ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Department of Electrical and Computer Engineering

Game Theory

Evolutionary Games Toolbox - Documentation

Group Number	2
Delopoulos Emmanouil	entelopo@ece.auth.gr
Vaporis Dimitrios	dvaporis@ece.auth.gr
Voulkidis Vlasios	vlasiosv@ece.auth.gr
Professor	Kehagias Athanasios

May 2025

Contents

4 Strategies of strategies subfolder in the Code folder **15**

1 Introduction

The following is a file containing the documentation for the Evolutionary Games Toolbox contained in this GitHub repository. It is split into three segments: in the first segment, the documentation of each function contained in the Code folder are presented, whereas in the second segment, the simulation corresponding to each example file in the Examples folder is noted. Lastly, the third segment contains documentation on the strategies in the strategies subfolder of the Code folder. Note that the details regarding the operation of each function are not presented in this file (a lot of this is done in the Report.pdf file in the Report folder). Instead, this is a showcase of what each function expects as input, what it returns as output, which example corresponds to which simulation and how each strategy of the toolbox behaves. Across the functions written, there are smaller helper functions that are not included in this documentation, for the sake of simplicity.

2 Functions of Code folder

This is the documentation of the functions contained in the Code folder. The inputs/outputs of each functions are presented, without showcasing the operation of each one. See also the Report.pdf as well as the source code of each function (it is properly commented) if necessary.

2.1 AnalyzeMarkovChain

The AnalyzeMarkovChain function is responsible for creating the directed state graphs of the Imitation Dynamics. It requires inputs P the transition matrix of the Markov Chain (usually calculated by the TourTheImi function), $POP0$ the initial $1 \times S$ (with S the number of strategies used in the simulation) population vector, *Strategies* an array containing the string names of the strategies used in the simulation (as named later on in the Strategies section) and *Title* the string containing the title of the outgoing graph. The function does not have any outputs, as it only creates the graph and exports it for use in the Report).

2.2 Axel

The Axel function computes the score of a single generation Axelrod tournament between specific strategies. It expects inputs B a 2×2 payoff matrix for each match, *Strategies* an array containing the string names of the strategies used in the simulation (as named later on in the Strategies section), *Pop* the initial $1 \times S$ (with S the number of strategies used in the simulation) population vector and T the number of rounds played in each match of the tournament.

It returns an $N \times 1$ array (with N being the total number of players in the simulation) containing the score of each player of the simulation at the end of the Axelrod tournament. Note that this function is not used for functions later on because of other faster implementations used.

2.3   plotFitnessVSImitation

The plotFitnessVSImitation function creates and exports a figure comparing the Fitness and Imitation Dynamics, using both "Total" and "Individual" mode for best strategy calculation in the Imitation Dynamics. It expects inputs *Strategies* an array containing the string names of the strategies used in the simulation (as named later on in the Strategies section), *POP*1 the population matrix for each generation of the Fitness Dynamics (created by TourSimFit), *POP*2 the population matrix for each generation of the Imitation Dynamics - "Individual" mode (created by TourSimImi), *POP*3 the population matrix for each generation of the Imitation Dynamics - "Total" mode (created by TourSimImi with "Total" mode input) and *Title* the title string of the created graph. The function does not have outputs and only creates and exports the graph. See example10 for an example usage of the function.

2.4   plotPopulationOfStrategiesOverGenerations

The plotPopulationOfStrategiesOverGenerations function constructs the diagram of populations of different strategies across generations after a completed simulation. It requires inputs *Strategies* an array containing the string names of the strategies used in the simulation (as named later on in the Strategies section), *POP* the population matrix for each generation of the given simulation, created by the corresponding function and *Title* the title string of the created diagram. The function does not have outputs and only creates and exports the diagram graphing the population of each strategy with respect to the generation number.

2.5   plotPopulationsOfStrategiesOverGenerations

The plotPopulationsOfStrategiesOverGenerations function creates the plots for the population of different strategies across generations for results generated by TourTheFit, TourSimFit with compensation and without compensation. It expects inputs *Strategies* an array containing the string names of the strategies used in the simulation (as named later on in the Strategies section), *POP*1 the theoretical population matrix for each generation of the Fitness Dynamics (created by TourTheFit), *POP*2 the population matrix for each generation of the Fitness Dynamics without compensation (created by TourSimFit), *POP*3

the population matrix for each generation of the Fitness Dynamics with compensation (created by TourSimFit with *compensation* = true input) and *Title* the title string of the created figure. The function does not have outputs and only creates and exports the figure. See the first few examples for example usages of the function.

2.6   plotPopulationsTourSimImi

The plotPopulationsTourSimImi function creates the plot for the population of different strategies across generations after a completed TourSimImi simulation. It expects inputs *Strategies* an array containing the string names of the strategies used in the simulation (as named later on in the Strategies section), *POP* the population matrix for each generation of the given simulation, created by the TourSimImi function and *Title* the title string of the created plot. The function does not have outputs and only creates and exports the diagram graphing the population of each strategy with respect to the generation number. See example07 for an example usage of the function.

2.7   TourSimFit

The TourSimFit function simulates the evolutionary tournament using Fitness Dynamics. The function expects inputs B a 2×2 payoff matrix for each match, *Strategies* an array containing the string names of the strategies used in the simulation (as named later on in the Strategies section), *POP*0 the initial $1 \times S$ (with S the number of strategies used in the simulation) population vector, T the number of rounds played in each match of the tournament, J the number of generations of the simulation and *compensation* an optional argument (with default value being false) that toggles the usage of random compensation to keep the total population of each generation constant (this is better explained in the Report). It returns a $(J + 1) \times S$ array *POP* containing the population of each generation for each strategy, a $J \times S$ array *BST* containing information about the best strategy of each generation (for the cell (i,j) of the array, if it has value 1 then strategy j was the best for generation i, else it has value 0) and *FIT* a $J \times S$ array that contains the fitness scores of each strategy for each generation.

2.8   TourSimImi

The TourSimImi function simulates the evolutionary tournament using Imitation Dynamics. The function expects inputs B a 2×2 payoff matrix for each match, *Strategies* an array containing the string names of the strategies used in the simulation (as named later on in the Strategies section), *POP*0 the initial

$1 \times S$ (with S the number of strategies used in the simulation) population vector, T the number of rounds played in each match of the tournament, J the number of generations of the simulation and *mode* an optional argument (with default value being "Individual") that toggles the mode for best strategy selection, with the other option being "Total" (this is better explained in the Report). It returns a $(J + 1) \times S$ array *POP* containing the population of each generation for each strategy and a $J \times S$ array *BST* containing information about the best strategy of each generation (for the cell (i,j) of the array, if it has value 1 then strategy j was the best for generation i, else it has value 0).

2.9   TourTheFit

The TourTheFit function conducts the theoretical analysis of an evolutionary tournament using Fitness Dynamics. The function expects inputs B a 2×2 payoff matrix for each match, *Strategies* an array containing the string names of the strategies used in the simulation (as named later on in the Strategies section), *POP0* the initial $1 \times S$ (with S the number of strategies used in the simulation) population vector, T the number of rounds played in each match of the tournament and J the number of generations of the simulation. It returns a $(J + 1) \times S$ array *POP* containing the population of each generation for each strategy, a $J \times S$ array *BST* containing information about the best strategy of each generation (for the cell (i,j) of the array, if it has value 1 then strategy j was the best for generation i, else it has value 0) and *FIT* a $J \times S$ array that contains the fitness scores of each strategy for each generation.

2.10   TourTheImi

The function TourTheImi computes the state transition matrix of a given evolutionary tournament using Imitation Dynamics using Markov Chain theory. The function expects inputs B a 2×2 payoff matrix for each match, *Strategies* an array containing the string names of the strategies used in the simulation (as named later on in the Strategies section), *POP0* the initial $1 \times S$ (with S the number of strategies used in the simulation) population vector, T the number of rounds played in each match of the tournament, J the number of generations of the simulation and *mode* an optional argument (with default value being "Individual") that toggles the mode for best strategy selection, with the other option being "Total" (this is better explained in the Report). It returns an array P which is the state transition matrix of the process. More specifically, the cell (i,j) of the array contains the probability that the system reaches state j at the next step, given that the system starts at state i (with system being the population distribution and each state being each possible population distribution). Use AnalyzeMarkovChain to visualize and understand the

results.

3   Examples of Examples folder

This is the documentation of the Examples contained in the Examples folder of the repository. More specifically, the situation each example simulates is presented and any possible links to the paper by Mathieu et al are mentioned.

3.1   example01

The first example shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in Figure 1, with Title "Defectors may be strong". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*per_ddc*","*Alternator*","*soft_majo*"], initial population *POP*0 = [100; 100; 100], number of rounds per match $T = 1000$ and number of generations $J = 90$.

3.2   example02

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in Figure 2, with Title "Monotonous convergence". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*Grim*","*TitForTat*","*Alternator*"], initial population *POP*0 = [100; 100; 100], number of rounds per match $T = 1000$ and number of generations $J = 10$.

3.3   example03

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in Figure 3, with Title "Attenuated oscillatory movements". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*per_ccd*","*per_ddc*","*soft_majo*"], initial population *POP*0 = [450; 1000; 100], number of rounds per match $T = 1000$ and number of generations $J = 420$.

3.4   example04

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in Figure 4, with Title "Periodic movements". The functions are called with payoff

matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*per_ccd*","*per_ddc*","*soft_majo*"], initial population *POP*0 = [300; 200; 100], number of rounds per match T = 1000 and number of generations J = 1000.

3.5   example05

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in Figure 5, with Title "Increasing oscillations". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*Alternator*","*per_ddc*","*soft_majo*"], initial population *POP*0 = [400; 300; 200], number of rounds per match T = 1000 and number of generations J = 450.

3.6   example06

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in Figure 6, with Title "Disordered oscillations". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*soft_majo*","*per_ccccd*","*Prober*"], initial population *POP*0 = [100; 500; 800], number of rounds per match T = 1000 and number of generations J = 280.

3.7   example07

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in the left plot of Figure 7, with Title "Sensitivity of dynamics to population's size". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*per_ccd*","*soft_majo*","*per_ddc*"], initial population *POP*0 = [300; 100; 244], number of rounds per match T = 1000 and number of generations J = 1000.

3.8   example08

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in the right plot of Figure 7, with Title "Sensitivity of dynamics to population's size". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*per_ccd*","*soft_majo*","*per_ddc*"], initial population *POP*0 = [300; 100; 245], number of rounds per match T = 1000 and number of generations J = 1000.

3.9 example09

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in the left plot of Figure 8, with Title "Sensitivity of winner to population's size". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*per_ddc*",- "*soft_majo*","*Alternator*"], initial population *POP*0 = [100; 159; 100], number of rounds per match T = 1000 and number of generations J = 120.

3.10 example10

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in the right plot of Figure 8, with Title "Sensitivity of winner to population's size". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*per_ddc*","*soft_majo*","*Alternator*"], initial population *POP*0 = [100; 160; 100], number of rounds per match T = 1000 and number of generations J = 120.

3.11 example11

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in the left plot of Figure 9, with Title "Sensitivity to game length". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*per_ccd*","*soft_majo*","*per_ddc*"], initial population *POP*0 = [300; 100; 244], number of rounds per match T = 7 and number of generations J = 1000.

3.12 example12

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in the right plot of Figure 9, with Title "Sensitivity to game length". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*per_ccd*","*soft_majo*","*per_ddc*"], initial population *POP*0 = [300; 100; 244], number of rounds per match T = 6 and number of generations J = 1000.

3.13 example13

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in the

left plot of Figure 10, with Title "Sensitivity to CIPD payoff". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 4.6 & 1 \end{bmatrix}$, *Strategies* = ["*per_ccd*","*soft_majo*","*per_ddc*"], initial population *POP0* = [300; 100; 244], number of rounds per match T = 1000 and number of generations J = 475.

3.14 example14

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in the right plot of Figure 10, with Title "Sensitivity to CIPD payoff". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 4.7 & 1 \end{bmatrix}$, *Strategies* = ["*per_ccd*","*soft_majo*","*per_ddc*"], initial population *POP0* = [300; 100; 244], number of rounds per match T = 1000 and number of generations J = 1000.

3.15 example15

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in the plots of Figure 11, with Title "Sensitivity to repartition computation method". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$, *Strategies* = ["*per_ccd*",-"*soft_majo*","*per_ddc*"], initial population *POP0* = [300; 100; 200], number of rounds per match T = 1000 and number of generations J = 1000. Note that the plot on the left corresponds to the plot created by TourSimFit and the plot on the right corresponds to the one created by TourTheFit.

3.16 example16

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in the left plot of Figure 12, with Title "Sensitivity to repartition computation method". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 4.6 & 1 \end{bmatrix}$, *Strategies* = ["*per_ccd*","*soft_majo*","*per_ddc*"], initial population *POP0* = [450; 100; 1000], number of rounds per match T = 1000 and number of generations J = 1000.

3.17 example17

This shows the results of TourTheFit, TourSimFit with compensation and TourSimFit without compensation in the case presented by Mathieu et al in the right plot of Figure 12, with Title "Sensitivity to repartition computation

method". The functions are called with payoff matrix $B = \begin{bmatrix} 3 & 0 \\ 4.6 & 1 \end{bmatrix}$, *Strategies* = ["*per_ccd*","*soft_majo*","*per_ddc*"], initial population *POP*0 = [45; 10; 100], number of rounds per match T = 1000 and number of generations J = 1000.

3.18   example18

This showcases the sensitivity of Imitation Dynamics, mode = "Individual" to the population's size, in the case N = 3. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = ["*All_C*","*All_D*","*Grim*"], initial population *POP*0 = [0; 0; 3], number of imitators per round K = 1, number of rounds per match T = 100 and number of generations J = 15. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population *POP*0 = [0; 0; 3], *Strategies* = ["*All_C*","*All_D*","*Grim*"] and a proper title.

3.19   example19

This showcases the sensitivity of Imitation Dynamics, mode = "Individual" to the population's size, in the case N = 5. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = ["*All_C*","*All_D*","*Grim*"], initial population *POP*0 = [0; 0; 5], number of imitators per round K = 1, number of rounds per match T = 100 and number of generations J = 15. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population *POP*0 = [0; 0; 5], *Strategies* = ["*All_C*","*All_D*","*Grim*"] and a proper title.

3.20   example20

This showcases the sensitivity of Imitation Dynamics, mode = "Individual" to the population's size, in the case N = 10. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = ["*All_C*","*All_D*","*Grim*"], initial population *POP*0 = [0; 0; 10], number of imitators per round K = 1, number of rounds per match T = 100 and number of generations J = 15. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population *POP*0 = [0; 0; 10], *Strategies* = ["*All_C*","*All_D*","*Grim*"] and a proper title.

3.21 example21

This showcases the sensitivity of Imitation Dynamics, mode = "Individual" to the population's size, in the case $N = 100$. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = ["*All_C*","*All_D*","*Grim*"], initial population $POP0 = [0; 0; 100]$, number of imitators per round $K = 1$, number of rounds per match $T = 100$ and number of generations $J = 15$. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population $POP0 = [0; 0; 100]$, *Strategies* = ["*All_C*","*All_D*","*Grim*"] and a proper title.

3.22 example22

This showcases the sensitivity of Imitation Dynamics, mode = "Individual" to the payoff matrix, in the case $a = 3.2$. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3.2 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = ["*All_C*","*All_D*","*Grim*"], initial population $POP0 = [0; 0; 10]$, number of imitators per round $K = 1$, number of rounds per match $T = 100$ and number of generations $J = 15$. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population $POP0 = [0; 0; 10]$, *Strategies* = ["*All_C*","*All_D*","*Grim*"] and a proper title.

3.23 example23

This showcases the sensitivity of Imitation Dynamics, mode = "Individual" to the payoff matrix, in the case $a = 3.4$. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3.4 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = ["*All_C*","*All_D*","*Grim*"], initial population $POP0 = [0; 0; 10]$, number of imitators per round $K = 1$, number of rounds per match $T = 100$ and number of generations $J = 15$. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population $POP0 = [0; 0; 10]$, *Strategies* = ["*All_C*","*All_D*","*Grim*"] and a proper title.

3.24 example24

This showcases the sensitivity of Imitation Dynamics, mode = "Individual" to the payoff matrix, in the case $a = 3.6$. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3.6 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = ["*All_C*","*All_D*","*Grim*"], initial population $POP0 = [0; 0; 10]$, number of imitators per round $K = 1$, number of rounds per match $T = 100$ and number of generations $J = 15$. The

function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population $POP0 = [0; 0; 10]$, *Strategies* = [*"All_C","All_D","Grim"*] and a proper title.

3.25   example25

This showcases the sensitivity of Imitation Dynamics, mode = "Individual" to the payoff matrix, in the case a = 3.8. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3.8 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = [*"All_C","All_D","Grim"*], initial population $POP0 = [0; 0; 10]$, number of imitators per round K = 1, number of rounds per match T = 100 and number of generations J = 15. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population $POP0 = [0; 0; 10]$, *Strategies* = [*"All_C","All_D","Grim"*] and a proper title.

3.26   example26

This showcases the sensitivity of Imitation Dynamics, mode = "Total" to the population's size, in the case N = 3. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = [*"All_C","All_D","Grim"*], initial population $POP0 = [0; 0; 3]$, number of imitators per round K = 1, number of rounds per match T = 100 and number of generations J = 15. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population $POP0 = [0; 0; 3]$, *Strategies* = [*"All_C","All_D","Grim"*] and a proper title.

3.27   example27

This showcases the sensitivity of Imitation Dynamics, mode = "Total" to the population's size, in the case N = 5. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = [*"All_C","All_D","Grim"*], initial population $POP0 = [0; 0; 5]$, number of imitators per round K = 1, number of rounds per match T = 100 and number of generations J = 15. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population $POP0 = [0; 0; 5]$, *Strategies* = [*"All_C","All_D","Grim"*] and a proper title.

3.28 example28

This showcases the sensitivity of Imitation Dynamics, mode = "Total" to the population's size, in the case $N = 10$. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = ["*All_C*","*All_D*","*Grim*"], initial population *POP*0 = [0; 0; 10], number of imitators per round $K = 1$, number of rounds per match $T = 100$ and number of generations $J = 15$. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population *POP*0 = [0; 0; 10], *Strategies* = ["*All_C*","*All_D*","*Grim*"] and a proper title.

3.29 example29

This showcases the sensitivity of Imitation Dynamics, mode = "Total" to the population's size, in the case $N = 100$. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = ["*All_C*","*All_D*","*Grim*"], initial population *POP*0 = [0; 0; 100], number of imitators per round $K = 1$, number of rounds per match $T = 100$ and number of generations $J = 15$. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population *POP*0 = [0; 0; 100], *Strategies* = ["*All_C*","*All_D*","*Grim*"] and a proper title.

3.30 example30

This showcases the sensitivity of Imitation Dynamics, mode = "Total" to the payoff matrix, in the case $a = 3.2$. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3.2 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = ["*All_C*","*All_D*","*Grim*"], initial population *POP*0 = [0; 0; 10], number of imitators per round $K = 1$, number of rounds per match $T = 100$ and number of generations $J = 15$. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population *POP*0 = [0; 0; 10], *Strategies* = ["*All_C*","*All_D*","*Grim*"] and a proper title.

3.31 example31

This showcases the sensitivity of Imitation Dynamics, mode = "Total" to the payoff matrix, in the case $a = 3.4$. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3.4 & 1 \\ 4 & 2 \end{bmatrix}$, *Strategies* = ["*All_C*","*All_D*","*Grim*"], initial population *POP*0 = [0; 0; 10], number of imitators per round $K = 1$, number of

rounds per match T = 100 and number of generations J = 15. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population $POP0$ = $[0; 0; 10]$, $Strategies$ = [”All_C”,”All_D”,”$Grim$”] and a proper title.

3.32   example32

This showcases the sensitivity of Imitation Dynamics, mode = "Total" to the payoff matrix, in the case a = 3.6. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3.6 & 1 \\ 4 & 2 \end{bmatrix}$, $Strategies$ = [”All_C”,”All_D”,”$Grim$”], initial population $POP0$ = $[0; 0; 10]$, number of imitators per round K = 1, number of rounds per match T = 100 and number of generations J = 15. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population $POP0$ = $[0; 0; 10]$, $Strategies$ = [”All_C”,”All_D”,”$Grim$”] and a proper title.

3.33   example33

This showcases the sensitivity of Imitation Dynamics, mode = "Total" to the payoff matrix, in the case a = 3.8. The function TourTheImi is called with payoff matrix $B = \begin{bmatrix} 3.8 & 1 \\ 4 & 2 \end{bmatrix}$, $Strategies$ = [”All_C”,”All_D”,”$Grim$”], initial population $POP0$ = $[0; 0; 10]$, number of imitators per round K = 1, number of rounds per match T = 100 and number of generations J = 15. The function PlotStateTransitionGraph is then called with arguments state transition matrix P as calculated by TourTheImi, initial population $POP0$ = $[0; 0; 10]$, $Strategies$ = [”All_C”,”All_D”,”$Grim$”] and a proper title.

4   Strategies of strategies subfolder in the Code folder

Lastly, this is the documentation for all strategies included in the strategies subfolder of the Code folder. For more details, the exact source code is available for each of the following strategies. Note that you can always enhance this collection with strategies of your own. For your created strategies to work with the toolbox, follow the format used in the strategies given: let the function have form function Move = YourStrategy(History), where History is the $T \times 2$ matrix of moves by both players for each turn, regard the player of your strategy as player 1 (the first column of the History matrix, it is flipped in case the player is actually player 2) and make Move = 2 in case of defection and Move = 1 in case of cooperation. Also, suppose that each strategy can only access past

moves (player 2 cannot access player 1's current round move) and that each strategy does not know the amount of rounds played in each match. Lastly, because of the current form of simulation and theoretical analysis functions (that prioritize low running times), strategies including random elements are not included (in case you wish to include some, most functions will probably fail). For the strategies below, C stands for Cooperation and D stands for Defection.

4.1 All_C

Strategy that always cooperates.

4.2 All_D

Strategy that always defects.

4.3 Alternator

Strategy that alternates between cooperating and defecting, starting with cooperation in the first round.

4.4 Cycler

Strategy that permanently cycles the moves C, C, C, D.

4.5 Detective

Strategy that starts with the moves C, D, C, C. If the opponent defects at least once in these first four rounds, play as TitForTat forever. Else, always defect.

4.6 Grim

Strategy that always cooperates unless the opponent defects: after the first opponent defection, always defect.

4.7 per_ccccd

Strategy that permanently cycles the moves C, C, C, C, D.

4.8 per_ccd

Strategy that permanently cycles the moves C, C, D.

4.9 per_ddc

Strategy that permanently cycles the moves D, D, C.

4.10 Prober

Strategy that starts with the moves D, C, C. If the opponent cooperates at moves 2 and 3, plays defect for the rest of the game. Otherwise plays as TitForTat.

4.11 SneakyTitForTat

Strategy that starts by cooperating in the first round and playing as TitForTat in the second round. If the opponent has not defected in these first two rounds, defect once and check the opponent's response. If the opponents retaliates, play a cooperation to repent and then play as TitForTat. If the opponent does not retaliate, keep playing defect and checking the opponent's response as before.

4.12 soft_majo

Strategy that plays the same move as the majority of the opponent's moves up until the previous round. In case of equality, cooperate.

4.13 SpitefulTitForTat

Strategy that starts by cooperating and then plays like TitForTat, until the opponent defects twice in a row, in which case the player always defects.

4.14 TitForTat

Strategy that cooperates in the first move and for the rest of the game plays the opponent's previous move.

4.15 TitForTwoTats

Strategy that plays defect only if the opponent has defected twice in a row. Otherwise, always cooperates.

4.16 TwoTitsForTat

Strategy that starts by cooperating and replies to each opponent defection by two defections of its own. More specifically, if there has been a defection by the opponent in the two previous rounds, the player chooses defect. Otherwise, plays cooperate.