groop4

WOCHE

DER

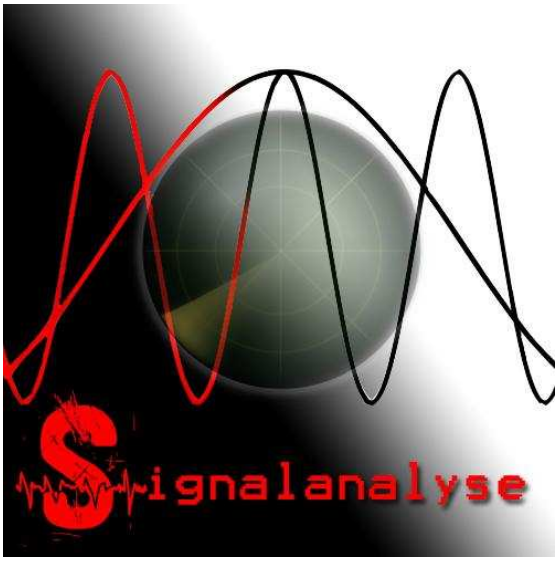
MODELLIERUNG

MIT

MATHEMATIK

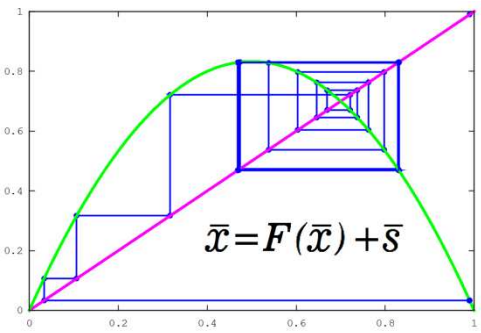
PageRank-Algorithmus

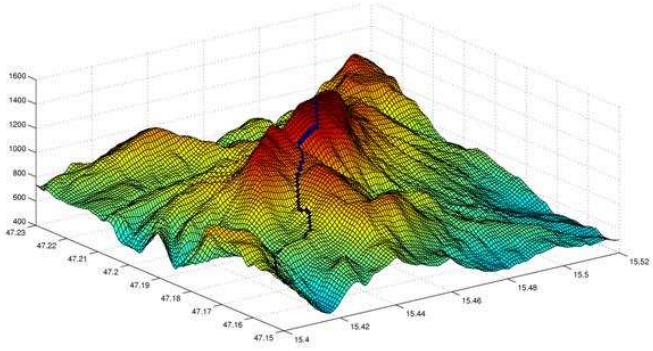
Google-Suche | Auf gut Glück!



DOKUMENTATIONS-BROSCHÜRE

6.2. – 12.2.2011





Das Land Steiermark

UNI GRAZ

MATHEMATIK

WOCHE DER MODELLIERUNG MIT MATHEMATIK

Pöllau bei Hartberg, 6.2. – 12.2.2011

Weitere Informationen:

http://math.uni-graz.at/modellwoche/2011/

Das Land Steiermark

UNI GRAZ MATHEMATIK

Vorwort

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

Nun ist leider für viele Menschen Mathematik ein Schulfach, mit dem sie eher unangenehme Erinnerungen verbinden. Umso erstaunlicher erscheint es, dass Schülerinnen und Schüler sich freiwillig melden, um eine ganze Woche lang mathematische Probleme zu wälzen - und dabei auch noch Spaß haben. Sie erleben hier offensichtlich die Mathematik auf eine Art und Weise, wie sie der Schulunterricht nicht vermitteln kann. Die jungen Leute arbeiten und forschen in kleinen Gruppen mit Wissenschaftler/innen an realen Problemen aus den verschiedensten Bereichen und versuchen, mit Hilfe mathematischer Modelle neue Erkenntnisse zu gewinnen. Sie arbeiten ohne Leistungsdruck, dafür mit Eifer und Enthusiasmus, rechnen, diskutieren, recherchieren, oft auch noch am späten Abend, in einer entspannten und kreativen Umgebung, die den Schüler/innen und betreuenden Wissenschaftler/innen gleichermaßen Spaß macht. Als Projektbetreuer konnte ich 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, 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 siebten Mal diese inzwischen zur Institution gewordene Veranstaltung. Ihm sei an dieser Stelle noch einmal ausdrücklich und herzlich gedankt. Besonders wichtig war in den vergangenen Jahren auch die Unterstützung durch den langjährigen Mentor der Modellierungswoche, Herrn o.Univ.-Prof. Dr. Franz Kappel, der oft auch eine eigene Gruppe mit interessanten Problemstellungen betreut hat.

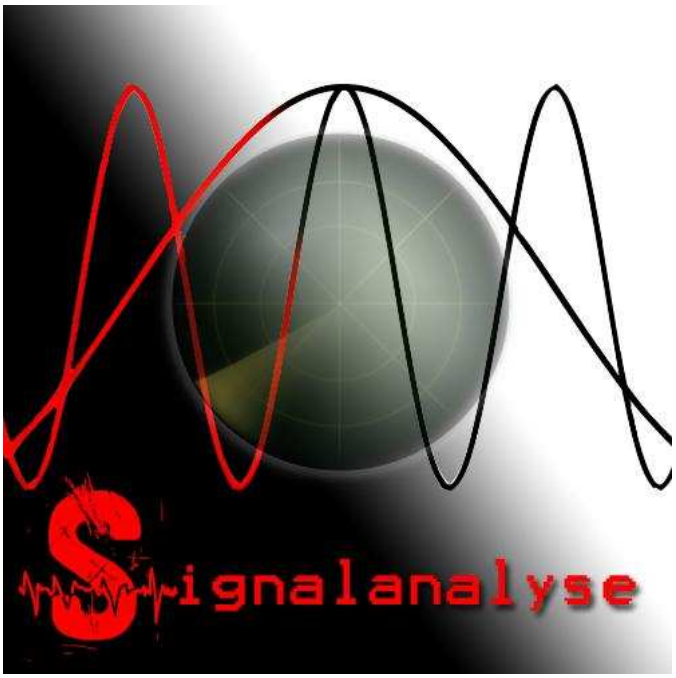
Wir danken dem Landesschulrat für Steiermark, und hier insbesondere Frau Landesschulinspektorin Frau HR Mag. Marlies Liebscher, für die Hilfe bei der Organisation und ihre kontinuierliche Unterstützung der Idee einer Modellierungswoche. Ohne den idealistischen, unentgeltlichen und engagierten Einsatz der direkten Projektbetreuer Dr. Peter Schöpf, Dr. Kristian Bredies, Dr. Georg Propst und Dr. Christian Clason - alle Institut für Mathematik und Wissenschaftliches Rechnen - hätte diese Modellierungswoche nicht stattfinden können.

Besonderer Dank gebührt ferner Herrn Mag. Christoph Gruber, der die ganze Veranstaltung betreut und auch die Gestaltung dieses Berichtes übernommen hat, Frau Michaele Seiwald für die tatkräftige Hilfe bei der organisatorischen Vorbereitung, und Herrn Dr. Georg Propst für die Hilfe bei der Betreuung der Hard- und Software. Es wird auch heuer die Modellierungswoche selbst wissenschaftlich untersucht. Die fachdidaktische Begleitforschung wird von Herrn Mag. Christoph Gruber im Rahmen einer Dissertation an unserem Institut und im Auftrag des Bundesministeriums für Wissenschaft und Forschung durchgeführt. Wir danken den Schülerinnen und Schülern für ihre Geduld beim Ausfüllen zahlreicher Fragebögen und für ihre Bereitschaft, sich psychologischen Tests aller Art zu unterziehen.

Finanzielle Unterstützung erhielten wir vom Land Steiermark durch Landesrätin Mag. Kristina Edlinger-Ploder und von der Karl-Franzens-Universität Graz durch Vizerektor Prof. Dr. Martin Polaschek, Dekan Prof. Dr. Karl Crailsheim und Prof. Dr. Karl Kunisch, dem Leiter des Instituts für Mathematik und Wissenschaftliches Rechnen.

Pöllau, am 12. Februar 2011

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



Geschwindigkeits- und Entfernungsmessung mit Signalwellen

Dr. Peter Schöpf

Modellierer/innen:

Nikolaus Leopold

Fabian Peter Hammerle

Dario Kaylani

Dino Mehic

Mario Messiha

Lissa Wilding

Inhalt

I. Problemstellung

Signalausbreitung und -analyse spielen eine wichtige Rolle bei Radarmessungen. Dabei müssen wir zwei Arten der Messung unterscheiden: Sendung von Impulsen und deren Echos, oder dauerhafte Aussendung von Wellen konstanter Frequenz. Die Impulse haben alle die gleiche Zeitdauer (PW) und werden in gleichmäßigen Abständen (PRT) ausgesendet. Ein Messvorgang besteht in der Aussendung von ca. 100 solchen Impulsen und der Registrierung ihrer Echos.

Im Rahmen der Modellierungswoche setzten wir uns das Ziel die Welt der Physik mit der Mathematik zu verbinden. Um uns den anschaulichen Einstieg in die Probleme zu erleichtern, modellierten wir die auftretenden Bewegungsvorgänge zunächst mit GeoGebra. Anschließend leiteten wir allgemein gültige Formeln für einige Fragestellungen her. Insbesondere bestimmten wir die Gültigkeitsbereiche für Ort und Geschwindigkeit des bewegten Objektes, die für ein korrektes Funktionieren des Messgerätes notwendig sind.

II. Grundlegende Überlegungen

1. Echoberechnung

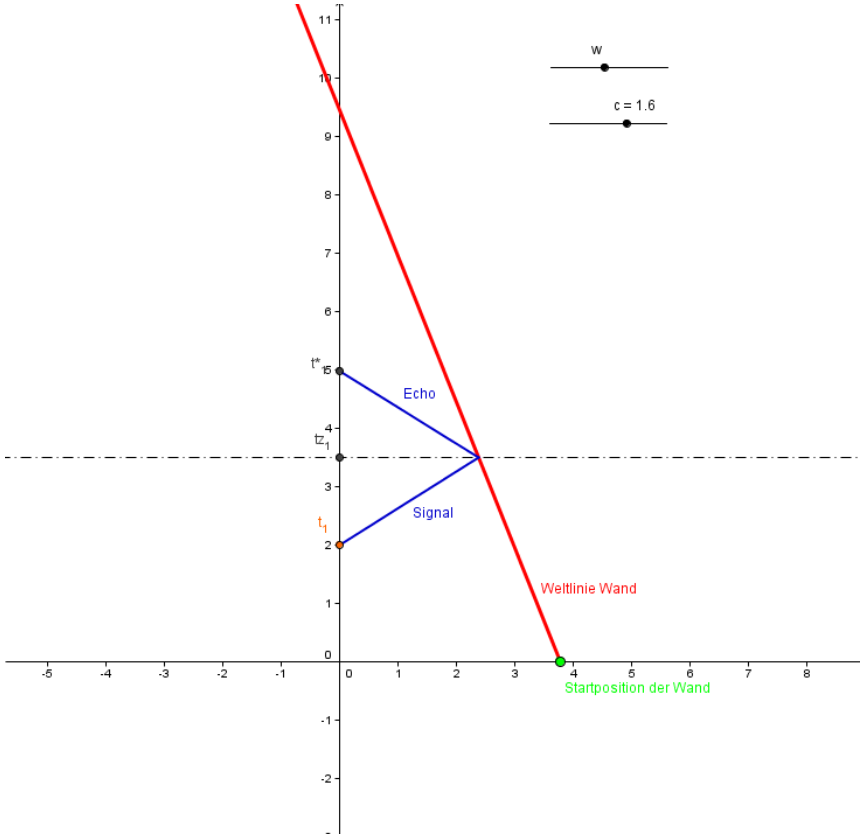


Abbildung 1

t_1 … Aussendungszeitpunkt des Signals
t^z_1 … Zeitpunkt des Zusammentreffens des Signals mit der Wand
w … Wandgeschwindigkeit (-w Näherung zum Sender)
t* … Zeitpunkt der Rückkehr des von der Wand reflektierten Signals (=Echo)
c … Lichtgeschwindigkeit bzw. Ausbreitungsgeschwindigkeit des Signals
x(0) … Startpunkt der Wand

Für die Formelherleitung haben wir folgende Größen gegeben: $c, x(0), w, t_1$

- Signalgeschwindigkeit $s(t) = c \cdot t$
- Wandgeschwindigkeit $x(t) = x(0) + w \cdot t$ (w=const.)

Ein ruhender Sender sendet ein Signal s_1 zum Zeitpunkt t_1 aus und trifft zum Zeitpunkt t^z_1 auf ein bewegte Wand (z.B. Auto) und wird durch Reflexion als Echo(signal) zurückgesandt, welches zum Zeitpunkt t* ankommt.

Nun wollten wir unsere geometrischen Überlegungen auch in mathematischen Formeln ausdrücken:

Dabei stellten wir Geradengleichungen für das Signal und die Wand auf:

$$s(t) = c \cdot t \rightarrow s(t) = (t - t_1) \cdot c$$

$$x(t) = x(0) + w \cdot t$$

Um den Zusammentreffzeitpunkt von Strahl und Wand zu berechnen müssen wir beide Gleichungen miteinander schneiden: $x(t) = s(t)$

$$x(o) + w \cdot t = (t - t_1) \cdot c$$

Daraus ergibt sich der Zusammentreffzeitpunkt $t_1^z = \frac{x(0)+t_1}{c-w}$

Daraus können wir t_1^*, den Ankunftspunkt des Echos, berechnen: $t_1^* = t_1 + (t_1^z - t_1)$

Danach haben wir t_1^z in diese Formel eingesetzt und daraus hat sich folgende Formel für t_1^* ergeben:

$$t_1^* = \frac{2 \cdot x(0)}{c - w} + t_1(c + w)$$

Zu diesem Zeitpunkt trifft das Signal auf die Wand.

2. Senden zweier Signale

Nun haben wir analog zum ersten Signal einen parallelen Signalstrahl zum Zeitpunkt t_2 ausgesendet und auch hier den Zusammentreffpunkt t_2^* und den Echopunkt t_2^z berechnet.

Es wird definiert:

$$\Delta t = t_2 - t_1$$
$$\Delta t^* = t_2^* - t_1^*$$

$$\Delta t^* = \frac{(c + w)}{(c - w)} \cdot \Delta t$$

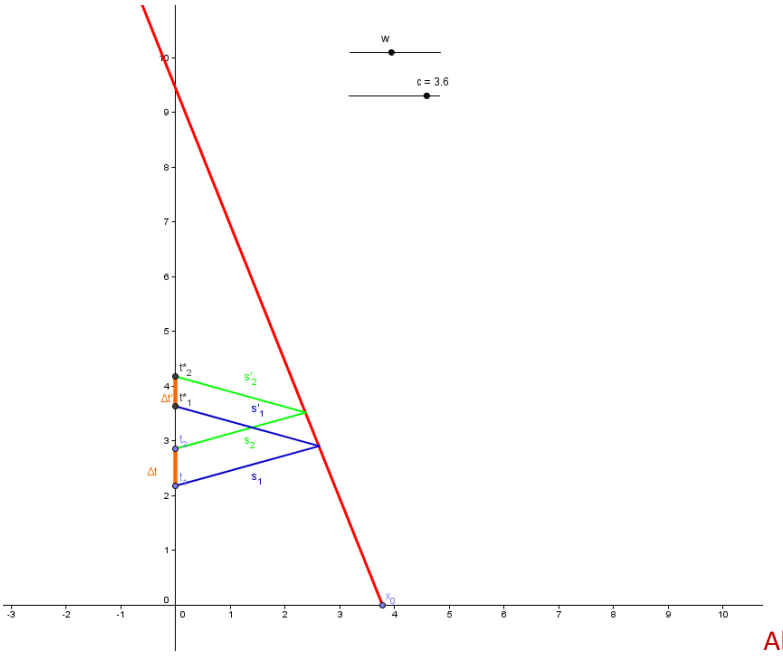


Abbildung 2

Unter der Annahme, dass c, t_1, t_2, t_1*und t_2* gegeben sind, haben wir das w (durch Umformung) berechnet: $w = \frac{\Delta t^* - \Delta t}{\Delta t^* + \Delta t} \cdot c$

Nun haben wir das w in die Formel von t_1^* eingesetzt und erhalten nach Umformungen die Herleitung von $x(0)$

$$x(0) = \frac{t_1^* \cdot t_2 - t_2^* \cdot t_1}{\Delta t^* + \Delta t} \cdot c$$

x(0) ist nun die Position der Wand $x(0)$ zum Zeitpunkt 0.

Unter Berücksichtigung, dass das Echo des ersten Signals vor dem Aussenden des zweiten Signals ankommen muss, galt es herauszufinden wie weit die maximale Entfernung der Wand sein darf.

$$t_1 < t_1^* < t_2 (= t_1 + \Delta t)$$

Gegeben waren:

c … Lichtgeschwindigkeit

Δt … Sendeintervall = $t_{i+1} - t_i$

Nun haben wir durch Überlegen anhand der Skizze folgende Formel hergeleitet:

$$x(0)_{max} = c \cdot \frac{t_2 - t_1}{2}$$

x(0) ist die maximale Entfernung die eine Wand (Auto) haben darf, sodass das Echo noch vor dem Aussenden des zweiten Signals ankommt!
(Durch 2 wird dividiert da der Ankunftspunkt nur die halbe Strecke ist)

3. Problemstellung

Die obige Formel gilt allerdings nur für punktförmige Signale, welche in der Realität nie vorkommen.

Also mussten wir im weiteren Verlauf überlegen, wie wir eine Formel aufstellen können, in der die Pulsweite der Signale (=Signaldauer) sowie die Totzeit berücksichtigt werden. Zwar gibt es noch eine sogenannte Umschaltzeit, diese wird aber aus Erleichterungsgründen nicht berücksichtigt.
Außerdem wurde die Annahme getroffen, dass die Echoweite gleich groß ist wie die Pulsweite.

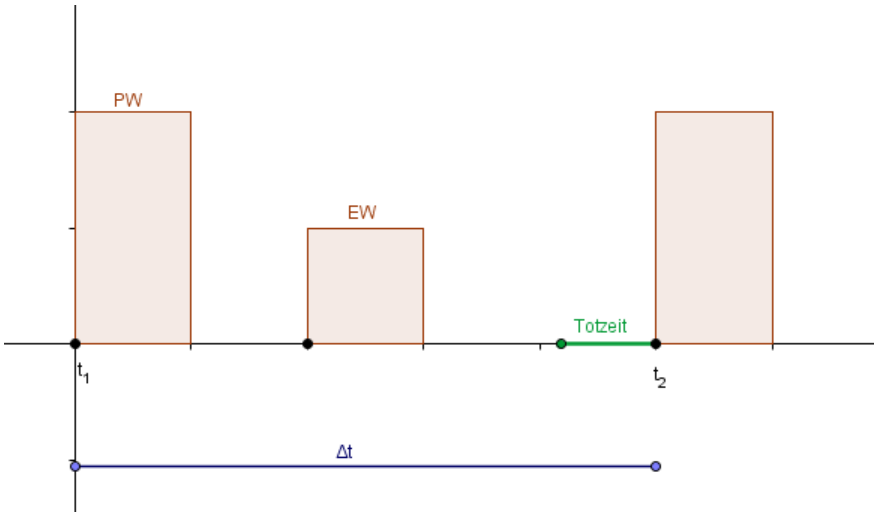


Abbildung 3

Gegeben waren:

c……. Lichtgeschwindigkeit
PW …… Pulsweite
PRT ….. Pulse Repetition Time = Δt
TZ ……. Totzeit (Sender und Empfänger schalten für kurze Dauer ab. Hier werden vom Radar interne Kontrollen der Systeme durchgeführt)

Für die Position des Objekts bei einem Signal zu einem Zeitpunkt t_i, welches selbst eine Signaldauer PW hat, ergibt sich nun folgende Formel:

$$x(0)_{min} = c \cdot \frac{PW}{2}$$

$$x(0)_{max} = c \cdot \frac{t_i + PRT - TZ - PW}{2}$$

→ sprich der Standort der Wand (Auto), mit jener minimalen bzw. maximalen Entfernung, dass das reflektierte Echosignal eines Signals zum Zeitpunkt t_i noch vor dem Signal zum Zeitpunkt t_{i+1} hineinfällt.

III. Dopplereffekt

1. Dopplereffekt – klassisch

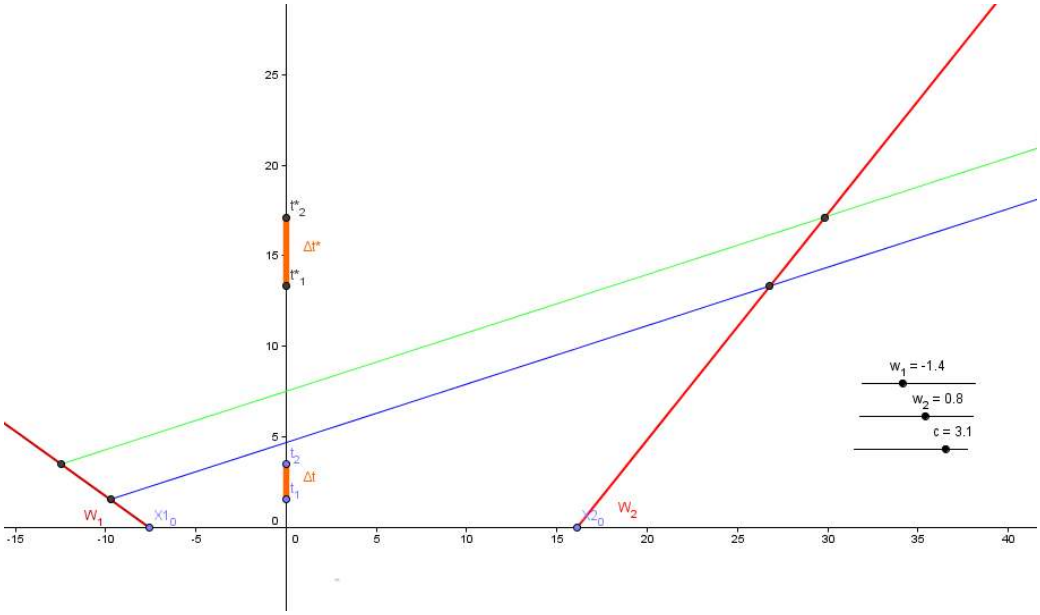


Abbildung 4

Nun haben wir uns den Doppler-Effekt unrelativistisch überlegt. Das heißt wir gehen davon aus, dass t = t* ist. Im Folgenden wollten wir uns den Zusammenhang von $\Delta t \; und \; \Delta t^*$ anschauen und die Änderungen in Bezug auf c und $w_{1,2}$ betrachten.

Gegeben war:

t_1… Aussendezeitpunkt des ersten Signals

t_2… Aussendezeitpunkt des zweiten Signals

w_1, w_2… die Geschwindigkeiten der zwei Wände (Auto)

$x_1(0), x_2(0)$… Standpunkte der Objekte zum Zeitpunkt 0

Um diesen Zusammenhang zu untersuchen haben wir zuerst Funktionen für die verschiedenen Startpunkte der beiden Wände (bzw. Autos) aufgestellt:

$$x_1(t) = x_1(0) + w_1 t$$

$$x_2(t) = x_2(0) + w_1 t$$

Danach haben wir die Strahlen der ausgesandten Signale ebenfalls als Funktionen ausgedrückt:

$$s_1(t): x = O_1 + \lambda \cdot \binom{c}{1}$$

$$s_2(t): x = O_2 + \lambda \cdot \binom{c}{1}$$

wobei $O_{1,2}$ als die Startpunkte der Signale von der ersten Wand w_1 aus zu sehen sind:

$O_1 = (x_1(t_1) \mid t_1) \quad \rightarrow O_1 = (x_1(0) + w_1 t \mid t_1)$

$O_2 = (x_2(t_2) \mid t_2) \quad \rightarrow O_2 = (x_1(0) + w_1 t \mid t_2)$

Wir formten s_1 auf eine Normalvektorform um und drückten x aus. Anschließend setzten wir dieses x in $x_2(t)$ ein um den Schnitt(zeit)punkt des ersten Signals mit dem zweiten Auto zu veranschaulichen. Analog dazu das gleiche mit s_2 und $x_2(t)$ um auch den Schnitt(zeit)punkt des zweiten Signals mit dem zweiten Auto darzustellen. Dafür erhalten wir folgende Formeln:

$$t_1^* = \frac{w_1 t_1 + x_1(0) - c t_1 - x_2(0)}{w_2 - c}$$

$$t_2^* = \frac{w_1 t_2 + x_1(0) - c t_2 - x_2(0)}{w_2 - c}$$

Die Differenz aus den beiden ist nun unser Δt^*

$$\Delta t^* = \frac{w_1 - c}{w_2 - c} \cdot (t_2 - t_1)$$

Nun haben wir die obigen Formeln mithilfe unserer GeoGebra-Konstruktion mit konkreten Zahlen verglichen. (siehe Abb. 4) Daraus konnten wir nun schließen, dass, wenn sich die Wände zueinander hin bewegen, Δt^* kleiner wird. Wenn sich die Wände allerdings voneinander wegbewegen wird Δt^* größer. Das Verhältnis von $\Delta t^* : \Delta t = \frac{w_1 - c}{w_2 - c}$

Da wir auch zeigen wollten, dass sich die Frequenz des Anfangssignals und des Echosignals verändern, haben wir die Frequenz wie folgt definiert → $f_1 = \frac{1}{\Delta t}$

$$f_2 = \frac{1}{\Delta f^*}$$

Wenn man unsere Formel für $\Delta t^* : \Delta t = \frac{w_1 - c}{w_2 - c}$ mithilfe der Frequenz ausdrückt erhalten wir Folgendes: $f_2 = \frac{w_2 - c}{w_1 - c} \cdot f_1$

Weiters haben wir uns auch noch einige Spezialfälle des Doppler-Effekts überlegt:

w_1… Geschwindigkeit des Senders (1. Wand bzw. Auto)

w_2… Geschwindigkeit des Empfängers (2. Wand bzw. Auto)

<u>1. Fall</u>: Beobachter in Ruhe und Sender nähert sich

$$w_2 = 0$$

$$f_2 = \frac{1}{1 - \frac{w_1}{c}} \cdot f_1$$

<u>2. Fall</u>: Beobachter in Ruhe und Sender entfernt sich

$$w_2 = 0$$

$$f_2 = \frac{1}{1 + \frac{w_1}{c}} \cdot f_1$$

3. Fall: Sender in Ruhe und Empfänger nähert sich

$$w_1 = 0$$

$$f_2 = (1 + \frac{w_2}{c}) \cdot f_1$$

4. Fall: Sender in Ruhe und Empfänger entfernt sich

$$w_1 = 0$$

$$f_2 = (1 - \frac{w_2}{c}) \cdot f_1$$

2. Doppler – Effekt – relativistisch

Nun haben wir uns auch noch den Doppler-Effekt in Bezug auf eine relativistische Anschauung überlegt. In einem relativistischen System können wir das Medium vernachlässigen.

Als Beispiel haben wir zwei Koordinatensysteme verwendet von denen wir eines gedanklich bewegt haben. Wir definierten ein Ereignis $\binom{x}{t}$ welches wir vom ersten (blauen) Koordinatensystem aus sehen können, und in Bezug auf das bewegte zweite (rote) Koordinatensystem mit Berücksichtigung der Relativitätstheorie, berechnen wollen. Wir senden ein Signal s zum Zeitpunkt t aus und beobachten, wann dieses Signal im zweiten bewegten Koordinatensystem ankommt. Dadurch können wir die Zeit- und Wegeinheit im zweiten (roten) Koordinatensystem bestimmen und in weiterer Folge das Ereignis in Bezug auf das zweite, bewegte (rote) Koordinatensystem.

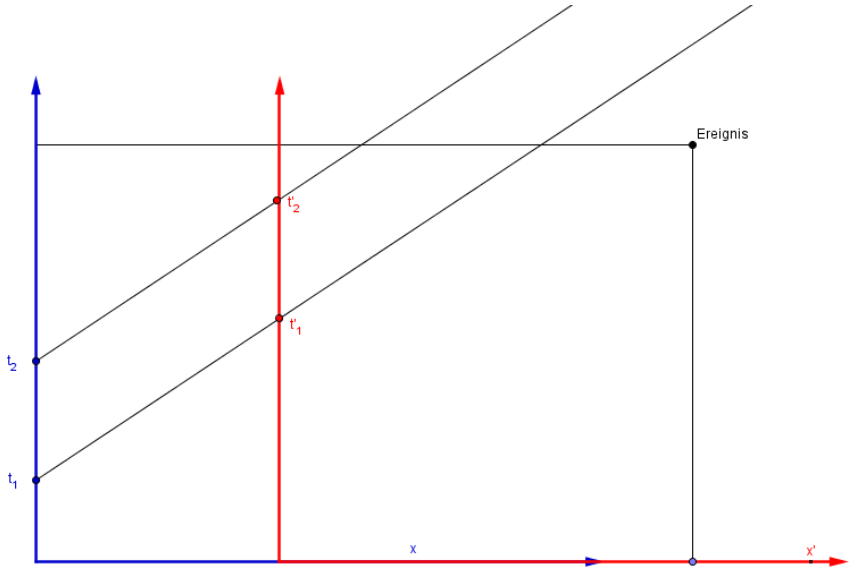


Abbildung 5

Dazu verwendeten wir die Lorentztransformation:

$$x' = \frac{1}{\sqrt{1-\left(\frac{v}{c}\right)^2}} \cdot x - \frac{v}{\sqrt{1-\left(\frac{v}{c}\right)^2}} \cdot t$$

$$t' = \frac{1-\frac{v}{c^2}}{\sqrt{1-\left(\frac{v}{c}\right)^2}} \cdot x - \frac{1}{\sqrt{1-\left(\frac{v}{c}\right)^2}} \cdot t$$

$$\begin{pmatrix} x' \\ t' \end{pmatrix} = \frac{1}{\sqrt{1-\left(\frac{v}{c}\right)^2}} \cdot \begin{pmatrix} 1 & -v \\ \frac{-v}{c^2} & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ t \end{pmatrix}$$

Das bewegte Koordinatensystem haben wir mit der Funktion $w(t) = v \cdot t$, und die Strahlen $s_1(t) = c \cdot (t - t_1)$ bzw. $s_2(t) = c \cdot (t - t_2)$ definiert.

Nun haben wir w(t) und s(t) geschnitten um die Punkte t_1^* und t_2^* zu berechnen:

$$t_1^* = \frac{c \cdot t_1}{c - v}$$

$$t_2^* = \frac{c \cdot t_2}{c - v}$$

$$x_1^* = v \cdot \frac{c \cdot t_1}{c - v}$$

$$x_2^* = v \cdot \frac{c \cdot t_2}{c - v}$$

Danach haben wir t_1^* und x_1^* in t' eingesetzt. (da wir x‘ bereits haben). Weiters haben wir $f_1 = \frac{1}{\Delta t}$ und $f' = \frac{1}{\Delta t'}$

Nach Umformen kamen wir auf folgende Formel, die die Empfangsfrequenz des roten Koordinatensystems darstellt:

$$f' = \sqrt{\frac{c-v}{c+v}} \cdot f_1$$

IV. Radarmessungen

1. Geschwindigkeitsintervall bestimmen

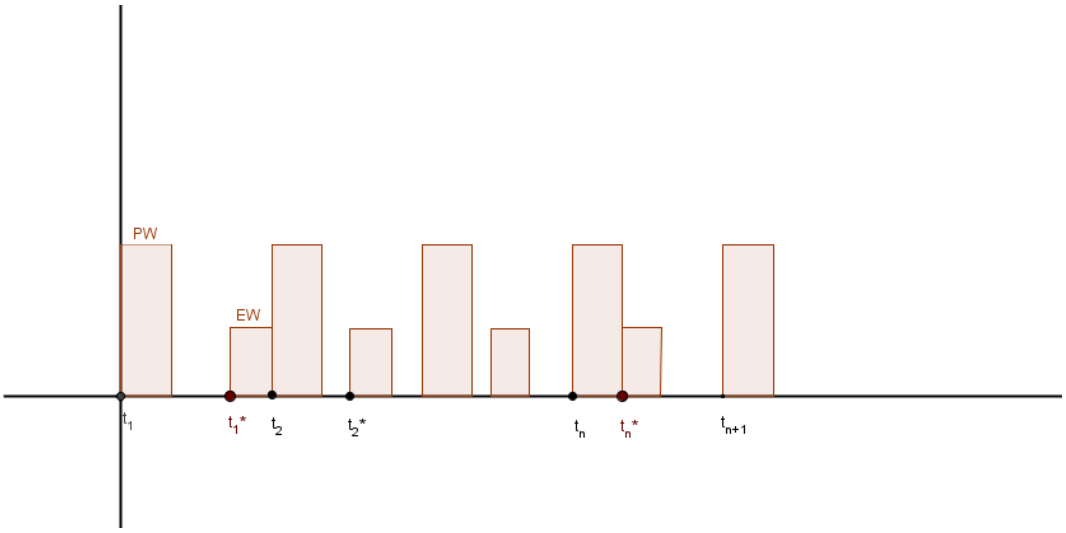


Abbildung 6

EW… Echoweite Gegeben: c; n; EW= a(w)*PW; PW≪PRT

PW…Pulsweite

PRT … Signalaussendungsintervall

n… Anzahl der Pulse

t_n…. Anfangspunkt Pulsweite = Startpunkt des Signals

t_n^*…Anfangspunkt Echoweite = Ankunftspunkt des Echos

Zum Vereinfachen setzten wir t_1=0

Somit ist $t_2 = PRT$ und t_n lässt sich durch folgende Funktion beschreiben: $t_n = PRT \cdot (n-1)$

Wir wollten nun das Intervall für die maximale und die minimale Geschwindigkeit, die die Wand (bzw. Auto) haben darf, um das Echosignal noch vor dem Aussenden des zweiten Signals zu empfangen, berechnen. Aus vorhergegangenen Berechnungen wissen wir, dass folgendes gilt →

$$\frac{\Delta t^*}{\Delta t} = \frac{c+w}{c-w}$$

Wir überlegten uns zuerst den Fall der **maximalen Geschwindigkeit** …

$$t_1^* = PRT - EW$$

$$t_n^* = (n-1) \cdot PRT + PW$$

… und setzten in die obige Formel ein:

$$\frac{c+w}{c-w} = \frac{(n-1) \cdot PRT + PW - PRT + EW}{(n-1) \cdot PRT}$$

Nach Umformen haben wir folgendes Ergebnis für w erhalten:

$$w_{max} = \frac{(2PW - PRT)}{PRT \cdot (2n-3)} \cdot c$$

$$w_{min} = \frac{PRT - 2PW}{PRT \cdot (2n-1)} \cdot c$$

Das Intervall für die „Wandgeschwindigkeit“ lautet wie folgt:

$$w_{max} \leq w \leq w_{min}$$

2. Ortsintervalle

Problemstellung: Gleich wie bei der Bestimmung des Geschwindigkeitsintervalls, war unser nächstes Ziel ein konkretes Ortsintervall zu bestimmen, indem sich die „Wand“ (Auto) befinden muss um eine zuverlässige Aussage über dessen Entfernung und Geschwindigkeit bestimmen zu können.

Zur ersten Analyse waren folgende Werte gegeben:

c … $Signalgeschwindigkeit\ (z.B.\ Lichtgeschwindigkeit)$

w … $Geschwindigkeit\ der\ Wand$

$-w$ … $Nähерung\ der\ Wand\ zum\ Sender$

$+w$ … $Entfernung\ der\ Wand\ vom\ Sender$

x_0 … $Startposition\ der\ Wand\ zum\ Zeitpunkt\ 0$

Δt … $Abstand\ zwischen\ den\ beiden\ gesendeten\ Signalimpulsen\ (pulse\ repetition\ time)$

PW … $Dauer\ eines\ Signalimpulse$

n … *Anzahl der Signale*

Aus der *Pulse Repetition Time* Δt kann der Aussendezeit jedes Signals berechnet werden:

$$t_1 = 1$$

$$t_i = (i-1) \cdot \Delta t$$

$t_i \ldots Aussendezeitpunkt\ des\ i.Signals$

Aufgrund dieser gegebenen Werte lassen sich Funktionen aufstellen, die zu jedem Zeitpunkt den Signalen bzw. der Wand einen eindeutigen Ort zuweisen:

$$s_i(t) = c \cdot (t - t_i)$$

$$x(t) = w \cdot t + x_0$$

$s_i(t) \ldots Position\ des\ i.Signals\ zum\ Zeitpunkt\ t$

x(t)…$Position\ der\ Wand\ zum\ Zeitpunkt\ t$

Damit der Bewegungsverlauf der Wand zuverlässig bestimmt werden kann, ist es in diesem Modell erforderlich, dass das Echo eines Signals erst nach dem Ende des Impulses des zugehörigen Signals und vor dem nächsten Signal eintrifft:

$$t_i + PW \leq t_i^* + EW \leq t_{i+1}$$

$t_i^* \ldots Zeitpunkt\ des\ Eintreffens\ des\ Echos\ des\ i.Signals$
$EW \ldots Impulsdauer\ des\ Echos$

Aus dieser Bedingung kann das erlaubte Zeitintervall des Echos des 1. und des n. Signals hergeleitet werden:

$$t_1 + PW \leq t_1^* \leq t_2 - EW$$
$$t_n + PW \leq t_n^* \leq t_{n+1} - EW$$

Anstelle der Zeitpunkte des Absendens der Signale können nun die entsprechenden Formeln eingesetzt werden:

$$PW \leq t_1^* \leq \Delta t - EW$$
$$(n-1) \cdot \Delta t + PW \leq t_n^* \leq n \cdot \Delta t - EW$$

Falls das 1. und n. Signal die genannten Bedingungen erfüllen, liegen auch alle anderen im zulässigen Intervall.

Um nun das Intervall, indem sich die Wand zum Zeitpunkt 0 befinden muss, bestimmten zu können, werden für $t_{1,n}^*$ und EW die bereits bekannten Formeln substituiert und die Ungleichungen nach x_0 aufgelöst:

1. Signal $$\frac{(c-w) \cdot PW}{2} \leq x_0 \leq \frac{\Delta t \cdot (c-w) - (c+w) \cdot PW}{2}$$

n. Signal $$\frac{(c-w) \cdot PW}{2} - w \cdot (n-1) \cdot \Delta t \leq x_0 \leq \frac{\Delta t \cdot (c - w \cdot (2 \cdot n - 1)) - (c+w) \cdot PW}{2}$$

Das endgültige Intervall wird nun auf den Bereich eingeschränkt, der alle aufgestellten Ungleichungen erfüllt:

$$min\left\{\frac{(c-w)\cdot PW}{2}\ ,\qquad \frac{(c-w)\cdot PW}{2}-\ w\ \cdot(n-1)\cdot\Delta\ t\ \right\}\leq\ x_0$$

$$x_0\ \leq max\left\{\frac{\Delta t\ \cdot(c-w)-\ (c+w)\cdot PW}{2},\ \frac{\Delta t\ \cdot(c-w\cdot(2\cdot n-1))-\ (c+w)\cdot PW}{2}\right\}$$

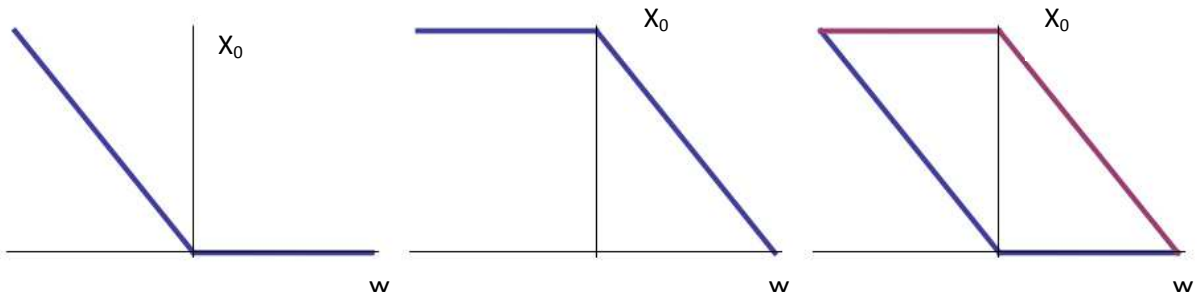


Abbildung 7: (a) stellt die Mindestentfernung des Objekts bei gegebener Geschwindigkeit w dar. (b) stellt die Maximalentfernung bei gegebener Geschwindigkeit w dar. (c) gibt den zulässigen Anfangsort des Objekts während der Messung wieder.

3. Erweiterung des Modells: Senderlage außerhalb der Bahn (realistisch)

Näherungsweise Berechunung

Zur Entwicklung eines realistischeren Modells, haben wir angenommen, dass sich der Sender nicht auf der gleichen Linie wie die Wand (Auto) befindet, sondern im Normalabstand d von der Straße. Es galt zu beachten dass eine Radarmessung nur im Winkel von 20°-22° möglich ist. Zur näherungsweisen Berechnung haben wir angenommen, dass sich ein imaginärer Sender auf der Bahn der Wand (Auto) befindet und eine im Vergleich zu c reduzierte Signalgeschwindigkeit besitzt, sodass die Signale beider Sender zur selben Zeit bei der Wand eintreffen, wobei sie gleichzeitig abgeschickt werden.

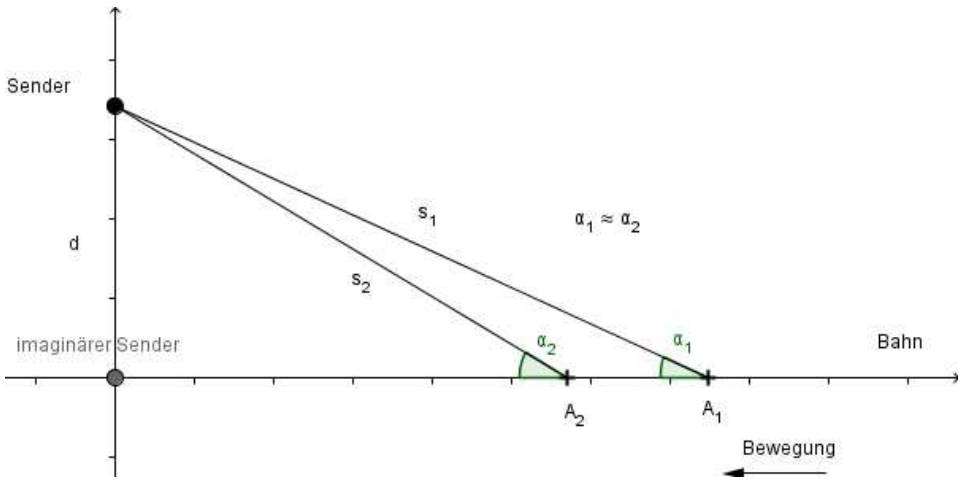


Abbildung 8

s_1… erstes Signal bewegt sich mit $c \cdot t$

s_{1*}… erstes Signal des imaginären Senders bewegt sich mit $c \cdot t \cdot k$

k … ist ein Faktor um den sich das imaginäre Signal langsamer bewegen muss als das erste Signal des realen Senders. Diesen galt es zu berechnen.

Somit kann man erkennen, dass der Faktor k, um den das imaginäre erste Signal langsamer sein muss, gleich cosα ist:

$$t \cdot c \cdot cos\alpha$$

Somit gelten alle alten Formeln, wobei wir für die Lichtgeschwindigkeit auch $c \cdot cos\alpha$ schreiben können, da wir nun den realistischen Fall haben, dass das Radarmessgerät sich nicht auf der gleichen Linie wie das Auto befindet.

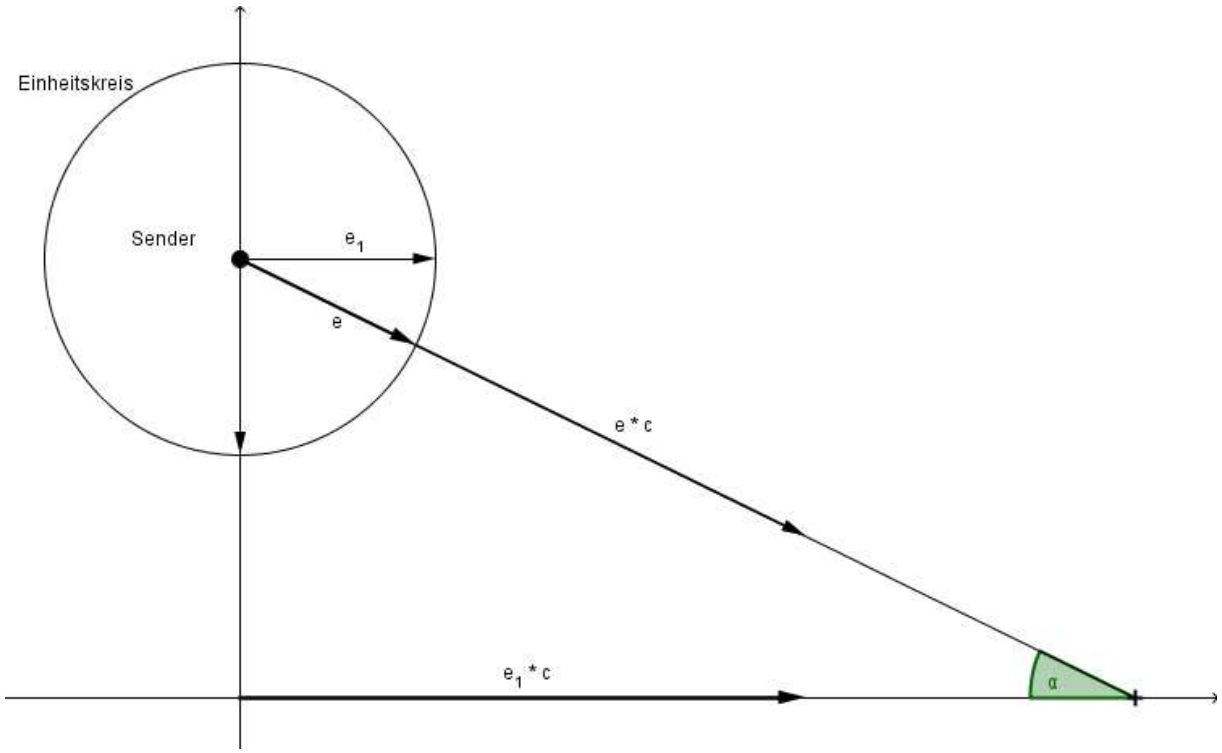


Abbildung 9

Exakte Bestimmung

Durch geometrische Überlegungen gelang es uns die exakte Formel für die Geschwindigkeit des Autos zu bestimmen:

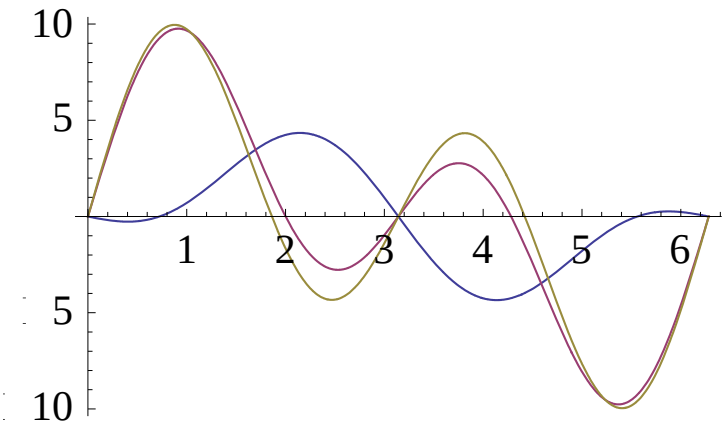
$$w = \frac{c}{\Delta t^* + \Delta t} \cdot \left(\sqrt{(t_2^* - t_2)^2 - \frac{4d^2}{c^2}} - \sqrt{(t_1^* - t_1)^2 - \frac{4d^2}{c^2 t_1^*}} \right)$$

4. Amplitudenmodulation eines hochfrequenten Trägersignals mit einem niederfrequenten Audiosignal

Es seien $as1(t)$, $as2(t)$ und $as3(t)$ drei Audiosignale, die in verschiedenen Studios aufgenommen wurden.
Wie man aus den unten definierten Audiosignalen abliest, handelt es sich jeweils um zwei Sinustöne im Abstand einer Oktave. Um das Prinzip der Amplitudenmodulation des Trägersignals mit dem Audiosignal zu verstehen und am Bildschirm anschauen zu können, wählen wir unrealistisch kleine Trägerfrequenzen nämlich $tf1 = 8$, $tf2 = 16$, $tf3 = 24$.

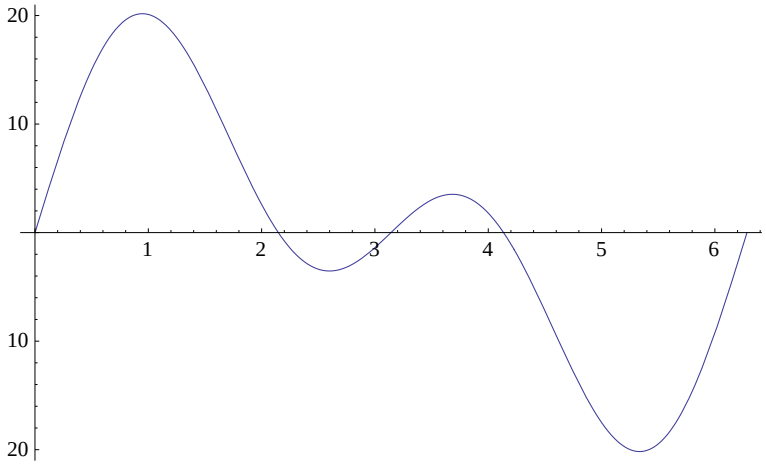
1) Im ersten Plot zeigen wir die Wellenbilder aller drei Audiosignale.

$$as1(t) = 3\sin(t) - 2\sin(2t)$$
$$as2(t) = 5\sin(t) + 6\sin(2t)$$
$$as3(t) = 4\sin(t) + 7\sin(2t)$$
$$tf1 = 8 \ \ (Hz)$$
$$tf2 = 16\ (Hz)$$
$$tf3 = 24\ (Hz)$$

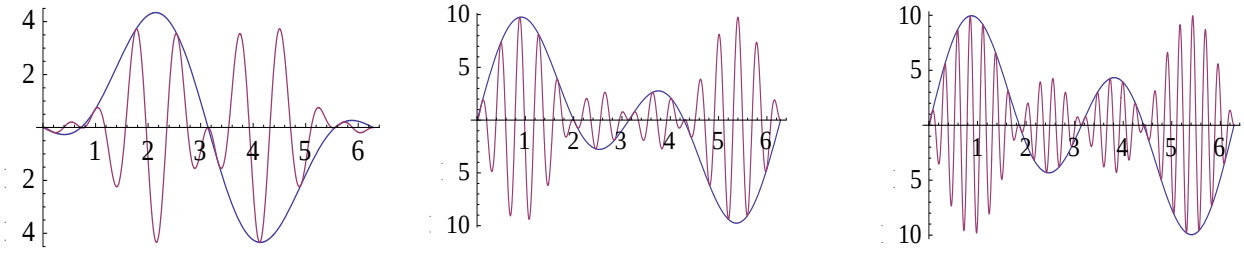


2. Im zweiten Plot sehen wir die Superposition der drei Audiosignale, wenn alle in einem Raum zugleich erzeugt werden. Aus dieser Superposition der Audiosignale sind die einzelnen Audiosignale nicht mehr rekonstruierbar, weil sich z.B. der Anteil mit der Audiofrequenz 2 auf verschiedene Arten in drei Signale zerlegen lässt:

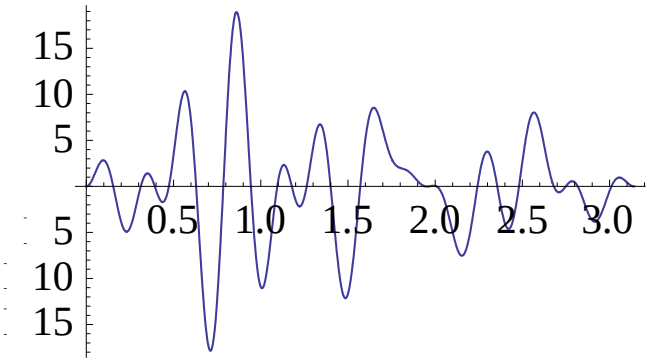
$$-2\sin(2t) + 6\sin(2t) + 7\sin(2t) = \sin(2t) - 3\sin(2t) + 14\sin(2t) = 12\sin(2t)$$



3. In den drei folgenden Plots sehen wir immer zugleich das ursprüngliche Audiosignal und das mit diesem Audiosignal amplitudenmodulierte Trägersignal. Das Audiosignal "umhüllt" das amplitudenmodulierte Signal, es ist bis zu einem gewissen Grad noch immer erkennbar.



4. Nun superponieren wir alle amplitudenmodulierten Trägersignale. Dies entspricht der elektromagnetischen Welle, die am Ort des Radioempfängers eintrifft.



Wie kann man das ursprüngliche Audiosignal $as2(t)$ aus diesem Wellengemisch wieder herausfinden?
Wir müssen dabei ganz genau zwischen physikalischen und mathematischen Problemen unterscheiden.
Der Empfangsapparat muss die Frequenzen der Frequenzbänder aller Sender bis auf diejenigen des Frequenzbandes des gewünschten Senders ausblenden können - das gehört zur Elektrotechnik.
Ferner muss der Empfangsapparat entweder analog oder digital empfangene Wellen (im mathematischen Sinne) integrieren können - auch dies gehört in die Elektrotechnik.

Wir wenden uns nun der mathematischen Problembehandlung zu. Alles Folgende beruht auf dem Additionstheorem für die Cosinunsfunktion und der Berechnung von Fourierkoeffizienten.
Aus den beiden Formeln

$$\cos[(T-A)t] = \cos(At)\cos(Tt) + \sin(At)\sin(Tt)$$

$$\cos[(T+A)t] = \cos(At)\cos(Tt) - \sin(At)\sin(Tt)$$

erhalten wir die fundamentale Formel zur Verwandlung von Produkten der Art $\sin(Tt) \cdot \sin(At)$ in eine Differenz von Cosinusfunktionen:

$$\sin(At)\sin(Tt) = \frac{1}{2}(\cos[(T-A)t] - \cos[(T+A)t]$$

Aus mnemotechnischen (gedächtnisstützend) Gründen soll man bei T an eine riesige Trägerfrequenz (MHz bis GHz) und bei A an die vergleichsweise sehr kleine Frequenz eines Audiosignals denken (20Hz bis 20kHz). Die obige Darstellung zeigt auch wie breit das sogenannte Frequenzband eines Senders mit Trägerfrequenz T sein muss, wenn $20\mathrm{kHz}$ die maximale Audiofrequenz ist die gesendet werden soll. Das Frequenzband muss alle Frequenzen zwischen $\mathrm{T}\text{-}20\mathrm{kHz}$ und $\mathrm{T}+20\mathrm{kHz}$ enthalten und die Frequenzbänder verschiedener Sender dürfen sich nicht überlappen.

Die letzte Formel gestatte uns die Umformung des elektromagnetischen Signals (Superposition aller Sender) folgender Weise:

$$f(t) := [3\sin(t) - 2\sin(2t)] \cdot \sin(8t) + [5\sin(t) + 6\sin(2t)] \cdot \sin(16t) + [4\sin(t) + 7\sin(2t)] \cdot \sin(24t)$$

$$= 3\sin(t) \cdot \sin - 2\sin(2t) \cdot \sin(8t) + 5\sin(t) \cdot \sin(16t) + 6\sin(2t) \cdot \sin(16t) + 4\sin(t) \cdot \sin(24t) + 7\sin(2t) \cdot \sin(24t)$$

$$= 3/2(\cos[(8-1)\,t] - \cos[(8+1)t)] - 2/2(\cos[(8-2)t] - \cos[(8+2)t] + 5/2(\cos[(16-1)t] - \cos[(16+1)t]) + 6/2(\cos[(16-2)t] - \cos[(16+2)t]) + 4/2(\cos[(24-1)t] - \cos[(24+1)t]) + 7/2(\cos[(24-2)t] - \cos[(24+2)\mathrm{t}])$$

$$= 3/2(\cos[7\mathrm{t}] - \cos[9\mathrm{t}]) - 2/2(\cos[6\mathrm{t}] - \cos[10\mathrm{t}]) + 5/2(\cos[15\mathrm{t}] - \cos[17\mathrm{t}]) + 6/2(\cos[14\mathrm{t}] - \cos[18\mathrm{t}]) + 4/2(\cos[23\mathrm{t}] - \cos[25\mathrm{t}]) + 7/2(\cos[22\mathrm{t}] - \cos[26\mathrm{t}]).$$

Die mathematische Umformung des Signals $f(t)$ in eine Summe von Cosinusfunktionen (mit verschiedenen ganzzahligen Frequenzen) wird uns später helfen die Fourieranalyse zu verstehen! Die Fourieranalyse ist ein Bestimmungsverfahren, das die in einem Signal versteckten Cosinusfunktionen (allgemeiner harmonischen Funktionen) und ihre Koeffizienten liefert. Mit unserer Umformung haben wir dies bereits für unser Signal $f(t)$ geleistet. Es geht jetzt darum, zu zeigen, wie man die Koeffizienten der Cosinusfunktionen auf andere Art aus dem Sendesignal bestimmen kann, denn das Sendesignal liegt beim Empfänger ja gar nicht als mathematische Formel vor: weder in der Gestalt als amplitudenmoduliertes Sinussignal noch als Summe von Cosinusfunktionen. Das einlangende Signal müssen wir uns als physikalisch repräsentiert denken, z.B. sichtbar gemacht auf einem Oszillographen oder unsichtbar als physikalischen Schwingvorgang in einem elektrischen Schaltkreis. Ein so repräsentiertes Signal kann man nur auf analoge oder digitale Weise integrieren. Letztlich möchten wir aber die Koeffizienten als Zahlen kennenlernen.

Es wäre nicht schwer die numerische Integration mit den Funktionswerten aus obiger Wertetabelle zu beschreiben, und damit die Fourierkoeffizienten $\mathrm{b_k}$ in der Fourierreihe

$$f(t) = \sum_{k=1}^{\infty} b_k \cdot \cos(kt)\,dt$$

mit der Formel

$$b_k = {}^{2}/_{\pi} \int_0^{\pi} f(t) \cdot \cos(kt)\,dt$$

berechnen. Wir zeigen nun, dass diese Berechnung in unserem Beispiel die (bereits bekannten) Koeffizienten liefert.

Dazu benötigen wir die Werte einiger bestimmter Integrale über das Intervall $[0,\pi]$. Die Integration ist in allen Fällen ganz einfach und liefert:

$$\int_0^{\pi} \cos(kt) \cdot \cos(mt) \cdot dt = 0 \quad für\ alle\ k \neq m \qquad (1)$$

$$\int_0^{\pi} \cos(kt) \cdot dt = \begin{cases} 0, für\ k \neq 0 \\ \pi, für\ k = 0 \end{cases}$$

$$\int_0^{\pi} \cos^2(\mathrm{kt}) \cdot \mathrm{dt} = \int_0^{\pi} \sin^2(\mathrm{kt}) \cdot \mathrm{dt} = \frac{\pi}{2}, für\ alle\ k \qquad (2)$$

$$\int_0^{\pi} \sin(kt) \cdot \sin(mt) \cdot dt = 0 \quad für\ alle\ k \neq m$$

$$\int_0^{\pi} \sin(kt) \cdot dt = \begin{cases} 0, für\ alle\ geraden\ k \\ \frac{2}{k}, für\ ungeraden\ k \end{cases}$$

Wenn wir in Analogie zur Vektorgeometrie des zwei - und dreidimensionalen Raumes die Funktionen $\cos(\mathrm{t})$, $\cos(2\mathrm{t})$, $\cos(3\mathrm{t})$, … , $\cos(\mathrm{kt})$, u.s.w. als Basisvektoren eines Vektorraumes von reellen Funktionen g: $[0, \pi] \rightarrow \mathrm{P}$ auffassen und die Zahl

$$g(t) * h(t) = \int_0^{\pi} g(t) * h(t) dt$$

als Skalarprodukt auffassen, dann sind die Funktionen $\cos(\mathrm{kt})$ und $\cos(\mathrm{mt})$ gemäß Formel (1) orthogonal zueinander und jede dieser Cosinusfunktionen hat wegen (2) die Norm $\sqrt{\frac{\pi}{2}}$.
Die Norm entspricht der Länge eines Vektors in der Geometrie und ist die Quadratwurzel aus dem Skalarprodukt des Vektors mit sich selbst.

Nun zurück zu unserem eigentlichen Problem der Berechnung der $\mathrm{b_k}$ aus dem Signal $\mathrm{f(t)}$.
Unser Empfänger kann die Frequenzen in der Nähe der Trägerfrequenzen $\mathrm{tf1}=8$ und $\mathrm{tf3}=24$ ausblenden, sodass wir nur mehr das reduzierte Signal

$$f_{red}(t) = \frac{5}{2}[\cos(15t) - \cos(17t)] + \frac{6}{2}[\cos(14\mathrm{t}) - \cos(18t)]$$

analysieren müssen (Siehe: mittlere Zeile in der Summe der Cosinusterme weiter oben).
Es genügen die Skalarprodukte von $f_{red}(t)$ mit den Funktionen $\cos(14\mathrm{t})$, $\cos(15\mathrm{t})$, $\cos(16\mathrm{t})$,$\cos(17\mathrm{t})$ und $\cos(18\mathrm{t})$ aus dem kleinen Frequenzband $[14, 18]$ um die Trägerfrequenz $\mathrm{tf2}$ zu berechnen.

V. Einblicke in die Fourieranalyse von Signalen

Durch Amplitudenmodulation können viele niedrig-frequente Signale (z.B. Audio, 20Hz-20kHz) auf sehr hoch-frequenten Trägerwellen (10MHz-10GHz) übertragen werden

Hauptproblem: Das Ausblenden unerwünschter Frequenzbänder beim Empfänger und der Empfang der amplitudenmodullierten Welle. Das Ausblenden unerwünschter Frequenzen bewerkstelligt der Empfangsapparat physikalisch, aber das Wiederherstellen der ursprünglichen Audiosignale aus dem amplitudenmodulierten Sendesignal erfordert Mathematik und dieser Vorgang kann mathematisch beschrieben werden. Das mathematische Hauptwerkzeug ist die Frequenzanalyse mit Hilfe der Fourieranalysis. Das Ziel der Frequenzanalyse besteht in der Darstellung des Sendesignals durch eine unendliche Reihe von Sinus- und Cosinus-Wellen mit ganzzahligen Frequenzen.

VI. Formelsammlung[1]:

Reflexionszeitpunkte

$$t_i^z = \frac{w(0) + c \cdot t_i}{c - w}$$

Reflexionsorte

$$w(t_i^z) = \frac{c}{c - w} \cdot (w(0) + w \cdot t_i)$$

Eintreffzeiten der Echos

$$t_i^* = \frac{2w(0)}{c - w} + \frac{c + w}{c - w} \cdot t_i$$

Zusammenhang zwischen $t_i, t_i^z;\ t_i^*$

$$t_i^z = \frac{t_1 + t_1^*}{2}$$

$$w(t_i^z) = c \cdot \frac{t_i^z - t_i}{2}$$

Zeitliche Echodistanz

$$\Delta t^* = \frac{c + w}{c - w} \cdot \Delta t$$

Geschwindigkeit aus 2 Messungen

$$w = c \cdot \frac{\Delta t^* - \Delta t}{\Delta t^* + \Delta t}$$

Ort aus 2 Messungen

$$w(0) = c \cdot \frac{t_1^* t_2 - t_2^* t_1}{\Delta t^* + \Delta t}$$

Laserdaten

n… Anzahl der Impulse

PW… Pulsweite

$t_i = (i - 1) \cdot \Delta t$

Bedingung für das 1. Echo

$$PW \leq t_1^* \leq \Delta t - \frac{c + w}{c - w} PW$$

Bedingung für das n-ten Echo	$t_n + PW \leq t_n^* t_n \leq +PW - \frac{c+w}{c-w} PW$
Zeit des 1. bis zum n-ten Echo	$(n-2)\Delta t + \left(1 + \frac{c+w}{c-w}\right) \cdot PW \leq t_n^* - t_1^*$ $\leq n \cdot \Delta t - \left(1 + \frac{c+w}{c-w}\right) PW$
Geschwindigkeit w1 bei minimalen Δt^*	$w_1 = c \cdot \frac{2PW - \Delta t^*}{(2n-3)\Delta t}$
Geschwindigkeit w_2 maximalen Δt^*	$w_2 = c \cdot \frac{\Delta t - 2PW}{(2n-1)\Delta t}$
Erlaubter Geschwindigkeitsbereich	$w_1 \leq w \leq w_2$
Obere Grenze für erlaubten Ort bei gegebener Geschwindigkeit	$w(0) \leq min\left\{-\frac{c+w}{2} PW + \frac{c-w}{2}\Delta t, -\frac{c+w}{2} PW - \frac{2n-1}{2} \cdot w \cdot \Delta t + \frac{c}{2}\Delta t\right\}$

[1] alle Formeln wurden selbständig hergeleitet

Projekt: Navigation

Florian Conrad Maximilian Köller Markus Krainz Klara Saller
Robert Schwarzl Martin Stadler

6.–12. Februar 2011

1 Einleitung und Problemstellung

Unser Projekt beinhaltet vier ähnliche Problemumgebungen: ein Labyrinth, einen Berg, die eigene Wohnung und ein Straßennetz.

- Die Aufgabe im Labyrinth besteht darin, den kürzesten Weg von einem Anfangspunkt zu einem beliebigen anderen Punkt zu finden. Wir gehen von einem zweidimensionalen Objekt aus, das aus beliebig vielen Kästchen besteht. Jedes dieser Kästchen hat vier Seiten, die entweder passierbar oder unpassierbar sind. Unpassierbare Seiten werden als Wände definiert. Wir unterscheiden zwischen zwei Arten von Labyrinthen: jenem mit einem möglichen Lösungsweg (maze) und jenen mit mehreren (strange maze).
- Beim Berg ist nicht der kürzeste Weg der entscheidende, sondern der kostenärmste bzw. der am wenigsten anstrengende Weg. Im Gegensatz zum Labyrinth ist der Berg dreidimensional und die Höhenkoordinaten jedes einzelnen Punktes sind gegeben. Auch variiert der Kostenaufwand um von einem Punkt zum nächsten zu gelangen je nach Steigung.
- Das dritte Problem beschäftigt sich mit der eigenen Wohnung und zwar mit dem Transport eines Klaviers durch die möblierte Umgebung. Man betrachtet hierbei den Grundriss der Wohnung als eine 2D-Bild. Allerdings gilt es die Form des Klavieres in weiterer Folge zu berücksichtigen. Die Möglichkeit, das Klavier auch zu drehen, erschwert die Modellierung zusätzlich.
- Das Straßennetz stellt durch seine großen Datenmengen eine besondere Herausforderung des Projekts dar. Man kann sich es als gigantisches Labyrinth vorstellen, bei welchem noch zusätzlich Straßenlängen, Geschwindigkeitsbegrenzungen und Sonderfälle wie Einbahnstraßen berücksichtigt werden müssen. Diese Art der Navigation ist wohl die uns bekannteste, da sie von einer großen Menschenmenge täglich in Form von Navigationsgeräten im Auto oder über das Internet, mit Seiten wie zum Beispiel 'google maps', genutzt wird.

Unsere Projektziele waren ein einheitliches Modell und verschiedene Lösungsstrategien zu finden und diese am PC mithilfe von MatLab umzusetzen.

2 Chronologische Entwicklung der Lösungsansätze

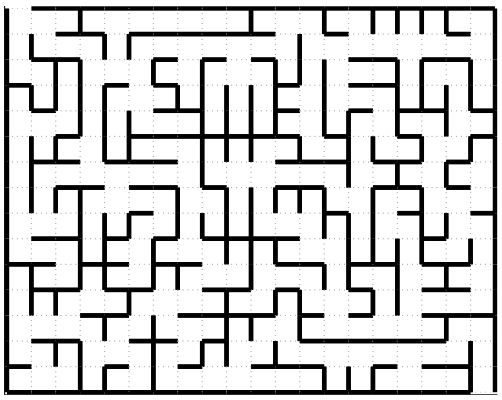
Am Anfang hatten wir vor, die vier Probleme nacheinander zu bearbeiten und Lösungsstrategien für sie zu entwickeln. Der Einfachheit wegen beschäftigten wir uns als erstes mit dem Labyrinth, da am wenigsten Zusatzfaktoren zu berücksichtigen waren. Mit der Zeit fielen uns die zahlreichen Gemeinsamkeiten der vier Probleme auf und wir begannen unsere Ideen auf alle Beispiele zu erweitern.

2.1 Rechtsregel & CutLoops-Algorithmus

Den ersten Lösungsansatz erhielten wir durch die uns bekannte Rechtsregel, mit der man von jedem Punkt eines Labyrinths zu jedem anderen erreichbaren Punkt finden kann, auch wenn dies viel Anstrengung und lange Wegstrecken bedeuten kann. Sie liefert immer einen möglichen, aber in den seltensten Fällen den kürzesten Weg.

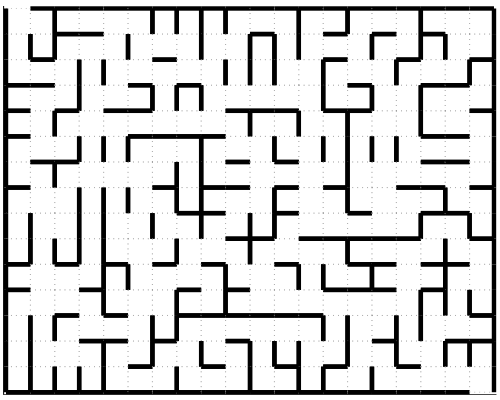
Die Rechtsregel besagt, dass man bei jeder Weggabelung nach rechts abbiegt, man hält sich also immer an der rechten Mauer.

Da man bei Anwendung dieser Regel gewisse Punkte im Labyrinth öfters passiert, nämlich genau dann, wenn man aus einer Sackgasse wieder herauskommt, entstehen gelaufene 'Schlaufen', welche für den Weg zum Zielpunkt unnötig wären. Wir entschieden diese 'Schlaufen' mit dem CutLoop-Algorithmus zu eliminieren. Dieser besagt, dass alle mehrfach passierten Kästchen der Ursprung und das Ende einer 'Schlaufe' sein müssen und, dass alle jene Einträge zwischen diesen zwei identen Werten, und somit die 'Schlaufe', gelöscht werden musste. Bei einer Sorte Labyrinthe ('maze') (Abb.1) erhielten wir ein positives Ergebnis, den kürzesten Weg. Mazes haben eine Wegbreite von genau einem Kästchen (Punkt) und enthalten keine Inseln, folglich sind alle Wände mit der Außenwand verbunden. Bei allen anderen Labyrinthen ('strange maze') (zum Beispiel Abb.2), bei denen mehrere Wege zum Zielpunkt führen, ohne dass sie einen Punkt mehrmals passieren müssen, führt diese Technik jedoch nicht zwangsläufig zum kürzesten Weg.



'Maze'

Abbildung 1: Labyrinth mit einem Lösungsweg



'Strange maze'

Abbildung 2: Labyrinth mit mehreren Lösungswegen

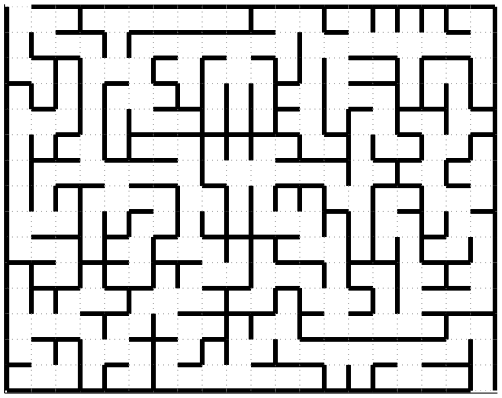


Abbildung 3: Ursprüngliches maze

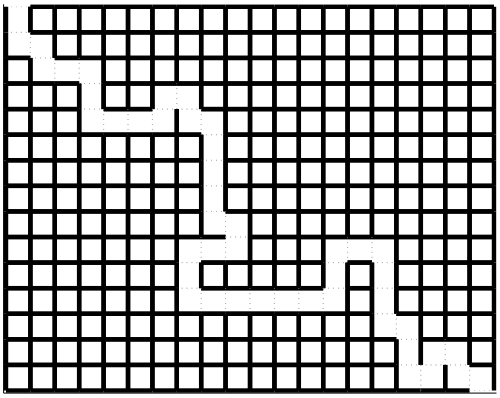


Abbildung 4: Maze nach Schritt 1

2.2 TAP-Algorithmus (Try All Possibilities)

Der TAP-Algorithmus versucht von einem Ausgangspunkt im Labyrinth jedes von dort aus erreichbare Kästchen, und von diesen wiederum jedes von diesen erreichbare Kästchen, zu begehen. Er probiert erst alle Wege der Länge 1, danach der Länge 2 bis zur Länge n, welche dann der Länge des kürzesten Weges zum Ziel entspricht, aus. Diese möglichen Wege steigen von der Weglänge abhängig exponentiell, somit wird dieser Algorithmus, der jeden möglichen Weg zum Ziel berechnet, mit zunehmender Weglänge in 'mazes' sowie in 'strange mazes', immer ineffizienter. Dennoch stellt dieser Algorithmus unsere erste, auf alle Labyrinthe anwendbare, Lösung dar.

2.3 FillUp-Algorithmus

Ein anderer Lösungsansatz war der von uns in zwei Schritten entwickelte FillUp-Algorithmus. Dieser schließt in 'mazes' alle Sackgassen, indem er zusätzliche Wände in das Labyrinth einbaut. Der erste Schritt (Abb.3 und 4) ist dem jeweils letzten Kästchen einer Sackgasse, welches bereits von drei Wänden umschlossen ist, eine vierte hinzuzufügen. Somit werden die jeweiligen Nachbarkästchen auch zu Kästchen mit drei Wänden, welche im nächsten Schritt ebenfalls geschlossen werden, außer der Anfang der Sackgasse ist bereits erreicht. So wird der kürzeste und einzige Lösungsweg sowohl visuell als auch mathematisch herauskristallisiert, da alle Wege bis auf den einen, völlig geschlossen sind. Daraufhin wenden wir den TAP-Algorithmus an, der nun den unverzweigten, einzigen Weg findet und ausgibt.

Da der FillUp-Algorithmus, welcher zu dem Zeitpunkt erst aus Schritt eins bestand, nicht ausreichend für das Lösen von 'strange mazes' war, entwickelten wir einen Schritt zwei. Der zweite Schritt (Abb.5 und 6) besteht darin, dass bei einem Kästchen, welches zwei aneinander liegende Wände besitzt, und das diagonal benachbarte, der Ecke gegenüber liegende Kästchen die gleichen zwei Wände nicht besitzt, alle noch offenen Seiten geschlossen werden. Von diesem Schritt, in Kombination mit dem ersten (Abb.7), erhofften wir uns, dass sämtliche 'strange mazes' und offenen Flächen deutlich verein-

facht wären und effizient mit dem TAP-Algorithmus gelöst werden könnten, was jedoch nicht immer zutraf.

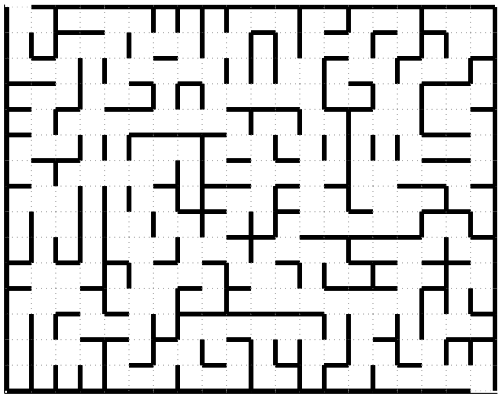


Abbildung 5: Ursprüngliches strange maze

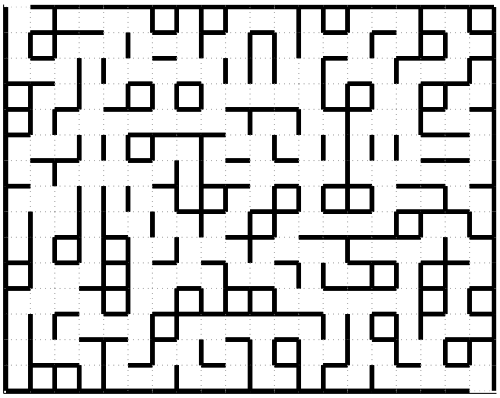


Abbildung 6: Strange maze nach Schritt 2

2.4 Der Spezielle Flow-Algorithmus

Auf diesen Lösungsweg kamen wir mit dem Gedanken, es wäre möglicherweise von Vorteil die Richtung von Start- nach Zielpunkt auf dem gesamten Weg mitzuberücksichtigen und nicht einfach willkürlich von jedem Kästchen aus eine Richtung zu wählen bzw. alle Möglichkeiten durchzuprobieren. Erst zeichneten wir also in jedes Kästchen des Labyrinths einen Pfeil, welcher von diesem aus genau auf den Zielpunkt zeigte. Das bleibende Problem hierbei war jedoch, dass Hindernisse bzw. die Wände des Labyrinths, von den Pfeilrichtungen nicht berücksichtigt wurden. Allerdings brachte uns diese Skizze die zündende Idee.

Wir sahen, dass der gleichmäßige 'Fluss' der Pfeilrichtungen einfach nur an die Hindernisse angepasst werden musste, um von jedem beliebigen Punkt den kürzesten Weg zum Zielpunkt zu erhalten. Wir begannen mit dem Pfeil vor dem Zielfeld, wessen Richtung als erste eindeutig definiert werden konnte. Dieses Feld zeigte direkt auf den Ausgang bzw. das Ziel. Danach richteten wir alle Pfeile von den benachbarten Kästchen des einen Kästchens, welche nicht durch Wände getrennt waren, auf dieses aus. Das gab uns bereits den kürzesten Weg der ersten Felder zum Zielpunkt. Daraufhin richteten wir alle weiteren Pfeile immer jeweils auf die bereits definierten Pfeile, sodass wir nun einen optimal an das Labyrinth angepassten 'Fluss' aus Pfeilen erhielten (Abb.8). Der kürzest mögliche Weg konnte nun für jedes Kästchen gefunden werden. Dieser Algorithmus war nun der erste, welcher auf jegliche Art von Labyrinthen anwendbar war.

Um diese Methode für einen Computer verständlich zu machen, ersetzten wir alle Pfeile mit aufsteigenden Zahlen. Das Zielfeld bekam den Wert 1, alle angrenzenden Felder den Wert 2, alle an diese Felder angrenzenden Felder den Wert 3 und so weiter. Der programmierte Algorithmus musste nun nur mehr von einem beliebigen Anfangskästchen

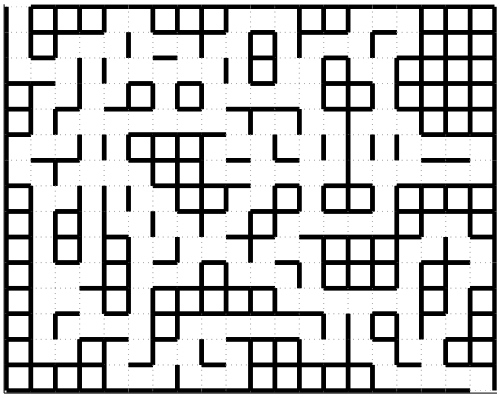


Abbildung 7: Strange maze nach Schritt 1 und 2

mit dem Wert n, welcher bereits die Länge des kürzest möglichen Weges anzeigt, immer auf das benachbarte Kästchen mit einem kleineren Wert als n (im Fall von Labyrinthen genau $n-1$) gehen, solang bis er nach genau n Zügen am Zielfeld ankäme (Abb.9).

3 Resultate

3.1 Einheitliches Modell

Während der Arbeiten an dem Speziellen Flow-Algorithmus wurde uns bewusst, dass wir diesen Algorithmus auf alle Problemstellungen anwenden können, vorausgesetzt wir fänden ein einheitliches Modell. Dies führte uns zu der Frage nach den Gemeinsamkeiten aller Probleme. Alle Probleme können auf folgendes reduziert werden:

- Gegeben sei:
  - Die Menge aller Punkte K,
  - weiter, die Menge aller benachbarten Punkte E
  - und eine Funktion $\omega : E \to \mathbf{R}_{\geq 0}$,
  - welche die Kosten angibt, von einem Punkt zu einem anderen zu gelangen.
- Gesucht ist:

  Ein Weg $W = [K_1, \ldots, K_n], \quad K_1 = A, \quad K_n = B$
  dessen Kosten gleich das Minimum der Kosten aller Wege von A nach B ist.

$$\underbrace{\sum_{i=1}^{n-1} \omega(K_i, K_{i+1})}_{\text{Kosten des Weges}} = \min_{\substack{X=[X_1,\ldots,X_m] \\ X_1=A,\ X_m=B}} \sum_{i=1}^{m-1} \omega(X_i, X_{i+1})$$

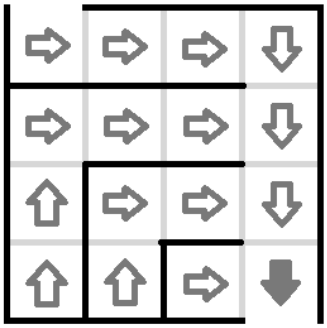


Abbildung 8: Erste Darstellung des Flows

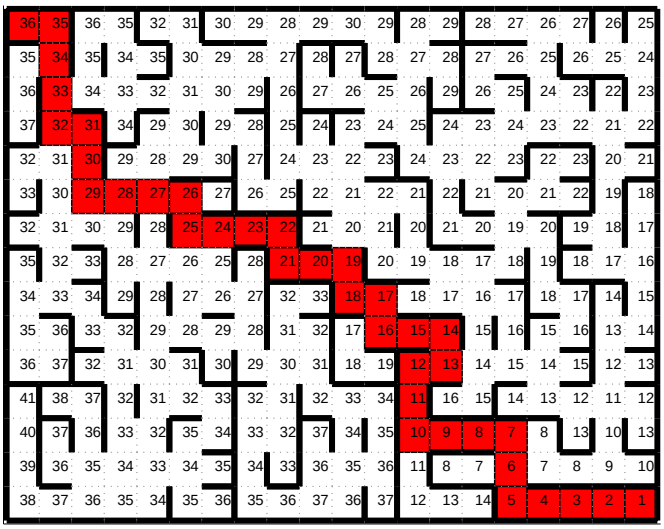


Abbildung 9: Der durchgezählte flow

3.2 Vom Problem zum Modell

3.2.1 Labyrinth

Das einzige, das wir mit dem Labyrinth machen mussten war offene Seiten und Wände mit Kosten zu versehen. Wir wählten die Kosten 1 um eine offene Wand zu passieren und die Kosten 0 bzw. ∞ um die Unüberwindbarkeit einer Wand zu beschreiben.

3.2.2 Berg

Das nächste Problem war die Routenplanung für eine Bergtour. Hier gilt es nicht unbedingt den kürzesten, sondern den am wenigsten anstrengenden Weg zu finden. Wir mussten überlegen in welcher Weise wir den Berg mathematisch darstellen könnten. Wir hatten die Idee Punkte auf der Bergoberfläche zu verteilen und ihre Abstände zueinander zu berechnen. Diese Idee brachte uns aber nicht weiter, daher mussten wir eine andere Strategie entwickeln.

Den Berg kann man auch als 'flachgedrückte' 2D-Ebene sehen. Diese ist eingeteilt in Kästchen, ähnlich wie ein Labyrinth, wobei jeder Punkt am Berg zu einem Kästchen wird, welches einen bestimmten Höhenwert hat. Auch gibt es keine herkömmlichen 'Wände' sondern jede Seite eines Kästchens bekommt einen errechneten Wert, welcher den nötigen Aufwand beschreibt, um diese Seite zu passieren. Um auf diesen Wert zu kommen erstellten wir folgende Funktion:

$$S(x) = \sqrt{1 + (xn)^2}$$

In dieser Funktion beschreibt x den Höhenunterschied zwischen den zwei benachbarten Kästchen und n beschreibt den Anstrengungsfaktor, welcher ausdrückt um wie viel

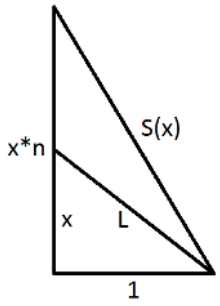


Abbildung 10: $S(x)$ visualisiert

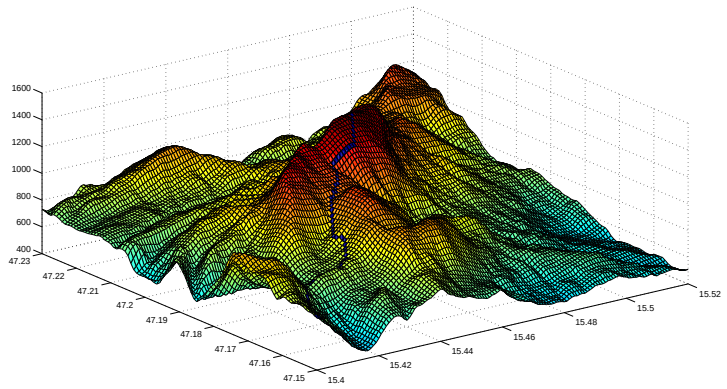


Abbildung 11: Berg mit optimalem Weg, errechnet mit dem Allgemeinen Flow-Algorithmus (Schöckl)

anstrengender ein Meter in die Höhe, als ein Meter in die Breite zu bewältigen ist. Der Funktionswert $S(x)$ (Abb.10) gibt den Aufwand an.

3.2.3 Straßenkarte

Die Straßenkarten des OpenStreetMap-Projekts liegen in einem Web-ähnlichen Format vor, was nicht gerade vorteilhaft für die Verarbeitung mit MatLab® ist. Darum schrieben wir ein Programm zur Verarbeitung der Daten; es liest die XML-Daten ein, berechnet die Abstände zwischen den Straßenpunkten und schreibt das Resultat wieder in Dateien, die mit MatLab® gut aufgenommen werden können.
Das downloadbare Format ist prinzipiell in zwei Teilen aufgebaut: Im Ersten werden Wegpunkte mit Koordinaten definiert, im Zweiten finden sich verschiedene Objekte (bzw. Wege), die durch mehrere Punkte verbunden sind. Im Programm werden die beiden Listen eingelesen; anschließend werden aus den Objekten die Straßen gefiltert, da Flüsse und Gebäude mit gebräuchlichen städtischen Verkehrsmitteln (ausgenommen natürlich Amphibienfahrzeuge) nicht befahrbar sind. Nach einer Umordnung nach IDs, die vordergründig für Performance und Sichtbarkeit wichtig sind, werden die verwendeten Punkte in die eine, und die Wege von einem Punkt zum nächsten, in eine andere Datei geschrieben. Außerdem werden automatisch die Punkte erkannt, die in mehreren Wegen vorkommen und somit Kreuzungen darstellen.

3.3 Der Allgemeine Flow-Algorithmus

Um den idealen Weg von A, dem Startpunkt, nach B, dem Endpunkt, zu finden, bedarf es zwei großer Schritte.

Der erste Teil besteht darin, auf jedem Punkt, wie es der Spezielle Flow-Algorithmus im Labyrinth getan hat, den genauen Wert der addierten Anstrengungswerten für den

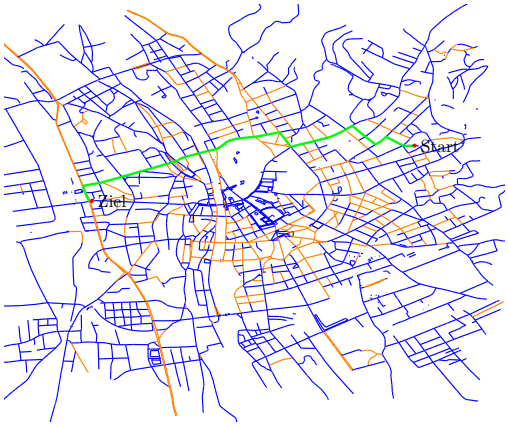


Abbildung 12: Navigation auf Straßenkarte (Graz)

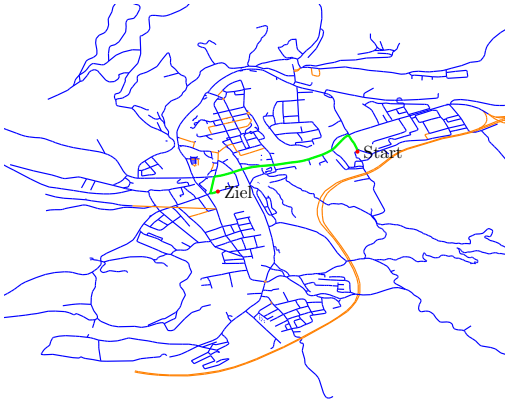


Abbildung 13: Navigation auf Straßenkarte (Leoben)

optimalen Weg anzugeben. Dies ist um einiges schwieriger als im Labyrinth da es nun verschiedene 'Kosten', um Seiten bzw. Verbindungen zu passieren, gibt.

Teil 1:

Der Algorithmus beginnt am Zielpunkt B, der anfangs der einzige Punkt in der Menge S ist, und berechnet die Kosten um von allen benachbarten Kästchen C auf dem 'direkten' Weg (über die trennende Seite beziehungsweise die direkte Verbindungsstrecke) zu ersterem Punkt B, zu kommen. Diese errechneten Werte kommen alle in eine Menge U, welche alle 'unsicheren Wegpunkte' beinhaltet. Dann wird die Menge U geprüft und das Kästchen mit den geringsten Kosten (wenn dies auf mehrere Kästchen zutrifft, gilt folgendes für alle diese) wird in die Menge S eingetragen und aus der Menge U gelöscht. Dies geschieht, da dieser Weg mit Sicherheit der kürzeste Weg von jenem Punkt zum Ziel ist, weil jeder andere Weg von diesem Punkt zum Ziel mindestens ein Element der Menge U passieren müsste und somit zwangsläufig größere Kosten als der ursprüngliche Weg haben müsste. Nun wird dieser Vorgang für alle angrenzenden Punkte des Punktes, der zuletzt zur Menge S hinzugefügt worden ist, wiederholt. Wiederum kommen alle möglichen angrenzenden Kästchen, in die Menge U, in der sich aber auch noch die unsicheren Kästchen des vorherigen 'Durchgangs' befinden. Die Kosten auf der Menge U werden erneut geprüft und das Kästchen mit dem kleinsten Wegwert wird zur Menge S hinzugefügt. Diese Prozedur wird solange wiederholt, bis jeder Punkt in der Menge S enthalten ist.

Diese Prozedur lässt sich auch wie folgt formal ausdrücken:

1. Setze:

$$K_x = B, \quad (B = \text{Zielpunkt}), \quad i = 1, \quad K_x \in S_i, \quad S_i = \{K_x\}$$

$\gamma_i : S_i \to \mathbf{R}_{\geq 0}, \quad \gamma_i(K_x) = 0, \quad N_i = \emptyset$

2. Wenn $S_i = K$ gehe zu Schritt (4.), sonst:

   Suche alle K_y für die gilt $K_y \in K$ und $(K_y, K_x) \in E$

   Setze $U_{i+1} = (U_i \cup \{K_y | (K_y, K_x) \in E\}) \backslash S_i$

   Definiere $\lambda_{i+1} : U_{i+1} \to \mathbf{R}_{\geq 0}$ für alle K_y wie folgt:

$$\lambda_{i+1}(K_y) = \begin{cases} \gamma_i(K_x) + \omega(K_y, K_x) & \text{falls } K_y \notin U_i \\ \lambda_i(K_y) & \text{falls } K_y \in U_i \\ & \text{und } \gamma_i(K_x) + \omega(K_y, K_x) \geq \lambda_i(K_y) \\ \lambda_i(K_x) + \omega(K_y, K_x) & \text{falls } K_y \in U_i \\ & \text{und } \gamma_i(K_x) + \omega(K_y, K_x) < \lambda_i(K_y) \end{cases}$$

3. Suche in U_{i+1} ein K_v für das gilt:

$$\lambda_{i+1}(K_v) = \min_{K_o \in U_{i+1}} \lambda_{i+1}(K_o)$$

   Setze $S_{i+1} = S_i \cup (K_y)$ und definiere $\gamma_{i+1} : S_{i+1} \to \mathbf{R}_{\geq 0}$

$$\gamma_{i+1}(K) = \begin{cases} \gamma_i(K) & \text{falls} \quad K \in S_i \\ \lambda_{i+1}(K) & \text{falls} \quad K = K_v \end{cases}$$

   Setze: $\quad i \leftarrow i + 1$

   Gehe zu Schritt (2.)

4. Ausgabe Funktion $\gamma : K \to \mathbf{R}_{\geq 0}$: $\gamma = \gamma_i$

Teil 2:

Der zweite Teil beschäftigt sich mit dem tatsächlichen Finden des idealen Weges von einem beliebigen Punkt aus, anhand der durch Teil 1 vorhandenen Daten. Man hat einen Wert für jeden Punkt (aus der Menge S, welche aus Teil 1 stammt), der die Kosten für den kürzesten Weg anzeigt, und die Kosten um jede Seite beziehungsweise Verbindung zu passieren, gegeben. Um den idealen Weg von A nach B (Anfangs- nach Zielpunkt) zu finden beginnt man bei Punkt A. Man prüft alle Nachbarpunkte von Punkt A und zieht den jeweilig eingeschlossenen Passierwert ab. Über den Punkt, bei welchem das

Ergebnis dieser Differenz der Wert des Punktes selbst ist, verläuft der gesuchte ideale Weg. Von diesem Punkt aus prüft man nun wieder alle Nachbarkästchen und zieht vom Wert in diesem Punkt wieder die jeweilig eingeschlossenen Passierkosten ab. Man findet auf diesem Weg den nächsten Punkt der auf dem idealen Weg liegt. Diese Technik muss man anwenden bis man am Endpunkt B, mit dem Wert 0, angekommen ist. Die somit passierten Punkte beschreiben den idealen Lösungsweg und können vom Allgemeinen Flow-Algorithmus ausgegeben werden.

Diese Prozedur lässt sich wieder auch formal ausdrücken:

Sei $\gamma : K \to \mathbf{R}_{\geq 0}$, wie in Teil 1 definiert, dann:

1. Setze $K_i = A, \quad i = 1$
2. Falls $K_i = B$, dann ist $W = [K_1, \ldots, K_i]$ gefunden, sonst:

   Wähle K_{i+1} so, dass $(K_i, K_{i+1}) \in E$ und $\gamma(K_{i+1}) = \gamma(K_i) - \omega(K_i, K_{i+1})$

   Setze $i \leftarrow i + 1$ und fahre mit Schritt 2 fort.

4 Realisierung am PC

Parallel zur Findung neuer Algorithmen versuchten wir diese auch auf dem Rechner zu verwirklichen. In MatLab fanden wir ein sehr mächtiges Werzeug, das uns erlaubte die Algorithmen relativ einfach auf dem Rechner zu implementieren. Einzig bei der Effizienz stießen wir an unsere Grenzen, unter anderem weil MatLab eine Skriptsprache ist, weil unsere Laptops mit ihren 2 GB Arbeitsspeicher nicht in der Lage waren größere Datenmengen (zum Beispiel eine hochaufgelöste Karte des Schöckls oder eine Straßenkarte von Graz) effizient zu verarbeiten und weil wir uns erst mit den Feinheiten dieser Sprache vertraut machen mussten. MatLab ist nämlich insgesamt eher an Mathematik orientiert als an der Architektur von Computern. Außerdem stehen sehr viele mathematische Funktionen und diverse Operationen, die man bei der Umsetzung von Algorithmen am Computer benötigt, zur Verfügung. Bei Beachtung dieser Eigenheiten ist ein geschriebenes Programm recht einfach zu verstehen, jedoch bei Nichtbeachtung und falscher Verwendung führt das zu sehr langsamen und ineffizienten Programmen, die dann auch nicht wirklich praxistauglich sind.

4.1 Labyrinth

Die auf das Labyrinth anwendbaren Algorithmen am Rechner zu implementieren gestaltete sich relativ einfach. Die computergenerierten Labyrinthe lagen in einer Matrix vor, die die Nachbarschaftsbeziehungen eines jeden Kästchens zu seinen Nachbarn beschrieb. Dies sah aus wie in Tabelle 1, die Matrix die dem Beispiel-Labyrinth (Abb. 14) zu Grunde liegt. Die Indizes rechts stehen für die Nummer eines jeden Kästchens des Labyrinths und die Einsen und Nullen rechts daneben an geben an, wo sich relativ zum Kästchen eine Wand befindet.

Indizes	Wand Oben	Wand Rechts	Wand Unten	Wand Links
1	1	1	0	1
2	0	1	0	1
3	0	0	1	1
4	1	0	0	1
5	0	1	0	1
6	0	1	1	0
7	1	1	0	0
8	0	1	0	1
9	0	1	1	1

Tabelle 1: Nachbarschaftsbeziehungen der Kästchen

Die Umlegung sämtlicher erdachter Algorithmen geschah ziemlich speziell und wurde direkt auf die vorliegenden Daten des Labyrinths zugeschnitten. So kamen für Labyrinthe immer eigene und spezielle Programme zum Einsatz. Nur der TAP benutzte einen etwas allgemeineren Datensatz als Eingang, einen Graphen. Das ist eine Matrix, die für jeden Punkt beschreibt zu welchen Punkten er benachbart ist. Dies würde für das Beispiel Labyrinth (Abb. 14) so aussehen wie in Tabelle 2.

Indizes	1	2	3	4	5	6	7	8	9
1	0	1	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0
3	0	1	0	0	0	1	0	0	0
4	0	0	0	0	1	0	1	0	0
5	0	0	0	1	0	1	0	0	0
6	0	0	1	0	1	0	0	0	0
7	0	0	0	1	0	0	0	1	0
8	0	0	0	0	0	0	1	0	1
9	0	0	0	0	0	0	0	1	0

Tabelle 2: Der Graph der Nachbarschaftsbeziehungen der Kästchen

Im Laufe der Woche entdeckten wir immer mehr MatLab-eigene Funktionen und programmierten unsere Funktionen immer MatLab-gerechter. Aufgrund der Einfachheit von Labyrinthen spielte es eigentlich keine Rolle, wie effizient unsere Programme arbeiteten. So legten wir nur Wert darauf das Programm möglichst zuverlässig zu machen.

4.2 Berg

Beim Berg wurde zum ersten Mal der allgemeine Flow verwendet, der ja auch beim Nachdenken über eine Lösungsstrategie für den Berg entstanden war. Zuerst mussten die Daten des Berges, die in einer Matrix mit den Höhenwerten vorlag, in einen Graphen

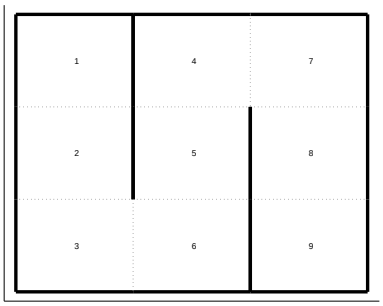


Abbildung 14: Ein Beispiel Labyrinth

umgerechnet werden. Konkret verwendeten wir die SRTM-3-Daten des Grazer Hausberges, dem Schöckl. Diese Daten werden vom USGS frei zur Verfügung gestellt und haben in unseren Breitengraden eine Auflösung von 90 m Nord-Süd und 60 m in West-Ost bei einer Höhengenauigkeit von 6 m. Jetzt mussten wir diese Matrix erst einmal in eine brauchbare Form für Flow bringen. Dafür schrieben wir eine Funktion, die uns, wie schon beschrieben, den Aufwand von einem Punkt zum nächsten zu kommen errechnete und mit diesem Werten den Graphen befüllte. Der Graph beschrieb somit nicht nur die Nachbarschaftsbeziehung, sondern auch gleichzeitig den Aufwand von einem Punkt zum nächsten zu kommen. Hier kam ein spezieller Matrixtyp zum Einsatz, da sonst riesige Datenmengen entstehen würden, die der Arbeitsspeicher nicht fassen könnte, und zwar die Sparse-Matrix. Das Besondere dabei ist, dass nicht alle Werte der Matrix gespeichert werden, sondern nur die, die nicht null sind, trotzdem kann man eine Sparse-Matrix genauso wie eine normale Matrix verwenden. Dies bringt eine enorme Verkleinerung des benötigten Speichers, da ein Punkt am Berg ja nur 8 Nachbarn haben kann. Die ursprüngliche Matrix des Schöckls aus SRTM-Daten hat eine Ausdehnung von 145x97 Feldern. Dies ergibt ein Produkt von 14065 Feldern. Also hat der Graph eine Ausdehnung von 14065x14065 Feldern. Die erste Implementierung von Flow hatte hier massive Effizienzprobleme und konnte nur ein 100stel der Felder des Schöckls in einer sinnvollen Zeit bearbeiten. Sie funktionierte richtig, jedoch zu langsam. Dies war das erste Mal, dass wir die Effizienz einer unserer Programme in den Vordergrund stellen mussten. Un-

ser Programm war deswegen so langsam, weil es viel zu viele sinnlose Abfragen machte und außerdem viel zu wenig an MatLabs Feinheiten angepasst war. Über zwei Schritte näherten wir uns immer mehr der idealen Lösung an und konnten schließlich den Schöckl in voller Auflösung innerhalb von einer halben Minute bearbeiten. Wir mussten unser Programm sehr genau auf die Eigenheiten von MatLab zuschneiden und ersetzten dabei viele der eigenen Operationen durch MatLab-eigene.

4.3 Straßenkarte

Um die Verarbeitung des realen Straßennetzes zu ermöglichen, griffen wir auf die Daten von OpenStreetMap zurück. Die Dateien sind mit freier Lizenz verfügbar.
Die Daten werden vom Web-Dienst in einer speziellen Version des XML-Standards gespeichert. Das Format besteht grob aus zwei Teilen: Einerseits finden sich im Anfangsbereich Koordinatenpunkte (bestehend aus Längen- und Breitengraden), andererseits werden mit diesen als Basis auf der zweiten Seite verschiedene Arten von Wegen, Gebäuden, Flüssen und vielem mehr dargestellt.

```
/* Koordinatenpunkte */
<node id="418731727" lat="46.8335555" lon="15.5459667"/>
<node id="418731047" lat="46.8335501" lon="15.5468256"/>

/* Wegdefinition */
<way id="35814197" user="StefanTiran" uid="56870"
visible="true" version="1" changeset="1485927"
timestamp="2009-06-11T11:55:41Z">
 <nd ref="418731727"/>
 <nd ref="418731047"/>
 <tag k="highway" v="track"/>
</way>
```

Obwohl diese Daten durchaus zur Verwendung in unserer Hauptprogrammiersprache MatLab® geeignet erschienen, stellten sie uns doch vor relativ große Probleme: Nur ein kleiner Teil der Wegpunkte wird verwendet, Straßen und Wege darzustellen, der Rest dient vor allem der Definition von Flüssen und Verwaltungsgrenzen. Um diesem Problem zu begegnen, wurde ein Programm in Microsoft® C# aufgesetzt, das die Daten verarbeitet und im CSV-Datentyp exportiert.

Der Aufbau dieses Converters besteht aus mehreren Teilen:

1. Vereinfachung der ID,
2. Filtern der Wegdefinitionen nach befahrbaren Straßen,
3. Erkennung der verfügbaren Bewegungsvektoren,
4. Ausgabe von Bewegungsvektoren (Strecken),

5. Ausgabe der benötigten Koordinatenpunkte,

6. Ausgabe der Kreuzungspunkte.

Der erste Schritt wurde notwending, da MatLab® Daten auf Basis von Indexdaten verarbeiten kann. Dies macht es möglich, statt mit Zahlen in der Größenordnung von Milliarden mit IDs von 1 bis zur Anzahl der Wegpunkte zu arbeiten. Im nächsten Teil werden alle Straßen ausgelassen, die nur für Fußgänger, Radfahrer und Eigentümer (Privatwege) benutzbar sind. Außerdem werden auf dieser Basis gemäß Gesetz und Fahrpraxis Annahmen für Geschwindigkeit getroffen (siehe Tabelle 3).

Straße	km/h
Autobahn	130
Bundesstraße	100
Landstraße	70
Ortsgebiet	50
Kleine Ortsstraßen	30

Tabelle 3: Annahmen über die Geschwindigkeit

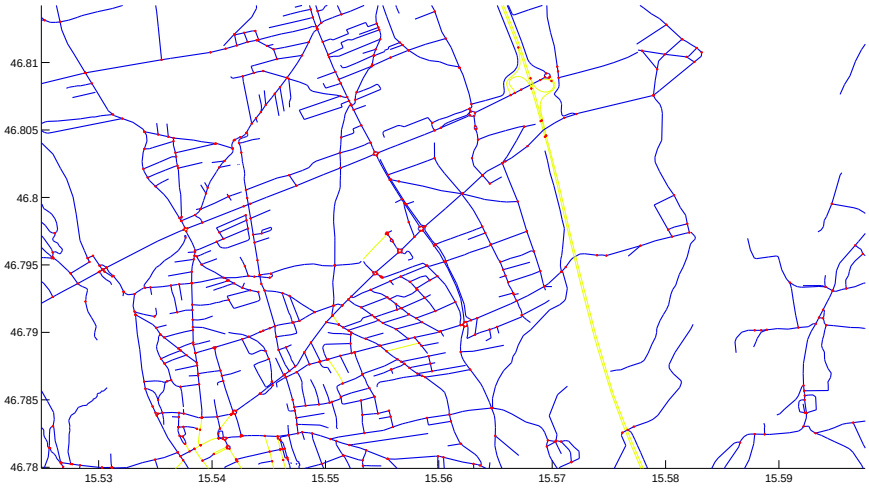


Abbildung 15: Straßenkarte von Leibnitz

Die Bewegungsvektoren geben an, welche Punkte untereinander als Start und Ziel fungieren (zwischen denen eine Verbindung existiert). Die Bestimmung erfolgt mit einem simplen Algorithmus: Alle Wegpunkte aller Wege werden mit einer foreach-Schleife

durchlaufen und mit dem jeweils nächsten verbunden; unter Zuhilfenahme der Geschwindigkeit wird als dritter Wert die benötigte Zeit berechnet. Im Anschluss erfolgt die Ausgabe als CSV(Comma Seperated Values)-Datei mit dem Format

```
ID 1,ID 2,Zeit
```

In der zweiten Datei finden sich Daten zu Wegpunkten nach dem Format

```
ID,Längengrad,Breitengrad
```

Der Abstand (in Kilometer) zwischen zwei Wegpunkten $P_1 = (l_1, b_1)$ und $P_2 = (l_2, b_2)$ (Längen- und Breitengrad) Kann durch die folgende Formel angenähert werden:

$$\overline{P_1P_2} = \sqrt{[x(l_2 - l_1)]^2 + [y(b_2 - b_1)]^2} \quad \text{mit}$$
$$x = 77,24249606 \quad y = 111,1949403$$

Sie ist anwendbar unter den Annahmen, dass man sich im Bereich von ca. 46 Grad Breite bewegt und die Erde als Kugel dargestellt wird.

Die dritte Datei stellt nun zum Zweck der Zeichnung eine Liste der Punkte dar, an denen mehrere Wege zusammentreffen; die Berechnung erfolgt über einen Vergleich aller Wegpunkte und Wegen untereinander.

5 Konklusion

Im Laufe dieser Modellierungswoche haben wir einige interessante Lösungsansätze und auch eine durchaus effektive Lösung für drei von den vier uns gegebenen Problemstellungen gefunden. Diese drei gelösten Aufgaben waren jedoch bereits so umfangreich, dass wir nicht mehr zur Bearbeitung der letzten Aufgabe, dem Klaviertransport, kamen. Die allgemeine Arbeitshaltung der Gruppe und das Klima beim Ausarbeiten der Probleme waren sehr positiv. Auch war die Motivation zu (beinahe) jeder Stunde gegeben und wir arbeiteten auch abends freiwillig an unserem Modell weiter. Einige lernten besser mit MatLab umzugehen, zu programmieren und wir bewiesen auch manche unserer relevanten Algorithmen mathematisch. Ohne Ausnahme jeder von uns wird von dieser Woche positive Erinnerungen mitnehmen.

Danksagung

Unser besonderer Dank gilt Dr. Kristian Bredies, der uns während der gesamten Woche unterstützt, geholfen und ertragen hat. Ebenfalls danken möchten wir dem gesamten Team der KF-Universität Graz, das eine solche Woche erst ermöglicht hat und dem Team des JUFA - Gästehauses.

Projekt Dynamische Systeme

Gibt es Zeit-diskrete Pumpen?

TeilnehmerInnen:

Anna Eibel
Tomas Kamencek
Nadja Kravanja
Armin Shan
Philipp Schröttner
Christopher Tscherne

Dr. Georg Propst

Inhaltsverzeichnis

1. Einleitung

In diesem Projekt geht es um die Erforschung bzw. Auffindung von Zeit-diskreten Pumpen. „Pumpen“ nennt man Systeme, die bei periodischer Erregung in ein Nichtgleichgewicht konvergieren. Unser Ziel war es, herauszufinden, ob es auch Zeit-diskrete Modelle gibt, für die dies zutrifft. „Zeit-diskret“ bedeutet, dass die Werte einer Funktion nicht für alle Argumente einzeln, sondern von Zeitpunkt zu Zeitpunkt berechnet werden.

Wir beschäftigten uns also mit zeit-diskreten Modellen der Form $x_{t+1} = F(x_t) + s_t$, wobei es sich bei s_t um eine periodische Störung handelt. Eine periodische Störung lässt sich z.B. durch eine Sinusfunktion oder einen Modulobefehl konstruieren.
Pumpen sind diese Modelle genau dann, wenn der Mittelwert der Argumente ungleich der Summe des Funktionswertes dieses Mittelwerts und dem Mittellwert der Störung ist, das heißt:

$$\bar{x} \neq F(\bar{x}) + \bar{s}$$

Wir erzeugten Modelle für verschiedene $F(\bar{x})$ und behandelten auch für die Störungen s_t verschiedene Fälle, wobei wir zunächst stets versuchten, auch Pumpen ohne Störung zu finden.

Für numerische Berechnungen verwendeten wir das Programm Octave und konnten so auch unsere Modelle grafisch darstellen. Weiters eignete sich Octave gut, um zu testen, ob die am Papier erhaltenen Ergebnisse richtig sind.

Zusätzlich verwendeten wir GeoGebra, um die untersuchten Funktionen grafisch darzustellen und z.B. Fixpunkte sofort zu erkennen.

2. Grundvoraussetzungen für Pumpen

Bei einem System kann es sich nur um eine Pumpe handeln, wenn bestimmte Bedingungen erfüllt sind:

1. $\bar{x} \neq F(\bar{x}) + \bar{s}$

2. Es muss periodisch sein. In einem bestimmten Intervall müssen also die gleichen Funktionswerte immer wieder auftreten. Die Periode T muss zwischen 1 und ∞ liegen. Ein System mit der Periode T=1 kann keine Pumpe sein, da für diese x immer gleich F(x) + s ist. Somit ist auch $\bar{x}$ immer gleich $F(\bar{x}) + \bar{s}$.
   Die Periode T darf den Wert ∞ nicht annehmen, da sonst der Mittelwert $\bar{x}$ nicht mehr berechnet werden kann.

3. Affin-lineare Systeme

Wir begannen unsere Suche nach zeit-diskreten Pumpen mit der Untersuchung von affin-linearen Systemen. Wir wählten diese als erstes, da von ihnen die einfachsten Funktionsterme zu erwarten waren.
Ein affin-lineares System hat die Form: $F(x) = G(x) + d$ mit linearem G. „Linear“ bedeutet, dass es sich bei der Funktion um eine durch den Ursprung verlaufende Gerade handelt.

Ein lineares System ist also eine Sonderform der affin-linearen Systeme mit d=0.

Aufgrund der Tatsache, dass eine Pumpe (1+n)-periodische Lösungen (für $n \in N^*$) besitzen muss, nahmen wir an, dass diese affin-lineare Systeme keine Pumpen sein können, weil sie keine mehrperiodischen Lösungen zu liefern schienen.
Um dies zu beweisen, behaupteten wir das Gegenteil, nämlich dass es affin-lineare Pumpen gibt, und suchten einen Widerspruch:

Behauptung: Es gibt affin-lineare Pumpen.
Beweis durch Annahme $\bar{x} \neq F(\bar{x}) + \bar{s}$:

$$\bar{x} = \frac{1}{T}\sum_{t=0}^{T-1} x_{t+1}$$

$$\bar{s} = \frac{1}{T}\sum_{t=0}^{T-1} s_t$$

$$F(\bar{x}) = G\left(\frac{1}{T}\sum_{t=0}^{T-1} x_t\right) + \frac{1}{T}\sum_{t=0}^{T-1} d$$

Jedoch folgt daraus: $\bar{x} = F(\bar{x}) + \bar{s}$

Affin lineare Systeme sind also keine Pumpen. Wir kamen zu dem Schluss, dass nur nicht-lineare Systeme Pumpen liefern können und untersuchten als nächstes quadratische Modelle.

4. Quadratische Modelle

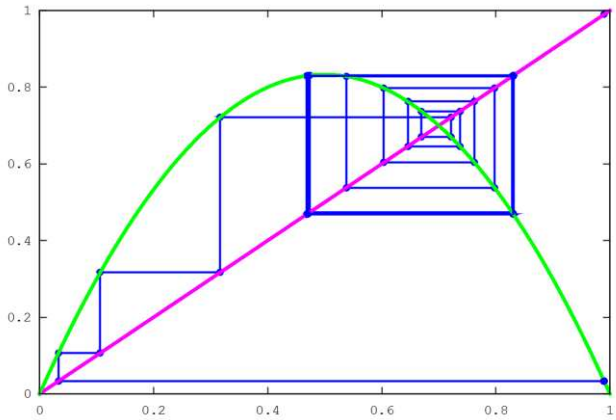
Wir betrachteten Parabeln im Intervall $[0,1]$, wobei nur die 4. Hauptlage sinnvolle Ergebnisse lieferte:
Versuch mit dem Modell $x_{t+1} = px_t(1 - x_t) + s$ für $x \in R|\{0 \leq x \leq 1\}$, p ist beliebiger positiver Parameter

4.1 s = 0

Zunächst versuchten wir der Einfachheit halber die Störung wegzulassen: $x_{t+1} = px_t(1 - x_t)$
Wir führten eine Kurvendiskussion durch:
Durch Nullsetzen der ersten Ableitung ergibt sich, dass bei x= $\frac{1}{2}$ ein Hochpunkt liegt. Weiters verläuft die Funktion immer durch den Ursprung und den Punkt (1|0).
Besitzt das Modell einen weiteren Fixpunkt außer dem Ursprung, so liegt dieser bei $x = 1 - \frac{1}{p}$:

$$px_t(1 - x_t) = x_t \rightarrow x_t = 1 - \frac{1}{p}$$

Für alle p > 2 besitzt der Schnittpunkt (S_1) der Funktion mit der ersten Mediane einen höheren x-Wert als der Hochpunkt, was Iterationen zu einer Ruhelage für $2 < x < 3$ ermöglicht.

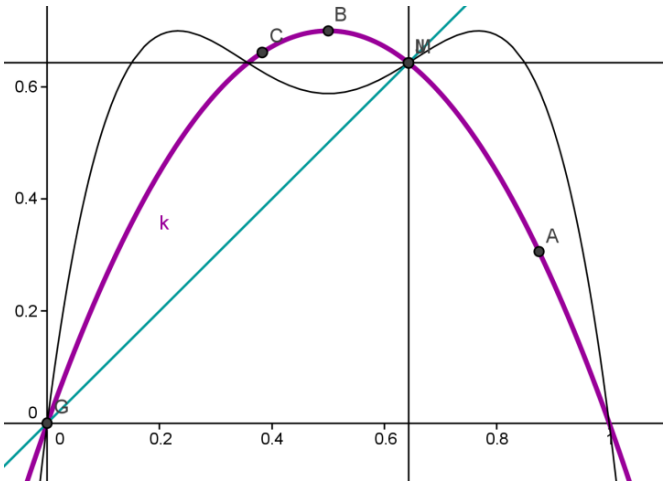


Iterationen ergeben sich durch wiederholtes Einsetzen des Funktionswertes in die Funktion.

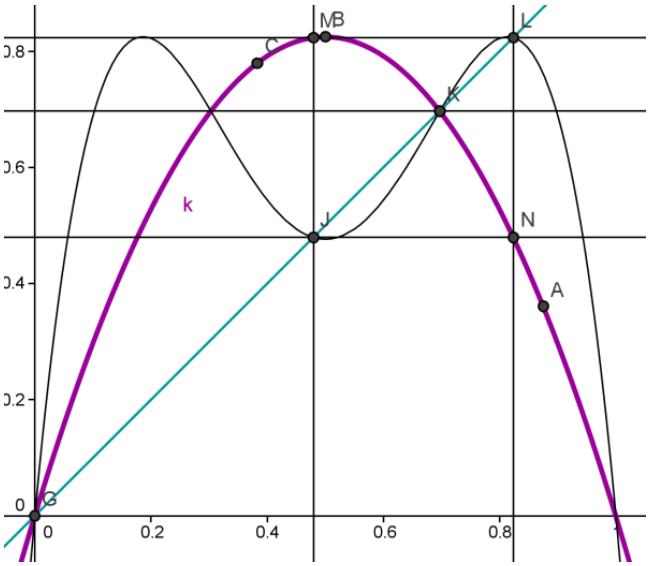
In der Abbildung beginnt die Folge in x_0 rechts unten und konvergiert gegen eine (2)-periodische Lösung (Rechteck)

Vergleich einer einfachen, einer zweifachen Periode und eines nicht periodischen Systems mit GeoGebra:

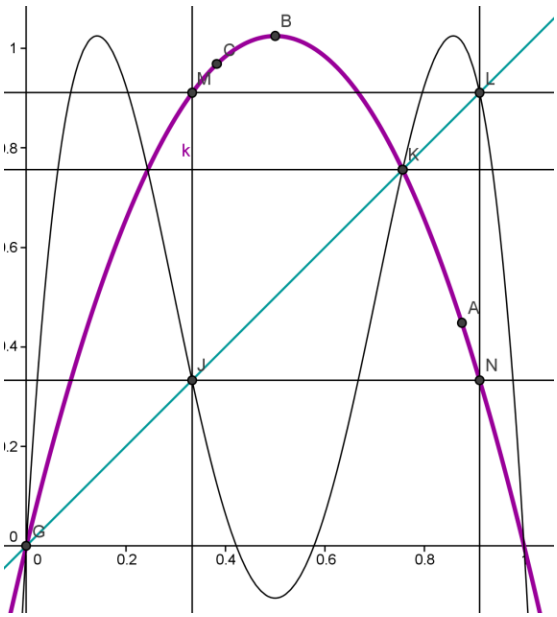
1er Periode für p = 2.8



2er Periode für p = 3.3



Chaos für p = 4.1



Man kann erkennen, dass in der ersten Abbildung die Funktion $F\big(F(x_t)\big)$ nur einen Schnittpunkt zwischen 0 und 1 mit der ersten Mediane hat, während in der zweiten und dritten Abbildung drei Schnittpunkte vorliegen. In der ersten Abbildung ist die Lösung daher nur (1)-periodisch, in den Abbildungen zwei und drei sind 2er-Perioden möglich. Da jedoch die dritte Funktion aus dem Intervall $[0,1]$ austritt, fällt die Iteration ins negativ Unendliche ab.

In weiterer Folge wollten wir beweisen, dass es (2)-periodische Pumpen für dieses System im Intervall $\left]0,\frac{p-1}{p}\right[$ gibt:

Das heißt, dass gelten muss: $x_{t+2} = F\big(F(x)\big) = x_t$

$F\big(F(x)\big) = p^2x(1-x)\big(1-px(1-x)\big)$

Durch Gleichsetzen mit x ergibt sich folgende Gleichung (x=0 ist nicht interessant):

$$px^3 - 2px^2 + px + x - 1 + \frac{1}{p^2} = 0$$

Um zu zeigen, dass diese Gleichung im Intervall $\left]0,\frac{p-1}{p}\right[$ eine Lösung besitzt, verwendeten wir den Zwischenwertsatz:

Wenn von einer stetigen Funktion f(x) ein positiver und ein negativer Funktionswert bekannt sind, muss die Funktion zwischen diesen mindestens eine Nullstelle besitzen.

An unser Beispiel angepasst, wählten wir zunächst für x den Wert 0:

$-1 + \frac{1}{p^2} < 0$ für p > 1

Für x = $\frac{p-1}{p}$ ergibt sich, dass f(x) > 0 ist für p > 2.

Folgerung: $\exists\, x \in [0,1]$: f(x) = 0, was bedeutet, dass es eine 2-periodische Lösung gibt, die grundsätzlich erst ab p > 3 auftreten können, wie wir mit GeoGebra zeigen konnten.

Nun bewiesen wir, dass es sich auch um eine Pumpe handelt, indem wir wiederum annahmen, dass gilt: $\bar{x} = F(\bar{x})$ und einen Widerspruch suchten:

$$\bar{x} = \frac{px_1 - px_1^2 + x_1}{2}$$

$$F(\bar{x}) = p\left(\frac{px_1 - px_1^2 + x_1}{2}\right)\left(1 - \frac{px_1 - px_1^2 + x_1}{2}\right)$$

Wir setzten $\bar{x}$ mit $F(\bar{x})$ gleich und erhielten dadurch folgende quadratische Gleichung:

$$px_1^2 - (p+1)x_1 + 2 - \frac{2}{p} = 0$$

Mithilfe der Lösungsformel $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ erhielten wir 2 Lösungen:

$x_{1_1} = \frac{p-1}{p}$

$x_{1_2} = \frac{2}{p}$

Die Lösung x_{1_1} ist nicht in $\left]0, \frac{p-1}{p}\right[$ enthalten und fällt somit als Lösung aus. Aus x_{1_2} lässt sich der Funktionswert $F(x_{1_2}) = x_{2_2}$ berechnen, jedoch ergibt sich aus $F(x_{2_2})$ nicht wieder x_{1_2}, was ein Widerspruch zur ursprünglichen Annahme $\bar{x} = F(\bar{x})$ ist. Das bedeutet, dass es sich um Pumpen handelt.

Konkretes Beispiel für $p = {}^{10}/_{3}$:

$x_{1_1} = {}^{7}/_{10}$ = Fixpunkt

$x_{1_2} = {}^{3}/_{5}$ $x_{2_2} = {}^{4}/_{5}$ $F(x_{2_2}) = {}^{8}/_{15} \neq {}^{3}/_{5}$

Folglich ist das System für $p = {}^{10}/_{3}$ eine Pumpe.

4.2 $s \neq 0$

Wir wählten eine konstante Störung s_0:

Auch dieses Modell ist bei geeigneter Einstellung von p und s eine Pumpe, wobei p-Werte, die ohne Störung keine Pumpen waren, durch Störung zu einer Pumpe werden können und umgekehrt. Dies lässt sich gut durch numerische Berechnungen zeigen, die wir mit Hilfe des Programmes Octave, durchführten.

4.3 Programm

Das oben erwähnte Octave-Programm ist folgendermaßen aufgebaut und dient zur Darstellung bzw. Berechnung des Modells mit oder ohne Störung:

x=(0.8236)	x ist der Startwert der Iteration
T=4	T ist die Periode
l=50	l gibt die Anzahl der Iterationen an
s=0.1	s ist die Störung
p=3.3	p ist der Parameter
v=zeros(3,l)	v speichert die erzeugten x-Werte
v(1,1)= p*x -p*x*x	dieser Befehl speichert den Startwert in v
z=1	z ist eine Zählvariable für v und eine while-Schleife

while z<(l+1)	startet die Schleife mit l Wiederholungen
v(1,z+1)= (-p)*v(1,z)*v(1,z)+p*v(1,z)+s	Berechnung und Speichern der restlichen x-Werte
z=z+1	dieser Befehl verhindert eine Endlosschleife
end	Beendet die Schleife
xquer=mean(v(1,(l-T+1):l))	mean berechnet den Mittelwert der Argumente
fxquer= (-p)* xquer * xquer +p* xquer	Berechnung des Funktionswertes des Mittelwertes
fxend=fxquer+s	Addieren der Durchschnittlichen Störung
z=1	Zurücksetzen der Zählvariable
while z<(l+1)	Beginn einer neuen Schleife
v(2,z)= xquer	dient zum Darstellen von $\bar{x}$
v(3,z)=fxend	dient zum Darstellen von $F(\bar{x}) + \bar{s}$
z=z+1	dieser Befehl verhindert eine Endlosschleife
end	beendet die Schleife
q=[1:1:l+1]	gibt den Intervall für die Darstellung an
plot(q,v)	Stellt den Vektor v im Intervall q dar

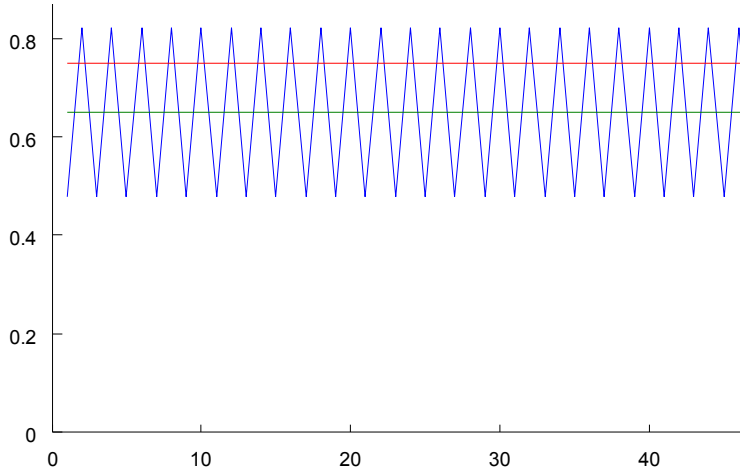
Stabile 2er-Periode mit:

p = 3,3

s = 0

Grüne (untere) Linie: $\bar{x}$

Rote (obere) Linie: F($\bar{x}$)



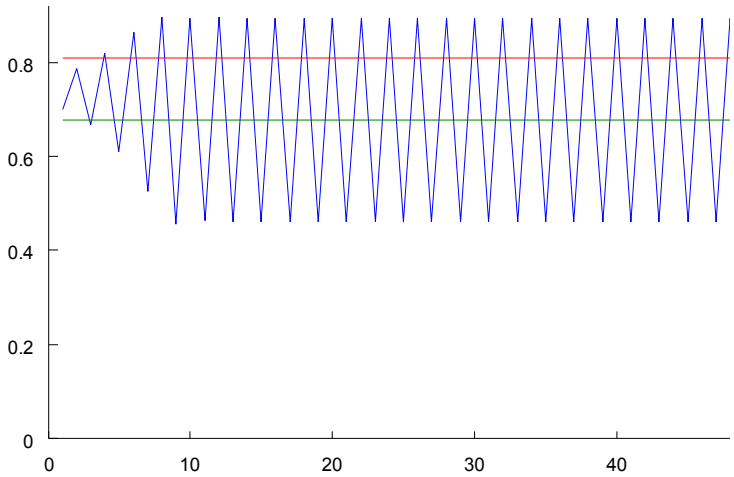
Stabile 2er-Periode mit:

p = 2,8

s = 0,2

Grüne (untere) Linie: $\bar{x}$

Rote (obere) Linie: F($\bar{x}$)



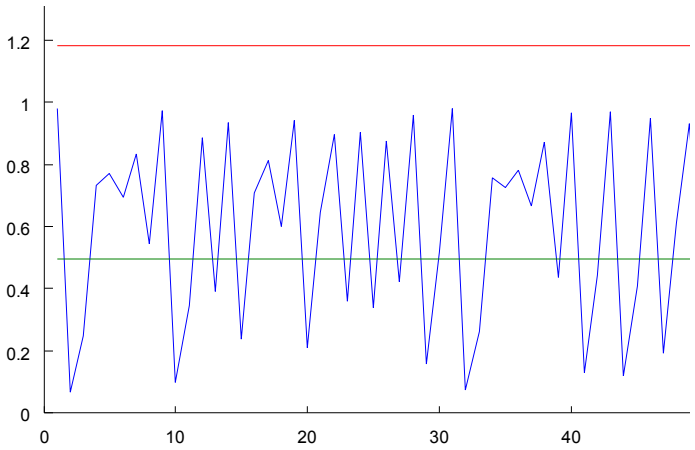
Ohne die Störung würde sich dieses Modell in eine Ruhelage einpendeln, da p < 3.

Chaotisches System mit :

p = 3,93

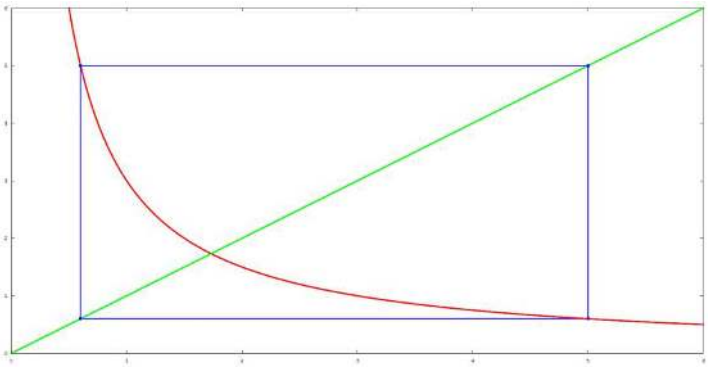
s = 0

Hier kann man gut erkennen, dass die Mittelwertberechnung nicht mehr funktioniert.



5. Hyperbolische Modelle

Wir versuchten analog zu den quadratischen Funktionen auch für Modelle, die durch Hyperbeln beschrieben werden, Pumpen zu finden, und zwar für s_t=0.



Die Abbildung zeigt eine (2)-periodische Folge x_t , welche rechts unten bzw. links oben beginnt.

Als Modell wählten wir die Funktion der Form $F(x_t) = \frac{p}{x_t}$ $mit\ p > 0\ ,\ x > 0$

Für 2-periodische Lösungen muss gelten:

$x_{t+2} = x_t$

$x_{t+2} = F\big(F(x_t)\big) = \frac{px_t}{p} = x_t$

Folglich gibt es 2-periodische Lösungen.

Um den Fixpunkt der Funktion zu berechnen, setzten wir $\frac{p}{x_t} = x_t$

Daher ergibt sich für den Fixpunkt: $x_t = \sqrt{p}$

Parallel zu den quadratischen Modellen bewiesen wir auch für dieses Modell, dass es Pumpen gibt.

$\bar{x} = {}^1\!/_2\left(\frac{p}{x_t} + x_t\right)$

$F(\bar{x}) = \frac{2px_t}{p + x_t^2}$

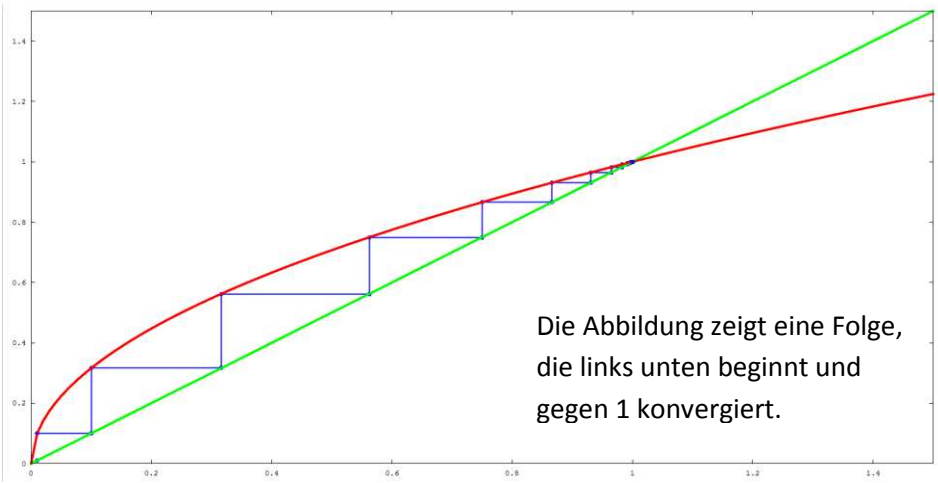
Wir setzten $\bar{x}$ mit $F(\bar{x})$ gleich und erhielten: ${x_t}^4 - 2p{x_t}^2 + p^2$

Wir substituierten $x_t{}^4$ mit u^2 und erhielten so für $u_{1,2} = p$ und somit für $x_1 = \sqrt{p}$, die negative Lösung -$\sqrt{p}$ fällt weg, da wir nur positive x-Werte betrachten.

Die Lösung ist genau der Schnittpunkt mit der ersten Mediane (=Fixpunkt), was bedeutet, dass nur für diesen Wert zutrifft, dass $\bar{x} = F(\bar{x})$.
Für alle anderen $x_t \neq \sqrt{p}$ gilt: $\bar{x} \neq F(\bar{x})$. Also haben wir bewiesen, dass dieses Modell für alle $x_0 \neq \sqrt{p}$ eine Pumpe ist.

6. Quadratwurzel-Modell

Wir untersuchten die Funktion $x_t = \sqrt{x_t} + s_t$
Für $s = 0$ kann das System keine Pumpe sein, da es keine periodischen Lösungen außer x_t=0,1 gibt. Diese sind jedoch die Fixpunkte der Funktion, die wir durch Gleichsetzen von $x_t = \sqrt{x_t}$ mit x_t errechneten, wodurch es sich um einperiodische Lösungen handelt, die, wie oben bewiesen, nie Pumpen sein können.



Die Abbildung zeigt eine Folge, die links unten beginnt und gegen 1 konvergiert.

Wir interessierten uns in weiterer Folge dafür, ob wir eine Störung so konstruieren können, dass wir eine 2-periodische Lösung erhalten.
Als Startwert wählten wir ein beliebiges x_0 und s_0 setzten wir 0.
Daraus ergibt sich für x_1 und x_2:
$x_1 = \sqrt{x_0}$, $x_2 = \sqrt[4]{x_0} + s_1$ wobei $x_0 \neq 1$ ist.

Wiederum setzten wir x_0 mit x_2 gleich und drückten $s_1$explizit aus: $s_1 = x_0 - \sqrt[4]{x_0}$

Weiters berechneten wir $\bar{x}$, $F(\bar{x})$ und $\bar{s}$ und konnten dann zeigen und nahmen an, dass gilt:
$\bar{x} = F(\bar{x}) + \bar{s}$:

$$\bar{x} = {}^1/_2\left(x_0 + \sqrt{x_0}\right) \qquad F(\bar{x}) = \sqrt{{}^1/_2\left(x_0 + \sqrt{x_0}\right)} \qquad \bar{s} = \frac{x_0 - \sqrt[4]{x_0}}{2}$$

Das Gleichsetzen führt zu folgender Gleichung:

$x_0{}^2$ ($(x_0 - 1)^2$ = 0

Die Lösungen lauten demnach: x_{0_1} = 0 und x_{0_2} = 1

Dies sind genau die x-Werte der Fixpunkte, weshalb für alle x außer den Fixpunkten gilt, dass $\bar{x}$ ungleich $F(\bar{x})$ + $\bar{s}$ ist, d.h. es handelt sich um Pumpen.

Googles PageRank-Algorithmus

Projekt Informationstechnik

Elke Schlager
Philipp Gabler
Michael Reichelt
Tristan Weniger
Lukas Grabenwarter
Konstantin Pollanz

Projektleiter: Dr. Christian Clason

Inhaltsverzeichnis

1 Einleitung

Wir alle kennen Google, und wir alle benutzen es, wahrscheinlich mehrmals täglich. Aber wie oft haben Sie sich schon gefragt, warum ausgerechnet Google? Es gibt ja auch andere Suchmaschinen, die man verwenden könnte. Warum nimmt selbst Google diese Konkurrenz nicht so ernst, wie sie sein könnte? Die Antwort ist leicht: Google ist einfach besser. Um Längen. Aber wieso? Einer der Gründe, warum Google so beliebt ist, ist die Tatsache, dass man oft sofort findet, was man gesucht hat. Dies geschieht mit Hilfe eines Programms, das die Seiten mit dem gewünschten Suchbegriff nach Relevanz ordnet. Es funktioniert ziemlich gut: Die meisten Suchen enden bereits auf der ersten Seite der Google-Ergebnisse, da diese genau die richtige Information beinhalten. Unsere Aufgabe diese Woche war es, ebenfalls so einen Algorithmus zu kreieren: Den sogenannten „PageRank-Algorithmus“. Zur Implementierung unserer Algorithmen haben wir uns für die Software „Octave“ entschieden, da diese eine effiziente Verarbeitung von Matrizen ermöglicht, mit deren Hilfe sich dieses Problem sehr gut modellieren lässt. Wir haben uns entschlossen mehrere Faktoren in unsere Seitenbewertung einzubeziehen, nämlich Linkstruktur, Inhalt, Klick-Statistik und Domain-Wichtigkeit.

2 Linkstruktur

2.1 Grundidee

Anfangs versuchten wir, über die Verlinkungen einer Seite ihren „Wert“ herauszufinden: Je öfter eine Seite verlinkt wird, desto wertvoller erscheint sie in unserem System. Natürlich können wir hier nicht alle Links zählen, sonst wäre das Ergebnis unbrauchbar: Mehrere gleiche Links von einer Seite zu einer anderen wären zum Beispiel nichts weiter als Spam und sollten dementsprechend auch nicht bewertet werden. Eine Seite, die sich selbst verlinkt, kann man auch nicht als besser betrachten als eine, die dies nicht tut. Nachdem wir das in Betracht gezogen hatten, kamen wir zu unseren Einschränkungen, wie zum Beispiel bei mehreren gleichen Links nur einen davon zu zählen und Verlinkungen auf die eigene Seite gar nicht. Da wir mathematische Software für die Modellierung verwendet haben, haben wir Linkstrukturen als Matrizen interpretiert. Wenn die Seite i auf die Seite j verweist, dann soll in der Matrix

$$\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n \times n} \tag{1}$$

der n Seiten das Element a_{ij} eins sein. Die übrigen Werte in der Matrix betragen null. Dadurch kann in der verwendeten Software mit einer sogenannten

Sparse-Matrix gearbeitet werden, die nur jene Werte speichert, die ungleich null sind und damit eine effizientere Speicherung und Verarbeitung ermöglicht. ¡Bild¿

2.2 Entwicklung

Unser erster Algorithmus arbeitete wie folgt: Man definiert einen Startwert, mit dem jede Seite in den Algorithmus einsteigt; in unserem Fall betrug dieser Wert eins. Jede Seite bekommt nun pro Link, der auf sie zeigt, den Wert der verlinkenden Seite addiert. Nach mehreren Wiederholungen dieses Schrittes bekommen wir eine Rangliste, welche sich nicht mehr verändert: Unseren PageRank, zumindest unseren ersten.

Nachdem wir den Algorithmus mit gewöhnlichen Programmstrukturen implementiert hatten, fanden wir einen Weg, ihn als Matrizenmultiplikation zu formulieren, wodurch er effizienter und übersichtlicher wurde. Dieser Algorithmus kann nun als folgende Gleichung geschrieben werden, wobei k die Anzahl der Iterationen ist:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{A}\mathbf{x}^k, \tag{2}$$

wobei

$$\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n \times n} \tag{3}$$

die Linkmatrix von n Seiten und

$$\mathbf{x}^1 = (1, \ldots, 1) \in \mathbb{R}^n \tag{4}$$

der Vektor der Startwerte ist.

Der Nachteil dieses Verfahrens ist aber, dass die einzelnen Werte mit k gegen Unendlich gehen, weshalb wir den resultierenden Vektor normieren, indem wir durch seinen Maximalwert dividieren:

$$\mathbf{x}^{k+1} = \frac{\mathbf{x}^k + \mathbf{A}\mathbf{x}^k}{\max\limits_{1 \le i \le n} (\mathbf{x}^k + \mathbf{A}\mathbf{x}^k)_i} \tag{5}$$

Dadurch erhalten wir relative Werte, die sich alle zwischen null und eins befinden und so gut vergleichbar sind.

Um zu verhindern, dass Seiten, die selbst nicht verlinkt sind, den Wert null an andere Seiten weitergeben, haben wir einen Parameter α eingeführt, der nach jedem Durchlauf zu x addiert wird:

$$\mathbf{x}^{k+1} = \frac{\mathbf{x}^k + \mathbf{A}\mathbf{x}^k}{\max\limits_{1 \le i \le n} (\mathbf{x}^k + \mathbf{A}\mathbf{x}^k)_i} + \alpha \tag{6}$$

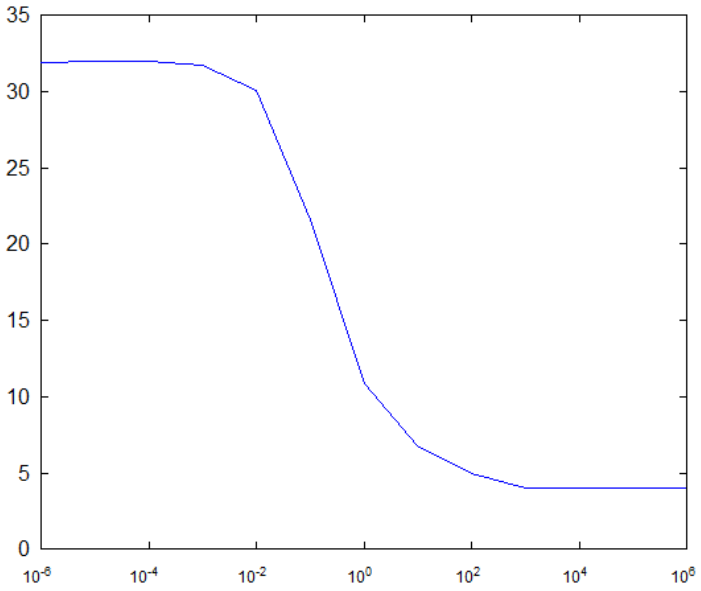


Abbildung 1: Anzahl der Iterationen als Funktion des Parameters α (halblogarithmisch)

Dieser Parameter α beeinflusste sowohl die Laufzeit als auch die Genauigkeit der Werte, und zwar so, dass bei größerem α die Anzahl der Iterationen, aber leider auch die Aussagekraft der Resultate sank (siehe Abb. 1).

Ein weiteres Problem dieses Modells taucht auf wenn Subnetzwerke existieren, die überhaupt nicht miteinander verbunden sind, da sich die Verhältnisse der Wertigkeiten dieser Netze nicht eindeutig bestimmen lassen. Daher haben wir uns in unserem endgültigen Modell für einen anderen Algorithmus entschieden.

2.3 Der Algorithmus

Der Algorithmus, den wir schließlich gewählt haben, löst das Problem mit den Subnetzwerken durch das Hinzufügen einer imaginären Seite, die auf alle anderen Seiten verweist und von allen anderen Seiten verlinkt wird. Deren Wert, der aufgrund der Funktionsweise unseres Algorithmus sehr groß wird, wird nach jeder Iteration unseres Programms auf alle anderen Seiten aufgeteilt, und zwar gewichtet nach deren derzeitigem Wert.

Da jedes Element der Matrix, auf die man den Algorithmus anwendet, durch die jeweilige Zeilensumme dividiert wird, summieren die Zeilen sich immer

auf eins, und daher beträgt auch die Summe der Seitenwerte eins. Dadurch sind sie leicht vergleichbar, können nicht gegen Unendlich gehen und sind als Wahrscheinlichkeit deutbar. Das bedeutet, dass ein Surfer, der zufälligen Links folgt, mit eben dieser Wahrscheinlichkeit x_i auf eine gewisse Seite i gelangt.

Die Startwerte jeder Seite werden als $\frac{1}{n}$ festgelegt, damit sie sich schon zu Beginn auf eins summieren. In jeder Iteration wird x_i^{k+1} festgelegt als die Summe aller auf i verweisenden Seiten, geteilt durch die Anzahl der Links auf der Seite i.

Wir beginnen wieder mit unserer Linkmatrix

$$\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n \times n}. \tag{7}$$

Nun bilden wir für jede Zeile i von $\mathbf{A}$ die Zeilensumme s_i erhöhen sie um eins und bilden den Kehrwert, um hinterher eine Division zu vermeiden, die die Berechnung verlangsamen würde:

$$\mathbf{s} = (s_i) \quad \text{mit} \quad s_i = \left(\left(\sum_{j=1}^{n} a_{ij} \right) + 1 \right)^{-1} \quad \text{für} \quad 1 \leq i \leq n. \tag{8}$$

Jedes Element von $\mathbf{A}^T$ wird durch die entsprechende Zeilensumme geteilt. Die Transponierte von $\mathbf{A}$ wird verwendet, da Octave Matrizen in dieser Form schneller verarbeiten kann, und um die Weiterverarbeitung zu vereinfachen. Als neue Matrix erhalten wir

$$\tilde{\mathbf{A}} = (\tilde{a}_{ij}) = (a_{ji} s_j) \quad \text{für} \quad 1 \leq i, j \leq n. \tag{9}$$

Wie oben beschrieben betragen die Startwerte jeweils $\frac{1}{n}$:

$$\mathbf{x}^1 = \frac{1}{n} \mathbf{e} \quad \text{mit} \quad \mathbf{e} = (1, \ldots, 1)^T \in \mathbb{R}^n \tag{10}$$

Somit erhalten wir die Folge $\mathbf{x}^k$, die für $k \to \infty$ gegen die von uns gewünschten Werte konvergiert:

$$\mathbf{x}^{k+1} = \tilde{\mathbf{A}} \mathbf{x}^k + \frac{\mathbf{s}^T \mathbf{x}^k}{n} \mathbf{e} \tag{11}$$

Die Berechnung wird beendet, sobald die Summe der Beträge der Differenzen kleiner 10^{-5} ist

$$\sum_{i=1}^{n} |x_i^{k+1} - x_i^k| < 10^{-5}, \tag{12}$$

weil dann angenommen werden kann, dass die Werte annähernd konstant bleiben.

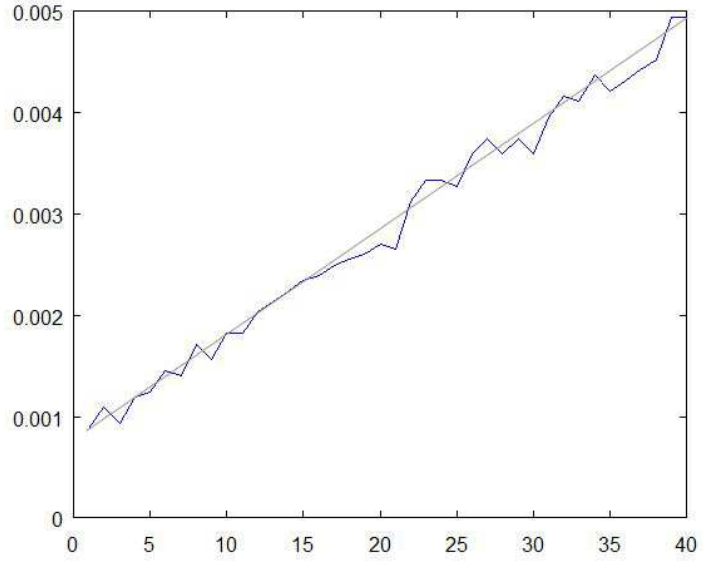


Abbildung 2: Laufzeit als Funktion der Seitenzahl (in Hunderterschritten)

2.4 Eigenschaften des Algorithmus

Die Laufzeit unseres Verfahrens ist, wie in Abb. 2 zu sehen, linear bezüglich der Seitenanzahl n, da die Iterationszahl unabhängig von der Seitenzahl und die in der Schleife ausgeführte Matrizenmultiplikation für eine Sparse-Matrix annähernd linear ist.

Eine weitere Besonderheit ergibt sich, wenn man unseren Algorithmus als Suche nach einem Eigenvektor sieht. Sei

$$\mathbf{A}^* = \tilde{\mathbf{A}} + \frac{1}{n} \begin{pmatrix} \mathbf{s}^T \\ \vdots \\ \mathbf{s}^T \end{pmatrix}, \tag{13}$$

dann kann man (11) auch in der Form

$$\mathbf{x}^{k+1} = \mathbf{A}^* \mathbf{x}^k \tag{14}$$

schreiben. Wenn jetzt $k \to \infty$ geht und $\mathbf{x}^k$ gegen $\tilde{\mathbf{x}}$ konvergiert, gilt

$$1\tilde{\mathbf{x}} = \mathbf{A}^* \tilde{\mathbf{x}}. \tag{15}$$

Damit lässt sich unser gesuchter Vektor der Wertigkeiten als Eigenwertgleichung der Matrix $\mathbf{A}^*$ mit bekanntem Eigenwert $\lambda_1 = 1$ und Eigenvektoren

$\mathbf{v}_1, \ldots, \mathbf{v}_n$ interpretieren, wobei λ_i zu $\mathbf{v}_i$ gehört. Sei nun $\mathbf{x}^k = (x_1^k, \ldots, x_n^k)^T$ ein Wertungsvektor der Größe n, und sei $\mathbf{x}^k$ eine Linearkombination $\alpha\mathbf{v}_1 + \beta\mathbf{v}_2$. Dann ist

$$\begin{aligned} \mathbf{x}^{k+1} &= \mathbf{A}^*\mathbf{x}^k \\ &= \mathbf{A}^*(\alpha\mathbf{v}_1 + \beta\mathbf{v}_2) \\ &= \alpha\mathbf{A}^*\mathbf{v}_1 + \beta\mathbf{A}^*\mathbf{v}_2 \\ &= \alpha\lambda_1\mathbf{v}_1 + \beta\lambda_2\mathbf{v}_2 \end{aligned} \tag{16}$$

und damit

$$\begin{aligned} \mathbf{x}^{k+2} &= \mathbf{A}^*\mathbf{x}^{k+1} \\ &= \alpha\lambda_1\mathbf{A}^*\mathbf{v}_1 + \beta\lambda_2\mathbf{A}^*\mathbf{v}_2 \\ &= \lambda_1^2\alpha\mathbf{v}_1 + \lambda_2^2\beta\mathbf{v}_2. \end{aligned} \tag{17}$$

Offensichtlich ergeben sich nach $k + i$ Wiederholungen i-te Potenzen der Eigenwerte. Wenn nun $\lambda_1 > \lambda_2$ und $\lambda_1 = 1$, strebt der zweite Summand durch die wiederholte Potenzierung gegen null, der erste dagegen gegen $\mathbf{v}_1$, also unser gesuchtes $\tilde{\mathbf{x}}$. Ist aber $\lambda_1 \approx \lambda_2$, werden durch die numerische Berechnung der Werte deutlich mehr Iterationen benötigt, um einen hinreichend genaues $\tilde{\mathbf{x}}$ zu erhalten, da die Konvergenzgeschwindigkeit viel geringer ist.

Die Differenz der Eigenwerte ist nun eine Eigenschaft, die nicht von der Größe, sondern nur von der Struktur der Linkmatrix, also den Verlinkungen der einzelnen Seiten, abhängt. Dadurch ist das eigentliche Hindernis einer effizienten Berechnung nicht die große Anzahl der Seiten, sondern die schwer zu berechnende Struktur der Verlinkungen.

Dieses Problem wird eben durch die imaginäre Seite gelöst, die das Problem der getrennten Subnetze behebt.

2.5 Vergleich mit Google

Google verwendet einen PageRank-Algorithmus, der unserem sehr ähnelt. Es gibt nur einen gravierenderen Unterschied. Während wir das Problem der Subnetze, also Netze die nicht untereinander verlinkt sind, durch die bereits beschriebene imaginäre Seite lösen, geht Google einen etwas anderen Weg. Google nimmt an, dass es von jeder Seite eine Chance gibt, einfach auf eine beliebige andere Seite mithilfe der Adressleiste zu wechseln, auch wenn diese auf der aktuellen Seite nicht verlinkt ist. So erzeugen sie, ohne Einführung einer Hilfsseite, ein vollständig verlinktes Seitennetzwerk. Wie groß die Wahrscheinlichkeit ist, dass der Surfer über diesen Weg die Seite wechselt, ist aber schwer einzuschätzen, und so macht sich auch Google nicht zu viele Gedanken darüber und wählt einen willkürlichen Wert: Eine Chance von 15%. Falls man nun diesen Wert auf 43% verändern würde, bekäme Google mit seinem Verfahren

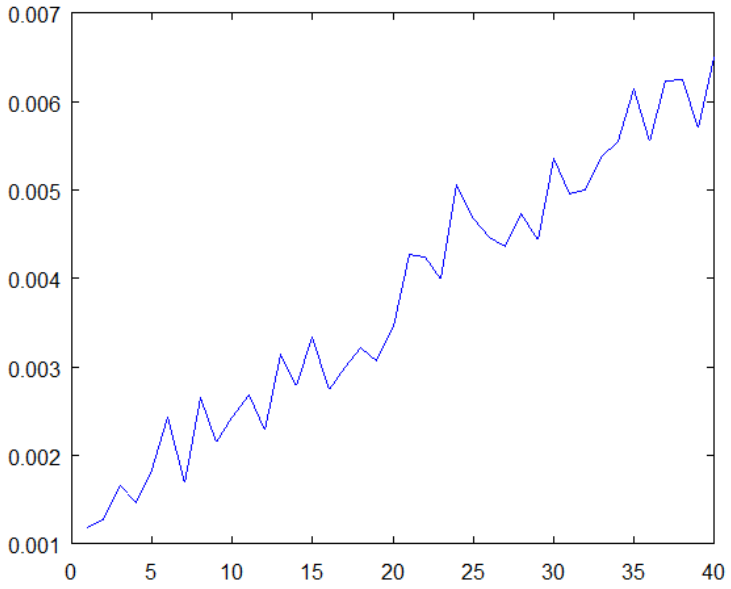


Abbildung 3: Laufzeit als Funktion der Seitenanzahl in Hunderterschritten – Googles Modell (siehe Abb. 2)

sehr ähnliche Wertigkeiten wie wir. Mit diesem höheren Wert enden wir mit dem Unterschied, dass unser Algorithmus kleinere Seiten stärker unterstützt. Außerdem ist unser Algorithmus schneller, da wir damit weniger Iterationen brauchen, doch bei genaueren Vergleichen hebt Google kleinere Rangunterschiede deutlicher hervor.

3 Inhalt

Unser Algorithmus berücksichtigt auch die Häufigkeit des Suchbegriffes im Text bzw. in den Überschriften und im Titel einer Seite. Die relative Häufigkeit im Text wird anhand einer Gaußschen Glockenkurve mit den Parametern $\mu = 0.05$ und $\sigma = 0.02$ bewertet (siehe Abb. 4):

$$\varphi_i(h_i) = \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \tag{18}$$

wobei h_i der relativen Häufigkeit des Suchwortes auf der Seite i und φ_i der Bewertung der jeweiligen Seite entspricht.

Die relative Häufigkeit in den Überschriften wird als solche beibehalten, für den Titel existieren nur die Werte 1 (kommt vor) und 0 (kommt nicht vor).

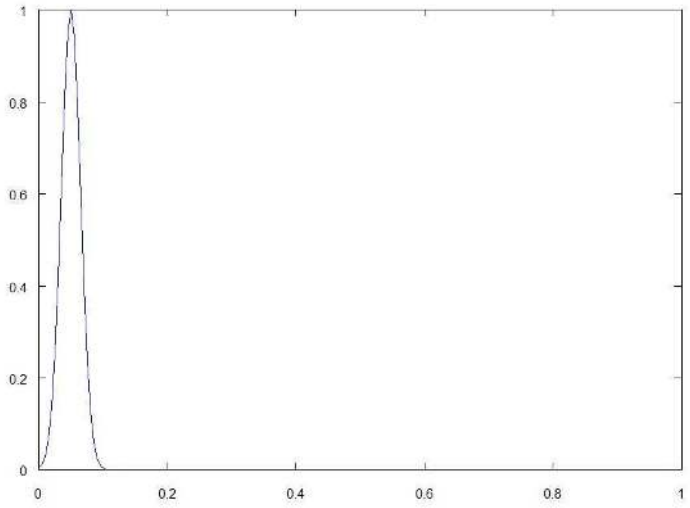


Abbildung 4: Glockenkurve, zur Gewichtung der relative Häufigkeit der Suchworte

Der Durchschnitt dieser drei Werte wird jeweils durch die Summe ebendieses Durchschnitts für alle Seiten dividiert, damit sich die Werte aller Seiten auf eins summieren:

$$\mathbf{w}_{C_i} = \frac{\varphi_i + \psi_i + \chi_i}{\sum_{j=1}^{n} (\varphi_j + \psi_j + \chi_j)} \tag{19}$$

wobei φ_i die Text-Wertung, ψ_i die Überschriften-Wertung, χ_i die Titel-Wertung für die Seite i und $\mathbf{w}_{C_i}$ der anhand des Inhalts erstellte Wertigkeitenvektor sei.

4 Klick-Statistik

Um die Klicks auf eine Website in unsere Bewertung mit relativem Wert einfließen zu lassen, dividieren wir die gesamten Klicks auf diese Website durch die Summe aller Klicks auf jede Seite. Hierbei werden nur die Seite berücksichtigt, die aufgrund der Suchabfrage ausgewählt wurden. Dazu benützen wir einen Spaltenvektor, der uns als Datenbank für die Klicks dient, und dividieren ele-

mentweise durch die Summe des Vektors:

$$\mathbf{w}_c = \mathbf{c} \left(\sum_{i=1}^{n} c_i \right)^{-1} \tag{20}$$

5 Domain-Wichtigkeit

In die Bewertung einer Seite soll in unserem Modell auch der Wert der Domain, auf der Seite liegt, miteinbezogen werden, weil eine renommierte Domain mit großer Wahrscheinlichkeit auch Seiten mit gutem Inhalt beherbergt. Der Wert einer Domain soll sich aus den Wertigkeiten ihrer einzelnen Seiten zusammensetzen.

Dies erledigt folgender Algorithmus: Wir gehen aus von einer Matrix $\mathbf{D}$ für die gilt $d_{ij} = 1$ wenn die Domain i die Seite j beherbergt und einem Vektor $\mathbf{w}_A$, der alle Seitenwerte enthält. Für jede Domain werden nun die Werte der zugehörigen Seiten addiert. Jeder einzelne Wert wird dann noch durch die Summe aller Domain-Wichtigkeiten geteilt, damit auch diese Werte sich auf eins summieren.

Durch eine Matrizen-Multiplikation und anschließende Division des Vektors durch die eigene Summe

$$\mathbf{w}_D = \frac{\mathbf{D}^T(\mathbf{D}\mathbf{w}_A)}{\sum_{i=1}^{n} \mathbf{D}^T(\mathbf{D}\mathbf{w}_A)_i} \tag{21}$$

erhält jedes Element $\mathbf{w}_{D_i}$, das der Domain i entspricht, einen aus den Seiten-Wichtigkeiten errechneten Wert. Alle Domain-Wertigkeiten summieren sich wiederum auf 1, wodurch die Vergleichbarkeit mit den aus den übrigen Algorithmen gewonnenen Werten gewährleistet ist.

6 Gesamtwertung

Die Gesamtwertung $\mathbf{w}_G$ entsteht aufgrund der einzelnen Wertungen als gewichtete Summe, wobei die Linkstruktur einen Anteil von 40% ausmacht, die Klick-Statistik 30% und die Inhaltswertung und die Domain-Wichtigkeit jeweils 15%. Daraus ergibt sich folgende Formel:

$$\mathbf{w}_G = 0.4\mathbf{w}_A + 0.15\mathbf{w}_B + 0.3\mathbf{w}_C + 0.15\mathbf{w}_D, \tag{22}$$

wobei $\mathbf{w}_A$ die Bewertung der Linkstruktur, $\mathbf{w}_B$ die Inhaltswertung, $\mathbf{w}_C$ die sich aus der Klick-Statistik ergebende Wertung und $\mathbf{w}_D$ die Domain-Wichtigkeit sind.

Das hat den Sinn, dass die Summe eins ergibt und als Wahrscheinlichkeit deutbar ist, und dass eine Einzelwertung immer weniger Einfluss hat als die Gesamtheit anderen, also keine absolute Mehrheit hat. Dadurch wird wiederum vermieden, dass ein Spammer durch Manipulation eines einzelnen Kriteriums den PageRank seiner Seite in zu großem Ausmaß verändern kann.

Projekt Politikwissenschaft

Entwicklung eines Wahlsystems

Dr. Stephen Keeling

Teilnehmer: Thessa Hinteregger, Simeon Kanya, Benedikt Maderbacher, Sarah Österreicher, Martin Schwarzl, Julian Greitler

PROBLEMSTELLUNG:

Es ist nicht unüblich, dass ein Bürger eine Kritik für seine Regierung hat. Impliziert bei dieser Kritik ist üblicherweise der Glaube, dass ein Kurswechsel existiert, der dem Gemeinwohl besser dienen würde. Kann es sein, dass diese Einstellung fundamental falsch ist? Was ist das Gemeinwohl überhaupt? Wie gewichtet man die vielen Präferenzen der Bürger um das Gemeinwohl zu definieren? Und wenn das Gemeinwohl sinnvoll gemacht werden könnte, wie sollen die Präferenzen gemessen werden? Würden die Bürger ihre Präferenzen ehrlich bekannt geben oder eher ein Wahlsystem manipulieren?
Nach dem Satz von Arrow (The Impossibility Theorem) ist es nicht möglich, die Präferenzen der Bürger in einer gemeinsamen Präferenzenliste zu kombinieren, während natürliche Gerechtigkeitsbedingungen erfüllt werden. Trotzdem müssen Entscheidungen von einer demokratischen Regierung getroffen werden, die alle Bürger beeinflussen und einigermaßen repräsentieren sollen. Oder soll die Regierung den Satz von Arrow bereitwillig annehmen? Sie könnte einen Konsens dynamisch entstehen lassen, während die Bürger ihre Präferenzen durch ihre Käufe auf einem freien Mark bekanntmachen. So ist die Politik in den USA von Spieltheoretikern gesteuert worden. Wenn ein Konsens dynamisch entstehen sollte, stellt sich die Frage ob die Dynamik tatsächlich zu einem stabilen Ergebnis führt und welche Beziehung dies zu den individuellen Präferenzen hätte.
Die Ziele des Projekts sind folgende: Für eine bevorstehende Gruppenentscheidung soll eine Zielfunktion definiert werden, mit der berechnet werden kann, wie erfolgreich ein Wahlergebnis gegenüber dem anderen ist. Dann sollen einige Wahlsysteme untersucht werden. Ein System kann aus einem Wahlgang oder aus mehreren wiederholten Wahlgängen bestehen. Es soll für ein System bestimmt werden, welche Bedingungen des Satzes von Arrow verletzt werden. Wenn das System aus mehreren wiederholten Wahlgängen besteht, soll bestimmt werden, ob das System tatsächlich zu einem stabilen Ergebnis führt. Man soll auch überlegen, wie die Gruppenmitglieder das System manipulieren können. Schließlich soll mit der eigenen Zielfunktion berechnet werden, welche der eigenen Ergebnisse erfolgreicher als andere sind.
WAS IST DAS GEMEINWOHL?

EINFÜHRUNG IN DEN SATZ VON ARROW

"Suppose we have a finite set V of voters who have to vote for a finite set of candidates C. Each voter ranks the candidates in some ordering, e.g. if C={X,Y,Z} then a voter v might rank Y > Z > X. We assume voter rationality- that every voter ranks candidates in a total ordering, and voter free will- each voter can rank candidates in any order (subject to voter rationality) and independently of all other voters.
A voting system is a function which takes as input the voting preferences of each voter in V, and returns as output a ranking of the candidates.
One would like the voting system to satisfy the following reasonable axioms.

- (Voting system rationality) The output of the voting system is a total ordering.

- (Determinism) The output of the voting system is determined only by the ranking preferences of the voters and on no other factors (in particular, no randomization is allowed).
- (Consensus) If all the voters prefer A to B, then the output of the voting system also prefers A to B.
- (Impartiality) All candidates are treated equally; in other words, the voting system is invariant under permutations of the candidates.
- (Independence of a third alternative) The relative ranking of X and Y in the output of a voting system is independent of the voters preferences for a third candidate Z.
- (No dictators) If a single voter prefers X to Y, but all other voters disagree, then the voting system should override the wishes of that single voter and rank Y higher than X.

Unfortunately, we have Arrow's theorem: If there are at least three candidates, then the above six axioms are inconsistent." [1]

ERGEBNISSE:

Zu Beginn stand die Frage, welche Möglichkeiten von Auswertungssystemen es gibt. Hierfür nahmen wir ein einfaches Beispiel, nämlich einen Schönheitswettbewerb zwischen drei Katzen. Jeder sollte die drei Katzen für sich mit einer Gesamtsumme von 10 bewerten, indem jeder Katze ein bestimmter Wert zwischen null und zehn zugewiesen wurde. Am Ende hatte jene Katze gewonnen, die insgesamt mehr Punkte hatte, ohne andere Faktoren zu berücksichtigen, die das Ergebnis in eine andere Richtung beeinflussen hätte können, zum Beispiel Streuung oder Unzufriedenheit. Später suchten wir Kritikpunkte dieser Methode und versuchten sie zu verbessern, indem wir oben genannte Faktoren einzubauten.

Gegen Abend bekamen wir eine kurze Einführung in Matlab um an den folgenden Tagen mögliche Wahlsysteme programmieren zu können.

Am Beispiel des Wählens eines Reiseziels stellten wir uns die Frage, welches Wahlsystem nun am besten sei. Dafür machten wir einen Selbstversuch, indem wir verschiedene Möglichkeiten ausprobierten und diskutierten. Die erste Möglichkeit war, die einzelnen Reiseziele mit Punkten von -5 bis 5 zu bewerten, wobei hier keine Gesamtsumme vorgegeben war. Probleme waren unter anderem, dass der Grad der Unzufriedenheit nicht berücksichtigt wurde und das System zudem im Laufe mehrerer Wahldurchgänge zunehmend manipulierbar wurde. In den weiteren Wahlgängen bewerteten wir die einzelnen Länder nur mehr mit plus und minus, wobei es je nach Wahlgang verschiedene Einschränkungen gab: Unbegrenzte Zahl an plus beziehungsweise minus, eine maximale Zahl an plus beziehungsweise minus, eine genaue Anzahl von plus beziehungsweise minus sowie verschiedene Kombinationen aus den eben genannten. Im Laufe dieser Wahldurchgänge stellte sich für uns heraus, dass die Zusammenführung von höchstens zwei negativen und unbegrenzt vielen positiven Werten (jede Möglichkeit muss bewertet werden!) am aussagekräftigsten sei. Um dies in größerem Rahmen zu testen, führten wir eine Wahl zum Thema Freizeitaktivitäten unter allen 30

[1] Quelle: Department of Mathematics, UCLA, Los Angeles, CA 90024

Teilnehmern der Modellierungswoche durch, wobei man seine negativen und positiven Wertungen noch zwischen null und zehn gewichten musste.

Im Laufe des Nachmittages werteten wir die Ergebnisse aus und versuchten aufgrund dieser Ergebnisse unser Wahlsystem zu beschreiben. Ziel war es, möglichst viele Leute zufrieden zu stellen beziehungsweise in weiterer Folge vor Unzufriedenheit zu bewahren, dies definierten wir zunächst als Gemeinwohl. Die Frage nach dem Gemeinwohl sollte uns noch länger beschäftigen. Um unser Ziel zu erreichen, brachten wir verschiedene Komponenten in eine Formel. Diese waren arithmetisches Mittel der einzelnen Wertungen bezüglich einer Wahlmöglichkeit, die Streuung (Varianz, Standardabweichung σ), Median, Modus, durchschnittliche Unzufriedenheit pro Person.
Unsere ursprüngliche Formel war folgende:

$$a = \frac{\sigma}{\bar{x}}$$

Zustande kam diese durch die Überlegung, dass die zwei wichtigsten Faktoren wohl die Standardabweichung und das arithmetische Mittel seien. Das arithmetische Mittel ist immer dann aussagekräftig, wenn ich von jeglicher Manipulation absehe. Auch die Standardabweichung ist wichtig, weil sie die Geeintheit der Wähler angibt. Ist sie klein, ist dies für die Gruppe gut. Je kleiner a ist, desto besser. Probleme sind, dass keiner der Werte null sein darf und dass diese Formel bei Entscheidungen zwischen zwei Wahlmöglichkeiten aufgrund der Standardabweichung nicht aussagekräftig ist. Dies ergibt sich dadurch, dass bei hohem arithmetischen Mittel und großer Standardabweichung im Vergleich zu niedrigem arithmetischen Mittel (vor allem im unteren Drittel der Werte) und kleiner Standardabweichung a annähernd gleich groß sind. Hier ist jedoch das erste Ergebnis besser, dies ist an dieser Formel aber nicht ersichtlich. Auch andere Systeme beziehungsweise Formeln kamen in Frage, diese verwarfen wir allerdings aufgrund verschiedenster Probleme wieder.

-2	3	0	2	0	1	1	4	1	-3	1	4	0	-2	1	0	-5	2	0	2	5	-3	-5	2	4	2	5	0	5	0
1	2	4	1	4	4	1	0	0	3	0	1	0	5	3	1	5	2	3	1	3	2	0	0	2	5	4	2	2	1
3	5	-5	-5	-5	3	1	-3	4	0	-5	3	-5	-5	-5	-5	-5	2	2	4	-5	4	5	-5	-2	2	-5	4	1	-4
3	4	4	4	0	0	2	3	0	1	1	3	-1	5	5	3	0	2	3	2	0	1	1	5	5	4	4	0	4	3
-2	-3	2	0	-3	-2	-5	-5	-5	0	-3	-5	2	2	0	0	3	-1	-3	0	5	-1	-3	0	-4	-2	-2	-2	-3	-5
0	-5	5	-5	2	0	0	2	0	1	2	0	5	1	-5	4	3	2	4	1	4	3	1	3	3	4	3	0	0	0

		Arithmet. Mittel																															Unzufriedenheit	Standardabweichung	Median	Modus
Eisstockschießen	25	0,83	1	0	0	0	0	0	0	0	0	2	0	0	0	1	0	0	2	0	0	0	0	2	2	0	0	0	0	0	0	0	10	2,66	1	0
Heimkino	62	2,07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2,03	2	1
Fußball	-26	-0,87	0	0	2	2	2	0	0	2	0	0	2	0	2	2	2	2	2	0	0	0	2	0	0	2	1	0	2	0	0	2	29	4,27	-1	-5
Kegeln	71	2,37	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2,38	3	3
Stadtbesichtigung	-45	-1,50	1	2	0	0	2	1	2	2	2	0	2	2	0	0	0	0	0	1	2	0	0	1	2	0	2	1	1	1	2	2	31	3,46	-2	-3
Spieleabend	38	1,27	0	2	0	2	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	2,66	1,5	0

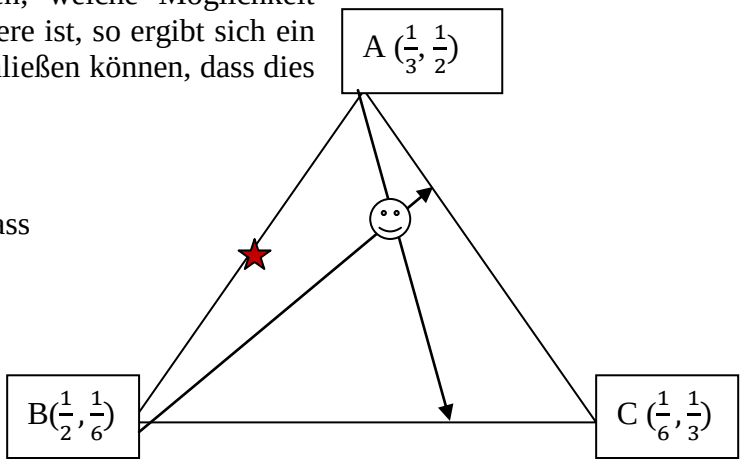
																															UZ/neg.P	UZ/P
2	0	0	0	0	0	0	0	0	3	0	0	0	2	0	0	5	0	0	0	0	3	5	0	0	0	0	0	0	0	20	3,33	0,67
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,00	0,00
0	0	5	5	5	0	0	3	0	0	5	0	5	5	5	5	5	0	0	0	5	0	0	5	2	0	5	0	0	4	69	4,60	2,30
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1,00	0,03
2	3	0	0	3	2	5	5	5	0	3	5	0	0	0	0	0	1	3	0	0	1	3	0	4	2	2	2	3	5	59	3,11	1,97
0	5	0	5	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	5,00	0,50

Als neuen Input bekamen wir eine kurze Einführung in die Spieltheorie. Wir gehen von drei verschiedenen Möglichkeiten aus, die von zwei Personen bewertet werden. Diese stellen wir graphisch als Dreieck dar. Die jeweiligen Eckpunkte entsprechen den Optionen, die erste Koordinate der Wertung der ersten Person (Summierung auf eins!), die zweite analog dazu der Wertung der zweiten Person. So sehe ich auf einen Blick, dass A die beste Lösung wäre. Die mathematische Erklärung dazu stellt die Spieltheorie dar.

		Person 2		
		x*A	y*B	z*C
Person 1	p*A	$(\frac{1}{3},\frac{1}{2})$	$(\frac{5}{12},\frac{1}{3})$	$(\frac{1}{4},\frac{5}{12}$[2]$)$
	q*B	$(\frac{5}{12},\frac{1}{3})$	$(\frac{1}{2},\frac{1}{6})$	$(\frac{1}{3},\frac{1}{4})$
	r*C	$(\frac{1}{4},\frac{5}{12})$	$(\frac{1}{3},\frac{1}{4})$	$(\frac{1}{6},\frac{1}{3})$

★ ….. Gleichgewicht
☺ …. Ergebnis

Wenn sich die Personen entscheiden müssten, welche Möglichkeit abgesehen von ihrer eigenen Präferenz die bessere ist, so ergibt sich ein Ergebnis, das A am nächsten ist, woraus wir schließen können, dass dies das beste Ergebnis ist.



Auch bei mehreren Wahldurchgängen bleibt dieses Ergebnis stabil, da man davon ausgeht, dass die Personen ihre Wertungen nur verstärken um ihr gewünschtes Ergebnis zu erreichen.

Als Ergänzung zu dieser Einführung haben wir später noch einen Film dazu angesehen.

Am dritten Tag besprachen noch einmal die Spieltheorie und griffen ein weiteres Wahlsystem auf, welches darauf aufbaut, dass die Wahlmöglichkeit die von den wählenden Personen mit dem geringsten Widerstand bedacht wurde, gewinnt.

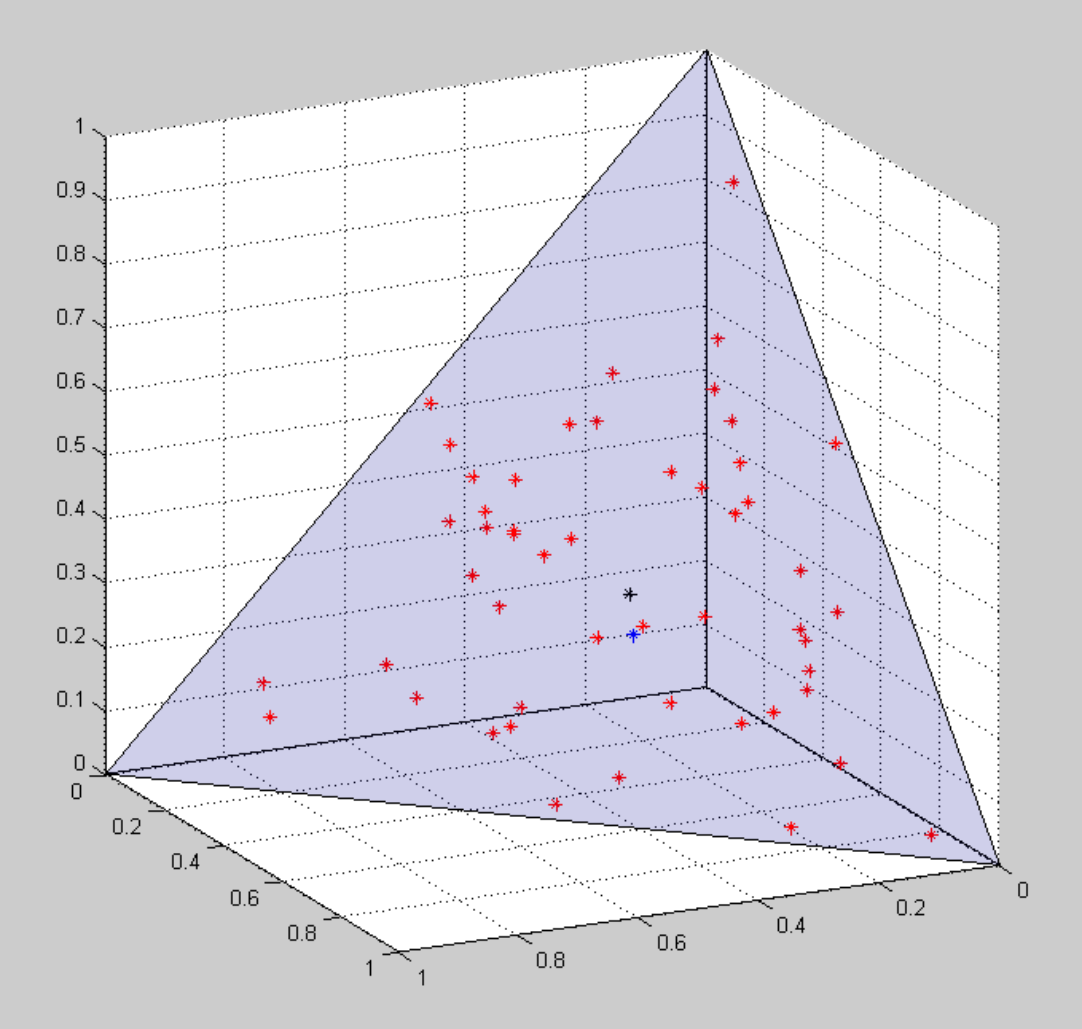
Dazu ein Beispiel: Es gibt drei Wahlmöglichkeiten und zwei Personen, die diese Möglichkeiten je nach Präferenz bewerten. Die Bewertungen haben insgesamt die Summe 1 und es gilt, je höher die Bewertung, desto beliebter die Wahlmöglichkeit. Daraus folgt, je kleiner die Bewertung desto größer ist der Widerstand der Person gegen die entsprechende Möglichkeit. Nun wird je Person auf der gegenüberliegenden Seite der beliebtesten Möglichkeit ein Widerstandspunkt gesetzt. Dieser befindet sich je näher an einer der verbliebenen Wahlmöglichkeiten desto unbeliebter die entsprechende Möglichkeit ist. Als nächstes werden die Widerstandspunkte der einzelnen Personen gemittelt und die Möglichkeit gewinnt, die am weitesten von diesem entfernt ist. Werden die Möglichkeiten um eine weitere Variante erhöht, so erhöht sich auch die Dimension der grafischen Darstellung um eins. Daher gilt, dass bei drei Möglichkeiten die grafische Darstellung zweidimensional und bei vier Möglichkeiten dreidimensional ist.

[2] $=(\frac{1}{3}+\frac{1}{2})/2$; analog dazu für alle anderen Felder

Nun teilten wir uns in drei Zweiergruppen auf.

Modellierung mithilfe von Widerstandspunkten

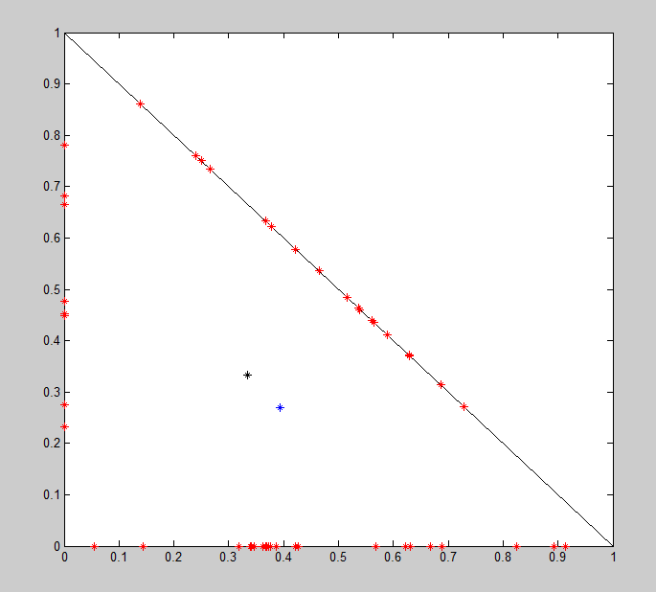
Eine Gruppe beschäftigte sich damit, das oben genannte Wahlsystem, basierend auf Widerstandspunkten, in Matlab zu programmieren, zu simulieren und auch grafisch darzustellen. Dabei wurde zuerst ein Programm geschrieben, das im Stande war, dies bei einem Wahlgang und drei Möglichkeiten zu tun. Die Verteilung der Stimmen wird jedes Mal mittels eines Zufallsgenerators neu ermittelt. Nach weiteren Arbeitsstunden wurde es auch möglich, eine Wahl bei vier Möglichkeiten zu simulieren und auch grafisch als Tetraeder darzustellen.

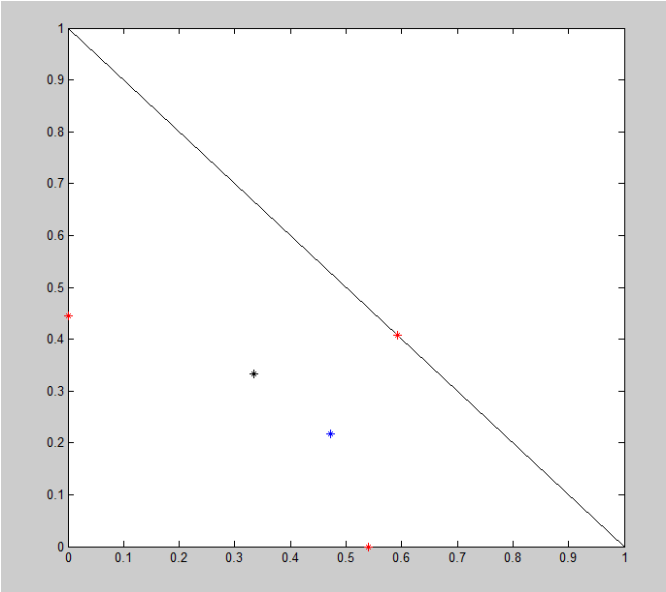


★ …Widerstandspunkte bei 50 Personen
★ …Punkt der gemittelten Widerstände (Die Option in der Spitze des Tetraeders gewinnt)
★ …Schwerpunkt des Tetraeders

Diese Gruppe beschäftigte sich auch damit, wie sich die wählenden Personen bei einem zweiten Wahlgang verhalten würden, wenn ihnen nur das Endergebnis des ersten Wahlgangs

und ihre Stimmabgabe bekannt sind. Nach längeren theoretischen Überlegungen kam diese Gruppe aber nur auf ein Ergebnis, bei dem sich die Widerstandspunkte in den Eckpunkten des Objekts, also den einzelnen Möglichkeiten sammeln würden. Diese Überlegungen stellten uns aber nicht zufrieden, da bei Selbstversuchen ein solches Ergebnis nicht zu Stande kam. Unser Gruppenbetreuer machte uns daraufhin auf eine Möglichkeit aufmerksam, nach der die wählenden Personen so handeln, dass sie beim Wählen den größtmöglichen Gewinn für sich erzielen. Der Gewinn ist dabei so definiert, dass die Person durch ihre neue Stimme versucht, das Ergebnis zu ihren Gunsten zu verändern beziehungsweise beizubehalten, sollte es für sie schon zufriedenstellend sein. Dazu wurde der mittlere Widerstandspunkt aus den Meinungen aller anderen berechnet und eine Gerade durch diesen und den beliebtesten Punkt gelegt. Dadurch schneidet die Gerade die Seite des Objekts, die dem beliebtesten Punkt gegenüber liegt; es entsteht ein neuer Punkt R. Nun fragt das Programm ab, wie hoch der Gewinn für die Person wäre, wenn bei der nächsten Wahl diese nun hier ihre Stimme setzt. Dasselbe fragt das Programm nun auch für die Eckpunkte, die nicht als beliebteste Möglichkeit ausgewählt wurden, und für den Mittelpunkt der Seite zwischen diesen. Das Programm errechnet nun, dass fast immer der Punkt R die beste Wahl ist. Dadurch wandern die Widerstandspunkte innerhalb einiger Wahlen zu den R-Punkten der Seiten.





Diese Simulation zeigt uns, dass mehrere Wahlgänge nicht sinnvoll sind, da der erste Wahlgang am aussagekräftigsten ist und mehrere Wahlgänge die Aussagekraft nur verfälschen.

Modellierung einer allgemein gültigen Formel

Eine weitere Gruppe beschäftigte sich an den folgenden Tagen weiter mit der idealen Formel für unsere Wahl.
Da die Formel, bei der wir arithmetisches Mittel sowie Unzufriedenheit pro Person berücksichtigten, sehr gut funktionierte, programmierten wir am Mittwoch ein Modell dieses Wahlsystems. Wir testeten das System mit einer unterschiedlichen Anzahl von Personen und Möglichkeiten, wobei die Ergebnisse immer übereinstimmten. An dem fertigen Modell testeten wir mögliche Wahlsituationen.

Eine weitere Tätigkeit an diesem Tag war der Versuch, eine allgemein gültige Formel aufzustellen. Da wir ja die Standardabweichung verworfen hatten, ergab es sich, die durchschnittliche Unzufriedenheit pro Person mit einzubauen, weil wir ja die Unzufriedenheit der Wähler trotz Verwerfens der Standardabweichung berücksichtigen wollten. Also entstand folgende Formel:

$$a = \frac{\bar{x}}{b+1}$$

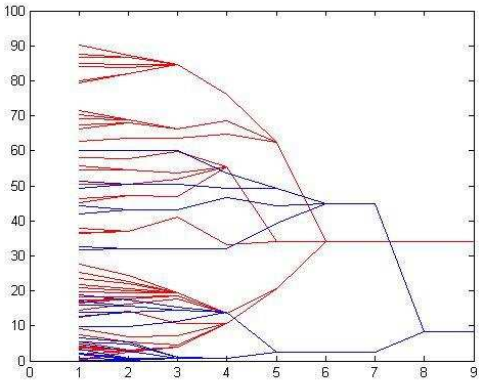
a…gesuchter Wert
b…durchschnittliche Unzufriedenheit pro Person

Es gilt nun: Je größer der gesuchte Wert, desto besser diese Wahlmöglichkeit. Ziel war es auch hier, die durchschnittliche Unzufriedenheit zu minimieren, während die Mehrheit für diese Möglichkeit ist (=arithmetisches Mittel). Im Gegensatz zur bisherigen Formel ist es auch wichtig, wie viele Leute wirklich unzufrieden sind und zusätzlich wird auch der Grad der Unzufriedenheit berücksichtigt, was in der bisherigen Formel nicht der Fall war. Durch die Addition von eins im Nenner verhindern wir eine Division durch null.
Wir probierten diese Formel anhand unserer Auswertung der Umfrage aus und befanden es für akzeptabel.
Veränderten wir einen negativen Wert in der Nähe von null bei der Möglichkeit Kegeln in einen klar negativen (zum Beispiel -5), so änderte sich a so, dass in diesem Fall Heimkino gewann, was unserer Meinung nach richtig war, da im Vergleich Heimkino- auch wenn das arithmetische Mittel nicht so groß war wie bei Kegeln- weitaus mehr zufriedene Leute hat als Kegeln. Bei unserer ursprünglichen Auswertung hatte Kegeln nur eine negative Stimme, diese Person war allerdings nicht strikt dagegen, was zur Folge hatte, dass Kegeln gewann.

Zusätzlich programmierten wir in Matlab eine Möglichkeit zur Gruppenbildung bei mehreren Wahlgängen. Wir gingen davon aus, dass in diesem Modell auch Dominanz eine wichtige Rolle spielt. Dies könnte de facto später umgelegt werden auf Manipulation oder Wahlwerbung vor einer Wahl. Im zweiten Wahlgang wäre die Meinung M dann

$$M(i) = \frac{\sum_i^j D(i) * Malt(i)}{\prod_i^j D(i)}$$

D…. Dominanz (Zufallsvariable)
M….Meinung nach n Wahlgängen
Malt…Meinung vor dem n-ten Wahlgang



An dieser Grafik sehen wir, wie sich die verschiedenen Meinungen so lange zusammenschließen, bis eine „Gesamtmeinung“ heraus kommt. Die Farben rot und blau stehen für die zwei Wahlmöglichkeiten. Zu jeder von diesen hat jeder der insgesamt 50 Personen eine Meinung zwischen 0 und 100. Gewonnen hat die Meinung, bei der am Schluss die Gesamtmeinung am höchsten ist, in diesem

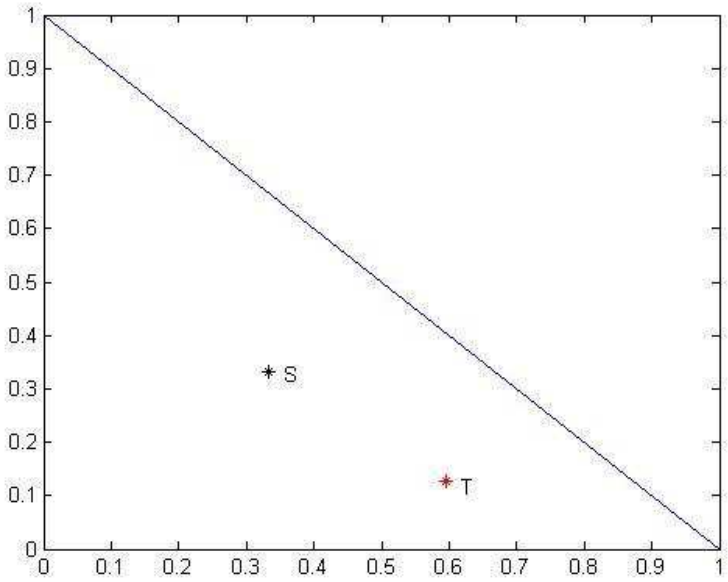
Fall Wahlmöglichkeit rot. Im Endeffekt entspricht die Endmeinung dem arithmetischen Mittel. Dies erschien für uns als richtig, da jeder andere Wert von Wahlmanipulation zeugen würde. Die Tatsache, dass auch hier wieder das arithmetische Mittel heraus kommt liegt daran, dass die Dominanz eine Zufallsvariable ist, die sich immer gleichmäßig verteilt, weswegen sie hier keinen Einfluss auf das Endergebnis hat.

Da wir nun das Wahlsystem mit den Wertungen zwischen -5 und 5 während eines Wahlganges ausreichend getestet hatten, überlegten wir uns, was passieren würde, gäbe es mehrere Wahlgänge und arbeiteten an einer Programmierung auf dem Computer.
Bei unseren theoretischen Überlegungen kamen wir zu dem Schluss, dass sich schließlich ein Gleichgewicht einstellen würde, bei dem keine der Wähler seinen Gewinn noch optimieren kann. Dieses Gleichgewicht ist dem Anfangsergebnis sehr ähnlich. Ist das Ergebnis anders und vergleicht man die beiden Ergebnisse anhand der ehrlichen Bewertungen miteinander, so sieht man, dass das erste Ergebnis besser ist, da es den ehrlichen Meinungen eher entspricht. Nach mehreren Wahlgängen wird auch die Unzufriedenheit dementsprechend größer, da jeder nur mehr auf seinen eigenen Vorteil bedacht ist. Auch diese wird irgendwann zu einem gewissen Gleichgewicht konvergieren.

Modellierung mithilfe von Präferenzpunkten

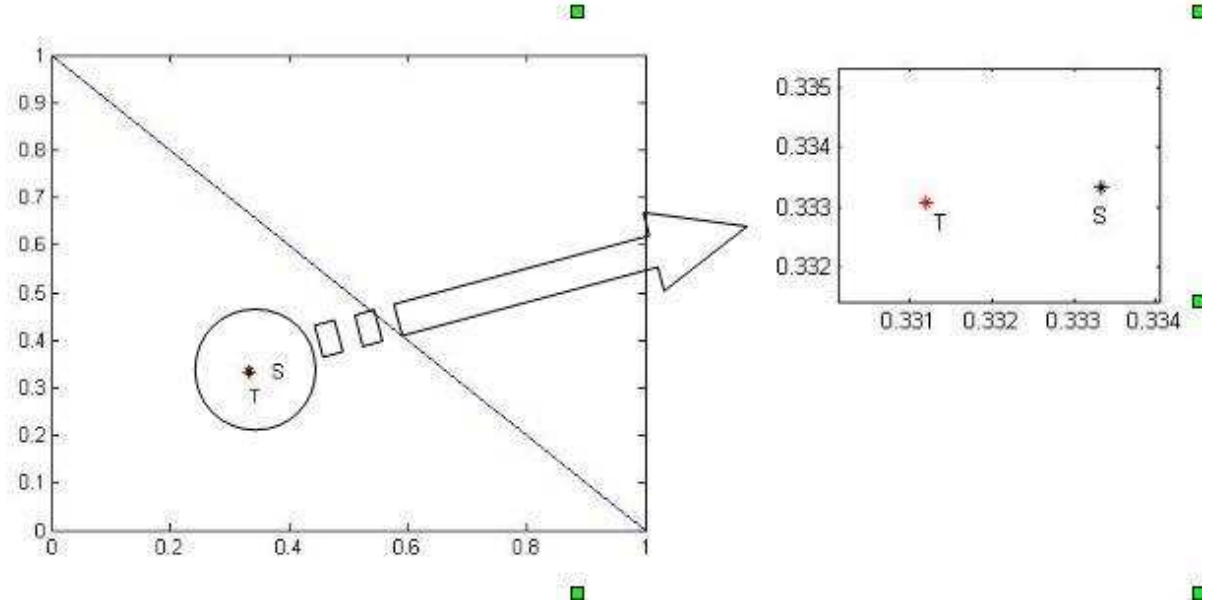
Die Basis unseres präferenzbezogenem Modells bilden drei Wahlmöglichkeiten m, die von zwei Spielern n bewertet werden. Zur Visualisierung wird ein Dreiecksmodell verwendet. Dieses Modell bezieht sich auf ein System mit nur einem Wahlgang. Im weiteren Evolutionsprozess unseres Modells extendieren wir auf mehrere Wahlgänge.

Den Punkten des Dreiecks werden einfachheitshalber die Koordinaten A(0/0), B(1/0) und C(0/1) zugewiesen, da diese Einteilung das Programmieren des Modells vereinfacht. Grundsätzlich sind die Koordinaten irrelevant für das Ergebnis eines gemeinsamen Präferenzpunktes T. Den Spielern werden zufällige Präferenzen für die Wahlmöglichkeiten zugewiesen, welche immer auf eins summieren. Durch diese spielerbezogenen Präferenzen ergibt sich für jeden Spieler ein Präferenzpunkt im Dreieck. Der Mittelwert dieser Präferenzpunkte P_n definiert den gemeinsamen Präferenzpunkt T.



Dreieicksmodell

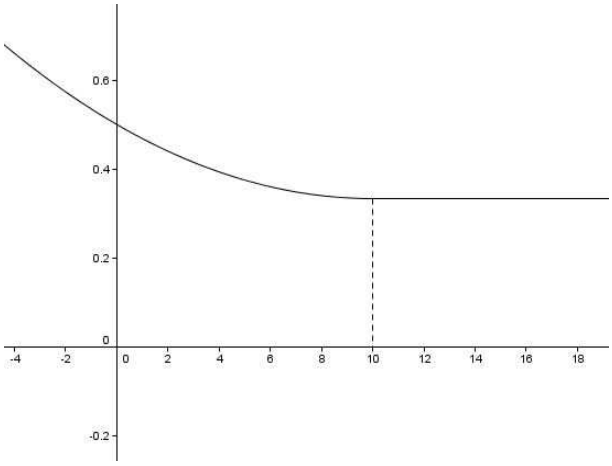
Steigert man die Anzahl der Spieler tendiert T zum Schwerpunkt des Dreiecks. Dieses Verhalten kann durch ein annäherndes Gleichgewicht der gesetzten Präferenzen erklärt werden.
Die Dimension des Modells verändert sich gleich wie bei der Modellierung mit Widerstandspunkten.



Wahlvorgang mit n= 10000

Für das Modell mit mehreren Wahlgängen ändern sich die Präferenzen der Spieler. Wir nehmen an, dass der Spieler seine bevorzugte Option höher gewichtet als im ersten Wahlgang, folglich müssen die anderen Präferenzen benachteiligt werden. Für diese Auf- beziehungsweise Abwertung wird ein dynamisches Gewicht verwendet, welche aus einer Funktion(1.1) besteht, die im Intervall [0;10] streng monoton fallend ist und das Verhalten einer Parabel bis zu ihrem Tiefpunkt aufweist, danach ist sie als lineare Funktion parallel zur x-Achse:$y = \frac{1}{3}$, definiert.

$$f(x) = \frac{1+sgn(10-x)}{2} \cdot \frac{(10-x)^2}{600} + \frac{1}{3} \qquad (1.1)$$



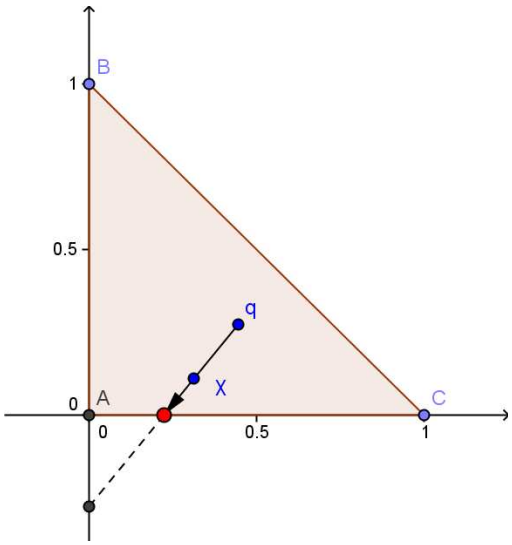
Funktion

Das heißt, mit zunehmenden Wahlgängen W_n nimmt das Gewicht ab und stabilisiert sich nach $W_n = 10$.

Bei Verwendung dieses Modells wird hingegen die Präzision vernachlässigt, da der Spieler unabhängig von dem Ausgang des vorhergehenden Wahlganges in den Punkt seiner größten Präferenz zieht.

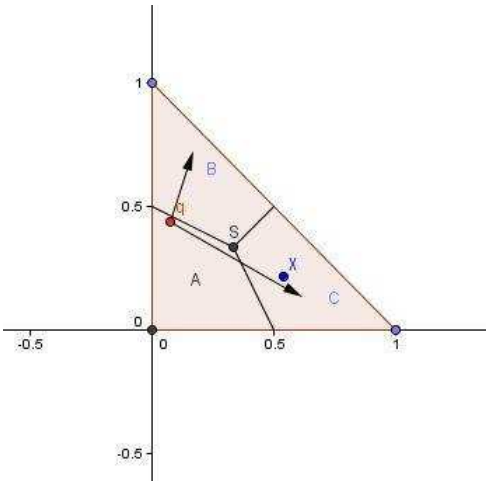
Zur Steigerung der Präzision bei weiteren Wahlgängen verwenden wir ein vordefiniertes taktisches Verhalten für jeden Spieler. Jeder Spieler versucht auf Basis des Präferenzpunktes der anderen Spieler q des vorhergehenden Wahlganges seinen Gewinn zu optimieren. Wir definieren eine Anzahl von $2 \cdot m + 1$ Möglichkeiten für taktische Züge. Diese setzen sich zusammen aus:

- den Eckpunkten
- den Mittelpunkten der Kanten des Polygons
- einem besonderen Punkt der wie folgt definiert wird: Der Vektor $\overrightarrow{qx}$, wobei x den eigenen Präferenzpunkt repräsentiert, wird so lange verlängert bis einer der Präferenzkoeffizienten gleich null ist.

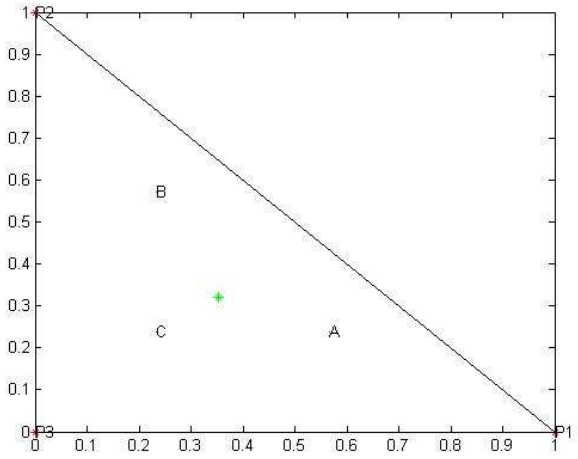


Berechnung des besonderen Punktes

Jeder Spieler überprüft anhand des q-Wertes der vorhergehenden Wahl seinen Gewinn, welchen er durch einen Zug zu einem der vorhin genannten Punkte erzielen könnte. Daraufhin wählt er diesen Punkt für den nächsten Wahlgang aus. Dabei kommt es oft zu Handlungen entgegen des eigenen Instinkts, da der Spieler in gewissen Situationen gezwungen ist näher zu seiner zweiten Wahl zu ziehen.
Zur Veranschaulichung ein Beispiel für eine derartige Situation:
Im Schwerpunkt S würde ein Gleichgewicht zwischen allen Optionen entstehen. Zurzeit befindet sich der Wert der andern Personen q im Bereich A, jedoch sehr nahe an der Grenze zu B. Wir gehen davon aus, dass die betroffen Person x die Optionen A, B, C folgendermaßen bewertet hat: C > B > A
Für die Person ist es nicht möglich mit ihrer Stimme q in den Bereich C zu ziehen also muss sie sich für ihre zweite Wahl B entscheiden.



Bei Durchführung mehrerer Wahlgänge dieses Systems ist eine klare Tendenz zu erkennen: Alle Spieler tendieren zu den Eckpunkten des Polygons.
In Abbildung 6 sehen wir das Ergebnis nach 2 Wahldurchgängen mit n=3 und m=3.

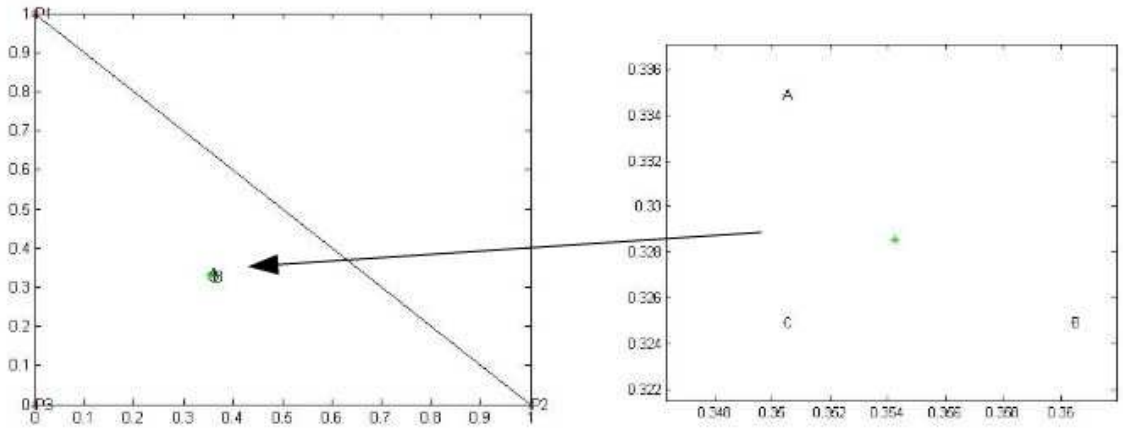


2 Wahlgänge; m=3; n=3

Die Personen P1, P2 und P3 befinden sich bereits in den Eckpunkten des Polygons, da sie nach dem ersten Wahlgang bereits eine Optimierung vorgenommen haben. Die Punkte A, B und C stellen eine erhoffte Veränderung der Person für den Fall, dass die anderen Personen ihre Wahl nicht verändert hätten (A für P1, B für P2 und C für P3), dar.
Diese Punkte A, B und C werden mit folgender Formel berechnet:

$$A, B, C, \ldots, Q(n) = \frac{(n-1)q + Pn}{n}$$

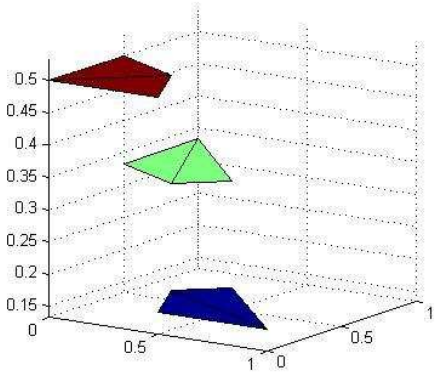
Da jedoch jede Person ihre Stimme optimiert läuft alles auf ein Gleichgewicht im Schwerpunkt hinaus.
Umso höher die Anzahl der Personen beziehungsweise die Anzahl der Wahlgänge ist umso weniger Möglichkeit hat eine einzelne Person ihren Gewinn nach mehreren Wahlgängen zu optimieren. Dieses scheinbare Gleichgewicht wird durch das nächste Beispiel belegt. In diesem haben wir auf n=100, m=3 und 5 Wahldurchgänge erweitert:



Beispiel n=100; m=3; 5 Wahlgänge

Es entsteht ein Gleichgewicht im Schwerpunkt, da sich alle Wähler (P1,P2,P3) in den Eckpunkten befinden. Eine weitere Gewinnoptimierung ist für jede Person auszuschließen. Mit mehreren Wahlgängen bemerken wir eine Konvergenz im Schwerpunkt. Wir haben durch dieses System ein gewünschtes Gleichgewicht erhalten, welches jedoch auch einer unlösbaren Konfliktsituation zwischen den Spielern entspricht.

Weiters kann man eine Unstetigkeit bei der Zielfunktion beobachten. Der Wechsel des Präferenzpunktes zwischen dem Bereich zweier Optionen geschieht sprunghaft. Bei der folgenden Repräsentation der Zielfunktion kann man diese Unstetigkeit durch den Höhenunterschied zwischen den verschiedenen Ebenen erkennen.



Zusammenfassung:

Von unseren verschiedenen Modellen können wir ableiten, dass das Ergebnis durch mehrere Wahlgänge nicht verfeinert wird, sondern nur auf ein Gleichgewicht hingeführt wird. Das scheinbar ehrlichste und ausschlaggebendste Ergebnis erhält man nach nur einem Wahlgang, ohne jegliche persönliche Präferenz- beziehungsweise Widerstandsoptimierung.

Der einzige Vorteil bei mehreren Wahlgängen wäre die Behauptung, ein stabiles Ergebnis zu erhalten, was allerdings in der Wirklichkeit wiederum nicht sehr vorteilhaft wäre, da das Ergebnis verfälscht wäre, sodass man keine Aussage mehr treffen könnte.

Wichtig für unser Modell ist aber, dass Gewichtungen trotzdem vorhanden sind, die sowohl in den positiven als auch in den negativen Bereich fallen. Dies soll garantieren, dass jeder seine Meinung möglichst nahe der Realität ausdrücken kann.

Unsere Definition des Gemeinwohls ist: Für uns ist es ausschlaggebend, dass jeweils die Meinung der unzufriedenen Wähler als auch der zufriedenen bezüglich der verschiedenen Wahlmöglichkeiten mit einbezogen wird. Ziel ist es, die Unzufriedenheit zu minimieren und die Zufriedenheit der Gesamtheit zu maximieren.
Also setzten wir dies, wie schon oben besprochen, in unserer Formel um, die unserer Meinung nach dem Gemeinwohl so nahe wie möglich kommt.

$$a = \frac{1/n \sum_{i=1}^{n} x(i)}{1/n \sum_{i=1}^{m} x(i) + 1}$$

a…gesuchter Wert
n… Anzahl der Wähler
m… Anzahl der unzufriedenen Wähler

Unser entwickeltes Wahlsystem bestätigt in vielen Fällen den Satz von Arrow, da immer mindestens ein Axiom verletzt wird. Verletzt werden unter anderem das erste, fünfte und sechste Axiom.