

**Akademia Górniczo-Hutnicza  
im. Stanisława Staszica w Krakowie**

---

Wydział Fizyki i Informatyki Stosowanej

KATEDRA INFORMATYKI STOSOWANEJ I FIZYKI KOMPUTEROWEJ



**PRACA INŻYNIERSKA**

**ERNEST JĘCZMIONEK**

**SYMULACJE EWOLUCJI KOALICJI MIESZANYCH**

PROMOTOR:

prof. dr hab. Krzysztof Kułakowski

Kraków 2017

## **OŚWIADCZENIE AUTORA PRACY**

OŚWIADCZAM, ŚWIADOMY ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZĄ PRACĘ DYPLOMOWĄ WYKONAŁEM OSOBIŚCIE I SAMODZIELNIE, I NIE KORZYSTAŁEM ZE ŹRÓDEŁ INNYCH NIŻ WYMIENIONE W PRACY.

.....

PODPIS

**AGH**  
**University of Science and Technology in Krakow**

---

Faculty of Physics and Applied Computer Science  
DEPARTMENT OF APPLIED INFORMATICS AND COMPUTATIONAL PHYSICS



**BACHELOR OF SCIENCE THESIS**

**ERNEST JĘCZMIONEK**

**SIMULATIONS OF EVOLUTION OF MIXED COALITIONS**

SUPERVISOR:  
Professor Krzysztof Kułakowski

Krakow 2017

Serdecznie dziękuję ... tu ciąg dalszych  
podziękowań np. dla promotora, żony,  
sąsiada itp.

## Spis treści

<b>1. Wprowadzenie</b>	6
<b>2. Opis teoretyczny</b>	7
2.1. Gra	7
2.2. Model gry	7
2.3. Równania standardowe	8
2.4. Równania replikatorów	9
2.5. Ograniczenie prawdopodobieństwa	9
2.6. Ograniczenie prawdopodobieństwa - przykłady	10
2.7. Rozwiązanie stacjonarne równań standardowych	12
2.8. Stabilność równań replikatorów	12
<b>3. Implementacja symulacji</b>	14
3.1. Środowisko QT	14
3.2. GLWidget	14
3.3. Schemat programu	14
3.4. Kod	14
3.5. Rysowanie 3D	18
3.6. Makefile	18
<b>4. Wyniki</b>	19
4.1. Gry 3-osobowe	19
4.2. Gry N-osobowe	21
<b>5. Podsumowanie</b>	23

# 1. Wprowadzenie

Teoria gier wielu osobom kojarzy się z opisem gier towarzyskich między dwojgiem graczy, lecz takie rozgrywki to rzadkość w naszym zróżnicowanym świecie, gdzie zwykle w grę ekonomiczną, społeczną czy polityczną angażuje się wielu uczestników. W niniejszej pracy zostaną umówione gry wieloosobowe o niepełnej informacji. W tym typie gier ważnym elementem strategii jest odpowiedni wybór koalicjantów. Oczywiście nie jest możliwe aby uwzględnić wszystkie czynniki mogące mieć wpływ na grę, ale zostaną przeanalizowane dwa równania ewolucyjne, które mogłyby sterować graczami oraz przeprowadzona będzie analiza ich stabilności. Modelami gry użytymi w niniejszej pracy będzie gra 3-osobowa oraz gra wieloosobowa, w której gracze będą ustawieni w okręgu. Symulację partii w przypadku gier 3-osobowych będą obrazowane jako trójwymiarowe funkcje prawdopodobieństwa, natomiast dla gier wieloosobowych jako funkcje prawdopodobieństwa od czasu dla poszczególnych graczy.

!!!TEST CYTATÓW!!! [Now06] [HS98] [P.01] [Qt] [Tut] [Sza] [Fsm] [Crf]

## 2. Opis teoretyczny

### 2.1. Gra

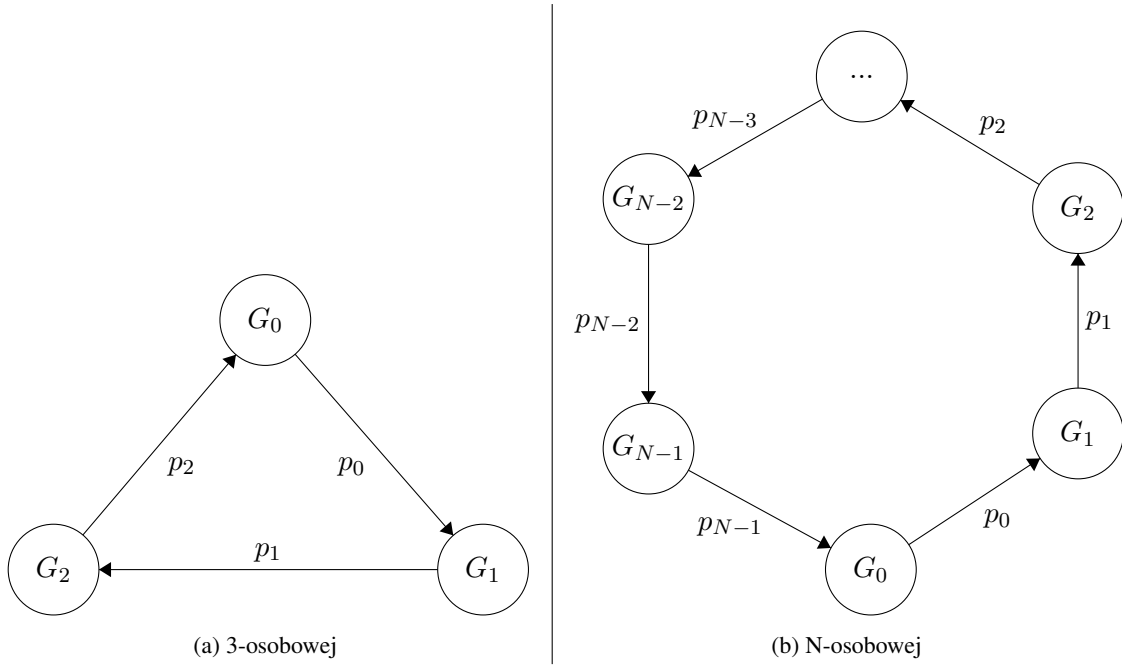
Niniejsza praca skupia się na grach wieloosobowych, których szczególnym przypadkiem jest gra 3-osobowa. Model gry 3-osobowej pokazuje, że decyzję jednego z zawodników mają bezpośredni wpływ na zachowanie sąsiadów. W modelu gry N-osobowej decyzje graczy będą wpływać nie tylko na najbliższych sąsiadów, ale pośrednio także na decyzje innych graczy. Omawiane tutaj gry są grami o niepełnej informacji, w tej pracy oznacza to grę, w której nie wszyscy uczestnicy znają prawdopodobieństwo wyborów przeciwników. Prawdopodobieństwo gry przeciwników będzie szacowane na podstawie obserwacji historii ich zagrań. Opisywane gry będą także grami ewolucyjnymi, czyli mającymi możliwość przewidywania i uczenia się na podstawie zachowań graczy.

### 2.2. Model gry

Jak wcześniej wspomniano będą potrzebne dwa modele gier. Pierwszym z nich będzie model gry 3-osobowej 2.1a. Graczy nazwiemy odpowiednio  $G_0$ ,  $G_1$ ,  $G_2$  (gracz zerowy, pierwszy, drugi). Każdy z nich posiada prawdopodobieństwo zagrania, w celu nawiązania koalicji z graczem o wyższym indeksie. To prawdopodobieństwo oznaczone jest jako  $p_i$  ( $i$  jest indeksem gracza), przy czym dla  $G_2$  gracz o wyższym indeksie to  $G_0$ . Prawdopodobieństwo zagrania, w celu nawiązania koalicji z graczem o niższym indeksie to  $1 - p_i$ , co nie będzie przechowywane, ponieważ w łatwy sposób można to wyliczyć. Analogicznie dla  $G_0$  gracz o niższym indeksie to  $G_2$ . Ponieważ omawiana jest gra o niepełnej informacji żaden z zawodników nie ma dostępu do prawdopodobieństw innych graczy, lecz każdy ma dostęp do statystyki gry na którą składa się:

- $liczba_{partii}$  liczba rozegranych w grze partii
- $nast_i$  ilość zagrań  $G_i$  w celu zawiązania sojuszu z  $G_{i+1}$ , ilość zagrań  $G_i$  aby nawiązać sojusz z  $G_{i-1}$  wyraża się jako  $liczba_{partii} - nast_i$

Model gry N-osobowej 2.1b będzie dysponował takimi samymi danymi jak model gry 3-osobowej. Różnicą między nimi będzie ustawienie graczy w okręgu, co będzie prowadziło do konsekwencji nieograniczających się do najbliższych sąsiadów. **UKŁAD**



Rysunek 2.1: Modele gry

## 2.3. Równania standardowe

W tym typie równań celem będzie maksymalizacja zysku( zyskiem jest prawdopodobieństwo ) w czasie. Wyprowadzenie będzie przeprowadzone dla  $G_0$ , przy założeniu gry o pełnej informacji. Pozostałe dwa równania można wyprowadzić analogicznie. Niech  $x = p_0$ ,  $y = p_1$ ,  $z = p_2$ . Wypłata dana jest przez:

$$W = \underbrace{x(1-y)}_{\text{wypłata sojuszu } G_0 \text{ z } G_1} + \underbrace{(1-x)z}_{\text{wypłata sojuszu } G_0 \text{ z } G_2} \quad (2.1)$$

Aby znaleźć dynamikę należy zróżniczkować wypłatę po  $\delta x$ .

$$\frac{\delta W}{\delta x} = 1 - x - y = W_x$$

Iloraz różnicowy na zmianę wypłaty od czasu daje:

$$\begin{aligned} W_x(t) &= \frac{W_x(t+dt) - W_x(t)}{\Delta t} \\ W_x(t)\Delta t &= W_x(t+dt) - W_x(t) = \Delta p_0 \\ \Delta p_0 &= \Delta t(1 - y - z) \end{aligned} \quad (2.2)$$

W dalszej części będzie stosowane oznaczenie  $\alpha = \Delta t$ . Równanie wygląda obiecująco, gdyż człon  $1 - y$  jest prawdopodobieństwem zagrania  $G_1$  aby zawiązać sojusz z  $G_0$ . Podobnie  $-z$  prowadzi do sojuszu  $G_2$  z  $G_0$ , minus przy  $z$  powoduje ubytek dla  $x$ , ponieważ prawdopodobieństwo zagrania  $G_0$  w celu zawarcia sojuszu z  $G_2$  jest dane przez  $1 - x$ .



W przypadku gry o niepełnej informacji należy zmodyfikować równanie 2.2. Opis użytych parametrów można znaleźć w podrozdziale 2.2.

$$\Delta p_i = \alpha \cdot \left(1 - \frac{nast_{i+1}}{liczba_{partii}} - \frac{nast_{i-1}}{liczba_{partii}}\right) \quad (2.3)$$

## 2.4. Równania replikatorów

Model dynamiki replikatorów jest najbardziej znanym różniczkowym modelem teorii gier ewolucyjnych, przez co może stanowić dobry wybór do sterowania zachowaniem graczy. Tak jak poprzednio przyjmijmy prawdopodobieństwa  $x = p_0$ ,  $y = p_0$  oraz  $z = p_0$  pamiętając, że prawdopodobieństwa przeciwników w symulacji są szacowane. Podstawowy wzór równania replikatorów wygląda następująco:

$$\dot{x} = x \cdot (W_x - \bar{W}) \quad (2.4)$$

Gdzie  $W_x$  jest średnią wypłatą dla strategii  $x$  (prawdopodobieństw strategii  $x$ ), natomiast  $\bar{W}$  jest średnią wypłatą co daje:

$$\begin{aligned} \dot{x} &= x \cdot \left( \overbrace{(1-y)}^{W_x} - \overbrace{(x(1-y) + (1-x)z)}^{\bar{W}} \right) \\ &\quad \Downarrow \\ \dot{x} &= x \cdot (1-x) \cdot (1-y-z) \end{aligned}$$

Iloraz różnicowy zmiany wypłaty od czasu ponownie wyznaczy  $\Delta p_0$ :

$$\begin{aligned} x(t) &= \frac{x(t + \Delta t) - x(t)}{\Delta t} \\ x(t)\Delta t &= x(t + \Delta t) - x(t) = \Delta p_0 \end{aligned}$$

W dalszej części tej pracy  $\alpha = \Delta t$ . Co dla gry 3-osobowej generuje równania:

$$\begin{aligned} \Delta p_0 &= \alpha p_0 \cdot (1 - p_0) \cdot \left(1 - \frac{n_1}{liczba_{partii}} - \frac{n_2}{liczba_{partii}}\right) \\ \Delta p_1 &= \alpha p_1 \cdot (1 - p_1) \cdot \left(1 - \frac{n_2}{liczba_{partii}} - \frac{n_0}{liczba_{partii}}\right) \\ \Delta p_2 &= \alpha p_2 \cdot (1 - p_2) \cdot \left(1 - \frac{n_0}{liczba_{partii}} - \frac{n_1}{liczba_{partii}}\right) \end{aligned} \quad (2.5)$$

## 2.5. Ograniczenie prawdopodobieństwa

Wszystkie zmiany prawdopodobieństwa muszą zostać poddane funkcji ograniczającej.

$$p_i = ogr(p_i + \Delta p_i) \quad (2.6)$$

W przeciwnym razie równania z poprzednich podrozdziałów mogą wyjść poza przedział  $< 0, 1 >$ . Każde nowo obliczone prawdopodobieństwo podawane jest jako parametr do funkcji *ogr*, a dopiero jej

rezultat jest przypisywany poszczególnym prawdopodobieństwom graczy. Funkcja dbająca o pozostanie prawdopodobieństwa w dziedzinie wygląda następująco:

$$ogr(p_i) = \begin{cases} 1 & \text{jeżeli } p_i > 1 \\ p_i & \text{jeżeli } 1 \geq p_i \geq 0 \\ 0 & \text{jeżeli } p_i < 0 \end{cases}$$

Wyjaśnienia wymaga wartość  $\alpha$ , która została przyjęta jako 0.1. Jest to wartość przyjęta arbitralnie aby spowolnić rozgrywkę oraz umożliwić łatwiejszą zmianę koalicji.

## 2.6. Ograniczenie prawdopodobieństwa - przykłady

Najpierw przeanalizowane zostaną równania standardowe dla gry 3-osobowej o prawdopodobieństwie początkowym  $\frac{1}{2}$  i  $\alpha = 1$ . Gdzie  $N$  oznacza zagranie  $G_i$  mające na celu zawiązanie sojuszu z  $G_{i+1}$ , a  $P$  ilość zagranie  $G_i$  mające na celu zawiązanie sojuszu z  $G_{i-1}$ . **UKŁAD**

$$\begin{array}{cc} G_0 = N, G_1 = P, G_2 = N & G_0 = N, G_1 = P, G_2 = P \\ \left\{ \begin{array}{ll} \Delta p_0 = (1 - 0 - 1) = 0 & p_0 = \frac{1}{2} \\ \Delta p_1 = (1 - 1 - 1) = -1 & p_1 = 0 \\ \Delta p_2 = (1 - 1 - 0) = 0 & p_2 = \frac{1}{2} \end{array} \right. & \left\{ \begin{array}{ll} \Delta p_0 = (1 - 0 - \frac{1}{2}) = 0 & p_0 = \frac{3}{4} \\ \Delta p_1 = (1 - \frac{1}{2} - 1) = -\frac{1}{2} & p_1 = 0 \\ \Delta p_2 = (1 - 0 - \frac{1}{2}) = \frac{1}{2} & p_2 = \frac{3}{4} \end{array} \right. \end{array}$$

Schemat przedstawia dwie osobne tury: po lewej pierwszą, a po prawej drugą. Parametr  $\alpha = 1$  powoduje szybkie zawiązanie mocnych koalicji, których przerwanie staje się mało prawdopodobne, co może praktycznie uniemożliwiać jakiegokolwiek zmiany sojuszy.

Wartym rozważenia jest czy w równaniach replikatorów potrzebna będzie  $\alpha < 1$ , skoro posiadają one człon postaci  $x(1 - x)$  gasnący na krańcach przedziału prawdopodobieństwa. Założenia zostały oparte na poprzednim przykładzie.

$$\begin{array}{cc} G_0 = N, G_1 = P, G_2 = N & \\ \left\{ \begin{array}{ll} \Delta p_0 = \frac{1}{2} \cdot (1 - \frac{1}{2}) \cdot (1 - 0 - 1) = 0 & p_0 = \frac{1}{2} \\ \Delta p_1 = \frac{1}{2} \cdot (1 - \frac{1}{2}) \cdot (1 - 1 - 1) = 0 & p_1 = \frac{1}{4} \\ \Delta p_2 = \frac{1}{2} \cdot (1 - \frac{1}{2}) \cdot (1 - 0 - 1) = 0 & p_2 = \frac{1}{2} \end{array} \right. & \\ G_0 = N, G_1 = P, G_2 = P & \\ \left\{ \begin{array}{ll} \Delta p_0 = \frac{1}{2} \cdot (1 - \frac{1}{2}) \cdot (1 - 0 - \frac{1}{2}) = \frac{1}{8} & p_0 = \frac{5}{8} \\ \Delta p_1 = \frac{1}{4} \cdot (1 - \frac{1}{4}) \cdot (1 - \frac{1}{2} - 1) = -\frac{1}{32} & p_1 = \frac{7}{32} \\ \Delta p_2 = \frac{1}{2} \cdot (1 - \frac{1}{2}) \cdot (1 - 0 - \frac{1}{2}) = \frac{1}{8} & p_2 = \frac{5}{8} \end{array} \right. & \end{array}$$

Jak widać najszybsza zmiana zachodzi dla pierwszego gracza, który traci 25% z prawdopodobieństwa sojuszu z graczem o wyższym indeksie. W kolejnej partii nie widzimy już tak dużych zmian. Należy odpowiedzieć na pytanie czy jest to na tyle dużo, aby zastosować  $\alpha$  taką jak w równaniach standardowych. Powinno się poruszyć dwie kwestie. Dynamika prawdopodobieństwa jest akceptowalna (wynosi do 10%), ale tylko dla prawdopodobieństw które oddalają się od środka przedziału, a grę rozpoczynamy

właśnie w nim. Drugą sprawą jest porównanie obu równań. Porównanie wyników, o różnym kroku czasowym nie jest najłatwiejszą rzeczą do opisanie.

Omówiony teraz zostanie szczególny przypadek, gdy w dwóch osobnych grach żaden z graczy nie współpracował. Użyte będą równania standardowe, prawdopodobieństwo początkowe  $\frac{1}{2}$  oraz  $\alpha = 1$ .

$$\begin{array}{cc} G_0 = N, G_1 = N, G_2 = N & G_0 = P, G_1 = P, G_2 = P \\ \left\{ \begin{array}{l} \Delta p_0 = (1 - 1 - 1) = -1 \quad p_0 = 0 \\ \Delta p_1 = (1 - 1 - 1) = -1 \quad p_1 = 0 \\ \Delta p_2 = (1 - 1 - 1) = -1 \quad p_2 = 0 \end{array} \right. & \left\{ \begin{array}{l} \Delta p_0 = (1 - 0 - 0) = 1 \quad p_0 = 1 \\ \Delta p_1 = (1 - 0 - 0) = 1 \quad p_1 = 1 \\ \Delta p_2 = (1 - 0 - 0) = 1 \quad p_2 = 1 \end{array} \right. \end{array}$$

Przypadek ten pokazuje skutki braku ograniczenia kroku czasowego. Następny ruch jest z góry znany, każdy z graczy wykona ruch przeciwny do poprzedniego co daje w obu grach:

$$\left\{ \begin{array}{l} \Delta p_0 = (1 - \frac{1}{2} - \frac{1}{2}) = 0 \\ \Delta p_1 = (1 - \frac{1}{2} - \frac{1}{2}) = 0 \\ \Delta p_2 = (1 - \frac{1}{2} - \frac{1}{2}) = 0 \end{array} \right.$$

Brak zmian prawdopodobieństw, gracze dokonują wyboru jak poprzednio.

$$\begin{array}{cc} G_0 = P, G_1 = P, G_2 = P & G_0 = N, G_1 = N, G_2 = N \\ \left\{ \begin{array}{l} \Delta p_0 = (1 - \frac{1}{3} - \frac{1}{3}) = \frac{1}{3} \quad p_0 = \frac{1}{3} \\ \Delta p_1 = (1 - \frac{1}{3} - \frac{1}{3}) = \frac{1}{3} \quad p_1 = \frac{1}{3} \\ \Delta p_2 = (1 - \frac{1}{3} - \frac{1}{3}) = \frac{1}{3} \quad p_2 = \frac{1}{3} \end{array} \right. & \left\{ \begin{array}{l} \Delta p_0 = (1 - \frac{2}{3} - \frac{2}{3}) = -\frac{1}{3} \quad p_0 = \frac{2}{3} \\ \Delta p_1 = (1 - \frac{2}{3} - \frac{2}{3}) = -\frac{1}{3} \quad p_1 = \frac{2}{3} \\ \Delta p_2 = (1 - \frac{2}{3} - \frac{2}{3}) = -\frac{1}{3} \quad p_2 = \frac{2}{3} \end{array} \right. \end{array}$$

Nie jest to powrót do punktu wyjścia, w którym wyjściowym prawdopodobieństwem jest  $\frac{1}{3}$  dla gry przedstawionej po lewej stronie i  $\frac{2}{3}$  dla gry po prawej. W pamięci każdego z graczy jest liczba rozegranych partii ze swoimi rywalami co będzie prowadziło do niekoniecznie oczywistych zachowań. Od teraz dokonywany wybór będzie tym o większym prawdopodobieństwie.

$$\begin{array}{cc} G_0 = P, G_1 = P, G_2 = P & G_0 = N, G_1 = N, G_2 = N \\ \left\{ \begin{array}{l} \Delta p_0 = (1 - \frac{1}{4} - \frac{1}{4}) = \frac{1}{2} \quad p_0 = \frac{5}{6} \\ \Delta p_1 = (1 - \frac{1}{4} - \frac{1}{4}) = \frac{1}{2} \quad p_1 = \frac{5}{6} \\ \Delta p_2 = (1 - \frac{1}{4} - \frac{1}{4}) = \frac{1}{2} \quad p_2 = \frac{5}{6} \end{array} \right. & \left\{ \begin{array}{l} \Delta p_0 = (1 - \frac{2}{5} - \frac{2}{5}) = -\frac{1}{2} \quad p_0 = \frac{1}{6} \\ \Delta p_1 = (1 - \frac{2}{5} - \frac{2}{5}) = -\frac{1}{2} \quad p_1 = \frac{1}{6} \\ \Delta p_2 = (1 - \frac{2}{5} - \frac{2}{5}) = -\frac{1}{2} \quad p_2 = \frac{1}{6} \end{array} \right. \end{array}$$

Obserwowana jest zmiana zachowania graczy spowodowana częstością gier przeciwników. Sytuacja takiej fluktuacji może się powtarzać. Dysponując odpowiednio dużą grupą instancji gry oraz dostatecznie długą rozgrywką możliwe byłoby zaobserwowanie funkcji prawdopodobieństwa od czasu (równą dla wszystkich graczy instancji) w kształcie sinusoidy o rosnącym okresie. Sytuacja taka nie jest pożądana, co stanowi dodatkowy argument za użyciem  $\alpha < 1$  skutecznie niwelującego wystąpienie takich sytuacji.

## 2.7. Rozwiązanie stacjonarne równań standardowych

Do ustalenia punktów stałych potrzebny jest zanik dynamiki  $\Delta p_i = 0$ . Krok czasowy zostanie pominięty, gdyż nie daje wkładu do obliczeń. Należy rozwiązać układ równań, gdzie przyjęte jest  $x = p_0$ ,  $y = p_1$ ,  $z = p_2$ :

$$\begin{cases} 1 - y - z = 0 \\ 1 - x - z = 0 \\ 1 - x - y = 0 \end{cases} \Rightarrow p_0 = p_1 = p_2 = \frac{1}{2} \quad (2.7)$$

Z czego wynika że gra startująca w punkcie  $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$  nie powinna z niego wyjść. Byłoby tak gdyby prawdopodobieństwa użyte w równaniu były faktycznymi prawdopodobieństwami  $p_i$ , są one natomiast jedynie obserwacją zachowania pozostałych graczy. Są one dane jako  $\frac{nast_j}{liczba_{partii}}$ . Użycie prawdopodobieństw obserwowanych umożliwia opuszczenie punktu stałego.

## 2.8. Stabilność równań replikatorów

Aby wyznaczyć stabilność równań replikatorów należy najpierw znaleźć punkty stałe. Przyjęte zostały następujące oznaczenia  $\dot{x} = \Delta p_0$ ,  $\dot{y} = \Delta p_1$ ,  $\dot{z} = \Delta p_2$ .

$$\begin{cases} \dot{x} = 0 \\ \dot{y} = 0 \\ \dot{z} = 0 \end{cases} \Rightarrow \text{kombinacje} \begin{matrix} (0, 0, 0) & i = 0 \\ (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}) & i = 1 \\ (1, 1, 1) & i = 2 \\ (0, 1, \xi) & i = 3 \end{matrix} \text{ dają punkty stałe } (x_i^*, y_i^*, z_i^*), \text{ gdzie } \xi \in < 0, 1 >$$

Przeanalizowane zostaną 4 przypadki, które nie są symetryczne względem siebie, wprawdzie jednak należy policzyć pochodne cząstkowe.

$$\begin{aligned} \frac{\delta \dot{x}}{\delta x} &= 1 - y - z - 2x + 2xy + 2xz & \frac{\delta \dot{y}}{\delta x} &= y^2 - y \\ \frac{\delta \dot{x}}{\delta y} &= x^2 - x & \frac{\delta \dot{y}}{\delta y} &= 1 - x - z - 2y + 2xy + 2yz \\ \frac{\delta \dot{x}}{\delta z} &= x^2 - x & \frac{\delta \dot{y}}{\delta z} &= y^2 - y \end{aligned}$$

$$\begin{aligned} \frac{\delta \dot{z}}{\delta x} &= z^2 - z \\ \frac{\delta \dot{z}}{\delta y} &= z^2 - z \\ \frac{\delta \dot{z}}{\delta z} &= 1 - x - y - 2z + 2xz + 2yz \end{aligned}$$

Kolejnym krokiem jest znalezienie wartości własnych. **ZRÓB PIONOWĄ KRESKĘ PRZY =**

$$J = \begin{pmatrix} \frac{\delta \dot{x}}{\delta x} & \frac{\delta \dot{x}}{\delta y} & \frac{\delta \dot{x}}{\delta z} \\ \frac{\delta \dot{y}}{\delta x} & \frac{\delta \dot{y}}{\delta y} & \frac{\delta \dot{y}}{\delta z} \\ \frac{\delta \dot{z}}{\delta x} & \frac{\delta \dot{z}}{\delta y} & \frac{\delta \dot{z}}{\delta z} \end{pmatrix} \begin{matrix} x = x_i^* \\ y = y_i^* \\ z = z_i^* \end{matrix} = J_i \quad \left| \begin{matrix} \frac{\delta \dot{x}}{\delta x} - \lambda & \frac{\delta \dot{x}}{\delta y} & \frac{\delta \dot{x}}{\delta z} \\ \frac{\delta \dot{y}}{\delta x} & \frac{\delta \dot{y}}{\delta y} - \lambda & \frac{\delta \dot{y}}{\delta z} \\ \frac{\delta \dot{z}}{\delta x} & \frac{\delta \dot{z}}{\delta y} & \frac{\delta \dot{z}}{\delta z} - \lambda \end{matrix} \right| = J_i - \lambda I = J_{i,\lambda}$$

$$\begin{aligned}
J_{i,\lambda} = 0 \Rightarrow \quad & \lambda_0 \in \{1, 1, 1\} \\
& \lambda_1 \in \{-\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\} \\
& \lambda_2 \in \{1, 1, 1\} \\
& \lambda_3 \in \{0, -z, z-1\}
\end{aligned}$$

Warunkiem na stabilność punktu jest  $Re\lambda_i < 0$ , co eliminuje nam wszystkie lambdy poza  $\lambda_3$ . Pokazuje to że punkty o kombinacji  $(0, 1, \xi)$  są marginalnie stabilne. Ponieważ analiza matematyczna nie przyniosła jednoznacznej odpowiedzi co do stabilności, można spróbować przeanalizować równanie  $\dot{z}$ , dla sytuacji gdy  $p_i$  dwóch z graczy dojdą do granicy prawdopodobieństwa. W pierwszym wierszu tabelki 2.1 widać, że  $x$  dąży do zawarcia sojuszu z  $z$ , natomiast  $y$  gra aby osiągnąć sojusz z  $x$ . W tej sytuacji prawdopodobieństwo strategii  $z$  będzie rosło do zawiązania koalicji z  $x$ . W kolejnych dwóch wierszach są stany ustalone. W wierszu drugim zmiana  $\dot{z}$  nie ma znaczenia gdyż obaj przeciwnicy są w wyłącznej koalicji z  $z$ . Wiersz trzeci pokazuje przeciwną sytuację, w której jakakolwiek zmiana  $\dot{z}$  nie wniesie nic do gry ze względu na trwały sojusz między  $x$  a  $y$ . Ostatni wiersz pokazuje przypadek symetryczny do pierwszego, z tym że jedynym możliwym sojusznikiem dla  $z$  jest  $y$ . Oczywiście omawiany teraz przykład do osiągnięcia stabilności wymagałby braku zmian decyzji pozostałych graczy. Musieli by oni znać realne prawdopodobieństwo przeciwników.

Tablica 2.1: Stabilność na krawędzi sześciangu

x	y	1-x-y	$\dot{z}$
0	0	1	$z \cdot (1 - z)$
0	1	0	0
1	0	0	0
1	1	-1	$-z \cdot (1 - z)$

## 3. Implementacja symulacji

W tym rozdziale chciałbym przedstawić technologie i narzędzia użyte do wykonania symulacji oraz sposoby ich uruchomienia.

### 3.1. Środowisko QT

**CHECK** Zdecydowałem się wykorzystać QT Creator IDE z kilku powodów, które zamierzam zaraz rozwinąć. Najważniejszą cechą środowiska jest udostępnienie go na kilku rodzajach licencji. Osobiście użyłem licencji LGPL, która pozwoliła mi bez ponoszenia kosztów korzystać ze środowiska. Kolejnym ważnym elementem jest multiplatformowość pozwalająca w łatwy sposób przenosić kod program między systemami operacyjnymi, o ile nie zostały użyte biblioteki dostępne tylko na jeden z systemów. Kolejną z zalet jest łatwy i intuicyjny interfejs tworzenia graficznego interfejsu użytkownika, osoba mająca wcześniej styczność z chociażby biblioteką Swing Java'y nie powinna mieć problemu z zaadaptowaniem się do formularza QT Creatora. Wykorzystywany jest model sygnałów i slotów, polegający na emitowaniu sygnału przez zdarzenie, który następnie trafia do podłączonego slotu. Jest to w stanie znacznie ułatwić komunikację między elementami. Używanie nowoczesnego języka C++ (ja używałem wersji 14) nie sprawia problemów, lecz powinniśmy być świadomi że przykładowe uruchomienie wątków w aplikacji powinno być zrobione przy użyciu klas i funkcji z biblioteki QT.

### 3.2. GLWidget

### 3.3. Schemat programu

### 3.4. Kod

W tym podrozdziale będą opisane kluczowe fragmenty kodu, niezbędne do zrozumienia implementacji programu. W kodzie będą używane oznaczenia:**ZASTANÓW SIĘ**

```
1 template<typename T> using tup3= tuple<T,T,T>;  
2 vector<vector<tup3<double>>> beginsp; //wektor punktów początkowych  
   odcinków  
3 vector<vector<tup3<double>>> endsp; //wektor punktów końcowych odcink  
   ów
```

```

4 vector<tup3<int>>>colorsp;
5 mutex points;

```

**Interfejs klasy *Game*** Cała symulacja przeprowadzana jest w klasie *Game*, dlatego chciałbym ją teraz omówić. Reprezentuje ona jedną instancję gry 3-osobowej. Parametrem konstruktora jest indeks tablicy *decision\_funs*, której element zostaje przypisany do *decision.function* jest wrapperem biblioteki standardowej dla dowolnej funkcji, wyrażenia lambda, wyrażenia bind, funkcjonału lub wskaźników do funkcji. W poniższym kodzie *function* jest parametryzowany bezargumentowym, bezwynikowym zachowaniem. Funkcja *next* przeprowadza rozgrywkę jednej partii. Jej implementacja zostanie przedstawiona później. Zmienna *current* odpowiada  $liczba_{partii}$ , natomiast  $nr[i]$  odpowiada  $nast_i$  (opisano w podrozdziale 2.2). Funkcja *checker* jest odpowiednikiem funkcji *ogr* (opisano w podrozdziale 2.5). Tablica *decision\_funs* zawiera funkcje lambda mające dostęp do prywatnych zmiennych klasy *Game*. W tablicach *result* znajdują się obliczone  $\Delta p_i$ , które następnie poddawane są funkcji ograniczającej.

```

1 class Game{
2 public:
3     Game(int f);
4     tup3<double> next();
5     int getCurrent(); //zwraca numer bieżącej partii
6     tup3<double> prelast(); //zwraca pre
7 private:
8     array<double,3> p; //prawdopodobieństwa
9     tup3<double> pre; //ostatnia krotka prawdopodobieństw na
        podstawie, których było dokonane losowanie sojuszników
10    int current; //numer bieżącej partii
11    array<int,3> nr;
12    double checker(double r);
13    function<void()> decision; //dokonuje modyfikacji prawdopodobień
        stw
14    array<function<void()>,2> decision_funs={{
15        [this]() { //standardowe
16            array<double,3> result= {
17                0.1*( 1 - static_cast<double>(nr[1])/current -
                    static_cast<double>(nr[2])/current),
18                0.1*( 1 - static_cast<double>(nr[2])/current -
                    static_cast<double>(nr[0])/current),
19                0.1*( 1 - static_cast<double>(nr[0])/current -
                    static_cast<double>(nr[1])/current)
20            };
21            for(int i=0; i<3; i++)

```

```

22         p[i]= checker(p[i]+result[i]);
23     },
24     [this]() { // replikatorów
25         array<double,3> result= {
26             0.1*(p[0]*(1-p[0])*(1-static_cast<double>(nr[1])/current-
                static_cast<double>(nr[2])/current)),
27             0.1*(p[1]*(1-p[1])*(1-static_cast<double>(nr[0])/current-
                static_cast<double>(nr[2])/current)),
28             0.1*(p[2]*(1-p[2])*(1-static_cast<double>(nr[0])/current-
                static_cast<double>(nr[1])/current))
29         };
30         for(int i=0; i<3; i++)
31             p[i]= checker(p[i]+result[i]);
32     }
33     }
34 };

```

**Implementacja *Game::next*** Tablica *choices* zawiera losowe wartości dla każdego z zawodników, na podstawie których dokonywane jest losowanie sojusznika.

```

1  tup3<double> Game::next() {
2      pre= make_tuple(p[0],p[1],p[2]);
3      current++;
4      array<double,3> choices= { static_cast<double>(rand())/RAND_MAX,
                static_cast<double>(rand())/RAND_MAX, static_cast<double>(rand
                ())/RAND_MAX};
5      for(int i=0; i<3; i++)
6          if(choices[i]<p[i])
7              nr[i]++;
8      decision();
9      return make_tuple(p[0],p[1],p[2]);
10 }

```

**Implementacja wątku symulującego** *QtConcurrent::run* jest funkcją uruchamiającą w osobnym wątku funkcję podaną jako parametr. Zwraca ona obiekt *QFuture*, który nie może przerwać lub wstrzymać wątku. Poprzez *beginasp[i].resize(p)* tworzony jest wektor posiadający p pustych elementów (wektorów krotek), ponieważ już w pierwszej partii będziemy potrzebować osobnego wektora dla każdej instancji gry. Natomiast *beginasp[i].reserve(g)* rezerwuje pamięć dla g partii, gdyby ta funkcja nie została użyta prawdopodobnie wystąpiłaby realokacja pamięci, w celu zapewnienia jej ciągłości. Przekładałoby się to na spowolnienie działania programu. Napisałem prawdopodobnie, ponieważ



standard języka nie precyzuje ilości pamięci domyślnie rezerwowanej przez wektor, co jest zostawione po stronie implementacji. Przed wykonaniem każdej partii wątek czeka na dostęp do zasobów. Funkcja `try_lock()` zwróci `true` w przypadku udanego przydzielenia zasobów, w przeciwnym razie zwróci `false`.

```

1  QtConcurrent::run(
2      [&]()->void{
3          ui->pushButton_Run->setEnabled( false );
4          const int g= nr_rounds;
5          const int p= nr_players;
6          const int f= fun;
7          vector<unique_ptr<Game>> tab;
8          std::default_random_engine generator;
9          std::uniform_int_distribution<int> distribution
              (0,255);
10         auto r= bind(distribution, generator);
11         clear_vectors(); //czyści beginsp, endsp, colorsp
12         chrono::milliseconds d(delay);
13         beginsp.resize(p);
14         endsp.resize(p);
15         for(int i=0; i<p;i++){
16             tab.push_back(make_unique<Game>(f));
17             colorsp.push_back(make_tuple( r(), r(), r() ));
18             beginsp[i].reserve(g);
19             endsp[i].reserve(g);
20         }
21         for(int j=0; j<g;j++){
22             while(! points.try_lock()); // próba blokady dostę
                pu do zasobów
23             for(int i=0; i<p;i++){
24                 beginsp[i].push_back(tab[i]->next());
25                 endsp[i].push_back(tab[i]->prelast());
26             }
27             points.unlock(); //zwolnienie dostępu do zasobów
28             emit copy();
29             this_thread::sleep_for(d);
30         }
31         ui->pushButton_Run->setEnabled( true );
32     });

```

## 3.5. Rysowanie 3D

## 3.6. Makefile

Symulując grę w okręgu postanowiłem rysować wykresy funkcji prawdopodobieństwa od numeru partii. Do tego celu uznałem, że najbardziej odpowiedni będzie plik *Makefile*, który wykona kompilację, uruchomienie oraz narysowanie wykresu przy pomocy programu gnuplot. Aby uruchomić program należy podać argument: G - ilość partii do rozegrania oraz P - ilość graczy. Poniżej przykładowe polecenie do wykonania symulacji dla 100 partii rozegranych przez 20 zawodników.

```
make G=100 P=20
```

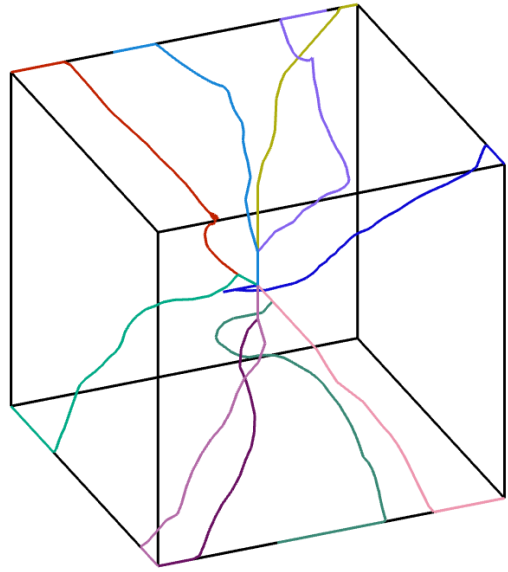
## 4. Wyniki

### 4.1. Gry 3-osobowe

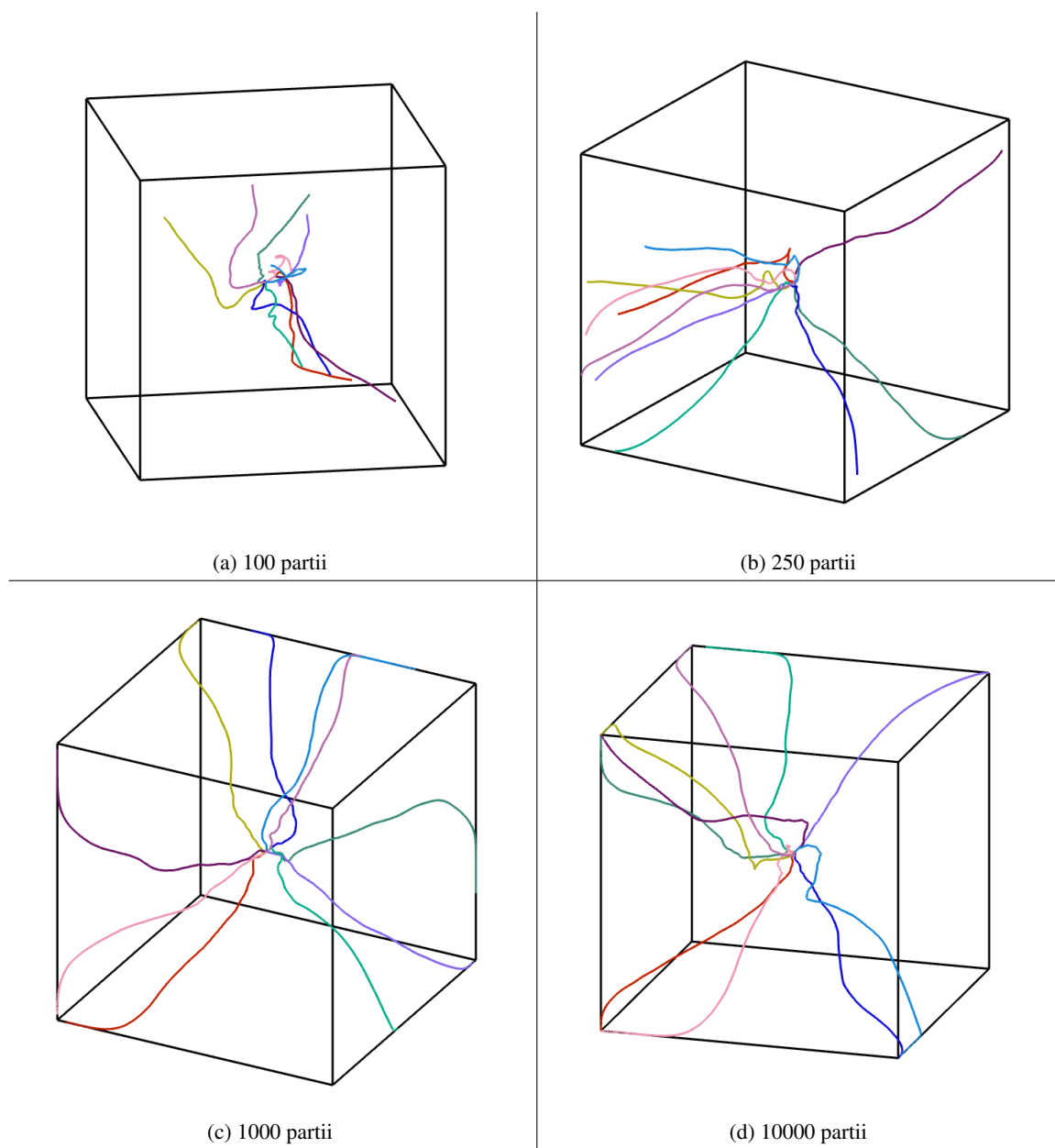
**Równania standardowe** Analiza teoretyczna równań wskazała punkt  $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$  jako punkt stały. Wynika to z założenia gry o pełnej informacji. Nie jest tak dla symulacji, której funkcje opuszczają centrum sześcianu. Osiągnięcie przez funkcję krawędzi sześcianu oznacza zawiązanie koalicji, po czym funkcja przesuwa się po krawędzi dążąc do wierzchołka. Odnosząc tabelkę 2.1 do równań standardowych należy stwierdzić, że funkcja powinna osiągać punkt stabilności na krawędzi i nie poruszać się dalej. Byłoby tak gdyby błąd szacowania prawdopodobieństw przeciwników był odpowiednio mały. Szacowanie prawdopodobieństw dane jest przez  $\frac{nast_j}{liczba_{partii}}$ . Wynika z tego, że wystarczy kilka zagrań przeciwników niezgodnych z zawartą koalicją, aby gracz szacujący ich prawdopodobieństwa nie mógł stwierdzić, że ich realne prawdopodobieństwa wynoszą 0 lub 1. Pokazuje to rysunek 4.1, na którym widać funkcje o wielu punktach przegięcia. Szybkość dążenia do wierzchołków będzie malała wraz z czasem spędzonym na krawędzi, gdyż

$$\lim_{nast_j \rightarrow liczba_{partii} \wedge liczba_{partii} \rightarrow \infty} \frac{nast_j}{liczba_{partii}} \in \{0, 1\}$$

**Równania replikatorów** Równania replikatorów charakteryzują się mniejszą dynamiką gry w stosunku do równań standardowych. Przyczyną jest człon  $x(1 - x)$ , którego maksimum wynosi 0.25. Daje to czterokrotnie mniejsze  $\Delta p$  w punkcie  $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ . Na krańcach dziedziny  $< 0, 1 >$  wspomniany człon szybko dąży do 0, co znacznie opóźnia osiągnięcie koalicji. Widać to porównując rysunki 4.1 oraz 4.2a. Natomiast rysunki 4.1 oraz 4.2d pokazują liczbę partii potrzebną do osiągnięcia podobnych miejsc w przestrzeni sześcianu. Gra używająca równań replikatorów musiała wykonać ich 100 razy więcej niż gra używająca równań standardowych.

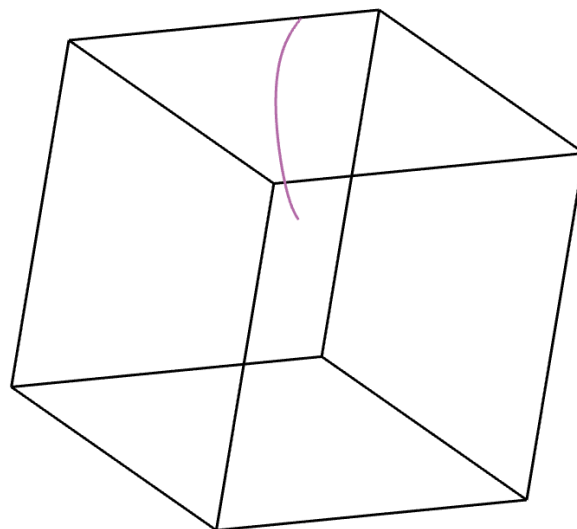


Rysunek 4.1: Równania standardowe: 100 partii, 10 instancji



Rysunek 4.2: Równania replikatorów: 10 instancji

Zaś analiza stabilności pokazała, że każda krawędź sześcianu powinna być linią punktów stabilnych, co obrazuje rysunek 4.3. Natomiast symulacja przedstawiona na rysunku 4.2c pokazuje przemieszczanie się funkcji ku wierzchołkom. Powodem tego jest błąd gracza w szacowaniu prawdopodobieństwa przeciwników, wynikający z gry o niepełnej informacji. Nie występuje on jednak we wszystkich instancjach gry, na przykład w jasnozielonej rozgrywce 4.2c widać, że funkcja nieznacznie porusza się po krawędzi, co świadczy o mniejszym błędzie szacowania prawdopodobieństwa w stosunku do innych rozgrywek. Analiza stabilności wskazuje, że punkty  $(0, 0, 0)$  oraz  $(1, 1, 1)$  nie są stabilne, co wynika z tego iż żadna z symulacji do nich nie zmierza. Co więcej pierwszy i ostatni wiersz tabelki 2.1 nie może wystąpić w symulacji. Osiągnięcie punktu  $(0, 0, \xi)$  lub  $(1, 1, \xi)$ , gdzie  $\xi$  nie jest w pobliżu 0 lub 1, nie jest możliwe w symulacji. Oznaczałoby to zawiązanie sojuszu pomiędzy graczami, których zagrania nie mają na celu zawiązania między nimi koalicji. Potwierdzają to symulacje z których żadna nie dochodzi do krawędzi wychodzących z punktów  $(0, 0, 0)$  oraz  $(1, 1, 1)$ .

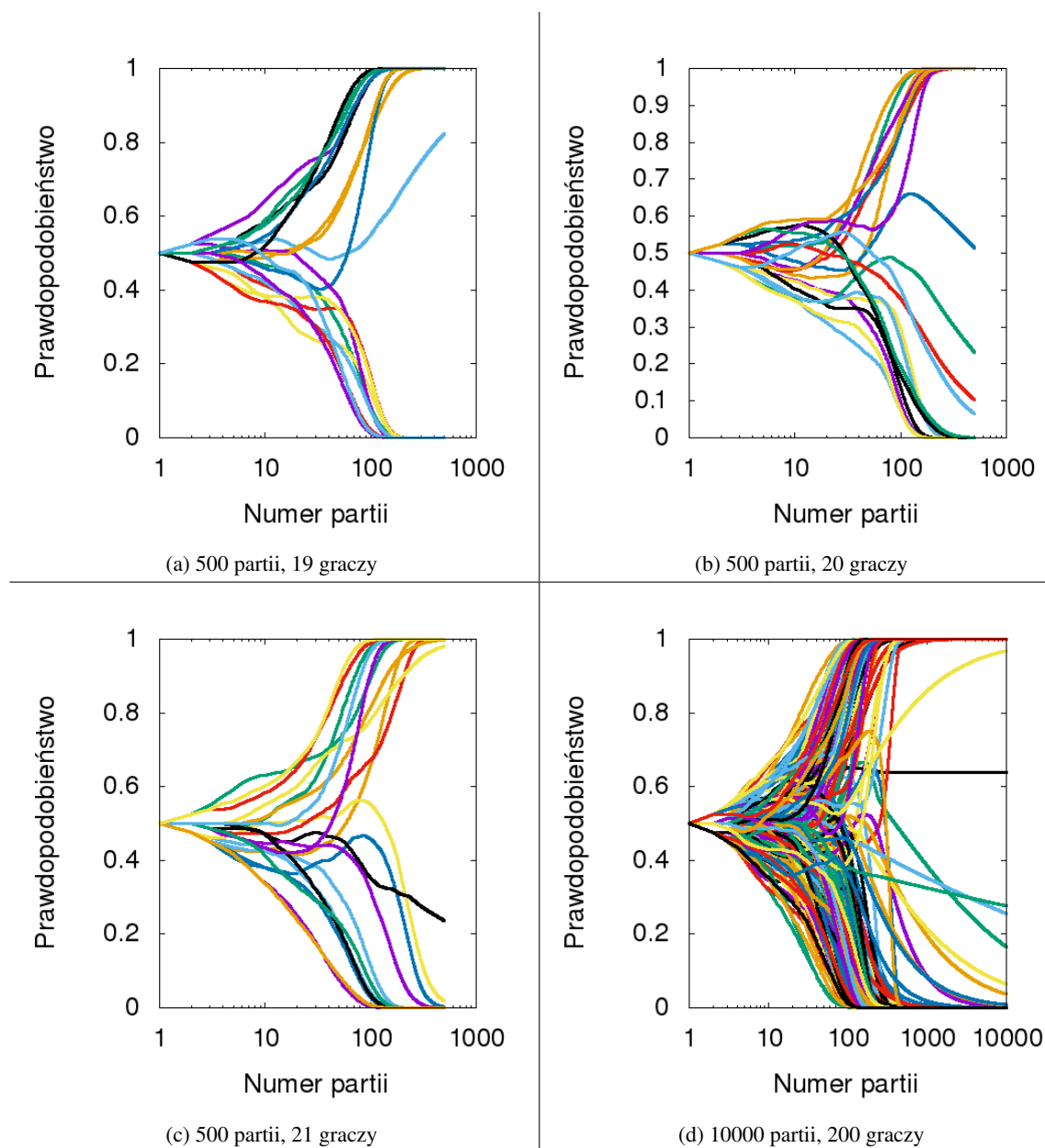


Rysunek 4.3: Równania replikatorów, gra o pełnej informacji: 300 partii, 10 instancji

## 4.2. Gry N-osobowe

W grze w której ustawionych w okręgu jest  $N$  graczy, spodziewamy się, że istotnym czynnikiem będzie parzystość ich liczby. Teoretyczna liczba graczy mogących nie znaleźć koalicjanta wynosi  $< N \bmod 2, \lfloor \frac{N}{3} \rfloor >$ . W grze o parzystej liczbie graczy zawsze istnieje rozwiązanie, które gwarantuje każdemu graczowi przynależność do koalicji. Natomiast w grach o nieparzystej liczbie graczy musi istnieć co najmniej jeden zawodnik, który nie zawrze sojuszu (posiadanie prawdopodobieństwa 0 lub 1, nie jest równoznaczne z byciem w sojuszu, co pokazały symulacje dla 3-graczy, kiedy funkcje dążyły do wierzchołków). Maksymalna liczba graczy bez sojuszu może być maksymalnie połową liczby graczy będących w sojuszu (maksymalnie co trzeci gracz może być bez sojuszu z zaokrągleniem w dół). Gdyby była większa oznaczałoby to, że istnieją pary sąsiadujących graczy, którzy nie są w żadnej koalicji. Taka sytuacja nie może mieć miejsca, ponieważ omawiane pary stałyby się koalicjami.

W przypadku występowania dużej ilości zawodników łatwiej można zaobserwować sytuacje, w których sąsiedzi zawodnika stosują w przewadze jedną taktykę. Co prowadzi do dokładnego szacowania ich prawdopodobieństw. Gracz wtedy nie dokonuje zmian w swoim zachowaniu, co wynika z tabelki 2.1. Przypadek taki można zaobserwować również na rysunku 4.4d, oznaczony kolorem czarnym.



Rysunek 4.4: Gry N-osobowe

## 5. Podsumowanie

Celem pracy było przeprowadzenie symulacji koalicji mieszanych przy użyciu ewolucyjnej teorii gier. Oczekiwane wyniki na podstawie analizy matematycznej nie odpowiadały ściśle wynikom uzyskanym z symulacji. Powodem tego było założenie gry o pełnej informacji, gdzie symulacja zakładała grę o niepełnej informacji. W celu zmniejszenia błędu szacowanego prawdopodobieństwa, można by uwzględniać do niego jedynie  $k$ -ostatnich gier. Spowodowałoby to trafne szacowanie prawdopodobieństwa podczas monotonicznej gry przeciwników, czego skutkiem byłyby punkty stabilne na krawędziach po przeprowadzeniu na nich  $k$  gier. Rodzi to jednak problem natury doboru  $k$ . Zgodnie z analizą funkcje dążyły do krawędzi sześcianu, omijając punkty niestabilne oraz krawędzie reprezentujące niemożliwe do zawiązania koalicje.

Symulacje gier  $N$ -osobowych pokazały dążenie graczy do trwałych koalicji. Wartą przeanalizowania byłaby sytuacja losowej zmiany prawdopodobieństw części graczy w stanie ustalonym. Powinno to z czasem doprowadzić do ustalenia nowego stanu ustalonego.

## Bibliografia

- [Crf] C++ reference. <http://en.cppreference.com/>. 2017-12-26.
- [Fsm] Finite state machine designer. <http://madebyevan.com/fsm/>. 2017-12-20.
- [HS98] J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge, 1998.
- [Now06] M. A. Nowak. *Evolutionary dynamics: exploring the equations of life*. The Belknap press of Harvard university press, Cambridge, Massachusetts and London, England, 2006.
- [P.01] Straffin P. *Teoria gier*. Wydawnictwo Naukowe SCHOLAR, Warszawa, 2001.
- [Qt] Qt main site. <http://doc.qt.io/>. 2017-09-01.
- [Sza] Msc/eng thesis template of university of science and technology in krakow (agh). <https://www.sharelatex.com/templates/thesis/agh>. 2017-09-01.
- [Tut] Qt5 tutorial opengl with qglwidget - 2017. [http://www.bogotobogo.com/Qt/Qt5\\_OpenGL\\_QGLWidget.php](http://www.bogotobogo.com/Qt/Qt5_OpenGL_QGLWidget.php). 2017-09-01.