Physik der sozio-ökonomischen Systeme *mit dem Computer*

JOHANN WOLFGANG GOETHE UNIVERSITÄT
07.11.2025

MATTHIAS HANAUSKE

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

4. Vorlesung

Plan für die heutige Vorlesung

- Kurze Wiederholung der Inhalte der 3. Vorlesung

- 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)
    - Zentrumspiele (Center Class Games)

Inhalte Vorlesung 3

- Symmetrische und nicht symmetrische Spiele
- Klassen symmetrischer (2x2)-Spiele
- Beispiel eines asymmetrischen Spiels: Kampf der Geschlechter
- Einführung in die Evolutionäre Spieltheorie
  - Die Replikatordynamik der Quasispezies
  - Die Differentialgleichung eines evolutionären (2xm)-Spiels
  - Die Differentialgleichung eines evolutionären (2x2)-Spiels (Bi-Matrix Spiel)
  - Die Differentialgleichung eines evolutionären, symmetrischen (2x2)-Spiels
  - Definition: Evolutionär stabile Strategien (ESS)

Klassen symmetrischer (2x2)-Spiele

	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)

Symmetrisches (2x2)-Spiel

Symmetrische (2x2)-Spiele lassen sich in drei unterschiedliche Klassen gliedern:

1. Dominante Spiele
2. Koordinationspiele
3. Anti-Koordinationsspiele

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 $(x,y)=(1,1)$.

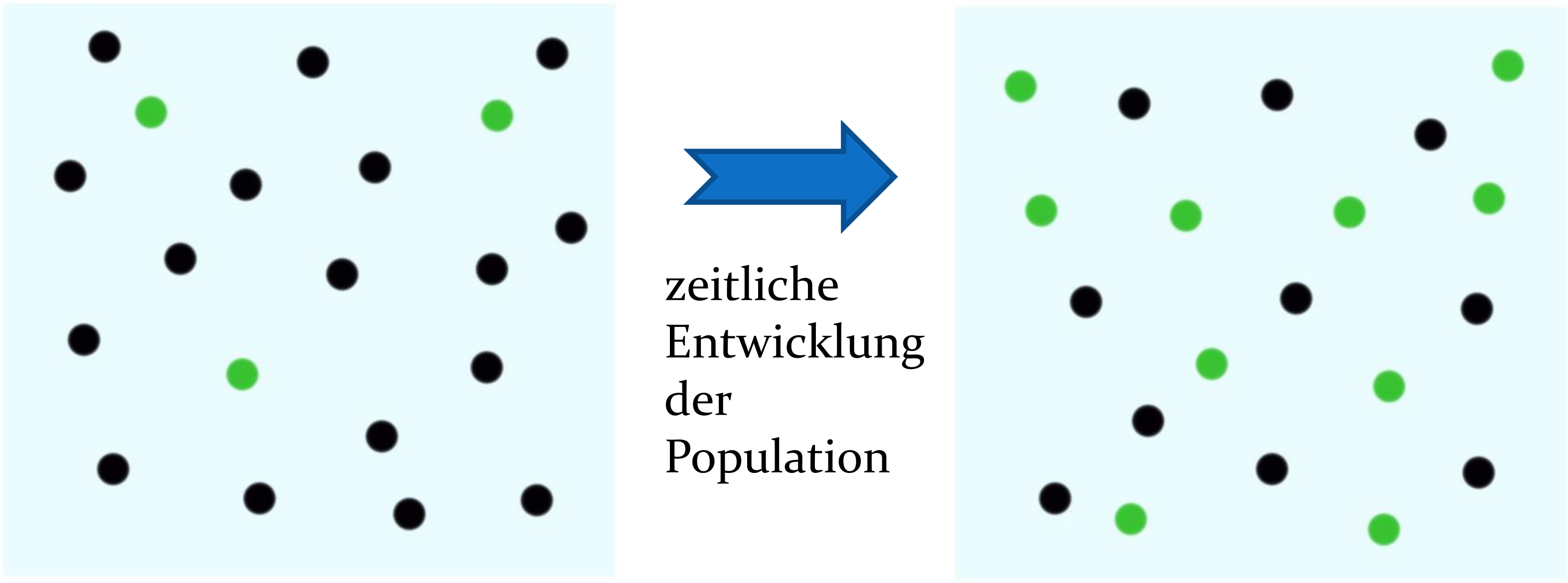
Anti-Koordinationsspiele (a < c und b > d)

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 $(x,y)=(0,1)$.

Evolutionäre Spieltheorie

Symmetrische (2x2)-Spiele einer Population

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



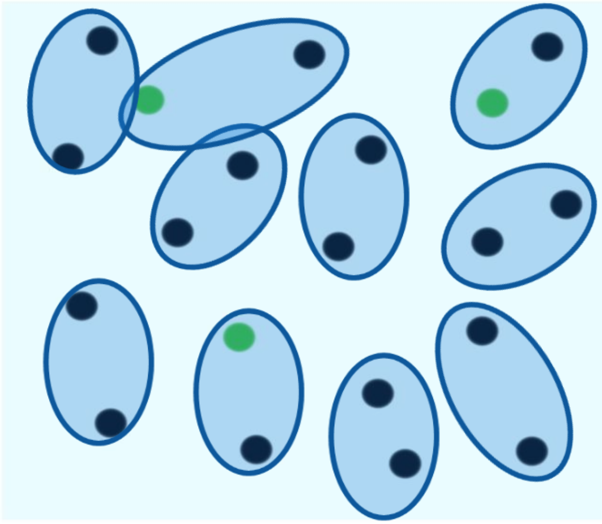
$x(0)=0.15$ $x(10)=0.5$

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

Evolutionäre Spieltheorie

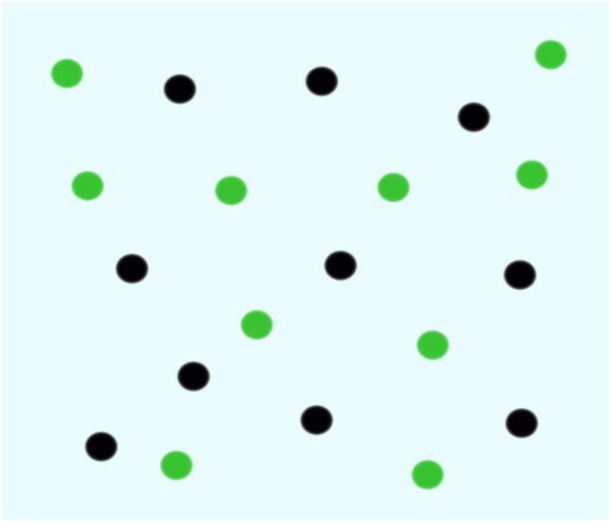
Symmetrische (2x2)-Spiele einer Population

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



x(0)=0.15

Die Anfangspopulation von Spielern spielt zum Zeitpunkt t=0 das erste Mal das Spiel. Die Spieler wählen im Mittel zu 15% die grüne Strategie.



x(10)=0.5

Das evolutionäre Spiel schreitet voran und die grüne Strategie wird für die Spieler zunehmend attraktiver. Zum Zeitpunkt t=10 spielen schon 50% grün.

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

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

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:

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

Replikatordynamik

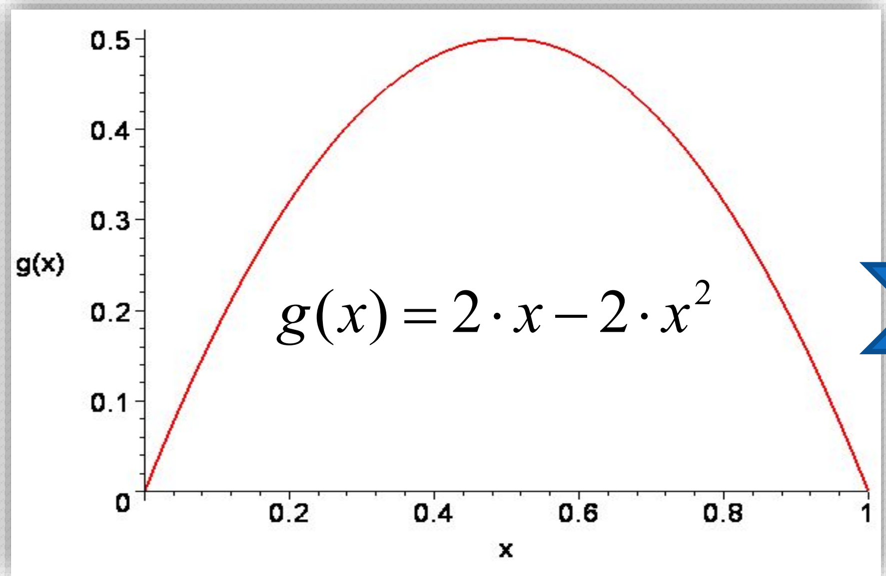
(für das Gefangenendilemma)

	Ge	Sc
Ge	(-7 , -7)	(-1 , -9)
Sc	(-9 , -1)	(-3 , -3)

Die Differentialgleichung der Replikatordynamik für das Gefangenendilemma lautet:

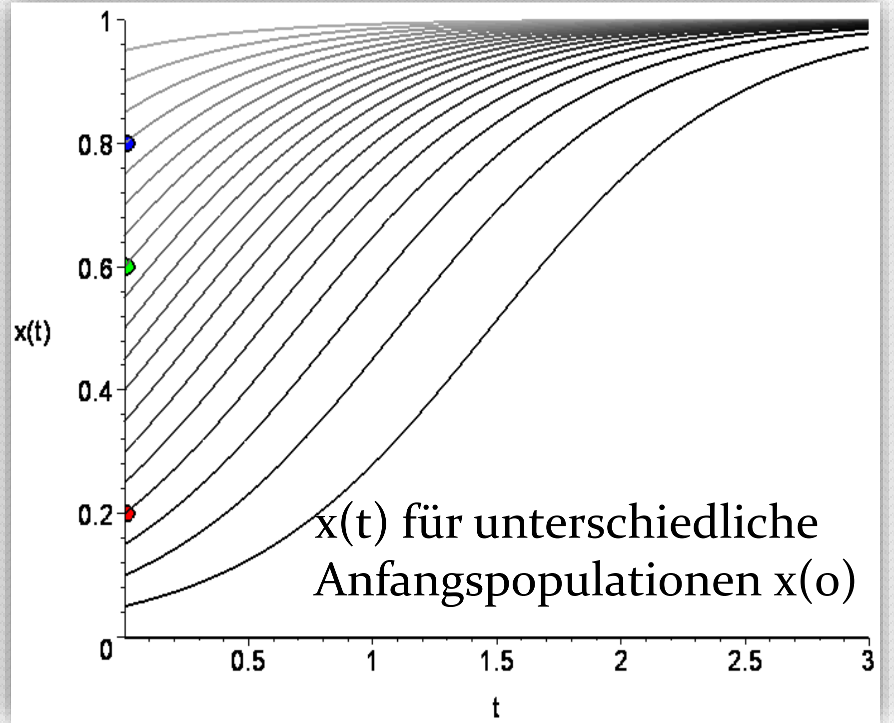
$$\dot{x}(t) := \frac{dx(t)}{dt} = 2 \cdot x(t) - 2 \cdot \left(x(t)\right)^2 = g(x(t))$$

$$g(x(t)) := x(t) \cdot \left((-7-(-9)) \cdot (x(t) - x(t)^2) + (-3-(-1)) \cdot (2 \cdot x(t) - 1 - x(t)^2) \right)$$



$$g(x) = 2 \cdot x - 2 \cdot x^2$$

Beispiel: Gefangenendilemma
g(x)=g(x(t)) im Bereich [0,1] dargestellt



x(t) für unterschiedliche Anfangspopulationen x(0)

Evolutionär stabile Strategien (ESS)

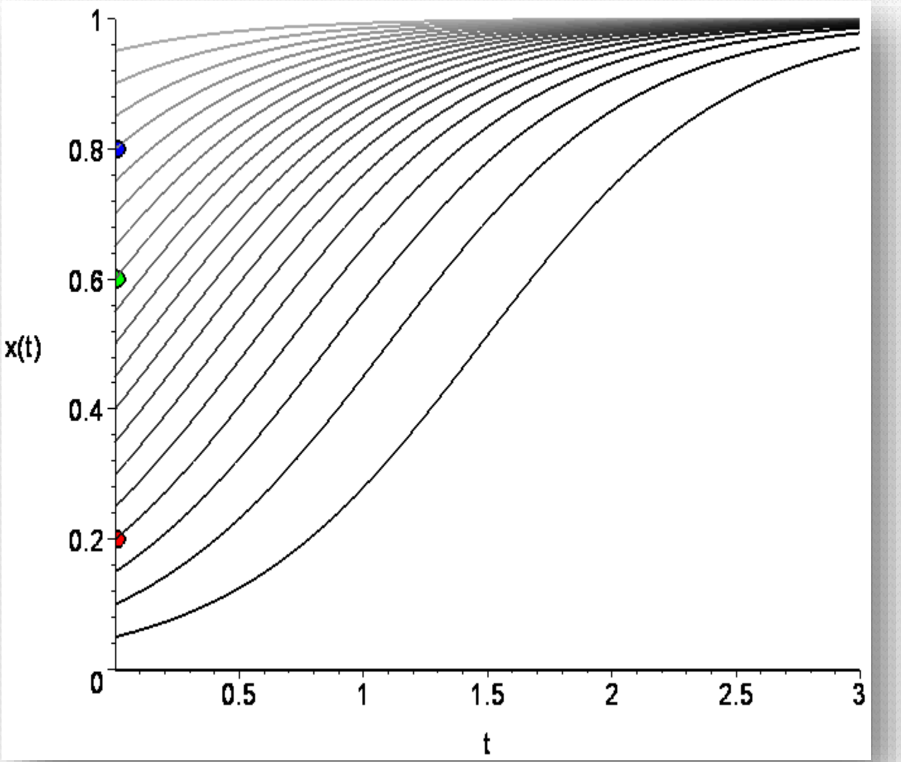
- Evolutionär stabile Strategien sind die stabilen Endzustände der Häufigkeitsverteilung x(t):

$$\underset{t\to\infty}{Limes}\left(x(t)\right)$$

Eine notwendige (aber nicht hinreichende) Bedingung für die Existenz einer ESS ist, dass diese ein Nash – Gleichgewicht des zugrundeliegenden Spiels ist.

Beispiel:
Gefangenendilemma ähnliche Spiele
Für die ESS des evolutionären Gefangenendilemma – Spiels ergibt sich:

$$\underset{t\to\infty}{Limes}\left(x(t)\right)=1 \quad\Rightarrow\quad \text{alle "gestehen"}$$



Replikatordynamik

(für das Hirschjagt-Spiel)

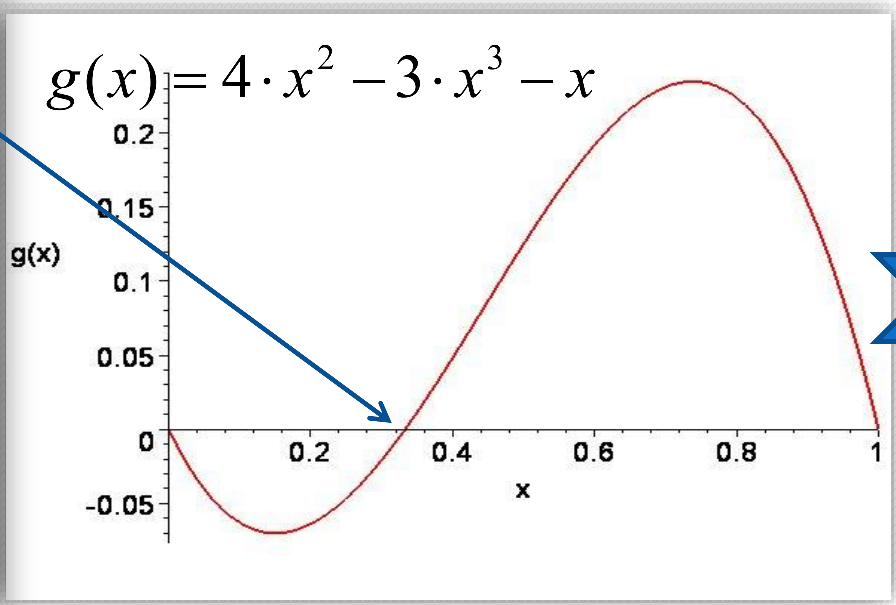
	Hasen	Hirsch
Hasen	(2 , 2)	(4 , 0)
Hirsch	(0 , 4)	(5 , 5)

Die Differentialgleichung der Replikatordynamik für das Hirschjagt-Spiel lautet:

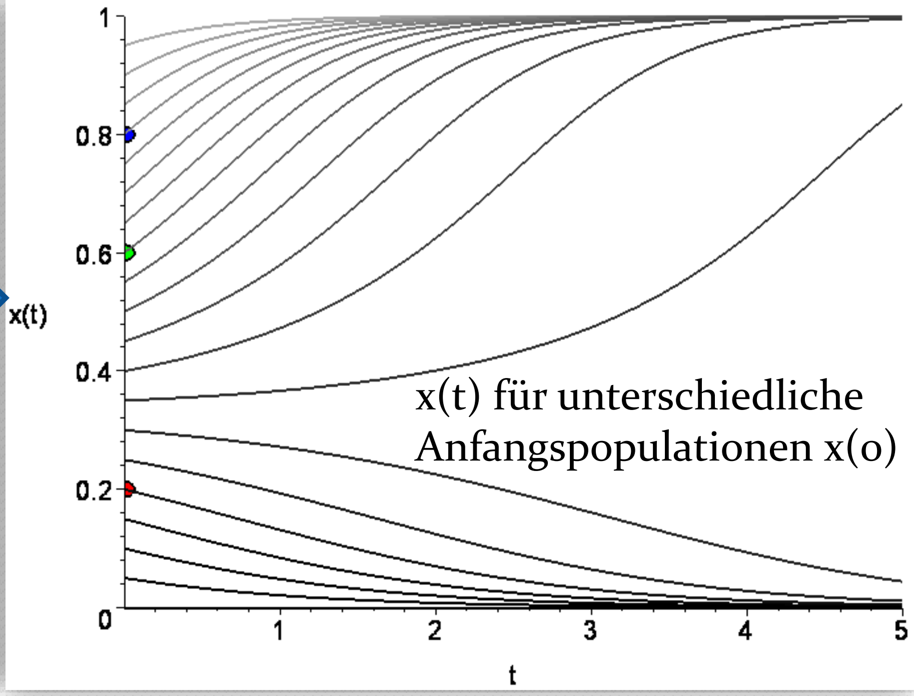
$$\dot{x}(t) := \frac{dx(t)}{dt} = 4 \cdot (x(t))^2 - 3 \cdot (x(t))^3 - x(t) - = g(x(t))$$

$$g(x(t)) := x(t) \cdot \left((2-0) \cdot (x(t) - x(t)^2) + (5-4) \cdot (2 \cdot x(t) - 1 - x(t)^2)\right)$$

Die interne Nullstelle der Funktion g(x) entspricht dem Wert des gemischten Nash-Gleichgewichts und bestimmt den separations-Anfangswert der Population



Beispiel: Hirschjagt-Spiel
g(x)=g(x(t)) im Bereich [0,1] dargestellt



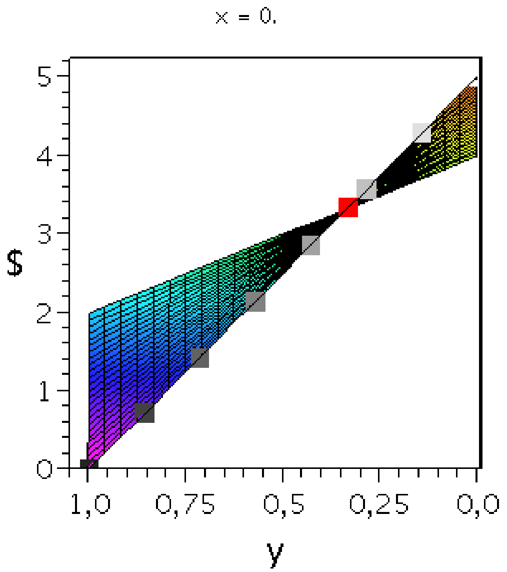
Das Nash-Gleichgewicht in gemischten Strategien

Ein Spezialfall des Nash-Gleichgewichts besteht, falls die partielle Ableitung der gemischten Auszahlungsfläche verschwindet. Man nennt dann ein solches Nash-Gleichgewicht $(\tilde{s}^{A\star}, \tilde{s}^{B\star})$ ein internes Nash-Gleichgewicht bzw. ein Nash-Gleichgewicht in gemischten Strategien.

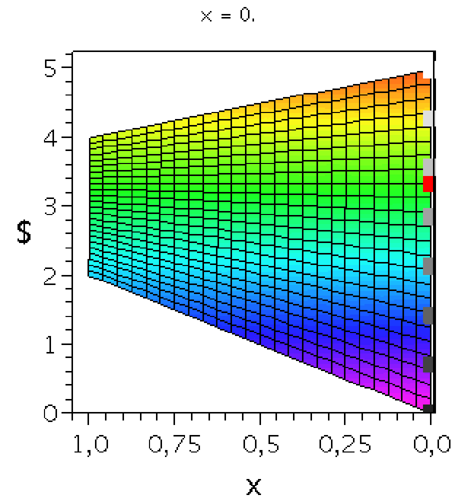
Nash-Gleichgewicht in gemischten Strategien:

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

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



Das gemischte Nash-Gleichgewicht im Hirschjagd-Spiel liegt bei der Strategien-kombination (x=1/3 , y=1/3). Das gemischte Nash-Gleichgewicht entspricht der internen Nullstelle der Funktion g(x).



Replikatordynamik

(für das Angsthasen-Spiel)

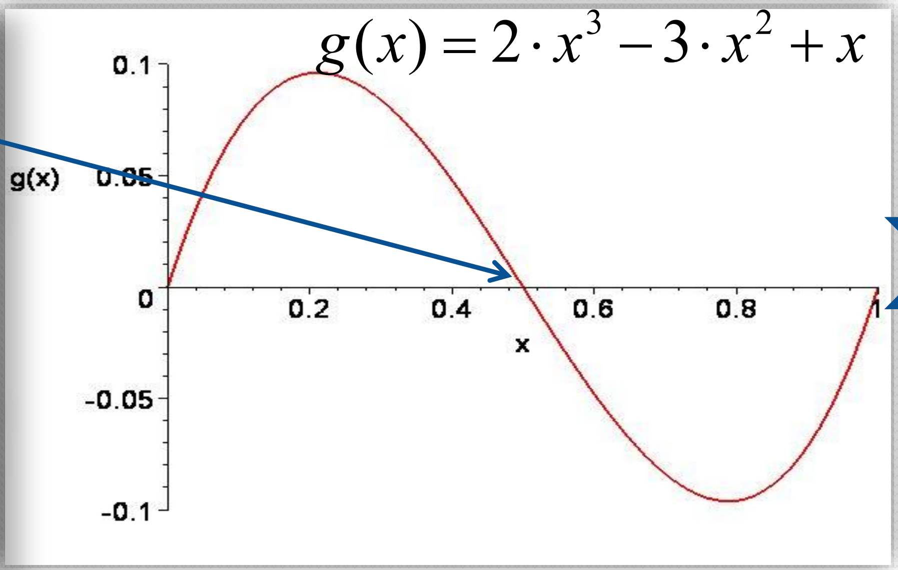
	Springe nicht	Springe
Springe nicht	(-1 , -1)	(2 , 0)
Springe	(0 , 2)	(1 , 1)

Die Differentialgleichung der Replikatordynamik für das Angsthasen-Spiel lautet:

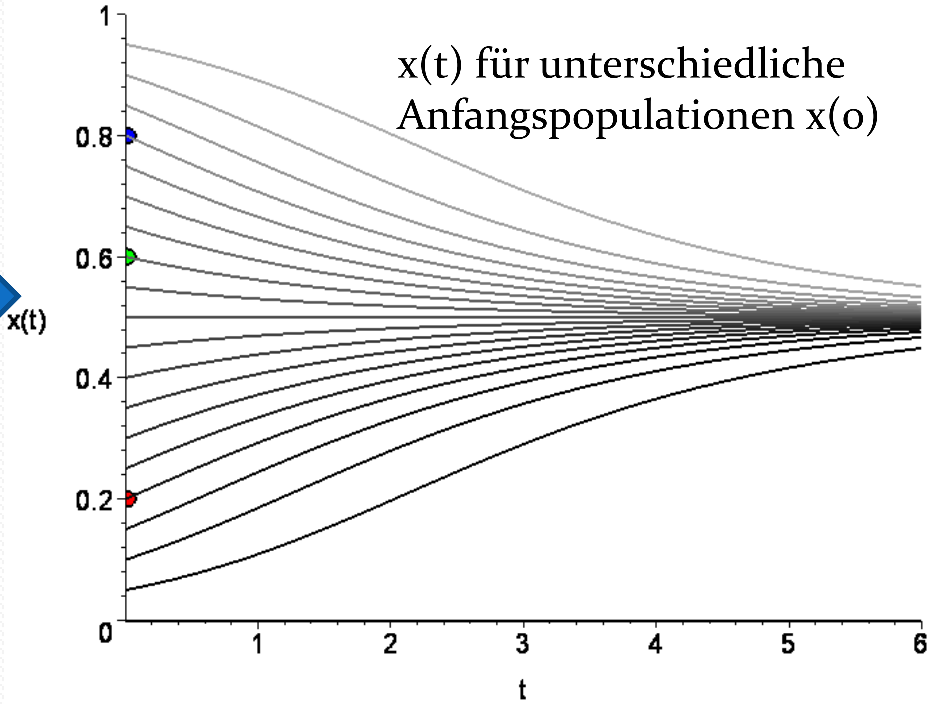
$$\dot{x}(t) := \frac{dx(t)}{dt} = 2 \cdot (x(t))^3 - 3 \cdot (x(t))^2 + x(t) = g(x(t))$$

$$g(x(t)) := x(t) \cdot \left((-1-0) \cdot (x(t) - x(t)^2) + (1-2) \cdot (2 \cdot x(t) - 1 - x(t)^2)\right)$$

Die interne Nullstelle der Funktion g(x) entspricht dem Wert des gemischten Nash-Gleichgewichts und bestimmt die Lage der ESS



Beispiel: Angsthasen-Spiel
g(x)=g(x(t)) im Bereich [0,1] dargestellt



Evolutionär Stabile Strategien

- Betrachten Sie die folgenden Beispiele:

Beispiel 1:

	Kugel	Keine Kugel
Kugel	(0 , 0)	(2 , -1)
Keine Kugel	(-1 , 2)	(1 , 1)

Beispiel 2:

	Kugel	Keine Kugel
Kugel	(-1 , -1)	(3 , 0)
Keine Kugel	(0 , 3)	(1 , 1)

Beispiel 3:

	Kugel	Keine Kugel
Kugel	(0 , 0)	(2 , -2)
Keine Kugel	(-2 , 2)	(3 , 3)

1. Geben Sie mögliche dominante Strategien und Nash-Gleichgewichte der Spiele an.
2. Bestimmen und zeichnen Sie die Funktion g(x) für alle drei Spiele?
3. Berechnen Sie die Nullstellen der Funktion g(x) (g(x)=0).
4. Geben Sie die evolutionär stabilen Strategien der Spiele an?

Dominante Strategien und Nash-Gleichgewichte

Beispiel 1	Kugel	Keine Kugel
Kugel	(0 , 0)	(2 , -1)
Keine Kugel	(-1 , 2)	(1 , 1)

Das erste Spiel besitzt nur ein Nash-Gleichgewicht das gleichzeitig die dominante Strategie des Spiels ist (Kugel, Kugel). Das Spiel gehört zur Klasse der dominanten Spiele.

Beispiel 2	Kugel	Keine Kugel
Kugel	(-1 , -1)	(3 , 0)
Keine Kugel	(0 , 3)	(1 , 1)

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). Das Spiel gehört zur Klasse der Anti-Koordinationsspiele.

Beispiel 3	Kugel	Keine Kugel
Kugel	(0 , 0)	(2 , -2)
Keine Kugel	(-2 , 2)	(3 , 3)

Das dritte Spiel besitzt ebenfalls keine dominante Strategie, aber zwei symmetrische Nash-Gleichgewicht in reinen Strategien ((K,K) und (KK,KK)) und ein gemischtes Nash-Gleichgewicht (0.33 K , 0.67 KK). Das Spiel gehört zur Klasse der Koordinationsspiele.

Nullstellen der Funktion g(x)

Beispiel 1:

$$g(x)=x-x^2$$

$$x-x^2=0$$

$$x\cdot(1-x)=0 \quad\Rightarrow\quad x_1=0$$

$$1-x=0 \quad\Rightarrow\quad x_2=1$$

Die Funktion hat zwei Nullstellen:

$$x_1=0 \text{ und } x_2=1$$

Beispiel 3:
Die Funktion hat drei Nullstellen:

$$g(x)=-3\cdot x^3+4\cdot x^2-x$$

$$x_1=0\ ,\ x_2=1 \text{ und } x_3=\frac{1}{3}$$

Beispiel 2:

$$g(x)=3\cdot x^3-5\cdot x^2+2\cdot x$$

$$x\cdot\left(3\cdot x^2-5\cdot x+2\right)=0 \quad\Rightarrow\quad x_1=0$$

$$3\cdot x^2-5\cdot x+2=0 \ \Big|\ /3$$

$$x^2-\frac{5}{3}\cdot x+\frac{2}{3}=0 \ \Big|\ \text{p - q Formel}$$

$$x_{2/3}=\frac{5}{6}\pm\sqrt{\left(\frac{5}{6}\right)^2-\frac{2}{3}}$$

$$x_{2/3}=\frac{5}{6}\pm\sqrt{\frac{25-24}{36}}$$

$$x_{2/3}=\frac{5}{6}\pm\frac{1}{6} \quad\Rightarrow\quad x_2=1,\ x_3=\frac{2}{3}$$

Die Funktion hat drei Nullstellen:

$$x_1=0\ ,\ x_2=1 \text{ und } x_3=\frac{2}{3}$$

Evolutionäre Strategien (Beispiel 1)

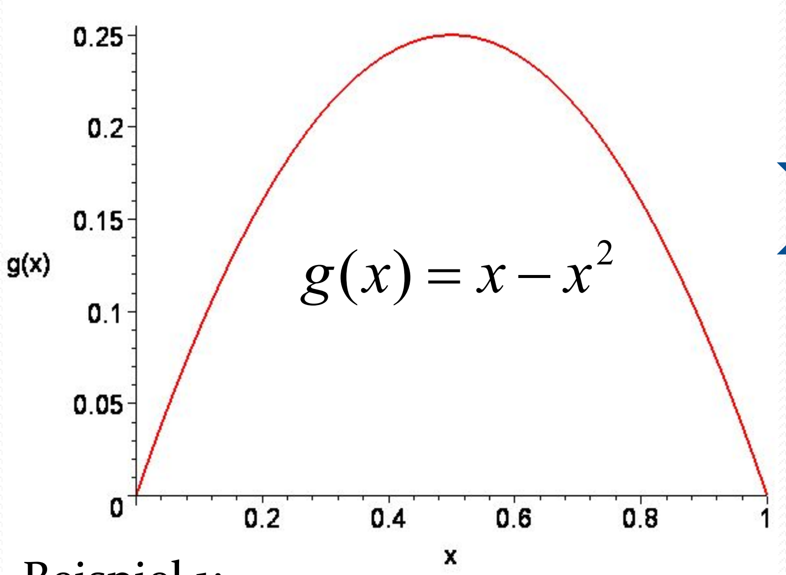
	Kugel	Keine Kugel
Kugel	(0 , 0)	(2 , -1)
Keine Kugel	(-1 , 2)	(1 , 1)

Die Differentialgleichung der Replikatordynamik für das erste Beispiel lautet:

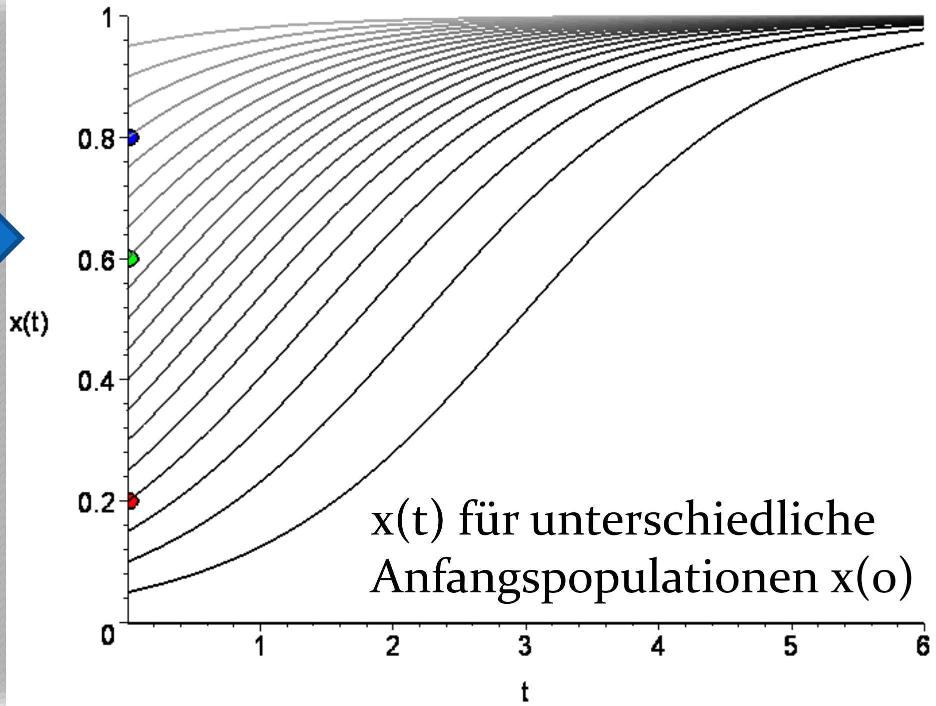
$$\frac{dx(t)}{dt} = g(x(t)) = x(t) - (x(t))^2$$

$\boxed{\text{Eine ESS bei x=1}}$

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 1.



Beispiel 1:
g(x)=g(x(t)) im Bereich [0,1] dargestellt



	Kugel	Keine Kugel
Kugel	(-1 , -1)	(3 , 0)
Keine Kugel	(0, 3)	(1 , 1)

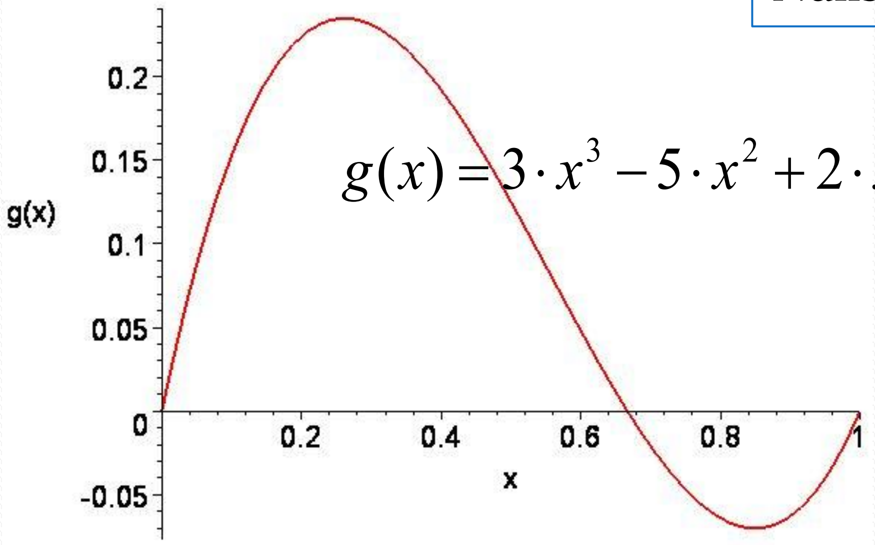
Evolutionäre Strategien (Beispiel 2)

Die Differentialgleichung der Replikatordynamik für das zweite Beispiel lautet:

$$\frac{dx(t)}{dt} = g(x(t)) = 3\cdot(x(t))^3 - 5\cdot(x(t))^2 + 2\cdot x(t)$$

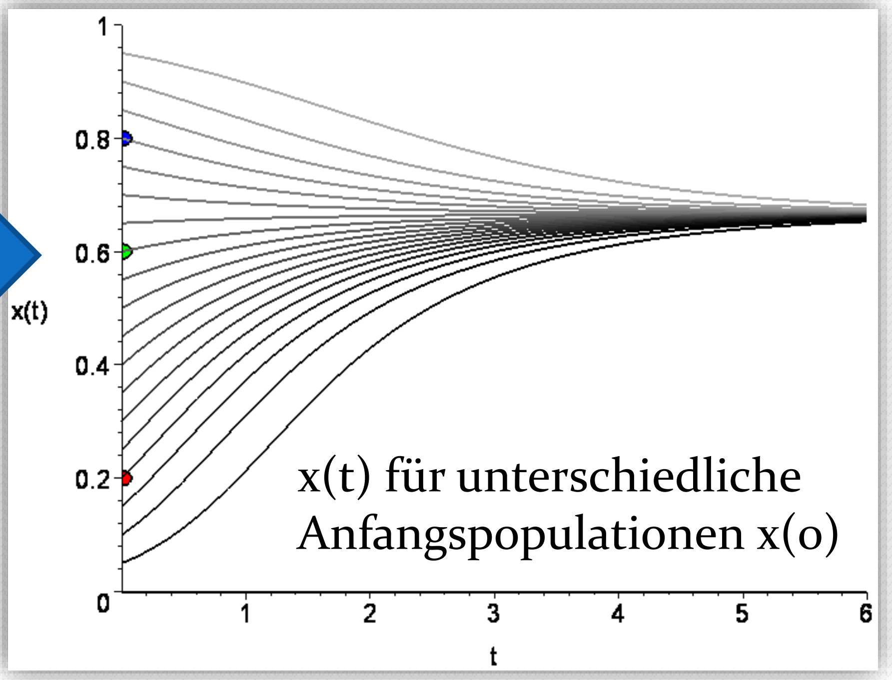
Da es sich bei diesem Beispiel um ein symmetrisches Anti-Koordinationsspiel handelt, strebt der Populationsanteil der Kugel-Spieler unabhängig vom Anfangswert immer zu dem gemischten Nash-Gleichgewicht, was identisch mit der mittleren Nullstelle der Funktion g(x) ist (x=0.67).

Eine ESS bei x=0.67



$$g(x) = 3\cdot x^3 - 5\cdot x^2 + 2\cdot x$$

Beispiel 2:
g(x)=g(x(t)) im Bereich [0,1] dargestellt



x(t) für unterschiedliche Anfangspopulationen x(0)

Evolutionäre Strategien (Beispiel 3)

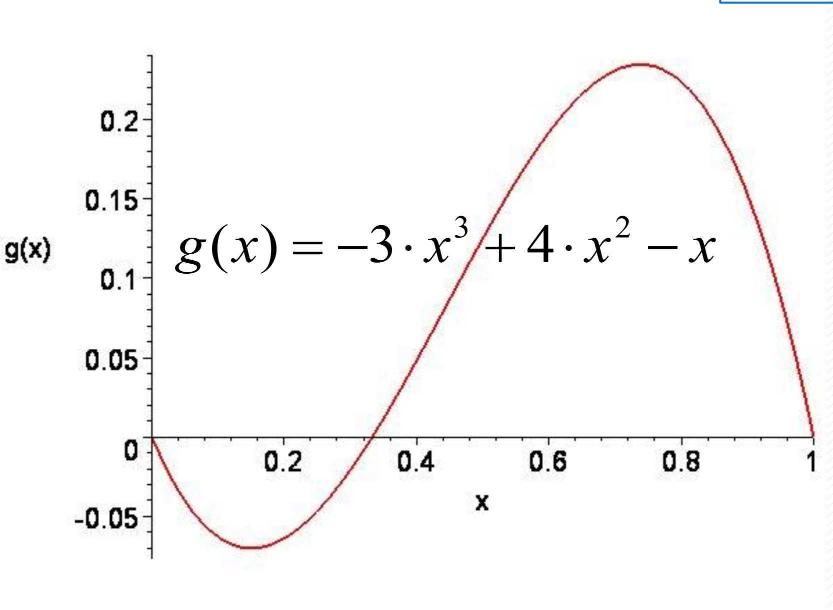
	Kugel	Keine Kugel
Kugel	(0, 0)	(2 , -2)
Keine Kugel	(-2, 2)	(3 , 3)

Die Differentialgleichung der Replikatordynamik für das zweite Beispiel lautet:

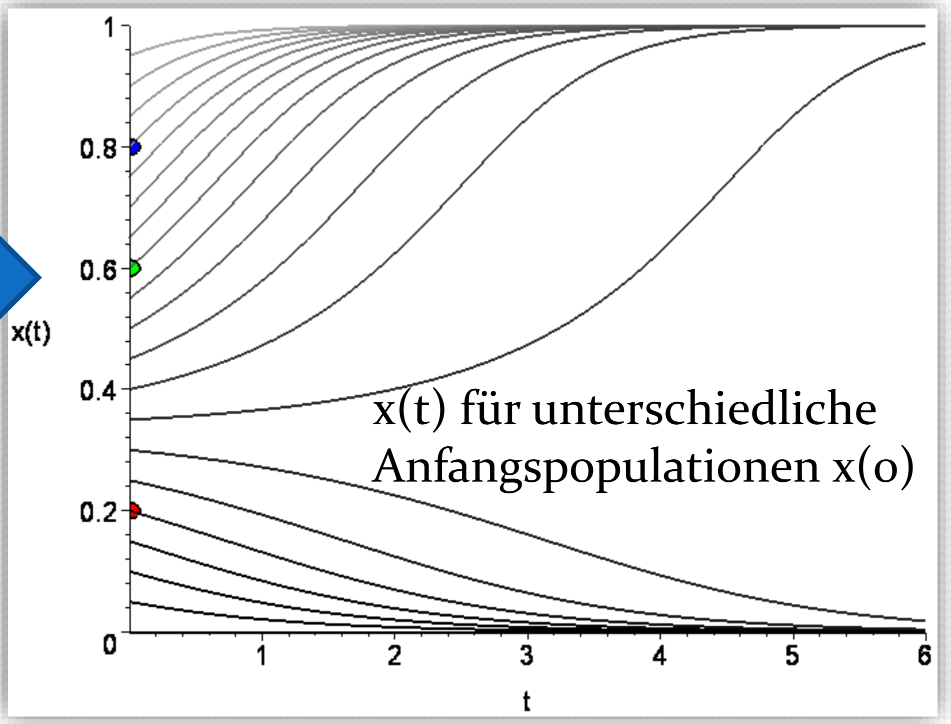
$$\frac{dx(t)}{dt} = g(x(t)) = -3\cdot(x(t))^3 + 4\cdot(x(t))^2 - x(t)$$

Zwei ESSs : (x=1 und x=0)

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).

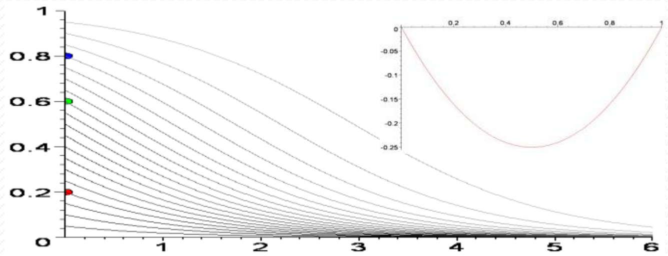


Beispiel 3:
g(x)=g(x(t)) im Bereich [0,1] dargestellt

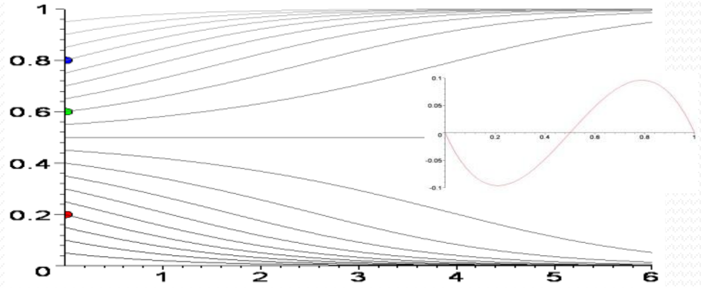


Klassifizierung von evolutionären, symmetrischen (2x2)-Spielen

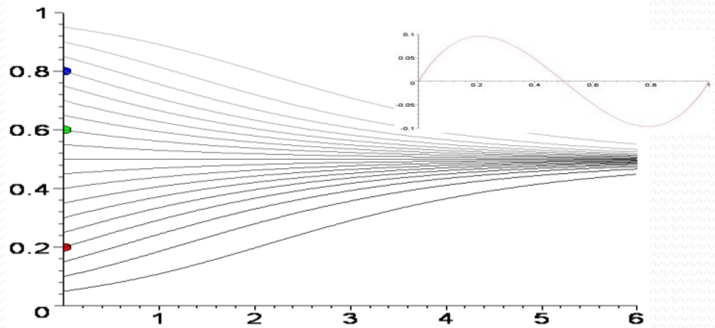
- **Dominante Spiele**
  (2. Strategie dominiert 1.Strategie)
  Es existiert ein Nash - Gleichgewicht, welches die anderen Strategien dominiert. ESS bei x=0.



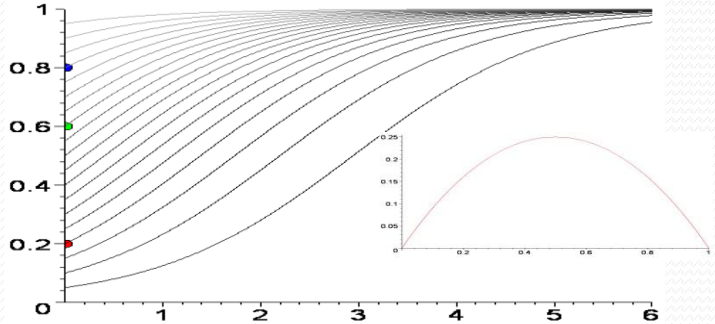
- **Koordinationsspiele**
  Es existieren drei Nash – Gleichgewichte und zwei reine ESS, die abhängig von der Anfangsbedingung realisiert werden.



- **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.



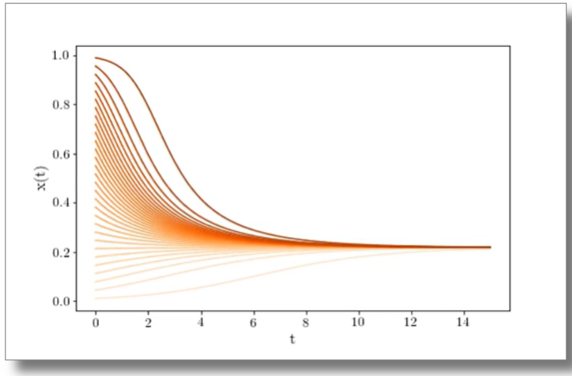
Internetseite der Vorlesung 4

Auf der Internetseite der Vorlesung 4, unter *Weiterführende Links*, sind ein Jupyter Notebook und drei Python Programme zum Herunterladen bereitgestellt, welche die nummerische Lösung von symmetrischen evolutionären 2-Personen 2-Strategien Spielen behandeln.

Vorlesung 4

Die Lösungen der im rechten Panel dargestellten Differentialgleichungen der *evolutionäre Spieltheorie* werden in dieser Vorlesung nummerisch gelöst und die möglichen zeitlichen Entwicklungen der Populationen in Klassen untergliedert. Die entsprechenden Python Programme (Python Skripts) und Jupyter Notebooks finden Sie weiter unten unter *Weiterführende Links*.

Evolutionäre symmetrische (2 × 2) Spiele



Die im unteren Bereich des rechten Panels dieser Vorlesung dargestellte Gleichung der Reproduktionsdynamik lässt sich nach Spezifikation der symmetrischen Auszahlungsmatrix $ auf unterschiedlichste Populationsspiele anwenden. Generell lassen sich symmetrische (2 × 2) Spiele und somit auch die Lösungen der Differentialgleichung der Reproduktionsdynamik für eine Populationsgruppe mit zwei Quasispezies in drei unterschiedliche Spielklassen gliedern: Dominante Spiele, Koordinationsspiele und Anti-Koordinationsspiele. Bei dominanten Spielen entwickelt sich, unabhängig von dem gewählten Anfangszustand, die Entscheidungswahl der Population stets zu einem Zustand, indem alle Spieler die dominante Strategie spielen - die dominante Strategie entspricht somit der evolutionär stabilen Strategie (ESS). Bei Koordinationsspielen entwickelt sich die Population, abhängig von ihrem Anfangszustand, entweder zu einem Zustand in dem alle Spieler die Strategie 2 (x=0) oder alle die Strategie 1 (x=1) wählen - man sagt es gibt zwei ESSs in Koordinationsspielen. Der kritische Anfangswert, der die beiden ESSs voneinander separiert, entspricht dem gemischten Nash-Gleichgewicht des zugrundeliegenden Koordinationsspiels und ist außerdem die interne Nullstelle der Funktion g(x). Bei Anti-Koordinationsspielen entwickelt sich die Population, unabhängig von ihrem Anfangs… asymptotisch erreichte Populatio… ist und die einzige evolutionär… Population im Falke-Taube Spiel … der negativen Auswirkung, wenn… V=[1,5]), so verschiebt man …

Vorlesung 4

In der vorigen Vorlesung wurden die zugrundeliegenden Differentialgleichungen der *evolutionäre Spieltheorie* für zeitlich sich wiederholende, unsymmetrische (2 Personen)-(m Strategien) Spiele dargestellt. Wir beschränken uns im Folgenden auf den 2-Strategien Fall ($m_A = m_B = 2$), lassen jedoch weiter eine Unsymmetrie der Auszahlungsmatrix zu (evolutionäre Bi-Matrix Spiele). 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:

$$\frac{dx(t)}{dt} = g_A(x,y) \quad , \quad \frac{dy(t)}{dt} = g_B(x,y)$$

$$g_A(x,y) := \left[\left(\$^A_{11} + \$^A_{22} - \$^A_{12} - \$^A_{21} \right) y(t) + \right.$$
$$\left. + \left(\$^A_{12} - \$^A_{22} \right) \right] \left(x(t) - (x(t))^2 \right)$$
$$g_B(x,y) := \left[\left(\$^B_{11} + \$^B_{22} - \$^B_{12} - \$^B_{21} \right) x(t) + \right.$$
$$\left. + \left(\$^B_{21} - \$^B_{22} \right) \right] \left(y(t) - (y(t))^2 \right)$$

In dieser Vorlesung betrachten wir jedoch zunächst die Lösungen der Differentialgleichung für sich wiederholende, symmetrische (2 Personen)-(2 Strategien) Spiele. Aufgrund der Symmetrie des Spiels besteht die betrachtete Population nun nur aus einer Gruppe von ununterscheidbaren Spielern (beschrieben durch den Populationsvektor $x(t)$). Die … ie vereinfacht sich in diesem … algleichung erster Ordnung:

$$\ldots)(1 - 2x + x^2)] \; x(t)$$

… en die Lösungen der

Weiterführende Links

- Folien der 4.Vorlesung
- Vorlesungsaufzeichnung der 4.Vorlesung: WS 2022/23 bzw. WS 2021/22
- View Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)
- View Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)

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 2025/26)

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

Frankfurt am Main 22.08.2025

Erster Vorlesungsteil:

Klassifizierung evolutionärer symmetrischer (2×2)-Spiele

Jupyter Notebook

Einführung

In diesem Unterkapitel werden die unterschiedlichen Spieltypen der gemischten Erweiterung eines simultanen (2 Spieler)-(2 Strategien) Spiels in strategischer Form mit symmetrischer Auszahlungsmatrix klassifiziert. Die Einteilung lehnt sich an das Buch von *Martin A. Nowak, Evolutionary Dynamics - Exploring the Equations of Life, 2006 (siehe Seite 49-51)* an. Eine alternative Klassifizierung findet man z.B. auch in *Matthias Hanauske, Evolutionäre Quanten-Spieltheorie im Kontext sozio-ökonomischer Systeme, 2011*. Ausgangspunkt ist die neben stehende allgemeine symmetrische Auszahlungsmatrix eines (2 Personen)-(2 Strategien) Spiels (a, b, c und d sind reellwertige Zahlen). Da es sich um eine symmetrische Auszahlungsmatrix handelt, gilt: $\hat{\$}^B = \left(\hat{\$}^A\right)^T$.

Abhängig von den gewählten Parametern der Auszahlungsmatrix lassen sich symmetrische (2×2) Spiele in drei unterschiedliche Spielklassen gliedern: Dominante Spiele, Koordinationsspiele und Anti-Koordinationsspiele. Wir nehmen die folgende Form der Spielmatrix an:

$$\hat{\$}^A = \hat{\$} = \begin{pmatrix} \$_{11} & \$_{12} \\ \$_{21} & \$_{22} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad (1)$$

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

Näheres zum numerischen Lösungen von Differenzialgleichungen finden Si
Einführung in die Programmierung für Studierende der Physik (Sommersem
Vorlesung 8 Differenzialgleichungen: Numerische Lösung von Anfangswert
Jupyter Notebook Differenzialgleichungen: Numerische Lösung von Anfang
(Bemerkung: In den vergangenen Semestern benutzten wir noch die Funkti
welche auf einem älteren Fortran-basierten Solver basiert.)

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from scipy.integrate import solve_ivp
```

Definition der Funktion g(x) und der Differenzialgleichung des evolutionäre

```
def g(x,a,b,c,d):
    g = ((a-c)*(x-x*x) + (b-d)*(1-2*x+x*x))*x
    return g
```

```
def DGL(t, x):
    dxdt = g(x,a,b,c,d)
    return dxdt
```

Numerisches Lösen der Differenzialgleichung

Beispiel eines dominanten Spiels (a=10 > c=7 und

Wir betrachten die zeitliche Entwicklung eines dominanten Spiels. x(t), der
zum Zeitpunkt t die Strategie 1 spielen, hängt neben der Funktion g(x) von
$x_0 := x(t=0)$ ab. Setzt man z.B. $x_0 = 0.2$ (das entspricht einer Anfangsp
spielt Strategie 1 und 80% spielt Strategie 2) so kann man die Differenzialg
lösen. Im Folgenden lösen wir das Anfangswertproblem im Bereich $t \in [0,$
von 1000 zeitlichen Gitterpunkten (mesh points).

```
a,b,c,d = 10,6,7,5
t_val = np.linspace(0, 5, 1000)
x0 = 0.2
Loes = solve_ivp(DGL, [0, 5], [x0], t_eval=t_val)
```

```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from scipy.integrate import solve_ivp

# 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

# Definition der DGL
def DGL(t, x):
    dxdt = g(x,a,b,c,d)
    return dxdt

# 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 symmetrischen (2x2)-Spiels
# Dominant Game
#a = -7
#b = -1
#c = -9
#d = -3
#tend=4
# Koordinantionsspiel
a = 2
b = 4
c = 0
d = 5
tend = 6
# Anti-Koordinantionsspiel
#a = -10
#b = 2
#c = 0
#d = 1
#tend = 3

#Weitere Festlegungen
```

Weiterführende Links

Auf der Internetseite der Vorlesung

- Folien der 4.Vorlesung
- Vorlesungsaufzeichnung der 4.Vorlesung: WS 2022/23 bzw. WS 2021/22
- View Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)
- View Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)

Python Programm Version 2 (evol1a.py)

Zum Lösen des evolutionären Spiels mit Python werden wir in dieser Version die Funktion 'solve_ivp(...)' verwenden, die sich im Untermodul 'scipy.integrate' befindet, welche Funktionen zum Lösen von gewöhnlichen Differenzialgleichungen bereitstellt.

Lösen des evolutionären Spiels mit Python (Version 2)

```python
#Weitere Festlegungen
numpoints = 500
t_val = np.linspace(0,tend,numpoints)
curvethick = 1.5

# Definition der Kurvenfarbe in Abhaengigkeit von der Spielklasse
# Dominant Games
if a > c and b > d:
    cmap = plt.cm.Greys
    texttitle = r'$\rm Dominantes\, Spiel $'
if a < c and b < d:
    cmap = plt.cm.Greys
    texttitle = r'$\rm Dominantes\, Spiel $'
# Koordinantionsspiele
if a > c and b < d:
    cmap = plt.cm.Blues
    texttitle = r'$\rm Koordinationsspiel $'
# Anti-Koordinantionsspiele
if a < c and b > d:
    cmap = plt.cm.Oranges
    texttitle = r'$\rm Anti-Koordinationsspiel $'

# Mehrere Anfangswerte der Population
num_x0 = 30
x0 = np.linspace(0,1,num_x0)
line_colors = cmap(np.linspace(0,1,num_x0+10))

# Loesung der DGL und plotten des Bildes
Loes = solve_ivp(DGL, [0, tend], x0, t_eval=t_val)
for j in range(num_x0):
    plt.plot(Loes.t,Loes.y[j],c=line_colors[j+5], linewidth=curvethick, linestyle='-')

# Plotten der Spielmatrix in das Bild
textstr1 = r'$\hat{\bf {\cal \$}} = \left( \begin{array}[c]{cc} a & b \\ \ c & d \end{array} \right)\\a='+str(a)+',\,b='+str(b)+'$'
textstr2 = r'$\\c='+str(c)+',\,d='+str(d)+'$'
props = dict(boxstyle='round', facecolor='white', alpha=0.92)
plt.text(tend-tend/50.0, 0.98, textstr1+textstr2, fontsize=16, verticalalignment='top', horizontalalignment='right', bbox=props)

# Achsenbeschriftungen usw.
plt.ylim(0,1)
plt.yticks([0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1],["$0$","","$0.2$","","$0.4$","","$0.6$","","$0.8$","","$1$"])
plt.ylabel(r"$\rm x(t)$")
plt.xlabel(r"$\rm t$")
plt.title(texttitle)

#Speichern der Bilder als (.png , benoetigt dvipng unter Linux)- und .pdf-Datei
saveFig="./evol.png"
plt.savefig(saveFig, dpi=100,bbox_inches="tight",pad_inches=0.05,format="png")
saveFig = "./evol.pdf"
plt.savefig(saveFig,bbox_inches="tight",pad_inches=0.05,format="pdf")
plt.show()
```

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

# 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

# Loesen der DGL
def solve_dgl(numpoints,tend,x0,a,b,c,d):
    sol = np.empty([numpoints,2])
    t = np.linspace(0,tend,numpoints)
    dt = t[1]-t[0]
    sol[0][0] = t[0]
    sol[0][1] = x0
    for i in range(1,numpoints):
        sol[i][0] = t[i]
        dx = g(sol[i-1,1],a,b,c,d)*dt
        sol[i][1] = sol[i-1,1] + dx
    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)

#Festlegung der Auszahlungsmatrix des symmetrischen (2x2)-Spiels
# Dominant Game
#a = -7
#b = -1
#c = -9
#d = -3
#tend=4
# Koordinantionsspiel
a = 2
b = 4
c = 0
d = 5
tend = 6
# Anti-Koordinantionsspiel
#a = -10
#b = 2
#c = 0
#d = 1
#tend = 3
```

Auf der Internetseite der Vorlesung

Weiterführende Links

- Folien der 4.Vorlesung
- Vorlesungsaufzeichnung der 4.Vorlesung: WS 2022/23 bzw. WS 2021/22
- View Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)
- View Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)

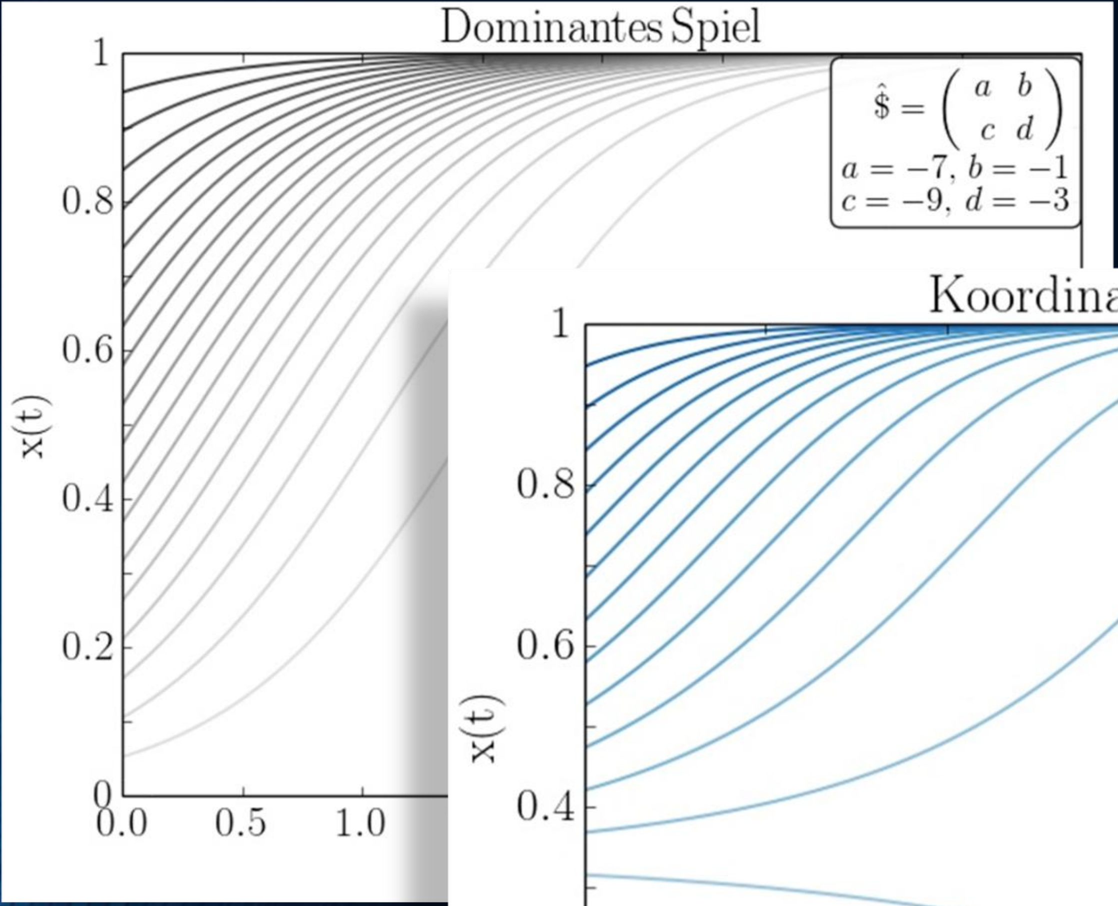
Python Programm Version 1 (evol1.py)

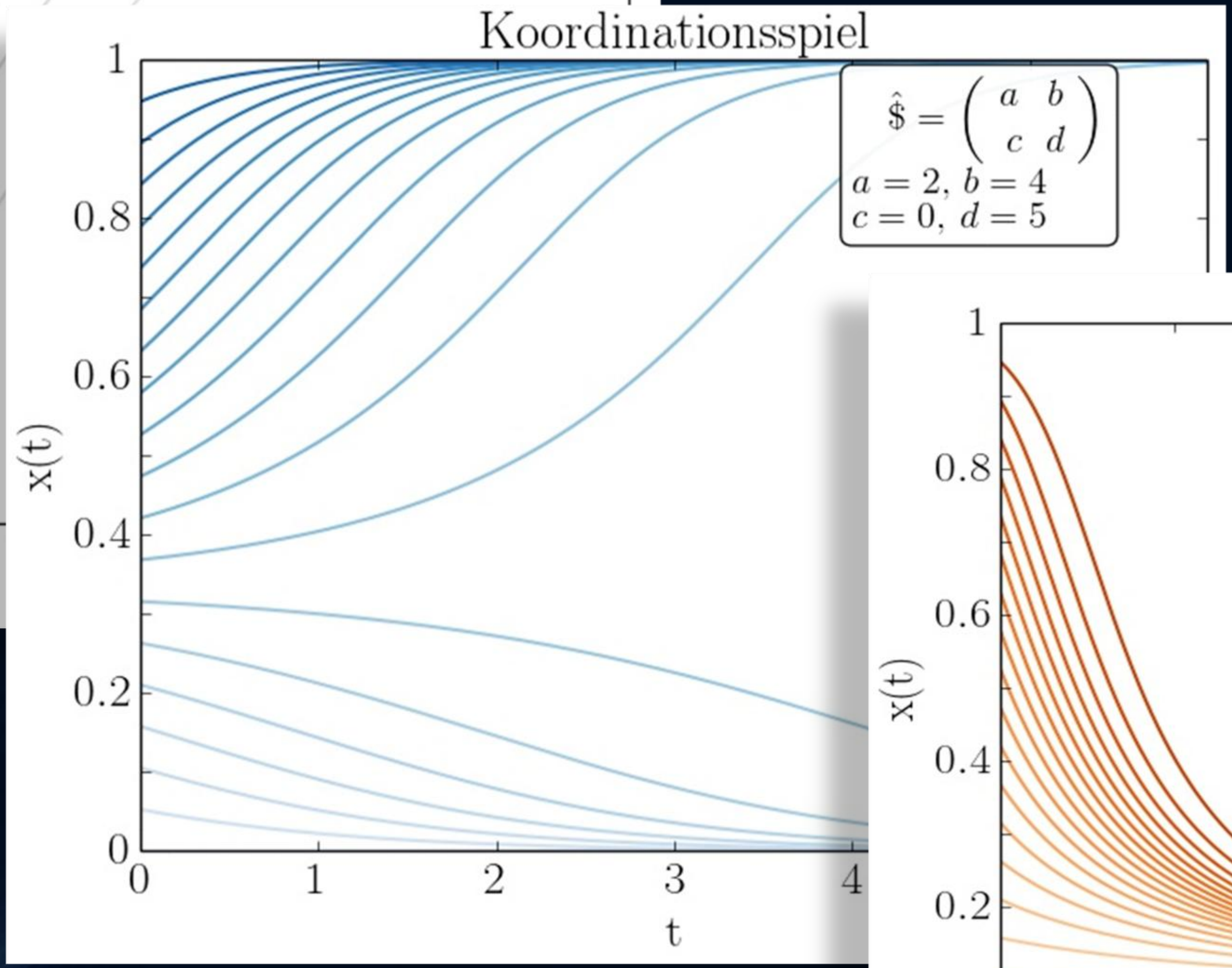
Lösen des evolutionären Spiels mit Python,
hier mit dem einfachen Euler-Verfahren

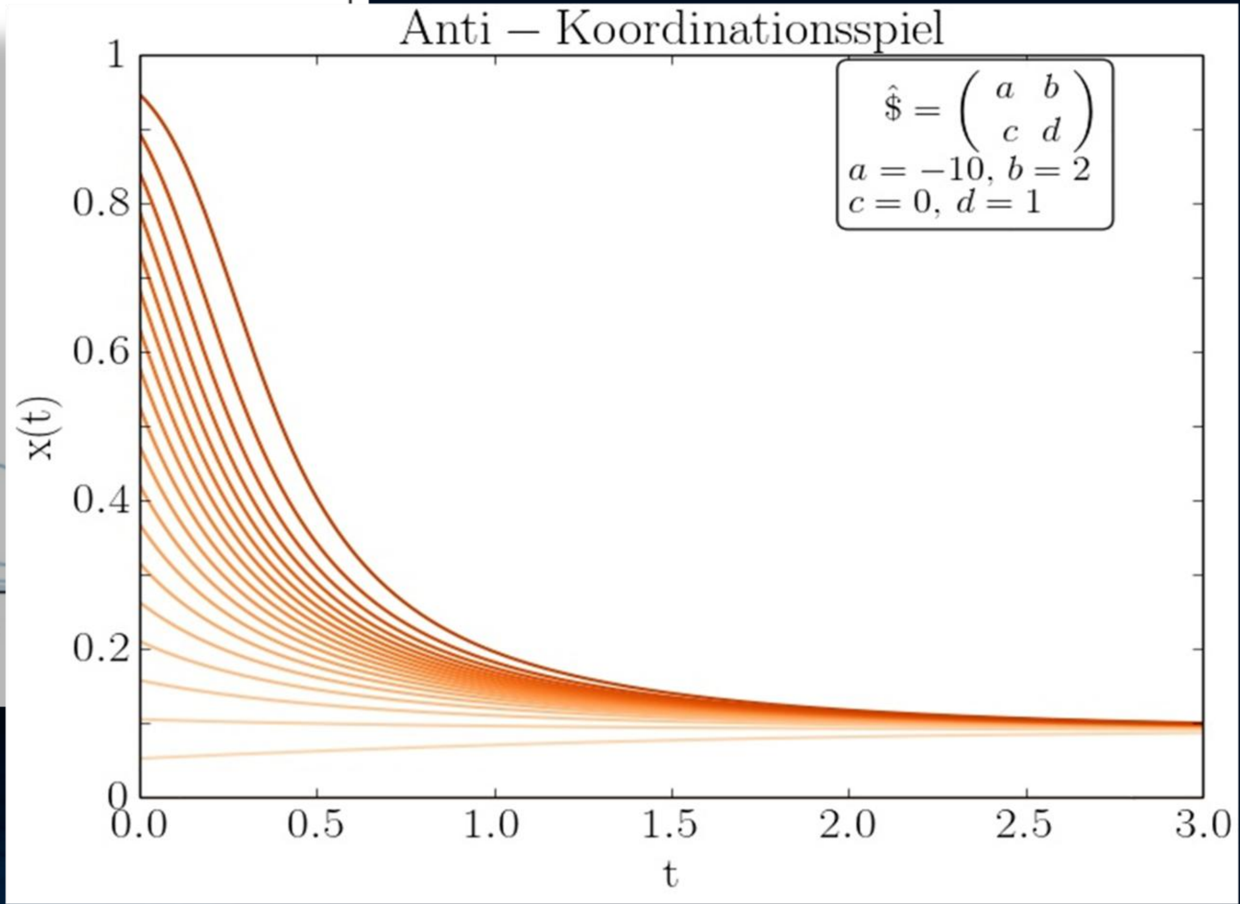
Lösen des evolutionären Spiels mit Python (Version 1)

```python
#Weitere Festlegungen
numpoints = 500
curvethick = 1.5

# Definition der Kurvenfarbe in Abhaengigkeit von der Spielklasse
# Dominant Games
if a > c and b > d:
    cmap = plt.cm.Greys
    texttitle = r'$\rm Dominantes\, Spiel $'
if a < c and b < d:
    cmap = plt.cm.Greys
    texttitle = r'$\rm Dominantes\, Spiel $'
# Koordinantionsspiele
if a > c and b < d:
    cmap = plt.cm.Blues
    texttitle = r'$\rm Koordinationsspiel $'
# Anti-Koordinantionsspiele
if a < c and b > d:
    cmap = plt.cm.Oranges
    texttitle = r'$\rm Anti-Koordinationsspiel $'

# Mehrere Anfangswerte der Population
num_x0 = 30
x0 = np.linspace(0,1,num_x0)
line_colors = cmap(np.linspace(0,1,num_x0+10))

# Loesung der DGL und plotten des Bildes
for j in range(num_x0):
    Loes = solve_dgl(numpoints,tend,x0[j],a,b,c,d)
    plt.plot(Loes[:,0],Loes[:,1],c=line_colors[j+5], linewidth=curvethick, linestyle='-')

# Plotten der Spielmatrix in das Bild
textstr1 = r'$\hat{\bf {\cal \$}} = \left( \begin{array}[c]{cc} a & b \\ \ c & d \end{array} \right)\\a='+str(a)+',\,b='+str(b)+'$'
textstr2 = r'$\\c='+str(c)+',\,d='+str(d)+'$'
props = dict(boxstyle='round', facecolor='white', alpha=0.92)
plt.text(tend-tend/50.0, 0.98, textstr1+textstr2, fontsize=16, verticalalignment='top', horizontalalignment='right', bbox=props)

# Achsenbeschriftungen usw.
plt.ylim(0,1)
plt.yticks([0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1],["$0$","","$0.2$","","$0.4$","","$0.6$","","$0.8$","","$1$"])
plt.ylabel(r"$\rm x(t)$")
plt.xlabel(r"$\rm t$")
plt.title(texttitle)


#Speichern der Bilder als (.png , benoetigt dvipng unter Linux)- und .pdf-Datei
saveFig="./evol.png"
plt.savefig(saveFig, dpi=100,bbox_inches="tight",pad_inches=0.05,format="png")
saveFig = "./evol.pdf"
plt.savefig(saveFig,bbox_inches="tight",pad_inches=0.05,format="pdf")
plt.show()
```

Lösen des evolutionären Spiels mit Python Versionen 1 oder 2







```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams
from scipy.integrate import solve_ivp

# DGL bestimmende Funktion g
def g(x, a, b, c, d):
    return ((a-c)*(x-x*x) + (b-d)*(1-2*x+x*x))*x

# Definition der DGL
def DGL(t, x, a, b, c, d):
    return g(x, a, b, c, d)

# Klassifizierung des Spiels, Festlegung der Farbe und des Titels
def classify_game(a, b, c, d):
    if (a > c and b > d) or (a < c and b < d):
        return plt.cm.Greys, r'$\rm Dominantes\, Spiel $'
    elif a > c and b < d:
        return plt.cm.Blues, r'$\rm Koordinationsspiel $'
    elif a < c and b > d:
        return plt.cm.Oranges, r'$\rm Anti-Koordinationsspiel $'
    else:
        return plt.cm.Greens, r'$\rm ?\, Spielklasse $'
```

```python
# Loesen der DGL und plotten der Populationsentwicklung
def solve_and_plot(a, b, c, d, t_end=6, num_x0=30, n_points=500):
    # Groessenfestlegung der Labels usw.
    rcParams.update({
        'text.usetex': True,
        'axes.titlesize': 22,
        'axes.labelsize': 20,
        'xtick.labelsize': 20,
        'ytick.labelsize': 20
    })

    # Mehrere Anfangswerte der Population
    x0 = np.linspace(0.01, 0.99, num_x0)
    t_eval = np.linspace(0, t_end, n_points)

    # Loesung der DGL
    Loes = solve_ivp(DGL, [0, t_end], x0, args=(a,b,c,d, ), t_eval=t_eval)

    # Festlegung der Farbe und des Titels
    cmap, title_text = classify_game(a, b, c, d)
    line_colors = cmap(np.linspace(0, 1, num_x0))
    line_width = 1.5

    # Plotten der Losungen
```

```python
# Berechnung der Nashgleichgewichte
def find_nash_equilibria(a, b, c, d):
    equilibria = []
    # Auszahlungsmatrix (symmetrisches 2x2-Spiel)
    D_A = np.array([[a,b],[c,d]])
    # Berechnung der reinen Nash-Gleichgewichte
    for i in range(2):
        for j in range(2):
            if D_A[i, j] == np.max(D_A[:, j]) and D_A[j, i] == np.max(D_A[:, i]):
                equilibria.append({'type': 'pure', 's': (i+1, j+1)})
    # Berechnung der gemischten Nash-Gleichgewichte (mittels sympy)
    Nenner = a - b - c + d
    if abs(Nenner) > 1e-10:
        s = (d - b) / Nenner
        if 0 < s < 1:
            equilibria.append({'type': 'mixed', 's_star': s})
    return equilibria
```

Weiterführende Links

Auf der Internetseite der Vorlesung

Folien der 4.Vorlesung

Vorlesungsaufzeichnung der 4.Vorlesung: WS 2022/23 bzw. WS 2021/22

View Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele

Download Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele

Download Python Programm: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)

View Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele

Download Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele

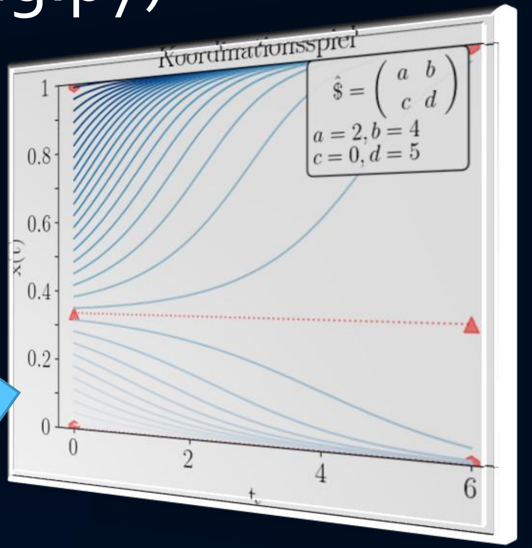
Download Python Programm: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)

Python Programm Version 3 (evol1_ng.py)

Modularisierung der Version 2 mittels unterschiedlicher Funktionen

Berechnung der Nash-Gleichgewichte

Automatische Kennzeichnung der Nash-Gleichgewichte im Bild



E-Learning und interaktive Übungsaufgaben

Zusäzlich zu den Informationen aus dieser Internetseite finden Sie in diesem Unterpunkt diverse interaktive Übungsaufgaben zu den folgenden Themen:

Aufgabe 1

Reine Nash-Gleichgewichte in einem simultanen (2x2)-Spiel in strategischer Form mit symmetrischer Auszahlungmatrix

Aufgabe 2

Gemischtes Nash-Gleichgewicht in einem simultanen (2x2)-Spiel in strategischer Form mit symmetrischer Auszahlungmatrix

Aufgabe 3

Das gemischtes Nash-Gleichgewicht im Hirschjagt-Spiel

Aufgabe 4

Spielklassen von simultanen (2x2)-Spielen in strategischer Form mit symmetrischer Auszahlungmatrix

Aufgabe 5

Zeitliche Entwicklung des Populationsvektors im evolutionären Spiel

Aufgabe 6

Evolutionär stabile Strategien

Aufgabe 7

Zeitliche Entwicklung des Populationsvektors im evolutionären Bi-Matrix Spiel

Aufgabe 8

Gemischtes Nash-Gleichgewicht in einem simultanen (2x2)-Spiel in strategischer Form mit unsymmetrischer Auszahlungmatrix

Aufgabe 9

Gemischtes Nash-Gleichgewicht und zeitliche Entwicklung des Populationsvektors in Zentrumsspielen

Aufgabe 10

Zeitliche Entwicklung des Populationsvektors im evolutionären (2x3)-Spiel

Aufgabe 11

Gemischtes Nash-Gleichgewicht im evolutionären (2x3)-Spiel

Aufgabe 12

Mittlere Distanz zwischen zwei Knoten in einem zufälligen Netzwerk

Aufgabe 4

A/B	s_1	s_2
s_1	(107 , 107)	(10 , 177)
s_2	(177 , 10)	(125 , 125)

Betrachten Sie ein simultanes (2 Spieler)-(2 Strategien) Spiel in strategischer Form mit symmetrischer Auszahlungmatrix. Die Menge der Spieler sei $\mathcal{I} = \{A, B\}$, die Menge der reinen Strategien sei $\mathcal{S}^A = \mathcal{S}^B = \{s_1, s_2\}$ und die Präferenzordnungen der Spieler sei durch die neben stehende Auszahlungstabelle quantifiziert.
Welcher Spielklasse gehört das Spiel an? Markieren Sie die zutreffende Aussage in der unteren Auswahl an

- ○ Dominantes Spiel
- ○ Koordinationsspiel
- ○ Anti-Koordinationsspiel

und vergleichen Sie indem Sie den folgenden *Button* drücken.

Lösung anzeigen

Das Spiel gehört zur Klasse der

Aufgabe 5

Lösung

Der Wert des Populationsvektors zur Zeit t=1.06 ist $x(t = 1.06)$ =

Weiter zur nächsten Aufgabe ...

Betrachten Sie die zeitliche Entwicklung eines evolutionären (2x2)-Spiels mit symmetrischer Auszahlungsmatrix. Der Populationsvektors $\vec{x}(t)$ (hier $x(t)$, da nur zwei Strategien) wird durch folgende Differentialgleichung bestimmt

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

$x(t)$, der Anteil der Spieler die zum Zeitpunkt t die Strategie s_1 spielen, hängt neben der Funktion g(x) von dem Anfangswert $x(t=0)$ ab. Die Auszahlungswerte des Spiels seien $\$_{11} = 1$, $\$_{12} = 4$, $\$_{21} = 0$ und $\$_{22} = 2$ und der Anfangswert der Population zum Zeitpunkt t=0 sei $x(0) = 0.1$. Berechnen Sie den Wert des Populationsvektors zur Zeit t=1.06. Tragen Sie bitte Ihren Wert in das untere Eingabenfeld ein

$x(t = 1.06)$ = 0,000

und vergleichen Sie indem Sie den folgenden *Button* drücken.

Lösung anzeigen

Aufgabe 6

Lösung

Die evolutionär stabile Strategie des Spiels lautet $x(t \to \infty)$ =

Weiter zur nächsten Aufgabe ...

Betrachten Sie die zeitliche Entwicklung eines evolutionären (2x2)-Spiels mit symmetrischer Auszahlungsmatrix. Der Populationsvektors $\vec{x}(t)$ (hier $x(t)$, da nur zwei Strategien) wird durch folgende Differentialgleichung bestimmt

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

$x(t)$, der Anteil der Spieler die zum Zeitpunkt t die Strategie s_1 spielen, hängt neben der Funktion g(x) von dem Anfangswert $x(t=0)$ ab. Die Auszahlungswerte des Spiels seien $\$_{11}$ = 139 , $\$_{12}$ = 110 , $\$_{21}$ = 135 und $\$_{22}$ = 145 und der Anfangswert der Population zum Zeitpunkt t=0 sei $x(0)$ = 0.5. Berechnen Sie die evolutionär stabile Strategie des Spiels; also gegen welchen Wert strebt der Populationsvektor fÃ¼r $t \to \infty$?

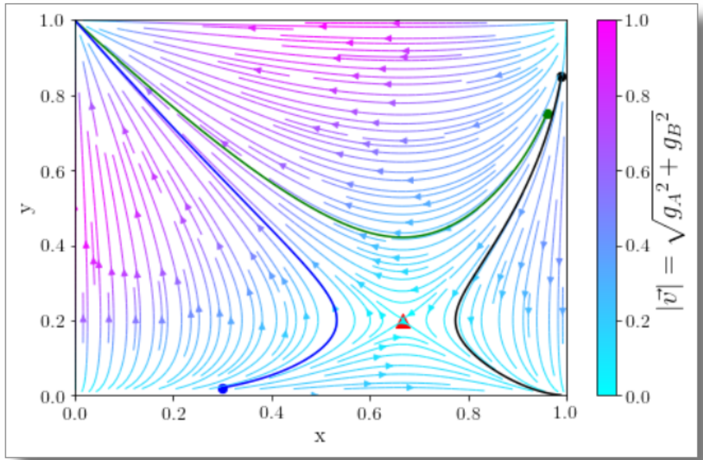
Tragen Sie bitte Ihren Wert in das untere Eingabenfeld ein

$x(t \to \infty)$ = 0,000

und vergleichen Sie indem Sie den folgenden *Button* drücken.

Lösung anzeigen

Evolutionäre unsymmetrische (2×2) Spiele (Bi-Matrix Spiele)



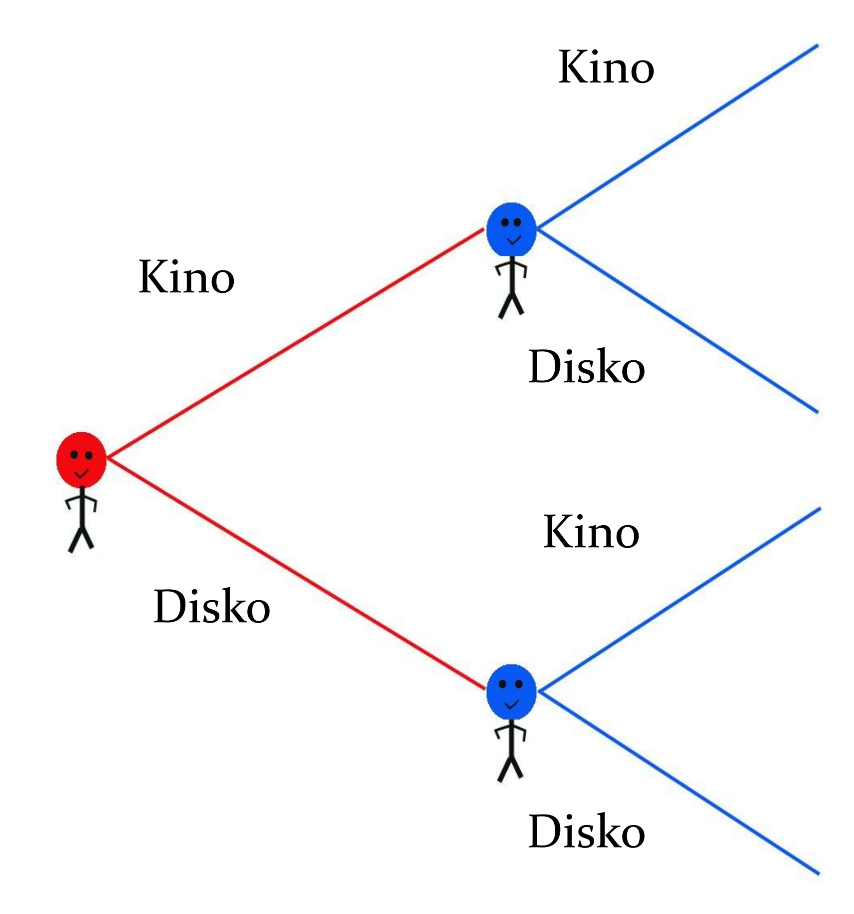
Die im oberen Bereich des rechten Panels dieser Vorlesung dargestellten Gleichungen der Reproduktionsdynamik lassen sich nach Spezifikation der Auszahlungsmatrizen $\hat{\$}^A$ und $\hat{\$}^B$ auf unterschiedlichste Populationsspiele anwenden. Generell lassen sich evolutionäre unsymmetrische $(\mathbf{2} \times \mathbf{2})$ Spiele in die folgenden drei Spielklassen gliedern: Eckenspiele, Sattelpunktspiele und Zentrumsspiele. Ein Eckenspiel liegt vor, falls zumindest eine der Auszahlungsmatrizen der Spielergruppen eine dominante Struktur hat. Aufgrund der Dominanz der Strategie endet die zeitliche Entwicklung der Populationen, unabhängig von der Anfangsbedingungen, in einer Ecke des x-y Populationsvektor Diagramms - die ESS des Eckenspiels. Ein Sattelpunktspiel liegt vor, falls beide Spielergruppen gleichzeitig ein Koordinationsspiel oder Anti-Koordinationsspiel spielen. Bei einem Sattelpunktsspiel das aus zwei Anti-Koordinationsspielen besteht, existieren zwei ESSs ((x=0,y=1) oder (x=1,y=0)) zu denen sich die Populationsgruppen im Laufe der Zeit entwickeln. Welche dieser evolutionär stabilen Strategien erreicht wird, hängt von der Anfangsstrategiewahl der Population ab. Die nebenstehende Abbildung zeigt die zeitliche Entwicklung von drei Anfangszuständen in einem solchen Eckenspiel. Auch bei sehr ähnlichen Werten der Anfangsstrategiewahl kann es geschehen, dass sich die Population im Laufe der Zeit zu unterschiedlichen Ecken entwickelt (siehe grüne und schwarze Trajektorien). Eine besondere Bedeutung hat der Sattelpunkt des Spiels (siehe rotes Dreieck). Die Position des Sattelpunktes im x-y Diagramm entspricht dem Wert des gemischten Nash-Gleichgewichtes bzw. lässt sich durch die Nullstellen der Funktionen $g_A(x,y)$ und $g_B(x,y)$ bestimmen. Bei einem Zentrumsspielen existiert keine ESS, da die Strategiewahl der Population sich im Laufe der Zeit ständig verändert und um ein Zentrum kreist. Die Position dieses Zentrums im x-y Diagramm entspricht dem Wert des gemischten Nash-Gleichgewichtes bzw. lässt sich durch die Nullstellen der Funktionen $g_A(x,y)$ und $g_B(x,y)$ bestimmen (siehe Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele).

Weiterführende Links

- Folien der 4.Vorlesung
- Vorlesungsaufzeichnung der 4.Vorlesung: WS 2022/23 bzw. WS 2021/22
- View Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)
- View Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)

Kampf der Geschlechter

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



Alexander und Bettina haben bei ihrem letzten Treffen nicht genau ausgemacht, wann und wo sie sich am Samstagabend treffen wollen. Bettina geht sehr gerne in das kleine Kino am Stadtrand (Spätvorstellung, Beginn 23.30 Uhr), Alexander aber würde gerne in die Diskothek im Zentrum der Stadt gehen. Keiner von ihnen hat ein Telefon. Bettina denkt, wird er mir zuliebe ins Kino gehen? Alexander denkt ähnlich, wird sie mir zuliebe in die Diskothek gehen? Beiden liegt aber viel daran sich am Samstagabend zu treffen. Der erzielte Nutzen dieses Spiels kann zum Beispiel wie oben angegeben quantifiziert werden.

Kampf der Geschlechter

(Nash-Gleichgewichte in reinen Strategien)

- Es gibt keine dominante Strategie bei diesem Spiel.
- Es gibt zwei reine Nash-Gleichgewichte:
  (Kino,Kino)
  (Disko, Disko)

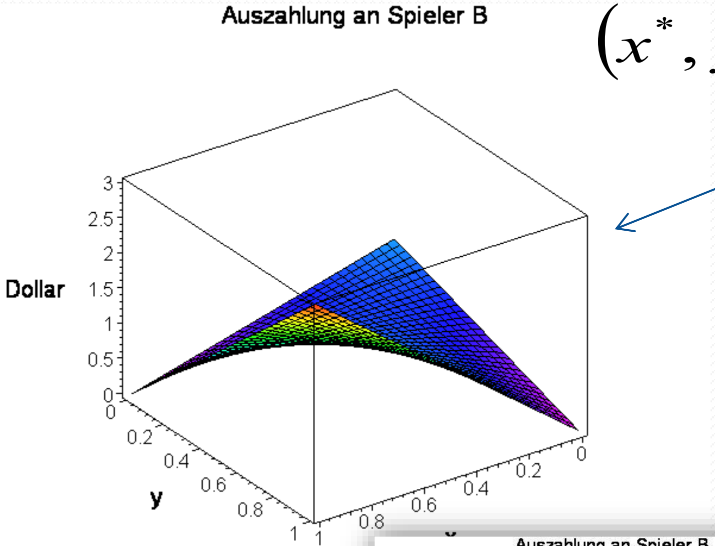
	Kino	Disko
Kino	(1 , 3)	(0 , 0)
Disko	(0 , 0)	(3 , 1)

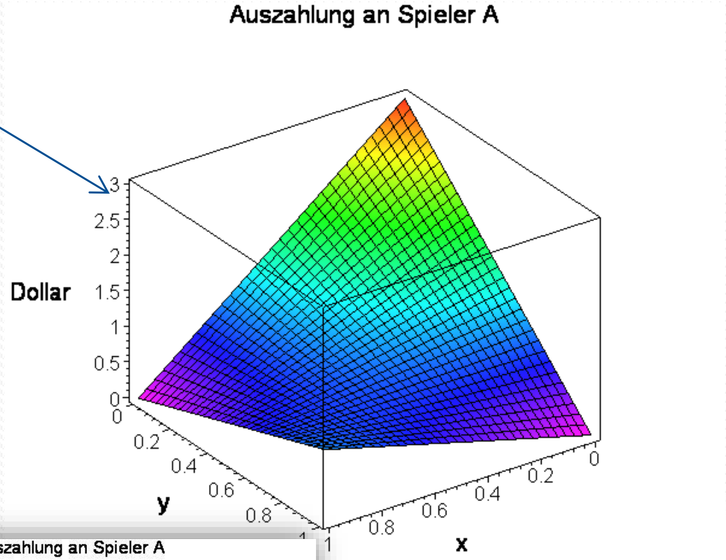
Kampf der Geschlechter

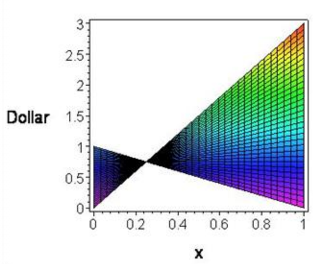
(Grafische Veranschaulichung des gemischten Nash-Gleichgewichts)

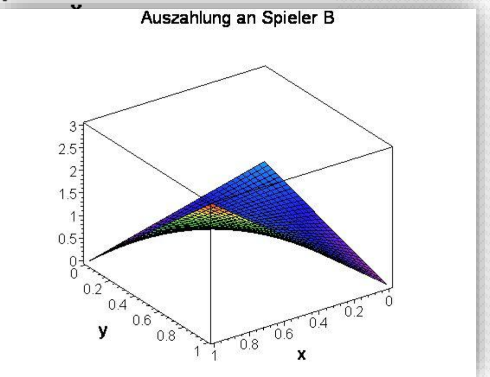
- Das Nash-Gleichgewicht in gemischten Strategien befindet sich bei

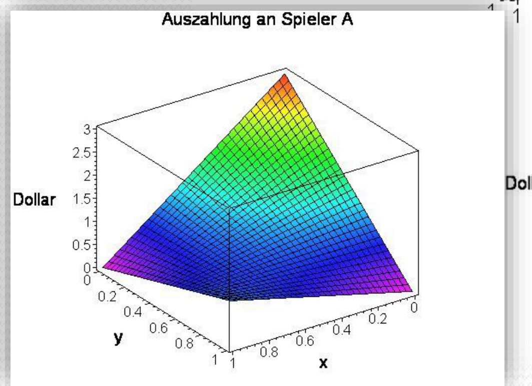
$$\left(x^*, y^*\right) = \left(\frac{1}{4}, \frac{3}{4}\right)$$

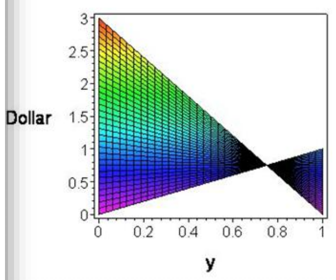








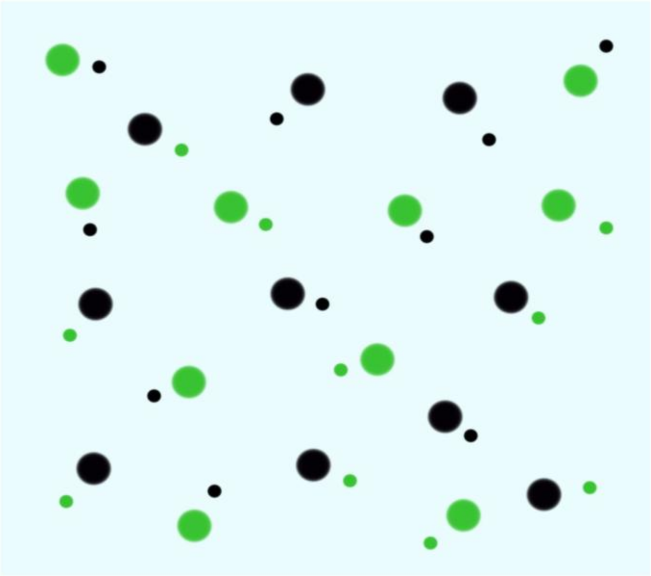




Evolutionäre Spieltheorie

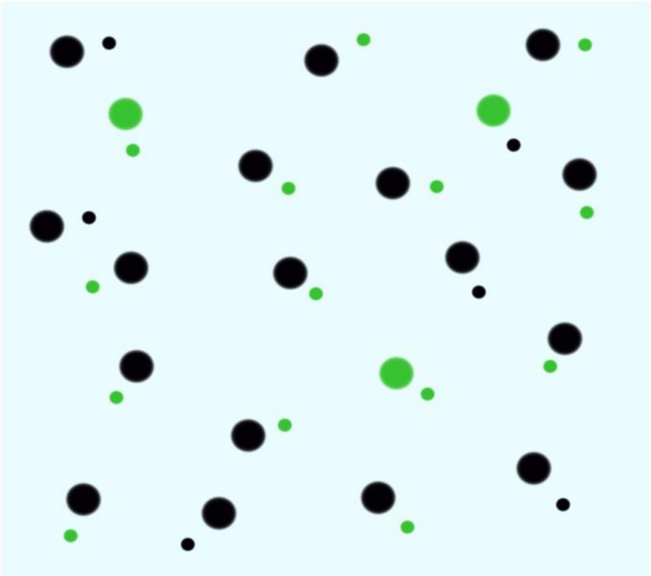
Unsymmetrische (2x2)-Spiele (Bimatrixspiele) zweier Populationen

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.



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

zeitliche Entwicklung der Population



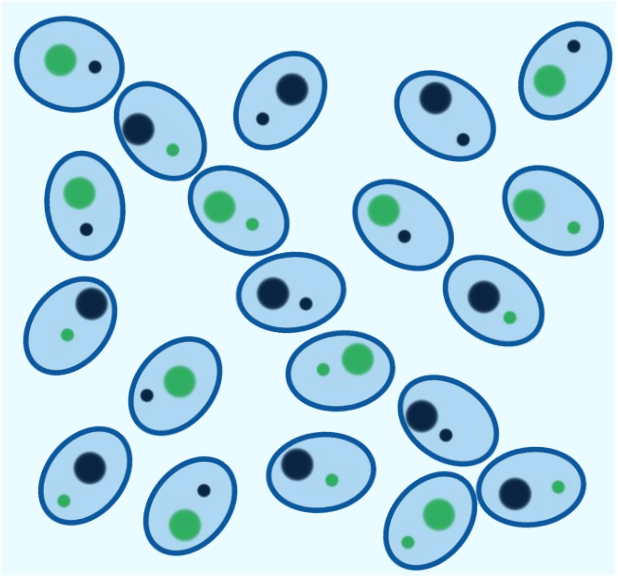
$x(10)=0.15$, $y(10)=0.7$

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

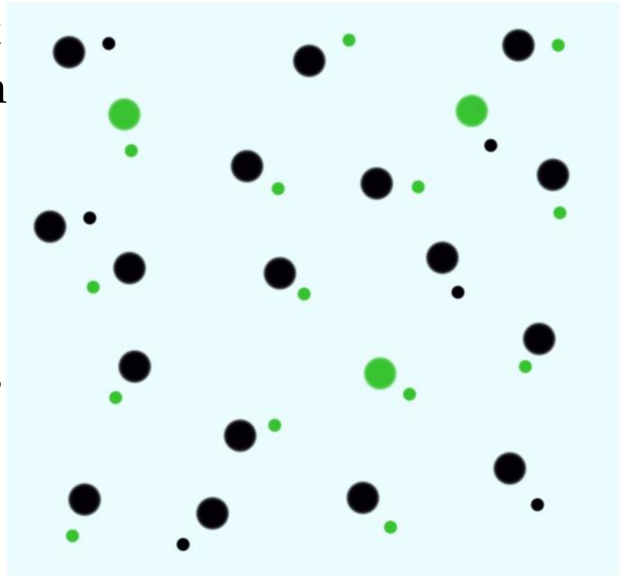
Unsymmetrische (2x2)-Spiele (Bimatrixspiele)

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):

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

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

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:

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

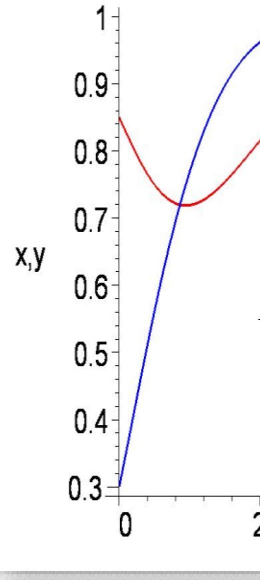
$$\frac{dy(t)}{dt} = y(t)\cdot\left(4\cdot x(t) - 4\cdot x(t)\cdot y(t) + y(t) - 1\right)$$

Die 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)).

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

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

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.



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

Bi-Matrix Spiele

Gemischte Nash-Gleichgewichte

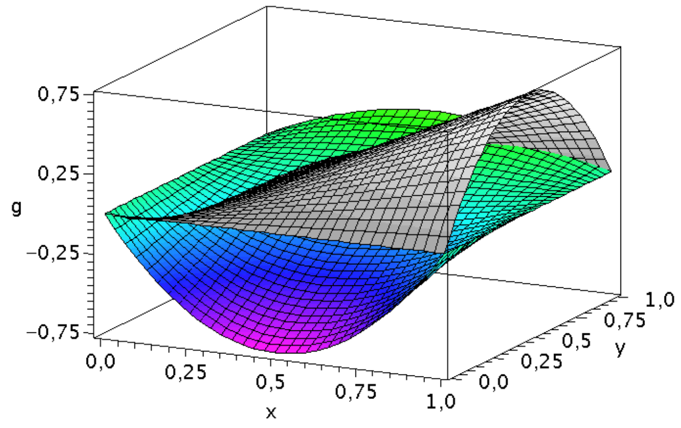
Nash-Gleichgewicht in gemischten Strategien:

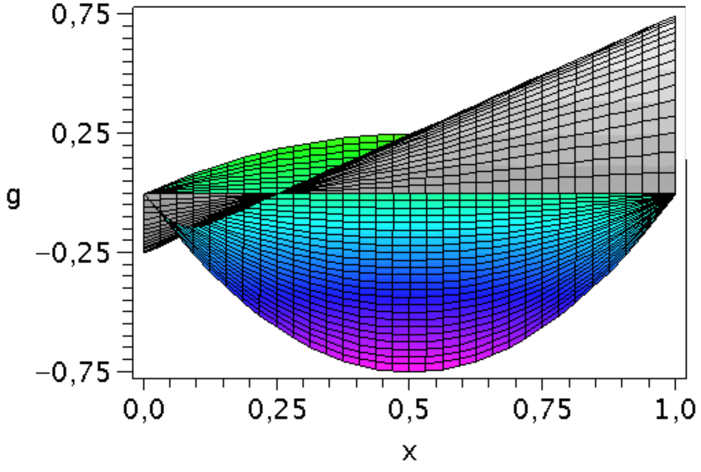
$$\left. \frac{\partial \tilde{\$}^{A}(\tilde{s}^{A}, \tilde{s}^{B})}{\partial \tilde{s}^{A}} \right|_{\tilde{s}^{B} = \tilde{s}^{B\star}} = 0 \quad \forall \;\; \tilde{s}^{A} \in [0,1] \;\;,\;\; \tilde{s}^{B\star} \in]0,1[$$

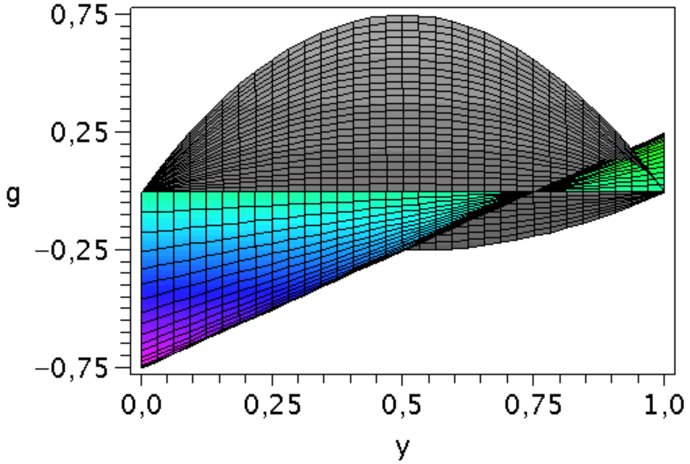
$$\left. \frac{\partial \tilde{\$}^{B}(\tilde{s}^{A}, \tilde{s}^{B})}{\partial \tilde{s}^{B}} \right|_{\tilde{s}^{A} = \tilde{s}^{A\star}} = 0 \quad \forall \;\; \tilde{s}^{B} \in [0,1] \;\;,\;\; \tilde{s}^{A\star} \in]0,1[$$

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:

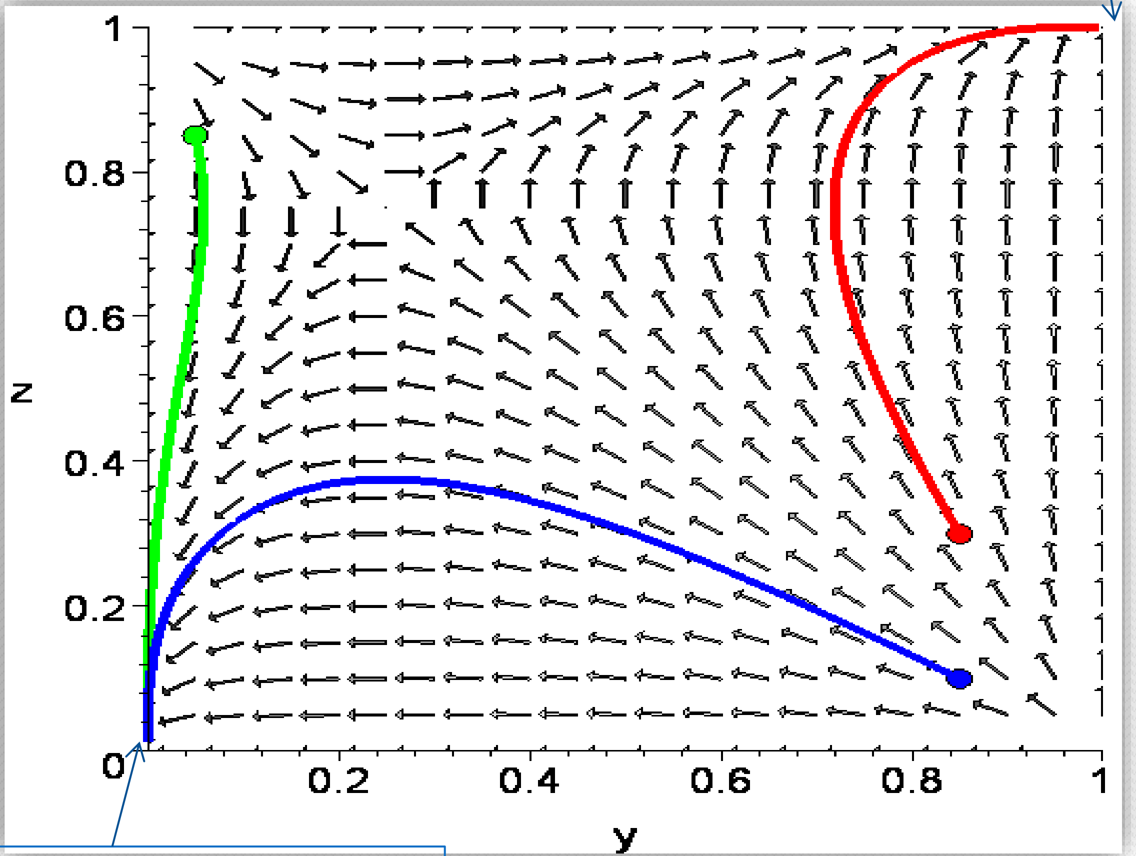
Kampf der Geschlechter als evolutionäres Spiel

	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:

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.

(1,1) entspricht (Kino,Kino)



(0,0) entspricht (Disko,Disko)

Beispiel 2:

Klasse der Zentrumsspiele (Center Class)

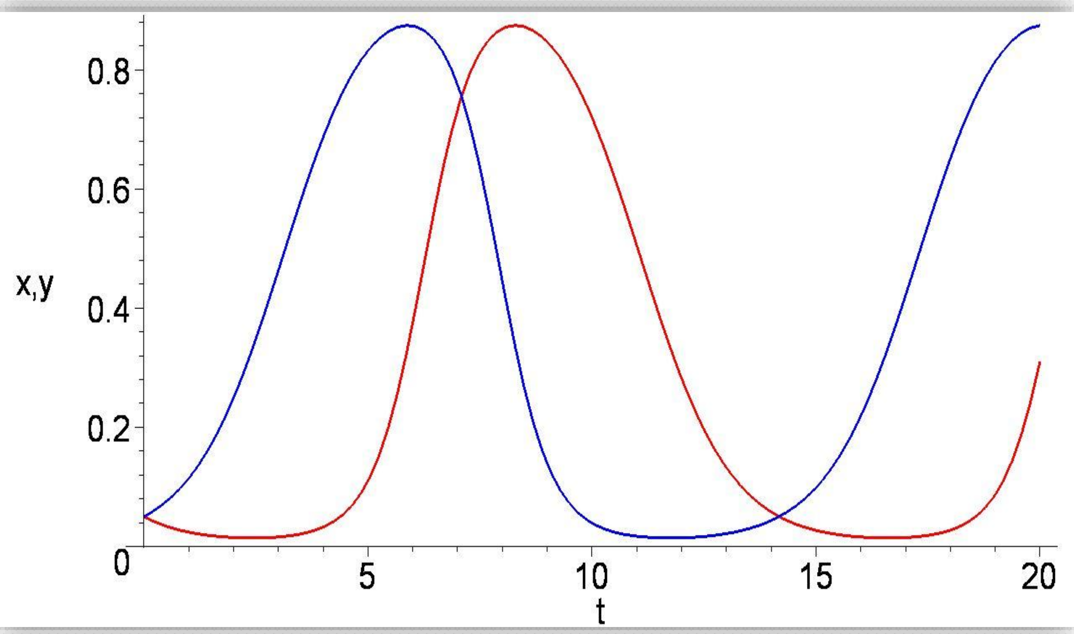
	Strat. 1	Strat. 2
Strat. 1	(2 , -2)	(0 , 0)
Strat. 2	(0 , 0)	(1 , -1)

Das gekoppelte System von Differentialgleichung für dieses Beispiel lautet:

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

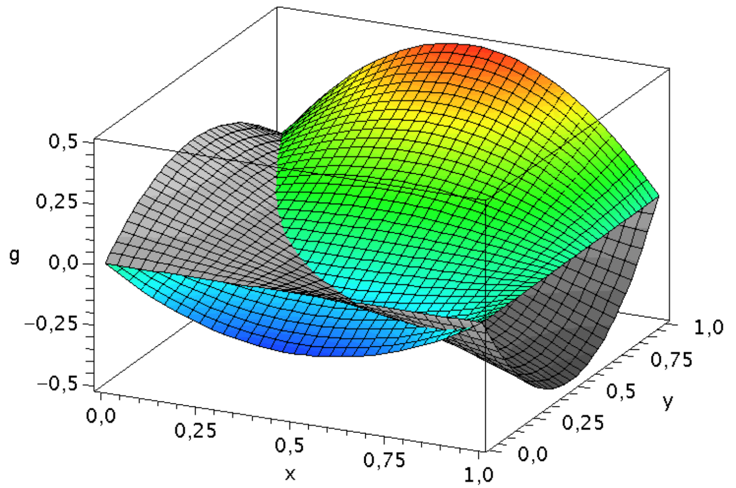
Die Lösung der Gleichung erfolgt wiederum durch Integration bzw. kann mithilfe des Computers nummerisch berechnet werden. Es wurde der Anfangswert zu (x(0),y(0)) =(0.05 , 0.05) gewählt.

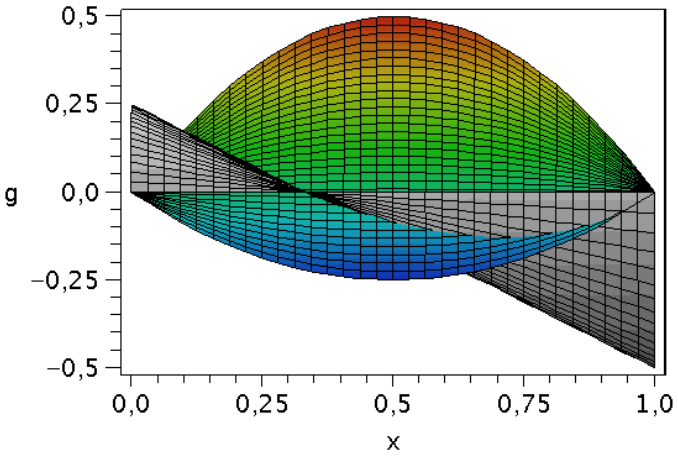
Die rechte Abbildung zeigt eine nummerische Lösung der obigen Gleichung, wobei der Anfangswert zu (x(0),y(0)) =(0.05 , 0.05) gewählt wurde. Im Gegensatz zu allen anderen möglichen Klassen von Bimatrixspielen, treten bei der Klasse der Zentrumsspiele periodische Verläufe der Populationsanteile x(t) und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

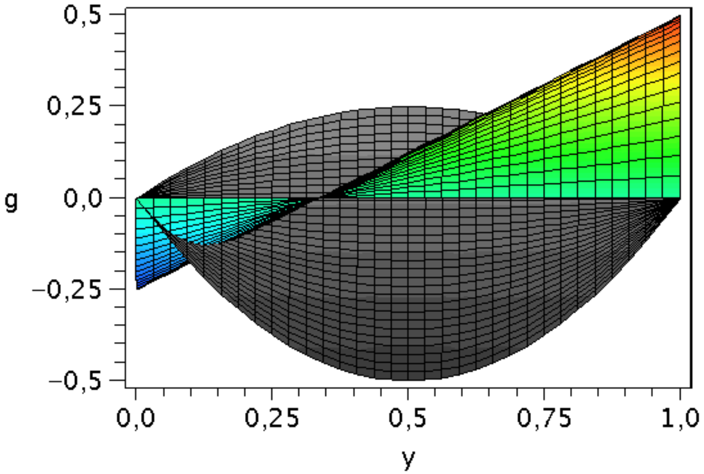


Eigenschaften der Funktionen $g_A(x,y)$ und $g_B(x,y)$

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.







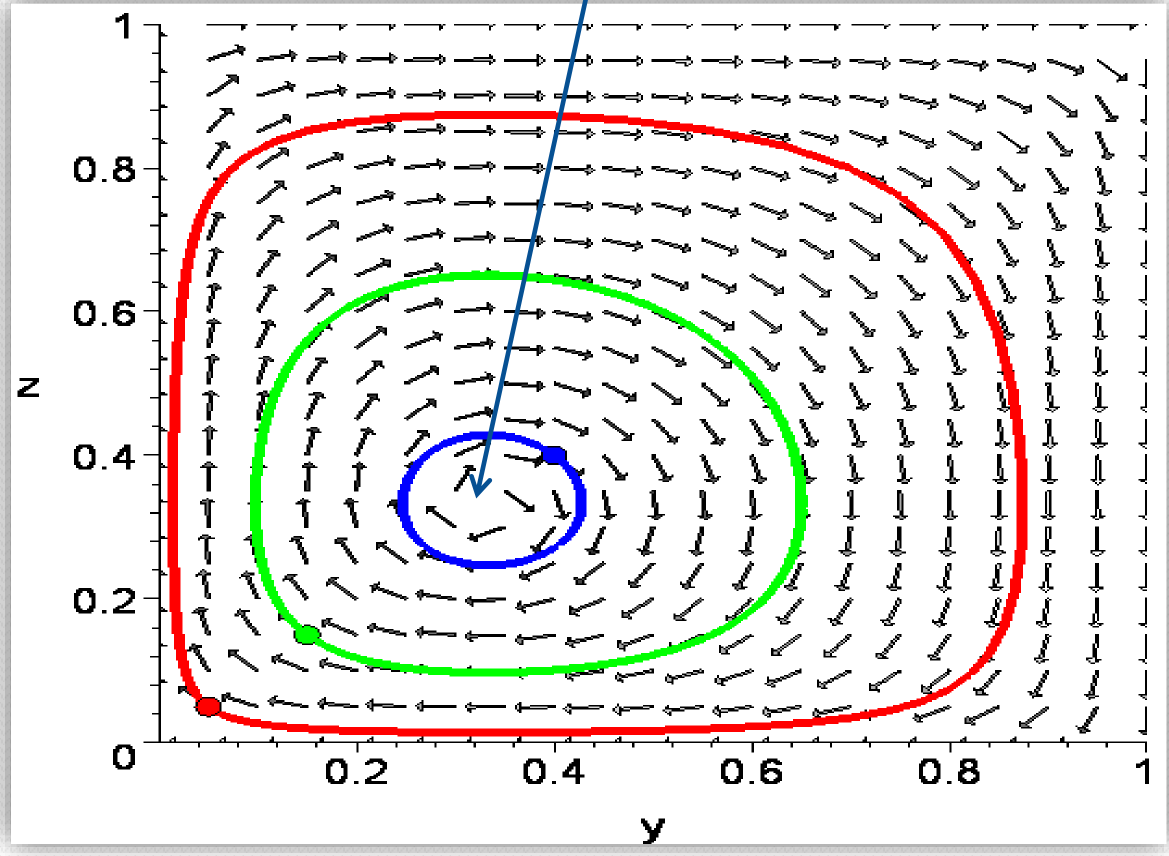
Beispiel 2:

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)

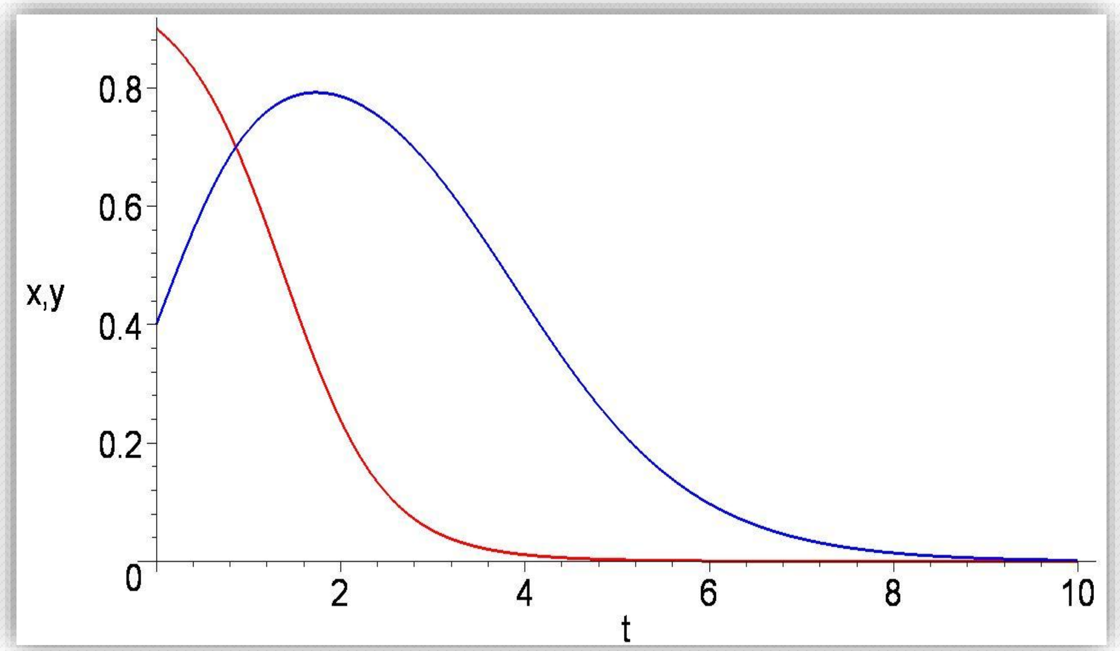
	Strat. 1	Strat. 2
Strat. 1	(-2 , 2)	(0 , 0)
Strat. 2	(0 , 0)	(1 , 1)

Das gekoppelte System von Differentialgleichung für dieses Beispiel lautet:

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

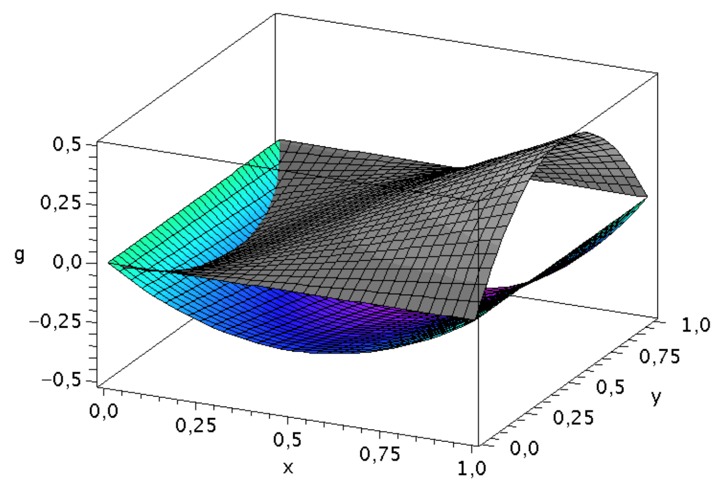
Die Lösung der Gleichung erfolgt wiederum durch Integration bzw. kann mithilfe des Computers nummerisch berechnet werden. Es wurde der Anfangswert zu
(x(0),y(0)) =(0.9 , 0.4) gewählt.

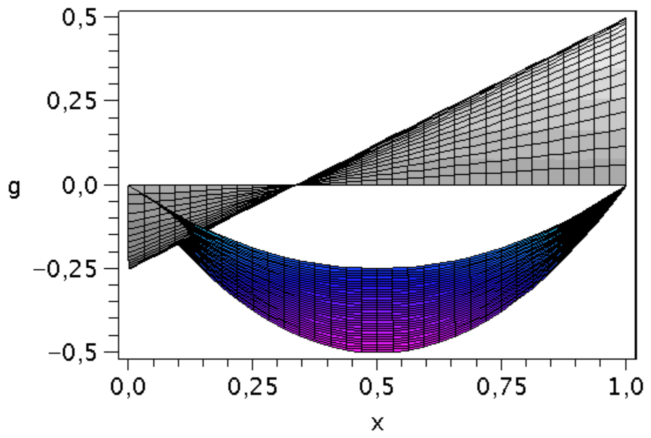
Die rechte Abbildung zeigt eine nummerische Lösung der obigen Gleichung, wobei der Anfangswert zu (x(0),y(0)) =(0.9 , 0.4) gewählt wurde. Bei der Klasse der „Eckspiele“ gibt es eine evolutionär stabile Strategie, die unabhängig von der gewählten Anfangsbedingung stets von der Population angestrebt wird.

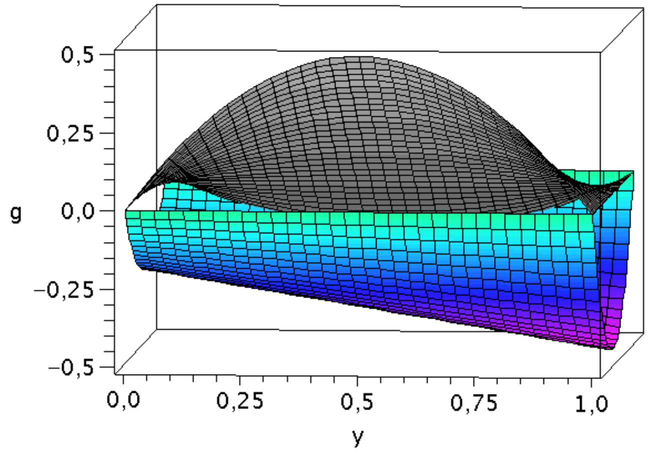


Eigenschaften der Funktionen $g_A(x,y)$ und $g_B(x,y)$

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:

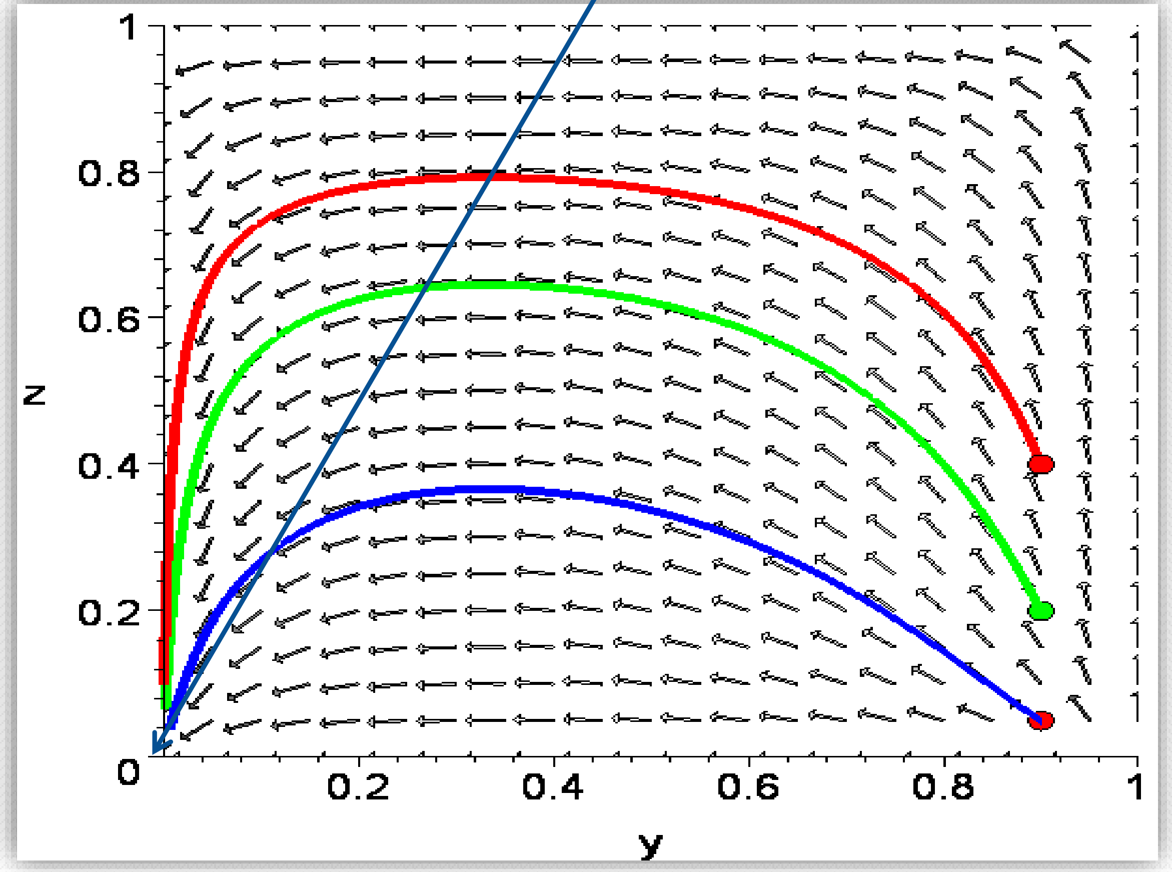
Klasse der Eckspiele (Corner Class)

	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

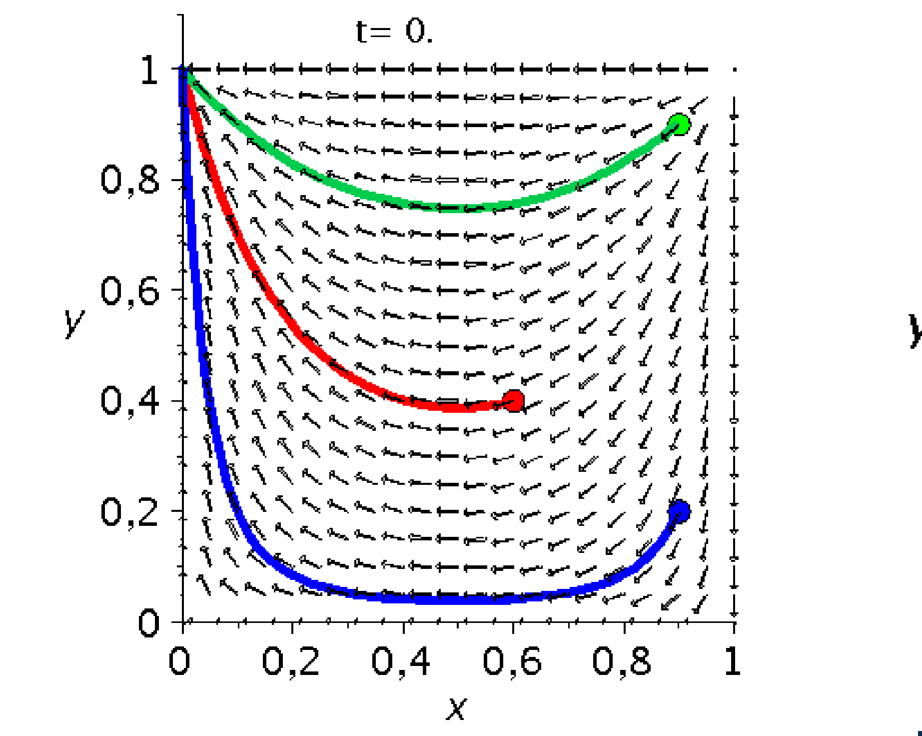
Die Spielklasse der Gruppe A oder der Gruppe B ist ein *Dominantes Spiel*

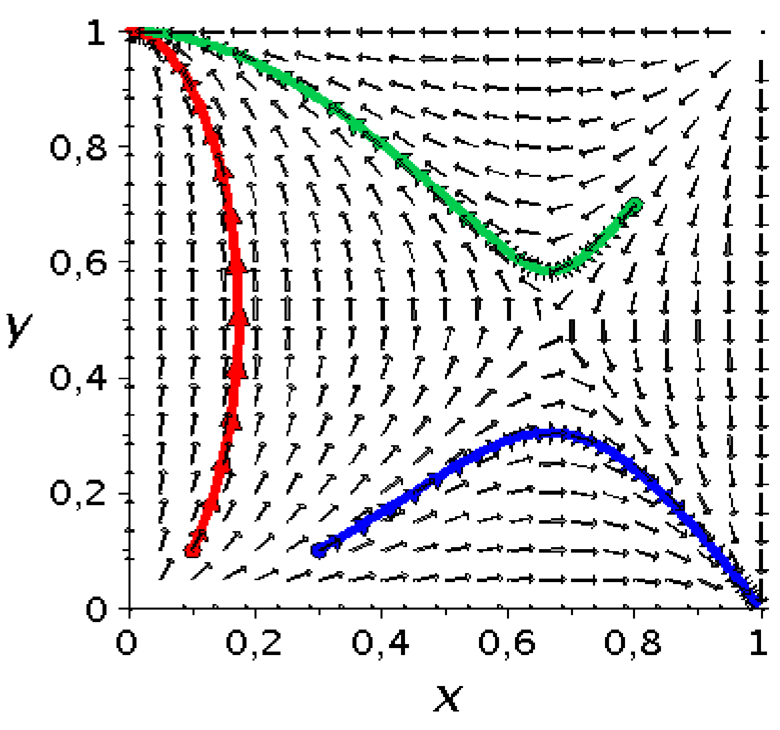
Sattelspiele

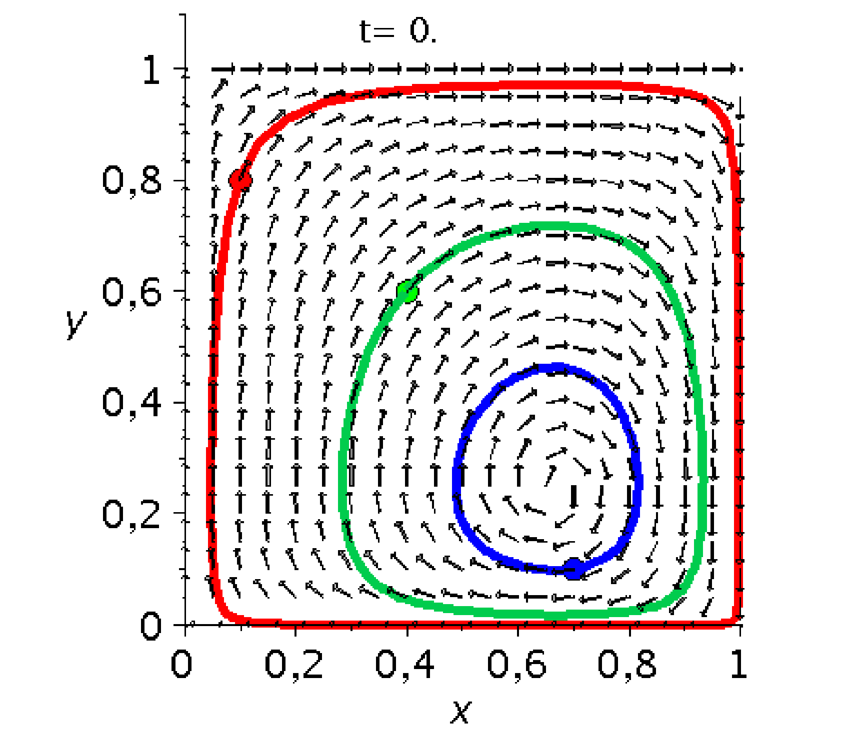
Spiel A: *Koordinationspiel*
Spiel B: *Koordinationspiel*
oder
Spiel A: *Anti-Koordinationspiel*
Spiel B: *Anti-Koordinationspiel*

Zentrumsspiele

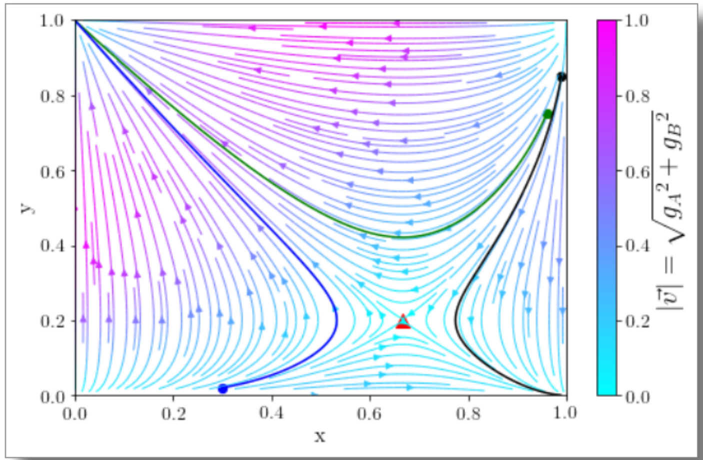
Spiel A: *Koordinationspiel*
Spiel B: *Anti-Koordinationspiel*
oder
Spiel A: *Anti-Koordinationspiel*
Spiel B: *Koordinationspiel*







Evolutionäre unsymmetrische (2×2) Spiele (Bi-Matrix Spiele)



Die im oberen Bereich des rechten Panels dieser Vorlesung dargestellten Gleichungen der Reproduktionsdynamik lassen sich nach Spezifikation der Auszahlungsmatrizen $\hat{\$}^A$ und $\hat{\$}^B$ auf unterschiedlichste Populationsspiele anwenden. Generell lassen sich evolutionäre unsymmetrische $(\mathbf{2} \times \mathbf{2})$ Spiele in die folgenden drei Spielklassen gliedern: Eckenspiele, Sattelpunktspiele und Zentrumsspiele. Ein Eckenspiel liegt vor, falls zumindest eine der Auszahlungsmatrizen der Spielergruppen eine dominante Struktur hat. Aufgrund der Dominanz der Strategie endet die zeitliche Entwicklung der Populationen, unabhängig von der Anfangsbedingungen, in einer Ecke des x-y Populationsvektor Diagramms - die ESS des Eckenspiels. Ein Sattelpunktspiel liegt vor, falls beide Spielergruppen gleichzeitig ein Koordinationsspiel oder Anti-Koordinationsspiel spielen. Bei einem Sattelpunktsspiel das aus zwei Anti-Koordinationsspielen besteht, existieren zwei ESSs ((x=0,y=1) oder (x=1,y=0)) zu denen sich die Populationsgruppen im Laufe der Zeit entwickeln. Welche dieser evolutionär stabilen Strategien erreicht wird, hängt von der Anfangsstrategiewahl der Population ab. Die nebenstehende Abbildung zeigt die zeitliche Entwicklung von drei Anfangszuständen in einem solchen Eckenspiel. Auch bei sehr ähnlichen Werten der Anfangsstrategiewahl kann es geschehen, dass sich die Population im Laufe der Zeit zu unterschiedlichen Ecken entwickelt (siehe grüne und schwarze Trajektorien). Eine besondere Bedeutung hat der Sattelpunkt des Spiels (siehe rotes Dreieck). Die Position des Sattelpunktes im x-y Diagramm entspricht dem Wert des gemischten Nash-Gleichgewichtes bzw. lässt sich durch die Nullstellen der Funktionen $g_A(x,y)$ und $g_B(x,y)$ bestimmen. Bei einem Zentrumsspielen existiert keine ESS, da die Strategiewahl der Population sich im Laufe der Zeit ständig verändert und um ein Zentrum kreist. Die Position dieses Zentrums im x-y Diagramm entspricht dem Wert des gemischten Nash-Gleichgewichtes bzw. lässt sich durch die Nullstellen der Funktionen $g_A(x,y)$ und $g_B(x,y)$ bestimmen (siehe Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele).

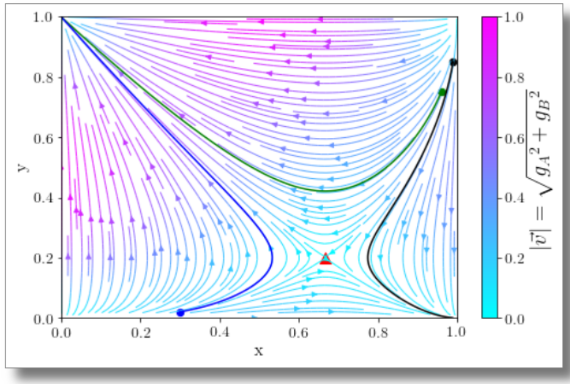
Weiterführende Links

- Folien der 4.Vorlesung
- Vorlesungsaufzeichnung der 4.Vorlesung: WS 2022/23 bzw. WS 2021/22
- View Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)
- View Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)

Jupyter Notebook

Auf der Internetseite der Vorlesung

Evolutionäre unsymmetrische (2×2) Spiele (Bi-Matrix Spiele)



Die im oberen Bereich des rechten Panels dieser Vorlesung dargestellten Gleichungen der Reproduktionsdynamik lassen sich nach Spezifikation der Auszahlungsmatrizen $\hat{\$}^A$ und $\hat{\$}^B$ auf unterschiedlichste Populationsspiele anwenden. Generell lassen sich evolutionäre unsymmetrische (2×2) Spiele in die folgenden drei Spielklassen gliedern: Eckenspiele, Sattelpunktspiele und Zentrumsspiele. Ein Eckenspiel liegt vor, falls zumindest eine der Auszahlungsmatrizen der Spielergruppen eine dominante Struktur hat. Aufgrund der Dominanz der Strategie endet die zeitliche Entwicklung der Populationen, unabhängig von der Anfangsbedingungen, in einer Ecke des x-y Populationsvektor Diagramms - die ESS des Eckenspiels. Ein Sattelpunktspiel liegt vor, falls beide Spielergruppen gleichzeitig ein Koordinationsspiel oder Anti-Koordinationsspiel spielen. Bei einem Sattelpunktsspiel das aus zwei Anti-Koordinationsspielen besteht, existieren zwei ESSs ((x=0,y=1) oder (x=1,y=0)) zu denen sich die Populationsgruppen im Laufe der Zeit entwickeln. Welche dieser evolutionär stabilen Strategien erreicht wird, hängt von der Anfangsstrategiewahl der Population ab. Die nebenstehende Abbildung zeigt die zeitliche Entwicklung von drei Anfangszuständen in einem solchen Eckenspiel. Auch bei sehr ähnlichen Werten der Anfangsstrategiewahl kann es geschehen, dass sich die Population im Laufe der Zeit zu unterschiedlichen Ecken entwickelt (siehe grüne und schwarze Trajektorien). Eine besondere Bedeutung hat der Sattelpunkt des Spiels (siehe rotes Dreieck). Die Position des Sattelpunktes im x-y Diagramm entspricht dem Wert des gemischten Nash-Gleichgewichtes bzw. lässt sich durch die Nullstellen der Funktionen $g_A(x,y)$ und $g_B(x,y)$ bestimmen. Bei einem Zentrumsspielen existiert keine ESS, da die Strategiewahl der Population sich im Laufe der Zeit ständig verändert und um ein Zentrum kreist. Die Position dieses Zentrums im x-y Diagramm entspricht dem Wert des gemischten Nash-Gleichgewichtes bzw. lässt sich durch die Nullstellen der Funktionen $g_A(x,y)$ und $g_B(x,y)$ bestimmen (siehe Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele).

Weiterführende Links

- Folien der 4.Vorlesung
- Vorlesungsaufzeichnung der 4.Vorlesung: WS 2022/23 bzw. WS 2021/22
- View Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)
- View Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)

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 2025/26)

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

Frankfurt am Main 27.10.2025

Erster Vorlesungsteil:

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

Einführung

In diesem Unterkapitel werden die unterschiedlichen Spieltypen evolutionärer Bi-Matrix Spiele (unsymmetrische (2×2)-Spiele) klassifiziert. Ausgangspunkt sind die folgenden allgemeinen Auszahlungsmatrizen der Spielergruppen A und B. Da es sich um unsymmetrische Auszahlungsmatrizen nehmen wir das Folgende an: $\hat{\$}^B \neq \left(\hat{\$}^A\right)^T$.

$$\hat{\$}^A = \begin{pmatrix} \$^A_{11} & \$^A_{12} \\ \$^A_{21} & \$^A_{22} \end{pmatrix} \quad , \quad \hat{\$}^B = \begin{pmatrix} \$^B_{11} & \$^B_{12} \\ \$^B_{21} & \$^B_{22} \end{pmatrix} \qquad (1)$$

Unsymmetrische (2×2) 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 $\$^A$ oder $\$^B$ ein dominantes Spiel ist.

Jupyter Notebook

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from scipy.integrate import solve_ivp
```

Definition der Funktion $g_A(x,y)$ und $g_B(x,y)$.

In [2]:
```
def gA(x,y,Aa,Ab,Ac,Ad):
    g = ((Aa+Ad-Ac-Ab)*y + (Ab-Ad))*(x-x*x)
    return g
def gB(x,y,Ba,Bb,Bc,Bd):
    g = ((Ba+Bd-Bc-Bb)*x + (Bc-Bd))*(y-y*y)
    return g
```

Definition des Systems der gekoppelten Differenzialgleichung des evolutionären Bi-Matrix Spiels.

In [3]:
```
def DGLsys(t,vx):
    x, y = vx
    dxdt = gA(x,y,Aa,Ab,Ac,Ad)
    dydt = gB(x,y,Ba,Bb,Bc,Bd)
    return [dxdt,dydt]
```

Beispiel I eines Eckenspiels

Wir betrachten die zeitliche Entwicklung eines Eckenspiels mit den folgenden Auszahlungswerten der beiden Spielergruppen:

$$\hat{\$}^A = \begin{pmatrix} 10 & 4 \\ 12 & 5 \end{pmatrix} \quad , \quad \hat{\$}^B = \begin{pmatrix} 10 & 12 \\ 7 & 5 \end{pmatrix} \tag{4}$$

Es handelt sich hierbei um eine Kombination von einem dominanten Spiel (Spielergruppe A: dominante Strategie 2 (x=0)) mit einem Anti-Koordinationsspiel (Spielergruppe B: zwei unsymmetrische Nash-Gleichgewichte).

Im Folgenden wird das evolutionäre Eckenspiel für die Anfangsbedingung $(x_0, y_0) = (0.9, 0.2)$ berechnet:

In [4]:
```
Aa,Ab,Ac,Ad = 10,4,12,5
Ba,Bb,Bc,Bd = 10,12,7,5
t_val = np.linspace(0, 7, 1000)
x0 = 0.9
y0 = 0.2
Loes = solve_ivp(DGLsys, [0, 7], [x0, y0], t_eval=t_val)
```

Auf der Internetseite der Vorlesung

```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from scipy.integrate import solve_ivp

# 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 + (c-d))*(y-y*y)
    return g

# Definition der DGL Systems
def DGLsys(t,vx):
    x, y = vx
    dxdt = gA(x,y,Aa,Ab,Ac,Ad)
    dydt = gB(x,y,Ba,Bb,Bc,Bd)
    return [dxdt,dydt]

# 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 unsymmetrischen (2x2)-
```

Weiterführende Links

- Folien der 4.Vorlesung
- Vorlesungsaufzeichnung der 4.Vorlesung: WS 2022/23 bzw. WS 2021/22
- View Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)
- View Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)

Python Programm Version 2 (bimatrix1a.py)

Zum Lösen des evolutionären Spiels mit Python werden wir in dieser Version die Funktion 'solve_ivp(...)' verwenden, die sich im Untermodul 'scipy.integrate' befindet, welche Funktionen zum Lösen von gewöhnlichen Differenzialgleichungen bereitstellt.

Lösen des evolutionären Spiels mit Python (Version 2)

```
# Weitere Festlegungen
numpoints = 1000
t_val = np.linspace(0, tend, numpoints)

# Loesung der DGL fuer eine Anfangspopulation (x0,y0)
Loes = solve_ivp(DGLsys, [0, tend], [x0, y0], t_eval=t_val)

# 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
fig, ax = plt.subplots()
strm = ax.streamplot(SX,SY,SgA,SgB,density=[2, 2], linewidth=1,color=colorspeed, cmap=plt.cm.cool)
ax.plot(Loes.y[0],Loes.y[1],c="black", linewidth=1.5, linestyle='-')
ax.plot(x0,y0, marker='o', color='grey', markersize=8)
for i in np.linspace(numpoints/30,numpoints-2,10):
    ax.arrow(Loes.y[0][int(i)],Loes.y[1][int(i)], Loes.y[0][int(i)+1] - Loes.y[0][int(i)], Loes.y[1][int(i)+1] - Loes.y[1][int(i)], head_width=0.02, head_length=0.03, fc='black', ec='black')
# Erzeugung der nebenstehenden Farblegende colorbar
cbar = fig.colorbar(strm.lines,ax=ax,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)
```

Bi-Matrix Spiele mit Python (Feldliniendiagramm)

Bi-Matrix Spiele mit Python V2 (Feldliniendiagramm)

```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from scipy.integrate import solve_ivp

# 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 + (c-d))*(y-y*y)
    return g

# Definition der DGL Systems
def DGLsys(t,vx):
    x, y = vx
    dxdt = gA(x,y,Aa,Ab,Ac,Ad)
    dydt = gB(x,y,Ba,Bb,Bc,Bd)
    return [dxdt,dydt]

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

#Festlegung der Auszahlungsmatrix des uns
```

```python
#y0=0.2
# Klasse der Sattelspiele (Saddle Class Game
Aa=10
Ab=4
Ac=9
Ad=5
Ba=10
Bb=7
Bc=4
Bd=5
tend=12
x0=0.6
y0=0.1
# Klasse der Zentrumsspiele (Center Class Ga
#Aa=10
```

```python
# Weitere Festlegungen
numpoints = 1000
t_val = np.linspace(0, tend, numpoints)

# Loesung der DGL fuer eine Anfangspopulation (x0,y0)
Loes = solve_ivp(DGLsys, [0, tend], [x0, y0], t_eval=t_val)

# 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
fig, ax = plt.subplots()
strm = ax.streamplot(SX,SY,SgA,SgB,density=[2, 2], linewidth=1,color=colorspeed, cmap=plt.cm.cool)
ax.plot(Loes.y[0],Loes.y[1],c="black", linewidth=1.5, linestyle='-')
ax.plot(x0,y0, marker='o', color='grey', markersize=8)
for i in np.linspace(numpoints/30,numpoints-2,10):
    ax.arrow(Loes.y[0][int(i)],Loes.y[1][int(i)], Loes.y[0][int(i)+1] - Loes.y[0][int(i)], Loes.y[1][int(i)+1] - Loes.y[1][int(i)], head_width=0.02, head_length=0.03, fc='black', ec='black')
# Erzeugung der nebenstehenden Farblegende colorbar
cbar = fig.colorbar(strm.lines,ax=ax,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.xticks([0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1],["$0$","","$0.2$","","$0.4$","","$0.6$","","$0.8$","","$1$"])
plt.yticks([0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1],["$0$","","$0.2$","","$0.4$","","$0.6$","","$0.8$","","$1$"])
plt.ylabel(r"$\rm y$")
plt.xlabel(r"$\rm x$")

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

Auf der Internetseite der Vorlesung

Weiterführende Links

- Folien der 4.Vorlesung
- Vorlesungsaufzeichnung der 4.Vorlesung: WS 2022/23 bzw. WS 2021/22
- View Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)
- View Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)

```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 + (c-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][0] = t[0]
    sol[0][1] = x0
    sol[0][2] = y0
    for i in range(1,numpoints):
        sol[i][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][1] = sol[i-1,1] + dx
        sol[i][2] = sol[i-1,2] + dy
    return sol

# Groessenfestlegung der Labels usw. im Bild
params = {
    'text.usetex'        : True,
    'axes.titlesize' : 22,
    'axes.labelsize' : 20
```

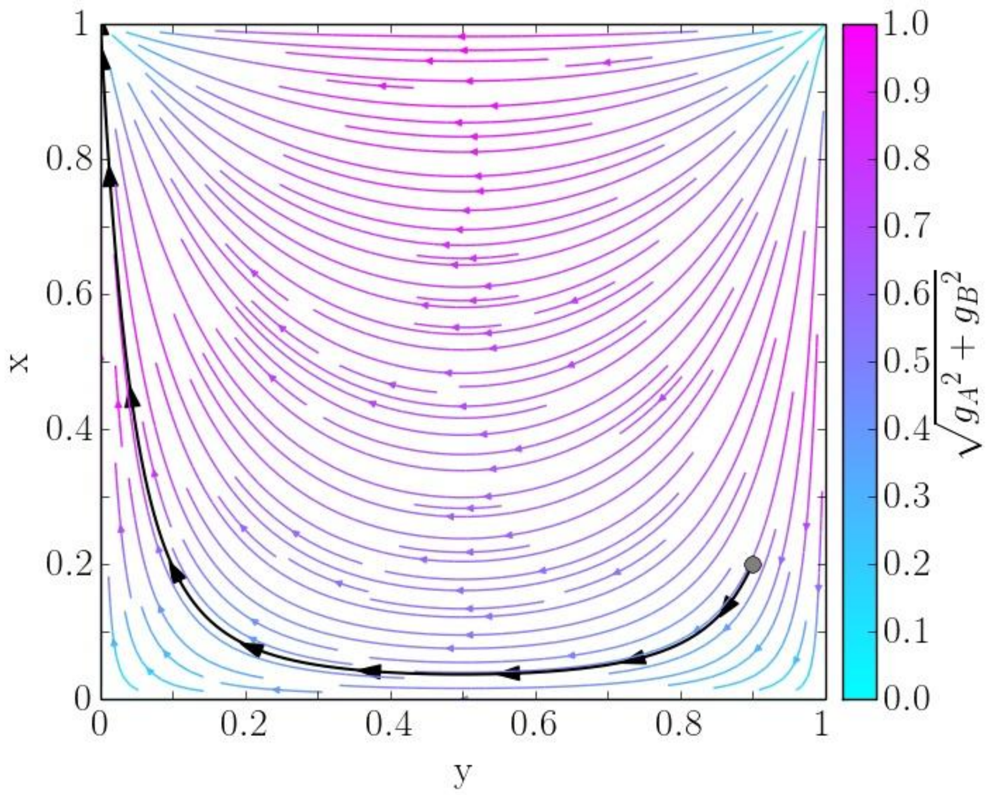
Python Programm Version 1 (bimatrix1.py)

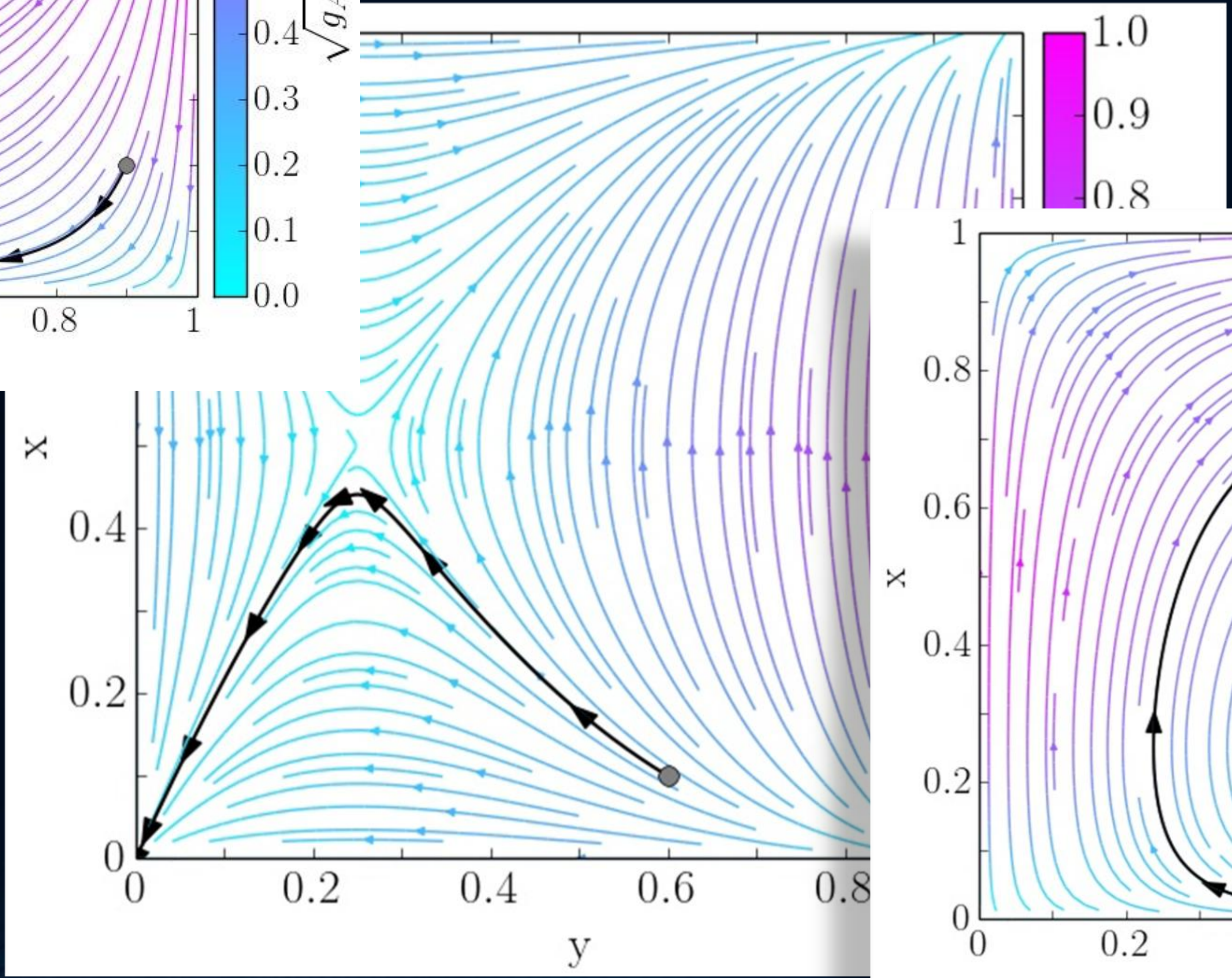
Lösen des evolutionären Spiels mit Python,
hier mit dem einfachen Euler-Verfahren

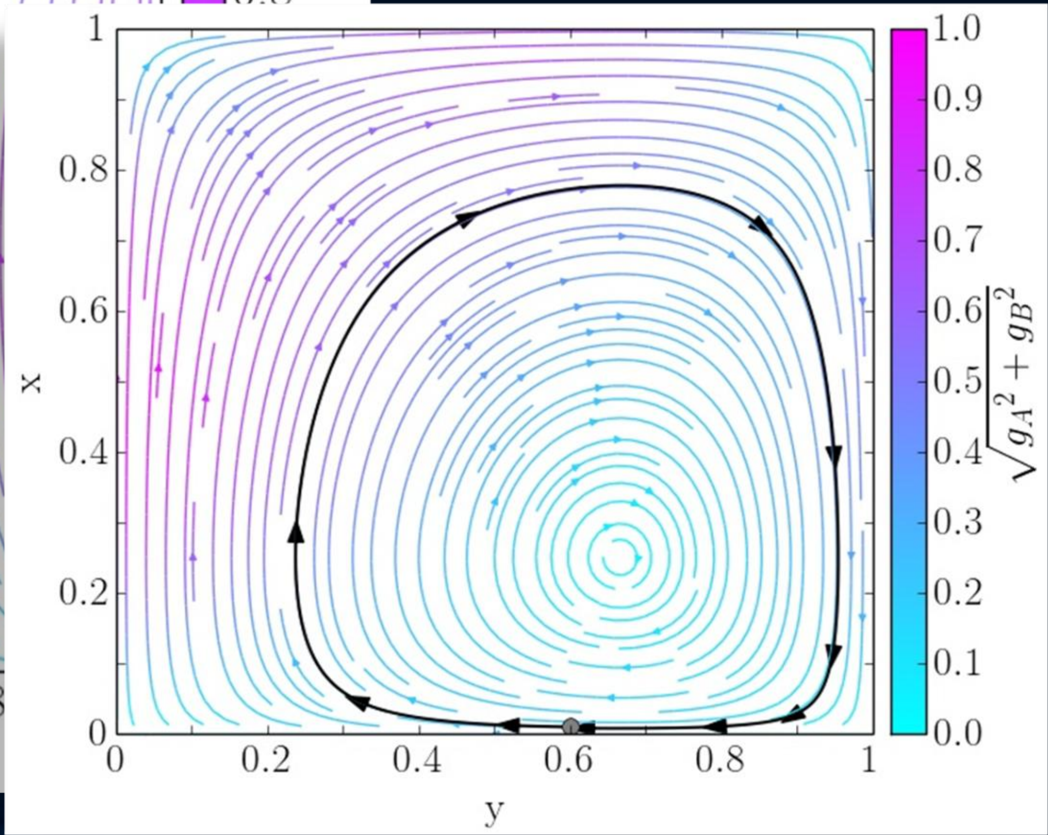
Lösen des evolutionären Spiels mit Python (Version 1)

```python
# 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)

# 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
fig, ax = plt.subplots()
strm = ax.streamplot(SX,SY,SgA,SgB,density=[2, 2], linewidth=1,color=colorspeed, cmap=plt.cm.cool)
ax.plot(Loes[:,1],Loes[:,2],c="black", linewidth=1.5, linestyle='-')
ax.plot(Loes[0,1],Loes[0,2], marker='o', color='grey', markersize=8)
for i in np.linspace(numpoints/30,numpoints-2,10):
    ax.arrow(Loes[int(i),1],Loes[int(i),2], Loes[int(i)+1,1] -Loes[int(i),1], Loes[int(i)+1,2] -Loes[int(i),2], head_width=0.02, head_length=0.03, fc='black', ec='black')
# Erzeugung der nebenstehenden Farblegende colorbar
cbar = fig.colorbar(strm.lines,ax=ax,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.xticks([0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1],["$0$","","$0.2$","","$0.4$","","$0.6$","","$0.8$","","$1$"])
plt.yticks([0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1],["$0$","","$0.2$","","$0.4$","","$0.6$","","$0.8$","","$1$"])
plt.ylabel(r"$\rm y$")
plt.xlabel(r"$\rm x$")

#Speichern der Bilder als (.png , benötigt dvipng unter Linux)- und .pdf-Datei
saveFig="./bimatrix.png"
```

Bi-Matrix Spiele mit Python
Version 1 und 2







Auf der Internetseite der Vorlesung

Weiterführende Links

- Folien der 4.Vorlesung
- Vorlesungsaufzeichnung der 4.Vorlesung: WS 2022/23 bzw. WS 2021/22
- View Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie symmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)
- View Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Jupyter Notebook: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele
- Download Python Programm: Evolutionäre Spieltheorie unsymmetrischer (2x2)-Spiele (Version 1 , Version 2 , Version 3)

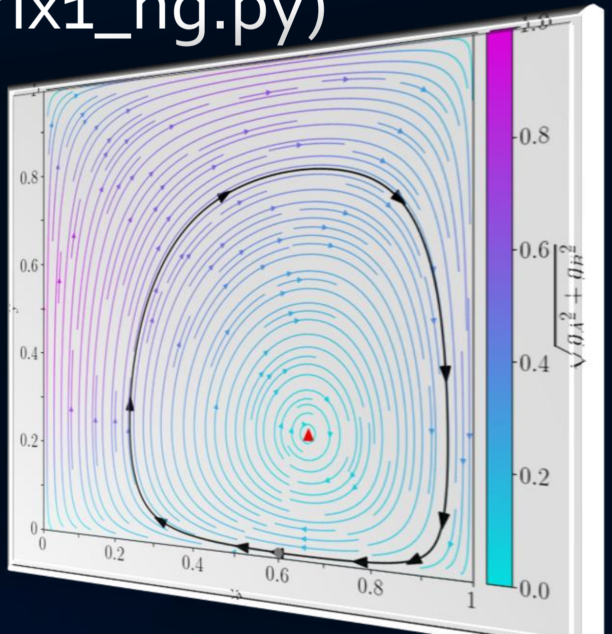
```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams
from scipy.integrate import solve_ivp
from sympy import *

# DGL bestimmende Funktion g_A und g_B
def gA(x,y,D):
    return ((D[0,0]+D[1,1]-D[0,1]-D[1,0])*y + (D[0,1]-D[1,1]))*(x-x*x)

def gB(x,y,D):
    return ((D[0,0]+D[1,1]-D[0,1]-D[1,0])*x + (D[1,0]-D[1,1]))*(y-y*y)

# Definition des DGL Systems
def DGLsys(t,vx,DA,DB):
    x, y = vx
    dxdt = gA(x,y,DA)
    dydt = gB(x,y,DB)
    return [dxdt,dydt]

# Berechnung der Nashgleichgewichte
def find_nash_equilibria(DA,DB):
    equilibria = []
    # Berechnung der reinen Nash-Gleichgewichte
    for i in range(2):
        for j in range(2):
            if DA[i, j] == max(DA[:, j]) and DB[i, j] == max(DB[i, :]):
                equilibria.append({'type': 'pure', 's': (i+1,j+1)})
    #Berechnung der gemischten Nash-Gleichgewichte
    x, y = symbols('x, y')
    Dollar_A = DA[0,0]*x*y + DA[0,1]*x*(1-y) + DA[1,0]*(1-x)*y + DA[1,1]*(1-x)*(1-y)
    Dollar_B = DB[0,0]*x*y + DB[0,1]*x*(1-y) + DB[1,0]*(1-x)*y + DB[1,1]*(1-x)*(1-y)
    GlGemNashA = Eq(Dollar_A.diff(x), 0)
    GlGemNashB = Eq(Dollar_B.diff(y), 0)
    if GlGemNashA != False and GlGemNashB != False:
        s_A = solve(GlGemNashB,x)[0]
        s_B = solve(GlGemNashA,y)[0]
        if 0 < s_A < 1 and 0 < s_B < 1:
            equilibria.append({'type': 'mixed', 's*': (s_A,s_B)})
    return equilibria
```

Python Programm Version 3 (bimatrix1_ng.py)

Modularisierung der Version 2 mittels unterschiedlicher Funktionen

Berechnung der Nash-Gleichgewichte

Automatische Kennzeichnung der Nash-Gleichgewichte im Bild



E-Learning und interaktive Übungsaufgaben

Zusäzlich zu den Informationen aus dieser Internetseite finden Sie in diesem Unterpunkt diverse interaktive Übungsaufgaben zu den folgenden Themen:

Aufgabe 1

Reine Nash-Gleichgewichte in einem simultanen (2x2)-Spiel in strategischer Form mit symmetrischer Auszahlungmatrix

Aufgabe 2

Gemischtes Nash-Gleichgewicht in einem simultanen (2x2)-Spiel in strategischer Form mit symmetrischer Auszahlungmatrix

Aufgabe 3

Das gemischtes Nash-Gleichgewicht im Hirschjagt-Spiel

Aufgabe 4

Spielklassen von simultanen (2x2)-Spielen in strategischer Form mit symmetrischer Auszahlungmatrix

Aufgabe 5

Zeitliche Entwicklung des Populationsvektors im evolutionären Spiel

Aufgabe 6

Evolutionär stabile Strategien

Aufgabe 7

Zeitliche Entwicklung des Populationsvektors im evolutionären Bi-Matrix Spiel

Aufgabe 8

Gemischtes Nash-Gleichgewicht in einem simultanen (2x2)-Spiel in strategischer Form mit unsymmetrischer Auszahlungmatrix

Aufgabe 9

Gemischtes Nash-Gleichgewicht und zeitliche Entwicklung des Populationsvektors in Zentrumsspielen

Aufgabe 10

Zeitliche Entwicklung des Populationsvektors im evolutionären (2x3)-Spiel

Aufgabe 11

Gemischtes Nash-Gleichgewicht im evolutionären (2x3)-Spiel

Aufgabe 12

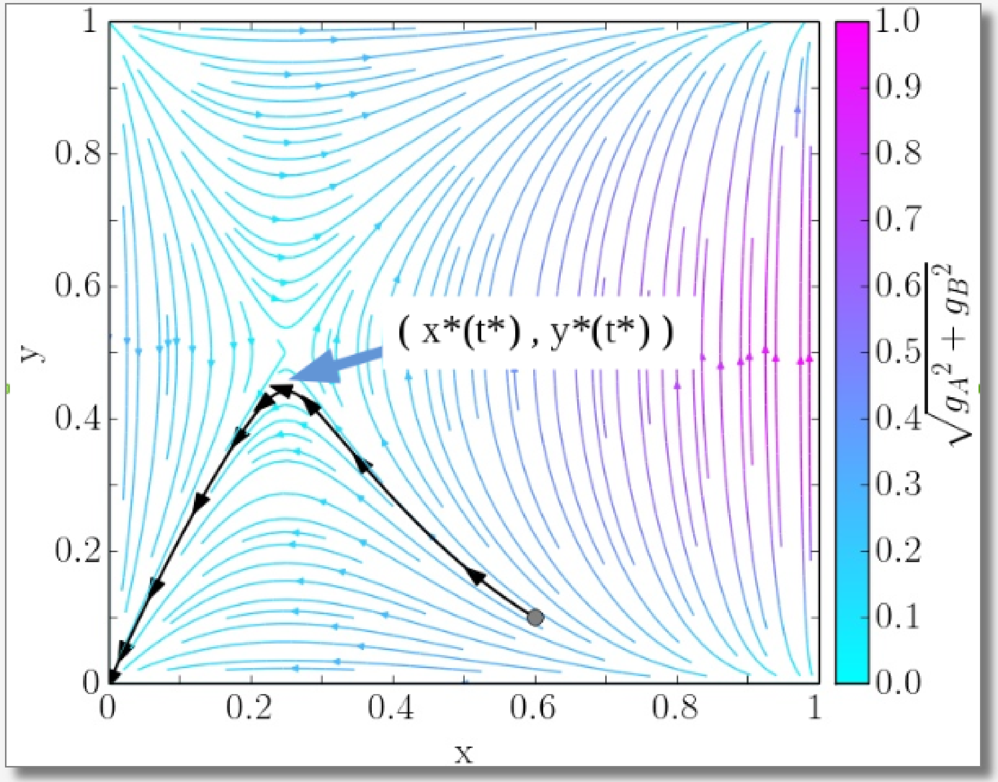
Mittlere Distanz zwischen zwei Knoten in einem zufälligen Netzwerk

Aufgabe 7

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:

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

Qualitative Veranschaulichung der Aufgabenstellung



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

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

Der Populationsvektor zur Zeit t=0 sei (x(0) = 0.6 , y(0) = 0.0937). Der Anteil der Spieler in der Gruppe B die die Strategie s_1^B spielen nimmt zunächst zu, erreicht dann ein Maximum und nimmt danach wieder ab (siehe nebenstehende Abbildung). Berechnen Sie den Zeitpunkt t^* an dem der maximale Wert y^* erreicht wird. Tragen Sie bitte die beiden Werte in die unteren Eingabenfelder ein

t^* = 0,000 , y^* = 0,000

und vergleichen Sie indem Sie den folgenden *Button* drücken.

Lösung anzeigen

Weiter zur nächsten Aufgabe ...

Lösung

Der maximale Wert y-Wert beträgt $y^* =$
und er wird zum Zeitpunkt $t^* =$ erreicht.

Aufgabe 8

A/B	s_1	s_2
s_1	(210 , 292)	(132 , 120)
s_2	(37 , 125)	(392 , 378)

Betrachten Sie die gemischte Erweiterung eines simultanen (2 Spieler)-(2 Strategien) Spiels in strategischer Form mit unsymmetrischer Auszahlungmatrix. Die Menge der Spieler sei $\mathcal{I} = \{A, B\}$, die Menge der reinen Strategien sei $\mathcal{S}^A = \mathcal{S}^B = \{s_1, s_2\}$ und die Präferenzordnungen der Spieler sei durch die neben stehende Auszahlungstabelle quantifiziert. Die reinen Strategien entsprechen den folgenden gemischten Strategien: $s_1 \hat{=} \tilde{s}^B = \tilde{s}^B = 1$ und $s_2 \hat{=} \tilde{s}^A = \tilde{s}^B = 0$.
Bei welcher gemischten Strategienkombination $(\tilde{s}^{A\star}, \tilde{s}^{B\star})$ befindet sich das gemischte Nash-Gleichgewicht? Tragen Sie bitte Ihre Werte in die unteren Eingabenfelder ein

$\tilde{s}^{A\star} =$ 0,000 , $\tilde{s}^{B\star} =$ 0,000

und vergleichen Sie indem Sie den folgenden *Button* drücken.

Lösung anzeigen

Lösung

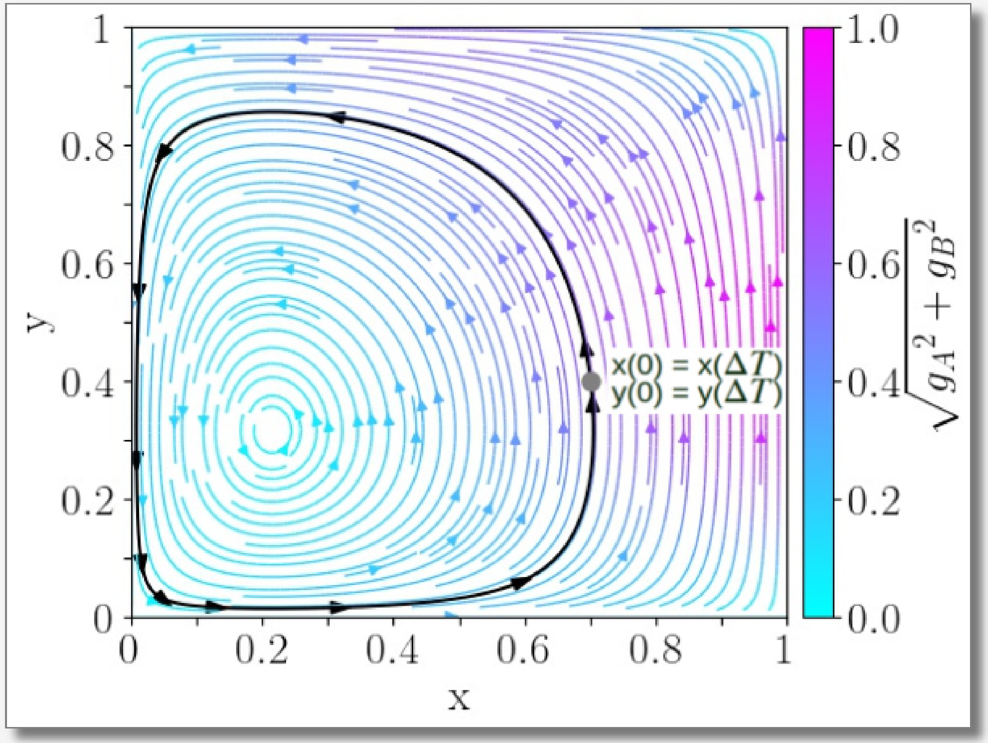
Das gemischte Nash-Gleichgewicht besindet sich bei
$\tilde{s}^{A\star} =$
$\tilde{s}^{B\star} =$

Aufgabe 9

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:

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

Qualitative Veranschaulichung der Aufgabenstellung



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

A/B	s_1^B	s_2^B
s_1^A	$(8, 8)$	$(6, c_B)$
s_2^A	$(c_A, 4)$	$(5, 5)$

Der Populationsvektor zur Zeit t=0 sei (x(0)=0.7 , y(0)=0.4). Wählen Sie die noch offenen Werte der Auszahlungsmatrix (c_A und c_B) so, dass sich das gemischte Nash-Gleichgewicht des Spiels bei $x^\star = \tilde{s}^{A\star} = 0.3927586$ und $y^\star = \tilde{s}^{B\star} = 0.4927586$ befindet. Berechnen Sie dann die Zeit ΔT, die die Population benötigt um wieder zu ihrer Anfangskonfiguration zurück zu kehren: x(0) = x(ΔT) und y(0) = y(ΔT). Tragen Sie bitte die berechneten Werte in die unteren Eingabenfelder ein

c_A = 0,000 , c_B = 0,000

ΔT = 0,000

und vergleichen Sie indem Sie den folgenden *Button* drücken.

Lösung anzeigen

Weiter zur nächsten Aufgabe ...

Lösung

Die Werte der Auszahlungsmatrix betragen
$c_A =$ und $c_B =$

und die Zeit eines Umlaufes des Populationsvektors beträgt
$\Delta T =$