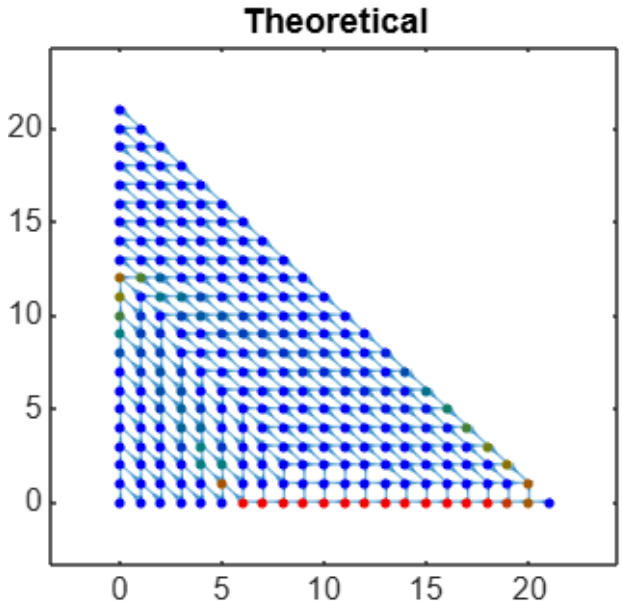
Example 4

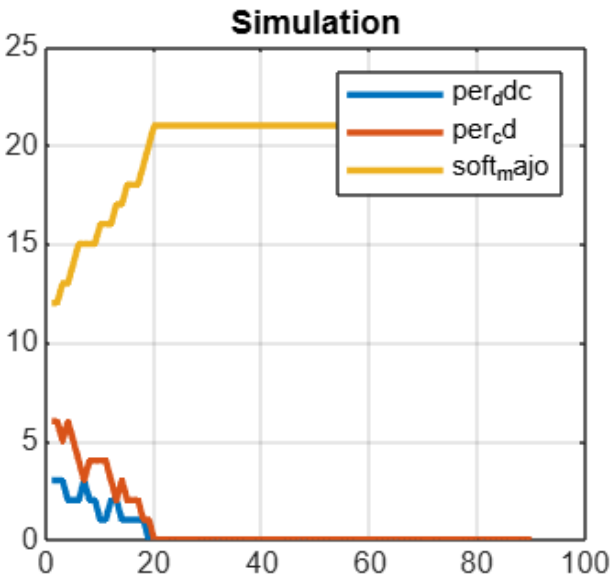
**a=3**



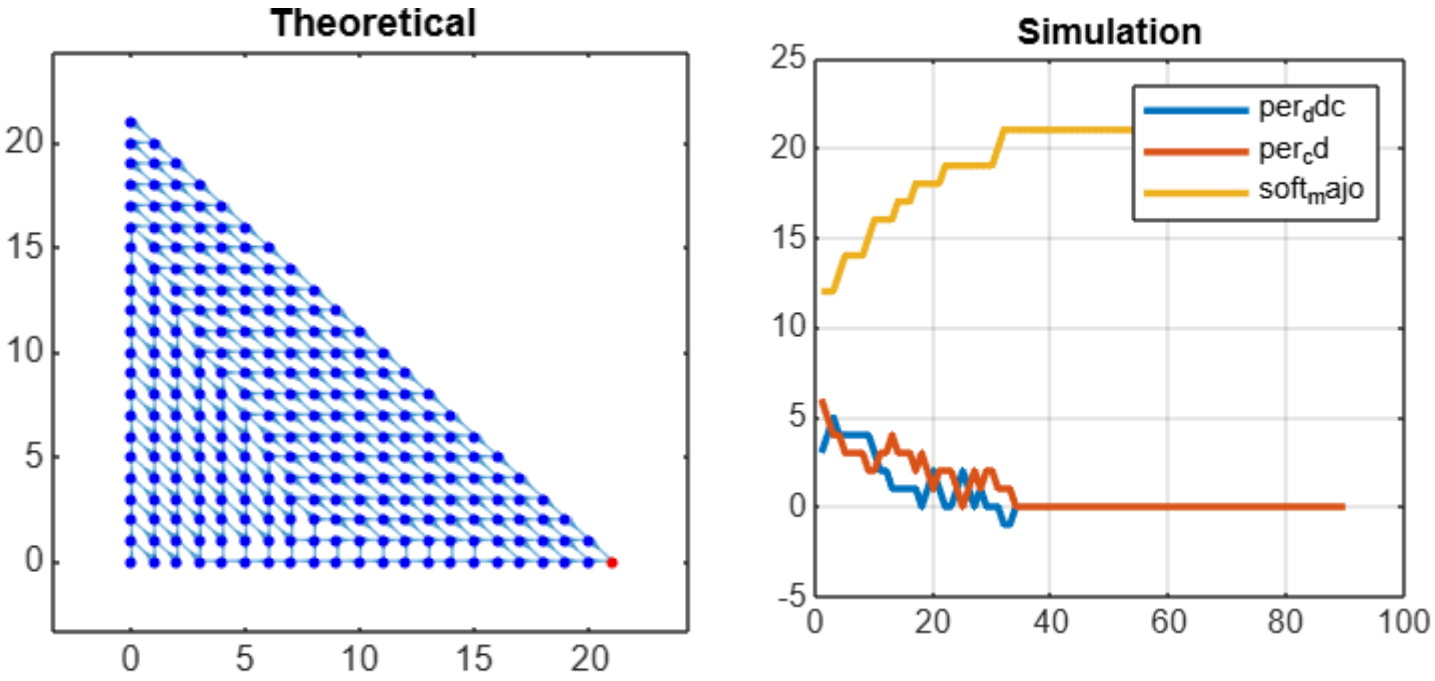




**a=3.4**



**a=3.8**

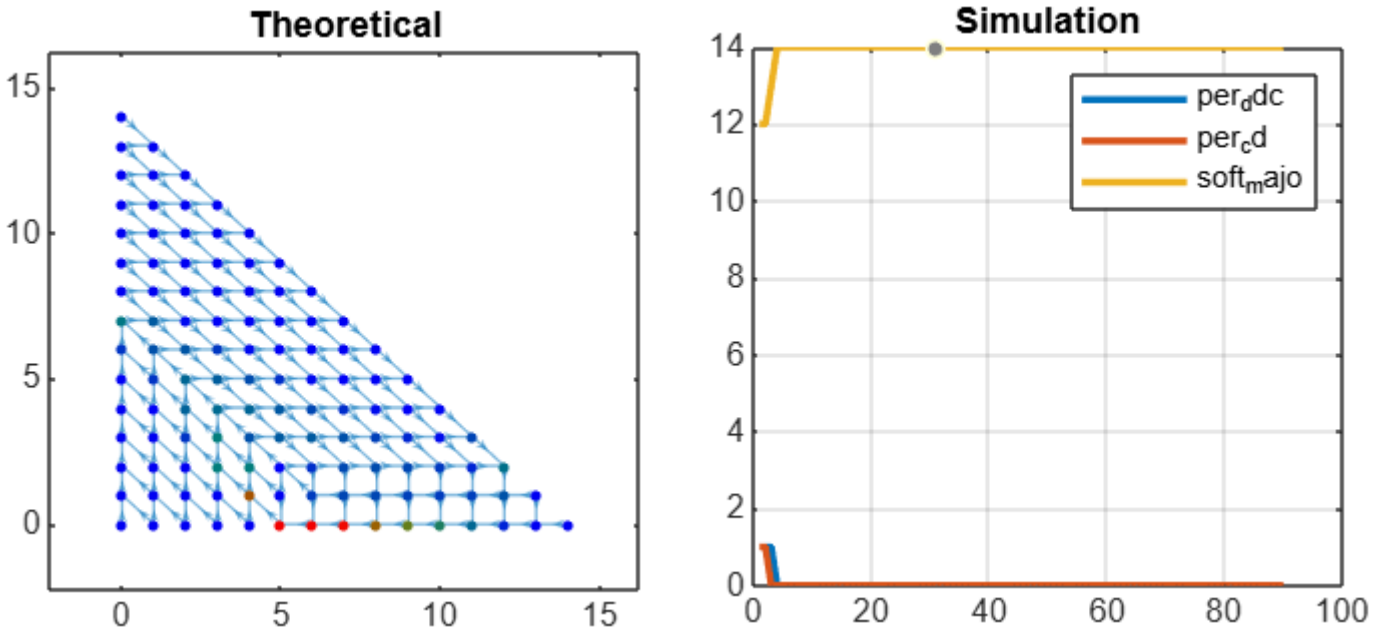



Same initial parameters as before, but the population sizes are reversed compared to Example 3, or in other words, per_ddc is represented by 3 individuals, per_cd by 6 and soft_majo by 12.

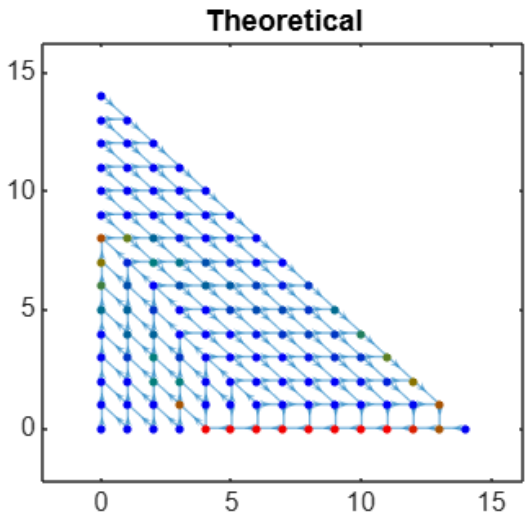
Now, for the first time so far, soft_maj is the eventual winner. We also see the same form for the diagram of the transition matrix.

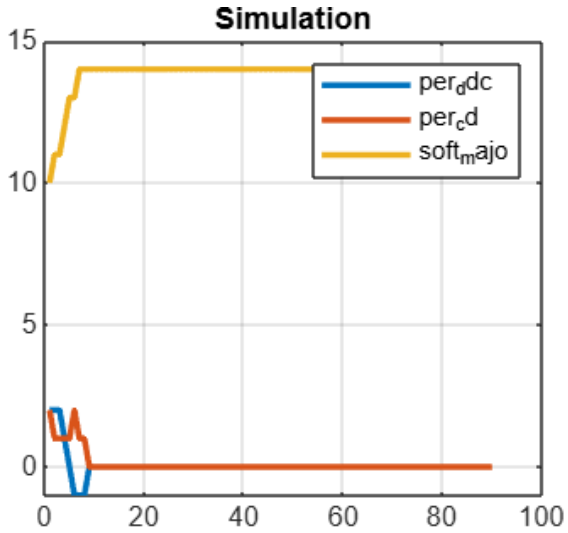
Example 5

**a=3**

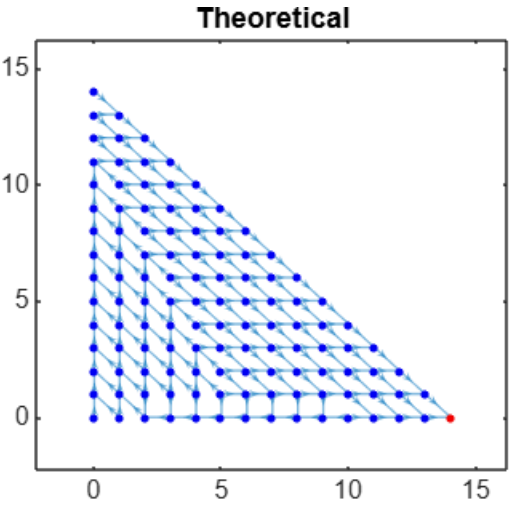


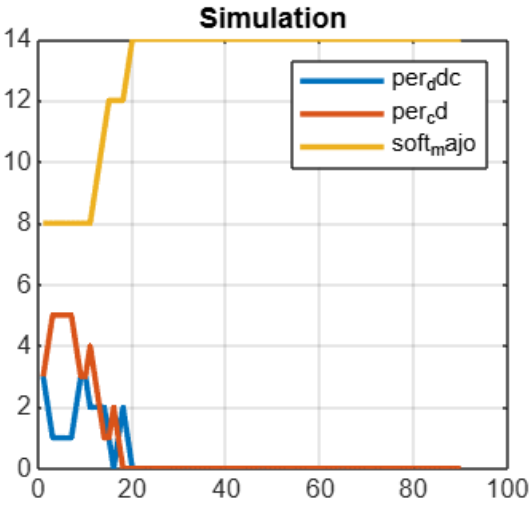
**a=3.4**




**a=3.8**



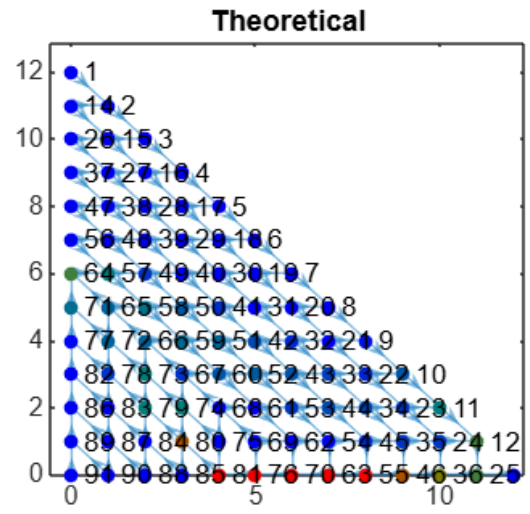



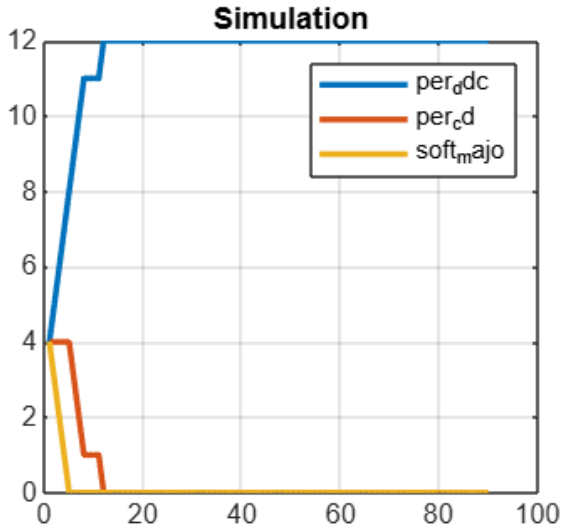

Once again, only the initial population of the strategies changes. Now, per_ddc and per_cd only have one person representing them, while soft_majo has 12.

Like the previous example, the winner is soft_majo (unsurprisingly) and we can see it dominated the other two strategies, compared to Example 4. Same comment about diagram of the transition matrix.

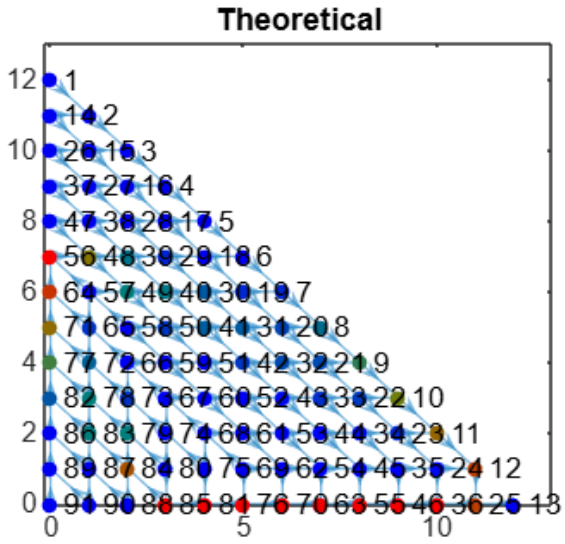
Example 6

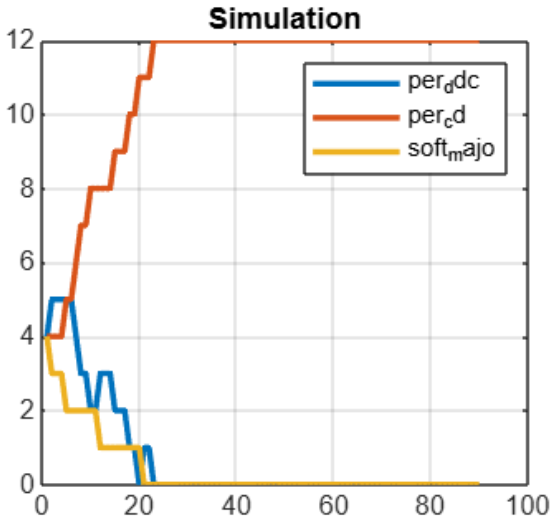
**a=3**


Theoretical

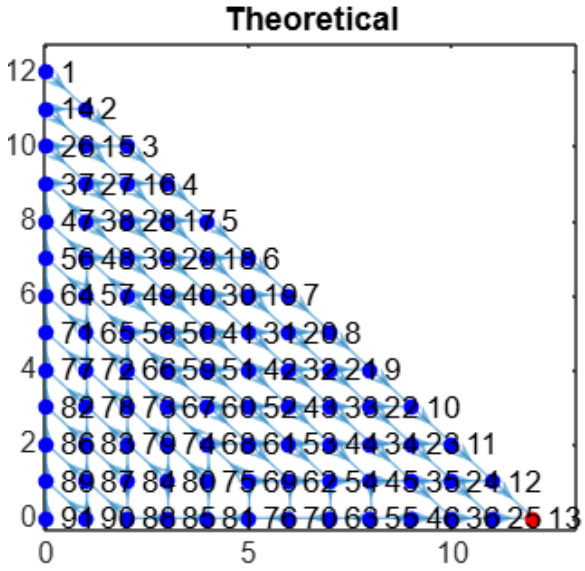

Simulation
perdc
percd
softmajo

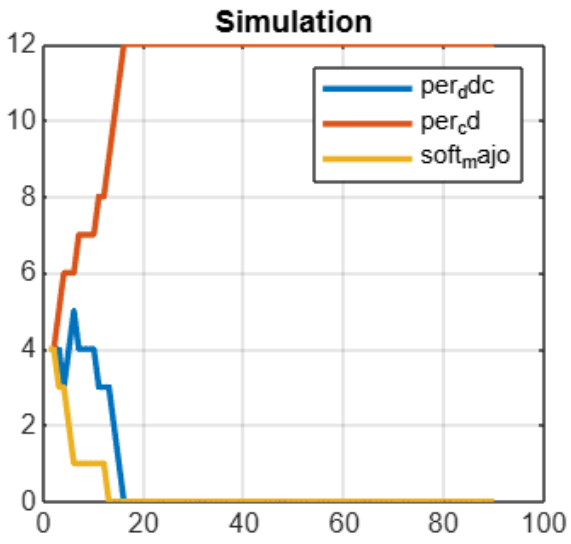
**a=3.4**


Theoretical


Simulation
perdc
percd
softmajo

**a=3.8**


Theoretical

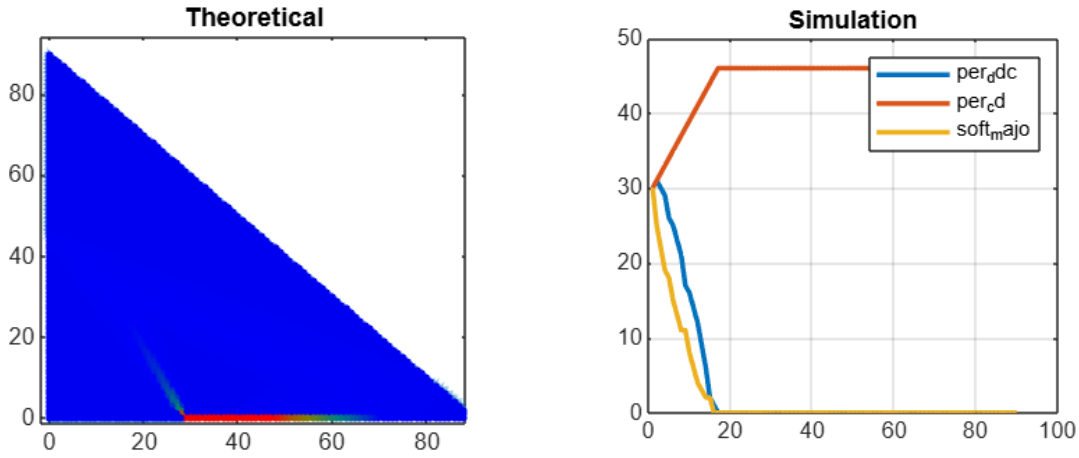

Simulation
perdc
percd
softmajo

Now, for the first time since Example 1, all three strategies have the same population, that population now being 4.

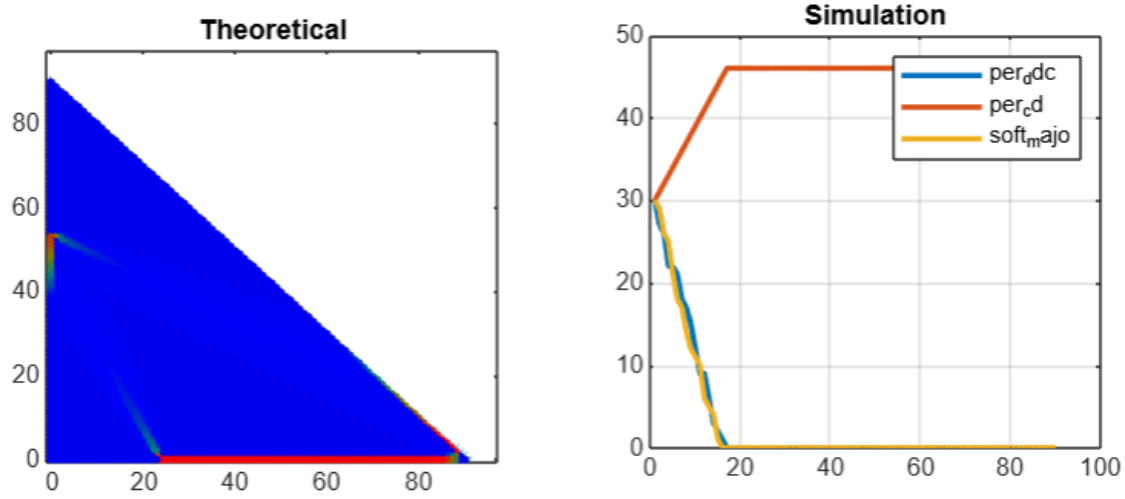
For a=3, per_ddc is the winner but for the other two scenarios, per_cd is. Worth noting that the graph shapes are similar to Example 1's, although now for a=3.4, the winner is different.

Example 7

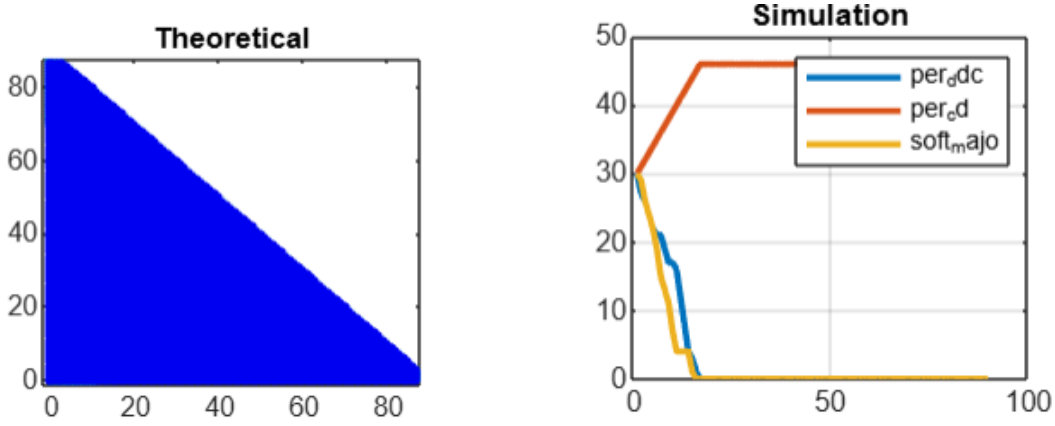
**a=3**




**a=3.4**




**a=3.8**

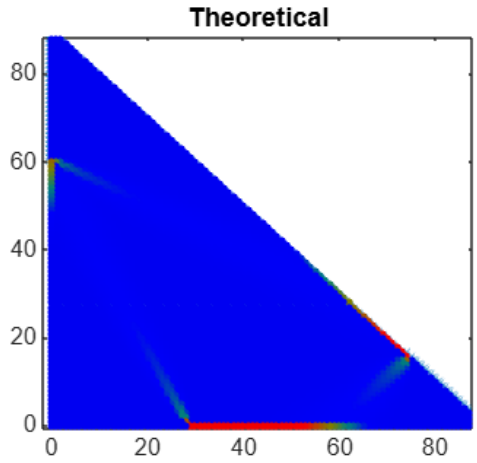


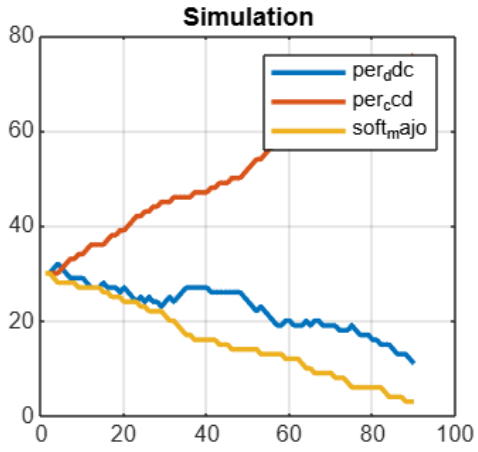
Once again, the three strategies have the same number of individuals representing them, 30.

Unlike the other scenarios where the populations are equal, the winning strategy never fluctuates between the three values of a we test. Also, we can see how concentrated the nodes are in the diagram of the transition matrix.

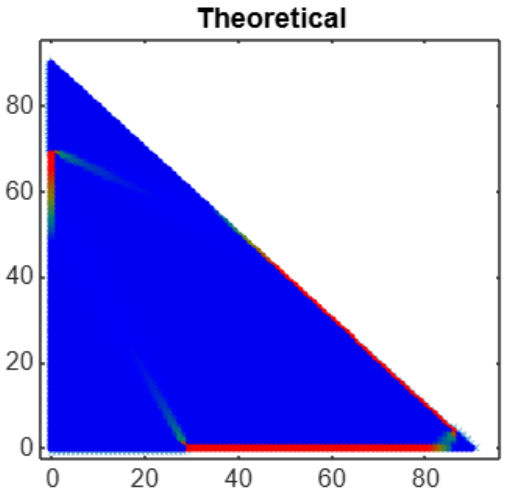
Example 8

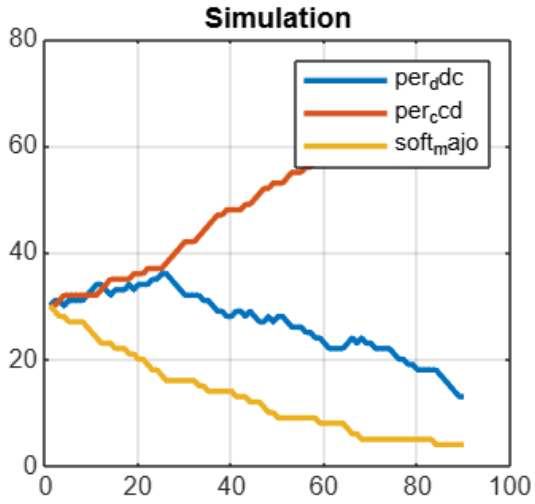
**a=3**



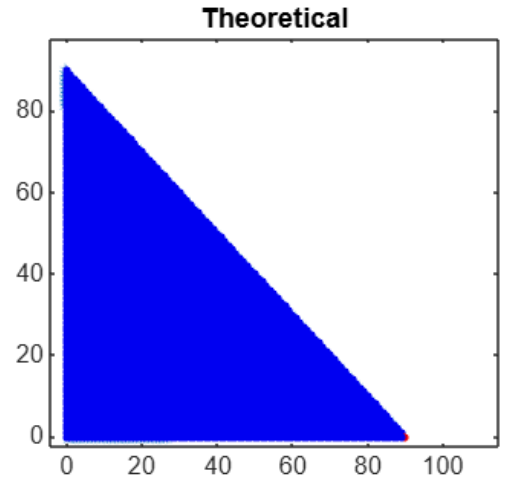



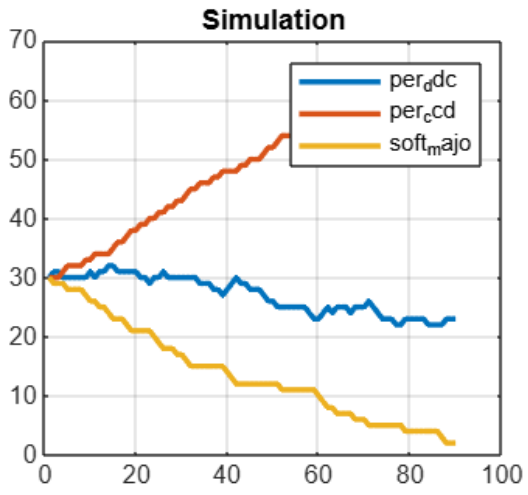

**a=3.4**




**a=3.8**



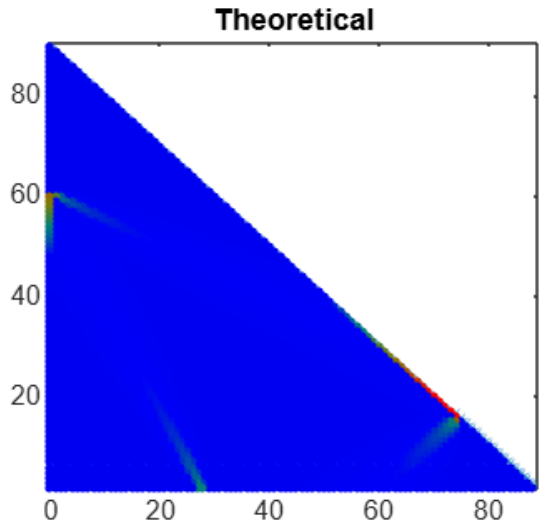



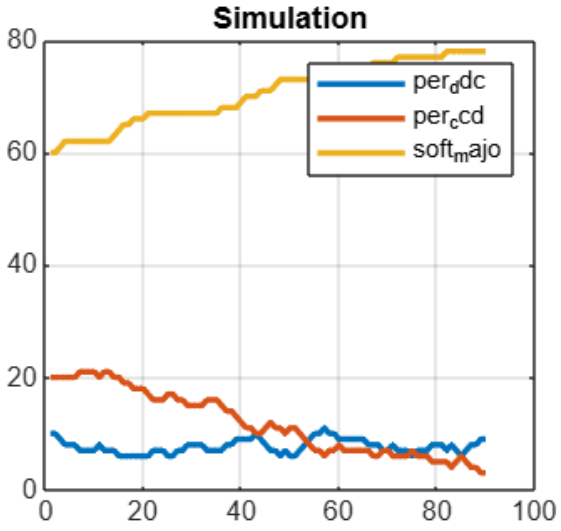
In this example we change things up a bit. All the populations are the same as in Example 7, but now instead of per_cd we have per_ccd.

Once again, per_ddc and soft_majo are the ultimate losers of the game.

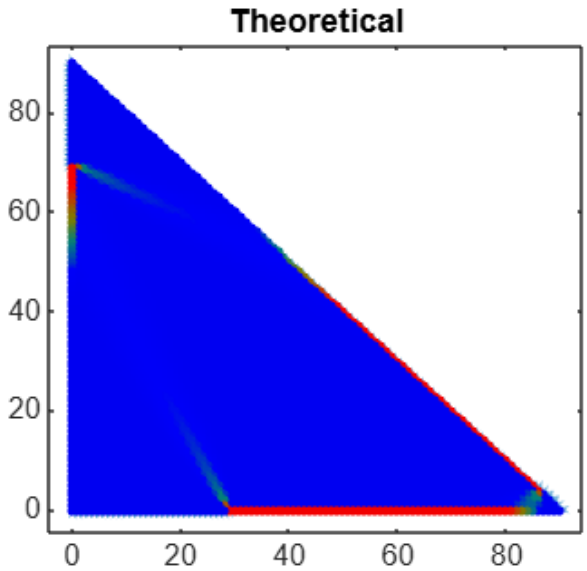
Example 9

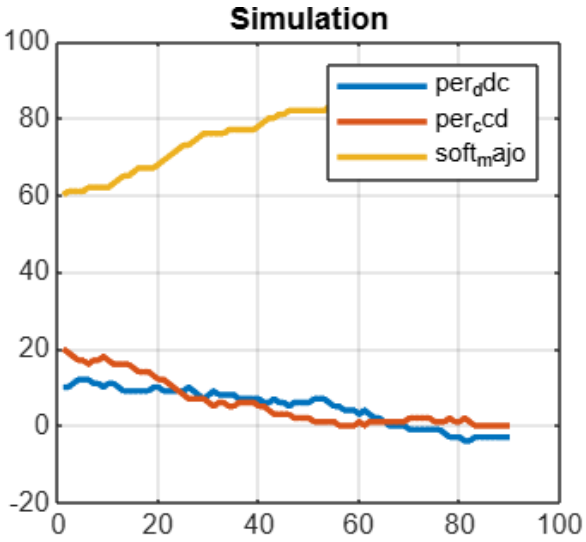
**a=3**



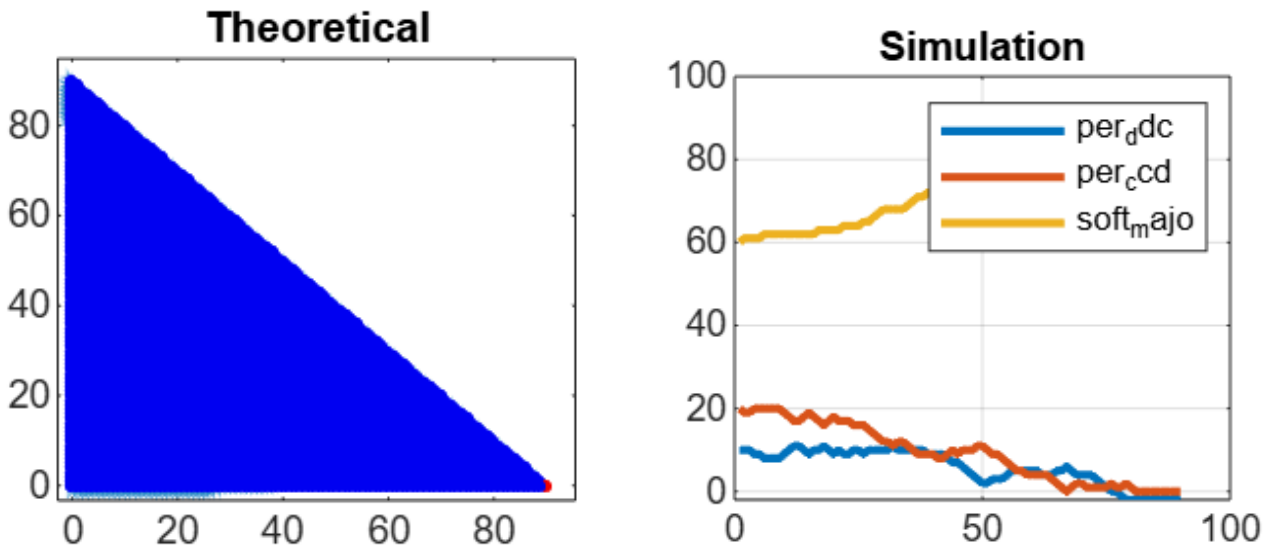




**a=3.4**



**a=3.8**

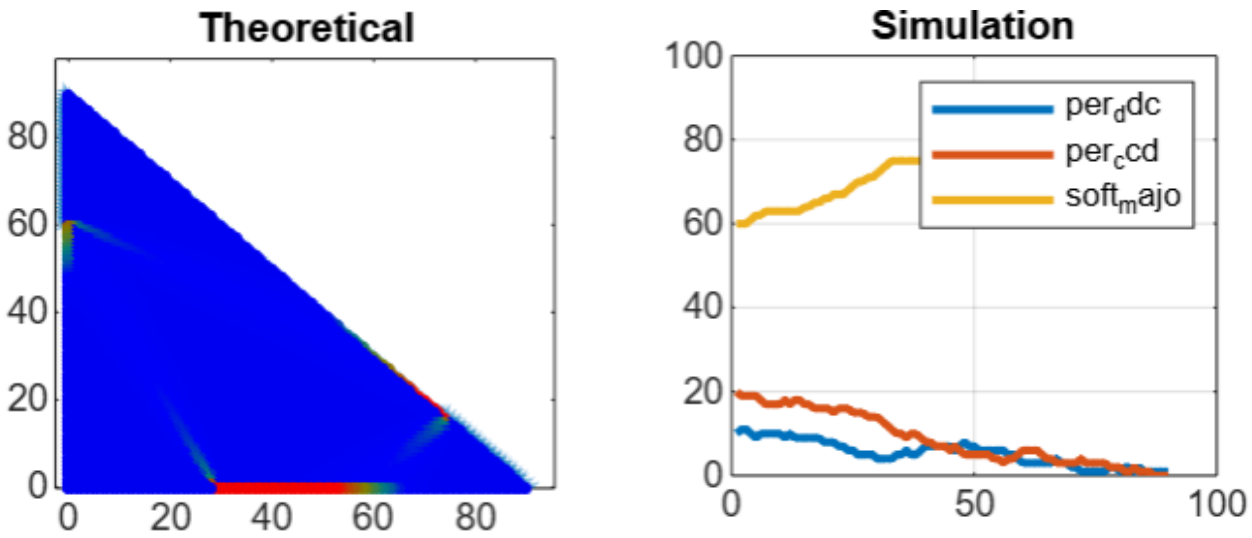



Now, continuing with the strategies of the previous examples, we change their populations as follows: per_ddc has 10, per_ccd has 20 and soft_majo has 60.

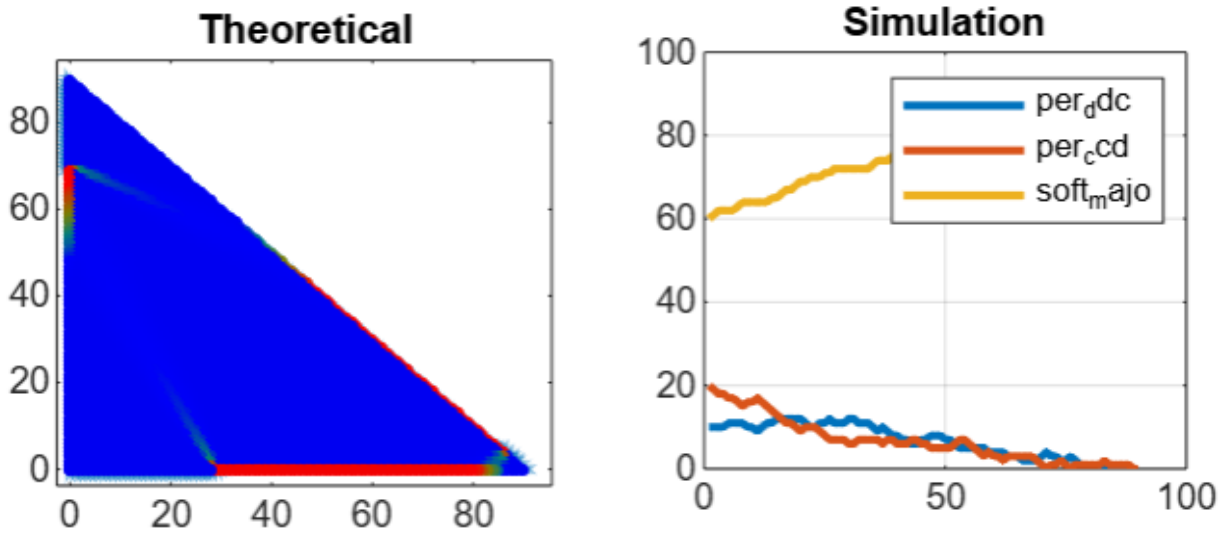
The winner this time is soft_majo, but it's interesting to see how the lines of the other two strategies intersect in different ways between the three values of a.

Example 10

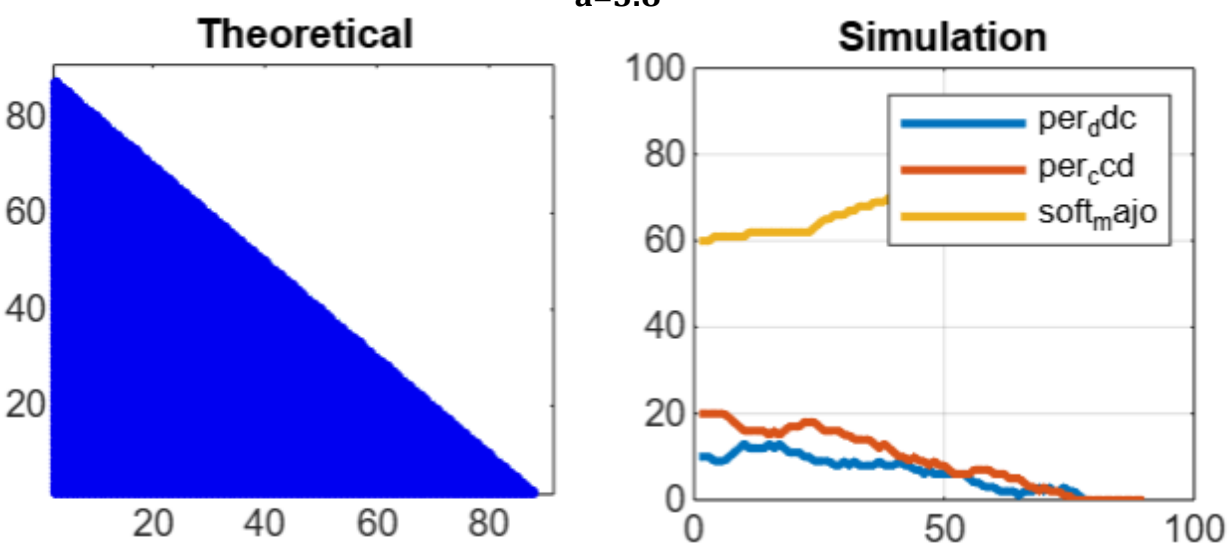
**a=3**



**a=3.4**




**a=3.8**

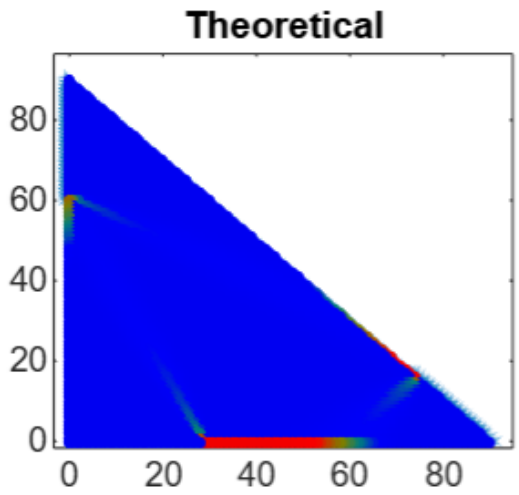


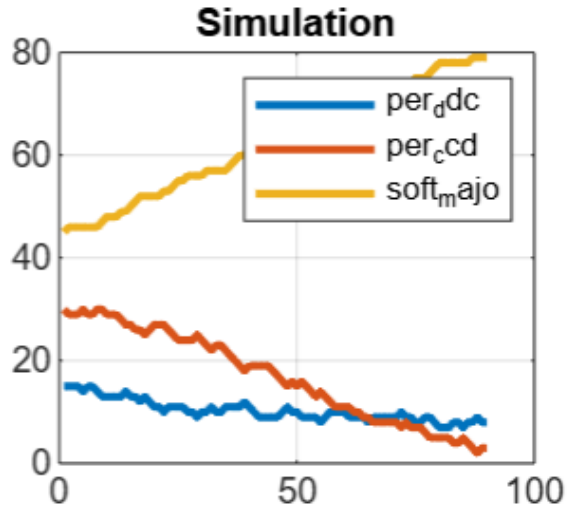

And now, we multiply all the strategies' populations by 10.

The winner remains soft_majo, but again, the lines of the other two intersect in different ways. Also, for the same value of a, the lines of the strategies differ between this example and the previous one.

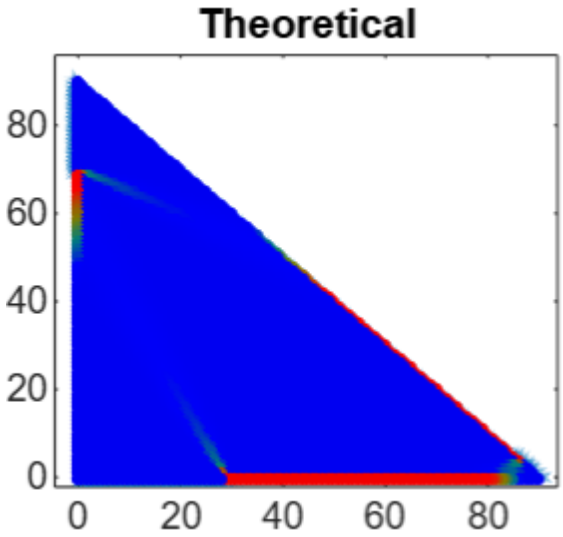
Example 11

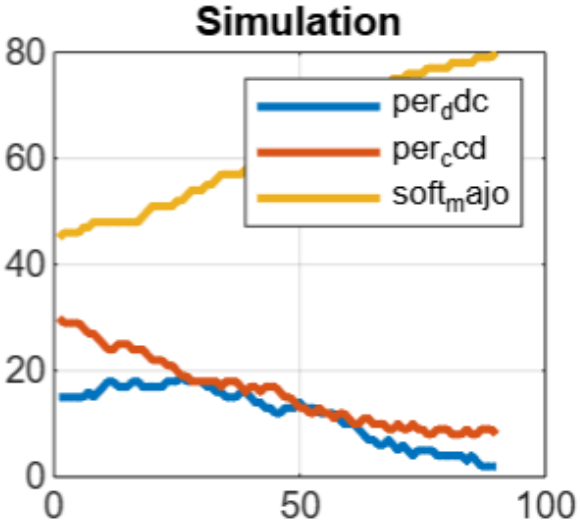
**a=3**



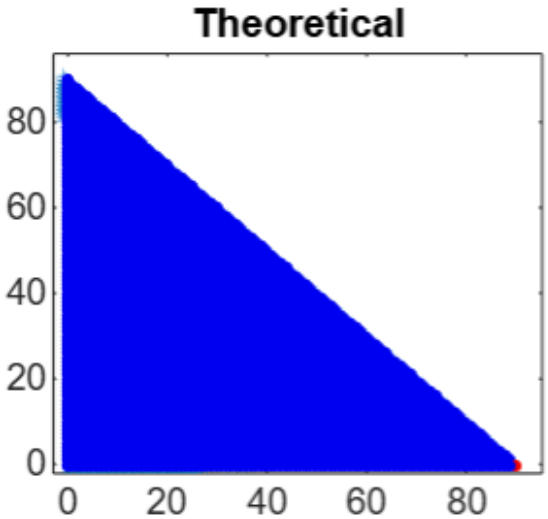



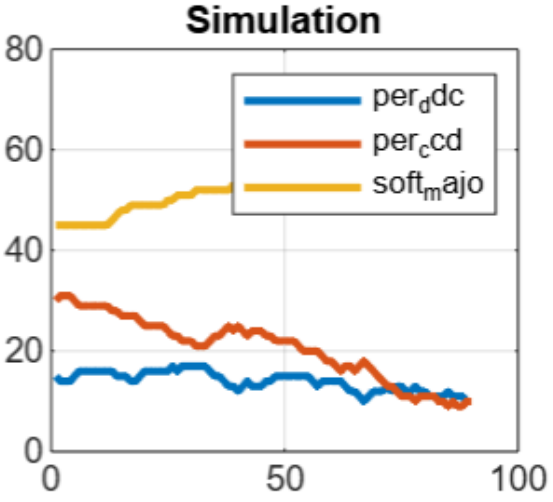

**a=3.4**




**a=3.8**



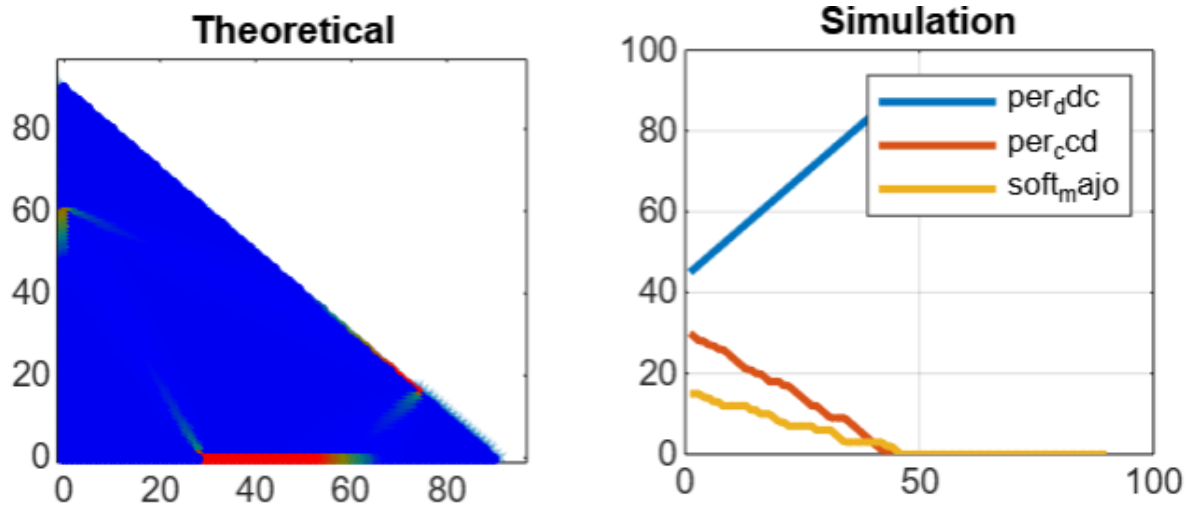



Now, we lower each strategies' populations: per_ddc has 15, per_ccd has 30 and soft_majo 45.

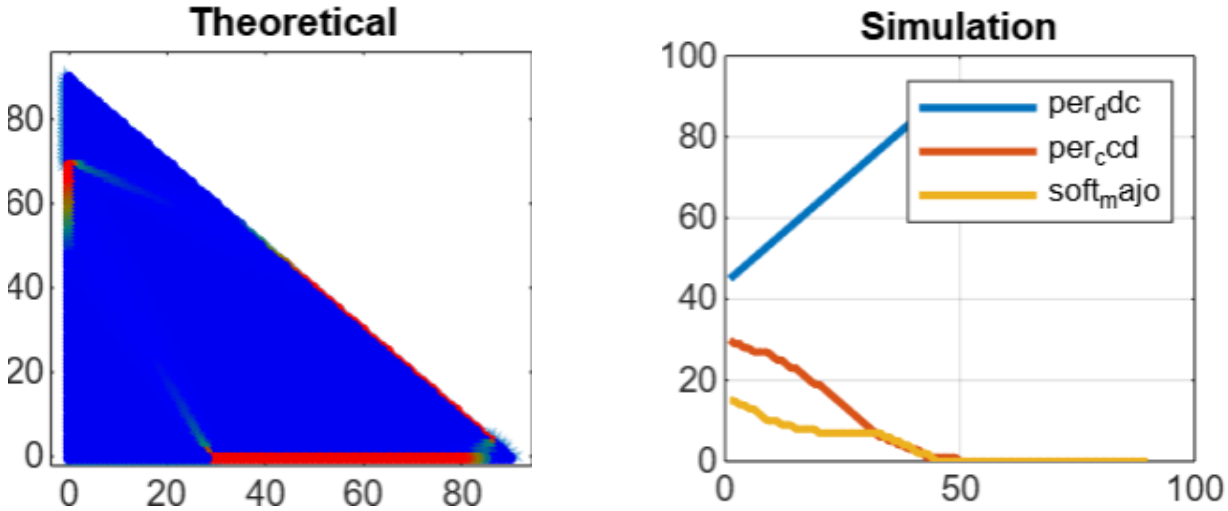
The winner remains to be soft_majo.

Example 12

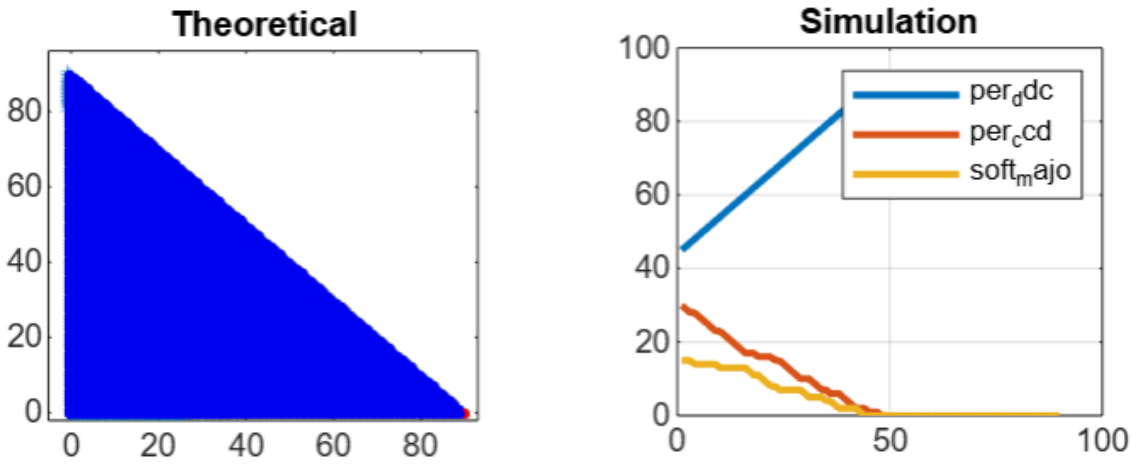
**a=3**




**a=3.4**




**a=3.8**



Now, we reverse the populations such as per_ddc has 45, per_ccd has 30 and soft majo has 15.

The winner is now per_ddc.

## (D.3) Discussion

From the above, we saw that:

1. The values of a influence the diagram of the transition matrix, but it seems that it tends to maintain the same general form, possibly because we were using the same strategies.

2. Even if all the strategies start off with the same population, depending on a's value ,the winner changes. Actually, it even varies depending on what the population is equal to, even if all of them have the same population.

3. Even if in two scenarios the populations are relative to each other, the eventual winner might change.

4. For different values of a, the lines of the graph vary.

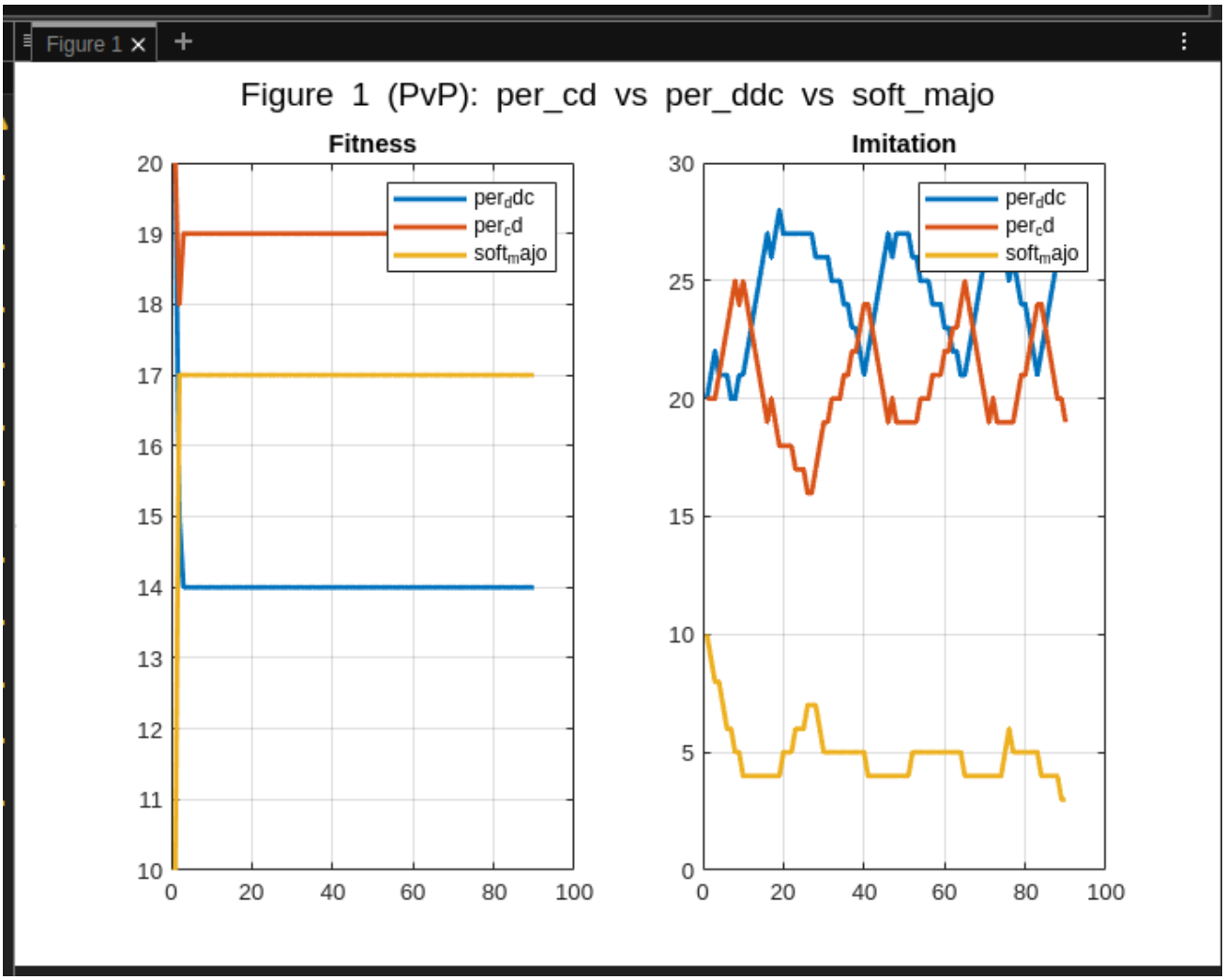
5. For several of our examples, our winner tended to be the one whose initial population is the highest.

# (E) DISCUSSION

## (E.1) Comparison of Fitness Vs. Imitation Dynamics (Player vs Player)

The following figures where taken by the script FitnessSimulationVsImitationSimulationScript.m

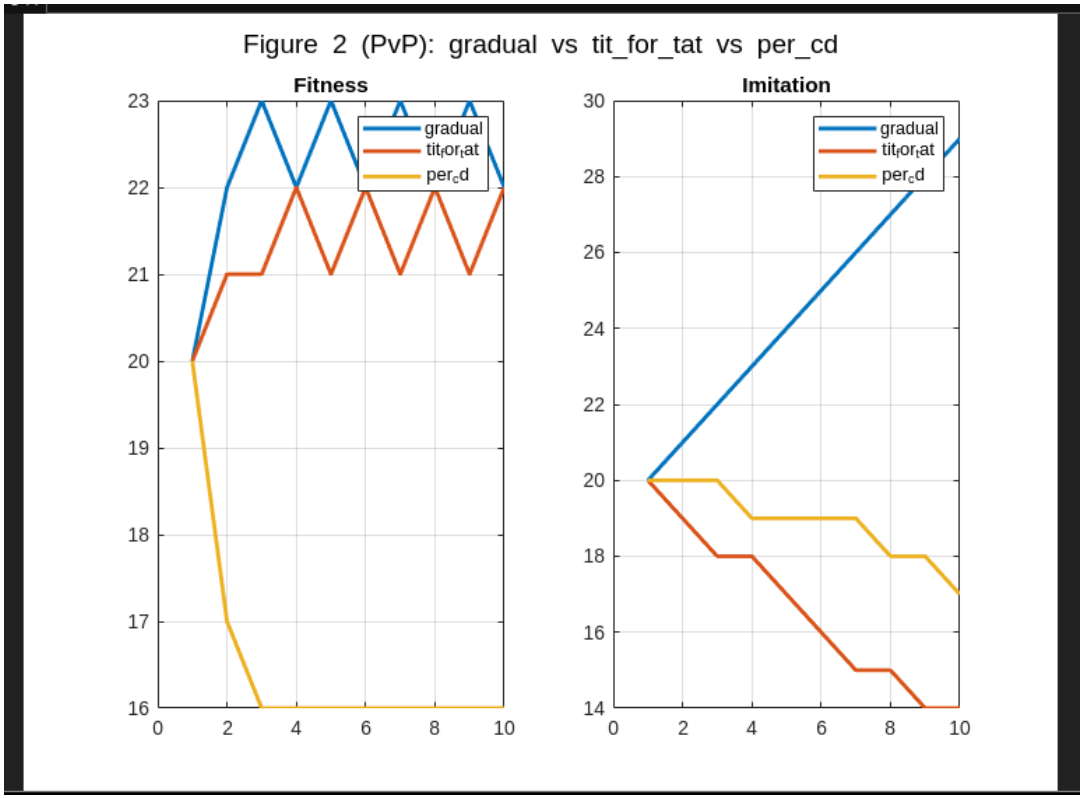
Figure 1




In this example, we see that Periodic CD and Periodic DDC have a clear advantage over Soft Majority. They keep exchanging the lead as the game goes on, with Soft Majority staying clearly behind, but managing to survive complete extinction.

Experiment Parameters:
Initial Populations:
a) Periodic DDC - 20
b) Periodic CD - 25
c) Soft Majority - 10

Figure 2



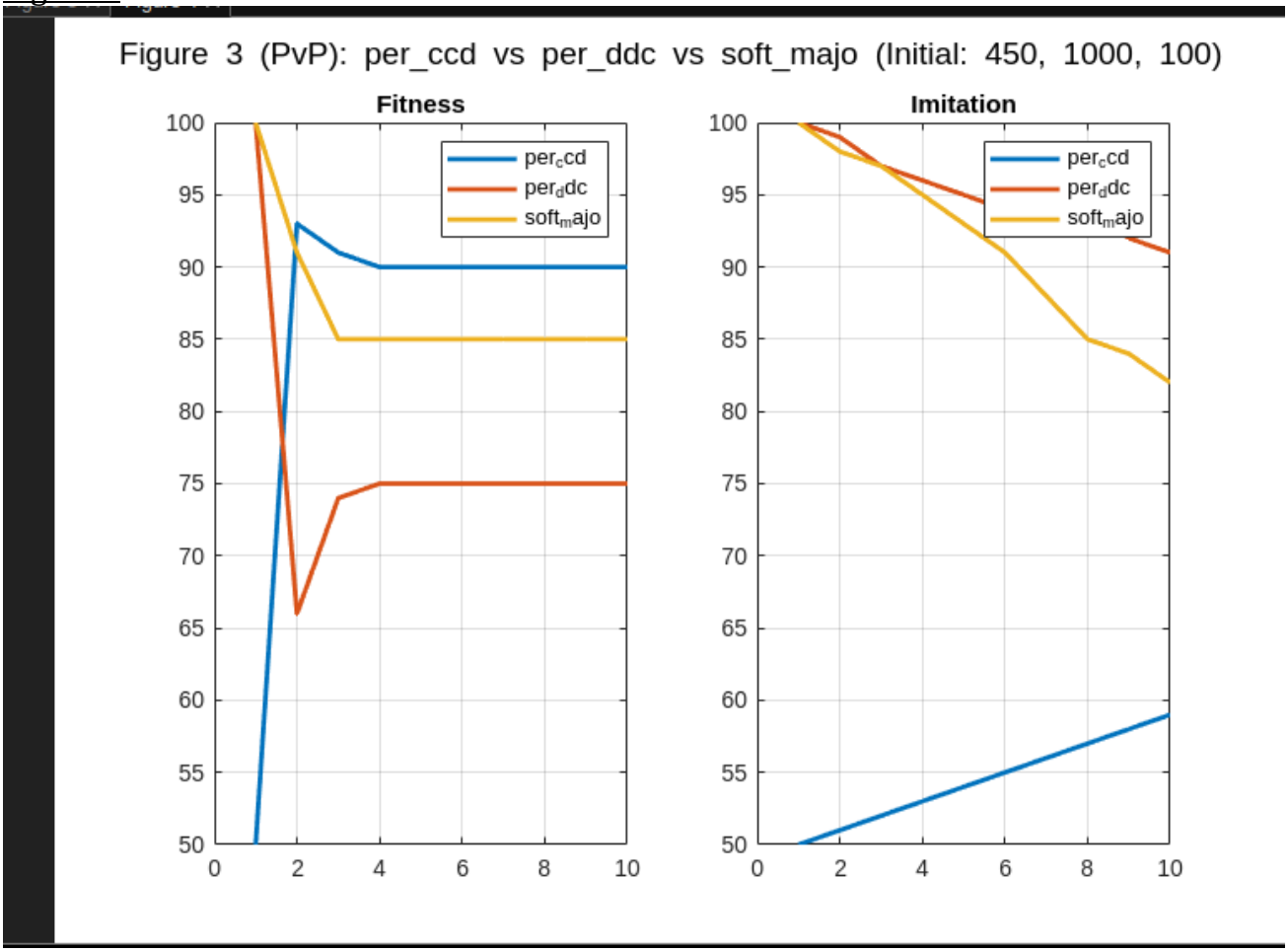

In this example, we place Gradual, Tit-Fot-Tat and Periodic CD head-to-head, for 10 generations. We can see that in both Fitness and Imitation simulations, Gradual is the winner, albeit in a tie with Tit-For-Tat in Fitness, whereas Tit-For-Tat performs noticeably worse in Imitation. Periodic CD gets eliminated in both simulations.

Experiment Parameters:

Initial Populations:

a) Gradual - 20
b) Tit-For-Tat - 20
c) Periodic CD - 20

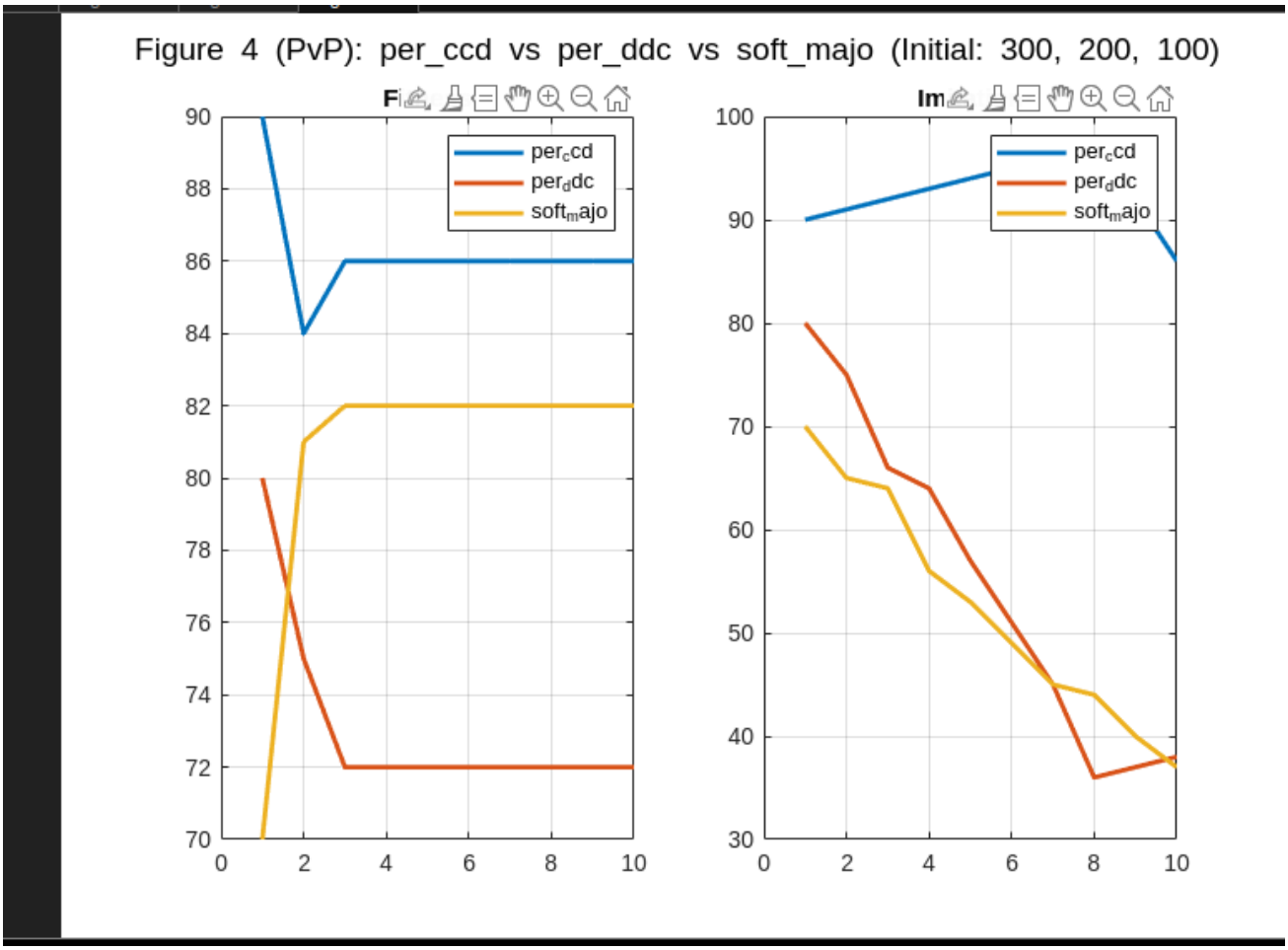
Figure 3




In this experiment, we once again summon Periodic DDC and Soft Majority, but swap Periodic CD for Periodic CCD, a nicer strategy than CD. The short 10-generation game again shows different outcomes, with Periodic CCD winning in Fitness, but coming last in Imitation, in which Periodic DDC takes the spoils. Soft Majority performs relatively well in both instances, narrowly missing out on victory in the Imitation simulation.

Experiment Parameters:
Initial Populations:
a) Periodic CCD - 50
b) Periodic DDC - 100
c) Soft Majority - 100

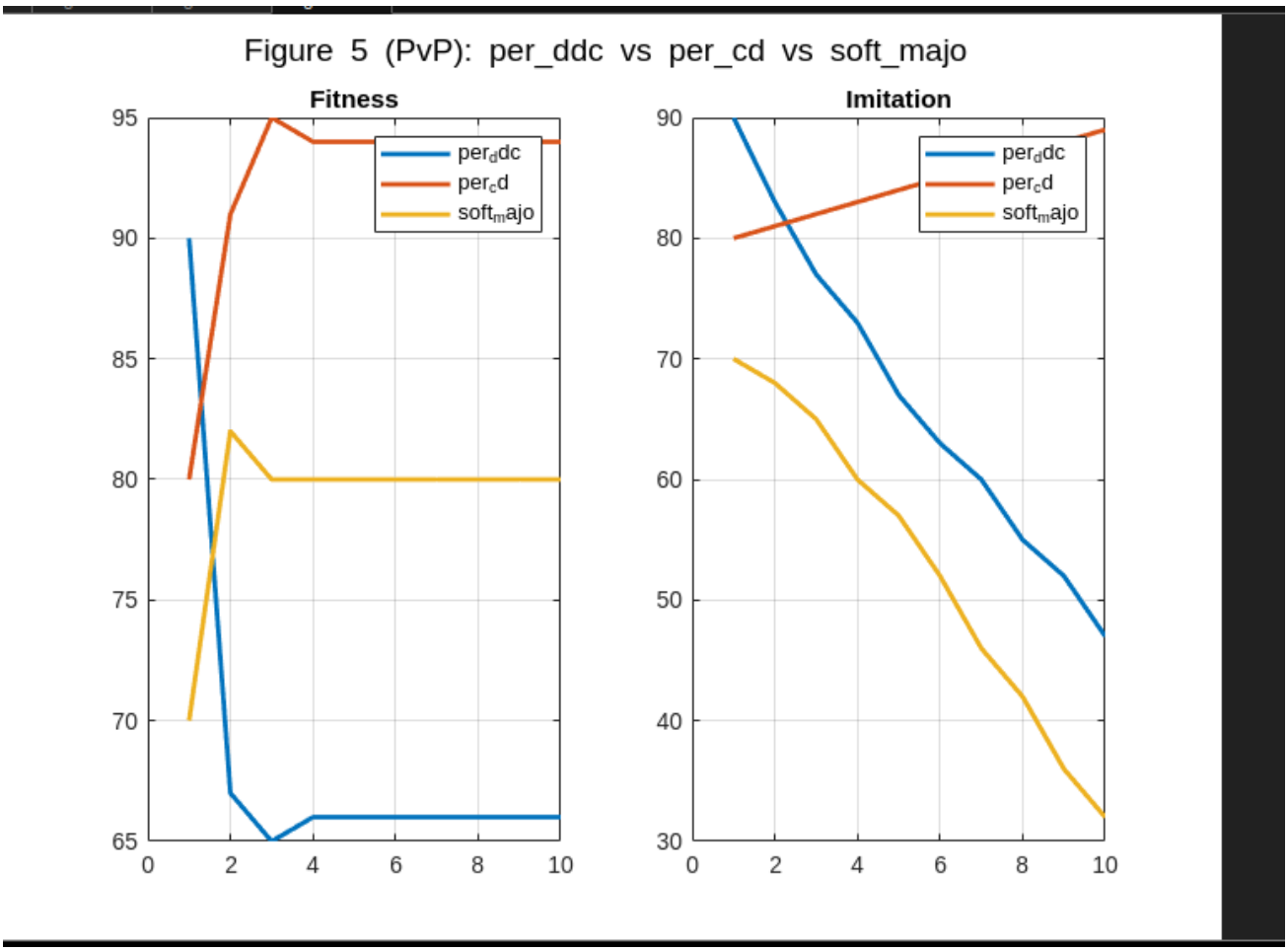
Figure 4




The erratic behavior continues in example 4, where the same strategies are once again put head-to-head, with a different initial population distribution. This time, Periodic CCD is the clear winner in both scenarios.

Experiment Parameters:
Initial Populations:
a) Periodic CCD - 90
b) Periodic DDC - 80
c) Soft Majority - 70

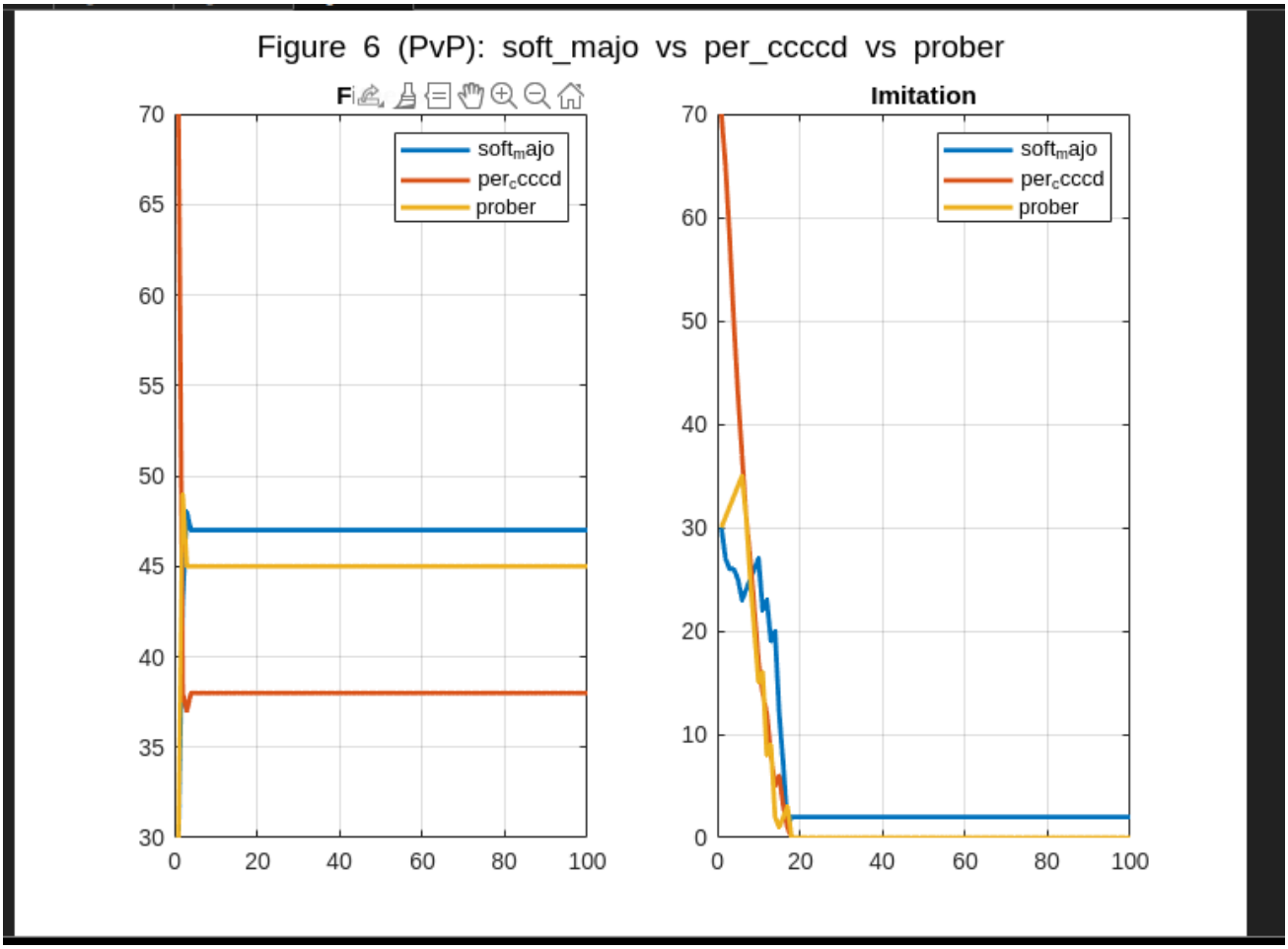
Figure 5




In this example, Soft Majority goes against Periodic DDC and CD. At the Fitness simulation, it fares quite decently and stabilizes at a population of 80, slightly higher than what it started with (70), however Imitation is completely different as its population diminishes more and more with each generation, dropping to 30 by generation 10.

Experiment Parameters:
Initial Populations:
a) Periodic DDC - 990
b) Periodic CD - 80
c) Soft Majority - 70

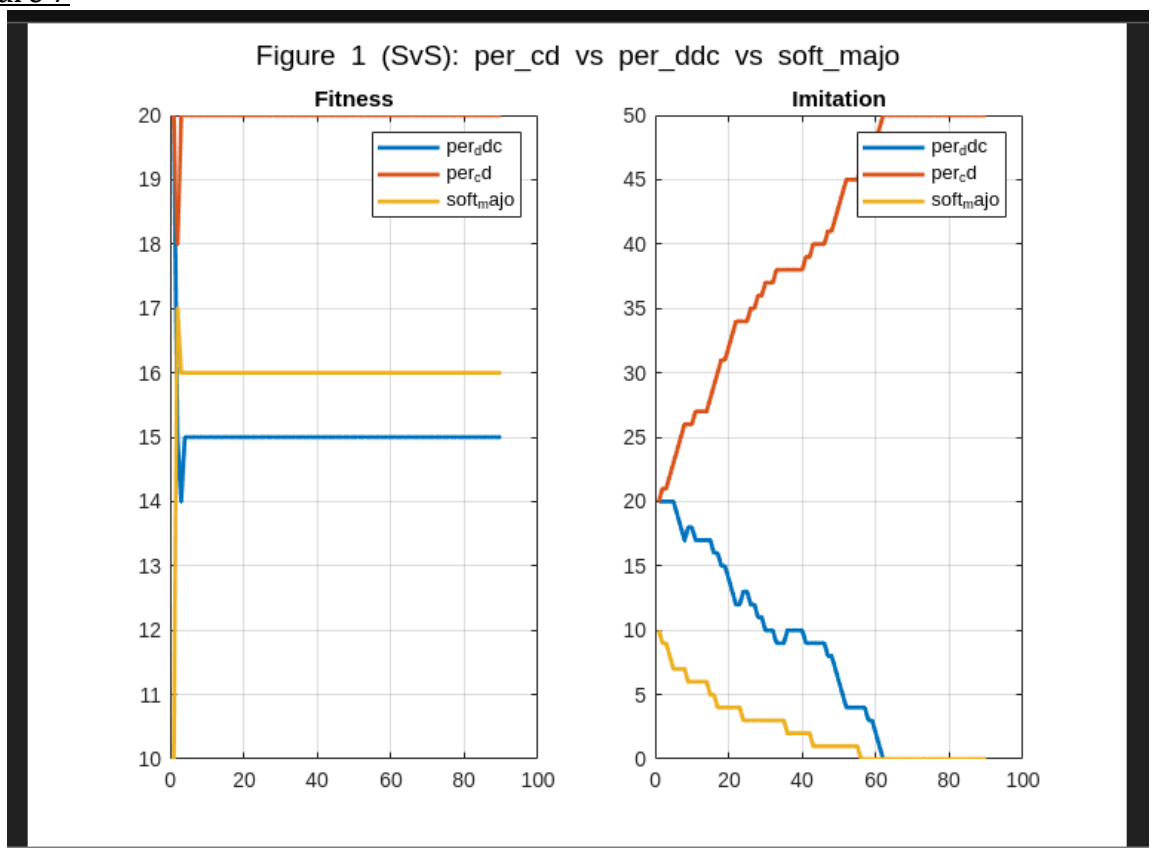
Figure 6




In this example, we can once again see Soft Majority against Prober and Periodic CCCCD. Despite being given a large handicap, the ultra-nice CCCCD strategy quickly gets exploited and its population plummets in both scenarios. What's interesting is that while Prober gets an initial growth spurt in Imitation, it completely fails shortly after and gets completely eliminated.

Experiment Parameters:
Initial Populations:
a) Soft Majority - 30
b) Periodic CCCCD - 70
c) Prober - 30

Figure 7




For our last experiment, we chose two of the periodic strategies (CD and DDC) against Soft Majority. We can see that Periodic CD dominates both scenarios, with Periodic DDC and Soft Majority surviving at Fitness but being completely eliminated at Imitation.

Experiment Parameters:
Initial Populations:
a) Periodic DDC - 20
b) Periodic CD - 20
c) Soft Majority - 10

## Comparison of Fitness Vs. Imitation Dynamics (Strategy vs Strategy)

The following figures follow the exact same experiments (and so, also initial parameters) as the ones in the previous section (Comparison of Fitness Vs. Imitation Dynamics (Player vs Player)). The only difference is now it is Strategy vs Strategy, instead of Player vs Player.

Figure 1

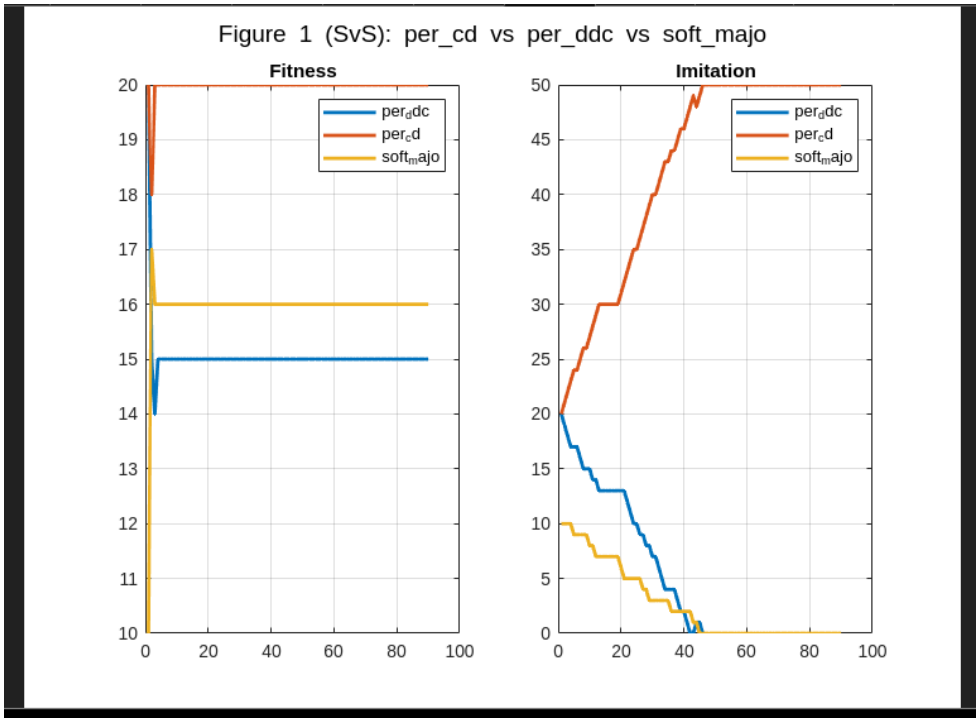



Figure 2

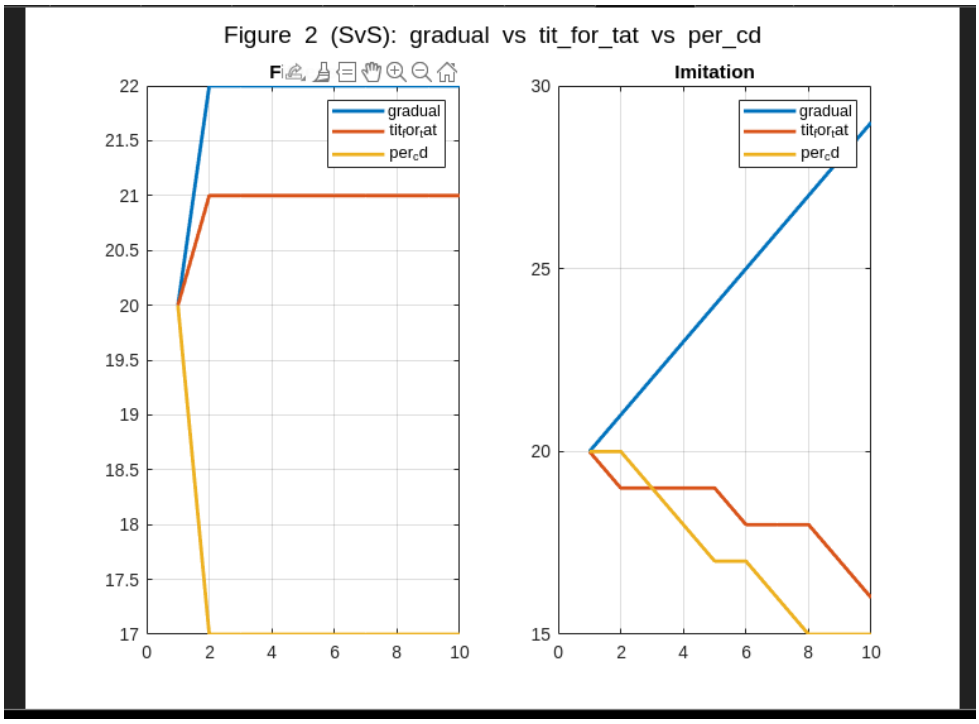


Figure 3

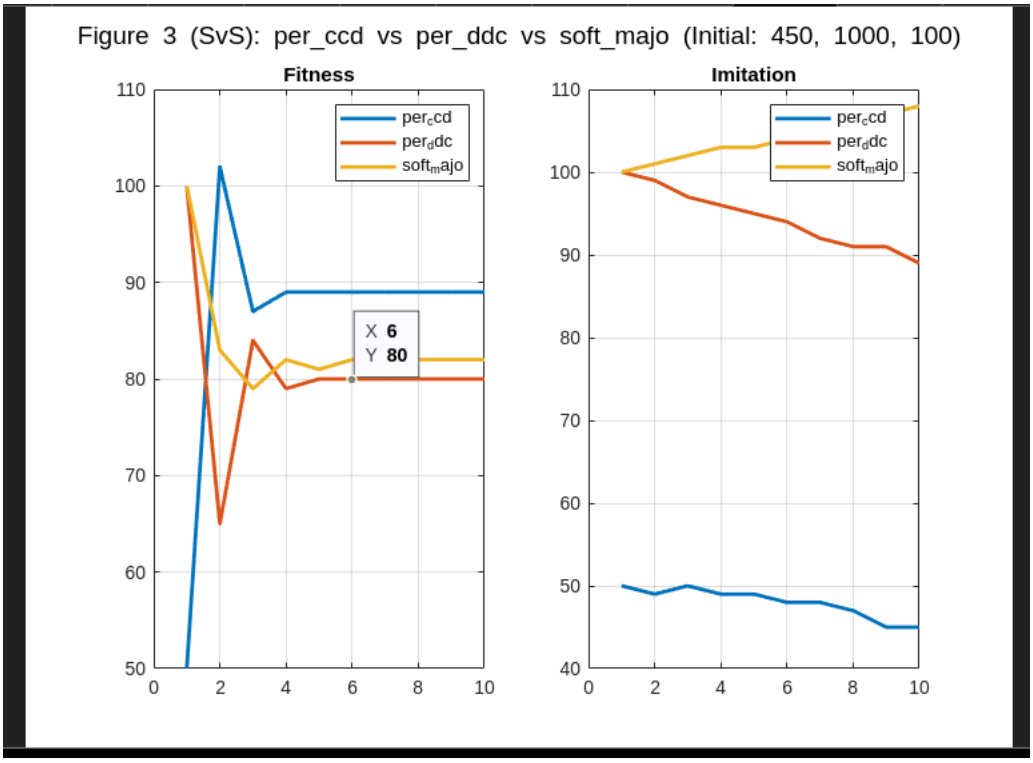



Figure 4

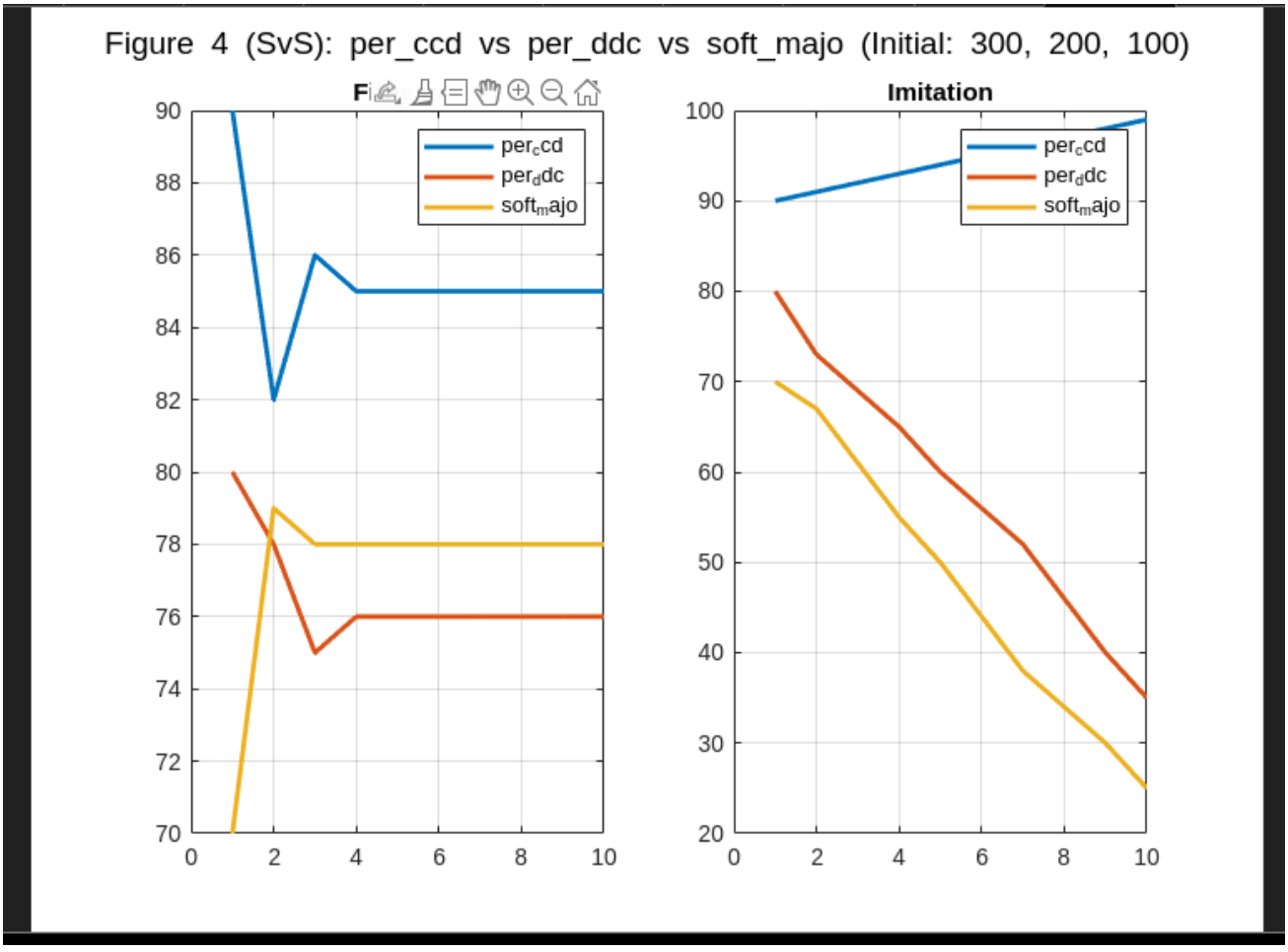


Figure 5

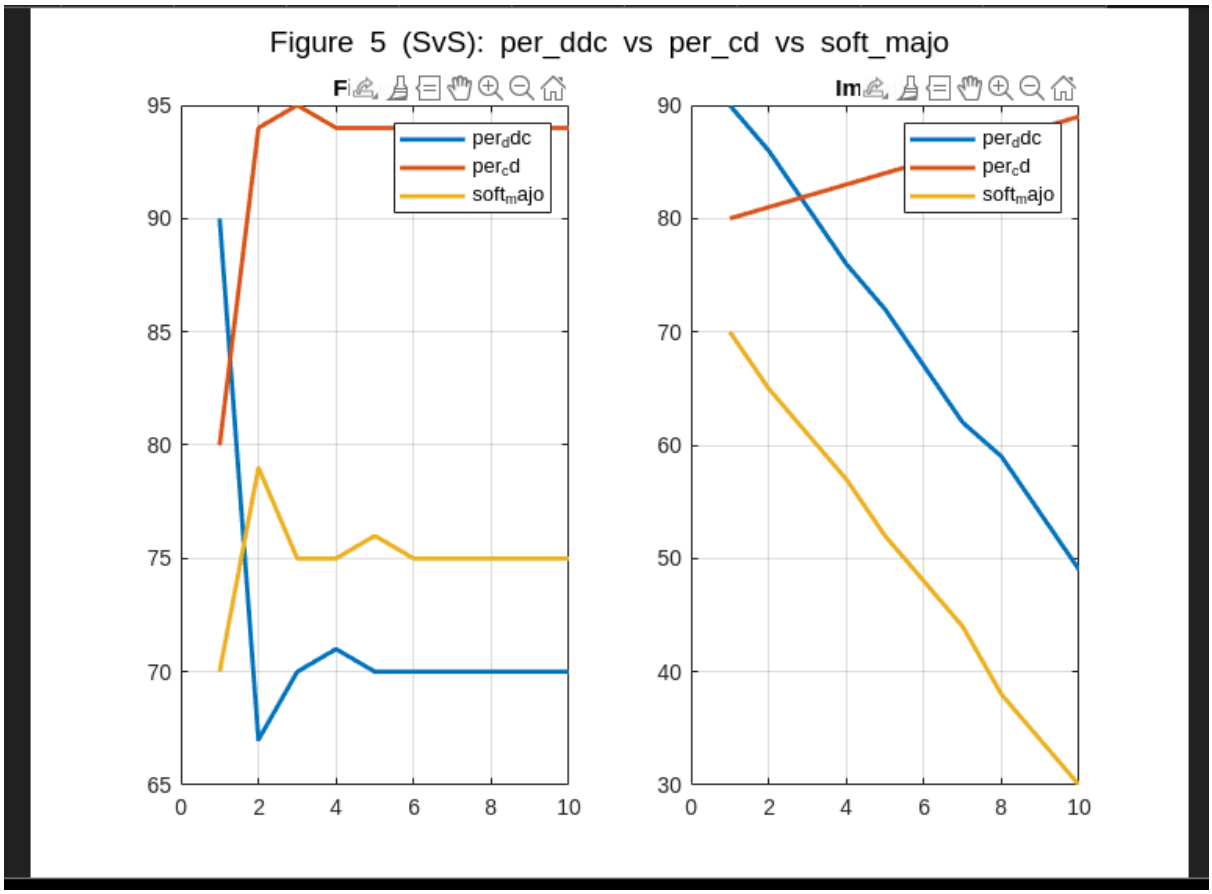
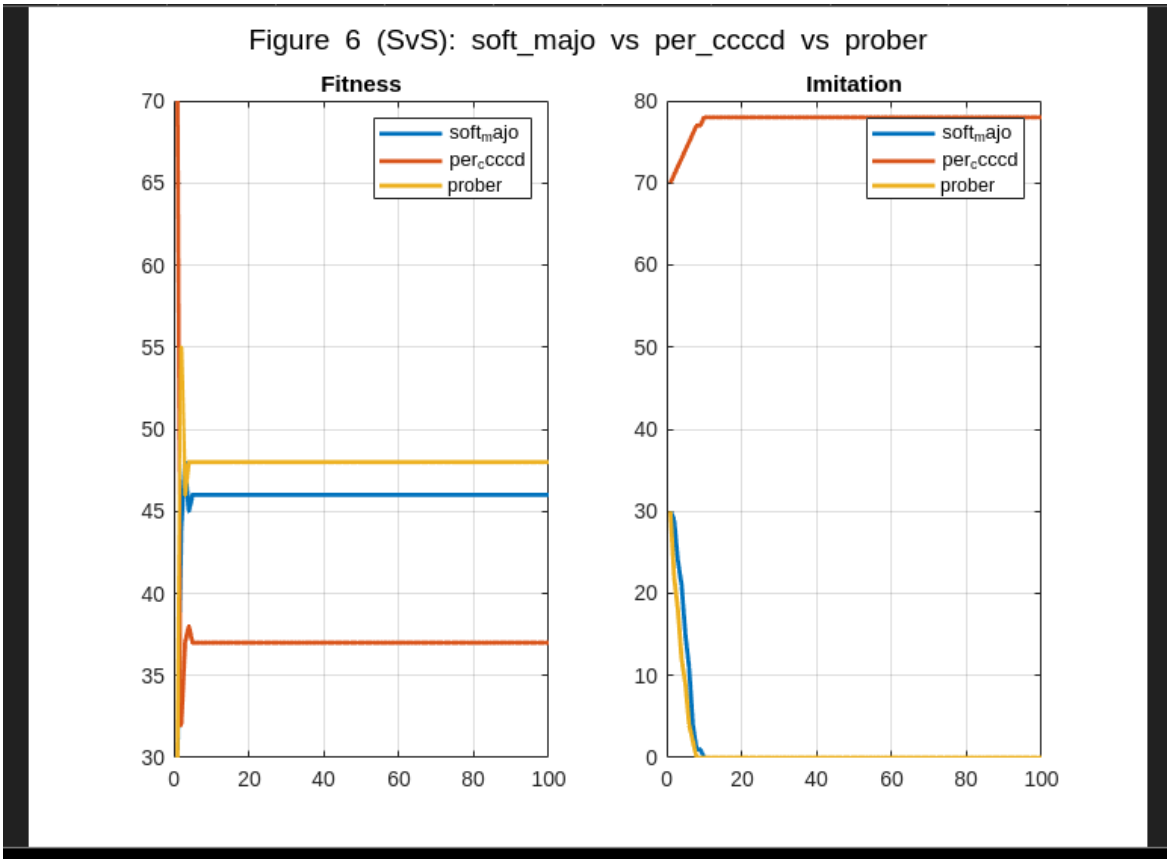



Figure 6



(E.2) Other Comments

I don't know if the following constitutes as 'interesting' in your eyes, but I will share how I first came into contact with the Prisoner's Dilemma and the reason it first fascinated with it.

It was through a game called Virtue's Last Reward. Now, what the game is about is long, and complicated because it's the second in a trilogy and has a bunch of fantasy science elements into it, but the main idea is that a group of people are locked in a warehouse and are each wearing a bracelet that is displaying the number 3.

They have to complete a puzzle room and afterwards, they have to play the following game of the Prisoner's Dilemma:


After everyone makes their choice, points will be added/subtracted to the number on your bracelet. The catch is that if the number on your bracelet is <=0, you die. The only way to escape is for the number on your bracelet to reach 9 or above.

The game follows the more moral side of the Prisoner's Dilemma, rather than the mathematical side. Is it always ok to follow strict logic? What if your opponent is unable to properly participate in the game? Are you ok with inadvertently killing someone to get out? And how will the rest of the players react to you betraying someone? Will there be a scenario when they trust you?

The characters do indeed follow their own logic and morals while choosing what move to play. Some stay constant with one choice, and some react to your earlier betrayal against a different opponent.

I know that in this class we don't really talk about this side of Game Theory and we stick to the more logical & mathematical side of it, but I always enjoyed the concept of moral dilemmas, and in the end it's the reason I'm currently taking this class.

- A comment by Γιώργος Χατζηλύρας

## (F) REFERENCES

[1] Alexander, J. McKenzie. Evolutionary game theory. Cambridge University Press, 2023.

[2] Axelrod, Robert, and William D. Hamilton. “The evolution of cooperation.” Science, Vol. 211, No. 4489, pp. 1390-1396, 1981.

[3] Axelrod, Robert, and Douglas Dion. “The further evolution of cooperation.” Science, Vol. 242, No. 4884, pp. 1385-1390, 1988.

[4] Mathieu, Philippe, Bruno Beaufils, and Jean-Paul Delahaye. “Studies on Dynamics in the Classical Iterated Prisoner’s Dilemma with Few Strategies” European conference on artificial evolution. Springer: Berlin Heidelberg, 1999.

# (G) APPENDICES

## (G.1) Documentation

Toolbox Functions:

Fitness

1. TourSimFit
2. TourSimFitPvP
3. TourTheFit

Imitation

1. TourSimImi
2. TourSimImiPvP
3. TourTheImi

Helper Functions

1. AssignStrategies
2. Axel
3. GrPlot
4. MatchPayoff
5. get_stationary_colors
6. enumerate_states

Strategies

1. All C
2. All D
3. Gradual
4. Periodic CCCCD
5. Periodic CCD
6. Periodic CD
7. Periodic DDC
8. Prober
9. Soft Majority
10. Tit-For-Tat

## (G.2) Github Repo

The link to access the toolbox and the related documents, including this report is the following:

https://github.com/tmpadasn/EvolutionaryGamesToolbox