Physik der sozio-ökonomischen Systeme *mit dem Computer*

PC-POOL RAUM 01.120
JOHANN WOLFGANG GOETHE UNIVERSITÄT
16.11.2018

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

Allgemeines zur Vorlesung

- Ort und Zeit:
  PC-Pool Raum 01.120, immer freitags von 15.00 bis 17.00 Uhr
- Vorlesungs-Materialien:
  http://th.physik.uni-frankfurt.de/~hanauske/VPSOC/
- Aufgaben auf der Online-Lernplatform Lon Capa:
  http://lon-capa.server.uni-frankfurt.de/
- Plan für die heutige Vorlesung:
  Wiederholung (Klassifizierung von symmetrischen (2x2)-Spielen), Evolution unsymmetrischer Spiele (Bi-Matrix Spiele), Klassifizierung von unsymmetrischen (2x2)-Spielen, Maple und Python Programme, Übungsaufgabe auf der Lon Capa Lernplatform

Allgemeines (2x2)-Spiel

	Spieler B wählt Strategie 1	Spieler B wählt Strategie 2
Spieler A wählt Strategie 1	$(\$_{11}^{A}, \$_{11}^{B})$	$(\$_{12}^{A}, \$_{12}^{B})$
Spieler A wählt Strategie 2	$(\$_{21}^{A}, \$_{21}^{B})$	$(\$_{22}^{A}, \$_{22}^{B})$

Symmetrisches (2x2)-Spiel

	Spieler B Strategie 1 y=1	Spieler B Strategie 1 y=0
Spieler A Strategie 1 x=1	(a , a)	(b , c)
Spieler A Strategie 2 x=0	(c , b)	(d , d)

Die Klasse der dominanten Spiele (a > c und b > d bzw. a < c und b < d)

Bei dieser Spielklasse dominiert eine Strategie die andere. Es existiert nur ein reines Nash-Gleichgewicht welches die dominante Strategie des Spiels darstellt. Dieser Fall tritt ein, falls:
a > c und b > d : Strategie 1 dominiert Strategie 2; dominante Strategie bei (x,y)=(1,1).
a < c und b < d : Strategie 2 dominiert Strategie 1; dominante Strategie bei (x,y)=(0,0).

Koordinationsspiele (a > c und b < d)

Ein Koordinationsspiel existiert, falls die Parameter a, b, c und d der Auszahlungmatrix die folgenden Bedingungen erfüllen: a > c und b < d . Bei dieser Spielklasse existieren drei Nash-Gleichgewichte, ein gemischtes Nash-Gleichgewicht und zwei reine, symmetrische Nash-Gleichgewicht bei (x,y)=(0,0) und (x,y)=(1,1).

Anti-Koordinationsspiele (a < c und b > d)

Ein Anti-Koordinationsspiel existiert, falls die Parameter a, b, c und d der Auszahlungmatrix die folgenden Bedingungen erfüllen: a < c und b > d . Bei dieser Spielklasse existieren drei Nash-Gleichgewichte, ein gemischtes Nash-Gleichgewicht und zwei reine, unsymmetrische Nash-Gleichgewicht bei (x,y)=(0,1) und (x,y)=(0,1).

$$\frac{dx_i^A(t)}{dt} = \left[\underbrace{\sum_{l=1}^{m_B} \$_{il}^A \, x_l^B(t)}_{\text{Fitness der Strategie i}} - \underbrace{\sum_{l=1}^{m_B} \sum_{k=1}^{m_A} \$_{kl}^A \, x_k^A(t) \, x_l^B(t)}_{\text{Durchschn. Fitness der Population A}} \right] x_i^A(t) \qquad (1)$$

$$\frac{dx_j^B(t)}{dt} = \left[\underbrace{\sum_{l=1}^{m_A} \$_{lj}^B \, x_l^A(t)}_{\text{Fitness der Strategie j}} - \underbrace{\sum_{l=1}^{m_A} \sum_{k=1}^{m_B} \$_{lk}^B \, x_l^A(t) \, x_k^B(t)}_{\text{Durchschn. Fitness der Population B}} \right] x_j^B(t) \quad ,$$

wobei $x_i^A(t),\ i = 1, 2, \ldots, m_A$ und $x_j^B(t),\ j = 1, 2, \ldots, m_B$ die Anteile der in den Spielergruppen A und B zur Zeit t gewählten Strategien widerspiegeln und in der Soziobiologie den Frequenzen der *Quasispezies* entsprechen.

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:

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

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

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

Lösen des evolutionären Spiels mit Maple

(Vorlage2.mw)

```
> restart:
  with(LinearAlgebra):
  with(plots):
```

Definition der Auszahlunsmatrix für Spieler A (USA) im Spiel des Wettrüstens:

```
> D_A11:=1:
  D_A12:=4:
  D_A21:=0:
  D_A22:=2:
  D_A:=Matrix(2,2,[D_A11,D_A12,D_A21,D_A22]);
```

Da sich das Spiel um ein symmetrisches (2 Personen)–(2 Strategien) Spiel handelt, erhält man die Auszahlunsmatrix für Spieler B (Nord Korea) durch die transponierte Matrix des Spielers A:

```
> D_B:=Transpose(D_A);
```

Wir betrachten im folgenden die evolutionäre Erweiterung des Dilemma des Wettrüstens. Die Population bestehe aus den Entscheidungsträgern von vielen Ländern und zu jedem Zeitpunkt treffen sie erneut die Entscheidung "Aufrüsten" oder "Abrüsten". Die Differentialgleichung, die die zeitliche Entwicklung des Populationsvektors x(t) (Anteil der Länder die "Aufrüsten") beschreibt lautet:

```
> DGL:=diff(x(t),t)=simplify(((D_A11-D_A21)*(x(t)-x(t)^2) + (D_A12-D_A22)*(1-2*x(t)+x
  (t)^2))*x(t));
  g:=simplify(((D_A11-D_A21)*(x-x^2) + (D_A12-D_A22)*(1-2*x+x^2))*x);
```

Die die zeitliche Entwicklung bestimmende Funktion g(x) besitzt das folgende Aussehen:

```
> plot(g, x=0..1);
```

Lösen des evolutionären Spiels mit Maple

(Vorlage2.mw)

x(t), der Anteil der Spieler (Länder) die zum Zeitpunkt t die Strategie 1 ("Aufrüsten") spielen , hängt neben der Funktion g(x) von dem Anfangswert x(t=0) ab.Setzt man z.B. x(t=0)=0.1 (entspricht einer Anfangspopulation von 10% der Länder rüstet auf und 90% rüstet ab) so kann man die Differentialgleichung analytisch Lösen (nicht immer möglich):

```
> LoesDGL:=simplify(dsolve({DGL,x(0)=0.1}));
```

Darstellen der Lösung:

```
> plot(rhs(LoesDGL),t=0..4);
```

Ausgabe des Wertes zu einer festen Zeit (z.B. t=2)

```
> evalf(subs({t=2},rhs(LoesDGL)));
```

Falls eine analytische Lösung nicht möglich ist kann man wie folgt die DGL numerisch lösen und grafisch darstellen:

```
> LoesDGL:=dsolve({DGL,x(0)=0.1},x(t),type=numeric):
  odeplot(LoesDGL,[t,x(t)],0..4);
```

Ausgabe des Wertes zu einer festen Zeit (z.B. t=2)

```
> LoesDGL(2);
```

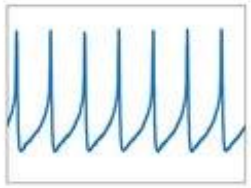
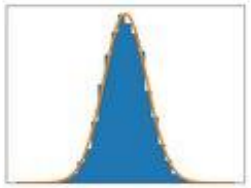
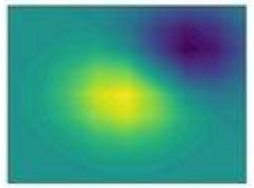
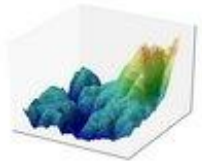
Python mit Matplotlib: http://matplotlib.org/

matplotlib

Version 3.0.0

home | examples | tutorials | API | docs »

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the `pyplot` module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Installation

Visit the Matplotlib installation instructions.

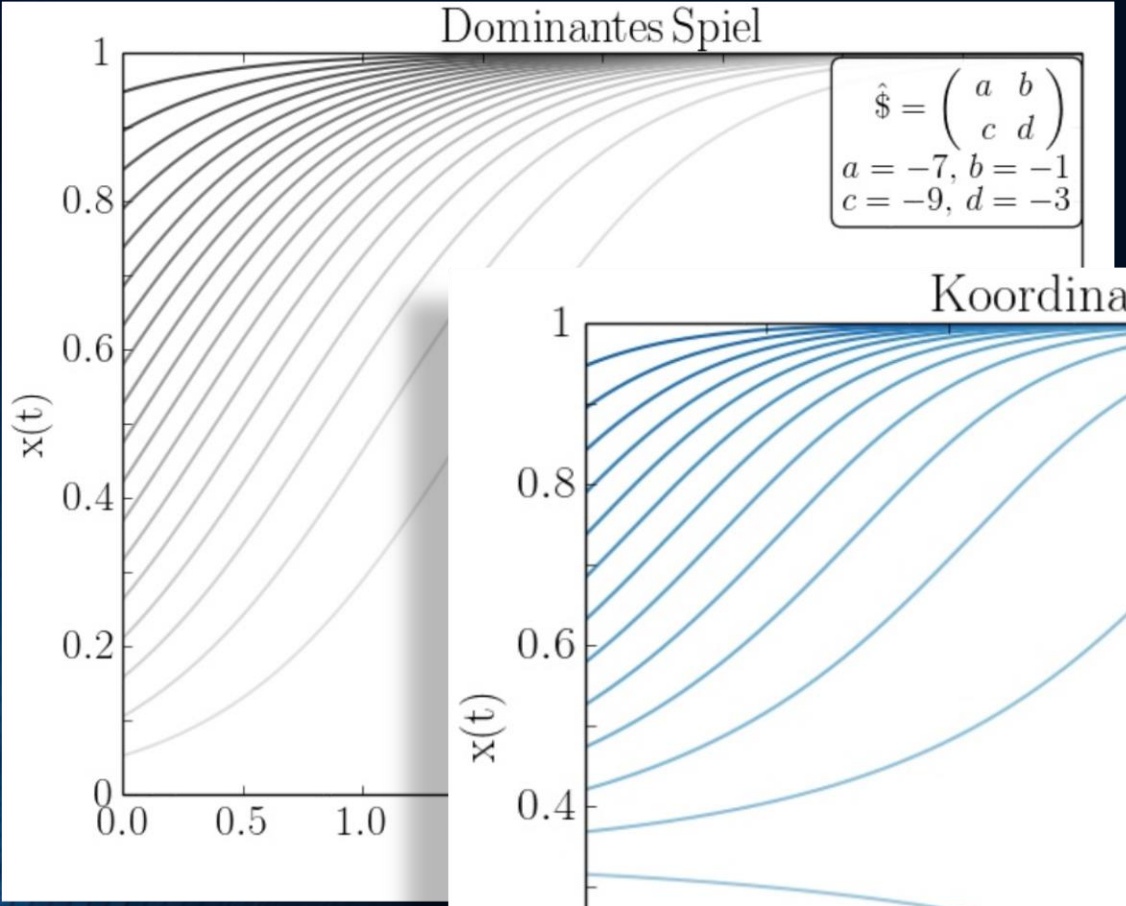
Lösen des evolutionären Spiels mit Python

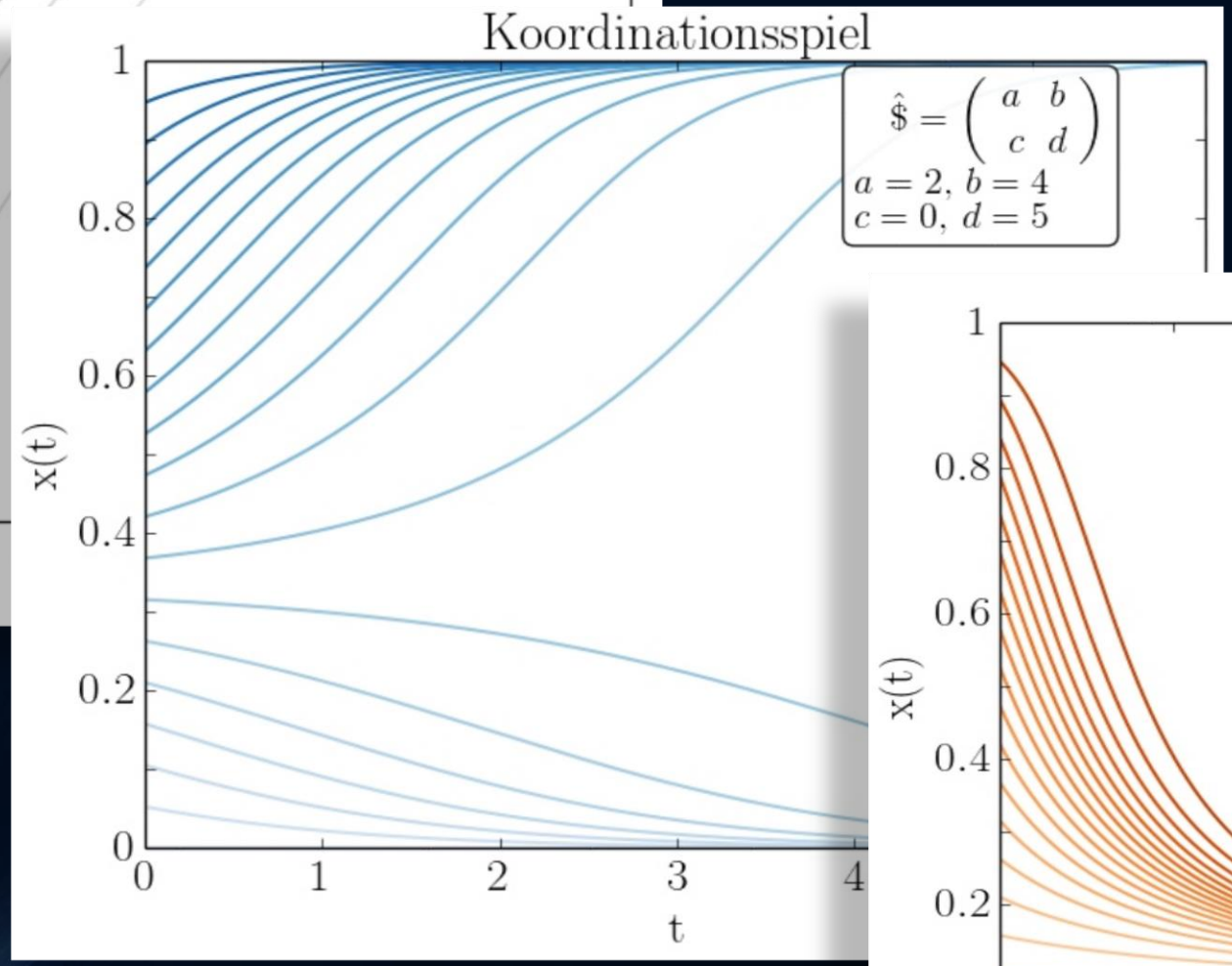
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib

# Groessenfestlegung der Labels usw. im Bild
params = {
    'text.usetex'       : True,
    'axes.labelsize'  : 20,
    'xtick.labelsize' : 20,
    'ytick.labelsize' : 20
}
matplotlib.rcParams.update(params)

# Definition der Funktion g
def g(x,a,b,c,d):
  g=((a-c)*(x-x*x) + (b-d)*(1-2*x+x*x))*x
  return g

#Festlegung der Auszahlungsmatrix des symmetrischen (2x2)-Spiels
a=2
b=4
c=0
d=5
#End(zeit)punkt, Anzahl der Zeitschritte
tend=6
numpoints=500
#Anfangswert der Population
x0=0.4
```

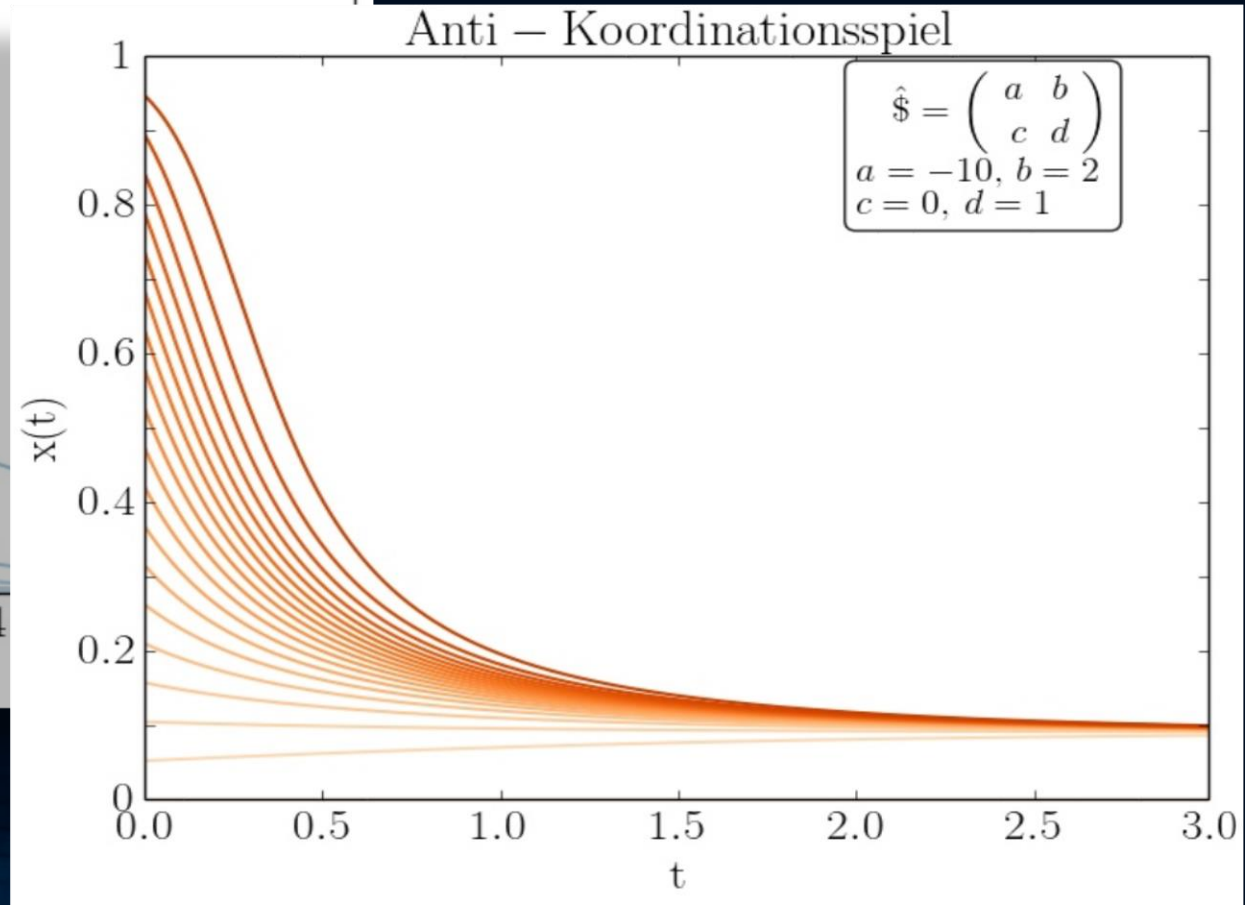
Lösen des evolutionären Spiels mit Python

```
#Loesen der DGL
Loes=np.empty([numpoints,2])
t=np.linspace(0,tend,numpoints)
dt=t[1]-t[0]
Loes[0].flat[0] = t[0]
Loes[0].flat[1] = x0
i = 1
while i < len(Loes):
    Loes[i].flat[0] = t[i]
    dx=g(Loes[i-1,1],a,b,c,d)*dt
    Loes[i].flat[1] = Loes[i-1,1] + dx
    i = i + 1

#Plotten des Bildes
plt.plot(Loes[:,0],Loes[:,1],c="black", linewidth=1.5, linestyle='-')

# Achsenbeschriftungen usw.
plt.ylim(0,1)
plt.ylabel(r"$\rm x(t)$")
plt.xlabel(r"$\rm t$")

#Speichern des Bildes als .jpg--Datei
saveFig="./evol.jpg"
plt.savefig(saveFig, dpi=100,bbox_inches="tight",pad_inches=0.05,format="jpg")
plt.show()
```

Lösen des evolutionären Spiels mit Python Version evol1.py



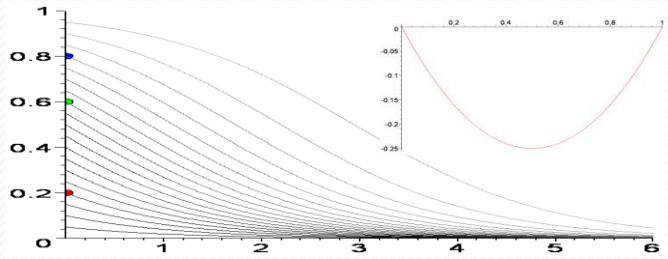




Klassifizierung von evolutionären, symmetrischen (2x2)-Spielen

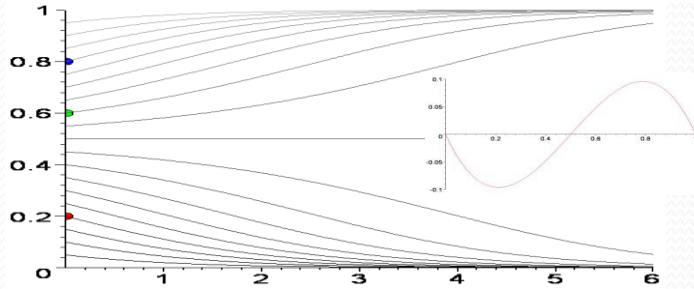
- **Dominante Spiele**
  (2. Strategie dominiert 1.Strategie)

Es existiert ein Nash - Gleichgewicht, welches die anderen Strategien dominiert. ESS bei x=0.



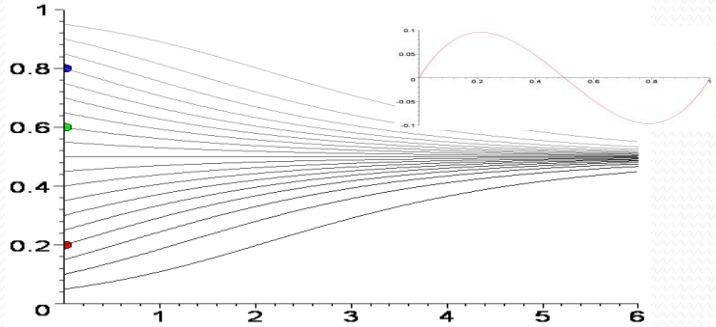
- **Koordinationsspiele**

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



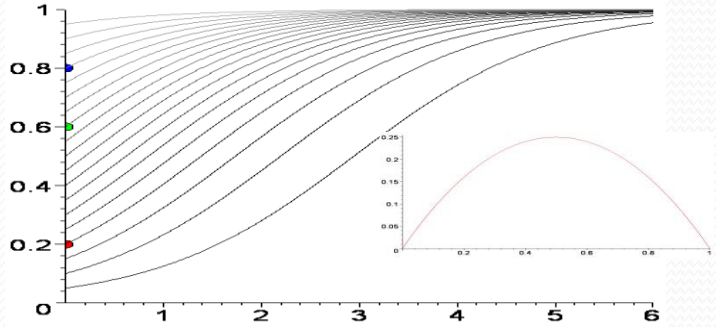
- **Anti – Koordinationsspiele**

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.



Symmetriebedingung der Auszahlungsmatrizen

	Spieler 2 wählt Strategie 1	**Spieler 2 wählt Strategie 2**
Spieler 1 wählt Strategie 1	$(\$_{11}^1, \$_{11}^2)$	$(\$_{12}^1, \$_{12}^2)$
Spieler 1 wählt Strategie 2	$(\$_{21}^1, \$_{21}^2)$	$(\$_{22}^1, \$_{22}^2)$

Auszahlungsmatrix des zweiten Spielers:

$$\hat{\$}^2 = \begin{pmatrix} \$_{11}^2 & \$_{12}^2 \\ \$_{21}^2 & \$_{22}^2 \end{pmatrix}$$

Auszahlungsmatrix des ersten Spielers:

$$\hat{\$}^1 = \begin{pmatrix} \$_{11}^1 & \$_{12}^1 \\ \$_{21}^1 & \$_{22}^1 \end{pmatrix}$$

Symmetriebedingung:

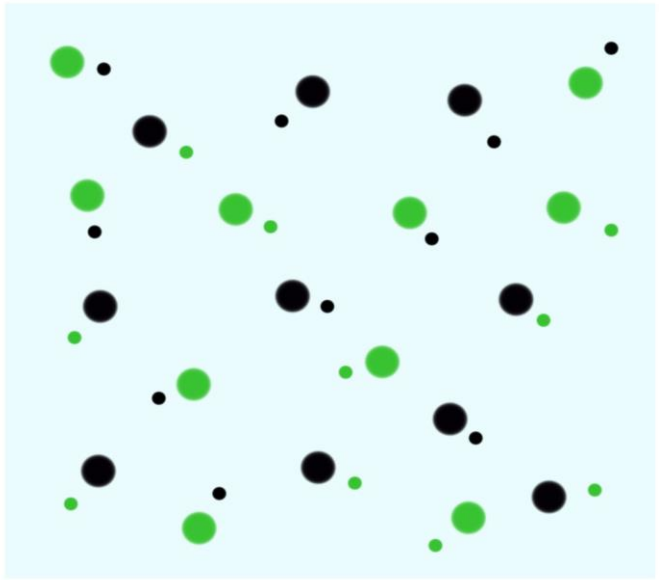
$$\left(\hat{\$}^2\right)^T = \begin{pmatrix} \$_{11}^2 & \$_{21}^2 \\ \$_{12}^2 & \$_{22}^2 \end{pmatrix} \overset{!}{=} \begin{pmatrix} \$_{11}^1 & \$_{12}^1 \\ \$_{21}^1 & \$_{22}^1 \end{pmatrix} = \hat{\$}^1$$

Transponierte Matrix

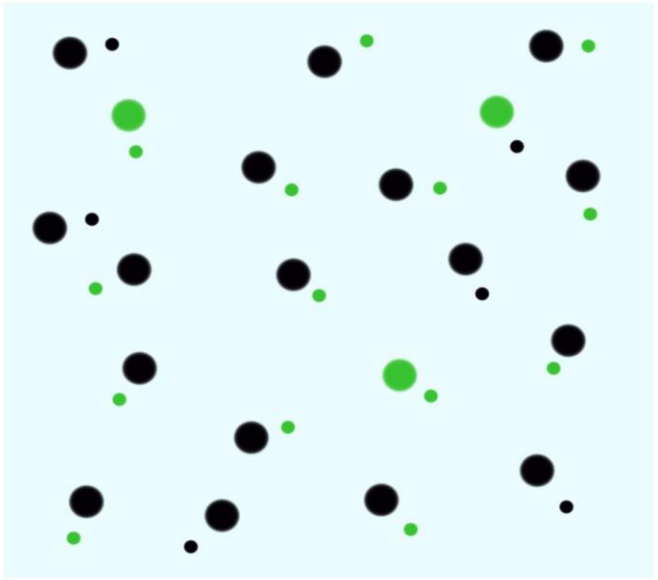
Evolutionäre Spieltheorie (I)

Unsymmetrische (2x2)-Spiele (Bimatrixspiele)

Bei unsymmetrischen (2x2)-Spielen 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.



zeitliche Entwicklung der Population



x(0)=0.5 , y(0)=0.5

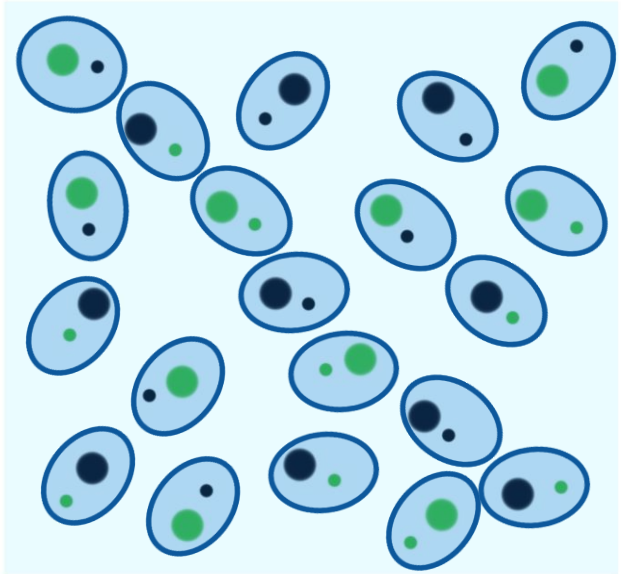
x(10)=0.15 , y(10)=0.7

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 (II)

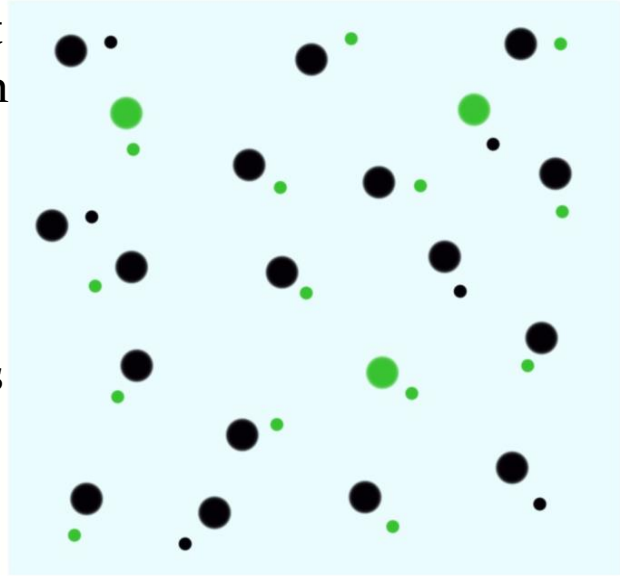
Unsymmetrische (2x2)-Spiele (Bimatrixspiele)

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).

$$\frac{dx_i^A(t)}{dt} = \left[\underbrace{\sum_{l=1}^{m_B} \$_{il}^A \, x_l^B(t)}_{\text{Fitness der Strategie i}} - \underbrace{\sum_{l=1}^{m_B} \sum_{k=1}^{m_A} \$_{kl}^A \, x_k^A(t) \, x_l^B(t)}_{\text{Durchschn. Fitness der Population A}} \right] x_i^A(t) \qquad (1)$$

$$\frac{dx_j^B(t)}{dt} = \left[\underbrace{\sum_{l=1}^{m_A} \$_{lj}^B \, x_l^A(t)}_{\text{Fitness der Strategie j}} - \underbrace{\sum_{l=1}^{m_A} \sum_{k=1}^{m_B} \$_{lk}^B \, x_l^A(t) \, x_k^B(t)}_{\text{Durchschn. Fitness der Population B}} \right] x_j^B(t) \quad ,$$

wobei $x_i^A(t),\ i = 1, 2, \ldots, m_A$ und $x_j^B(t),\ j = 1, 2, \ldots, m_B$ die Anteile der in den Spielergruppen A und B zur Zeit t gewählten Strategien widerspiegeln und in der Soziobiologie den Frequenzen der *Quasispezies* entsprechen.

Replikatordynamik (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 Replikatordynamik beschreibt wie sich die einzelnen Populationsanteil der zur Zeit t gewählten Strategien $x_j(t)$, j=1,2,...M im Laufe der Zeit entwickeln.

$$\dot{x}_j(t) := \frac{dx_j(t)}{dt} = x_j(t) \cdot \left[\sum_{k=1}^{M} \$_{jk} \cdot x_k(t) - \sum_{l=1}^{M} \sum_{k=1}^{M} \$_{kl} \cdot x_k(t) \cdot x_k(t) \right]$$

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

Durchschnitt-licher Erfolg der j-ten Strategie

Durschnittliche Fitness (Auszahlung) der gesamten Population

Bi-Matrix Spiele

unsymmetrische (2x2)-Spiele, zwei unterscheidbare Populationsgruppen

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):

$$\frac{dx(t)}{dt} = \left[\left(\$_{11}^A + \$_{22}^A - \$_{12}^A - \$_{21}^A\right) y(t) + \left(\$_{12}^A - \$_{22}^A\right)\right] \left(x(t) - (x(t))^2\right) =: g_A(x,y) \qquad (2)$$

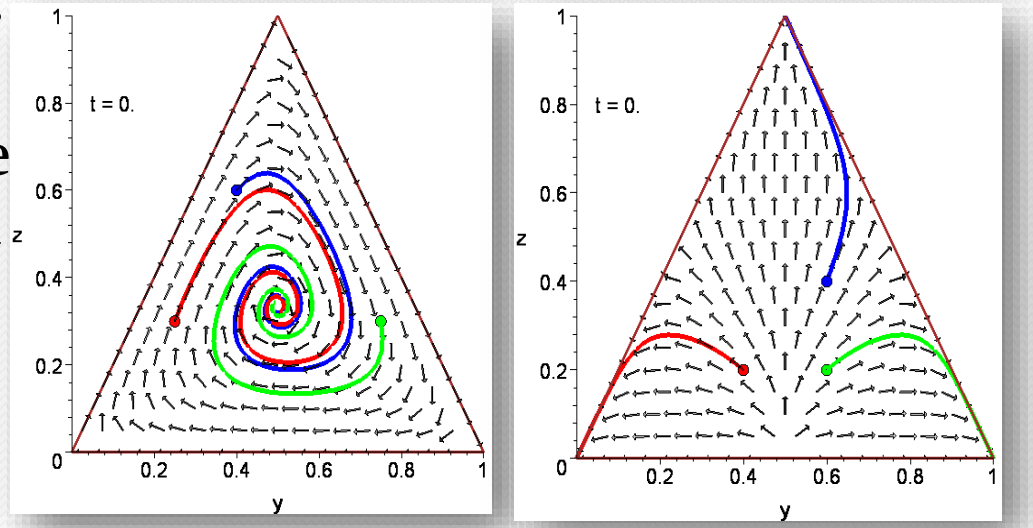
$$\frac{dy(t)}{dt} = \left[\left(\$_{11}^B + \$_{22}^B - \$_{12}^B - \$_{21}^B\right) x(t) + \left(\$_{12}^B - \$_{22}^B\right)\right] \left(y(t) - (y(t))^2\right) =: g_B(x,y)$$

Weitere Arten von Spieltypen

(Ausblick: Vorlesungsteil 5)

- **Mehr als zwei Strategien:**

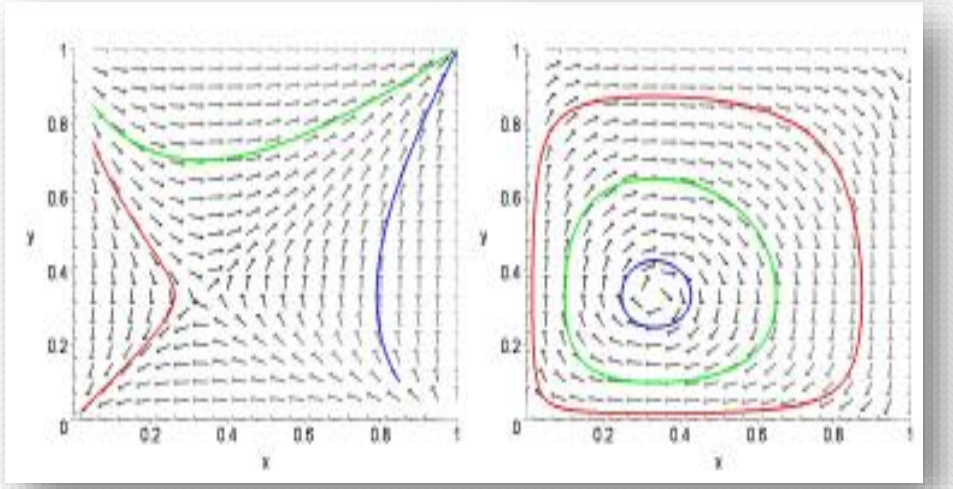
Schon bei drei Strategien können 19 unterschiedliche Spielklassen unterschieden werden.



- **Bimatrix Spiele**

Unsymmetrische (2x2) Spiele:

Setzt sich die Population aus zwei unterschiedlichen Spielergruppen (x(t) und y(t)) zusammen, so spricht man von Bimatrix Spielen.



Beispiel 1:

Kampf der Geschlechter als evolutionäres Spiel

	Kino	Disko
Kino	(1 , 3)	(0 , 0)
Disko	(0 , 0)	(3 , 1)

Das gekoppelte System von Differentialgleichung lautet:

$$\frac{dx(t)}{dt} = x(t)\cdot\left(4\cdot y(t) - 4\cdot x(t)\cdot y(t) + 3\cdot x(t) - 3\right)$$

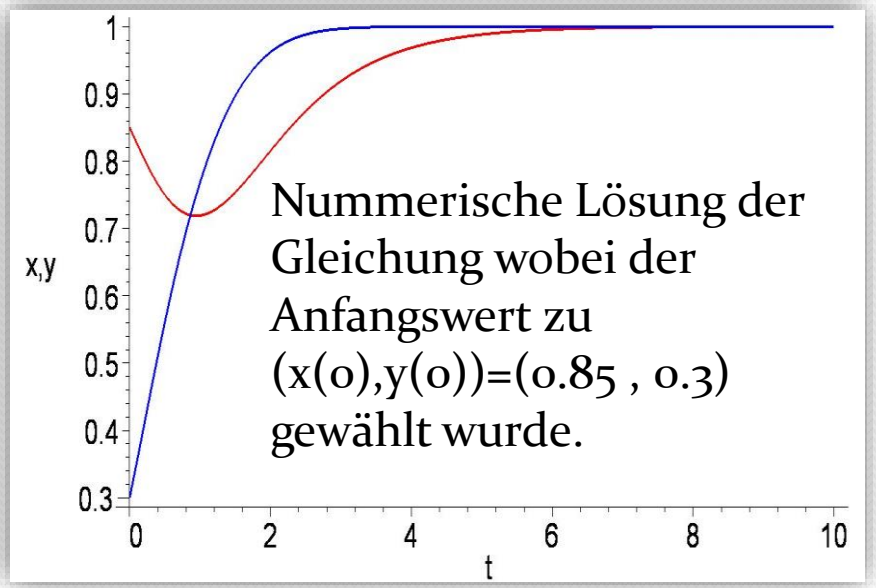
$$\frac{dy(t)}{dt} = y(t)\cdot\left(4\cdot x(t) - 4\cdot x(t)\cdot y(t) + y(t) - 1\right)$$

Die Lösung der Gleichung erfolgt wiederum durch Integration bzw. kann mithilfe des Computers nummerisch berechnet werden. Es muss in beiden Fällen ein fester Anfangswert (x(0),y(0)) gewählt werden.

Die beiden Gruppen der Population sind Männer (A, x(t))und Frauen (B, y(t)). Da nur zwei Strategien wählbar sind, lassen sich die jeweiligen Populationsanteile durch lediglich eine Größe ausdrücken (x(t) und y(t)).

$$x(t) := x_1^A(t) \;,\; x_2^A(t) = 1 - x(t)$$

$$y(t) := x_1^B(t) \;,\; x_2^B(t) = 1 - y(t)$$



Nummerische Lösung der Gleichung wobei der Anfangswert zu (x(0),y(0))=(0.85 , 0.3) gewählt wurde.

Bi-Matrix Spiele

Gemischte Nash-Gleichgewichte

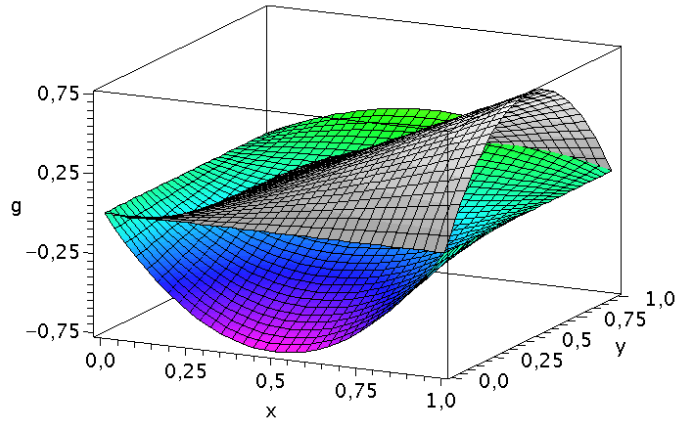
Nash-Gleichgewicht in gemischten Strategien:

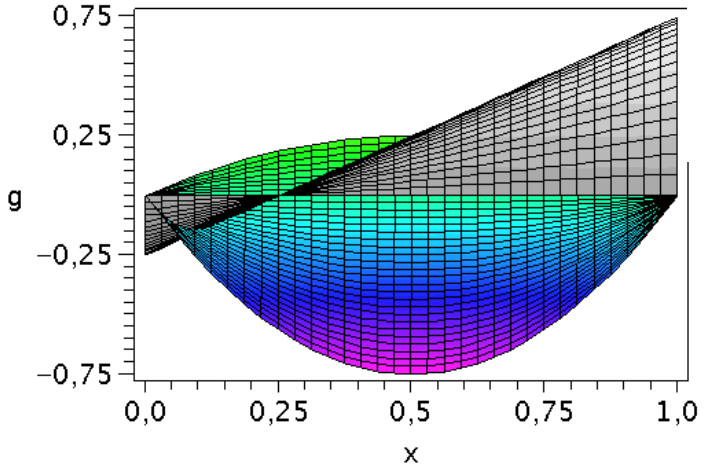
$$\left.\frac{\partial \tilde{\$}^{A}(\tilde{s}^{A}, \tilde{s}^{B})}{\partial \tilde{s}^{A}}\right|_{\tilde{s}^{B}=\tilde{s}^{B\star}} = 0 \quad \forall \ \tilde{s}^{A} \in [0,1] \ , \ \tilde{s}^{B\star} \in]0,1[$$

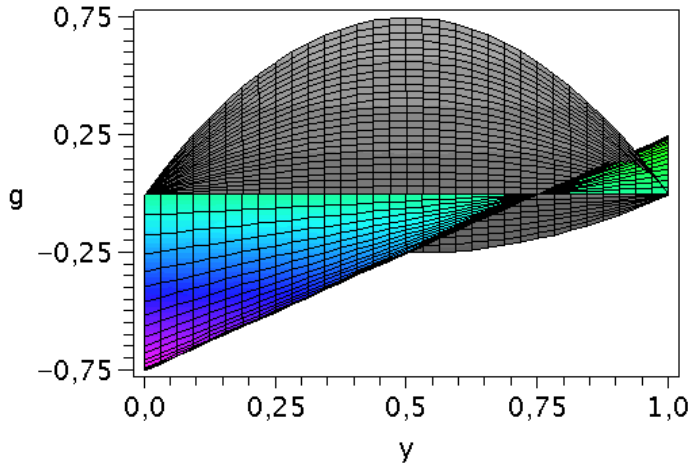
$$\left.\frac{\partial \tilde{\$}^{B}(\tilde{s}^{A}, \tilde{s}^{B})}{\partial \tilde{s}^{B}}\right|_{\tilde{s}^{A}=\tilde{s}^{A\star}} = 0 \quad \forall \ \tilde{s}^{B} \in [0,1] \ , \ \tilde{s}^{A\star} \in]0,1[$$

Eigenschaften der Funktionen $g_A(x,y)$ und $g_B(x,y)$

Die das Bimatrix Spiel bestimmenden Funktionen $g_A(x,y)$ (farbige Fläche) und $g_B(x,y)$ (graue Fläche) sind in den unteren Abbildungen veranschaulicht. Beide Teilgruppenspiele gehören der Klasse der Koordinationsspiele an.







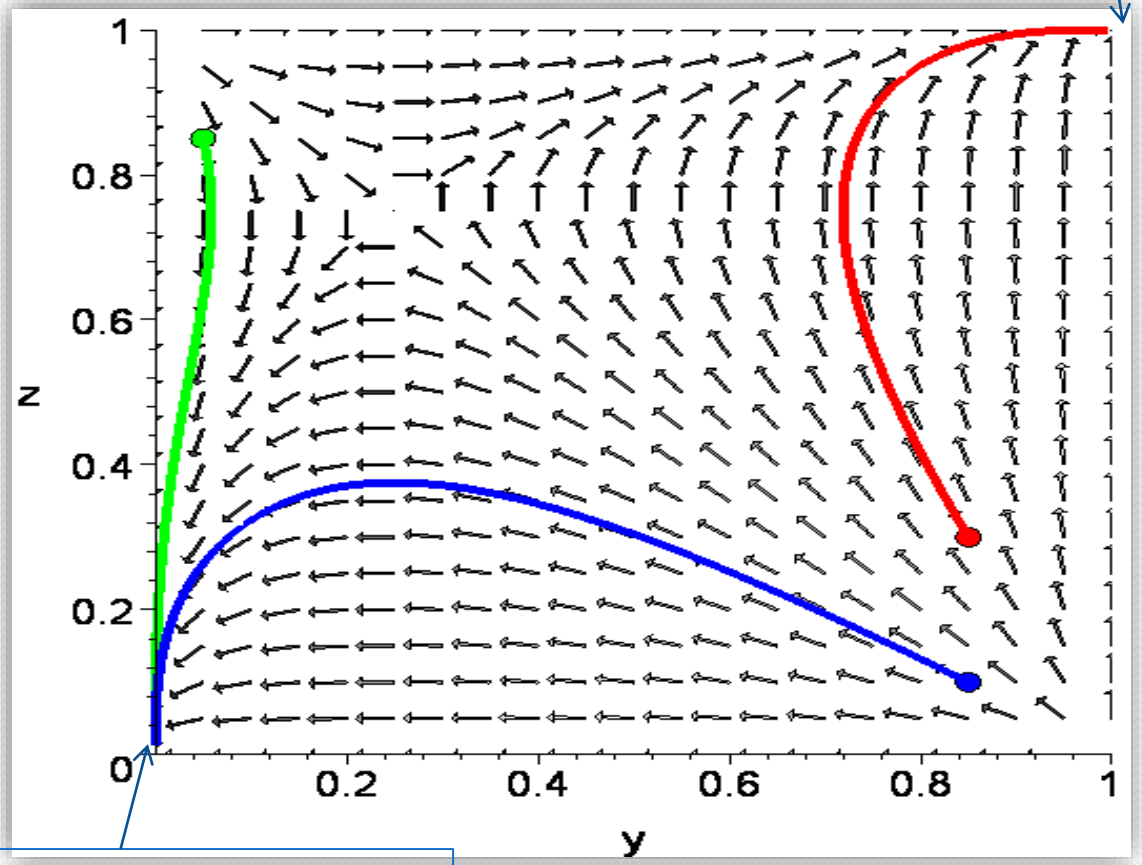
Beispiel 1:

Kampf der Geschlechter als evolutionäres Spiel

	Kino	Disko
Kino	(1 , 3)	(0 , 0)
Disko	(0 , 0)	(3 , 1)

Das „Kampf der Geschlechter“- Spiel gehört der Klasse der „Sattelpunktsspiele“ (Saddle Class) an. Das Phasenportrait des Spiels besitzt das folgende Aussehen:

(1,1) entspricht (Kino,Kino)



(0,0) entspricht (Disko,Disko)

Die rechte Abbildung zeigt das zeitliche Verhalten von drei nummerischen Lösungen mit unterschiedlichen Anfangsbedingungen. Die evolutionäre Erweiterung des Spiels besitzt zwei evolutionär stabile Strategien ((0,0) und (1,1)). Die blaue und grüne Populationsentwicklung enden bei (0,0) während die Anfangsbedingung der roten Population bei (1,1) endet.

Beispiel 2:

Klasse der Zentrumsspiele (Center Class)

	Strat. 1	Strat. 2
Strat. 1	(2 , -2)	(0 , 0)
Strat. 2	(0 , 0)	(1 , -1)

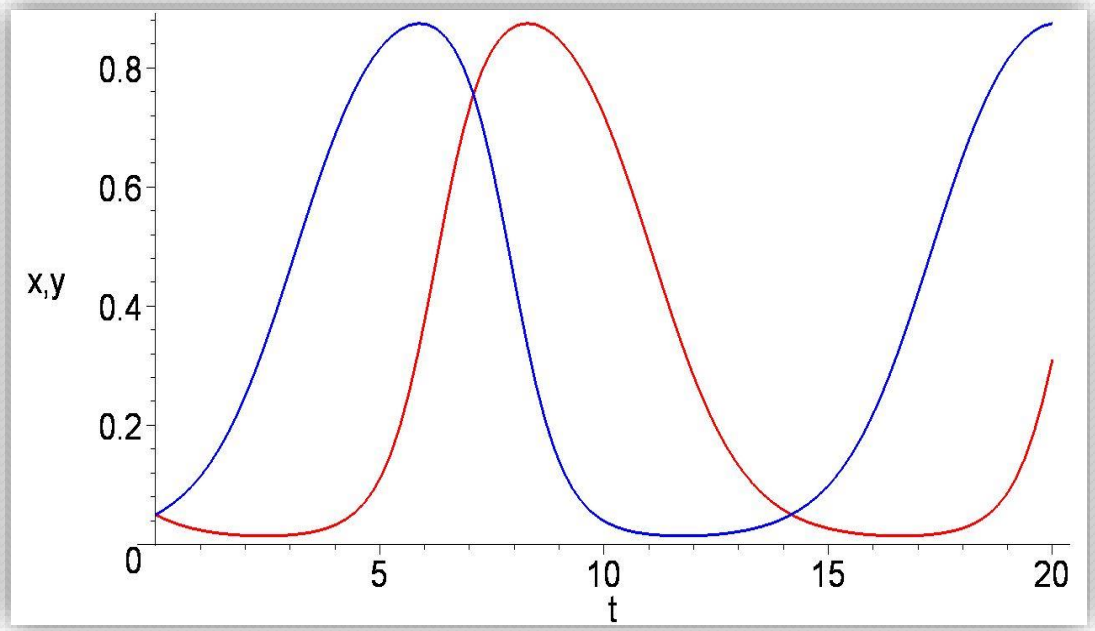
Das gekoppelte System von Differentialgleichung für dieses Beispiel lautet:

$$\frac{dx(t)}{dt} = x(t)\cdot\left(3\cdot y(t) - 3\cdot x(t)\cdot y(t) + x(t) - 1\right)$$

$$\frac{dy(t)}{dt} = y(t)\cdot\left(-3\cdot x(t) + 3\cdot x(t)\cdot y(t) - y(t) + 1\right)$$

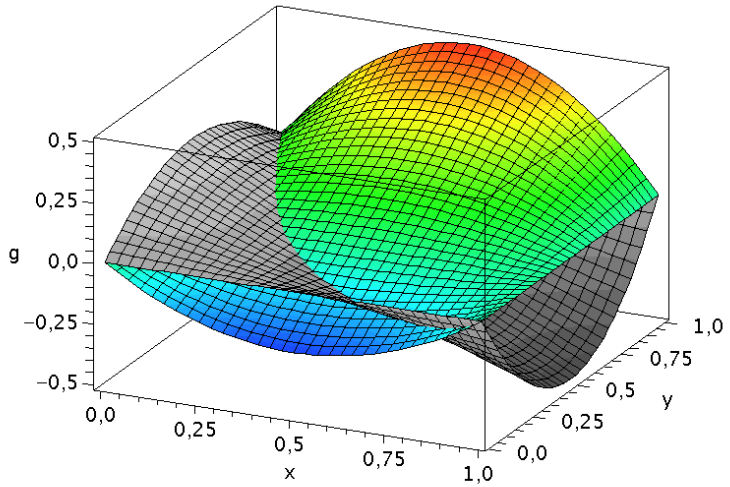
Die Lösung der Gleichung erfolgt wiederum durch Integration bzw. kann mithilfe des Computers nummerisch berechnet werden. Es wurde der Anfangswert zu (x(0),y(0)) =(0.05 , 0.05) gewählt.

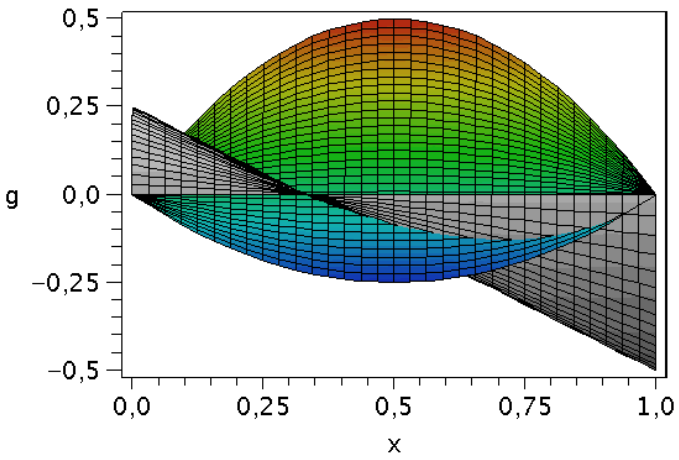
Die rechte Abbildung zeigt eine nummerische Lösung der obigen Gleichung, wobei der Anfangswert zu (x(0),y(0)) =(0.05 , 0.05) gewählt wurde. Im Gegensatz zu allen anderen möglichen Klassen von Bimatrixspielen, treten bei der Klasse der Zentrumsspiele periodische Verläufe der Populationsanteile x(t) und y(t) auf - die Populationsanteile enden nicht in einer evolutionär stabilen Strategie.

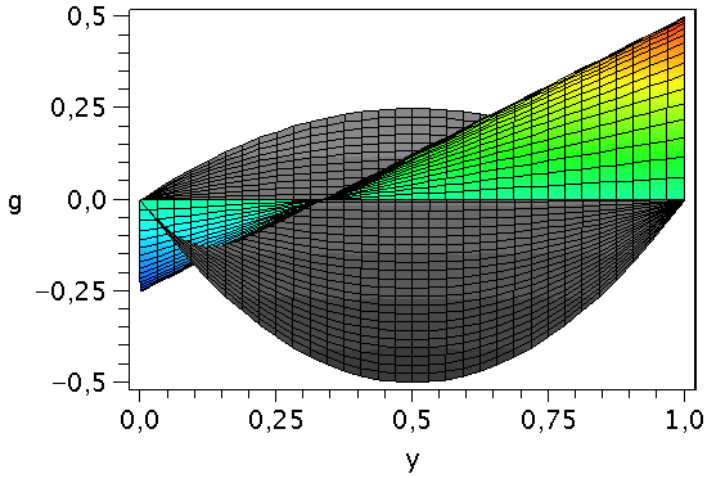


Eigenschaften der Funktionen $g_A(x,y)$ und $g_B(x,y)$

Die das Bimatrix Spiel bestimmenden Funktionen $g_A(x,y)$ (farbige Fläche) und $g_B(x,y)$ (graue Fläche) sind in den unteren Abbildungen veranschaulicht. Das Spiel der Gruppe A gehört der Klasse der Koordinationsspiele an, das der Gruppe B der Klasse der Anti-Koordinationsspiele.







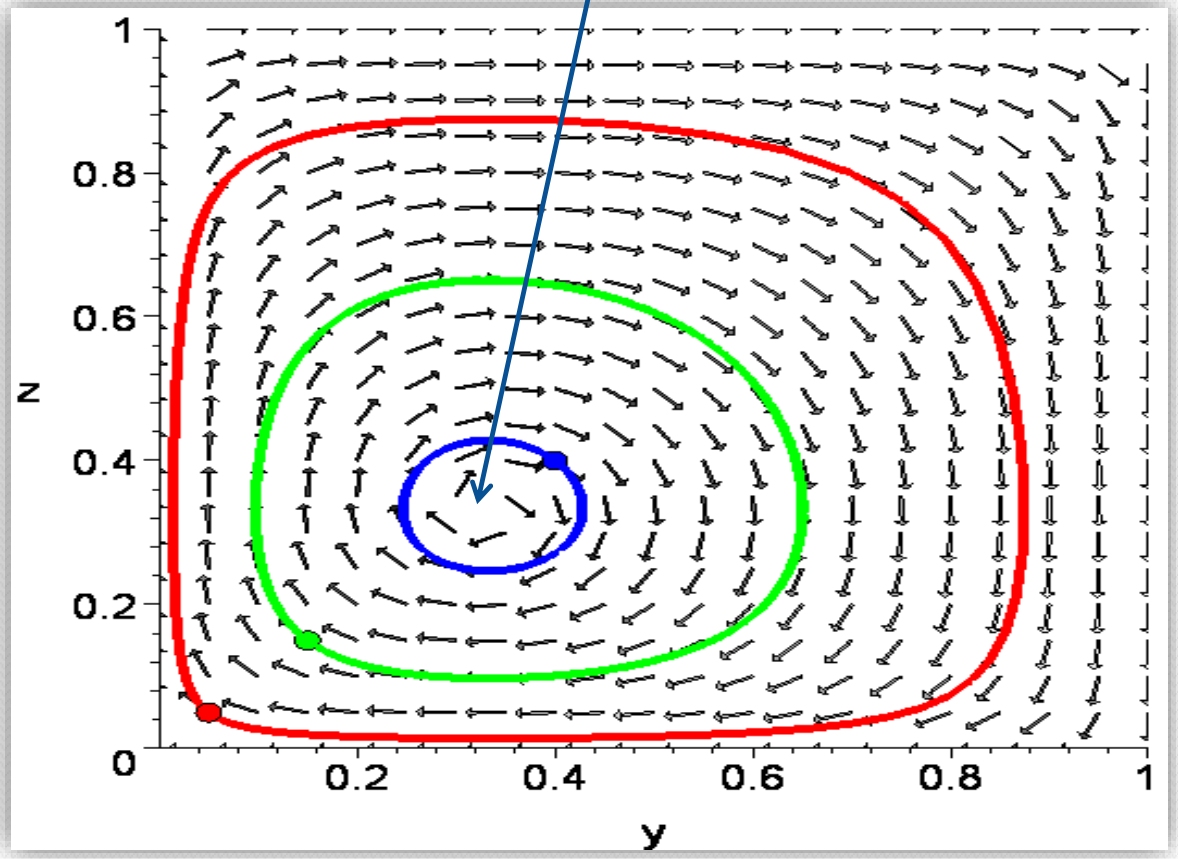
Beispiel 2:

Klasse der Zentrumsspiele (Center Class)

	Strat. 1	Strat. 2
Strat. 1	(2 , -2)	(0 , 0)
Strat. 2	(0 , 0)	(1 , -1)

Das Phasenportrait des zweiten Beispiels besitzt das folgende Aussehen:

Zentrum: Gemischtes Nash-Gleichgewicht des Spiels



Die rechte Abbildung zeigt das zeitliche Verhalten von drei nummerischen Lösungen mit unterschiedlichen Anfangsbedingungen. Dieses Bimatrixspiel besitzt keine evolutionär stabile Strategie, da die einzelnen Phasenraum-Trajektorien sich keinem Punkt annähern, sondern auf einer geschlossenen, zyklischen Bahn um das gemischte Nash-Gleichgewicht kreisen.

Beispiel 3:

Klasse der Eckenspiele (Corner Class)

	Strat. 1	Strat. 2
Strat. 1	(-2 , 2)	(0 , 0)
Strat. 2	(0 , 0)	(1 , 1)

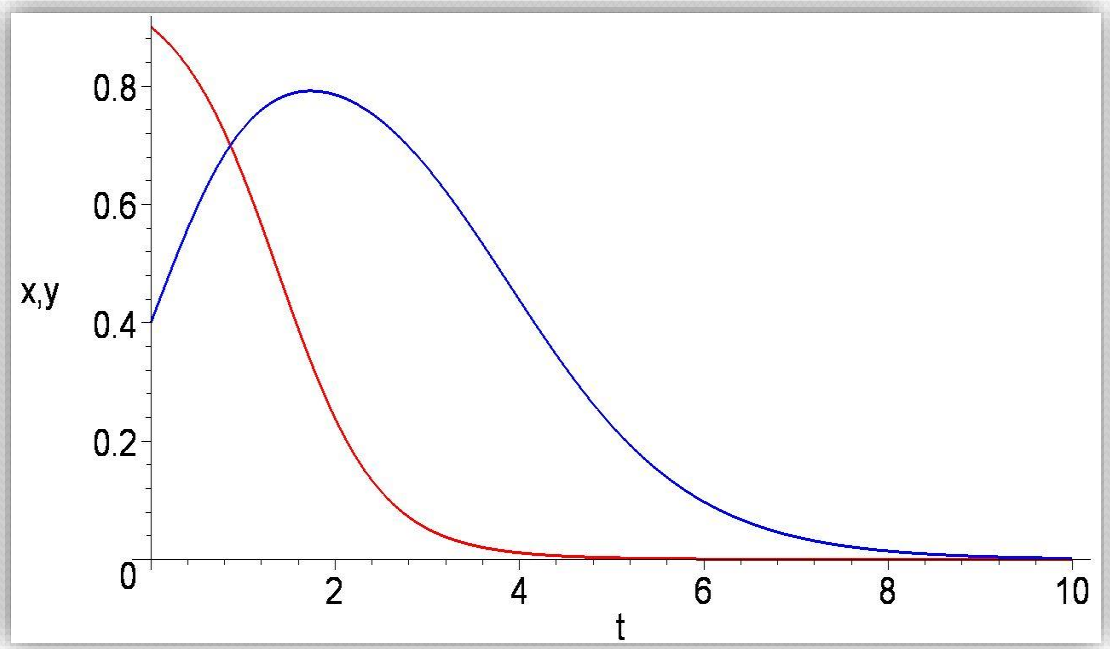
Das gekoppelte System von Differentialgleichung für dieses Beispiel lautet:

$$\frac{dx(t)}{dt} = x(t)\cdot\left(-y(t)+x(t)\cdot y(t)+x(t)-1\right)$$

$$\frac{dy(t)}{dt} = y(t)\cdot\left(3\cdot x(t)-3\cdot x(t)\cdot y(t)+y(t)-1\right)$$

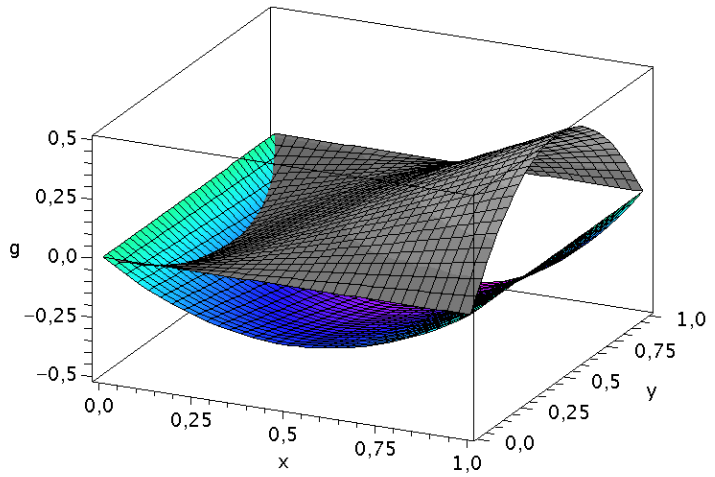
Die Lösung der Gleichung erfolgt wiederum durch Integration bzw. kann mithilfe des Computers nummerisch berechnet werden. Es wurde der Anfangswert zu
(x(0),y(0)) =(0.9 , 0.4) gewählt.

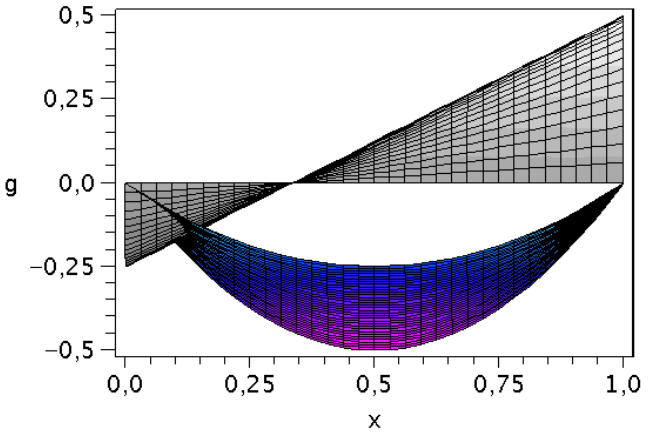
Die rechte Abbildung zeigt eine nummerische Lösung der obigen Gleichung, wobei der Anfangswert zu (x(0),y(0)) =(0.9 , 0.4) gewählt wurde. Bei der Klasse der „Eckspiele“ gibt es eine evolutionär stabile Strategie, die unabhängig von der gewählten Anfangsbedingung stets von der Population angestrebt wird.

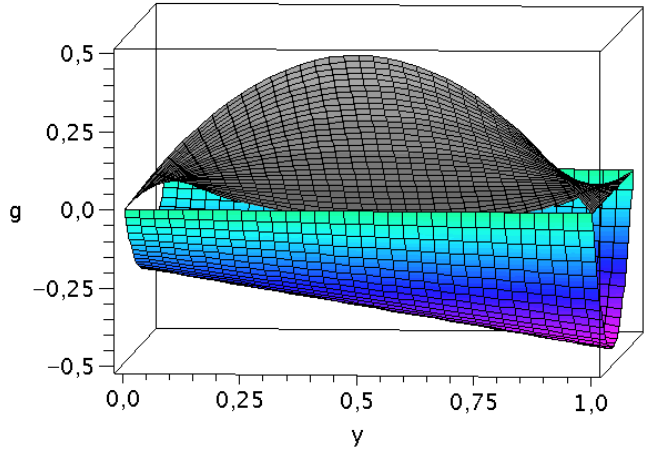


Eigenschaften der Funktionen $g_A(x,y)$ und $g_B(x,y)$

Die das Bimatrix Spiel bestimmenden Funktionen $g_A(x,y)$ (farbige Fläche) und $g_B(x,y)$ (graue Fläche) sind in den unteren Abbildungen veranschaulicht. Das Spiel der Gruppe A gehört der Klasse der dominanten Spiele an, das der Gruppe B der Klasse der Koordinationsspiele.







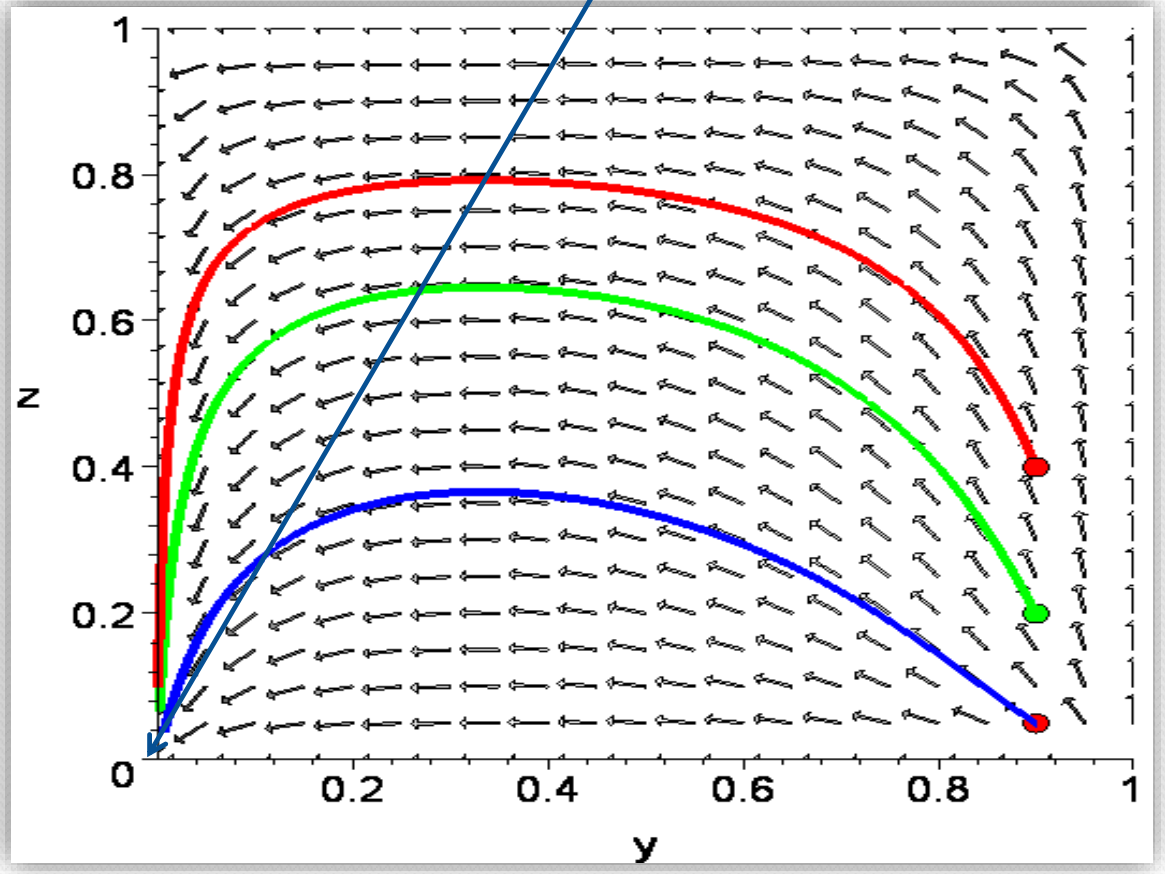
Beispiel 3:

Klasse der Eckspiele (Corner Class)

	Strat. 1	Strat. 2
Strat. 1	(-2 , 2)	(0 , 0)
Strat. 2	(0 , 0)	(1 , 1)

Das Phasenportrait des dritten Beispiels besitzt das folgende Aussehen:

Einziges gemeinsames Nash-Gleichgewicht des Spiels

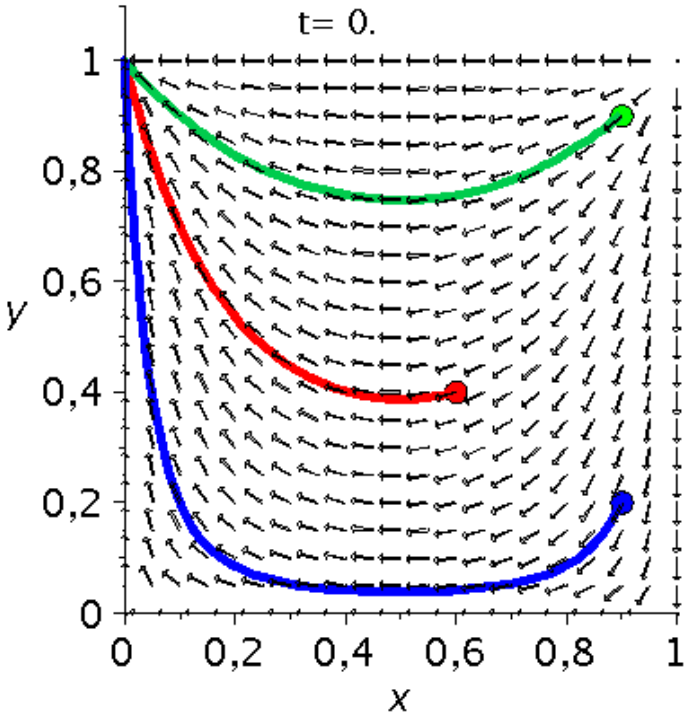


Die rechte Abbildung zeigt das zeitliche Verhalten von drei nummerischen Lösungen mit unterschiedlichen Anfangsbedingungen. Dieses Bimatrixspiel besitzt eine evolutionär stabile Strategie, da es nur ein gemeinsames symmetrisches Nash-Gleichgewicht gibt ((x,y)=(0,0)). Der rote Spieler besitzt sogar bei (0,0) eine dominante Strategie.

Klassifizierung von Bi-Matrix Spielen

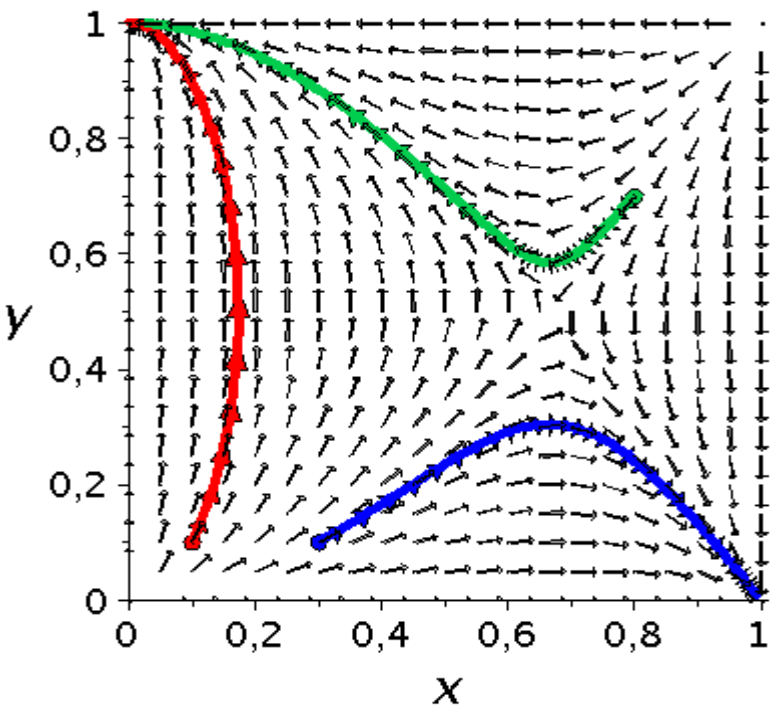
Eckspiele

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



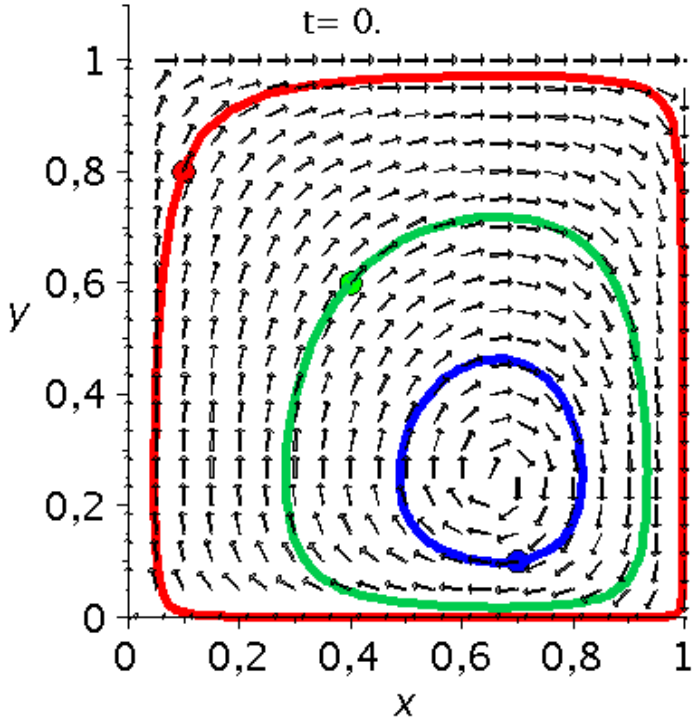
Sattelspiele

Spiel A: *Koordinationspiel*
Spiel B: *Koordinationspiel*
oder
Spiel A: *Anti-Koordinationspiel*
Spiel B: *Anti-Koordinationspiel*



Zentrumsspiele

Spiel A: *Koordinationspiel*
Spiel B: *Anti-Koordinationspiel*
oder
Spiel A: *Anti-Koordinationspiel*
Spiel B: *Koordinationspiel*



Bi-Matrix Spiele mit Python V1

```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib

# Definition der Funktionen g_A und g_B
def gA(x,y,a,b,c,d):
  g=((a+d-c-b)*y + (b-d))*(x-x*x)
  return g
def gB(x,y,a,b,c,d):
  g=((a+d-c-b)*x + (b-d))*(y-y*y)
  return g

# Loesen der DGL
def solve_dgl(numpoints,tend,x0,y0,Aa,Ab,Ac,Ad,Ba,Bb,Bc,Bd):
  sol=np.empty([numpoints,3])
  t=np.linspace(0,tend,numpoints)
  dt=t[1]-t[0]
  sol[0].flat[0] = t[0]
  sol[0].flat[1] = x0
  sol[0].flat[2] = y0
  i = 1
  while i < len(sol):
    sol[i].flat[0] = t[i]
    dx=gA(sol[i-1,1],sol[i-1,2],Aa,Ab,Ac,Ad)*dt
    dy=gB(sol[i-1,1],sol[i-1,2],Ba,Bb,Bc,Bd)*dt
    sol[i].flat[1] = sol[i-1,1] + dx
    sol[i].flat[2] = sol[i-1,2] + dy
    i = i + 1
  return sol

# Groessenfestlegung der Labels usw. im Bild
params = {
    'text.usetex'       : True,
    'axes.titlesize' : 22,
    'axes.labelsize' : 20,
    'xtick.labelsize' : 20 ,
    'ytick.labelsize' : 20
}
matplotlib.rcParams.update(params)
```

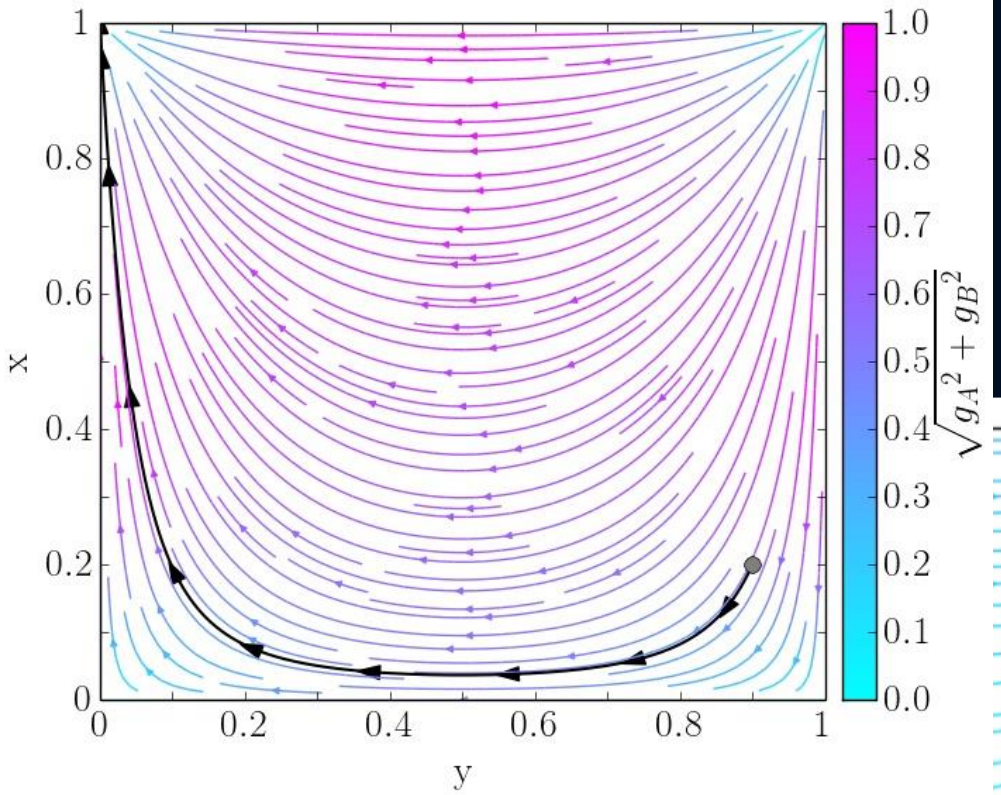
```python
#Festlegung der Auszahlungsmatrix des unsymmetrischen (2x2)-Spiels, Anfangspopulation, Endzeit
Aa=10
Ab=4
Ac=9
Ad=5
Ba=10
Bb=4
Bc=7
Bd=5
tend=12
x0=0.6
y0=0.1

# Weitere Festlegungen
numpoints=1000

# Loesung der DGL fuer eine Anfangspopulation (x0,y0)
Loes=solve_dgl(numpoints,tend,x0,y0,Aa,Ab,Ac,Ad,Ba,Bb,Bc,Bd)

# Plotten des Bildes
plt.plot(Loes[:,1],Loes[:,2],c="black", linewidth=1.5, linestyle='-')
plt.plot(Loes[0,1],Loes[0,2], marker='o', color='grey', markersize=8)

# Achsenbeschriftungen usw.
plt.xlim(0,1)
plt.ylim(0,1)
plt.ylabel(r"$\rm y$")
plt.xlabel(r"$\rm x$")

#Ausgabe des Wertes des Populationsvektors zur Endzeit im Terminal
print "Zur Zeit t=", Loes[numpoints-1,0]
print "Der Wert von x ist: x=", Loes[numpoints-1,1]
print "Der Wert von y ist: y=", Loes[numpoints-1,2]


#Speichern der Bilder als .jpg- und .pdf-Datei
saveFig="./bimatrix.jpg"
plt.savefig(saveFig, dpi=100,bbox_inches="tight",pad_inches=0.05,format="jpg")
plt.show()
```

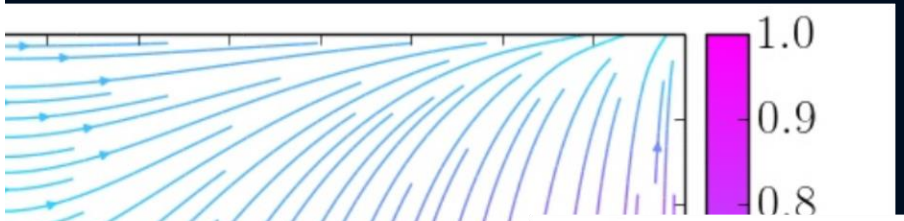
Bi-Matrix Spiele mit Python V2 (Feldliniendiagramm)

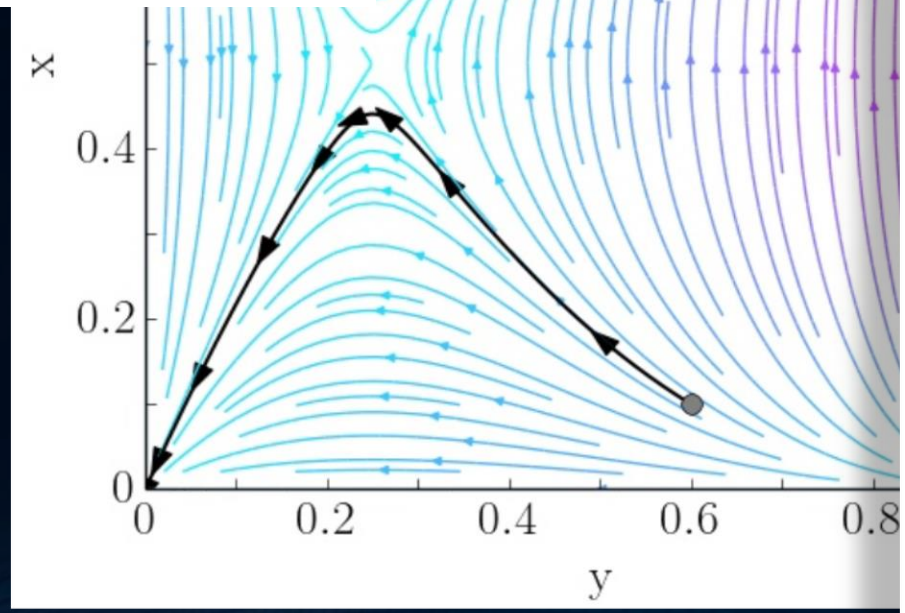
```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib

# Definition der Funktionen g_A und g_B
def gA(x,y,a,b,c,d):
  g=((a+d-c-b)*y + (b-d))*(x-x*x)
  return g
def gB(x,y,a,b,c,d):
  g=((a+d-c-b)*x + (b-d))*(y-y*y)
  return g

# Groessenfestlegung der Labels usw. im Bild
params = {
    'text.usetex'        : True,
    'axes.titlesize' : 22,
    'axes.labelsize' : 20,
    'xtick.labelsize' : 20 ,
    'ytick.labelsize' : 20
}
matplotlib.rcParams.update(params)

#Festlegung der Auszahlungsmatrix des
Aa=10
Ab=4
Ac=9
Ad=5
Ba=10
Bb=4
Bc=7
Bd=5
tend=12
x0=0.6
y0=0.1
```

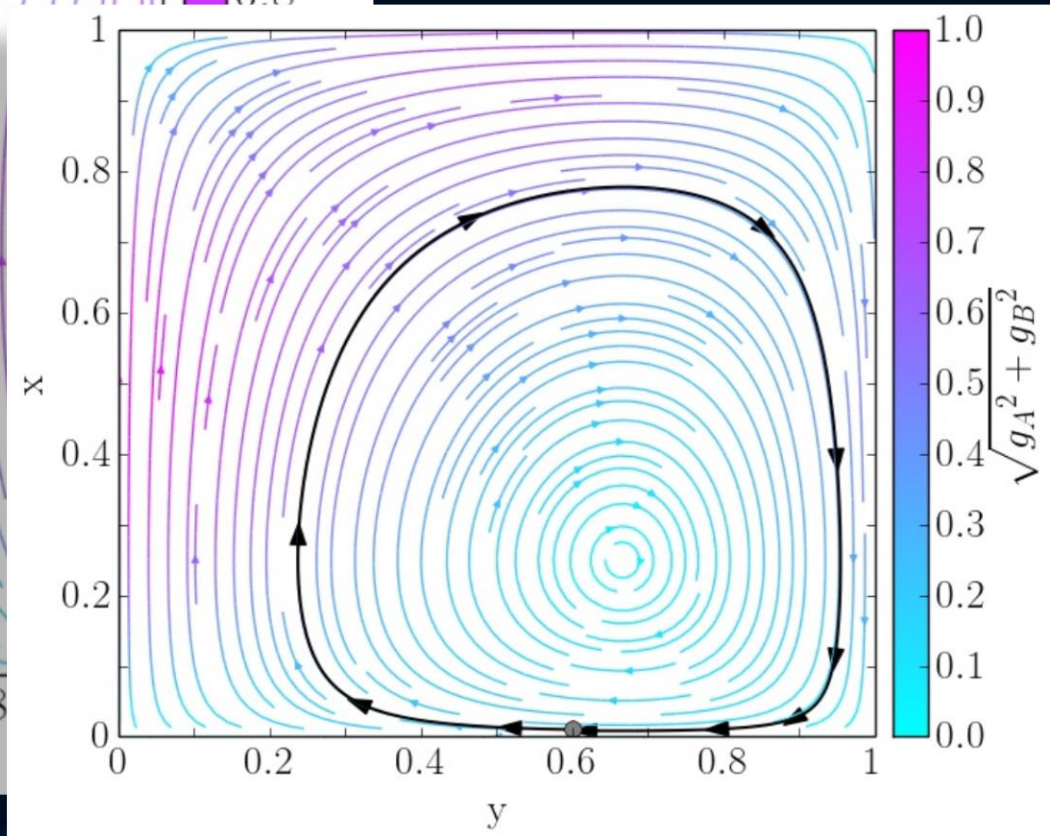
```python
# Fuer die Darstellung des Feldliniendiagramms streamplot
SY,SX=np.mgrid[0:1:100j,0:1:100j]
SgA=gA(SX,SY,Aa,Ab,Ac,Ad)
SgB=gB(SX,SY,Ba,Bb,Bc,Bd)
# Die Farbe wird die Geschwindigkeit der Aenderung des Populationsvektors anzeigen
speed=np.sqrt(SgA*SgA + SgB*SgB)
colorspeed = speed/speed.max()

# Plotten des Bildes
strm = plt.streamplot(SX,SY,SgA,SgB,density=[2, 2], linewidth=1,color=colorspeed, cmap=plt.cm.cool)
# Erzeugung der nebenstehenden Farblegende colorbar
cbar=plt.colorbar(strm.lines,pad=0.02)
cbar.set_label(r'$\sqrt{{g_A}^2 + {g_B}^2}$',size=20)

# Achsenbeschriftungen usw.
plt.xlim(0,1)
plt.ylim(0,1)
plt.ylabel(r"$\rm y$")
plt.xlabel(r"$\rm x$")

#Speichern der Bilder als .jpg- und .pdf-Datei
saveFig="./bimatrix.jpg"
plt.savefig(saveFig, dpi=100,bbox_inches="tight",pad_inches=0.05,format="jpg")
plt.show()
```

Bi-Matrix Spiele mit Python
Version bimatrix1.py









E-Learning

Zusatzmaterial auf der Online-Lernplatform Lon Capa

Zusäzlich zu den Informationen aus dieser Internetseite wurde ein separater Kurs auf der Online-Lernplatform Lon Capa eerstellt. Er beinhaltet neben den hier dargestellten Informationen zusäzliche Erläuterungen, diverse interaktive Übungsaufgaben, Feedbackmöglichkeiten und Probeklausuren. Falls Sie schon einen Lon Capa Account besitzen können Sie sich einfach mit diesem unter dem unten angegebenen Link einloggen. Falls Sie Student der Universiät Frankfurt sind, können Sie sich mit Ihrem HRZ-Account einloggen. Anderenfalls kontaktieren Sie bitte per E-Mail und ich werde für Sie einen Account für die Lernplatform erstellen.

Hier gehts zu Lon Capa

Aufgaben auf Lon-Cappa

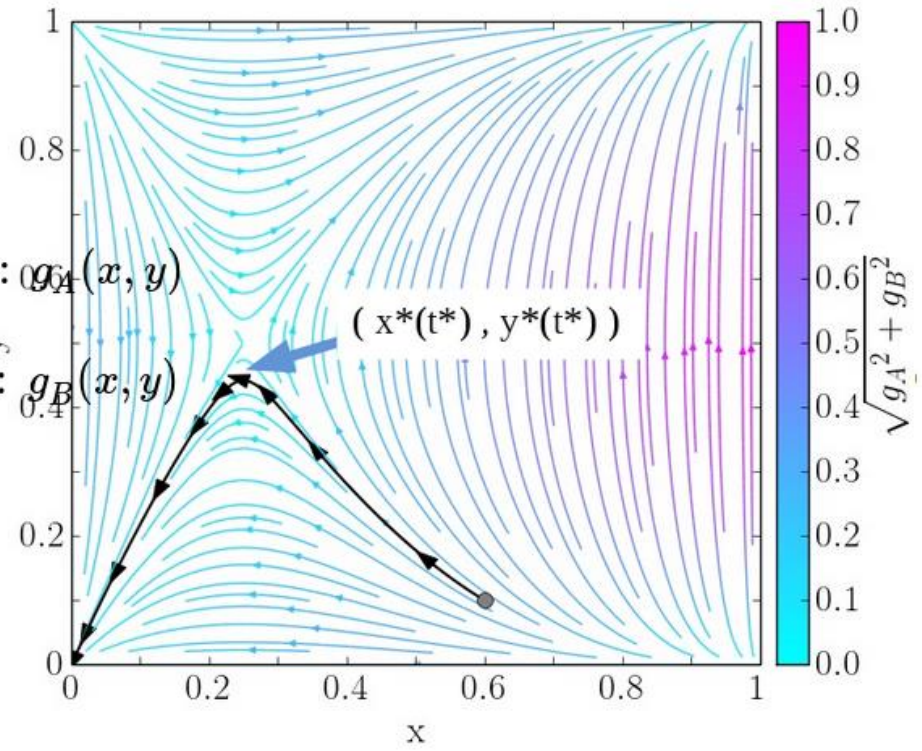
Das zeitliche Verhalten der Komponenten der Populationsvektoren (Gruppe A: $x(t) := x_1^A(t)$ und Gruppe B: $y(t) := x_1^B(t)$) wird in der Reproduktionsdynamik mittels des folgenden Systems von Differentialgleichungen beschrieben:

$$\frac{dx(t)}{dt} = \left[\left(\$_{11}^A + \$_{22}^A - \$_{12}^A - \$_{21}^A\right) y(t) + \left(\$_{12}^A - \$_{22}^A\right)\right]\left(x(t) - (x(t))^2\right) =: g_A(x,y)$$

$$\frac{dy(t)}{dt} = \left[\left(\$_{11}^B + \$_{22}^B - \$_{12}^B - \$_{21}^B\right) x(t) + \left(\$_{12}^B - \$_{22}^B\right)\right]\left(y(t) - (y(t))^2\right) =: g_B(x,y)$$

Das durch die folgende Auszahlungstabelle definierte Bimatrix Spiel gehört der Klasse der Sattelpunktsspiele an.

A/B	s_1^B	s_2^B
s_1^A	(10 , 10)	(4 , 7)
s_2^A	(9 , 4)	(5 , 5)



t^* = [ ] , y^* = [ ] (bitte geben Sie den numerischen Wert auf mindestens 4 Nachkommastellen an; z.B. 0.4135)

Antwort einreichen Versuche 0/5

Der Populationsvektor zur Zeit t=0 sei (x(0)=0.6 , y(0)=0.0565). Der Anteil der Spieler in der Gruppe B die die Strategie s_1^B spielen nimmt zunächst zu, erreicht dann ein Maximum und nimmt dannach wieder ab (siehe nebenstehende Abbildung). Berechnen Sie den Zeitpunkt t^* an dem der maximale Wert y^* erreicht wird.

I.2.6 Mögliche Projekte im Teil I

Projekt I: Spielklassen symmetrischer (2x3)-Spiele

E. C. Zeeman zeigt in seinem im Jahre 1980 veröffentlichten Artikel, dass man evolutionäre, symmetrische (2x3)-Spiele in 19 Klassen einteilen kann. Es gibt Spielklassen, die besitzen lediglich eine evolutionär stabile Strategie und Klassen die sogar drei evolutionär stabile Strategien besitzen. (siehe E.C. Zeeman, POPULATION DYNAMICS FROM GAME THEORY, In: Global Theory of Dynamical Systems, Springer 1980). Stellen Sie Beispiele dieser Spielklassen mittels eines Maple Worksheets oder eines Python Skriptes dar. Das folgende Maple Worksheet kann Ihnen dabei helfen (View Maple Worksheet, Download Maple Worksheet).

Projekt II: Äquivalenz der Räuber-Beute-Gleichung für 2-Populationen mit der Replikatordynamik der evolutionären Spieltheorie für 3 Strategien

Stellen Sie die Äquivalenz der Räuber-Beute-Gleichung für 2-Populationen mit einem evolutionären Spiel mit 3 Strategien mittels eines Maple Worksheets oder eines Python Skriptes dar.