

Physik der sozio-ökonomischen Systeme *mit dem Computer*

*JOHANN WOLFGANG GOETHE UNIVERSITÄT
18.11.2022*

MATTHIAS HANAUSKE

*FRANKFURT INSTITUTE FOR ADVANCED STUDIES
JOHANN WOLFGANG GOETHE UNIVERSITÄT
INSTITUT FÜR THEORETISCHE PHYSIK
ARBEITSGRUPPE RELATIVISTISCHE ASTROPHYSIK
D-60438 FRANKFURT AM MAIN
GERMANY*

5. Vorlesung

In diesem Semester (WS 2022/23)
findet die Vorlesung und die Übungstermine
in Präsenz (und auch Online) statt.

Plan für die heutige Vorlesung

- Kurze Wiederholung der Inhalte der 4. Vorlesung
- Numerisches Lösen von Differentialgleichungen
- Einführung in die evolutionäre Spieltheorie
 - Die Differentialgleichung eines evolutionären, symmetrischen (2×3) -Spiels
 - Die 19 Zeeman – Klassen
- Anwendungsfelder Spieltheorie
 - Anwendungsfelder in den Wirtschafts- Sozialwissenschaften und Biologie
 - Experimentelle Ökonomie
 - Die Finanzkrise als Falke-Taube Spiel
 - Die Entstehung einer dritten Strategie im Elfmeter-Spiel (Nesken Effekt)
 - Evolutionäre Entwicklung einer Eidechsen Population als symmetrisches (2×3) -Spiel
 - Das Räuber-Beute Spiel und die Lotka-Volterra-Gleichung
 - Die Klimakrise als Populationsdilemma

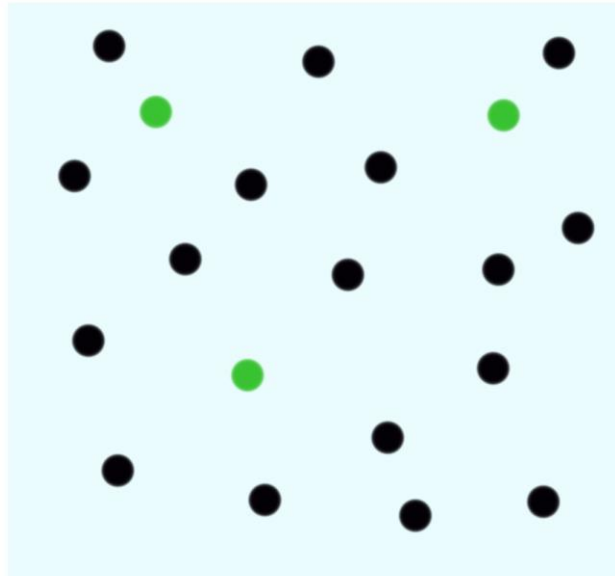
Inhalte Vorlesung 4

- Einführung in die Evolutionäre Spieltheorie
 - Die Differentialgleichung eines evolutionären, symmetrischen (2x2)-Spiels
 - Dominante Spiele
 - Koordinationsspiele
 - Anti-Koordinationsspiele
 - Das System von Differentialgleichung eines evolutionären, unsymmetrischen (2x2)-Spiels (Bi-Matrix Spiele)
 - Eckenspiele (Corner Class Games)
 - Sattelpunktspiele (Saddle Class Games)
 - Zentrumspele (Center Class Games)

Evolutionäre Spieltheorie

Symmetrische (2x2)-Spiele einer Population

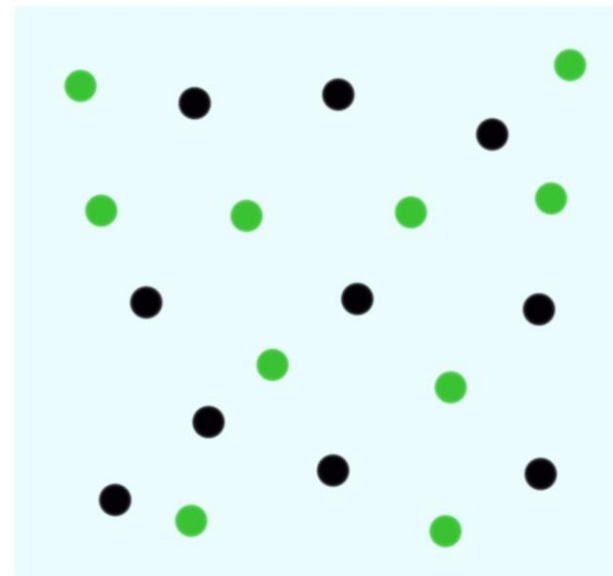
Die evolutionäre Spieltheorie betrachtet die zeitliche Entwicklung des strategischen Verhaltens einer gesamten Spielerpopulation.



$$x(0)=0.15$$



zeitliche
Entwicklung
der
Population



$$x(10)=0.5$$

Mögliche Strategien: (grün , schwarz), Parameter t stellt die „Zeit“ dar.
 $x(t)$: Anteil der Spieler, die im Zeitpunkt t die Strategie „grün“ spielen.

Nimmt man zusätzlich ein symmetrisches Spiel an ($\hat{\$} := \hat{\$}^A = \left(\hat{\$}^B\right)^T$), in welchem die Auszahlungswerte (Fitness-Werte) der Populationsgruppen gleich sind, so kann man die beiden Gruppen von ihrer mathematischen Struktur her als ununterscheidbare Spielergruppen mit identischen Populationsvektoren $x(t) = y(t)$ annehmen. Die Differentialgleichung schreibt sich dann wie folgt:

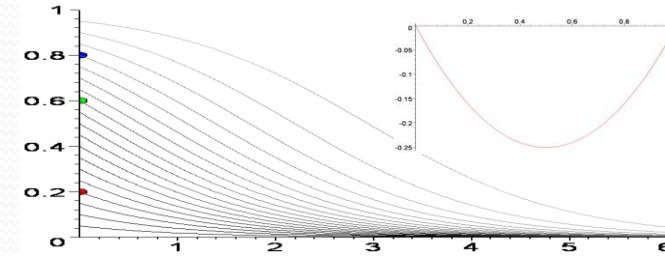
$$\frac{dx(t)}{dt} = [(\$_{11} - \$_{21})(x - x^2) + (\$_{12} - \$_{22})(1 - 2x + x^2)] x(t) =: g(x) \quad (3)$$

Verallgemeinert man diese Differentialgleichung wieder auf mehr als zwei Strategien, so kann man abkürzend die folgende Formulierung schreiben:

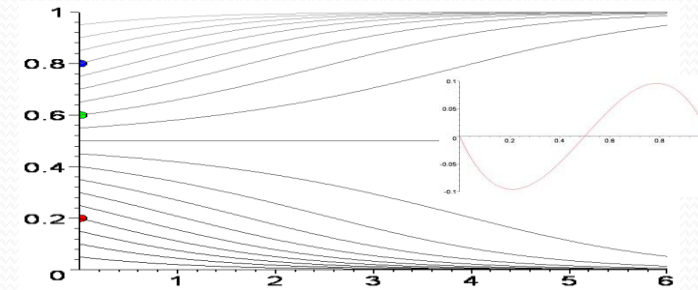
$$\frac{d\vec{x}}{dt} = \hat{\mathbf{x}} \left(\hat{\$} \vec{x} \right) - \left(\left(\hat{\$} \vec{x} \right)^T \vec{x} \right) \vec{x}$$

Klassifizierung von evolutionären, symmetrischen (2x2)-Spielen

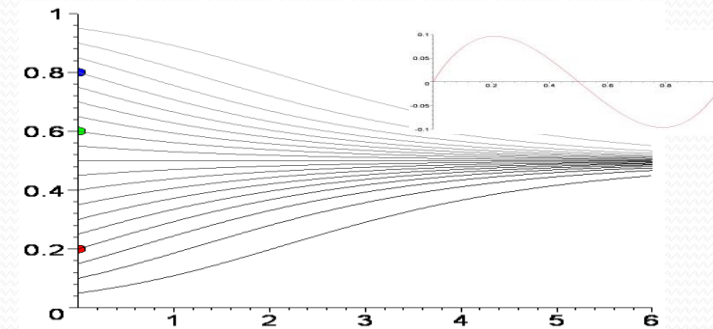
- **Dominante Spiele**
(2. Strategie dominiert 1.Strategie)
Es existiert ein Nash - Gleichgewicht, welches die anderen Strategien dominiert. ESS bei $x=0$.



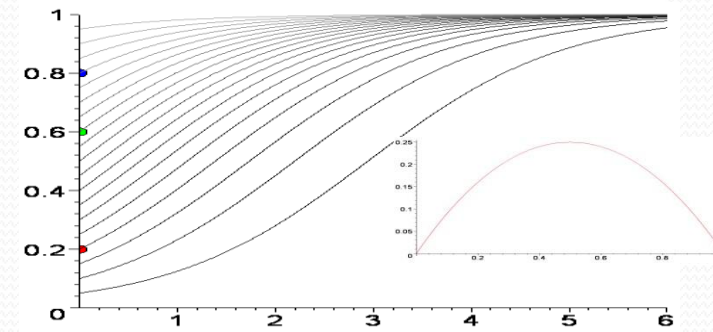
- **Koordinationsspiele**
Es existieren drei Nash - Gleichgewichte und zwei reine ESS, die abhängig von der Anfangsbedingung realisiert werden.



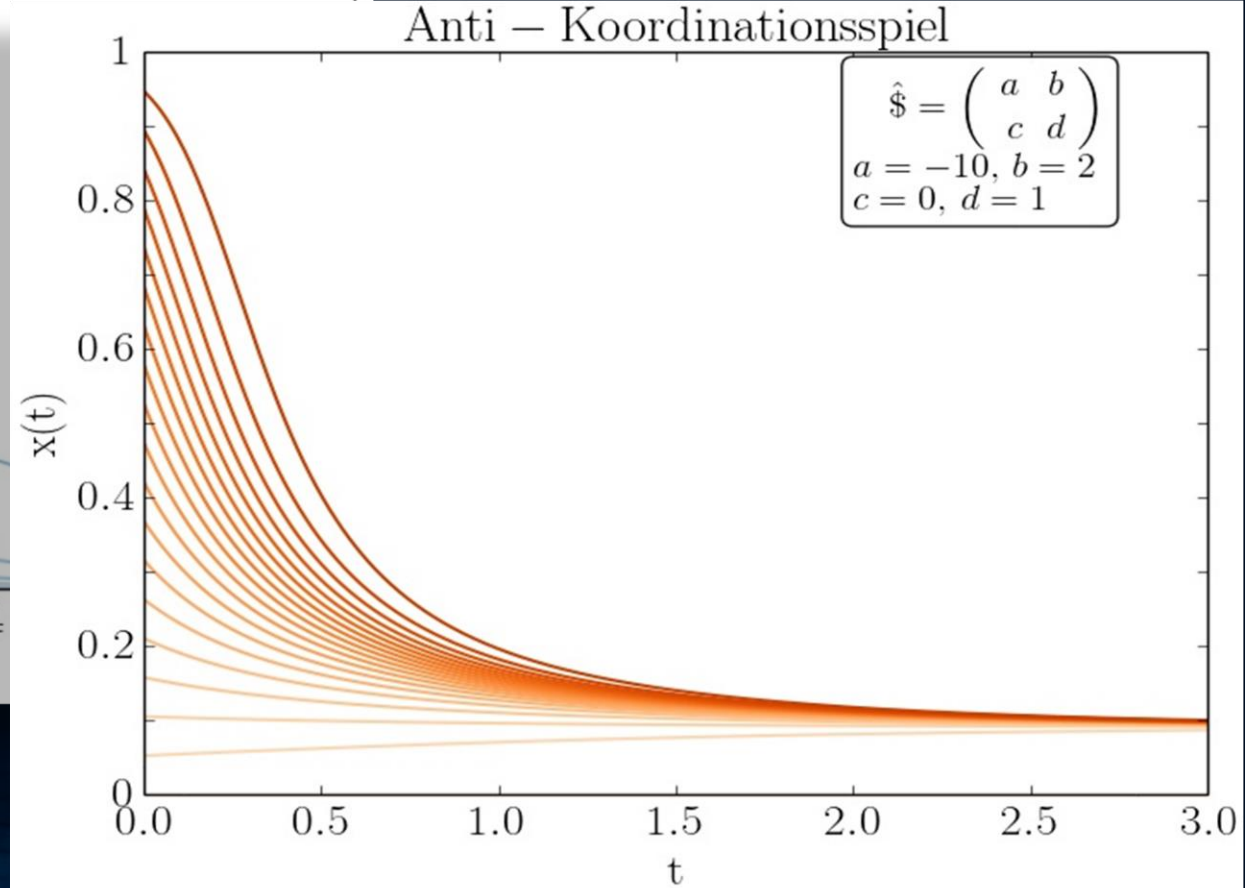
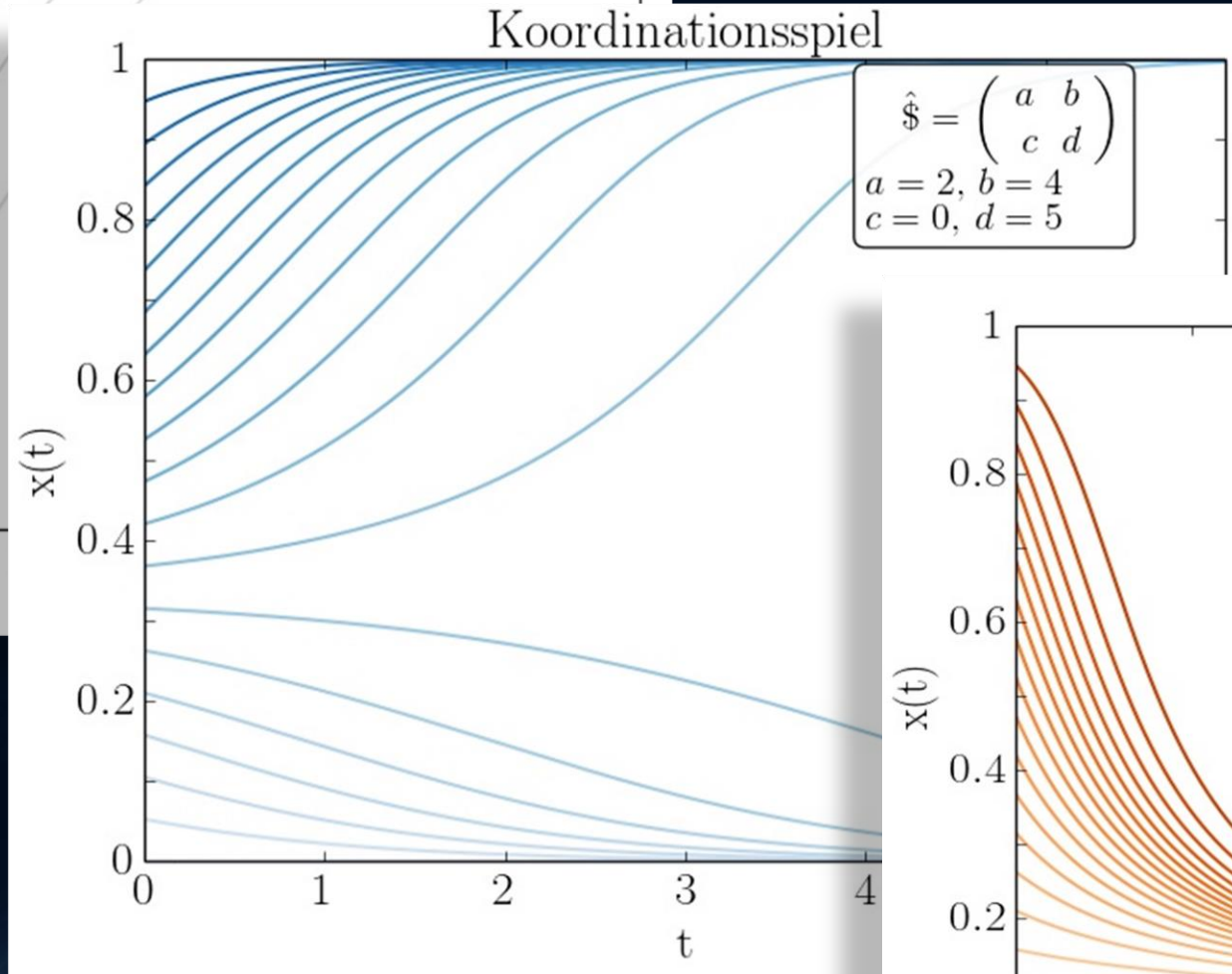
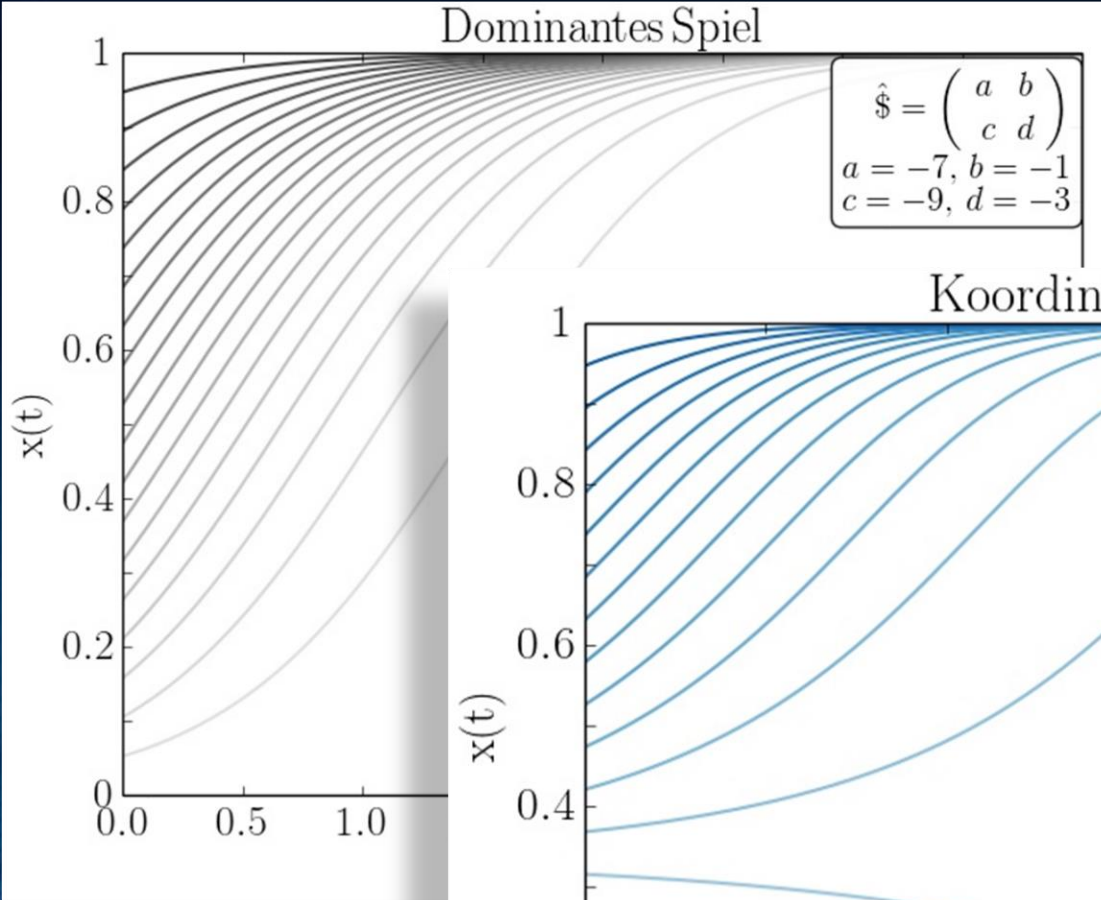
- **Anti - Koordinationsspiele**
Es existieren drei Nash - Gleichgewichte aber nur eine gemischte ESS, die unabhängig von der Anfangsbedingung realisiert wird.



- **Dominante Spiele**
(1. Strategie dominiert 2.Strategie)
Es existiert ein Nash - Gleichgewicht, welches die anderen Strategien dominiert. ESS bei $x=1$.



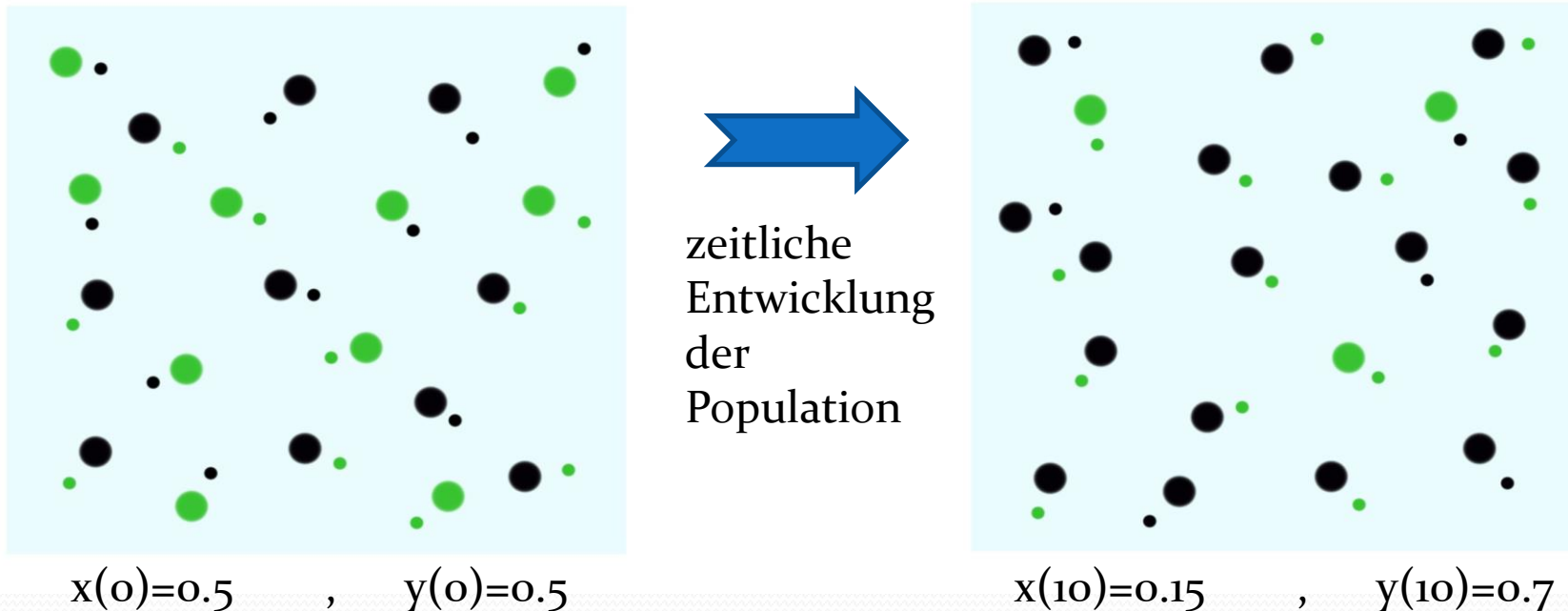
Lösen des evolutionären Spiels mit Python Version evol1.py



Evolutionäre Spieltheorie

Unsymmetrische (2x2)-Spiele (Bimatrixspiele) zweier Populationen

Bei unsymmetrischen (2x2)-Spiele besteht die zugrundeliegende Population aus zwei Gruppen (hier große und kleine Kreise). Aufgrund der unterschiedlichen Auszahlungsmatrizen können die Populationsgruppen sich in ihren Strategieentscheidungen (**grün**, schwarz) unterschiedlich entwickeln.



Mögliche Strategien: (**grün**, schwarz), Parameter t stellt die „Zeit“ dar.

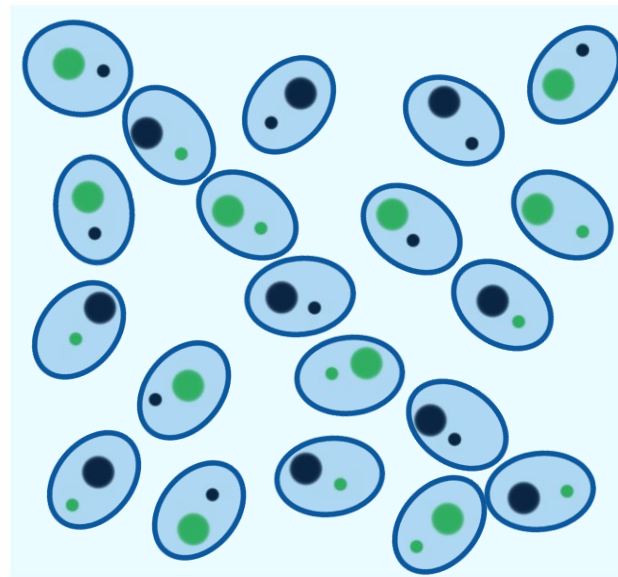
$x(t)$: Anteil der großen Spieler, die im Zeitpunkt t die Strategie „**grün**“ spielen.

$y(t)$: Anteil der kleinen Spieler, die im Zeitpunkt t die Strategie „**grün**“ spielen.

Evolutionäre Spieltheorie

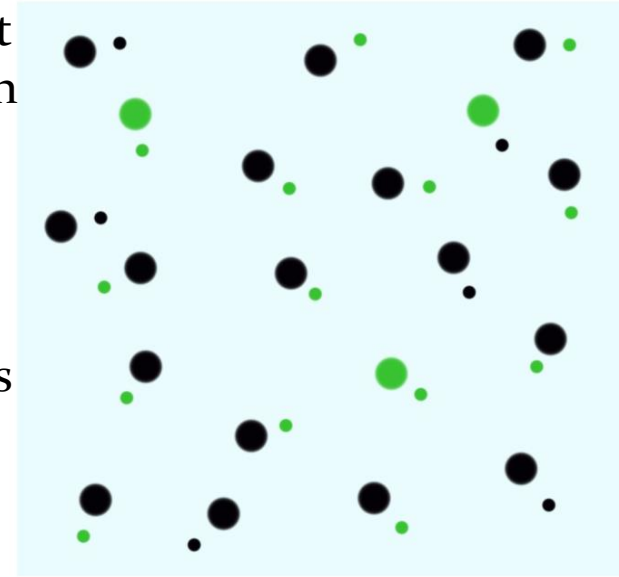
Unsymmetrische (2x2)-Spiele (Bimatrixspiele)

Die einzelnen Akteure innerhalb der betrachteten gesamten Population spielen ein andauernd sich wiederholendes Spiel miteinander, wobei sich jeweils zwei Spieler mit unterschiedlichen Gruppenzugehörigkeiten zufällig treffen, das Spiel spielen und danach zu dem nächsten Spielpartner der anderen Gruppe wechseln .



$x(10)=0.5$, $y(10)=0.5$

Die Anfangspopulation von Spielern spielt zum Zeitpunkt $t=0$ das erste Mal das Spiel. Es bilden sich stets Zweier-Gruppen aus großen und kleinen Kreisen.



$x(10)=0.15$, $y(10)=0.7$

Das evolutionäre Spiel schreitet voran und die grüne Strategie wird für die kleinen Spieler zunehmend attraktiver ($y(10)=0.7$), wohingegen sie für die großen Spieler zunehmend weniger attraktiv wird ($x(10)=0.15$).

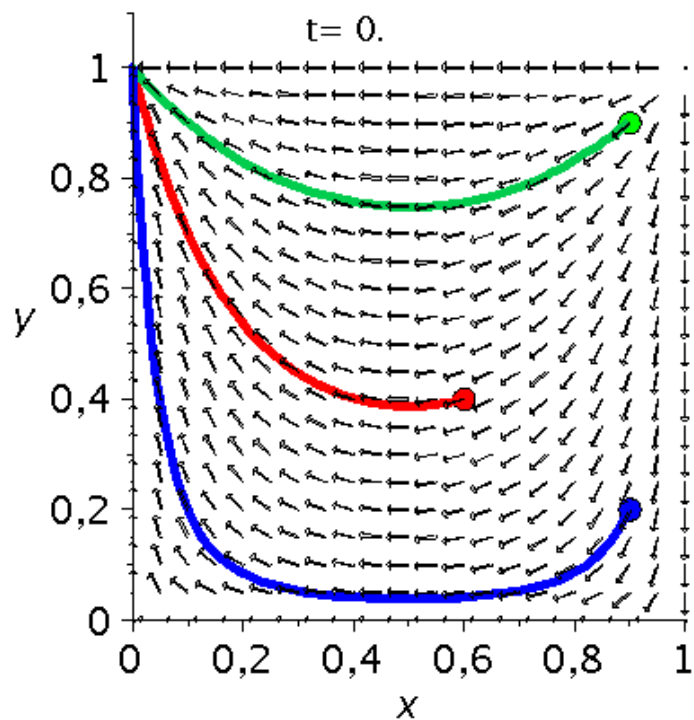
Wir beschränken uns im folgenden auf den 2-Strategien Fall ($m_A = m_B = 2$), lassen jedoch weiter eine Unsymmetrie der Auszahlungsmatrix zu. Die beiden Komponenten der zweidimensionalen gruppenspezifischen Populationsvektoren lassen sich dann, aufgrund ihrer Normalisierungsbedingung, auf eine Komponente reduzieren ($x_2^A = 1 - x_1^A$ und $x_2^B = 1 - x_1^B$). Das zeitliche Verhalten der Komponenten der Populationsvektoren (Gruppe A: $x(t) := x_1^A(t)$ und Gruppe B: $y(t) := x_1^B(t)$) wird in der Reproduktionsdynamik mittels des folgenden Systems von Differentialgleichungen beschrieben (siehe z.B. [4], S:116 oder [3], S:69):

$$\begin{aligned} \frac{dx(t)}{dt} &= [(\beta_{11}^A + \beta_{22}^A - \beta_{12}^A - \beta_{21}^A) y(t) + (\beta_{12}^A - \beta_{22}^A)] (x(t) - (x(t))^2) =: g_A(x, y) \\ \frac{dy(t)}{dt} &= [(\beta_{11}^B + \beta_{22}^B - \beta_{12}^B - \beta_{21}^B) x(t) + (\beta_{12}^B - \beta_{22}^B)] (y(t) - (y(t))^2) =: g_B(x, y) \end{aligned} \quad (2)$$

Klassifizierung von Bi-Matrix Spielen

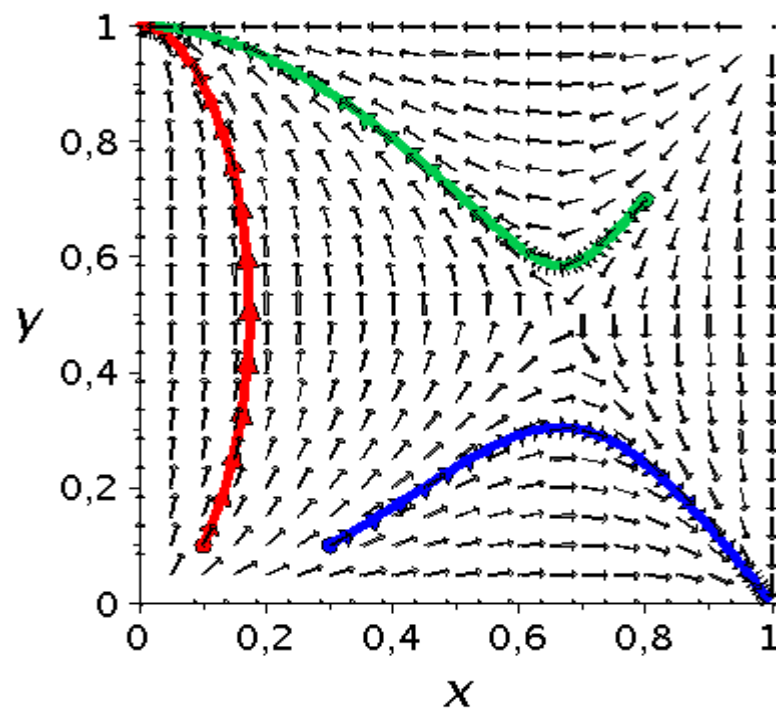
Eckspiele

Die Spielklasse der Gruppe A
oder der Gruppe B ist ein
Dominantes Spiel



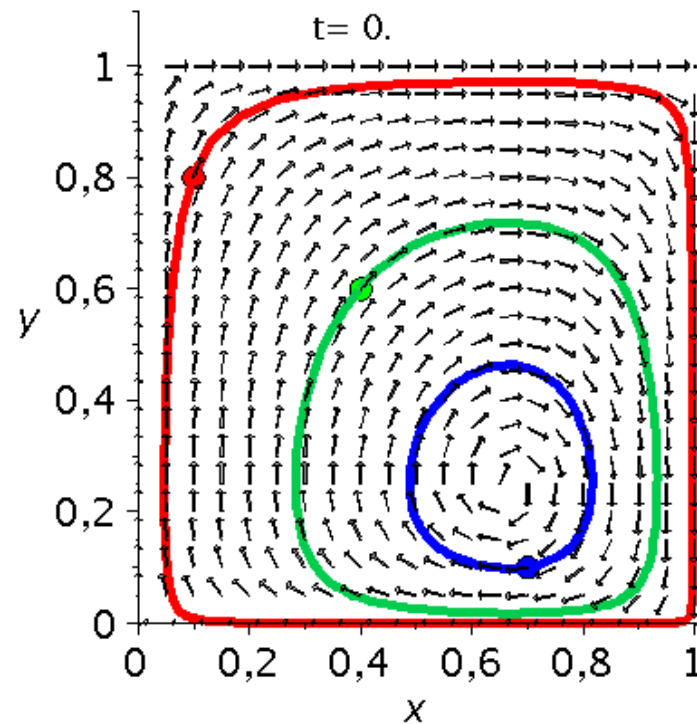
Sattelspiele

Spiel A: Koordinationsspiel
Spiel B: Koordinationsspiel
oder
Spiel A: Anti-Koordinationsspiel
Spiel B: Anti-Koordinationsspiel



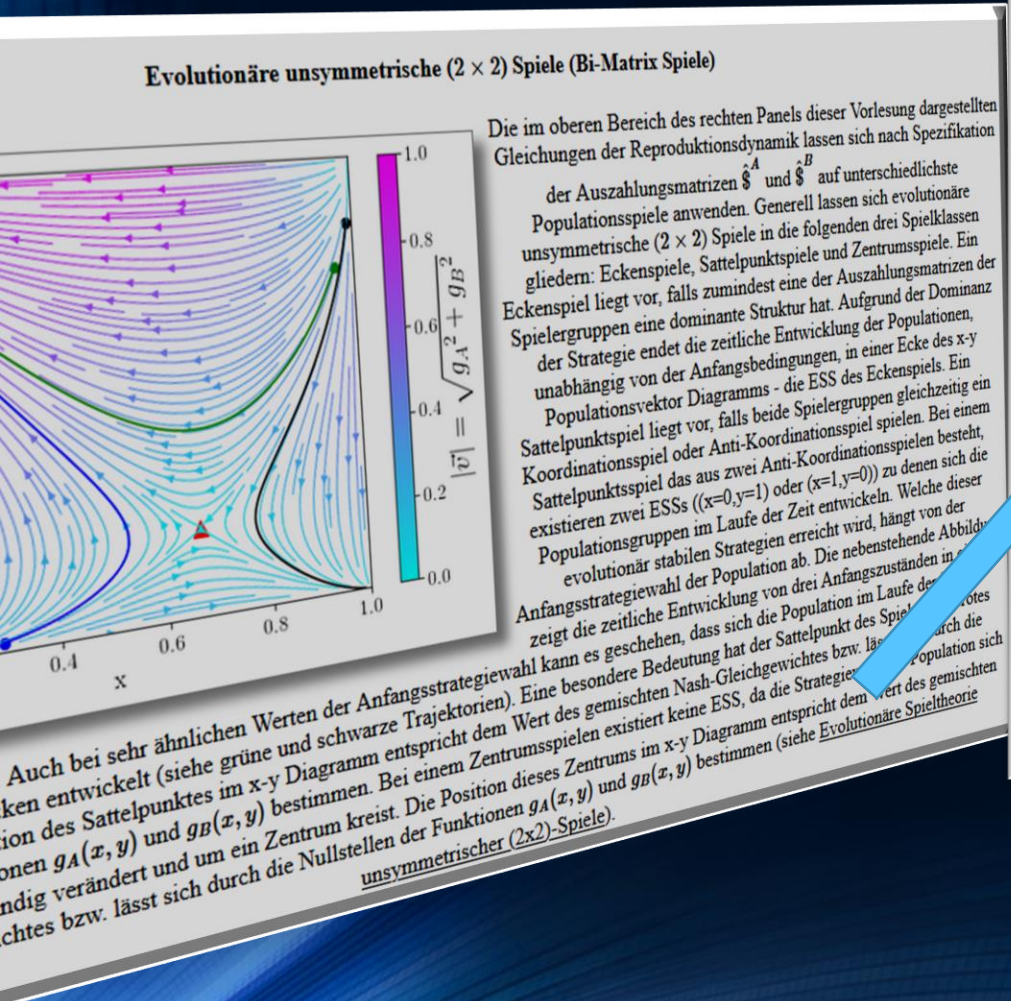
Zentrumsspiele

Spiel A: Koordinationsspiel
Spiel B: Anti-Koordinationsspiel
oder
Spiel A: Anti-Koordinationsspiel
Spiel B: Koordinationsspiel



Jupyter Notebook

Auf der Internetseite der Vorlesung



Physik der sozio-ökonomischen Systeme mit dem Computer

(Physics of Socio-Economic Systems with the Computer)

Vorlesung gehalten an der J.W.Goethe-Universität in Frankfurt am Main

(Wintersemester 2020/21)

von Dr.phil.nat. Dr.rer.pol. Matthias Hanauske

Frankfurt am Main 02.11.2020

Erster Vorlesungsteil:

Klassifizierung evolutionärer Bi-Matrix Spiele (unsymmetrische (2 x 2)-Spiele)

Einführung

In diesem Unterkapitel werden die unterschiedlichen Spieltypen evolutionärer Bi-Matrix Spiele (unsymmetrische (2 x 2)-Spiele) klassifiziert. Ausgangspunkt sind die folgenden allgemeinen Auszahlungsmatrizen der Spielergruppen A und B. Da es sich um unsymmetrische Auszahlungsmatrizen nehmen wir das folgende an:

$$\hat{\$}^B \neq (\hat{\$}^A)^T$$

$$\hat{\$}^A = \begin{pmatrix} \$_{11}^A & \$_{12}^A \\ \$_{21}^A & \$_{22}^A \end{pmatrix}, \quad \hat{\$}^B = \begin{pmatrix} \$_{11}^B & \$_{12}^B \\ \$_{21}^B & \$_{22}^B \end{pmatrix}$$

Unsymmetrische (2 x 2) Spiele lassen sich in die folgenden Spielklassen gliedern:

Die Klasse der Eckenspiele (engl.: corner class games)

Ein Eckenspiel liegt vor, falls eine der Auszahlungsmatrizen der Spielergruppen eine dominante Struktur hat; falls $\hat{\A oder $\hat{\B ein dominantes Spiel ist.

Die Klasse der Sattelpunktspiele (engl.: saddle class games)

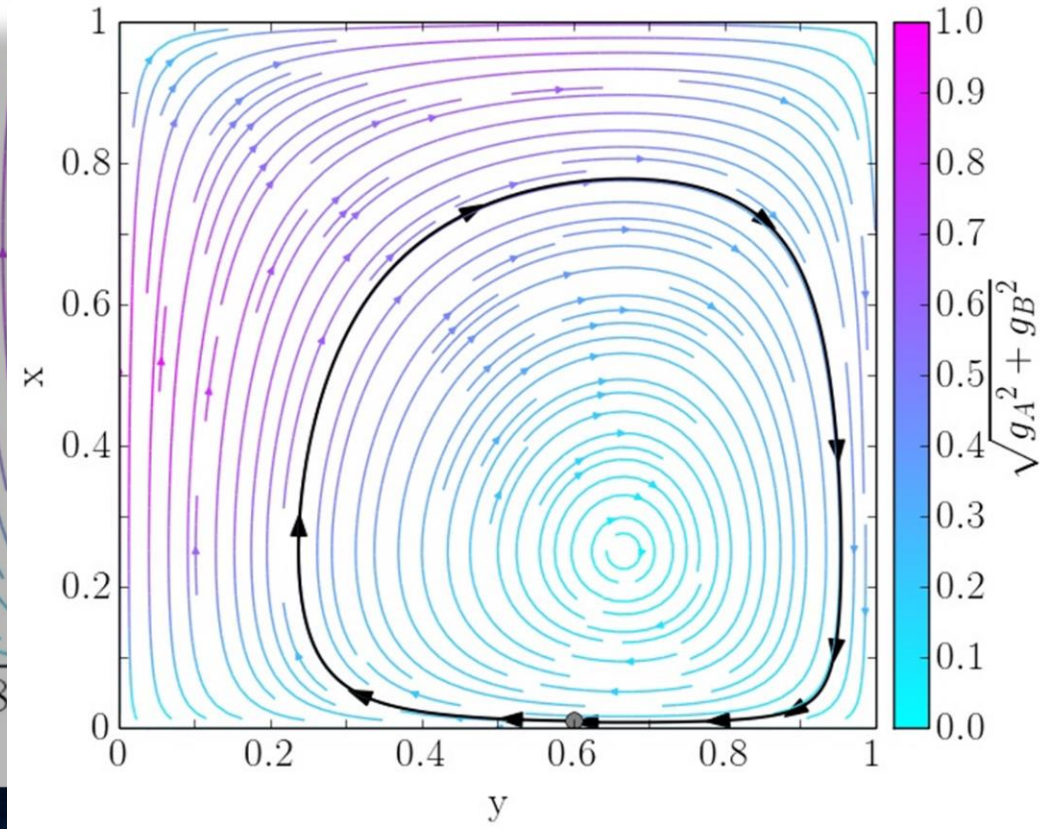
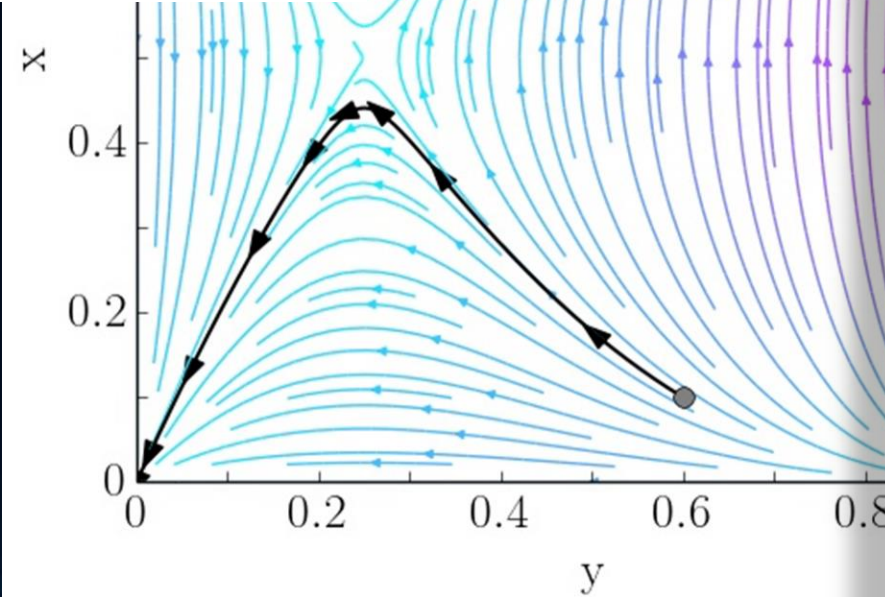
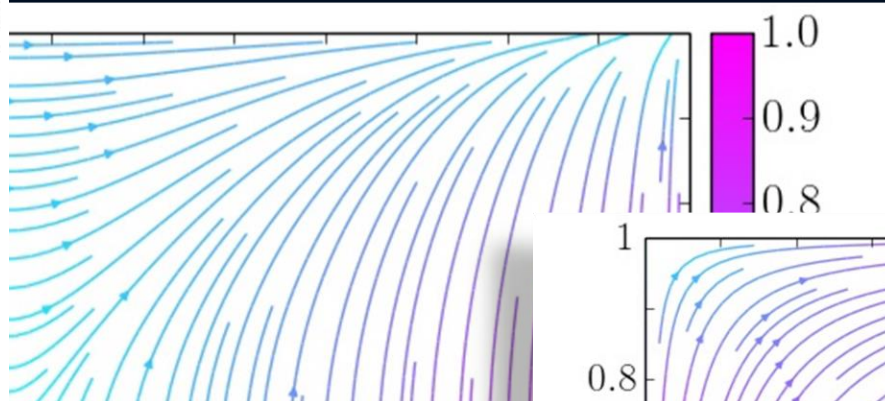
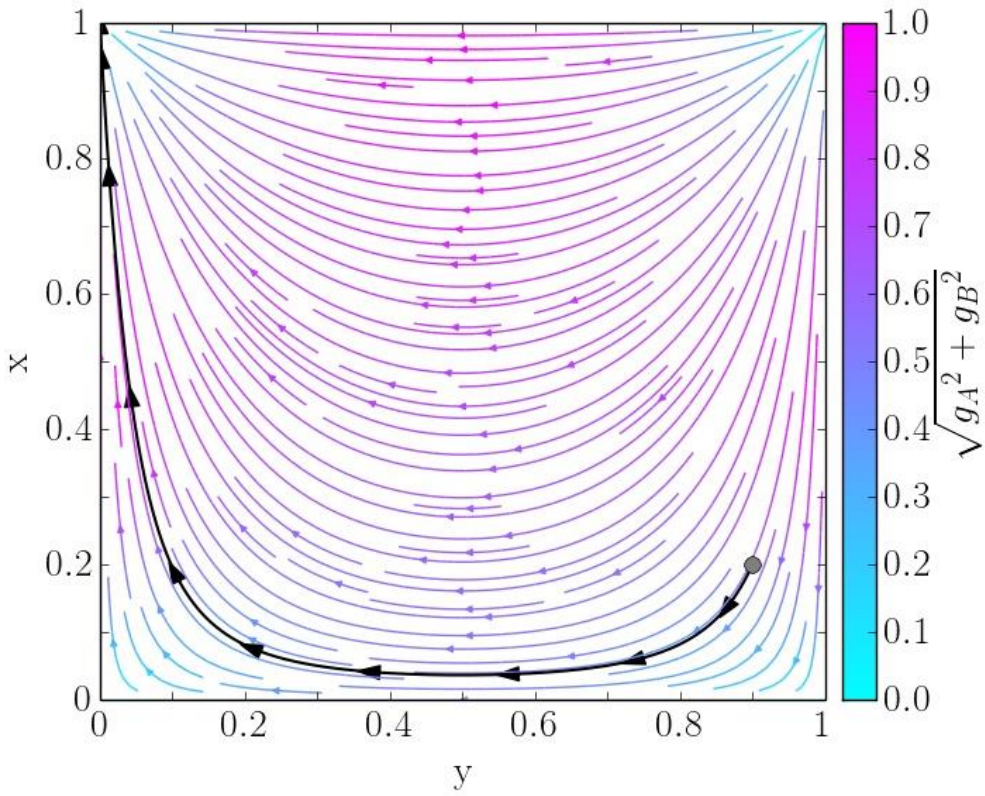
Ein Sattelpunktspiel liegt vor, falls beide Spielergruppen gleichzeitig ein Koordinationsspiel oder Anti-Koordinationsspiel spielen.

Die Klasse der Zentrumsspiele (engl.: center class games)

Ein Zentrumsspiel liegt vor, falls Spielergruppe A ein Koordinationsspiel und Spielergruppe B ein Anti-Koordinationsspiel spielen; falls Spielergruppe A ein Anti-Koordinationsspiel und Spielergruppe B ein Koordinationsspiel spielen.

Zusätzlich befindet sich das Jupyter Notebook auch auf der OLAT Seite des Kurses

Bi-Matrix Spiele mit Python Version bimatrix1.py



Differentialgleichungen: Numerische Lösung von Anfangswertproblemen

Im vorigen Unterpunkt hatten wir die Bewegung einzelner Teilchen in einer Kiste simuliert. In der Physik ist die zeitliche Entwicklung eines Systems oft in Form von Differentialgleichungen (DGLs) gegeben. In diesem Unterpunkt betrachten wir das numerische Lösen einer Differentialgleichung erster Ordnung der Form

$$\dot{y}(t) = \frac{dy(t)}{dt} = f(t, y(t)) \quad , \text{ mit: } a \leq t \leq b, \quad y(a) = \alpha \quad .$$

Die Funktion $f(t, y(t))$ bestimmt die DGL und somit das Verhalten der gesuchten Funktion $y(t)$. Es wird hierbei vorausgesetzt, dass $f(t, y(t))$ auf einer Teilmenge $\mathcal{D} = \{(t, y) | a \leq t \leq b, -\infty \leq y \leq \infty\}$ kontinuierlich definiert ist. Weiter wird angenommen, dass das so definierte Anfangswertproblem "well-posed" ist und eine eindeutige Lösung $y(t)$ existiert ("well-posed" bedeutet hier, dass die Differentialgleichung eine Struktur hat, bei der kleine Störungen im Anfangszustand nicht exponentiell anwachsen). Wir hatten bereits gesehen, wie man Differentialgleichungen mittels Jupyter Notebooks und SymPy DGLs analytisch löst (siehe [Jupyter Notebooks und das Rechnen mit symbolischen Ausdrücken](#)). Nicht jede DGL lässt sich analytisch lösen und falls der Befehl "dsolve()" keine sinnvollen Resultate liefert, muss man die zeitliche Entwicklung der Funktion $y(t)$ numerisch berechnen. Die numerische Lösung der DGL kann man sich auch direkt in Python mittels der Methode "integrate.odeint()" berechnen (Python-Modul "scipy"). Möchte man die Lösung jedoch in einem C++ Programm berechnen, so ist man auf die Anwendung eines numerischen Verfahrens angewiesen.

Das einfache Euler Verfahren zum Lösen einer DGL

Das wohl einfachste Verfahren zum Lösen einer DGL erster Ordnung ist die Euler Methode. Hierzu schreibt man die DGL als eine Differenzengleichung um

$$\frac{dy(t)}{dt} = f(t, y(t)) \rightarrow \Delta y = f(t, y) \cdot \Delta t \rightarrow \Delta y = h \cdot f(t, y)$$

und unterteilt das Zeitintervall $[a, b]$ in $N + 1$ äquidistante Zeit-Gitterpunkte $(t_0, t_1, t_2, \dots, t_N)$, wobei $t_i = a + i h \quad \forall i = 0, 1, 2, \dots, N$. Im Algorithmus der Euler Methode startet man bei $t = t_0$ und $y = y_0 = \alpha$ (Anfangsbedingungen des Systems) und erhöht dann iterativ die Zeit t um den Wert von h . Den neuen y -Wert erhält man mittels $y_1 = y_0 + h \cdot f(t_0, y_0)$ und man führt das Verfahren so lange aus, bis man an den letzten zeitlichen Gitterpunkt gelangt.

Betrachten wir z.B. die einfache Differentialgleichung

$$\frac{dy(t)}{dt} = f(t, y(t)) = -y(t) \quad ,$$

die den exponentiellen Abfall einer Funktion $y(t)$ beschreibt. Obwohl sich die allgemeine Lösung der DGL einfach bestimmen lässt ($y(t) = \alpha \cdot e^{-t}$, mit $\alpha = y(0)$), möchten wir die DGL auf numerischem Wege lösen. Das folgende C++ Programm benutzt die Eulermethode und entwickelt die obere Differentialgleichung im Zeitintervall $[a, b] = [0, 2]$ mittels 101 Gitterpunkten. Die simulierten Daten werden dann, zusammen mit der analytischen Lösung im Terminal ausgegeben.

DGL_0.cpp

```
/* Berechnung der Lösung einer Differentialgleichung der Form y'=f(t,y)
 * mittels der einfachen Euler Methode und f(t,y) = y
 * Zeitentwicklung der fuer unterschiedliche t-Werte in [a,b]
 */
#include <stdio.h> // Standard Input- und Output Bibliothek in C, z.B. printf(...)
#include <cmath> // Bibliothek für mathematisches (e-Funktion, Betrag, ...)

double f(double t, double y){ // Definition der Funktion f(t,x)
    double wert; // Eigentliche Definition der Funktion
    wert = - y; // Rueckgabewert der Funktion f(t,x)
    return wert; // Ende der Funktion f(t,x)
}

double y_analytisch(double t, double alpha){ // Analytische Loesung der DGL
    double wert; // bei gegebenem Anfangswert y(a)=alpha
    wert = alpha*exp(-t); // Eigentliche Definition der analytische Loesung
    return wert; // Rueckgabewert
} // Ende der Definition

int main(){ // Hauptfunktion
    double a = 0; // Untergrenze des Zeit-Intervalls [a,b] in dem die Loesung berechnet werden soll
    double b = 2; // Obergrenze des Intervalls [a,b]
    int N = 100; // Anzahl der Punkte in die das t-Intervall aufgeteilt wird
    double h = (b - a)/N; // Abstand dt zwischen den aequidistanten Punkten des t-Intervalls (h=dt)
    double alpha = 0.5; // Anfangswert bei t=a: y(a)=alpha
    double t; // Aktueller Zeitwert
    double y = alpha; // Deklaration und Initialisierung der numerischen Loesung der Euler Methode

    printf("# 0: Index i \n# 1: t-Wert \n# 2: Euler Methode \n"); // Beschreibung der ausgegebenen Groessen
    printf("# 3: Analytische Loesung \n"); // Beschreibung der ausgegebenen Groessen

    for(int i = 0; i <= N; ++i){ // for-Schleife ueber die einzelnen Punkte des t-Intervalls
        t = a + i*h; // Zeit-Parameter wird um h erhoeht
        printf("%3d %19.15f %19.15f %19.15f\n",i, t, y, y_analytisch(t,0.5)); // Ausgaben der Loesung

        y = y + h*f(t,y); // Euler Methode
    } // Ende for-Schleife
} // Ende der Hauptfunktion
```

Lösen einer einfachen DGL erster Ordnung mittels des Euler Verfahrens

Numerische Verfahren zum Lösen von Differentialgleichung erster Ordnung

Das Euler Verfahren:

$$y_{i+1} = y_i + h \cdot f(t_i, y_i)$$

Mittelpunktmethode:

$$y_{i+1} = y_i + h \cdot \left[f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2} f(t_i, y_i)\right) \right]$$

Modifizierte Euler Methode:

$$y_{i+1} = y_i + \frac{h}{2} \cdot [f(t_i, y_i) + f(t_{i+1}, y_i + h f(t_i, y_i))]$$

Runge-Kutta Ordnung vier:

$$y_{i+1} = y_i + \frac{1}{6} \cdot (k_1 + 2k_2 + 2k_3 + k_4) \quad , \text{ wobei:}$$

$$k_1 = h f(t_i, y_i)$$

$$k_2 = h f\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_1\right)$$

$$k_3 = h f\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_2\right)$$

$$k_4 = h f(t_{i+1}, y_i + k_3)$$

```

* Zeitentwicklung der fuer unterschiedliche t-Werte in [a,b]
* Ausgabe zum Plotten mittels Python Jupyter Notebook DGL_1.ipynb: "./a.out"
*/

```

```

#include <stdio.h>
#include <cmath>

double f(double t, double y){
    double wert;
    wert = y - pow(t,2) + 1;
    return wert;
}

double y_analytisch(double t, double alpha){
    double wert;
    wert = (alpha + (pow(t,2) + 2*t + 1)*exp(-t) - 1)*exp(t);
    return wert;
}

int main(){
    double a = 0;
    double b = 2;
    int N = 10;
    double h = (b - a)/N;
    double alpha = 0.5;
    double t;
    double y_Euler = alpha;
    double y_Midpoint = alpha;
    double y_Euler_M = alpha;
    double y_RungeK_4 = alpha;
    double k1, k2, k3, k4;

    printf("# 0: Index i \n# 1: t-Wert \n# 2: Euler Methode \n");
    printf("# 3: Mittelpunkt Methode \n# 4: Modifizierte Euler Methode \n");
    printf("# 5: Runge-Kutta Ordnung vier \n# 6: Analytische Loesung \n");

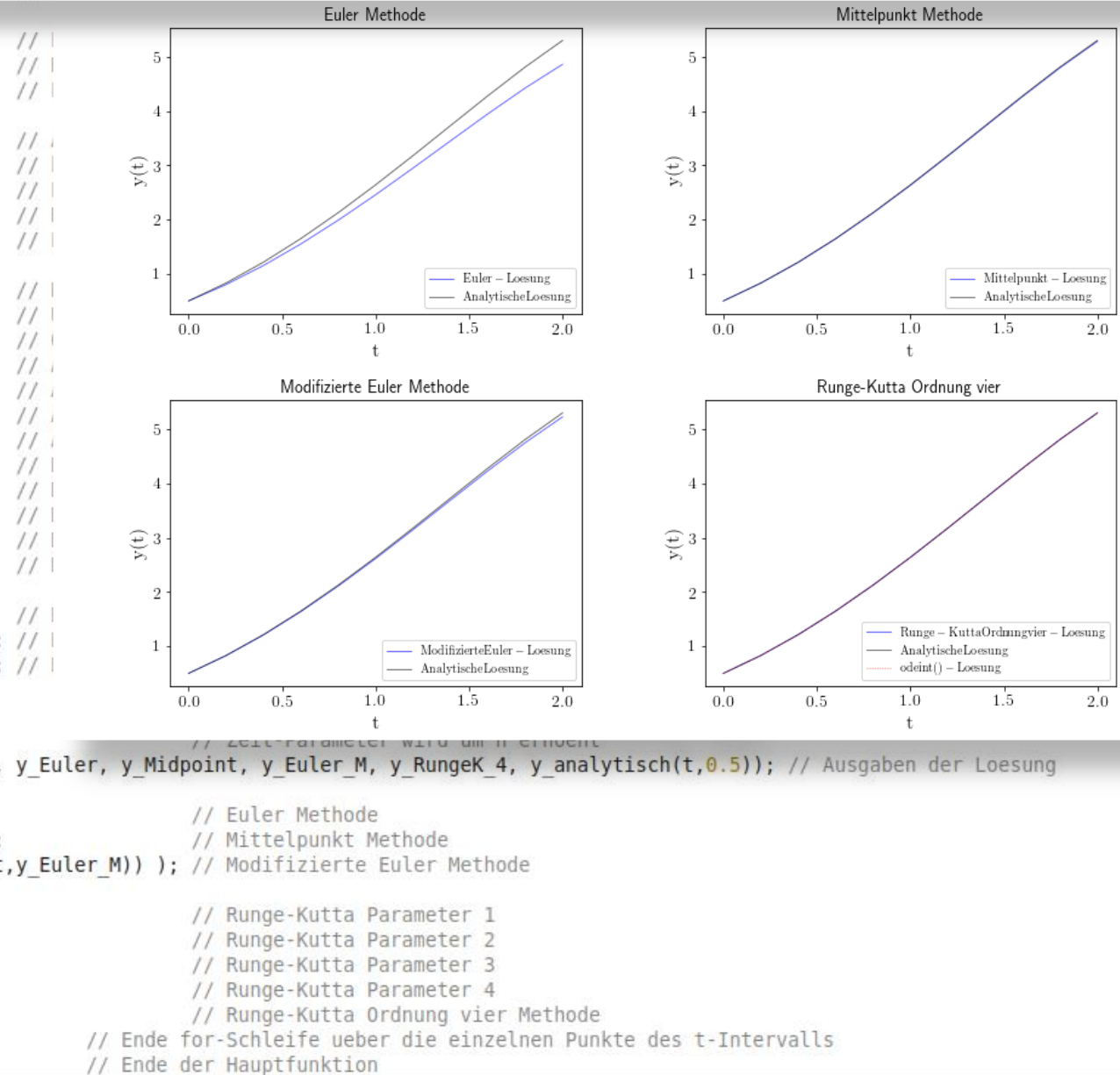
    for(int i = 0; i <= N; ++i){
        t = a + i*h;
        printf("%3d %19.15f %19.15f %19.15f %19.15f %19.15f %19.15f\n",i, t, y_Euler, y_Midpoint, y_Euler_M, y_RungeK_4, y_analytisch(t,0.5));

        y_Euler = y_Euler + h*f(t,y_Euler);
        y_Midpoint = y_Midpoint + h*f(t+h/2,y_Midpoint+h/2*f(t,y_Midpoint));
        y_Euler_M = y_Euler_M + h/2*( f(t,y_Euler_M) + f(t+h,y_Euler_M+h*f(t,y_Euler_M)) );

        k1 = h*f(t,y_RungeK_4);
        k2 = h*f(t+h/2,y_RungeK_4+k1/2);
        k3 = h*f(t+h/2,y_RungeK_4+k2/2);
        k4 = h*f(t+h,y_RungeK_4+k3);
        y_RungeK_4 = y_RungeK_4 + (k1 + 2*k2 + 2*k3 + k4)/6;
    }
}

```

Anwendung auf die DGL: $\frac{dy}{dt} = y - t^2 + 1$





Einführung in die Programmierung für Studierende der Physik

(Introduction to Programming for Physicists)

Vorlesung gehalten an der J.W.Goethe-Universität in Frankfurt am Main

(Sommersemester 2022)

von Dr.phil.nat. Dr.rer.pol. Matthias Hanauske

Frankfurt am Main 01.04.2022

Numerisches Lösen einer DGL erster Ordnung mit Python

Numerisches Lösen von Differentialgleichungen (das Anfangswertproblem)

Zunächst wird das Python Modul "sympy" eingebunden, das ein Computer-Algebra-System für Python bereitstellt und eine Vielzahl an symbolischen Berechnungen im Bereich der Mathematik und Physik relativ einfach möglich macht. Falls Sie das "sympy" Modul das erste Mal verwenden, müssen Sie es zunächst in Ihrer Python 3 Umgebung installieren (z.B. in einem Linux Terminal mit "pip3 install sympy").

```
In [1]: from sympy import *  
init_printing()
```

Wir betrachten in diesem Jupyter Notebook das numerische Lösen einer Differentialgleichung (DGL) erster Ordnung der Form

$$\dot{y}(t) = \frac{dy(t)}{dt} = f(t, y(t)) \quad , \text{ mit: } a \leq t \leq b, \quad y(a) = \alpha \quad .$$

Die Funktion $f(t, y(t))$ bestimmt die DGL und somit das Verhalten der gesuchten Funktion $y(t)$. Es wird hierbei vorausgesetzt, dass $f(t, y(t))$ auf einer Teilmenge $D = \{(t, y) | a \leq t \leq b, -\infty \leq y \leq \infty\}$ kontinuierlich definiert ist. Weiter wird angenommen, dass das so definierte Anfangswertproblem "well-posed" ist und eine eindeutige Lösung $y(t)$ existiert ("well-posed" bedeutet hier, dass die Differentialgleichung eine Struktur hat, bei der kleine Störungen im Anfangszustand nicht exponentiell anwachsen).

Vorlesung 9



In dieser Vorlesung befassen wir uns zunächst mit dem numerischen Lösen von Systemen gekoppelter Differentialgleichungen und Differentialgleichungen zweiter Ordnung und stellen im darauf folgenden Teil mögliche Programmierprojekte vor, die von den Studierenden bearbeitet werden können. Beim Klicken auf die Überschriften der Projekte gelangen Sie zu einer detaillierteren Beschreibung der einzelnen Projektthemen.

Systeme von gekoppelten Differentialgleichungen und Differentialgleichungen zweiter Ordnung

In der vorigen Vorlesung hatten wir die unterschiedlichen Verfahren zum Lösen von Differentialgleichungen erster Ordnung kennengelernt. Die Bewegungsgleichungen vieler physikalischer Systeme sind jedoch von zweiter Ordnung in der Zeit und in diesem Teilkapitel beschreiben wir die Vorgehensweise wie man solche Differentialgleichungen höherer Ordnung numerisch mittels eines C++ Programmes löst. Um ein Differentialgleichung zweiter Ordnung mittels des Computers lösen zu können, schreibt man zunächst die DGL in ein System von zwei gekoppelten Differentialgleichungen erster Ordnung um und diese löst man dann mit den Verfahren, die in der vorigen Vorlesung behandelt wurden. In diesem Unterpunkt werden wir uns zunächst mit Systemen von gekoppelten Differentialgleichungen erster Ordnung befassen und dann das numerische Lösen von Differentialgleichungen zweiter Ordnung vorstellen (näheres siehe Systeme von gekoppelten Differentialgleichungen und Differentialgleichungen zweiter Ordnung).

Studentische Projekte

Das Foucaultsche Pendel

Im Jahre 1851 gelang Jean Bernard Léon Foucault ein anschaulicher Beweis der Erdrotation. Aufgrund der, in rotierenden Bezugssystemen auftretenden Coriolisbeschleunigung, dreht sich die Schwingungsebene des Pendels langsam. Dieses Projekt ist ein Anwendungsfall der Newtonschen Mechanik in bewegten Bezugssystemen (siehe z.B. Walter Greiner, 'Klassische Mechanik II' [8. Auflage, 2008, Kapitel I3. Seite 18]) und das zugrundeliegende System von drei gekoppelten Differentialgleichungen zweiter Ordnung gilt es numerisch mittels eines C++ Programmes zu lösen und die berechneten Daten mittels Python zu visualisieren.

Das periodisch angetriebene Pendel

Das Projekt *periodisch angetriebenes Pendel* ist ein Anwendungsfall aus der klassischen Mechanik (siehe z.B. Walter Greiner, 'Klassische Mechanik II' [8. Auflage, 2008, Kapitel VII27. Seite 496]). Das System besteht aus einem Pendel, auf welches zusätzlich eine äußere Kraft mit periodischer Zeitabhängigkeit wirkt. Außerdem soll das Pendel durch

Vorlesung 9

Das numerische Lösen von Differentialgleichungen ist ein mathematisch anspruchsvolles Thema und kann in dieser Vorlesung nicht im Detail erläutert werden. Im ersten Teil dieser Vorlesung sollen die im vorigen Unterpunkt (Differentialgleichungen: Numerische Lösung von Anfangswertproblemen) besprochenen Verfahren auf Systeme von gekoppelten Differentialgleichungen und Differentialgleichungen zweiter Ordnung angewendet werden. Die Bewegungsgleichungen vieler physikalischer Systeme sind von zweiter Ordnung in der Zeit und es soll die Vorgehensweise besprochen werden, wie man solche Differentialgleichungen höherer Ordnung numerisch mittels eines C++ Programmes löst. Ab der nächsten Vorlesung (Vorlesung 10) werden die Studierenden dann an eigenen Projekten arbeiten, wobei viele dieser Projekte Probleme behandeln, die man nur mittels einer numerischen Lösung adäquat beschreiben kann (siehe linkes Panel dieser Vorlesung). Beim Klicken auf die Überschriften der Projekte gelangen Sie zu einer detaillierteren Beschreibung der einzelnen Projektthemen. Neben den hier vorgestellten Projekten, können auch eigene Projektthemen behandelt werden. Bei Interesse können auch Projekte aus dem Bereich der Physik der sozio-ökonomischen Systeme (z.B. Replikatorndynamik der Evolutionären Spieltheorie, Simulationen von komplexen Netzwerken), oder Projekte im Themengebiet der allgemeinen Relativitätstheorie (z.B. Bewegung um ein schwarzes Loch mittels der Geodätengleichung, näheres siehe Allgemeine Relativitätstheorie mit dem Computer) behandelt werden. Die studentischen Projekte können alleine oder in Gruppen (bis zu drei Personen) durchgeführt werden.

Systeme von gekoppelten Differentialgleichungen und Differentialgleichungen zweiter Ordnung

Im vorigen Unterpunkt hatten wir die unterschiedlichen Verfahren zum Lösen von Differentialgleichungen erster Ordnung kennengelernt. Die Bewegungsgleichungen vieler physikalischer Systeme sind jedoch von zweiter Ordnung in der Zeit und dieses Unterkapitel der Vorlesung 9 befasst sich damit, wie man solche Differentialgleichungen höherer Ordnung numerisch mittels eines C++ Programmes löst. Um ein Differentialgleichung zweiter Ordnung mittels des Computers lösen zu können, schreibt man zunächst die DGL in ein System von zwei gekoppelten Differentialgleichungen ersten Ordnung um und diese löst man dann mit den Verfahren, die in der vorigen Vorlesung behandelt wurden. In diesem Unterpunkt werden wir uns zunächst mit Systemen von gekoppelten Differentialgleichungen erster Ordnung befassen und dann das numerische Lösen von Differentialgleichungen zweiter Ordnung vorstellen.

Systeme von gekoppelten Differentialgleichungen

Wir betrachten zunächst das numerische Lösen eines Systems von m -gekoppelten Differentialgleichungen (DGLs) erster Ordnung der Form

$$\begin{aligned} \dot{y}_1(t) &= \frac{dy_1}{dt} = f_1(t, y_1, y_2, \dots, y_m) \\ \dot{y}_2(t) &= \frac{dy_2}{dt} = f_2(t, y_1, y_2, \dots, y_m) \\ \dot{y}_3(t) &= \dots = \\ &\dots = \dots \\ \dot{y}_m(t) &= \frac{dy_m}{dt} = f_m(t, y_1, y_2, \dots, y_m) \quad , \end{aligned}$$

wobei die zeitliche Entwicklung der Vektorfunktion $\vec{y}(t) = (y_1(t), y_2(t), \dots, y_m(t))$ in den Grenzen $a \leq t \leq b$ gesucht wird. Die m -Funktionen $f_i(t, y_1, y_2, \dots, y_m)$, $i \in [1, 2, \dots, m]$ bestimmen das System der DGLs und somit das Verhalten der gesuchten Funktion $\vec{y}(t)$. Es wird hierbei vorausgesetzt, dass die Funktionen $f_i(t, y_1, y_2, \dots, y_m)$ auf einer Teilmenge $\mathcal{D} \subset \mathbb{R}^{m+1}$ kontinuierlich definiert sind und das so definierte Anfangswertproblem "well-posed" ist und eine eindeutige Lösung $\vec{y}(t)$ existiert. Bei gegebener Anfangskonfiguration

$$y_1(a) = \alpha_1, y_2(a) = \alpha_2, \dots, y_m(a) = \alpha_m$$

ist es dann numerisch möglich das System von gekoppelten DGLs zu lösen.

Systeme von gekoppelten Differentialgleichungen ▶

Wir betrachten zunächst das numerische Lösen eines Systems von m -gekoppelten Differentialgleichungen (DGLs) erster Ordnung der Form

$$\begin{aligned}\dot{y}_1(t) &= \frac{dy_1}{dt} = f_1(t, y_1, y_2, \dots, y_m) \\ \dot{y}_2(t) &= \frac{dy_2}{dt} = f_2(t, y_1, y_2, \dots, y_m) \\ \dot{y}_3(t) &= \dots = \\ &\dots = \dots \\ \dot{y}_m(t) &= \frac{dy_m}{dt} = f_m(t, y_1, y_2, \dots, y_m) \quad ,\end{aligned}$$

wobei die zeitliche Entwicklung der Vektorfunktion $\vec{y}(t) = (y_1(t), y_2(t), \dots, y_m(t))$ in den Grenzen $a \leq t \leq b$ gesucht wird.

Die m -Funktionen $f_i(t, y_1, y_2, \dots, y_m)$, $i \in [1, 2, \dots, m]$ bestimmen das System der DGLs und somit das Verhalten der gesuchten Funktion $\vec{y}(t)$. Es wird hierbei vorausgesetzt, dass die Funktionen $f_i(t, y_1, y_2, \dots, y_m)$ auf einer Teilmenge $\mathcal{D} \subseteq \mathbb{R}^{m+1}$ kontinuierlich definiert sind und das so definierte Anfangswertproblem "well-posed" ist und eine eindeutige Lösung $\vec{y}(t)$ existiert. Bei gegebener Anfangskonfiguration

$$y_1(a) = \alpha_1, y_2(a) = \alpha_2, \dots, y_m(a) = \alpha_m$$

ist es dann numerisch möglich das System von gekoppelten DGLs zu lösen.

Beispiel: Numerische Lösung eines Systems von zwei gekoppelten Differentialgleichungen erster Ordnung

Wir betrachten speziell das folgende System bestehend aus zwei gekoppelten DGLs ($m = 2$):

$$\begin{aligned}\dot{y}_1(t) &= \frac{dy_1}{dt} = 3y_1 + 2y_2 - (2t^2 + 1) \cdot e^{2t} =: f_1(t, y_1, y_2) \\ \dot{y}_2(t) &= \frac{dy_2}{dt} = 4y_1 + y_2 + (t^2 + 2t - 4) \cdot e^{2t} =: f_2(t, y_1, y_2) \quad ,\end{aligned}$$

und sind an der numerischen Lösung $\vec{y}(t) = (y_1(t), y_2(t))$ im Zeitintervall $t \in [0, 1]$ interessiert. Die Anfangsbedingungen lauten

$$y_1(0) = \alpha_1 = 1, \quad y_2(0) = \alpha_2 = 1 \quad .$$

Das Lösen dieses Systems von DGLs ist auf gleichem Wege möglich, wie man einzelne Differentialgleichungen numerisch approximiert.


```

int main(){
    double a = 0;
    double b = 1;
    int N = 100;
    double h = (b - a)/N;
    double alpha_1 = 1;
    double alpha_2 = 1;
    double t;
    double y_Euler_1 = alpha_1;
    double y_RungeK_4_1 = alpha_1;
    double k1_1,k2_1,k3_1,k4_1;
    double y_Euler_2 = alpha_2;
    double y_RungeK_4_2 = alpha_2;
    double k1_2,k2_2,k3_2,k4_2;
    double tmp;

    // Hauptfunktion
    // Untergrenze des Zeit-Intervalls [a,b] in dem die Loesung berechnet werden soll
    // Obergrenze des Intervalls [a,b]
    // Anzahl der Punkte in die das t-Intervall aufgeteilt wird
    // Abstand dt zwischen den aequidistanten Punkten des t-Intervalls (h=dt)
    // 1.Anfangswert bei t=a: y_1(a)=alpha_1
    // 2.Anfangswert bei t=a: y_2(a)=alpha_2
    // Aktueller Zeitwert
    // Deklaration und Initialisierung der numerischen Loesung der Euler Methode fuer y_1
    // Deklaration und Initialisierung der numerischen Loesung der Runge-Kutta Ordnung vier Methode
    // Deklaration der vier Runge-Kutta Parameter fuer y_1
    // Deklaration und Initialisierung der numerischen Loesung der Euler Methode fuer y_2
    // Deklaration und Initialisierung der numerischen Loesung der Runge-Kutta Ordnung vier Methode
    // Deklaration der vier Runge-Kutta Parameter fuer y_2
    // Variable zum Zwischenspeichern von Ergebnissen

    printf("# 0: Index i \n# 1: t-Wert \n# 2: Euler Methode y1 \n# 3: Euler Methode y2 \n"); // Beschreibung der ausgegebenen Groessen
    printf("# 4: Runge-Kutta Ordnung vier y1 \n# 5: Runge-Kutta Ordnung vier y2 \n"); // Beschreibung der ausgegebenen Groessen
    printf("# 6: Analytische Loesung y1 \n# 7: Analytische Loesung y2 \n"); // Beschreibung der ausgegebenen Groessen
    printf("%3d %19.15f %19.15f %19.15f %19.15f ",0, t, y_Euler_1, y_Euler_2, y_RungeK_4_1); // Ausgaben der t=a Werte
    printf(" %19.15f %19.15f %19.15f \n", y_RungeK_4_2, y_1_analytisch(t), y_2_analytisch(t)); // Ausgaben der t=a Werte

    for(int i=0; i <= N; ++i){ // for-Schleife ueber die einzelnen Punkte des t-Intervalls
        t = a + i*h; // Zeit-Parameter wird um h erhoeht
        printf("%3d %19.15f %19.15f %19.15f %19.15f ",i, t, y_Euler_1, y_Euler_2, y_RungeK_4_1); // Ausgaben der Loesungen
        printf(" %19.15f %19.15f %19.15f \n", y_RungeK_4_2, y_1_analytisch(t), y_2_analytisch(t)); // Ausgaben der Loesungen

        tmp = y_Euler_1 + h*f_1(t,y_Euler_1,y_Euler_2); // Euler Methode
        y_Euler_2 = y_Euler_2 + h*f_2(t,y_Euler_1,y_Euler_2); // y_2 Euler Methode
        y_Euler_1 = tmp; // y_1 Euler Methode

        k1_1 = h*f_1(t,y_RungeK_4_1,y_RungeK_4_2); // Runge-Kutta Parameter k1 fuer y_1
        k1_2 = h*f_2(t,y_RungeK_4_1,y_RungeK_4_2); // Runge-Kutta Parameter k1 fuer y_2
        k2_1 = h*f_1(t+h/2,y_RungeK_4_1+k1_1/2,y_RungeK_4_2+k1_2/2); // Runge-Kutta Parameter k2 fuer y_1
        k2_2 = h*f_2(t+h/2,y_RungeK_4_1+k1_1/2,y_RungeK_4_2+k1_2/2); // Runge-Kutta Parameter k2 fuer y_2
        k3_1 = h*f_1(t+h/2,y_RungeK_4_1+k2_1/2,y_RungeK_4_2+k2_2/2); // Runge-Kutta Parameter k3 fuer y_1
        k3_2 = h*f_2(t+h/2,y_RungeK_4_1+k2_1/2,y_RungeK_4_2+k2_2/2); // Runge-Kutta Parameter k3 fuer y_2
        k4_1 = h*f_1(t+h,y_RungeK_4_1+k3_1,y_RungeK_4_2+k3_2); // Runge-Kutta Parameter k4 fuer y_1
        k4_2 = h*f_2(t+h,y_RungeK_4_1+k3_1,y_RungeK_4_2+k3_2); // Runge-Kutta Parameter k4 fuer y_2
        y_RungeK_4_1 = y_RungeK_4_1 + (k1_1 + 2*k2_1 + 2*k3_1 + k4_1)/6; // Runge-Kutta Ordnung vier Methode fuer y_1
        y_RungeK_4_2 = y_RungeK_4_2 + (k1_2 + 2*k2_2 + 2*k3_2 + k4_2)/6; // Runge-Kutta Ordnung vier Methode fuer y_2
    } // Ende for-Schleife ueber die einzelnen Punkte des t-Intervalls
} // Ende der Hauptfunktion

```

**main()-Programm
Zur Lösung des
Systems bestehend
aus zwei DGLs erster
Ordnung**

```
int main(){
double a = 0;
double b = 1;
int N = 100;
double h = (b - a)/N;
double alpha_1 = 1;
double alpha_2 = 1;
double t;
double y_Euler_1 = alpha_1;
double y_Euler_2 = alpha_2;
double y_RungeK_4_1 = alpha_1;
double y_RungeK_4_2 = alpha_2;
double k1_1,k2_1,k3_1,k4_1;
double k1_2,k2_2,k3_2,k4_2;
double tmp;

printf("# 0: Index i \n#");
printf("# 4: Runge-Kutta Ordnung vier y1");
printf("# 6: Analytische Loesung y1");
printf("# 1: t-Wert");
printf("# 2: Euler Methode y1");
printf("# 3: Euler Methode y2");
printf("# 4: Runge-Kutta Ordnung vier y1");
printf("# 5: Runge-Kutta Ordnung vier y2");
printf("# 6: Analytische Loesung y1");
printf("# 7: Analytische Loesung y2");

for(int i=0; i <= N; ++i){
t = a + i*h;
printf("%3d %19.15f %19.15f %19.15f %19.15f ",i, t, y_Euler_1, y_Euler_2, y_RungeK_4_1); // Ausgaben der Loesungen
printf(" %19.15f %19.15f %19.15f \n", y_RungeK_4_2, y_1_analytisch(t), y_2_analytisch(t)); // Ausgaben der Loesungen

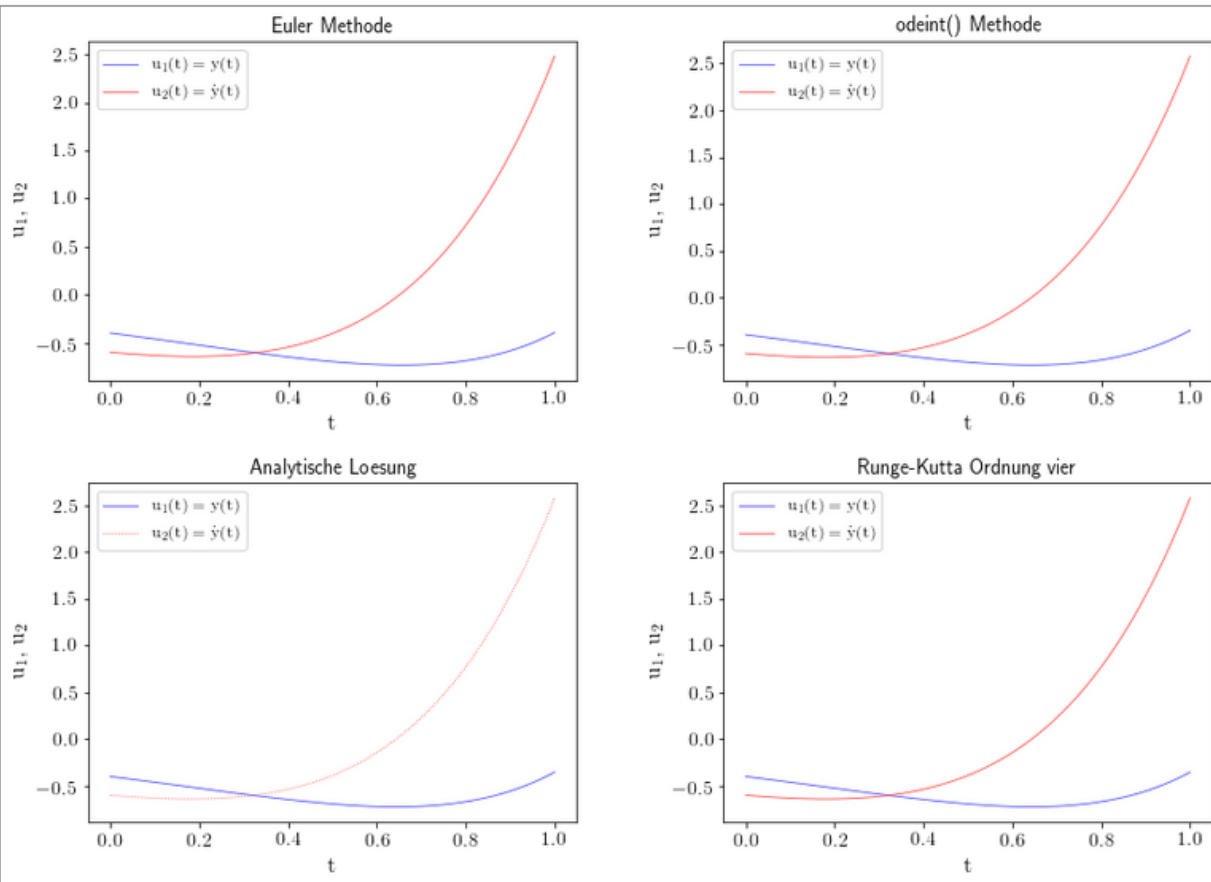
tmp = y_Euler_1 + h*f_1(t,y_Euler_1,y_Euler_2,y_RungeK_4_1);
y_Euler_2 = y_Euler_2 + h*f_2(t,y_Euler_1,y_Euler_2,y_RungeK_4_1);
y_Euler_1 = tmp;

k1_1 = h*f_1(t,y_RungeK_4_1,y_RungeK_4_2,k1_1);
k1_2 = h*f_2(t,y_RungeK_4_1,y_RungeK_4_2,k1_1);
k2_1 = h*f_1(t+h/2,y_RungeK_4_1,y_RungeK_4_2,k1_1,k2_1);
k2_2 = h*f_2(t+h/2,y_RungeK_4_1,y_RungeK_4_2,k1_1,k2_1);
k3_1 = h*f_1(t+h/2,y_RungeK_4_1,y_RungeK_4_2,k1_1,k2_1,k3_1);
k3_2 = h*f_2(t+h/2,y_RungeK_4_1,y_RungeK_4_2,k1_1,k2_1,k3_1);
k4_1 = h*f_1(t+h,y_RungeK_4_1,y_RungeK_4_2,k1_1,k2_1,k3_1,k4_1);
k4_2 = h*f_2(t+h,y_RungeK_4_1,y_RungeK_4_2,k1_1,k2_1,k3_1,k4_1);
y_RungeK_4_1 = y_RungeK_4_1 + (k1_1 + 2*k2_1 + 2*k3_1 + k4_1)/6; // Runge-Kutta Ordnung vier Methode fuer y_1
y_RungeK_4_2 = y_RungeK_4_2 + (k1_2 + 2*k2_2 + 2*k3_2 + k4_2)/6; // Runge-Kutta Ordnung vier Methode fuer y_2
}
}

(base) hanauske@hanauske-Aspire-A717-72G:~/PPROG/EigProg/V9$ g++ DGL_2.cpp
(base) hanauske@hanauske-Aspire-A717-72G:~/PPROG/EigProg/V9$ ./a.out
# 0: Index i
# 1: t-Wert
# 2: Euler Methode y1
# 3: Euler Methode y2
# 4: Runge-Kutta Ordnung vier y1
# 5: Runge-Kutta Ordnung vier y2
# 6: Analytische Loesung y1
# 7: Analytische Loesung y2
0 0.0000000000000000 1.0000000000000000 1.0000000000000000 1.0000000000000000 1.0000000000000000 1.0000000000000000 1.0000000000000000 1.0000000000000000
0 0.0000000000000000 1.0000000000000000 1.0000000000000000 1.0000000000000000 1.0000000000000000 1.0000000000000000 1.0000000000000000 1.0000000000000000
1 0.0100000000000000 1.0400000000000000 1.0100000000000000 1.040608426315013 1.010558940181763 1.040608427569041 1.010558941425456
2 0.0200000000000000 1.081195946197052 1.021097006868275 1.082468186487541 1.022272409932680 1.082468189115352 1.022272412539396
3 0.0300000000000000 1.123637330492212 1.033343871369918 1.125632778808522 1.035197418735856 1.125632782938618 1.035197422833657
4 0.0400000000000000 1.167375849311085 1.046795999898338 1.170158168240702 1.049393799070299 1.170158174010907 1.049393804796552
5 0.0500000000000000 1.212465508925469 1.061511473409989 1.216102907913251 1.064924341689591 1.216102915471323 1.064924349191579
6 0.0600000000000000 1.258962735934773 1.077551171969109 1.263528266739016 1.081854937478282 1.263528276243294 1.081854946913853
7 0.0700000000000000 1.306926493163091 1.094978905171568 1.312498363466237 1.100254726211197 1.312498375086330 1.100254737749452
8 0.0800000000000000 1.356418401240555 1.113861548730063 1.363080307492500 1.120196252557172 1.363080321410022 1.120196266379195
9 0.0900000000000000 1.407502866150967 1.134269187516644 1.415344346785449 1.141755629685859 1.415344363194790 1.141755645985495
10 0.1000000000000000 1.460247213041785 1.156275265373008 1.469364023272402 1.165012710854272 1.469364042381559 1.165012729838951
11 0.1100000000000000 1.514721826607265 1.179956742014139 1.525216336079555 1.190051269368648 1.525216358111003 1.190051291260258
89 0.8900000000000000 30.930880042824768 30.063568069606163 34.335278279597091 33.513116889352368 34.335285835073890 33.513124438401135
90 0.9000000000000000 32.306838455932400 31.516766532214152 35.919826655719859 35.176963303871311 35.919834678006687 35.176971319508823
91 0.9100000000000000 33.747878176687038 33.040117636018252 37.581811296515667 36.923393059891680 37.581819813513697 36.923401570012345
92 0.9200000000000000 35.257179970562206 34.636996585061937 39.325128086735063 38.756498858038242 39.325137127976433 38.756507892167264
93 0.9300000000000000 36.838082119087382 36.310942423138201 41.153871130268442 40.680578010174486 41.153880727010005 40.680587599561534
94 0.9400000000000000 38.494088264736298 38.065666064560588 43.072342882456958 42.700142759460462 43.072353067779481 42.700152937178956
95 0.9500000000000000 40.228875647423045 39.905058721679517 45.085064801277419 44.819931124470799 45.085075610190323 44.819941925522599
96 0.9600000000000000 42.046303752173394 41.833200748846942 47.196788543991907 47.044918294100583 47.196800013543999 47.044929755526724
97 0.9700000000000000 43.950423388516661 43.854370923514082 49.412507737215932 49.380328601353476 49.412519906612403 49.380340762351139
98 0.9800000000000000 45.945486223170619 45.973056186177487 51.737470349791899 51.831648105542975 51.737483260518005 51.831661007589204
99 0.9900000000000000 48.035954788670189 48.193961861970656 54.177191699361046 54.404637813947794 54.177205395312626 54.404651500929958
100 1.0000000000000000 50.226512991722970 50.522022387834483 56.737468125110773 57.105347575549835 56.737482652732375 57.105362093903814
(base) hanauske@hanauske-Aspire-A717-72G:~/PPROG/EigProg/V9$
```

Visualisierung und numerische Lösung mittels Python

Beim Klicken auf das untere rechte Bild gelangen Sie zu dem Jupyter Notebook [DGL_2.ipynb](#) in dem eine Visualisierung der Ergebnisse des oberen C++ Programms programmiert ist. Die untere Abbildung auf der linken Seite zeigt z.B. die Visualisierung der Daten von [DGL_2.cpp](#). Zusätzlich wird in dem Notebook auch die numerische Lösung direkt in Python generiert (Methode "integrate.odeint()" im Python-Modul "scipy") und mit den simulierten Daten der unterschiedlichen Verfahren verglichen.



Einführung in die Programmierung für Studierende der Physik

(Introduction to Programming for Physicists)

Vorlesung gehalten an der J.W.Goethe-Universität in Frankfurt am Main

(Sommersemester 2022)

von Dr.phil.nat. Dr.rer.pol. Matthias Hanauske

Frankfurt am Main 01.04.2022

Numerisches Lösen von Differentialgleichungen

Systeme von gekoppelten Differentialgleichungen und Differentialgleichungen zweiter Ordnung

Im Jupyter Notebook [DGL_1.ipynb](#) haben wir einige in Python implementierte Lösungsmethoden für Differentialgleichungen erster Ordnung kennengelernt. In diesem Notebook werden wir uns zunächst mit Systemen von gekoppelten Differentialgleichungen erster Ordnung befassen und dann das numerische Lösen von Differentialgleichungen zweiter Ordnung vorstellen.

Systeme von gekoppelten Differentialgleichungen

Wir betrachten zunächst das numerische Lösen eines Systems von m -gekoppelten Differentialgleichungen (DGLs) erster Ordnung der Form

$$\dot{y}_1(t) = \frac{dy_1}{dt} = f_1(t, y_1, y_2, \dots, y_m)$$

$$\dot{y}_2(t) = \frac{dy_2}{dt} = f_2(t, y_1, y_2, \dots, y_m)$$

$$\dot{y}_3(t) = \dots =$$

Replikatorodynamik (2xM)-Spiele

Wir beschränken uns zunächst auf symmetrische (2xM)-Spiele, d.h. zwei Personen - M Strategien Spiele. Da es sich um symmetrische Spiele handelt, sind alle Spieler gleichberechtigt und man kann von einer homogenen Population ausgehen. Die Differentialgleichung der Replikatorodynamik beschreibt wie sich die einzelnen Populationsanteile der zur Zeit t gewählten Strategien $x_j(t)$, $j=1,2,\dots,M$ im Laufe der Zeit entwickeln.

$$\dot{x}_j(t) := \frac{dx_j(t)}{dt} = x_j(t) \cdot \left[\underbrace{\sum_{k=1}^M \$_{jk} \cdot x_k(t)}_{\text{Fitness der Strategie } j} - \underbrace{\sum_{l=1}^M \sum_{k=1}^M \$_{kl} \cdot x_k(t) \cdot x_l(t)}_{\text{Durschnittliche Fitness (Auszahlung) der gesamten Population}} \right]$$

Wobei die Parameter $\$_{kl}$ die einzelnen Einträge in der Auszahlungsmatrix des 1. Spielers darstellen

$$\hat{\$} = \hat{\$}^1 = \begin{pmatrix} \$_{11} & \$_{12} & \$_{13} & \dots & \$_{1M} \\ \$_{21} & \$_{22} & \$_{23} & \dots & \$_{2M} \\ \$_{31} & \$_{32} & \$_{33} & \dots & \$_{3M} \\ \dots & \dots & \dots & \dots & \dots \\ \$_{M1} & \$_{M2} & \$_{M3} & \dots & \$_{MM} \end{pmatrix}$$

Fitness der Strategie j
Durchschnittlicher Erfolg der j-ten Strategie

Durschnittliche Fitness (Auszahlung) der gesamten Population

Replikatorodynamik

(für symmetrische (2x3)-Spiele)

Wir beschränken uns nun auf symmetrische (2x3)-Spiele, d.h. zwei Personen - 3 Strategien Spiele (M=3). Die Differentialgleichung der Replikatorodynamik vereinfacht sich unter dieser Annahme wie folgt:

$$\frac{dx_j(t)}{dt} = x_j(t) \cdot \left[\sum_{k=1}^3 \$_{jk} \cdot x_k(t) - \sum_{l=1}^3 \sum_{k=1}^3 \$_{kl} \cdot x_k(t) \cdot x_l(t) \right]$$

$$\frac{dx_j}{dt} = x_j \cdot \left[\begin{array}{l} \$_{j1} \cdot x_1 + \$_{j2} \cdot x_2 + \$_{j3} \cdot x_3 - \\ \underbrace{\left(\begin{array}{l} \$_{11} \cdot x_1 \cdot x_1 + \$_{12} \cdot x_1 \cdot x_2 + \$_{13} \cdot x_1 \cdot x_3 + \\ + \$_{21} \cdot x_2 \cdot x_1 + \$_{22} \cdot x_2 \cdot x_2 + \$_{23} \cdot x_2 \cdot x_3 + \\ + \$_{31} \cdot x_3 \cdot x_1 + \$_{32} \cdot x_3 \cdot x_2 + \$_{33} \cdot x_3 \cdot x_3 \end{array} \right)}_{\overline{\$}} \end{array} \right]$$

$j = 1, 2, 3$

Replikatorodynamik

(für symmetrische (2x3)-Spiele)

Man erhält ein System von drei gekoppelten Differentialgleichungen:

$$\frac{dx_1}{dt} = x_1 \cdot [\$_{11} \cdot x_1 + \$_{12} \cdot x_2 + \$_{13} \cdot x_3 - \bar{\$}]$$

$$\frac{dx_2}{dt} = x_2 \cdot [\$_{21} \cdot x_1 + \$_{22} \cdot x_2 + \$_{23} \cdot x_3 - \bar{\$}]$$

$$\frac{dx_3}{dt} = x_3 \cdot [\$_{31} \cdot x_1 + \$_{32} \cdot x_2 + \$_{33} \cdot x_3 - \bar{\$}]$$

Das System von Differentialgleichungen lässt sich bei gegebener Auszahlungsmatrix $\hat{\$}$ und Anfangsbedingung $(x_1(0), x_2(0), x_3(0))$ meist nur numerisch (auf dem Computer) lösen. Die Lösungen bestehen dann aus den drei (zeitlich abhängigen) Populationsanteilen $(x_1(t), x_2(t), x_3(t))$.

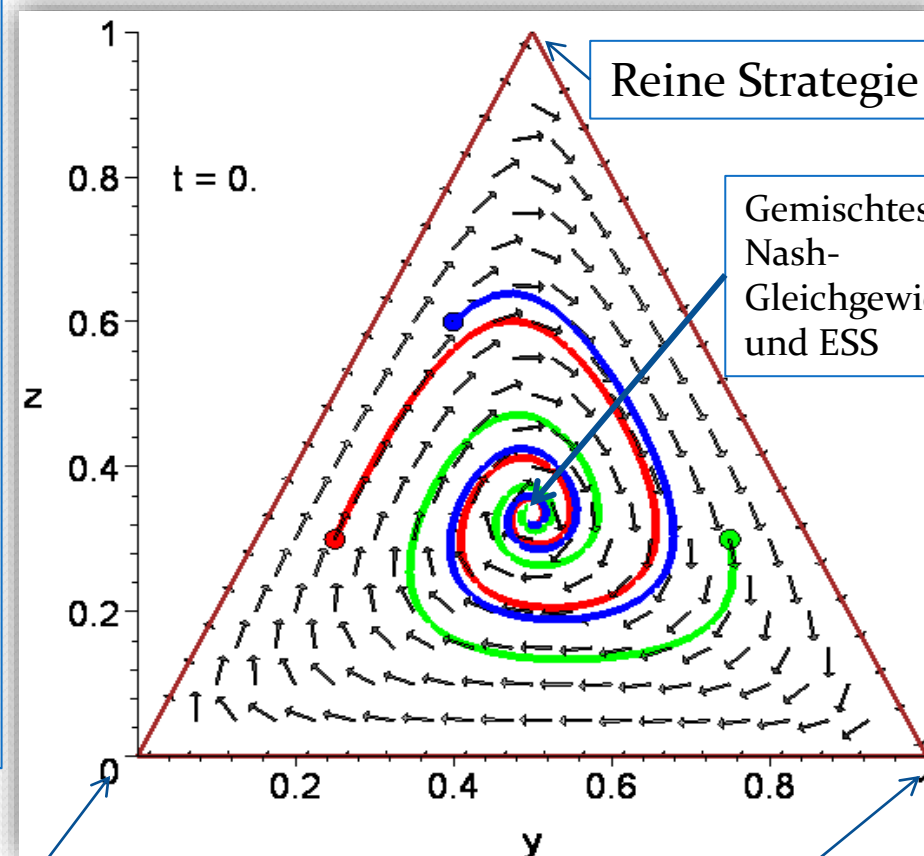
Replikatorodynamik

(für symmetrische (2x3)-Spiele, **Beispiel 1**)

Wir betrachten im Folgenden ein Beispiel eines (2x3)-Spiels mit der rechts angegebenen Auszahlungsstruktur:

	Strategie 1	Strategie 2	Strategie 3
Strategie 1	(0, 0)	(2, -1)	(-1, 2)
Strategie 2	(-1, 2)	(0, 0)	(2, -1)
Strategie 3	(2, -1)	(-1, 2)	(0, 0)

Die rechte Abbildung zeigt die zeitliche Entwicklung der relativen Populationsanteile der gewählten Strategien für drei mögliche Anfangsbedingungen. Die einzige evolutionär stabile Strategie dieses Beispiels befindet sich beim gemischten Nash-Gleichgewicht $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$. Die einzelnen Pfeile im Dreieck veranschaulichen den durch die Spielmatrix bestimmten Strategien-„Richtungswind“, dem die Population zeitlich folgen wird.



Zur Visualisierung der evolutionären Entwicklung benutzt man oft die sogen. barycentric coordinates:

$$y := x_2 + \frac{x_3}{2}$$

$$z := x_3$$

Reine Strategie 1

Reine Strategie 2

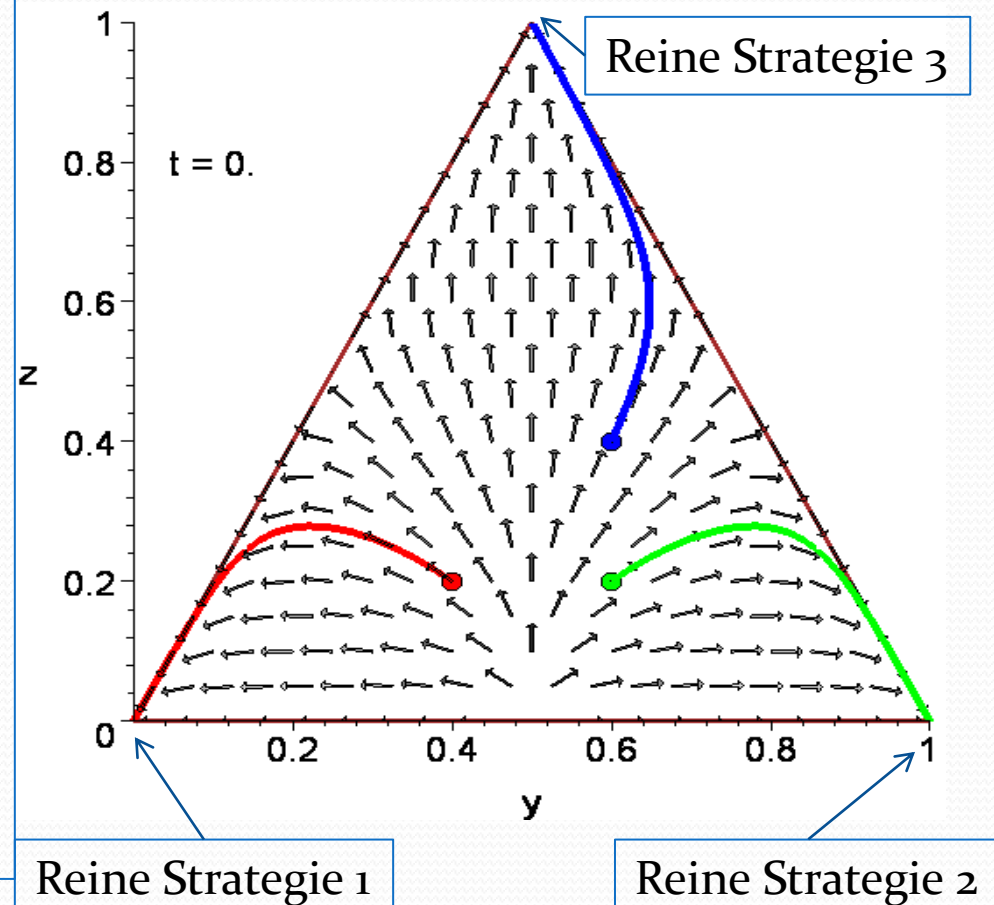
Replikatorodynamik

(für symmetrische (2x3)-Spiele, **Beispiel 2**)

Wir betrachten im Folgenden ein Beispiel eines (2x3)-Spiels mit der rechts angegebenen Auszahlungsstruktur:

	Strategie 1	Strategie 2	Strategie 3
Strategie 1	(0, 0)	(-3, -3)	(-1, -1)
Strategie 2	(-3, -3)	(0, 0)	(-1, -1)
Strategie 3	(-1, -1)	(-1, -1)	(0, 0)

Die rechte Abbildung zeigt die zeitliche Entwicklung der relativen Populationsanteile der gewählten Strategien für drei mögliche Anfangsbedingungen. Das Spiel besitzt drei Nash-Gleichgewichte in reinen Strategien, die ebenfalls evolutionär stabile Strategien darstellen. Welche der drei ESS die Population realisiert hängt von dem Anfangswert der Populationsanteile ab. Die zeitliche Entwicklung folgt wieder dem Strategien-„Richtungswind“ der zugrundeliegenden Auszahlungsmatrix.

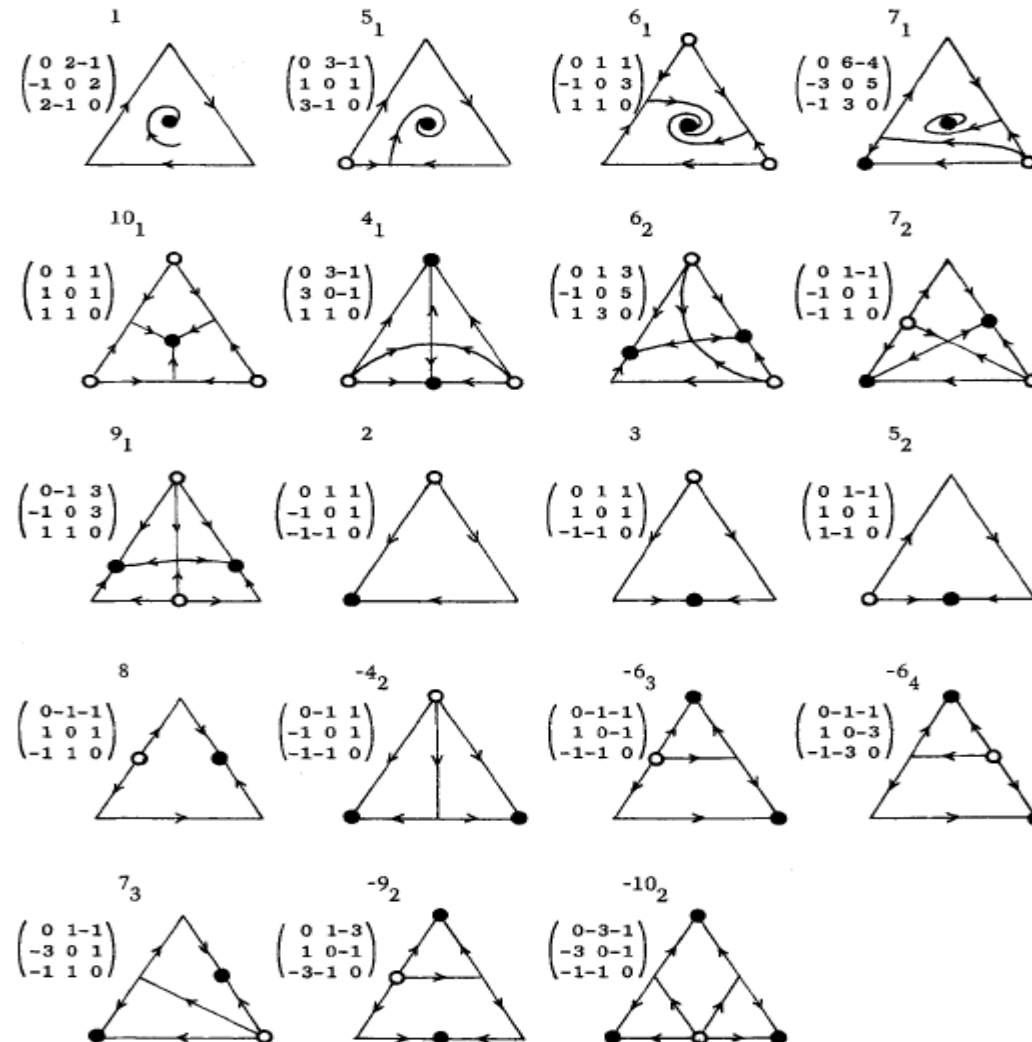


Replikatorodynamik

(Klassifizierung symmetrische (2x3)-Spiele)

E. C. Zeeman, *POPULATION DYNAMICS FROM GAME THEORY*,
In: *Global Theory of Dynamical Systems*, Springer 1980

E. C. Zeeman zeigt in seinem im Jahre 1980 veröffentlichten Artikel, dass man evolutionäre, symmetrische (2x3)-Spiele in 19 Klassen einteilen kann. Die Abbildung rechts zeigt das evolutionäre Verhalten dieser 19 Spieltypen. Die ausgefüllten schwarzen Punkte markieren die evolutionär stabilen Strategien der jeweiligen Spiele. Es gibt Spielklassen, die besitzen lediglich eine ESS und Klassen die sogar drei ESS besitzen.



Jupyter Notebook Evolutionenspiel4.ipynb

Physik der sozio-ökonomischen Systeme mit dem Computer

(Physics of Socio-Economic Systems with the Computer)

Vorlesung gehalten an der J.W.Goethe-Universität in Frankfurt am Main

(Wintersemester 2020/21)

von Dr.phil.nat. Dr.rer.pol. Matthias Hanauke

Frankfurt am Main 01.12.2020

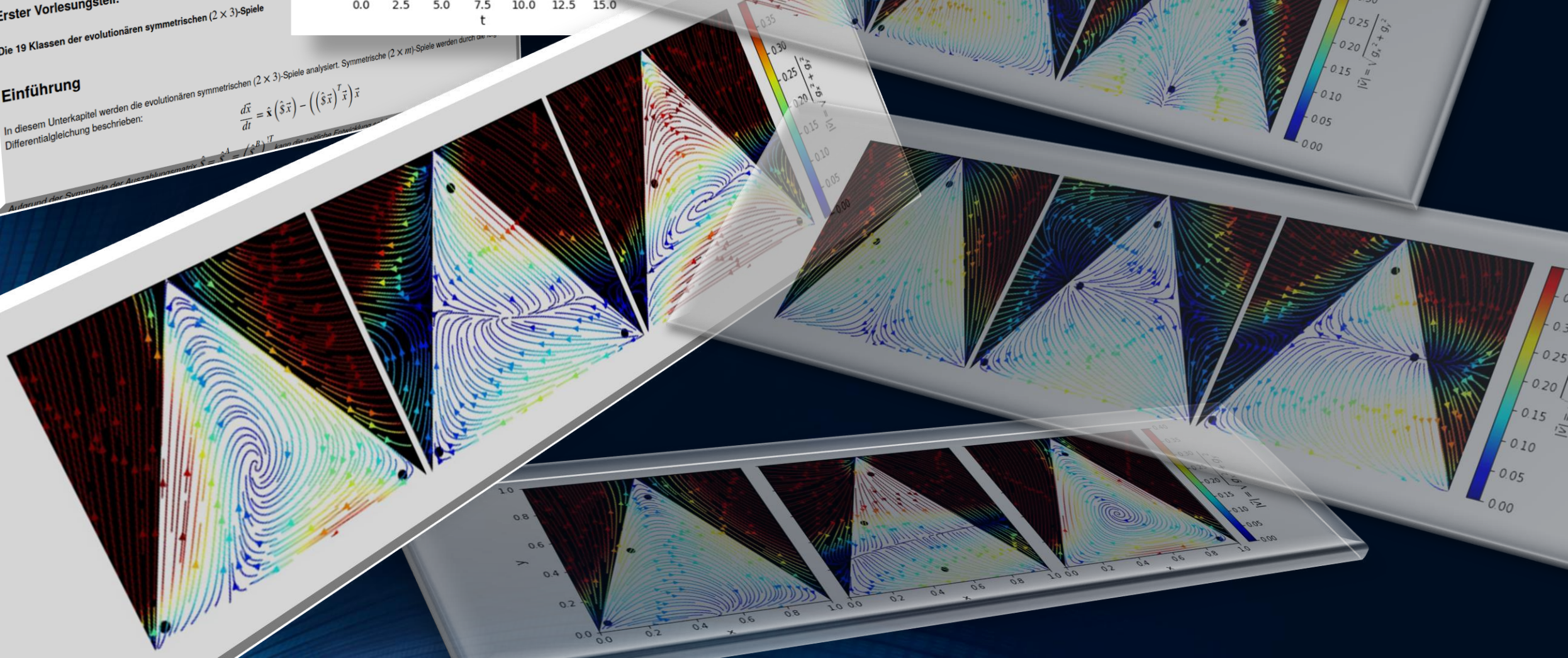
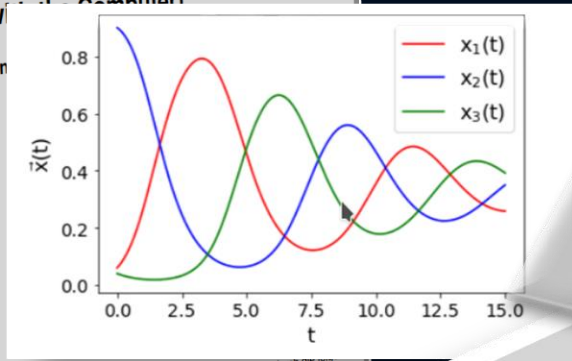
Erster Vorlesungsteil:

Die 19 Klassen der evolutionären symmetrischen (2×3) -Spiele

Einführung

In diesem Unterkapitel werden die evolutionären symmetrischen (2×3) -Spiele analysiert. Symmetrische $(2 \times m)$ -Spiele werden durch die Differentialgleichung beschrieben:

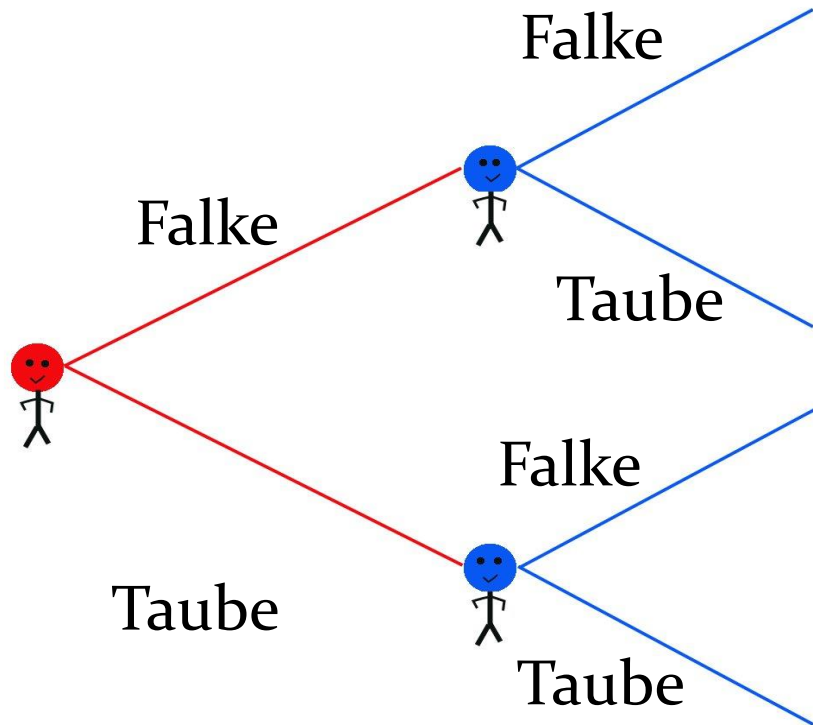
$$\frac{d\vec{x}}{dt} = \vec{x}(\hat{s}\vec{x}) - \left((\hat{s}\vec{x})^T \vec{x} \right) \vec{x}$$



Anwendungsfelder Spieltheorie

- Anwendungsfelder in den Wirtschafts- Sozialwissenschaften und Biologie
 - Experimentelle Ökonomie
 - Die Finanzkrise als Falke-Taube Spiel
 - Die Entstehung einer dritten Strategie im Elfmeter-Spiel (Nesken Effekt)
 - Evolutionäre Entwicklung einer Eidechsen Population als symmetrisches (2x3)-Spiel
 - Das Räuber-Beute Spiel und die Lotka-Volterra-Gleichung
 - Die Klimakrise als Populationsdilemma

Das Falke-Taube Spiel



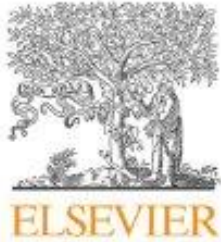
	Falke	Taube
Falke	$\left(\frac{(p_h-d)}{2}, \frac{(p_h-d)}{2}\right)$	$(p_h, 0)$
Taube	$(0, p_h)$	$\left(\frac{p_m}{2}, \frac{p_m}{2}\right)$

Das Falke-Taube-Spiel modelliert ursprünglich den Wettkampf um eine Ressource (z.B. Nistplatz). Das Spiel wird jedoch oft auch auf andere Systeme angewendet, wobei die Taube-Strategie eine friedliche Verhaltensweise symbolisiert und die Falke-Strategie ein aggressives Verhalten. Im folgenden Artikel wird das Falke-Taube-Spiel auf den Immobilien-Investmentmarkt angewendet (Spieler-Population: Investmentbanker).

Parameter setting	Risk of destabilisation	d	p_h	p_m
P1	LOW	6	5	3
P2	MEDIUM	10	5	3
P3	HIGH	20	5	3

TABLE II: Parameters of the three different sets of the underlying payoff matrix used to model the investment market of the Hawk-Dove game.

TABLE I: Payoff matrix for investment bankers A and B within the Hawk-Dove game. The parameters are defined as follows: p_h : high selling premium, d : disutility resulting from fighting and p_m : moderate selling premium.



Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa



Doves and hawks in economics revisited: An evolutionary quantum game theory based analysis of financial crises

Matthias Hanauske^{a,*}, Jennifer Kunz^b, Steffen Bernius^a, Wolfgang König^c

^a Institute of Information Systems, Goethe-University, Grüneburgplatz 1, 60323 Frankfurt/Main, Germany

^b Chair of Controlling & Auditing, Goethe-University, Grüneburgplatz 1, 60323 Frankfurt/Main, Germany

^c House of Finance, Goethe-University, Grüneburgplatz 1, 60323 Frankfurt/Main, Germany

ARTICLE INFO

Article history:

Received 14 April 2009

Received in revised form 22 April 2010

Available online 15 June 2010

Keywords:

Evolutionary game theory

Quantum game theory

Hawk–dove game

Financial crisis

ABSTRACT

The last financial and economic crisis demonstrated the dysfunctional long-term effects of aggressive behaviour in financial markets. Yet, evolutionary game theory predicts that under the condition of strategic dependence a certain degree of aggressive behaviour remains within a given population of agents. However, as a consequence of the financial crisis, it would be desirable to change the “rules of the game” in a way that prevents the occurrence of any aggressive behaviour and thereby also the danger of market crashes. The paper picks up this aspect. Through the extension of the well-known hawk–dove game by a quantum approach, we can show that dependent on entanglement, evolutionary stable strategies also can emerge, which are not predicted by the classical evolutionary game theory and where the total economic population uses a non-aggressive quantum strategy.

© 2010 Elsevier B.V. All rights reserved.

Wie entwickelt sich der Populationsvektor $x(t)$ der Investmentbanker im Laufe der Zeit?

Benutzen Sie hierbei die drei unterschiedlichen Parametersets der vorigen Folie.

Das Spiel der Geldpolitik

Fiskalbehörde Geldbehörde	Keine neuen Schulden	Weiter Schulden machen
Finanzierung des Staatsdefizits über inflationäre Geldschöpfung	(3, 3)	(2, 4)
Stabile Geldpolitik	(4, 2)	(0, 0)

Eine nationale, oder auch europäische Geldpolitik ist stets in einem fiskalpolitischen Diskurs. Die Geldbehörde (Zentralbank), die z.B. durch eine Verknappung der Geldmenge (kontraktive/restriktive) Geldpolitik bzw. eine Ausdehnung der Geldmenge (expansive Geldpolitik), eine stabile bzw. un stabile Strategie wählen kann, ist bestrebt ihre geldpolitischen Ziele (z.B. Preisniveaustabilität) durchzusetzen. Sowohl die Entscheidungsträger der Geldpolitik als auch die Politiker, welche eine fiskalpolitische Entscheidungen zu treffen haben, befinden sich in einem wiederholten Spiel. Laut Gerhard Illing (Theorie der Geldpolitik, Kapitel 10.2) ist das gesamte geldpolitische Spiel, in erster Näherung, wie in der obigen Spielmatrix zu approximieren. Zusätzlich wirkt das globale Finanznetzwerk, zusammengesetzt (unter anderem) aus einer Vielzahl von Spekulanten, auf die Regierung ein, indem sie durch spekulativen Devisenhandel Währungskurse attackieren. Näheres siehe: Hochschul-Sommerkurses 2011, „Money, Money, Money: Deutschlands Wirtschafts- und Finanzleben“ (https://itp.uni-frankfurt.de/~hanauske/new/HSK_2011/index.html)

Das Spiel der Geldpolitik

Fiskalbehörde Geldbehörde	Keine neuen Schulden	Weiter Schulden machen
Finanzierung des Staatsdefizits über inflationäre Geldschöpfung	(3, 3)	(2, 4)
Stabile Geldpolitik	(4, 2)	(0, 0)

Obwohl hier eine symmetrische Spielmatrix vorliegt, ist das zugrundeliegende Spiel als Bi-Matrix Spiel zu beschreiben. In welche Klasse von Bi-Matrix Spielen ist das Spiel einzuordnen? Beschreiben Sie die möglichen zeitlichen Entwicklungen.

Benutzen Sie hierbei das folgende Maple oder Python Programm:

- 1) Bi-Matrix Spiele (Maple): <https://itp.uni-frankfurt.de/~hاناuske/VPSOC/T1/maple/I-2-4/BiMatrix1.html>
- 2) Bi-Matrix Spiele (Python): <https://itp.uni-frankfurt.de/~hاناuske/VPSOC/T2/python/bimatrix1.py>

Anwendungsfelder der Spieltheorie (I)

- **Biologie**

- **Verteilung von Bakterien in Organismen**

Siehe z.B.: Kerr, Feldmann, Nature 2002

- **Kooperation von Virus-Populationen**

Siehe z.B.: Turner, Chao, Nature 1999

- **Paarungsstrategien von Eidechsen**

Siehe z.B.: Sinervo, Hazard, Nature 1996

- **Evolutionäre Entwicklung von Makromolekülen**

Siehe z.B.: Eigen, Schuster, Naturwissenschaften 64, 1977

Evolutionäre Spieltheorie

Evolutionäre Entwicklung von biologischen Systemen

Quasispezies und die Fitness der Genom Sequenz

Viele der in diesem Unterkapitel behandelten Systeme sind dem Buch Martin A. Nowak, *Evolutionary Dynamics - Exploring the Equations of Life*, 2006 entnommen, welches eine sehr gute und allgemein verständliche Einführung in das Themengebiet der evolutionären Dynamik darstellt. Obwohl der Fokus dieses Buches im Bereich der Evolution von biologischen Systemen liegt (siehe Kapitel 10: HIV Infection, Kapitel 11: Evolution of Virulence, Kapitel 12: Evolutionary Dynamics of Cancer, und Kapitel 13: Language Evolution), sind die Kapitel 1-9 weitgehend allgemein formuliert. Die evolutionäre Dynamik unterschiedlicher Spezies einer Tierart und der Mechanismus wie Tierarten ineinander übergehen wurde von Charles Darwin bereits im Jahre 1840 beschrieben. Im Jahre 1973 stellte John Maynard Smith eine Verbindung zwischen den Populationsgleichungen der Biologie und der evolutionären Spieltheorie her. Das Konzept der *Quasi-Spezies* (Ensemble von ähnlichen Genomen Sequenzen (Erbgut eines Lebewesens) welches durch einen Prozess der Mutation und Selektion entstanden ist) wurde von Manfred Eigen und Peter Schuster entwickelt (siehe Kapitel 3.3: Martin A. Nowak, *Evolutionary Dynamics - Exploring the Equations of Life*). Die Struktur der Quasi-Spezies Differentialgleichung ist den Gleichungen der evolutionären Spieltheorie sehr ähnlich (siehe Bild 3.4 und 4.5: Martin A. Nowak, *Evolutionary Dynamics - Exploring the Equations of Life*). Die evolutionäre Vorteilhaftigkeit einer Genom Sequenz wird hierbei als die *Fitness* der Quasi-Spezies bezeichnet. *Quasi-Spezies* entsprechen den Strategien der Spieltheorie und die Fitness kann als der Auszahlungswert einer Strategie aufgefasst werden. Die zeitliche Entwicklung der *Quasi-Spezies* am Beispiel des Paarungsverhalten von Eidechsen wird z.B. in siehe Sinervo, Barry, and Curt M. Lively. 'The rock-paper-scissors game and the evolution of alternative male strategies.' *Nature* 380.6571 (1996): 240. analysiert (siehe auch Vorlesung 6). Die evolutionäre Dynamik hängt von der unterliegenden Netzwerkstruktur der beteiligten Akteure ab und skalenfreie Netzwerkstrukturen agieren hier als Verstärker der evolutionären Selektion (siehe Kapitel 8, Evolutionary Graph Theory: Martin A. Nowak, *Evolutionary Dynamics - Exploring the Equations of Life*). Im folgenden Unterpunkt werden sie sogenannten *Spatial Games* behandelt (eine ausführliche Einführung findet sich im Kapitel 9: Martin A. Nowak, *Evolutionary Dynamics - Exploring the Equations of Life*).

Siehe Teil III der Vorlesung

III.3 Anwendungsfelder der Spieltheorie

III.3.1 Die Wissenschaft als komplexes Netzwerk (Models of Science Dynamics)

III.3.2 Sozio-ökonomische Labor- und Feldexperimente

III.3.3 Anwendungen in der Biologie

III.3.4 Anwendungen in den Politikwissenschaften

III.3.5 Spieltheorie und Auktionskonzepte

III.3.6 Finanzkrisen und evolutionäre Spiele

III.3.7 Sozio-ökonomische Netzwerke

Beispiel

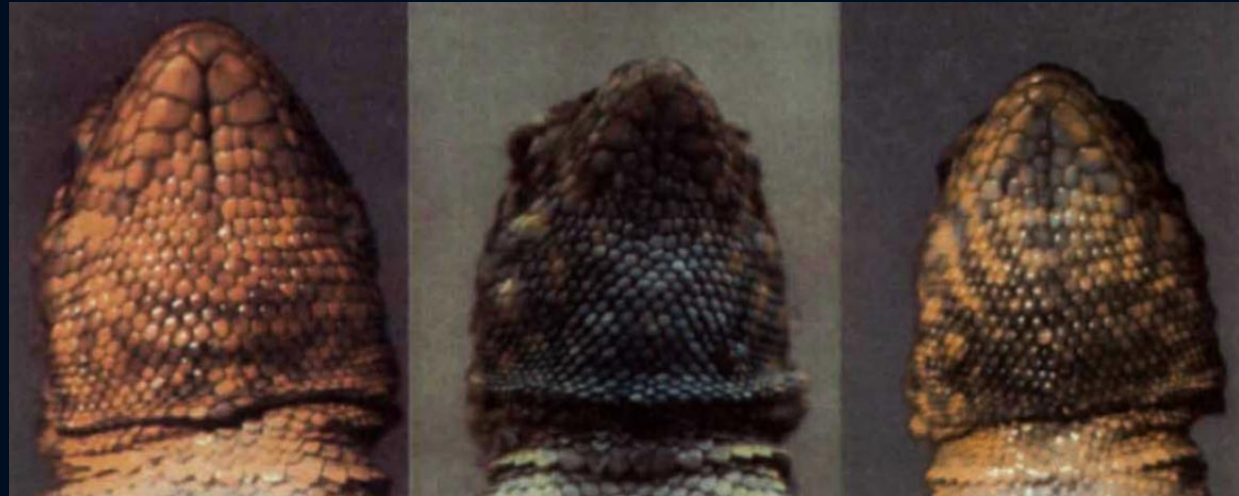
The rock–paper–scissors game and the evolution of alternative male strategies

B. Sinervo & C. M. Lively

Department of Biology and Center for the Integrative Study of Animal Behavior, Indiana University, Bloomington, Indiana 47405, USA



Evolutionäre
Strategie
(Quasi-Spezies)



Orange

Blau

Gelb

The Rock-Siccor-Paper Game Replicatordynamics and ESS

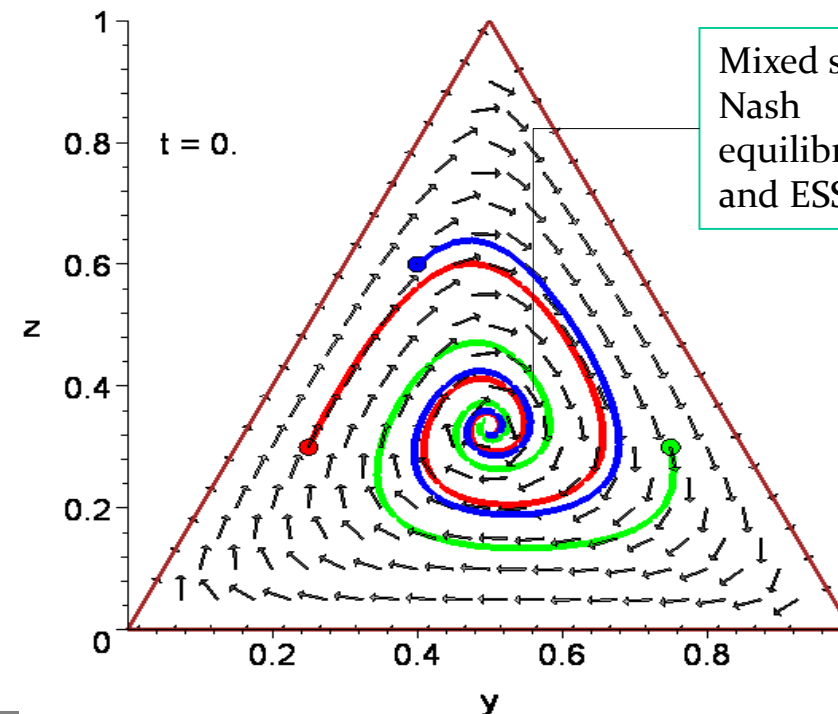
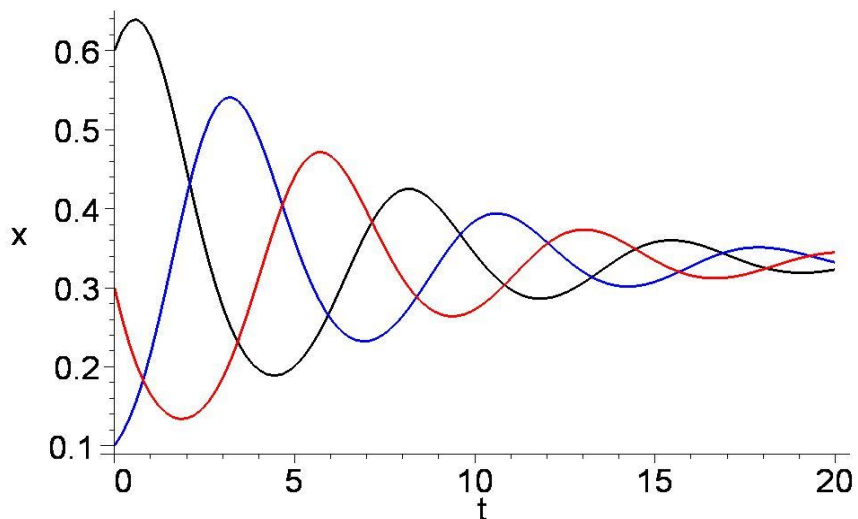
$$\frac{dx_1}{dt} = x_1 \cdot [2 \cdot x_2 - x_3 - \bar{\$}]$$

$$\frac{dx_2}{dt} = x_2 \cdot [-x_1 + 2 \cdot x_3 - \bar{\$}]$$

$$\frac{dx_3}{dt} = x_3 \cdot [2 \cdot x_1 - x_2 - \bar{\$}]$$

with: $\bar{\$} = x_1 \cdot x_2 + x_1 \cdot x_3 + x_2 \cdot x_3$

	Strategie 1	Strategie 2	Strategie 3
Strategie 1	(0, 0)	(1, -1)	(-1, 1)
Strategie 2	(-1, 1)	(0, 0)	(1, -1)
Strategie 3	(1, -1)	(-1, 1)	(0, 0)



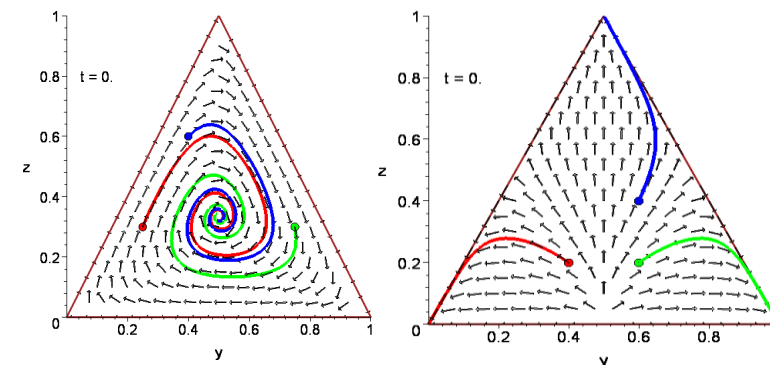
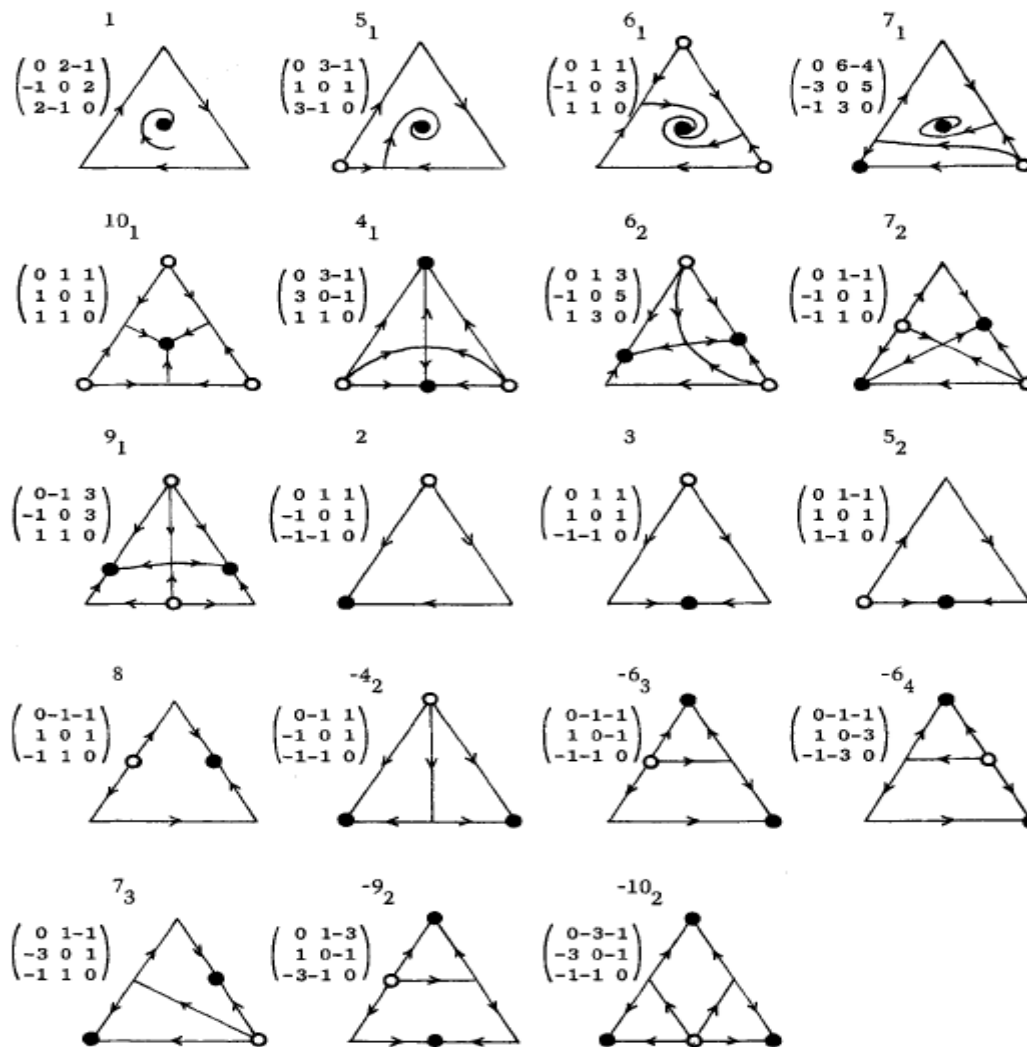
Mixed strategy
Nash
equilibrium
and ESS

Using
barycentric
coordinates:

$$y := x_2 + \frac{x_3}{2}$$

$$z := x_3$$

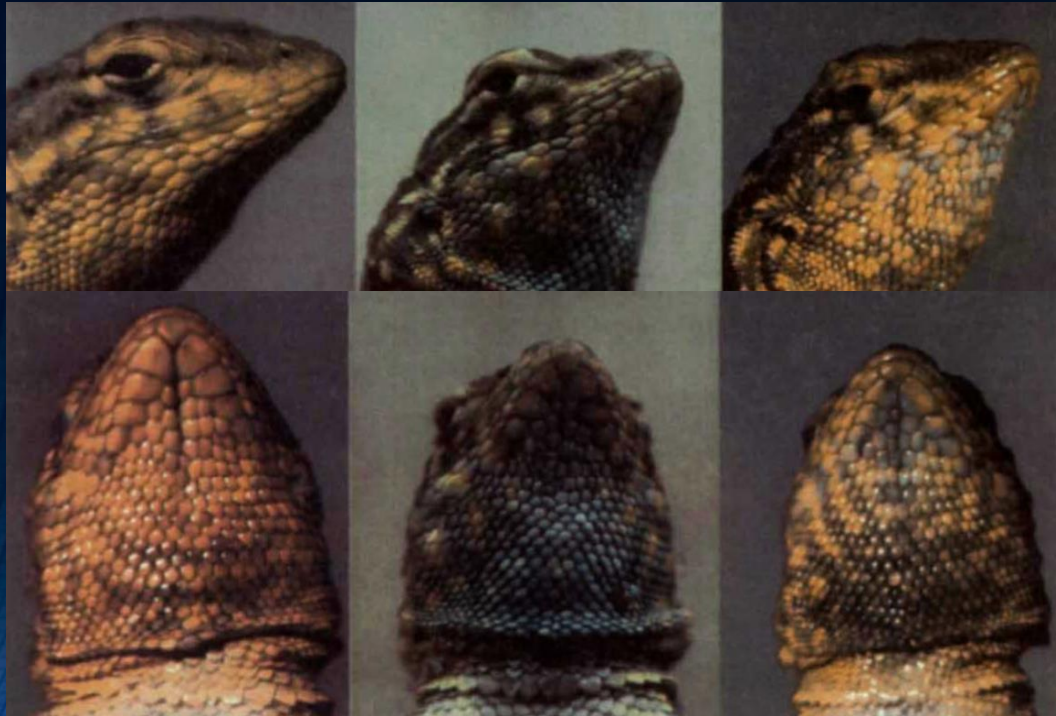
Classes of symmetric (2x3) games



FE. C. Zeeman proved in his article that one can categorize symmetric evolutionary (2x3) games into 19 different classes. The figure on the left side shows that some classes have only one ESS (filled black circles), while others can have three ESSs.

E. C. Zeeman, *POPULATION DYNAMICS FROM GAME THEORY*,
In: *Global Theory of Dynamical Systems*, Springer 1980

The Rock-Papers-Scissors Game and the Evolution of Sexual Selection



B.Sinervo and C.M.Lively focus within their article (The rock-paper-scissors game and the evolution of alternative male strategies, *Nature*, Vol.380 (1996)) on the sexual selection of male side-blotched lizards. From 1990-1995 they studied experimentally these animals and proposed an evolutionary model to explain their data.

Male fitness:

Number of monopolized + shared females

The male lizards have substantially three different colors, which are strongly connected to their behavior: **Orange** (very aggressive, defend large territories), **Blue** (less aggressive, defend small territories), **Yellow** (sneakers, look like females, not aggressive, do not defend territories). The payoff for the male lizards (their fitness) was estimated by the number of monopolized females (exclusively on his home range) and shared females (overlap to other territories),
51.64,11

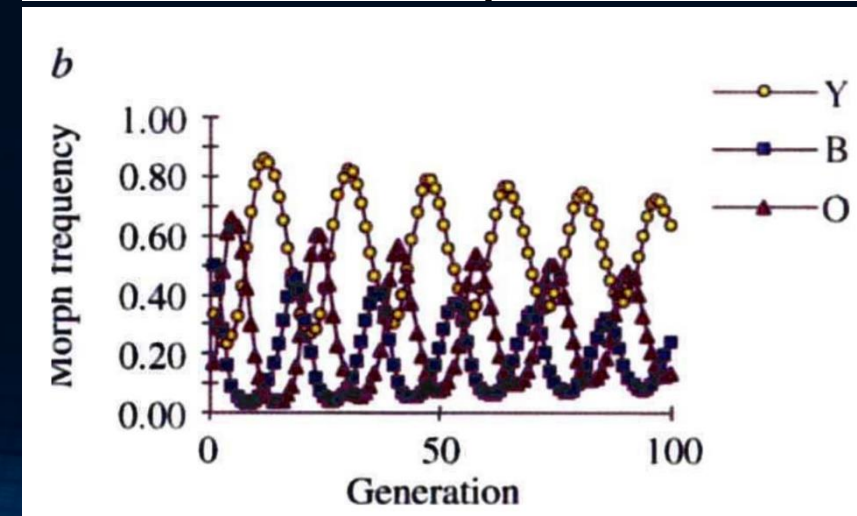
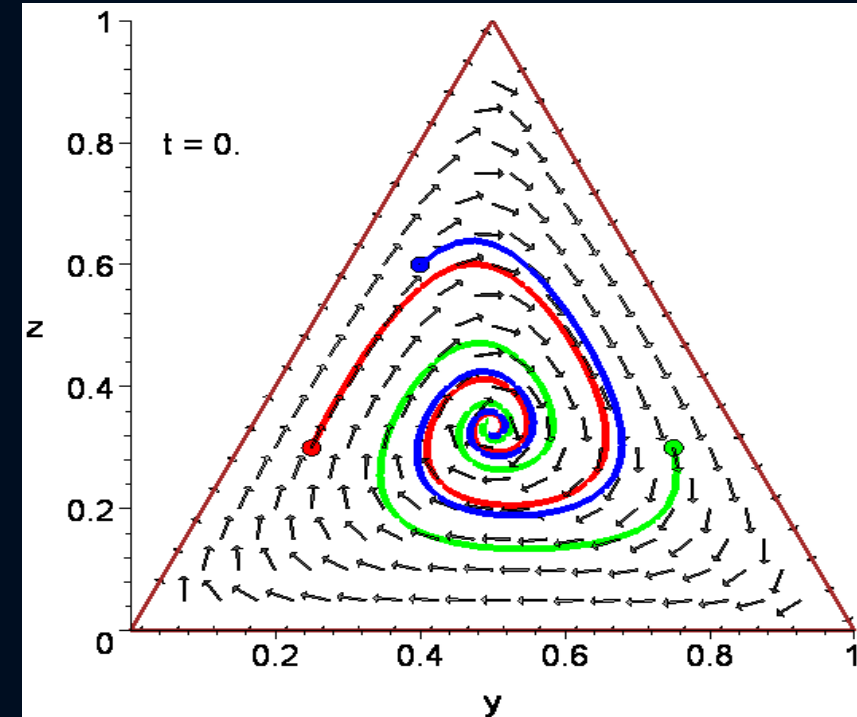
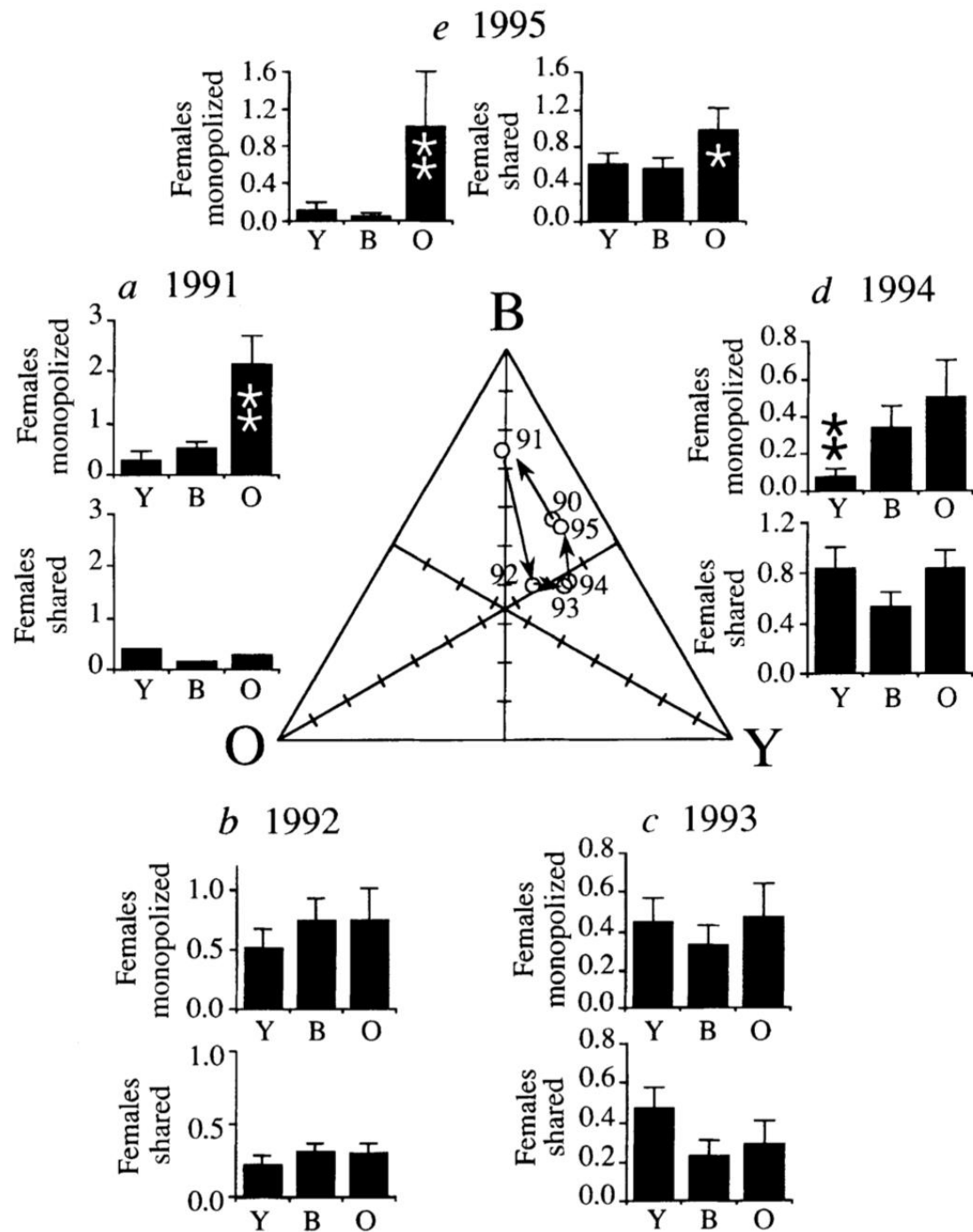
The Rock-Papers-Scissors Game and the Evolution of Sexual Selection

MANY species exhibit colour polymorphisms associated with alternative male reproductive strategies, including territorial males and ‘sneaker males’ that behave and look like females¹⁻³. The prevalence of multiple morphs is a challenge to evolutionary theory because a single strategy should prevail unless morphs have exactly equal fitness^{4,5} or a fitness advantage when rare^{6,7}. We report here the application of an evolutionary stable strategy model to a three-morph mating system in the side-blotched lizard. Using parameter estimates from field data, the model predicted oscillations in morph frequency, and the frequencies of the three male morphs were found to oscillate over a six-year period in the field. The fitnesses of each morph relative to other morphs were non-transitive in that each morph could invade another morph when rare, but was itself invadable by another morph when common. Concordance between frequency-dependent selection and the among-year changes in morph fitnesses suggest that male interactions drive a dynamic ‘rock–paper–scissors’ game⁷.



We have described the first biological example of a cyclical ‘Rock–paper–scissors’ game⁷. As in the game where paper beats rock, scissors beat paper, and rock beats scissors, the wide-ranging ‘ultradominant’ strategy of orange males is defeated by the ‘sneaker’ strategy of yellow males, which is in turn defeated by the mate-guarding strategy of blue males; the orange strategy defeats the blue strategy to complete the dynamic cycle. Frequency-dependent selection maintains substantial genetic variation in alternative male strategies, while at the same time prohibiting a stable equilibrium in morph frequency. □

The Rock-Papers-Scissors Game and the Evolution of Sexual Selection



Anwendungsfelder der Spieltheorie (II)

- **Ökonomie**

- **„Public Goods“- (Öffentliches Gut)- Spiele**

- **Trust in Private and Common Property Experiments**, Elinor Ostrom, et al.
 - **Evolutionary Dynamics in Public Good Games**, CHRISTIANE CLEMENS and THOMAS RIECHMANN, Computational Economics (2006) 28: 399–420
 - **Institution Formation in Public Goods Games**, Michael Kosfeld, Akira Okada, and Arno Riedl, American Economic Review 2009, 99:4, 1335–1355

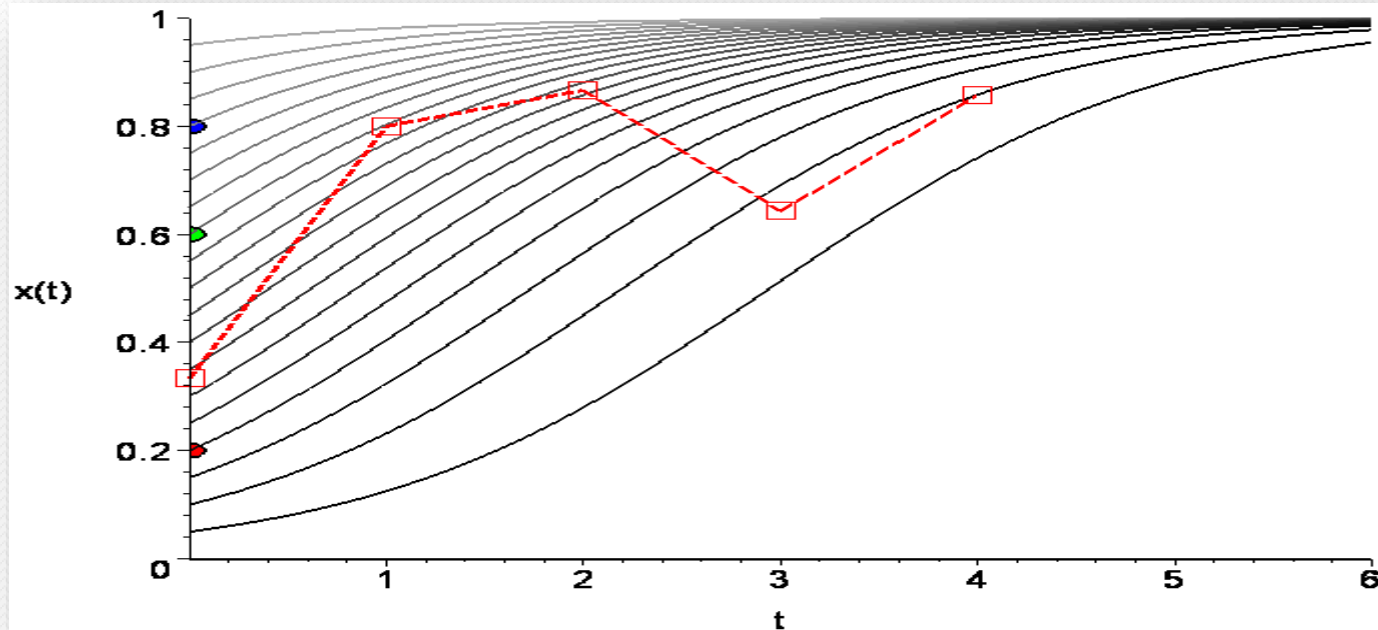
- **Experimentelle Ökonomie**

- **Cooperation in PD games: Fear, greed, and history of play**, T.K. AHN, ELINOR OSTROM, DAVID SCHMIDT, ROBERT SHUPP, Public Choice 106: 137–155, 2001.
 - **„Behavioral“- Verhaltensökonomie (Altruismus, Empathie, ...)** z.B.: Fehr et al.
 - **Evolution von Informationsnetzwerken**

Theorie ↔ Experiment

Experimentelle Ergebnisse des in Lyon gespielten Beispiels 1

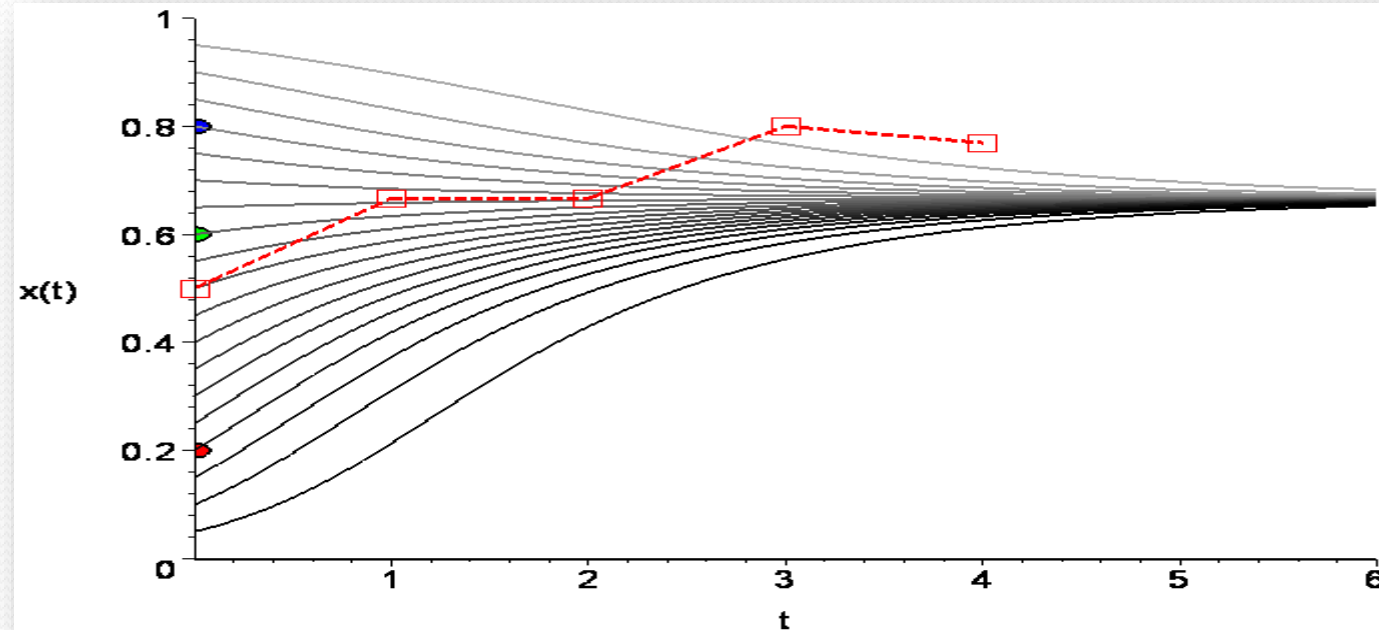
Experimentelle Ökonomie



Das erste Spiel besitzt nur ein Nash-Gleichgewicht das gleichzeitig die dominante Strategie des Spiels ist (Kugel, Kugel). Da es sich bei diesem Beispiel um ein dominantes, symmetrisches (2x2)-Spiel handelt und die Funktion $g(x)$ im relevanten Bereich ($x=[0,1]$) immer größer-gleich Null ist, strebt der Populationsanteil der Kugel-Spieler unabhängig vom Anfangswert immer gegen die evolutionär stabile Strategie $x=1$. Die klassische evolutionäre Spieltheorie sagt demnach voraus, dass die Spieler innerhalb der betrachteten Population nach einer gewissen Zeit maßgeblich die Strategie Kugel wählen ($x=1$). Die rote Kurve in der obigen Abbildung zeigt die experimentellen Ergebnisse des im Vorlesungsteil 4 gespielten Beispiels 1.

Theorie ↔ Experiment

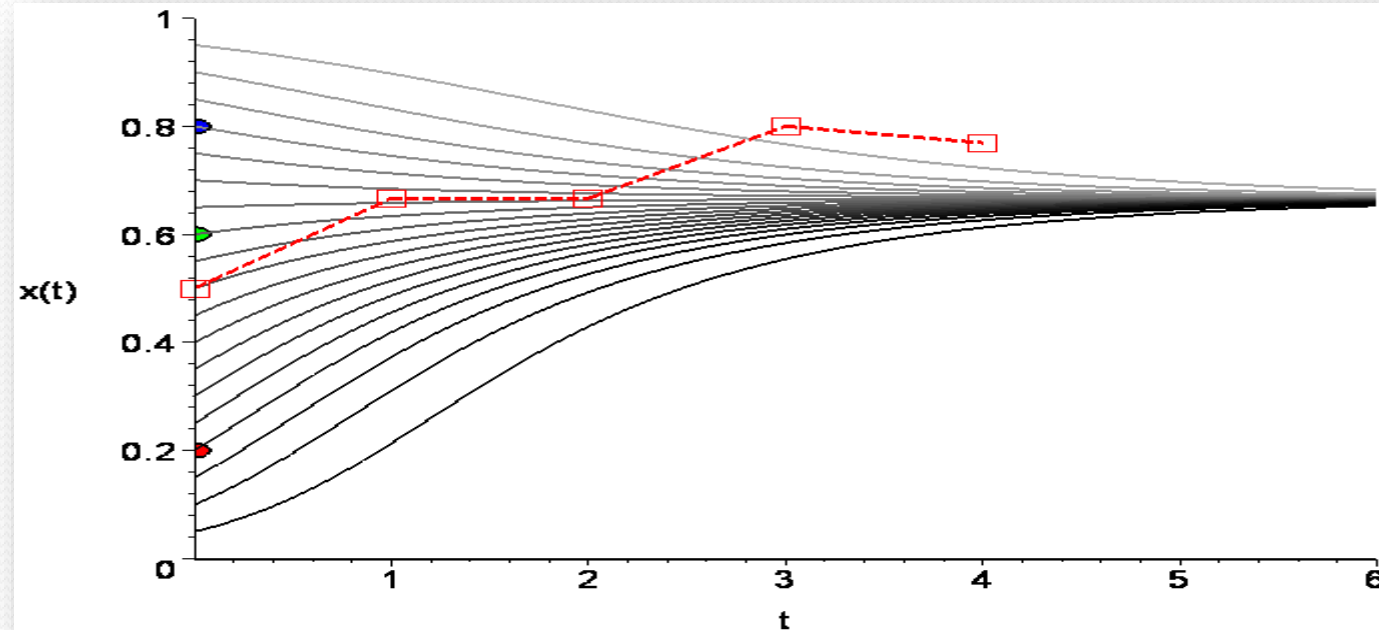
Experimentelle Ergebnisse des in Lyon gespielten Beispiels 2



Das zweite Spiel besitzt keine dominante Strategie, aber zwei unsymmetrische Nash-Gleichgewichte in reinen Strategien ((K, KK) und (KK, K)) und ein gemischtes Nash-Gleichgewicht (0.67 K , 0.33 KK). Da es sich bei diesem Beispiel um ein symmetrisches Anti-Koordinationsspiel handelt, strebt der Populationsanteil der Kugel-Spieler unabhängig vom Anfangswert immer zu dem gemischten Nash-Gleichgewicht (der einzigen evolutionär stabilen Strategie des Spiels), was identisch mit der mittleren Nullstelle der Funktion $g(x)$ ist ($x=0.67$). Die rote Kurve in der obigen Abbildung zeigt die experimentellen Ergebnisse des im Vorlesungsteil 4 gespielten Beispiels 2.

Theorie ↔ Experiment

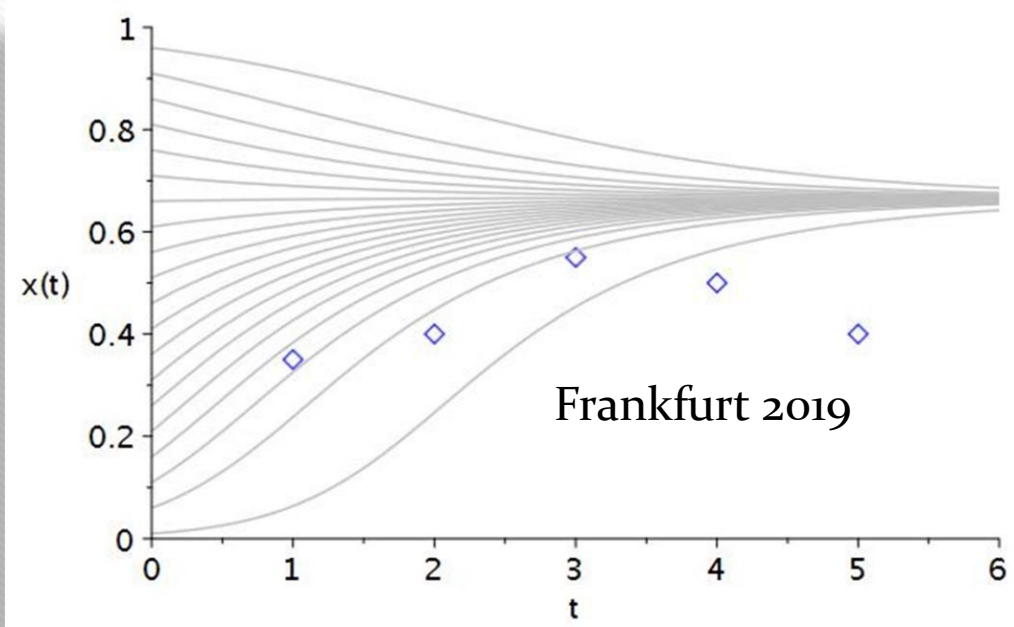
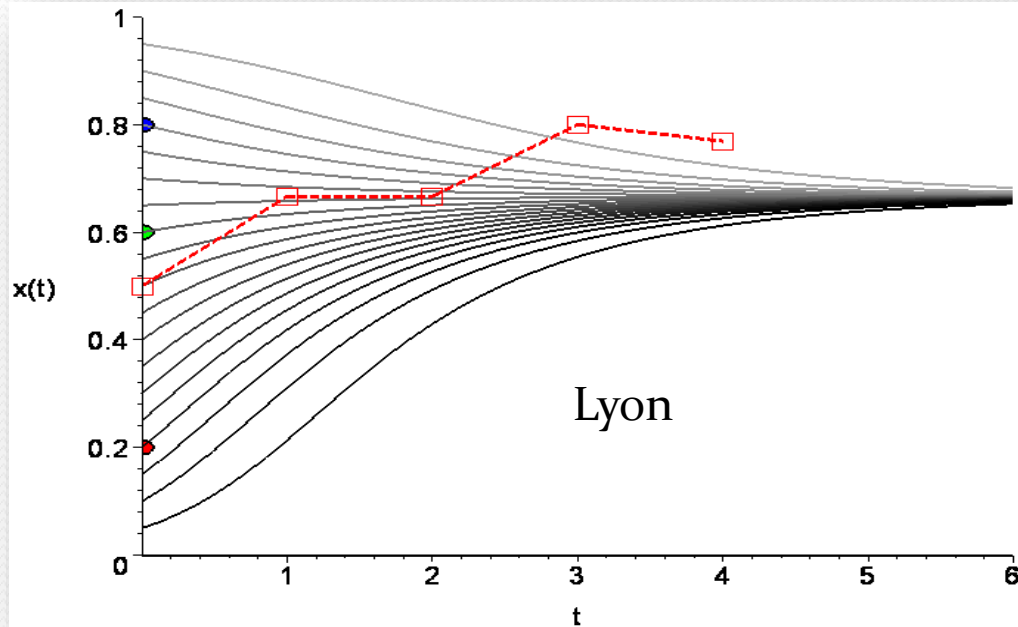
Experimentelle Ergebnisse des in Lyon gespielten Beispiels 2



Das zweite Spiel besitzt keine dominante Strategie, aber zwei unsymmetrische Nash-Gleichgewichte in reinen Strategien ((K, KK) und (KK, K)) und ein gemischtes Nash-Gleichgewicht (0.67 K , 0.33 KK). Da es sich bei diesem Beispiel um ein symmetrisches Anti-Koordinationsspiel handelt, strebt der Populationsanteil der Kugel-Spieler unabhängig vom Anfangswert immer zu dem gemischten Nash-Gleichgewicht (der einzigen evolutionär stabilen Strategie des Spiels), was identisch mit der mittleren Nullstelle der Funktion $g(x)$ ist ($x=0.67$). Die rote Kurve in der obigen Abbildung zeigt die experimentellen Ergebnisse des im Vorlesungsteil 4 gespielten Beispiels 2.

Theorie ↔ Experiment

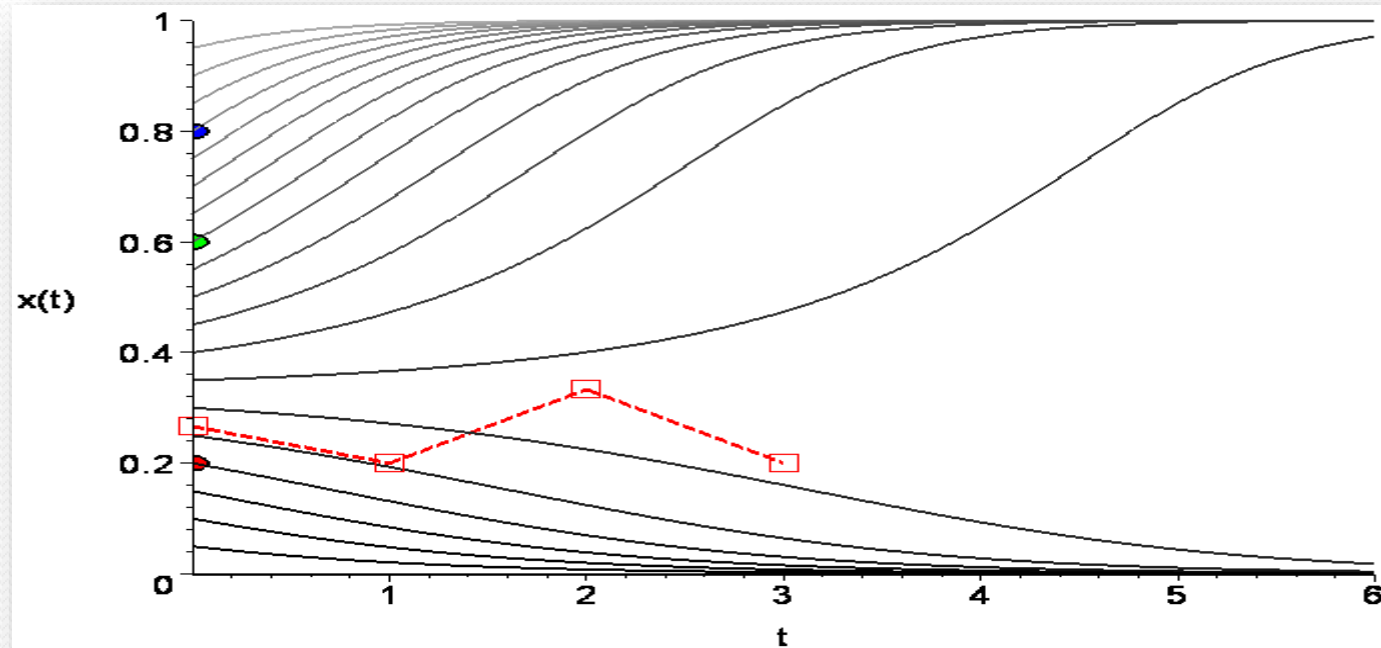
Experimentelle Ergebnisse des Beispiels 2



Das zweite Spiel besitzt keine dominante Strategie, aber zwei unsymmetrische Nash-Gleichgewicht in reinen Strategien ((K, KK) und (KK, K)) und ein gemischtes Nash-Gleichgewicht (0.67 K , 0.33 KK). Da es sich bei diesem Beispiel um ein symmetrisches Anti-Koordinationsspiel handelt, strebt der Populationsanteil der Kugel-Spieler unabhängig vom Anfangswert immer zu dem gemischten Nash-Gleichgewicht (der einzigen evolutionär stabilen Strategie des Spiels), was identisch mit der mittleren Nullstelle der Funktion $g(x)$ ist ($x=0.67$). Die rote Kurve in der linken obigen Abbildung zeigt die experimentellen Ergebnisse des in Lyon gespielten Beispiels 2. Die blauen Punkte in der rechten obigen Abbildung zeigt die experimentellen Ergebnisse des in Frankfurt gespielten Beispiels 2 (zum Vergleich bitte die blauen Punkte um -1 auf der Zeitachse verschieben).

Theorie ↔ Experiment

Experimentelle Ergebnisse des in Lyon gespielten Beispiels 3



Das dritte Spiel besitzt ebenfalls keine dominante Strategie, aber zwei symmetrische Nash-Gleichgewichte in reinen Strategien ((K,K) und (KK,KK)) und ein gemischtes Nash-Gleichgewicht (0.33 K , 0.67 KK). Da es sich bei diesem Beispiel um ein symmetrisches Koordinationsspiel handelt, strebt der Populationsanteil der Kugel-Spieler abhängig vom Anfangswert zu einem der beiden reinen Nash-Gleichgewichte ($x=1$ oder $x=0$). Die klassische evolutionäre Spieltheorie sagt demnach voraus, dass es zwei evolutionär stabile Strategien gibt ($x=1$ oder $x=0$). Die rote Kurve in der obigen Abbildung zeigt die experimentellen Ergebnisse des im Vorlesungsteil 4 gespielten Beispiels 3.

Anwendungsfelder der Spieltheorie (III)

- **Sozialwissenschaft**
 - **Kulturelle und moralische Entwicklungen**
 - **Evolution of social learning does not explain the origin of human cumulative culture**, Magnus Enquist, Stefano Ghirlanda, *Journal of Theoretical Biology* 246 (2007) 129–135
 - **EVOLUTION OF MORAL NORMS**, William Harms and Brian Skyrms, *For Oxford Handbook on the Philosophy of Biology* ed. Michael Ruse
 - **Evolution der Sprache**
 - **Finite populations choose an optimal language**, Christina Pawlowitsch, *Journal of Theoretical Biology* 249 (2007) 606–616
 - **Soziales Lernen**
 - **Evolution of social learning does not explain the origin of human cumulative culture**, Magnus Enquist, Stefano Ghirlanda, *Journal of Theoretical Biology* 246 (2007) 129–135
 - **Evolution von sozialen Normen**
 - **Collective Action and the Evolution of Social Norms**, Elinor Ostrom, *The Journal of Economic Perspectives*, Vol. 14, No. 3 (Summer, 2000), pp. 137-158
 - **Evolution von sozialen Netzwerken**
 - **GOVERNING SOCIAL-ECOLOGICAL SYSTEMS**, MARCO A. JANSSEN and ELINOR OSTROM
 - **A General Framework for Analyzing Sustainability of Social-Ecological Systems**, Elinor Ostrom, et al., *Science* 325, 419 (2009)

DPG Spring Meeting Berlin, March 25 - 30, 2012

SCOPE

- Financial Markets and Risk Management
- Economic Models and Evolutionary Game Theory
- Traffic Dynamics, Urban and Regional Systems
- Social Systems, Opinion and Group Dynamics
- Networks: From Topology to Dynamics

KEYNOTE TALK

H. Eugene Stanley
(Boston, USA)

“Interdependent Networks and Switching Phenomena”

YOUNG SCIENTIST AWARD FOR SOCIO- AND ECONOPHYSICS*

Keynote Speaker: **Stefan Thurner** (Wien, A)
“The Role of Agent Based Models in Understanding Human Societies”

* supported by d-fine



Registration via <http://berlin12.dpg-tagungen.de/index.html?lang=en>
Conference Languages: English and German

Deadline: December 1st 2011

Young Scientist Award: Call for nominations and applications at <http://www.dpg-physik.de/dpg/gliederung/fv/soe/YSA/call.html>

Deadline: December 1st 2011

CONTACT

Prof. Dr. Dirk Helbing, Dr. Jörg Reichardt and Dr. Tobias Preis,
Chairmen of the Physics of Socio-Economic Systems Division (Φ ·SOE), <http://www.phi-soe.de/>

TUTORIAL “Scientific Writing”**

Hernan Rozenfeld (APS, USA)
Tim Smith (IOP Publishing, UK)

INVITED TALKS

Thilo Gross (Bristol, UK) “Adaptive Networks of Opinion Formation in Humans and Animals”

Marc Hütt (Bremen) “Common Design Principles of Metabolic Networks and Industrial Production”

Focus SESSION: BIG DATA**

Rosario Mantegna (Palermo, IT)
“Econophysics and Social Research with Large Sets of Data”

Philip Treleaven (London, UK)
“Experimental Computational Finance & Big Data Environment”

Tiziana Di Matteo (London, UK)
“Embedding High Dimensional Data on Networks”

Michael Batty (London, UK)
“Cities and Complexity”

FOCUS SESSION: MODELS OF WAR, CONFLICT AND REVOLUTIONS

Neil Johnson (Miami, USA)
“Escalation, Timing and Severity of Insurgent and Terrorist Events: Robust Patterns and a Generic Model”

Aaron Clauset (Boulder, USA)
“Fatality Dynamics and the Limits of Civil and Interstate Wars”

Ravinder Bhavnani (Geneva, CH)
“Group Segregation and Urban Violence”

**Sessions are organized with the JDPG

Tutorial

SOE 1.1 Sun 16:00–18:00 HSZ 04 **Collective Dynamics of Firms: A Statistical Physics Approach** — ●FRANK SCHWEITZER

Focus Session: Swarm Intelligence

SOE 2.1 Mon 10:15–10:45 GÖR 226 **Social Media and Attention** — ●BERNARDO HUBERMAN
SOE 2.2 Mon 10:45–11:15 GÖR 226 **Mobilizing society with a red balloon** — ●RILEY CRANE
SOE 2.3 Mon 11:15–11:45 GÖR 226 **Collective behaviour and swarm intelligence** — ●JENS KRAUSE

Focus Session: GPU-Computing (with DY)

SOE 5.1 Mon 14:00–14:30 GÖR 226 **Applications of GPU-Computing in Statistical Physics** — ●PETER VIRNAU
SOE 5.2 Mon 14:30–15:00 GÖR 226 **Accelerating Monte Carlo Simulations in Statistical Physics with GPU's** — ●DAVID LANDAU

Focus Session: Experimental Methods

SOE 10.1 Tue 13:30–14:00 GÖR 226 **Complex Economic Systems in the Laboratory** — ●CARS HOMMES
SOE 10.2 Tue 14:00–14:30 GÖR 226 **Multiplicative Cascades: How to model trip within cities** — ●MARTA C. GONZÁLEZ
SOE 10.3 Tue 14:30–15:00 GÖR 226 **Human behavior on networks: lessons and perspectives from game theory** — ●ANGEL SÁNCHEZ
SOE 10.4 Tue 15:00–15:30 GÖR 226 **Measuring Happiness** — ●PETER S. DODDS

Young Scientist Award for Socio- and Econophysics

SOE 8.1 Mon 17:00–17:45 HSZ 02 **Dragon-kings versus black swans: diagnostics and forecasts for the on-going world financial crisis** — ●DIDIER SORNETTE
SOE 8.1 Mon 18:00–18:30 HSZ 02 **Community structure in networks and statistical physics of social dynamics** — ●SANTO FORTUNATO

Joint Symposium on Foundations and Perspectives of Climate Engineering (with AKE, UP)

See SYCE for the full program of the symposium.

SYCE 1.1 Tue 10:30–11:00 HSZ 01 **Oceanic carbon-dioxide removal options: Potential impacts and side effects** — ●ANDREAS OSCHLIES

SYCE 1.2 Tue 11:00–11:30 HSZ 01 **Climate Engineering through injection of aerosol particles into the atmosphere: physical insights into the possibilities and risks** — ●MARK LAWRENCE

SYCE 1.3 Tue 11:30–12:00 HSZ 01 **Geoengineering - will it change the climate game?** — ●TIMO GOESCHL
SYCE 1.4 Tue 12:00–12:30 HSZ 01 **The gamble with the climate - an experiment** — ●MANFRED MILINSKI

Plenary Talks related to SOE

PV X Thu 8:30– 9:15 H1 **Complex Networks: From Statistical Physics to the Cell** — ●ALBERT LASZLO BARABASI

Tutorial

SOE 1.1 Sun 16:00–18:00 H10 **Time Series Analysis in Sociophysics and Econophysics** — ●JOHANNES J. SCHNEIDER, ●TOBIAS PREIS

Invited Talks

SOE 2.1 Mon 9:30–10:15 H44 **Don't panic! - The physics of pedestrian dynamics and evacuation processes** — ●ANDREAS SCHAUSCHNEIDER

SOE 7.1 Tue 9:30–10:00 H44 **Humans playing spatial games** — ●ARNE TRAUlsen

SOE 12.1 Wed 9:30–10:15 H44 **The hidden complexity of open source software** — ●FRANK SCHWEITZER

SOE 17.1 Thu 9:30–10:15 H44 **Wave localization in complex networks** — ●JAN W. KANTELHARDT

SOE 22.1 Fri 9:30–10:15 H44 **Hypergraphs and social systems** — ●GUIDO CALDARELLI

Focus Session: Science of Science

SOE 4.1 Mon 13:30–14:00 H44 **Following the actors: individual and collective behavior in epistemic landscapes** — ●ANDREA SCHARNHORST

SOE 4.2 Mon 14:00–14:30 H44 **Tracking science in real-time from large-scale usage data.** — ●JOHAN BOLLEN

SOE 4.3 Mon 14:45–15:15 H44 **Mapping change in science** — ●MARTIN ROSVALL, CARL BERGSTROM

SOE 4.4 Mon 15:15–15:45 H44 **Statistical physics of citation behavior** — ●SANTO FORTUNATO

Elfmeter im Fussball: Übergang von einem (2x2)-Spiel zu einem (2x3)-Spiel

Wolfgang Leininger and Axel Ockenfels*

The Penalty-Duel and Institutional Design: Is there a Neeskens-Effect?

Abstract

We document an increase in the scoring probability from penalties in soccer, which separates the time period before 1974 significantly from that after 1976: the scoring probability increased by 11%. We explain this finding by arguing that the *institution* of penalty-shooting before 1974 is best described as a *standard of behaviour* for striker and goal-keeper, which in game-theoretic terms represents a 2x2-game. In contrast to this, after 1976 the institution of the penalty-duel is best described by a 3x3 game form constrained by certain behavioural rules. Those rules can be parameterized by a *single* parameter, which nevertheless allows the theoretical reproduction (and hence explanation) of all the empirically documented regularities. The scoring probability in equilibrium of the latter institution is higher than in the former one. We present historical evidence to the effect, that this change in the perception of penalty-duels (as two different games), was caused by Johan Neeskens' shrewd and "revolutionary" penalty-taking during World-Cup 1974, when he shot a penalty in the first minute of the final between Germany and the Netherlands right into the *middle* of the goalmouth.

The following application is based on a working paper by W.Leininger and A.Ockenfels (CESIFO WORKING PAPER NO. 2187, 2008). The article focuses on the 'Penalty-Duel' in soccer and describes it as a simultaneous two player game – a game between the goalkeeper and the kicker.

Neeskens Elfmeter:

<https://www.youtube.com/watch?v=44HvFzhV9xI>

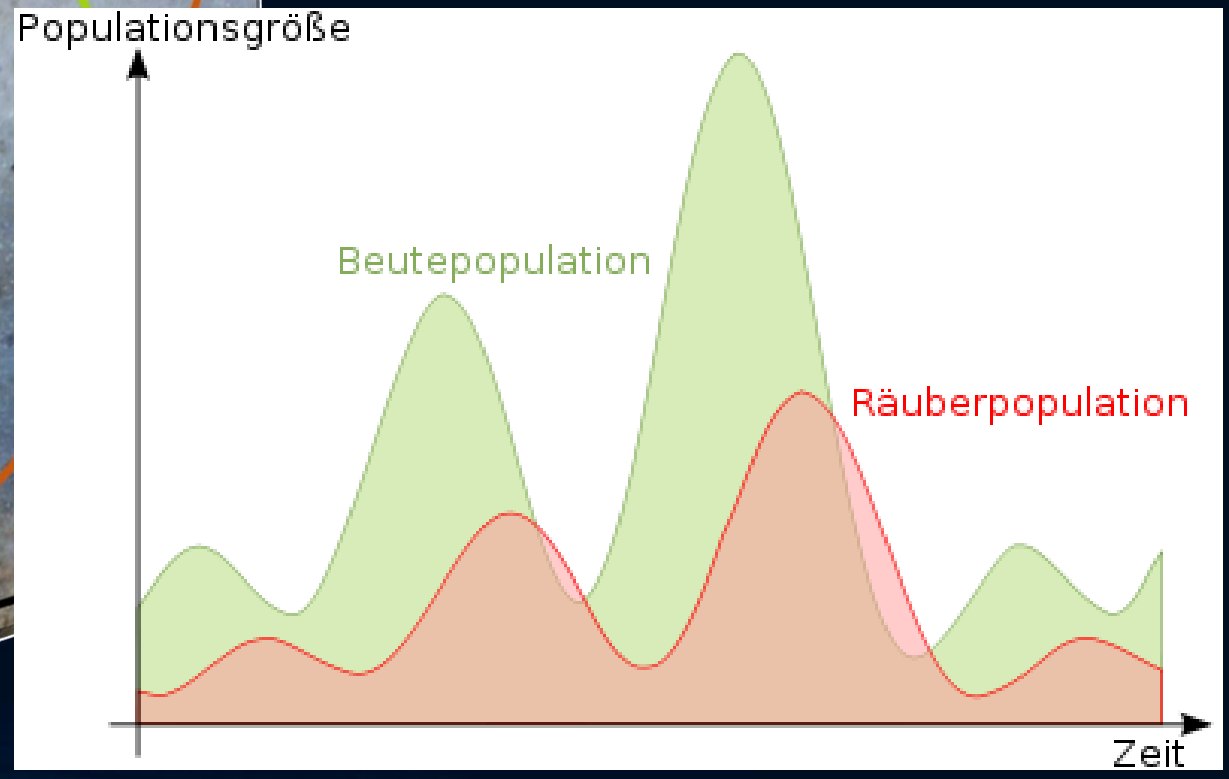
Artikel:

<https://www.econstor.eu/bitstream/10419/26769/1/528420186.PDF>

Räuber-Beute-Beziehung



Das Räuber-Beute Spiel



Die Lotka-Volterra-Gleichung (Räuber-Beute-Gleichung) für N-Populationen

Anzahl der Räuber/Beute Wesen
der i-ten Population zur Zeit t

$$\frac{dx_i(t)}{dt} = \left(r_i + \sum_{j=1}^N b_{ij} x_j(t) \right) x_i(t)$$

Reproduktions-
bzw. Sterberaten

Interaktionsmatrix