

WOCHE DER MODELLIERUNG MIT MATHEMATIK



PÖLLAU, 6.2.–12.2.2016

WEITERE INFORMATIONEN:

[HTTP://MATH.UNI-GRAZ.AT/MODELLWOCHE/2016/](http://math.uni-graz.at/modellwoche/2016/)

ORGANISATOREN UND SPONSOREN



KOORDINATION

Thomas Russold, BA MA



Alexander Sekkas



Mag. DDr. Patrick-Michel Fröhmann



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 elften 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 Herrn Landesschulinspektor Mag. Gerhard Sihorsch und Frau Fachinspektorin Mag. Michaela Kraker, für die Hilfe bei der Organisation und seine kontinuierliche Unterstützung der Idee einer Modellierungswoche.

Ohne den idealistischen, unentgeltlichen und engagierten Einsatz der direkten Projektbetreuer Dr. Dipl.-Math.techn. Tobias Breiten, Dr. Konstantinos Xylouris, Michael Kniely, BSc BSc MSc MSc und Dr. Laurent Pfeiffer – Institut für Mathematik und Wissenschaftliches Rechnen – hätte diese Modellierungswoche nicht stattfinden können.

Besonderer Dank gebührt ferner Herrn Mag. DDr. Patrick-Michel Frühmann, der die ganze Veranstaltung betreut und auch die Gestaltung dieses Berichtes übernommen hat, Herrn Thomas Russold, BA MA, 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 von der Karl-Franzens-Universität Graz durch Vizerektor Prof. Dr. Martin Polaschek und Dekan Prof. Dr. Christof Gattringer, vom regionalen Fachdidaktikzentrum für Mathematik und von Comfortplan.

Pöllau, am 12. Februar 2016

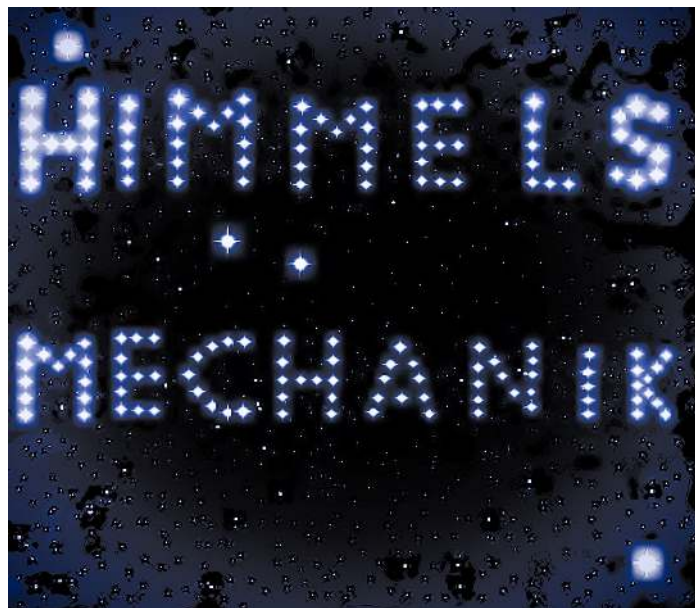
Bernd Thaller
Institut für Mathematik und Wissenschaftliches Rechnen
Karl-Franzens-Universität Graz



Himmelsmechanik

Lagrange-Punkte im Sonnensystem

Siegfried Kaidisch, Nina Maria Lampl, Niklas
Tscheppe, Julia Stadlmann, Michael Karl
Steinbauer, Victoria Zeiler



Betreuer: Michael Kniely BSc BSc MSc MSc

Inhaltsverzeichnis

1	Zielsetzung	3
2	Physikalischer Hintergrund	3
2.1	Newton'sche Axiome	3
2.2	Natürliche Einheiten und Dimensionen	4
2.3	Definition der Skalen	5
3	Berechnung von Ort und Geschwindigkeit	6
3.1	Differentialgleichungen	6
3.2	Algorithmus	6
3.3	Umsetzung	8
3.4	Impulserhaltung	9
3.5	Energieerhaltung	10
4	n-Körper-Problem	11
5	Lagrange-Punkte	12
5.1	Lokalisierung	12
5.2	Gesamtpotential	14
5.3	Corioliskraft	14
5.4	Matrizen	15
6	Ergebnisse	16
6.1	3-Körper-Problem	16
6.2	Sonnensystem	16
6.3	Lagrange-Punkte	18



1 Zielsetzung

Das Ziel hinter der Projektgruppe „Himmelsmechanik“ war es einerseits sowohl unser Sonnensystem als auch 3-Körper-Probleme zu modellieren und andererseits etwas in die Materie der sogenannten „Lagrange-Punkte“ einzutauchen.

Das 2-Körper-Problem wurde ja schon vor einiger Zeit analytisch gelöst, doch das 3-Körper Problem ist auf diese Weise nicht zu lösen. Das Beste, was man sich hier erhoffen kann, ist eine numerische Annäherung und genau solch eine haben wir mit unserer Simulation versucht zu finden.

Doch wie sieht es mit mehr als 3 Körpern aus — mit 4, 5, 10 oder gar 100? Tatsächlich ähnelt das 3-Körper-Problem stark dem allgemeinen n -Körper-Problem. Eines dieser Probleme stellt natürlich auch die Modellierung unseres Sonnensystems dar. Bei der realistischen Simulation rund um unseren Heimatstern haben uns allerdings natürlich einige neue Probleme erwartet. Energie- und Impulserhaltung, (natürliche) Einheiten sowie einiges Andere stellten Hindernisse dar, die es zu überwinden galt.

Letztlich galt es die sogenannten Lagrange-Punkte, spezielle Gleichgewichtspunkten im Kosmos, zu finden. In ihnen gleichen sich die Gravitationskräfte zu Sonne und Erde als auch die Zentrifugalkraft genau aus. Ein Objekt, das sich einmal genau in solch einem Punkt befindet, bleibt (zumindest theoretisch) für immer an dieser Stelle.

2 Physikalischer Hintergrund

2.1 Newton'sche Axiome

Als Grundlage unseres Projektes fungiert das zweite Newton'sche Axiom, welches besagt, dass ein Körper durch eine einwirkende Kraft eine Beschleunigung erfährt, die zur einwirkenden Kraft proportional ist.

Angenommen es existieren n Körper mit den Massen $m_1, \dots, m_n$. Dann gilt für alle diese Körper i mit der Masse m_i in der Position r_i :

$$m_i \cdot \ddot{r}_i = \sum_{j \neq i} \vec{F}_{ij}$$

Die Summe aller Kräfte, die von anderen Körpern j auf i wirken, entspricht also dem Produkt der Masse von i und seiner Beschleunigung. Nach dem Newton'schen Gravitationsgesetz setzen wir für F_{ij} Folgendes ein:

$$\vec{F}_{ij} = G \cdot \frac{m_i \cdot m_j}{r_{ij}^2} \cdot \hat{r}_{ij}$$

Wenn man nun m_i kürzt, ergibt sich folgende Formel:

$$\ddot{\vec{r}}_i = G \cdot \sum_{j \neq i} \frac{m_j}{r_{ij}^2} \cdot \hat{r}_{ij}$$

In dieser Gleichung entspricht $G = 6.674 \cdot 10^{-11} m^3 / (kg \cdot s^2)$ der Gravitationskonstante, $\hat{r}_{ij}$ dem Einheitsvektor von i nach j und r_{ij} dem Abstand zwischen i und j .

2.2 Natürliche Einheiten und Dimensionen

Um Berechnungen übersichtlicher zu gestalten und aus technischen Gründen werden natürliche Einheiten verwendet. Das heißt, dass zuerst der Ausdruck $\ddot{\vec{r}}_i$ durch Division durch L dimensionslos gemacht wird und dann eine neue Längenskala $L, [L] = 1m$, Zeitskala $T, [T] = 1s$, und Massenskala $M, [M] = 1kg$, definiert wird.

Nach Division durch L erweitern wir im zweiten Schritt mit L^2 . Das liefert

$$\frac{\ddot{\vec{r}}_i}{L} = \frac{G}{L} \sum_{j \neq i} \frac{m_i}{r_{ij}^2} \cdot \hat{r}_{ij} = \frac{G}{L} \sum_{j \neq i} \frac{L^2}{L^2} \cdot \frac{m_i}{r_{ij}^2} \cdot \hat{r}_{ij} = \frac{G}{L^3} \sum_{j \neq i} \frac{m_i}{\left(\frac{r_{ij}}{L}\right)^2} \cdot \hat{r}_{ij}.$$

$(r_{ij}/L)^2$ ist nun offensichtlich dimensionslos. Daher wird nun mit M erweitert.

$$\frac{\ddot{\vec{r}}_i}{L} = \frac{G}{L^3} \sum_{j \neq i} \frac{M}{M} \cdot \frac{m_i}{\left(\frac{r_{ij}}{L}\right)^2} \cdot \hat{r}_{ij} = \frac{G \cdot M}{L^3} \sum_{j \neq i} \frac{\frac{m_i}{M}}{\left(\frac{r_{ij}}{L}\right)^2} \cdot \hat{r}_{ij}.$$

Auch m_i/M ist nun dimensionslos. Für den nächsten Schritt kann eine andere Schreibweise für Ableitungen verwendet werden. Es gilt $\ddot{\vec{r}}_i = d^2 \vec{r}_i / dt^2$. Eine Multiplikation mit T^2 führt zum gewünschten Ergebnis.

$$\frac{d^2}{d\left(\frac{t}{T}\right)^2} \left(\frac{\vec{r}_i}{L}\right) = \frac{T^2}{L} \left(\frac{d^2}{dt^2} \vec{r}_i\right) = \frac{T^2 \cdot \ddot{\vec{r}}_i}{L} = \frac{G \cdot M \cdot T^2}{L^3} \sum_{j \neq i} \frac{\frac{m_i}{M}}{\left(\frac{r_{ij}}{L}\right)^2} \cdot \hat{r}_{ij}.$$

Nun ist die gesamte linke Seite wegen t/T und $\vec{r}_i/L$ dimensionslos. Da dies auch für die Summe auf der rechten Seite gilt, muss auch $(G \cdot M \cdot T^2)/L^3$ dimensionslos sein. Tatsächlich gilt für die Einheit von G : $[G] = 1m^3/(kg \cdot s^2)$.

2.3 Definition der Skalen

Nachdem die Gleichung dimensionslos gemacht wurde, können Einheiten für die Längenskala, Zeitskala und Massenskala gewählt werden. Soll zum Beispiel das Sonnensystem modelliert werden, ist $L = \text{Abstand von Sonne und Erde} = 149597870700 \text{ m}$ und $M = \text{Erdmasse} = 5,97 \cdot 10^{24} \text{ kg}$ sinnvoll. Da zur Vereinfachung weiters $(G \cdot M \cdot T^2)/L^3 = 1$ gelten soll, kann hieraus T errechnet werden:

$$6,67408 \cdot 10^{-11} \cdot \frac{5,97 \cdot 10^{24} \cdot T^2}{149597870700^3} = 1$$

Daraus folgt $T = 2898713202 \text{ s}$ und $V = L/T = 51,61 \text{ m/s}$ folglich als natürliche Geschwindigkeitseinheit. Für unsere Planeten ergeben sich die folgenden Daten, wobei r den Abstand von der Sonne, m die Masse und v die Geschwindigkeit bezeichnet:

HK	r in m	r in NE	m in kg	m in NE	v in m/s	v in NE
Sonne	0	0	$1,9 \cdot 10^{30}$	332946	0	0
Merkur	$58 \cdot 10^9$	0,397	$3,3 \cdot 10^{23}$	0,055	47.870	927,25
Venus	$108 \cdot 10^9$	0,72	$4,87 \cdot 10^{24}$	0,816	35.020	678,34
Erde	$150 \cdot 10^9$	1	$5,97 \cdot 10^{24}$	1	29.780	576,84
Mars	$228 \cdot 10^9$	1,52	$6,42 \cdot 10^{24}$	0,108	24.130	467,40
Jupiter	$778 \cdot 10^9$	5,187	$1,9 \cdot 10^{27}$	318,258	13.070	253,17
Saturn	$1433 \cdot 10^9$	9,553	$5,69 \cdot 10^{26}$	95,31	9.690	187,70
Uranus	$2872 \cdot 10^9$	19,147	$8,68 \cdot 10^{25}$	14,54	6.810	131,91
Neptun	$49105 \cdot 10^9$	30,047	$1,02 \cdot 10^{26}$	17,085	5.430	105,18

3 Berechnung von Ort und Geschwindigkeit

Das vierstufige Runge-Kutta-Verfahren, kurz RK4, dient zur numerischen Lösung von Anfangswertproblemen, zu denen das Lösen von gewöhnlichen Differentialgleichungen gehört. Dieses Verfahren wurde in unserem Projekt verwendet, um Positionen und Geschwindigkeiten der verschiedenen Massen zu bestimmten Zeitpunkten zu ermitteln.

3.1 Differentialgleichungen

Da wir eine Differentialgleichung 2. Ordnung haben, das später erläuterte Runge-Kutta-Verfahren aber nur auf Differentialgleichungen 1. Ordnung angewendet werden kann, müssen wir diese erst mit einem Trick umformen.

Angenommen $\ddot{y} = f(y)$. Sei $\vec{z} := \begin{pmatrix} y \\ \dot{y} \end{pmatrix}$, dann folgt

$$\dot{\vec{z}} = \begin{pmatrix} \dot{y} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} \dot{y} \\ f(y) \end{pmatrix} =: \vec{F}(\vec{z}).$$

Somit haben wir eine Differentialgleichung 2. Ordnung in eine Differentialgleichung 1. Ordnung umgewandelt. Selbstverständlich lässt sich dieses Verfahren auch auf Differentialgleichungen höherer Ordnung anwenden.

Sei $y''''(t) = f(y)$. Dann gilt

$$\vec{z} = \begin{pmatrix} y \\ y' \\ y'' \\ y''' \end{pmatrix} \Rightarrow \vec{z}' = \begin{pmatrix} y' \\ y'' \\ y''' \\ y'''' \end{pmatrix} = \begin{pmatrix} y' \\ y'' \\ y''' \\ f(y) \end{pmatrix}.$$

3.2 Algorithmus

Um das Runge-Kutta-Verfahren anzuwenden werden eine erste Ableitung und ein Ausgangspunkt benötigt. Es sind also $\dot{\vec{z}} = \vec{F}(\vec{z})$ und $\vec{z}(0) = \vec{z}_0$ gegeben. Gesucht wird $\vec{z}(h), \vec{z}(2h), \dots, \vec{z}(nh)$. Es wird $\vec{z}(t)$ betrachtet, um eine Approximation für $\vec{z}(t+h)$ zu finden. Man setzt

$$\begin{aligned}
\vec{a}_1 &= \vec{F}(\vec{z}(t)) \\
\vec{a}_2 &= \vec{F}\left(\vec{z}(t) + \frac{h}{2}\vec{a}_1\right) \\
\vec{a}_3 &= \vec{F}\left(\vec{z}(t) + \frac{h}{2}\vec{a}_2\right) \\
\vec{a}_4 &= \vec{F}(\vec{z}(t) + h\vec{a}_3)
\end{aligned}$$

Dann gilt

$$\vec{z}(t+h) = \vec{z}(t) + \frac{h}{6}(\vec{a}_1 + 2\vec{a}_2 + 2\vec{a}_3 + \vec{a}_4) + O(h^5)$$

$O(h^5)$ ist ein Fehler, der beim Anwenden des Verfahrens entsteht. Dies bedeutet grob, dass bei einer Verdoppelung der Schrittweite der Fehler sich verzweihunddreißigfacht und umgekehrt. Eine etwas kleinere Schrittweite führt also zu einem viel genaueren Ergebnis.

Das Ergebnis wird in einer Matrix der Form

$$\begin{pmatrix} \vec{r}(0) & \vec{r}(h) & \dots & \vec{r}(nh) \\ \vec{v}(0) & \vec{v}(h) & \dots & \vec{v}(nh) \end{pmatrix}$$

gespeichert. $\vec{r}(t)$ gibt hier die Position aller Körper zum Zeitpunkt t an und $\vec{v}(t)$ gibt die betreffende Geschwindigkeit an:

$$\vec{r}(t) = \begin{pmatrix} \vec{r}_1(t) \\ \vdots \\ \vec{r}_m(t) \end{pmatrix}, \quad \vec{v}(t) = \begin{pmatrix} \vec{v}_1(t) \\ \vdots \\ \vec{v}_m(t) \end{pmatrix}.$$

Weiters setzen wir

$$\dot{\vec{z}}(t) := \begin{pmatrix} \vec{r}_1(t) \\ \vdots \\ \vec{r}_m(t) \\ \vec{v}_1(t) \\ \vdots \\ \vec{v}_m(t) \end{pmatrix}, \quad \dot{\vec{z}}(t) = \begin{pmatrix} \vec{v}_1(t) \\ \vdots \\ \vec{v}_m(t) \\ \sum \frac{m_j}{r_{1j}^2} \hat{r}_{1j} \\ \vdots \\ \sum \frac{m_j}{r_{mj}^2} \hat{r}_{mj} \end{pmatrix} = \vec{F}(\vec{z}).$$

3.3 Umsetzung

Dieser Algorithmus wurde in Folge in MATLAB implementiert.

Bevor eine neue Simulation gestartet werden kann, bedarf es zuerst einiger Variablen. Mit *dim* wird die Dimension festgelegt und mit *num* die Anzahl der Objekte bestimmt. *h* gibt, wie zuvor beschrieben, die Genauigkeit der Simulation an. In der Praxis hat sich ein Wert von 10^{-5} als idealer Kompromiss zwischen Präzision und Rechendauer ergeben. *n* bezeichnet die Anzahl der Iterationen. Danach wird das Runge-Kutta-Verfahren wie beschrieben umgesetzt.

```
1 z = zeros (2 * num * dim, n);
2
3 z(:, 1) = [r0, v0]';
4
5
6 for k = 2:n
7
8     a1 = f(z(:, k - 1), mm);
9     a2 = f(z(:, k - 1) + h/2 * a1, mm);
10    a3 = f(z(:, k - 1) + h/2 * a2, mm);
11    a4 = f(z(:, k - 1) + h * a3, mm);
12    z(:, k) = z(:, k - 1) + h/6 * (a1 + 2 * a2 + 2 * a3 + a4);
13
14 end
```

```
1 function vec = f(rv, mm)
2
3 global dim num;
4
5 vec = zeros(size(rv));
6
7 vec(1:dim*num) = rv(dim*num + 1:end);
8 vec(dim*num + 1:end) = a(rv(1:dim*num), mm);
9
10 end
```

```

1 function vec = a(pos, mm)
2
3 global dim num;
4
5 vec = zeros(size(pos));
6
7 for k = 1:num - 1
8     for l = k + 1:num
9         r = pos(dim * (l-1) + 1:dim * l) - ...
10            pos(dim * (k-1) + 1:dim * k);
11         aux = r / (norm(r)^3);
12
13         vec(dim * (k-1) + 1:dim * k) = ...
14            vec(dim * (k-1) + 1:dim * k) + mm(l) * aux;
15         vec(dim * (l-1) + 1:dim * l) ...
16            = vec(dim * (l-1) + 1:dim * l) - mm(k) * aux;
17
18     end
19 end

```

3.4 Impulserhaltung

Die zuvor beschriebenen Schritte reichen aus, um die Positionen und Geschwindigkeiten der Objekte zu berechnen. Allerdings wurde hier noch nicht der Gesamtimpuls des Systems bedacht. Dieser ist eine Erhaltungsgröße; das bedeutet, dass der Gesamtimpuls eines isolierten Systems sich nicht mit der Zeit ändert.

Da aber zur Vereinfachung alle Startgeschwindigkeiten in y-Richtung gewählt wurden, hat dies zur Folge, dass sich das gesamte System aufgrund der Impulserhaltung mit der Zeit auch in diese Richtung bewegt. Dies wird auch im letzten Abschnitt grafisch veranschaulicht. Dieser Impuls soll nun zu Beginn kompensiert werden um unsere Simulation zu vervollständigen.

Für den Anfangsimpuls gilt

$$\vec{P}_0 = \sum_{\text{Planet } i} m_i \cdot \vec{v}_i(0).$$

Es muss also ein $\vec{v}_*$ mit $\sum_i m_i (\vec{v}_i(0) + \vec{v}_*) = 0$ gefunden werden:

$$\vec{v}_* = -\frac{\sum_i m_i \vec{v}_i(0)}{\sum_i m_i}$$

Diese Berechnung ist allerdings nur für das n-Körper-Problem geeignet, durch die verschiedenen Besonderheiten der Lagrange-Punkte-Simulation ist es notwendig eine speziellere Formel zu entwickeln.

Hier gibt es nur drei Objekte, nämlich die Sonne mit Masse m_s , die sich zu Beginn nicht bewegt, die Erde mit Masse m_e und Anfangsgeschwindigkeit v_e und ein 3. Objekt, das masselos gewählt wurde. Da der Anfangsimpuls aus der Summe der Produkte von Masse und Geschwindigkeit der Objekte berechnet wird und im Fall von Sonne und dem 3. Objekt ebendieses Produkt 0 ist, beträgt der Gesamtimpuls $m_e \cdot v_e$. Die Impulskompensation an sich beschränkt sich auf die Sonne. Es wird also die Geschwindigkeit

$$v_* = -\frac{m_e \cdot v_e}{m_s}$$

zu der Geschwindigkeit der Sonne addiert.

3.5 Energieerhaltung

In einem geschlossenen System wie dem unseren muss die vorhandene Energie erhalten bleiben. Das besagt der Energieerhaltungssatz. Daher kann durch das Ermitteln der Gesamtenergie unser Ergebnis überprüft werden. In unserem Fall bedeutet dies, dass sich, aufgrund Mangel an anderen Energieformen, die potentielle und kinetische Energie ausgleichen müssen.

$$E_{kin} = \sum_i \frac{m_i \cdot \vec{v}_i^2}{2}$$

$$E_{pot} = - \sum_{i < j} \frac{m_i m_j}{r_{ij}}$$

Es werden in MATLAB also zwei Formeln implementiert, wobei die erste die kinetische Energie aller Objekte und die zweite die potentielle Energie aller Objekte summiert. Im Gegensatz zur kinetischen Energie müssen hier jedoch nicht alle Objekte für sich betrachtet werden, sondern die Beziehungen, in Form der wirkenden Gravitationskräfte, zwischen ihnen. Weiters ist anzumerken, dass die potentielle Energie, abgeleitet nach der Position des Teilchens i , die negative Kraft, welche auf dieses Teilchen i wirkt, ergibt.

```

1 function [ekin, epot] = energie(rr, vv, mm)
2
3 global dim num;
4
5 n = size(rr, 2);
6 ekin = zeros(1, n);
7 epot = zeros(1, n);
8
9 for h = 1:n
10
11     for k = 1:num
12         ekin(h) = ekin(h) + mm(k) * sum(vv(dim * (k-1) ...
13             + 1:dim * k, h).^2) / 2;
14
15         for l = k+1:num
16             epot(h) = epot(h) - mm(k)*mm(l) / ...
17                 norm(rr(dim * (k-1) + 1:dim * k, h) - ...
18                     rr(dim * (l-1) + 1:dim * l, h));
19         end
20     end
21 end
22
23 end

```

4 n-Körper-Problem

Beim n-Körper-Problem handelt es sich um das Problem der Bestimmung der Bahnen von n Körpern. Diese beeinflussen sich gegenseitig durch ihre Gravitationskräfte.

Eine Form des n-Körper-Problems ist das 3-Körper-Problem. Im Allgemeinen resultieren daraus chaotische Bahnen. Jedoch gibt es auch instabile Konstellationen, bei denen kein chaotisches Verhalten sichtbar ist, etwa bei drei Körpern in den Ecken eines gleichseitigen Dreiecks, die um ihren Massenschwerpunkt kreisen.

Auch unser Sonnensystem ist ein Beispiel für ein n-Körper-Problem. Hier haben wir 9 Körper, die alle um den gemeinsamen Massenschwerpunkt, der in der Nähe der Sonne liegt, kreisen. Verändert man gewisse Massen in dieser Konstellation, kann dies das komplette System destabilisieren. Würde man zum Beispiel den Jupiter um ein Vielfaches schwerer machen, würde das ganze Sonnensystem mit der Zeit aus dem Gleichgewicht geraten.

5 Lagrange-Punkte

5.1 Lokalisierung

Lagrange-Punkte sind Punkte, in denen die Zentrifugalkraft die Gravitationskräfte von Sonne und Erde genau ausgleicht. Sie können zwischen zwei Objekten, zum Beispiel Sonne und Erde lokalisiert werden.

Schon durch Überlegen lassen sich die fünf Lagrange-Punkte bestimmen. Am leichtesten zu finden ist der erste Lagrange-Punkt. Dieser muss nämlich zwischen dem Zentralgestirn und dem Planeten liegen, da dort alle Kraftvektoren auf einer Linie liegen, wobei die Zentrifugalkraft nach außen zeigt. Der zweite und dritte Lagrange-Punkt lassen sich ähnlich bestimmen, da auch deren Kraftvektoren auf einer Linie liegen. Beide befinden sich außerhalb der Umlaufbahn des Planeten.

Gedanklich schwieriger zu finden sind die letzten beiden Lagrange-Punkte. Diese bilden jeweils ein gleichseitiges Dreieck mit den beiden Körpern. Die Punkte 4 und 5 sind die stabilsten Lagrange-Punkte, da Objekte in der Nähe dieser Punkte bei passenden Geschwindigkeiten immer wieder durch die später erklärte Corioliskraft auf Bahnen um den jeweiligen Punkt zurückgelenkt werden.

Um die Punkte exakt zu berechnen, kann die Gleichung $\vec{F}_S + \vec{F}_E + \vec{F}_{ZF} = \vec{0}$ verwendet werden. Hier sollen die Lagrange-Punkte für Sonne und Erde ermittelt werden. Die Formel für die Zentrifugalkraft ist

$$\vec{F}_{ZF} = -mr_0G\frac{M_S}{r_{SE}^2r_{0E}}\hat{r}_0.$$

G ist hier die Gravitationskonstante, m die Masse eines Objekts, r_0 der Abstand zum Massenschwerpunkt, M_S die Masse der Sonne, r_{SE} der Abstand von Sonne und Erde, r_{0E} der Abstand der Erde vom Massenschwerpunkt und $\hat{r}_0$ der Richtungsvektor zu eben diesem. Es gilt daher

$$G\frac{M_Sm}{r_S^2}\hat{r}_s + G\frac{M_Em}{r_E^2}\hat{r}_E - mr_0G\frac{M_S}{r_{SE}^2r_{0E}}\hat{r}_0 = 0.$$

Befindet sich die Sonne in $(0,0)$ und setzt man $\mu := M_E/(M_E + M_S)$ sowie $q := \sqrt[3]{\mu/3}$ so befinden sich die Lagrange-Punkte in den folgenden Positionen:

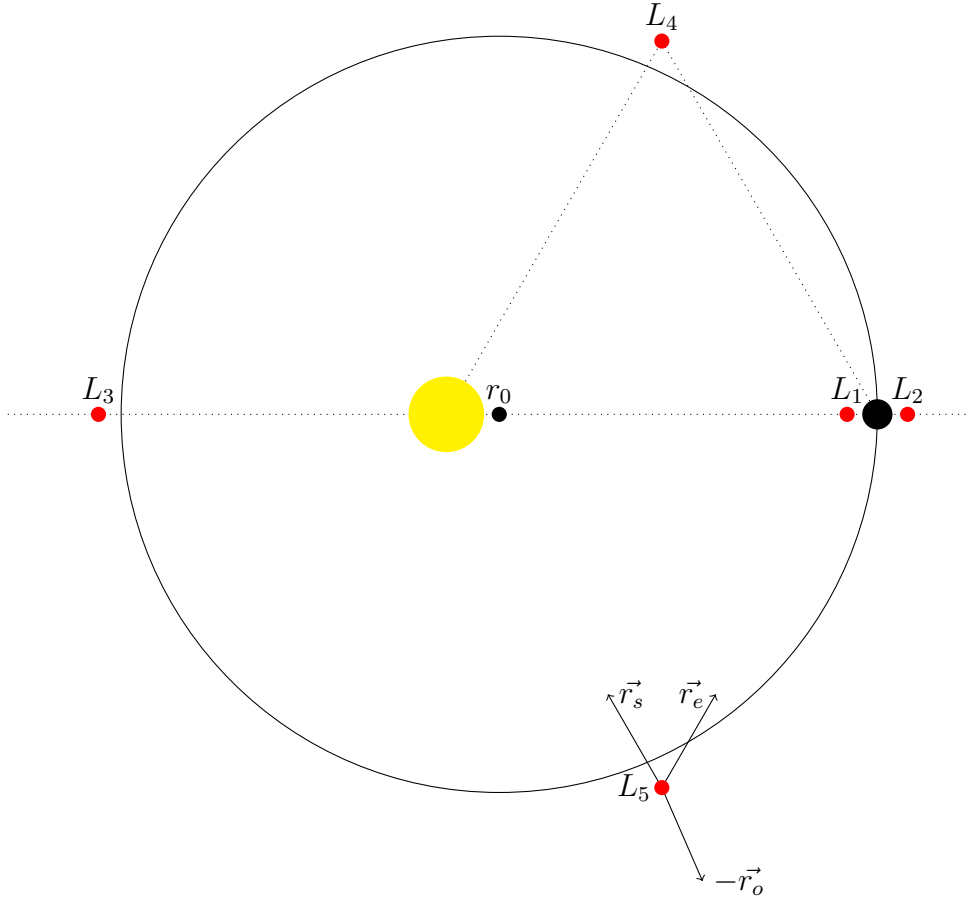
$$L_1: x \approx r_{SE} \left(1 - q + \frac{q^2}{3} + \frac{q^3}{9} \right), y = 0$$

$$L_2: x \approx r_{SE} \left(1 + q + \frac{q^2}{3} - \frac{q^3}{9} \right), y = 0$$

$$L_3: x \approx r_{SE} \left(-1 + \frac{7}{12}\mu \right), y = 0$$

$$L_4: x = \frac{r_{SE}}{2}, y = \frac{r_{SE}}{2} \cdot \sqrt{3}$$

$$L_5: x = \frac{r_{SE}}{2}, y = -\frac{r_{SE}}{2} \cdot \sqrt{3}$$



5.2 Gesamtpotential

Um die Position der Lagrange-Punkte grafisch zu visualisieren kann eine Darstellung des Gesamtpotentials herangezogen werden. Das Gesamtpotential gibt, wie der Name schon sagt, an, wie viel Kraft in welche Richtung auf jeden Punkt unserer Simulation wirkt. Dabei werden hier sowohl Zentrifugal-, als auch Gravitationskraft von Erde und Sonne betrachtet.

Man kann sich diese Darstellung wie eine reale Oberfläche vorstellen. Legt man eine Kugel an ein Gefälle so wird sie sich schnell von ihrer ursprünglichen Lage entfernen – eine Kraft dominiert über der anderen. Legt man sie jedoch an eine flache Stelle, so wird sie nur langsam davonrollen. Das heißt, dass sich die Kräfte in diesem Bereich nahezu aufheben.

Bewegt sie sich nicht, so hat man einen sogenannten kritischen Punkt gefunden, bei dem sich die Kräfte genau aufheben und damit der Gradient 0 ist. Dies sind die gesuchten Lagrange-Punkte. Überraschenderweise findet man diese jedoch allesamt an Hochpunkten oder Sattelpunkten. Man kann sich dieses filigrane Gleichgewicht wie das Balancieren eines Bleistiftes auf seiner Spitze vorstellen. Sie sind also sehr instabil. Lediglich die Bereiche um Lagrange-Punkte 4 und 5 sind flach genug um aufgrund der Coriolis-Kraft stabile Bahnen für sogenannte Trojaner um sie herum zuzulassen.

5.3 Corioliskraft

Ein weiteres Problem, das im Rahmen des Projektes untersucht wurde, ist die Bewegung von Objekten in der Nähe von Lagrange-Punkten. Eine weitere Kraft, die für das Problem von Bedeutung ist, ist die Corioliskraft. Sie ist eine Scheinkraft, die einen bewegten Körper quer zu seiner Bewegungsrichtung ablenkt. So wird beispielsweise eine Kugel auf einem Drehtisch, die gerade nach vorne gerollt wird, scheinbar nach rechts abgelenkt, wenn sich der Tisch mathematisch positiv dreht.

Die Formel für die Corioliskraft lautet

$$\vec{F}_{Cor} = -2m(\vec{\omega} \times \dot{\vec{r}})$$

Diese Kraft beeinflusst also die Bewegung von Objekten und so muss die Geschwindigkeit ebendieser verändert werden, sollen stabile Bahnen in der Nähe der Lagrange-Punkte 4 und 5 erzeugt werden.

Jupitertrojaner halten sich beispielsweise in der Nähe von L4 und L5 auf. Diese sind eine Gruppe von Asteroiden, die dieselbe Umlaufperiode wie Jupiter haben, ihm aber im Mittel 60° vor- oder nacheilen und dabei L4 und L5 in periodischen Bahnen umkreisen.

5.4 Matrizen

Soll die Umlaufbahn eines Objekts relativ zur Erde betrachtet werden, ist noch ein letzter Schritt notwendig um dies im Code zu implementieren. Mit jedem neuen Bild muss dieses so gedreht werden, dass sich die Erde wieder im selben Punkt befindet. Um dies zu erzielen, können Drehmatrizen verwendet werden.

Eine Drehung eines Vektors wird durch Multiplikation einer Matrix mit diesem Vektor erzeugt. Die Matrix ist von der Form

$$\begin{pmatrix} a & -b \\ b & a \end{pmatrix}$$

wobei $\sqrt{a^2 + b^2} = 1$ gilt.

Im Prinzip funktioniert die Drehung folgendermaßen. Angenommen die Erde befindet sich im Punkt (x, y) . Nach der Drehung soll sie sich wieder im Punkt $(1, 0)$ befinden. Die Sonne soll in $(0, 0)$ bleiben. Da die Erde sich (in dieser vereinfachten Beschreibung) auf einer Bahn mit Radius 1 um die Sonne bewegt, gilt also $\sqrt{x^2 + y^2} = 1$ und daher weiters $x^2 + y^2 = 1$. Das Produkt

$$\begin{pmatrix} x & -y \\ y & x \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x^2 + y^2 \\ -xy + xy \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

erfüllt daher diese Voraussetzungen. Tatsächlich ist die Rechnung etwas komplizierter, da die Rotation nicht um die Sonne, sondern um den Schwerpunkt erfolgt.

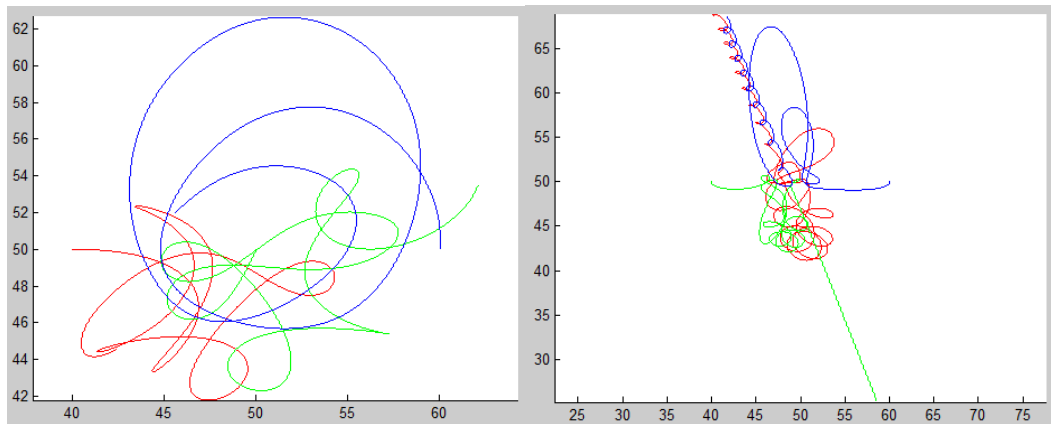


Abbildung 1: 3-Körper-Problem

6 Ergebnisse

6.1 3-Körper-Problem

Abbildung 1 zeigt die Bahnen dreier Körper mit gleichen Massen, aber unterschiedlichen Startpositionen und Geschwindigkeiten. Dabei laufen die Körper zumeist auf chaotischen Bahnen.

6.2 Sonnensystem

Abbildung 2 zeigt eine Simulation unseres Sonnensystems im Zeitraum von 92 Jahren. Es wurden die zu Beginn aufgelisteten Massen, Geschwindigkeiten und Abstände verwendet und das Runge-Kutta-Verfahren eingesetzt. Die Bahn der Erde ist blau eingezeichnet. Zusätzlich wurde links der Anfangsimpuls korrigiert. Die rechte Abbildung zeigt den Einfluss des hier fehlenden Gegenimpulses. Man erkennt im Laufe der Zeit eine Verschiebung der Bahnen in Richtung der y-Achse. Dies ist ein Resultat des nicht verschwindenden Anfangsimpulses des Gesamtsystems in Richtung der y-Achse.

Die in Abbildung 3 rot eingezeichnete Linie entspricht der kinetischen Energie, wohingegen die blaue Linie die potentielle Energie darstellt. Addiert man diese, erhält man die Gesamtenergie, hier grün dargestellt. Aufgrund der Energieerhaltung ist die Gesamtenergie konstant. Durch numerische Fehler variiert die Gesamtenergie jedoch geringfügig.

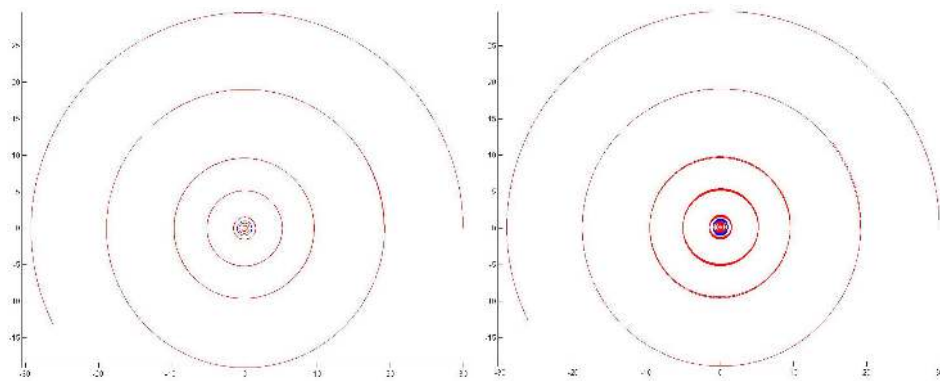


Abbildung 2: Simulation unseres Sonnensystems

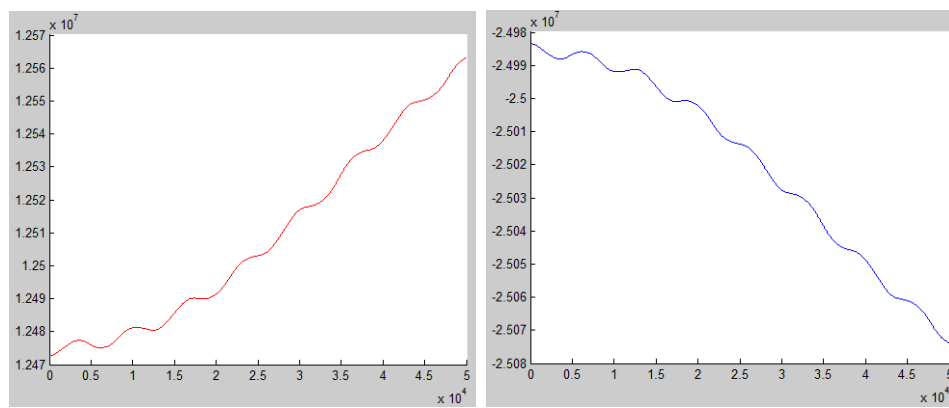


Abbildung 3: Kinetische und potentielle Energie

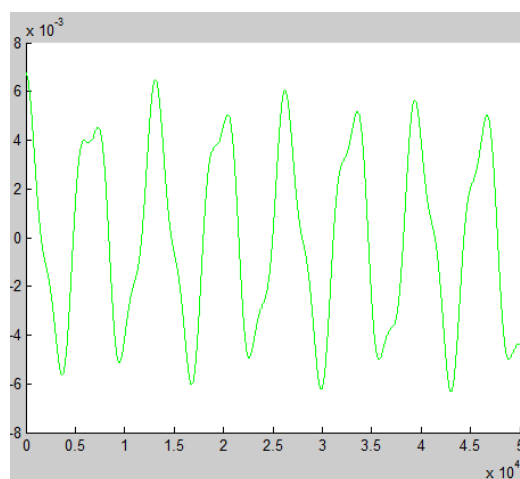


Abbildung 4: Gesamtenergie

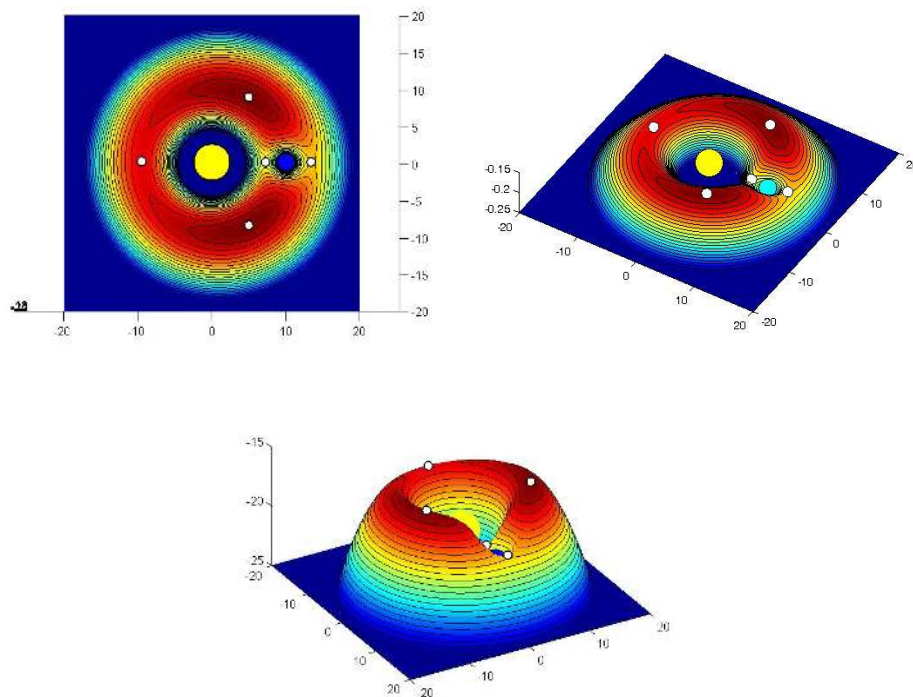


Abbildung 5: Potential und Lagrange-Punkte

6.3 Lagrange-Punkte

Abbildung 5 zeigt den Einfluss der Massen auf das Gesamtpotential und die Lagrange-Punkte. In der Grafik kann man erkennen, dass die Lagrange-Punkte genau auf den kritischen Punkten der Potentialfläche liegen.

In den folgenden Abbildungen wird veranschaulicht, wie sich Körper, die sich in der Nähe der Lagrange-Punkte befinden aus Perspektive der Erde bewegen. Das heißt, das System wurde so gedreht, dass die Erde still steht.

Das linke Bild in Abbildung 6 zeigt einen Körper, der gerinfügig vom 1. Lagrange-Punkt in Richtung Erde verschoben wurde. Aufgrund der Instabilität des 1. Lagrange-Punktes, die auch in der Abbildung des Gesamtpotentials erkennbar ist, genügt schon eine geringe Abweichung in Richtung der Erde, um die Flugbahn des Körpers zu ändern. Dieser nähert sich der Erde; da in unserer Simulation jedoch keine Körper kollidieren können, umkreist er ungehindert die Erde. Rechts wird ein ähnliches Verhalten in der Nähe des 2. Lagrange-Punktes gezeigt.

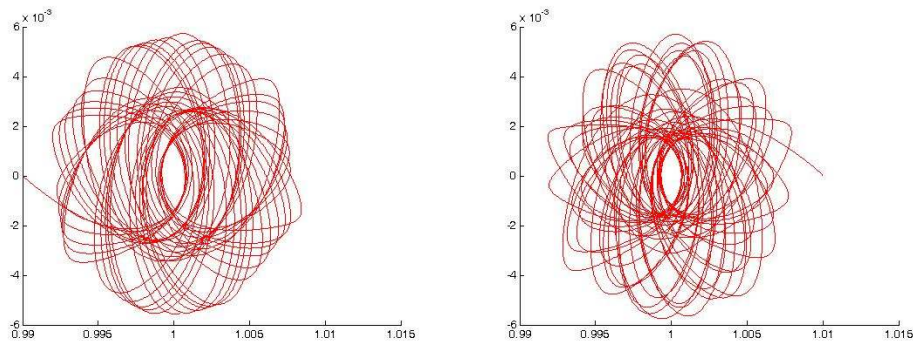


Abbildung 6: Verhalten in der Nähe von L1 und L2 nach Bewegung in Richtung der Erde aus Sicht der Erde

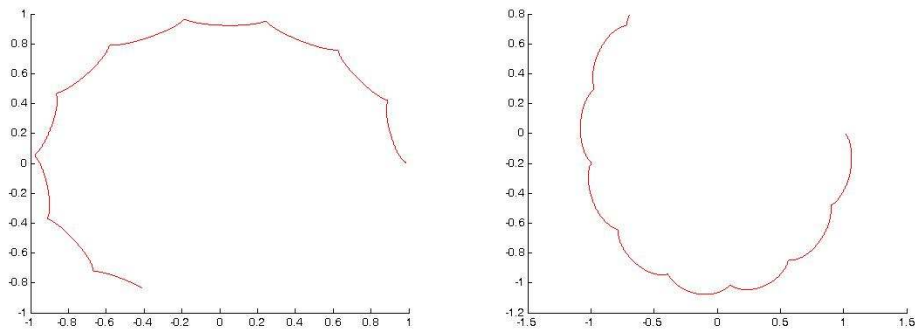


Abbildung 7: Verhalten in der Nähe von L1 und L2 nach Bewegung in bzw. entgegen der Richtung der Sonne aus Sicht der Erde

Auch in Abbildung 7 befindet sich das Objekt nicht genau im Lagrange-Punkt, sondern wurde geringfügig von der Erde weg verschoben, wodurch es eine Rosettenbahn beschreibt. Links wird das Verhalten in der Nähe von L1 gezeigt, rechts in der Nähe von L2.

Abbildung 8 zeigt das Verhalten in der Nähe von L3. Links ergibt sich durch das Versetzen in Richtung der negativen x-Achse eine instabile Bahn von der Sonne weg, während rechts nur minimale Abweichungen erkennbar sind. Dies liegt daran, dass sich das Objekt direkt in L3 befindet. Durch Änderung der Entfernung und Geschwindigkeit können verschiedenste Bahnen erzielt werden, wie zum Beispiel die mandelförmige Bahn unten.

Abbildung 9 veranschaulicht das Verhalten um L4. Bei der Positionierung eines Objektes in L4 sollte die Bahn des Objektes relativ zur Erde einen

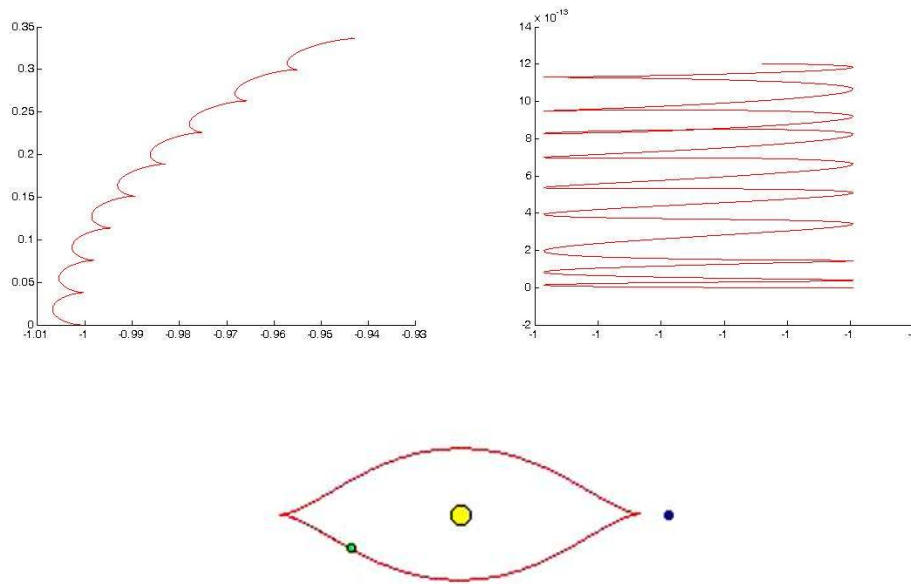


Abbildung 8: Verhalten in der Nähe von L3

Punkt darstellen. Durch numerische Fehler ergibt sich jedoch eine Linie. Verschiebt man das Objekt in y-Richtung und passt die Geschwindigkeit an, ergibt sich eine bohnenförmige Umlaufbahn aus Sicht der Erde, die einer ellipsenförmigen Umlaufbahn um die Sonne entspricht. Eine weitere mögliche Umlaufbahn wird unten rechts gezeigt. Ähnliche Bahnen können in der Nähe von L5 erzielt werden.

Abbildung 10 zeigt eine hufeisenförmige Bahn relativ zur Erde von L4 aus. Das Objekt wurde um 1.5 Millionen km in die negative y-Richtung verschoben und die Startgeschwindigkeit um 723 m/s in Richtung der negativen x-Achse erhöht.

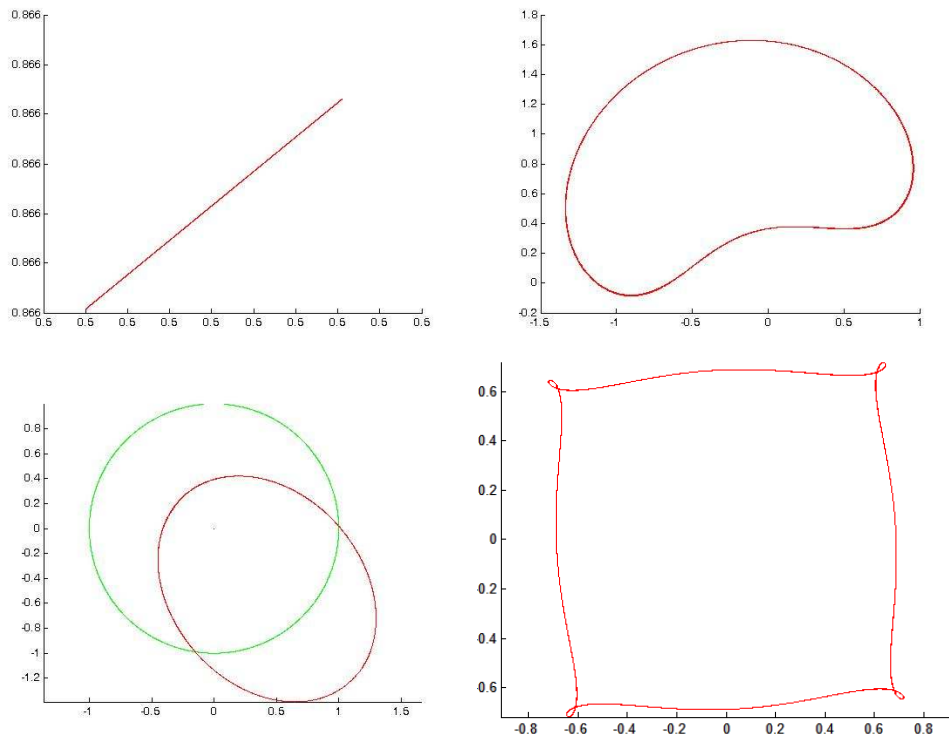


Abbildung 9: Verhalten in der Nähe von L4

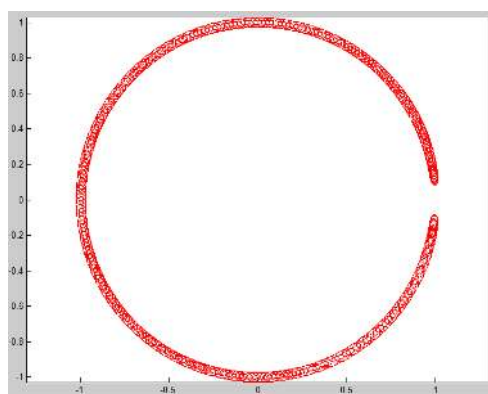


Abbildung 10: Hufeisenförmige Bahn

Informatik

Wie funktionieren Empfehlungsdienste?

Luca Cerroni, Sebastian Halbwirth, Elisabeth Höldrich,

Charlotte Meyer, Sophie Reiter, Simon Vasold

```
00010100 0011011 00001101001101101011 1100100101010000010100 1100100110111010100 11011001 10010000 10011110
0100110110 01101100 00000000110110011010 010101011111101101001111 0110100110100011001111 101010111 011001110 101100110
11010010010 00111101 10001000100111011100 000001111111001101111100 00101100000001111110111 110101011 011101100 11000001
0100000100111 01000101 00000100 0100011111001100 01110000 101011101 100001011 100010000100111101 10001000
1101100110010 01110000 1001000101010000010 100100101 00110101 011101111001 110101000 10011001100101 1001010000
110111011011010 01011111 1110001101110111010 010010110 101101110 10110111010001100101 000100100 00010100000
10101111 0010111001101110 0100010001011000010 00110110 110001000010110001 01000110 0110100111110
01011001 11110011110101 10001010 100010100 01110011 1110001000 011110111 01110111101010111
00100011 001110110100 101111100111111000101 00010010 01000010 110010011 00010101 010011111 001010100
01100111 0111000010 000100110001010100001 01101000 01010110 110111010 001010001 010000100 101010100
00111010 010011010 01011100001111000110 11101010 01110011 111011001 10111001 010001011 100111101
01110110 0110001 100010110 11101100 00000101 00011100 11101100 00110110 100110011
```

Betreuer: Dipl.-Math.techn. Dr. Tobias Breiten

Inhalt

Problemstellung	3
Ziele des Projekts.....	3
1.0 Matrix	4
1.1 Aufstellen eines Wahrscheinlichkeitsvektors.....	5
1.2 Behebung eines „Dangling Node“	7
1.3 Suchleiste.....	8
1.4 Personalisierte Wahrscheinlichkeitsvektoren	10
2.0 Genauere Definitionen	12
2.1 Eigenwertproblem.....	15
2.2 Iteration	15
3.0 Filmempfehlungen	16
3.1 Erstellung der benutzer_film_base_matrix.....	16
3.2 Erstellung der benutzer_film_verbindungs_matrix	16
3.3 Erstellung der genre_film_verbindungs_matrix.....	17
3.4 Ergebnis	17
4.0 Bewerten unserer Empfehlungen	19
5.0 Zusammenfassung.....	19

Problemstellung

In den letzten Jahren, in denen sich das Internet immer weiter vergrößert hat, wurde es immer wichtiger Möglichkeiten zu finden es nach seiner Wichtigkeit zu sortieren. Dazu wird zum Beispiel das Pagerank-System verwendet, welches Google zu seiner Größe verhalf. Dieses ist nun in veränderter Form auch verwendbar andere Dinge außer Webseiten zu sortieren, in unserem Fall Filme. Aber wie funktioniert es, dass man jedem beliebigen Nutzer eine auf ihn zugeschnittene Empfehlung geben kann, welche Webseite er suchen könnte, beziehungsweise welcher Film ihm gefallen könnte.

Ziele des Projekts

- Rechnen mit Matrizen und Vektoren
- Aufstellen eines Wahrscheinlichkeitsvektors, welcher die Wichtigkeiten eines von uns virtuell aufgestellten Netzwerkes beschreibt
- Diesen Wahrscheinlichkeitsvektor auf beliebige Nutzer zuschneiden
- Verschieden Schwierigkeiten wie zum Beispiel, ein „Dangling Node“ zu beheben
- Filmempfehlungen aufgrund von Matrizen aufstellen
- Genauigkeit der Filmempfehlungen optimieren
- Verständnis des genauen Hintergrunds der Pagerank-Rechnungen

1.0 Matrix

Einführung

Das Umgehen mit Matrizen war ein wesentlicher Teil unseres Projekts, da wir schon anfangs mit Matrizen rechnen mussten. Sie sind besonders hilfreich, wenn man mehrere Informationen einander zuordnen muss. Matrizen kann man sich als mehrspaltige Vektoren vorstellen. Mit Matrizen haben wir addiert, subtrahiert und multipliziert.

Wenn $A = \begin{pmatrix} a1 & a2 \\ a3 & a4 \end{pmatrix}$ dann sind $\begin{pmatrix} a1 \\ a3 \end{pmatrix}$ und $\begin{pmatrix} a2 \\ a4 \end{pmatrix}$ die Spalten und $(a1 \ a2)$ und $(a3 \ a4)$ die Zeilen.

Addition und Subtraktion

Matrizen kann man addieren, solange sie vom gleichen Typ sind, also gleich viele Zeilen und Spalten haben. Es funktioniert also so, dass man den ersten Wert der Matrix (1te Zeile, 1te Spalte) mit dem ersten Wert der zweiten Matrix (1te Zeile, 1te Spalte) addiert. Es müssen also immer Zeile und Spalte übereinstimmen. Nimmt man also Matrix A und Matrix B, welche die gleichen Zeilen- und Spaltenanzahlen haben, so bekommt man durch $A+B$ das Ergebnis C, somit $A+B=C$. Auch bei Subtraktionen findet eine Subtraktion der einzelnen Elemente statt, somit gilt $D=E-F$.

$$\text{Also } \begin{bmatrix} a1 & a2 \\ a3 & a4 \end{bmatrix} + \begin{bmatrix} a5 & a6 \\ a7 & a8 \end{bmatrix} = \begin{bmatrix} a1 + a5 & a2 + a6 \\ a3 + a7 & a4 + a8 \end{bmatrix}$$

Multiplikation

Die Multiplikation von Matrizen ist nicht kommutativ, was bedeutet, dass die beiden Faktoren nicht vertauscht werden dürfen. Also ist $A*B \neq B*A$. Bei einer Matrix-Vektor-Multiplikation muss die Spaltenzahl der Matrix gleich der Zahl der Komponenten des Vektors sein, damit das Ergebnis ein Vektor mit der Anzahl der Spalten der Matrix ist.

$$A * B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

Skalarmultiplikation

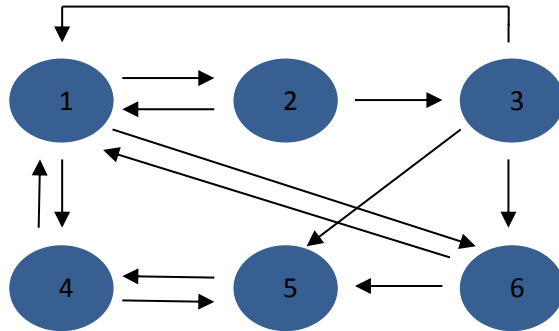
Bei einer Skalarmultiplikation, multipliziert man die Matrix mit einem Skalar, sodass jeder Wert in der Matrix mit dem Skalar multipliziert wird. Es gilt also:

$$y * A = y * \begin{bmatrix} a1 & a2 \\ a3 & a4 \end{bmatrix} = \begin{bmatrix} y * a1 & y * a3 \\ y * a2 & y * a4 \end{bmatrix}$$

1.1 Aufstellen eines Wahrscheinlichkeitsvektors

Als erster Schritt muss ein „Mini-Internet“ erstellt werden, da man die Anzahl der Webseiten auf einige wenige beschränken muss. Als nächstes beginnt man, Verbindungen zwischen den verschiedenen Webseiten aufzustellen, welche die Links darstellen. So kommt man in unserem virtuellen Netzwerk von einer Seite zur nächsten. Erst später wird die Funktion der Suchleiste hinzugefügt. Jetzt wo man dieses Netzwerk aufgestellt hat, kann man mit dem ersten „rechnerischen“ Schritt beginnen. Man stellt eine Matrix auf, die durch Einsen und Nullen die Verbindungen zwischen den einzelnen Seiten beschreibt und so genau die Dimensionen unseres Netzwerkes widerspiegelt. Als nächsten Schritt, kann man die Einsen durch verschiedene Brüche ersetzen, welche die Wichtigkeit der jeweiligen Seite zeigen. Als Beispiel könnte man zeigen, dass eine Webseite, von der 3 Links weggehen, den anderen Seiten auf die sie zeigt $\frac{1}{3}$ ihrer Wichtigkeit gibt. Eine Webseite mit 2 ausgehenden Links gibt jedem Empfänger des Links $\frac{1}{2}$ (also die Hälfte) ihrer Wichtigkeit. Durch die Gleichung $x=A*x$, in der A die Matrix mit unserem Netzwerk und x unseren Einheitsvektor darstellt, können wir jetzt erstmals unser Netzwerk nach seiner Wichtigkeit sortieren. Die Webseiten werden nach ihrer Wahrscheinlichkeit gerankt, welche auch anzeigt, wo sich ein zufälliger Surfer, welcher sich nur durch das Klicken auf Links durch unser Netzwerk bewegt, am wahrscheinlichsten aufhält. Als Ergebnis dieser Gleichung bekommen wir einen Vektor, eben diesen Wahrscheinlichkeitsvektor, welcher uns in jeder Zeile i, die jeweilige Wichtigkeit unserer Seite angibt. Interessant zu erwähnen ist, dass egal wo man in unserem Netzwerk startet, vorausgesetzt man startet nicht auf einer Seite ohne Verbindungen, sich die Wahrscheinlichkeiten für die jeweiligen Seiten, nach einer gewissen Anzahl von Klicks nicht mehr verändert. Das heißt, egal wo wir in unserem Netzwerk starten, nachdem wir eine gewisse Anzahl von Klicks erreicht haben, sind wir immer am wahrscheinlichsten auf unserer

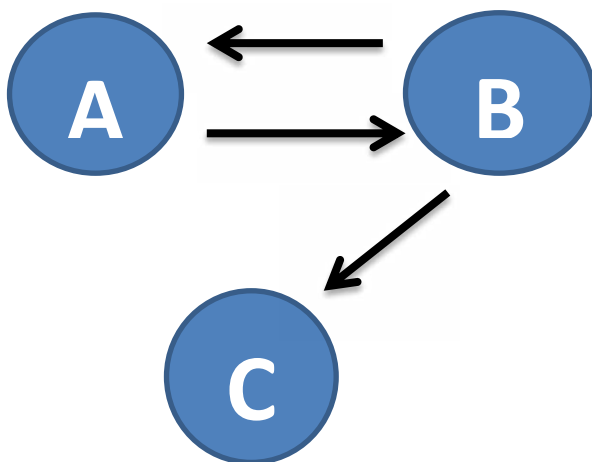
Seite S, welche die höchste Wahrscheinlichkeit aufweist. Hier unten angegeben, erkennt man ein von uns aufgestelltes Netzwerk, und daneben die zugehörige Matrix. Bei genauer Analyse kann man die gerade erklärten Zusammenhänge erkennen. Aus dieser Matrix können wir nun einen Wahrscheinlichkeitsvektor kreieren.



$$\begin{bmatrix} 0 & 1/2 & 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 0 & 1/3 & 0 & 0 & 0 \end{bmatrix}$$

1.2 Behebung eines „Dangling Node“

Am Anfang erwiesen sich auch Schwierigkeiten mit „Dangling Nodes“, da wir noch nicht die Benutzung der Suchleiste berücksichtigt hatten. Dieses Wort bedeutet in der Pageranking-Fachsprache, dass es sich um eine Webseite handelt, welche nur Verbindungen von anderen Webseiten bekommt, aber keine Verbindungen an andere Webseiten gibt. Bei einer „Dangling Node“ verbindet die Webseite den Benutzer nicht mit einer weiteren Webseite, was heißt, dass ein Zufallsserver bei dieser „Dangling Node“ Webseite landen und nicht zu einer weiteren Webseite weiterklicken kann. Daher kommen nach jeder Iteration immer mehr Benutzer zu dieser „Dangling Node“, wo sie feststecken und zu keiner weiteren Webseite kommen. Dies führt dazu, dass sich der Wichtigkeitsvektor nach einer bestimmten Anzahl an Klicks nicht mehr auf 1 aufsummiert, sondern die Wahrscheinlichkeiten, das man auf anderen Links landet zu 0 konvergieren. Damit nach einer bestimmten Anzahl an Klicks gewisse Werte in dem Vektor von x bei $x=A*x$ nicht zu 0 konvergieren, muss die Summe der Spalte 1 ergeben und das nach jeder Iteration, damit keine „Energie“ im Prozess verloren geht und die Wahrscheinlichkeit so genau wie möglich bleibt. Wir haben in unseren Code deshalb $x=x/\text{sum}(x)$ eingefügt. Das bezweckt, dass x immer durch die Summe von ihrer Spalte dividiert wird, sich der Wahrscheinlichkeitsvektor immer selbst zu 1 addiert und keine Energie aus dem System verloren geht. Um aber wirklich keine Energie während jeder Durchführung der Rechnung zu verlieren, gibt man diesen Code $x=x/\text{sum}(x)$ in die for-Schleife, damit dieser Schritt bei jeder Iteration durchgeführt wird.

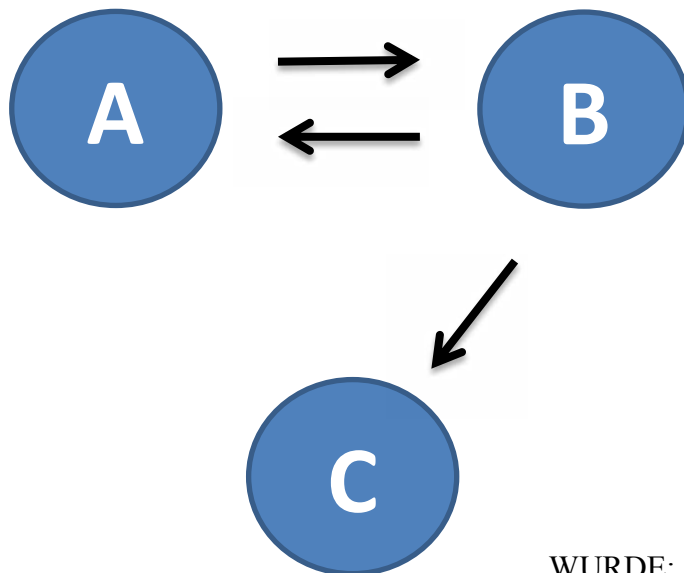


Von C geht kein Link weg, das heißt, dass alle Benutzer die auf C landen nicht auf eine andere Webseite weiter kommen können.

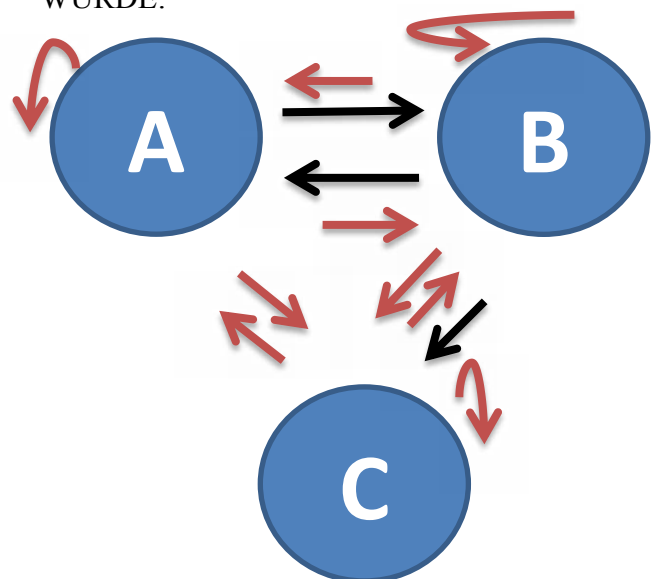
1.3 Suchleiste

Anfangs haben wir nur gewisse Webseiten mit anderen Webseiten verbunden und haben erst später die Suchleiste in Betracht gezogen mit der man von jeder Webseite zu allen anderen kommt, da wir durch das Problem mit dem „Dangling Node“ eine Lösung brauchten, damit man auch von einer Webseite die nicht zu anderen Webseiten verlinkt ist, zu weiteren kommt. Die Suchleiste half, da man sich mit ihr von einer Webseite zu allen anderen in unserem Netzwerk verlinken konnte, auch zu sich selbst.

AUS:



WURDE:



Allerdings haben wir zwei verschiedene Matrizen aufgestellt, einmal eine für die roten Pfeile und einmal eine für die schwarzen Pfeile.

Die Matrix der schwarzen Pfeile: $X = \begin{bmatrix} 0 & 1/2 & 0 \\ 1 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix}$

Die Matrix der roten Pfeile: $Y = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$

Die Matrix Y besteht nur aus den Verbindungen der Suchleiste. Man kreiert zwei Matrizen, da man sie für das Pageranking verschieden gewichtet. Die Endmatrix Z besteht aus der Summe der gewichteten Matrizen, also:

$$Z = \alpha * X + (1 - \alpha) * Y$$

Die Suchleiste wurde bei uns mit 15% Wichtigkeit gewichtet, daher wählten wir für das alpha 0.85. Diese Gewichtung basiert auf der von Google. Somit hatten die Verbindungen zwischen den Webseiten eine größere Wichtigkeit, als wenn man zwischen Webseiten mit der Suchleiste des Webbrowsers wechselt. Lässt man nun die Matrix Z iterieren und lässt $x = Z * x$, so bekommt man den Vektor, welcher das Ranking der Webseiten angibt.

1.4 Personalisierte Wahrscheinlichkeitsvektoren

In unserem nächsten Schritt haben wir uns vorgenommen, unser Pagerank nicht mehr für den Zufallssurfer zu erstellen, sondern auf realistische Interessen des Nutzers einzugehen. Das heißt, zusätzlich zur Matrix welche die Verbindungen angibt, erstellen wir eine zweite Matrix, welche auf die Interessen des Benutzers eingeht. Einfach gesagt heißt das, dass unsere „benutzerdefinierte“ Matrix den Seiten aus unserem Netzwerk, welche zu den Interessen unseres Benutzers passen, höhere Wahrscheinlichkeiten gibt. Diese müssen sich natürlich wieder zu 1 summieren, da nie eine Wahrscheinlichkeit von über 100% erreichen kann. Nun kann man mit der Formel

$$C = \alpha * A + (1 - \alpha) * B$$

C berechnen, wobei C alle benutzerdefinierten Wahrscheinlichkeiten für unser Netzwerk nebeneinander in einer Matrix anzeigt. Diese Formel wird auch die „Google-Matrix“ genannt, da sie jeden Monat von Google aufgestellt wird, um das Pageranking System zu aktualisieren, da ja unser Internet stetig anwächst. Angehängt ist hier ein Programm, auf MATLAB, geschrieben, welches eine Matrix A mit dem zur Matrix gemachten Vektor B multipliziert. Hier wird Googles Pageranksystem verwendet, was man an der Formel $C = \alpha * A + (1 - \alpha) * B$ sehen kann. Die Matrix B stellt in diesem Fall unsere „personalisierte“ Wahrscheinlichkeitsmatrix dar, welche die Interessen des Nutzers nun in unser Pageranking einfließen lässt.

```

A=A
for i=1:20
    %i-te Spalte
    A(:,i)=1/sum(A(:,i))*A(:,i);
end

B=[4/20;1/20;2/20;1/40;1/20;1/40;0;1/40;0;1/40;0;1/40;3/20;2/20;2/20;1/40;0;1/40;1/20;1/40]

B=B*ones(1,20);

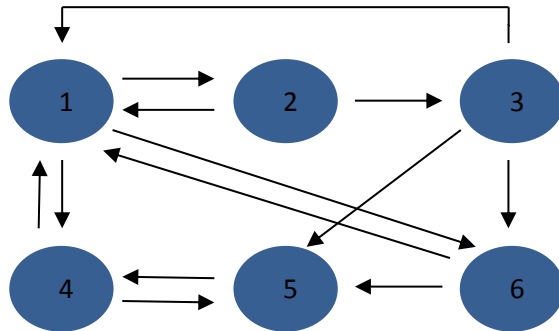
alpha=0.85;
%B=1/20*ones(20,20);
C=alpha*A+(1-alpha)*B;
x=1/20*ones(20,1);

for i=1:1000
    x=C*x;
    x=x/sum(x);
end
x

```

2.0 Genauere Definitionen

Um genauer auf den mathematischen Hintergrund unserer Rechnungen einzugehen, müssen wir uns zuerst unser erstes „Mini-Internet“ noch einmal ansehen:



Wir haben also verschiedene Seiten erstellt die untereinander mit Links verbunden sind. Diese Links können zu einer Webseite hin- oder wegführen. In unserer Matrix kann man sich zum Beispiel die in Spalte 1 stehenden Werte als die von Webseite 1 auf die anderen fünf Seiten verteilten Wichtigkeiten vorstellen. Seite 1 „gibt“ also den Seiten 2, 4 und 6 seine Wichtigkeit. Zeile 1 gibt somit die Wichtigkeiten von Webseite 1 an. Damit sich die Spalten auf eins aufsummieren, wurden alle Werte skaliert. Beim Skalieren wird jeder Wert einer Spalte durch die Summe der jeweiligen Spalte dividiert.

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 3 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1/2 & 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 0 & 1/3 & 0 & 0 & 0 \end{bmatrix}$$

(zum Beispiel: $(1,2) = 1/3$, $(4,5) = 1/2$)

Die in Zeile 1 stehenden Werte geben folgend die Wichtigkeiten an, die Seite 1 von den anderen Seiten zugewiesen bekommen hat. Um die tatsächlichen Wichtigkeiten der einzelnen Seiten zu errechnen, stellten wir auf der Matrix basierende Formeln auf:

$$x_1 = 1/2 \cdot x_2 + 1/3 \cdot x_3 + 1/2 \cdot x_4 + 1/2 \cdot x_6$$

$$x_2 = 1/3 \cdot x_1$$

$$x_3 = 1/2 \cdot x_2$$

$$x_4 = 1/3 \cdot x_1$$

$$x_5 = 1/3 \cdot x_3 + 1/2 \cdot x_4 + 1/2 \cdot x_6$$

$$x_6 = 1/3 \cdot x_1 + 1/3 \cdot x_3$$

Um die Rechnungen fortzuführen, drückten x_2 bis x_6 durch x_1 aus und entwickelten daraus diese Gleichung:

$$x_1 = x_1/6 + x_1/18 + x_1/6 + x_1/18 + x_1/6 + x_1/6 + x_1/36 + x_1/6 + x_1/36 =$$

$$(6 \cdot x_1 + 2 \cdot x_1 + 6 \cdot x_1 + 2 \cdot x_1 + 6 \cdot x_1 + 6 \cdot x_1 + x_1 + 6 \cdot x_1 + x_1)/36 = (36 \cdot x_1)/36 = x_1$$

Daraus folgt also $x_1 = x_1$, was bedeutet, dass wir jeden beliebigen Wert für x_1 wählen können.

Mit dieser Erkenntnis wählten wir als erstes für $x_1 = 1$:

$$x_1 = 1$$

$$x_2 = 1/3$$

$$x_3 = 1/6$$

$$x_4 = 1/3$$

$$x_5 = 1/2$$

$$x_6 = 2/3$$

Damit die Summe des Vektors 1 ergibt, skalierten wir alle Werte.

$$x_1 = 0,3333$$

$$x_2 = 0,1111$$

$$x_3 = 0,0556$$

$$x_4 = 0,1111$$

$$x_5 = 0,1667$$

$$x_6 = 0,2222$$

Der Vektor ergibt sich also so:

$$x = A * x$$

$$\begin{bmatrix} 0,3333 \\ 0,1111 \\ 0,0556 \\ 0,1111 \\ 0,1667 \\ 0,2222 \end{bmatrix} = \begin{bmatrix} 0 & 1/2 & 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 0 & 1/3 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0,3333 \\ 0,1111 \\ 0,0556 \\ 0,1111 \\ 0,1667 \\ 0,2222 \end{bmatrix}$$

2.1 Eigenwertproblem

Dies ist eine spezielle Form des Eigenwertproblems, bei dem der Eigenwert $\lambda = 1$ ist, womit wir zu unserer nächsten Definition kommen:

Sei A eine quadratische Matrix ($m \times m$). Die Aufgabe, eine Zahl λ und einen dazugehörigen Vektor $x \neq 0$ zu finden, damit

$$A * x = \lambda * x$$

nennt man Eigenwertproblem.

- Die Zahl λ heißt Eigenwert, wobei λ eine komplexe Zahl ist.
- Der Vektor x heißt Eigenvektor, wobei auch $c * x$ (c ist eine beliebige reelle Zahl $\neq 0$) ein Eigenvektor ist. x darf nicht der Nullvektor sein.

2.2 Iteration

Die Iteration, also in MATLAB der Befehl: `for i=1:n`, heißt das die Variable i von 1 bis n durchläuft. Für eine Matrix, A , wie wir sie haben und einem Anfangsvektor x konvergiert die Iteration zu einem konstanten Vektor x .

Das heißt zum Schluss haben wir $A * x = x$. Um dies zu verstehen muss man wissen, dass es zu einer 2×2 Matrix zwei Eigenwerte λ_1 und λ_2 mit zwei Eigenvektoren x und y gibt, sodass $A * x = \lambda_1 * x$ beziehungsweise $A * y = \lambda_2 * y$ gilt. Wenn wir einen Startvektor x_0 haben können wir ihn schreiben als $x_0 = a * x + b * y$. Damit gilt nach i Schritten:

$$A^i * x_0 = a * x * \lambda_1^i + b * y * \lambda_2^i$$

Weil $\lambda_1 = 1$ und $\lambda_2 < 1$, bleibt $a * x * \lambda_1$ während der Iteration gleich und $b * y * \lambda_2^i$ gegen 0 läuft. Deshalb lautet die Formel jetzt:

$$A^i * x_0 \approx a * x * 1 + 0 \Rightarrow A * (A^i * x_0) \approx a * A * x$$

$$A(a * x) \approx a * \lambda_1 * x \quad a * x = x'$$

$$A(a * x) \approx a * x$$

$$A * x' \approx x'$$

Hiermit wären wir wieder bei der anfänglichen Aussage $A*x=x$.

3.0 Filmempfehlungen

Das Ziel unseres Projektes ist aufgrund von Filmbewertungen und gesehenen Genres eines Nutzers von einer Filmplattform, weitere passende und gute Filmempfehlungen zu geben. Durch Verbindungen zwischen Filmen und Filmen, Filmen und Benutzern, Benutzern und Genres und Filmen und Genres werden Matrizen aufgestellt und miteinander verglichen, um schlussendlich auf einen benutzerdefinierten Vektor zu kommen, der aufsteigend die besten Empfehlungen für Filme für diesen Benutzer darstellt.

3.1 Erstellung der benutzer_film_base_matrix

Zuerst wird eine Matrix aus den Daten der Base-Datei erstellt in der jede Spalte für einen Benutzer steht und jeder Film für eine Zeile. Wenn ein Benutzer einen bestimmten Film gesehen hat, steht in der Spalte von dem Benutzer und in der Zeile von dem Filme ein Einser. Daraus wird dann die benutzer_film_base_matrix erstellt. Eine gleiche Matrix wird auch aus der Test-Datei erstellt.

```
for i=1:size(base_matrix,1)
    benutzer_film_base_matrix(base_matrix(i,2),base_matrix(i,1))=1;
end
```

Abbildung 1: Erstellung von benutzer_film_base_matrix

3.2 Erstellung der benutzer_film_verbindungs_matrix

Durch das Vergleichen von zwei Zeilen in der benutzer_film_base_matrix, wird nach einer Übereinstimmung von Benutzern, die zwei gleiche Filme gesehen haben, gesucht. Dieses Übereinstimmungssystem besteht darin, dass zuerst 2 Zeilen ausgewählt werden und addiert werden. Dann wird von jedem Wert in dieser Zeile 1 abgezogen. Als nächstes wird zu jedem Wert in der Zeile sein eigener Betrag addiert und wird dann durch 2 dividiert. Dann wird jeder Wert in der Zeile addiert und wird dann in der neuen Matrix an die Stelle (1. verglichene Zeile, 2. verglichene Zeile) und (2. verglichene Zeile, 1. verglichene Zeile) geschrieben. Dadurch entsteht eine Matrix deren Symmetrieachse die Diagonale ist, wobei in den

Diagonalen nur Nullen stehen. Der Vergleich kreiert Verbindungen zwischen Benutzern die in der `benutzer_film_verbindungs_matrix` dargestellt werden.

```
for i=1:size(benutzer_film_verbindungs_matrix,1)
    for u=1:i-1
        t=benutzer_film_base_matrix(i,:)+benutzer_film_base_matrix(u,:);
        t=t-1;
        t=t+abs(t);
        t=t/2;
        t=sum(t);
        benutzer_film_verbindungs_matrix(i,u)=t;
        benutzer_film_verbindungs_matrix(u,i)=t;
    end
end
```

Abbildung 2: Erstellung von `benutzer_film_verbindungs_matrix`

3.3 Erstellung der `genre_film_verbindungs_matrix`

Das gleiche Übereinstimmungssystem wird benutzt um die `genre_film_verbindungs_matrix` zu erstellen, nur werden hier die Übereinstimmungen in der `genre_matrix` durchgeführt. Außerdem werden die Verbindungen bei Genres, die der User präferiert, stärker gewichtet, als bei denen die er sich nicht so gerne anschaut. Wie sehr er ein Genre mag, lesen wir aus der Base-Datei und der Genre-Matrix aus. Wir zählen sozusagen ab wie oft jedes Genre in seinen bereits geschauten Filmen vorkommt.

3.4 Ergebnis

Um unseren Algorithmus auszuprobieren, benutzten wir die Daten von *movielens*. Eine Webseite die einerseits in einer Datei die Matrix von Benutzern, und deren Filme mit Bewertungen von 1 bis 5 die sie angeschaut haben. Andererseits eine Datei, die jedem Film gewisse Genres zuteilt. Um den von uns geschriebenen Algorithmus am Schluss auch zu testen, stellt *movielens* zwei verschiedene Dateien zur Verfügung. Eine davon ist die Base-Datei, die eben aus Benutzern, gesehenen Filmen und deren Bewertungen besteht. Die andere ist die Test-Datei und beschreibt die Filme die der Benutzer danach gesehen hat, wieder mit den gegebenen Bewertungen. Unser Ziel war es, mit unserem Algorithmus vorherzusagen welche weiteren Filme ein gewisser Benutzer sich ansehen wird. Mit der Test-Datei konnten wir dies dann vergleichen. Wir haben die Richtigkeit und Genauigkeit unsere Empfehlungen mit diesem Vergleich berechnet. Unseren benutzerdefinierter Endvektor haben wir mit dieser Formel berechnet:

$$x = \alpha * A * x + \beta * b + C * x * \gamma$$

Wobei A=benutzer_film_verbindungs_matrix die Matrix der Verbindungen, aufgrund der Benutzer die sie gesehen haben, zwischen den Filmen ist, b=bewertungsvektor ist der Vektor der Bewertungen der benutzerdefiniert ist,

C=genre_film_verbindungs_matrix ist die Genre-Matrix und x ist der zufällige Startvektor. Diese Formel läuft durch eine Iteration, wo x immer wieder durch das neue x ersetzt wird.

Durch das Variieren von alpha, beta und gamma, suchten wir die perfekte Gewichtung der verschiedenen Matrizen und Vektoren. Als Ergebnis, welches die besten Empfehlungen herausgibt fanden wir:

alpha=0,007

beta=0,99

gamma=0,003

Hier wird deutlich gemacht, dass der Vektor der Bewertungen am wichtigsten für eine gute Empfehlung ist. Im Vergleich sind die Verbindungen zwischen einzelnen Filmen und den Genres weniger wichtig. Damit sich der Endvektor unten auf 1 summiert haben wir x durch die Summe von allen x-Werten dividiert, also:

$$x = \frac{x}{\text{sum}(x)}$$

4.0 Bewerten unserer Empfehlungen

Ein großer Teil unseres Programms, war auch dafür da unsere Empfehlungen zu bewerten, da es einfach zu viele Benutzer und Filme gegeben hat um es selbst zu machen.

Wir haben zwei Bewertungsmethoden verwendet, die erste haben wir selbst gefunden und die zweite haben wir die Idee aus einem wissenschaftlichen Paper und haben diese dann implementiert.

Die erste Methode hat unsere besten Empfehlungen mit den danach tatsächlich gesehenen Filmen aus der Test-Datei verglichen und umso mehr Filme in beiden vorgekommen sind, umso besser war die Empfehlung.

Bei der zweiten Methode vergleichen wir unsere Empfehlungswerte von den in der Test-Datei geschauten Filmen mit den Empfehlungs-Werten von allen Filmen die der Nutzer nie geschaut hat und jedes Mal wenn erstere größer sind wird das Ergebnis besser.

```
score_counter=0;
for i=1:size(geschaute_filme_test_vektor,1)
    for j=1:length(nie_geschaute_filme)

        if x(geschaute_filme_test_vektor(i,1),1)>x(nie_geschaute_filme(j),1)
            score_counter=score_counter+1;
        end
    end
end

score=score_counter/(size(geschaute_filme_test_vektor,1)*length(nie_geschaute_filme))
```

Abbildung 3: Schließliche Berechnung nach der 2. Methode

Wir haben einmal für alle 943 Nutzer für unsere Empfehlungen die Bewertung nach der 2. Methode berechnet und einen Durchschnittswert von 90% heraus bekommen.

5.0 Zusammenfassung

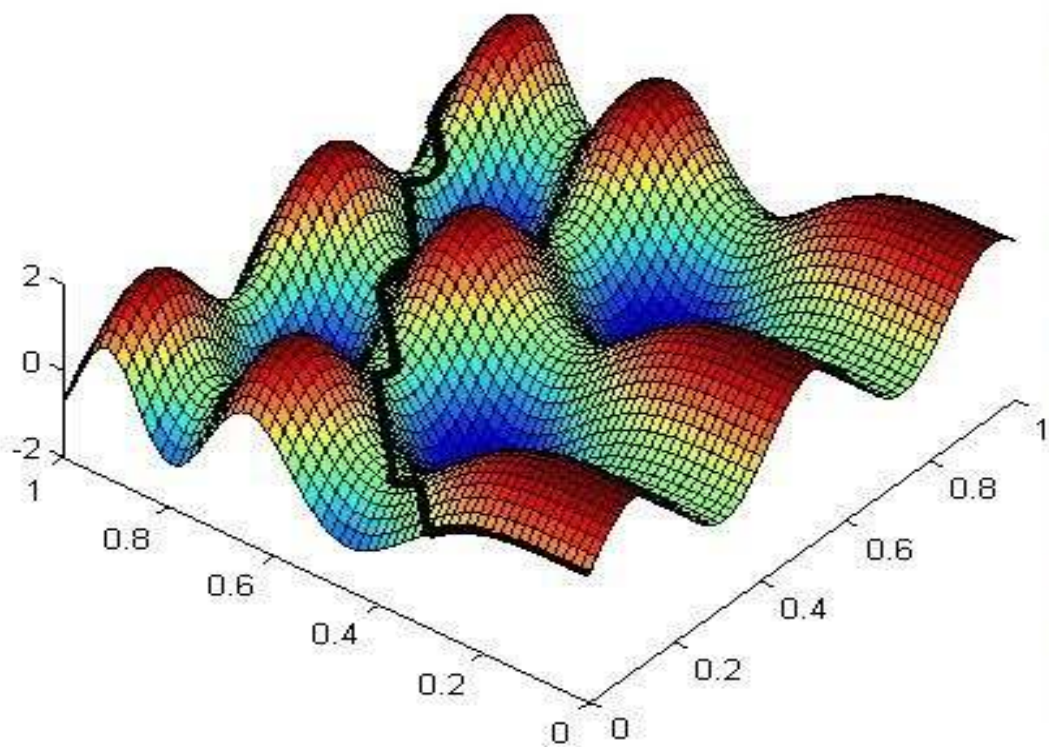
Nach diesen 5 Tagen haben wir ein Programm zusammengestellt mit dem wir von einer Filmdatenbank für einen bestimmten Benutzer weitere Filme empfehlen können. Unser Endergebnis mit unserer Bewertungsmethode betrug über 90%.

Großer Dank gehört unserem Betreuer Tobias.



Woche der Modellierung mit Mathematik

Voraussage des Straßenverkehrs in einem Transportnetzwerk



Gruppenmitglieder: Almana Graf, Cynthia Zeya, Doris Prach, Stefan Fischer, Sebastian Kölbl, Fuad Farajov

Betreuer: Ass.Prof. Dr. Laurent Pfeiffer

Jufa Pöllau 6. – 12. Februar 2016

Inhaltsverzeichnis

Einleitung	3
Berechnung des schnellsten Weges	4
Algorithmus von Dijkstra	4
Anwendung in einer dreidimensionalen Umgebung.....	6
Aufteilung des Verkehrs: einfache Beispiele	9
Beispiel 1: 2 Städte/ 2 Straßen	9
Beispiel 2: 3 Städte/ 6 Straßen	11
Aufteilung des Verkehrs: allgemeine Methoden	14
Beschreibung der Methoden.....	14
Grundfunktionen	15
Aktualisierung	16
Beispiel: 7 Städte / 12 Straßen	18

Einleitung

In diesem Projekt wird versucht ein Modell zu erstellen, welches den Verkehr voraussagen kann. In besagtem Modell nimmt jeder Autofahrer den schnellsten Weg, um sein Ziel zu erreichen. Zunächst werden numerische Methoden untersucht, um diesen schnellsten Weg zu berechnen. Solche Methoden sind übrigens in GPS implementiert. Danach wird analysiert, wie sich die Verbraucher des Netzes gegenseitig beeinflussen. Zum Beispiel, wenn sie alle versuchen, die Autobahn zu nehmen, entsteht ein Stau: es kann dann schneller sein, eine kleine Landstraße zu nehmen.

Im ersten Abschnitt wird zur Feststellung des schnellsten Weges der Algorithmus von Dijkstra beschrieben.¹ Im folgenden Kapitel wird die Aufteilung des Verkehrs anhand einfacher Beispiele erläutert. Insbesondere wird die gegenseitige Beeinflussung der Autofahrer berücksichtigt. Im letzten Teil dieser Arbeit wird ein Programm, mit dem der Verkehr modelliert werden kann, erstellt.

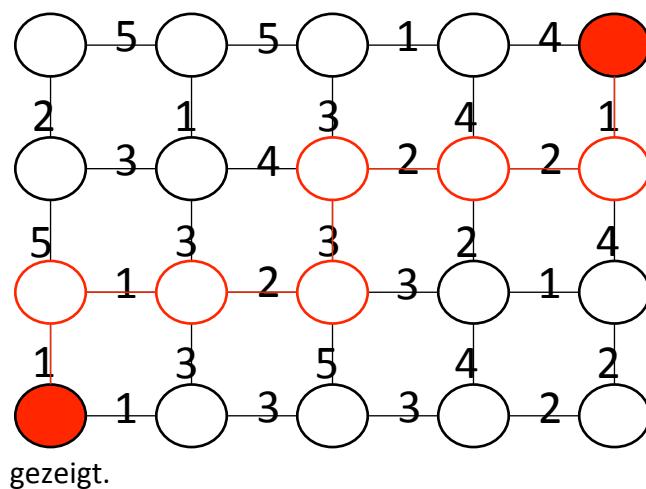
¹ DIJKSTRA, Edsger W.; *A note on two problems in connexion with graphs*. In: *Numerische Mathematik*. 1, 1959, S. 269–271

Berechnung des schnellsten Weges

Algorithmus von Dijkstra

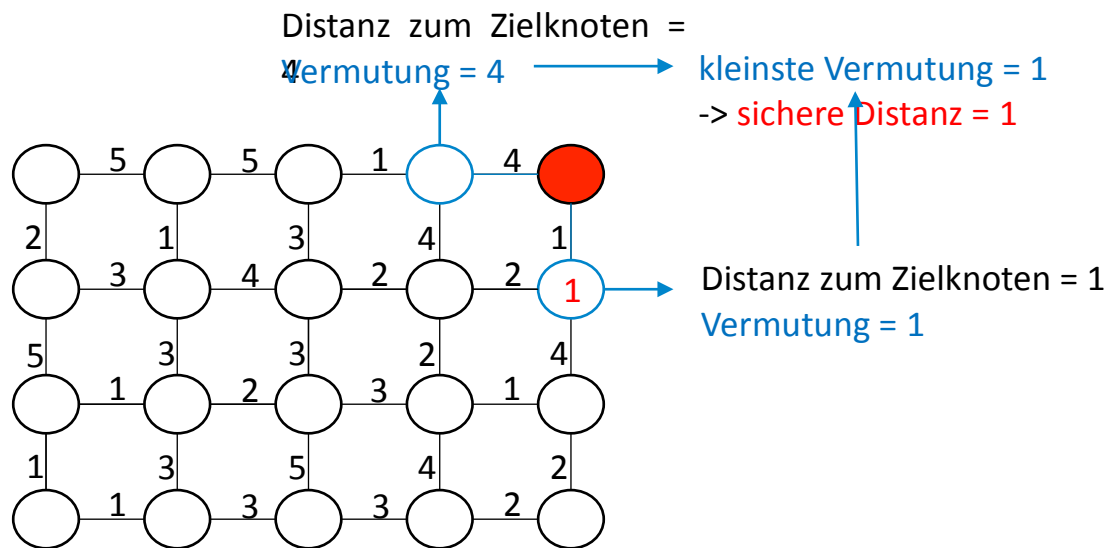
Ein Graph (Straßensystem) kann durch Knoten (Städte) und Kanten (Straßen) beschrieben werden. Jede Straße hat einen bestimmten Zahlenwert, welcher die Distanz angibt um von einem Knoten zum nächsten Knoten zu kommen. Um den schnellsten Weg zwischen zwei Knoten zu berechnen, kann der Algorithmus von Dijkstra verwendet werden. Die Grundidee dieses Algorithmus ist es, vom Zielpunkt aus immer derjenigen Kante zu folgen, die den kürzesten Streckenabschnitt verspricht, bis der Startpunkt erreicht wurde.

In einem einfachen Beispiel wird der kürzeste Weg zwischen den rot markierten Punkten

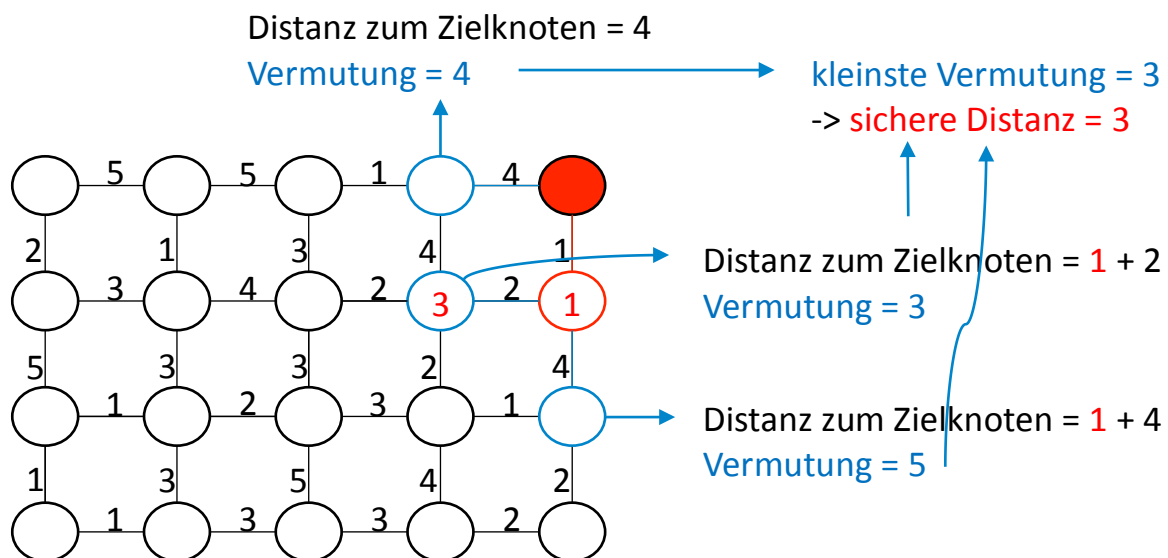


1. Zunächst wird der Distanz zum Zielknoten der Wert 0 zugeordnet.
2. Die Distanzen zu allen benachbarten Knoten werden als Vermutungen gespeichert. Diese Knoten bekommen nun den Zahlenwert, der die Distanz bis zum Zielknoten beschreibt.² Dies ist jedoch nur ein vorläufiges Ergebnis und kann später aktualisiert werden.
3. In diesem Schritt werden alle Vermutungen miteinander verglichen und die kleinste als fixe Distanz verwendet. Alle anderen Vermutungen werden wieder zurückgesetzt damit sie im nächsten Schritt wieder neu gespeichert werden können.

² Die gesamte Distanz setzt sich aus den Teildistanzen der bestmöglichen Wege bis zum Zielpunkt zusammen.



4. Die Schritte 2 und 3 werden wiederholt, wobei die Distanzen nicht nur von einem Zielknoten, sondern von allen bekannten Knoten berechnet werden. Die Anzahl der Wiederholungen hängt davon ab, ob alle Knoten erreicht wurden.

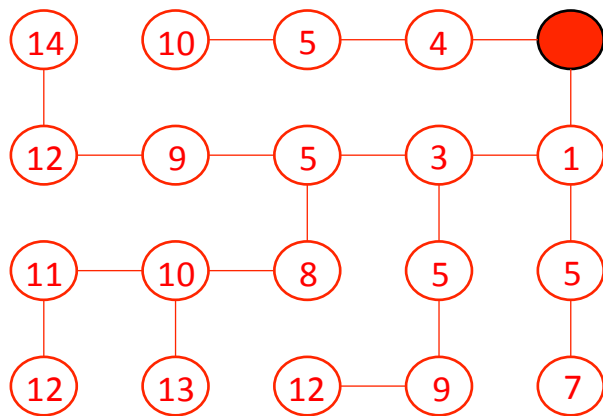


In dieser Form berechnet der Algorithmus ausgehend von einem Zielknoten die kürzesten Wege zu allen anderen Knoten. Ist man dagegen nur an dem Weg zu einem ganz bestimmten Knoten interessiert, so können Schritt 2 und 3 abgebrochen werden, wenn es zum gesuchten Knoten keinen kürzeren Weg gibt.

Ist für jeden Knoten die Distanz fixiert, wird ein neues Straßennetz erstellt, bei dem alle Knoten gleich bleiben, aber nur die „sinnvollen“ Straßen angezeigt werden. Dadurch existiert zu jedem Knoten, außer zum Startknoten, nur eine eindeutige Verbindung. Aus diesem Netz kann nun der kürzeste Weg zwischen zwei beliebig gewählten Punkten durch die folgende Methode bestimmt werden:

1. Vom Zielknoten ausgehend, wird ein vorläufiger Anfangsknoten bestimmt. Der Anfangsknoten ist derjenige, über den die Strecke vom Startknoten bis zum Zielknoten verläuft und nur eine Straße entfernt liegt.
2. Der vorläufige Anfangsknoten wird nun zum vorläufigen Endknoten, von dem aus wieder ein neuer Anfangsknoten gewählt wird, bis der Anfangsknoten gleich dem Startknoten ist.
3. In jedem Schritt wird der Name der Teilstraße gespeichert und kann am Ende zu einem vollständigen Weg zusammengefügt werden.

Hier sind die Distanzen zum Zielpunkt von jedem Knoten dargestellt.



Anwendung in einer dreidimensionalen Umgebung

Es wird zuerst ein dreidimensionales Koordinatensystem mit den Achsen x, y und z definiert. In dem Koordinatensystem können nach Belieben sogenannte „Berge“ und „Täler“ eingefügt werden.

Von einem bestimmten Anfangspunkt bis zu einem bestimmten Zielpunkt wird der optimale Weg berechnet. Der optimale Weg ist derjenige, bei dem die Dauer am geringsten ist. Wenn der Weg über Berge oder durch Täler führt, erhöht sich die Dauer des Weges. Daher wird versucht möglichst alle Berge und Täler zu umgehen.

Die Funktion, die die dreidimensionale Umgebung beschreibt, ist durch die x- und y-Koordinaten dim1 und dim2 begrenzt. Nachdem diese Dimensionen festgelegt wurden, kann eine Matrix mit den Daten der Städte (StädteMatrix) und eine Matrix mit den Daten der Straßen (StraßenMatrix) erstellt werden. Diese Matrizen beschreiben die Koordinaten der Städte, die irgendwo auf der Funktion liegen können, sowie die Straßen, die diese Städte verbinden.

In der Funktion „höhe“ wird die z-Koordinate durch die x- und y-Koordinaten berechnet.

Abhängig von den Werten, die sich dadurch für z ergeben, kann der Graph im Bereich dim1/dim2 dargestellt werden. Durch Veränderung von Koeffizienten der Funktion „höhe“ kann die Position der Berge und Täler verändert werden.

Der Algorithmus von Dijkstra, der vorher bereits näher beschrieben wurde, wird verwendet, um durch die x- und y-Koordinaten, sowie durch die StraßenMatrix die Distanzen aller Teilstrecken und den optimalen Weg vom Anfangspunkt zum Zielpunkt zu berechnen.

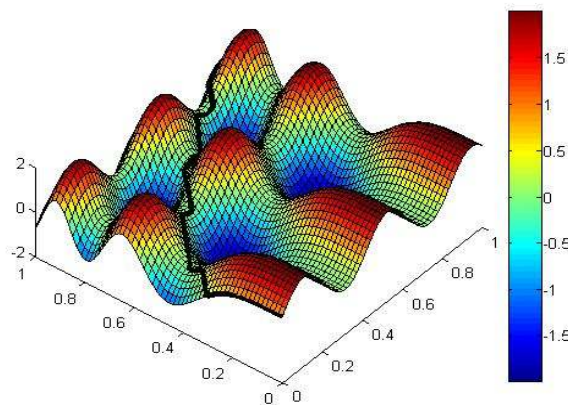
Mit dem Satz von Pythagoras kann die Distanz von 2 Punkten in einem dreidimensionalen Raum mit der folgenden Formel berechnet werden:

$$dis = \sqrt{(x' - x)^2 + (y' - y)^2 + (\text{höhe}(x) - \text{höhe}(y))^2}$$

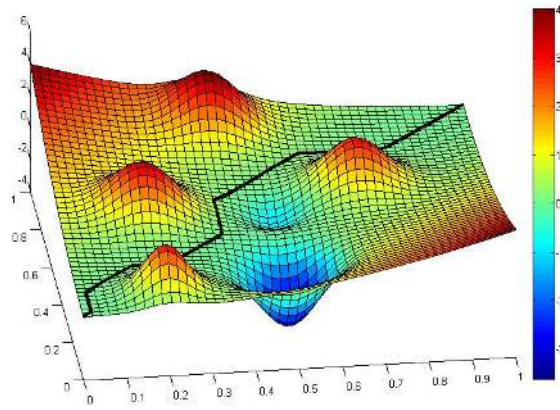
In diesem Fall sind (x|y) die Koordinaten für den ersten, und (x'|y') die Koordinaten für den zweiten Punkt.

In den folgenden 2 Beispielen stellt der schwarze Strich den optimalen Weg dar. Höhere Stellen sind rot und tiefere Stellen blau eingezeichnet. Wie man in beiden Bildern sehen kann, verläuft der Weg einigermaßen überall auf der selben Höhe und weicht Bergen und Tälern aus.

Unter den Graphiken ist die Definition von z angegeben.



$$z = \cos(15x^2) + \sin(15y)$$



$$g(y_1, y_2) = \exp(-y_1^2 - y_2^2)$$

$$\begin{aligned}
& 3g(14(x - 0,1), 14(y - 0,2)) + 2g(10(x - 0,5), 10(y - 0,1)) \\
& - 3g(10(x - 0,2), 10(y - 0,5)) + 3g(10(x - 0,9), 10(y - 0,4)) \\
& + 3g(10(x - 0,6), 10(y - 0,7)) - 1g(14(x - 0,5), 14(y - 0,5)) \\
& - 1g(14(x - 0), 14(y - 1)) - 1g(14(x - 0,8), 14(y - 0,1)) \\
& - 0,5g(14(x - 0,1), 14(y - 0,1)) - 0,5g(14(x - 0,5), 14(y - 0,9)) \\
& + 1g(10(x - 0,8), 10(y - 0,4)) + (x - y)^2
\end{aligned}$$

Aufteilung des Verkehrs: einfache Beispiele

Im täglichen Straßenverkehr wird davon ausgegangen, dass jeder Autofahrer, der von einem Punkt A zu einem Punkt B fahren will, bei der Auswahlmöglichkeit von mehreren Strecken immer die Strecke wählt, deren Fahrtzeit am geringsten ist. Allerdings ist dabei das Verhalten aller anderen Autofahrer zu berücksichtigen, die diese Dauer beeinflussen könnten. Die Berechnung des Verhaltens der Autofahrer lässt sich mit den folgenden Beispielen gut darstellen.

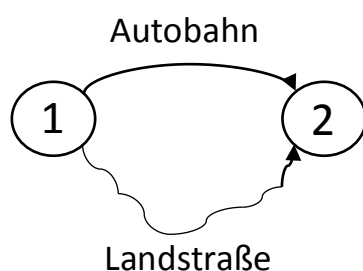
Um die optimale Aufteilung des Verkehrs zu berechnen, wird ein iteratives Verfahren benötigt. Mit diesem kann man sich der Lösung immer weiter annähern, bis sich die Lösung von der Aktualisierung fast nicht mehr unterscheidet, denn bei manchen Problemen lassen sich die Lösungen nicht gleich berechnen.

Beispiel 1: 2 Städte/ 2 Straßen

Gegeben:

- y ...Anzahl der Autofahrer gesamt
- x_j ...Anzahl der Autofahrer, die über eine bestimmte Straße j fahren
- p_j ...Reisezeit über die Straße j , wenn niemand darüber fährt
- q_j ...Steigung der Dauer der Autofahrt, wenn x Autofahrer mehr auf der Straße j fahren
- Reisezeit über die Straße j : $p_j + q_j x$

Beispiel:



Gegeben:

- Stadt 1 (Anfangsstadt)
- Stadt 2 (Zielstadt)
- Autobahn (A, schneller)
- Landstraße (L, langsamer)

Annahme:

Alle Autofahrer wollen über die kürzeste Strecke, also die Strecke mit der geringsten Dauer fahren. Da man von der Anfangsstadt aus die Zielstadt viel schneller über die Autobahn erreichen kann, fahren alle Autofahrer über die Autobahn. Wenn aber zu viele Autofahrer über die Autobahn fahren, entsteht ein Stau, dadurch steigt die Dauer der Fahrt über die Autobahn pro Autofahrer, der über diese fährt. Wenn eine bestimmte Anzahl an Autofahrern über die Autobahn fahren, wird die Dauer der Fahrt über die Autobahn größer als die der Fahrt über die Landstraße. Sobald dieser Fall eintritt, fahren Autofahrer über die neue kürzere Strecke, also die Landstraße. Das Ziel ist die Verteilung der Autofahrer bei einer ausgeglichenen Dauer der Fahrzeiten von Autobahn und Landstraße zu berechnen.

Gegeben:

- $Z_A(x_A)$...Dauer der Autobahnfahrt in Abhängigkeit von den Autofahrern
- $Z_L(x_L)$...Dauer der Landstraßenfahrt in Abhängigkeit von den Autofahrern
- y ...Anzahl der Autofahrer gesamt
- $x_A(y)$...Anzahl der Autofahrer, die über die Autobahn fahren
- $x_L(y)$...Anzahl der Autofahrer, die über die Landstraße fahren
- p_j ...Reisezeit über die Straße j , wenn niemand darüber fährt
- q_j ...Steigung der Dauer der Autofahrt, wenn x Autofahrer mehr auf der Straße j fahren
- Reisezeit über die Straße j : $p_j + q_j x$
- x ...1000 Autofahrer

Mit dem Beispiel der Landstraße und der Autobahn könnten Z_A und Z_L als lineare Funktionen beschrieben werden:

$$Z_A(x_A) = x_A + 1$$

$$Z_L(x_L) = 2x_L + 2$$

Um die Dauer der Landstraßenfahrt zu beschreiben, ist eine höhere Steigung sinnvoll, da diese Straße nur einspurig ist und es schneller zu Verzögerungen kommen kann.

Für ein System mit nur zwei Städten und zwei Straßen ist diese Aufgabe relativ einfach mit einem Gleichungssystem zu lösen, dabei muss jedoch zwischen zwei Fällen unterschieden werden. Wie man an den Formeln schnell erkennen kann, sind die Fahrtzeiten der beiden Straßen genau gleich, wenn $y = 1$.

1. Fall: $y \leq 1$;

$$x_A(y) = y,$$

$$x_L(y) = 0$$

In diesem Fall fahren alle Autofahrer über die Autobahn, da es bis 1000 Autofahrer über diese schneller ist.

2. Fall:

$y > 1$;

$$y = x_A + x_L,$$

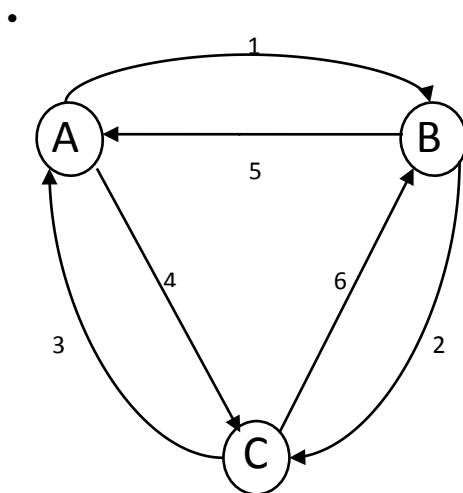
$$\rightarrow x_L(y) = \frac{y-1}{3}$$

$$\rightarrow x_A(y) = \frac{2}{3}y + \frac{1}{3}$$

Dieser Fall beschreibt das Verhalten der Autofahrer, sobald mehr als 1000 Autofahrer auf der Autobahn fahren, und dieser Weg somit so lange dauert wie die Landstraße. Das Ergebnis bedeutet, dass die Steigung der Anzahl der Autofahrer der Autobahn ab 1000 Autofahrern $\frac{2}{3}$, und die Steigung der Anzahl der Autofahrer der Landstraße $\frac{1}{3}$ beträgt.

Beispiel 2: 3 Städte/ 6 Straßen

Sollte es jedoch ein größeres Netz mit mehr als zwei Städten und Straßen geben, wird die Berechnung um Einiges komplizierter:



- i...Stadt i
- S j...Straße j
- Gruppen g: Autofahrer, die von einer bestimmten Stadt zu einer bestimmten anderen Stadt fahren.
- y_g ...Anzahl der Autofahrer einer bestimmten Gruppe
- x_g ...Autofahrer jeder Gruppe, die nur über eine Straße zu ihrem Ziel fahren
- u_g ...Autofahrer jeder Gruppe, die über zwei verschiedene Straßen an ihr Ziel fahren

Autofahrer jeder Gruppe:

1. G_1 : St 1→2
 $x_1(S \quad 1)$
 $u_1(S \ 6,5)$
2. G_2 : St 2→3
 $x_2(S \quad 2)$
 $u_2(S \ 4,6)$

3. G_3 : St 3→1
 $x_3(S \quad 3)$
 $u_3(S \ 5,4)$
4. G_4 : St 2→1
 $x_4(S \quad 4)$
 $u_4(S \ 2,3)$

5. G_5 : St 3→2
 $x_5(S \quad 5)$
 $u_5(S \ 3,1)$
6. G_6 : St 1→3
 $x_6(S \quad 6)$
 $u_6(S \ 1,2)$

Autofahrer über jede Straße:

1. S: x_1, u_5, u_6
2. S: x_2, u_4, u_6
3. S: x_3, u_4, u_5
4. S: x_4, u_2, u_3
5. S: x_5, u_3, u_1
6. S: x_6, u_2, u_1

Bedingungen des Problems:

1. Bedingung für jede Gruppe:
 $y_g = x_g + u_g$
2. 2 Bedingungen für jede Gruppe:
Beispiel Gruppe 1:
Dauer der Straße 1:
 $Z_1(x_1 + u_5 + u_6)$
Dauer der Straßen 5 und 6:
 $Z_5(x_5 + u_3 + u_1) + Z_6(x_6 + u_2 + u_1)$
 - Falls $Z_5 + Z_6 > Z_1$, gilt $u_1 = 0$;
Die Dauer der Straßen 5 und 6 ist größer als die Dauer der Straße 1, daher fahren alle Autofahrer der Gruppe 1 über die Straße 1.
 - Falls $Z_1 > Z_5 + Z_6$, gilt $x_1 = 0$;
Die Dauer der Straße 1 ist größer als die Dauer der Straßen 5 und 6, daher fahren alle Autofahrer der Gruppe 1 über die Straßen 5 und 6.

→ Für die 6 verschiedenen Gruppen gibt es insgesamt 18 Bedingungen.

Es werden also die Variablen x_g und u_g gesucht, sodass alle Bedingungen erfüllt sind. Das Problem wird mit einem iterativen Verfahren gelöst.

Man kann sich den Variablen annähern, indem man annimmt, dass nur x und u der entsprechenden Gruppe unbekannt sind.

Dies kann anhand des Beispiels der Gruppe 1 erkannt werden:

Dauer der Strecke 1 (Straße 1):

$$p_1 + q_1(u_5 + u_6) + q_1x_1$$

Zur Vereinfachung werden die bekannten Variablen der Formel mit a zusammengefasst:

$$a = p_1 + q_1(u_5 + u_6)$$

Dauer der Strecke 2 (Straßen 5 und 6):

$$p_5 + q_5(x_5 + u_3) + p_6 + q_6(x_6 + u_2) + (q_5 + q_6)u_1$$

Zur Vereinfachung werden die bekannten Variablen der Formel mit b zusammengefasst:

$$b + p_5 + q_5(x_5 + u_3) + p_6 + q_6(x_6 + u_2)$$

Bei der folgenden Aktualisierung der Variablen wird wieder zwischen drei möglichen Fällen unterschieden.

1. Alle Autofahrer der Gruppe 1 fahren über die Straße 1, wenn $b + (p_6 + q_6)y_1 \leq a$ gilt. $x_1 = y_1, u_1 = 0$
2. Alle Autofahrer der Gruppe 1 fahren über die Straßen 5 und 6, wenn $a + q_1y_1 \leq b$ gilt.
 $u_1 = y_1, x_1 = 0$
3. Die Anzahl der Autofahrer der Gruppe 1 sind regelmäßig auf die Straße 1 und die Straßen 5 und 6 verteilt, wenn $a + q_1x_1 = b + (q_5 + q_6)$ und $y_1 = x_1 + u_1$ gilt. Beim Lösen des Gleichungssystems ergeben sich folgende Ergebnisse für die Variablen:

$$u_1 = \frac{a-b+q_1y_1}{q_5+q_6+q_1}$$

$$x_1 = y_1 - u_1 \quad x_1 = y_1 - u_1$$

Mit diesen Ergebnissen können die Variablen nun aktualisiert werden.

Im in Matlab geschriebenen Programm „Verkehr“ werden diese Berechnungen für alle sechs Gruppen durchgeführt und mithilfe einer while-Schleife zusammengefügt. So werden die Variablen immer weiter aktualisiert, bis alle Bedingungen erfüllt sind. Die Lösung des Programms „Verkehr“ gibt an, wie viele Autofahrer der jeweiligen Gruppe die Strecke 1 mit nur einer Straße, und wie viele die Strecke 2 mit zwei Straßen benutzen.

Beispiel:

Wenn angenommen wird, dass die Fahrtzeiten aller Straßen ohne Autofahrer mit $y=x+1$ beschrieben werden und y für alle Straßen mit $y = [1 \ 2 \ 3 \ 4 \ 5 \ 6]$ definiert wird, ergeben sich folgende Werte:

$$x = [1 \ 2 \ 3 \ 4 \ 5 \ 5,33]$$

$$u = [0 \ 0 \ 0 \ 0 \ 0 \ 0,66]$$

Das Ergebnis bedeutet, dass alle Autofahrer der Gruppen 1-5 nur über eine Straße fahren, nur die Gruppe 6 teilt sich auf, dabei fahren $5\frac{1}{3}$ der Autofahrer über eine Straße und der Rest über zwei Straßen.

Aufteilung des Verkehrs: allgemeine Methoden

Das Programm, ist aus mehreren Funktionen zusammengesetzt, und diese kann schlussendlich ein Verkehrssystem in einer großen Stadt berechnen. Darin ist auch enthalten, welche Straßen die Autofahrer nehmen würden, damit sie am schnellsten bei ihrer Zielstadt ankommen.

Beschreibung der Methoden

Initialisierung des Verkehrs

Nun muss man die Funktion Getdauer in eine weitere Funktion einfügen, welche schlussendlich den gesamten Verkehr im Netz bestimmen kann. Also die Inputvariablen der Funktion „Verkehr“ sind „Strassenmatrix, Gruppenmatrix und nStaedte“, wobei nStaedte die Anzahl der Städte ist, und die Inputvariable ist x. Um nun die Initialisierung des Verkehrs durchzuführen, muss man zuerst x definieren. Dazu erstellt man zuerst eine Tabelle, mit der Zeilenanzahl „nStrassen“ und der Spaltenanzahl „nGruppen“.

Danach muss man eine Abfrage erstellen. In der Abfrage sollte für die Anfangsstadt, die Zielstadt und für die Anzahl der Autofahrer eine Matrix erstellt werden. In die Abfrage wird die Funktion „Getdauer“ aber auch die Funktion „schnellsterWeg“ eingefügt, damit Matlab als ersten Schritt alle Autofahrer in die Straße mit dem schnellsten Weg verlegen kann.

Iterative Aktualisierung des Verkehrs

In diesem Abschnitt des Programms wird die Verteilung des Verkehrs aktualisiert. Man kann die Autofahreranzahl in einer Gruppe auf einer Straße nur mithilfe eines iterativen Verfahrens aktualisieren. Das bedeutet, dass man die Lösungen des Problems nicht mit einer Formel direkt berechnen kann, sondern dass man sich nur mit bestimmten Schritten, die die Werte aktualisieren, der Lösung nähern kann.

Dabei braucht man eine while-Schleife. Diese Schleife führt die Befehle, die man festlegt, solange aus, bis die Bedingung in der while-Schleife nicht mehr erfüllt wird.

Um die Aktualisierung in der while-Schleife durchzuführen, braucht man eine neue Funktion, die den Straßenverkehr aktualisiert.

Für diese neue Funktion braucht man die Gruppen, die durch eine for-Schleife immer eine andere Gruppennummer annehmen, um alle Gruppen zu untersuchen. Dann braucht man auch die Straßenmatrix, in der alle Daten, wie die Anfangsstadt, Zielstadt, Reisedauer ohne andere Autofahrer und die Steigung der Reisezeit pro Autofahrer einer Straße gespeichert

sind. Auch benötigt man die Gruppenmatrix, welche die Anzahl der Autofahrer und die Anfang- und Zielstadt gespeichert hat. Schließlich braucht man auch die alte Matrix, welche aktualisiert werden muss. Natürlich benötigt man auch die Anzahl der Städte, die überhaupt angefahren werden können. Aus dieser Funktion erwartet man als Ergebnis eine aktualisierte Verkehrsmatrix.

Schließlich muss man am Anfang dieser Funktion verschiedene Variablen definieren, wie zum Beispiel die Anzahl der Straßen oder die Startstadt und die Zielstadt jeder Gruppe, um mit diesen Variablen in der Funktion arbeiten zu können. Die Hauptidee dieser Funktion ist, dass man mit ihr Autofahrer von der langsamsten Strecke auf die schnellste Strecke jeder Gruppe verlegt, um fast gleiche Reisezeiten für alle Streckenmöglichkeiten zu bekommen und so optimale Wege zu finden. Um auf diese Autoverteilung einzugehen benötigt man die schnellste und langsamste Strecke, die eine Gruppe für die Erreichung einer Stadt nehmen muss. Somit definiert man zwei neue Funktionen; eine, die den schnellsten Weg einer Gruppe berechnet und eine, die den langsamsten Weg einer Gruppe zwischen zwei Städten berechnet. Der schnellste Weg wird dabei mit dem Algorithmus von Dijkstra und aus der Dauer für die Bewältigung jeder Straße aus der „getDauer“-Funktion berechnet und auch der langsamste Weg wird auf eine ähnliche Weise wie der Dijkstra Algorithmus berechnet. Dabei kommen aus den beiden Funktionen zwei Vektoren heraus, welche mithilfe von 1 (wahr) und 0 (falsch) sagen, welche Straße eine schnellste bzw. eine langsamste Strecke bildet.

Jetzt braucht man nur noch die Autofahrerzahl jeder Gruppe, welche man aus der Gruppenmatrix lesen kann.

Nun hat man einmal alle Eingangsdaten, die man für die Aktualisierung benötigt.

Grundfunktionen

Straßenbefahrung

Die Anzahl der Autofahrer für jede Straße ist dabei auch enthalten, weshalb eine eigene Funktion nötig ist, denn die Anzahl der Autofahrer auf einer bestimmten Straße wirkt auch auf die Dauer der Straße ein. Die neue Funktion wurde „Straßenbefahrung“ genannt, und die Inputvariable ist „x“, eine Matrix, die die Anzahl der Autofahrer von jeder Gruppe auf jeder Straße beschreibt. Die Outputvariable ist „Straßenbenutzer“, ein Vektor, welcher von einer bestimmten Straße alle Autofahrer bestimmt.

Getdauer

Diese Funktion wurde in eine andere Funktion mit dem Namen Getdauer eingefügt. Dessen Inputvariablen sind „x“ und „Strassenmatrix“. Die Outputvariable der Funktion ist

„Dauervektor“, wobei dies die Dauer einer bestimmten Straße bestimmen kann. Dazu muss man zuerst „p“ und „q“ mit einer Matrix definieren, wobei p die Reisezeit einer Straße ohne Autofahrer ist und q die Steigung der Zeit pro Autofahrer. Am Ende dieser Funktion braucht man die folgende Gleichung, damit Matlab ausrechnen kann wie groß der Dauervektor ist.

```
Dauervektor(i)=pStrassen+qStrassen*Strassenbenutzer(i);
```

Dabei beschreibt „Strassenbenutzer“ die Autofahrer einer bestimmten Straße.

Aktualisierung

In der ersten for-Schleife untersucht man jede Straße und bestimmt, ob die jeweiligen Wege der langsamste Weg von einer Stadt zu einer anderen ist. Falls sie eine langsamste Strecke ist, wird die kleinste Autofahrerzahl auf der Strecke gesucht, um die gesamte Autofahrerzahl auf der langsamsten Strecke zu bestimmen. Das macht man alles in der ersten for-Schleife. In der zweiten for-Schleife werden die schnellste Wege von einer Stadt zu einer anderen Stadt untersucht. Dabei werden in dieser for-Schleife alle Steigungen pro Autofahrer der beteiligten Straßen für die Dauer zur Überwältigung des jeweiligen schnellsten Weges aufsummiert und gespeichert. Das Gleiche wird auch für die langsamsten Wege gemacht.

Wenn man jetzt Autos von der langsameren Strecke auf die schnellere Strecke verlegt, muss man bedenken, dass von der langsameren Strecke Autos weggenommen werden, daher wird die Dauer, um diese Strecke zu bewältigen, kürzer und da diese Autos der schnelleren Strecke hinzugefügt werden, wird die Dauer für die Bewältigung der Strecke auch größer. Nun muss man zwei verschiedene Fälle betrachten: Wenn alle Autos der langsameren Strecke auf die schnellere Strecke verlegt werden und dabei die Zeit für die Bewältigung der langsameren Strecke noch immer größer als die Bewältigung der schnelleren Strecke ist, können alle Autos von der langsameren Strecke auf die schnellere verlegt werden. Wenn dieser Fall aber nicht eintritt, muss man ein Gleichgewicht zwischen den beiden Zeiten finden. Dabei dürfen nur so viele Autos auf die schnellere Strecke verlegt werden, bis es gleich lange dauert, um die beiden Strecken zu bewältigen. Im Programm kann man diese zwei Fälle mithilfe einer if-Abfrage unterscheiden.

Jetzt werden in einer for-Schleife alle Straßen noch einmal untersucht und wenn diese Straße eine langsamste Strecke bildet, wird in der Straßen-Gruppenanzahl die verlegte Autofahreranzahl vom alten Wert für die Autofahreranzahl in einer Gruppe auf einer Straße abgezogen. Falls die Straße eine schnellste Strecke bildet, wird diese Autofahreranzahl, die von der langsameren Strecke verlegt worden ist, auf die Verkehrsmatrix der schnelleren Strecke zum alten Wert addiert und die Funktion wird beendet.

Jetzt haben wir die Daten für eine Gruppe aktualisiert. Nun wird der gleiche Vorgang für die nächste Gruppe gemacht und so weiter. Wenn jetzt alle Gruppen aktualisiert wurden, geht das Ganze wieder von vorne mit den neu aktualisierten Daten los. So kommt man Schritt für Schritt der Lösung näher.

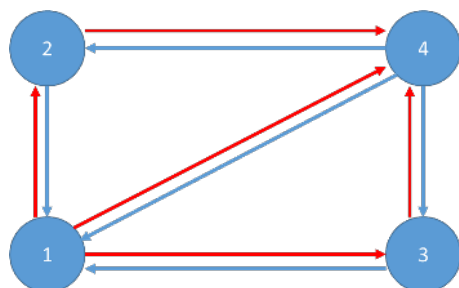
Die while-Schleife endet erst, wenn die alte Verkehrsmatrix von der neuen sich um weniger als 0,01 unterscheidet.

Nun kann man das Ganze als Matrix ausgeben lassen und man sieht dann, wie sich die Autofahrer in jeder Gruppe auf die Straßen verteilen.

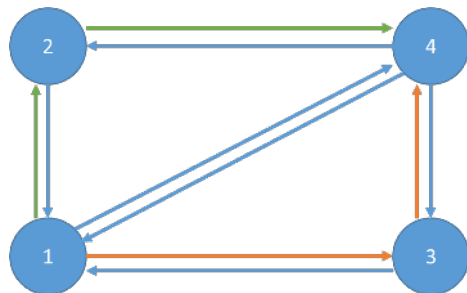
Auch kann man jetzt das Ganze in einem Graphen ausgeben lassen, nachdem man die Koordinaten der Städte eingegeben hat und man sieht das entstandene Netz und wie die Städte miteinander verbunden sind.

Mithilfe des iterativen Verfahrens hat man sich der Lösung immer weiter genähert, bis sich die Lösung von der Aktualisierung fast nicht mehr unterscheidet, denn bei manchen Problemen lassen sich die Lösungen nicht gleich berechnen.

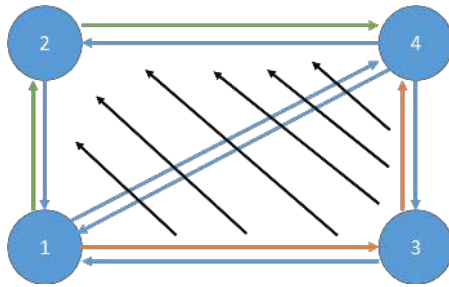
Erklärung des Verfahrens anhand eines Beispiels



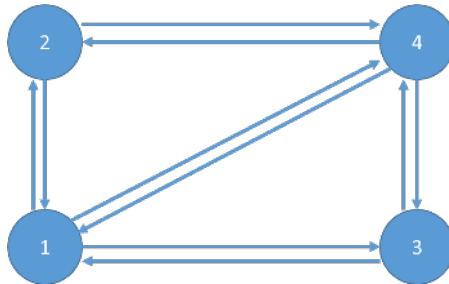
Man möchte von der Stadt 1 die Stadt 4 erreichen. Es gibt 3 verschiedene Wege, dies zu machen (1-4; 1-2-4; 1-3-4).



Jetzt sind auf der schnelleren Strecke mehr Autos und auf der langsameren Strecke weniger Autos. Da jetzt die Fortbewegung auf der schnelleren Strecke langsamer und auf der langsameren Strecke schneller geworden ist, hat sich jetzt ein Gleichgewicht eingestellt.



Aber die Dauer, die verschiedenen Strecken zu bewältigen, ist unterschiedlich lang. Grüne Pfeile bedeuten, dass dort die schnellere Strecke ist, orange bedeutet, dass die Strecke dort am langsamsten zu überwinden ist. Und blau bedeutet, dass die Reisezeit mittellang ist.

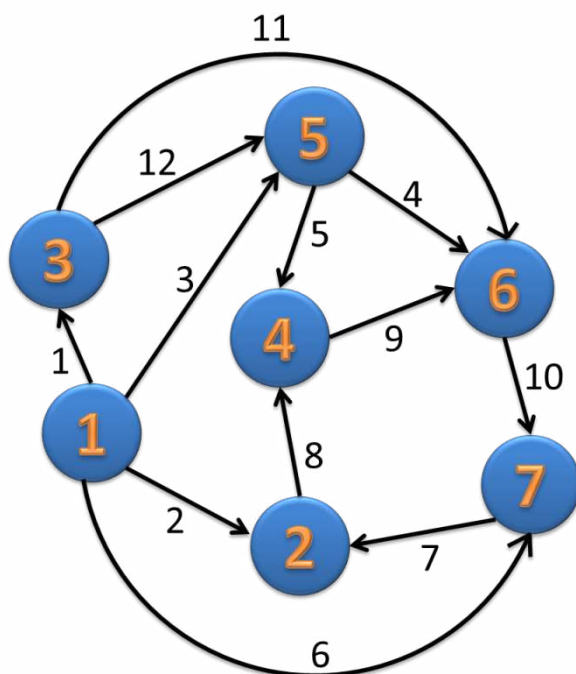


Deswegen muss man Autofahrer von der langsameren Strecke auf die schnellere übertragen, um entweder alle Autofahrer von der langsamen Strecke auf die schnellere zu übertragen oder wenigstens ein Gleichgewicht herzustellen.

Beispiel: 7 Städte / 12 Straßen

In diesem Teil wird ein Verkehrssystem mit 7 Städten und 12 Straßen modelliert. Die Verteilung der Autofahrer im System wird mithilfe der Funktion „Verkehr“ dargestellt.

Straßenmatrix und Gruppenmatrix des Verkehrssystems:



Straßenmatrix	Anfangsstadt	Zielstadt	p	q
Straße 1	1	3	1	3
Straße 2	1	2	2	1
Straße 3	1	5	1	2
Straße 4	5	6	1	1
Straße 5	5	4	2	3
Straße 6	1	7	1	1
Straße 7	7	2	4	1
Straße 8	2	4	3	1
Straße 9	4	6	2	2
Straße 10	6	7	3	1
Straße 11	3	6	5	0.5
Straße 12	3	5	2	1

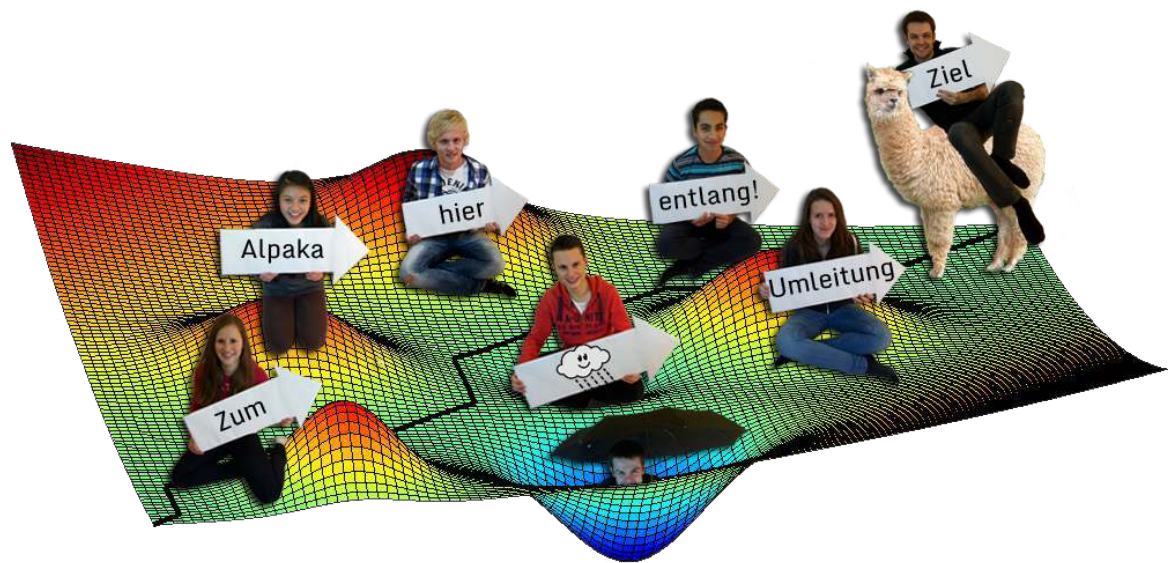
Gruppenmatrix	Anfangsstadt	Zielstadt	Autofahrer der Gruppe n (1-12)
Gruppe 1	1	3	1
Gruppe 2	1	2	9000
Gruppe 3	1	5	50
Gruppe 4	5	6	1558
Gruppe 5	5	4	1
Gruppe 6	1	7	9642
Gruppe 7	7	2	1
Gruppe 8	2	4	1480
Gruppe 9	4	6	1
Gruppe 10	6	7	1260
Gruppe 11	3	6	460
Gruppe 12	3	5	8340

Lösung	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12
S1	0.001	0	0	0	0	1.19	0	0	0	0	0	0
S2	0	8.46	0	0	0	0	0	0	0	0	0	0
S3	0	0.53	0.05	0	0	0.52	0	0	0	0	0	0
S4	0	0.10	0	1.55	0	0.52	0	0	0	0	0	0
S5	0	0.43	0	0.001	0.01	0.01	0	0	0	0	0	0
S6	0	0	0	0	0	7.92	0	0	0	0	0	0
S7	0	0.53	0	0	0	0	0.001	0	0	0	0	0
S8	0	0	0	0	0	0	0	1.48	0	0	0	0
S9	0	0.43	0	0.001	0	0.001	0	0	0.001	0	0	0
S10	0	0.53	0	0	0	1.71	0	0	0	1.26	0	0
S11	0	0	0	0	0	1.19	0	0	0	0	0.46	0
S12	0	0	0	0	0	0	0	0	0	0	0	8.34

In dieser Tabelle sieht man wie viele Autofahrer aus der Gruppe n (=beliebige Gruppe von 1-12) die Straße n (1-12) benutzen. Man sieht auch, dass sich die Gruppen 2 und 6 am meisten aufteilen.

Wenn man nun die Gruppe 2 nimmt, die von der Anfangsstadt 1 beginnen und zur Zielstadt 2 fahren wollen, kann man erkennen, dass es 5 schnellste Wege gibt, die alle die gleiche Reisezeit haben, da sich die Autofahrer richtig verteilen. Dies sind die Wege (2), (3, 5, 9, 10, 7), (3, 4, 10, 7), (6, 7) und (1, 11, 10, 7). Die Autofahrer der Gruppe 2 benutzen aber nur 3 der 5 schnellsten Wege, obwohl es möglich wäre alle 5 Straßen zu benutzen. All diese Wege

haben eine Reisezeit von 8466 Zeiteinheiten. Es gibt nur zwei langsamste Wege und dies sind (1, 12, 4, 10, 7) und (1, 12, 5, 9, 10, 7). Die zwei Wege werden von den Autofahrern nicht benutzt, da sie eine Reisezeit von 18163 Zeiteinheiten haben.





Statistische Invarianten und Trading-Strategien für den S&P500

Konstantin Andritsch,
Alexander Burtscher,
Hansi Droschl,
Tomas James Hain,
Florian Huebler,
Valentin Szilavec,
Konstantinos Xylouris

Modellierungswoche 2016 in Pöllau

If by Rudyard Kipling

If you can keep your head when all about you
Are losing theirs and blaming it on you,
If you can trust yourself when all men doubt you,
But make allowance for their doubting too;
If you can wait and not be tired by waiting,
Or being lied about, don't deal in lies,
Or being hated, don't give way to hating,
And yet don't look too good, nor talk too wise:

If you can dream—and not make dreams your master;
If you can think—and not make thoughts your aim;
If you can meet with Triumph and Disaster
And treat those two impostors just the same;
If you can bear to hear the truth you've spoken
Twisted by knaves to make a trap for fools,
Or watch the things you gave your life to, broken,
And stoop and build 'em up with worn-out tools:

If you can make one heap of all your winnings
And risk it on one turn of pitch-and-toss,
And lose, and start again at your beginnings
And never breathe a word about your loss;
If you can force your heart and nerve and sinew
To serve your turn long after they are gone,
And so hold on when there is nothing in you
Except the Will which says to them: 'Hold on!'

If you can talk with crowds and keep your virtue,
Or walk with Kings—nor lose the common touch,
If neither foes nor loving friends can hurt you,
If all men count with you, but none too much;
If you can fill the unforgiving minute
With sixty seconds' worth of distance run,
Yours is the Earth and everything that's in it,
And—which is more—you'll be a Man, my son!

Inhaltsverzeichnis

If by Rudyard Kipling	1
1 Einleitung	3
1.1 Vokabeln	4
1.2 Standard & Poor's 500	4
1.3 Methodische Instrumente	5
1.3.1 Fourier-Transformation	5
1.3.2 Fourier-Reihe	5
1.3.3 Monte-Carlo	6
1.4 Randomtrader	6
2 Strategie nach statistischen Invarianten	9
2.1 Tägliche Schwankungsbreiten pro Monat	9
2.2 Fallende Zeiträume	9
2.3 Rebound-Strategie	11
2.4 Curvature-Strategie	11
2.5 Maximale Schwankungsbreite jedes Monats	12
2.6 Just do it-Strategie	12
2.7 Repetitive-Bull-Strategy	14
3 Hoch- und Tiefpunkte Strategie	17
3.1 Corridor-Strategie	18
3.2 Max/Min-Layer-Trap-Strategie	19
3.3 Long-wins-everything-Strategie	20
3.4 Similar Triangles-Strategie	22
4 Historische Strategien	23
4.1 Livermore-Strategie	23
4.2 W-Formation-Strategie	25
4.3 Regressionsgeraden	26
4.4 Labourmarket-Strategie	26
5 Fourierreihen	28
6 Perceptron	30
6.1 Perceptron-Long-Strategie	30
6.2 Perceptron-Short-Long-Strategie	31
6.3 Fourier-High-Energy-Frequenzy-Perceptron-Strategy	33

Kapitel 1

Einleitung

Obwohl der Handel von Produkten eventuell so alt wie Menschheit ist, hat die moderne Börse ihre Gründungsjahre im zwölften Jahrhundert gefeiert, als in Frankreich erstmals Wertpapiere in Form von Schuldverpflichtungen landwirtschaftlicher Gemeinden gehandelt wurden. Im Laufe der Zeit hat sich die Börse zu einem riesigen Basar entwickelt, der von vielen Unternehmen genutzt wird, um sich durch den Verkauf von Unternehmensanteilen (Wertpapieren) Geldmittel zu beschaffen. Stark vereinfacht kann man sich die Börse also als großen und sehr liquiden Handelsplatz vorstellen, bei der sich eine Angebotsseite mit einer Nachfragepartei zusammenfindet, um einen Austausch zwischen Geld und Gütern vorzunehmen. Vor dem Hintergrund, dass diese Handlungen stets Menschen ausführen, geprägt mit allen ihre Emotionen, den Ängsten mögliche Gewinne verlieren und den Hoffnungen, Verluste möglichst bald ausgleichen zu können, oder aber der Gier nach mehr, liegt es durchaus Nahe, dass bei diesen Transaktionen über alle Zeiten hinweg in der Masse wiederkehrende Muster zu erwarten sind. Auf der Suche nach solchen Mustern ist Jesse Livermore einer der erfolgreichsten Spekulanten des 20ten Jahrhunderts geworden. Ihm war es gelungen aus diesen Mustern erfolgreiche Tradingstrategien zu formen.

Im Schatten dieser Erfolge ist das Ziel dieser Arbeit, durch Anwendung mathematischer Methoden statistische Invarianten im Markt zu finden, die auf Strategien für das Investieren in zukünftige Marktentwicklungen hindeuten. Durch eine exzessive Analyse der Kursdaten aus den letzten 54 Jahren des S&P500 Markts haben wir auch Algorithmen gefunden, die in einem statistisch verifizierbarem Setting zwischen verschiedenen zeitlichen Stichproben erheblichen Erfolg zeigen. Diese Strategien haben nicht nur den Markt geschlagen, sondern auch einen parallel dazu konstruierten Zufallstrader. Letzterer wiederum wird die Vermutung bestätigen, dass ein rein zufälliges Investieren in den Markt den Erfolgchancen eines Münzwurfs entspricht. Die Strategien und Methoden dieser Arbeit beruhen auf statistische Größen, wiederkehrende Muster, Fourier-Transformationen und -reihen, neuronale Netzwerke, ausgezeichnete Zeitpunkte - wie Arbeitsmarktdaten, Crashauslöser und Boomanfänge, sowie Monte-Carlo ähnliche Methoden.

Die Rohdaten entstammen aus der Finanzdatenbank von Yahoo-Finance. Diese stellt unter anderem die Daten des S&P 500 seit dem 3. Jänner 1950 bis heute zur Verfügung. Dabei werden zu jedem Tag der Eröffnungskurs, der Schlusskurs, der Höchstkurs und der Tiefstkurs bereit gestellt. Weil diese Daten – insbesondere in den Anfangsjahren – jedoch nur unzureichend aufgezeichnet sind, – Schlusskurs, Eröffnungskurs und Tiefstkurs sind in diesen Jahren identisch angegeben – werden die ersten 12 Jahre, bei denen dieses zutrifft, aus der betrachteten Datenmenge unberücksichtigt gelassen. Es werden also Kursdaten vom 2. Jänner 1962 – 8. Februar 2016 betrachtet, analysiert und zum

Entwickeln und Testen der Strategien verwendet. Um die entwickelten Methoden statistisch robust gegenüber verschiedenen Stichproben zu machen, ist dieser Zeitraum – vom 2.1.1962 – 8.2.2016 – noch einmal in zwei Zeiträumen unterteilt worden. In einen Referenzzeitraum, der die Zeitspanne vom 2.1.1962 – 31.12.2014 umfasst und einen Testzeitraum von 2.1.2015 – 8.2.2016. Die Idee ist nun, statistische Größen der Kursentwicklung und Tradingstrategien im Referenzzeitraum zu finden und anschließend diese im Testzeitraum unabhängig zu überprüfen.

1.1 Vokabeln

Der S&P 500 (Standard & Poor's 500) ist der amerikanische Markt, der die 500 größten amerikanischen Unternehmen zusammenfasst. Er hat eine Marktkapitalisierung von knapp 12.5 tausend Milliarden Dollar. Steigt er zum Beispiel um einen Prozentpunkt, so ändert sich dieser Wert um 125 Milliarden nach oben. Wenn man Anteile des Markts in der Hoffnung einer Wertsteigerung kauft, spricht man davon, dass man in den Markt long geht. Im Falle, dass man auf fallende Kurse setzt, spricht man von einer Short-Position. Solche Short-Positionen können durch Leerverkäufe oder andere meist von Banken herausgegebene Wertpapieren bezogen werden. Während sich der Gewinn einer Long-Position durch die Differenz zwischen Verkaufs- und Einkaufswert ergibt, ist er bei der Short-Position genau umgekehrt. In dieser Arbeit werden auch Wertpapiere berücksichtigt, die die prozentuale Bewegung eines zugrundeliegenden Kurses hebeln. In diesem Fall kann man überproportional an seiner Position gewinnen bzw. verlieren. Es wird kurz von Hebel gesprochen mit der Angabe seines Faktors. Beispielsweise wenn der S&P500 mit einem Hebel von 5 geshortet wird und er am Ende des Tages um 3% gefallen ist, so beträgt der prozentuale Gewinn dieser Position 15%.

Jede Transaktion wird zumeist von Banken ausgeführt, welche dafür sogenannte Transaktionskosten ansetzen. Bei Gewinnen fallen zusätzlich Kapitalertragssteuern an, welche knapp 25 Prozent des Gewinns ausmachen.

In dieser Arbeit wird unter dem Begriff Schwankungsbreite in einen Zeitraum die Differenz zwischen dem Höchstkurs und Tiefstkurs in demselben Zeitraum verstanden.

Die Börse ist nur an regulären Wochentagen geöffnet. Sie steht an Feiertagen und am Wochenende still. Damit ergeben sich pro Jahr im Schnitt knapp 251 Handelstage. Auf dem Parkett gehandelt wird der S&P500 zwischen 9:30 Uhr und 16:00 Uhr New Yorker Ortszeit.

1.2 Standard & Poor's 500

In folgender Grafik (Abbildung 1.1) ist die Kursentwicklung des S&P 500 vom 02.01.1962 bis zum 31.12.2014 zu sehen. Dieser Zeitraum entspricht genau 13342 Handelstagen. Abgebildet wird für jeden Tag der Eröffnungskurs. Der Kurs besaß im Jahr 1962 einen Wert von knapp 70 Punkten und konnte sich in den folgenden 53 Jahren auf knapp 2000 Punkten nahezu verdreifachen. Seine Marktkapitalisierung im Jahr 1962 betrug also knapp 400 Milliarden Dollar. Es wäre interessant festzustellen, ob diese Verdreifachung mit der durch die Zentralbanken bedingten Geldvermehrung in einem Verhältnis steht. Ein Sparbuch mit einer jährlichen Verzinsung von 3% hätte im selben Zeitraum das angelegte Kapital knapp verfünffacht (genauer um den Faktor 4.7904 vermehrt).

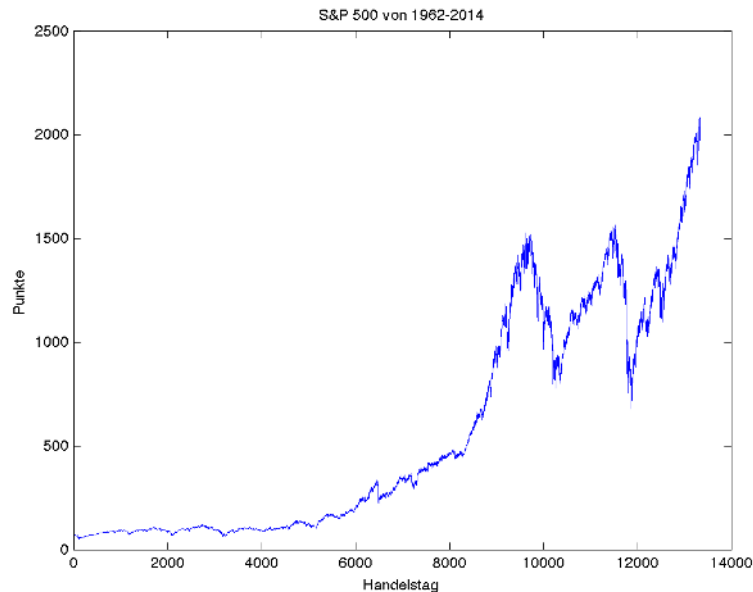


Abbildung 1.1: Verlauf des S&P 500 in den letzten 53 Jahren, von 1962- 2014, angegeben in Handelstagen. Sein Kurs hat sich in diesem Zeitraum knapp verdreißigfacht. Das Kapital eines Sparbuchs hätte sich im selben Zeitraum nur verfünffacht.

1.3 Methodische Instrumente

1.3.1 Fourier-Transformation

Die Fourier-Transformation einer Funktion über die Zeit liefert eine eindeutig Darstellung dieser Funktion im Frequenzraum. Diese Darstellung gibt an, welche Frequenzen in der ursprünglichen Funktion dominieren- man spricht auch von Frequenzen hoher Energie. Besonders bei durch Rauschen gestörte Daten erkennt die Fourier-Transformation zugrundeliegende Oszillationen mit deren Periode. Im Grunde kann die Fourier-Transformation also als Frequenzfilter bezüglich beliebigen Funktionen verstanden werden. Besonders bei den stark verrauschten Börsendaten ist sie damit ein sehr mächtiges Werkzeug, um zeitliche Einflussbereiche zu erkennen und auszunutzen.

1.3.2 Fourier-Reihe

Die Fourier-Reihe approximiert eine beliebige Funktion als Linearkombination trigonometrischer Funktionen – $\cos(\alpha t)$ - und $\sin(\alpha t)$ -Funktionen. In dieser Arbeit werden die α der Fourier-Reihe aus den gewonnenen Frequenzen der Fourier-Transformation angenähert und die Koeffizienten ihrer Linearkombination mit Hilfe des Lösen eines Gleichungssystems bestimmt, das aus der Interpolation des betrachteten Zeitraums entsteht. Die Fourier-Reihe kann zum Beispiel zur Extrapolation verwendet werden. Es ist interessant ihre Aussagekraft aus den gegebenen Daten zu überprüfen und Hinweise auf ihre Gültigkeit zu finden.

1.3.3 Monte-Carlo

In dieser Arbeit wird unter Monte-Carlo jene numerische Methode verstanden, die bezüglich eines stochastischen Prozesses ihren Erwartungswert bestimmt. Im Effekt werden dabei viele verschiedene Realisationen berechnet und anschließend über deren Anzahl gemittelt.

1.4 Randomtrader

Um sämtliche Strategien zu evaluieren, die auf Grundlage der Daten des Referenzzeitraums (zwischen 1962 und 2014 entwickelt) sich ergeben, werden einerseits Ihre Vorhersagen auf den Testzeitraum (von 2. Januar 2015 bis zum 8. Februar 2016) angewendet und auf Richtigkeit überprüft. Andererseits wird ihre Performance mit jener eines Zufallstraders verglichen. Dieser Zufallstrader handelt dabei auf folgender Grundlage: Er hat stets einen Zustand, der nur zwei Werte annehmen kann. Entweder ist der Zufallstrader im Markt investiert oder er ist befindet sich außerhalb. Jeden Tag entscheidet er sich durch je einen Münzwurf am Morgen und am Abend, ob er seinen Zustand ändert. Wenn er seinen Zustand ändert und er im Markt investiert ist, dann stellt er sofort seine Position glatt. Wenn die Änderung wiederum impliziert, dass er in den Markt einsteigen soll, so wirft er wieder eine Münze, um seine Position zu bestimmen, ob er auf steigende oder auf fallende Kurse setzen soll. Der Zufallstrader handelt nur den S&P 500 und er möge bei jedem Trade (aus Gründen der Einfachheit) sein komplett zur Verfügung stehendes Kapital einsetzen. Er kauft stets ungehebelte Produkte. Sein Verhalten wird für den Fall mit und ohne Transaktionskosten verglichen. Im letzteren betragen die Transaktionskosten genau 13 Dollar. Der Zufallstrader beginnt mit einem Startkapital von 100.000 Dollar im Jahr 1962. Methodisch werden 1000 Realisationen dieses Zufallstraders berechnet und anschließend deren Mittelwert bestimmt.

Der Zufallstrader besitzt besonderes Interesse, weil wenn man annimmt, dass sämtliche Strategien dem absoluten Zufall untergeordnet sind, diese zu Ergebnissen führen müssten, die mit dem Zufallstrader korreliert sind. Mathematisch betrachtet ist der Zufallstrader ein stochastischer Prozess über die Zeit von seinem Startkapital. Die Realisationen dieses Prozesses über den Referenzzeitraum ist in Abbildung 1.2 festgehalten. Wie erwartet ist sein Erwartungswert zum Ende des Referenzzeitraums unverändert genau bei ca. 100.000Dollar. Er handelt über 13342 Handelstage und macht dabei 6673 Trades. Von diesen hat er 3337 Longpositionen bezogen und 3336 Shortpositionen. Von den Longpositionen sind 1437 erfolgreich (das sind 43%) und von den Shortpositionen 1260(ca. 38% Erfolgsquote). Diese Zahlen und Quoten ändern sich für den Fall mit Transaktionskosten nicht. Jedoch sind die Transaktionskosten sehr entscheidend über einen so langen Zeitraum. Im Mittel verliert dadurch der Zufallstrader über 85% seines Kapitals. Sein durchschnittliches Kapital am Ende des Zeitraums beträgt nur noch 12000Dollar. Interessant ist hierbei, dass er zu 94.5% aller Fälle weniger besitzt wie zu Beginn. Außerdem ist er bei 554 Fällen bankrott gegangen (Abbildung 1.3).

Analoge Graphiken (Abbildungen 1.4, 1.5) ergeben sich für den Ertrag des Zufallstrader für die Stichprobe des Testzeitraums. Dieser umfasst insgesamt 277 Handelstage. Bei Berücksichtigung der Transaktionskosten von 13Dollar hat der Randomtrader am Ende im Mittel knapp 98000 tausend Dollar, ohne Transaktionskosten sind es ziemlich genau das Startkapital von 100000Dollar. Er hat dabei 138 Trades ausgeführt, von denen er 69 Longpositionen und 69 Shortpositionen eingegangen ist. 34 dieser Longpositionen sind erfolgreich und 35 der Shortpositionen.

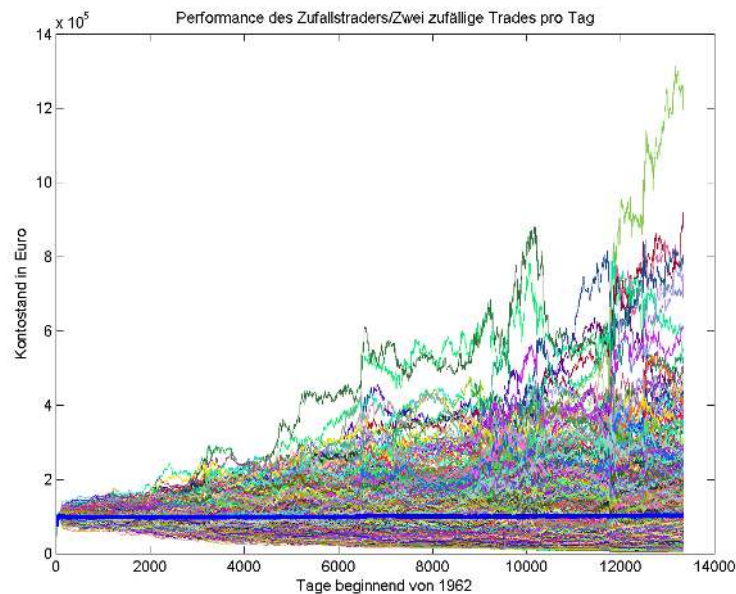


Abbildung 1.2: Leistung des Randomtraders ohne Transaktionskosten über den Referenzzeitraum. Im Mittel verändert sich das Kapital des Traders nicht. Während es einige Fälle gibt, in denen er über diesen Zeitraum Pleite geht, gibt es auch einen Case, bei dem er fast über 40% gewinnt. Er handelt über 13342 Handelstage. Dabei macht er 6673 Trades, von denen 3337 long sind und 3326 short. Von den Longpositionen gewinnt er 1437 mal und bei den Shortpositionen 1260 mal.

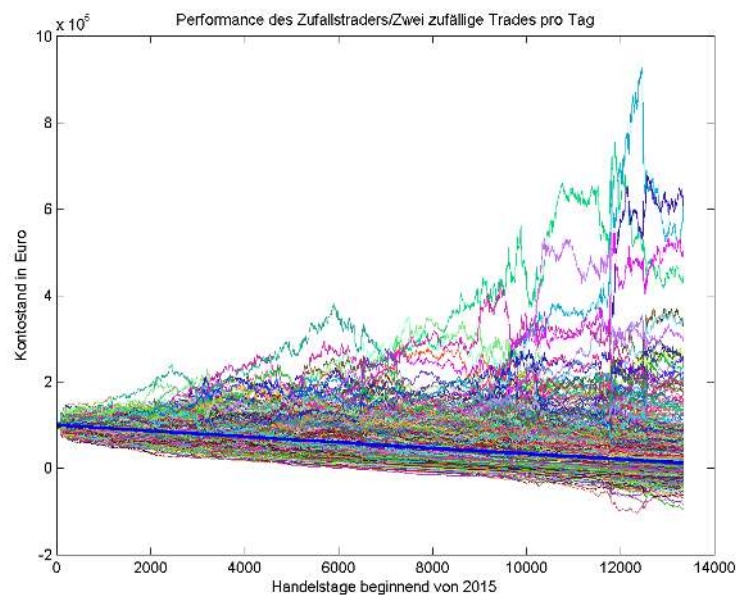


Abbildung 1.3: Leistung des Randomtraders mit Transaktionskosten über den Referenzzeitraum. Beginnend bei 100000 Dollar hat der Trader im Mittel am Ende nur noch 12000Dollar. In 945 Fällen macht der Randomtrader Verlust - in 554 Fälle geht er sogar pleite.

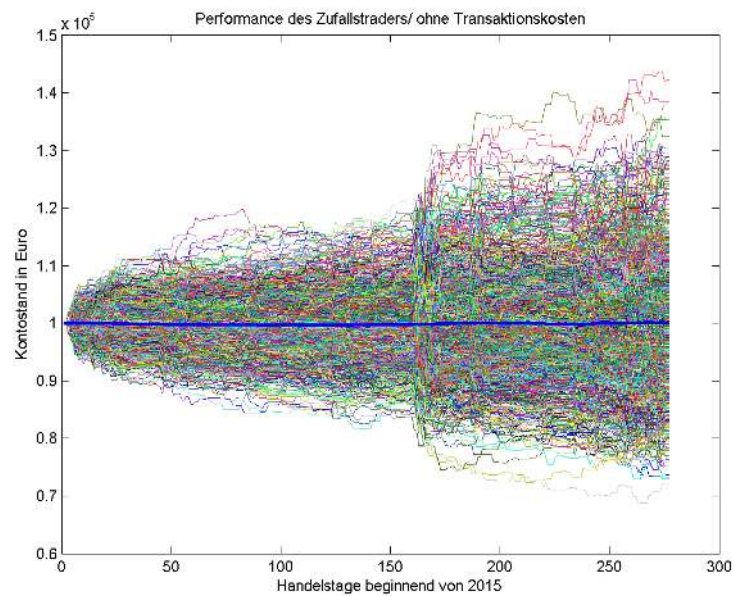


Abbildung 1.4: Leistung des Randomtraders ohne Transaktionskosten über den Testzeitraum. Im Mittel ändert sich das Kapital des Traders nicht.

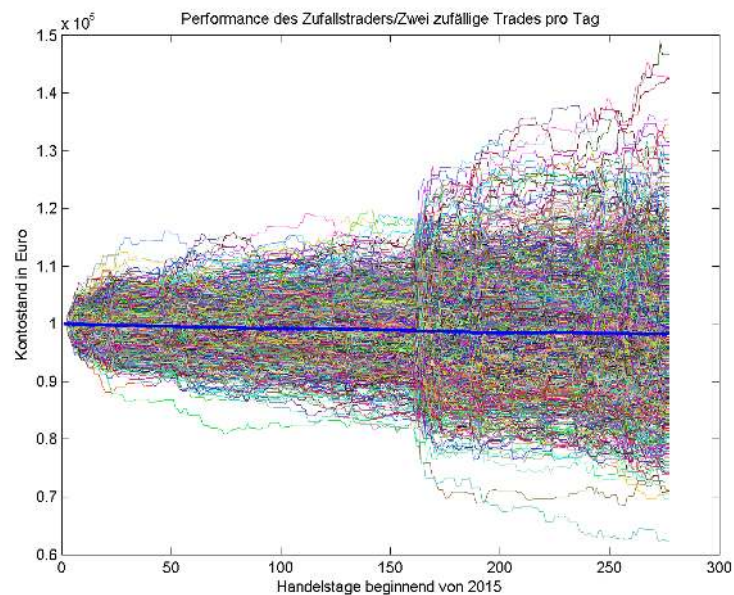


Abbildung 1.5: Leistung des Randomtraders mit Transaktionskosten über den Testzeitraum. Im Mittel hat der Trader am Ende noch 98000 Dollar. Die Transaktionskosten pro Trade betragen 13 Dollar und er hat insgesamt 138 Trades in 277 Handelstagen vorgenommen. Davon sind jeweils 69 Long und Short. Ca. die Hälfte waren erfolgreich.

Kapitel 2

Strategie nach statistischen Invarianten

2.1 Tägliche Schwankungsbreiten pro Monat

Eine der ersten Möglichkeiten die Kurse zu betrachten ist es, die Schwankungsbreite jedes Tages, sprich den prozentualen Unterschied des Tageshöchst- und des Tagestiefstwerte monatsweise zusammenzufassen. Wenn diese Schwankungsbreite für alle Tage des zur Verfügung stehenden Zeitraumes errechnet wird und für jeden Monat der Durchschnitt gebildet wird, ergibt sich der Graph in Abbildung 2.1. Das erste Detail, das einem ins Auge fällt ist der überdurchschnittliche hohe Wert im Oktober und der daraufhin starke Abfall bis Dezember, der das Tief des Graphen bestimmt. Daraus lässt sich interpretieren, dass der Monat Oktober sehr unruhig verläuft, also ein Tag durchschnittlich sehr hoch steigt und auch wieder fällt. Diese Instabilität ist für Käufer, die sich nur kurzfristig im Markt aufhalten, sehr gefährlich, da sich der Kurs des Markts sehr schnell verändern kann. Diese Schwankungsbreite kann auch über die 53 Jahre hinweg dargestellt werden (Abbildung 2.2). Die Werte für den Oktober sind generell etwas über denen der anderen Monate, jedoch verhalten sich alle Monate recht ähnlich zueinander. Das bedeutet, dass sich ein Jahr nicht nur in wenigen Monaten instabil verhält, sondern über das gesamte Jahr.

2.2 Fallende Zeiträume

Eine weitere interessante statistische Kennzahl des Kurses ist die Anzahl der aufeinanderfolgenden Tage, in denen der Kurs fällt (Abbildung 2.3) - von Kurseröffnung des heutigen Tages bis zur Kurseröffnung des nächsten Tages. Über den Referenzzeitraum von 13342 Handelstagen ist der Kurs insgesamt knapp 3200 Male gefallen. Genau nur ein Mal in diesem Zeitraum beim ca. 300 Mal, in dem er gefallen ist, ist er 12 Tage hintereinander gefallen. Relativ häufig fällt er an bis zu zwei aufeinander folgenden Tagen. Je länger dieser Zeitraum jedoch andauert, umso seltener tritt er auf. Die Wahrscheinlichkeit, dass der Kurs am nächsten Tag steigen wird, nachdem er fünf Tage lang gefallen ist beträgt 58%. (Eine Durststrecke von 13 Tagen ist in dem Referenzzeitraum kein einziges Mal gefunden worden.) Diese Erkenntnis führt direkt zur Rebound-Strategie.

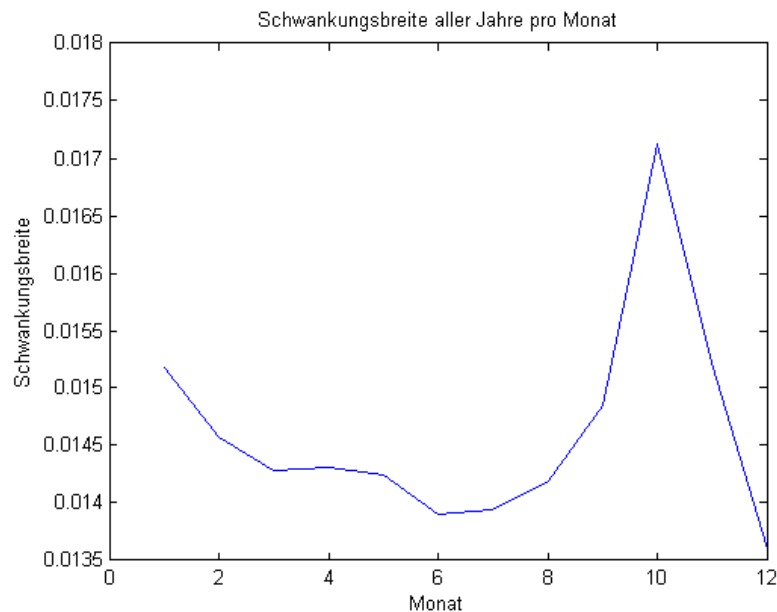


Abbildung 2.1: Schwankungsbreite eines Tages pro Monat gemittelt über den gesamten Referenzzeitraum. Der Markt ist im Oktober über diesen Zeitraum am volatilsten gewesen mit einer täglichen durchschnittlichen Bewegung von knapp 1.8%. Die Sommermonate und der Dezember haben eine auffallend geringe Schwankungsbreite.

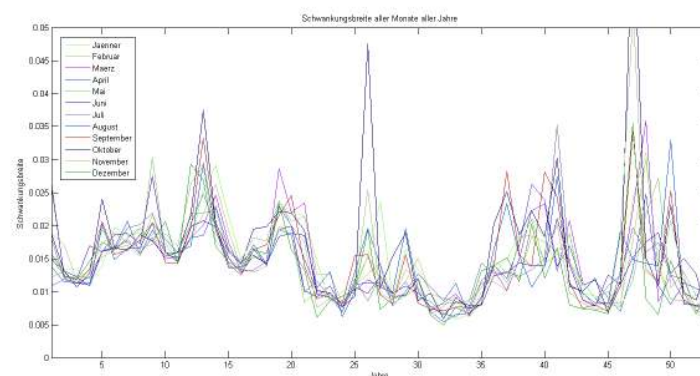


Abbildung 2.2: Durchschnittliche tägliche Schwankungsbreite pro Monat und Jahr. Die tägliche Schwankungsbreite pro Monat hängt vom Jahr ab. Ist ein Jahr besonders volatil, so sind es alle in im enthaltenen Monate. Der Monat Oktober sticht dabei immer ein wenig heraus.

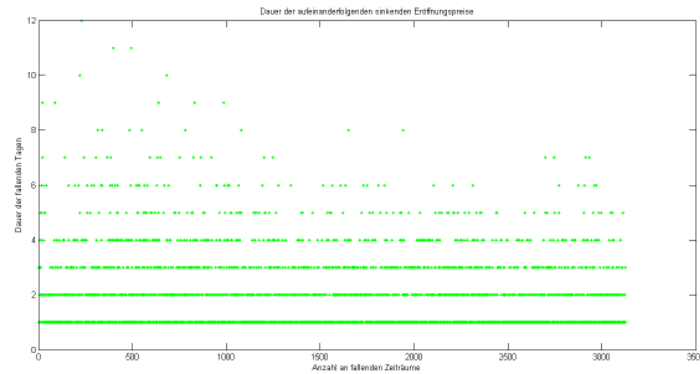


Abbildung 2.3: Häufigkeit der aufeinanderfolgenden fallenden Tage. Der S&P 500 ist von den 13342 Handelstagen, insgesamt 3200 Mal gefallen. Die grünen Punkte geben an, bei welchem Mal der Kurs an aufeinanderfolgenden Tagen gefallen ist.

2.3 Rebound-Strategie

Diese Daten legen nahe, dass nach fünftägigem Fallen des Kurses ein Anstieg folgt. Diese Strategie möge Rebound-Strategie heißen. Dazu wird, wenn der Kurs nach fünf Tagen um insgesamt mehr als 6% gefallen ist, das Ausgangskapital von 100.000\$ in den Markt investiert. Erwartet wird, dass der Kurs sich wieder erholt, also der Preis steigt. Die Position wird also in Erwartung eines unmittelbaren Anstiegs am nächsten Tag zum Eröffnungskurs wieder verkauft. War dieser Preis höher als der Kaufpreis, so macht man Gewinn, andernfalls Verlust.

Wird diese Strategie über den Referenzzeitraum der 53 Jahre angewandt, so ergibt sich ein Steigerung des Kapitals auf 350.000\$, das ist 3.5 mal so viel wie zu Beginn (Abbildung 2.4). Ein Vorteil dieser Strategie ist die relativ Hohe Verlässlichkeit. Man macht in 65% der Fälle einen Gewinn. Allerdings kann man sein Kapital auch sehr schnell wieder verlieren, sollte der Kurs noch weiter sinken. Des Weiteren macht man oft nur sehr kleine Gewinne, da der Kurs an solch einem Rebound-Tag sehr selten um mehr als 1% steigt. Daher wurde diese Strategie nicht für das Jahr 2015 angewandt, da mit eher niedrigen Gewinnen zu rechnen ist.

2.4 Curvature-Strategie

Könnte es sinnvoll sein, dem Strom der Massen zu folgen? Sollte man also dann zuschlagen und kaufen, wenn alle kaufen und genau das tun was jeder tut? An dem Kursverlauf ist es leicht festzustellen, ob die Mehrheit der Menschen kauft oder verkauft. Werden viele Aktien gekauft, so ist die Nachfrage groß und somit auch der Preis hoch. Wird viel verkauft, so sinkt die Nachfrage und mit ihr auch der Preis.

Diese Gedanken führen zur Curvature-Strategie, das sich Positionieren nach Krümmung des Kurses. Die letzten fünf Tage des Kurses werden ungefähr als Teil einer quadratischen Funktion betrachtet. Ist die Krümmung positiv, also die zweite Ableitung des Kurses größer Null, so wird das ganze Kapital, beginnend 100.000\$ im Jahr 1962 in den S&P 500 investiert. Verkauft wird die Position, sobald der Kurs zu fallen beginnt, oder nach spätestens drei Tagen. Da der Kurs in den letzten Tagen immer gestiegen ist, erwartet

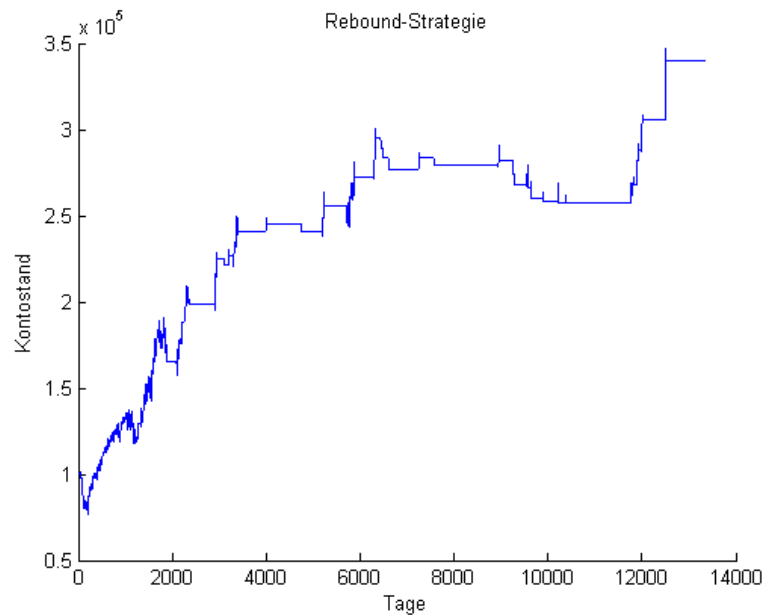


Abbildung 2.4: Kontostand der Rebound-Strategie über Referenzzeitraum. Das anfängliche Kapital von 100000\$ verdreifacht sich auf 300000\$.

man diese Entwicklung auch in den nächsten Tagen. Bei einer negativen Krümmung werden Short-Positionen eingegangen und später um einen zu erwartenden niedrigeren Preis wieder glatt gestellt, sodass man in beiden Varianten mit mehr Geld aussteigt.

Nach Ausführung dieser Strategie auf den Zeitraum von 1962 bis 2014 ergab sich ein Gewinn von 70%. Das ist eine Summe von 170.000\$ (Abbildung 2.5). Wie in der Grafik zu sehen ist, gibt es zwei sehr hohe Verluste, die den Gewinn der vorherigen Transaktionen wieder annullieren. Aufgrund der seltenen Erfüllungen dieser Bedingungen werden auch sehr wenige Transaktionen durchgeführt. Schwächt man diese jedoch ab, um öfter handeln zu können, kann es zu noch höheren Verlusten führen, die womöglich ein Pleite nach sich ziehen. Den Massen zu folgen und in deren Strom zu schwimmen bringt einen an der Börse also nicht ans Ziel.

2.5 Maximale Schwankungsbreite jedes Monats

In der Abbildung 2.6 ist die maximale Schwankungsbreite des Kurses für jeden Monat von Januar 1962 bis Dezember 2014 dargestellt. Dabei wird zuerst die Differenz zwischen dem Monatshöchstwert und dem Monatstiefstwert für jeden Monat gebildet und dann der daraus resultierende Wert durch den Monatsöffnungswert des jeweiligen Monats dividiert. Die maximalen Schwankungsbreiten der jeweiligen Monate bewegen sich zwischen rund 2.5% und maximal 35%. Dabei ist kein erkennbares Muster hinter der Veränderung der maximalen Schwankungsbreiten erkennbar.

2.6 Just do it-Strategie

Bei dieser Strategie geht es darum, sich vor Beginn der Anwendung dieser Methode einen einzelnen Monat auszuwählen (z.B. Jänner). Nun kauft man immer am Anfang

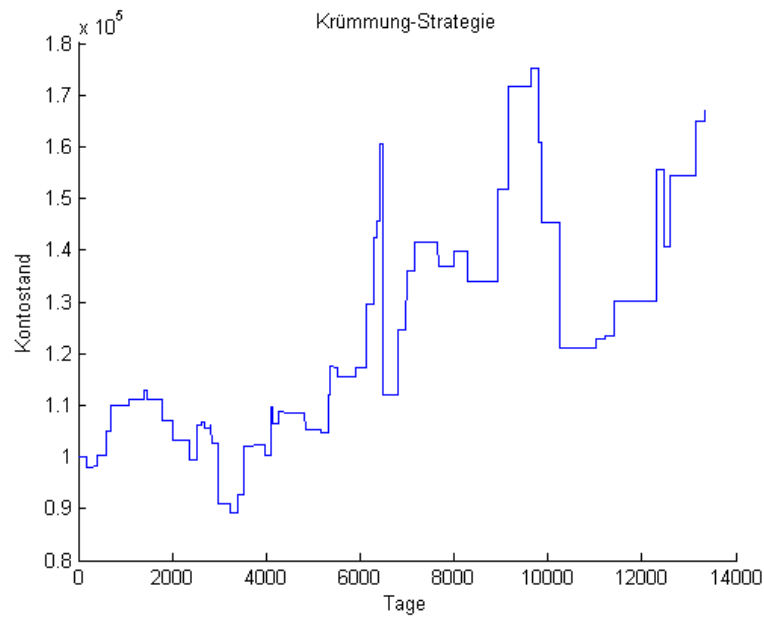


Abbildung 2.5: Kapitalverlauf nach der Curvature-Strategie. Die Anwendung der Strategie, den Massen zu folgen, führt zu einem Gewinn von nur lediglich 70% über den Referenzzeitraum.

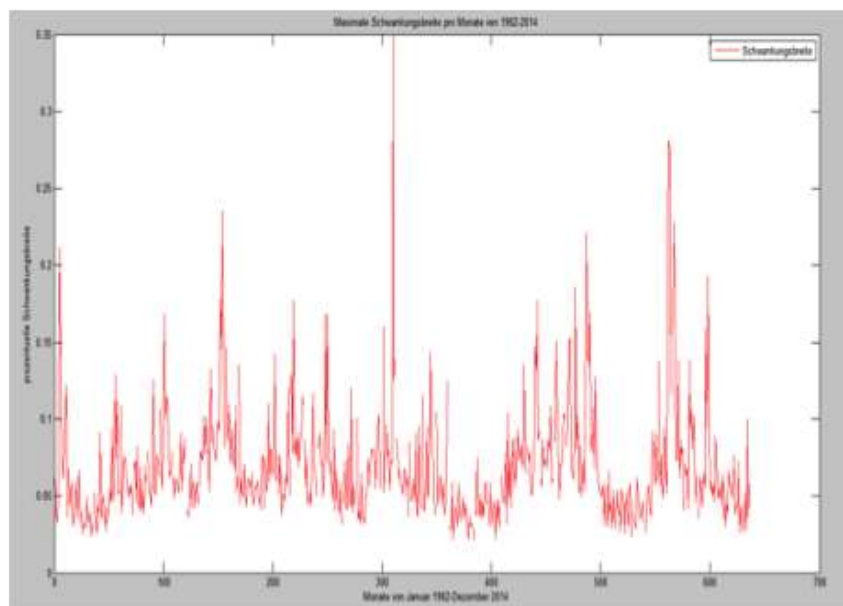


Abbildung 2.6: Monatliche Schwankungsbreite aufgezeichnet über alle Monate beginnend von 1962 bis Ende 2014. Diese schwankt zwischen 2,5% und 35%

des gewählten Monats Aktien zum Monatsöffnungswert dieses Monats unabhängig von der momentanen Situation des Marktes. Am Ende des Monats verkauft man seine Aktien zum Monatsschlusswert wieder, wobei auch hier wiederum nicht auf den momentanen Wert der Aktien geachtet wird. Mithilfe der zur Verfügung gestellten Daten, die von 2.1.1962 bis 31.12.2014 reichen, ist es möglich den resultierenden Gewinn/Verlust zu berechnen, wenn man die oben genannte Strategie von 1962 bis 2014 verfolgt hätte (Abbildung 2.7).

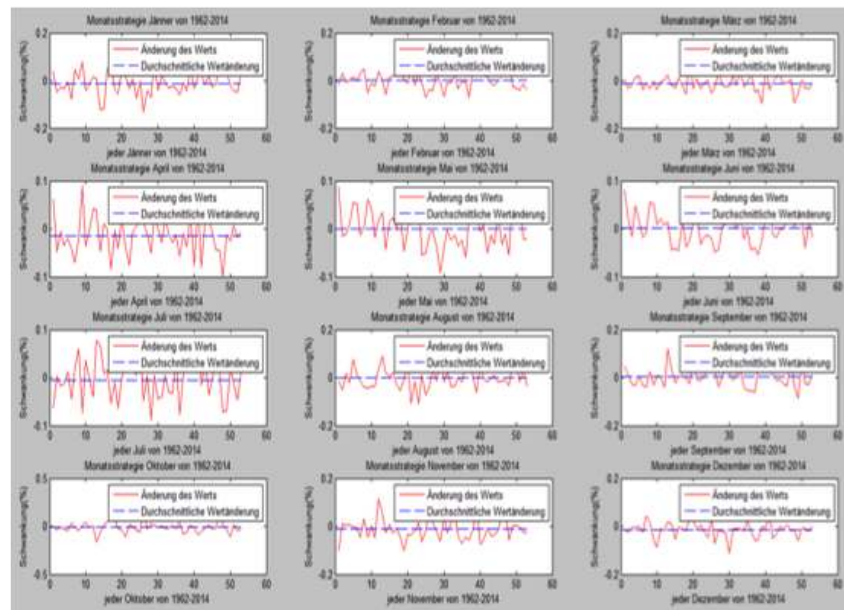


Abbildung 2.7: Prozentuale Rendite der Just-Do-It-Strategie für verschiedene Monate. Kaufen am Monatsanfang und verkaufen am Monatsende bezüglich eines speziellen Monats. Der Handelszeitraum ist der gesamte Referenzzeitraum. Für alle Monate bis auf den September macht man im Durchschnitt einen Verlust. Der Gewinn für den September ist jedoch so klein, dass er akkumuliert über die 53 Jahre nur 20% beträgt.

In der Abbildung 2.7 ist die Veränderung des Kontostandes im Zeitraum von 1962 bis 2014 dargestellt und jede der 12 Darstellungen bildet den Kontostand für einen anderen von Anfang an gewählten Handelsmonat ab. Nur wenn man den Monat September als Handelsmonat gewählt hätte, hätte man am Ende der Zeitspanne einen Gewinn zu vermelden. Bei allen anderen Monaten hätte man deutlichen Verlust gemacht. Allerdings ist diese Handelsstrategie auch im Monat September nicht wirklich erfolgreich, da man bei einer Steigerung des Grundkapitals um 20% in 53 Jahren nicht von einer effektiven Strategie sprechen kann.

2.7 Repetitive-Bull-Strategy

Das Ziel einer Long Strategie ist es immer jene Tage zu finden, an denen der Kurs steigt – umso besser, wenn mehrere davon in einer Reihe auftreten. Um Auslöser für ein repetitives Ansteigen des Marktes zu finden, bietet es sich an, jene Tage zu finden, an denen der Markt dieses Verhalten aufweist. Mit der Annahme, dass dieses Verhalten Auslöser haben muss, bieten sich nun die vorhergehenden Tage als Grundlage für weitere Untersuchungen an. Bei genauerer Betrachtung dieser Daten ließ sich nun eine Ähnlichkeit in den Veränderungsraten feststellen, die vor allem am Tag vor dem ersten Anstieg des Marktes wenige Abweichungen aufweisen (Abbildung 2.8).

Diese Daten legen nun die Strategie nahe, dann zu kaufen, wenn die Änderungsraten an 5 aufeinanderfolgenden Tagen in den Korridor zwischen den Standardabweichungen fallen und die Aktien für mehrere Tage zu halten. Wenn man diese Strategie nun auf den Referenzzeitraum (siehe Einleitung) anwendet so ist sie durchaus erfolgreich (Abbildung 2.9).

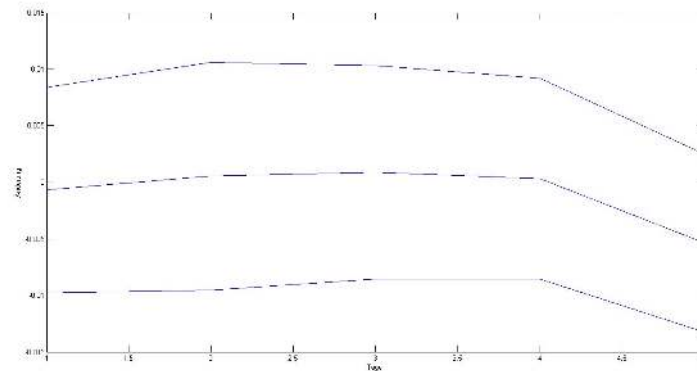


Abbildung 2.8: Durchschnittlicher relativer Kursverlauf über fünf Tage, gemittelt über den Referenzzeitraum, bevor der Kurs an vier aufeinanderfolgenden Tagen steigt mit Standardabweichung.

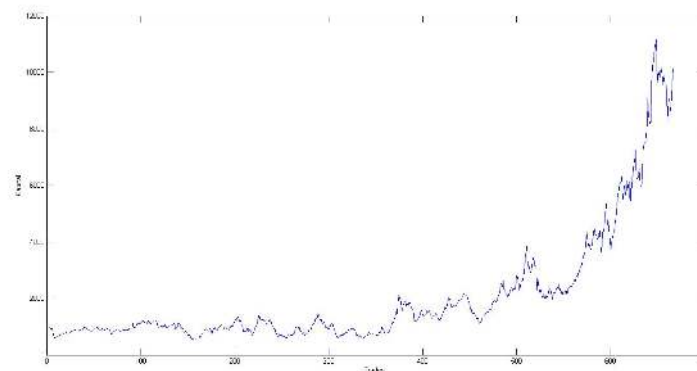


Abbildung 2.9: Verlauf des Kontostandes im Referenzraum bei Anwendung der Repetitive-Bull-Strategy. Es ergibt sich eine ca. Verzehnfachung des Anfangskapitals von 1000 Dollar über den gesamten Referenzzeitraum. Man schlägt also das 3%-Sparbuch um den Faktor 2.

Überprüft man diese Strategie nun allerdings im Testzeitraum, so führt man 7 Trades durch, von denen nur einer mit einem Gewinn endete, und man endet ausgehend von einem Startkapital von 1000 Dollar bei 650 Dollar - also einem Verlust von 35% (Abbildung 2.10). Diese Strategie ist somit ein Beispiel dafür, dass zwar Techniken in der Vergangenheit sehr erfolgreich sind, im Testzeitraum (siehe Einleitung) allerdings nicht funktionieren.

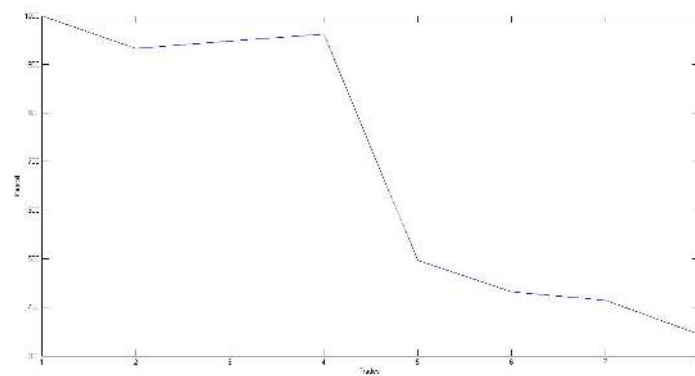


Abbildung 2.10: Verlauf des Kontostandes nach der Repetitive-Bull-Strategy für den Testzeitraum, abgebildet nach Trades. Im Jahr 2015 werden also nur 7 Trades mit dieser Strategie ausgeführt, von denen jedoch nur ein einziger erfolgreich ist.

Kapitel 3

Hoch- und Tiefpunkte Strategie

Ein Tief- bzw. Hochpunkt eines Kurses ist definiert als der tiefste bzw. höchste Wert in einer Umgebung von n Tagen um diesen Tag herum. Für $n = 14$ sind alle Tiefpunkte im Referenzzeitraum in Abbildung 3.1 dargestellt.

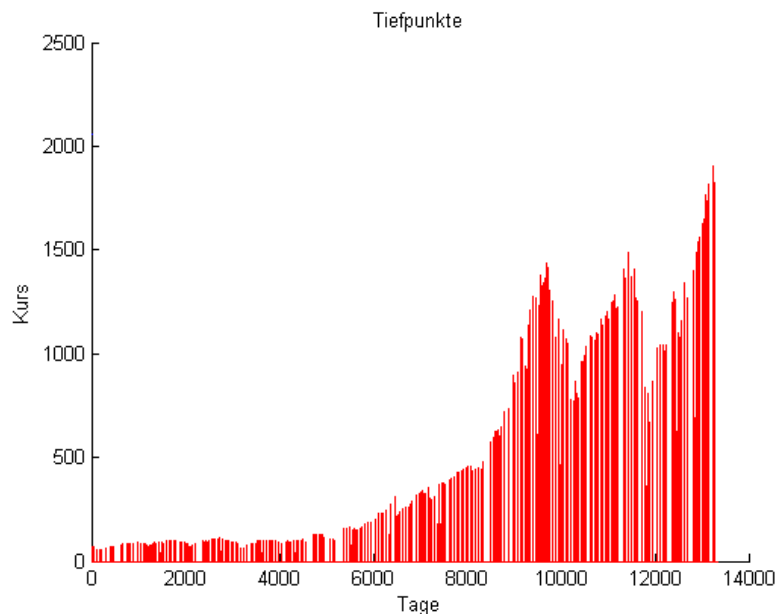


Abbildung 3.1: Tiefpunkte von 1962 bis 2014 mit einem Abstand um 14 Tagen

In den hier verwendeten Strategien wird meist diese Umgebung auf entweder 7 oder 14 gesetzt. Um solche Extrempunkte zu bestimmen, wird ein Vektor gebildet, in dem $2n + 1$ Elementen von n Kurswerten vorher und n Kurswerte nach dem möglichen Extrempunkt gespeichert werden. Falls ein höherer bzw. tieferer Punkt sich in diesem Vektor befindet, wird der Vektor so verschoben, dass dieser Punkt der neue Mittelpunkt des Vektors ist. Um den Gewinn zu maximieren, ist außerdem interessant, wie lange nach einem Tiefpunkt bzw. Hochpunkt, der Kurs steigt oder fällt. Für beide Punkte ergibt sich ein Durchschnitt von ca. 3 Tagen. Somit weiß ein Händler, dass er, falls einen Tiefpunkt vorliegt, mit 3 Tagen Wachstum rechnen darf. Jedoch weiß der Händler nicht, ob ein Tiefpunkt gerade geschehen ist. Deswegen müssen Muster erkannt werden, die einen Extrempunkt ankündigen.

Eine Möglichkeit einen potenziellen Tiefpunkt zu bestimmen ist es, die Bewegungen des Kurses in den Tagen vor allen bekannten Tiefpunkten zu beobachten und diese dann übereinanderlegen und somit ein gemittelttes Muster zu erhalten, nach dem sich die Kurse normalerweise vor einem Tiefpunkt bewegen.

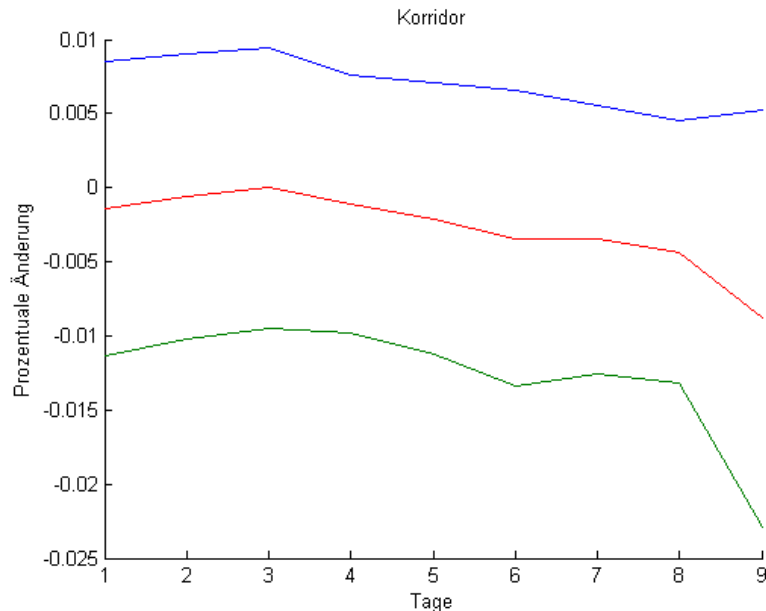


Abbildung 3.2: Durchschnittlicher Korridor zehn Tage vor einem Tiefpunkt.

In Abbildung 3.2 sieht man, wie sich der Kurs zehn Tage vor dem Tiefpunkt durchschnittlich verhält, wobei zum Durchschnitt die Standardabweichung addiert oder subtrahiert wird. Dadurch entsteht ein Korridor, in dem sich der Kurs vor einem Tiefpunkte aufhält. Analog kann auch man mit den Hochpunkten verfahren.

3.1 Corridor-Strategie

Bei der Corridor-Strategie verwendet man genau diesen Korridor, der ein Tiefpunkt bzw. Hochpunkt definieren soll. Da nach einem Tiefpunkt die Aktien durchschnittlich drei Tage steigen, bzw. nach einem Hochpunkt fallen werden, setzt man nach einem potentiellen Extrempunkt auf long bzw. short und steigt nach drei Tagen aus. Falls sich der Händler für eine der beiden Optionen entscheidet, investiert er sein ganzes Geld in seine Position. Ein Problem bei dieser Strategie ist allerdings, dass der Korridor relativ allgemein gefasst ist, und somit nur 6.17% der potentiellen Tiefpunkte reale Tiefpunkte und nur 5.64% der potentiellen Hochpunkte auch reale Hochpunkte sind, was in einem relativ kleinen Gewinn resultiert (Abbildung 3.3).

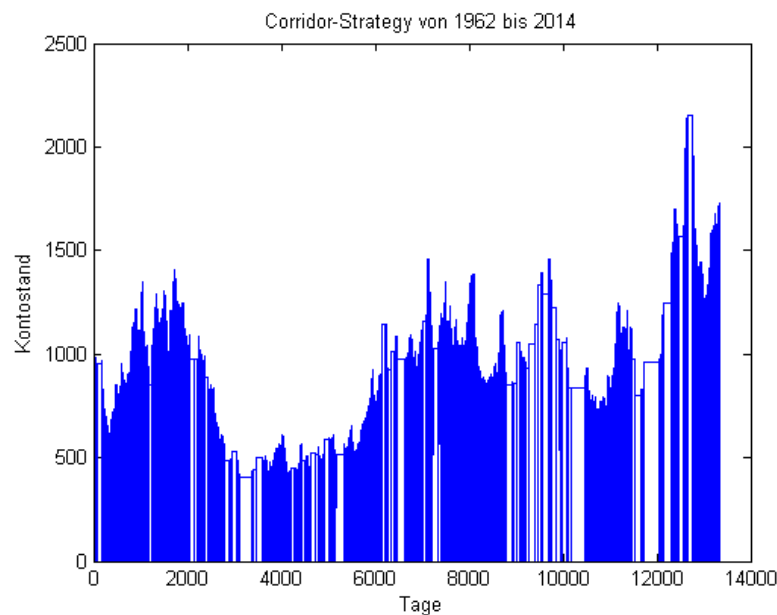


Abbildung 3.3: Verlauf des Kontostandes mit einem Startkapital von 1000\$ mit der Corridor-Strategie im Referenzzeitraum.

In den zwei Diagrammen sieht man den Verlauf des Kontostandes im jeweiligen Referenzzeitraum mit einem Startkapital von 1000\$ und mit einem Hebel von 5. Im Jahr 2015 hätte man mit dieser Strategie sein Kapital um 20% vermehrt (Abbildung 3.4). Im letzteren Fall sind die Transaktionskosten von 13 \$ pro Trade mit berücksichtigt.

3.2 Max/Min-Layer-Trap-Strategie

Ähnlich wie bei der Corridor-Strategie werden bei der Max/Min-Layer-Trap-Strategie bei den Hoch- bzw. Tiefpunkten Short- bzw. Longpositionen eingegangen. Bei jedem Trade wird wieder das ganze Kapital investiert. Der Unterschied zur Corridor-Strategie liegt darin, dass die Positionen nicht nach drei Tagen verkauft werden, sondern sie werden erst gekauft, wenn die Positionen aus einem Bereich von 1% um den Extrempunkt fallen bzw. steigen. Da bei einem Tiefpunkt eine 92% Wahrscheinlichkeit besteht, dass falls er einmal in diesem Bereich hineinfällt, diesen nach oben hin verlässt. Bei Hochpunkten ist es ein 90% Wahrscheinlichkeit, dass der Kurs den Bereich fallend verlässt. Um Tiefpunkte zu erkennen, wird die Methode aus der Corridor-Strategie verwendet. Sobald der Auslöser eines Hochs bzw. Tiefs nach der Corridor-Strategie aktiviert wird, wird noch zwei weitere Tage gewartet und wenn der Kurse während dieser Zeit den Extrempunkt nicht um ein Prozent verlassen hat, so wird zur Eröffnung des dritten Tages eine Short- bzw. Longposition eingegangen. Erstaunlicherweise verhalten sich die so erkannten Punkte in der obigen Eigenschaft ähnlich wie echte Extremstellen. Das resultiert in einem deutlich größeren Gewinn. Das Startkapital von 1000\$ konnte innerhalb von 62 Jahre auf 26 000 000\$ erhöht werden (Abbildung 3.5), mit einer eher geringen Anzahl an abgeschlossenen Trades von 630.

Im Jahr 2015 konnte aufgrund der hohen Volatilität mit dieser Strategie nie gehandelt werden, weil die Kurse nie lange genug in dem Bereich um den Extrempunkt blieben.

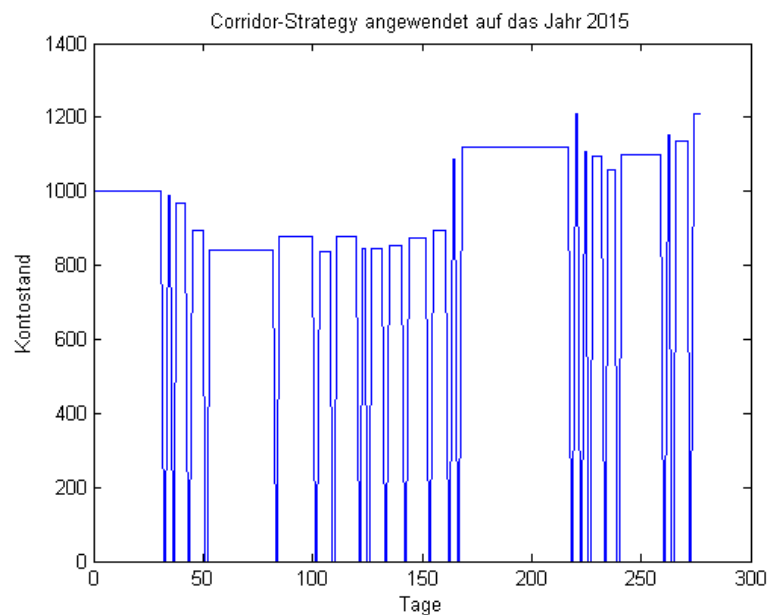


Abbildung 3.4: Verlauf des Kontostandes mit einem Startkapital von 1000\$ mit der Corridor-Strategie im Testzeitraum, Anfang 2015 bis 8. Februar 2016.

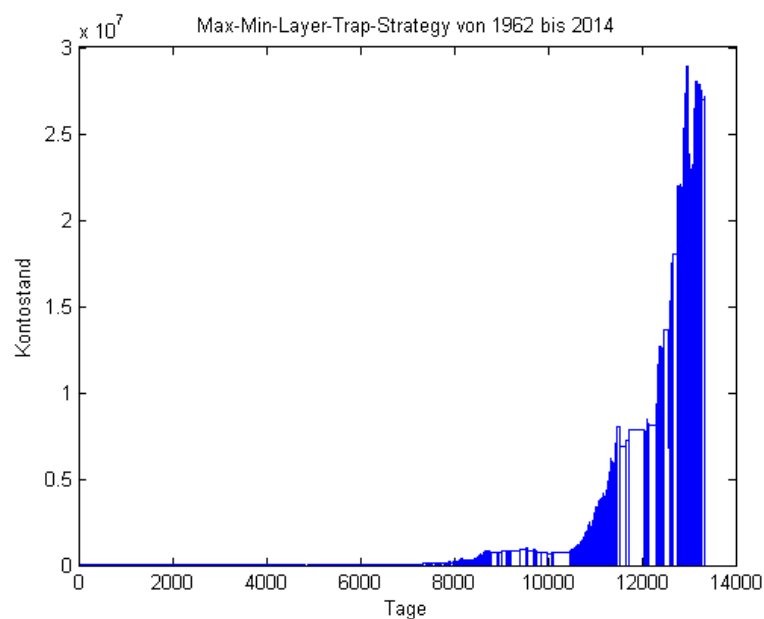


Abbildung 3.5: Kontostand des Händlers mit einem Startkapital von 1000\$ der nach der Max/Min-Layer-Trap-Strategie handelt im Referenzzeitraum resultiert in 26 000 000. Diese Strategie hat das Kapital damit in diesem Zeitraum versechundzwanzigtausendfacht und schlägt damit selbst den Markt um ca. den Faktor 1000.

3.3 Long-wins-everything-Strategie

Bei dieser Handelsstrategie geht es in erster Linie darum, den Markt gut zu beobachten. Wenn man einen Tiefpunkt erkennt und auf diesen Tiefpunkt ein Hochpunkt folgt, wartet

man darauf, dass der Kurs wieder zu einem Tiefpunkt absinkt. Tiefpunkte sind bei dieser Strategie als Punkte definiert, an denen der Wert kleiner ist als an allen 7 Handelstagen vor dem vermeintlichen Tiefpunkt und an allen 7 Handelstagen nach diesem Zeitpunkt. Die Definition eines Hochpunktes lautet beinahe gleich. Allerdings muss der Wert des Marktes bei einem Hochpunkt größer sein als an allen 7 Tagen davor und danach. Wenn man nun die Konstellation Tiefpunkt-Hochpunkt-Tiefpunkt erkannt hat, kauft man am siebenten Tag nach dem Erreichen des zweiten Tiefpunkts Aktien zum Tagesöffnungskurs. Der Grund, warum man sieben Tage warten muss, um zu investieren, liegt ganz einfach darin, dass man aufgrund der Definition eines Tiefpunkts erst 7 Tage nach Erreichen eines Tiefpunkts mit Sicherheit sagen kann, dass ein solcher vorliegt. Der Anteilschein wird erst wieder verkauft, wenn die Aktie einen gewissen Wert angenommen hat. Dieser gewünschte Verkaufswert ist dann erreicht, wenn der momentane Marktwert größer ist als die Summe aus dem Marktwert zur Zeit des ersten Tiefpunkts und der zweifachen Differenz zwischen dem Marktwert zur Zeit des ersten Hochpunkts und zur Zeit des ersten Tiefpunkts. Da es möglich ist, dass eine neue Tiefpunkt-Hochpunkt-Tiefpunkt-Konstellation erreicht wird, bevor man die zuvor gekauften Aktien verkauft, wird das Investitionskapital bei jedem neuen Kauf halbiert. Falls eine Aktie verkauft wird, erhöht sich das Investitionskapital um den Erlös aus dem Verkauf der Aktie. Deshalb kann es sein, dass das zur Verfügung stehende Kapital bei manchen Investitionen größer ist als bei anderen. In der Grafik ist das verfügbare Kapital für jeden Aktienankauf dargestellt, das sich circa zwischen 30\$ und 1,5 Millionen Dollar bewegt. Weiters wird das Investitionskapital, falls dieses unter 30\$ sinkt, verdoppelt, damit die Möglichkeit eines negativen Kapitals, das durch die Transaktionskosten in Höhe von 13\$ pro erworbener Aktie erreicht werden könnte, ausgeschlossen wird. Wenn man die Long wins everything-Strategie seit Jänner 1962 mit einem Hebel von 5 und einem Grundkapital von 1000\$ angewandt hätte, würde man den Referenzdaten zufolge Ende Dezember 2014 7.078.500\$ besitzen (Abbildung 3.6). Diese Strategie angewendet auf den Testzeitraum würde zum Ende noch 8 offene Positionen besitzen. Deswegen ist es für diese Strategie nicht möglich, eine belastbare Aussage für den Testzeitraum zu treffen.

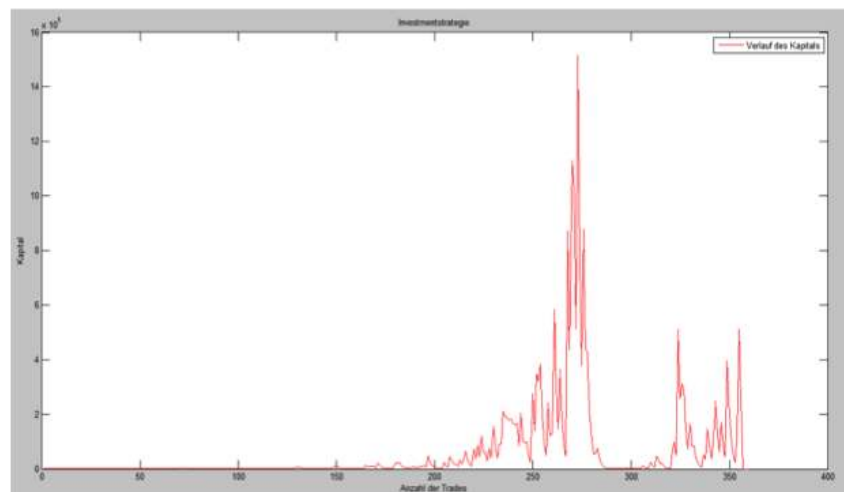


Abbildung 3.6: Verlauf des Kontostandes mit einem Startkapital von 1000\$ mit der Long wins everything-Strategie. Zum Ende ist das gesamte Kapital in den Markt investiert und bei Glattstellen aller Position würde der Kontostand ca. 7 000 000 betragen. Diese Strategie versiebenhundertfacht also das anfängliche Investitionskapital.

3.4 Similar Triangles-Strategie

Ähnlich wie bei der Long-wins-everything-Strategie, muss man bei der Dreiecksstrategie den Markt beobachten und nach Tief- und Hochpunkten suchen. Das Hauptaugenmerk liegt dabei auf dem Auffinden von Tiefpunkt-Hochpunkt-Tiefpunkt-Konstellationen und der Berechnung des Flächeninhalts des Dreiecks, das man erhält, wenn man die drei Punkte einer solchen Konstellation durch Strecken verbindet.

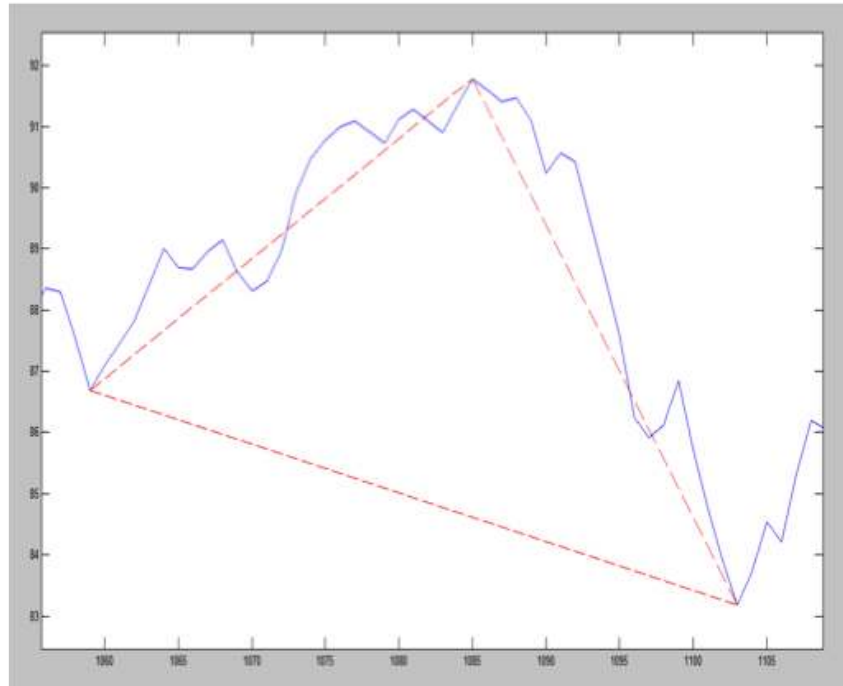


Abbildung 3.7: Mögliche Dreiecksformation

In der Grafik (Abbildung 3.7) ist eine Tief-Hoch-Tiefpunkt-Konstellation mit den eingezeichneten Verbindungsstrecken dargestellt, um die beschriebene Vorgehensweise zu veranschaulichen. In weiterer Folge versucht man Verhältnisse zwischen den Dreiecksflächen mehrerer Tief-Hoch-Tiefpunkt-Konstellationen oder die Länge der einzelnen Seiten jeder Dreiecksfläche und die Fläche des dazugehörigen Dreiecks zu finden. Allerdings war im Zeitraum 1962 bis 2014 keine Regelmäßigkeit hinter diesen Verhältnissen erkennbar, weshalb kein Anhaltspunkt für die Durchführung der Dreiecksstrategie gefunden werden konnte.

Kapitel 4

Historische Strategien

4.1 Livermore-Strategie

Auch die berühmte Strategie von Livermore, mit der er zwischen 1907 und 1929 enorme Gewinne einfahren konnte, wurde analysiert. Laut der Strategie von Livermore sollte man Longpositionen eingehen, wenn das Tief des vorherigen Tages über dem Tief des aktuellen Tages ist und der Schlusskurs vom aktuellen Tag über dem Schlusskurs des vorherigen Tages. In diesem Fall haben die Kurse zu seiner Zeit häufig zu steigen begonnen. Falls das Hoch des vorherigen Tages unter dem Hoch des aktuellen Tages und außerdem der Schlusskurs von gestern höher ist als der Schlusskurs von heute, sollte man Shortpositionen eingehen, da hier der Kurs laut Livermore fallen würde. Wenn man diese Strategie auf die Jahre zwischen 1962 und 2014 anwendet, erhält man folgende Entwicklung eines Startkapitals von 1000\$ (Abbildung 4.1).

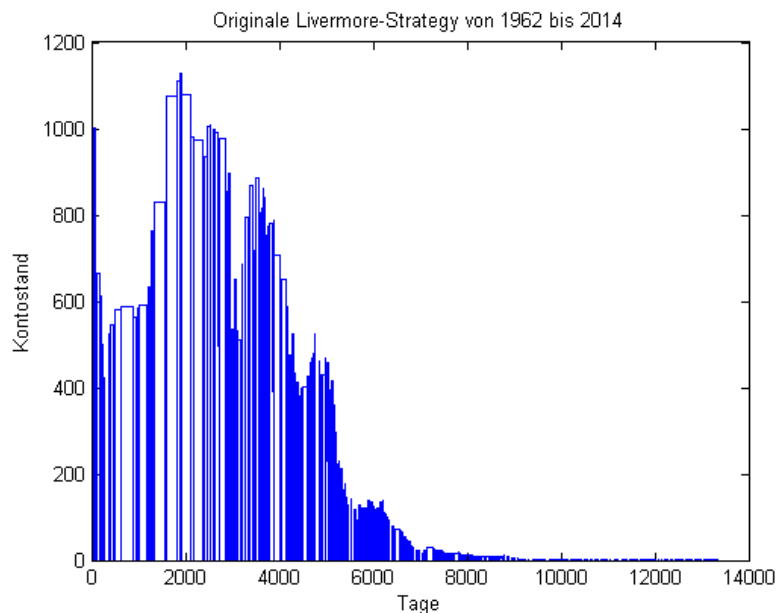


Abbildung 4.1: Entwicklung des Startkapitals von 1000\$ nach der originalen Livermore-Strategie im Referenzzeitraum. Das befolgen dieser Strategie würde zur Pleite führen.

Wie man sieht, hat sich der Markt so angepasst, dass diese Strategie nicht mehr gewinn-

bringend ist. Falls man jedoch die Bedingungen um short bzw. long zu gehen umkehrt, erhält man die Entwicklung in Abbildung 4.2.

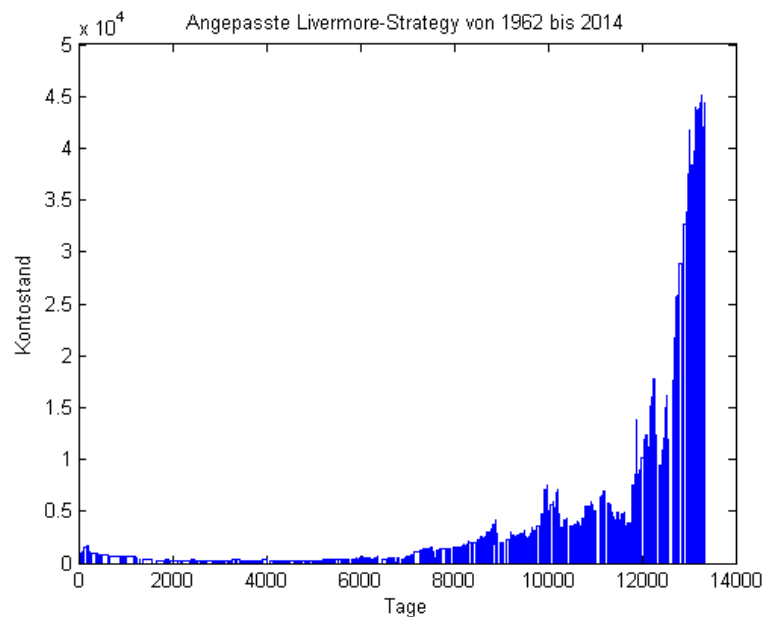


Abbildung 4.2: Angepasste Livermore-Strategie im Referenzzeitraum mit einem Startkapital von 1000\$.

Offenbar hat der Markt die originale Strategie von Livermore überkompensiert. Bei beiden Grafen wurde mit einem Hebel von 3.5 gehandelt bei einem Startkapital von 1000\$. Wenn man die angepasste Strategie von Livermore auf das Jahr 2015 anwendet erhält man folgende die Entwicklung aus Abbildung 4.3. Falls bei einem Tag der Kontostand 0 erreicht wird, hat der Händler sein ganzes Geld im Markt investiert.

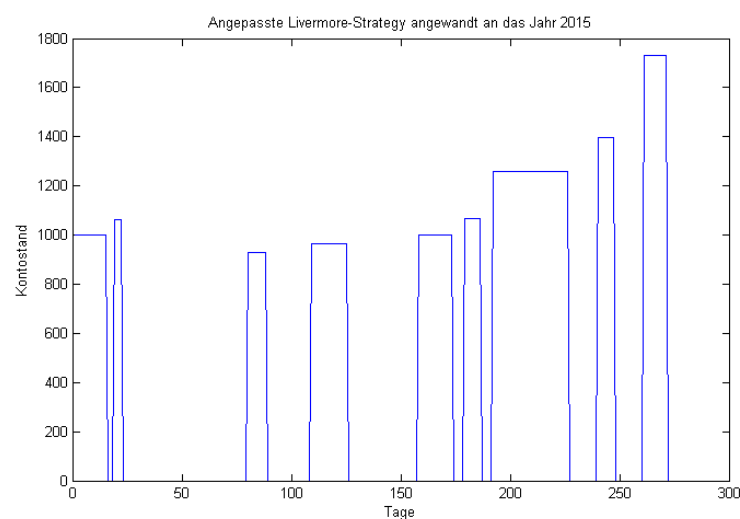


Abbildung 4.3: Angepasste Livermore-Strategie im Testraum mit einem Startkapital von 1000\$ erwirtschaftet einen Gewinn von über 70%.

Am Ende des Testjahres 2015 hätte man sein Vermögen von 1000\$ auf 1729\$ vervielfacht. Auch hier wurde mit dem Hebel 3.5 gehandelt.

4.2 W-Formation-Strategie

Mit der W-Formation-Strategie wird die Strategie bezeichnet, bei der man auf einen Verlauf des Kurses hofft, der einem W gleicht, d.h. er sinkt zuerst stark ab, erholt sich leicht, sinkt wieder leicht ab und steigt schließlich wieder stark an, um zirka das Ausgangsniveau wieder zu erreichen. Um mit dieser Strategie arbeiten zu können, wurden alle Tiefpunkte mit einem Mindestabstand von 2 Tagen zum nächsten Tiefpunkt gesammelt. Falls vor einem solchen Tiefpunkt der Kurs um mehr als zwei Tagen gefallen ist und zwischen diesem und dem nächsten Tiefpunkt, der nur in einem 1% Bereich um den ersten Tiefpunkt liegen darf, der Kurs um mindestens 1.5% gestiegen ist und nach dem zweiten Tiefpunkt zwei Tage gestiegen ist, würde das nach diesem Modell signalisieren, dass man sich in einer W-Formation befindet und somit auf long setzten müsste. Diese Position würde solange behaltet werden bis der Kurs insgesamt jenes Niveau erreicht, von dem er vor dem ersten Tiefpunkt gefallen ist. Ein mögliche W-Formation könnte folgendermaßen aussehen (Abbildung 4.4).

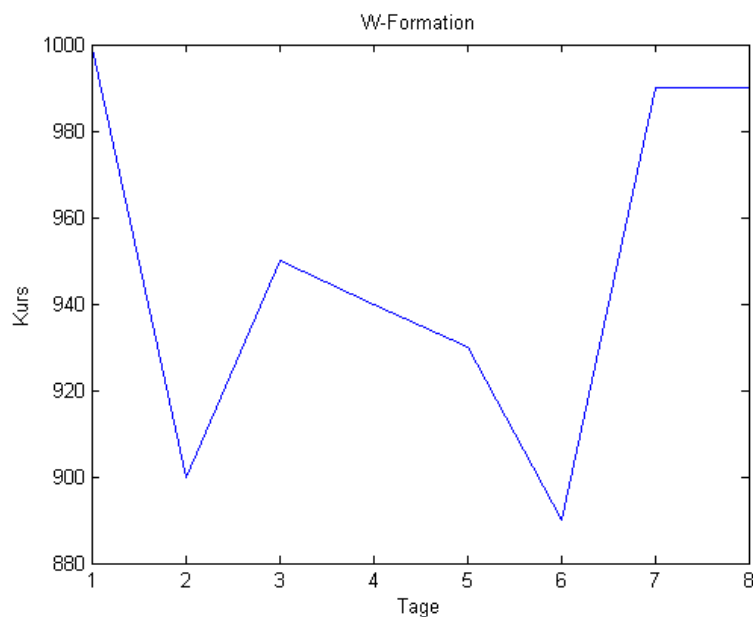


Abbildung 4.4: Mögliche W-Formation

Diese Strategie funktioniert nur mäßig, da der Kurs, falls er obige Kriterien erfüllt, genauso wahrscheinlich nach den zwei Tagen fällt, wie er steigt. Außerdem kann nur eine relativ kleine Anzahl von Trades abgeschlossen werden. Falls der Händler mit 1000\$ beginnt, wird er sein Kapital nach 53 Jahren um 34% auf 1660\$ steigen können.

4.3 Regressionsgeraden

Um den derzeitigen Trend des Marktes feststellen zu können, ist es von Vorteil nicht nur die letzten Tage genauer unter die Lupe zu nehmen, sondern auch einen größeren Zeitraum. Einen Zeitraum, in dem der Kurs lange nicht mehr unter einen Punkt gefallen, oder schon lange kein Hoch mehr erreicht hat.

Durch die Bestimmung der Hoch- und Tiefpunkte innerhalb von mehreren Tagen, ist es möglich den genauen Verlauf zwischen zwei aufeinanderfolgenden Extremstellen zu betrachten. Dazu ist es sinnvoll ein Tagesmittel des Kurses für jeden Tag festzulegen. Dieses setzt sich aus Eröffnungs-, Höchst-, Tiefst- und Abschlusskurs zusammen.

Aufgrund der möglicherweise breiten räumlichen Verteilung dieser erhaltenen Punktwolke bietet es sich an, eine lineare Regressionsgerade durch sie zu legen. Eine Regressionsgerade ist eine Gerade, welche die Datenpunkte möglichst gut approximiert. Dazu wird von jedem Punkt das Lot auf die Gerade gefällt und quadriert. Diese Zahlen werden nun summiert. Das Ziel ist es diese Summe möglichst klein zu halten. Wenn sie ihr Minimum erreicht, hat man die lineare Regressionsgerade gefunden.

Somit kann der Trend bis zur nächsten Extremstelle festgestellt werden. Ist die Steigung der Geraden positiv, so wird eine Longposition eingegangen. Ist sie negativ, so setzt man auf fallende Kurse. Da man allerdings nicht in die Zukunft sehen kann, ist es wichtig den Trend so früh wie möglich festzustellen und zu wissen, wie lange der Trend ungefähr anhalten wird.

Deshalb ist zu untersuchen, wie lange ein Trend durchschnittlich hält. Dann bekommt man abzüglich der Standardabweichung einen Zeitraum, nach dem es ratsam ist, wieder auszusteigen. Da es sehr lange dauert bis man den tatsächlichen Trend erkennt, gibt es nur sehr selten Fälle, in denen man die Aktien über mehrere Tage lang hält, sodass sich der Trend stark genug auf den Kurs auswirken könnte. Deshalb werden nur sehr selten eher geringere Gewinne erzielt. Des Weiteren sind keine offensichtlichen Zusammenhänge zwischen Längen des Trend beziehungsweise deren Steigungen ersichtlich.

4.4 Labourmarket-Strategie

Am ersten Freitag in jedem Monat werden die US-amerikanischen Arbeitsmarktdaten bekannt gegeben, die einen erheblichen Einfluss auf das Geschehen an der Börse haben. In der Abbildung 4.5 sieht man, wie sich der Kurs prozentuell ändert. Dabei ist der sechste Tag jener Freitag, an dem die Arbeitsmarktdaten bekannt gegeben werden.

Um den allgemeinen Kursverlauf in den fünf Tagen vor dem Freitag zu bestimmen, bildet man für die ersten fünf Punkte in obiger Grafik (Abbildung 4.5) die Regressionsgerade. Falls die Steigung dieser Gerade größer als -1% ist, geht man vom Freitag bis zum nächsten Freitag eine Longposition ein. Falls dies nicht der Fall sein sollte, geht man vom übernächsten Montag bis zum darauffolgenden Dienstag, den sechsten und siebten Handelstag nach dem Freitag eine Longpositionen ein. Im Jahr 2015 konnte mit dieser Strategie ein Gewinn von 21.5% erzielt werden, da von den 12 möglichen Trades, 9 positiv abgeschlossen werden. Zwischen 1962 und 2014 konnte der Gewinn von 1000\$ auf 115000\$ (auf das 115-fache) vermehrt werden. Die Anzahl der positiven Trades beträgt für diesen Fall 58.5%.

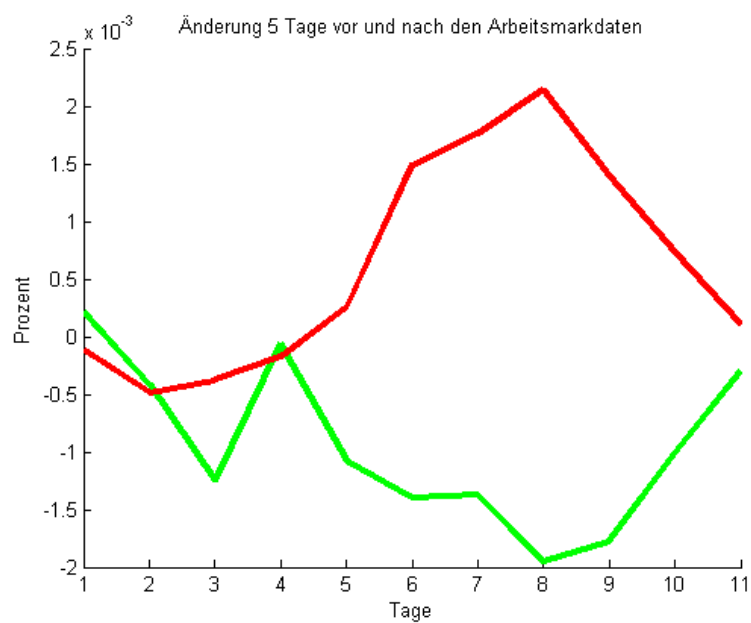


Abbildung 4.5: Durchschnittliche relative Kursänderungen vor und nach Bekanntgabe der Arbeitsmarktdaten gemittelt über alle Monate des Referenzzeitraums. Tag sechs ist der Freitag, an dem die Ankündigung stattfindet.

Kapitel 5

Fourierreihen

Oft können scheinbar zufällig angeordnete Daten durch Fourierreihen erstaunlich genau approximiert werden. Diese können dann für eine Extrapolation für kleine zukünftige Zeiträume genutzt werden. Daher liegt der Gedanke nicht fern, die Kursdaten durch solche Fourierreihen zu beschreiben und vorauszusagen.

Allgemein besitzen Fourierreihen die Form

$$f(x) = \sum_{i=1}^n \alpha_i \sin(\omega_i x) + \beta_i \cos(\omega_i x).$$

Je größer n , desto genauer wird die Fourierreihe an den Datensatz angeschmiegt. Um alle α_i , β_i und ω_i zu bestimmen, muss man einerseits die Fouriertransformation der Datensätze betrachten und die Frequenzen mit der höchsten Energie ausfindig machen und dann mithilfe dieser und $2n$ Punkte ein Interpolationsgleichungssystem lösen. Für den Probefall wird $n = 10$ gewählt. Die Fouriertransformation im relevanten Bereich ist in Abbildung 5.1 dargestellt.

Die ω s mit den höchsten Energien sind 0.0005, 0.0010, 0.0055, 0.0015, 0.0050, 0.0030, 0.0035, 0.0025, 0.0040, 0.0020, 0.0045. Als 20 Stützpunkte $t_1, \dots, t_{20}$, auf denen das Gleichungssystem errichtet wird, um alle α und β zu bestimmen, werden 10 zufällig gewählte Hoch- und 10 zufällige gewählte Tiefpunkte gewählt, da diese sehr viel mehr Information über den Datensatz tragen, als ein beliebiger Punkt. Um dieses System zu lösen, muss zuerst eine 20×20 Matrix C aufgestellt werden, mit $C_{ij} = \sin(t_j * \omega_i)$ und $C_{i+1j} = \cos(t_j * \omega_i)$ und ein Vektor γ in dem die Kurswerte der Stützpunkte gespeichert werden. Indem man $C^{-1}\gamma$ bildet, bekommt man einen Vektor, in dem alle α_i und β_j eingetragen sind. Durch diese erhält man dann eine Fourierreihe, die im Vergleich zu den Originaldaten in Abbildung 5.2 dargestellt ist.

Damit die Daten des Kurses mit der Kurve der Fourierreihe verglichen werden können, muss die Fourier-Reihe $f(t)$ zu $|\frac{f(t)}{100}| + 0.07t$ transformiert werden, wobei t der Tag ist. Wie man sieht, kommt diese Kurve nicht wirklich an die realen Kurswerte heran. Besser wäre es gewesen als Interpolationspolynom eine um eine lineare Funktion gestörte Fourier-Reihe zu wählen und deren zusätzlichen Parameter zu bestimmen. Ein weitere Steigerung der Stützpunkte wurde nichts bringen, da mit zunehmender Größe der Matrix C numerische Ungenauigkeiten beim Invertieren überwiegen.

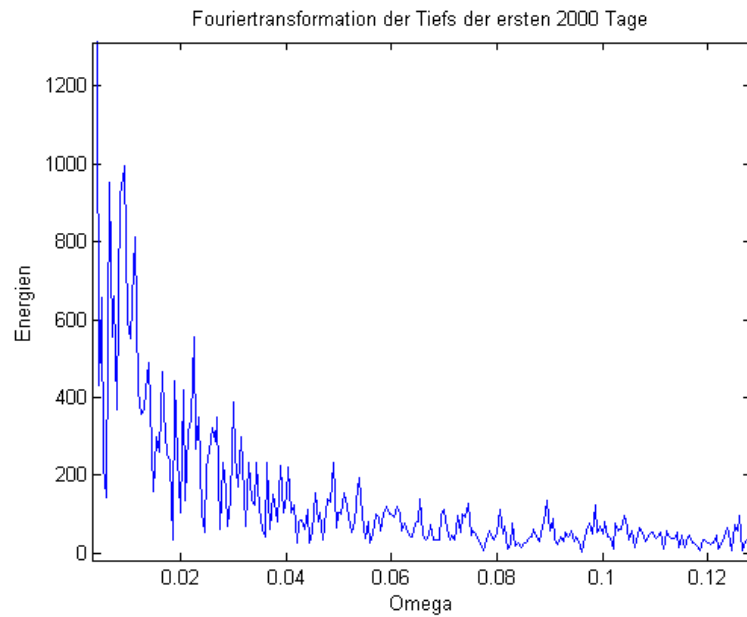


Abbildung 5.1: Fouriertransformation über den gesamten Referenzzeitraum von 13342 Handesltagen. Die niederfrequenten Anteile besitzen im Kurs die höchste Energie.

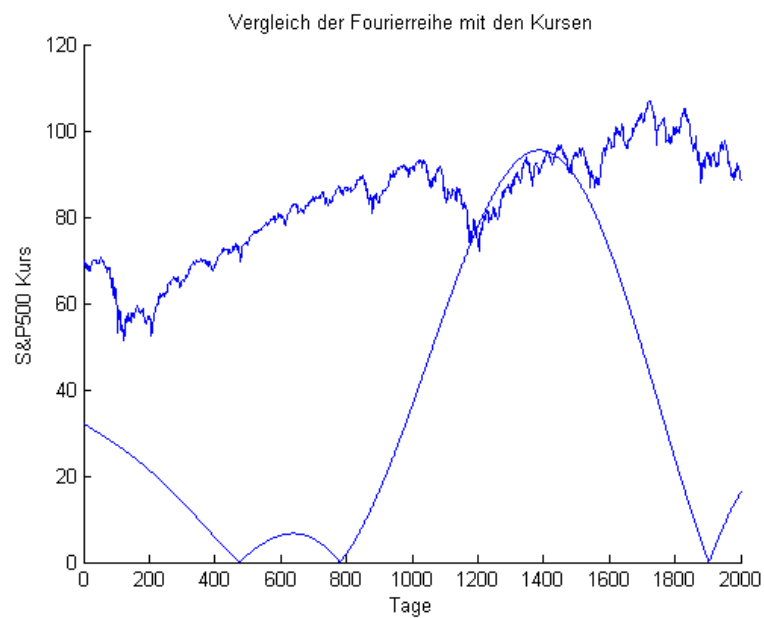


Abbildung 5.2: Vergleich der wirklichen Datensätze mit der aufgestellten Fourierreihe.

Kapitel 6

Perceptron

Doch eine Trading-Strategie muss keinesfalls ein immer gleichbleibendes Konstrukt von verarbeitenden Daten sein. Sie kann sich durchaus auch verändern, sich selbst immer wieder erneuern, bis eine nahezu optimale Verarbeitung der Daten möglich ist. Eine Variante dieses zu erreichen, ist ein neuronales Netz aufzubauen, das im einfachsten Fall ein Perceptron ist. Diese Strategie wird als Perceptron-Strategie bezeichnet. Aufgrund von Inputdaten wird ein Output generiert, der mit dem echten Output verglichen wird. Sind diese voneinander verschieden, werden sogenannte Gewichte derart angepasst, dass sich der Fehler verringert. Führt man diesen Prozess nun wiederholte Male durch, nähern sich diese Gewichte jenem Optimum an, sodass für jeden Input der richtige Output errechnet wird. Dies wird schließlich verwendet, um die Zukunft vorauszusagen.

Will man nun den zukünftigen Kurs berechnen, so bietet sich an, im Referenzzeitraum fünf aufeinanderfolgende Eröffnungspreise als Input festzulegen. Der echte Output ist der Startkurs des nächsten Tages. Mit fünf Gewichten, die zu Anfang auf eine zufälligen Zahl zwischen -1 und 1 gesetzt werden, wird nun ein Output berechnet, der mit dem echten Wert verglichen wird.

Nachdem diese Verbesserung circa 150.000 Mal erfolgt ist, ergeben sich zukünftige Werte, die den echten sehr nahe kommen. Die dadurch entstandenen Gewichte wurden nun auf den Testzeitraum, das Jahr 2015, angewandt. Die Abbildung 6.1 ganz in blau zeigt die tatsächlichen Werte des Jahres 2015, der rote Graph in der Abbildung 6.2 zeigt die errechneten Werte für das Jahr, wobei der blaue Teil davor das Ende des Jahres 2014 darstellt.

Da dies sehr vielversprechende Ergebnisse liefert, liegt es auf der Hand diese Perceptron-Strategie auch anzuwenden. Hierbei wird unter zwei Varianten unterschieden: Die Perceptron-Long-Strategie und die Perceptron-Short-Long-Strategie.

6.1 Perceptron-Long-Strategie

Bei der Perceptron-Long-Strategie wird aufgrund der berechneten Gewichte und der Eröffnungskurse der letzten fünf Tage der Kurs des nächsten Tages berechnet. Ist dieser höher als der heutige Eröffnungskurs, so wird das gesamte Kapital in eine Longposition investiert. Diese werden am nächsten Tag zur Öffnung der Börse verkauft. Da in dieser Simulation mit einem Kleininvestor gerechnet wird, liegt das Startkapital bei 1.000\$. Innerhalb eines Jahres, des Testzeitraums, steigt sein Kapital auf 120.000\$, was einer 120fachen Steigerung entspricht (Abbildung 6.3). Somit zeigt sich, dass dies eine sehr gewinnbringende Strategie ist, die sogar in nur einem Jahr sehr hohe Gewinne erzielt.

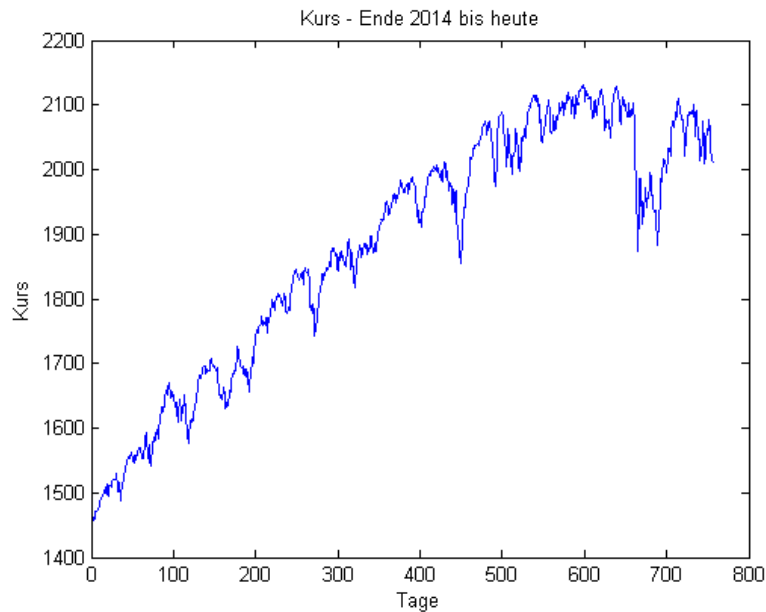


Abbildung 6.1: Realer Kursverlauf des S&P 500 von Ende 2012 bis zum 8. Februar 2016, an insgesamt knapp 800 Handelstagen.

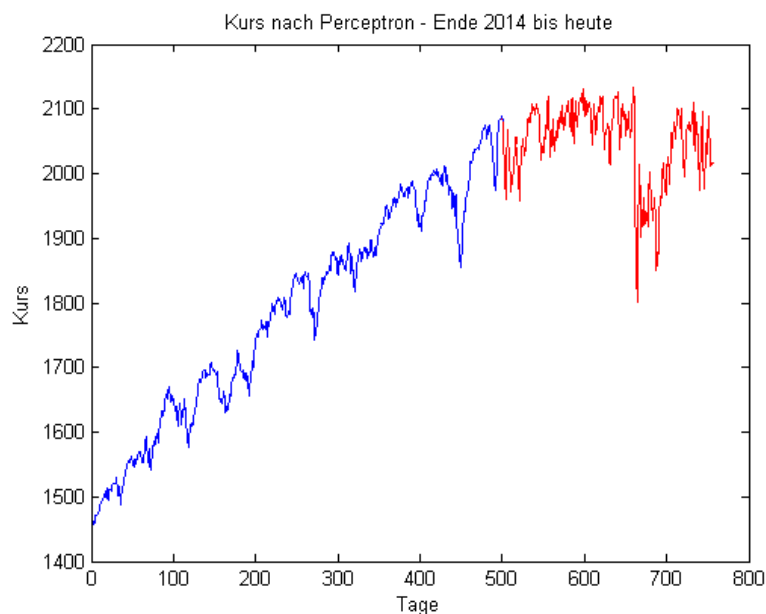


Abbildung 6.2: Durch das Perceptron prognostizierter Kursverlauf des S&P 500 für das Jahr 2015 bis zum 8. Februar 2016 in rot, an insgesamt 277 Handelstage. Dieser Verlauf kommt dem realen Verlauf aus Abbildung 6.1 qualitativ sehr nahe.

6.2 Perceptron-Short-Long-Strategie

Die Perceptron-Short-Long-Strategie funktioniert sehr ähnlich, allerdings werden bei prognostizierten fallenden Kursen Shortpositionen eingegangen, die genau bis zum nächsten Eröffnungskurs gehalten werden. Dieses passiert, wenn der für den nächsten Tag vom

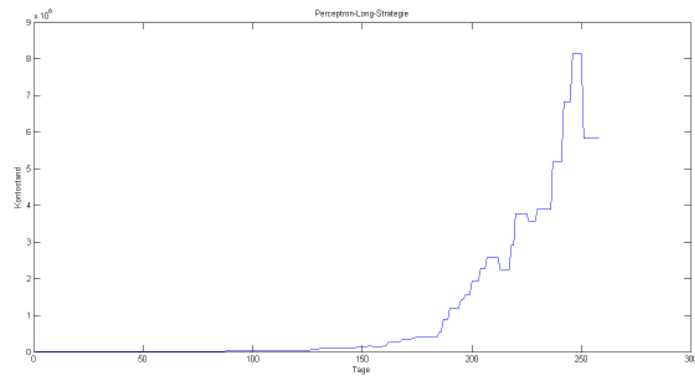


Abbildung 6.3: Kontoverlauf mit einem Startkapital von 1000\$ im Testzeitraum mit Perceptron-Long-Strategie. Das Kapital wird 120fach.

Perceptron prognostizierte Eröffnungskurs unter dem heutigen Eröffnungskurs liegt. Des Weiteren wurde in dieser Simulation die Gebühr für den Börsenmakler, der einem die Aktien verschafft, mit einbezogen. Dieses ist ein Pauschalbetrag von 13\$ pro Transaktion. Da es sehr unwahrscheinlich ist, um den genauen Eröffnungspreis Aktien zu erlangen, wurde auch noch eine zufällige Abweichung von -3 bis +3 beim Eröffnungskurs in die Simulation hinzugefügt, die die Abweichung des gekauften Kurses von dem Eröffnungskurs beschreibt. Da man nun bei mehrmaligem Ausführen der Simulation unterschiedliche Ergebnisse pro Ausführung erhält, erweist es sich als sinnvoll die Monte-Carlo Methode anzuwenden. Es werden mehrere Beispiele gemacht und für jeden erhaltenen Wert wird der Durchschnitt gebildet. Dadurch entsteht ein eindeutiger Verlauf des Kapitals, wobei man nicht genau vorhersagen kann wie viel Geld man durch Anwendung dieser Strategie bekommt, aber zumindest einen ungefähren Richtwert. Das Startkapital betrug wie zuvor 1.000\$.

Die entstandene Grafik zeigt alle 100 Beispiele die gemacht wurden, gezeichnet in einer zufälligen Farbe, der durchschnittliche Kontoverlauf wird durch die doppelt so dicke schwarze Linie angegeben (Abbildung 6.4).

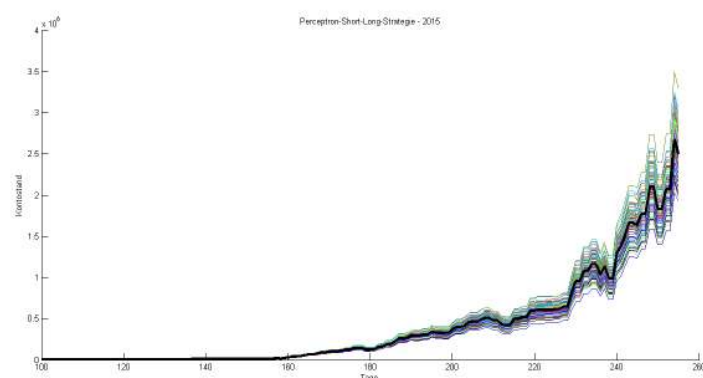


Abbildung 6.4: Kontoverlauf mit einem Startkapital 1000\$ im Testzeitraum mit der Perceptron-Short-Long-Strategie. Im Mittel hat sich das Kapital verdreitausendfacht. Transaktionsgebühren sind mit einberechnet. Allerdings keine Kapitalertragssteuern.

Mit der Perceptron-Short-Long-Strategie wurden rund 2.5 Millionen\$ Endkapital erzielt. Dies ist eine der erfolgreichsten Trading-Strategien, da man die erzielten Werte mit einem noch längeren Trainingszeitraum weiter verbessern kann.

6.3 Fourier-High-Energy-Frequenzy-Perceptron-Strategy

Da die Furientransformation über den gesamten Zeitraum der Daten eine hohe Energie für die Frequenz von 2000 Tagen liefert – sprich der zugrundeliegende Kurs besitzt Oszillationen in der Frequenz von 2000 Handelstagen – und da das Perceptron in seiner Funktion aus unzusammenhängenden Daten logische Zusammenhänge finden kann, erscheint es in diesem Zusammenhang als sehr passend, wenn diese beiden Techniken in einer Strategie vereint werden. Mit den Öffnungs- sowie Schlusskurs der 2000 +/- 5 Handelstage zurückliegenden Tage ausgestattet gab das Perceptron nun allerdings nur zu 55,6% den Richtigen Anstieg für den Tag wieder, was über einen Zeitraum von 53 Jahre in ca. einer Verdopplung der Startkapitals endete. Der Grund dafür liegt in den Daten selbst, auch wenn es vor einem repetitiven Anstieg dieses Verhalten im Durchschnitt geben mag, so gibt es dieses, durch die Menge von rund 14000 Handelstagen, auch an vielen anderen Stellen, an welchen die Strategie nun Verlust einfährt (Abbildung 6.5).

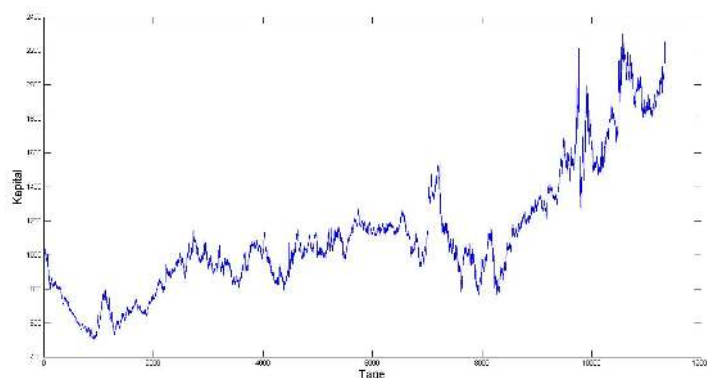


Abbildung 6.5: Kontoverlauf mit einem Startkapital 1000\$ im Refernzzeitraum mit der Fourier-High-Energy-Frequenzy-Perceptron-Strategy

So wie in bereits vorhergehenden Strategien findet aber auch diese keine Anwendung im Testzeitraum, da nach rund 270 Tagen von einem Startkapital von 1000 Dollar nur mehr rund 850 übrig bleiben (Abbildung 6.6).

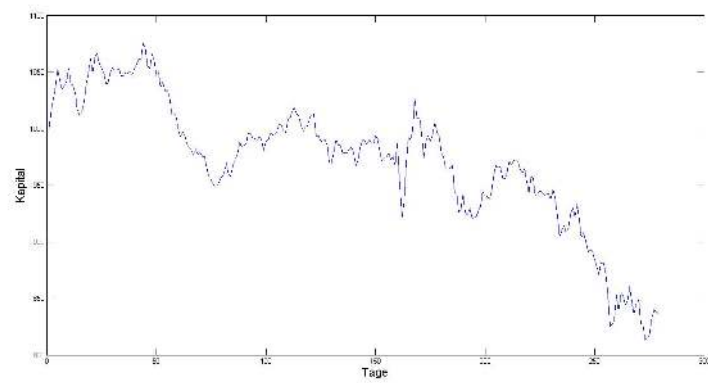


Abbildung 6.6: Kontoverlauf mit einem Startkapital 1000\$ im Testzeitraum mit der Fourier Fourier-High-Energy-Frequenzy-Perceptron-Strategy



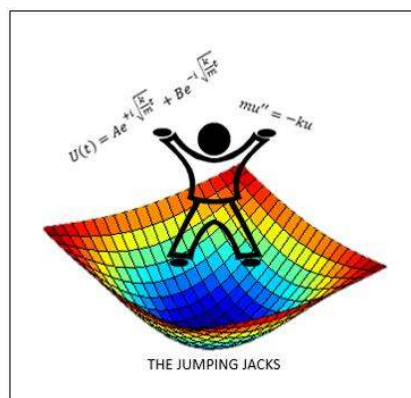
Woche der Modellierung mit Mathematik

Projekt: Kontinuumsmechanik

Betreuer: a.o. Univ.-Prof. Mag. Dr. Stephen Keeling

Titel: Deformationswellen elastischer Materialien

Wie man bereits am Titel des Projekts erkennen kann, war es unser Ziel, mithilfe von MATLAB die Bewegung und die Deformation eines Bungeeseils, in Form einer Kette an Federn und Massen, möglichst realistisch zu simulieren. Ebenso betrachteten wir durch Weiterführung dieses Codes komplexere Simulationen, bis hin zu einer elastischen Kugelmembran.



Teilnehmer:

Zahra Azizi

Lea Bischof

Franziska Harich

Daniel Horiatakis

Elias Hrad

Paul Lafer

Inhaltsverzeichnis

1	Grundvorstellung	3
1.1	$M \cdot u''(t) = F(t) = -k \cdot u(t)$	3
1.2	Integrieren	3
2	Feder mit einer Masse	4
2.1	Rechnerisch	5
2.2	Numerisch.....	5
3	Federn mit 2 Massen.....	7
4	n Federn mit n Massen.....	8
5	Modellerweiterung	9
5.1	Innere Reibung.....	9
5.2	Große Auslenkung.....	10
6	Masse-Feder-Kette im Dreidimensionalen Raum	10
7	Gitarrensaite	11
8	Membran	12
9	Scheibenmembran	13
10	Die Kugelmembran	14

1 Grundvorstellung

1.1 $M \cdot u''(t) = F(t) = -k \cdot u(t)$

Laut dem zweiten Newton'schen Gesetz ist Kraft gleich Masse mal Beschleunigung und laut dem Hook'schen Gesetz ist Kraft gleich Auslenkung mal der rückwirkenden Federkraft, der Federkonstante. Da beide Gesetze dieselbe Kraft beschreiben, kann man sie gleichsetzen.

Aufgrund dieses physikalischen Prinzips wollten wir als erstes die Bewegung einer Masse an einer Feder darstellen. Um jedoch die Grundvorstellung zu verstehen, lernten wir zu Beginn zu Integrieren.

1.2 Integrieren

Um aus einer Ableitung einer Funktion die Grundfunktion zu ermitteln muss man integrieren. Daher kann diese Rechentechnik als Gegenstück zum Differenzieren gesehen werden.

Um zu integrieren betrachtet man die zu integrierende Funktion, wie etwa:

$$f(x) = x^2$$

Nun bestimmt man die Grundfunktion, das heißt, man bestimmt die Funktion, die abgeleitet unsere Funktion $f(x)$ ergibt. Diese Grundfunktion bezeichnet man mit $F(x)$.

In unserem Fall wäre $F(x) = x^3/3$.

Probe: Die Ableitung von $F(x)$: $F'(x) = 3 \cdot x^2/3 = x^2 = f(x)$

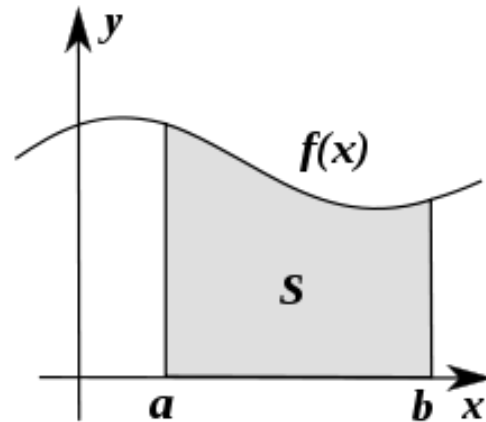
Allgemein geschrieben sieht dies so aus:

$$\int f(x) dx = F(x)$$

Diese Art des Integrierens nennt man „unbestimmtes Integrieren“.

Legt man nun zusätzlich Grenzen des Integrals fest, kann man auch den Flächeninhalt der Funktion zwischen dem Funktionsgraphen, der x-Achse und den beiden Grenzen berechnen.

$$\int_a^b f(x) dx = \text{Flächeninhalt}(s)$$



Die Formel dazu lautet:

$$\int_c^b f(x) dx = F(b)$$

$$F(b) - F(a) = \text{Flächeninhalt}$$

$$F'(x) = \lim_{h \rightarrow 0} \frac{1}{h} * \left[\int_c^{x+h} f(t) dt - \int_c^x f(t) dt \right] = \lim_{h \rightarrow 0} \frac{1}{h} * \left[\int_x^{x+h} f(t) dt \right] = f(x)$$

Beispielsweise kann die Fläche für $f(x)=x$ im Bereich 0 bis 1 auf diese Weise berechnet werden:

$$\begin{aligned} F(x) &= x^2/2 \\ F(1) - F(0) &= \frac{1^2}{2} - \frac{0^2}{2} \\ &= 1/2 \end{aligned}$$

Diese Art von Integrieren nennt man „bestimmtes Integrieren“.

2 Feder mit einer Masse

$$m * u''(t) = -k * u(t)$$

Dieses Problem lässt sich rechnerisch und numerisch lösen:

2.1 Rechnerisch

Als Hilfsmittel verwenden wir die Euler'sche Formel:

$$u(t) = e^{a*t}$$

$$u'(t) = a * e^{a*t}$$

$$u(t) = a^2 * e^{a*t}$$

$$m * a^2 * e^{a*t} + k * e^{a*t} = 0$$

durch e^{a*t}

Wir dividieren

$$m * a^2 + k = 0$$

$$a = \pm i * \sqrt{k/m}$$

$$\text{Lösungen: } e^{i*\frac{\sqrt{k}}{m}*t}, e^{-i*\frac{\sqrt{k}}{m}*t}$$

$$u(t) = A * e^{i*\frac{\sqrt{k}}{m}*t} + B * e^{-i*\frac{\sqrt{k}}{m}*t}$$

$$\tilde{A} = (A + B), \quad \tilde{B} = (A * i - B * i)$$

$$u(t) = \tilde{A} * \cos\sqrt{\frac{k}{m}} * t + \tilde{B} * \sin\sqrt{\frac{k}{m}} * t$$

löst allgemein $m * u''(t) = -k * u(t)$

2.2 Numerisch

MATLAB kann keine zweiten Ableitungen lesen, daher mussten wir unsere Formel umschreiben, sodass sie nur erste Ableitungen beinhaltet.

```
[t,y] = ode45 (@myf,tspan,y0,opts);
```

Diese Zeile berechnet in MATLAB numerisch alle Lösungen einer Funktion.

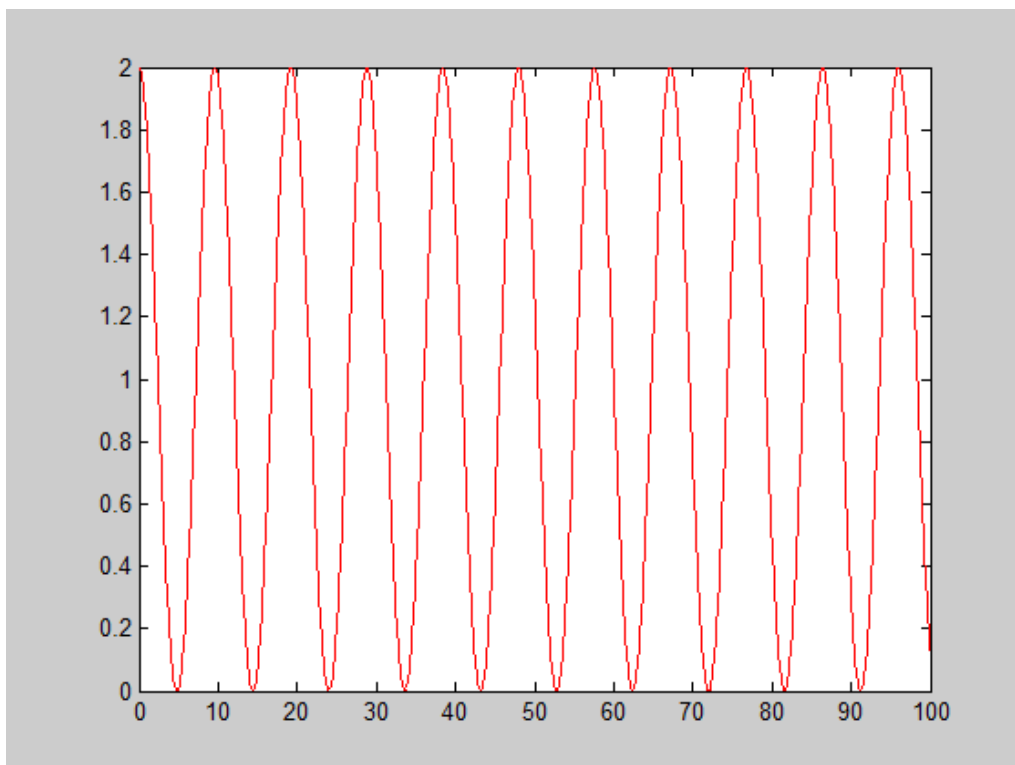
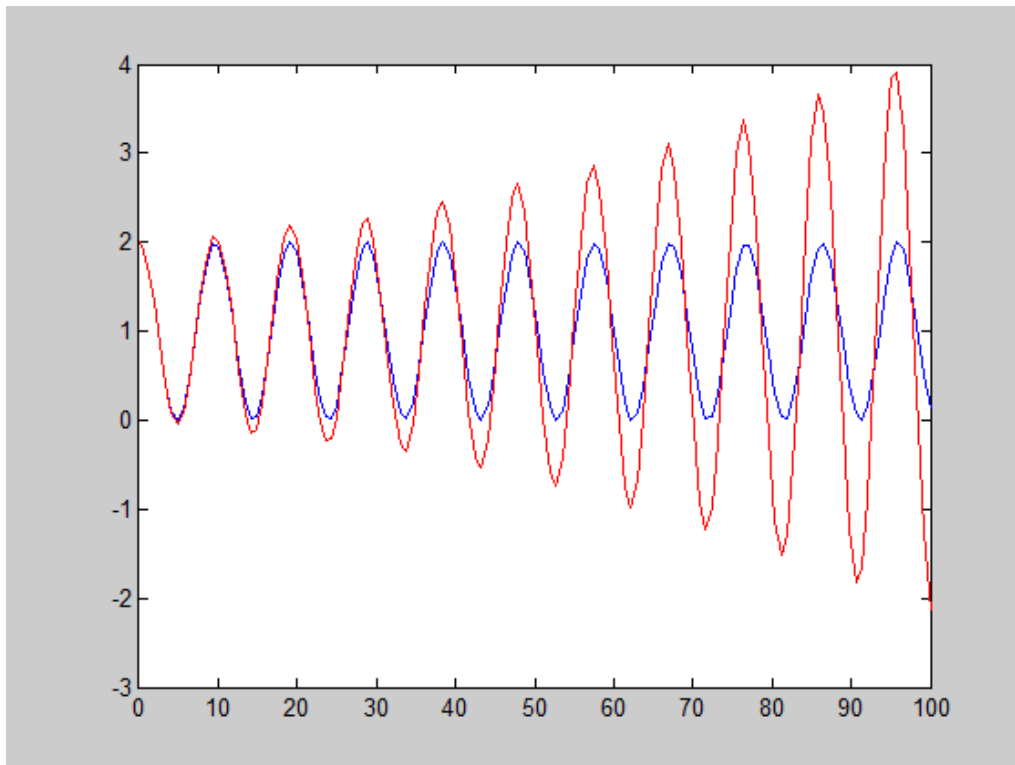
$$y(0) = y_0 = \begin{pmatrix} u(0) \\ u'(0) \end{pmatrix}$$

$$y'(t) = f(t, y(t))$$

Diese Form erwartet MATLAB. Daher müssen wir in $y(t)$ bereits eine erste Ableitung einfügen, damit $y'(t)$ dann eine zweite Ableitung ausgibt.

$$\text{Daher ist } y'(t) = \begin{pmatrix} u(t) \\ u'(t) \end{pmatrix}' = \begin{pmatrix} 0 & 1 \\ -k/m & 0 \end{pmatrix} * \begin{pmatrix} u(t) \\ u'(t) \end{pmatrix}$$

$$y'(t) = \text{Matrix} \cdot y(t) \text{ oder } y'(t) = M \cdot y(t)$$



Das erste Bild zeigt einen Vergleich der rechnerischen und numerischen mit ungenauer numerischer Lösung, während für das zweite Bild die Genauigkeit erhöht wurde.

3 Federn mit 2 Massen

Auch bei dieser Simulation verwendeten wir das Grundmodell, allerdings mussten wir nun den Einfluss der zweiten Feder auf die erste Masse mit berücksichtigen.

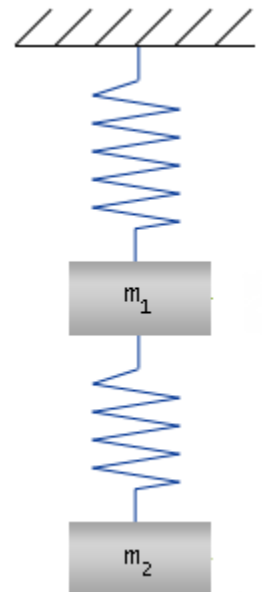
Unser angepasstes Modell lautet daher:

$$m_1 u_1'' = -k_1 * (u_1 - l_1) + k_2 * (u_2 - u_1 - l_2)$$

$$m_2 u_2'' = -k_2 * (u_2 - u_1 - l_2)$$

$$y' = \begin{pmatrix} u_1 \\ u_2 \\ u_1' \\ u_2' \end{pmatrix}' = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1}{m_1} - \frac{k_2}{m_1} & \frac{k_2}{m_1} & 0 & 0 \\ \frac{k_2}{m_2} & -\frac{k_2}{m_2} & 0 & 0 \end{pmatrix} * \begin{pmatrix} u_1 \\ u_2 \\ u_1' \\ u_2' \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{k_1 * l_1}{m_1} - \frac{k_2 * l_2}{m_1} \\ \frac{k_2 * l_2}{m_2} \end{pmatrix}$$

$$y' = M * y + b$$



diese Formel kann man auch umschreiben als:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{m_1} & 0 \\ 0 & 0 & 0 & \frac{1}{m_2} \end{pmatrix} * \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -k_1 - k_2 & k_2 & 0 & 0 \\ k_2 & -k_2 & 0 & 0 \end{pmatrix}$$

$$b = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{m_1} & 0 \\ 0 & 0 & 0 & \frac{1}{m_2} \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \\ k_1 * l_1 - k_2 * l_2 \\ k_2 * l_2 \end{pmatrix}$$

Diese Form ist für den weiteren Verlauf praktischer, da sie leichter verallgemeinert werden kann.

4 n Federn mit n Massen

Nun haben wir dieses Programm zu einer allgemeinen Form von n Massen und Federn umgewandelt.

Dabei konnten wir nun nicht mehr m_1, m_2 usw. schreiben, sondern mussten alle m als Matrizen

→ $m = \text{ones}(n, 1)$; umschreiben. Bei diesem Befehl legt $\text{ones}()$ fest, dass die gesamte Matrix mit Einsen gefüllt ist, während $(n, 1)$ die Dimension festlegt, die in diesem Fall 1 breit und n Elemente hoch ist.

Dasselbe musste mit den Definitionen für k und l gemacht werden. Will man nun den Wert festlegen, dann multipliziert man den gesamten Ausdruck mit der gewünschten Zahl (zB: $2 * \text{ones}(n, 1)$);

Die Matrix der zweiten Funktion konnte nun mit einigen kurzen Funktionen von MATLAB stark vereinfacht werden, indem wir sie in ihre Viertel aufteilen und diese aufgrund eines eindeutig darin erkennbaren Systems schnell modellieren konnten.

$$\begin{bmatrix} u1 \\ u2 \\ u3 \\ u1' \\ u2' \\ u3' \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m1^{\wedge} - 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & m1^{\wedge} - 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & m1^{\wedge} - 1 \end{pmatrix} * \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -(k1 + k2) & k2 & 0 & -c1 & 0 & 0 \\ k2 & -(k2 + k3) & k3 & 0 & -c2 & 0 \\ 0 & k3 & -k3 & 0 & 0 & -c3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ k1l1 - k2l2 \\ k2l2 - k3l3 \\ k3l3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ g \\ g \\ g \end{bmatrix}$$

Eine 3*3 Matrix die nur in der Diagonale mit Einsen gefüllt kann als $\text{eye}(3)$ dargestellt werden. In dieser 3*3 Matrix kann man ein System erkennen, wodurch sie sich verallgemeinern lässt.

Daher sieht der MATLAB-Code für n Massen dann folgendermaßen aus:

```
I=eye(n);
Z=zeros(n);
M1=diag(1./m);
C=diag(c);

k0=[k(2:n);0];
l0=[l(2:n);0];

K=-diag(k+k0)+diag(k(2:n),1)+diag(k(2:n),-1);

b1=zeros(n,1);
b2=k.*l-k0.*l0;
```

```
z=zeros(n,1);
o=ones(n,1);
```

Hier legen wir einzelne Variablen für Teile unserer Matrix fest, um diese dann vereinfacht darzustellen als:

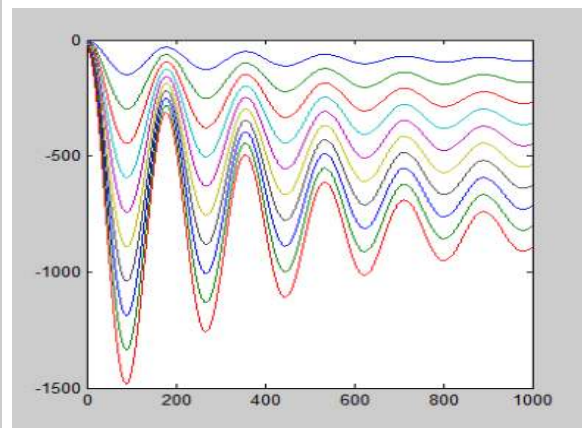
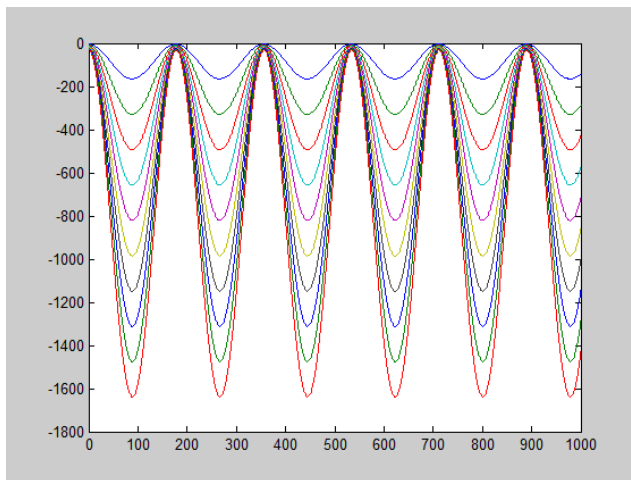
```
yt=[I,Z;Z,M1]*([Z,I;K,Z]*y+[b1;b2]+g*[z;o]);
```

5 Modellerweiterung

5.1 Innere Reibung

$$m * u''(t) = -k * u(t) - c * u'(t)$$

Um die innere Reibung miteinzubeziehen haben wir zu unserem Grundmodell den Term $-c * u'(t)$ hinzugefügt.



Diese Abbildungen beschreiben die Bewegung der Massen in Abhängigkeit von der Zeit, wobei bei der zweiten die innere Reibung miteinberechnet ist, bei der ersten jedoch nicht.

5.2 Große Auslenkung

Da bei größerer Auslenkung das Modell in dieser Version nicht mehr zutreffend war, da sich die Massen kreuzten, mussten wir es anpassen. Die neue Version hatte die folgende Form:

$$m_1 u_1'' = -k_1 \left(1 - \frac{l_1}{|u_1|}\right) * u_1 + k_2 \left(1 - \frac{l_2}{|u_2 - u_1|}\right) * (u_2 - u_1) - c_1 * u_1'$$

$$m_i * u_i'' = -k_i \left(1 - \frac{l_i}{|u_i - u_{(i-1)}|}\right) * (u_i - u_{(i-1)}) + k_{(i+1)} \left(1 - \frac{l_{(i+1)}}{|u_{(i+1)} - u_i|}\right) * (u_{(i+1)} - u_i) - c_i * u_i'$$

(i geht in dieser Darstellung von 2 bis n-1)

$$m_n * u_n'' = -k_n \left(1 - \frac{l_n}{|u_n - u_{(n-1)}|}\right) * (u_n - u_{(n-1)}) - c_n * u_n'$$

Auch unsere Funktion in MATLAB musste an diese Form angepasst werden:

```
utt(1)=-k(1).*dl(1).*du(1)+k(2).*dl(2).*du(2)-c(1).*ut(1);
utt(i)=-k(i).*dl(i).*du(i)+k(i+1).*dl(i+1).*du(i+1)-c(i).*ut(i);
utt(n)=-k(n).*dl(n).*du(n)-c(n).*ut(n);
```

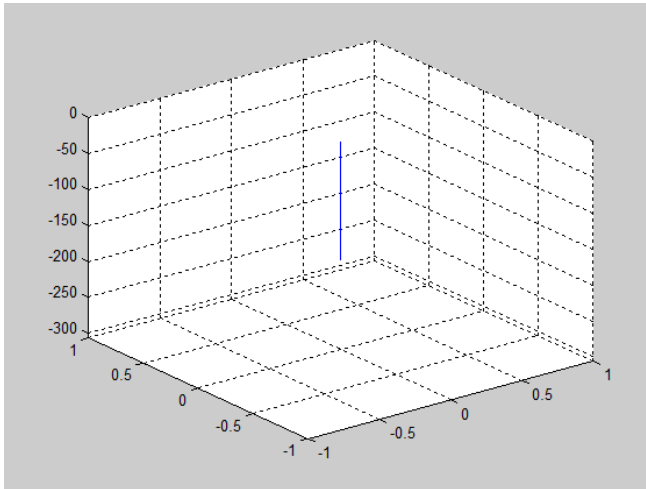
Zuvor liest man alle u und u' in die `myf`-Funktion ein, dann berechnet man die oben genannten Differenzen zwischen u_n und $u_{(n-1)}$, welche in den Variablen `du` gespeichert werden. Dann berechnet man die Absolutwerte von `du` und deklariert sie als Variable `L`. Um die Formel noch weiter zu vereinfachen legen wir in den Variablen `dl` den Wert 1 minus die Längen durch `L` fest. Zuletzt dividieren wir noch durch die Massen und rechnen anschließend die Schwerkraft dazu.

6 Masse-Feder-Kette im Dreidimensionalen Raum

Nun musste die `myf`-Funktion ein weiteres Mal erweitert werden, da wir mittlerweile nicht nur eine u -Variable benötigten, sondern für jede Koordinate eine eigene Auslenkung, wodurch u zu u_x, u_y, u_z und u' zu u'_x, u'_y und u'_z wurden. Diese sechs Variablen wurden daraufhin eingelesen, wodurch dasselbe Verfahren in dreifacher Ausführung durchgeführt werden musste.

```
uttx(1)=-k(1).*dl(1).*dux(1)+k(2).*dl(2).*dux(2)-c(1).*utx(1);
uttx(i)=-k(i).*dl(i).*dux(i)+k(i+1).*dl(i+1).*dux(i+1)-c(i).*utx(i);
uttx(n)=-k(n).*dl(n).*dux(n)-c(n).*utx(n);
```

Diese Zeilen wirken zwar auf den ersten Blick neu, sind jedoch genau dieselben wie im letzten Kapitel, die für die x -Koordinate erweitert wurden, und analog auch für y und z existieren. Anschließend dividieren wir u_x'' , u_y'' und u_z'' durch die Massen und rechnen danach bei u_z'' die Schwerkraft hinzu.



7 Gitarrensaite

Bei der Modellierung der Gitarrensaite, stellten wir uns vor, dass das untere Ende der Federkette auch an einer Wand befestigt worden ist. Ansonsten blieb das Meiste im Vergleich zum Aufbau mit n-Massen gleich.

Um die „Gitarrensaite“ dann auch zum Schwingen anzuregen, schrieben wir eine Störung in unser Programm ein.

```
dux = [ux(1); (ux(2:n)-ux(1:(n-1))) ; -ux(n)];
duy = [uy(1); (uy(2:n)-uy(1:(n-1))) ; -uy(n)];
duz = [uz(1)-zmin; (uz(2:n)-uz(1:(n-1))) ; zmax-uz(n)];
```

Wobei zmin und zmax folgendermaßen definiert sind:

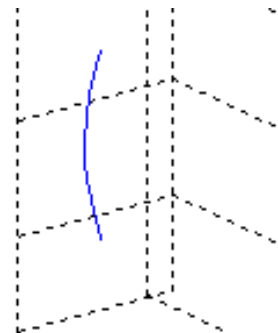
```
zmin = 0;
zmax = 1;
```

Die hinzugefügten Terme (in rot) bewirken das die Masse-Feder-Kette nun an beiden Enden befestigt ist.

```
ux=0.1*sin(ux(2:(n+1)));
```

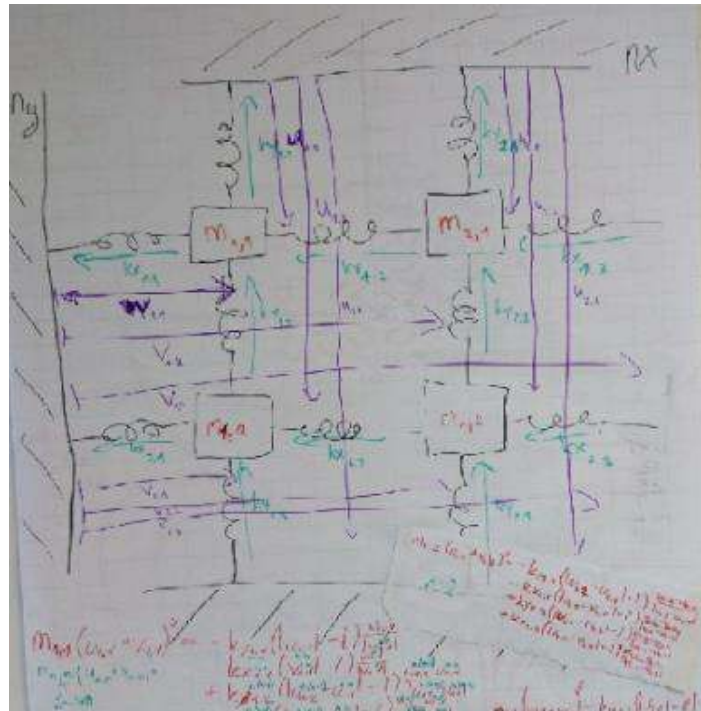
In dieser Zeile fügten wir eine Störung hinzu, um die Gitarrensaite zum Schwingen zu bringen.

Zusätzlich änderte sich der Code dahingehend, dass die n-te Masse, also die letzte Masse, nun von zwei Federn beeinflusst wird.



8 Membran

Zum besseren Verständnis fertigten wir zuerst eine Skizze an, welche unserer Vorstellung einer Membran mit 4 Massen und rundherum gelagerten Federn entsprach.



Danach passten wir die Formel an, sodass sie einer Störung einer Membran mit n Massen entsprach.

Wir haben als Koordinaten statt x, y und z , u, v und w verwendet um Verwirrung im Code zu vermeiden.

Wir haben die Grenzen der Membran als u_u, u_o, u_l, u_r (u unten, u oben, u links, u rechts), v_u, v_o, v_l, v_r und w_u, w_o, w_l, w_r festgelegt. Wir änderten die Gesamtgröße so, dass die gesamte Membran 1×1 Einheiten groß ist, daher mussten u_l und u_r im gleichmäßigen Abstand von $1/(n_x+1)$ aufgeteilt werden.

```
uo = zeros(1,ny);
uu = ones(1,ny);
ul = (1:nx)'/(nx+1);
ur = (1:nx)'/(nx+1);
```

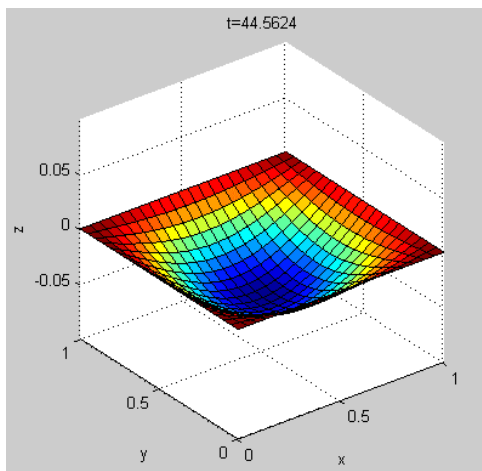
Die Membran wurde mit folgendem Code gespannt, da sie sonst nicht die gewünschte realistische Form annehmen würde:

```
lx = lx/10;
ly = ly/10;
```

Außerdem agieren die Kräfte in diesem Modell jetzt in 2 Dimensionen:

```
kx = ones(nx+1,ny);
```

```
ky = ones(nx,ny+1);
```



9 Scheibenmembran

Um die Modellierung einer Blase zu verstehen, veränderten wir den vorhandenen Code der Membran, um ihn vorerst in die Form einer Scheibe zu bringen. Dazu änderten wir die Grenzen der Membran wie folgt:

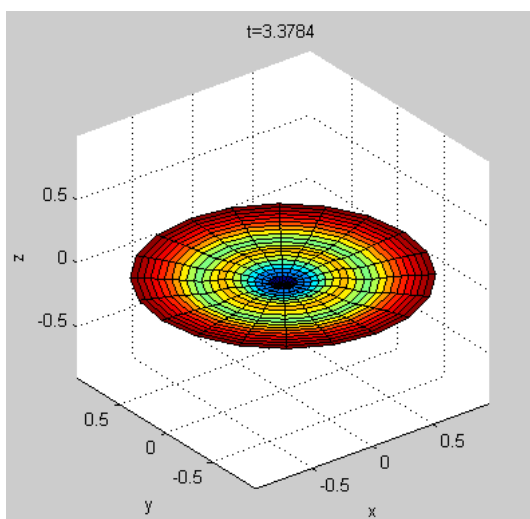
Die obere Grenze geht über in den Mittelpunkt der Membran, wodurch die untere Grenze zum Umfang wird.

Und die linke Grenze und die rechte Grenze gehen ineinander über, wodurch die von links gesehen erste Masse durch eine Feder mit der von rechts gesehen ersten Masse verbunden wird.

```
uu= cos(2*pi*(0:(ny-1))/ny);
```

Mit dieser Zeile teilen wir die Befestigungspunkte der äußersten Federn gleichmäßig auf den Umfang der Scheibe auf.

Zusätzlich haben wir die Massen abhängig von den Ruhelängen der Federn gemacht.



10 Die Kugelmembran

Um eine Kugel zu modellieren nahmen wir das bereits vorhandene Modell der Scheibenmembran und veränderten die Randkoordinaten so, dass sie eine Blase bildeten.

```
x = pi*(1:nx)'/ (nx+1);
y = 2*pi*(0:(ny-1))/ny;
```

x beschreibt die Koordinaten der Federaufhängungen um die Breitengrade und y jene um die Längengrade.

Wir haben die Störung in s umbenannt, um den Code zu vereinfachen da wir nun eine Störung in allen 3 Achsen verursachen wollten.

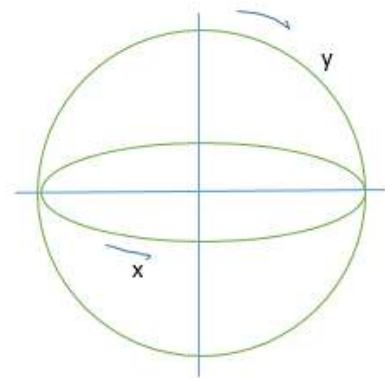
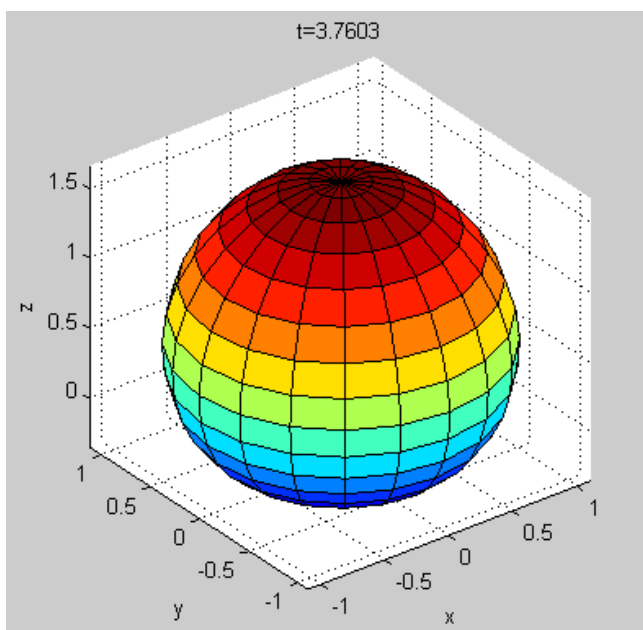
Wir haben kein uu mehr definiert, da es keine äußere Grenze mehr gibt und das auch für vu und wu übernommen. Daher mussten auch die, bei der Scheibenmembran außen liegenden Massen mit den ihnen nun benachbarten Massen durch Federn verbunden werden:

Vorher: $dux(nx+1,1:ny) = uu(1:ny)-u(nx,1:ny);$

Nachher: $dux(nx+1,1:ny) = u(nx,[(ny/2+1):ny,1:(ny/2)])-u(nx,1:ny);$

```
u = kron(sin(x), cos(y));
v = kron(sin(x), sin(y));
w = kron(cos(x), ones(1,ny));
```

u , v und w sind unsere Kugelkoordinaten.







„Kaiko“-Bassist Georg Schober (links) kommt aus Oberwart

MANI FROH

Genügend Geld für Groovepop mit Dingding

Die Band „Kaiko“ hat ihr erstes Album mit Hilfe von Crowdfunding finanziert.

FELDBACH. Dass „Kaiko“, eine junge Band mit südoststeirischen Wurzeln, es auch mit kleinem Budget ins Studio schaffte, verdanken die fünf Musiker 151 Unterstützern via Crowdfunding. Und vielleicht auch ihrem „Dingding“.

Normalerweise bringen die fünf Musiker Groovepop auf die Bühnen. Im Vorjahr waren sie unter anderem am Augartenfest zu hören, heuer stehen sie auch auf den Line-Ups einiger Festivals, wie dem „Picture On“ in Bildein. In ihrem Crowdfunding-Videoauftragaben sie aber ausnahmsweise ihre Rapkünste zum Besten. „Es sollte etwas zum Schmunzeln sein“, erzählt Gitarrist Philipp Maier. Nicht

nur das, am Ende des ersten Videos kam dann eine Tischglocke ins Spiel. Damit war das „Crowdfunding-ding“ geboren, das sich wie ein roter Faden durch alle weiteren Videos zog.

Mehr als 6000 Euro sind zusammengekommen. Gemeinsam mit Werner Angerer feilen die Musiker derzeit im „4 Tune Studio“ in Wien an ihrem ersten Album, auf dem ihre Eigenkompositionen zu hören sein werden. Im Sommer soll das Werk fertig sein. Für die Unterstützer gibts CDs, Vinyls und Pakete, in denen auch ein Dingding und ein Blingding, eine Discokugel, zu finden sind. „Wir sind sehr erleichtert und dankbar“, so Maier.

VERENA GANGL

In einer Woche ins All und an die Börse

In Pöllau gab es ein einwöchiges Trainingscamp für angehende Mathematik-Genies. Die jungen Rechenkünstler lösten dabei komplexe Aufgaben mit Bravour.

FRANZ BRUGNER

Eine Woche lang stand Pöllau ganz im Zeichen der Mathematik. Bereits zum zwölften Mal bezogen 30 Mathematik-Talente – durchwegs angehende steirische Maturanten – Quartier im Jugend- und Familiengästehaus. Unter der wissenschaftlichen Anleitung von einem halben Dutzend Professoren der Universität Graz, angeführt von Stephen Keeling, weiteten sie hier bei der „Woche der Modellierung mit Mathematik“ die Kapazitäten ihres logischen Denkvermögens aus. Koordiniert wurde dieses Stelldichein der Jungmathematiker vom Mittelschullehrer Patrick-Michel Frühmann aus Gleisdorf. Laut dem zweifachen Doktor, der am BG/BRG Leibnitz Mathematik unterrichtet und von dort insgesamt sieben Schüler nach Pöllau gelotst hat, müsse jedes Jahr eine Reihe von Bewerbern abgewiesen werden. „Wir schauen dabei auf den Zeitpunkt der Anmeldung. Wenn sich jemand zum ersten Mal bewirbt, hat er bessere Chancen als jemand, der schon einmal teilgenommen hat“, nennt Frühmann die Kriterien. Eine Art Vorselektion treffe der jeweilige Mathematiklehrer.

Wie schon in den Jahren zuvor wurden auch diesmal wieder fünf komplexe Problemstellungen dem Teilnehmerfeld offeriert:



Die teilnehmenden Schüler in Pöllau mit ihren Betreuern von der Karl-Franzens-Universität Graz

BRUGNER

Himmelsmechanik, Informatik, Straßenverkehr, Finanzwesen und Kontinuumsmechanik boten ein weites wie spannendes Feld, um die Hirn-Ganglien der mathematisch überdurchschnittlich begabten Gymnasiasten zum Glücken zu bringen.

Unter ihnen war auch der 18-jährige Grafendorfer Stefan Fischer, Achtklässler des Realgymnasiums Hartberg. Als künftiger Physik-Student tendierte er eigentlich dazu, sich rechnerisch intensiv mit den

sogenannten Lagrange-Punkten nahe den Umlaufbahnen von Planeten um die Sonne zu beschäftigen, landete dann aber bei den Berechnungsmodellen für das Transportnetzwerk. Sein Resümee: „Mir hat es ausgezeichnet gefallen. Ich habe wichtige Erfahrungen sammeln können. Innerhalb der Gruppe gab es einen regen Gedankenaustausch. Diese

eine Woche hat mir so viel gebracht, wie ich sonst in zwei Monaten in der Schule lerne.“

Börsengewinner

Ein anderes fünfköpfiges Team beschäftigte sich mit Handelsstrategien an der Börse. Zielsetzung war, den Markt zu schlagen, sprich, satte Gewinne – zumindest auf dem Papier – einzufahren. Mit Hilfe von Algorithmen war versucht worden im Hai-

fischbecken der Finanzwelt Oberwasser zu behalten. 53 Jahre des „Standard & Poor's Index“ an der Wall Street wurden analysiert und daraus Strategien entwickelt, das Geld bestmöglich zu vermehren. Der Großteil davon waren Flops, während es mit einzelnen Taktiken unter Zuhilfenahme künstlicher Intelligenz eine hundertfache Verzinsung des Kapitals gegeben hätte.

Stephen Keeling, Gründervater

dieser Begegnungswoche mathematischer Rohdiamanten, meinte zusammenfassend: „Das Endergebnis war eigentlich sekundär. Es ging uns darum, dass die jungen Leute möglichst viel Spaß an der Mathematik hatten.“ Den strahlenden Gesichtern der Teilnehmer nach zu schließen war bei diesem mathematischen Fitnessparcours neben dem Lern- auch der Genussfaktor ein sehr hoher.

ANZEIGE

23. ONLINE-AUKTION PROFITIEREN SIE ALS HÄNDLER



Willy Semmler
Hotel Restaurant Genusswirt Erla
Stubenberg/See

Ich nehme teil, weil durch die Online-Auktion neue Kunden auf mich aufmerksam werden.



Wenn auch Sie dabei sein möchten:
Karl Gutmann
0664 / 44 53 007
karl.gutmann@kleinezeitung.at

KLEINE ZEITUNG
Meine Kleine.

10. März 2016
Teilnahmeschluss

WAS, WANN, WO?

DONNERSTAG, 18.2.

OBERWART. Alles unter Kontrolle. Filmvorführung und Diskussion mit Werner Boote und Wolfgang Spitzmüller. Gratis-Ticket reservieren, Kartenabholung ab 19.30 Uhr. Diesel Kino, 20 Uhr. Tel. 0664-831 75 16, bgld@gbw.at.

ÄRZTE/KRANKENHAUS

Die Dienstbereitschaft der praktischen Ärzte erfahren Sie telefonisch unter der Kurzwahl 141 sowie im Burgenland bei der Bezirksstelle des **Roten Kreuzes Oberwart** unter Tel. (03352) 322 44 und in der Bezirksstelle des **Roten Kreuzes Güssing** unter Tel. (03322) 422 03. Der **Rotkreuz-Notruf** ist unter der Telefonnummer 144 erreichbar. **Rotkreuz-Krankentransporte** sind unter der Kurzwahl 14844 bestellbar.

Das Landeskrankenhaus Hartberg ist unter Tel. (03332) 605-0 erreichbar, das **Marienenkrankenhaus Vorau** unter Tel. (03337) 22 54-0, das **Landeskrankenhaus Fürstenfeld** unter Tel. (03382) 506-0, das **Landeskrankenhaus Feldbach** unter Tel. (03152) 899-0. Das **Krankenhaus Oberwart** erreichen Sie unter Tel. (05) 79 79-32 000, das **Krankenhaus Güssing** erreichen Sie unter Tel. (05) 79 79-31 000.

Vergiftungs-Informationsbüro: Tel. (01) 406 43 43.

Apothekenruf 1455.

TELEFONSEELSORGE

Die Telefonseelsorge ist unter der Kurzwahl 142 erreichbar.

SONSTIGES

Festnahme-Notruf der Österreichischen Rechtsanwälte: Tel. 0800 376 386.

MEIN V%RTEIL:
50 % AUF SPEISEN

MAHLZEITPASS NEU!

z. B. im Hotel-Restaurant „Pusswald“
Grazer Straße 18
Hartberg



Alle Infos unter kleinezeitung.at/vorteilsclub
Nur für Vorteilsclub-Mitglieder der Kleinen Zeitung.

FÜR SIE DA **REGIONALREDAKTION HARTBERG** Ressavarstraße 26, 8230 Hartberg.
Tel.: (03332) 63 610; Fax-DW: 74; hartred@kleinezeitung.at;
Redaktion: Thomas Pilch (DW 76), Harald Hofer (DW 71), Kirin Kohlhauser (DW 80), Daniela Buchegger (DW 75); **Sekretariat:** Gertrude Reitbauer, Heidemarie Rappold; Mo. – Do. 8 bis 16 Uhr, Fr. 8 bis 13 Uhr; E-mail: hartberg@kleinezeitung.at
Werbeberater: Karl Gutmann (DW 72), Sonja Fuchs (DW 79); **Abo-Service:** (0316) 875 3200

Vielen Dank für Ihr Interesse!

Woche der Modellierung mit Mathematik, JUFA Pöllau,

6.–12. Februar 2016

