Woche der Modellierung mit Mathematik

Dokumentationsbroschüre
05.02.-11.02.2012

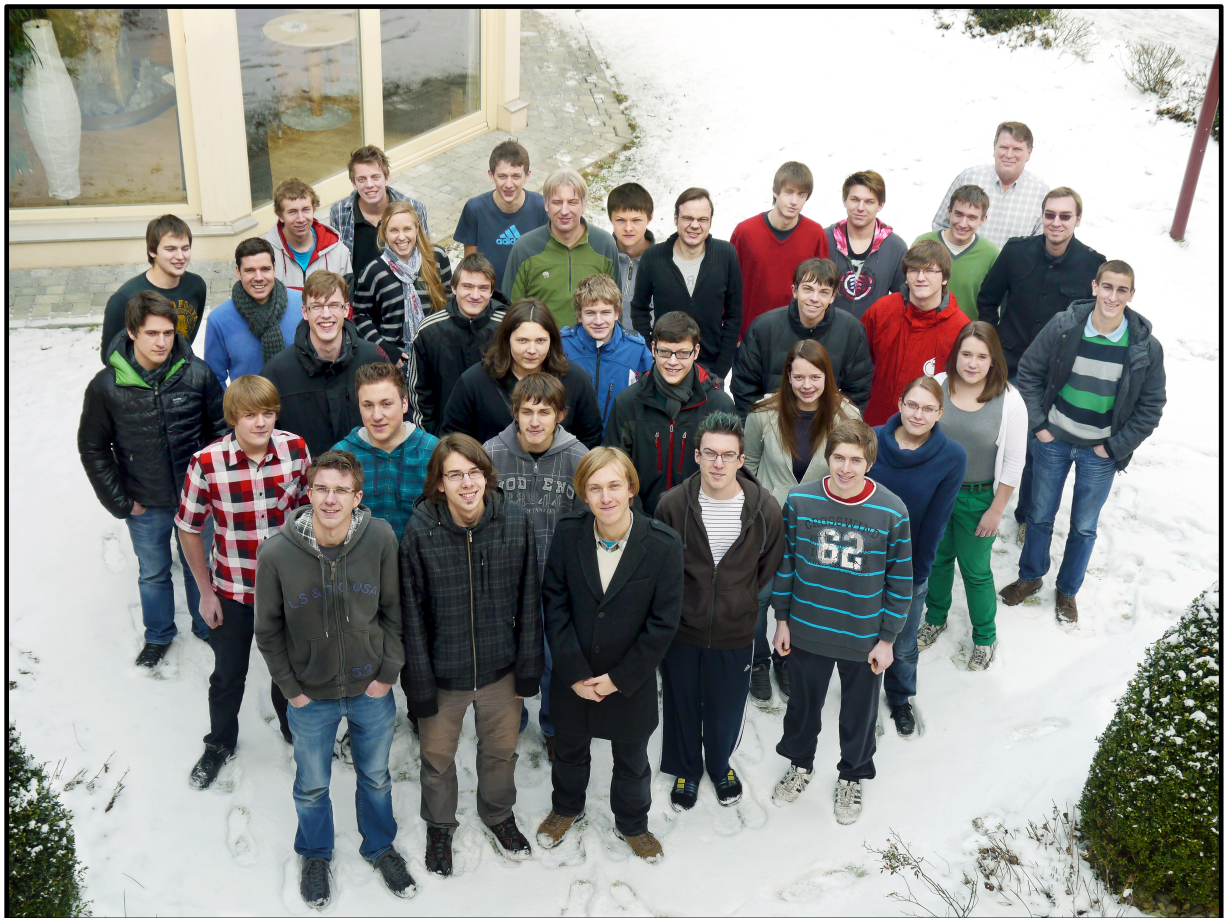
<we do awesome stuff>

Das Land Steiermark

UNI GRAZ

MATHEMATIK

WOCHE DER MODELLIERUNG MIT MATHEMATIK



PÖLLAU BEI HARTBERG, 05.02.-11.02.2012

WEITERE INFORMATIONEN:

HTTP://MATH.UNI-GRAZ.AT/MODELLWOCHE/2012/

Das Land Steiermark

UNI GRAZ

MATHEMATIK

Vorwort

Viele Wissenschaften erleben zurzeit einen ungeheuren Schub der Mathematisierung. Mathematische Modelle, die vor wenigen Jahrzehnten noch rein akademischen Wert hatten, können heute mit Hilfe von Computern vollständig durchgerechnet werden und liefern praktische Vorhersagen, die helfen, Phänomene zu verstehen, Vorgänge zu planen, Kosten einzusparen. Damit unsere Gesellschaft auch in Zukunft mit der technologischen Entwicklung schritthält, ist es wichtig, bereits junge Leute für diese Art mathematischen Denkens zu begeistern und in der Gesellschaft das Bewusstsein für den Nutzen angewandter Mathematik zu heben. Dies war für uns einer der Gründe, die Woche der Modellierung mit Mathematik zu veranstalten.

Nun ist leider für viele Menschen Mathematik ein Schulfach, mit dem sie eher unangenehme Erinnerungen verbinden. Umso erstaunlicher erscheint es, dass Schülerinnen und Schüler sich freiwillig melden, um eine ganze Woche lang mathematische Probleme zu wälzen - und dabei auch noch Spaß haben. Sie erleben hier offensichtlich die Mathematik auf eine Art und Weise, wie sie der Schulunterricht nicht vermitteln kann. Die jungen Leute arbeiten und forschen in kleinen Gruppen mit Wissenschaftler/innen an realen Problemen aus den verschiedensten Bereichen und versuchen, mit Hilfe mathematischer Modelle neue Erkenntnisse zu gewinnen. Sie arbeiten ohne Leistungsdruck, dafür mit Eifer und Enthusiasmus, rechnen, diskutieren, recherchieren, oft auch noch am späten Abend, in einer entspannten und kreativen Umgebung, die den Schüler/innen und betreuenden Wissenschaftler/innen gleichermaßen Spaß macht. Die Projektbetreuer konnten auch in diesem Jahr wieder erleben, wie eigenes Entdecken und Selbstmotivation das Verhalten der Schüler/innen während der ganzen Modellierungswoche bestimmen. Sie lernen eine Arbeitsmethode kennen, die in beinahe allen Details den Arbeitsmethoden einer Forschergruppe entspricht. Bei keiner anderen Gelegenheit erfahren Schüler/innen so viel über Forschung wie bei so einer Veranstaltung.

Modellierungswochen gab bzw. gibt es zum Beispiel auch in den USA, in Deutschland oder in Italien. Wir verdanken Herrn Prof. Dr. Stephen Keeling den Vorschlag, auch durch die Universität Graz so eine Woche zu veranstalten, und seiner unermüdlichen Organisationsarbeit das tatsächliche Zustandekommen. Er leitet nun bereits zum achten Mal diese inzwischen zur Institution gewordene Veranstaltung. Ihm sei an dieser Stelle noch einmal ausdrücklich und herzlich gedankt. Besonders wichtig war in den vergangenen Jahren auch die Unterstützung durch den langjährigen Mentor der Modellierungswoche, Herrn o.Univ.-Prof. Dr. Franz Kappel, der oft auch eine eigene Gruppe mit interessanten Problemstellungen betreut hat.

Wir danken dem Landesschulrat für Steiermark, und hier insbesondere Frau Landesschulinspektorin Frau HR Mag. Marlies Liebscher, für die Hilfe bei der Organisation und ihre kontinuierliche Unterstützung der Idee einer Modellierungswoche. Ohne den idealistischen, unentgeltlichen und engagierten Einsatz der direkten Projektbetreuer Dr. Kristian Bredies, Dr. Frank Heyde, Daniel Kraft, BSc BSc und Mag. Stefan Fürtinger – alle Institut für Mathematik und Wissenschaftliches Rechnen – hätte diese Modellierungswoche nicht stattfinden können.

Besonderer Dank gebührt ferner Herrn Mag. Dr. Christoph Gruber, der die ganze Veranstaltung betreut und auch die Gestaltung dieses Berichtes übernommen hat, Frau Michaele Seiwald für die tatkräftige Hilfe bei der organisatorischen Vorbereitung, und Herrn Alexander Sekkas für die Hilfe bei der Betreuung der Hard- und Software.

Finanzielle Unterstützung erhielten wir vom Land Steiermark durch Landesrätin Mag. Kristina Edlinger-Ploder und von der Karl-Franzens-Universität Graz durch Vizerektor Prof. Dr. Martin Polaschek und Dekan Prof. Dr. Karl Crailsheim.

Pöllau, am 10. Februar 2012

Bernd Thaller
Institut für Mathematik und Wissenschaftliches Rechnen
Karl-Franzens-Universität Graz

Modellierung des dynamischen Verhaltens von Fußgängern

Bereich: Sozialwissenschaften

Betreuer: Dr. Kristian Bredies

Erarbeitet von:

Julia Kraner, Klaus Haider, Michael Kaiser, Patricia Offerman,
Thomas Kunzfeld und Michael Gernot Sumper

Einleitung

In der modernen urbanen Welt gehören sich bewegende Menschenmassen zum Alltag. Zum Beispiel sind sie in der Fußgängerzone, im Einkaufszentrum, bei einem Konzert, in einem Stadion oder auf einer Messe anzutreffen. Obwohl sich jeder Mensch als Individuum anders verhält, erfolgt die Bewegung von Fußgängern nach Gesetzmäßigkeiten und organisiert sich häufig selbst in bestimmten Mustern. Engpässe, Kreuzungen oder Durchgänge können gefährliche Stauung von Menschen verursachen, ebenso kann eine Notfallsituation zu Massenpanik führen.

Mit diesem Hintergrund stellt sich die Frage, wie die Dynamik von Fußgängern erfolgt, wodurch sie beeinflusst wird und wie sie mathematisch beschrieben werden kann. Ziel dieses Projektes ist es daher einerseits, ein Modell zur Abbildung der Bewegung zu entwerfen, und andererseits dieses Konzept in Form von experimentellen Computer-simulationen umzusetzen.

1. Die Fußgängerdynamik wird von verschiedenen Faktoren beeinflusst:

Umweltfaktoren

- Hindernisse
- Wetter
- Wegbreite
- Tageszeit
- Verkehr
- Andere Menschen
- Menschenströme

Physikalische Aspekte

- Geschwindigkeit
- Beschleunigung
- Richtung
- Position
- Größe des Individuums
- Gepäcksstücke

Persönlichkeitsaspekte

- Temperament
- Erfahrung
- Motivation: Ziel, Plan
- Bevorzugte Richtungen
- Egoismus
- Intelligenz vs. soziale Normen

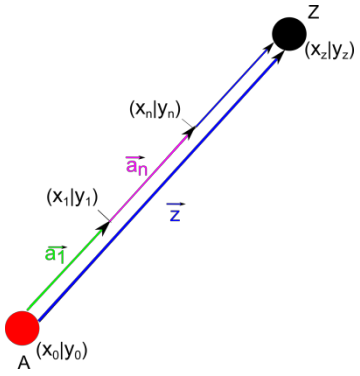
Auch Willkür und Zufall können über den konkreten Verlauf von Bewegungen entscheiden.

2. Mathematische Grundlagen für die Bewegung eines Individuums

Person A befindet sich am Startpunkt $(x_0|y_0)$ und will das Ziel$(x_z|y_z)$ erreichen. Deshalb bewegt sie sich im einfachsten Fall auf Luftlinie mit konstanter Geschwindigkeit darauf zu.

Von $(x_0|y_0)$ bewegt sie sich auf den Punkt $(x_1|y_1)$ zu und so weiter. Zum Zeitpunkt s_n befindet sich A an der Position $(x_n|y_n)$. $n \in \mathbb{N}$

Auch der Bewegungsvektor von A ist zeitabhängig. Wir bezeichnen ihn mit $\vec{a}_n = \begin{pmatrix} a_n^1 \\ a_n^2 \end{pmatrix}$



Der Subskript indiziert den Zeitpunkt *n* und der Superskript die Komponenten (x- und y-Wert) des Vektors.

Die Komponenten des Vektors werden für jede Zeiteinheit neu berechnet, sie genügen den Gleichungen:

$a_{n-1}^1 = \frac{x_n - x_{n-1}}{s}$ $\qquad$ $a_{n-1}^2 = \frac{y_n - y_{n-1}}{s}$

Der Weg von der aktuellen Position zum Ziel ist der Vektor $\vec{z}_n = \begin{pmatrix} x_z - x_n \\ y_z - y_n \end{pmatrix}$.

Die Länge des Vektors a_n entspricht in diesem Fall der Durchschnittsgeschwindigkeit $v_\emptyset$ eines Fußgängers. Durch Division mit seinem Betrag wird $\vec{z}_n$ ein Einheitsvektor. Nun kann er mit der gewünschten Durchschnittsgeschwindigkeit multipliziert werden, damit er diese Länge hat.

$$a_n = \frac{\vec{z}_n}{|z_n|} \cdot v_\emptyset$$

Durch Umformen der oben angeführten Gleichungen kann man die jeweils neue Position $(x_n|y_n)$ von A ermitteln. für i = 1, 2, 3, …. $\mathbb{N}$

$x_n = x_{n-1} + s \cdot a_{n-1}^1$ $\qquad$ $x_i = x_{i-1} + s \cdot a_{n-1}^1$

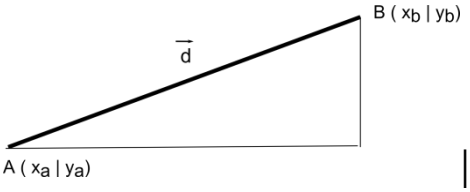
$y_n = y_{n-1} + s \cdot a_{n-1}^2$ $\qquad$ $y_i = y_{i-1} + s \cdot a_{n-1}^2$

3. Mathematisches Modell für die Bewegung des Individuums

Annahmen über die Bewegung einer Einzelperson werden zunächst in die mathematische Sprache übersetzt. Schrittweise werden diese aufgestellten Formeln und Bedingungen in die Programmiersprache *C++* übertragen, um dies zu simulieren. Die Zahl der sich bewegenden Objekte in der Simulation kann auch erhöht werden, um Menschenmassen darzustellen. Zur Vereinfachung werden die Bewegungen zweidimensional betrachtet und im Grundriss dargestellt.

3.1. Mindestabstand zwischen zwei Personen

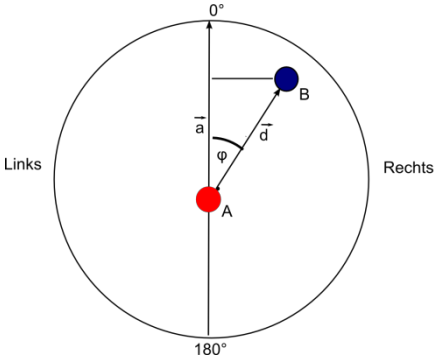
Fußgänger vermeiden es sich anzurempeln oder ineinanderzulaufen. Erste Regel für die Bewegung der Person A ist daher das Einhalten eines Mindestabstandes k_{safe} zu einer Person B. Der Betrag des Vektors $\vec{d}$ mit der Spitze B$(x_b|y_b)$ und dem Schaft A$(x_a|y_a)$ muss mindestens einen konstanten Wert annehmen.



$$|\vec{d}| = \left|\begin{pmatrix} x_b - x_a \\ y_b - y_a \end{pmatrix}\right| = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2} \geq k_{safe}$$

3.2. Befindet sich Person B rechts oder links von Person A?

Person A bewegt sich in Richtung des Vektors $\vec{a}$ während Person B steht. Der Vektor $\vec{d}$ verläuft von A nach B. Anhand des Winkels φ zwischen $\vec{a}$ und $\vec{d}$ kann ermittelt werden, ob sich Person B aus der Sicht von Person A links oder rechts befindet.



$$\cos\varphi = \frac{\vec{a} \cdot \vec{d}}{|\vec{a}| \cdot |\vec{d}|}$$

$$0 < \varphi < \pi \quad \Leftrightarrow \quad B\ liegt\ rechts$$

$$\pi < \varphi < 2\pi \quad \Leftrightarrow \quad B\ liegt\ links$$

3.3. Person A weicht Person B aus

Nachdem Person A festgestellt hat, ob sich Person B rechts oder links von ihr befindet, will A der Person B in die jeweils andere Richtung ausweichen.

$$B\ steht\ rechts \Leftrightarrow A\ weicht\ nach\ links\ aus$$

$$B\ steht\ links \quad \Leftrightarrow A\ weicht\ nach\ rechts\ aus$$

A orientiert sich beim Ausweichen zunächst an dem Vektor $\overrightarrow{d'}$. Dieser von B wegführende Richtungsvektor $\overrightarrow{d'}$ entsteht durch Spiegelung des Vektors $\vec{d}$ an der Achse, die von A in Richtung $\overrightarrow{a}$ ausgeht. Die Länge eines Vektors entspricht der Geschwindigkeit in die angezeigte Richtung. Beim Ausweichen beschleunigt A. Daher wird der Vektor $\overrightarrow{d'}$ entsprechend seiner Geschwindigkeit auf die richtige Länge gebracht. Daraus resultiert der Vektor $\overrightarrow{v_{avoid}}$. Neben der Ausweichrichtung $\overrightarrow{v_{avoid}}$. berücksichtigt A auch die Zielrichtung und wählt einen Kompromiss zwischen den beiden Richtungen, der durch Vektoraddition erreicht wird.

Die genannten mathematischen Maßnahmen des Ausweichens werden im Folgenden genauer erklärt.

Spiegelung des Vektors $\vec{d}$

Durch den Punkt A und seinen Bewegungsvektor $\vec{a}$ kann eine Gerade g aufgestellt werden. Sie dient als Spiegelachse, um den Punkt B zu spiegeln. Da ein fester Zeitschritt betrachtet werden soll, wird die Indizierung durch den Zeitschritt *n* weggelassen und die Komponenten des Vektors mit a_1 und a_2 bezeichnet.

$g\colon X = A + \sigma \cdot \vec{a} = \begin{pmatrix} x_a \\ y_a \end{pmatrix} + \sigma \cdot \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$

Sei der Vektor $\vec{p}$ derjenige, der die kürzeste Verbindung von B zu g darstellt.

$\vec{p} = \overrightarrow{BP} = \begin{pmatrix} x_p - x_b \\ y_p - y_b \end{pmatrix}$

Wobei der Projektionspunkt $P \in g$ ermittelt werden soll. Dieser zeichnet sich aus durch:

I. $\begin{pmatrix} x_p \\ y_p \end{pmatrix} = \begin{pmatrix} x_a \\ y_a \end{pmatrix} + \sigma \cdot \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$

II. $\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \cdot \begin{pmatrix} x_p - x_b \\ y_p - y_b \end{pmatrix} = 0 \qquad \vec{a} \cdot \vec{p} = 0 \Leftrightarrow \vec{a} \perp \vec{p}$

Durch Substitution erhält man: $\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \cdot \left[\begin{pmatrix} x_a \\ y_a \end{pmatrix} + \sigma \cdot \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} - \begin{pmatrix} x_b \\ y_b \end{pmatrix} \right] = 0$

Nach dem Ausmultiplizieren und dem Herausheben von a_1 und a_2 ergibt sich:

$$\sigma = \frac{a_1(x_b - x_a) + a_2(y_b - y_a)}{{a_1}^2 + {a_2}^2}$$

Der Zähler entsteht durch die skalare Multiplikation $\vec{a} \cdot \vec{d}$.

Der Nenner ist das Quadrat des Betrags von $\vec{a}$, denn $|\vec{a}| = \sqrt{{a_1}^2 + {a_2}^2}$.

$$\rightarrow \quad \sigma = \frac{\vec{a} \cdot \vec{d}}{|\vec{a}|^2}$$

Durch Einsetzen von σ in die Geradengleichung kann P ermittelt werden.

$$P = A + \frac{(\vec{a}\cdot\vec{d})}{|\vec{a}|^2}\cdot\vec{a}$$

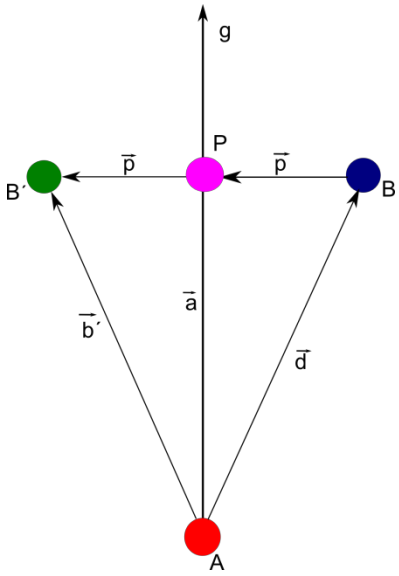
Durch P kann der gespiegelte Punkt B‘ ermittelt werden.

$$B' = P + \vec{p}$$

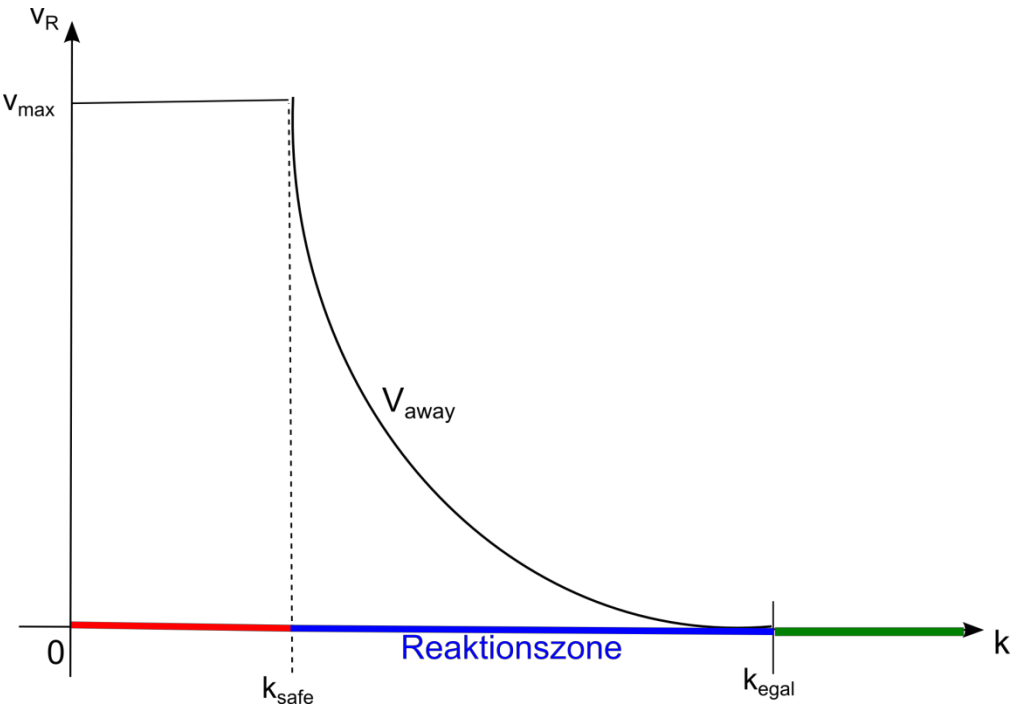
$$B' = A + \frac{(\vec{a}\cdot\vec{d})}{|\vec{a}|^2}\cdot\vec{a} + \vec{p}$$

Mit B‘ und A kann nun der gespiegelte Vektor $\overrightarrow{d'}$ aufgestellt werden.

$$\vec{d}' = B' - A = \frac{(\vec{a}\cdot\vec{d})}{|\vec{a}|^2}\cdot\vec{a} + \vec{p}$$



Änderung der Geschwindigkeit beim Ausweichen



Nähert sich A der Person B, ist die Reaktion von A das Ausweichen. Dabei beschleunigt A, je nachdem wie nahe er Person B bereits ist. Zum Geschwindigkeitsvektor von A, dargestellt durch $\vec{a}$, wird ein Geschwindigkeitsvektor addiert. Das Diagramm zeigt dessen Betrag v_R (Geschwindigkeit der Reaktion) in Abhängigkeit von dem Abstand k zwischen A und B.

Zwei für die Reaktion charakteristische Abstände sind die Werte k_{egal} und k_{safe}.

$$v_{away} = 0 \quad \Leftrightarrow \quad k \geq k_{egal}$$

$$v_{max} > v_{away} > 0 \quad \Leftrightarrow \quad k_{safe} < k < k_{egal}$$

$$v_{away} = v_{max} \Leftrightarrow 0 \leq k \leq k_{safe}$$

In verwendetem Modell kann v_{away} durch einen Viertelkreises beschrieben werden. Der Mittelpunkt dieses Kreises ist $M = (r - k_{safe} | r)$. Daraus ergibt sich durch Umformen nach V_R

$$V_R = r - \sqrt{r^2 - (k - r - k_{safe})^2}$$

$r = k_{egal} - k_{safe}$

Da v_R nicht unendlich groß werden kann, sondern maximal so groß wie v_{max} ,muss v_R noch normiert und mit v_{max} multipliziert werden.

$$v_R(k) = \frac{(k_{egal} - k_{safe}) - \sqrt{r^2 - (k - r - k_{safe})^2}}{k_{egal} - k_{safe}} \cdot v_{max}$$

Damit gilt: $v_R(k_{safe}) = v_{max}$

Anpassen der Länge des Vektors $\vec{d'}$

Der Vektor $\vec{d}'$ hat zunächst die Länge des Vektors $\vec{d}$, er muss jedoch noch auf die entsprechende Länge laut der Funktion V_R gebracht werden.

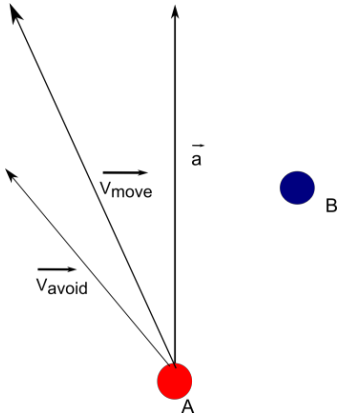
Deshalb dividiert man den Vektor $\vec{d}'$ zunächst durch seinen Betrag, damit er hat die Länge 1 hat und multipliziert ihn dann mit der gewünschten Länge. Daraus ergibt sich:

$\frac{\vec{d}'}{|\vec{d}'|} \cdot v_R(|\vec{d}|) = \vec{V}_{avoid}$

Aufstellen des tatsächlichen Vektors des Ausweichens

Neben der Vermeidung von körperlichem Kontakt mit anderen Fußgängern wird das Erreichen des individuellen Ziels berücksichtigt. Daher muss, um eine ideale

Bewegungsrichtung zu erreichen, in weiterer Folge ein Kompromiss zwischen der ursprünglichen Bewegungsrichtung und der neu berechneten Richtung $\vec{V}_{avoid}$ erfolgen.



$\vec{V}_{avoid} + \vec{a} = \vec{V}_{move}$

Der durch Vektoraddition entstandene Vektor $\vec{V}_{move}$ gibt tatsächliche Richtung und Geschwindigkeit des Ausweichens vor.

4.4. Sonderfall: B liegt annähernd auf der Achse *g*

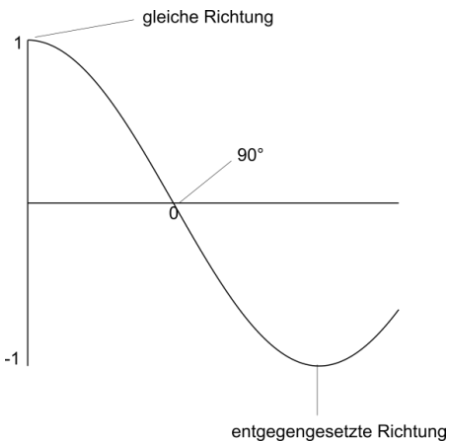
Wenn sich A zu sehr auf B zubewegt, ist der Abstand von B zur Achse g sehr klein. Befindet sich B auf der Achse g, so bewirkt die Spiegelung dieses Punktes B an der Achse keine Änderung. In diesen Fällen ist der Betrag von $\vec{p}$ sehr klein oder null. Dieses Problem wird gelöst, indem bei der Beschreibung von B‘ zwischen zwei Fällen unterscheiden wird..

$$B' = \begin{cases} A + \dfrac{\vec{a} \cdot \vec{d}}{|\vec{a}|^2} \cdot \vec{a} + \vec{p} & falls\ |\vec{p}| \geq p_{min} \\ A + \dfrac{\vec{a} \cdot \vec{d}}{|\vec{a}|^2} \cdot \vec{a} + \dfrac{\overrightarrow{n_a}}{|\overrightarrow{n_a}|} & falls\ |\vec{p}| \leq p_{min} \end{cases}$$

p_{min} bezeichnet dabei die Länge ab der B‘ die Spiegelung an der Achse realisiert.

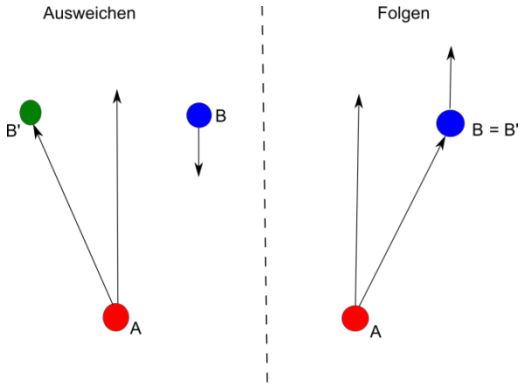
4.5. Person A folgt Person B

In der Realität kann man bei Menschenmassen, die sich in entgegengesetzte Richtungen bewegen, eine Strombildung in die jeweilige Richtung beobachten. Für die Darstellung in einem Modell soll daher zu nächst der Winkel zwischen den Vektoren $\vec{a}$ und $\vec{b}$ berechnet werden. Dafür benötigt man den Cosinus-Satz: $cos\varphi = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|}$



Im Graphen kann abgelesen werden, in welche Richtung sich A nach dem Graphen der Cosinus Funktion, in Bezug auf B bewegen soll.

Ist der Cosinus von φ gleich 1, sind die Richtungsvektoren gleich orientiert, d.h. A soll B folgen. Ist der Cosinus von φ gleich -1, bewegen sich A und B in entgegengesetzte Richtungen, daher weicht A nach vorherigen Überlegungen in Richtung von B‘ aus. Ist der Winkel zwischen den Richtungsvektoren gleich 90° bleibt die Bewegungsrichtung von A unverändert. Um die beiden Modelle des Ausweichens und des Folgens mit einander zu verbinden, sind weitere Überlegungen notwendig. Ist A nach dem vorherigen Modell in Richtung von B‘ ausgewichen, soll es sich nun abhängig vom Bewegungsvektor von B bewegen. Bewegt sich B in eine annähernd gleiche Richtung wie A, soll A seinen Bewegungsvektor an B orientieren, also B folgen.



Nebenstehend sind die beiden Annahmen graphisch gegenübergestellt.

Der Punkt C der in diesem Fall anstelle von B‘ angepeilt wird, soll auf der Verbindungsstrecke von B zu B‘ liegen. Dies lässt sich mathematisch durch eine Konvexkombination ausdrücken. Diese beschreibt einen beliebigen Punkt auf der Verbindungsstrecke zwischen B und B‘. Mathematisch bedeutet das, dass ausgehend vom Punkt B‘ ein Vielfaches des Vektors von B‘ nach B addiert wird, wobei dieses Vielfache λ zwischen 0 und 1 liegen soll:

$0 \leq \lambda \leq 1$

$C = \lambda \cdot (B - B') + B'$

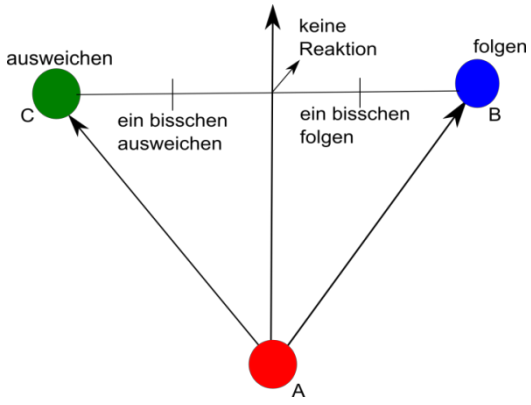
Daraus folgt für den Punkt C

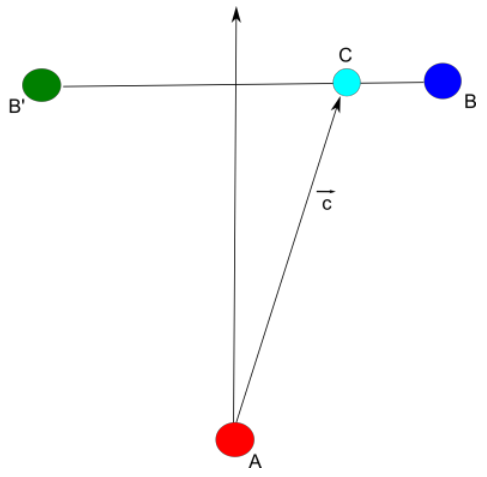
$C = \lambda \cdot B + (1 - \lambda) \cdot B'$

Dies ist besagte Konvexkombination.

Auch wird der Vektor von A nach C benötigt:

$\vec{c} = C - A$





Damit dies mit der Berechnung des Winkels zwischen den Bewegungsvektoren in Verbindung gebracht werden kann wird folgendes angenommen:

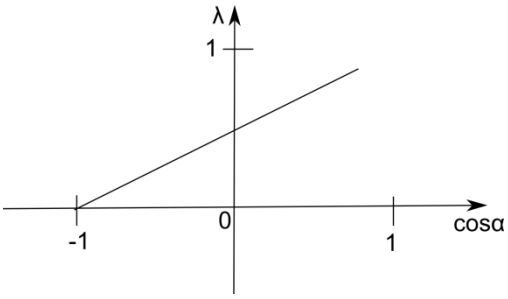
$cos\varphi = 1 \Leftrightarrow \ \lambda = 1$

$cos\varphi = -1 \Leftrightarrow \ \lambda = 0$

Um auch Punkte, die zwischen B‘ und B liegen, zu beschreiben soll außerdem folgendes gelten:

$-1 \leq cos\varphi \leq 1 \Leftrightarrow \ 0 \leq \lambda \leq 1$

Diese Annahme lässt sich am Einfachsten durch eine lineare Funktion beschreiben



$$\lambda = 0{,}5 \cdot cos\varphi + 0{,}5$$

Aus den drei Annahmen

- $C = \lambda \cdot B + (1 - \lambda) \cdot B'$
- $\vec{c} = C - A$
- $\lambda = 0{,}5 \cdot cos\varphi + 0{,}5$

lässt sich die Formel $\vec{v}_{avoid} = \frac{\vec{c}}{|\vec{c}|} \cdot \vec{v}_R(|\vec{d}|)$ für die Beschreibung des Folgens aufstellen.

5. Simulationen des Ausweichens mit *C++*

Die aufgestellten Formeln und Überlegungen wurden auch in mit Hilfe eines Programms simuliert. Zuerst sollte diese Simulation mit Hilfe des Programms NetLogo realisiert werden, jedoch stellte sich die Abstandsmessung zu den anderen *turtles* als schwierig heraus. Auch laufen Programme in NetLogo relativ langsam, sobald viele Individuen simuliert werden. Da das entwickelte Modell aber auch an simulierten Menschenmassen mit mehreren hundert Individuen getestet werden sollte, wird entschieden, ein Programm mit C++ zu schreiben. Dieses greift auf die Programmbibliothek OpenCV zurück, um die Individuen etc. zu zeichnen.

Die Individuen werden durch Instanzen der Klassen „human“ beschrieben. Jedes Objekt dieser Klasse besitzt Variablen, die unter anderem seine momentane Geschwindigkeit, seine Position und sein Ziel beschreiben. Alle Individuen werden in einem Array verwaltet, dessen Größe über die Variable POPULATION bestimmt wird.

Die Funktion human::updateVelocityVect ist für das Ausweichen zuständig, sie verändert den Geschwindigkeitsvektor (im mathematischen Modell: $\vec{a}$) so, dass es zu keiner Kollision zwischen Individuen kommt, unter Berücksichtigung der Richtung zum Ziel und der Bewegungsrichtung der anderen Individuen. Sie berechnet den Abstand vom aufrufenden Objekt zu jeder anderen Instanz der human-Klasse und falls der Abstand ungleich 0 ist (was dem Abstand zum aufrufenden Objekt selbst beschreibt) und kleiner als die Konstante k_{egal} wird ein Ausweichvektor berechnet und zu $\vec{a}$ addiert. Nach dem Durchlauf der Schleife stellt $\vec{a}$ die Summe aller Korrekturvektoren dar, wird normiert und mit der Geschwindigkeit des Objekts multipliziert.

Die Berechnung des Korrekturvektors basiert auf dem entwickelten Modell, jedoch waren ein paar Anpassungen erforderlich, um zufriedenstellende Simulationsergebnisse zu erhalten. Wird nur das Modell verwendet, bewegen sich die Objekte oft übereinander. Um das zu verhindern, wird im Programm zuerst geprüft, ob es zu einer Kollision zwischen den Objekten kommt, sprich ob die Distanz zwischen beiden Mittelpunkten kleiner als der Radius mal 2 plus einem Sicherheitsabstand von einem Pixel ist. Ist das der Fall, bewegt sich das aufrufende Objekt vom anderen Individuum weg.

```
Dvect.x = humanArray[i].position.x - position.x;
Dvect.y = humanArray[i].position.y - position.y;
if(dist <= size*2+1) {
        length = sqrt(pow(Dvect.x, 2) + pow(Dvect.y, 2));
        Dvect.x /= length;
        Dvect.y /= length;
        Dvect *= (-1)*velocity;
        position += Dvect;
}
```

Um Problemen bei direktem Aufeinandertreffen vorzubeugen, wird die Länge von $\vec{p}$ geprüft und falls sie die Länge von 10 Pixeln unterschreitet wird $\vec{p}$ als Normalvektor von a neu aufgestellt, normiert und auf eine zufällige Länge gebracht.

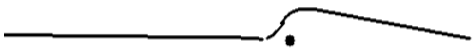
```
float length = sqrt(pow(p.x, 2) + pow(p.y, 2));
if(length < 10) {
        p.x = velocityVect.y;
        p.y = (-1.0f)*velocityVect.x;

        float length = sqrt(pow(p.x, 2) + pow(p.y, 2));
        p.x /= length;
        p.y /= length;

        p *= (rand() % 5 + 6.0f);
}
```

Eine weitere größere Anpassung war das Einführen einer Bremsfunktion, falls sich viele Individuen auf engem Raum befinden. Dazu wird jede Distanz zwischen aufrufendem Objekt und anderen Individuen in einer Variablen gespeichert, sofern diese Distanz die Konstante k_{safe} unterschreitet. Danach wird der Mittelwert gebildet und der Geschwindigkeitsvektor mit einem Faktor zwischen 0 und 1 multipliziert, welcher durch die Formel $\frac{x}{(x+c)}$ berechnet wird. Die Konstante c bestimmt die Steilheit der Funktion und ist im Moment auf 15 gesetzt.

Ergebnisse der Simulationen

1. **Ein dynamischer und ein statischer Punkt**

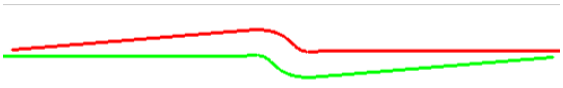
Die erste Simulation zeigt ein Individuum, welches sich leicht versetzt auf ein zweites, statisches zubewegt.



Die schwarze Linie zeigt die Bewegung des Individuums von links nach rechts. Sobald der festgelegte Abstand k_{egal} unterschritten wird, beginnt das Ausweichmanöver, wobei die Ausweichrichtung immer einen Kompromiss zwischen dem Wegbewegen vom potenziellen Kollisionsobjekt und der Richtung zum Ziel darstellt.

2. **Zwei dynamische Punkte**

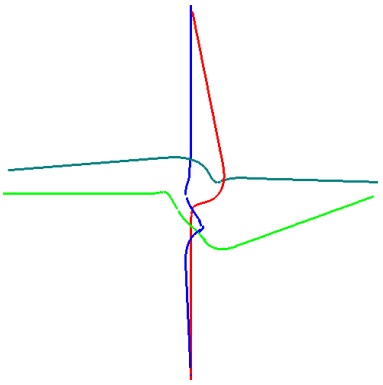
In der zweiten Simulation bewegen sich zwei Individuen, wieder leicht versetzt, aufeinander zu. Die Bewegungslinien sind sehr ähnlich zu der in der ersten Simulation.



3. Zwei frontale dynamische Punkte

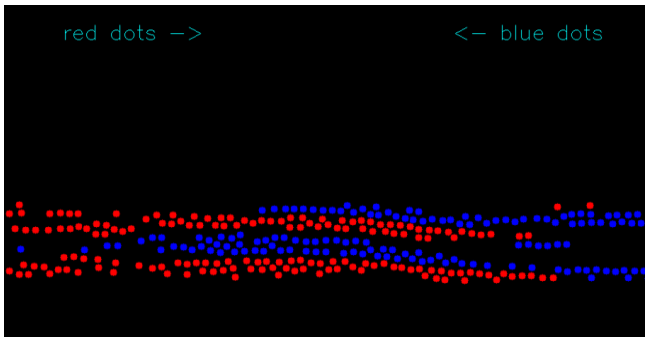
In der nächsten Simulation bewegen sich beide direkt aufeinander zu. In diesem Fall ist der p-Vektor gleich 0, was dazu führt, dass das Programm einen neuen mit zufälliger Länge aufstellt. Das Ergebnis ist nahezu identisch zu Simulation 2.

4. Vier dynamische Punkte



Als nächstes wurde die Anzahl der Individuen auf 4 verdoppelt und sie bewegen sich nun von allen vier Seiten aufeinander zu. Man erkennt, dass alle unterschiedlich stark ausweichen. Die Ausweichbewegung erscheint plausibel, jedoch können sich in der Realität andere Konstellationen ergeben.

5. Menschenströme durch Simulation des Folgens

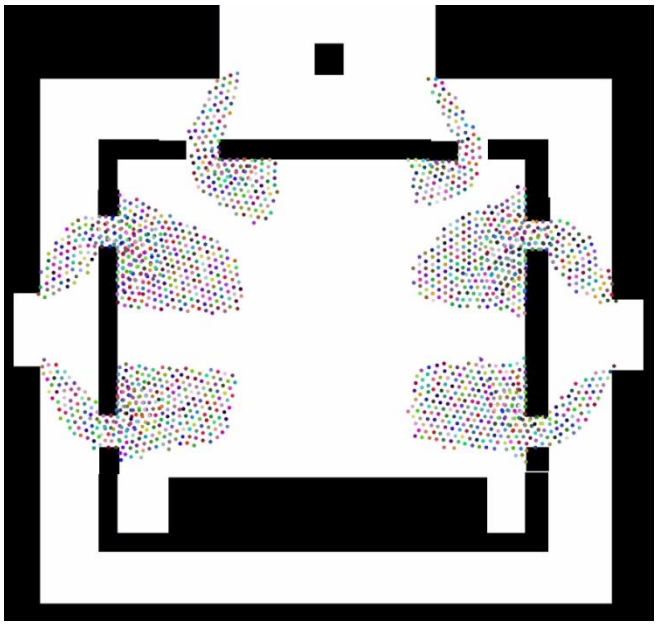


Die folgenden Simulationen zeigen das Verhalten von Menschenmassen, die sich in entgegengesetzte Richtungen bewegen. Dadurch, dass das mathematische Modell das Verfolgen von Individuen möglich macht, bilden sich wie in der Realität kleine Ströme, also Linien von Individuen , die sich in die gleiche Richtung bewegen. Vergleiche mit Videos von Experimenten zeigen, dass die Simulation in diesem Aspekt beinahe exakt dem Verhalten von realen Menschenmassen entspricht.

6. Erweiterung durch Hindernisumgehung

Die zuvor geschriebenen Simulationen wurden so erweitert, dass die Individuen auf ihrem Weg zum Ziel Hindernisse umgehen können. Das funktioniert folgendermaßen: Ein spezielles Programm berechnet ein Graustufenbild, welches für jeden Punkt die Distanz zum Ziel

angibt. Ein Verlauf von Weiß nach Schwarz zeigt den Weg. Die Individuen müssen sich nun nur noch unter Berücksichtigung der zuvor erstellten Regeln in die Richtung bewegen, in der die Pixelwerte am stärksten abnehmen. Dadurch kann das Verhalten von Menschen zum



Beispiel bei der Evakuierung eines Gebäudes sehr realitätsnahe simuliert werden.

6. Simulationen der Bewegung von Menschenmassen mit Netlogo 4.1.3

6.1. Menschenmassen an einer T-Kreuzung

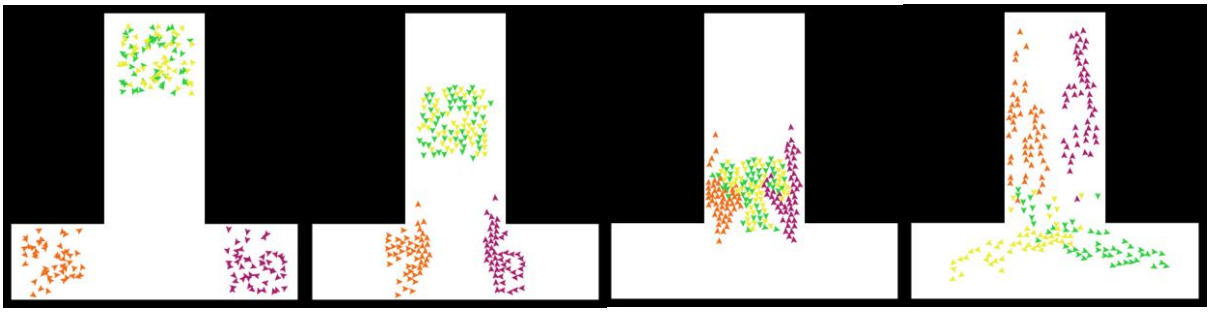
Mit der Software Netlogo wurde eine Simulation über das Verhalten von kleineren Menschenmassen an einer Kreuzung simuliert. Mit einem simplen Modell, das drei Gruppen mit jeweils verschiedenen Ausgangspositionen zu verschieden definierten Zielen starten lässt, wurde begonnen.

Die gesteuerten *turtles* (Objekte, die Menschen simulieren sollen) bewegten sich zunächst noch übereinander. Durch Abstandsberechnungen konnte dieses Problem vermieden werden um die Simulation realistisch zu gestalten. Für diese Berechnungen wurde der Vektor zum Ziel durch die Differenz der Koordinaten des Ziels und denen des Ausgangspunktes angegeben. Die Länge des Richtungsvektors wird mithilfe der Quadratwurzel der addierten Quadrate von X und Y berechnet und die Geschwindigkeit jedes einzelnen *turtles* in Richtung X und Y wurde dann durch den Einheitsvektor definiert. Das Programm berechnet fortlaufend den Abstand zwischen jeden *turtle* und multipliziert die Geschwindigkeit mit vel_fac > 0 um die Geschwindigkeit bei einem zu geringen Abstand

zu verringern und um das Betreten „verbotener“ Zonen zu vermeiden. Wenn die kritische Distanz zwischen einem orangen oder violetten mit einem gelben oder grünen *turtle* und gleichzeitig der „verbotenen“ Zone erreicht wird, wird automatisch die Geschwindigkeit der betroffenen orangen und violetten *turtles* auf 0 gesetzt. Ist das Ziel von den *turtles* erreicht verringert sich die Geschwindigkeit auf 0.

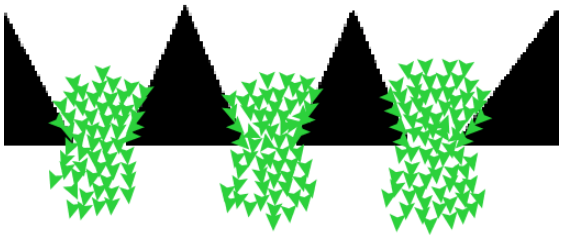
```
ask turtles
   set dirX xTarget - xcor                    Berechnung des Richtungsvektors
   set dirY yTarget – ycor
   set len sqrt(dirX * dirX + dirY * dirY)    Berechnung der Länge des Vektors
   set vx dirX / len                          Definieren der Geschwindigkeit durch den
                                              Einheitsvektors
   set vy dirY / len
  if (vx != 0 or vy != 0)                     Ausrichtung der turtles in Richtung der
                                                Zielkoordinaten bei
                                                Geschwindigkeit ≠ 0
     set heading (atan vx vy)
```

Diese Simulation soll zeigen, wie 4 Menschengruppen reagieren, wenn sie an einer Kreuzung interagieren und verschiedene Ausweichbewegungen erfolgen müssen. Es entstehen 3 Ströme mit 2 verschiedenen Zielrichtungen. 2 mit denselben Ziel an den Seiten und in der Mitte ein Gegenstrom. In dieser Simulation kann man die Zahl der Versuchspersonen beliebig erhöhen um die Auswirkungen der Größenänderungen genau beobachten zu können.



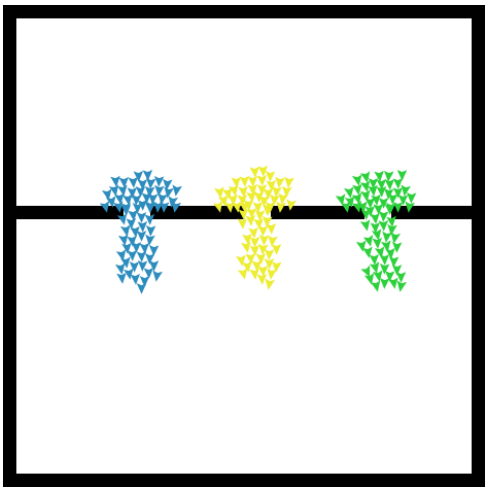
6.2. Evakuierung durch 3 Türen

Danach wurde eine Simulation mit NetLogo gestartet, die die Idee hatte eine große Menge von Menschen durch 3 gleich große Fluchtwege zu schicken um so eine Evakuierung zu simulieren. Es wurden zwei Varianten für diese Evakuierung angefertigt, um beobachten zu können, welche Auswirkungen die jeweiligen Bedingungen auf die Dauer der Evakuierung haben. Die Basis bildet die vorherige Simulation (Abschnitt 6.1.) mit der Abstandsberechnung. Es wurden lediglich die Ziele mit Checkpoints anders angeordnet. Die Checkpoints wurden mit Bedingungen verbunden. Sollten *turtles* in einem bestimmten Bereich (x- und y-Koordinate) *spawnen* dann wird das Ziel auf den nächstgelegenen Notausgang gesetzt. Wird dieses Ziel erreicht, wird es automatisch auf *xNewTarget* gesetzt, das in einem gestreuten Bereich außerhalb des zu evakuierenden Gebäudes liegt. Somit konnte ein grobes Modell für eine Evakuierung erstellt werden.



6.3. Stau vor Türen

Mit der interaktiven Programmieroberfläche *NetLogo 4.1.3* wurde die Bewegung von drei Gruppen simuliert. Die Gruppen haben je einen verschiedenen Ausgangspunkt und bewegen sich auf verschiedene Ziele durch enge Öffnungen, Türen eines Hauses. Die Simulation zeigt die Stauung vor den Türen. In solch einer Notsituation fällt es schwerer, von der Seite durch die Tür zu kommen, als direkt durchzugehen. Um Panik zu vermeiden, sollten sich die Individuen in Reihen anstellen.



Zuerst platziert man drei verschiedene Gruppen von *turtles*, in drei Farben definiert, an verschiedenen Punkten. Jede Gruppe soll eine bestimmte Öffnung durchqueren. Wenn sie

an die schwarzen *patches* an gewissen x-Koordinaten stoßen, müssen die *turtles* entweder nach links oder rechts gehen, um durch die Türen zu kommen.

```
if (pcolor = black)                ;; wenn die Farbe des Patches schwarz ist
[
if (xcor > -1 and xcor < 9)         ;; und wenn die x-Koordinaten größer als
                                   -1 und  kleiner als 9 sind
[set vx 1                          ;; dann gehen sie eines nach rechts
set vy 0]                          ;; bleiben aber auf der gleichen Höhe
```

Mit Hilfe der Abstandsberechnung wurde erreicht, dass die *turtles* sich gegenseitig ausweichen und nicht übereinander laufen.

```
ask turtles with [who != [ who ] of myself]
[
if (distance myself < 0.6)
[set vx_temp vx_temp + ([ xcor ] of myself - xcor) * 8
set vy_temp vy_temp + ([ ycor ] of myself - ycor) * 8
set critical_dist 1]

if (distance myself < 0.5)          ;; wenn die Distanz zwischen dem
                                   Turtle und einem anderen Turtle kleiner als 0.5 ist
[set vel_fac vel_fac * 0.25]    ;;verringert sich die Geschwindigkeit um den Faktor 0.25
```

7. Simulationen der Bewegung von Menschenmassen mit *Matlab*

6.1. Konfrontation ohne Hindernisse

Parallel zu den mit NetLogo und C++ angefertigten Simulationen wurde auch eine Simulation mit Matlab erstellt. Nach einer kurzen Einführung in das Programm durch Dr. Kristian Bredies wurde die Idee gefasst die Reaktionen von zwei Gruppen, deren Laufbahnen sich bei der Hälfte ihrer Wege normal zueinander schneiden, zu beobachten, doch zuvor musste man das Problem der Programmierung einer zum Ziel gerichteten Bewegung lösen. Dies gelang mit folgendem Code:

```
d = [wz pz] - [w(i) p(i)];
dbetrag = sqrt(d(1)^2 + d(2)^2);
      if (dbetrag > 0)
```

```
    v = d/dbetrag;
  else
    v = [0 0];
  end
```

Dieser Code gibt an, dass der Vektor zwischen den Zielkoordinaten [wz pz] und den Koordinaten des Punktes/der Person [w(i) p(i)] berechnet wird. Dieser wird hier als d (=distance) bezeichnet. Danach wird der Betrag dieses Vektors (dbetrag) berechnet. Wenn die Bedingung , dass der Betrag dieses Vektors größer als null ist, was gleichbedeutend mit der Tatsache , dass noch ein Abstand zwischen Person und Ziel vorherrscht , ist, erfüllt ist wird der Geschwindigkeitsvektor v (=velocity) der Person gleich d durch dbetrag gesetzt um seinen Betrag auf die Größe eins zu bringen. Ist diese Bedingung jedoch nicht erfüllt (= die Person hat ihr Ziel erreicht) wird v zum Vektor [0 0].

Das nächste zu lösende Problem war, dass der fehlenden Kollisionen und Ausweichbewegungen der Personen untereinander. Dazu wurde bei den Berechnungen zwischen Ausweichbewegungen zwischen derselben Gruppe angehörenden und verschiedenen Gruppen angehörenden Personen unterschieden.

```
for k = eigenegruppe
        b = [w(k) p(k)] - [w(i) p(i)];
      bbetrag = sqrt(b(1)^2 + b(2)^2);
      if (bbetrag < 1) && (bbetrag > 0)
        ce = b/bbetrag *(-1) ;
      else
      ce = [0 0];
      end
end
```

Dazu wurde wieder ein Abstand (b) berechnet. Auch dieser Abstand zwischen den Personen einer Gruppe untereinander wurde wieder auf die Einheitsgröße gebracht. Wenn der Betrag von b zwischen eins und null liegt gilt ce (=*correction* eigene Gruppe) = b/bbetrag *(-1) um eine Korrekturbewegung in die entgegengesetzte Richtung zu bewirken, ansonsten ist ce [0 0]. Auf ähnliche Art und Weise wurden ca (=*correction* andere Gruppe) und später auch co (=*correction obstacle*) berechnet und in die Berechnung der Bewegung eingefügt:

```
w(i) = w(i) + s*(v(1)+ce(1)+ca(1)+2*co(1));
p(i) = p(i) + s*(v(2)+ce(2)+ca(1)+2*co(2));
```

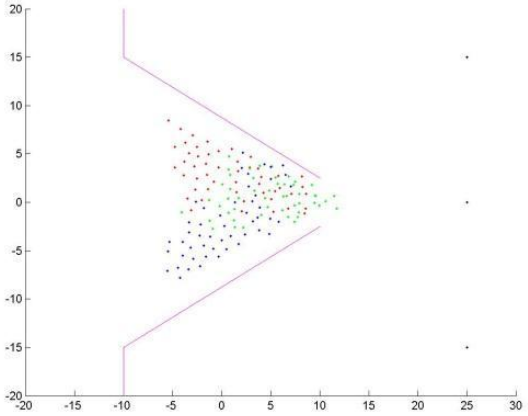
Die neuen Koordinaten der Personen [w(i) p(i)] setzen sich aus den alten Koordinaten, dem Geschwindigkeitsvektor v und den Korrekturvektoren ce, ca und co mit dem Zeitschritt s multipliziert zusammen.

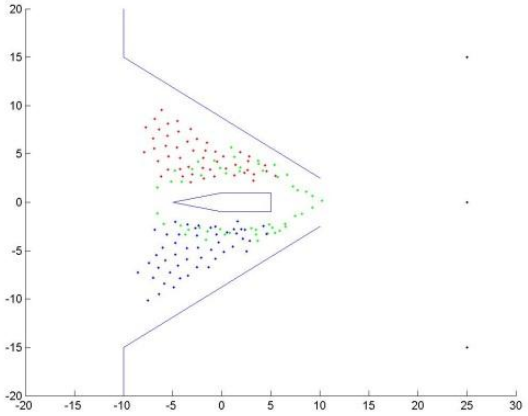
6.2. Evakuierung durch trichterförmigen Ausgang

Nachdem die Programmierarbeit getan war, war das oben beschriebene Szenario (6.1.) mehr oder weniger realistisch simulierbar. Dabei stellte sich heraus, dass es keinerlei Probleme bei der Konfrontation von Menschenmassen gibt, wenn genug Platz zum Ausweichen vorhanden ist, also wurde ein 2. Szenario programmiert, bei dem drei Menschenmassen nahezu zeitgleich ein trichterartiges Hindernis passieren müssen um ihre Ziele zu erreichen. Dies verursachte schon wesentlich mehr Probleme und somit dauerte es auch wesentlich länger, bis sich die Situation auflöste und die Personen ihr Ziel erreichten. Außerdem zeigten sich auch Differenzen zwischen der Realität und der Simulation. So wichen die Personen in der Simulation den Hindernissen nicht nur aus, sondern hielten auch Abstand von ihnen auch wenn dadurch eine Abweichung vom kürzesten Weg erfolgte. Dies liegt daran, dass der Grenzwert, ab welchem die Bewegungskorrektur für Hindernisse in Kraft tritt, über den Betrag des Abstandsvektors definiert wurde. Ist dieser Wert zu groß bildet sich um die Hindernisse ein Bereich der nicht betreten wird, ist er jedoch zu klein findet die Ausweichbewegung zu spät statt oder reicht nicht mehr aus um ein Betreten des Hindernisses zu verhindern.

Beim Beobachten der Simulation stellte sich folgende Frage: Ist es möglich die Zeit, die vergangen ist bis auch die letzte Person den Trichter passiert hat, zu verringern? Es kam die Idee auf ein keilartiges Hindernis in der Mitte des Trichters zu platzieren um die Strombildung der Personen zu begünstigen. Die benötigte Zeit wurde pro Variante Zehn mal gemessen. Jeweils der höchste und der geringste Wert wurden aus der Wertung genommen um mögliche Ausreißer zu verhindern. Schließlich wurden die Durchschnittswerte berechnet und verglichen. Dabei kam man zu folgendem Ergebnis: Die durchschnittlich benötigte Zeit ohne Keil beträgt ca. 56 Sekunden, mit jedoch nur ca. 51 Sekunden. Dies entspricht einer Reduktion um fast 9%. Es wäre sehr interessant diesen Versuch auch in der Realität durchzuführen. Leider stehen uns die Mittel dazu nicht zur Verfügung. Auf der folgenden

Seite sieht man zwei Bilder auf denen man das Verhalten der Personen, einmal mit und einmal ohne Keil, erkennen kann.





Resumé

Nach einem raschen Kennenlernen hatten wir die Möglichkeit entsprechend unseren Interessen für Mathematik, Physik und Programmieren eine Woche lang uns mit dem Thema der Modellierung der Dynamik von Fußgängern zu beschäftigen. Innerhalb der Gruppe fanden wir uns automatisch in kleinen Teams von zwei oder drei Personen zusammen. In den meisten Fällen in Partnerarbeit, teils mit Hilfe von Kristian, teils in Gemeinschaftsarbeit und teils in Einzelarbeit organisierten wir uns in Prozessen des Ideensammelns, des Aufstellens von mathematischen Aussagen sowie beim Experimentieren mit Computersimulationen und schließlich beim Verfassen dieser Dokumentation. Wir hatten viel zu lachen, viel zu durchdenken und haben trotz der langen Arbeit viel Spaß gehabt.
Wir haben das Gefühl etwas Gemeinsames geschaffen zu haben, denn die Ergebnisse wirken interessant, da sie mit der Realität der Menschenbewegungen gut zu vergleichen sind.

Spielstärkeeinschätzung und Ergebnisvorhersage bei Paarvergleichen

Dr. Frank Heyde

Modellierer/innen:

Felix Bloder

Maria Buchsteiner

Randos Megjidi

Michael Missethan

Lukas Posch

Mathias Pützl

Inhalt

1 Einleitung

Wir alle kennen verschiedene Ranglisten. Sie zeigen uns durch eine Ratingzahl eine Bewertung von Spielern oder Mannschaften in Punkten und jene Mannschaften mit einer höheren Punktezahl belegen höhere Positionen. Doch wie werden beispielsweise Rankings in Fußball, Schach oder Tennis aufgestellt? Und wie bekommt man solche Punkteratingsysteme? Kann man nun auch ein Ergebnis mit Hilfe eines solchen Ratings vorhersagen? Wie kann man vorige Spiele in ein Ratingsystem einfließen lassen? Diese und noch viel mehr Fragen kann man sich zu diesem Thema stellen und wir haben in unserem Projekt versucht möglichst viele Fragen zu beantworten und im Speziellen Arten des ELO-Ratings zu erforschen, benutzen und zu beobachten.

2 Verschiedene Ranglisten

2.1 Tennis

Im Tennis werden Ratings mittels des ATP-Rankings, dem so genannten Entry System vergeben und daraus entsteht eine internationale Rangliste.

Das System des Entry Systems berücksichtigt nur die Punkte eines Spielers innerhalb der letzten 52 Wochen. Solche Punkte kann man durch das Erreichen einer bestimmten Runde in einem Turnier bekommen. Neben den Punkten, die man durch die Siege in einem Turnier bekommen kann, werden zusätzlich noch Zusatzpunkte, wie Qualifikationspunkte addiert. Aber nicht in jedem Turnier sind die Punkteverteilungen gleich, es gibt 12 Pflichtturniere: 4 Grand Slam Turniere und 8 Masters-1000 Turniere. Ein Sieg eines Grand Slam Turniers bringt mehr Punkte ein, als ein Sieg eines kleinen Turniers. Außerdem fließen nicht alle Punkte, die man in verschiedenen Turnieren gewinnt bei dem Rating ein, sondern nur die 18 besten Turnierergebnisse. Bei dem Frauen-Ranking, dem WTA Tour Ranking, ist das System ziemlich gleich, die Unterschiede liegen nur darin, dass es im ATP-Ranking mehrere kleine Regeln gibt, die man zusätzlich beachten muss, wie beispielsweise die Regel, dass bei den TOP 30 der Herren mindestens 4 ATP Turniere in die Wertung einfließen müssen, und dass bei dem Frauen-Rating nur die besten 16 Turnierpunkte innerhalb der letzten 52 Wochen gezählt werden.

Außerdem gibt es noch das ATP-Race. Dies ist eine Form des Rankings, bei der alle Spieler zu Beginn der Saison 0 Punkte bekommen. Man kennt solche Ranglisten, die jede Saison bei 0 starten auch beispielsweise von Schiläufern oder in der Formel 1. Der Gewinn von Punkten erfolgt nun ziemlich gleich wie bei ATP-Rankings. Es gibt aber auch Rankings für Frauentennis, die mit Beginn einer neuen Saison wieder von 0 beginnen, sowie ein Ranking für Teams, jedoch funktionieren alle Systeme sehr ähnlich. Man kann anhand einer Aufteilungstabelle vergleichen, wie viele Punkte welcher Sieg in welcher Runde bei welchem Turnier einbringt.

Nach der Analyse dieses Systems haben wir uns Gedanken gemacht, was bei dieser Bewertung gut ist und was eventuell noch berücksichtigt oder verbessert werden kann. Zunächst haben wir erkannt, dass man die Punkte nur für Siege bekommt. Diese Punktezahl variiert zwar bei den verschiedenen Turnieren und Runden, jedoch nicht bei den Gegnern. Das heißt wenn man einen sehr starken Gegner besiegt, bekommt man gleich viele Punkte dafür, als ob man gegen einen der schwächsten Konkurrenten gewinnt.

Andererseits kann ein Tennisspieler verletzungsbedingt oder wegen gesundheitlicher Probleme ausfallen und beispielsweise nicht an einem Turnier teilnehmen, das sehr groß und wichtig ist und wo man viele Punkte erzielen kann, so schlägt sich das Fehlen dieser

wichtigen Punkte für 52 Wochen nieder, denn erst nach diesem Zeitraum werden die alten Ergebnisse nicht mehr im Rating berücksichtigt.

2.2 FIFA-Rating System

Das meistbenutzte, und wohl auch bekannteste Rating System für Fußballnationalmannschaften ist jenes der ,,Fédération Internationale de Football Association´´, zu Deutsch die ,,Internationale Föderation des Verbandsfußballs´´, oder kurz FIFA. Jenes Rating System errechnet die Punkte der Nationalmannschaften anhand des Ergebnisses eines Spiels, der Priorität eines Spiels, der Stärke des Gegners und auch anhand der Stärke der Konföderation (z.B. UEFA, CONMEBOL, CONCACAF) welcher die Mannschaften angehören. Die Punkte verlieren aber auch ihren Wert direkt proportional zu ihrem Alter.

All jene Daten werden nach Spielende in folgende Formeln eingesetzt:

$$P = P_1 + P_2 + P_3 + P_4$$

$$P_2 = P_{x_2} * 0{,}5$$

$$P_3 = P_{x_3} * 0{,}3$$

$$P_4 = P_{x_4} * 0{,}2$$

$$P_x = P_A + (M * I * T * C)$$

P: Punktestand der Mannschaft nach Spielende

P_1: sind alle Punkte die in einem Zeitraum, der jünger als 12 Monate ist, erlangt worden sind

P_2: sind alle Punkte die in einem Zeitraum, der jünger als 24 Monate, aber älter als 12 Monate ist, gesammelt worden sind (P_{x_2}) und dann mit dem Entwertungsfaktor von 0,5 multipliziert worden sind

P_3: sind alle Punkte die in einem Zeitraum, der jünger als 36 Monate, aber älter als 24 Monate ist, gesammelt worden sind (P_{x_3})und dann mit dem Entwertungsfaktor von 0,3 multipliziert worden sind

P_4: sind alle Punkte die in einem Zeitraum, der jünger als 48 Monate, aber älter als 36 Monate ist, gesammelt worden sind (P_{x_4}) und dann mit dem Entwertungsfaktor von 0,2 multipliziert worden sind

P_x: ist eine stellvertretende Zahl, statt P_x kann man auch P_{x_2} bis P_{x_4} einsetzen

P_A: der Punktestand der Mannschaft vor Spielbeginn

M: Spielergebnis

M wird zu

3: wenn die Mannschaft das Spiel nach regulärer Spielzeit oder nach Verlängerung gewonnen hat

2: wenn die Mannschaft das Spiel durch Elfmeter schießen gewonnen hat

1: wenn die Mannschaft unentschieden gespielt hat, oder das Spiel durch Elfmeterschießen verloren hat

0: wenn die Mannschaft verloren hat

I: Priorität des Spiels

I wird zu

$1{,}0$: bei Freundschaftsspielen

$2{,}5$: bei Qualifikationsspielen für WM bzw. Kontinentalmeisterschaftsendrunden (z.B. EM, Afrika-Cup)

$3{,}0$: bei Kontinentalmeisterschaftsendrunden

$4{,}0$: bei Weltmeisterschaftsendrunden

T: Stärke des Gegners

$T = 200 - P_G$

P_G: Position des Gegners im FIFA-Ranking

T nimmt automatisch den Wert 200 an wenn der Gegner auf Weltranglistenposition eins steht.

Wenn der Gegner eine schlechtere Weltranglistenposition als 150 innehat, nimmt T trotzdem den Wert 50 an.

C: Mittelwert der Stärken beider Kontinentalverbände denen die Mannschaften angehören.

$$C = \frac{K_A + K_B}{2}$$

K_A:Stärke des Kontinentalverbandes von Team A

K_B:Stärke des Kontinentalverbandes von Team B

Die Stärken der Kontinentalverbände werden folgendermaßen berechnet:

Zuallererst analysiert man die Spiele der vergangenen drei Weltmeisterschaftsendrunden und sortiert jene Spiele aus in denen Mannschaften mit dem gleichen Kontinentalverband gegeneinander gespielt haben. Nun werden jedem Kontinentalverband Punkte vergeben. Ein Punkt für einen Sieg und $0{,}5$ Punkte für ein Unentschieden. Der nächste Schritt ist es die so erlangten Punkte durch die Anzahl der Spiele zu dividieren in denen Mannschaften jenes Kontinentalverbandes beteiligt gewesen sind. Des Weiteren werden nun die erhaltenen Zahlen verglichen, die höchste dieser Zahlen wird als P_{MAX} bezeichnet. Der letzte Schritt, um die Stärke des Kontinentalverbandes (S_{con}) zu ermitteln, ist es die aus der Division erhaltenen Zahl P_N in folgende Formel einzusetzen:

$$S_{con} = \sqrt[4]{\frac{P_n}{P_{max}}}$$

Aus jener Reihe von Rechnungen und Analysen haben sich für die Kontinentalverbände folgende Zahlen ergeben:

UEFA (Europa):	$1{,}00$
CONMEBOL (Südamerika):	$1{,}00$
CONCACAF (Nord- und Mittelamerika, sowie die Karibik):	$0{,}88$
CAF (Afrika):	$0{,}86$
AFC (Asien und Australien):	$0{,}86$
OFC (Ozeanien):	$0{,}85$

Vor- und Nachteile

Die Vorteile des FIFA-Rating Systems sind, dass man die aktuellen Stärken der Mannschaften misst, d.h. dass man nicht jene Punkte miteinberechnet die erreicht worden sind, als die Mannschaft noch viel stärker war. Würde man diese Punkte miteinberechnen, dann würde man nur das Ergebnis verzerren und es wäre nicht mehr so aussagekräftig. Ein weiterer Vorteil ist, dass man keine Punkte horten kann, um eine bessere Bewertung zu bekommen, da alle Punkte sich bereits nach einem Jahr nach und nach entwerten.

Die Nachteile jenes Systems sind, dass weder der Heimvorteil, noch die Dominanz des Siegers mit einberechnet werden. Es lassen sich anhand der Rangliste mathematisch auch keine Ergebnisse berechnen. Ein weiterer Nachteil ist, dass anhand der Punkteentwertung ein stetiger Spielzwang herrscht. Dem ist aus jenem Grund nicht immer leicht entgegenzuwirken, da nicht jede Mannschaft bei Turnierendrunden

mitspielen, dadurch gehen viele Punkte verloren und man kommt nur schwer von den hinteren Ranglistenpositionen weg.

2.3 ELO-System

Besser geeignet für die Vorhersage von Ergebnissen ist das Elo-System, das beim Schach und Go zum Einsatz kommt. Das Modell hinter diesem Rating ist schon etwas komplizierter als bei den vorhin erwähnten Systemen. Die Formel für die Berechnung der Eloänderung lautet:

$$R'_A = R_A + k * (S_A - E_A)$$

R'_A; neues Rating für Spieler A

R_A: altes Rating von Spieler A

k: Faktor, der bestimmt wie sehr die neue Partie gegenüber älteren Partien gewichtet wird, also umso größer k ist, umso mehr Bedeutung hat der Ausgang der Partie für die Ratingveränderung von Spieler. ab 2400 Elo gilt $k = 10$ und darunter $k = 15$.

S_A: Ergebnis, das der Spieler A in der Partie erzielte (1 für Sieg, ½ für Remis, 0 für Niederlage)

E_A: Punkteerwartung für Spieler A

Anmerkung: Der zurzeit beste Spieler hat ein Rating von 2835 Elo, ein guter Vereinsspieler etwa 2000 Elo, ein Anfänger 1200.

Hier fällt auf, dass es im Gegensatz zu den anderen Rating-Systemen auch eine Punkteerwartung, also in anderen Worten die Wahrscheinlichkeit, dass ein Spieler gewinnt, berücksichtigt wird. Nun stellt sich aber die Frage, wie man diese Punkteerwartung berechnen kann. Klar sollte sie höher sein wenn man gegen einen schlechteren Gegner spielt und umgekehrt niedriger wenn man sich mit einem besseren Spieler misst. Doch um auf wirklich genaue Werte zu kommen, müssen wir uns genauer mit der Wahrscheinlichkeitsrechnung beschäftigen.

Da die tatsächliche Spielstärke die ein/e Spieler/Mannschaft in einer Begegnung an den Tag legt von vielen Faktoren (Tagesverfassung, Trainingsaktivität, Verletzungen, Ausfälle) abhängt, entspricht die Spielstärke nie genau dem Rating. Sei nun der Einfachheit halber mit Spielstärke die Ratingzahl gemeint, die der Leistung entspricht die der Spieler in dieser Begegnung an den Tag legt.

Nun nimmt man an, dass die Wahrscheinlichkeiten, mit der gewisse Spielstärken auftreten, nach einem gewissen Muster verteilt sind. Beim Elo-Rating nimmt man an, dass die Spielstärken normalverteilt sind, eine weitere Verteilung die bei Ratingsystemen verwendet wird, ist die Extremwertverteilung.

Bevor wir mit der Berechnung des Erwartungswertes fortfahren, ist es sinnvoll die Normalverteilung genauer zu betrachten, um die folgenden Schritte besser verstehen zu können.

Bei einer Normalverteilung sind die Werte symmetrisch um einen Mittelwert verteilt. Wobei die Werte umso seltener auftreten, umso weiter sie vom Mittelwert entfernt sind. Charakteristisch für die Normalverteilung ist folgender Satz: Zufallsvariablen sind dann normalverteilt, wenn sie durch Überlagerung einer großen Zahl von unabhängigen Einflüssen entstehen, wobei jede einzelne Einflussgröße einen im Verhältnis zur Gesamtsumme unbedeutenden Beitrag liefert. Die Dichtefunktion der Normalverteilung ist die berühmte Gauß'sche Glockenkurve. Die Formel für die Dichtefunktion der Normalverteilung lautet:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

μ: Mittelwert (Werte im Bereich um den Mittelwert treten am häufigsten auf)

σ: Standardabweichung (Wie sehr die Werte vom Mittelwert abweichen, wird sie größer, ist die Kurve flacher)

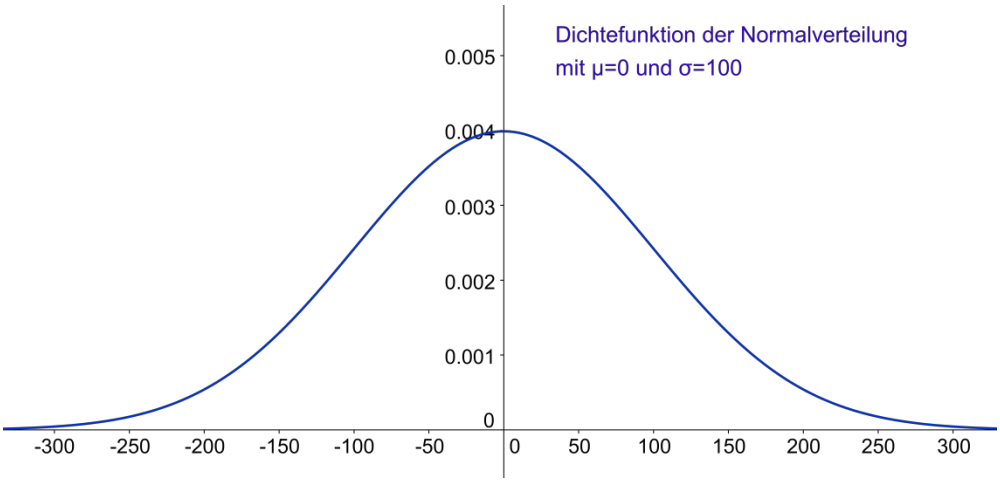
Die Formel für die dazugehörige Verteilungsfunktion lautet:

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2} dt$$

Die Verteilungsfunktion an der Stelle x gibt an, wie groß die Wahrscheinlichkeit ist, dass der Wert der Zufallsvariable kleiner gleich x ist. Logischerweise liegen alle Werte im Bereich von 0 und 1.

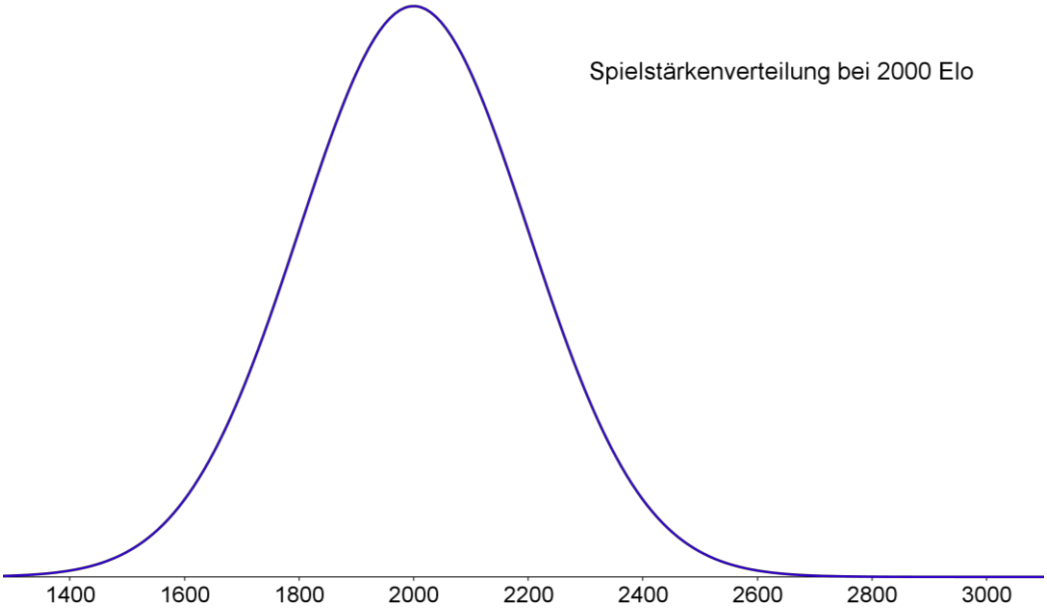
Die Normalverteilung mit $\mu = 0$ und $\sigma = 1$ wird Standardnormalverteilung genannt. Zwischen einer Verteilungsfunktion und der Standardnormalverteilung herrscht folgender Zusammenhang:

$$F(x) = \Phi\left(\frac{x-\mu}{\sigma}\right)$$



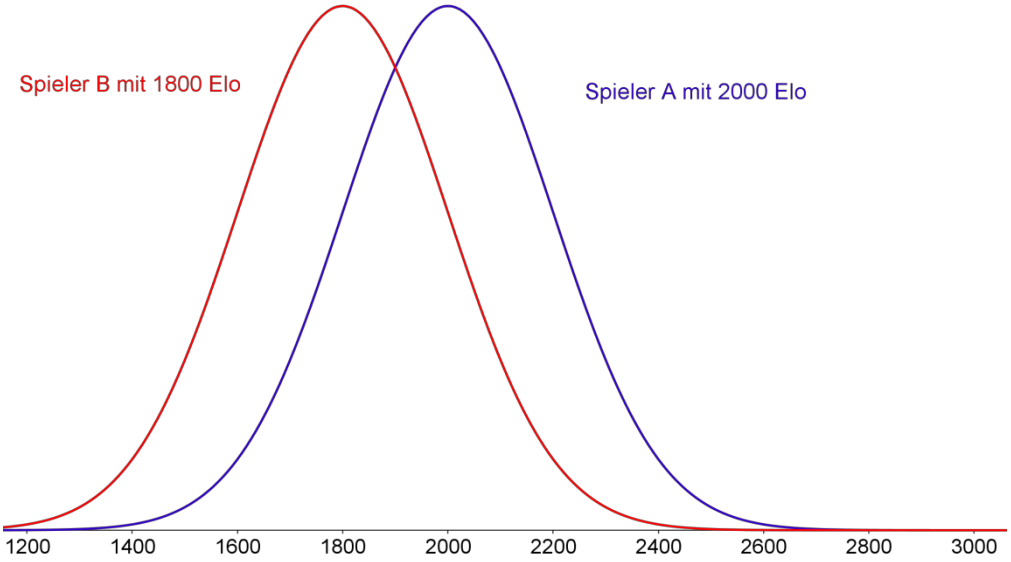
Ein Beispiel aus der Praxis sind die Fertigungsfehler von Werkstücken. Nehmen wir an jemand produziert Schrauben mit einer Solllänge von $50mm$. Da er aber sehr ungenau arbeitet, werden einige Schrauben $49mm$ lang, andere wiederrum $51mm$. Die meisten werden aber $50\ mm$ lang, ebenso ist der Durchschnitt aller Längen $50mm$ und die Länge kann von vielen Faktoren beeinflusst werden, die Werte der nun auftretenden Längen der Schrauben sind also normalverteilt. Die Standardabweichung gibt nun an, wie stark die Werte streuen. Ist sie klein liegen beinahe alle Längen im Bereich um $50mm$, ist sie groß, weichen viele Längen vom Sollwert ab.

Sehen wir uns die Berechnung des Erwartungswerts nun anhand eines Beispiels an. Nehmen wir an ein Spieler hätte 2000 Elopunkte. Nun ist klar, dass seine Spielstärke nicht immer 2000 Elo entspricht, einmal wird sie womöglich 1900 Elo entsprechen, an einem guten Tag vielleicht 2200. Trotzdem wird die Spielstärke meist in einem Bereich um 2000 Elo sein und im Mittelwert wird sie genau dem Rating des Spielers entsprechen, also in unserem Fall 2000 Elo. Stellt man diese Verteilung nun anhand eines Graphen dar, erhält man die bekannte Gauß´sche Glockenkurve:



Die Frage wie die Standardabweichung nun sinnvoll zu wählen ist, benötigt schon längere Beobachtungen von Partien. Durch Nachprüfung von einigen Eloberechnungen konnten wir leicht feststellen, dass die Standardabweichung mit 200 gewählt worden ist, ob dieser Wert allerdings sinnvoll ist, wird in einem späteren Kapitel behandelt.

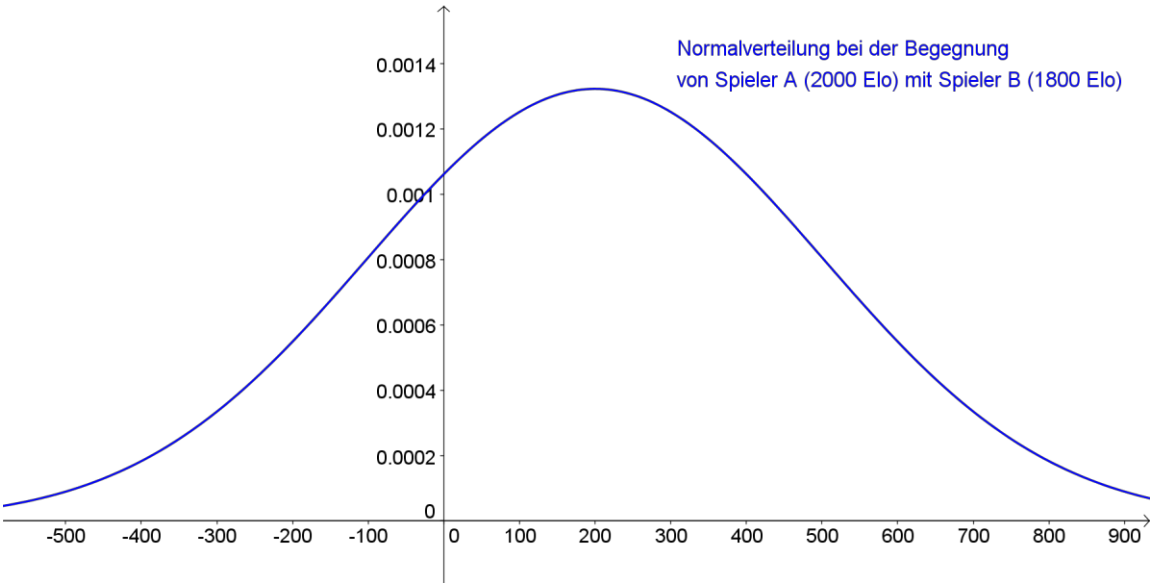
Nun stellen wir uns vor unser Spieler tritt gegen einen Kontrahenten mit 1800 Elo an, dabei sind die beiden erwarteten Spielstärken der beiden normalverteilt:



Wenn jetzt die Spielstärke des Spielers A höher als die des Spielers B ist, gewinnt er, wenn sie niedriger ist, verliert er. In der Praxis kann natürlich eine Partie auch Remis enden, dies können wir vorerst aber einmal vernachlässigen, da es kaum Änderungen der Punkteerwartung bewirkt.

Nun ist für den Ausgang der Partie also der Unterschied der beiden aufgetretenen Spielstärken (Spielstärke A − Spielstärke B) entscheidend. Ist sie größer als 0, gewinnt Spieler A, ist sie kleiner, Spieler B. Um nun die Wahrscheinlichkeit für einen Sieg von Spieler A zu berechnen, muss man die Wahrscheinlichkeit berechnen, dass der Unterschied größer 0 ist. Mit Hilfe der Integralrechnung können wir zeigen, dass auch die Differenzen der beiden Spielstärken ebenso normal verteilt sind, der Beweis dafür erfordert jedoch höhere Mathematik und ist daher hier nicht angeführt.

Die Differenz der beiden Spielstärken sind mit dem Mittelwert $\mu_1 - \mu_2$ und einer Standardabweichung von $\sqrt{2} * \sigma$ normalverteilt, wobei μ_1 …Elozahl Spieler A, μ_2 …Elozahl Spieler B und σ…Standardabweichung der beiden Spielstärke $(= 200)$. In unserem Beispiel wäre nun der Mittelwert $2000 - 1800 = 200$ und die Standardabweichung $\sqrt{2} * 200$. Graphisch sieht dies wie folgt aus:



Nun ist die Wahrscheinlichkeit, dass Spieler B gewinnt, also ident mit der Wahrscheinlichkeit, dass die Differenz kleiner 0 ist. Dies ist bekanntlich gleich der Fläche unter der Kurve von $-\infty$ bis 0, also das Integral im Intervall von $-\infty$ bis 0:

$$P(B) = \int_{-\infty}^{0} f(x) = \int_{-\infty}^{0} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = \int_{-\infty}^{0} \frac{1}{200\sqrt{2}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-200}{200\sqrt{2}}\right)^2} dx$$

Der Wert ist nun derselbe wie der Funktionswert an der Stelle 0 von der zugehörigen Verteilungsfunktion:

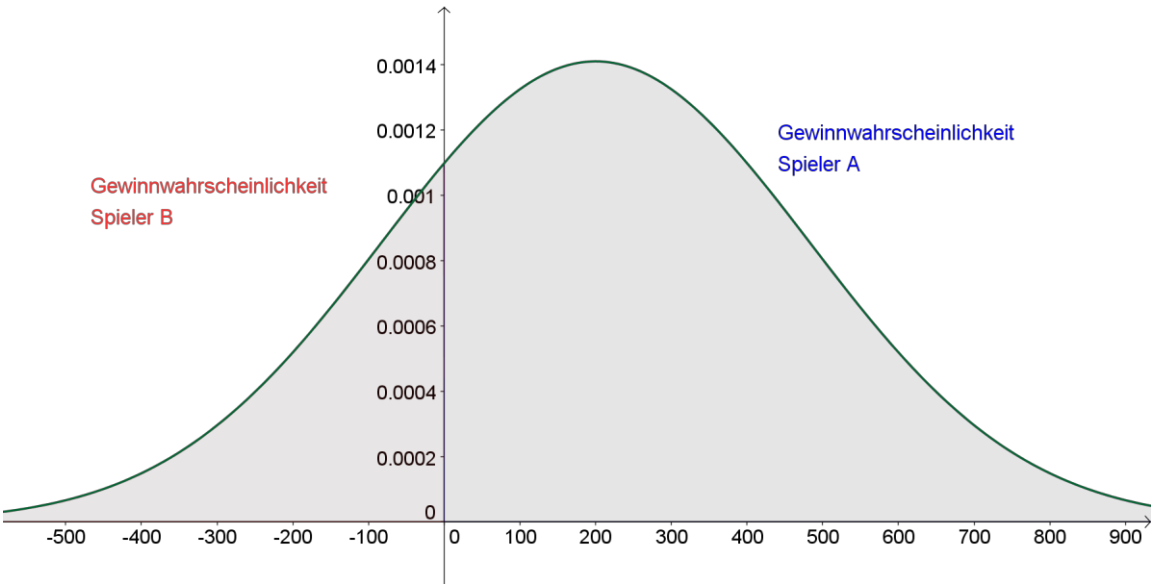
$$F(0) = \frac{1}{200\sqrt{2}\sqrt{2\pi}} \int_{-\infty}^{0} e^{-\frac{1}{2}\left(\frac{t-200}{\sigma}\right)} dt$$

Da man diesen Wert nicht explizit ausrechnen kann, müssen wir den oben erwähnten Zusammenhang verwenden. Die Werte der Standardnormalverteilung kann man dann in einer Formelsammlung nachschlagen:

$$P(B) = F(0) = \Phi\left(\frac{0-200}{200\sqrt{2}}\right) = \Phi\left(\frac{-\sqrt{2}}{2}\right) = 0{,}24$$

Analog könnte man nun die Siegwahrscheinlichkeit von Spieler A berechnen, da aber die Gewinnwahrscheinlichkeiten der beiden Spieler zusammen 1 ergeben müssen, können wir uns die vorigen Schritte ersparen:

$$P(A) = 1 - P(B) = 1 - 0{,}24 = 0{,}76$$



Nun wird also angenommen, dass der Spieler A mit einer Wahrscheinlichkeit von 76% gewinnt, oder anders gesagt in 100 Partien 76 Punkte erzielt. Würde nun unser Spieler A die Partie gewinnen, würde sich sein neues Rating also wie folgt berechnen lassen:

$$R'_A = R_A + k * (S_A - E_A) = 2000 + 15(1 - 0{,}76) = 2003{,}6$$

Dies wäre nun sein neues Rating, da bei dieser Spielstärke üblicherweise $k = 15$ gilt.

Nun wissen wir also, wie die Berechnung des Elo-Rating erfolgt, in einem der folgenden Kapitel werden wir uns noch intensiver damit beschäftigen und anhand einer Datenbank von 65000 Partien prüfen, ob die Berechnung eigentlich die Realität widerspiegelt.

2.4 ELO-Rating für Fußball

Ein weiteres, aber nicht so bekanntes, Rating System für Fußballnationalmannschaften ist das ELO-Rating System. Wie auch das der FIFA berechnet es die Punkte anhand des Ergebnisses und der Priorität des Spiels. Jedoch werden die Stärke der Konföderationen und die Position des Gegners nicht berücksichtigt. Dafür lässt es die Dominanz des Siegers, anhand der Tordifferenz, das erwartete Ergebnis, anhand der

Siegeswahrscheinlichkeit der Mannschaft, für die die Punktezahl berechnet wird, und auch den Heimvorteil in das Ergebnis miteinfließen. Anders als bei der Rating-Berechnungs Methode der FIFA, verjähren die Punkte nicht. Dafür verliert der Verlierer der Partie, bzw. bei Unentschieden die Mannschaft mit der höheren Siegeswahrscheinlichkeit, Punkte in Höhe der Punkte die das andere Team gewonnen hat.

All jene Daten werden in folgende Formeln eingesetzt:

$R_A = R'_A + P$

$P = K * G * (W - W_e)$

R_A: neuer Punktestand der Mannschaft A nach Spielende

R'_A: alter Punktestand der Mannschaft A

P: jene Punkte, die anhand des Spielergebnisses eruiert werden und dann zu den alten Punkten dazu addiert werden

K: Priorität des Spieles

K wird zu:

60: bei Weltmeisterschaftsendrunden

50: bei Kontinentalmeisterschaftsendrunden

40: bei WM- und Kontinentalmeisterschaftsqualifikationen

30: bei anderen Turnieren

20: bei Freundschaftsspielen

G: Tordifferenzfaktor $= \begin{cases} 1 & wenn\ TD < 2 \\ \frac{3}{2} & wenn\ TD = 2 \\ \frac{11+TD}{8} & wenn\ TD > 2 \end{cases}$

TD: Tordifferenz

W: Ergebnis des Spiels (1 für Sieg, 0,5 für Remis und 0 für Niederlage)

W_e: Gewinnerwartung der Mannschaft

$$W_e = \frac{1}{1 - 10^{\frac{R'_A - R'_B}{400}}}$$

R'_A: alte Ratingpunkte der Mannschaft für die die neuen Punkte ermittelt werden

R'_B: Ratingpunkte, vor Spielbeginn, der gegnerischen Mannschaft

Der Heimvorteil wird miteinkalkuliert, indem man bei der Berechnung der Siegeswahrscheinlichkeit derjenigen Mannschaft einen Punktebonus von 100 Punkten gewährt, welche zu Hause spielt, d.h.:

$R'_A = R_{A_x} + 100 * x_1$

$R'_B = R_{B_x} + 100 * x_2$

R_{A_x}: Punkte der Mannschaft A vor Matchbeginn ohne Miteinberechnung des Heimvorteils

R_{B_x}: Punkte der Mannschaft B vor Matchbeginn ohne Miteinberechnung des Heimvorteils

x_1 bzw. x_2 werden zu

1: bei vorhandenem Heimvorteil

0: bei nicht vorhandenem Heimvorteil

Vor- und Nachteile

Die Vorteile dieser Art von Punkteberechnung sind, dass man auch die Gewinnerwartungen der jeweiligen Mannschaften ermitteln kann. Gut ist auch, dass sowohl die Dominanz des Siegers, als auch der Heimvorteil von Wert sind.

Nachteile jenes Systems sind, dass alle je erlangten Punkte miteinberechnet werden, d.h. es ist keine Punkteentwertung vorhanden. Dadurch fließen auch Punkte in die Wertung ein, die erlangt wurden als die Mannschaft noch eine andere Stärke besaß. Dies führt zu leichter Stärkeverzerrung. Ein weiterer Nachteil der nicht vorhandenen Punkteentwertung ist, dass man Punkte horten kann um einen besseren Rang zu erhalten, d.h. man spielt so wenig Spiele wie möglich, um nahezu keine Punkte zu verlieren und um sich somit den bereits vorhandenen guten Platz zu sichern. Etwas was auch durch dieses Rating nicht zu eruieren ist, ist die Höhe der Wahrscheinlichkeit eines Unentschiedens.

3 Unsere Ergebnisse

3.1 Genauere Betrachtung des Elo-Systems und Änderungsvorschläge

Wir haben das Elo-System genauer unter die Lupe genommen und allen Einzelheiten des Systems auf den Zahn gefühlt. Basierend auf einen Datensatz von rund 65000 Partien aus den Jahren 2000-2008 haben wir uns Verbesserungsvorschläge überlegt.

3.1.1 Verbesserung der Standardabweichung

Nach ausführlicher Analyse der offiziellen Berechnungen der Elozahl haben wir erkannt, dass die Standardabweichung der Spielstärken mit 200 festgelegt wurde. Allerdings stimmt bei einem Wert von 200 die Punkteerwartung, die das Elo-System berechnet, nicht mit den tatsächlich erzielten Ergebnissen in unserer Datenbank überein. Damit die Punkteerwartung mit den tatsächlich erreichten Punkten übereinstimmt, müsste man die Standardabweichung auf $213{,}2$ erhöhen. Dadurch würde der Vorteil, den die schlechteren Spieler aus dem Standardabweichung von 200 ziehen, verschwinden, da somit die Punkteerwartung für sie etwas erhöht wird. Zum Beispiel wäre nun die Punkteerwartung für einen Spieler der 200 Elopunkte weniger besitzt als sein Gegner anstatt $0{,}242$ jetzt $0{,}254$.

3.1.2 Berücksichtigung des Vorteils des Weißspielenden

Ebenso wird im Elo-System der Vorteil, den der Weißspielende hat, nicht berücksichtigt, allerdings konnten wir feststellen, dass Spieler mit Weiß deutlich mehr Punkte erzielten als mit Schwarz. Ebenso sahen wir, dass diesen Vorteil Spieler mit einem höheren Rating besser ausnützen können als Spieler mit einer geringeren Wertungszahl. Daher müssten stärkere Spieler auch mehr von einer möglichen Korrektur erfasst werden. Wir versuchten diese beiden Aspekte in einer Formel zu vereinigen und kamen auf folgendes Ergebnis:

$$E_{w_b} = (E_w - 1200) * 1{,}0302 \ + 1200$$

E_{w_b}: Elozahl des Spielers mit Weiß, die für die Berechnung verwendet werden soll

E_w: tatsächliche Elozahl des Spielers mit Weiß

Nimmt man nun E_b anstatt E_w in der Berechnung, steigt die Punkteerwartung für den Weißspielenden. Allerdings steigt sie mehr, wenn der Spieler eine höheres Rating besitzt, da bessere Spieler den Weißvorteil effektiver nützen können. Nach dieser Korrektur haben Schwarz und Weiß durchschnittlich gleich viele Punkte erzielt, wie wir es mit unserer Erwartungsformel berechnet haben.

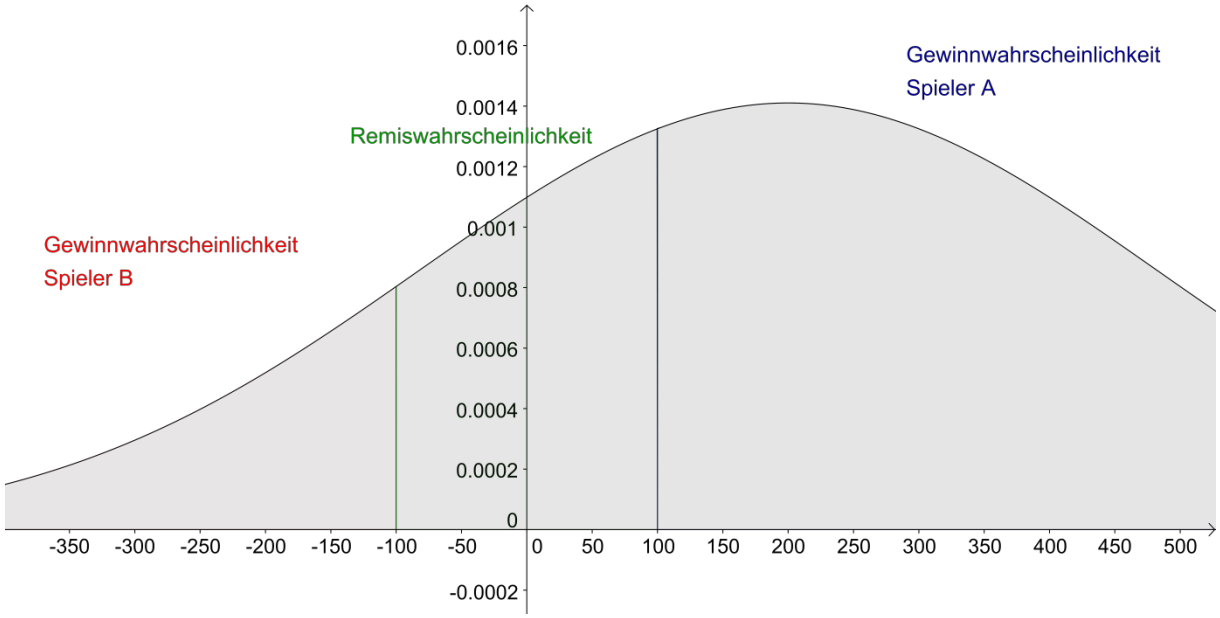
3.1.3 Gewichtung der neuen Partien

Wie auch bei den anderen Systemen ist auch bei Schachratings wichtig, wie viel die neuen Ergebnisse im Vergleich zu den Alten zählen. Dabei haben wir herausgefunden,

dass die Gewichtung der neuen Ergebnisse logischerweise vom Faktor k abhängt, allerdings auch von dem Elo-Unterschied zum Gegner. Neue Partien zählen natürlich am meisten, älterer immer weniger. So zählt bei Elogleichstand und einem Faktor von 10 das neueste Ergebnis 1,42% aller Partien, die zehntletzte Partie hat einen Wert von $1{,}25\%$, die hundertletzte nur mehr $0{,}34\%$ unter der Annahme, dass der Spieler 100 Partien gegen einen gleichstarken Spieler bestreitet. Zwischen der Gewichtung der neuen Partie und dem Faktor k besteht näherungsweise ein direkt proportionaler Zusammenhang, erst bei sehr großen Faktoren, die in der Praxis aber nie vorkommen, sind Abweichungen dieses Zusammenhanges erkennbar. Das heißt also, dass wenn der Faktor verdoppelt wird, auch die Gewichtung der neuesten Partie verdoppelt wird. Die Gewichtung der neuesten Partie nimmt aber auch ab, wenn ein größerer Elounterschied zum Gegner vorhanden ist. Bei einem Unterschied von 100 Elopunkten zählt die neueste Partie nur mehr $1{,}313\%$, bei 200 sogar nur mehr $1{,}098\%$ und bei 500 Elopunkten gar nur mehr $0{,}31\%$. Allerdings ist zu beachten, dass eine neuere Partie immer mehr gewichtet ist als eine ältere, auch wenn die neuere Partie gegen einen Gegner mit einem hohen Elounterschied gespielt wurde und die ältere mit einem niedrigen Elounterschied. Jedoch ist der Unterschied der Gewichtungen der beiden Partien nicht so groß, wie wenn die Partie mit dem niedrigen Elounterschied die jüngere Partie wäre.

3.1.4 Remiswahrscheinlichkeit

Bei unserem vorigen Modell des Elo-Systems sind wir davon ausgegangen, dass Spieler A gewinnt, sobald die Spielstärke, also die tatsächlich erbrachte Leistung, von Spieler A größer als die Spielstärke von Spieler B ist. Nun gibt es beim Schach aber auch noch Remis. Deswegen sind wir davon ausgegangen, dass die Spielstärke eines Spielers um einen gewissen Wert t größer sein muss als die des anderen, damit er gewinnt. Nehmen wir nun an, dass bei unserm vorigen Beispiel, wo Spieler A mit 2000 Elo gegen Spieler B mit 1800 Elo angetreten ist, die Partie dann Remis endet, wenn die beiden Spielstärken innerhalb von nur 100 Elopunkten liegen, also die Differenz der beiden im Bereich von -100 bis 100. Graphisch sieht dies wie folgt aus:



Nun wäre die Remiswahrscheinlichkeit die Fläche unter der Kurve im Intervall von -100 bis 100. Mit Hilfe der Verteilungsfunktion und Standardnormalverteilung können wir diesen Wert wie vorhin berechnen und bekommen ein Ergebnis von $21{,}74\%$. Die Gewinnwahrscheinlichkeit für B ist nun die Fläche unter der Kurve von $-\infty$ bis -100. Dies entspricht im diesen Fall eine Wahrscheinlichkeit von $14{,}44\%$. Die Punkteerwartung für Spieler B ist nun, die Gewinnwahrscheinlichkeit plus ein halb Mal die Remiswahrscheinlichkeit. In unserem Fall also:

$$E_b = 0{,}1444 + \frac{0{,}2174}{2} = 0{,}2532$$

Im vorigen Modell, als wir die Möglichkeit des Remis nicht betrachtet haben, kamen wir auf einen Erwartungswert von $0{,}242$. Nun sieht man, dass diese beiden Erwartungswerte kaum voneinander abweichen, deshalb wird die Remiswahrscheinlichkeit beim Elo-System, das beim Schach zum Einsatz kommt, so gut wie nicht beachtet.

Nun stellt sich aber die Frage wie man dieses t wählen sollte, um möglichst genau die Remiswahrscheinlichkeit der Praxis anzupassen. Anhand unserer Datenbank bemerkten wir, dass Spieler mit höheren Ratingzahlen häufiger Remis spielen, als schwächere Spieler, da nicht so geübte Spieler eher dazu neigen einen spielentscheidenden Fehler zu machen. Unter Berücksichtigung dieses Aspektes kamen wie nun auf folgende Formel:

$$t = \frac{(1200 - D)^{1{,}7948}}{4000 - \frac{(E_w + E_s)}{2}}$$

D: Betrag von der Differenz der beiden Elozahlen

E_w: Elozahl von Weiß

E_s: Elozahl von Schwarz

Anhand dieser Formel stimmten in unserer Datenbank, die berechneten Remiswahrscheinlichkeiten mit den tatsächlichen Ergebnissen genau miteinander überein.

3.1.5 Einstiegswert

Eine weitere Frage ist, wie man Spieler die noch keine Ratingzahl haben, bewertet. Hierbei versuchten wir heraus zu finden, wie stark das Einstiegsrating eines Spielers, den weiteren Verlauf seines Ratings beeinflusst. In der Praxis ist der Vorteil eines hohen Einstiegswertes nur für die ersten Partien gegeben. Wenn zum Beispiel ein Spieler A mit einem um 100 Punkte höheren Einstiegswert als ein Spieler B einsteigt und sie danach dieselben Leistungen erbringen, so wären in etwa 40 Partien von Nöten, sodass deren Ratingunterschied unter 10 Punkte fällt. Wäre der Startunterschied 200, so wäre dies nach 65 Partien der Fall. Auf lange Sicht ist der Einstiegswert sowieso zu vernachlässigen.

Hätten unsere beiden Spieler einen Startunterschied von 2000, der in der Praxis beinahe unmöglich ist, so würden sich deren Elozahl bei gleich guter Leistung nach 400 Partien nur mehr um einen tausendstel Elopunkt unterscheiden.

3.1.6 Änderungen durch 30-Züge Regel

Da seit 2008 bei den meisten Turnieren, bei denen die Weltspitze antritt, mit der 30 Züge-Regel gespielt wird, muss diese Formel für solche Turniere etwas angepasst werden. Diese Regel verbietet nämlich Remisangebote vor dem 30. Zug. Dadurch sinkt natürlich die Wahrscheinlichkeit, dass eine Partie Remis endet. Anhand von einer Datenbank mit 10000 neueren Partien versuchten wir eine leichtabgeänderte Formel zu entwerfen und kamen auf das Ergebnis, dass man bei der „alten" Formel für t nur einen Faktor von $0{,}886$ hinzufügen muss, damit sie wieder mit den Ergebnissen aus der Praxis übereinstimmt:

$$t_n = \frac{(1200 - D)^{1{,}7948}}{4000 - \frac{(E_w + E_s)}{2}} * 0{,}886$$

Ebenso nimmt dadurch auch der Vorteil des Weißspielenden etwas ab, sodass unsere vorhin erwähnte Formel für Turniere mit dieser Regel wie folgt abgeändert werden muss:

$$E_b = (E_w - 1200) * 1{,}01583126 \; + 1200$$

E_b: Elozahl des Spielers mit Weiß, die für die Berechnung verwendet werden soll

E_w: tatsächliche Elozahl des Spielers mit Weiß

3.1.7 Beispielpartie

Zum Abschluss wenden wir unsere entworfenen Formeln anhand einer Beispielpartie an:

Spieler A hat 2100 Elo und gewinnt mit Weiß gegen Spieler B, der 2000 Elo hat. Nehmen wir zuerst an, dass die 30 Züge-Regel bei dieser Begegnung nicht angewendet wird. Als Erstes berechnen wir den Vorteil den Spieler A daraus zielt, da er Weiß hat, ein. Seine „neue" Elozahl, die wir für unsere Berechnung verwenden ist nun:

$$E_{A_b} = (E_A - 1200) * 1{,}030159944 \; + 1200 = 2100 * 1{,}030159944 = 2163{,}3358824$$

Nun berechnen wir den Erwartungswert des Spielers A:

$$P(A) = \Phi\left(\frac{0 - (R_B - R_{A_b})}{\sigma\sqrt{2}}\right) = \Phi\left(\frac{0 - (2000 - 2163{,}3358824)}{213{,}2\sqrt{2}}\right) = 0{,}706$$

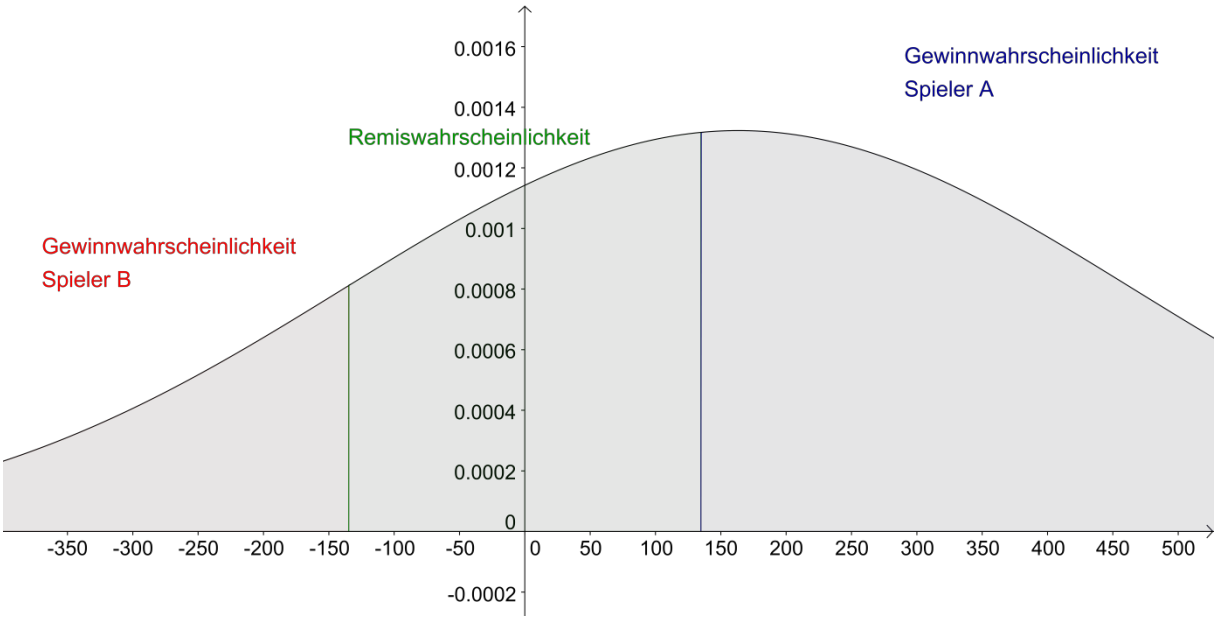
Das neue Rating ergibt sich nun:

$$R'_A = R_A + k * (S_A - E_A) = 2100 + 15 * (1 - 0{,}706) = 2104{,}41$$

Wollen wir nun auch die Remiswahrscheinlichkeit wissen, müssen wir den Wert t berechnen.

$$t = \frac{(1200 - D)^{1,7948}}{4000 - \frac{(E_w + E_s)}{2}} = \frac{(1200 - 163{,}3358824)^{1,7948}}{4000 - \frac{2163{,}335 + 2000}{2}} = 134{,}755$$

Nun müsste also die Spielstärke von A um $134{,}75$ Elo besser als die von B sein, damit A gewinnt und $134{,}75$ Elo schlechter, damit A verliert. Liegt die Differenz der beiden Spielstärken innerhalb dieser beiden Werte endet die Partie Remis. Die Wahrscheinlichkeit für ein Remis ist nun die grüne Fläche, die in diesem Fall $30{,}08\%$ ist.



Berechnen wir nun die neue Elozahl von Spieler A mit der „normalen“ Formel, wo kein Weißvorteil berücksichtigt wird und die Standardabweichung kleiner ist.

Der Erwartungswert ist nun:

$$P(A) = \Phi\left(\frac{0 - (R_B - R_{A_b})}{\sigma\sqrt{2}}\right) = \Phi\left(\frac{0 - (2000 - 2100)}{200\sqrt{2}}\right) = 0{,}638$$

Und das neue Rating:

$$R'_A = R_A + k * (S_A - E_A) = 2100 + 15 * (1 - 0{,}638) = 2105{,}43$$

Der Spieler A hätte nun bei dem herkömmlichen Modell in etwa einen Elopunkt mehr gewonnen als bei unserem Modell. Würde die Partie in einem Turnier mit der 30 Züge-Regel stattfinden, würde die Berechnung analog folgen, nur müssten wir die modifizierte Formel für den Weißvorteil verwenden und als Ergebnis hätte Spieler A ein Rating von $2105{,}29$ Elopunkten.

3.2 Berechnung der Wahrscheinlichkeit auf ein Unentschieden zweier Nationalmannschaften anhand ihres ELO-Ratings

Zuerst nehme man die Formel für die Siegeswahrscheinlichkeit für Mannschaft A aus der Formel für das ELO-Rating.

$$W_e = \frac{1}{1 + 10^{\frac{R'_B - R'_A}{400}}}$$

Anhand dieser Formel erkennen wir:

wenn $R'_A - R'_B > 0$, gewinnt Mannschaft A mit einer höheren Wahrscheinlichkeit

wenn $R'_A - R'_B < 0$, gewinnt Mannschaft B mit einer höheren Wahrscheinlichkeit

Die Formel für die Siegeswahrscheinlichkeit entsteht aus folgenden Überlegungen, ähnlich denen bei Schach. Nimmt man an, dass die beiden Spielstärken S der beiden Teams einer Extremwertverteilung mit Verteilungsfunktion

$$F_S(x) = e^{-10^{\frac{R-x}{400}}}$$

R: Rating der Mannschaft

unterliegen, so unterliegt die Differenz der Spielstärken beider Mannschaften $S_B - S_A$ einer logistischen Verteilung mit Verteilungsfunktion:

$$F_{S_B - S_A}(x) = \frac{1}{1 + 10^{\frac{R'_B - R'_A - x}{400}}}$$

Die Gewinnwahrscheinlichkeit für Mannschaft A beträgt:

$$P(S_B - S_A < 0) = F_{S_B - S_A}(0) = \frac{1}{1 + 10^{\frac{R'_B - R'_A}{400}}}$$

Um die Wahrscheinlichkeit für ein Unentschieden zu erhalten, muss man in diese Berechnung, neben R'_A und R'_B, noch eine dritte Konstante t einführen. Aus Erfahrung weiß man, dass die Wahrscheinlichkeit für ein Unentschieden immer höher ist als die Siegeswahrscheinlichkeit der Außenseitermannschaft. Weiters weiß man noch, dass die Wahrscheinlichkeit auf ein Unentschieden umso größer wird, je näher die Stärken der Mannschaften beieinander liegen.

Anhand dieser Erkenntnis, lässt sich auf Folgendes schließen:

wenn $S_A - S_B > t$, gewinnt Mannschaft A

wenn $-t < S_A - S_B < t$, endet das Spiel Unentschieden

wenn $S_A - S_B < -t$, gewinnt Mannschaft B

Die Wahrscheinlichkeiten für diese Ereignisse werden mithilfe der Verteilungsfunktion ähnlich wie beim Schach berechnet.

Nun galt es noch die Konstante t zu bestimmen. Hierzu muss man sich Quoten von Wettanbietern im Internet ansehen und auf Prozent umrechnen. Dies bewerkstelligt am besten indem man 1 durch die angebotene Quote dividiert. Wenn man die so erhaltenen Zahlen nun miteinander addiert fällt einem auf, dass die so erhaltene Zahl größer als eins ist. Die Differenz zu eins ist der Selbsterhalt der Wettanbieter. Um nun die wahren Wahrscheinlichkeiten zu erhalten, muss man die falschen Wahrscheinlichkeiten durch die zuvor erhaltene Zahl dividieren.

$$S = 1 - \left({Q_1}^{-1} + {Q_2}^{-1} + {Q_3}^{-1}\right)$$

S: Selbstbehalt

Q_1: Quote 1

Q_2: Quote 2

Q_3: Quote 3

Wahrscheinlichkeit für $Q_1 = 1 + S$

Der nächste Schritt ist es die Konstante t und die Zahl durch die $(R_a - R_b - t)$ dividiert $(= t')$ wird, so anzupassen, sodass die ausgerechneten Wahrscheinlichkeiten denen des Wettanbieters gleichen. Da aber nicht jeder Wettanbieter die gleiche Formel zur Errechnung der Quoten benutzt, muss man die zuvor erwähnten Schritte so oft wiederholen und die dabei erlangten Zahlen für t und t' so oft niederschreiben, bis man genügend Daten gesammelt hat. Zum Schluss errechnet man sich jeweils den Mittelwert von t und von t'. Wir kamen durch unsere Recherchen auf folgende Zahlen:

$t = 184{,}55$

$t' = 600$

3.3 Anwendung eines ELO-Ratingsystems auf eine Fußball-Liga

Zu Beginn betrachteten wir Ratings verschiedener Fußballclubs und fanden einen Quotenrechner im Internet. Man konnte zwei beliebe Teams auswählen und bekam Wahrscheinlichkeiten heraus, wie wahrscheinlich ein Heimsieg, ein Unentschieden und einen Auswärtssieg ist. Das Interessante war nun herauszufinden, wie man sich selbst die Wahrscheinlichkeiten und die damit verbundenen Wettquoten ausrechnen kann. Schnell bemerkten wir auch, dass die Quoten, wie man sie kennt, nicht die errechneten sind, sondern die Buchmacherquoten, was so viel heißt wie, dass sie verkleinert wurden um

Gewinne zu reduzieren. Bei den verschiedenen Rechnungen stellte sich auch heraus, dass die Wahrscheinlichkeit eines Unentschiedens bei dem Online-Rechner zwischen 10% und 35% liegt. Nun versuchten wir experimentell durch die gleichen Ratings auf ähnliche Wahrscheinlichkeiten zu kommen. Wir versuchten zuerst ein ELO-System mit zwei verschiedenen Variablen t_1 und t_2 zu verwenden, die das Intervall eines Unentschiedens eingrenzten, kamen aber auf kein einheitlich verwendbares Ergebnis. Durch Umformungen und Veränderungen kamen wir zu einem geeigneten System.

Nun folgte die Suche nach einem Datenblock, den man analysieren konnte. Die Wahl fiel auf die erste spanische Fußballliga, die Primera Division. Da Daten einer ganzen Saison benötigt wurden, nahmen wir nicht die aktuellen Ratings und Spiele, da diese Daten nicht ausgereicht hätten. Deswegen beschlossen wir Daten der letzten Saison 2010/11 zu verwenden. Wir haben den Spielplan auf Microsoft Excel übertragen und als Startratings der Mannschaften jene Ratings von www.soccer-rating.com genommen, und für die erste Runde jeder Mannschaft bei den Heim- und Auswärtsspielen verwendet. Nun rechneten wir die Erwartungswerte EW der Mannschaften und verglichen jene mit den Erwartungen im Internet und den Spielausgängen. Dadurch konnten wir eine Anpassung der Parameter K und α durchführen. Nun verwendeten wir folgende Formel jede Runde, um die neuen Heim- und Auswärtsratings, die in diesem Ratingsystem unterschieden werden, auszurechnen:

$$R'_A = R_A + \big(G * K * (AW - EW)\big)$$

R'_A: Rating nach dem Spiel

R_A: Rating vor dem Spiel

G: Tordifferenzfaktor $= \begin{cases} 1 & wenn\ TD < 2 \\ \frac{3}{2} & wenn\ TD = 2 \\ \frac{11+TD}{8} & wenn\ TD > 2 \end{cases}$

TD: Tordifferenz $= |T_A - T_B|$

T_A: erzielte Tore

T_B: erzielte Tore des Gegners

K: Konstante $= 40$

AW: Ausgangswert $= \begin{cases} 1 & wenn\ gewonnen \\ 0{,}5 & wenn\ unentschieden \\ 0 & wenn\ verloren \end{cases}$

EW: Erwartungswert $= P(HS) + 0{,}5 * P(U)$

$$P(HS) = \frac{1}{1 + \frac{\pi_2}{\pi_1} + \alpha * \sqrt{\frac{\pi_2}{\pi_1}}}$$

$$P(U) = \frac{\alpha * \sqrt{\frac{\pi_2}{\pi_1}}}{1 + \frac{\pi_2}{\pi_1} + \alpha * \sqrt{\frac{\pi_2}{\pi_1}}}$$

α: Konstante $= 0{,}8$

$\pi_1 = 10^{R_A/400}$

$\pi_2 = 10^{R_B/400}$

R_B: Rating des Gegners

Man kann sehen, dass prinzipiell die ELO-Formel verwendet wurde. Allerdings wurde noch die Wahrscheinlichkeit für ein Unentschieden eingearbeitet. Dies geschah folgenderweise:

Die Wahrscheinlichkeit eines Sieges aus der ELO-Formel $P(HS) = \frac{1}{1+10^{(R_B-R_A)/400}}$ kann man auch mit π_1 und π_2 angeben, wie oben angeführt. Aus dieser Formel kann man dann mit π_1 erweitern und erhält folgendes: $P(HS) = \frac{1}{1+\frac{\pi_2}{\pi_1}} = \frac{\pi_1}{\pi_1+\pi_2}$. Dies kann man auch mit der Wahrscheinlichkeit eines Sieges des Gegners machen und erhält: $P(AS) = \frac{\pi_2}{\pi_1+\pi_2}$.Nun kann man ein Verhältnis zwischen $P(HS)$ und P(AS) feststellen: $\frac{P(HS)}{P(AS)} = \frac{\pi_1}{\pi_2}$.

Wenn man nun eine Wahrscheinlichkeit eines Unentschiedens einbringen möchte, kann man zunächst den Nenner von $P(HS)$ und $P(AS)$ mit dem gleichen Summanden erweitern, ohne das Verhältnis zu verändern: $P(HS) = \frac{\pi_1}{\pi_1+\pi_2+u}; P(AS) = \frac{\pi_2}{\pi_1+\pi_2+u}$.

Nun können wir die Wahrscheinlichkeit für ein Unentschieden $P(U)$ so einbringen, sodass: $P(HS) + P(AS) + P(U) = 1$. Somit kann man die Wahrscheinlichkeit eines Unentschiedens folgendermaßen festlegen: $P(U) = \frac{u}{\pi_1+\pi_2+u}$. Jenen Summanden u kann man nun noch abhängig von den Faktoren α, $\pi_1$und π_2 angeben: $u = \alpha * \sqrt{\pi_1 * \pi_2} \rightarrow P(U) = \frac{\alpha*\sqrt{\pi_1+\pi_2}}{\pi_1+\pi_2+\alpha*\sqrt{\pi_1*\pi_2}}$. Den Faktor α kann man nun auch als Verhältnis sehen, wie wahrscheinlich 2 Unentschieden im Gegensatz zu einem Sieg und einer Niederlage bei einer Wiederholung eines Versuchs sind:

$$P(U) * P(U) = \frac{\alpha^2 * \pi_1 * \pi_2}{(\pi_1 + \pi_2 + \alpha * \sqrt{\pi_1 * \pi_2})^2}$$

$$P(HS) * P(AS) = \frac{\pi_1 * \pi_2}{(\pi_1 + \pi_2 + \alpha * \sqrt{\pi_1 * \pi_2})^2}$$

$$\frac{P(U) * P(U)}{P(HS) * P(AS)} = \alpha^2$$

Somit kann man erkennen, dass ein größeres α für eine größere Wahrscheinlichkeit eines Unentschiedens sorgt. In unserer speziellen Anwendung der Formel kann man also das α für die jeweilige Fußballliga individuell einstellen.

Der letzte Schritt um auf die oben aufgeführten Formeln für $P(HS)$ und $P(U)$ zu kommen, ist die Erweiterung des Zählers und Nenners mit $\frac{1}{\pi_1}$:

$$P(HS) = \frac{\frac{\pi_1}{\pi_1}}{\frac{\pi_1 + \pi_2 + \alpha * \sqrt{\pi_1 * \pi_2}}{\pi_1}} = \frac{1}{1 + \frac{\pi_2}{\pi_1} + \alpha * \sqrt{\frac{\pi_2}{\pi_1}}}$$

$$P(U) = \frac{\frac{\alpha * \sqrt{\pi_1 + \pi_2}}{\pi_1}}{\frac{\pi_1 + \pi_2 + \alpha * \sqrt{\pi_1 * \pi_2}}{\pi_1}} = \frac{\alpha * \sqrt{\frac{\pi_2}{\pi_1}}}{1 + \frac{\pi_2}{\pi_1} + \alpha * \sqrt{\frac{\pi_2}{\pi_1}}}$$

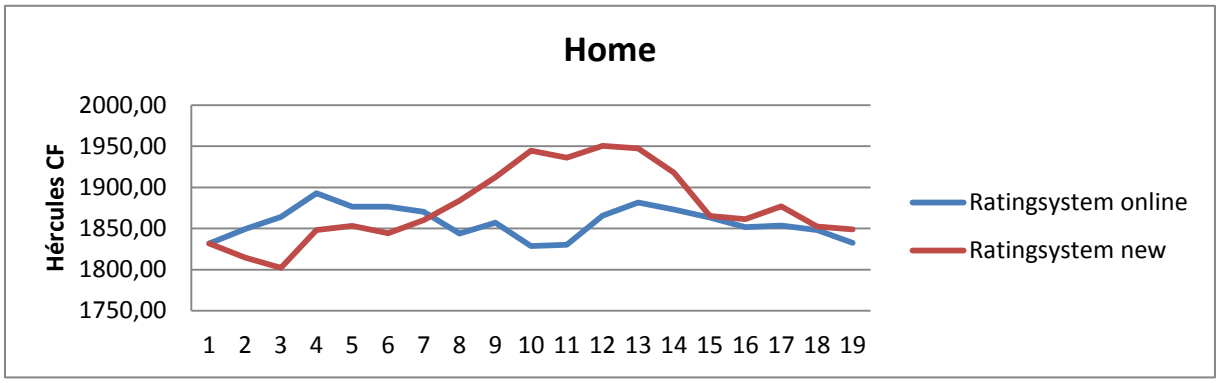
Die Formel sagt nun aus, wie groß das Rating eines Teams nach einem Spiel ist. Dieses neue Rating setzt sich zusammen aus dem alten Rating R_A des Teams zu dem ein Wert, der sich aus verschiedenen Faktoren zusammen setzt und darüber entscheidet wie viele Punkte das Team gewinnt oder verliert, addiert wird. Diese Faktoren sind zum einen der Tordifferenzfaktor G, den man aus der Tordifferenz wie in der Formel angegeben ausrechnen kann, dann der Konstanten K, die in diesem Fall 40 ist, aber von Liga zu Liga verschieden sein kann. Dieser Faktor K ist in der ELO-Formel für Fußball für die Gewichtung eines Spiels verantwortlich, da aber in einer Liga alle Spiele gleich viel zählen, ist der Faktor in dieser Formel konstant. Der letzte Faktor $(AW - EW)$ besteht aus dem Ausgangswert, wie ein Spiel aus der Sicht der Mannschaft mit dem Rating R_A ausgegangen ist und dem Erwartungswert für diese Mannschaft, den man errechnet hat. Anstatt der üblichen Verteilung mit 3 Punkten für einen Sieg, 1 Punkt für ein Unentschieden und 0 Punkte für ein verlorenes Spiel, werden in diesem Rating aber andere Werte für den Ausgangswert verwendet, nämlich 1 Punkt für einen Sieg, $0{,}5$ Punkte für ein Unentschieden und 0 Punkte für einen Sieg des Gegners. Der Erwartungswert EW ist die Wahrscheinlichkeit eines Sieges $P(HS)$ der Mannschaft mit dem Rating R_A, zu der noch 50% der Wahrscheinlichkeit eines Unentschiedens $P(U)$ addiert werden. Diese Wahrscheinlichkeiten kann man wie in der Formel angegeben. Sie wird aus den Ratings der zwei Mannschaften und einer weiteren Konstanten, die entscheidet, wie groß die Wahrscheinlichkeit eines Unentschiedens ist und auch von Liga zu Liga variieren kann, berechnet.

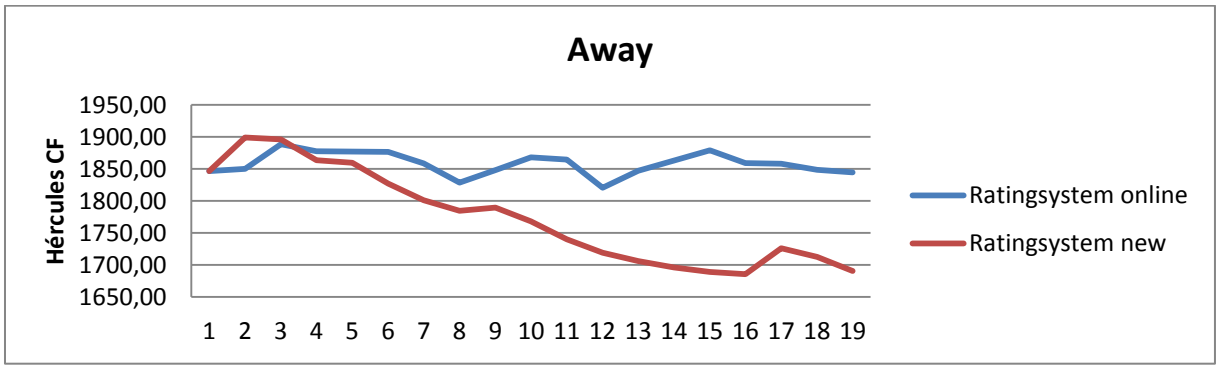
Nachdem wir diese Formel auf alle 38 Runden der Primera Division der letzten Saison angewendet hatten, sammelten wir in einer weiteren Tabelle auch die Ergebnisse einer zufälligen Mannschaft, die in diesem Fall Hércules CF war. Nun betrachteten wir die Ratingentwicklung sowie die Erwartungswerte, die wir berechneten, mit den Ausgangswerten und stellten sie den Erwartungswerten aus dem Internet gegenüber. Die quadratische Gesamtabweichung unseres Ratingsystems unterscheidet sich bei dieser Mannschaft nur um einen minimalen Wert gegenüber der quadratischen Gesamtabweichung, die wir mit Daten aus dem Internet errechnet haben.

4878	0,355945122	0,597361398	0,402638602	0,5	0,5	-0,144054878	-0,097361398	0,020751808	0,009479242
5152	0,348484848	0,674016905	0,325983095	1	0	0,348484848	0,325983095	0,12144169	0,106264978
0000	0,34375	0,613043655	0,386956345	1	0	0,34375	0,386956345	0,118164063	0,149735213
3939	0,260606061	0,631164642	0,368835358	1	0	0,260606061	0,368835358	0,067915519	0,136039522
9033	0,329380967	0,610860927	0,389139073	0	1	-0,670619033	-0,610860927	0,449729887	0,373151072
2414	0,367267586	0,58274896	0,41725104	0,5	0,5	-0,132732414	-0,08274896	0,017617894	0,00684739
9091	0,127840909	0,721056897	0,278943103	1	0	0,127840909	0,278943103	0,016343298	0,077809254
7619	0,380952381	0,590261551	0,409738449	0,5	0,5	-0,119047619	-0,090261551	0,014172336	0,008147148
						0,527245014	2,853494941	5,697687078	6,210998745

Quadratische Gesamtabweichung online	5,697687078
Quadratische Gesamtabweichung neu	6,210998745

Die Charaktere der Ratingänderungen sind aber dennoch sehr verschieden, wie man in den folgenden Grafiken gut erkennen kann:





Physik

Analyse der Inhalte der Jules Verne Romane

Christoph, Felix, Jakob, Lissa, Paul, Uli
Teamleiter: Kraft Daniel

Inhalt

Einleitung

Unsere Gruppe hat sich mit zwei Büchern und den darin vorkommenden Fakten und den entnommenen Problemstellungen des französischen Autors Jules Verne beschäftigt. Wir wählten vier verschiedene Aufgabenstellungen aus, welche wir souverän bearbeiteten. Aus dem Buch „Von der Erde zum Mond“ (französischer Originaltitel: De la Terre à la Lune) entnahmen wir die Angaben zu einer menschengefertigten Kanone deren Ziel es war ein bemanntes Projektil zum Mond zu schießen. Zunächst wurde vom Kanonenklub Baltimore ein Versuch mit zwei speziell ausgewählten Tieren gestartet. Dieser verlief erfolgreich, abgesehen davon dass der Kater das Eichhörnchen auffraß. Daraufhin wurde das bestehende Versuchsprogramm geändert und erweitert so dass schlussendlich ein für menschliche Versuchspersonen ausgestattetes Projektil gebaut wurde. Dieses schaffte es aber nicht bis zum Mond, da es in der Erdatmosphäre abgelenkt wurde, so die vorausberechnete Bahn verfehlte und um den Mond zu kreisen begann. Die uns zugeteilte Aufgabenstellung beinhaltete die von Jules Verne berechneten Werte zu überprüfen und wenn nötig so zu optimieren, um das Projektil bis zum Mond fliegen zu lassen. Es ergaben sich interessante Ergebnisse, welche durch verschiedene Kalkulationsprogrammen graphisch dargestellt und animiert wurden.

Im zweiten von uns bearbeiteten Werk Jule Vernes kollidiert ein Komet mit der Erde und nimmt einige Gesteinsmassen mit auf seine „Reise durch die Sonnenwelt“ (französischer Originaltitel: Hector Servadac). Zunächst bemerken die Mitreisenden nicht einmal, dass sie sich nicht mehr auf der Erde sondern auf einem Kometen befinden. Da sich aber verschiedene Faktoren, wie zum Beispiel Schwerkraft, Luftdruck und Erdachsenrichtung in kurzer Zeit verändern, bemerken sie nach und nach, dass etwas nicht mit rechten Dingen zugeht. Auch die Größe des Mondes scheint sich vervielfacht zu haben und die Sonne steigt jeden Morgen anstatt im Osten im Westen auf. Im Buch werden als Peripheriepunkte einerseits die Umlaufbahn der Venus anderseits die Umlaufbahn Jupiters angegeben. Spannend ist auch der daraus resultierende Temperaturverlauf, welcher von uns genauestens unter die Lupe genommen wurde. Da die Temperatur im Bereich der Jupiterbahn eiszeitartige Züge annimmt müssen die Bewohner in das Höhlensystem eines aktiven Vulkans flüchten. Um dem tödlichen Aufschlag zu entrinnen konstruierten die wackeren Weltraumpioniere einen Heißluftballon, der kurz nach dem Eintritt in die Atmosphäre abheben sollte um so den Höllenritt zu überleben.

Die aus diesem kurz beschriebenen Inhalt resultierenden Fragestellungen lauteten:

Modelliere ein Modell für die Orbitale der Planeten und den von Jules Verne beschriebenen Kometen.

Versuche den Temperaturverlauf an unterschiedlichsten Abständen des Kometen zur Sonne zu modellieren.

Versuch ein Modell der im Buch beschriebenen Atmosphäre des Kometen zu erstellen.

Kanone

In dem Buch „From the earth to the moon“ beschreibt Jules Verne einen Klub der sich mit einer Kanone in einem Projektil auf den Mond schießen und diesen kolonisieren möchte. Durch die vielen technischen Details die Jules Verne angibt haben wir den Flug der Kanone rekonstruiert und die Möglichkeit auf einen wirklichen Flug zu dem Mond modelliert und optimiert.

Angaben aus dem Buch:

Sprengstoff:

$1{,}8*10^5$ Tonnen

Kanonenrohr:

300 Meter tief

3 Meter Durchmesser

Annahmen aus dem Buch:

$6*10^9$ Liter Gas

Kein Luftwiderstand nach der Explosion

Als erstes galt es das Verhalten der Kanone in einer Differentialgleichung zu formulieren, um von dem aus dem Gasvolumen berechneten Druck die Beschleunigung des Projektils zu berechnen.

$$p = \frac{v_{gas}}{v_{kammer}}$$

Die Beschleunigung ist aus dem Druck berechenbar:

$$a(h) = \frac{n * R * T}{h * m}$$

Wobei: $$nRT = v_{kammer} * p$$

Die zweite Ableitung des Weges, bei uns die Höhe, ergibt die Beschleunigung:

$$h'' = a$$

Als Differenzialgleichung ergibt sich:

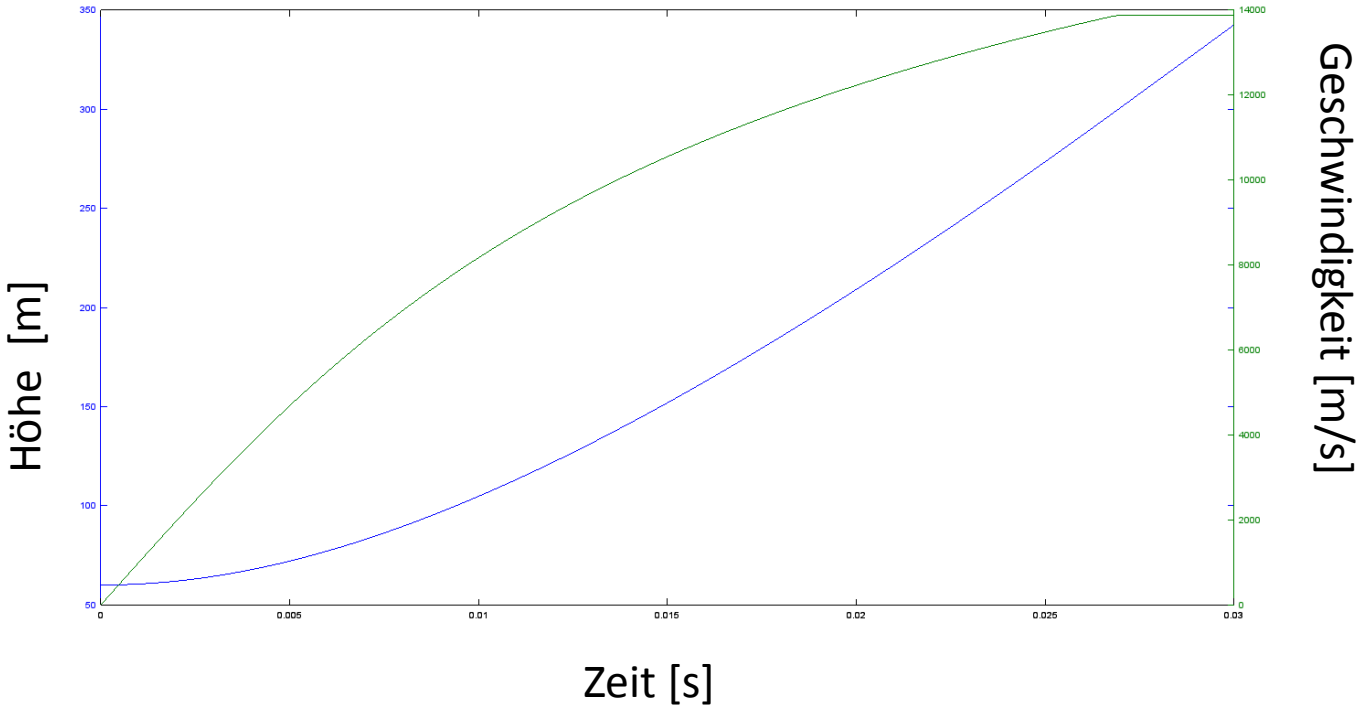
$$h''(t) = \frac{n * R * T}{h * m}$$

Wenn man diese nun algebraisch auflöst, erhält man folgende Gleichung:

Kanone

$$\left\{\left\{h[t] \to e^{\frac{-m\,C[1]-2\,n\,R\,T\,\text{InverseErf}\left[-\frac{\frac{m\,C[1]}{i\,e^{2\,n\,R\,T}}\sqrt{n}\,\sqrt{\frac{2}{\pi}}\,\sqrt{R}\,\sqrt{T}\,(t+C[2])}{\sqrt{m}}\right]^2}{2\,n\,R\,T}}\right\}, \left\{h[t] \to e^{\frac{-m\,C[1]-2\,n\,R\,T\,\text{InverseErf}\left[\frac{\frac{m\,C[1]}{i\,e^{2\,n\,R\,T}}\sqrt{n}\,\sqrt{\frac{2}{\pi}}\,\sqrt{R}\,\sqrt{T}\,(t+C[2])}{\sqrt{m}}\right]^2}{2\,n\,R\,T}}\right\}\right\}$$

Der Graph dieser Funktion :



Laut diesem Graphen steht einem Mondflug nichts im Weg. Es wurde allerdings die Erdgravitation vernachlässigt. In dem nächsten Schritt der Modellierung dieses Problems haben wir nun die Erdgravitation eingebunden.

Diese lautet:

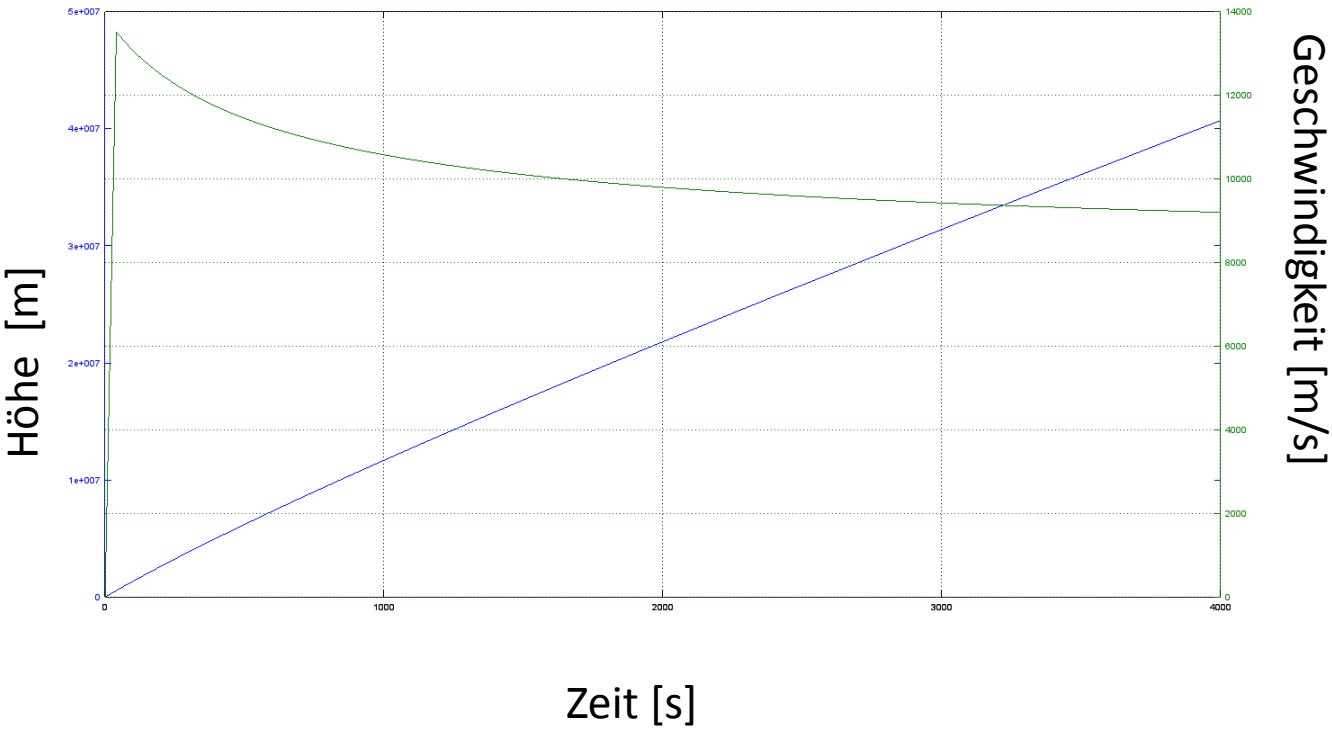
$$g = \frac{G * M}{r^2}$$

In die Differentialgleichung eingebunden ergibt sich:

$$h''(t) = \frac{n * R * T}{h(t) * m} - \frac{G * M}{(r + h)^2}$$

Eine algebraische Auflösung diese Differentialgleichung war nicht mehr möglich.

Numerisch aufgelöst in einem Graphen erhält man:



Auch mit dieser Rechnung erreicht das Projektil den Mond. Allerdings gibt es noch zwei weitere Komponenten die den Flug des Projektils wesentlich beeinflussen. Diese sind der Luftwiderstand und die Gravitationskraft des Mondes. Der Luftwiderstand wird folgendermaßen berechnet:

$$F_{luft} = c_d * \frac{\rho}{2} * A * v^2$$

Der Koeffizient unseres Projektils, welches als perfekter Kegel angenommen wurde, ist:

$$c_d = 0{,}5$$

Um diese Formel aber bei unser Problem anwenden zu können, müssen wir sie noch für die Beschleunigung umformen:

$$a_{luft} = \frac{c_d * \frac{\rho}{s} * A * v^2}{m}$$

Diese ist als negativ anzusehen, da sie unser Projektil abbremst.

Die Gravitationsbeschleunigung des Mondes wird ähnlich wie bei der Erde berechnet:

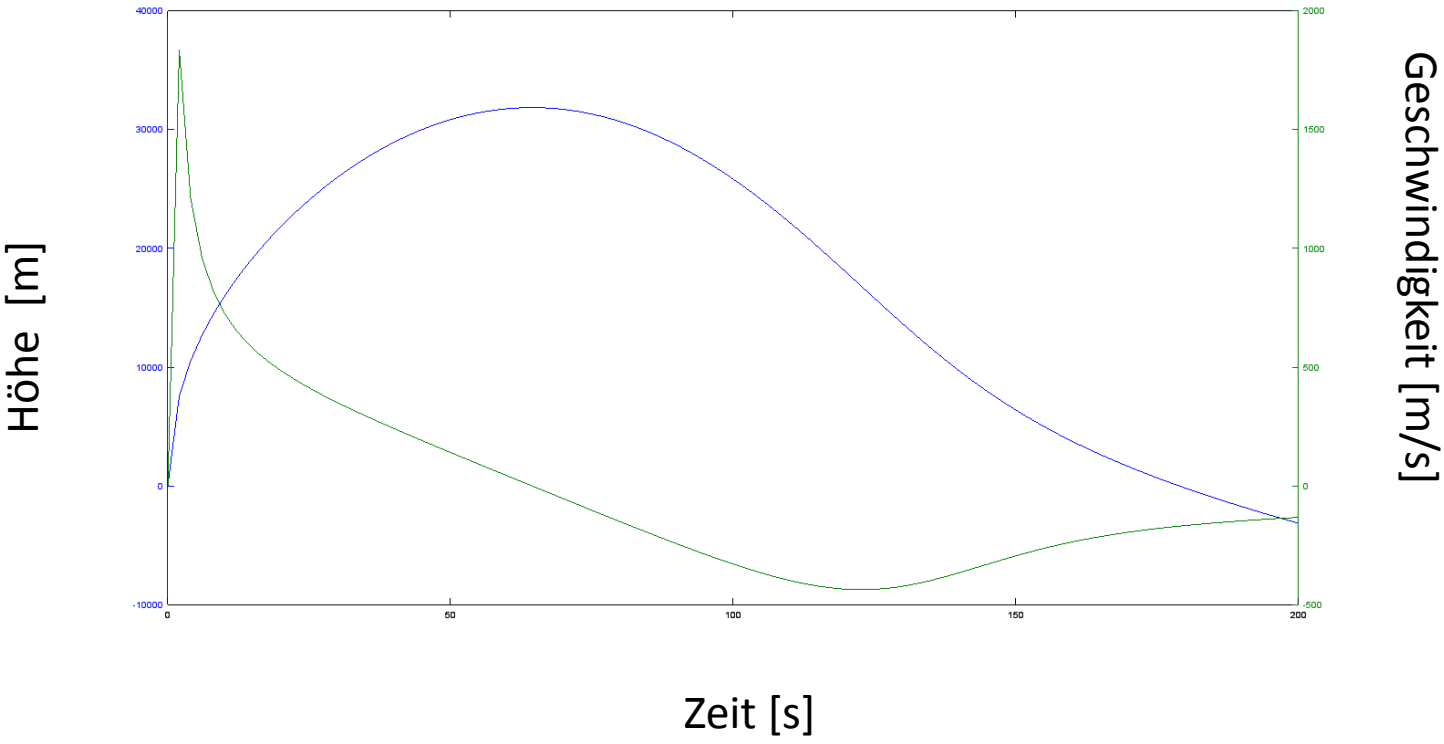
$$g_{mond} = \frac{G * M_{mond}}{(r_{mond} + d_{e-m} - h)^2}$$

Die Beschleunigung ist positiv da sie das Projektil zum Mond zieht.

Unter dieser Berücksichtigung erhalten wir folgende Differentialgleichung

$$h''(t) = \frac{n*R*T}{m*h(t)} + \frac{G*M_{mond}}{(r_{mond}+d_{e-m}-h(t))^2} - \frac{G*M}{(r+h(t))^2} - \frac{v^2*c_d*A*\frac{\rho}{2}}{m}$$

Der Graph dieser Funktion:



Dieses Diagramm zeigt nun, dass das Projektil unter den „wirklichen“ Verhältnissen und den Angaben von Jules Verne den Mond nicht erreichen kann.

Jules Verne hatte allerdings in noch einem Punkt Unrecht. Er gibt an, dass die Schießbaumwolle nach der Explosion 6 Milliarden Liter Gas erzeugt. Das tatsächliche Schwaden Volumen von Schießbaumwolle liegt aber bei:

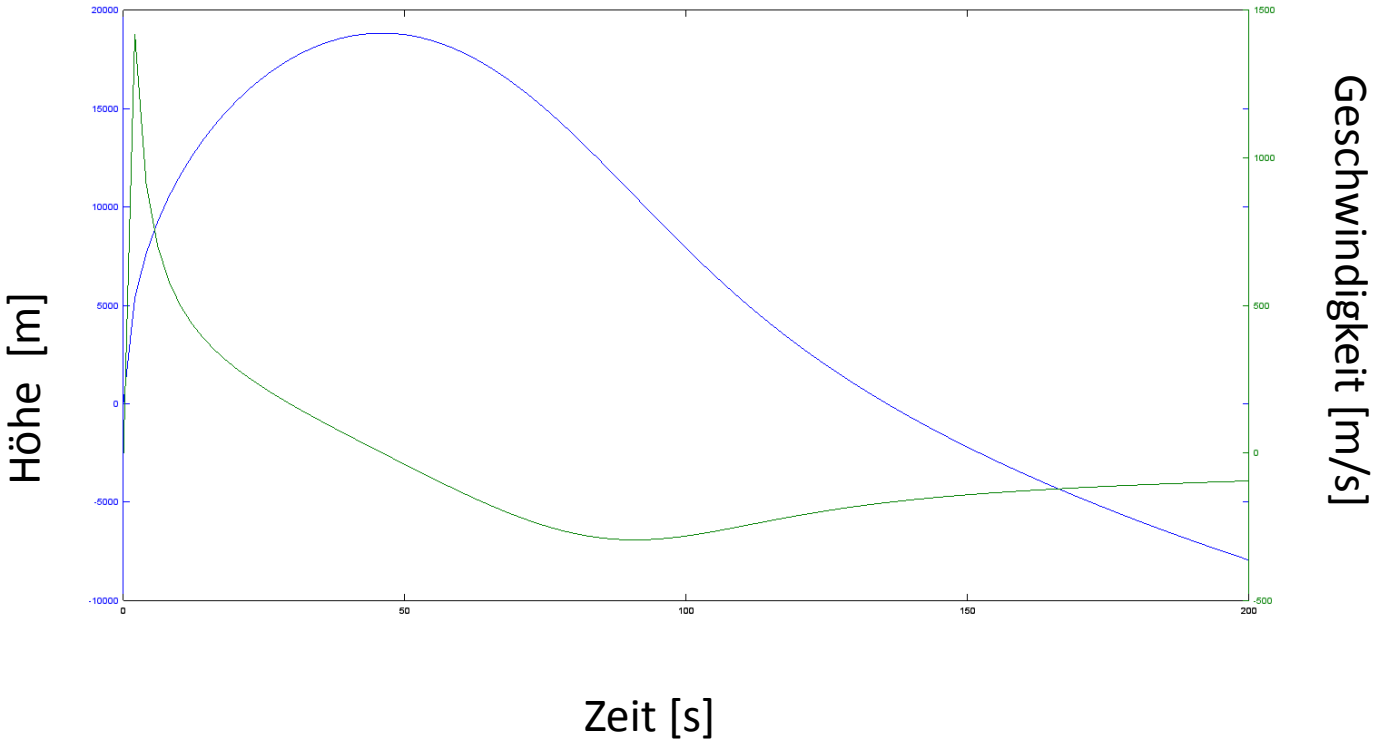
$$V_{sbw} = 869\frac{liter}{kg}$$

Mit diesem Schwaden Volumen ergibt sich ein neuer Druck von:

$$\rho = 1.564 * 10^{11} \ \ pascal$$

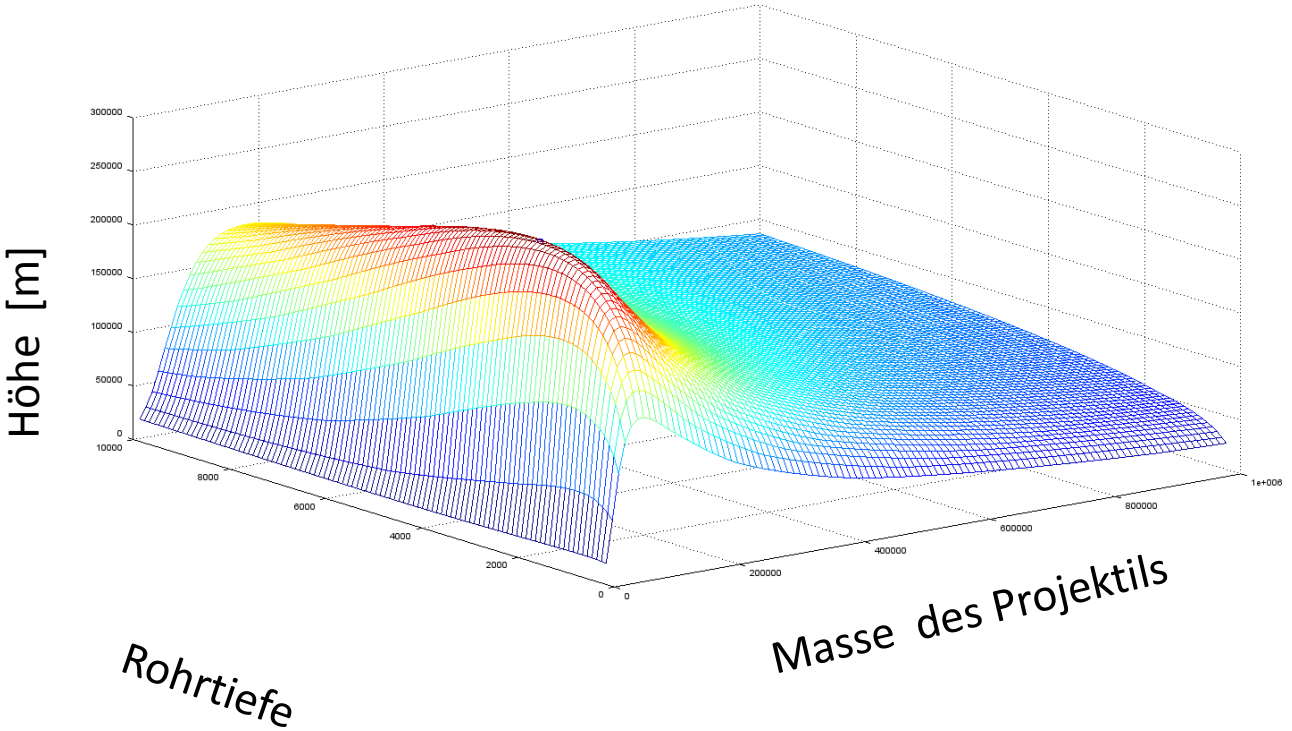
Dieser ist um den Faktor 3,8 kleiner als der von Jules Verne angenommene Druck.

Das daraus errechnete Diagramm :



zeigt, dass das Projektil wesentlich weniger hoch fliegen würde.

Der letzte Punkt unserer Modellierung an der „Jules Kanone“ war die Optimierung der Kanone. Es hat sowohl die Rohrlänge als auch die Masse des Projektils einen Einfluss auf die Flughöhe des Projektils. Bei beiden Parametern gibt es ein optimales Maximum.
Der 3D Graph stellt die Masse und



die Rohrlänge in Abhängigkeit zu der Maximalen Flughöhe dar:

Das Optimum liegt bei:

Masse=85500 kg
Rohr Höhe=2651,5 Meter

Resultierende Flughöhe=$2{,}77*10^5$

Es ist zu beachten, dass eine Überlänge des Rohres wieder zu einer geringeren Flughöhe führt. Dies folgt daraus, dass die Höhe von der Erdoberfläche gemessen wird und das Projektil bei einem längeren Rohr mehr Rohrweg zurücklegen muss ohne ,dass diese Höhe in die Flughöhe einfließt.

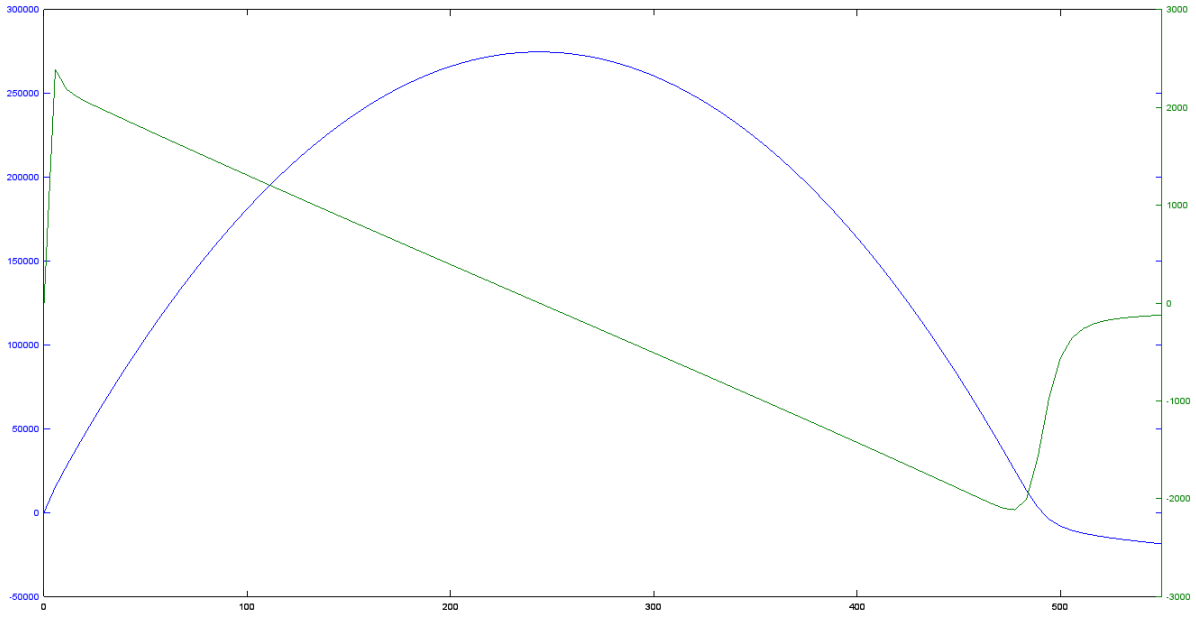
Auch Beachtenswert ist, dass das Projektil bei erhöhter Masse höher fiegt als mit niedriger Masse. Das kann erklärt werden wenn man sich die Formel zur Berechnung des Fluges noch einmal genauer anschaut:

$$h''(t) = \frac{n*R*T}{m*h(t)} + \frac{G*M_{mond}}{(r_{mond}+d_{e-m}-h(t))^2} - \frac{G*M}{(r+h(t))^2} - \frac{v^2*c_d*A*\frac{\rho}{2}}{m}$$

Die Masse kommt zweimal im Nenner vor. Das eine Mal verringert es die bei der Explosion der Schießbaumwolle entstehende Beschleunigung, das andere Mal den Luftwiderstand.

Da der Luftwiderstand negativ ist fliegt das Projektil mit einer Masse von 85 500 kg höher als ein Projektil mit einer Masse von 10 000 oder 100 000 kg.

Es ist allerdings trotz der Optimierung nicht möglich mit dieser Menge an Schießbaumwolle und diese Form und Größe des Projektils den Mond zu erreichen. Dies ist im untenstehenden Diagramm verdeutlicht.



Atmosphärendruck

Laut Jules Vernes Angaben ist der Luftdruck am Kometen „Gallier“ wesentlich geringer als auf der Erde. Diese Folgerung ist auf das kleinere Volumen des Kometen zurückzuführen. Die daraus resultierende Masse beträgt mit 1,98*10^11 kg ebenfalls nur einen Bruchteil der Erdmasse (5,974*10^24kg). Um auf den Luftdruck zurück zu kommen, kann man die Angabe, dass Wasser bei 66°C und nicht wie auf der Erde bei ca. 100°C kocht, als gegeben nehmen.

Wir haben den Atmosphärendruck und dessen Abnahme in Bezug auf die steigende Höhe rechnerisch ermittelt. Dafür haben wir in Octave ein Programm geschrieben, welches die jeweiligen Werte des Druckes p auf die vorergehenden Werte aufbaut.

$$p(z) = p(z+h) + \frac{g * m}{A}$$

Um eine möglichst große Genauigkeit des Graphen zu ermitteln, ließen wir die Spannweiten der einzelnen Abstände (h) gegen 0 laufen. Da sich die Masse aus dem Produkt der Dichte (p), der Fläche(A) und der Höhe (z) ergibt, erhält man folgende Formel:

$$m = \rho * A * h$$

In die obere Formel eingesetzt:

$$p(z) = p(z+h) + g\ \frac{p}{RsT} * h$$

Weiter umgeformt:

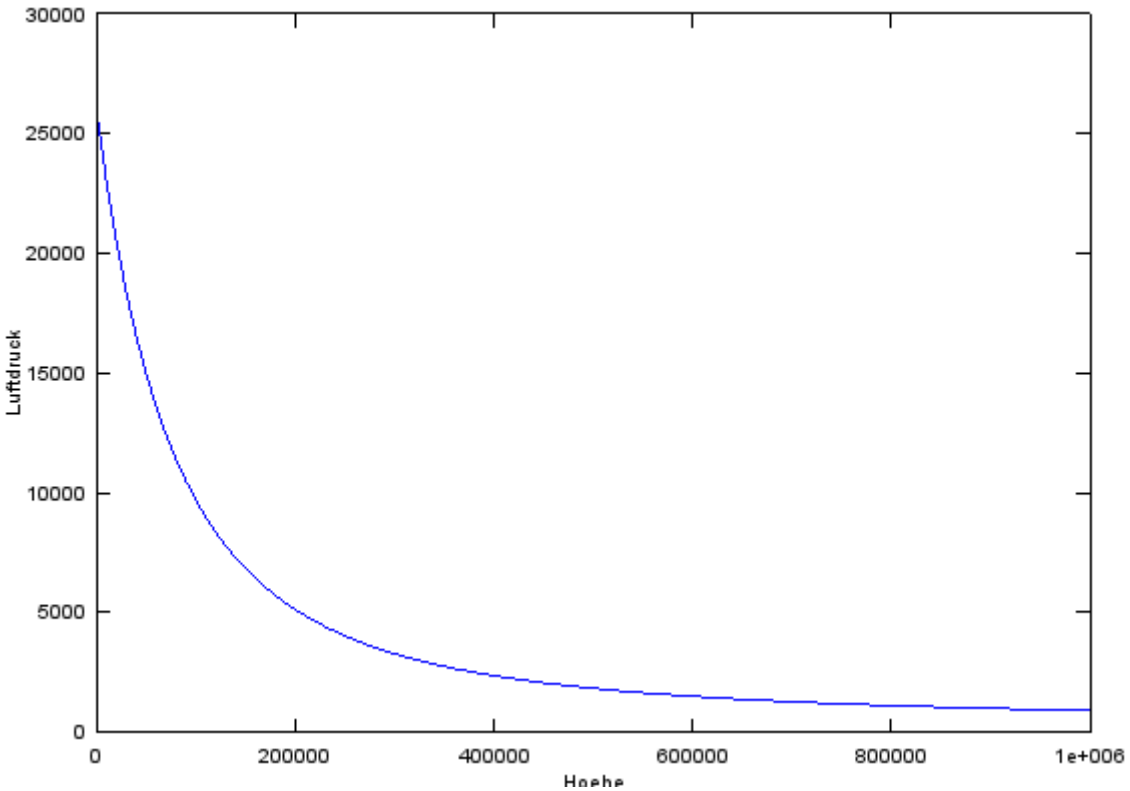
$$\frac{p(z+h) - p(z)}{h} = -\frac{g * p}{RsT}$$

Die Ableitung des Druckes an der Stelle z beträgt somit auch:

$$p'(z) = -\frac{g * p}{RsT}$$

Das Integral von der vorhergehenden Funktion in Octave mit den Angaben des Kometen in Octave eingegeben, ergibt folgenden Graphen.

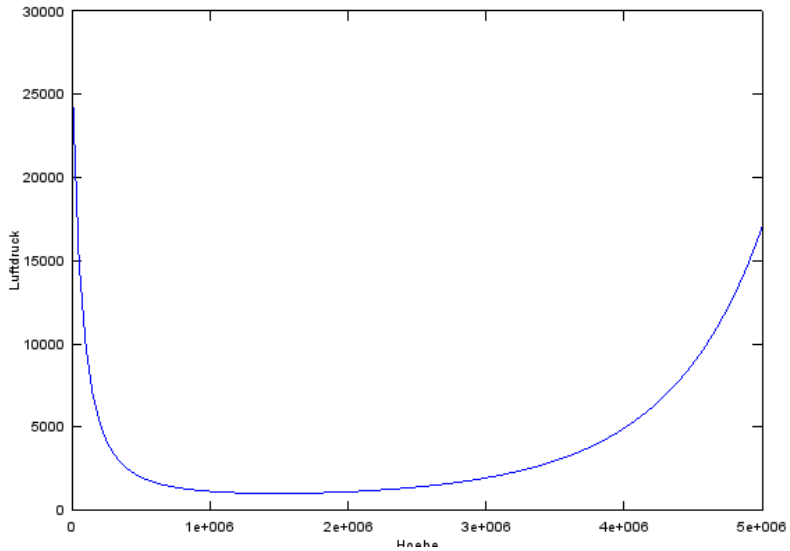
Atmosphärendruck



Hier lässt sich erkennen, dass der Luftdruck mit steigender Höhe exponentiell abnimmt, wobei der Luftdruck in Pascal gegeben ist. Auf Meereshöhe herrscht also ein Druck von 0,25bar. Dieser Wert ist im Vergleich zur Erde sehr realistisch.

Bindet man nun einen weiteren wesentlichen Faktor, und zwar die Zentrifugalkraft, die gegen die Gravitation wirkt, in die Überlegung ein, so erhält man ein stark abgeändertes Ergebnis. Die veränderte Formel lautet:

$$p = p_0 * e^{\left(-\frac{G*M}{Rs*T}\right)*\left(\frac{1}{r}-\frac{1}{z+r}\right)+\left((w^2*z*(z+2r))/2\right)}$$



Atmosphärendruck

Wie man dem Graphen entnehmen kann beträgt der Luftdruck nie null, wodurch man schließen kann, dass die Bildung einer Atmosphäre unmöglich ist. Der Komet ist nicht in der Lage eine Atmosphäre zu halten.

Orbitallaufbahn des Kometen

Als nächstes galt es zu beweisen, ob Jules Verne mit seinen Aussagen, der Komet sei innerhalb der Venusbahn der Sonne am nächsten, und außerhalb der Jupiterbahn von der Sonne am weitesten entfernt, richtig lag. Jules Verne beschrieb eine Umlaufbahn von 2 jähriger Dauer. Um die Richtigkeit dieser Annahmen zu überprüfen, haben wir die Umlaufbahnen der Planeten sowie die Umlaufbahnen des Kometen grafisch mit GeoGebra dargestellt.

Mit Hilfe der Kepler'schen Gesetze (insbesondere des 3. Kepler'schen Gesetzes), haben wir uns zuerst überlegt, ob der Komet mit einer Umlaufzeit von 2 Jahren eine solche große Distanz (Venus-Jupiter) zurücklegen kann.

$$3.\ Kepler'sche\ Gesetz:$$

$$\frac{t_1^2}{t_2^2} = \frac{a_1^3}{a_2^3}$$

Als t_1 haben wir die Umlaufzeit der Erde definiert, als t_2 die (im Buch angegebene) Umlaufzeit des Kometen. Das Verhältnis der beiden Umlaufzeiten beträgt also 1:4. Dies wiederum haben wir gleichgesetzt mit den Kuben der großen Halbachsenlängen. Die Halbachsenlänge der Kometenbahn beträgt $2{,}37 \cdot 10^{11} m$. Mit dieser Halbachsenlänge haben wir die Umlaufbahn grafisch dargestellt (siehe grüne Ellipse). Man sieht, dass der Komet mit einer Umlaufzeit von 2 Jahren zwar an die Venusbahn heranreicht, jedoch die Jupiterbahn nicht erreicht oder gar schneidet.

Angegeben war jedoch ein Aphel von 220 Mio. Leugen ($\equiv 880 \cdot 10^6$ km), das Perihel entspricht dem Abstand der Sonne zur Venus, also $108{,}1608 \cdot 10^6$ km. Die Summe aus Perihel und Aphel ist die Hauptachsenlänge 2a.

$$2a = Perihel + Aphel$$

Daraus ergibt sich für a ein Abstand von $494{,}0804 \cdot 10^6 km$ und für die Brennweite

$385{,}9196 \cdot 10^6 km$, sowie im Folgenden für die halbe Nebenachse $308{,}514998\ km$. Konstruiert man nun die Ellipse sieht man, dass der Komet die Jupiterbahn schneidet.

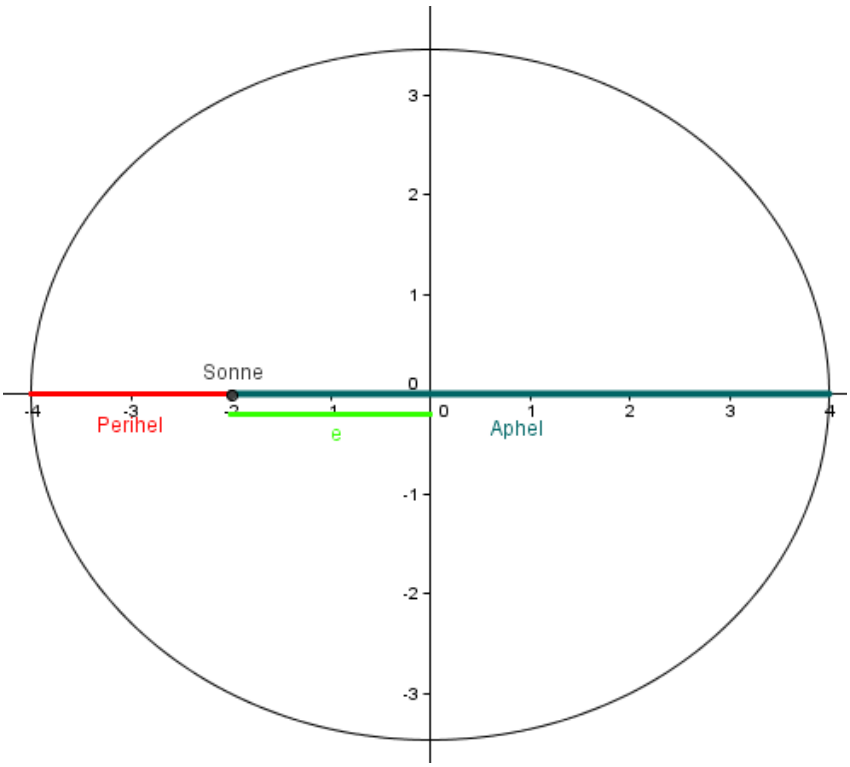
Rechnerisch setzt man die große Halbachsenlänge in das 3. Kepler'sche Gesetz ein und berechnet so die nötige Zeit für einen kompletten Umlauf. Dabei haben wir eine Zeit von 6 Jahren berechnet. Der von Jules Verne beschriebene Komet hatte also die ursprünglich beschriebene Umlaufbahn, jedoch brauchte er dafür 6 anstatt den angegebenen 2 Jahren.

Grundlagen dieser Überlegungen waren die Tatsache, dass sowohl bei den Planeten, als auch bei dem Kometen die Sonne im Brennpunkt der elliptischen Umlaufbahnen liegt. Um die elliptischen Bahnen der Kometen zu beschreiben, verwendeten wir die Ellipsengleichungen:

$$1 = \frac{x^2}{a^2} + \frac{y^2}{b^2}$$

$$a^2b^2 = x^2b^2 + y^2a^2$$

Der Mittelpunkt der Umlaufbahn der Erde liegt im Koordinatenursprung. Die große Halbachsenlänge der elliptischen Umlaufbahnen haben wir uns aus den Keplergesetzen berechnet. Perihel und Aphel der Erde waren bekannt. Kennt man die große Halbachsenlänge a sowie den kürzesten Abstand der Erde zur Sonne, also das Perihel, so kann man sich mit folgendem Zusammenhang die Nebenachsenlänge b berechnen:



$$e = a - Perihel$$

e ist der Abstand vom Ellipsenmittelpunkt zur Sonne, also die Brennweite einer Ellipse.

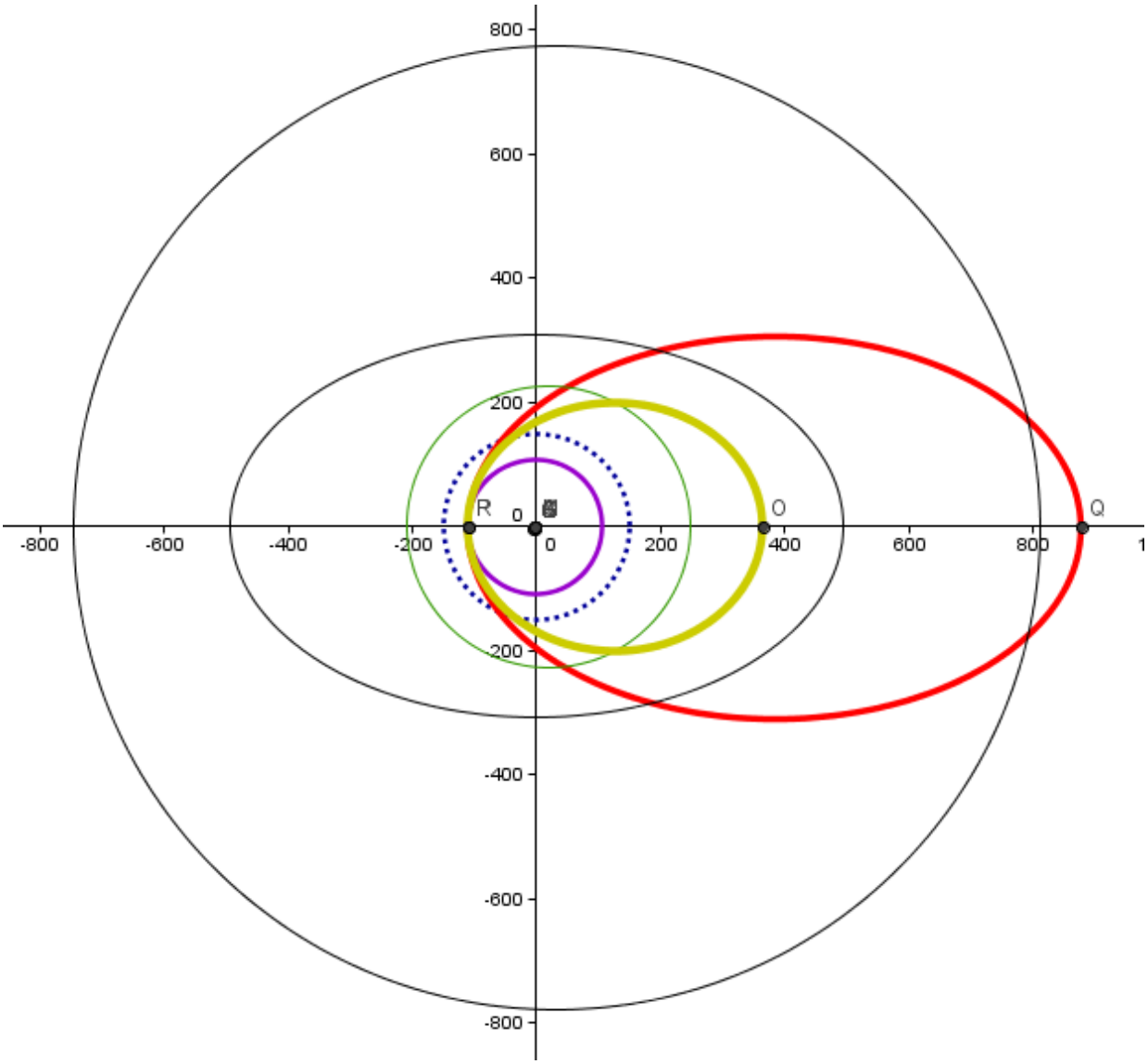
Mit folgendem Zusammenhang kann man sich nun die Nebenachsenlänge b ausrechnen:

$$b^2 = a^2 - e^2$$

Man setzt beide Werte in die Gleichung ein und erhält eine Formel die die elliptische Umlaufbahn der Erde beziehungsweise im Folgenden auch die Bahnen der anderen Planeten sowie des Kometen beschreibt.

Aus diesen Zusammenhängen lässt sich berechnen, dass die Erde sowie sämtliche andere Planeten eine annähernd kreisförmige Bahn beschreiben. Die Sonne liegt somit annähernd im Mittelpunkt der Umlaufbahn der Erde:

Orbitallaufbahn



In seinem Buch „Hector Servadac“ beschreibt Jules Verne unteranderem die Laufbahn des Kometen „Gallia“, in Bezug auf die anderen Planeten unseres Sonnensystems, sehr genau. Die Umlaufbahn von Gallia beschreibt, lauf Verne, eine extrem elliptische Rotationsbahn um die Sonne. Diese Bahnen wurden bereits genauer erläutert, jedoch fehlten bisher mathematische Beweise, welche nun fortfolgend erläutert werden. Um unser Sonnensystem, und die Interaktion zwischen den jeweiligen Planeten und Gallia zu veranschaulichen, haben wir mit Hilfe des Rechen- und Plottingprogramms „Gnu Octave“ die für unser Modell relevanten Planeten und den Kometen simuliert.

Wie bei jeder physikalischen Aufgabenstellung musste zuerst eine Sammlung der notwendigen Formeln erstellt werden. Die Masse des Kometen konnte dem Roman direkt entnommen werden, und die Massen der Planeten konnten durch Internetrecherchen ermittelt werden. Für die Abstände zwischen den Planeten und der Sonne nahmen wir Angaben aus vertrauenswürdigen Internetquellen. Bei der Simulation der Bewegung des Kometen definierten wir eine Anzahl x an

fortlaufenden Zeitschritten, deren Abstände wir gegen null laufen ließen. Dadurch wurde die Position des Kometen zu jedem Zeitabschnitt neu errechnet und die Simulation einer Ellipse ermöglicht.

Diese Berechnung beruht auf folgender Differentialgleichung:

$$x_1 = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \qquad x_2 = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$$

Die erste Ableitung ergibt die Geschwindigkeit:

$$\frac{d}{dt}x_1 = v_1 \qquad \frac{d}{dt}x_2 = v_2$$

$$v_1 = \begin{pmatrix} v_{x1} \\ v_{y1} \end{pmatrix} \qquad v_2 = \begin{pmatrix} v_{x2} \\ v_{y2} \end{pmatrix}$$

Die zweite Ableitung ergibt die Beschleunigung:

$$\frac{d}{dt}v_1 = a_1 \qquad \frac{d}{dt}v_2 = a_2$$

$$a_1 = \frac{G*m_2*(x_2-x_1)}{|x_2-x_1|^3} \qquad a_2 = \frac{G*m_1*(x_1-x_2)}{|x_1-x_2|^3}$$

Umgelegt auf unser Beispiel kann man die Bewegung des Kometen in Bezug auf die Sonne nun folgendermaßen ausdrücken:

$$a_{Komet} = \frac{G * m_{Sonne} * r_1}{|r_1|^3}$$

„r_1“ beschreibt den Abstand zwischen Komet und Sonne.

Will man nun die Laufbahn des Kometen im Bezug zu den relevanten Himmelskörpern (Sonne, Venus, Erde, Jupiter) bestimmen, muss man die Formel folgendermaßen ummodellieren. (Die Erde an sich beeinflusst den Meteoriten nur vernachlässigbar, sie wurde nur als Referenz in die Simulation miteinbezogen.)

$$a_{Komet} = \frac{G * m_{Sonne} * r_1}{|r_1|^3} + \frac{G * m_{Venus} * r_2}{|r_2|^3} + \frac{G * m_{Erde} * r_3}{|r_3|^3} + \frac{G * m_{Jupiter} * r_4}{|r_4|^3}$$

„r_1“ beschreibt den Abstand zwischen Komet und Sonne.

„r_2“ beschreibt den Abstand zwischen Komet und Venus.

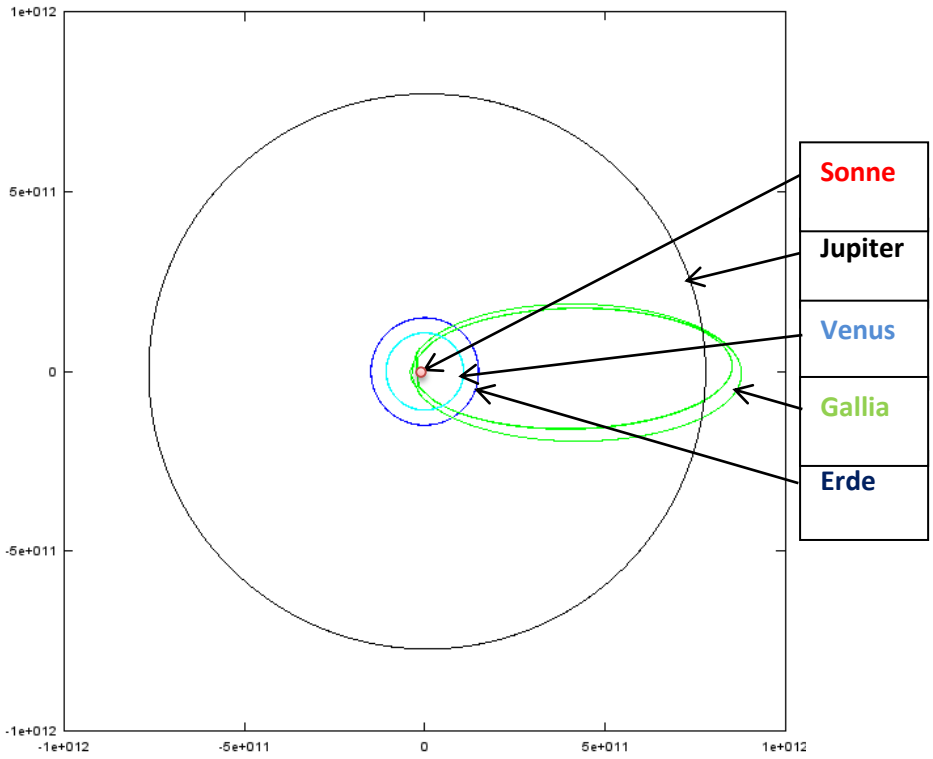
„r_3“ beschreibt den Abstand zwischen Komet und Erde.

„r_4“ beschreibt den Abstand zwischen Komet und Jupiter.

Orbitallaufbahn

Um die Simulation graphisch darzustellen, haben wir einen Plotbefehl geschrieben, in dem die Koordinaten jedes einzelnen Zeitpunktes in einem Diagramm eingezeichnet werden. Mit einem weiteren Befehl haben wir auch ein dynamisches Diagramm erzeugt, welches hier jedoch nicht dargestellt werden kann.

Die folgende Grafik zeigt die Orbitale der genannten Himmelkörper, wobei die verzerrte Bahn des Kometen durch die Beeinflussung von Venus und Jupiter hervorgerufen wird.



Man erkennt eindeutig, dass die Bahn, die Jules Verne im Buch beschreibt der errechneten entspricht, auch wenn der Komet statt zwei, sechs Jahre für einen Umlauf braucht. Die Extrempunkte der Umlaufbahn liegen aber wie von Verne angegeben innerhalb der Venusbahn und außerhalb der Jupiterbahn.

Temperatur auf Gallia

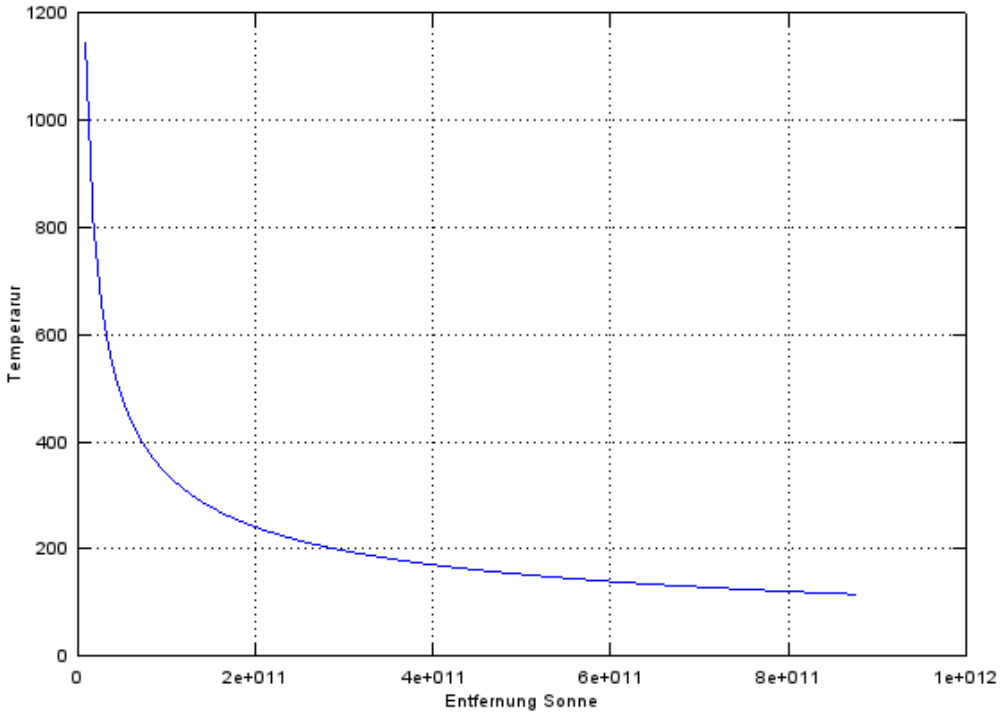
Um zu überprüfen ob der Roman von Jules Verne, in dem Menschen auf dem Komet zwei Jahre überleben, in diesem Detail wirklich Sinn ergeben kann, haben wir den Kometen auf sein Klima, sprich auf seine Temperaturverhältnisse untersucht. Wie wir bereits erläutert haben beschreibt der Komet eine sehr elliptische Bahn um die Sonne, ist also sehr starken Temperaturschwankungen ausgesetzt.

Anfangs haben wir uns die Temperatur abhängig von der Entfernung zur Sonne, jedoch unbeachtet des Treibhauseffekts und der Temperatur-Emission betrachtet.

Mit folgender Formel haben wir unsere Berechnungen durchgeführt.

$$T = \sqrt[4]{\frac{Es}{(16 * \pi * \sigma * r^2)}} * A$$

Es	Strahlungsleistung der Sonne
σ	Stefan-Boltzmann Konstante
r	Abstand zur Sonne
A	Querschnittsfläche des Kometen

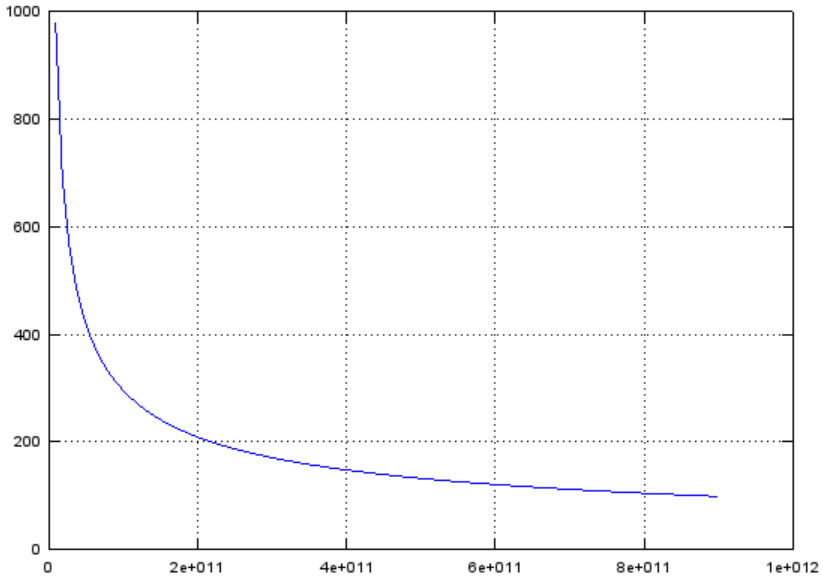


Temperatur auf Gallier

Zur Erinnerung: Der Komet bewegt sich in einer Distanz in Bezug auf die Sonne von 1,1*10^{11}m bis 8,7*10^{11}m, also würden ohne Treibhauseffekt Temperaturen zwischen 50°C und -158°C betragen.

Um den Treibhauseffekt miteinzubeziehen, gehen wir davon aus, dass die Atmosphäre des Kometen, der der Erde entspricht. Um eine Konstante zu errechnen, die wir einbeziehen können, errechnen wir uns die entsprechende Konstante auf der Erde, wenn man von einer mittleren Temperatur von 15°C, also 288,15 Kelvin, ausgeht. Die dadurch errechnete Konstante $x = 1{,}2626$ wird nun in der oben genannten Formel eingesetzt. Da Atmosphäre und Bodenbeschaffenheit in etwa der der Erde entsprechen haben wir auch den Emissionskoeffizienten der Erde $\varepsilon = 0{,}694$ verwendet.

$$T = \sqrt[4]{\frac{Es}{(16 * \pi * \sigma * r^2 * \varepsilon * x^4)}} * A$$



Nun liegen die Temperaturen zwischen 10°C und -173°C

Da das Ziel unserer Ermittlung die Antwort auf die Frage war, ob die Menschen auf dem Kometen lebensfreundliche Bedingungen hatten, muss man sagen, dass eher extreme Temperaturen vorherrschen. Es ist jedoch möglich, da Verne beschreibt, dass bei den Hitzeextremen die Bewohner auf die sonnenabgewandte Seite geflüchtet sind. Diese dauerten nur kurz an, da der Komet an dieser Stelle seine höchste Geschwindigkeit erreicht. Bei dem Temperaturminimum sind die Bewohner laut Verne in warme Vulkanhöhlen geflüchtet um zu überleben.

Zurück zum Seitenanfang

Modellierungswoche

Mathematik-Informatik

5. – 11. Februar 2012

JUFA Pöllau

Clemens Andritsch

Philipp Gaulhofer

Fabian Peter Hammerle

Sebastian Kollegger

Michael Pucher

Matthias Stocker

betreut von

Mag. Stefan Fürtinger

Projekt:

Bildverarbeitung mit neuronalen Netzen

Inhaltsverzeichnis

Bildverarbeitung mit neuronalen Netzen

Einführung

Für einen Menschen ist es einfach, eine Katze von einem Hund zu unterscheiden, doch mathematische Aufgaben zu lösen erweist sich als deutlich schwieriger. Das exakte Gegenteil dazu ist der Computer:
Er kann problemlos schwierige mathematische Probleme lösen, doch wenn es um das Thema Bilderkennung geht, tut sich ein Computer genauso schwer wie ein Mensch, der mathematische Probleme lösen muss. Jedoch ist es für uns Menschen möglich, mathematische Problemstellungen und ihren Lösungsweg zu erlernen und zu trainieren. Warum sollte das nicht auch ein Computer können?
Das Ziel unseres Projektes war es also, auch einen Computer dazu zu bringen zu erlernen wie man Bilder unterscheiden kann. Dazu beschäftigten wir uns mit neuronalen Netzen, die es auch in unserem Gehirn gibt. Dabei stießen wir auf die sogenannte McCulloch-Pitts Zelle.

Die McCulloch-Pitts Zelle:

Die McCulloch-Pitts Zelle, auch Schwellwertgatter genannt, ist das erste Neuronenmodell für künstliche Intelligenz und wurde im Jahr 1943 von Warren McCulloch und Walter Pitts entwickelt. Das Modell soll reale Vorgänge in einem einfachen System modellieren und vereinfachen.

Das Netzwerk in dem die McCulloch-Pitts Zellen angewandt werden hat drei Ebenen: Die Inputebene (mit Inputwerten von x_1 bis x_D), den Hiddenlayer (mit Aktivitätslevel z_1 bis z_M) und die Outputebene (mit Aktivitätslevel y_1 bis y_K). Die Informationen erhält das System über die „Inputebene“. Dabei hat jedes Neuron dieser Ebene einen bestimmten Inputwert. Die Neuronen der Inputebene senden an die Neuronen des Hiddenlayers einen Wert, der abhängig von ihrer Wichtigkeit ist. Dafür hat man für Verbindung zwischen einem Neuron der Inputebene und einem Neuron im Hiddenlayer einen eigenen Faktor, der „Gewicht“ genannt wird, und die Wichtigkeit des Neurons der Inputebene auf das des Hiddenlayers steuert ($w_{1\ 1}^{(1)}$ bis $w_{MD}^{(1)}$, dabei steht die erste Zahl für das Ziel im Hiddenlayer, die zweite Zahl für das Ausgangsneuron in der Inputebene und die Zahl in der Klammer für die Schicht, in der sich das abspielt).

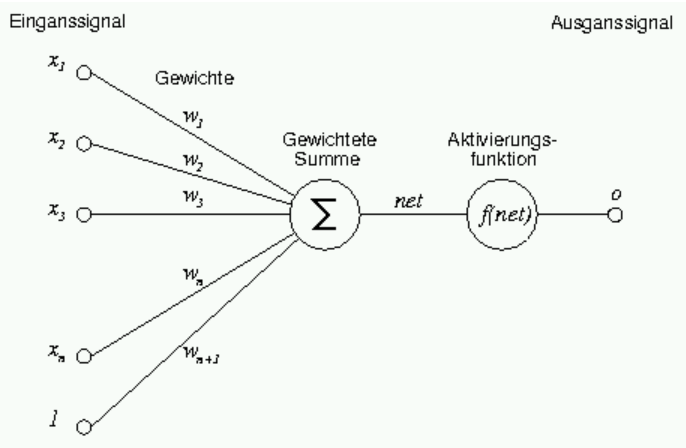


Abbildung 1

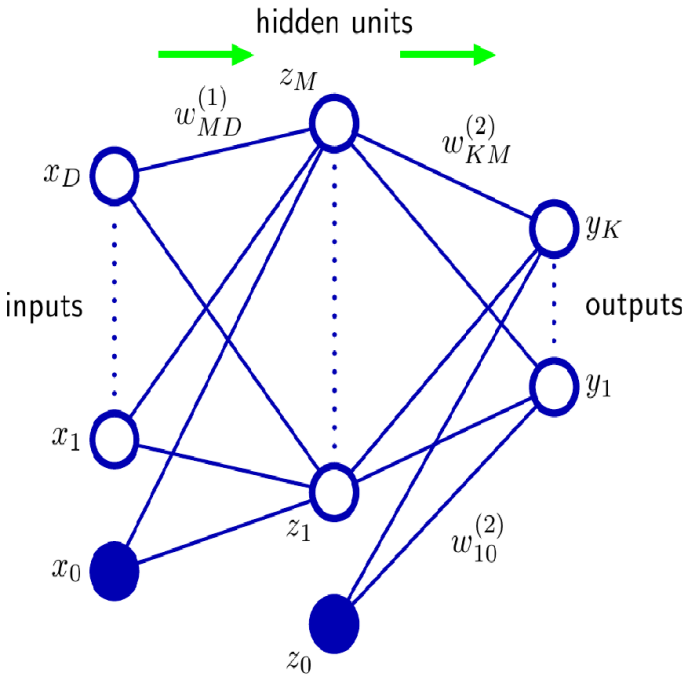


Abbildung 2

Ab einem gewissen Wert, der bei jedem Neuron des Hiddenlayers unterschiedlich ist, schickt das Neuron des Hiddenlayers Impulse an die Outputebene. Der Wert, der dabei überschritten werden muss, nennt man den Schwellwert (meist als Theta bezeichnet). Um diesen Schwellwert zu beeinflussen gibt es Bias-Units(x_0, z_0), die je Schicht einen bestimmten Wert für jedes Neuron annehmen können.
Von den Neuronen des Hiddenlayers gehen jeweils wieder Verbindungen zu den Neuronen der Outputebene aus. Diese haben wieder Gewichte, die dieses Mal von $w_{11}^{(2)}$ bis $w_{KM}^{(2)}$ gehen. Das Neuron, das in der Outputebene am stärksten angesprochen wurde, bestimmt den Output des Netzwerks.

PyBrain

Um neuronale Netzwerke bauen zu können, verwendeten wir eine auf künstliche Intelligenz spezialisierte Python-Bibliothek namens PyBrain.
Dabei fokussierten wir uns hauptsächlich auf Supervised Learning, darunter versteht man, dass dem Programm vorgegeben wird welches Ergebnis es errechnen soll. Hierfür benötigt man ein Dataset mit den dazugehörenden Targets.

- Datasets

Um ein Dataset zu generieren wandelten wir unsere Bilder in Arrays um, in welche wir die Graustufenwerte jedes Pixels schrieben. Für die Targets wird ein weiteres Array definiert, in dem festgelegt wird, welches Neuron der Outputebene am stärksten angesprochen werden soll. Das Dataset wird in weiterer Folge in ein Trainingsset und ein Testset aufgeteilt.

- Netzwerk

In PyBrain ist es möglich mit wenigen Befehlen die Struktur des Netzwerks zu bestimmen, wobei die Inputebene (Anzahl der Pixel) und die Outputebene (Anzahl der möglichen Targets), bereits vorgegeben sind. Einzig die Anzahl der Hiddenlayer und die Anzahl der Hiddenunits bleiben uns überlassen.
Es wird auch noch zwischen Feedforward Netzwerken und Recurrent Netzwerken

unterschieden. Der Unterschied zwischen den beiden liegt darin, dass bei einem Feedforward Netzwerk alle Informationen nur von einer Schicht zur nächsten gesendet werden. Bei einem Recurrent Netzwerk schicken die einzelnen Neuronen auch ein Feedback an sich oder an ein Neuron aus den vorigen Schichten zurück, und beeinflussen sich damit zusätzlich.

- Trainer

Als nächstes benötigt man einen Trainer, der das Netzwerk trainiert. Dazu verwendeten wir den sogenannten Backpropagation-Trainer. Wichtige Einstellungen, die am Trainer getätigt werden können sind zum Beispiel die Anzahl der Epochen (Anzahl der Trainings), die Learningrate und ob Bias-Units verwendet werden sollen. Je mehr Epochen man benützt, desto bessere Netzwerke werden erstellt (dies gilt aber nicht immer, da sich das Netzwerk manchmal innerhalb eines Trainingslaufs auch verschlechtern kann). Der Trainer verwendet die Ableitung der Übertragungsfunktionen. Damit sucht er nach lokalen Minima. Oft passiert es aber, dass das Netzwerk diese überspringt. Dafür gibt es die Learningrate, die genau das verhindern soll. Da man für jedes Neuron verschiedene Schwellwerte braucht, gibt es die Bias-Units.

- Trainingsvorgang

Das Netzwerk wird nun auf das Trainingsset trainiert. Um es zu trainieren werden die Gewichte im Netzwerk verändert. Zuerst wählt der Trainer für jedes Gewicht einen zufälligen Wert, den er nach jedem Trainingsdurchlauf zu verbessern versucht. Ein wichtiger Begriff hierbei ist das Overfitting. Dies tritt auf, wenn sich das Netzwerk zu sehr auf das Trainingset einstellt und somit das Testset nicht mehr gut erkennt.

- Auswertung

Nachdem die angegebenen Epochen abgehandelt worden sind, wird das Netzwerk gespeichert und mit dem Testset getestet. Mit diesem es das Ziel ist, eine ähnlich hohe Hitrate wie beim Trainingsset zu erreichen. Zur besseren Übersicht, wird der Verlauf der Trainingsdaten in einem Diagramm angezeigt. So kann man sehr leicht und schnell erkennen, ob es sinnvoll ist, mit dem Netzwerk weiter zu arbeiten, oder nicht.

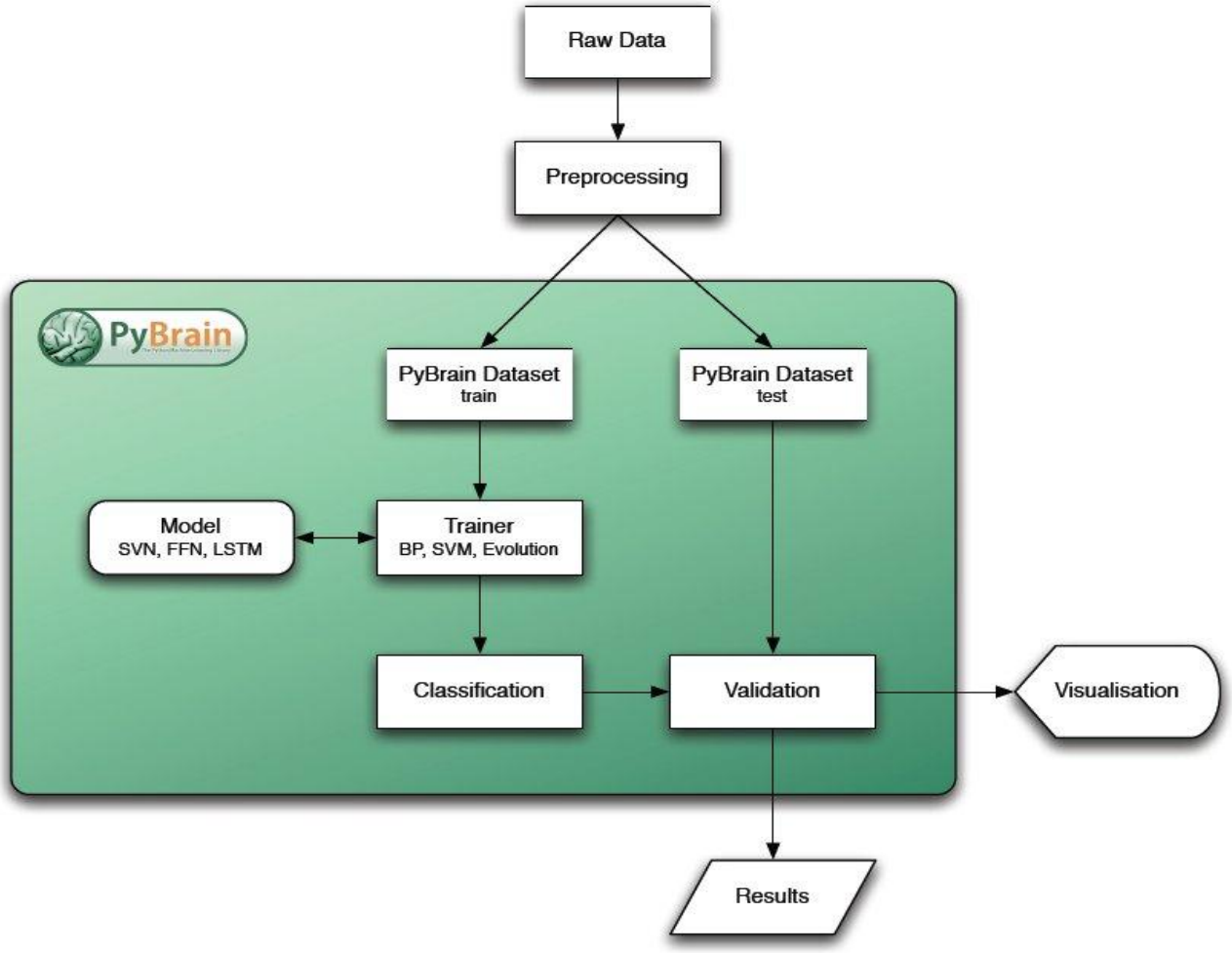


Abbildung 3

Übertragungsfunktionen

Die McCulloch-Pitts Zelle arbeitet mit einer "Heavy-Side"-Funktion. Das bedeutet, wenn das Argument x einen Schwellenwert überschreitet, nimmt H(x) den Wert eins an. Für komplexere Aufgaben braucht man jedoch komplexere Funktionen, um bessere Ergebnisse zu erzielen.

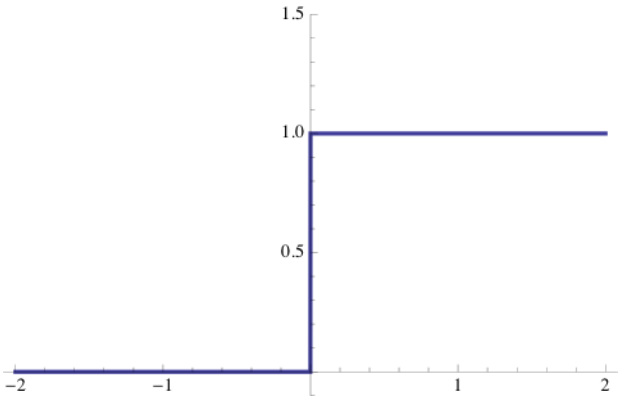


Abbildung 4

Der Trainer von Pybrain verwendet automatisch eine lineare Funktion. Daher gilt: Je größer die Werte, die das Neuron erhält, desto mehr sendet das Neuron an das nächste Neuron. Dies ist für einfache Problemstellungen sinnvoll, doch bei komplizierteren braucht man eine der folgenden Funktionen:

Tangenshyperbolicus-Funktion:

$$\tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$

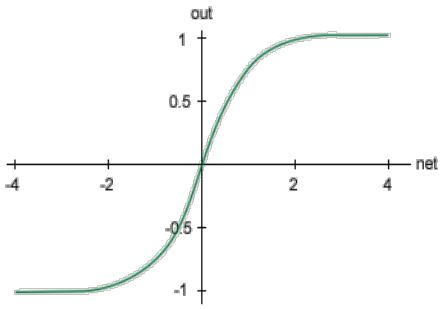


Abbildung 5

Sigmoid-Funktion:

$$\mathrm{sig}(x) = \frac{1}{(1 + e^{-x})}$$

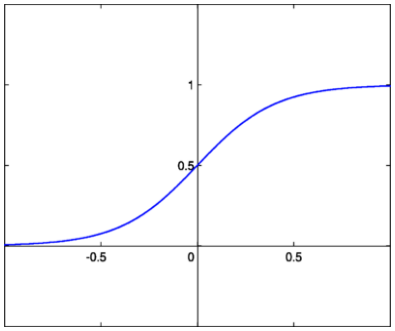


Abbildung 6

Softmax-Funktion:

$$\omega(x) = \frac{e^{a_1}}{\sum_{i=2}^{2} e^{a_i}}$$

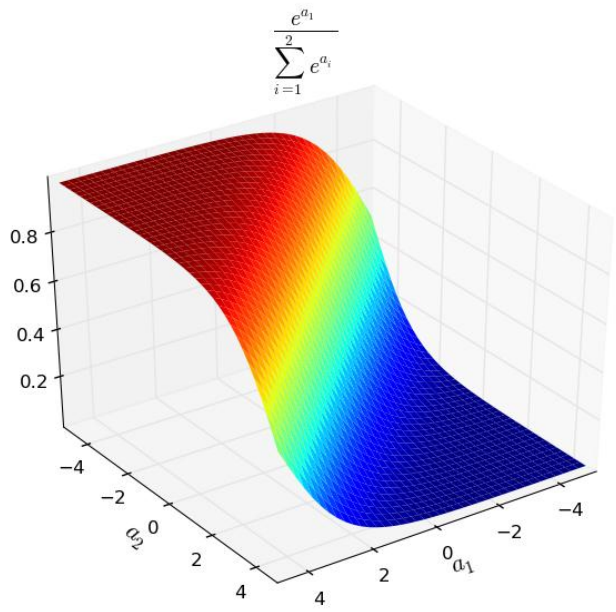


Abbildung 7

Funktionale Darstellung eines Netzwerks:

N_i … i-tes Neuron

x_i … Inputwert von N_i

z_i … Aktivitätslevel von N_i im Hiddenlayer

y_i … Aktivitätslevel von N_i im Outputlayer

a_i … Netinput von N_i im Hiddenlayer

b_i … Netinput von N_i im Outputlayer

h(x) … Übertragungsfunktion des Hiddenlayers

g(x) … Übertragungsfunktion des Outputlayers

$w_{ab}^{(c)}$ … Gewicht, wobei: a … Zielneuron

b … Ausgangsneuron

c … Schicht in der es wirkt

Die Werte der Outputneuronen in Abhängigkeit von den Gewichten und der Inputwerte errechnen sich wie folgt:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i$$

$$z_j = h(a_j) = h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i \right)$$

$$b_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j$$

$$y_k = g(b_k) = g\left(\sum_{j=1}^{M} w_{kj}^{(2)} z_j \right) = g\left(\sum_{j=1}^{M} w_{kj}^{(2)} * h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i \right) \right)$$

Recurrent Netzwerke

Im Gegensatz zu Feedforward Netzwerken, wo die Übertragung immer nur in eine Richtung geschieht, benutzen Recurrent Netze auch Rückkopplungen. Deshalb können auch Verbindungen zur gleichen Neuronenschicht (direkte Rückkopplung) oder zu einer früheren Schicht (indirekte Rückkopplung) des Netzwerks existieren. Die Anwendungsgebiete von Recurrent Netzwerken liegen bei dem Treffen von Prognosen über zukünftige Ereignisse und der Simulation von menschlichen Verhaltensweisen.

Für die Anwendung im Bereich der Bilderkennung sind Recurrent Netze im Gegensatz zu Feedforward Netzen weniger gut geeignet. In Bezug auf Leistung und Ergebnisse haben Feedforward Netze einen entscheidenden Vorteil. Ein Recurrent Netz mit zwei Layern und drei Hidden Units beispielsweise, versagte bei der Erkennung von Ziffern: Die Hitrate lag bei unter 40%. Nach einigen Versuchen konnte allerdings mit einem Recurrent Netz eine akzeptable Leistung von 87% bei der Erkennung von handschriftlichen Ziffern erzielt werden, wenn auch mit gewaltigem Rechenaufwand.

Da mit einfachen Recurrent Netzen keine brauchbaren Ergebnisse produziert werden konnten, wurde ein wesentlich komplexeres Netzwerk mit fünf Layern entwickelt:

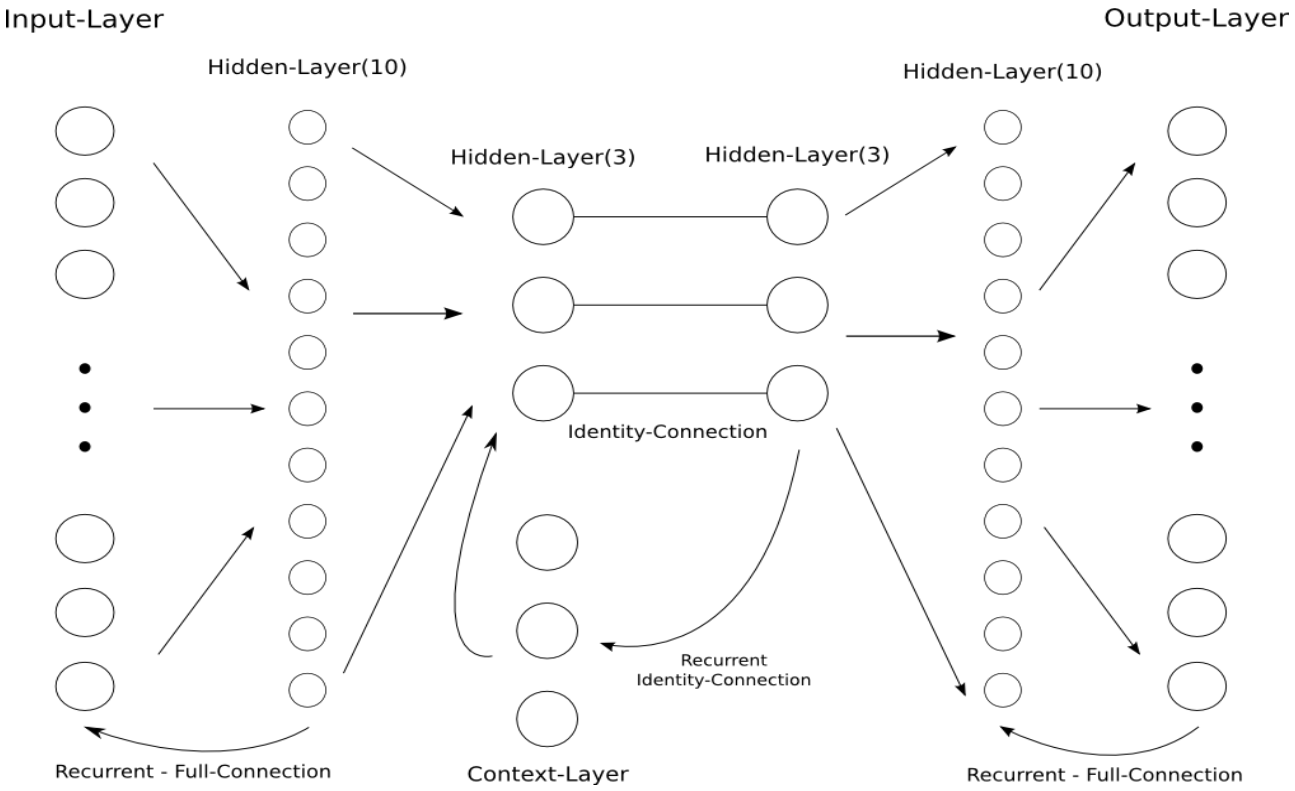


Abbildung 8

Projekt 1: Ziffernerkennung

Unsere erste Problemstellung bestand darin Bilder im Format von 16 x 16 Pixel von handgeschriebenen Ziffern zu erkennen und die Ziffern identifizieren können. Die Bilder wurden in Graustufen umgewandelt um die Erkennung zu erleichtern.

Das Netzwerk, das wir erstellten, bestand aus 256 Inputneuronen, eines pro Pixel und zehn Outputneuronen, eines für jede Ziffer. Die Anzahl der Neuronen im Hiddenlayer sollte sich erst nach und nach ergeben.

Die weiteren Parameter für das Netzwerk wurden erst mit der Zeit nach vielen Testläufen bestimmt. Wir erkannten, dass die Wahl der Übertragungsfunktionen der verschiedenen Schichten essentiell für das Ergebnis war.

Nach vielen Tests kristallisierten sich einige gute Netzwerke heraus und es wurde beschlossen, dass die besten sechs Netze über Nacht austrainiert werden sollten und im Anschluss daran das Beste genommen wird.

Es stellte sich heraus, dass ein FeedForward Netzwerk mit sigmoiden Funktionen, einer Learningrate von 0,1 und neun Neuronen im Hiddenlayer am besten die Bilder zuweisen konnte.

Dieses Netzwerk erreichte eine Hitrate von 92% für das Trainingsset und von 90% für das Testset. Dabei soll jedoch erwähnt werden, dass die Bilder auch für uns nicht immer eindeutig zu erkennen sind. (siehe Abb.: 10)

Zur besseren Veranschaulichung, programmierten wir eine Oberfläche, die das Bild der Ziffer, den Tipp des Computers und das tatsächliche Ergebnis ausgibt. Das Programm zeigt dabei die Ziffer in grün oder rot an, je nachdem ob das Bild richtig oder falsch erkannt wurde.

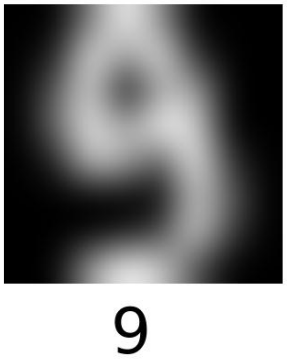


Abbildung 9

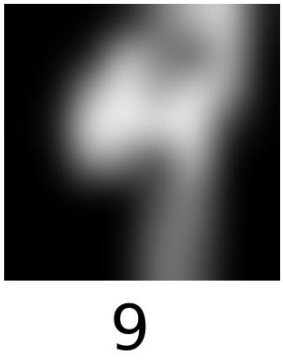


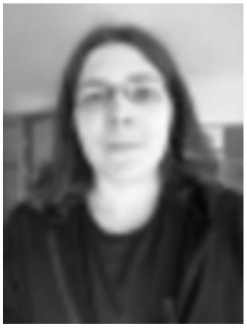
Abbildung 10

Projekt 2: Gesichtserkennung

Nach dem Erfolg beim Erkennen der Ziffern widmeten wir uns einer größeren Herausforderung, der Gesichtserkennung. Zunächst benötigten wir ausreichend Fotos (20.000 Stück), um ein Trainingsset und ein Testset erstellen zu können. Dazu haben wir uns vor einem einheitlichen Hintergrund selbst fotografiert und die Bilder in Graustufen umgewandelt. Für das neuronale Netz haben wir ein Feedforward-Netz mit fünf Hiddenunits, sigmoiden Funktionen, Biasunits und einer Learningrate von 0,1 verwendet. Unsere Fotos hatten eine Größe von 48 x 64 Pixel, daher betrug die Anzahl der Neuronen in der Inputebene 3072. Wir benötigten sechs Neuronen in der Outputebene, da wir sechs Teammitglieder waren (sechs mögliche Antworten). Das Ergebnis sprach für sich selbst: Die Fotos mit einem einheitlichen Hintergrund wurden von unserem Netz zu 96% richtig erkannt.

Danach gingen wir noch einen Schritt weiter: Wir fotografierten uns vor unterschiedlichen Hintergründen mit teils schwierigen Lichtverhältnissen (nochmals mehr als 20.000 Bilder). Es wurden abermals ein Trainingsset und ein Testset erstellt und wir verwendeten dieselben Netzwerkeinstellungen, wie bei den normierten Gesichtern mit dem Ergebnis, dass immer noch circa 90% der Bilder von unserem Netz richtig erkannt wurden.

Nachdem unsere Netzwerke gut funktionierten, interessierte uns noch, welche Hitrate bei vertauschten Datensätzen erreicht werden kann. Wir ließen also das Netzwerk, das auf den einfacheren Datensatz trainiert war, die Bilder mit den ständig wechselnden Hintergründen auswerten und umgekehrt. Dabei bemerkten wir, dass sich das Netzwerk, das den schwierigeren Datensatz zum Trainieren hatte viel besser abschnitt, als das, das mit dem monotonen Hintergrund arbeitete. Ersteres erreichte eine Trefferquote von über 50%, Letzteres eine von circa 35%.



Philipp

Abbildung 11

Matthias

Abbildung 12

Quellen

Abbildung 1:
http://www.iicm.tugraz.at/greif/node10.html

Abbildung 2, 5, 6:
Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Cambridge: Springer

Abbildung 3:
Schaul, T.; Bayer, J.; Wierstra, D.; Sun, Y.; Felder, M.; Sehnke, F.; Rückstieß, T.; Schmidhuber, J. (2010). PyBrain. Journal of Machine Learning Research(S. 743-746)

Abbildung 4:
http://www.cmg.org/measureit/issues/mit62/m_62_15.html

Signalverarbeitung

Pricipal Component Analysis und Independent Component Analysis

Modellierungswoche der Mathematik

Unter Betreuung von a.o.Univ.Prof. Mag.Dr. Stephen Keeling

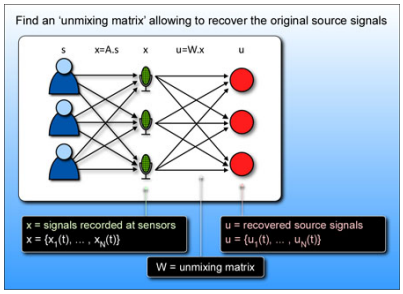
Jasmin Leitner, Joachim Auer, Georg Maierhofer, Jeremias Regner,
Andreas Weiss, Daniel Windisch

Inhaltsverzeichnis

1 Einführung

Die Woche der Modellierung bot für uns die einzigartige Möglichkeit sich den beiden Verfahren PCA (Principal Component Analysis) und ICA (Independent Component Analysis) unter Betreuung von a.o.Univ.Prof. Mag.Dr. Stephen Keeling in mathematisch interessanter Weise und auf Grundlage von Schulstoff zu nähern. Die beiden Verfahren bieten die Möglichkeit - wie ihr Name bereits verspricht - auf Basis von Matrixalgebra und Statistik die Hauptkomponenten und die unabhängigen Komponenten einer bzw. mehrere Datenmengen zu analysieren und damit die Quellen zu trennen. Als einführendes Beispiel kann hier das "Cocktail-Party-Problem"genannt werden, welches das Prinzip hinter diesen Verfahren sehr gut veranschaulicht:

Wie die Grafik zeigt führen mehrere Personen unkorrelierte Gespräche untereinander. Im Raum befinden sich nun Mikrophone, welche die Sprachquellen als Summe mit verschiedenen Gewichten aufzeichnet. Nun steht man vor dem hypothetischen Problem nur die Aufnahmen zu besitzen und stellt sich die Frage, ob es möglich ist, die ursprünglichen Quellen zumindest ungefähr wieder zu rekonstruieren. Zu beachten ist dabei, wie später klar wird, dass mindestens gleich viele Mischungen (Aufnahmen) wie Quellen verfügbar sein müssen, um diese rekunstruieren zu können.



Genau dies ermöglichen PCA und ICA. Anwendung finden die Verfahren beispielsweise in medizinischer Diagnostik und in der Netzwerk-Tomographie. Diese Arbeit bietet eine Sammlung des von uns in dieser Woche Gelernten und Erarbeiteten und somit eine Einführung in die Grundlagen der Verfahren und in die Arbeitsweise mit jenen.

2 Grundlagen

2.1 Matrixalgebra

Wir wollen uns den Matrizen auf einfachem Weg, zunächst mit linearen Gleichungssystemen nähern: Eine lineare Gleichung ist wohl jedem bekannt (z.B.: $4x + 2y = 3$), auch ein Gleichungssystem mit selben Grad, wie:

$$\begin{aligned} 4x + 2y &= 3 \\ -4x + 2y &= -1 \end{aligned}$$

Die Lösung hier ist leicht ersichtlich $\left(\frac{1}{2}, \frac{1}{2}\right)$, jene kann man einfach durch Addition und Einsetzen der beiden Gleichungen erhalten. Nun aber möchten wir uns dem Problem auf etwas algemeinere Weise nähern - gesucht sind nun die Lösungen (x_1, x_2) des folgenden Gleichungssystemes:

$$a_{11}x_1 + a_{12}x_2 = b_1 \quad (1)$$
$$a_{21}x_1 + a_{22}x_2 = b_2 \quad (2)$$

Wir wählen $m_{21} = \frac{a_{11}}{a_{21}}$ und berechnen (1)-m_{21}(2):

$$\begin{aligned} (a_{12} - m_{21}a_{22})\, x_2 &= (b_1 - m_{21}b_2) \\ \Rightarrow x_2 &= \frac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{21}a_{12}} \\ \Rightarrow x_1 &= \frac{b_1a_{22} - b_2a_{12}}{a_{11}a_{22} - a_{21}a_{12}} \end{aligned}$$

Dabei bemerken wir die notwendige Bedingung, dass für eindeutige Lösungen:

$$det[A] = a_{11}a_{22} - a_{21}a_{12} \neq 0$$

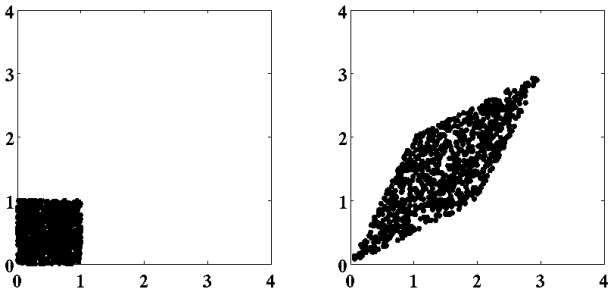
gelten muss. Wir bezeichnen das Polynom als $det[A]$, und somit als Determinante einer Matrix A. Das Gleichungssystem kann nämlich als Matrixmultiplikation interpretiert werden. Dies würde hier so aussehen:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

In den folgenden beiden Grafiken wird eine Matrix Multiplikation grafisch veranschaulicht: Dabei wird ersichtlich: eine Matrix Multiplikation entspricht einer Streckung des Ortvektors eines jeden Punktes in mehrere Richtungen (hier in 2).

Die Grafik entspricht einer Multiplikation mit der Matrix:

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

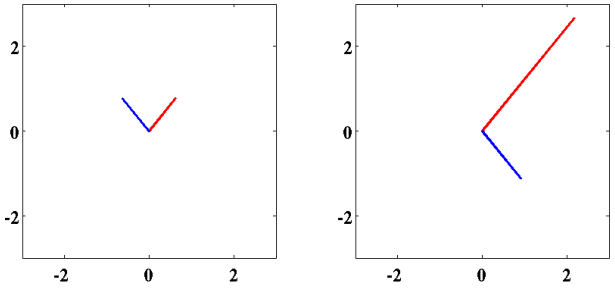


2.1.1 Eigenräume

Eigenräume sind Vektoren, welche sich nach Multiplikation mit der Matrix als Vielfaches abbilden (siehe Grafik).

Die Grafik entspricht einer Multiplikation mit der Matrix:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}.$$



Sie sind für jede Matrix eindeutig bestimmt, nämlich durch Lösung der Gleichung

$$A\mathbf{v} = \lambda\mathbf{v}$$

2.1.2 Eigenraum-Zerlegung

Es seien die Eigenräume gegeben mit:

$$\begin{aligned} A\mathbf{v_1} &= \mathbf{v_1}\lambda_1 \\ A\mathbf{v_2} &= \mathbf{v_2}\lambda_2 \end{aligned}$$

Mit den Matrizen,

$$V = \{\mathbf{v_1}, \mathbf{v_2}\}, \Lambda = diag\{\lambda_1\lambda_2\}$$

kann die Eigenraum-Zerlegung so umgeschrieben werden:

$$AV = V\Lambda$$

Ein beliebiger Vektor $\mathbf{v}$ kann durch die Lösung,

$$\mathbf{v} = V\mathbf{z}, \mathbf{z} = V^{-1}\mathbf{v}, \mathbf{z} = \langle z_1, z_2 \rangle^T$$

bezüglich der neuen Basis $\{\mathbf{v_1}, \mathbf{v_2}\}$ dargestellt werden,

$$\mathbf{v} = z_1\mathbf{v_1} + z_2\mathbf{v_2}$$

So wird die Wirkung der Matrix A,

$$A\mathbf{v} = z_1 A\mathbf{v_1} + z_2 A\mathbf{v_2} = z_1\lambda_1\mathbf{v_1} + z_2\lambda_2\mathbf{v_2}$$

d.h.

$$A\mathbf{v} = \left(V\Lambda V^{-1}\right)(V\mathbf{z}) = V(\Lambda\mathbf{z})$$

entkoppelt in 1 dimensionale Wirkungen auf jeder Achse $\{\mathbf{v_1}, \mathbf{v_2}\}$ eines neuen ver**z**errten Achsensystems.

Der Einfachheit zuliebe normalisiert man die Eigenvektoren:

$$\mathbf{v_1}^T\mathbf{v_1} = \mathbf{v_2}^T\mathbf{v_2} = 1$$

Wenn A symmetrisch ist, sind die Eigenvektoren senkrecht aufeinander.

$$\mathbf{v_1}^T\mathbf{v_2} = 0$$

Diese Bedingungen können mit Matrizen so umgeschrieben werden:

$$V^TV = I, V^{-1} = V^T$$

Somit ist V eine Drehungsmatrix. Folgendes Umschreiben entspricht einer Drehung:

$$\mathbf{v} = V\mathbf{d}, \mathbf{d} = V^T\mathbf{v}$$

So wird die Wirkung der Matrix A,

$$A\mathbf{v} = \left(V\Lambda V^T\right)(V\mathbf{d}) = V(\Lambda\mathbf{d})$$

enkoppelt in 1 dimensionale Wirkungen auf jeder Achse $\{\mathbf{v_1}, \mathbf{v_2}\}$ eines neuen ge**d**rehten Achsensystems.

2.2 Statistik

Es sei $\mathbf{x} = \{x_i\}_{i=1}^n$ eine Abtastung, also eine Folge von Messwerten. In der Statistik sind die folgenden Begriffe definiert, um die Abtastung vernünftig beschreiben zu können:

- Mittelwert:
$$\mu(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} x_i$$
- Varianz:
$$\sigma^2(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu(\mathbf{x}))^2$$

Wobei $\sigma(\mathbf{x})$ Standartabweichung heißt.

2.2.1 Zentraler Grenzwertsatz

Sei $\{x_i\}_{i=1}^n$ eine Folge von unabhängigen gleich verteilten Zufallsvariablen, mit (theoretischem) Mittelwert μ und Standartabweichung σ. Mit

$$\overline{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i$$

sei ϕ_n die Verteilung der Zufallsvariable

$$\phi_n = \frac{\overline{x}_n - \mu}{\frac{\sigma}{\sqrt{n}}}$$

Sei ϕ die Verteilung einer Gaußschen (normalverteilten) Zufallsvariable $N(0,1)$ mit Mittelwert 0 und Standartabweichung 1. Dann gilt

$$\phi_n \overset{n\to\infty}{\longrightarrow} \phi.$$

Die praktische Konsequenz aus dem Zentralen Grenzwertsatz ist: Eine lineare Mischung (Summe) ist normalverteilter als die Quellen. Dies gilt sogar für eine annähernd gleichmäßig verteilte Quellen.

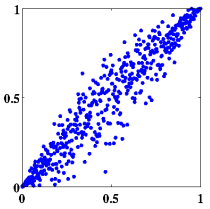
2.2.2 Kovarianz von Abtastungen

Die Varianz $\sigma^2(\mathbf{x})$ lässt sich so umschreiben:

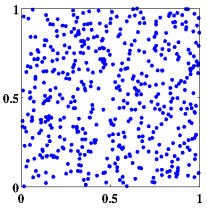
$$\sigma^2(\mathbf{x}) = \frac{1}{n}[\mathbf{x} - \mu(\mathbf{x})]^{\mathrm{T}}[\mathbf{x} - \mu(\mathbf{x})]$$

Analog gibt es eine *Kovarianz* zwischen 2 Abtastungen $\{x_i\}_{i=1}^n$ und $\{y_i\}_{i=1}^n$,

$$\kappa(\mathbf{x}, \mathbf{y}) = \frac{1}{n}[\mathbf{x} - \mu(\mathbf{x})]^{\mathrm{T}}[\mathbf{y} - \mu(\mathbf{y})]$$



$\kappa(\mathbf{x}, \mathbf{x}) = 0.0837$
$\kappa(\mathbf{x}, \mathbf{y}) = 0.0831$
$\kappa(\mathbf{y}, \mathbf{y}) = 0.0910$



$\kappa(\mathbf{x}, \mathbf{x}) = 0.0844$
$\kappa(\mathbf{x}, \mathbf{y}) = 0.0026$
$\kappa(\mathbf{y}, \mathbf{y}) = 0.0789$

Wie in der Grafik ersichtlich: je höher $\kappa(\mathbf{x}, \mathbf{y})$ ist, desto enger liegen die Punkte in Richtung des Eigenvektors von K mit dem größten Eigenwert. Je kleiner, desto ähnlicher sind die Eigenwerte.

Wir definieren die *Kovarianzmatrix* als:

$$K = \begin{bmatrix} \kappa(\mathbf{x}, \mathbf{x}) & \kappa(\mathbf{x}, \mathbf{y}) \\ \kappa(\mathbf{y}, \mathbf{x}) & \kappa(\mathbf{y}, \mathbf{y}) \end{bmatrix}$$

Für mehrere Abtastungen ergibt sich analog: Seien m Abtastungen $\{\mathbf{x}_k\}_{k=1}^m$ gegeben, und definiere

$$X = \frac{1}{\sqrt{n}} \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} - \mu(\mathbf{x}_1) \\ \vdots \\ \mathbf{x}_m^{\mathrm{T}} - \mu(\mathbf{x}_m) \end{bmatrix} \in \mathbb{R}^{m \times n}$$

so ist die Kovarianzmatrix:

$$K = XX^{\mathrm{T}} \in \mathbb{R}^{m \times m}$$

2.2.3 Korrelation

Die *Korrelation* zwischen $\mathbf{x} = \{x_i\}_{i=1}^n$ und $\mathbf{y} = \{y_i\}_{i=1}^n$ ist definiert als:

$$\chi(\mathbf{x}, \mathbf{y}) = \frac{[\mathbf{x} - \mu(\mathbf{x})]^{\mathrm{T}}[\mathbf{y} - \mu(\mathbf{y})]}{\|\mathbf{x} - \mu(\mathbf{x})\|\|\mathbf{y} - \mu(\mathbf{y})\|} = \frac{\kappa(\mathbf{x}, \mathbf{y})}{\sigma(\mathbf{x})\sigma(\mathbf{y})}$$

Wegen der Cauchy-Schwarz Ungleichung $|\mathbf{a}^{\mathrm{T}}\mathbf{b}| \leq \|\mathbf{a}\|\|\mathbf{b}\|$, ist:

$$|\chi(\mathbf{x}, \mathbf{y})| \leq 1.$$

Die *Korrelationsmatrix* ist

$$\mathrm{X} = \begin{bmatrix} \chi(\mathbf{x}, \mathbf{x}) & \chi(\mathbf{x}, \mathbf{y}) \\ \chi(\mathbf{y}, \mathbf{x}) & \chi(\mathbf{y}, \mathbf{y}) \end{bmatrix} = \begin{bmatrix} 1 & \chi(\mathbf{x}, \mathbf{y}) \\ \chi(\mathbf{x}, \mathbf{y}) & 1 \end{bmatrix} \tag{3}$$

Die Variablen sind *unkorreliert* wenn $\chi(\mathbf{x},\mathbf{y}) = \kappa(\mathbf{x},\mathbf{y}) = 0$ und damit

$$\mathrm{X} = I.$$

In diesem Fall gilt

$$K = \begin{bmatrix} \kappa(\mathbf{x},\mathbf{x}) & \kappa(\mathbf{x},\mathbf{y}) \\ \kappa(\mathbf{y},\mathbf{x}) & \kappa(\mathbf{y},\mathbf{y}) \end{bmatrix} = \begin{bmatrix} \sigma^2(\mathbf{x}) & 0 \\ 0 & \sigma^2(\mathbf{y}) \end{bmatrix} \tag{4}$$

und die Kovarianzmatrix ist diagonal.

2.2.4 Unabhängigkeit

Zum besseren Verständnis wird hier der Begriff ünkorreliert"beziehungsweise dessen Unterschied zum Begriff ünabhängig"mit 2 Beispielen veranschaulicht: Die Zufallsvariablen x und y sind unabhängig, wenn

$$P(x \in M \text{ und } y \in N) = P(x \in M) \cdot P(y \in N)$$

Beispiel: Zwei Münzen werden geworfen, und für

$$\text{1. Münze,} \quad x = \begin{cases} 1, & \text{Kopf} \\ 0, & \text{Zahl} \end{cases} \qquad \text{2. Münze,} \quad y = \begin{cases} 1, & \text{Kopf} \\ 0, & \text{Zahl} \end{cases}$$

Ein übliches Modell, das mit Erfahrung übereinstimmt, ist

$$P(x = m \text{ und } y = n) = P(x = m) \cdot P(y = n), \quad m, n \in \{0, 1\}$$

Bemerke: *Unabhängig* impliziert *Unkorrelliert*, aber nicht umgekehrt. Beispiel: Abtastungen $\mathbf{x}$ und $\mathbf{y}$ sind gleich oft in einem der Punkte $\{(\pm 1, 0), (0, \pm 1)\}$. Sie sind unkorreliert:

$$\mu(\mathbf{x}) = \mu(\mathbf{y}) = 0$$

$$\kappa(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} (0)(\pm 1)/n = 0.$$

Sie sind aber nicht unabhängig:

$$0 = P(\mathbf{x} = 1 \text{ und } \mathbf{y} = 1) \neq P(\mathbf{x} = 1) \cdot P(\mathbf{y} = 1) = \frac{1}{8}.$$

2.2.5 Gaußianität

Um in weiterer Folge den gaußschen Charakter einer Verteilung zu minimieren, stellen wir uns die Frage: Welche Eigenschaften hat eine Gaußsche Verteilung, die zu vermeiden sind, um die Gaußianität zu reduzieren? Wenn eine Abtastung $\mathbf{n} = \{n_i\}_{i=1}^n$ nach der Verteilung $N(\mu, \sigma)$ Gauß verteilt ist, erfüllen die *Momente*,

$$M_k(\mathbf{n}) = \frac{1}{n}\sum_{i=1}^{n}(n_i - \mu)^k \overset{n\to\infty}{\longrightarrow} \begin{cases} \sigma, & k = 2 \\ 3\sigma^2, & k = 4 \\ (k-1)\cdot(k-3)\cdots 1\cdot\sigma^{\frac{k}{2}}, & k \text{ gerade} \\ 0, & k \text{ ungerade} \end{cases}$$

Wölbung oder *Kurtosis* ist definiert für $\mathbf{n}$ (z.B. $\mathbf{n}_i = Y\mathbf{w}_i$) durch

$$W(\mathbf{n}) = M_4(\mathbf{n}) - 3M_2(\mathbf{n})^2$$

Für eine eher gleichmäßige Verteilung gilt $W(\mathbf{n}) < 0$. Für eine eher spitzige Verteilung gilt $W(\mathbf{n}) > 0$. Für ein Gauß verteiltes $\mathbf{n}$ gilt

$$W(\mathbf{n}) = 3\sigma^2 - 3\sigma^2 = 0.$$

Um die Gaußianität zu minimieren, könnte man folgende Funktion minimieren:

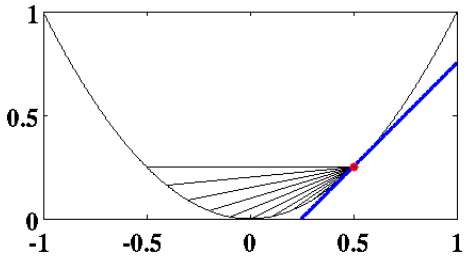
$$J(\mathbf{n}) = -W^2(\mathbf{n}).$$

2.3 Minimierung von Funktionen

2.3.1 Abstiegsverfahren

Im Folgenden wollen wir uns überlegen, wie man eine Funktion minimieren kann, wobei allerdings nur einzelne Funktionswerte, sowie Funktionswerte der 1. Ableitung ausgewertet werden können - die Funktion an und die Ableitung an sich sind unbekannt.
Sei die zu minimierende Funktion $f(x)$, mit $x \in \mathbb{R}^1$. Die Ableitung $f'(x)$ an der Stelle x ist der Limes der Steigungen von Sekantenstrecken:

$$D_x f(x) = f'(x) = \lim_{h\to 0}\frac{f(x+h) - f(x)}{(x+h) - x} = \lim_{h\to 0}\frac{f(x+h) - f(x)}{h}$$



Sei $f(x) = x^2$ mit $f'(x) = 2x$. An der Stelle $x = \frac{1}{2}$ gilt $f'(\frac{1}{2}) = 1$.
Da $f'(\frac{1}{2}) > 0$ gilt, erfüllen die nächsten vernünftigen Proben zur Minimierung von f die Ungleichung $x < \frac{1}{2}$.
Das Abstiegsverfahren ist (neues x) $x \leftarrow x - \alpha f'(x)$ (altes x) wobei das neue x das alte x ersetzt, und α wird ausgewählt, sodass $f(x)$ reduziert wird. Vernünftige Proben für α sind empirischer Weise gegeben durch

$$\alpha \in \{0, \frac{1}{2}, 1, 2\}$$

und das α wird gewählt, wo f am kleinsten ist.

2.3.2 Abstiegsverfahren für Systeme

In $\mathbb{R}^2$ (oder höher) verwendet man Richtungsableitungen für das Abstiegsverfahren. Die Richtungsableitung $D_{\mathbf{u}}f(\mathbf{x})$ an der Stelle $\mathbf{x} = (x, y)$ in der Richtung $\mathbf{u} = (u, v)$ ($\|\mathbf{u}\| = 1$) ist der Limes der Steigungen von Sekantenstrecken in einer senkrechten Ebene durch: $\mathbf{x}$ und $\mathbf{x} + \mathbf{u}$:

$$D_{\mathbf{u}}f(\mathbf{x}) = \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{u}) - f(\mathbf{x})}{h} = D_x f(\mathbf{x})u + D_y f(\mathbf{x})v$$

Für $f(x, y) = x^2 + y^2$, $\mathbf{x}_0 = (1, 1)$ und $\mathbf{u} = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ gelten

$$D_x f(x) = 2x, \quad D_y f(x) = 2y$$

und

$$D_{\mathbf{u}}f(\mathbf{x}_0) = D_x f(\mathbf{x}_0)u + D_y f(\mathbf{x}_0)v = 2\frac{1}{\sqrt{2}} + 2\frac{1}{\sqrt{2}} = 2\sqrt{2} > 0.$$

Daher läuft f bergauf an der Stelle $\mathbf{x}_0$ in der Richtung $\mathbf{u}$.
Die Richtung des steilsten Anstiegs ist

$$\mathbf{u}^*(\mathbf{x}) = \langle D_x f, D_y f \rangle^{\mathrm{T}} / \|\langle D_x f, D_y f \rangle\|$$

und das Abstiegsverfahren ist,

$$(\text{neues } \mathbf{x}) \quad \mathbf{x} \leftarrow \mathbf{x} - \alpha \mathbf{u}^*(\mathbf{x}) \quad (\text{altes } \mathbf{x})$$

für das minimierende $\alpha \in \{0, \frac{1}{2}, 1, 2\}$.

3 Principal Component Analysis

Gegeben seien nun m Abtastungen $\{\mathbf{x_k}\}_{k=1}^m$, welche Mischungen von Hauptkomponenten darstellen. Damit sind die Abtastungen korreliert. Um nun im ersten Schritt die Hauptkomponenten zu analysieren und zu trennen, könnte man die Erkenntnis aus Kapitel 2.2.3 benutzen:

> Die Variablen sind *unkorreliert* wenn $\chi(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{x}, \mathbf{y}) = 0$.

Wir haben die Kovarianzmatrix für mehrere Abtastungen $\{\mathbf{x}_k\}_{k=1}^m$ in Kapitel 2.2.2 definiert als:

$$K = XX^{\mathrm{T}} \in \mathbb{R}^{m \times m}$$

wobei

$$X = \frac{1}{\sqrt{n}} \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} - \mu(\mathbf{x}_1) \\ \vdots \\ \mathbf{x}_m^{\mathrm{T}} - \mu(\mathbf{x}_m) \end{bmatrix} \in \mathbb{R}^{m \times n}$$

Entspräche nun K der Einheitsmatrix, so wären die Kovarianzen unter den neuen Äbtastungen"null, somit auch die Korrelation von je zweien. Wir suchen nun also eine Transformation Y der Matrix X, sodass:

$$YY^{\mathrm{T}} = I$$

Die Matrix K ist symmetrisch und positiv definit, d.h.:

$$v^{\mathrm{T}} K \mathbf{v} = \mathbf{v}^{\mathrm{T}} XX^{\mathrm{T}} \mathbf{v} = (X^{\mathrm{T}}\mathbf{v})^{\mathrm{T}}(X^{\mathrm{T}}\mathbf{v}) = \|X^{\mathrm{T}}\mathbf{v}\|^2 > 0, \quad 0 \neq \mathbf{v} \in \mathbb{R}^2$$

Solche Matrizen haben Eigenvektoren mit $\mathbf{v}_i^{\mathrm{T}}\mathbf{v}_j \in \{0, 1\}$ und positive Eigenwerte $\lambda_i > 0$.
Mit der in Kapitel 2.1.2 überlegten Eigenraum-Zerlegung,

$$KV = V\Lambda, \quad K = V\Lambda V^{\mathrm{T}}$$

folgt

$$V\Lambda V^{\mathrm{T}} = XX^{\mathrm{T}}, \quad \Lambda = V^{\mathrm{T}} XX^{\mathrm{T}} V$$

Da Λ eine diagonale Matrix ist, ist $\Lambda^{-\frac{1}{2}}$ wohl definiert. Es folgt

$$I = \Lambda^{-\frac{1}{2}}\Lambda\Lambda^{-\frac{1}{2}} = \Lambda^{-\frac{1}{2}} V^{\mathrm{T}} XX^{\mathrm{T}} V \Lambda^{-\frac{1}{2}} = (X^{\mathrm{T}} V \Lambda^{-\frac{1}{2}})^{\mathrm{T}}(X^{\mathrm{T}} V \Lambda^{-\frac{1}{2}})$$

Daher ist die gesuchte Transformation $Y = \Lambda^{-\frac{1}{2}} V^{\mathrm{T}} X$, sie erfüllt $YY^{\mathrm{T}} = I$.

Wenn $m \gg n$ gilt, wird die obige Zerlegung leichter durch folgende Eigenraum-Zerlegung bestimmt:

$$X^{\mathrm{T}}X = \hat{Y}\hat{\Lambda}\hat{Y}^{\mathrm{T}} \in \mathbb{R}^{n\times n}$$

wobei $\Lambda = \begin{bmatrix} \hat{\Lambda} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{m\times m}, Y = \begin{bmatrix} \hat{Y} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{m}$ und $\hat{Y}^{\mathrm{T}}\hat{Y} = I$. Mit $\hat{V} = X\hat{Y}\hat{\Lambda}^{-\frac{1}{2}} \in \mathbb{R}^{m\times n}$ folgt

$$K\hat{V} = X(X^{\mathrm{T}}X)\hat{Y}\hat{\Lambda}^{-\frac{1}{2}} = X\hat{Y}\hat{\Lambda}\hat{Y}^{\mathrm{T}}\hat{Y}\hat{\Lambda}^{-\frac{1}{2}} = (X\hat{Y}\hat{\Lambda}^{-\frac{1}{2}})\hat{\Lambda} = \hat{V}\hat{\Lambda}.$$

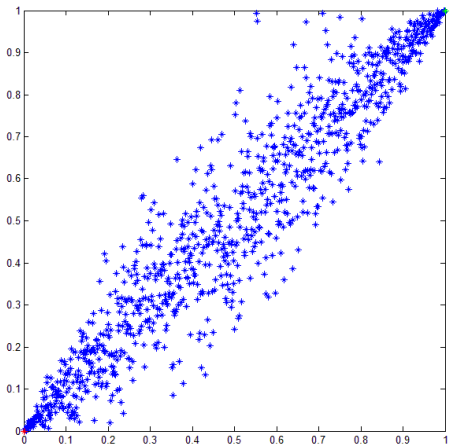
Für jede Ergänzung $V = [\hat{V}, \tilde{V}]$ mit $V^{\mathrm{T}}V = I$ folgt $KV = V\Lambda$.

Dieser Prozess - also die Erstellung der Transformation - heißt Principal Component Analysis. Diese Daten werden auch als „whitened“ bezeichnet.

3.1 Unsere Arbeit mit PCA

Wir setzten uns intensiv mit der PCA auseinander, die wir zuerst theoretisch bearbeiteten und durch deren Hilfe es uns auch ermöglicht war eine Videodatei stark zu komprimieren.

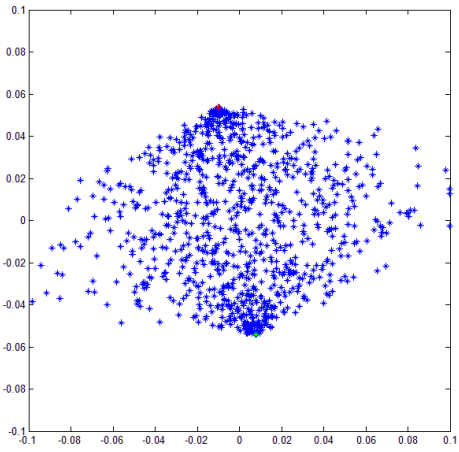
Als Einstieg in das System der PCA sollten wir dieses Verfahren zuerst mit künstlichen von uns selbst am Computer erzeugten Daten durchführen. Die knstlichen und gemischten Quellen stellten wir grafisch dar und ordneten somit den Werten jeder einzelnen Spalte unserer Matrix mit den sogenannten rohen Daten einen x- und einen y-Wert in einem Streudiagramm zu, was uns zu dem nebenstehenden Resultat führte.



Unsere Aufgabe war nun diese Daten zu zentrieren, so dass ihr Mittelwert null ergab und eine Kovarianzmatrix (vgl. Kapitel 3) zu berechnen, die uns angibt wie stark die Daten um eine bestimmte Hauptachse verstreut sind. Man müsste nun die exakten Eigenräume, also sozusagen die Hauptachsen der ursprünglichen Achsen und die Eigenwerte, also die Länge dieser Hauptachsen berechnen, was wir mit einer auf die Kovarianz basierenden Matlab-Funktion erreicht haben.

Der Sinn dieser Durchführung ist die Normierung der Eigenwerte auf den

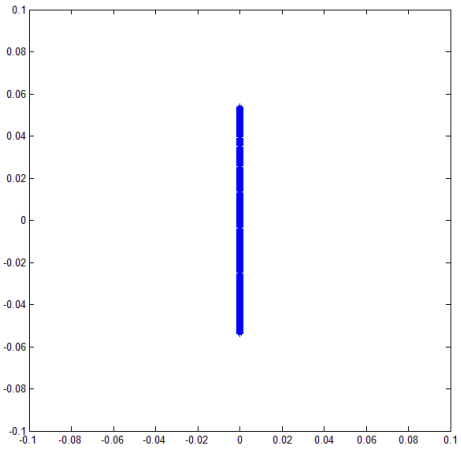
Wert 1, sodass auch die Varianz um den Koordinatenursprung 1 beträgt. Die Durchführung funktioniert durch die Umformung der ursprünglich in einer Diagonalmatrix angegeben Eigenwerte zu einem Vektor, von dem

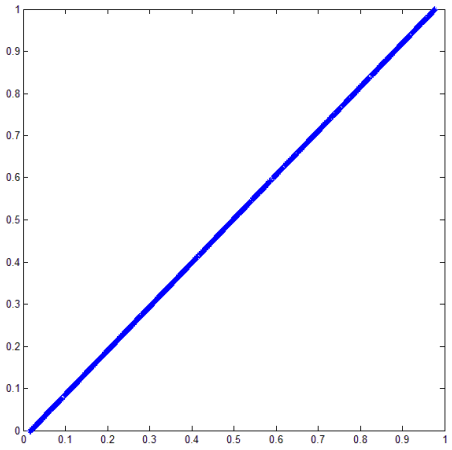


der Kehrwert der Wurzel berechnet wird, um diesen danach wieder in eine Diagonalmatrix umzuwandeln. Durch die Multiplikation dieser Matrix mit der auf Eigenräumen basierenden Drehungsmatrix und den zentrierten Daten (vgl. Kapitel 3) erhielten wir einen Datensatz, der die von uns erwarteten Anforderungen erfüllte - die Daten waren unkorreliert. Diesen stellten wir wiederum grafisch dar, wie links zu sehen ist.

Somit war die zentrale Frage der PCA geklärt und wir waren für neue Problemstellungen offen. Die neue Aufgabe war es alle entstandenen Daten auf eine Achse zusammenfallen zu lassen. Die Überlegung dahinter war, dass man für die Speicherung dieser Daten bei einer Matrix mit zwei Zeilen nach dem Zusammenfallen nur mehr einen Wert für jeden Datensatz, also insgesamt 1001 Werte und zusätzlich dazu eine 2x2 Matrix benötigt während

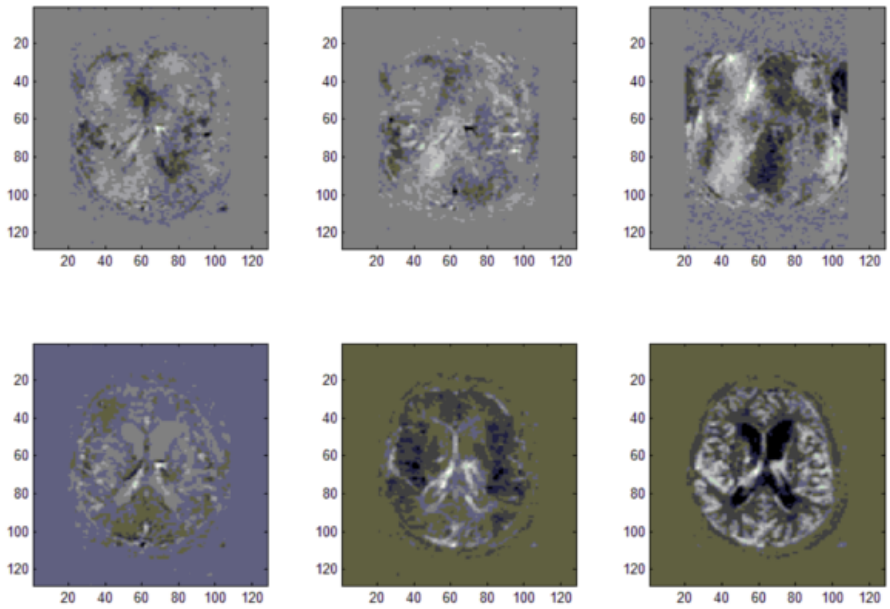
es für jeden Datensatz vorher zwei also insgesamt 2002 Werte waren. Das brachten wir zustande indem wir den schwächeren Eigenwert der Diagonalmatrix, die wir vorher bei der Multiplikation (vgl. „Λ“ Kapitel 3) verwendet hatten, vernüllten und sich alle Werte somit auf dem Eigenraum mit dem größeren Eigenwert befanden, was in einem Koordinatensystem eingetragen eine Strecke von Werten ergibt, wie sie rechts ersichtlich ist.





In weiterer Folge wurden die erhaltenen Daten wieder mit der durch Eigenwerte befüllten Diagonalmatrix multipliziert, um sie in ihre ursprüngliche Dehnung bzw. Stauchung zu bringen. Dadurch war es uns möglich das Datenvolumen fast auf die Hälfte des ursprünglichen Datensatzes zu verringern. Das Ergebnis haben wir wieder in einem Koordinatensystem darstellen lassen, wie links zu sehen ist.

Nun war unsere Vermutung, dass sich dieses Verfahren auch auf andere Datensätze, beispielsweise auf Audio- oder Videodateien anwenden ließe. Wir versuchten es also mit einem Video einer Magnetresonanztomographie eines Gehirns, das die Auflösung von 128x128 Pixel und eine Folge von 50 Bildern enthielt. Dieses stellten wir zuerst als zweidimensionale Matrix mit 50 Zeilen und 128 Spalten dar, um es mit unserem vorgefertigten Matlab-Code bearbeiten zu können. Der Unterschied zu unserem theoretischen Code war, dass wir es statt mit einem zweidimensionalen mit einem fünfzigdimensionalen Raum zu tun hatten. Vorher konnten wir nur einen Eigenwert gleich null setzten, jetzt waren es von den insgesamt fünfzig 44 Eigenräume, die unbrauchbar waren, weil Eigenwerte sehr nahe bei Null besan, somit nahezu nichts zum ursprnglichen Video betrugen und deswegen vernachlässigt werden konnten. Wir waren also in der Lage den fünfzigdimensionalen Raum auf einen sechsdimensionalen zu reduzieren, was das Datenvolumen von 806 KB auf 72 KB verringerte, weil wir mit folgenden sechs Bildern das Video so rekonstruieren konnten, dass für das menschliche Auge kein Unterschied erkennbar ist. Bei diesem Video gab es keine gro Bewegung innerhalb des Bildes, deswegen waren die einzelnen Eigenwerte für die ersten 43 Einzelbilder so klein und die anderen Bilder konnten leicht durch die folgenden sechs Hauptkomponenten ausgedrückt werden.



Wir vermuteten, dass, wenn wir ein Video mit einer stärkeren Bewegung verwenden würden, die Trennung nicht möglich wäre, weil alle Eigenwerte zu stark sein würden, um sie gleich null zu setzen. Wir erzeugten also ein Video, in dem sich eine Person vom einen zum anderen Bildrand bewegte und unsere Vermutung erwies sich als richtig. Es lien sich hier nicht alle Bilder durch nur wenige Hauptkomponenten darstellen. Insgesamt bringt das Verfahren bei Kontrastvideos also unglaublich gro Einsparungsmöglichkeiten, bei „normalen“ Videos hingegen fast nichts.

4 Independent Component Analysis

Wir haben nun die Korrelation zwischen den Komponenten vernullt. Damit sind diese schon recht gut getrennt, was auch bei den Großteil unserer Arbeiten bereits ausreichend war. Man kann aber die Komponenten noch weiter trennen - und damit das Verfahren für spezielle Anwendungen optimieren. Dabei wird der Zentrale Grenzwertsatz (vgl. Kapitel 2.2.1) ausgenützt, denn es ist nämlich:

Eine lineare Mischung (Summe) ist normalverteilter als die Quellen.

Könnte man nun den Gaußschen Faktor minimieren, so würde man in den meisten Fällen die Komponenten weiter trennen. Im Kapitel 2.2.5 haben

wir bereits überlegt, man könnte dafür folgende Funktion minimieren:

$$J(\mathbf{w}) = -W^2(\mathbf{w}Y) \quad \text{wobei} \quad Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}.$$

Wobei hier Y die whitened Daten (vgl. Kapitel 3 sind und $\mathbf{w}$ ein Vektor mit Koeffizienten für die Darstellung der optimierten Komponenten X_o als gewichtete Summe der Y_i ist. Das optimierte Ergebnis hätte also die Form:

$$X_o(i) = \mathbf{w_i}Y$$

beziehungsweise für die Matrix W mit

$$W(i) = \mathbf{w_i}^{\mathrm{T}}$$

ergibt sich:

$$X_o = WY.$$

Gesucht ist nun eine Matrix W dessen Zeilen w_i die minimale Gaußianität von X bewirken, somit also Minima der Funktion $J(\mathbf{w})$ sind.
Um diese $\mathbf{w}$ zu bestimmen verwendeten wir ein Abstiegsverfahren - wie in Kapitel 2.3.2 gezeigt.
Die zu minimierende Funktion ist also nun:

$$J(\mathbf{w}) = -W^2(Y\mathbf{w}) = -[M_4(Y\mathbf{w}) - 3M_2^2(Y\mathbf{w})]^2$$

Der Gradient dieser Funktion ist

$$D_{\mathbf{w}}J(\mathbf{w}) = -2[M_4(Y\mathbf{w}) - 3M_2^2(Y\mathbf{w})]\times$$

$$[D_{\mathbf{w}}M_4(Y\mathbf{w}) - 6M_2(Y\mathbf{w})D_{\mathbf{w}}M_2(Y\mathbf{w})]$$

wobei

$$D_{\mathbf{w}}M_k(Y\mathbf{w}) = \frac{k}{n}\sum_{i=1}^{n}(\mathbf{e}_i^{\mathrm{T}}V\mathbf{w})^{k-1}(V^{\mathrm{T}}\mathbf{e}_i)$$

Die Richtung des steilsten Anstiegs ist

$$\mathbf{u}^*(\mathbf{x}) = D_{\mathbf{w}}J(\mathbf{w})/\|D_{\mathbf{w}}J(\mathbf{w})\|$$

und das Abstiegsverfahren ist,

$$(\text{neues } \mathbf{w}) \quad \mathbf{w} \leftarrow \mathbf{w} - \alpha\mathbf{u}^*(\mathbf{w}) \quad (\text{altes } \mathbf{w})$$

für das minimierende $\alpha \in \{0, \frac{1}{2}, 1, 2\}$, wobei mit

$$\mathbf{w} \leftarrow \mathbf{w}/\|\mathbf{w}\|$$

normalisiert wird. Auch wenn $\mathbf{w}^\star$ schon für eine andere Komponente bestimmt worden ist, wird $\mathbf{w}$ so projiziert

$$\mathbf{w} \leftarrow \mathbf{w} - \mathbf{w}^\star(\mathbf{w}^{\mathrm{T}}\mathbf{w}^\star).$$

Wir erhalten also nun die Drehungsmatrix W, bei der die Gaußianität der X_o minimiert wurde. Durch die Drehung werden die unabhängigen Komponenten voneinander getrennt. Dieses Verfahren heißt Indepentent Component Analysis.

4.1 Unsere Arbeit mit ICA

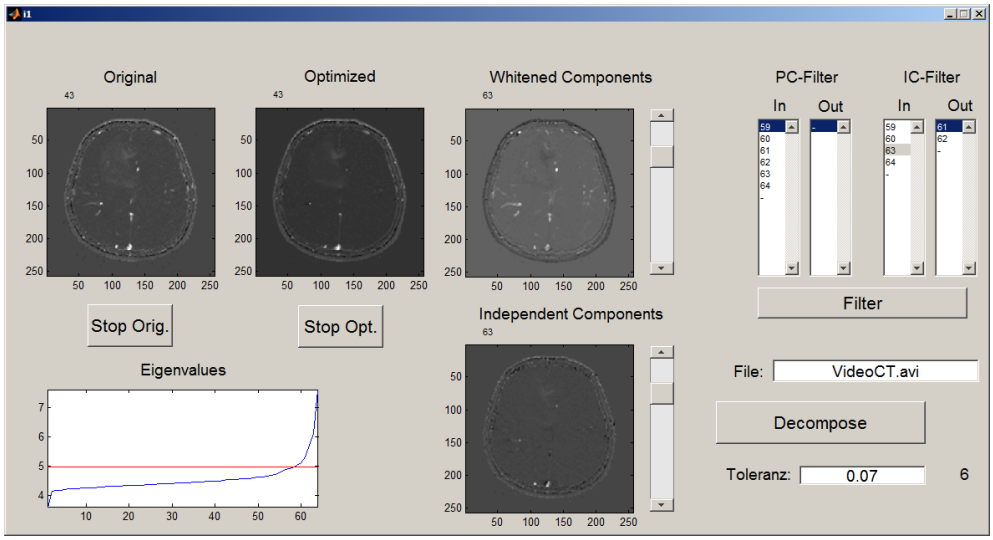
In der Woche wurde uns die Aufgabe gestellt mehrere Videos in ihre Haupt und unabhige Komponenten zu zerteilen. Dies würde es erlauben unerwünschte Komponenten (z.B. rauschen) zu entfernen um ein hetischeres Endprodukt zu erhalten. Grundslich ist diese Problemstellung dieselbe wie bei dem "Cocktail-Party-Problem", mit dem entscheidenden Unterschied, dass wir Komponenten hinauswerfen (Dimensionen entfernen) und anschliend die ICA und PCA zu reversieren um ein neues vollstiges Bild zu erhalten.

Um dies zu erreichen, stellen wir jedes Bild als einen Vektor dar, wobei die Zeilen des Bildes hintereinander gereiht werden. Somit erhält man eine $xy \times t$ Matrix wobei x und y die Auflösung und t die Bilderzahl des Videos ist. Unsere verschiedenen Vektorbilder sind somit analog zu vorher unsere Aufnahmen, wobei wir jetzt für den allgemeinen Fall einem Raum $\mathbb{R}^t$ erstellen, wo wir jeden Pixel über alle t als t-dimensionalen Punkt auffassen. Diesen t-dimensionalen Datensatz können wir zentrieren, und mithilfe der Eigenvektoren der Kovarianzmatrix so stauchen und dehnen, das die Kovarianzmatrix des *whitened* Datensatzes der Einheitsmatrix entspricht.

Somit haben wir die Hauptkomponenten in unserer $xy \times t$ Matrix in getrennter Form. Bei unserer Bearbeitung stellte sich heraus das bei verschiedenen Filmen sich eine Grenze existiert unter der alle Hauptkomponenten reines rauschen darstellen. Somit können wir die Matrix so umformen das sie alle Komponenten deren Eigenwerte unter einem gewissen Wert fallen nicht beinhalten. Jetzt besitzt unsere Matrix eine weitaus geringere Dimension und somit lasst sich mithilfe des Abstiegsverfahrens effizient und schnell dieselbe Anzahl and unabhängigen Komponenten berechnen.

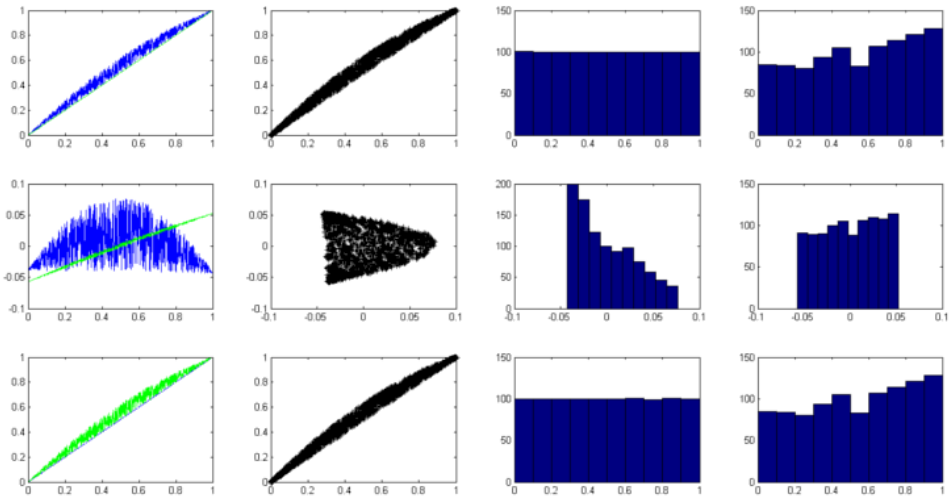
Aus unserer unabhängigen Komponente (die aus den reduzierten Hauptkomponenten berechnet wurden) konnten wir wiederum eine Menge an Kompo-

nenten auswählen die wir entfernen wollten, und daraus ein Endvideo berechnen. Damit diese Komponenten Zeiteffizient evaluiert werden können, entwickelten wir ein MatLab Programm das es uns einfach ermöglicht Haupt und unabhängige Komponenten herauszufiltern.



4.1.1 Optimierung mithilfe einer Drehungsmatrix

Eine unserer weiteren Arbeiten war mithilfe einer „Brute-Force“ Methode eine Optimierung zur ICA durchzuführen. Dazu wurden wieder künstliche Aufnahmen erzeugt, mit denen dann durch den bereits bestehenden Code zwei getrennte Hauptkomponenten berechnet wurden. Die jeweiligen Histogramme der Hauptkomponenten zeigten nun keine gleichmäg verteilten Werten, wie sie bei den rohen Daten ersichtlich waren. Unser Ziel war es diese Werte möglichst zu verflachen, sie also auf ein ungefähr gleiches Level zu bekommen. Das gelang uns durch suchen der Stelle in den Werten, bei der die kleinste Gauanität aufzufinden war, die wir durch eine zweidimensionale Drehungsmatrix auffanden. Somit war es uns möglich, die Quellen wieder zu trennen. Grafisch dargestellt war das Ergebnis Folgendes.



In der ersten Zeile befinden sich die künstlich erzeugten Quellen, die anschliend gemischt wurden, in der zweiten die unkorrelierten Hauptkomponenten dieser Mischungen und in der dritten die wieder getrennten unabhängigen Komponenten. Wir wiederholten dieses Beispiel auch mit einer Matrix der rohen Daten, die statt zwei drei Zeilen aufwies und das Ergebnis war das gleiche. Mithilfe dieses Codestücks war es auch möglich tatsächliche Tonaufnahmen zu trennen und somit das Cocktail-Party-Problem zu lösen, wie im nächsten Kapitel gezeigt wird.

5 „Cocktail-Party-Problem“

Hintergrund

Das so genannte „Cocktail-Party-Problem“ beschäftigt die Wissenschaft schon seit Jahrzehnten. Eine Grundfrage dabei ist: Wie gelingt es dem Gehirn während einer Cocktail-Party bestimmte Quellen, z.B. Geräusche, Stimmen, Musik, zu erkennen? Man vermutet, dass wir im Laufe unseres Lebens eine Art Musterbibliothek in unserem Gehirn anlegen.

Bisherige Modelle arbeiten nach folgendem Prinzip: Ein Archiv im menschlichen Gehirn vergleicht die ankommenden Signale einzeln mit bereits vorhandenen Mustern. Als Ergebnis erhält es die übereinstimmungsmenge mit allen Mustern und schafft es somit die Quellen zu identifizieren.

Problemstellung

Unser Modell unterscheidet sich im wesentlichen vom oben genannten Verfahren darin, dass wir kein komplexes Nervensystem wie ein menschliches Gehirn zur Verarbeitung und Signaltrennung der Aufnahmedaten verwenden, sondern versuchen einer Maschine, die mit einzelnen Stimmen, Tnen, etc. nicht viel anzufangen wei beizubringen mehrere Aufnahmemischungen in ihre einzelnen Ausgangsquellen zu trennen.

Vorbereitung

Nach der Frage der richtigen Arbeitsoberfläche fiel die Entscheidung auf die Skriptsprache 'Matlab'. Sie ermglicht uns einen angenehmen Umgang mit Matrizen, welcher für uns ntig ist, da wir alle unsere Aufnahmen in einer Matrix speichern wollen. Des weiteren wurde eine kleine Auswahl an Mikrofonen zur Verwendung für unsere Aufnahmen zusammengestellt.

Planung

Die Idee hinter unserem Vorhaben ist es mit zwei Aufnahmegeräten gleichzeitig zwei Quellen aufzunehmen und die Daten in Matlab einzuschreiben. Da bei den beiden Aufnahmen, aufgrund der verschiedenen Abstände zu den Mikrofonen, jeweils die eine Quelle lauter als die andere ist, sollte es mglich sein mit Matlab ein Skript zu schreiben, das ähnliche Daten bzw. linear abhängige Entwicklungen in den Matrizen erkennt und die beiden Ursprungsquellen dann wieder trennt. Aufgrund der stark variierenden Qualität der Aufnahmegeräte mussten wir die tatsächliche Umsetzung allerdings etwas verändern, das hei wir haben heben zwei gleichzeitige Aufnahmen zweier Quellen lediglich simuliert.

Künstliche Simulation

Versuch 1: Um uns mit der Durchführung des Cocktail Party Problems vertraut zu machen, begannen wir das Grundprinzip unseres Vorhabens in einem Skript künstlich zu simulieren. Anstelle zweier gleichzeitig von zwei Mikrofonen aufgenommener Quellen, machten wir zwei voneinander unabhängige Aufnahmen zweier unterschiedlicher Geräuschquellen.

- Geräuschquelle 1: Eine etwas tiefere Stimme liest etwas 'ha.wav'
- Geräuschquelle 2: Eine etwas hhere Stimme singt ein Lied 'ko.wav'

Die Daten der beiden Audiospuren werden mit dem Befehl „wavread“ eingelesen und definiert.

```
ha = wavread('ha.wav');
ko = wavread('ko.wav');

ha = ha(:,1);
ko = ko(:,1);
```

Die Längen werden definiert und angeglichen.

```
hale = length(ha);
kole = length(ko);

if (kole>hale)
    ha = [ha;zeros(kole-hale,1)];
else
    ko = [ko;zeros(hale-kole,1)];
end
n = length(ha);
```

In Z werden die beiden, nun gleich langen, Audiospuren gespeichert.

```
Z = [ha';ko'];
```

Mit A verwenden wir eine Matrix um die Audiospuren künstlich zu mischen.

```
A = [4,3;2,1];
```

Wir definieren mit Y die künstliche Mischung, also die Ausgangsquellen mit unserer Matrix A.

```
Y = A*Z;
```

Basierend auf dem Befehl 'linspace', n Werte werden zwischen 0 und 10 gleichmäg gestreut, definieren wir t.

```
t = linspace(0,10,n);
```

Wir verwenden W, die inverse Matrix von A, um die Mischungsmatrix Y wieder in X, die vermeintlichen Ausgangswerte zu trennen.

```
W = [-0.5,1.5;1,-2]
X = reshape(W,2,2)*Y;
```

Mit 'subplot' definieren wir die einzelnen Darstellungsfenster für unsere Grafik, 'plot' wiederum definiert die Diagramme und Histogramme der Quellwerte, Mischungsmatrix und Ausgangwerte.

```
subplot(3,4,1)


plot(t,Z(1,:),'b',t,Z(2,:),'g');
subplot(3,4,2)
plot(Z(1,:),Z(2,:),'k*');
subplot(3,4,3)
hist(Z(1,:),30)
subplot(3,4,4)
hist(Z(2,:),30)

subplot(3,4,5)
plot(t,Y(1,:),'b',t,Y(2,:),'g');
subplot(3,4,6)
plot(Y(1,:),Y(2,:),'k*');
subplot(3,4,7)
hist(Y(1,:),30)
subplot(3,4,8)
hist(Y(2,:),30)

subplot(3,4,9)
plot(t,X(1,:),'b',t,X(2,:),'g');
subplot(3,4,10)
plot(X(1,:),X(2,:),'k*');
subplot(3,4,11)
hist(X(1,:),30)
subplot(3,4,12)
hist(X(2,:),30)
```

Mit dem Befehl 'wavwrite' geben wir die gemischten und getrennten Audiospuren aus.

```
wavwrite (Y(1,:),45000,'gemischt1.wav')
wavwrite (Y(2,:),45000,'gemischt2.wav')
wavwrite (X(1,:),45000,'getrennt1.wav')
wavwrite (X(2,:),45000,'getrennt2.wav')
```

Wie erwartet gleichen die getrennten Audiospuren den ursprünglichen Quellspuren, haben wir die Mischung doch nur künstlich mit einer uns bekannten

Matrix erzeugt, und dann wieder mit der inversen Matrix getrennt.

Die eigentliche Problemstellung liegt aber darin die Mischung zu trennen ohne die Matrix A, mit der sie erzeugt wurde, zu wissen. Um uns die Suche nach einer uns unbekannten inversen Matrix W zu erleichtern, haben wir uns 'Simulink', einer Erweiterung Matlabs, zugewendet.

Mit der Annahme die Matrix A nicht zu kennen, erstellten wir ein Simulink-Programm, welches uns beliebig die Parameter unserer inversen Matrix W verändern lässt, bis diese ein verwertbares Ergebnis liefert. Angenommen wir verändern die Parameter so lange bis diese schlussendlich ein brauchbares Ergebnis liefern, stimmen diese mit unseren zuvor errechneten Werten der inversen Matrix überein und liefert eine Trennung der Mischung.

Durchführung

Als tatsächliche Lösung des 'Cocktail-Party-Problems' kann man unsere Durchführung auf Basis einer veränderten Version des 'Bildkomprimierung'-Skripts unserer Kollegen betrachten. In diesem Fall verwenden wir (wie bereits im Absatz 'Planung' erwähnt) zwei Audiospuren von zwei gleichzeitig aufgenommenen Quellen, die in unserem konkreten Fall nur simuliert werden, da das für ein optimales Ergebnis bentigte Material nicht zur Verfügung steht. Es wird also vorab simuliert, dass je eine Schallquelle näher beim einen Mikrofon ist, als die andere. Wir mischen also zwei getrennt aufgenommene Schallquellen miteinander in insgesamt zwei Audiodateien, wobei in je einer davon, die eine Quelle lauter als die andere gemacht wird.

Die Daten der beiden simulierten Audiospuren werden mit dem Befehl 'wavread' eingelesen und definiert. Auch die beiden getrennten Aufnahmen werden zum nachträglichen Vergleich eingeschrieben.

```
glass = wavread('cocktail.wav');
joe = wavread('party.wav');
glass_single = wavread('cocktail_single.wav');
joe_single = wavread('party_single.wav');
```

Die Längen werden angeglichen.

```
n = min([length(joe),length(glass),length(joe_single),length(glass_single)]);
joe=joe(1:n);
glass=glass(1:n);
joe_single=joe_single(1:n);
glass_single=glass_single(1:n);
```

Wir reservieren Platz für die rohen Daten. Die Quellen Xr werden mit der erfundenen Matrix A zur Mischung Yr.

```
t = linspace(0,1,n);
n1 = sqrt(n);

Xr=zeros(2,n);
Xr(1,:)=glass_single(1:n);
Xr(2,:)=joe_single(1:n);

Yr = zeros(2,n);
Yr(1,:) = glass(1:n);
Yr(2,:) = joe(1:n);

A = [2,1;1,2]/3;
Yr = A*Xr;
```

Mit dem Befehl 'zeros' wird Platz für zentrierte Daten reserviert.

```
Y0 = zeros(2,n);
```

Es werden die Mittelwerte der einzelnen Datensätze von Yr berechnet und von den Datensätzen abgezogen, damit man zentrierte Daten bekommt. Es wird der Eigenraum berechnet um später beim so genannten Whitening"die Achsen des Eigenraumes zu stauchen bzw. zu dehnen, sodass die Daten als Wolke um den Koordinatenursprung dargestellt werden.

```
Ym(1) = mean(Yr(1,:));
Ym(2) = mean(Yr(2,:));

Y0(1,:) = (Yr(1,:) - Ym(1))/n1;
Y0(2,:) = (Yr(2,:) - Ym(2))/n1;

K = Y0*Y0';

[V,D] = eig(K);

S=sqrt(diag(D));
S=(S ~= 0)./(S +(S == 0));

S=diag(S);

Yw = S*V'*Y0;
```

Mithilfe einer 2x2 Drehungsmatrix wird der Datensatz nach der Stelle mit der kleinsten Gaußianität abgetastet. Dieser wurde durch eine Matrix dargestellt

und mit dieser Matrix konnten die unkorrelierten Daten voneinander getrennt werden.

```
im = 100;
J = [];
J1 = [];
J2 = [];

for i=1:im
    th = 2*(i-1)*pi/im;
    W = [cos(th),sin(th);-sin(th),cos(th)];
    Z = W*Yw;

    M2z1 = sum(Z(1,:).^2)/n;
    M4z1 = sum(Z(1,:).^4)/n;

    M2z2 = sum(Z(2,:).^2)/n;
    M4z2 = sum(Z(2,:).^4)/n;

    Wz1 = M4z1 - 3*(M2z1)^2;
    Wz2 = M4z2 - 3*(M2z2)^2;
    J1 = [J1,-Wz1^2];
    J2 = [J2,-Wz2^2];
    J = [J,J1(end)+J2(end)];
end
```

Nun berechnet man i, das man zur Berechnung von th, dem Winkel der Drehungsmatrix W, benötigt. Mithilfe von W werden die Datensätze getrennt.

```
i  = find(J==min(J),1);
Ji = J(i);
th = 2*(i-1)*pi/im;
W = [cos(th),sin(th);-sin(th),cos(th)];

Zn = W*Yw;
Zn(1,:) = Zn(1,:)*n1 + Ym(1);
Zn(2,:) = Zn(2,:)*n1 + Ym(2);
Zn = Zn/max(Zn(:));
```

Die Datensätze werden als Wavesound-Dateien exportiert

```
wavwrite (Zn(1,:),45000,'getrennteins.wav')
wavwrite (Zn(2,:),45000,'getrenntzwei.wav')
```

Die einzelnen Exportdateien sind bis auf winzige Restgeräusche der jeweils anderen Quelle getrennt, das heißt das „Cocktail-Party-Problem“ wurde auf Basis einer Simulation erfolgreich gelst.

Fehlerdiskussion

Aufgrund der erforderlichen Genauigkeit beim Erstellen einer Mischung kam es häufig zu unerwünschten Ergebnissen. Bevor man sich schlussendlich dazu entschied die Quellaufnahmen künstlich zu erstellen, gab es viele Fehler mit auf zwei Mikrofonen basierenden durchgeführten Aufnahmen. Es stellte sich als ein Problem heraus, dass die beiden Tonaufnahmen nicht synchron aufgenommen werden konnten und auch nachträglich war es nicht mglich die Aufnahmen exakt abzustimmen, da auch nur die kleinsten Abweichungen verhindern knnen, dass Matlab die Daten richtig nutzt. Des weiteren führten Qualitätsmängel der Mikrofone dazu, dass die Aufnahmen teilweise durch ein Rauschen verfälscht wurden. Es besteht die Vermutung, dass Matlab mglichst reineÄufnahmen benötigt um brauchbare Ergebnisse zu liefern.

Eine weitere Ursache für entstandene Fehler könnte die zu niedrige Lautstärke mancher Schallquelle in den Audiospuren sein, eine Folgeerscheinung von falscher Positionierung der Mikrofone.

Auf Basis der oben aufgeführten Simulation gelang schlieich die Durchführung des Experiments.