Graphentheoretische Algorithmen für dynamische Spiele

Diplomarbeit

von

Diana Balbus

FAKULTÄT FÜR MATHEMATIK UND PHYSIK

MATHEMATISCHES INSTITUT

Datum: 21.Januar 2008

Aufgabenstellung/Betreuung:
Prof. Dr. Lars Grüne

Danksagung

Mein Dank geht zunächst an Herrn Prof. Dr. Lars Grüne und Dipl. Math. Marcus v. Lossow für die interessante Themenbereitstellung und hervorragende Betreuung meiner Arbeit. Gleichgültig wann ich mit Fragen an den Lehrstuhl kam, sie haben sich immer Zeit für mich genommen.

Danken möchte ich an dieser Stelle auch meinen Eltern, die mir die Zeit an der Universität überhaupt erst ermöglicht haben. Sie haben jede Phase meiner Studienlaufbahn seelisch und moralisch mit durchlebt und jede meiner Entscheidungen immer unterstützt.

Meiner besten Freundin möchte ich im Besonderen danken. Seit Jahren steht sie mir zur Seite und ist in allen Lebenslagen für mich da. Ich schätze mich glücklich, sie kennengelernt zu haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Seit Jahrhunderten dienen Spiele den Menschen als Unterhaltung, als Reiz der Spannung und/oder als Mittel zur Modellierung mathematischer, finanzwirtschaftlicher und strategischer Fragestellungen. Das Streben nach dem bestmöglichen Gewinn beziehungsweise nach dem Optimum ist das Ziel jedes Einzelnen. Konkurrenzdenken wird geschürt, gemeinsame Interessen werden in Form von Kooperationen vertreten. Sobald zwei oder mehr Individuen an einem Spiel beteiligt sind, entsteht ein Kampf um die Verwirklichung von Zielen, Wünschen und Vorstellungen.
Immer häufiger werden Computer eingesetzt, um den Wert des Spiels oder optimale Strategien einzelner Spieler zu berechnen. Wie hoch ist der Wert für unterschiedliche Ausgangssituationen? Wie agiert ein Spieler, wenn er seinen Zielen entgegenstrebt?
Diese Arbeit soll sich mit dynamischen Spielen beschäftigen, die mittels einer Modifikation des Dijkstra-Algorithmus numerisch behandelt werden.

Bei einem dynamischen Spiel gibt es zwei Spieler, die miteinander in Konflikt stehende Interessen verfolgen. Jeder der beiden möchte die bestmögliche Entscheidung für sich treffen und zwar in dem Wissen, dass sein Antagonist dasselbe möchte. Während der eine Spieler eine Strategie des Minimierens verfolgt, verfolgt der andere eine des Maximierens. Was dem einen von Vorteil ist, gereicht dem anderen zum Nachteil. In dieser Arbeit werden Verfolgungsspiele betrachtet, die sich in der Literatur einen Namen als „Pursuit-Evasion-Games“ gemacht haben. Der Pursuer (zu deutsch Jäger oder Verfolger) hat zum Ziel, seinen Gegenpart, den Evader (zu deutsch Gejagter oder Verfolgter), zu fassen, sei dies in Form des Fangs an sich, des Überfahrens (Homicidal Chauffeur Game) oder des Zerstörens (Raketenabwehrsysteme).
Der Wert des Spiels errechnet sich über die Kosten, die die Spieler in Kauf nehmen, respektive den Nutzen, den sie erreichen. Das Auffinden optimaler Strategien sowie die Bestimmung des eben genannten Wertes für jede erdenkliche Ausgangssituation ist oft ein schwieriges Unterfangen. Dank der heute sehr hochwertigen Computertechnologie kommen neben analytischen Methoden immer häufiger numerische Verfahren hierfür zum Einsatz. Der Ermittlung der optimalen Wertefunktion und dem Behandeln sowie Vergleichen unterschiedlicher Strategien wird in dieser Arbeit eine Modifikation des bekanntesten Kürzeste-Wege-Algorithmus, Dijkstra-Algorithmus, dienen. Grundlage für diesen Algorithmus bilden allerdings Kontrollsysteme, die ebenfalls eine Dynamik beschreiben und sowohl in kontinuierlicher als auch in diskreter Zeit vorliegen können. Numerisch ist die Auswertung eines Verfolgungsspiels nicht zu jedem Zeitpunkt möglich,

weshalb eine Diskretisierung unabdingbar sein wird. Aus diesem Grund wird bei der Übertragung des Spiels auf ein gestörtes Kontrollsystem nur auf die zweite Variante zugegriffen werden.
Gestörte Kontrollsysteme hängen von zwei Parametern, $u \in U$ und $w \in W$, ab, die sich zeitabhängig und/oder zustandsabhängig verändern können und später unter dem Einfluss der Spieler stehen werden. Ohne den Zusammenhang mit dynamischen Spielen kann der Parameter u als Steuergröße interpretiert werden, die von außen aktiv beeinflussbar ist, während w als Störung angesehen wird, die auf das System einwirkt.
Wem der Algorithmus von Dijkstra geläufig ist, der dürfte wissen, dass die kürzesten Wege von einem Punkt A zu anderen Punkten B_i, $i = 1, ..., m$, innerhalb der Struktur eines Graphen berechnet werden. Aufgrund der Existenz zweier veränderlicher Parameter, u und w, muss allerdings eine Hypergraphstruktur herangezogen werden, die jedem Übergang des Systems mehrere mögliche Folgezustände zuweist.

In Kapitel 2 werden alle erforderlichen theoretischen Grundlagen gelegt. Begriffe wie das gestörte Kontrollsystem in diskreter Zeit, dynamisches Spiel und Hypergraph werden formal eingeführt. Anhand eines kleinen Beispiels soll die Ausführung der gestörten Variante des Dijkstra-Algorithmus dem Leser nahe gebracht werden. Eine Reihe von Routinen werden zeigen, wie die Berechnung der optimalen Wertefunktion numerisch umgesetzt wird. Für die graphische Veranschaulichung der Resultate werden einige Implementierungen in dem Software-Paket Matlab vorgestellt.

Kapitel 3 und 4 behandeln zwei gänzlich unterschiedliche Verfolgungsspiele. Die erste Anwendung des Algorithmus erfolgt für das unter dem Namen „Homicidal Chauffeur Game“ bekannte Differentialspiel; die zweite wiederum für das dynamische Spiel „The Lady In The Lake“. Das Hauptaugenmerk liegt neben der Approximation der optimalen Wertefunktion sowohl von unten als auch von oben auf dem Verhalten der Spieler. Man muss sich die Frage stellen, was passiert, wenn die Spieler von einer optimalen Strategie abweichen. Ausführliche Erläuterungen der Konsequenzen der Parametersetzung sollen die Folgen kenntlich machen. Darüberhinaus wird ein Vergleich mit den Ergebnissen einer analytischen Untersuchung dieser Spiele, die in [2] mit Hilfe der Isaacs-Gleichung durchgeführt wurde, gezogen.

2 Theoretische Grundlagen

Kontrollsysteme finden Anwendung sowohl in den Naturwissenschaften und der Medizin als auch in der Technik und der Ökonomie. Das betrachtete System, hier das dynamische Spiel, soll aus einem bekannten Ist-Zustand x_0 effizient möglichst nahe an einen gewünschten, vorgegebenen Endzustand herangeführt werden. Aufgrund der numerischen und damit computergestützten Herangehensweise wird auf die diskrete Variante gestörter Kontrollsysteme zurückgegriffen. Der Endzustand wird in einer Menge $\mathcal{D}$ liegen, die die Terminierung des Spiels symbolisiert.
Ziel dieses Kapitels ist es, die diskrete optimale Wertefunktion, später mit $V_{\mathcal{P}}$ bezeichnet, eines dynamischen Spiels zu berechnen. Dazu wird der Zustandsraum partitioniert, das heißt in viele, gleich große Zellen unterteilt werden. Für jeden Punkt x_0 einer Zelle wird der Wert des Spiels dem kürzesten Weg von x_0 nach $\mathcal{D}$ entsprechen, wobei dieser Wert innerhalb einer Zelle gleich sein wird, was mittels entsprechender Projektionen erreicht werden kann. Die Modifikation des Dijkstra-Algorithmus, die zur Berechnung von $V_{\mathcal{P}}$ eingesetzt wird, erfordert Grundlagen aus der Graphentheorie, die in 2.2 gegeben werden sollen. Die Unterkapitel 2.3 bis 2.5 beschäftigen sich mit der formalen Einführung dynamischer Spiele, ihrer Handhabung als mehrwertige Abbildungen, ihrer Diskretisierung und letztendlich ihrer graphentheoretischen Formulierung. Den Abschluss von 2.5 bildet der Algorithmus selbst. 2.6 erläutert einige Routinen der Implementierung.

2.1 Diskretes gestörtes Kontrollsystem

Definition 2.1 (Diskretes gestörtes Kontrollsystem).
Ein diskretes gestörtes Kontrollsystem ist gegeben durch

$$x_{k+1} = f(x_k, u_k, w_k), \qquad k = 0, 1, \ldots,$$

wobei $f : X \times U \times W \to X$ stetig, $X \subset \mathbb{R}^d$ und $x_k \in X$ der Zustand des Systems ist; $u_k \in U$ heißt der Kontrollinput und $w_k \in W$ der Störinput des Systems.
Zusätzlich, um die Entwicklung zu beschreiben, ist eine stetige Kostenfunktion $g : X \times U \to [0, \infty)$, die jedem Übergang $x_{k+1} = f(x_k, u_k, w_k)$ die Kosten $g(x_k, u_k)$ zuweist, gegeben.

Aus dieser Definition lässt sich leicht der Begriff der Trajektorie herleiten. Für einen gegebenen Anfangswert $x \in X$, eine Kontrollsequenz $\boldsymbol{u} = (u_k)_{k\in\mathbb{N}} \in U^{\mathbb{N}}$ und eine Störfolge $\boldsymbol{w} = (w_k)_{k\in\mathbb{N}} \in W^{\mathbb{N}}$ erhält man induktiv eine Trajektorie, die der Vorschrift

$$\begin{aligned} x_0 &= x \\ x_{k+1} &= f(x_k(x, \boldsymbol{u}, \boldsymbol{w}), u_k, w_k), \qquad k = 0, 1, \ldots \end{aligned}$$

genügt.
Die zugehörigen akkumulierten Kosten sind durch

$$J(x, \boldsymbol{u}, \boldsymbol{w}) = \sum_{k=0}^{\infty} g(x_k(x, \boldsymbol{u}, \boldsymbol{w}), u_k) \tag{1}$$

bestimmt.

Zum einen werden sich die dynamischen Spiele der Kontrolltheorie bedienen sowie die Kontrolltheorie des Konstruktes mehrwertiger Spiele. Je nachdem, aus welchem Blickwinkel man sich das betrachtet. Die Methoden der dynamischen Programmierung und des Bellmannschen Optimalitätsprinzips, die eine Formel für die optimale Wertefunktion liefern, werden auf die spieltheoretische Herangehensweise der Behandlung von Kontrollsystemen verallgemeinert werden. Und doch liegt der Fokus auf den Spielen selbst.
Der folgende Abschnitt wird den Grundstock für die Berechnungen legen:

2.2 Graphentheorie

Mit Hilfe von Graphen können viele Probleme modelliert werden. Sie können dazu benutzt werden, um kürzeste Wege zwischen zwei Orten zu ermitteln, die durch den Algorithmus von Dijkstra (z.B. in [4]) effizient berechnet werden können.
In dieser Arbeit wird eine modifizierte Variante des Algorithmus von Dijkstra als Grundlage zur Behandlung dynamischer Spiele dienen. Doch bevor der Algorithmus vorgestellt werden kann, bedarf es einiger Grundbegriffe. Es muss letztendlich eine Struktur geschaffen werden, die es ermöglicht, die Systemübergänge des gestörten Kontrollsystems zu beschreiben und zu verwalten.

2.2.1 Graph und Hypergraph

Definition 2.2 (**Graph**)**.**
Ein Graph G ist ein Tupel (V, E), wobei $V = \{v_1, v_2, \ldots, v_n\}$ eine Menge von Knoten und $E = \{E_1, E_2, \ldots, E_m\}$ eine Menge von Kanten bezeichnet. Dabei ist E in einem gerichteten Graphen eine Teilmenge des kartesischen Produkts $V \times V$ und wird anschaulich mit einem Pfeil dargestellt. Ein Graph heißt (Kanten)gewichtet, wenn er an seinen Kanten semantische Inhalte in Form sogenannter Gewichte besitzt. Hierbei ordnet die Funktion $w : E \to \mathbb{R}$ jeder Kante eine reelle Zahl zu.

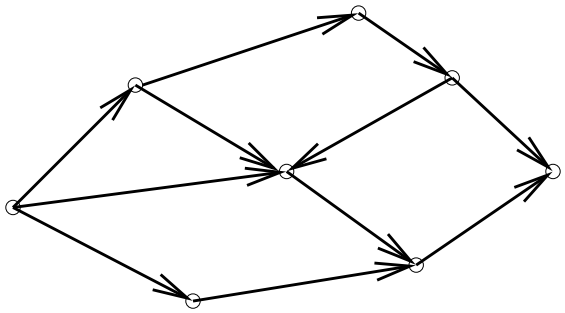


Abbildung 1: Gerichteter Standardgraph

Einfache Graphen wie in Abbildung 1 reichen für diese Arbeit jedoch nicht aus. Von einem Zustand x_k ausgehend können durch die verschiedenen Störparameter $w \in W$ mehrere Nachfolgezustände x_{k+1} erreicht werden. Dies wird durch folgende Definition veranschaulicht (nach [6]):

Definition 2.3 (Hypergraph).
Ein Hypergraph H ist ein Tupel (V, E), wobei $V = \{v_1, v_2, \ldots, v_n\}$ die Kantenmenge und $E = \{E_1, E_2, \ldots, E_m\}$, $E_i \subseteq V$ für $i = 1, .., m$, die Menge der Hyperkanten bezeichnet.

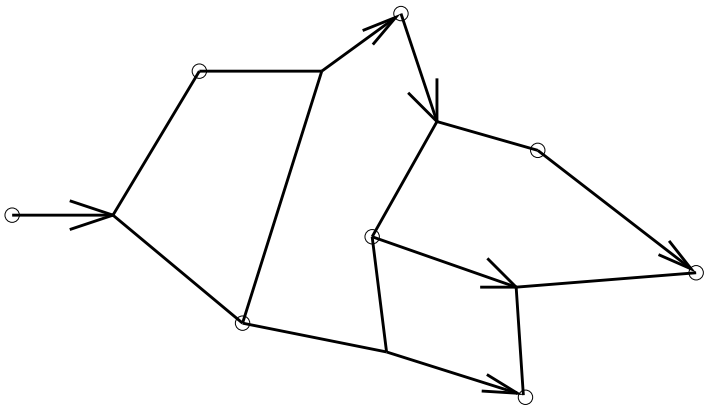


Abbildung 2: Hypergraph

Bei Hypergraphen (s. Abb.2) verbindet eine Kante also nicht nur zwei, sondern mehrere Knoten gleichzeitig. Zudem spricht man nicht nur von Kanten, sondern von Hyperkanten. Offensichtlich ist der Hypergraph ein Standardgraph, falls $|E_i| = 2$ für $i = 1, .., m$ gilt.
Die Größe eines Standardgraphen ist komplett durch die Größen n und m bestimmt. Die Größe eines Hypergraphen hängt jedoch auch noch von der Mächtigkeit seiner Hyperkanten ab, also $\mathsf{size}(H) = \sum_{E_i \in E} |E_i|$. Dies deutet bereits an, dass ein Hypergraph sehr schnell sehr groß werden kann. Dieser Fakt wird später noch einmal aufgegriffen werden.

Definition 2.4 (**Gerichtete Hyperkante**)**.**
Eine gerichtete Hyperkante ist ein geordnetes Paar $E = (T(E), H(E))$ *von disjunkten Teilmengen von Knoten aus* V*.* $T(E)$ *heißt der Schwanz (engl: tail) von* E*, während* $H(E)$ *der Kopf (engl: head) von* E *genannt wird.*

Es gibt zwei spezielle Arten von gerichteten Hyperkanten: die 'B-arcs' und die 'F-arcs'. Bereits in Abbildung 2 waren beide Typen vertreten. Während sich die B-arcs durch die Existenz nur eines einzigen Endknotens auszeichnen, rühmen sich die F-arcs durch das Vorhandensein nur eines einzigen Anfangsknotens:

(a) B-arc (b) F-arc

Abbildung 3: Typen von Hyperkanten

In dieser Arbeit werden ausschließlich F-Graphen betrachtet, das heißt Hypergraphen, deren Hyperkanten allesamt Vorwärtshyperkanten, d.h. 'F-arcs', sind. Formal ist eine Vorwärtshyperkante das Tupel $E = (T(E), H(E))$ mit $|T(E)| = 1$ und ist in Abbildung 4 in verschiedener Ausführung zu finden. Mal werden zwei Knoten durch eine Hyperkante miteinander verbunden, mal drei und mal vier.

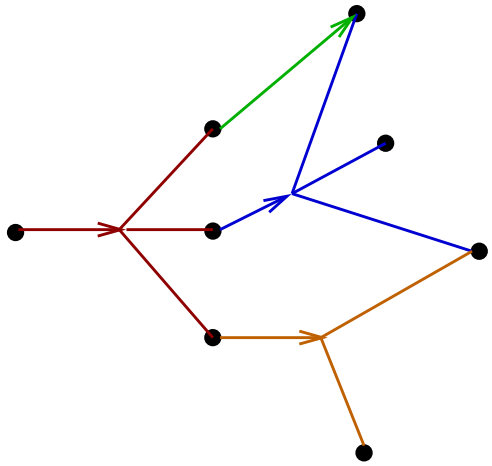


Abbildung 4: F-Graph

Bereits im Jahr 1984 sind Hypergraphen unter dem Namen 'labelled graphs' durch Dowling und Gallier in wissenschaftlichen Veröffentlichungen aufgetreten. Insbesondere F-Graphen wurden im Zusammenhang mit städtischen Transitproblemen von Nguyen und Pallottino (1988,1989) diskutiert. Hier sollen sie im Gegenzug der Berechnung der optimalen Wertefunktion von dynamischen Spielen dienen. Aus diesem Grund müssen die gerichteten Hyperkanten zusätzlich noch gewichtet werden.

Bemerkung 2.5. *Hypergraphen können ebenso wie Graphen gewichtet werden. Die genaue Definition der Gewichte der Hyperkanten wird in Abschnitt 2.3 über die Definition eines mehrwertigen Spiels eingeführt werden.*

Die optimale Wertefunktion, ebenfalls aus Abschnitt 2.3, wird der Länge des kürzesten Weges entsprechen, wobei die Länge als Summe der Bewertungen der Kanten des Weges definiert sein wird.

Definition 2.6 (Weg).
Ein Weg P_{st} der Länge q in einem Hypergraphen $H = (V, E)$ ist eine Folge von Knoten und Hyperkanten $P_{st} = (v_1 = s, E_{i_1}, v_2, E_{i_2}, \ldots, E_{i_q}, v_{q+1} = t)$, $s \in T(E_{i_1})$, $t \in H(E_{i_q})$, $v_j \in H(E_{i_{j-1}}) \cap T(E_{i_j})$, $j = 1, \ldots, q$.
Knoten s und t heißen Quelle bzw. Senke von P_{st} und man sagt, dass t und s verbunden sind.

Existiert kein Weg von s nach t, wird die Länge auf ∞ gesetzt.

2.3 Dynamische Spiele

Dynamische Spiele – genauer: Verfolgungsspiele – sind Nullsummen-Differentialspiele. Zwar handelt es sich im ursprünglichen Spiel um kein Nullsummenspiel, aber die Strategien der Spieler verfolgen ein minimax-Konzept, das ein Nullsummenspiel löst. Um sich das klarzumachen, erfolgt hier der Verweis auf [2, S.11]. Die Spieldauer ist im Vornhinein nicht festgelegt, weshalb man von deterministischen 2-Personen-Nullsummen-Differentialspielen mit variabler Endzeit spricht. Ein Verfolgungsspiel terminiert, falls der Verfolgte gefangen wird, das heißt sich innerhalb einer Distanz kleiner ε zum Jäger befindet. Die Bewegung der Spieler in der Zeit wird als Differentialgleichungssystem beschrieben. Daher auch der Zusatz „dynamisch“. Dynamisch bedeutet hier, dass die Spieler nicht nur einmal agieren (ihre Bewegungsrichtung ändern), wie es in der statischen Variante sein würde, sondern abhängig vom Anfangszustand mehrmals. Ebenso die zeitliche Abhängigkeit trägt zu diesem Adjektiv bei.
Eine zweite wichtige Komponente des Spiels bildet die Kostenfunktion. Der Jäger möchte die Kosten bis zum Fang minimieren, der Gejagte hingegen maximieren. Um das Verfolgungsspiel zugleich auf diskrete gestörte Kontrollsysteme zu übertragen, obliegt dem einen Spieler dabei die Kontrolle u_k, mit der er seine Bewegungsrichtung im Raum bestimmen kann. Der andere Spieler nimmt die Rolle des Störfaktors ein und bestimmt damit w_k bei jedem Übergang des Systems. Zu jedem Zeitpunkt unterliegen sowohl der Kontrollinput als auch der Störinput den Spielern. Man kann sagen, dass der Übergang von einem Zustand zum nächsten von zwei Individuen beeinflusst wird.

Bemerkung 2.7. *Die Bezeichnung „Störung“ für den Parameter w wird beibehalten, auch wenn er für einen der beiden Spieler als eine Art Kontrolle fungiert.*

Die dynamischen Spiele werden formal gleich als mehrwertige Spiele beschrieben werden, um sie später leichter als Hypergraphen darstellen zu können, die die Basis für die gestörte Variante des Dijkstra-Algorithmus bilden.

Definition 2.8 (**Mehrwertiges Spiel**)**.**
Eine mehrwertige Abbildung $F : X \times U \times W \rightrightarrows X$ bildet einen Zustand $x \in X$ mittels einer Kontrolle $u \in U$ und einer Störung $w \in W$ auf eine kompakte Teilmenge von Zuständen in X ab, wobei $X \subset \mathbb{R}^d$ abgeschlossen ist, $U \subset \mathbb{R}^m$ Kontrollmenge und $W \subset \mathbb{R}^l$ Störmenge genannt werden.
Die zugehörige Kostenfunktion ist definiert als $G : X \times X \times U \times W \to [0, \infty)$.
Ein mehrwertiges Spiel ist das Tupel (F, G).

Das zusätzliche X in der Definition der Kostenfunktion steht lediglich für den Zustand, der beim Übergang des Systems erreicht wird. Man beachte, dass den Kanten nur Kosten aus

$\mathbb{R}_0^+$ zugewiesen werden. Auf diese Weise werden Zykel mit negativen Kosten vermieden, die das System am Erreichen der Zielmenge hindern würden.

Auch hier ist der Begriff der Trajektorie leicht aus der Definition abzuleiten:
Für einen gegebenen Anfangswert $x \in X$, eine gegebene Kontrollfolge $\boldsymbol{u} = (u_k)_{k\in\mathbb{N}} \in U^{\mathbb{N}}$ und eine Störfolge $\boldsymbol{w} = (w_k)_{k\in\mathbb{N}} \in W^{\mathbb{N}}$ ist eine Trajektorie des Spiels gegeben durch eine Sequenz $\boldsymbol{x} = (x_k)_{k\in\mathbb{N}} \in X^{\mathbb{N}}$, so dass

$$\begin{aligned} x_0 &= x \\ x_{k+1} &\in F(x_k, u_k, w_k), \qquad k = 0, 1, \ldots \end{aligned}$$

Die Menge aller Trajektorien von F wird mit

$$\chi_F(x, \boldsymbol{u}, \boldsymbol{w}) = \{(x_k)_k \in X^{\mathbb{N}} |\ x_0 = x,\ x_{k+1} \in F(x_k, u_k, w_k)\ \forall k \in \mathbb{N}\}$$

bezeichnet. Damit ergeben sich folgende zugehörige akkumulierte Kosten:

$$J_{(F,G)}(x, \boldsymbol{u}, \boldsymbol{w}) = \inf_{(x_k)_k \in \chi_F(x,\boldsymbol{u},\boldsymbol{w})} \sum_{k=0}^{\infty} G(x_k, x_{k+1}, u_k, w_k) \tag{2}$$

Im Gegensatz zu den akkumulierten Kosten von gestörten Kontrollsystemen aus Abschnitt 2.1 muss hier noch das Infimum über alle Trajektorien von F gebildet werden. Dieses wird direkt über die Eigenschaft der mehrwertigen Abbildung impliziert. Es kann mehrere Wege vom Startwert zum Endzustand geben und es wird derjenige ausgewählt, der gesamt die niedrigsten Kosten liefert.

Bei einer Open-loop-Strategie beruht die Entscheidung eines Spielers auf der Kenntnis der gesamten Zukunft des Verlaufs des Spiels. Es können keine Feedback-Informationen für künftige Entscheidungen verwendet werden. Eine solche Annahme wäre aber für ein Verfolgungsspiel nicht gerechtfertigt. Die Spieler können die Aktionen des anderen nicht vorausahnen. Darum beschränkt man sich darauf, dass zumindest einem der beiden Spieler ein Informationsvorteil gewährt wird. Zumindest für einen Zeitschritt weiß einer der beiden Spieler, was der andere macht. Der Spieler, dem die Störung w_k zu eigen ist, soll damit die Wahl von u_k im aktuellen Zeitschritt kennen, um auf sie reagieren zu können. Dieser Vorteil soll über eine Funktion gemanagt werden, die die Störung w sowohl in Abhängigkeit vom Parameter u als auch vom Zustand x setzt:

Definition 2.9. *Man definiere sich eine nichtantizipative Strategie $\beta : U^{\mathbb{N}} \times X^{\mathbb{N}} \to W^{\mathbb{N}}$ zu einer gegebenen Kontrollsequenz $u \in U^{\mathbb{N}}$ mit $\beta(x, u) = w$. Die Menge $\mathcal{B}$ enthalte hierzu alle nichtantizipativen Strategien β.*

w wird also über zwei Größen durch den entsprechenden Spieler gewählt. Zum einen ist es entscheidend, was der Gegenspieler macht, also wie dieser u setzt. Zum anderen spielt der Zustand des Systems eine Rolle. Die Wahl von w hängt zusätzlich davon ab, wo sich das System im Raum befindet.

Das Interesse der Arbeit liegt an der Berechnung der Wertefunktion

$$V_{(F,G)}(x) = \sup_{\beta \in \mathcal{B}} \inf_{u \in U^{\mathbb{N}}} J_{(F,G)}(x, u, \beta(u, x)), \tag{3}$$

die zusätzlich zu (2) über u minimiert und die Strategien β maximiert. Nach dynamischen Optimierungsgrundsätzen ist leicht zu sehen, dass sie das Optimalitätsprinzip

$$V_{(F,G)}(x) = \inf_{u \in U} \inf_{x_1 \in F(x,u,w)} \sup_{w \in W} \{G(x, x_1, u, w) + V_{(F,G)}(x_1)\} \tag{4}$$

erfüllt.
Das Optimalitätsprinzip besagt, dass sich jede Optimallösung aus optimalen Teillösungen zusammensetzt. Jede getroffene Entscheidung (Wahl von u und w) ist optimal in Bezug auf den Zustandsübergang von x_0 nach $\mathcal{D}$. Da in den Anwendungsbeispielen ein kompakter Zustandsraum X sowie kompakte Mengen U und W vorausgesetzt werden, werden das Minimum und das Maximum in der Berechnung der optimalen Wertefunktion angenommen werden. Der Wert $V_{(F,G)}(x)$ des Spiels ist kurz gesagt eine Funktion des Anfangszustandes x.

2.4 Diskretisierung des dynamischen Spiels

Da eine Implementierung auf einem kontinuierlichem Zustandsraum nicht möglich ist, muss das mehrwertige Spiel diskretisiert werden. Auf diese Weise wird das Objekt unendlicher Datenmenge zu einem Objekt endlicher Datenmenge, das „nur“ einen endlichen Speicherplatz benötigt, wobei selbst dieser die heute sehr große RAM-Kapazität übersteigen kann.
Der Zustandsraum X wird dazu partitioniert und damit in viele, gleich große Zellen unterteilt (Abb.5).

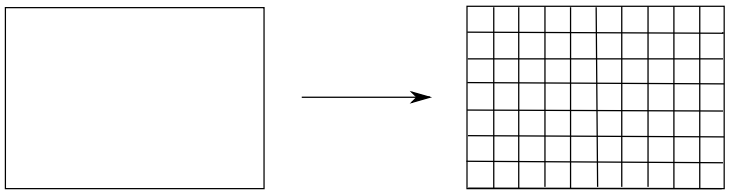


Abbildung 5: Zerlegung des Zustandsraumes

Definition 2.10 (**Zellenüberdeckung**).
Eine Zellenüberdeckung $\mathcal{P} = (P_1, \ldots, P_n)$ *von* X *ist eine endliche Familie abgeschlossener Mengen* P_i, $i = 1, \ldots, n$, $n \in \mathbb{N}$, *mit* $P_i = \overline{\text{int}P_i}$, *so dass* $\text{int}P_i \cap \text{int}P_j = \emptyset$ *für alle* $i \neq j$ *und* $\bigcup_{i=1,\ldots,n} P_i = X$ *gilt. Die Mengen* P_i *heißen Zellen der Zellenüberdeckung.*

Damit jedem Punkt $x \in X$ die Zelle der räumlichen Diskretisierung zugewiesen werden kann, in der x liegt, definiere man die Abbildungen $\pi : X \to \mathcal{P}$ mit $\pi(x) = P$, $x \in P$, und $\rho : X \rightrightarrows X$, $\rho = \pi^{-1} \circ \pi$.
Das Ziel wird im Folgenden sein, die Wertefunktion (4) auf stückweise konstante Funktionen zu projizieren. Für alle Punkte einer Zelle bedeutet das, dass ihnen ein- und derselbe Wert zugewiesen wird. Die Werte im Vergleich zweier Zellen können sich natürlich weiterhin unterscheiden. Hierzu wird das mehrwertige Spiel (F, G) mit

$$F(x, u, w) = \rho(f(x, u, w)) \quad \text{und} \quad G(x, x_1, u, w) = g(x, u)$$

betrachtet, wobei $f(x, u, w)$ und $g(x, u)$ aus der Definition des gestörten Kontrollsystems inklusive dessen Kostenfunktion stammen.
Das Optimalitätsprinzip lautet in diesem Fall

$$\begin{aligned} V_{(F,G)}(x) &= \inf_{u \in U} \inf_{x_1 \in F(x,u,w)} \sup_{w \in W} \{g(x, u) + V_{(F,G)}(x_1)\} \\ &= \inf_{u \in U} \inf_{x_1 \in F(x,u,w)} \{g(x, u) + \sup_{w \in W} V_{(F,G)}(x_1)\} \end{aligned} \tag{5}$$

Für die Projektion wird der Operator der dynamischen Programmierung $L : \mathbb{R}^X \to \mathbb{R}^X$,

$$L[v](x) = \inf_{u \in U} \inf_{x_1 \in F(x,u,w)} \{g(x, u) + \sup_{w \in W} V_{(F,G)}(x_1)\}$$

benötigt, den gerade die rechte Seite von (5) definiert.
Mittels der Projektion $\varphi : \mathbb{R}^X \to \mathbb{R}^{\mathcal{P}}$,

$$\varphi[v](x) = \inf_{x' \in \rho(x)} v(x')$$

wird der diskrete Operator der dynamischen Programmierung $L : \mathbb{R}^{\mathcal{P}} \to \mathbb{R}^{\mathcal{P}}$, $L_{\mathcal{P}} = \varphi \circ L$ definiert.
Genauer:

$$L_{\mathcal{P}}[v](x) = \inf_{x' \in \rho(x), u \in U, x_1 \in F(x', u, w)} \{g(x', u) + \sup_{w \in W} v(x_1)\}$$

Angewendet auf die optimale Wertefunktion ergibt sich dann:

Definition 2.11. *Definiere die diskrete optimale Wertefunktion $V_{\mathcal{P}} \in \mathbb{R}^{\mathcal{P}}$ als einzigen Fixpunkt von $L_{\mathcal{P}}$ mit $V_{\mathcal{P}}(P) = 0 \ \forall P \in \mathcal{P}$ mit $\pi^{-1}(\mathcal{P}) \cap D \neq \emptyset$ (Punkt in Zielmenge). Dann erfüllt $V_{\mathcal{P}}$ das Optimalitätsprinzip*

$$V_{\mathcal{P}}(x) = \inf_{x' \in \rho(x), u \in U, x_1 \in F(x',u,w)} \{g(x', u) + \sup_{w \in W} V_{\mathcal{P}}(x_1)\} \tag{6}$$

2.5 Dynamische Spiele in der Graphentheorie

In Abschnitt 2.4 wurde der Zustandsraum X in Zellen unterteilt. Nun soll für das so entstandene Gitter das Spiel graphentheoretisch formuliert werden.

Der Weg in einem Gitter gemäß Definition 2.6 lässt sich leicht veranschaulichen. In der grünen Zelle befindet sich der Startwert, der mit s gekennzeichnet ist. Über eine Folge von Vorwärtshyperkanten ist er mit dem Punkt $t \in \mathcal{D}$ verbunden. Es muss darauf hingewiesen werden, dass die Zeichnung nicht vollständig ist und auf die weiteren Hyperkanten mit den markierten, grünen Zellen als Anfangspunkt der Kante verzichtet wurde. Zwar sind sie im Hypergraph vorhanden, doch aus Übersichtlichkeitsgründen wurden sie nicht eingezeichnet. Dies muss man berücksichtigen, da aufgrund der Supremumsbildung über w der rot unterlegte Weg samt endlicher Gewichte sonst nicht dem Wert der optimalen Wertefunktion entsprechen würde, der mittels dem Dijkstra-Algorithmus berechnet würde.

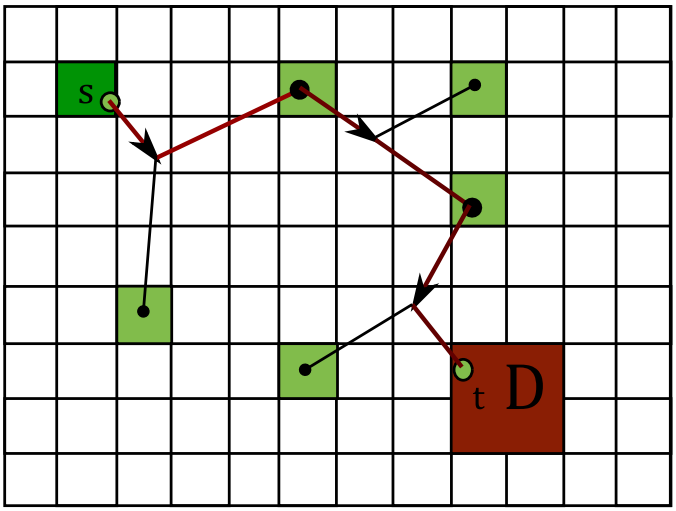
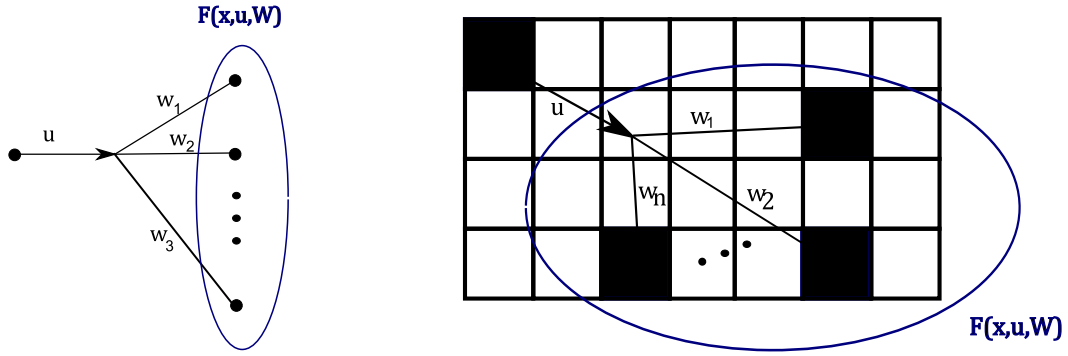


Abbildung 6: Weg im Gitter

Bemerkung 2.12. *Ab jetzt heißen Vorwärtshyperkanten nur noch schlicht Hyperkanten. Auf das Präfix 'Vorwärts' wird verzichtet, da Rückwärtshyperkanten respektive B-arcs nicht mehr vorkommen werden.*
Ebenso wird aus Kompatibilitätsgründen zu der verwendeten Literatur nur noch von Hypergraphen und nicht mehr von F-Graphen gesprochen, auch wenn es sich ausschließlich um letztere handelt.

Um den Bezug von dynamischen Spielen zur Graphentheorie herzustellen, wird auf die mehrwertige Abbildung F aus Definition 2.8 zurückgegriffen. Für jedes Paar $(x, u) \in X \times U$ ist die Menge $F(x, u, W) \subset X$ die Vereinigung einer endlichen Menge von Elementen der Partition $\mathcal{P}$. Das heißt $F(x, u, W)$ impliziert nichts anderes als eine gerichtete Hyperkante (s.Abb.7).



(a) im Hypergraphen (b) im Gitter

Abbildung 7: Implizierte Hyperkante durch F

Auf die Partition $\mathcal{P}$ übertragen wird jedes Element P aus $\mathcal{P}$ auf eine endliche Familie $\{\mathcal{N}_i\}_{i=1,\ldots,i(P)}$, $\mathcal{N}_i \subset \mathcal{P}$, von Teilmengen der Partition abgebildet und zwar unter allen auftretenden Störungen, die sich unter dem Einfluss eines der beiden Spieler befinden. Auf diese Weise ergibt sich ein gerichteter Hypergraph $H = (\mathcal{P}, E)$ mit der Menge von Hyperkanten $E \subset \mathcal{P} \times 2^{\mathcal{P}}$ [1],

$$E = \{(P, \mathcal{N}) |\ \pi(F(x, u, W)) = \mathcal{N} \text{ für einige } (x, u) \in P \times U\}.$$

Als äquivalente mehrwertige Abbildung $\mathcal{F} : \mathcal{P} \rightrightarrows 2^{\mathcal{P}}$ ist die Menge der Hyperkanten definiert durch

$$\mathcal{F}(P) = \{\pi(F(x, u, W)) : (x, u) \in P \times U\}.$$

Der gerichtete Hypergraph wird durch

$$\mathcal{G}(P, \mathcal{N}) = \inf\{g(x, u) : (x, u) \in P \times U, \pi(F(x, u, W)) = \mathcal{N}\}$$

[1] $2^{\mathcal{P}}$ bezeichnet die Potenzmenge von $\mathcal{P}$

noch gewichtet. Das Optimalitätsprinzip lautet damit

$$V_{\mathcal{P}}(P) = \inf_{\mathcal{N} \in \mathcal{F}(P)} \{\mathcal{G}(P, \mathcal{N}) + \sup_{N \in \mathcal{N}} V_{\mathcal{P}}(N)\} \tag{7}$$

Ziel des Spielers P, also des Verfolgers, ist es, seinen Gegner zu fangen, in den meisten Fällen je nach Modellierung schnellstmöglich. Graphentheoretisch wird der Fang in einer Zielmenge $\mathcal{D}$ umgesetzt. Diese Menge beinhaltet für das erste Anwendungsbeispiel „Homicidal Chauffeur Game“ diejenigen Zellen, die innerhalb des Fangradius von P liegen. Im zweiten Anwendungsbeispiel „The Lady In The Lake“ werden es diejenigen sein, die die Flucht der Frau garantieren.
Der für die Mathematik und Informatik fundamentale Algorithmus von Dijkstra, der u.a. in [4] zu finden ist, berechnet für jeden Anfangswert aus X den kürzesten Weg in die Zielmenge $\mathcal{D}$, ganz im Sinne des Spielers P im Spiel „Homicidal Chauffeur Game“. Doch hierbei darf man den zweiten Spieler nicht außer Acht lassen. Dieser möchte den Fang so lange wie möglich hinauszögern beziehungsweise gänzlich verhindern und agiert mittels der Störung w, die ihm obliegt, entsprechend wider seines Verfolgers. Das berücksichtigt der ungestörte Algorithmus nach Dijkstra aber nicht. Darum wird eine gestörte Variante benötigt, die die optimale Wertefunktion $V_{\mathcal{P}}$ in einem gerichteten und gewichteten Hypergraphen berechnet. Von „wahren“ kürzesten Wegen kann hier nicht mehr gesprochen werden, da annahmegemäß beide Spieler eine optimale Strategie verfolgen und durch die Aktionen des Spielers, dem die Störung w unterliegt, der Kürze des Weges entgegengewirkt wird. Ein Vergleich zwischen optimaler und nicht optimaler Strategie des Spielers E bezüglich der Länge des Weges von einem beliebigen Anfangswert bis in die Zielmenge $\mathcal{D}$ hinein wird im ersten Anwendungsbeispiel ausführlich gezogen werden.
Folgende Variante des Algorithmus von Dijkstra wurde der Quelle [1] entnommen:

Algorithmus 2.13 (**Gestörte Variante des Dijkstra-Algorithmus ($(\mathcal{P}$,E$)$,$\mathcal{G}$,$\mathcal{D}$)**).
Eingabe: *Hypergraph H mit der Knotenmenge $\mathcal{P}$ und der Kantenmenge E, Gewichtsfunktion $\mathcal{G}$ und die Menge der Endzustände $\mathcal{D}$*
Arbeitsweise:

1. *Für jedes $P \in \mathcal{P}$ setze $V(P) := \infty$*
2. *Für jedes $P \in \mathcal{D}$ setze $V(P) := 0$*
3. *Definiere die Menge $\mathcal{Q} := \mathcal{P}$*
4. *Solange $\mathcal{Q} \neq \emptyset$*

5. *Bestimme* $P := argmin_{P' \in \mathcal{Q}} V(P')$

6. *Entnimm der Menge* $\mathcal{Q}$ *das minimierende Argument:* $\mathcal{Q} := \mathcal{Q} \backslash \{P\}$

7. *Für jedes Paar* $(Q, \mathcal{N}) \in E$ *mit* $P \in \mathcal{N}$ *unterscheide:*

8. *Falls* $\mathcal{N} \subset \mathcal{P} \backslash \mathcal{Q}$, *dann*

9. *Falls* $V(Q) > \mathcal{G}(Q, \mathcal{N}) + V(P)$, *dann*

10. *Setze* $V(Q) := \mathcal{G}(Q, \mathcal{N}) + V(P)$

Ausgabe: *Optimaler Wert* $V(P)$ *für jede Zelle* $P \in \mathcal{P}$

Falls nach Ausführung des Algorithmus $V(P) = \infty$ für ein $P \in \mathcal{P}$ ist, so existiert kein Weg von $x_0 \in P$ nach der Zielmenge $\mathcal{D}$. Genauer gesagt gibt es eine Strategie β, so dass kein solcher Weg existiert.

An einem kleinen Beispiel soll eine Vertrautheit zum Algorithmus geschaffen werden:

Beispiel 2.14.

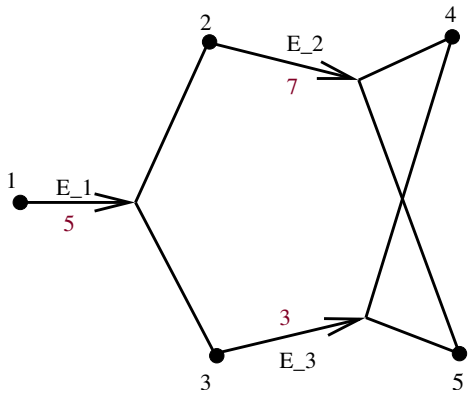


Abbildung 8: Beispiel

Der Algorithmus hat als Eingabe vier entscheidende Größen:

1. Die Knotenmenge $\mathcal{P} = \{1, 2, 3, 4, 5\}$
2. Als die Menge der Endzustände werden hier die Knoten 4 und 5 gewählt, also $\mathcal{D} = \{4, 5\}$
3. Die Menge der Hyperkanten $E = \{E_1, E_2, E_3\}$ mit den Gewichten

4. $\mathcal{G} = \{\mathcal{G}(1,\{2,3\}) = 5, \mathcal{G}(2,\{4,5\}) = 7, \mathcal{G}(3,\{4,5\}) = 3\}$

Schritte 1 und 2 des Algorithmus liefern:

P	1	2	3	4	5
$V(P)$	∞	∞	∞	0	0

Schritt 3: $\mathcal{Q} = \{1,2,3,4,5\}$

Schritte 4 bis 10:

- $P = \{4\}$
  $\mathcal{Q} = \{1,2,3,5\}$
  Betrachte: $(2,\{4,5\}) \in E$ mit $P \in \{4,5\}$

  $\{4,5\} \subset \mathcal{P} \setminus \mathcal{Q} = \{4\}$ nicht erfüllt

  Betrachte: $(3,\{4,5\}) \in E$ mit $P \in \{4,5\}$

  $\{4,5\} \subset \mathcal{P} \setminus \mathcal{Q} = \{4\}$ nicht erfüllt

P	1	2	3	4	5
$V(P)$	∞	∞	∞	0	0

- $P = \{5\}$
  $\mathcal{Q} = \{1,2,3\}$
  Betrachte: $(2,\{4,5\}) \in E$ mit $P \in \{4,5\}$

  $$\begin{array}{rcrcrcl} V(2) & > & \mathcal{G}(2,\{4,5\}) & + & V(5) & \Leftrightarrow & \\ \infty & > & 7 & + & 0 & = & 7 \end{array}$$

  $\Rightarrow V(2) = 7$

  Betrachte: $(3,\{4,5\}) \in E$ mit $P \in \{4,5\}$

  $$\begin{array}{rcrcrcl} V(3) & > & \mathcal{G}(3,\{4,5\}) & + & V(5) & \Leftrightarrow & \\ \infty & > & 3 & + & 0 & = & 3 \end{array}$$

  $\Rightarrow V(3) = 3$

P	1	2	3	4	5
$V(P)$	∞	7	3	0	0

- $P = \{3\}$
  $\mathcal{Q} = \{1, 2\}$
  Betrachte: $(1, \{2, 3\}) \in E$ mit $P \in \{2, 3\}$

  $\{2, 3\} \subset \mathcal{P} \setminus \mathcal{Q} = \{2, 4, 5\}$ nicht erfüllt

P	1	2	3	4	5
$V(P)$	∞	7	3	0	0

- $P = \{2\}$
  $\mathcal{Q} = \{1\}$
  Betrachte: $(1, \{2, 3\}) \in E$ mit $P \in \{2, 3\}$

  $$\begin{array}{rcl} V(1) & > & \mathcal{G}(1, \{2, 3\}) + V(2) \quad \Leftrightarrow \\ \infty & > & 5 \quad + \quad 7 \quad = \quad 12 \end{array}$$

  $\Rightarrow V(1) = 12$

P	1	2	3	4	5
$V(P)$	12	7	3	0	0

- $P = \{1\}$
  $\mathcal{Q} = \{\}$
  Keine Kanten mit $P \in \mathcal{N}$

P	1	2	3	4	5
$V(P)$	12	7	3	0	0

2.6 Implementierung

Eine einzige Zelle der Partition des Zustandsraums allein besteht schon aus unendlich vielen Punkten, dass eine Abbildung dieser mittels der Dynamik $f(x_k, u_k, w_k)$ in endlicher Zeit unmöglich ist. Da der Raum X in sehr viele Zellen zerlegt werden wird, ist es daher vonnöten, eine Auswahl von Punkten zu treffen, die die Entwicklung der Dynamik gut repräsentieren. Doch nicht nur bezüglich X sind solche Testpunkte zu wählen, sondern auch bezüglich des Kontrollraumes U und des Störraumes W. Auf diese Weise wird gewährleistet, dass Algorithmus 2.13 in adäquater Zeit die optimale Wertefunktion und damit auch die Trajektorien des Spiels liefern wird und es überhaupt speichertechnisch möglich ist, diesen Algorithmus am Rechner umzusetzen. Die berechneten Funktionen sind dadurch aber lediglich Approximationen. Die exakte optimale Wertefunktion beziehungsweise die exakten Trajektorien werden durch diesen Vorgang nur angenähert.
Es treten einige Arten von Diskretisierungsfehlern auf. Zum einen durch die zeitliche Diskretisierung, wenn die Differentialgleichung mittels eines Runge-Kutta-Verfahrens diskretisiert wird. Zum anderen durch die räumliche Diskretisierung, wenn der Zustandsraum in Zellen unterteilt wird und aus diesen lediglich Testpunkte entnommen werden. Hinzu kommt noch der gravierendste Fehler, den man bei der Testpunktwahl bezüglich des Kontroll- respektive des Störraumes macht.

Bisher wurde nur erörtert, wie die optimale Wertefunktion dynamischer Spiele anhand der gestörten Variante des Dijkstra-Algorithmus berechnet werden kann. Vielmehr ist man aber daran interessiert, wie dieser nun in der Praxis umgesetzt wird. Hierzu liegt bereits eine Implementierung von M. von Lossow vor [11], die für Zwecke dieser Arbeit teilweise erweitert und auf die Beispiele aus Kapitel 3 und 4 angepasst wurde. Im Folgenden werden eine Reihe von Routinen vorgestellt, die die programmiertechnische Umsetzung von 2.13 im Groben darlegen soll.

2.6.1 Gestörte Variante des Dijkstra-Algorithmus

a) Bisherige Routinen

```
double make_grid(double *xu, double *xo, int dim, int r)
```

Mittels dieser Routine wird eine Zellenüberdeckung $\mathcal{P}$ des zugrunde liegenden Raumes $X \in \mathbb{R}^{\texttt{dim}}$ erzeugt, der nur durch die linke, untere Ecke `xu` und die rechte obere Ecke `xo` definiert ist. Dabei besteht $\mathcal{P}$ aus genau `r` Zellen pro Koordinatenrichtung.

`hgraph * neuer_graph(int anz)`

Speicherprüfung und -bereitstellung für die Hyperkanten und Hyperknoten.

`void verbinde_gitter_graph(hgraph* graph)`

Im Durchlauf der Zellen des Gitters werden sie als Hyperknoten des Graphen `graph` in einem Array gespeichert.

`void erzeuge_hkanten(hgraph* graph,int dim_x,int anzahl_u,double** U, int anzahl_w, double** W,int tp,void f(double *x, double *y, double *u, double *w), double g(double *x, double *u))`

Im Durchlauf der Zellen und deren Testpunkte werden jedem Hyperknoten des Graphen `graph` die zugehörigen Hyperkanten zugewiesen. Es werden aber nur diejenigen berücksichtigt, die garantieren, dass `f(x,u,w)` nicht aus dem Raum X hinausführt.

`void erzeuge_hkanten_min_max_max(hgraph* graph,int dim_x,int anzahl_u, double** U,int anzahl_w,double** W,int tp,void f(double *x, double *y, double *u, double *w), double g(double *x, double *u))`

Der Unterscheid zu `erzeuge_hkanten` liegt darin begründet, dass innerhalb einer Zelle nicht der Testpunkt `tp`, der die niedrigsten Kosten liefert, sondern der, der die höchsten impliziert, in jedem Zeitschritt gewählt wird.

`void dijkstra(hgraph * graph, int anz_D, hknoten** D)`

Für jede Zelle des Zustandsraumes, also jeden Hyperknoten des Hypergraphen `graph`, wird der Wert der diskreten optimalen Wertefunktion berechnet. Dafür wird dem Algorithmus die Menge der Endzustände $\mathcal{D}$ übergeben.

`void ausgabe_wertefunktion(int dim, char* dname)`

Es wird eine Datei `dname` erzeugt, in der neben der linkeren unteren und rechten oberen Ecke einer Zelle auch ihr Wert der optimalen Wertefunktion steht.

b) Erweiterung

Mit dem Ziel, die Verfolgung des einen Spielers durch den anderen nachvollziehen zu können, wurden einige zusätzliche Routinen implementiert. Die Trajektorien des Spiels werden wie folgt berechnet:

Für das Spiel „Homicidal Chauffeur Game“ aus Kapitel 3:

`void trajektorie_hcg_opt(int dim,int anzahl_u, int anzahl_w, double **U, double **W, void f(double *x, double *y, double *u, double *w), double *x, double tol, char* dname)`

> Es wird eine Datei `dname` erzeugt, in der neben der Zustandsfolge $x_k, k = 0, 1, \ldots$, auch die nach dem Optimalitätsprinzip gewählte Kontrollfolge $\boldsymbol{u} = (u_k)_{k\in\mathbb{N}} \in U^{\mathbb{N}}$ des Spielers P und die nach dem Optimalitätsprinzip gewählte Störsequenz $\boldsymbol{w} = (w_k)_{k\in\mathbb{N}} \in W^{\mathbb{N}}$ des Spielers E verzeichnet sind.
> `dim` gibt hierbei an, welche Dimension das zugrundeliegende Differentialgleichungssystem `f` hat. Darüberhinaus werden die Testpunkte aus dem Kontroll- und aus dem Störraum übergeben und ein beliebiger Startzustand `x`, von dem aus die Trajektorie bis in die Zielmenge $\mathcal{D}$ hinein berechnet werden soll. `tol` gibt an, wie viel Toleranz man in etwa gegenüber Diskretisierungsfehlern gewährt, was sich in der Anzahl der zusätzlichen Zellenschichten um die ursprüngliche Endzustandsmenge widerspiegelt.
> Diese Routine beruht auf der Annahme, dass beide Spieler je eine optimale Strategie verfolgen, was heißt, dass der Fang durch Spieler E möglichst lange hinausgezögert wird. Genau diese Eigenschaft unterscheidet sich von den nachfolgenden Implementierungen.

`void trajektorie_hcg_zufall(int dim,int anzahl_u, int anzahl_w, double **U, double **W, void f(double *x, double *y, double *u, double *w), double *x, double tol, char* dname)`

> Im Gegensatz zu `trajektorie_hcg_opt` gibt `trajektorie_hcg_zufall` die Trajektorie zu einem beliebigen Startzustand aus, wenn Spieler E nicht optimal spielt. Die Wahl des Störparameters w wird dabei dem Zufall überlassen, der in einer kleinen Extraroutine `zufall` implementiert ist. Das heißt, dass Spieler E keinen Einfluss mehr auf seine Bewegung hat, sondern sich willkürlich auf dem Gitter fortbewegt.
> Der Routine `int zufall(int anzahl_w)` wird hierbei die Größe `anzahl_w` übergeben, so dass eine Zahl zwischen 0 und `anzahl_w-1` nach bestimmten, mathematischen Grundsätzen erzeugt und an `trajektorie_hcg_zufall` übergeben wird.

```
void trajektorie_hcg_kooperativ(int dim,int anzahl_u,int anzahl_w, double
**U, double **W, void f(double *x, double *y, double *u, double *w),
double *x, double tol, char* dname)
```

Um eine weitere Variante des dynamischen Spiels betrachten zu können, wird in dieser Routine angenommen, dass die Wahl der Störung w durch Spieler E immer zugunsten des Spielers P ausfällt. Die Wahl von u und w erfolgt simultan. Während u dem Optimalitätsprinzip genügt, wird w auf den Wert gesetzt, so dass der Fortschritt in der Bewegung für Spieler P am günstigsten ist; das Tupel (u, w) bildet also das minimierende Argument.

Für das Spiel „The Lady In The Lake“ aus Kapitel 4:

```
void trajektorie_litl_opt(int dim,int anzahl_u, int anzahl_w, double **U,
double **W, void f(double *x, double *y, double *u, double *w), double *x,
double tol, double tol2, char* dname)
```

Völlig analog zu `trajektorie_hcg_opt` bis auf die Übergabe der Toleranzwerte. `tol` und `tol2` grenzen den Fangbereich von der übrigen Randmenge ab.
Beiden Spielern wird insofern wieder ein optimales Strategieverhalten unterstellt, dass Spieler P versucht, Spieler E so lange wie möglich im Wasser verharren zu lassen, während dieser möglichst schnell fliehen möchte.

```
void trajektorie_litl_kooperativ(int dim,int anzahl_u, int anzahl_w, double
**U, double **W, void f(double *x, double *y, double *u, double *w),
double *x, double tol, double tol2,char* dname)
```

Analog zu `trajektorie_hcg_kooperativ` bis auf den Fakt, dass Spieler P nun zugunsten von Spieler E entscheidet, und bis auf die Übergabe der Toleranzwerte. `tol` und `tol2` grenzen auch hier den Fangbereich von der übrigen Randmenge ab.

```
void dijkstra_gewichtet(hgraph * graph, int anz_D, hknoten** D, double *K)
```

Den Zellen aus $\mathcal{D}$ wird im Gegensatz zu Schritt 2 aus 2.13 nicht der Wert 0, sondern Kosten $\neq 0$ über den Vektor `K` zugewiesen.
Ansonsten bleibt der Algorithmus unverändert.

2.6.2 Graphische Darstellung

Als Veranschaulichung der ausgegebenen Trajektorien respektive der optimalen Wertefunktion dienen eigens implementierte Matlab-Routinen. Eine kleine Animation soll die Verfolgung zudem etwas lebendiger darstellen.

wertefunktion.m

3D-Visualisierung der optimalen Wertefunktion mittels der in Matlab vorliegenden Funktion `waterfall`.
Aufruf mit wertefunktion('Filename.dat'), z.B. wertefunktion('hcg_64_3.dat'). Hier wird die Wertefunktion des Spiels „Homicidal Chauffeur Game“ mit 64 Zellen je Koordinatenrichtung und 3 zusätzlichen Zellenschichten um den Nullpunkt graphisch erzeugt. Dazu werden drei Arrays angelegt. X und Y enthalten die Eckkoordinaten der Zellen. Z weist jeder Kante der Zelle einen Wert zu und vergibt entsprechend der Höhe dieses Wertes eine Farbe. Deshalb ist darauf zu achten, dass jeder der vier Kanten einer Zelle derselbe Wert zugeordnet wird.

trajektorie.m

Plot der in den C-Routinen `trajektorie_hcg_opt`, `trajektorie_hcg_zufall` und/oder `trajektorie_hcg_kooperativ` bzw. `trajektorie_litl_opt` und/oder `trajektorie_litl_kooperativ` berechneten Punkte, ausgehend von einem beliebigen Startwert bis hin in die Zielmenge D. Dazu werden die Daten mittels dem Befehl `load` eingelesen und der Aufruf erfolgt mit trajektorie('Filename.dat').

trajektorie2.m

Es handelt sich um eine Erweiterung der Routine `trajektorie.m`, die in der Verbindung der einzelnen Punkte begründet ist. Auf diese Weise soll nachvollzogen werden können, wie die Verfolgung genau vonstatten geht. Umso dichter die Punkte beieinander liegen, umso schwieriger wird es ansonsten, den tatsächlichen Weg bis in D hinein zu erahnen, zumal Diskretisierungsfehler das Aussehen der Trajektorie maßgeblich beeinflussen können.

animation.m

Routine, die `trajektorie.m` animiert, um den Verlauf der Verfolgung „mitansehen“ zu können. Verwendet wird die Option des `movies`. Dazu werden einzelne Frames geplottet und mittels dem Befehl `getframe` abgespeichert. Benötigt werden genau `size(b)` viele, denn so viele Punkte wurden vom Startwert aus bis in die Zielmenge D mittels der Modifikation des Dijkstra-Algorithmus berechnet. Mit `movie(M,1,1)` lässt man den Film, der im Array M abgespeichert ist, 1 Mal ablaufen, wobei 1 Frame pro Sekunde gezeigt wird.
In `set` werden einige Einstellungen getroffen. `(gca,'NextPlot','replacechildren')` übernimmt die aktuellen Achsenabmessungen und behält sie in den folgenden Frames bei.

animation2.m

Animierte Verfolgung von `trajektorie2.m` mit der analogen Erweiterung der Verbindungslinien zwischen den einzelnen Punkten. Es werden zusätzliche Frames erzeugt und abgespeichert, die die Verbindungsstrecke von einem Punkt zu seinem nachfolgenden effektuieren.

2.6.3 Diskretisierungsverfahren

dopri5

Zur zeitlichen Diskretisierung des Systems gewöhnlicher Differentialgleichungen wird ein explizites Runge-Kutta-Verfahren der Ordnung (4)5 benutzt. Nähere Ausführungen finden sich in dopri5.h.

3 Homicidal Chauffeur Game

Das erste Anwendungsbeispiel bildet das unter dem Namen "Homicidal Chauffeur Game" bekannte Differentialspiel. Wie in der Einleitung bereits angedeutet, agieren zwei Spieler und, wie bei einem Verfolgungsspiel üblich, gegeneinander. Als Allegorie des Spiels dient dem ersten Spieler, dem sogannten Pursuer (dt: Verfolger, kurz: P), ein Chauffeur in einem Auto. Sein Ziel ist es, den zweiten Spieler, den sogenannten Evader (dt: Verfolgter, kurz: E), in Form eines Passanten zu überfahren. Ein recht makaberes, aber im Gegenzug sehr illustratives Beispiel für ein Verfolgungsspiel.
Annahmegemäß bewegen sich die Spieler in einem 2-dimensionalen Raum, einer Ebene, mit konstanter Geschwindigkeit. Dem Verfolger, also dem Chauffeur, obliegt die Kontrolle u und dem Verfolgten hingegen die Störung w. Das heißt nichts weiter, als dass beide Spieler eine Möglichkeit besitzen ihre Bewegungsrichtung zu ändern.
Um die optimale Wertefunktion aus Kapitel 2 berechnen zu können, bedarf es einer mathematischen Beschreibung des Spiels. In Abschnitt 3.1. wird ein Differentialgleichungssystem aufgestellt, das anschließend zeitlich und räumlich in 3.2 diskretisiert wird. Simultan findet eine Parameterwahl für die computergestützte Realisierung des Spiels, auf die in 3.3 gesondert eingegangen wird, statt. Unterkapitel 3.4 beinhaltet einige Beispiele für und beobachtbare Erkenntnisse über die optimale Wertefunktion. 3.5 zeigt eine Auswahl an optimalen Trajektorien und ihre Variation bei Strategieänderung auf. In 3.6 wird die exakte Wertefunktion von oben angenähert werden. Es wird also eine obere Schranke für die optimale Wertefunktion geben. Und zum Abschluss wird in 3.7 ein Vergleich mit einer analytischen Untersuchung des Spiels mittels der Isaacs-Gleichung erfolgen.

3.1 Mathematische Beschreibung des Spiels

Die Bewegungen der Spieler werden als Differentialgleichungen beschrieben. Als Ausgangs-Differentialgleichungssystem dient nach [2]:

DGL-System 3.1.

$$\begin{aligned} \dot{x}_{1_p} &= v_1 \sin\Theta_p & \dot{x}_{1_e} &= v_2 \sin\Theta_e \\ \dot{x}_{2_p} &= v_1 \cos\Theta_p & \dot{x}_{2_e} &= v_2 \cos\Theta_e \\ \dot{\Theta}_p &= w_1 u & \dot{\Theta}_e &= w_2 w \end{aligned}$$

mit

(i) v_1 *und* v_2 *sind die konstanten Geschwindigkeiten von P und E*

(ii) w_1 *bzw.* w_2 *sind die maximalen Winkelgeschwindigkeiten von P bzw. E*

(iii) Θ *ist der Winkel der Bewegungsrichtung*

(iv) Kontrolle $|u| \leq 1$*, Störung* $|w| \leq 1$

Da das System aufgrund seiner hohen Dimension und der zusätzlichen Differentialgleichung bezüglich des Winkels Θ nicht unbedingt praktikabel ist, wird eine geeignete Koordinatentransformation mittels

$$\begin{aligned} x_1 &= (x_{1_e} - x_{1_p}) \cos\Theta_p - (x_{2_e} - x_{2_p}) \sin\Theta_p \\ x_2 &= (x_{1_e} - x_{1_p}) \sin\Theta_p + (x_{2_e} - x_{2_p}) \cos\Theta_p \\ \Theta &= \Theta_e - \Theta_p \end{aligned}$$

durchgeführt. Dadurch erhält man ein Differentialgleichungssystem in Relativkoordinaten. Das bedeutet hier, dass der Nullpunkt des zugrunde liegenden Raumes stets als Position für den Pursuer dient. Das Koordinatensystem ist also nicht fix.

DGL-System 3.2.

$$\begin{aligned} \dot{x}_1 &= -w_1 u x_2 + v_2 \sin\Theta \\ \dot{x}_2 &= -v_1 + w_1 u x_1 + v_2 \cos\Theta \\ \dot{\Theta} &= w_2 w - w_1 u \end{aligned}$$

Die Tatsache, dass es sich bei Spieler E um einen Passanten handelt, impliziert, dass er seine Richtung beliebig scharf ändern kann. Die Winkelgeschwindigkeit w_2 ist damit beliebig groß wählbar. Daher kontrolliert Spieler E den Winkel Θ gänzlich, weshalb Θ zur neuen Störung w wird. Das 3-dimensionale System wird zu einem 2-dimensionalen. Das System wird nach einer Normierung der Geschwindigkeiten v_1 und w_1 jeweils auf 1 zu

DGL-System 3.3.

$$\begin{aligned} \dot{x}_1 &= -u x_2 + v_2 \sin w \\ \dot{x}_2 &= -1 + u x_1 + v_2 \cos w \end{aligned} \tag{8}$$

mit

(i) $|u| \leq 1$

(ii) w *frei*

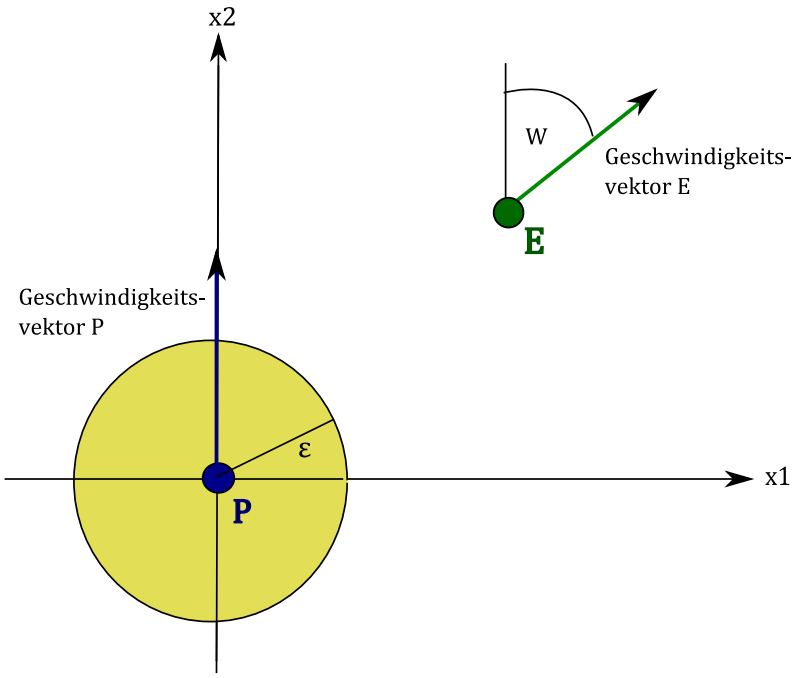


Abbildung 9: Homicidal Chauffeur Game

Abbildung 9 soll das Spiel veranschaulichen. Man beachte, dass der Winkel w im Uhrzeigersinn gemessen wird, was R. Isaacs als besondere Eigenschaft eingeführt hat.

Es liegt nur noch eine Restriktion, nämlich $|u| \leq 1$, vor. Und diese ist leicht einzusehen: Spieler P, der Chauffeur, ist der Bürde des Wendekreises seines Autos unterlegen und kann daher keine scharfe Drehung vollführen, weshalb u beschränkt sein muss. Der Fußgänger hingegen ist in der Lage, seine eingeschlagene Richtung abrupt zu ändern, weshalb die Störung w frei wählbar ist.

3.2 Diskretisierung

Die Diskretisierung des Spiels (8) erfolgt auf verschiedenen Ebenen. Die zeitliche Diskretisierung des Differentialgleichungssystems erledigt das in Abschnitt 2.6.3 erwähnte Einschrittverfahren. Für die räumliche Diskretisierung hingegen sind ein paar mehr Gedankengänge erforderlich. Bei der Wahl eines passenden Zustandsraums ist unbedingt darauf zu achten, dass der Endzustand des Spiels – der Fang! – innerhalb einer Zelle des Gitters über dem Raum liegt. Aufgrund von Diskretisierungsfehlern kann die Zuordnung von $f(x_k, u_k, w_k)$ – man fasse die Diskretisierung von (8) als $f(x, u, w)$ auf – zur Zelle schwierig werden, wenn der Zustand in der Nähe oder auf einer Kante liegt. Wenigstens für grobe Gitter kann auf die Lage des Nullpunkts geachtet werden.

Der zugehörige Kontrollraum U wird als das Intervall $[-1, 1]$ und der zugehörige Störraum W als das Intervall $[-\pi, \pi]$ gewählt. Die Wahl des Kontrollraumes ist sofort einsichtig, da die Restriktion $|u| \leq 1$ erfüllt werden muss. Die Störung w ist hingegen unbeschränkt und wird hier als Winkel aufgefasst. Mit dem Intervall $[-\pi, \pi]$ ist jede erdenkliche Drehung des Passanten gewährleistet und damit seine uneingeschränkte Handlungsfreiheit ihrbezüglich, wobei $-\pi$ und π dieselbe Bewegungsrichtung implizieren. Als Kontroll- bzw. Störparameter wählt man sich repräsentative Werte aus den jeweiligen Intervallen. Für den Chauffeur zum Beispiel bedeutet $u = -1$ eine Bewegung nach links und $u = 1$ eine nach rechts. Diese beiden Werte sollten damit auf jeden Fall in der Testpunktwahl enthalten sein. Durch die Diskretisierung der Menge U wird die ursprünglich messbare Kontrollfunktion $\boldsymbol{u}$ zu einer stückweise konstanten Funktion. Ebenso verhält es sich für die Kontrollfunktion $\boldsymbol{w}$ durch die Diskretisierung von W. Die auf diese Weise resultierenden Fehler in den Berechnungen der optimalen Wertefunktion sind die schwerwiegendsten aller auftretenden Diskretisierungsfehler.
Im Homicidal Chauffeur Game ist die Menge der Endzustände $\mathcal{D}$ nichts anderes als der Nullpunkt plus etwaigen erforderliche Zellenschichten um diesen herum. Der Nullpunkt aus dem Grund, weil das Differentialgleichungssystem (8) in Relativkoordinaten beschrieben ist und sich das Auto immer in diesem Punkt befindet, der Fang also unweigerlich dort erfolgt. Ab einer gewissen Feinheit des Gitters sind zusätzliche Zellenschichten um den Nullpunkt erforderlich, da die einzelnen Zellen sehr klein sind und Diskretisierungsfehler die Steuerung des Systems in die Null in den seltensten Fällen, wenn überhaupt, zulassen. Darum muss gewährleistet werden, dass auch Zustände nahe der Null als Endzustände fungieren dürfen.
In dieser Arbeit wird als Zustandsraum $X = [-5.4, 5.0] \times [-5.4, 5.0]$ gewählt. Aus Sicht des Autos könnte man sich das als freie Fläche, die 540 Meter nach links und nach hinten und 500 Meter nach vorne und nach rechts führt, vorstellen. Von Zustandsraumbeschränkungen wie Häuser oder Bäume wird hier abgesehen. Es treten also keine Hindernisse auf, die es zu umfahren bzw. zu umgehen gilt.

Insgesamt:

(i) $X = [-5.4, 5.0] \times [-5.4, 5.0]$

(ii) $u \in U = [-1, 1]$

(iii) $w \in W = [-\pi, \pi]$

(iv) $\mathcal{D}$ besteht aus dem Nullpunkt inklusive einer kleinen variablen Umgebung

Eine entscheidende Größe fehlt bislang aber noch: Die Kostenfunktion $g(x, u)$! Instinktiverweise möchte Spieler P den Spieler E möglichst schnell fangen, in dem Beispiel überfahren, und letzterer möchte dies so lange wie möglich verhindern. Daher bietet sich als Kostenfunktion $g(x, u) = h$ an, da zum einen h nichts anderes als der Zeitschritt der Diskretisierung ist und zum anderen auf diese Weise die Zeit bis zum Fang gemessen werden kann. Somit reiht sich das Spiel in die Serie der Ordnungsspiele ein. Es wird nicht danach gefragt, ob der Fang erfolgt oder nicht (Gattungsspiel), sondern die Zeit bis zum Fang liegt im Fokus des Interesses. Die benötigte Zeit entspricht dem Wert des Spiels und lässt sich über das Streben nach einer Minimierung (Spieler P) und einer Maximierung (Spieler E) dieser Größe berechnen.

3.3 Umsetzung als Programm in C

Die Umsetzung des Beispiels als Programm erfordert zunächst eine ganze Reihe von Definitionen. Während im Hauptprogramm sämtliche Parameter gesetzt werden, werden das Differentialgleichungssystem, die Kostenfunktion und der Aufruf des Differentialgleichungslösers als Funktionen ausgelagert.
Durch den Aufruf von

- make_grid
- neuer_graph
- verbinde_gitter_graph
- erzeuge_hkanten

wird sowohl das Gitter angelegt als auch der Hypergraph erzeugt.
Nachdem im Anschluss noch die Zielmenge $\mathcal{D}$ bestimmt wurde, wird die gestörte Variante des Dijkstra-Algorithmus aufgerufen. Nun bestehen die Optionen, sich die approximierte Wertefunktion mit `ausgabe_wertefunktion` und/oder die Trajektorien mit Hilfe einer der Routinen `trajektorie_hcg_opt`, `trajektorie_hcg_zufall` oder `trajektorie_hcg_kooperativ` zu beliebigen Startwerten ausgeben zu lassen.

3.4 Wertefunktion

Wie bereits erwähnt, kann mittels Algorithmus 2.13 nur eine Approximation der optimalen Wertefunktion berechnet werden. Da 2.13 immer ein $\min_u \max_w$-Problem behandelt, wird beiden Spielern ein optimales Strategieverhalten unterstellt. In jedem Schritt der Diskretisierung wird sowohl ein für Spieler P optimales u als auch ein für Spieler E optimales

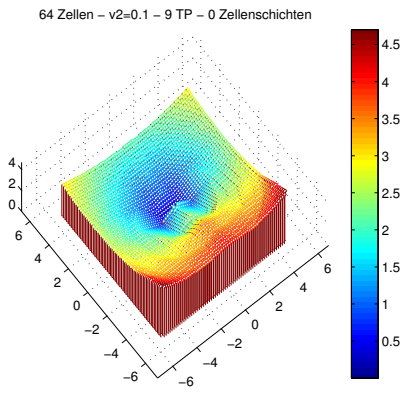
w bestimmt, die in ihrer Folge $\boldsymbol{u} = (u_k)_{k\in\mathbb{N}}$, $\boldsymbol{w} = (w_k)_{k\in\mathbb{N}}$ den Weg vom Startwert x_0 bis in die Zielmenge $\mathcal{D}$ festlegen.
Über

$$\min_{u\in U}\max_{w\in W}\{g(x_i,u) + V(f(x_i,u,w))\} \tag{9}$$

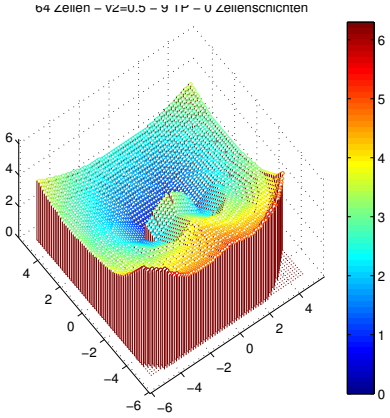
berechnet sich die optimale Kontrolle $\tilde{u}$ für Spieler P und die optimale Störung $\tilde{w}$ für Spieler E im $(i+1)$-ten Diskretisierungsschritt.
Der Zustand x_{i+1} lässt sich also durch $f(x_i, \tilde{u}, \tilde{w})$ berechnen.
Da die Kostenfunktion g für dieses Beispiel als konstant angenommen wird und zudem nicht von u abhängig ist, ist sie in (9) in der Bestimmung von $\tilde{u}$ respektive $\tilde{w}$ eigentlich nicht von Belang.

Bei näherer Betrachtung des Differentialgleichungssystems DGL-System 3.3 fällt auf, dass der Parameter v_2 nicht explizit festgelegt ist, sondern lediglich kleiner als 1 sein muss. Zudem hat man weitere Freiheiten inne, die den Wert der optimalen Wertefunktion beeinflussen können, aber nichts am optimalen Strategieverhalten der beiden Spieler ändern. Die Zellenanzahl pro Koordinatenrichtung spielt unter anderem eine Rolle. Je feiner das Gitter ist, desto aussagekräftiger ist das Ergebnis für die optimale Wertefunktion, da die Approximationsgüte mit zunehmender Zellenanzahl steigt. Bezüglich der Güte der Approximation sei auf das Kapitel „Konvergenzanalyse“ aus [1] verwiesen. Dort wird gezeigt, dass die numerisch berechnete optimale Wertefunktion gegen die exakte optimale Wertefunktion für $d \to 0$ konvergiert, wobei d der Diameter einer Zelle des Gitters ist. Zwar wird in den Beweisen des Artikels mit einer anderen nichtantizipierenden Strategie β, $\beta : U \to W$, und einem von Beginn an diskretem Kontrollsystem gearbeitet, aber wie die nachfolgenden Berechnungen zeigen sollen, ändert sich an dem Fakt der Konvergenz nichts, nur der Formalismus ist ein anderer. Allerdings wird hier keine genaue und formal korrekte Analyse erfolgen.

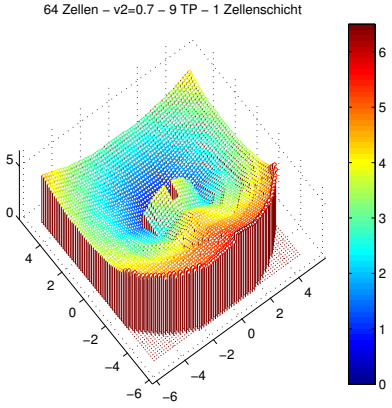
Ein paar Beispiele sollen den Einfluss einzelner Parameter veranschaulichen. v_2 ist die konstante Geschwindigkeit, mit der sich der Passant fortbewegt. Darum sollte der Wert der optimalen Wertefunktion kleiner werden, je langsamer Spieler E, der Passant, ist. Ergo, der Fang sollte schneller erfolgen. Für die Berechnungen wurden aus der Kontrollmenge 11 Testpunkte äquidistant entnommen, d.h. $-1, -0.8, \ldots, 0.8$ und 1. Aus der Störmenge hingegen nur 10, d.h. $-\pi, -0.8\pi, \ldots, 0.6\pi$ und 0.8π, da sich $-\pi$ und π in der Bewegungsrichtung des Spielers entsprechen.



(a) $v_2 = 0.1$



(b) $v_2 = 0.5$



(c) $v_2 = 0.7$

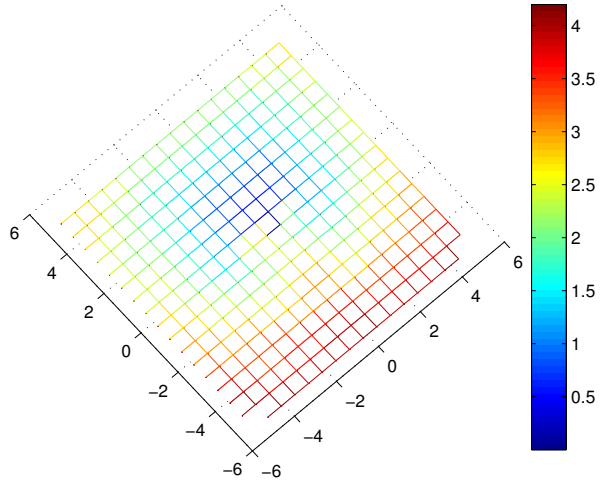
Abbildung 10: Optimale Wertefunktion, 64 Zellen je Richtung

Die Farbskalen in Abbildung 10 geben den Zuwachs der Werte mit höherer Geschwindigkeit des Fußgängers schön wieder. Bei $v_2 = 0.1$ beläuft sich der größte Wert noch auf 4.7, bei $v_2 = 0.7$ im Gegenzug auf 6.5. Der Fang wird also zeitlich hinausgezögert, je schneller sich der Passant fortbewegen kann.

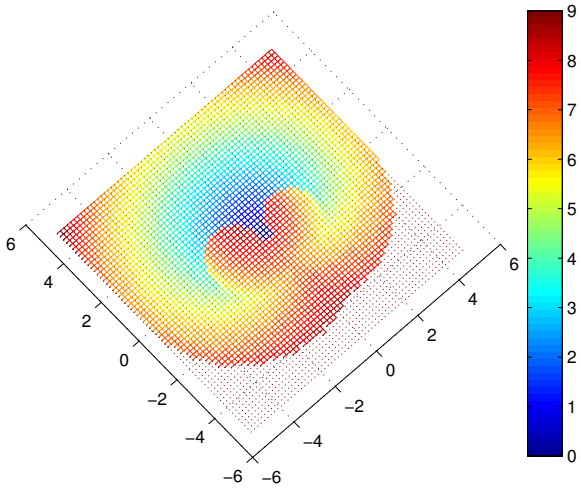
Ein Resultat, das nicht verwundern sollte. Aber eine interessante Eigenschaft spiegelt Abbildung 10 dennoch wider. Betrachtet man sich vor allem Figur (c) genauer, so erkennt man, dass nicht alle Zellen des zugrunde liegenden Raumes, das Quadrat $X = [-5.4, 5.0] \times [-5.4, 5.0]$, mit einem Wert belegt worden sind. Diese Zellen weisen bei der Berechnung der optimalen Wertefunktion durch die gestörte Variante des Algorithmus von Dijkstra den Wert „unendlich“ auf. Dieses Phänomen lässt sich ganz leicht erklären, da bereits in Abschnitt 2.5 erwähnt wurde, dass der Wert dann auf „unendlich“ gesetzt wird, wenn durch die Existenz einer geeigneten nichtantizipativen Strategie β kein Weg von x_0 nach $\mathcal{D}$ existiert. Bei der Erstellung des Hypergraphen wird außerdem auf all jene

Kanten verzichtet, die $f(x, u, w)$ aus dem Zustandsraum X hinausführen. Gibt es nach dem Optimalitätsprinzip also keine Kante, die das System von einem beliebigen Startwert x_0 aus auf dem Weg nach $\mathcal{D}$ nicht aus X hinausführt, so bleibt der Wert für die zugehörige Zelle (die Zelle, in der der Startwert liegt) aus Schritt 1 des Algorithmus auf unendlich und der gewünschte Fang des Passanten erfolgt nie. Je schneller der Passant ist, desto mehr Zellen sind von diesem Phänomen betroffen.

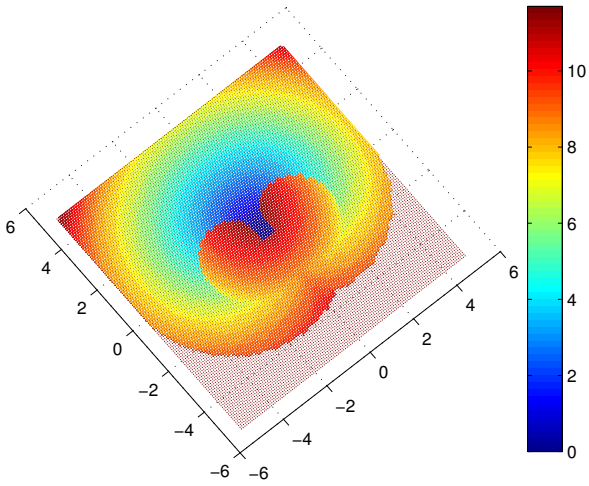
Obwohl keine formal korrekte Konvergenzanalyse folgen wird, so ist es dennoch interessant zu sehen, was passiert, wenn das Gitter immer feiner gewählt wird (Abb.11). Die Struktur der optimalen Wertefunktion wird mit jeder Verdopplung der Zellenanzahl je Koordinatenrichtung feiner und der Rand (die Zellen, für die gerade noch ein Wert $\neq \infty$ angenommen wird) wird immer glatter. Die Fortbewegung des Chauffeurs, vor allem ein Wendemanöver, wird in der Realität auch glatt bzw. rund verlaufen und keine „Ecken“ aufweisen. Wie man in 3.7 sehen wird, findet man zudem die Eigenschaften einer analytischen Untersuchung des Spiels wieder.



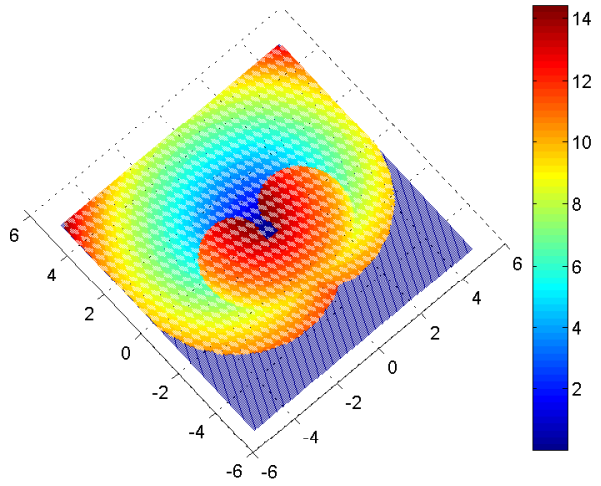
a) 16 Zellen



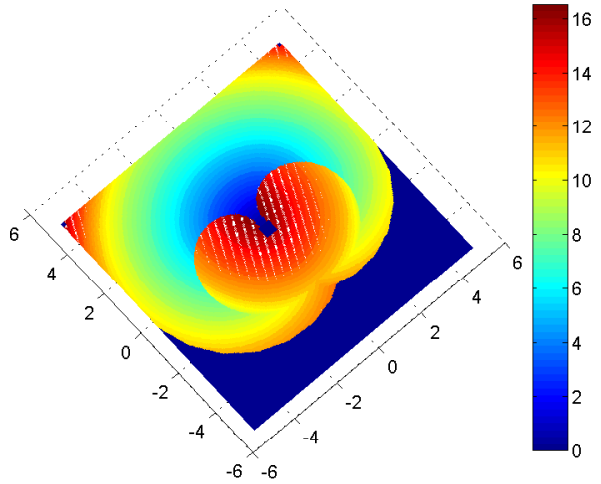
b) 64 Zellen



c) 128 Zellen



d) 256 Zellen



e) 512 Zellen

Abbildung 11: Wertefunktion mit zunehmender Zellenanzahl

Von den dunkelblauen Bereichen in Abbildung 11 d) und e) sollte man sich nicht beirren lassen. Ab einer gewissen Gitterfeinheit werden alle Zellen, die den Wert ∞ aufweisen (diese Werte stehen in der Textdatei als Ascii-Code „inf“), mit dem Wert 0 geplottet und beim Exportieren als eps-File durch die entsprechende Farbe (resultierend aus der Farbskala) belegt. Dies ist nicht beabsichtigt und resultiert aus der verwendeten Matlab-Routine.

3.5 Trajektorien und Strategienvergleich

3.5.1 Trajektorien des Spiels

Die Approximation der Wertefunktion ist nicht das einzige, was der Dijkstra-Algorithmus 2.13 als Ergebnis liefert. Mittels einer kleinen Routine kann man sich simultan den Weg, der zum Wert der Annäherung an die optimale Wertefunktion gehört, vom Startwert bis in die Zielmenge, ausgeben lassen. Die Länge des Weges respektive der optimalen Trajektorie bestimmt ohnehin durch die Gewichtung der Kanten den Wert der optimalen Wertefunktion.
In den Abschnitten 2.6.1 b) und 2.6.2 sind bereits die Routinen vorgestellt worden, mit denen die Trajektorien erhalten werden können. Im Folgenden soll ein ausführlicher Vergleich zwischen einem optimalen und einem nicht optimalen Verhalten des Spielers E, dem Passanten, gezogen werden. Was passiert, wenn er seine optimale Strategie plötzlich ablegt? Was ist, wenn er zugunsten seines Widersachers entscheidet und damit seinen freiwilligen Tod in Kauf nimmt?
Die Antworten liegen nahe und werden in Abschnitt 3.5.2 näher diskutiert werden.

Bemerkung 3.4. *Zur Berechnung der Trajektorien wurde aus Übersichtlichkeitsgründen ein recht grobes Gitter gewählt. Jeder berechnete Punkt im Gitter wird durch einen kleinen Kreis dargestellt und bei einem sehr feinen Gitter würde man letztendlich aufgrund der teilweisen Überlagerung der Kreise nichts mehr erkennen. 64 Zellen pro Koordinatenrichtung reichen außerdem aus, um den Effekt einer Strategienänderung kenntlich zu machen, zumal sich die Trajektorien für verschiedene Gitterfeinheiten nur unwesentlich unterscheiden.*
Als Zeitschritt wurde h auf den Wert 0.5 *gesetzt.*

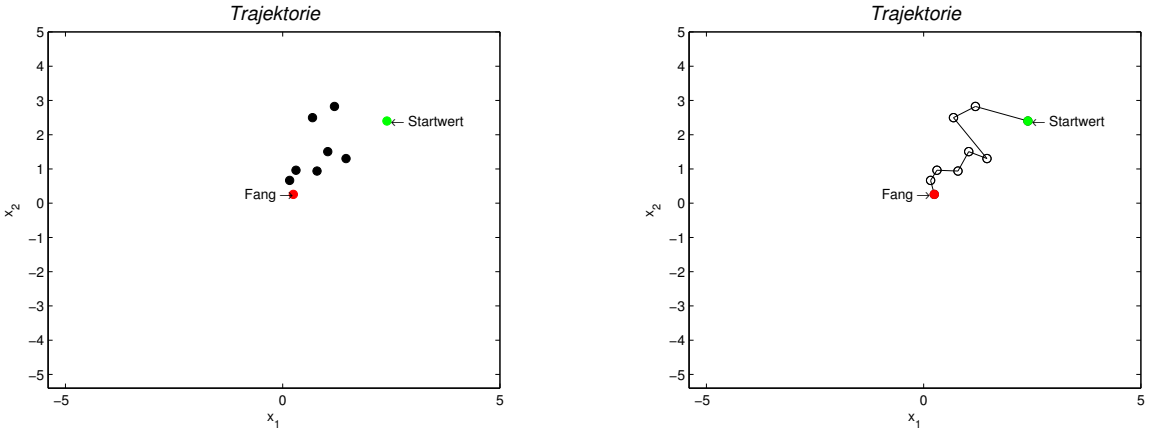
Zuerst werden einige Trajektorien betrachtet, die ein optimales Strategieverhalten beider Spieler unterstellen. Die Startwerte wurden so gewählt, dass alle möglichen Szenarien der Position von Chauffeur zu Passant durchgespielt werden. Hier gibt es drei grundsätzliche Typen zu unterscheiden:

(i) Der Fußgänger steht zu Beginn in der Front des Autos und hat daher keine Chance zu fliehen.

(ii) Der Fußgänger steht am Anfang an der Seite von Spieler P und nahe genug, um für den Moment in dessen Wendekreis zu flüchten.

(iii) Der Fußgänger steht zu Beginn hinter dem Chauffeur ($x_2 < 0$) und es gibt damit keine Möglichkeit zum sofortigen Fang.

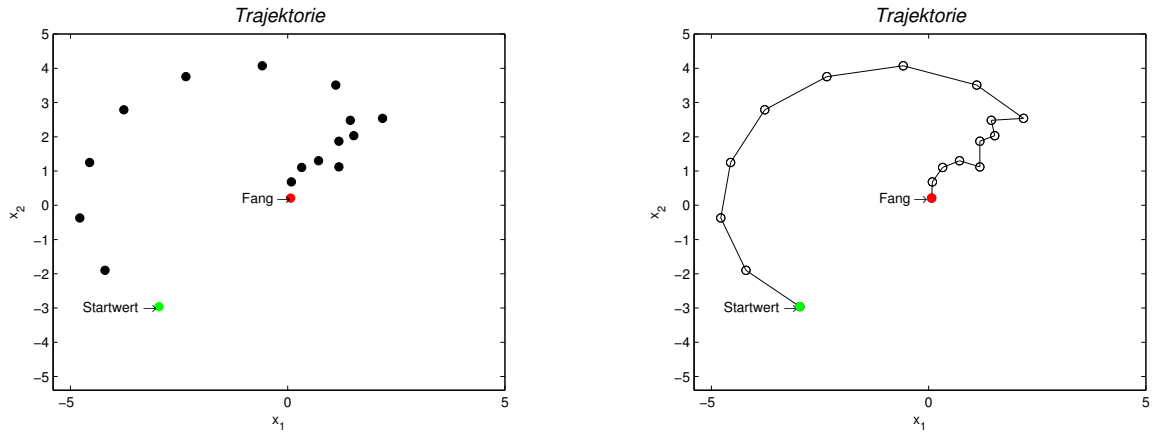
Der Annahme, dass der Passant zu Beginn in einiger Entfernung vor dem Auto steht, wird beispielsweise der Startwert (2.4,2.4) gerecht. Das Auto verfolgt seinen Gegner unmittelbar! Für die Darstellung des Ergebnisses werden zwei Arten gewählt. Eine veranschaulicht lediglich die Punkte, die während des Algorithmus 2.13 berechnet werden. Die andere spiegelt zusätzlich die Reihenfolge wider, in der das System in die Zielmenge $\mathcal{D}$ gesteuert wird.



(a) unverbunden (b) verbunden

Abbildung 12: Optimale Strategie I, Startwert (2.4,2.4)

Während Abbildung 12 Typ (i) wiedergibt, zeigt Abbildung 13 den Verlauf des Spiels, wenn eine Mischung aus Typ (ii) und (iii) den Startzustand bestimmt. Steht der Passant also seitlich hinter dem Chauffeur, so muss dieser erst eine Kurve fahren, ehe es ersteren direkt verfolgen kann. Der Bogen zu Beginn der Trajektorie ist ohnehin typisch für dieses Spiel. Der Chauffeur wird grundsätzlich zu Beginn seine Richtung ändern müssen, um eine direkte Verfolgung des Passanten aufnehmen zu können. Und da er am Steuer eines Autos sitzt, wird er diese Richtungsänderung in Form einer Rechts- oder einer Linkskurve beschreiben müssen.



(a) unverbunden (b) verbunden

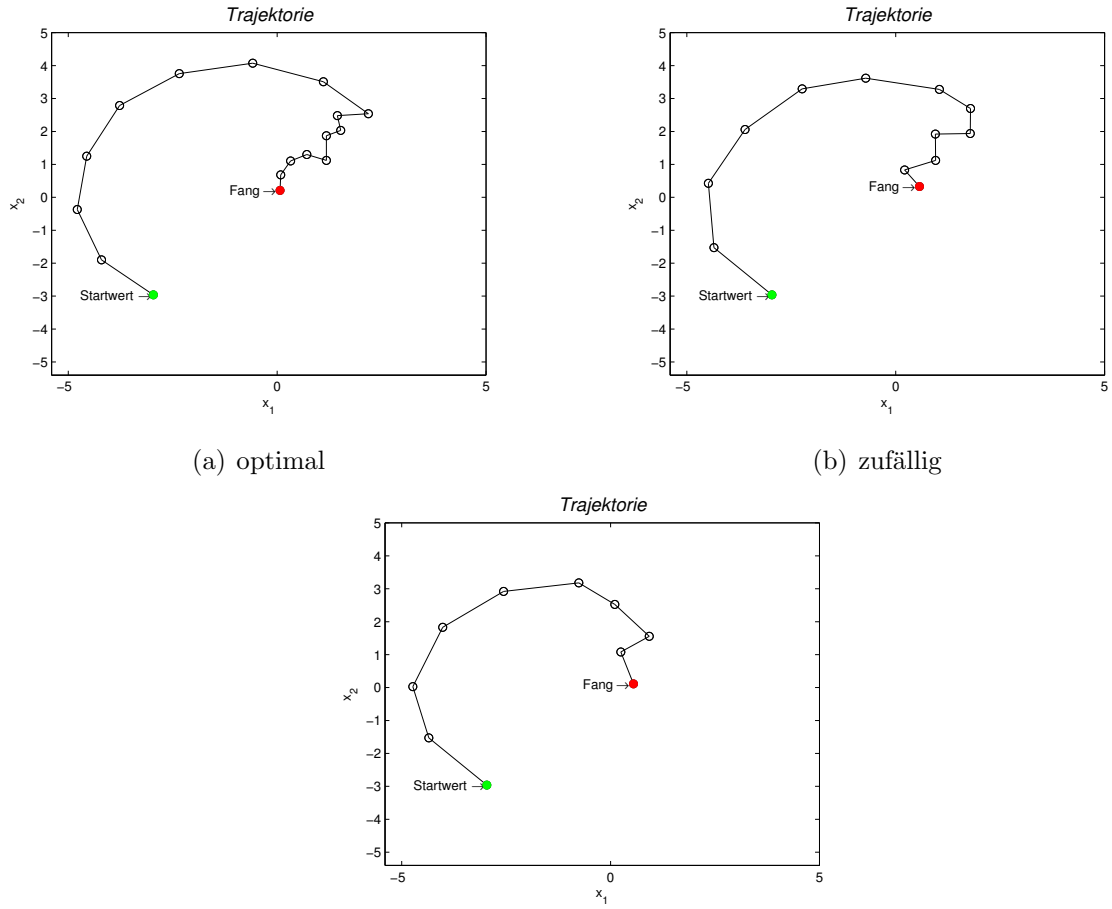
Abbildung 13: Optimale Strategie II, Startwert (-2.9625,-2.9625)

Schon jetzt kann leicht beobachtet werden, dass sich Diskretisierungsfehler zum Ende hin stärker auswirken. Die Punkte in den Abbildungen liegen „zickzackförmig“ und weichen daher von der exakten Trajektorie stärker ab. Der Bogen, den das System zu Beginn beschreibt, ist global gesehen für alle Berechnungen gleich, nur gerade diese Struktur des „Zickzacks“ ist abhängig von der zugrundeliegenden Diskretisierung.
Ein weiteres Merkmal für den Einsatz numerischer Mathematik ist die Position des Punktes, in dem der Fang erfolgt. Wenn man einzig von der Tatsache ausgehen würde, dass der Berechnung ein relatives Koordinatensystem zugrunde liegt – der Ursprung ist immer Position des Chauffeurs, so dürfte nur der Ursprung selbst als Fangpunkt dienen. Der gestörten Variante des Dijkstra-Algorithmus wird jedoch eine größere Zielmenge $\mathcal{D}$ übergeben, die man auch als Fangradius des mordlustigen Chauffeurs interpretieren könnte. Der eigentliche Grund für die Expansion von $\mathcal{D}$ ist aber, dass das System 3.3 aufgrund von gewissen Fehlern in der Berechnung als Folge der Diskretisierung nicht unbedingt direkt in die 0 gesteuert wird, sondern lediglich in ihre unmittelbare Umgebung.

3.5.2 Strategienwahl und ihre Folgen

Zwar wird der Passant wohl kaum den Drang verspüren, gegen seinen Gegner nicht optimal zu spielen, aber ihm werden nun dennoch zum Zweck der Forschung abweichende Strategievarianten unterstellt. Was passiert also, wenn der Fußgänger seine ihm obliegende Störung w eher zufällig wählt?
Er könnte eine Richtung einschlagen, die ihn direkt in die Arme seines Widersachers treibt oder einfach nur „halbherzig und wahllos“ davonlaufen, um den Fang vielleicht doch noch hinauszuzögern. Von dem Konjunktiv „könnte“ darf eigentlich nicht die Rede sein, denn die Wahrscheinlichkeit, dass per Zufall in jedem Diskretisierungsschritt immer das nach dem Optimalitätsprinzip bestimmte $\tilde{w}$ gewählt wird, ist verschwindend gering. Egal, wie die Zufallsfolge $\boldsymbol{w} = (w_k)_{k\in\mathbb{N}} \in W^{\mathbb{N}}$ daher aussehen mag, der Passant wird bis auf den Fall $w_k = \tilde{w}_k \; \forall k \in \mathbb{N}$ grundsätzlich zu einem früheren Zeitpunkt von dem Chauffeur überfahren werden. Abbildung 14 macht deutlich, wie sich die Trajektorie entsprechend verändert. Sie wird eindeutig kürzer und insgesamt werden in dem gewählten Beispiel mit Startwert (-2.9625,-2.9625) nur 12 Übergänge des Systems 3.3 bis zum Fang gefordert, während es im optimalen Fall noch 16 waren. Der Unterschied mag unbedeutend erscheinen, doch hierzu muss darauf hingewiesen werden, dass die Strategien für andere Startwerte enorm voneinander abweichen können.

Die Strategie, per Zufall zu entscheiden, welchen Wert w annimmt, führt auf eine ganz simple Strategie: Der Passant kann auch immer genau die Störung wählen, die zugunsten des Spielers P ausfällt. Das würde bedeuten, dass er das w auswählt, das als minimierendes Argument im Optimalitätsprinzip auftritt. Dazu müsste Spieler E allerdings einen Informationsvorteil besitzen und zwar in der Form, dass er weiß, wie sein Gegenspieler agiert, damit er sich danach richten kann. Diesen hat er aber aufgrund seiner durch β festgelegten Strategie ohnehin inne.
Der Weg bis zum Fang wird nun im wahren Sinne des Wortes minimiert. Es werden für den Startwert (-2.9625,-2.9625) nur noch 9 Übergänge des Systems benötigt, um die Zielmenge $\mathcal{D}$, und damit den Fang, zu erreichen.



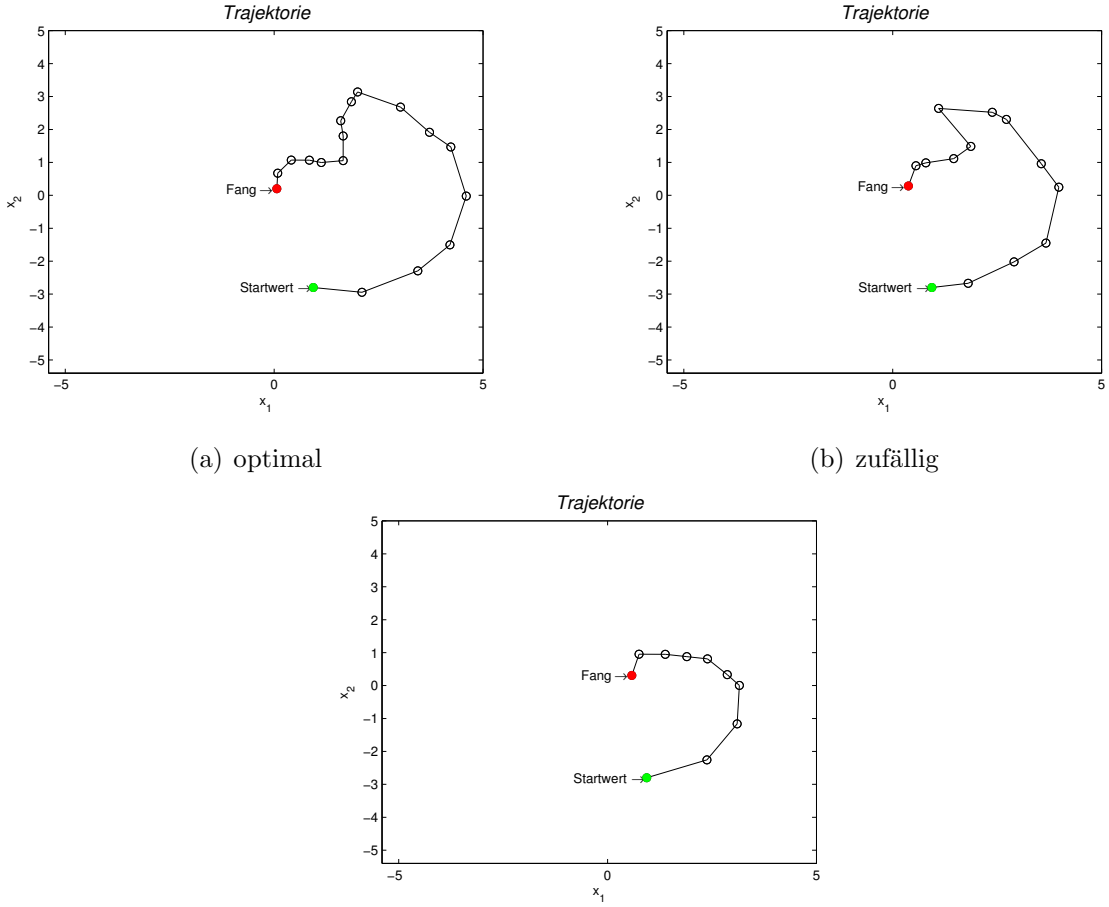
(a) optimal

(b) zufällig

(c) kooperativ

Abbildung 14: Strategiengegenüberstellung, 64 Zellen pro Richtung I

Wie bereits angedeutet, kann sich die Strategienwahl enorm auf das Aussehen der Trajektorie auswirken. Unter anderem belegt der Startwert (0.9375,-2.8) diese Aussage. Abbildung 15 zeigt, wie wichtig die Strategienwahl für Spieler E ist. Während die Trajektorie im optimalen Fall eine lange Flucht garantiert, wird der Passant im einseitig kooperativen Fall sehr schnell überfahren. Die Strategie des Fußgängers, immer direkt auf das Auto zuzulaufen, wirkt sich überaus positiv auf das Ziel des Chauffeurs, den Fang – und damit den Mord, um bei der ursprünglichen Versinnbildlichung des Spiels zu bleiben – zu erzwingen, aus. Sobald sich Spieler E zu Beginn des Spiels direkt im Rücken des Chauffeurs befindet, machen sich die Konsequenzen der Strategienwahl besonders bemerkbar.



(a) optimal

(b) zufällig

(c) kooperativ

Abbildung 15: Strategiengegenüberstellung, 64 Zellen pro Richtung II

3.6 Approximation der Wertefunktion von oben

Bisher wurde in der Berechnung der optimalen Wertefunktion über die Testpunkte einer Zelle minimiert, vgl. (6):

$$V_{\mathcal{P}}(x) = \inf_{x' \in \rho(x), u \in U, x_1 \in F(x',u,W)} \{g(x',u) + \sup_{w \in W} V_{\mathcal{P}}(x_1)\}.$$

Damit nähert man die exakte optimale Wertefunktion von unten her an und erhält gleichzeitig eine untere Schranke. Nun kann man aber auch eine Approximation von oben erzwingen, indem man nicht über die Testpunkte minimiert, sondern maximiert. Damit ergeben sich die akkumulierten Kosten

$$\hat{J}_{(F,G)}(x, \boldsymbol{u}, \boldsymbol{w}) = \sup_{(x_k)_k \in \chi_F(x, \boldsymbol{u}, \boldsymbol{w})} \sum_{k=0}^{\infty} G(x_k, x_{k+1}, u_k, w_k),$$

die Wertefunktion bleibt unverändert

$$\hat{V}_{(F,G)}(x) = \sup_{\beta \in \mathcal{B}} \inf_{u \in U^{\mathbb{N}}} J_{(F,G)}(x, u, \beta(u,x)),$$

die nun aber dem Optimalitätsprinzip

$$\hat{V}_{(F,G)}(x) = \inf_{u \in U} \sup_{x_1 \in F(x,u,w)} \sup_{w \in W} \{G(x, x_1, u, w) + \hat{V}_{(F,G)}(x_1)\}$$

bzw. mit $G(x, x_1, u, w) = g(x, u)$

$$\hat{V}_{(F,G)}(x) = \inf_{u \in U} \sup_{x_1 \in F(x,u,w)} \sup_{w \in W} \{g(x, u) + \hat{V}_{(F,G)}(x_1)\}$$

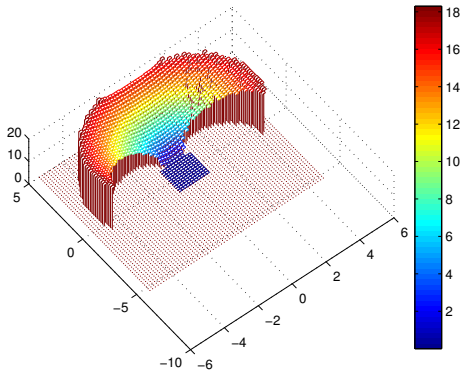
genügt.
Setze nun den diskreten Operator der dynamischen Programmierung auf

$$\hat{L}_{\mathcal{P}}[v](x) := \inf_{u \in U} \sup_{x' \in \rho(x)} \sup_{x_1 \in F(x',u,w)} \sup_{w \in W} \{g(x',u) + v(x_1)\} \tag{10}$$

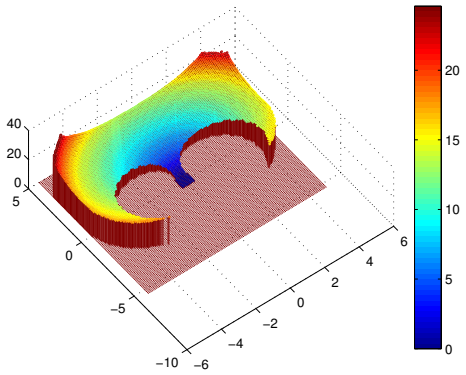
Zuguterletzt erhält man die diskrete optimale Wertefunktion als

$$\hat{V}_{\mathcal{P}}(x) = \inf_{u \in U} \sup_{x' \in \rho(x)} \sup_{x_1 \in F(x',u,w)} \sup_{w \in W} \{g(x',u) + \hat{V}_{\mathcal{P}}(x_1)\}. \tag{11}$$

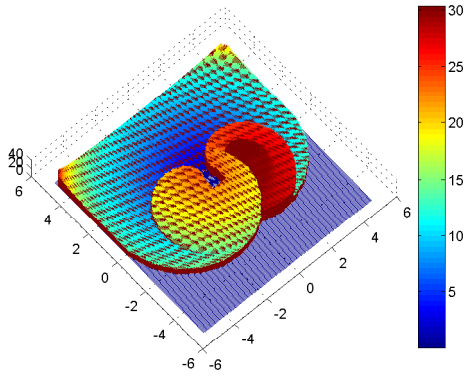
Die Werte von $\hat{V}_{\mathcal{P}}$ werden folglich größer als vorher ausfallen. Für Abbildung 16 wurde v_2 auf den Wert 0.5 gesetzt. Der Passant kann sich also nur mittelschnell fortbewegen. Ab $v_2 = 0.59$ werden nämlich beispielsweise für 64 Zellen je Richtung keine Werte neben ∞ mehr angenommen. Ab $v_2 = 0.76$ keine für 128 Zellen je Richtung. Wie man schön sehen kann, ist hier die Feinheit des Gitters von großer Bedeutung. Erst ab 512 Zellen pro Koordinatenrichtung erhält man repräsentative Ergebnisse. Die Approximation ist wie in der Theorie in [1] erst für einen kleinen Diameter, gleich der Durchmesser der Zelle, von annehmbarer Güte.



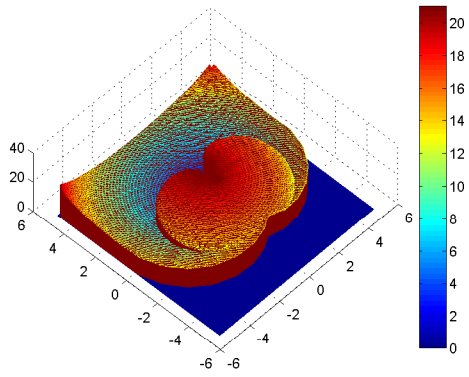
(a) 64 Zellen



(b) 128 Zellen



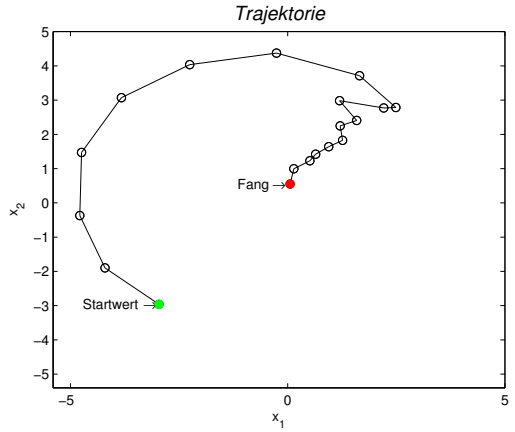
(c) 256 Zellen



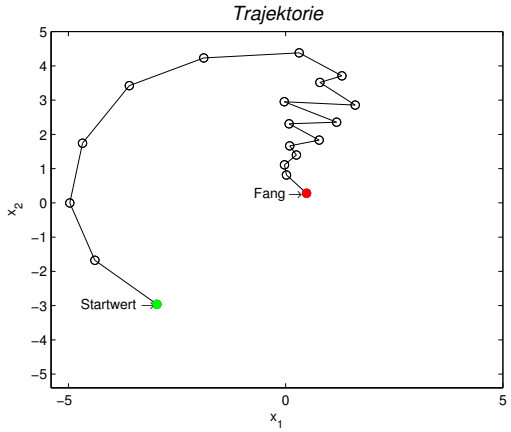
(d) 512 Zellen

Abbildung 16: Approximation der optimalen Wertefunktion von oben

Ein Trajektorienvergleich soll zeigen, dass sich die Wege sowohl für die Annäherung von unten als auch von oben nur unwesentlich unterscheiden. Allerdings fällt auf, dass dieser Unterschied für grobe Gitter, zum Beispiel 128 Zellen je Koordinatenrichtung, schon beachtlicher ausfallen kann. Dies macht sich insbesondere im „zickzackförmigen“ Verlauf bemerkbar.

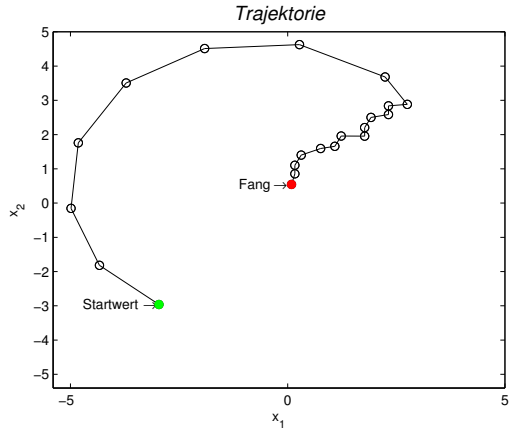


(a) 128 Zellen unten

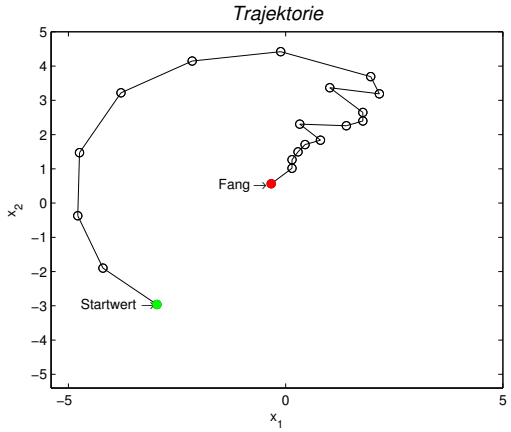


(b) 128 Zellen oben

Abbildung 17: Annäherung von unten und von oben I: Startwert (-2.9625,-2.9625)

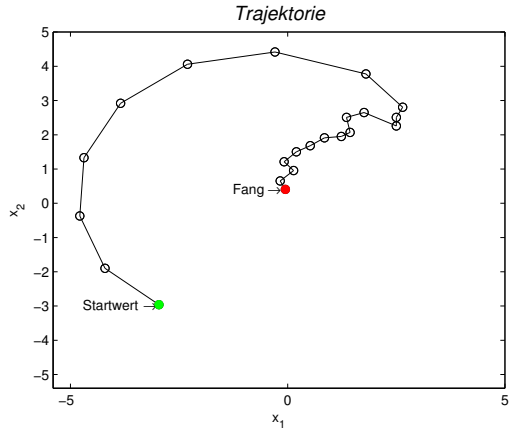


(a) 256 Zellen unten

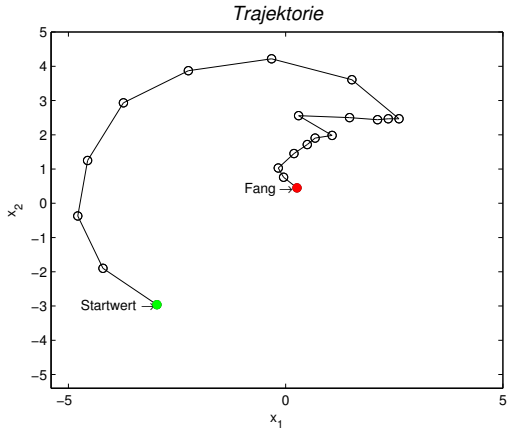


(b) 256 Zellen oben

Abbildung 18: Annäherung von unten und von oben II: Startwert (-2.9625,-2.9625)

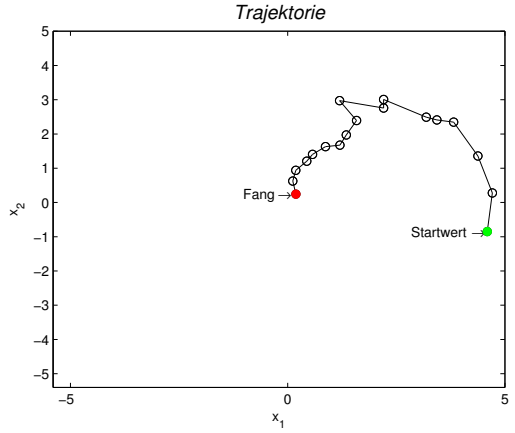


(a) 512 Zellen unten

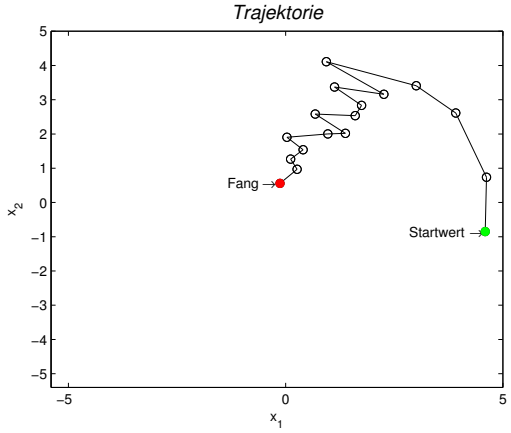


(b) 512 Zellen oben

Abbildung 19: Annäherung von unten und von oben III: Startwert (-2.9625,-2.9625)



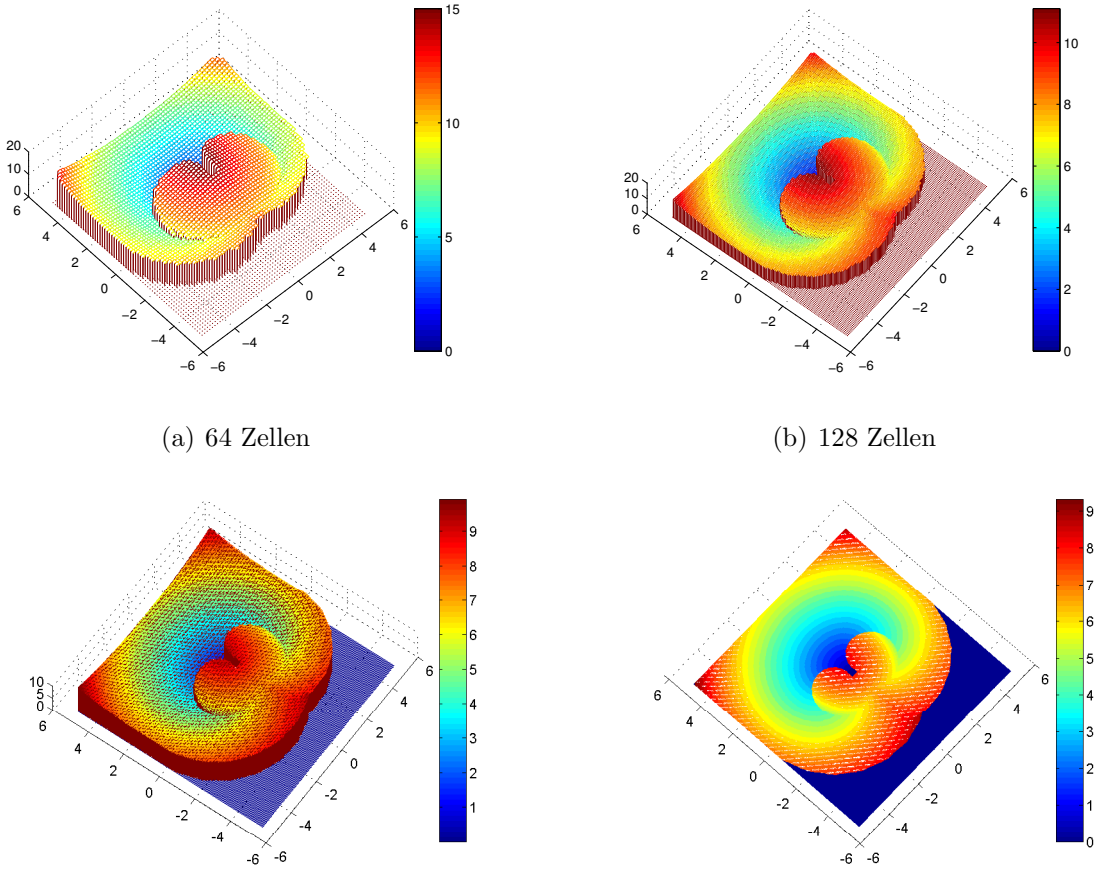
(a) 128 Zellen unten



(b) 128 Zellen oben

Abbildung 20: Annäherung von unten und von oben IV: Startwert (4.59375,-0.85)

Für $v_2 = 0.1$ sieht die Wertefunktion im Gegenzug bereits für 64 Zellen passabel aus. Da der Passant sehr langsam ist, kann ihn das Auto schnell einholen, weshalb die Werte insgesamt kleiner sind und selbst der maximale Testpunkt offenbar noch gut genug abschätzen kann.



(a) 64 Zellen

(b) 128 Zellen

(c) 256 Zellen

(d) 512 Zellen

Abbildung 21: Annäherung der Wertefunktion von oben

3.7 Ergebnisse und Vergleich mit der Theorie nach Isaacs

In [2,S.438ff] wurden Differentialspiele bereits analytisch behandelt. Auf die genaue Theorie des Lösungsansatzes der Isaacs-Gleichung und die einzelnen Rechnungen wird an dieser Stelle nicht eingegangen, doch Details können in der angegebenen Quelle nachgelesen werden. Auch wenn auf die Reproduktion der analytischen Untersuchung verzichtet wird, so sollen die aus ihr gewonnenen Resultate mit denen in dieser Arbeit erhaltenen Ergebnisse verglichen werden. Wie sich bald herausstellen wird, sind einige Parallelen zu verzeichnen.

Die Grundlage für die Behandlung dynamischer Spiele in [2] bildet die sogenannte Isaacs-Gleichung, die eine Bedingung für Sattelpunkt-Gleichgewichte liefert und eine Zwei-Personen-Erweiterung der Hamilton-Jacobi-Bellmann-Gleichung verkörpert.

Ein Strategienpaar (u^*, w^*) befindet sich im Sattelpunkt-Gleichgewicht, falls

$$J(u^*, w) \leq_{\forall w \in W} J(u^*, w^*) \leq_{\forall u \in U} J(u, w^*).$$

Das heißt, dass sich die Situation des Spielers verschlechtert, sobald er von seiner optimalen Strategie abweicht. Nehme man an, Spieler E weicht von seiner optimalen Strategie w^* ab, so verkürzt sich die Dauer bis zu seinem Fang, die über das Funktional J berechnet wird. Dies konnte bereits in Kapitel 3.5.2 beobachtet werden.
In [2] sind weitere Begriffe wie der Usable Part (UP) und die Barriere S von großer Wichtigkeit. Es wird zwischen den Anfangszuständen, die zur Terminierung des Spiels führen, und denen, die es nicht notwendigerweise tun, unterschieden. Dabei besteht S aus den Punkten, die diese beiden Anfangszustandsmengen voneinander abgrenzen. Der Usable Part ist die Menge aller Punkte des Randes der Zielmenge $\mathcal{D}$, die der Forderung der Terminierung des Spiels gerecht werden. Das sind also die Punkte, in denen die optimalen Trajektorien erstmals die Zielmenge $\mathcal{D}$ „treffen“.
Anhand der in [2] vorgenommenen Rechnungen kann zwischen zwei möglichen Konfigurationen des Spiels abhängig vom Fangradius ε des Chauffeurs unterschieden werden:

Fall 1

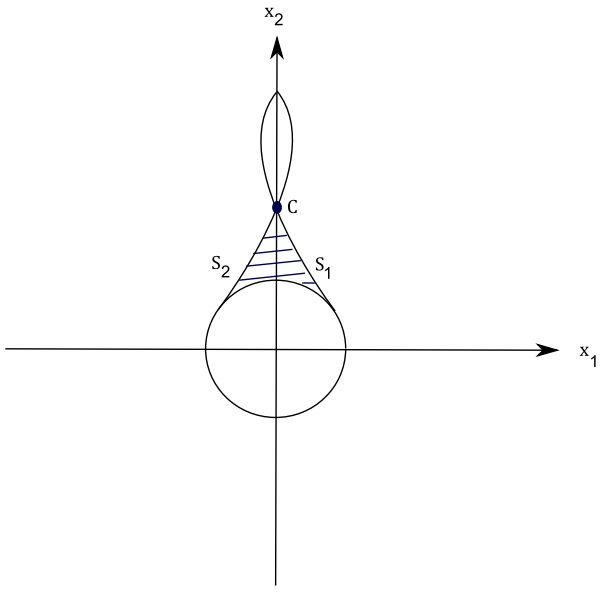


Abbildung 22: $\varepsilon \leq v_2 \arcsin v_2 + \sqrt{1 - v_2^2} - 1$

Abbildung 22 setzt voraus, dass sich Spieler E zu Beginn des Spiels innerhalb des schraffierten Bereiches befindet. Die Kurven S_1 und S_2 definieren semipermeable

Barrieren. Spieler P versucht zu verhindern, dass sie von Spieler E passiert werden; denn sobald der Passant den Bereich außerhalb der abgegrenzten, gestreiften Fläche betreten würde, würde er entkommen. Die Terminierung des Spiels kann folglich nur in dem schraffierten Bereich erfolgen. Dazu wird dem Chauffeur ein Fangradius unterstellt, der den Wert $v_2 \arcsin v_2 + \sqrt{1 - v_2^2} - 1$ nicht überschreitet. Die Situation, dass Spieler E die Flucht ergreifen kann, entspricht nicht dem Ziel der Berechnungen aus den Abschnitten 3.4 und 3.5. Die Zielmenge $\mathcal{D}$ (die dem Fangradius des Chauffeurs entspricht) wurde für jede Berechnung neu angepasst. Es wurde stets die kleinste Zielmenge gewählt, so dass für fast alle Anfangszustände die Terminierung des Spiels – der Fang – garantiert war. Deshalb beschränkt sich der erwünschte Vergleich der beiden Lösungsansätze auf

Fall 2

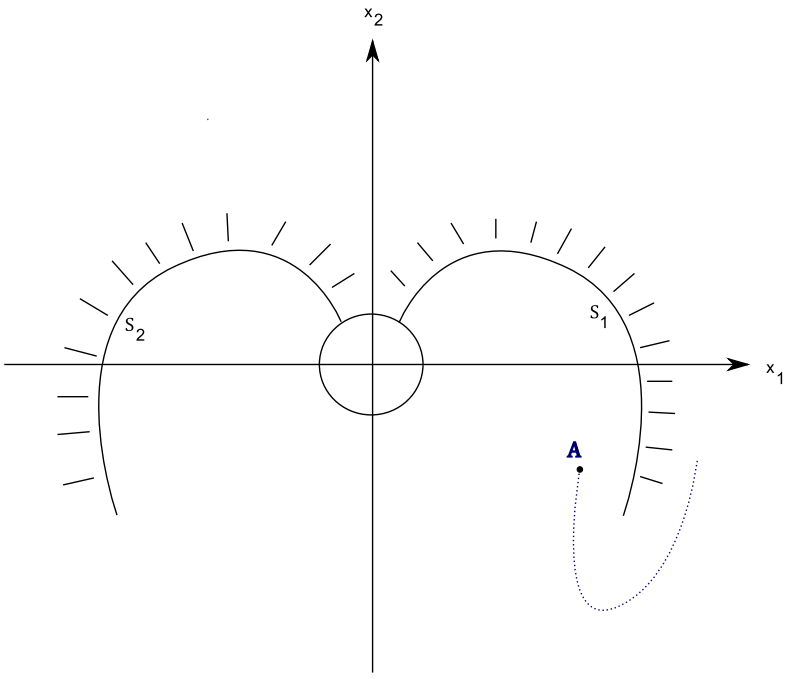


Abbildung 23: $\varepsilon > v_2 \arcsin v_2 + \sqrt{1 - v_2^2} - 1$

Obwohl, wie in Abbildung 23 zu sehen ist, die Existenz von semipermeablen Barrieren S_1 respektive S_2 immer noch gewahrt ist, so gilt selbst für Anfangszustände außerhalb des durch sie induzierten Bereiches der sicheren Terminierung des Spiels, dass der Fang herbeigeführt werden kann. Hierzu betrachte man sich den Punkt A aus Abbildung 23. Zwar wird der Spieler E nicht gezwungen, S_1 zu passieren, und doch kann Spieler P eine Bewegung des Systems entlang der blau gestrichelten Linie erzwingen, womit sich Spieler E ab einem gewissen Zeitpunkt im angedeuteten schraffierten Bereich befindet und

letztlich überfahren wird. Dazu muss der Fangradius ε größer als $v_2 \arcsin v_2 + \sqrt{1 - v_2^2} - 1$ sein.

Die semipermeablen Barrieren S_1 und S_2 bilden zusammen eine Struktur, die dem aufmerksamen Leser bekannt vorkommen dürfte. Diese „herzförmige“ Form findet sich nämlich in den Abbildungen 10 und 11 der optimalen Wertefunktion wieder. Sie kommt gerade durch die Startwerte, für die sich der Passant direkt neben dem Chauffeur oder hinter diesem befindet, zustande. Wenn man den beliebigen Wert $v_2 = 0.5$ hernimmt, so muss ε größer als 0.1278 sein, damit diese Struktur des Spiels erhaltbar ist. Die numerischen Berechnungen haben für dieses Resultat einen höheren Wert erfordert. Ab einer Gitterfeinheit von 128 Zellen je Koordinatenrichtung beläuft sich ε auf etwa 0.25. Grund hierfür ist insbesondere die Diskretisierung des Kontroll- und des Störraumes. Wie bereits erwähnt, werden die einst messbaren Kontrollfunktionen zu stückweise konstanten Funktionen, womit die Spieler nicht mehr in der Lage sind, bestmöglich zu agieren und zu reagieren. Ihre Bewegungsfreiheit wird sozusagen eingeschränkt, was sich vor allem kurz vor dem Fang bemerkbar macht. Daraus resultierte in den optimalen Trajektorien auch der zickzackförmige Verlauf.
Die Numerik benötigt für dieselbe Struktur der optimalen Wertefunktion einen fast doppelt so großen Fangradius und doch hat sie ihre Vorzüge. Nicht nur, dass sich die Wertefunktion aufgrund der gewonnenen Daten leicht veranschaulichen lässt, auch die optimalen Trajektorien können ohne großen Aufwand erhalten werden. Selbst solche Lösungskurven, die eine nicht-optimale Strategie für mindestens einen der Spieler unterstellen, sind einfach zu berechnen, sobald eine Modellierung für das Spiel gewählt wurde. Die Fehler, die beim Einsatz der numerischen Mathematik entstehen, dürfen jedoch nicht unbeachtet bleiben. Durch die Versinnbildlichung des Spiels mittels Chauffeur und dem Mann, der überfahren werden soll, ist ein etwas größerer Fangradius vielleicht leicht hinzunehmen, aber meist dienen solche Allegorien nur dem besseren Verständnis. Man muss sich vorher überlegen, welcher Zweck hinter dem Kontrollsystem steckt. Was möchte man regulieren? Was ist das eigentliche Ziel? Sind solche Abweichungen wie die Varianz des Fangradius hinnehmbar?

4 The Lady In The Lake

Ein zweites, interessantes Anwendungsbeispiel stellt das Spiel „The Lady In The Lake“ dar. Wie der Name schon preisgibt, dreht sich das dynamische Spiel um eine Frau (Spieler E), die in einem kreisrunden See schwimmend versucht, ans Ufer zu gelangen, ohne dabei in die Hände ihres Verfolgers (Spieler P) zu fallen. Der Jäger, ein unterstellter Nichtschwimmer, kann sich nur entlang des Ufers bewegen. Sobald die Frau den Rand des Sees erreicht, ist das Spiel mit einem der beiden möglichen Spielausgänge beendet: Entweder entkommt die Frau, wenn sie an den Teil des Gestades stößt, wo sich der Mann gerade nicht aufhält, oder sie wird von jenem gefangen genommen.
Auch hier wird wie im „Homicidal Chauffeur Game“ die Annahme getroffen, dass sich beide Spieler mit konstanter Geschwindigkeit fortbewegen. Damit das Spiel nicht trivial wird, soll die Frau an Land schneller sein als ihr Jäger, dessen maximale Geschwindigkeit auf $v_1 = 1$ gesetzt wird. Im See soll sie sich allerdings langsamer fortbewegen, weshalb $v_2 < 1$ ist. Zudem wird vorausgesetzt, dass keiner der Spieler ermüdet, die Frau jedoch nicht permanent im See verweilen möchte. Auf diese Weise wird immer die Terminierung des Spiels garantiert. Nur der Ausgang kann nicht vorhergesagt werden.

4.1 Mathematische Beschreibung des Spiels

Nach [2] kann man „The Lady In The Lake“ wie folgt in Polarkoordinaten beschreiben (gleichzeitig sei auf Abbildung 24 verwiesen):

DGL-System 4.1.

$$\begin{aligned} \dot{\Theta} &= \frac{v_2 \sin w}{r} - \frac{u}{R} \\ \dot{r} &= v_2 \cos w \end{aligned}$$

mit

(i) Θ *ist der Winkel zwischen P und E*

(ii) r *ist der Abstand von E zum Mittelpunkt des Sees (Punkt C)*

(iii) R *ist der Radius des Sees*

(iv) v_1 *und* v_2 *sind die konstanten Geschwindigkeiten von P (Mann) und E (Frau)*

(v) $|u| \leq 1$ *Kontrolle des Systems, Bewegungsrichtungsänderung von P (Vorzeichen hängt von der Richtung ab, mit der P um den See läuft)*

(vi) Störung $w \in [-\pi, \pi]$, Winkel des Geschwindigkeitsvektors von E bezüglich CE

Zu beachten ist das r im Nenner der ersten Differentialgleichung. Für $r = 0$ ergäbe sich eine Division durch 0, was aber nicht erlaubt ist. Die Frau befände sich für diesen Fall direkt im Mittelpunkt des Sees. Bei den späteren Berechnungen ist daher auf die Position der Frau zu achten.

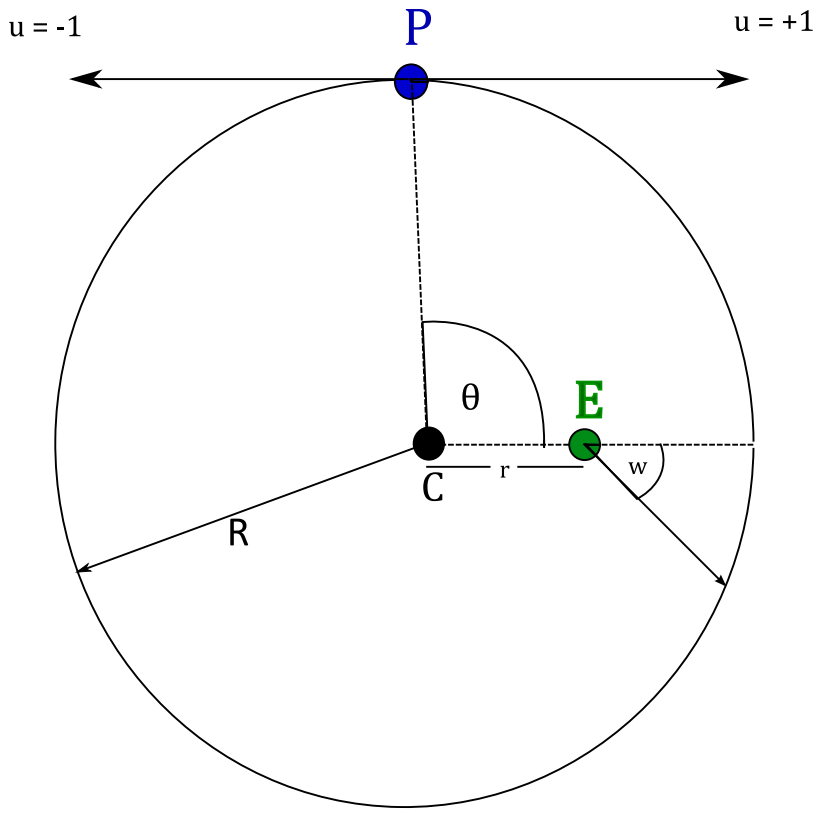


Abbildung 24: The Lady In The Lake

Da die Modifikation des Dijkstra-Algorithmus, die hier verwendet wird, immer ein $\min_u \max_w$-Problem simuliert, muss an dieser Stelle eine kleine Veränderung am Differentialgleichungssystem vorgenommen werden. Im Folgenden wird nicht das Erreichen des Ufers der Frau allgemein betrachtet, sondern einzig ihre Flucht wird als Zielmenge dienen. Das Differentialgleichungssystem 4.1 ist so modelliert worden, dass der Rand des Sees das Spielende bedeutet, gleichgültig, welcher Spielausgang zugrundeliegt. Unter der Voraussetzung, dass die Zellen des Fangs nicht zu $\mathcal{D}$ gehören, kann es während der Ausführung von 2.13 vorkommen, dass nach geraumer Zeit die Zellen, die den Fang garantieren, wieder verlassen würden und das Spiel in einer anderen Zelle enden würde.

Das heißt, die Frau würde entkommen, obwohl sie bereits gefangen genommen wurde. Das widerspräche jeglicher Vernunft. Außerdem berücksichtigt 2.13 nach bisheriger Implementierung keine Endkosten. Alle Zellen der Zielmenge werden auf den Wert 0 gesetzt und bleiben während der Berechnung von $V_{\mathcal{P}}$ unverändert. Somit kann nach Beendigung des Algorithmus bei Betrachtung der Daten nicht unterschieden werden, welche Anfangswerte zum Fang führen und welche nicht. Aufgrund der großen Anzahl von Zellen kann dies auch nicht durch eine graphische Veranschaulichung der Trajektorien erreicht werden. Das bedeutet, dass nicht der gesamte Rand des Sees als Zielmenge fungieren kann.
Aus diesem Grund ist nur die Flucht der Frau für die Modellierung von Interesse. Die Flucht und nicht der Fang aus dem Grund, weil sie den Großteil des Seerands ausmacht. Sobald sich das System innerhalb des Fangradius befindet, kostet der Übergang zum nächsten Zustand unendlich viel, unabhängig von der Wahl von u und w. So soll erzwungen werden, dass keine Zelle, die innerhalb des Fangradius liegt, während der Ausführung des Algorithmus als Aufenthaltsort der Frau – nicht einmal für kurze Zeit – dienen kann, da die optimale Wertefunktion der Länge des kürzesten Weges entspricht und die Länge ja nichts anderes als die Summe der Bewertungen der Kanten ist.

Was ist der Fangradius nun eigentlich? Was kann man sich unter Fangzellen vorstellen? – Damit sind diejenigen Zellen gemeint, in denen der Mann die Frau ergreifen würde (Abbildung 25). Für jede Feinheit des Gitters wird derselbe Fangradius unterstellt. Ob der Zustandsraum nun in 256 ($= 16^2$) oder in 262144 ($= 512^2$) Zellen unterteilt wird, die Armlänge des Mannes, welche man als Fangradius interpretieren könnte, bleibt immer gleich.

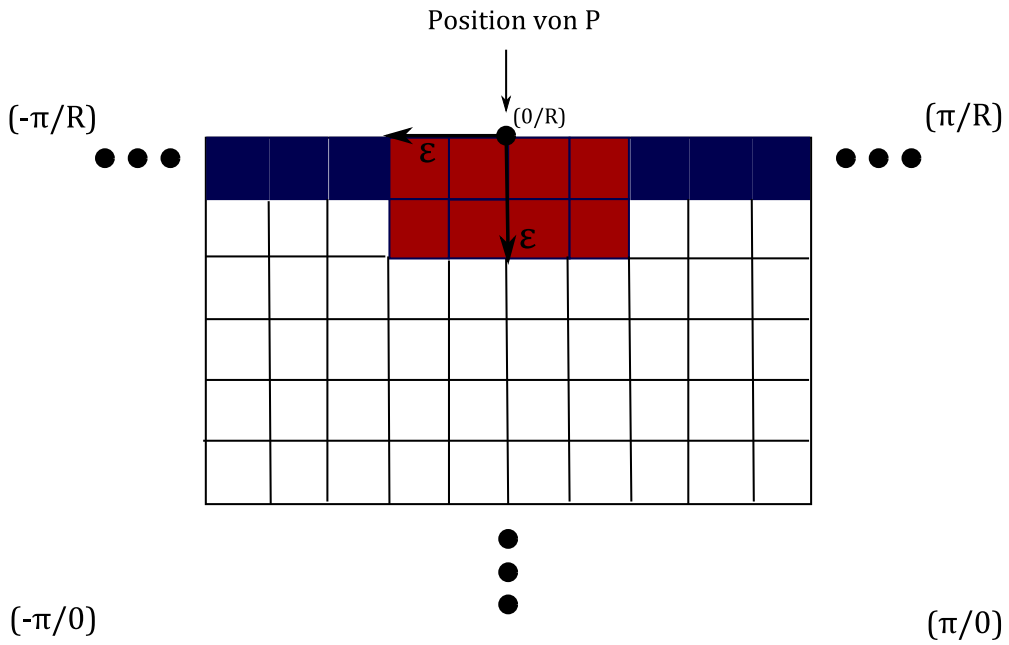


Abbildung 25: Fangbereich mit Fangradius ε

Bemerkung 4.2. *Es zählen auch die Zellen zum Fangbereich hinzu, die durch den Halbkreis, der durch die Armlänge ε bestimmt wäre, nicht getroffen werden, aber Teil des Rechtecks $[-\varepsilon, \varepsilon] \times [R - \varepsilon, R]$ sind.*
Abbildung 25 ist ein Vorgriff auf die Wahl des Zustandsraumes, der im Folgenden noch erläutert wird.

In den Untersuchungen in [2] hatte Spieler P noch das Interesse, den Abstand zu Spieler E, der durch den Winkel Θ gegeben ist, zu minimieren, während E ihn maximieren wollte (vgl. Abb.24). Als Kostenfunktion diente daher

$$g(\Theta, r) = |\Theta(T)|, \quad T = \min\{t \mid r(t) = R\}.$$

Wenn $g(\Theta, r)$ also den Wert 0 annahm, dann erfolgte der Fang.

Der Hypergraph $H = (\mathcal{P}, E)$ wird nach bisheriger Argumentation aber aus Sicht der im See schwimmenden Frau aufgebaut. Das soll heißen, dass der Frau die Kontrolle u unterliegt und der Mann die Störung w beeinflusst. Für 4.1 bedeutet das eine Umbenennung der Variablen u in w und umgekehrt. Differentialgleichungssystem 4.1 wird also zu

DGL-System 4.3.

$$\begin{aligned} \dot{\Theta} &= \frac{v_2 \sin u}{r} - \frac{w}{R} \\ \dot{r} &= v_2 \cos u. \end{aligned}$$

Als Kostenfunktion fungiert nicht mehr der Winkel zwischen den Spielern zum Zeitpunkt der Terminierung des Spiels, sondern es werden laufende Kosten analog zum vorherigen Beispiel eingeführt. Zunächst wird g wieder gleich dem Zeitschritt h gesetzt und wird mit g_1 bezeichnet; also

$$g_1(\Theta, r, u) = h.$$

Damit ist es irrelevant, ob die Frau sehr nahe dem Mann das Ufer erreicht oder weit entfernt von ihm. Nur die Zeit, bis sie einen Fuß an Land setzt, wird hier gemessen. Das soll in Unterkapitel 4.2 allerdings noch ein wenig mehr dem ursprünglichen Spiel angepasst werden.
Mittels dieser Modellierung möchte die Frau so schnell wie möglich das Ufer erreichen und ihr Gegenpart möchte das so lange wie möglich hinauszögern. Da als Zielmenge nur solche Zellen, die immer die Flucht und nie den Fang ermöglichen, gewählt werden, ist das Verhalten der Spieler durchaus einsichtig: Wenn der Mann die Frau schon nicht fangen kann, dann will er sie zumindest so lange wie möglich im See verharren lassen.
Neben den laufenden Kosten werden im Hauptprogramm weitere Kosten zugewiesen. Das System soll daran gehindert werden, in den Fangbereich gesteuert zu werden. Dies wird erreicht, indem jeder Übergang des Systems innerhalb der Fangzellen mit den Kosten ∞ belegt wird. Die Kostenfunktion lässt sich damit folgendermaßen aufspalten:

$$g_1(\Theta, r, u) = \begin{cases} h, & \text{wenn } (\Theta, r) \text{ nicht im Fangbereich} \\ \infty, & \text{wenn } (\Theta, r) \text{ im Fangbereich.} \end{cases}$$

Die Unstetigkeit in der Kostenfunktion bereitet keine Probleme, da das Spiel ohnehin diskretisiert wird.
Die Fangzellen bekommen eine Art Sonderstellung. Sie gehören weder zur Zielmenge $\mathcal{D}$ noch dienen sie je als Aufentshaltort des Systems. Das heißt, sie bekommen im ersten Schritt des Algorithmus 2.13 den Wert unendlich zugewiesen, doch in den nachfolgenden Schritten wird aufgrund der immensen Kosten beim Systemübergang dieser Wert nicht mehr geändert. Nur für Startwerte innerhalb des Fangbereichs erfolgt auch wirklich der Fang. Ansonsten wird stets die Flucht der Frau garantiert.

Wahl von X, U und W

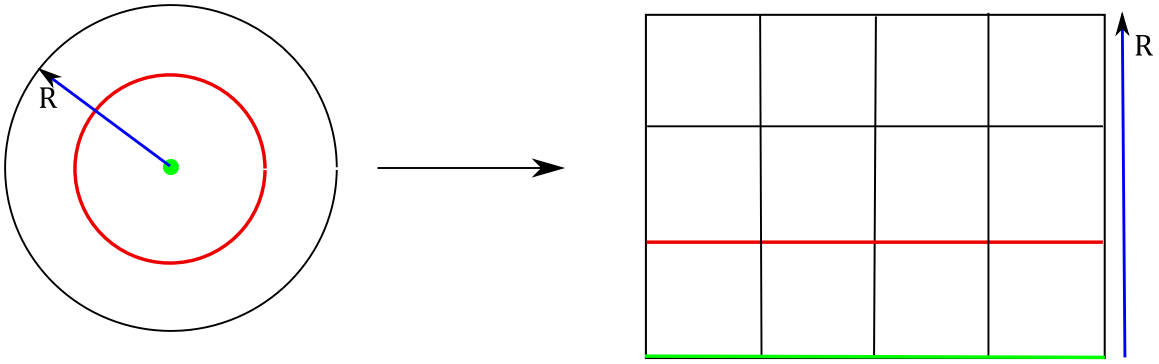


Abbildung 26: Zustandsraum

Das Differentialgleichungssystem 4.3 ist in Polarkoordinaten gegeben, weshalb die Wahl des Zustandsraums auf $X = [-\pi, \pi] \times [0, R]$ gefallen ist (Abb.26). Da es während den Systemübergängen durchaus passieren kann (zu beachten ist, dass der Algorithmus in kartesischen Koordinaten rechnet), dass Θ Werte außerhalb des Intervalls $[-\pi, \pi]$ annimmt, muss in der Implementierung eine entsprechende Projektion auf dieses Intervall vorgenommen werden. Aber aufgrund der Periodizität des Winkels verursacht dies keine Probleme.
Für den Kontrollraum bietet sich ebenfalls das Intervall $[-\pi, \pi]$ an, da sich die Bewegungsart der Frau nach System 4.1 nicht geändert hat. Als Störraum dient das Intervall $[-1, 1]$, wobei $+1$ bedeutet, dass der Mann im Uhrzeigersinn um den See läuft und -1 gegen den Uhrzeigersinn. Man beachte, dass er den See nicht durchqueren kann. Hierzu sei nochmals auf Abbildung 24 verwiesen.

Bemerkung 4.4. *Ob der Modellierung, insbesondere der Wahl des Zustandsraums, ist Vorsicht geboten. Der Mittelpunkt des Sees wird als Menge unendlich vieler Punkte gehandhabt. Etwaige Singularitäten haben sich in den Berechnungen allerdings nicht bemerkbar gemacht.*

4.2 Wertefunktion und Trajektorien für g_1

4.2.1 Wertefunktion

Das Optimalitätsprinzip für „The Lady In The Lake“ entspricht dem des dynamischen Spiels „Homicidal Chauffeur Game“, da dieselbe Kostenfunktion zugrunde liegt. $\tilde{u}$ als auch $\tilde{w}$ lassen sich wieder über (9) bestimmen.
Ebenfalls wie im vorherigen Spiel kann man diverse Parameter beeinflussen: Die Geschwindigkeit v_2 der Frau und die Feinheit des Gitters, die in der Implementierung in z festgelegt wird, wirken sich auf die Gestalt der optimalen Wertefunktion aus. Die Anzahl der Testpunkte einer Zelle wie die aus U und W werden nicht gesondert diskutiert.
Für die folgenden Berechnungen samt ihrer Interpretationen muss darauf hingewiesen werden, dass der Abstand Θ zwischen den beiden Spielern, ob er nun $-\pi$ oder π beträgt, derselbe ist, selbst wenn dies in der Darstellung als Rechteck nicht sofort ersichtlich ist.

Aufgrund des Fangbereiches, der von der Frau gemieden werden möchte, ergibt sich eine besondere Struktur der optimalen Wertefunktion. Anhand eines ausgewählten Beispieles soll diese vor diversen Variationen erläutert werden. Hierzu wird ein recht feines Gitter mit je 256 Zellen pro Koordinatenrichtung gewählt und v_2 auf den beliebigen Wert 0.5 gesetzt.

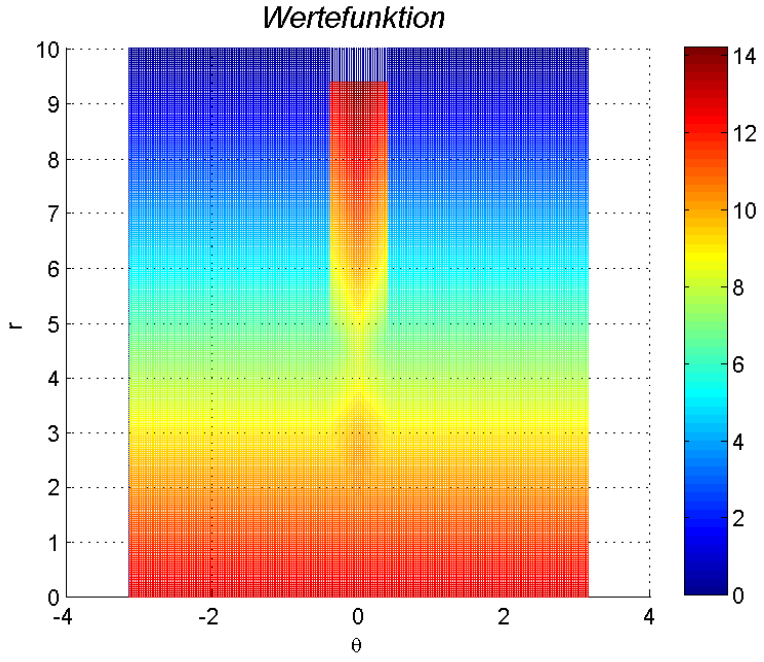
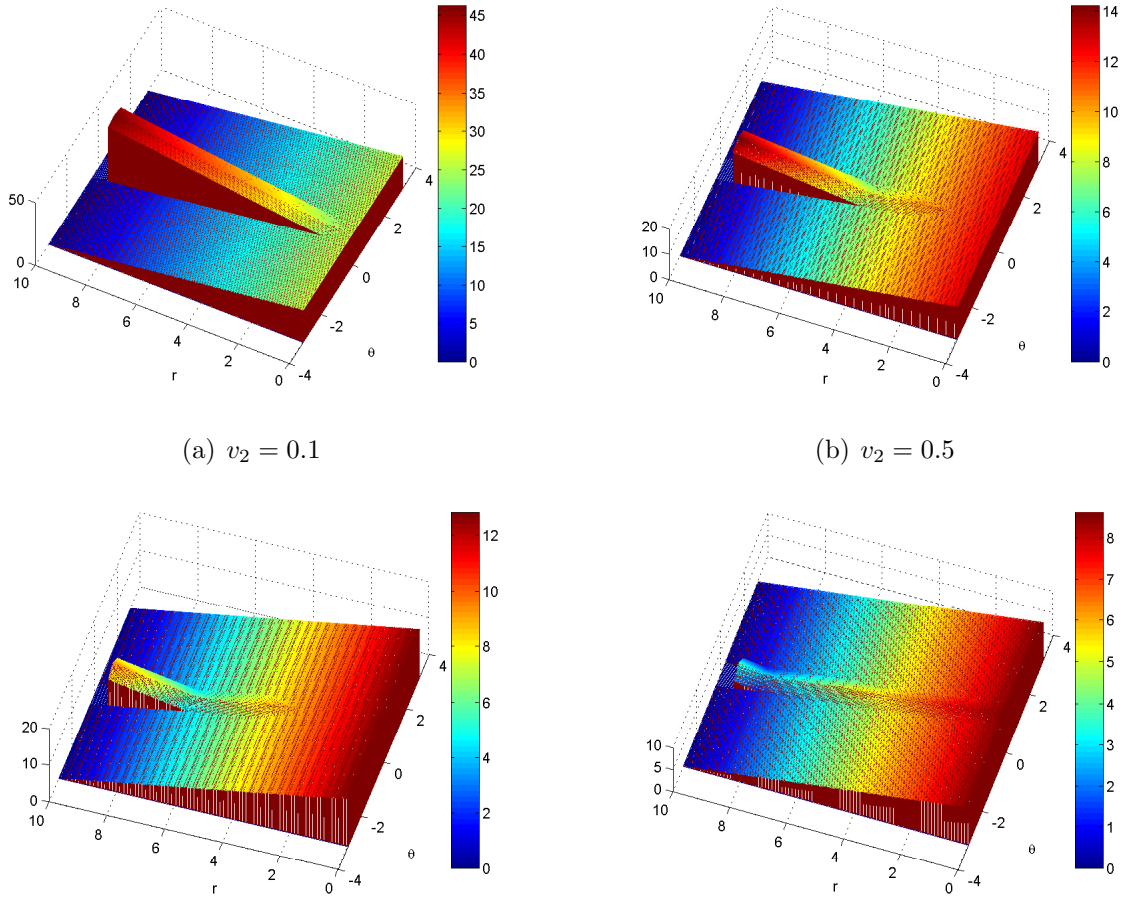


Abbildung 27: Optimale Wertefunktion - Struktur

Diejenigen Zellen, die mit den Kosten ∞ belegt werden, beeinflussen den Wert der Zellen, deren Θ-Wert nahe der 0 ist. Bevor dies genauer erklärt wird, ist zu erwähnen, dass alle Zellen der optimalen Wertefunktion, die denselben Radius r haben, den gleichen Wert zugewiesen bekommen mit Ausnahme derjenigen, die durch die hohen Kosten der Fangzellen beeinträchtigt werden. Die Begründung hierfür ist trivial, da lediglich rein das Entkommen der Frau modelliert wird und der genaue Ort, wo sie das Ufer erreicht, keine Rolle spielt. Damit ist es nicht von Belang, ob die Frau nahe dem Mann dem See entsteigt oder weit entfernt von ihm. Dieselbe Entfernung vom Ufer impliziert daher auch denselben Wert der optimalen Wertefunktion.
Ohnehin ist eine gewisse Symmetrie in der Wertefunktion zu erkennen. Die Funktion ist achsensymmetrisch entlang der r-Achse. Denn ob die Frau sich nun entscheidet, nach rechts oder nach links von $(0, r)$, $r \in [0, R - \varepsilon)$, ε Fangradius, aus gen Ufer zu schwimmen, an der Dauer ihres Aufenthalts im See ändert sich nichts. Die Symmetrie liegt aber auch darin begründet, dass ihre Geschwindigkeit als konstant angenommen und von unterschiedlichen Strömungen im See abgesehen wird.
In der unmittelbaren Umgebung des Aufenthaltsortes des Mannes sind die Werte sehr groß, da die Frau nicht auf direktem Wege zum Ufer schwimmen kann, sondern erst dem „kritischen Bereich“ (damit ist der Bereich gemeint, der in Abb. 27 entlang der r-Achse aufgrund der großen Werte hervorsticht) entfliehen muss. Das vollzieht sie, indem sie gen Mitte des Sees schwimmt und von dort aus ihre höhere Winkelgeschwindigkeit nutzend Richtung Ufer. Je näher sie also dem Mann ist, einen desto weiteren Weg legt sie letztendlich zurück, um ihm zu entkommen und desto größer ist damit der Wert von $V_{\mathcal{P}}$.

Nachdem die Struktur von $V_{\mathcal{P}}$ nun klar ist, kann auf eine Variation der Geschwindigkeit, mit der sich die Frau im See fortbewegt, eingegangen werden. Je kleiner v_2 ist, desto größer werden die Werte nahe des Fangbereiches und auch sonst ausfallen. Die Frau braucht länger, bis sie das Ufer erreicht, je langsamer sie ist. Dies impliziert aber auch, dass der kritische Bereich „ausgeprägter“ sein wird im Sinne seines Ausmaßes. Nur wenn die Frau schnell schwimmen kann, entkommt sie dem kritischen Bereich recht zügig, so dass sich die Ränder dieses Bereiches wertmäßig den Zellen außerhalb annähern können. Das würde in diesem Fall bedeuten, dass nur die Startwerte in einer kleinen Umgebung der r-Achse besonders durch den Fangbereich in der Länge des Weges bis zum Ufer beeinflusst werden. Wichtig zu verstehen ist hierbei noch, dass die Frau eine Strategie des Minimierens verfolgt und sich daher immer direkt dem Ufer nähern möchte. Sobald ihre Ausgangsposition auf der r-Achse oder in ihrer Nähe liegt, kann sie dies nicht tun, da der Mann keine feste Position innehat, sondern sich ebenfalls bewegen kann, sei es wie in diesem Spiel lediglich um den See herum.

Graphisch sieht das folgendermaßen für repräsentative v_2 aus:

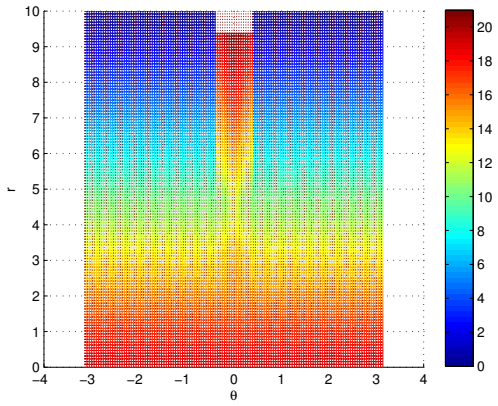


(a) $v_2 = 0.1$ (b) $v_2 = 0.5$

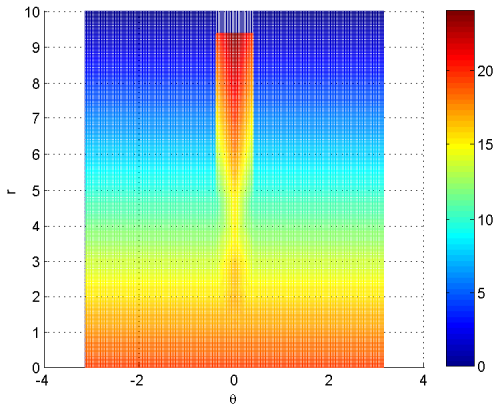
(c) $v_2 = 0.7$ (d) $v_2 = 0.9$

Abbildung 28: Optimale Wertefunktion mit unterschiedlichem v_2

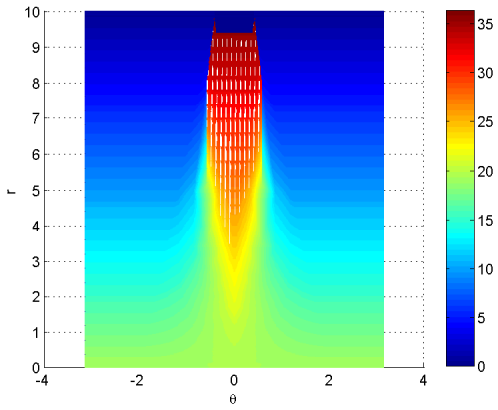
Ein weiteres interessantes Resultat ergibt sich durch die Betrachtung verschiedener Feinheiten des Gitters (vgl. Abb.29). Je kleiner der Diameter, also der Durchmesser einer Zelle, desto größere Werte werden von $V_{\mathcal{P}}$ angenommen und desto weiter erstreckt sich der kritische Bereich gen Seemittelpunkt. Hier wird deutlich, dass die Genauigkeit numerischer Berechnungen stark von der gewählten Diskretisierung abhängt, ebenso wie es sich bei der Setzung entscheidender Parameter wie v_2 verhält.
Ein Test mit 1024 oder mehr Zellen je Koordinatenrichtung konnte leider in Ermangelung größerer Arbeitsspeicher nicht durchgeführt werden. Aus diesem Grund liegt kein Vergleich mit weiteren Gitterfeinheiten vor, weshalb auf eine weiterführende Diskussion an dieser Stelle verzichtet wird.



(a) 128 Zellen



(b) 256 Zellen



(c) 512 Zellen

Abbildung 29: Optimale Wertefunktion mit zunehmender Zellenanzahl

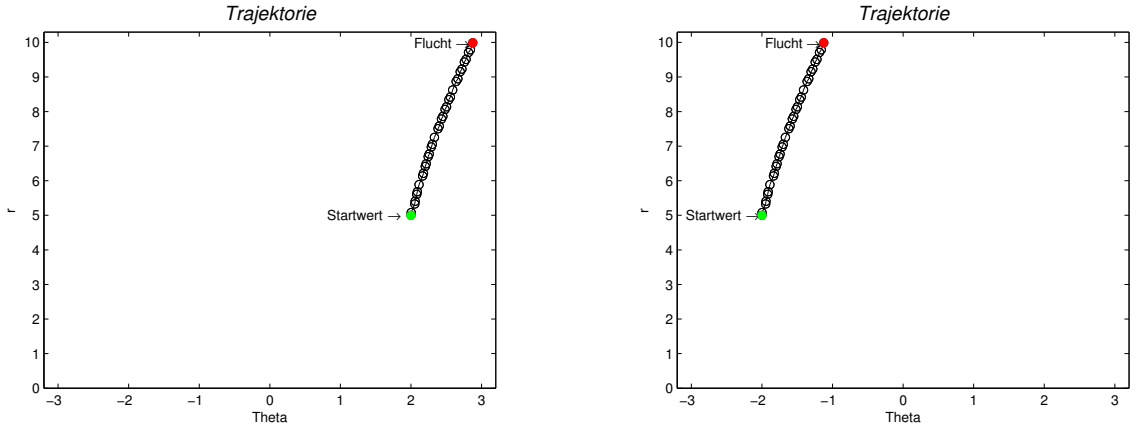
4.2.2 Trajektorien

Auffällig ist die Wahl von w in jedem Zeitschritt durch den Mann, aber auch die Wahl von u durch die Frau. Sobald die Position der Frau außerhalb des kritischen Bereichs liegt, wird der Mann natürlich das Ziel verfolgen, schnellstmöglich seinem Opfer hinterherzulaufen. Und genau das geschieht, wenn er mit $+1$ nach links oder mit -1 nach rechts steuert. Sobald er eine Richtung eingeschlagen hat, wird er diese beibehalten und sich daher unentwegt mit $+1$ respektive -1 fortbewegen. Zumindest bis sich die Frau in der Nähe seines Fangradius befindet. Als optimale Strategie resultiert daraus $\boldsymbol{w} = (\star, ..., \star, +1, +1, +1, ..., +1, \star, ..., \star)$ oder $\boldsymbol{w} = (\star, ..., \star, -1, -1, -1, ..., -1, \star, ..., \star)$, wobei $\star \in W$ und für unbekannte Werte steht. Da die Frau mit der unterstellten Kostenfunktion g_1 lediglich das Ufer erreichen möchte, entscheidet sie sich ebenfalls für eine Richtung und behält diese bei. Aufgrund der

Diskretisierung des Kontrollraumes U passiert es allerdings, dass ihre gewählte Kontrolle nicht über mehrere Zeitschritte hinweg konstant ist, sondern zwischen einigen Werten hin- und herspringt. Ihre optimale Strategie lässt sich daher nicht pauschalisieren.
Eine weitere Eigenschaft lässt sich aus den Daten herauslesen: Je kleiner v_2, desto mehr Systemübergänge werden benötigt, bis die Zielmenge $\mathcal{D}$ erreicht wird. Während es im Fall $v_2 = 0.3$ für den beliebig gewählten Startwert (1.0,7.0) noch 49 Stück sind, sind es für $v_2 = 0.9$ gerade einmal 12. Aufgrund der schnelleren Fortbewegungsmöglichkeit der Frau wird das System in jedem Zeitschritt näher gen Ufer gesteuert und dieses folglich auch eher erreicht. In Abb. 28 war die Abnahme der Werte bereits deutlich zu erkennen.
Nachdem bereits in der Wertefunktion Symmetrieeigenschaften festgestellt wurden, würde man sie auch bei den Trajektorien vermuten. An sich gibt es sie auch, aber Algorithmus 2.13 verhindert sie in gewisser Weise. Die gestörte Variante des Dijkstra-Algorithmus wählt in der Berechnung des Kontroll- und des Störinputs immer dasjenige u bzw. w, das minimiert bzw. maximiert, und tauscht bei gleichem Wert nicht aus. Wenn man diese Wahl ein wenig modifizieren würde, könnte man die Symmetrie sicherlich erzwingen.

Bemerkung 4.5. *Die Berechnungen wurden alle mit 256 Zellen je Koordinatenrichtung durchgeführt, da dies ausreicht, um alle sichtbar auftretenden Effekte zu signalisieren. Der Zeitschritt h wird auf* 0.5 *gesetzt.*

Ein paar repräsentative Startwerte sollen das Verhalten der Spieler verdeutlichen. Ist der Winkel Θ zu Beginn zwischen den beiden Spielern größer als ε (Fangradius), so wird die Frau auf geradem Weg gen Ufer schwimmen. Sobald sich Startwerte auf derselben Kreisbahn (Mittelpunkt ist Zentrum des Sees) befinden, gleichen sich die Trajektorien und sind lediglich um den Wert Θ verschoben, was bereits das Aussehen der optimalen Wertefunktion angekündigt hatte. Abbildung 30 zeigt die optimalen Trajektorien für die Startwerte (2.0,5.0) und (-2.0,5.0). Es verwundert auf den ersten Blick, dass die Trajektorie für den Startwert (-2.0,5.0) gen Fangbereich tendiert, wenn die Trajektorie für Startwert (2.0,5.0) doch in die andere Richtung strebt. Dies liegt an Algorithmus 2.13. Jeweils das erste minimierende u bzw. maximierende w wird gewählt und während der Ausführung von 2.13 nicht getauscht, selbst wenn ein nachfolgendes u bzw. w denselben Wert für die optimale Wertefunktion liefern würde. Darum kann es wie in diesem Beispiel auftreten, dass der Verlauf der Trajektorie entgegen den Erwartungen ausfällt.



(a) Start (2.0/5.0)

(b) Start (−2.0/5.0)

Abbildung 30: Trajektorien mit Startwert auf derselben Kreisbahn

Für Werte innerhalb des kritischen Bereichs ist ein weiteres Phänomen zu beobachten(s. Abb.31). Denn dort verhält sich die Frau anders. Ein gerader Kurs würde sie direkt in die Arme ihres Widersachers treiben, weshalb sie erst in Richtung Seemittelpunkt schwimmt und dort ihre größere Winkelgeschwindigkeit (bezüglich dem Zentrum C aus Abb.24) gegenüber der des Mannes ausnutzt. Ihr Entkommen ist garantiert, nur benötigt sie dafür für bestimmte Startwerte mehr Zeit.

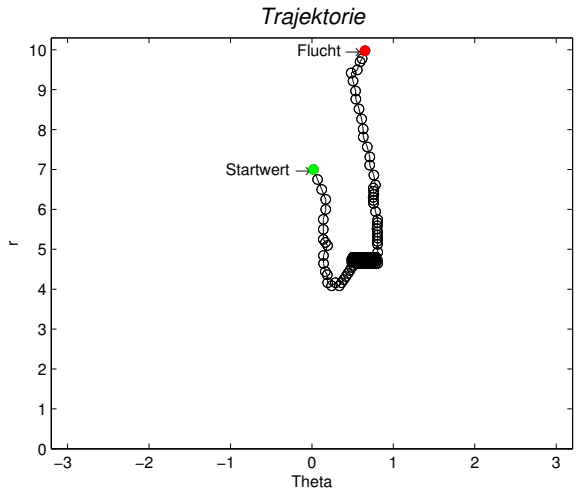
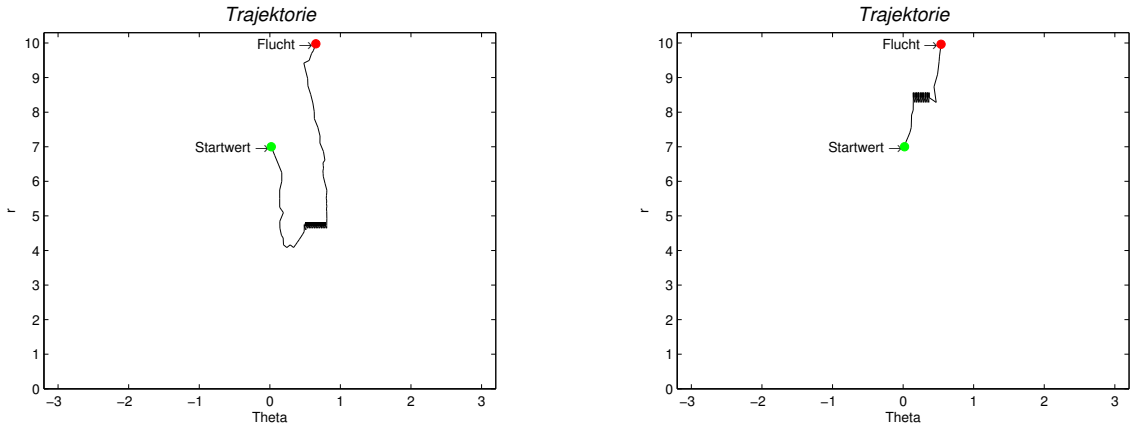


Abbildung 31: Trajektorie mit Startwert innerhalb des kritischen Bereichs

Die Geschwindigkeit v_2, mit der sich die Frau im See fortbewegt – sie wird als konstant angenommen, beeinflusst das Aussehen der Trajektorien zum Teil enorm. Es gibt vor allem mit Startwerten innerhalb des kritischen Bereichs deutliche Unterschiede. In Abb. 27 deutete die Approximation der optimalen Wertefunktion bereits Diversifikationen an.

Je schneller die Frau schwimmen kann, desto besser kann sie ihren Pluspunkt, eine höhere Winkelgeschwindigkeit innezuhaben, nutzen. Ihr Zwangsaufenthalt im See verkürzt sich entscheidend.



(a) $v_2 = 0.5$ (b) $v_2 = 0.9$

Abbildung 32: Trajektorien mit unterschiedlichem v_2

Wie deutlich zu erkennen ist, verfolgt die Frau in Abb. 32(b) nicht mehr die Strategie erst gen Mittelpunkt des Sees zu schwimmen, sondern versucht aufgrund ihrer hohen (Winkel-)Geschwindigkeit sofort das Ufer zu erreichen.

4.2.3 Nicht optimales Verhalten von Spieler P

Nun soll Spieler P keine optimale Strategie mehr verfolgen, was hier hieß, dass er die Zeit der Frau im See zu maximieren versuchte. Sondern er soll ebenfalls über seinen Input, die Störung w, minimieren wie seine Gegenspielerin über u. Genau genommen soll er immer dasjenige w wählen, das der Frau ermöglicht, mit geringsten Kosten, d.h. in geringster Zeit, das Ufer zu erreichen.

Abbildung 33 zeigt eine kleine Auswahl von Startwerten und den dazugehörigen Trajektorien. Auf die achsensymmetrischen Spiegelbilder wurde wohlweislich verzichtet, da Algorithmus 2.13, wie oben schon erwähnt, die Symmetrie ohnehin verhindert.

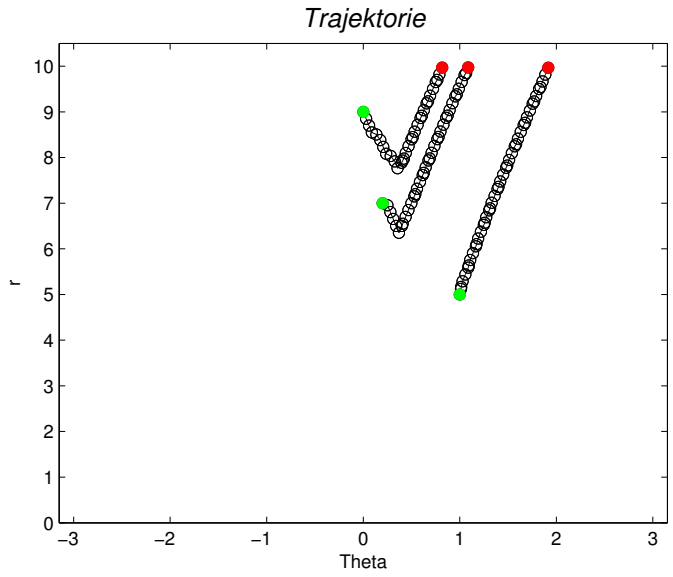


Abbildung 33: Trajektorien bei kooperativem Spielverhalten von Spieler P, $v_2 = 0.5$

Aber im Vergleich zu Abbildung 31 und 30a) reichen diese wenigen Werte bereits aus, um den Effekt des strategischen Entgegenkommens von Spieler P zu sehen. Die Frau kann das Ufer nun eher erreichen. Das tut sie nicht unbedingt weiter von ihm entfernt, weil lediglich die Zeit ausschlaggebend ist.
Dieses Resultat ist für den folgenden Abschnitt von Bedeutung. Denn dem Ort, an dem die Frau das Ufer erreicht, soll nun eine Relevanz zugesprochen werden.

4.3 Wertefunktion und Trajektorien für g_2

4.3.1 Wertefunktion

Bisher wurde unterstellt, dass der genaue Ort, an dem die Frau den Rand des Sees erreicht, nicht wichtig ist, von Interesse war ausnahmslos ihre Flucht. Diese Annahme wird nun aufgehoben. Dafür wird die Implementierung des Algorithmus 2.13 entsprechend angepasst. Anstatt jeder Zelle der Zielmenge $\mathcal{D}$ den Wert 0 zu übergeben, wird ihnen ein Wert zugewiesen, der einer Art Endkosten des Spiels entspricht. Die Flucht ist immer noch entscheidend, nur soll die Frau nun möglichst weit von dem Mann entfernt das Ufer erreichen.
Die Struktur der Implementierung des Dijkstra-Algorithmus bleibt an sich erhalten, da beim Vergeben der Endkosten im Hauptprogramm bereits auf eine ansteigende Reihenfolge bezüglich der Höhe des Wertes geachtet wird. Die Werte müssen in der Routine, in der 2.13 umgesetzt ist, den Zellen der Zielmenge nur noch zugewiesen werden.

Dazu wird neben g_1 für die laufenden Kosten die Kostenfunktion

$$g_2(\Theta, r, u) = \frac{10}{|\Theta|}$$

für die Modellierung der Endkosten gewählt.
Die Wahl von g_2 hat folgende Gründe:

(i) Erwünscht ist eine symmetrische Verteilung der Werte und zwar auf die Weise, dass für die Zellen des Seerandes = Zielmenge gilt:

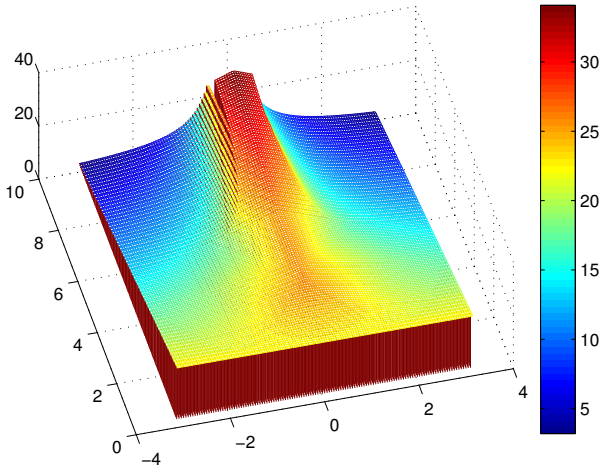
Abbildung 34: Symmetrische Verteilung der Kosten

mit K1 > K2 > K3 > ...

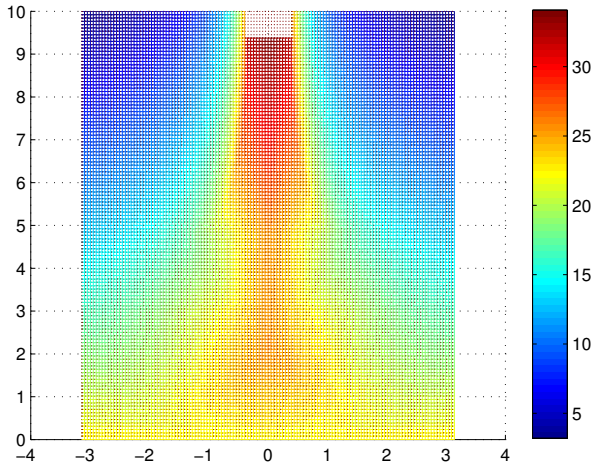
(ii) Die Endkosten sollen wertmäßig an die laufenden Kosten, die gleich dem Zeitschritt h gesetzt sind, angepasst werden. Damit ist gemeint, eine Art Gleichgewichtung zwischen den einzelnen Kostenarten herzustellen. Die Abbildungen der optimalen Wertefunktion werden später zeigen, dass sich der kritische Bereich von seinem Wert her in die restliche Wertefunktion „einschmiegen“ wird.

(iii) Der Abstand von der Frau zum Mittelpunkt des Sees, also r, soll keine gesonderte Rolle spielen, da lediglich die Zellen der Zielmenge mit den Kosten g_2 belegt werden.

Die optimale Wertefunktion weist weiterhin die Symmetrieeigenschaften aus Abschnitt 4.1 auf, nur sieht man jetzt deutlich einen Unterschied im Verhalten der im See schwimmenden Frau. Die Zellen, die dem Fangbereich am entferntesten liegen, weisen eindeutig kleinere Werte auf als die, die ihm am nächsten liegen.
Die Werte auf einer Kreisbahn (der kritische Bereich bleibt wieder einmal unberücksichtigt) entsprechen sich hier nicht allesamt, was aus dem Verhalten der beiden Spieler resultiert. Die Frau möchte die Kosten minimieren, der Mann jedoch maximieren. Er versucht alles, damit die Frau in seiner Nähe das Ufer erreicht und dort sind die Kosten, die aus g_2 stammen, wesentlich höher. Alle Startwerte garantieren gemäß der Modellierung die Flucht, aber der Einfluss von g_2 macht sich bereits in der Approximation der optimalen Wertefunktion stark bemerkbar, wie die Abbildungen 35 bis 37 zeigen sollen:

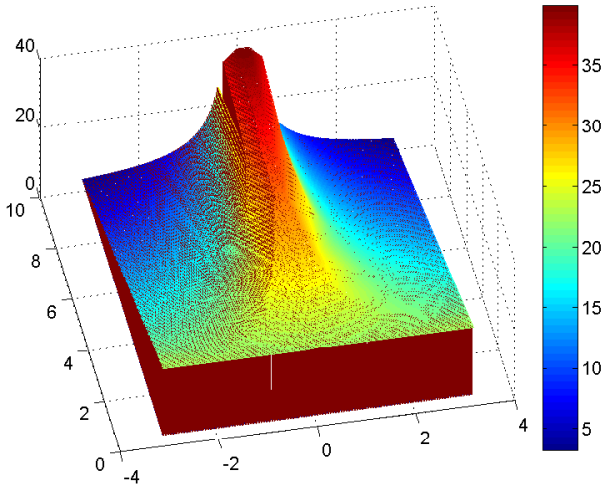


a) 3D

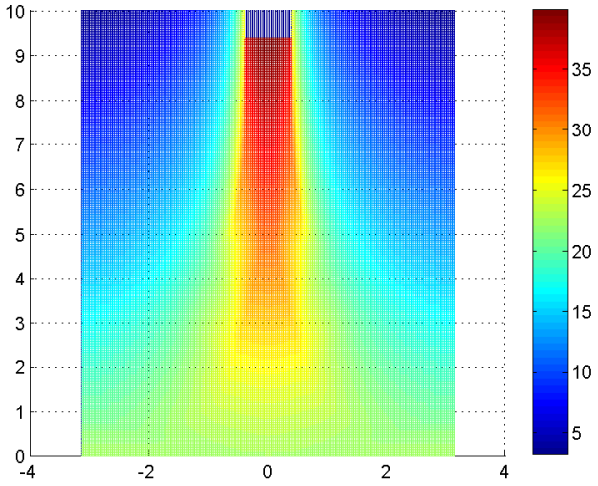


b) 2D

Abbildung 35: Wertefunktion mit 128 Zellen

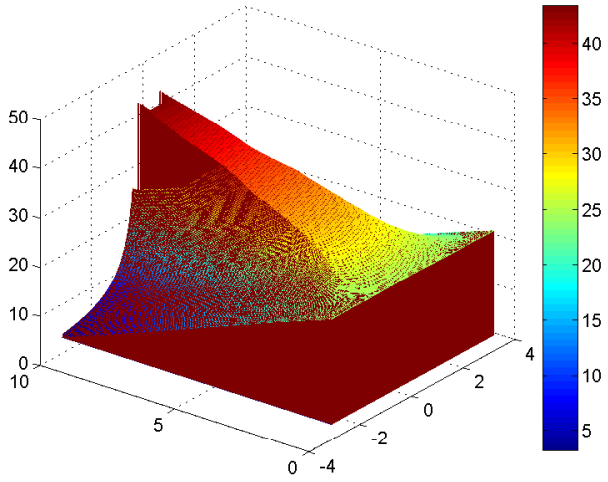


a) 3D

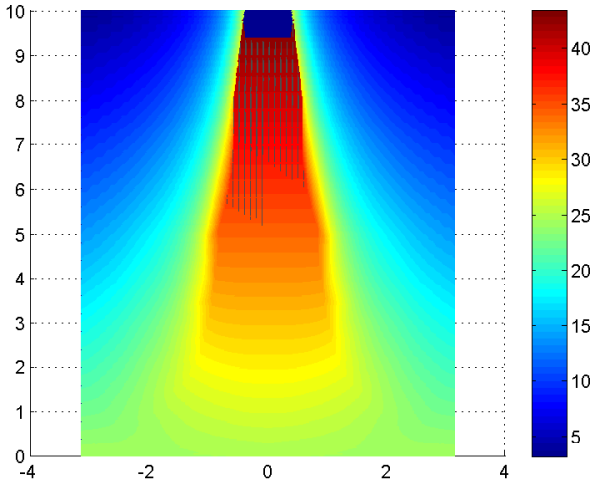


b) 2D

Abbildung 36: Wertefunktion mit 256 Zellen



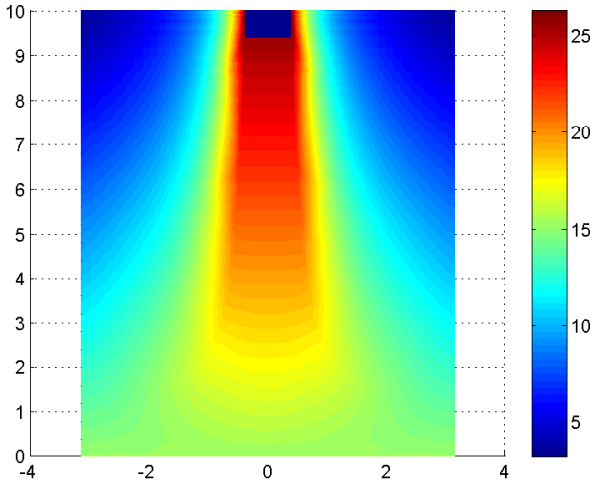
a) 3D



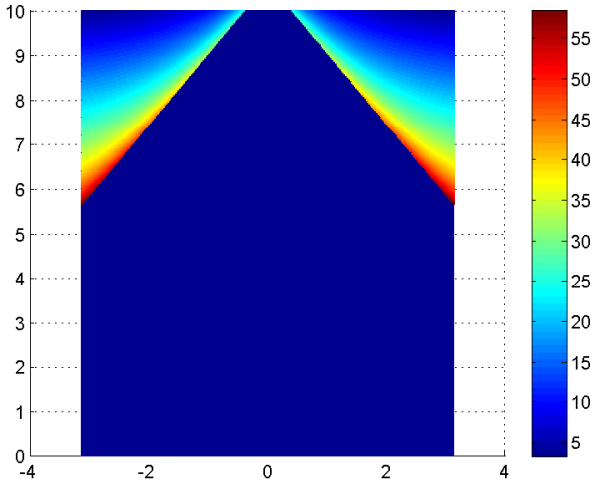
b) 2D

Abbildung 37: Wertefunktion mit 512 Zellen

Die blauen Bereiche beinhalten all die Startwerte, für die das System sehr kostengünstig in die Zielmenge gesteuert wird.
Auch bei Änderung von v_2 sind genau diese Werte von Bedeutung. Abbildung 38b) zeigt unwiderruflich, dass die Einführung von g_2 Konsequenzen für Spieler E hat. Die Frau ist mit $v_2 = 0.1$ sehr langsam und kann daher nur für Startwerte nahe des Ufers wirklich die Flucht ergreifen. Für alle anderen Ausgangssituationen muss sie durch die viel schnellere Fortbewegungsmöglichkeit des Mannes endlos im See verharren. Dies widerspricht zwar der anfangs getroffenen Annahme, dass das Spiel immer terminiert, doch diese kann man leicht wiederherstellen, indem man ihre Unfähigkeit der Flucht so interpretiert, dass sie sich letztendlich doch fangen lässt. Die wahre Ursache für dieses Spielende sind aber die Diskretisierungsfehler. Sie äußern sich so stark, dass die Flucht nicht mehr erreicht werden kann, obwohl sie das Ziel der Modellierung war.



a) 0.9

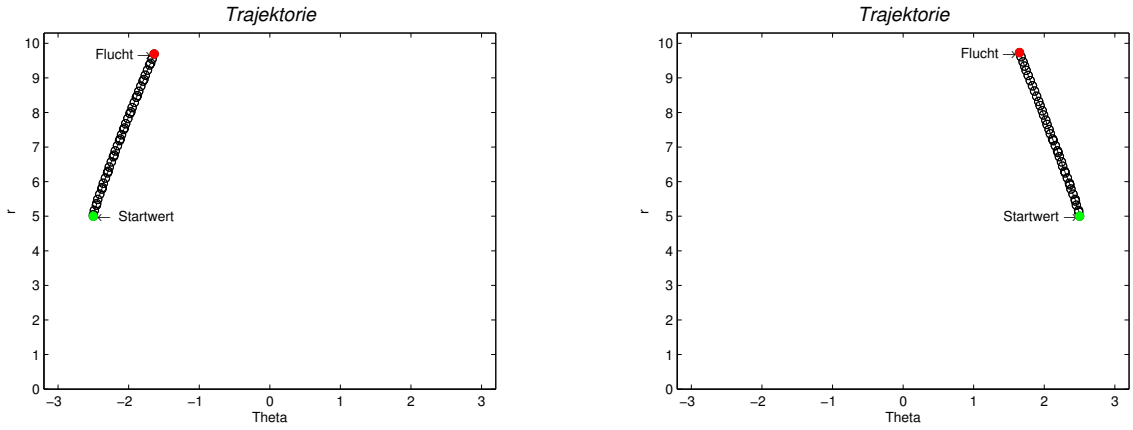


b) 0.1

Abbildung 38: Wertefunktion mit 512 Zellen

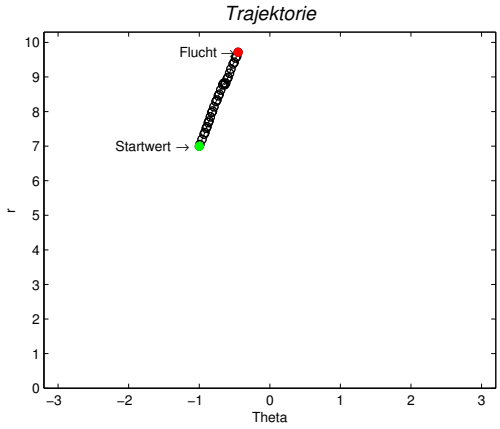
4.3.2 Trajektorien

Auch die Trajektorien haben sich gewandelt. Während in Abschnitt 4.1 die Frau noch geradewegs zum Ufer geschwommen ist, schafft es Spieler P durch seine Maximierung über w hier, dass es so aussieht, als schwimme sie auf ihn zu. Die Gewichtung des Randes ist ein Anreiz für ihn, dass sie ihm sehr nahe das Land erreicht.

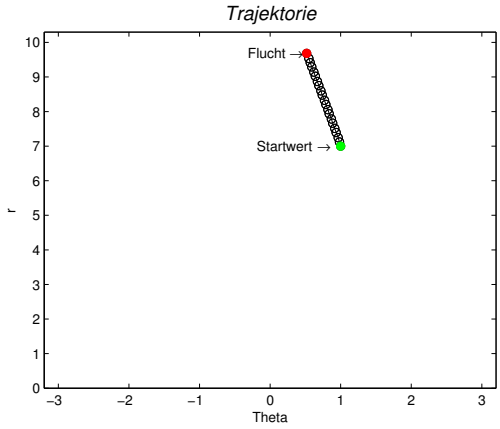


(a) Start (-2.5/5.0) (b) Start (2.5/5.0)

Abbildung 39: Trajektorien bei optimalem Spielverhalten I

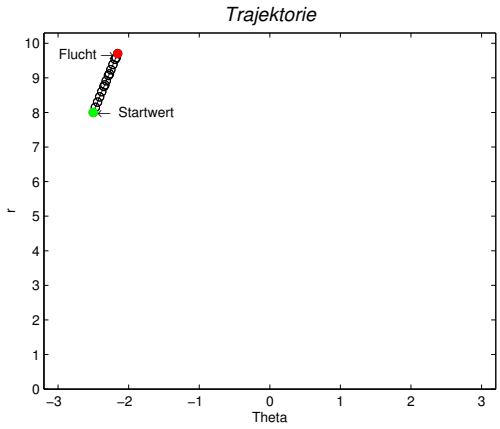


(a) Start (-1.0/7.0)

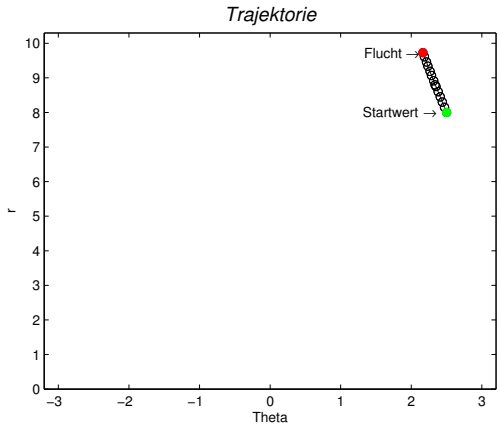


(b) Start (1.0/7.0)

Abbildung 40: Trajektorien bei optimalem Spielverhalten II



(a) Start (-2.5/8.0)



(b) Start (2.5/8.0)

Abbildung 41: Trajektorien bei optimalem Spielverhalten III

Für die einzelnen Berechnungen wurde $h = 0.3$ gewählt. Eine weitere Veränderung wurde bezüglich der Wahl der Diskretisierung des Störraumes W vorgenommen. Der Parameter w konnte nur die Werte -1, 0 und 1 annehmen.

4.3.3 Nicht optimales Verhalten von Spieler P

Was passiert, falls Spieler P nicht optimal spielt? Unterstellt wird ihm eine Strategie, dass er kein Interesse mehr hat, die Frau zu fangen, da er sie ohnehin nicht erwischen kann. Darum agiert er so, dass er es ihr so leicht wie möglich macht, das Ufer mit geringen Kosten zu erreichen.
Für die Veranschaulichung wurde eine Reihe von Startwerten ausgewählt. Aufgrund der

Gewichtung des Randes wird nicht wie in 4.2.3 auf die Spiegelbilder verzichtet, da 2.13 der Symmetrie nicht mehr entgegenwirken kann.

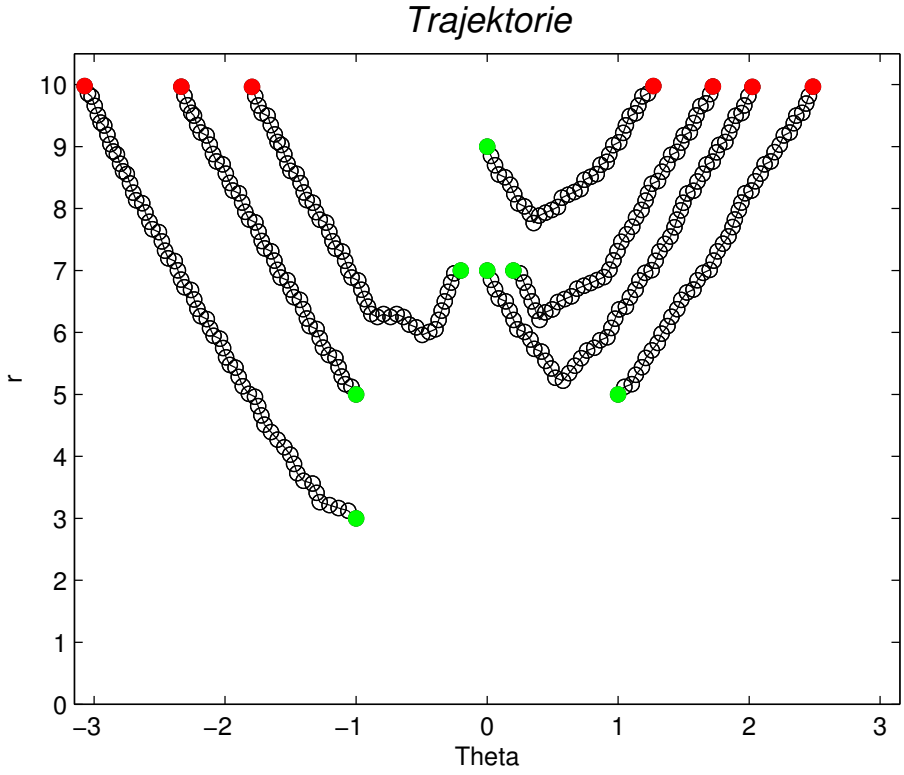


Abbildung 42: Trajektorien bei kooperativem Spielverhalten von Spieler P, $v_2 = 0.5$

Aufgrund von Diskretisierungsfehlern ist eine exakte Symmetrie der Trajektorien in Abildung 42 dennoch nicht erhaltbar.
Von Startwerten mit kleinerem r wird abgesehen, da die Trajektorien – $-\pi$ und π implizieren dieselbe Richtung, sie entsprechen sich – beispielsweise von (π, r) bei $(-\pi + \alpha, r + \beta)$, α, β hinreichend klein, fortgesetzt werden und damit die Veranschaulichung „unschön“ machen.
Im Vergleich zu Abbildung 33 erreicht die Frau in der Tat das Ufer mit einem größeren Winkel zwischen sich und dem Mann.

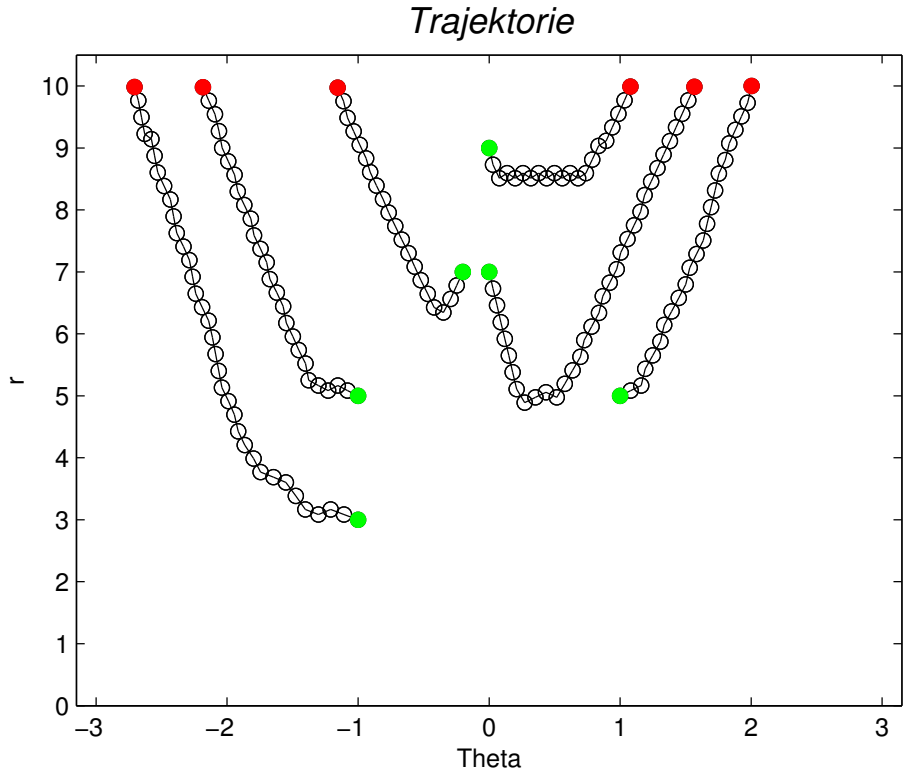


Abbildung 43: Trajektorien bei kooperativem Spielverhalten von Spieler P, $v_2 = 0.9$

Im Fall einer schnelleren Fortbewegungsmöglichkeit der Frau gibt es zwischen den Zuständen x_k und x_{k+1}, $k \in \mathbb{N}$, größere Abstände, wie Abbildung 43 zeigt. Darüberhinaus erreicht sie das Ufer eher, aber dafür dem Mann etwas näher. An der Tatsache ihrer Flucht ändert sich abermals nichts.

4.4 Approximation der Wertefunktion von oben

Wie in Kapitel 3 kann auch für „The Lady In The Lake“ eine Approximation der optimalen Wertefunktion von oben vorgenommen werden. Diese wird erneut über die Supremumsbildung über die Testpunkte einer Zelle durchgeführt. Daher ist zu vermuten, dass insbesondere die Intervallgrenzen des Inputraumes $W = [-1, 1]$ eine Rolle für die Berechnung von $V_{\mathcal{P}}$ spielen. Nur mit $w = 1$ bzw. $w = -1$ kann Spieler P seine Gegenspielerin schnellstmöglich jagen. Diese Vermutung bestätigt sich in der Tat. Die Wahl der Testpunkte -1, 0 und 1 für den Input w des Systems reicht vollkommen aus, um repräsentierbare Ergebnisse zu erhalten. Eines ist allerdings in der Implementierung unbedingt zu beachten: die Zielmenge muss durch zusätzliche Zellenschichten verstärkt werden, was für die Annäherung von unten nicht nötig war!

a) g_1

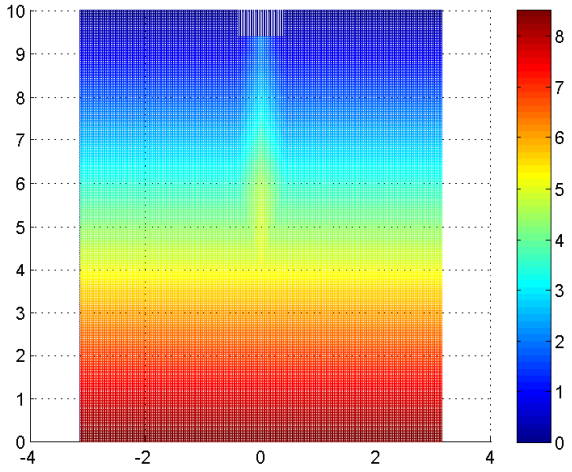


Abbildung 44: Annäherung der optimalen Wertefunktion von unten, g_1

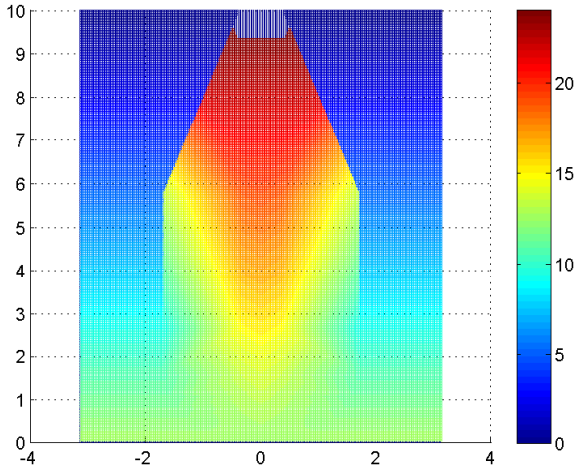


Abbildung 45: Annäherung der optimalen Wertefunktion von oben, g_1

Legt man dem System lediglich die Kosten g_1 zugrunde, so vergrößert sich der kritische Bereich zwar enorm, aber die in 4.1.1 diskutierte Struktur der optimalen Wertefunktion ist weiterhin gut erkennbar. Alle Startwerte auf einer Kreisbahn – mit Ausnahme des kritischen Bereichs – weisen denselben Wert auf. Der Fangbereich von Spieler P wirkt sich auf das Verhalten der Frau aus und folgert demnach größere Werte.
Nimmt man nun noch die Endkosten g_2 hinzu, so sieht man schön die Art „Rand“, der um den kritischen Bereich entsteht. Die höheren Kosten beim Erreichen des Ufers wirken

sich gesamt auf die Werte und das Aussehen der Wertefunktion aus.

b) g_2

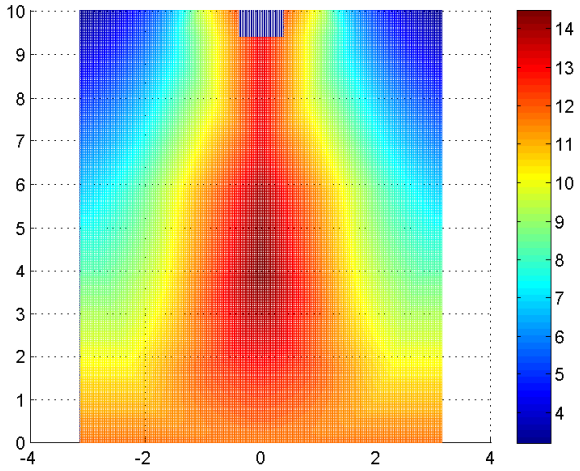


Abbildung 46: Annäherung der optimalen Wertefunktion von unten, g_1 und g_2

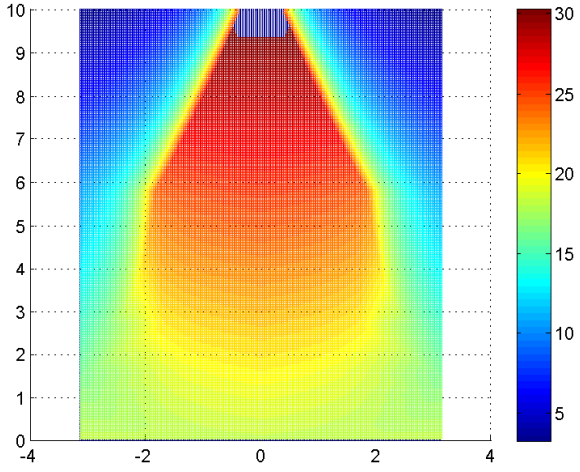


Abbildung 47: Annäherung der optimalen Wertefunktion von oben, g_1 und g_2

4.5 Ergebnisse und Vergleich mit der Theorie nach Isaacs

Auch für „The Lady In The Lake“ soll ein Vergleich mit den analytischen Untersuchungen des Differentialspiels aus [2] stattfinden. Ein wichtiger Aspekt der dort festgestellten Resultate ist bereits mehrmals als Argumentation für das Aussehen der optimalen Trajektorien verwendet worden: die größere Winkelgeschwindigkeit der Frau. Es kann gezeigt werden, dass diese Behauptung immer dann wahr ist, wenn $r < Rv_2$ ist, also wenn der Abstand der Frau zum Seemittelpunkt durch eine Größe, die von ihrer Geschwindigkeit abhängt, begrenzt ist. In Abbildung 32(b) ist schön zu sehen, dass die Frau, wenn sie sich mit konstantem $v_2 = 0.9$ fortbewegt, mit Startwert innerhalb des Kreises mit Radius Rv_2 nicht erst gen Zentrum des Sees schwimmen muss, um entkommen zu können. Ihre Flucht erfolgt „auf direktem Weg“ zum Ufer. Das bedeutet, sie kann ihre Fähigkeit, schneller als ihr Gegner die Richtung ändern zu können, in der Weise ausnutzen, keine Art Umweg über den Seemittelpunkt in Kauf nehmen zu müssen und kann damit die Terminierung des Spiels recht zügig herbeiführen.

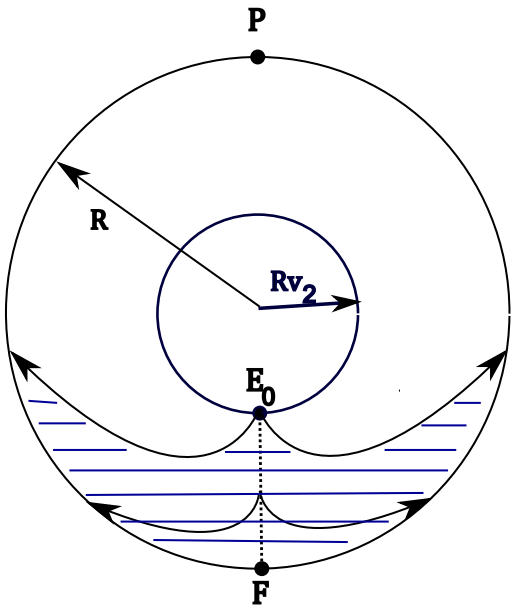


Abbildung 48: Verhalten der Spieler nach [2]

Im ursprünglichen Spiel, das auch in dieser Form in [2] behandelt wurde, versucht Spieler P den Winkel zwischen sich und seiner Gegenspielerin zum Zeitpunkt der Terminierung zu minimieren, während sie ihn maximieren möchte. Dieser Winkel $\Theta(T)$ lässt sich mittels einiger analytischer Vorgehensweisen wie folgt abschätzen:

$$|\Theta(T)| = \pi + \arccos v_2 - \frac{1}{v_2}\sqrt{1 - v_2^2} \tag{12}$$

Damit ergibt sich eine untere Schranke für die Geschwindigkeit von Spieler E,

$$v_2 > 0.21723...,$$

die für jeden Startwert die Flucht der Frau impliziert.
Dieses Resultat lässt sich für das Spiel mit den beiden in dieser Arbeit verwendeten Modellierungen nicht wiederfinden. Obgleich Abbildung 38b) zeigt, dass nicht mehr alle Anfangszustände zur Flucht führen, so ist dies mit den Fehlern aus den Diskretisierungen zu begründen. Vor allem die räumliche Diskretisierung ist für das irreführende Aussehen der optimalen Wertefunktion verantwortlich. Beide Modellierungen haben stets die Flucht der Frau modelliert, unabhängig davon, ob dem Rand des Sees nun eine Bedeutung zugemessen worden ist oder nicht.

Abbildung 48 grenzt zwei Bereiche voneinander ab. Jeder Startwert aus dem nicht schraffierten Bereich impliziert denselben Wert für V und zwar den, der durch (12) gegeben ist. Der Frau wird hierzu aber ein Informationsvorteil zugeschrieben. Sie weiß, wie ihr Widersacher zur selben Zeit wie sie agiert, und kann entsprechend darauf reagieren. Diese Annahme verhält sich genau gegensätzlich zur nichtantizipativen Strategie β des Spielers P.
Die optimalen Trajektorien des schraffierten Bereichs berechnen sich über die Isaacs-Gleichung und sind nur gültig, wenn der Endwinkel zwischen den Spielern den Wert π beträgt, die Frau das Ufer exakt dem Mann gegenüber erreicht. Für alle Anfangszustände auf der Strecke $\overline{E_0F}$ ist es an dem Mann zu entscheiden, ob er nach rechts oder nach links läuft. Etwas Wichtiges ist noch zu erwähnen: Auch wenn die Frau einen Weg eingeschlagen haben sollte, der sie in die Arme ihres Widersachers treiben würde, so bleibt ihr immer die Option zurück zum Seemittelpunkt zu schwimmen und von dort aus ihre Strategie zu überdenken. Da die ursprüngliche Kostenfunktion $g(\Theta, r) = |\Theta(T)|$ nur das Ufer des gesamten Zustandsraums, der See, mit Kosten belegt, erhöht sich der Wert von V nicht, selbst wenn sie mehrere Anläufe bräuchte, um letztlich fliehen zu können. Dies konnte in der Modellierung aus Abschnitt 4.2 nicht eintreten, da zum einen laufende Kosten in die Berechnung der optimalen Wertefunktion eingehen, zum anderen es an der Frau liegt, die Zeit bis zur Terminierung des Spiels zu minimieren.

Dass keine eindeutigen Parallelen in den beiden Untersuchungen zu verzeichnen sind, liegt bereits an der Umbenennung der Parameter u und w und damit am Ändern des Differentialgleichungssystems. Durch das Modellieren der Flucht und das Behandeln des Spiels aus Sicht der Frau ist das ursprüngliche Ziel zweckentfremdet worden. Das Ziel des Fangs wurde zum Wunsch der Flucht. Die Frau trat als Minimierer auf, während

der Mann die Rolle des Maximierers innehatte. Die Spieler haben sozusagen ihre Rollen getauscht.

5 Zusammenfassung der Ergebnisse und Ausblick

Die letzten beiden Kapitel haben gezeigt, wie wirkungsvoll die Wahl einzelner Parameter sein kann. Das Aussehen der optimalen Wertefunktion und der optimalen Trajektorien ist abhängig von den Rahmenbedingungen, die an das System gestellt werden. Variierte man beispielsweise die Geschwindigkeit v_2 (des Passanten in Kapitel 3 respektive der Frau in Kapitel 4), so konnte gesehen werden, dass selbst diese kleine Änderung weitreichende Folgen haben konnte. Insbesondere Abbildung 38 macht deutlich, dass der Einfluss eines einzigen Parameters entscheidend sein kann. Während für größere Werte von v_2 noch für alle Anfangswerte die Flucht der Frau garantiert werden konnte, so musste diese Forderung für sehr kleines v_2 aufgegeben werden. Selbst eine optimale Steuerfolge $\boldsymbol{u}$ konnte die Frau nicht vor ihrem Widersacher retten.
Doch nicht nur die Parameter, die das Differentialgleichungssystem direkt beeinflussen, können sich stark auf das Resultat der optimalen Wertefunktion auswirken, auch die Wahl der Testpunkte aus U und W kann von großer Bedeutung sein. Entnimmt man dem Kontroll- bzw. dem Störraum viele Testpunkte, so wird der Hypergraph, der vor Ablauf des Algorithmus 2.13 aufgebaut wird, enorm groß und die Berechnungen dauern entsprechend lange. Zudem wird der Speicher eines Rechners ziemlich gefordert. Würde man sich im Vorfeld überlegen, welche Kontrollinputs und Störinputs überhaupt vorkommen können (hierzu sollte man auch an andere Kontrollsysteme als die in dieser Arbeit behandelten denken), so könnte man vielleicht die Anzahl der Testpunkte in der Weise optimieren, dass der Hypergraph eine bestimmte Größe nicht übersteigt und die Berechnung der optimalen Wertefunktion schneller vonstatten geht.
Überhaupt ist ein Überdenken der Modellierung von „The Lady In The Lake“ erforderlich. Der gewählte Zustandsraum $X = [-\pi, \pi] \times [0, R]$ bleibt ein wenig heikel, obgleich die Berechnungen schöne Resultate ergeben haben. Den Seemittelpunkt als Strecke aufzufassen kann eine mögliche Fehlerquelle bedeuten. Auch wenn die in dieser Arbeit vorkommenden Trajektorien regulär sind, so kann es eventuell Singularitäten geben. Das System wurde nicht speziell auf solche überprüft. Um sie jedoch von Beginn an zu vermeiden, wäre an der Wahl des Zustandsraumes anzusetzen. Dabei sollte aber die Kompatibilität mit der gestörten Variante des Dijkstra-Algorithmus nicht außer Acht gelassen werden.
Ein weiterer Ansatz für die zukünftige Forschung betrifft die Kostenfunktion $g(x, u)$. In beiden behandelten Beispielen ist g so gewählt worden, dass keine Abhängigkeit von der Steuerung u gegeben war. Wie kann eine geeignete Kostenfunktion gewählt werden, dass die Bewegungen des Chauffeurs im „Homicidal Chauffeur Game“ und der Frau in „The Lady In The Lake“ mit in die Funktion eingehen?

A Inhalt der CD

Die CD enthält alle in der Arbeit erwähnten und verwendeten Programme. Im Ordner „C“ finden sich alle für die Berechnung der optimalen Wertefunktion und der optimalen Trajektorien nötigen C-Programme. Der Ordner „Matlab“ hält alle Routinen für die graphische Veranschaulichung von $V_{\mathcal{P}}$ und $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ sowie alle in der Arbeit vorkommenden und darüberhinaus weitere illustrative Beispieldatensätze parat. Im Unterordner „Beispieldaten_HCG“ des Ordners „Matlab“ finden sich alle Daten für das dynamische Spiel „Homicidal Chauffeur Game“ und im Unterordner „Beispieldaten_LITL“ alle Daten für das Spiel „The Lady In The Lake“ wieder.

Aufruf für die Berechnung von $V_{\mathcal{P}}$ und/oder $\boldsymbol{x} = (x_k)_{k\in\mathbb{N}}$ im Ordner „C“

Der Berechnung der optimalen Wertefunktion geht eine wichtige Routine voraus:

Entweder

`erzeuge_hkanten(...)`,

um eine Approximation der exakten optimalen Wertefunktion von unten,
oder

`erzeuge_hkanten_min_max_max(...)`,

um eine Approximation von oben zu erhalten.

$V_{\mathcal{P}}$ lässt sich dann mittels

`ausgabe_wertefunktion(...)`,

ausgeben.

Diese Optionen werden im Hauptprogramm `hcg.c` reguliert. Dort hat man zudem die Wahl, welche Strategie die Spieler verfolgen sollen.
Für das Spiel „Homicidal Chauffeur Game“ kann zwischen

`trajektorie_hcg_opt(...)`,
`trajektorie_hcg_zufall(...)`,

und

`trajektorie_hcg_kooperativ(...)`

gewählt werden. Dabei bezieht sich das Adjektiv „opt“, „zufall“ bzw. „kooperativ“ auf den Spieler E, also auf den Passanten.

Für das Spiel „The Lady In The Lake“ kann zwischen

`trajektorie_litl_opt(...)`

und

`trajektorie_litl_kooperativ(...)`

gewählt werden. „opt“ bzw. „kooperativ“ bezieht sich hier auf Spieler P, also auf den den See umrundenden Mann.

Die Programme `grid.c` bzw. `grid.h` enthalten unter anderem die Routinen für die Ausgabe der Wertefunktion und der Trajektorien. `hypergraph.c` bzw. `hypergraph.h` enthalten die Implementierung des Dijkstra-Algorithmus,

`dijkstra(...)`,

sowie seine gewichtete Fassung für die Modellierung der Endkosten g_2 aus „The Lady In The Lake“,

`dijkstra_gewichtet(...)`.

In `dopri5.c` bzw. `dopri5.h` ist das Runge-Kutta-Verfahren für die zeitliche Diskretisierung der Differentialgleichungssysteme, die die Spiele mathematisch beschreiben, realisiert.

Um die Berechnungen zu starten, ist dank des `makefiles` in der Konsole nur der Befehl

`make hcg`

bzw.

`make litl_`g_1 oder `make litl_`g_2

und nach dem Compilieren

`hcg`

bzw.

`litl_`g_1 oder `litl_`g_2

notwendig.

Graphische Ausgabe von $V_{\mathcal{P}}$ und/oder $\boldsymbol{x} = (x_k)_{k\in\mathbb{N}}$ im Ordner „Matlab“

Alle implementierten Matlab-Files wurden bereits in Kapitel 2.6 erläutert.
Während

`wertefunktion.m`

der Veranschaulichung der gewonnenen Daten der optimalen Wertefunktion dient, sind alle weiteren Programme für das Visualisieren der optimalen Trajektorien geschrieben worden. `trajektorie3` sollte statt `trajektorie2` bzw. statt `trajektorie` für „The Lady In The Lake“ aufgrund der angepassten Achsenbeschriftungen und -abmessungen verwendet werden. Ebenso verhält es sich für die Animation. Hierfür gibt es die zusätzliche Routine `animation3`.

Der Aufruf der einzelnen Funktionen erfolgt stets nach demselben Schema:

`program_name('.\Pfad\filename.dat')`

`wertefunktion('.\Beispieldaten_HCG\Annäherung_von_unten\hcg_0.3_0.5_128_3.dat')` plottet beispielsweise die Wertefunktion für das Spiel „Homicidal Chauffeur Game“ mit dem Zeitschritt $h = 0.3$, der konstanten Geschwindigkeit $v_2 = 0.5$, 128 Zellen je Koordinatenrichtung und 3 zusätzlich benötigten Zellenschichten um die Zelle, die den Nullpunkt enthält.

`trajektorie3('.\Beispieldaten_LITL\g_1\Annäherung_von_unten\litl_g1_opt_start_256_1.000000_5.000000_0.5.dat')` plottet beispielsweise die Trajektorie für das Spiel „The Lady In The Lake“ mit optimalen Strategieverhalten seitens Spieler P, dem Startwert (1.000000/5.000000), 256 Zellen je Koordinatenrichtung und der konstanten Geschwindigkeit $v_2 = 0.5$.

Abkürzungen

opt	-	optimal
zuf	-	zufällig
ko	-	kooperativ
start	-	Startwert/Anfangszustand
hcg	-	Homicidal Chauffeur Game
litl	-	The Lady In The Lake

Literatur

[1] L. Grüne and O. Junge. *Global optimal control of perturbed systems*. Technical Report, arXived at math. OC/0703874 (2006).

[2] T. Basar and G. Olsder. *Dynamic noncooperative game theory*. 2nd ed., SIAM, Philadelphia, 1999.

[3] R. Isaacs. *Differential Games*. Robert E. Krieger Publishing Company Huntington, New York, 1975.

[4] R. Diestel. *Graphentheorie*. 3.Auflage, Springer-Verlag, 2006.

[5] S.O. Krumke und H. Noltemeier. *Graphentheoretische Konzepte und Algorithmen*. 1.Auflage, Teubner-Verlag, 2005.

[6] G. Gallo, G. Longo, S. Nguyen and S.Pallottino. *Directed hypergraphs and applications*. Discrete Applied Mathematics, 42(2-3)(1993), 177-201.

[7] L. Grüne. *Numerik dynamischer Systeme*. Vorlesungsskript. http://www.uni-bayreuth.de/departments/math/ lgruene/numdyn0506/

[8] L. Grüne. *Numerische Dynamik von Kontrollsystemen*. Vorlesungsskript. http://www.math.uni-bayreuth.de/ lgruene/ndks04/

[9] O. Junge and H. Osinga. *A set oriented approach to global optimal control*. ESAIM : Control, Optimisation and Calculus of Variations 10 (2) (2004), 259-270.

[10] L. Grüne and O. Junge. *A set oriented approach to optimal feedback stabilization*. Systems & Control Letters, 54 (2) (2005), 169-180.

[11] M. v. Lossow. *Mengenorientierte optimale Steuerung und Fehlerschätzung*. Diplomarbeit, 2005.

[12] L. Grüne. *Mathematische Kontrolltheorie I: Lineare Systeme*. Vorlesungsskript. http://www.uni-bayreuth.de/departments/math/ lgruene/kontrolltheorie0506/ skript_kt1.pdf

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Bayreuth, den 21. Januar 2008 .....

(Diana Balbus)