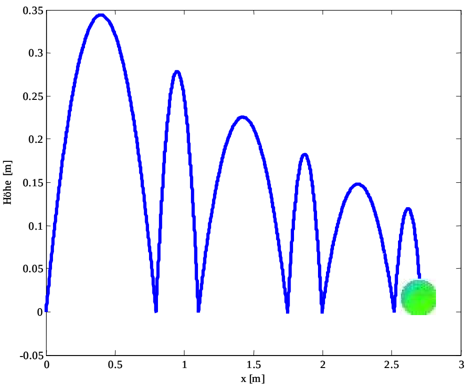
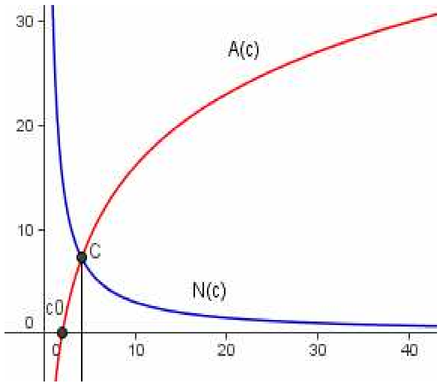
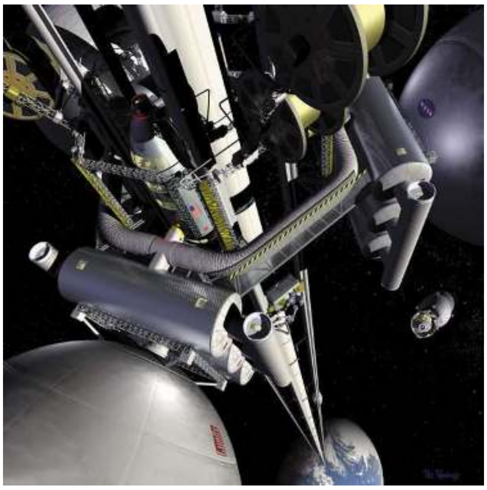
WOCHE DER MODELLIERUNG MIT MATHEMATIK

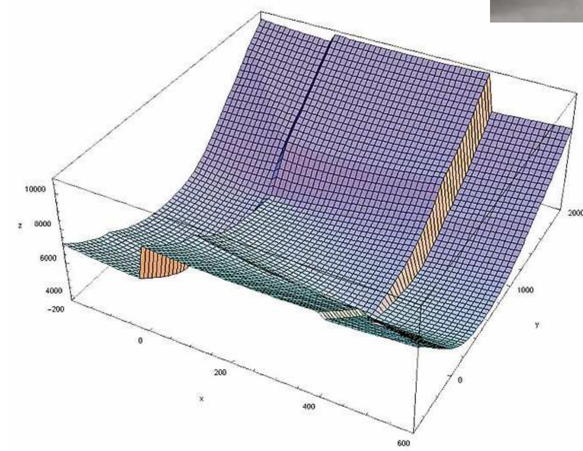
Dokumentationsbroschüre

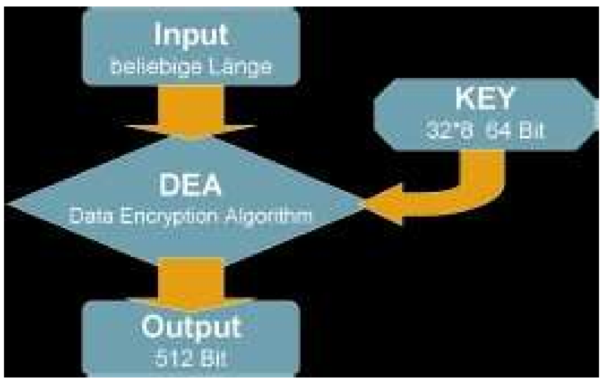
13. – 19. Jänner 2008



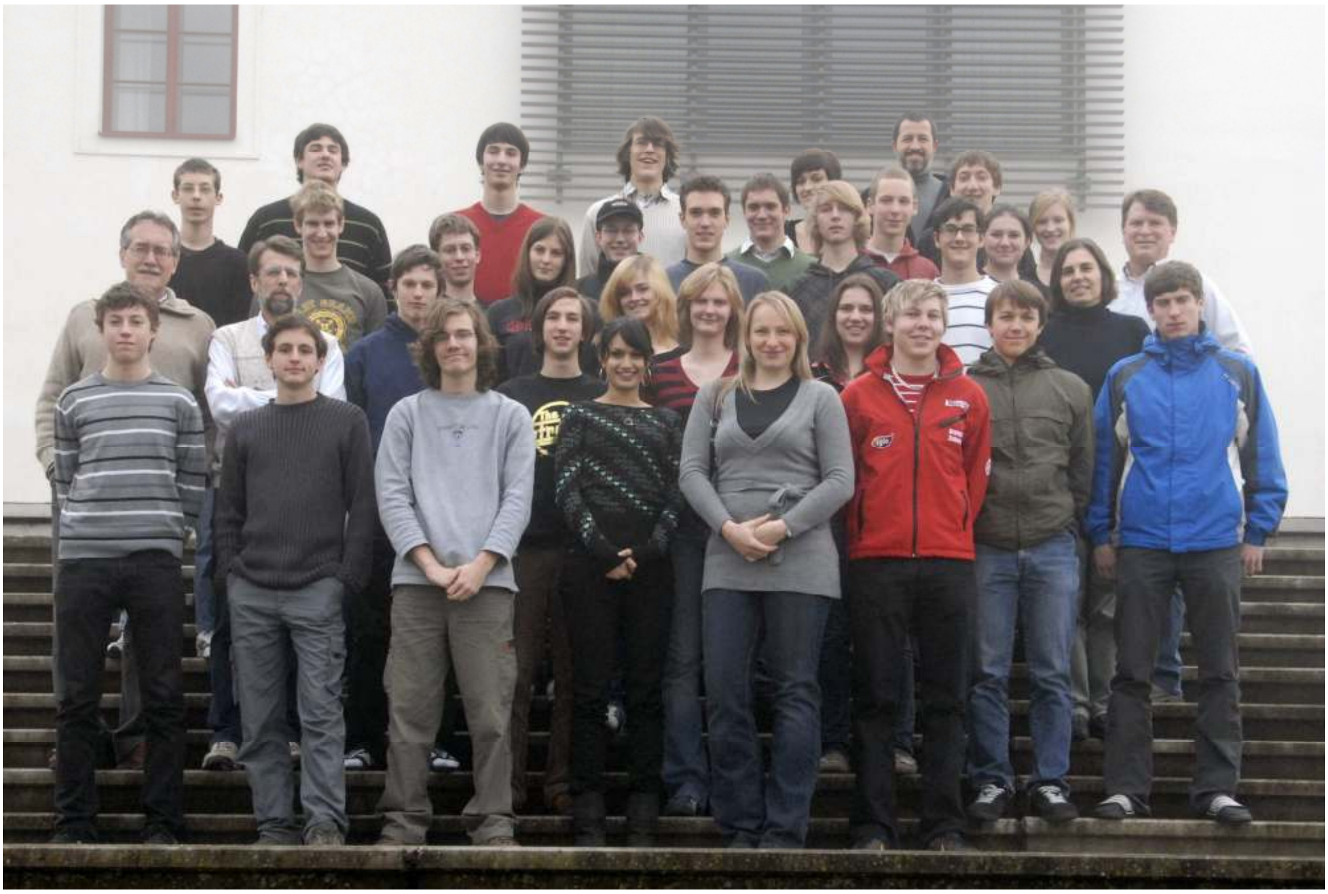








Woche der Modellierung mit Mathematik



Schloss Seggau, 13.01. – 19.01.2008

Weitere Informationen:
http://math.uni-graz.at/modellwoche/2008/

Vorwort

Die Idee zu der in der Steiermark durchgeführten „Modellierungswoche“ für Schüler der 7. und 8. Klasse der AHS wurde schon längere Zeit am Institut für Mathematik und Wissenschaftliches Rechnen der Universität Graz diskutiert. Vorbild waren ähnliche Vorhaben, die bereits in Kaiserslautern, in Bozen und auch in Linz durchgeführt wurden. Mitglieder des Institutes haben bereits Erfahrungen mit ähnlichen Veranstaltungen für Studierende und angehende Wissenschafter. Im Jahre 2005 wurde vom Institut für Mathematik und Wissenschaftliches Rechnen der Karl-Franzens-Universität erstmals eine Modellierungswoche durchgeführt, die bei allen teilnehmenden Schülern und Schülerinnen großen Anklang gefunden hat. Ermutigt durch diesen durchschlagenden Erfolg haben wir seither jedes Jahr wieder eine Modellierungswoche angeboten. Hauptziel der Modellierungswoche ist es, Schüler und Schülerinnen mit einem Aspekt der Mathematik zu befassen, der unserer Meinung nach im Unterricht an den AHS unterrepräsentiert ist: Die Rolle der Mathematik als Werkzeug zum Verständnis der Welt, die uns in Alltag und Wissenschaft umgibt. Während ihrer gesamten Geschichte stand die Mathematik immer in Wechselwirkung mit angewandten Bereichen. Viele mathematische Theorien entstanden in Reaktion auf Anforderungen aus den verschiedensten Anwendungsbereichen. Die Verfügbarkeit immer leistungsfähigerer Computer hat neue Möglichkeiten für die mathematische Behandlung verschiedenster komplexer Probleme eröffnet. Quantitative Resultate statt qualitativer Aussagen sind immer wichtiger und erfordern zu ihrer Bewältigung die mathematische Modellierung komplexer Systeme in interdisziplinärer Zusammenarbeit.

Den an der Modellierungswoche teilnehmenden Schülern und Schülerinnen sollte an Hand sorgfältig ausgewählter Projektaufgaben Gelegenheit gegeben werden, den angewandten Aspekt der Mathematik durch Teamarbeit in Projektgruppen zu erleben. Es wurde versucht, den Teilnehmenden die wesentlichen Phasen eines Modellierungsprozesses nahe zu bringen: Einarbeiten in das Anwendungsgebiet, Wahl der Modellstruktur in Hinblick auf die Aufgabenstellung, Einsatz numerischer Methoden, Interpretation der Ergebnisse, Präsentation der Resultate.

Treibende Kraft für die Realisierung der Modellierungswoche ist Dr. Stephen Keeling, dem hier für seinen großen Einsatz gedankt sei. Besonderer Dank gebührt dem Landesschulrat für Steiermark, und hier insbesondere Frau Landesschulinspektorin Hofrat Mag. Marlies Liebscher. Sie hat die Idee einer gemeinsamen Veranstaltung sofort sehr positiv aufgenommen und tatkräftig unterstützt. Ohne den großen Einsatz der direkten Projektbetreuer, Dr. Sigrid Thaller – Institut für Sportwissenschaft, Dr. Günter Lettl, Dr. Stephen Keeling, Dr. Peter Schöpf und Dr. Bernd Thaller – alle Institut für Mathematik und Wissenschaftliches Rechnen, und der Betreuerin aus dem Kreis der Lehrerschaft, Mag. Melanie Wogrin, die auch die Gestaltung dieses Berichtes übernommen hat, wäre die Modellierungswoche nicht durchführbar gewesen. Eine wesentliche Rolle im Organisationsteam der Modellierungswoche spielten Gerlinde Krois und Dr. Georg Probst vom Institut für Mathematik und Wissenschaftliches Rechnen. Wir danken der Abteilung für Wissenschaft und Forschung der Landesregierung Steiermark und der Bank Austria – Creditanstalt für ihre Subvention.

Wir danken Frau Landesschulinspektorin Hofrat Mag. Marlies Liebscher und Herrn Präsident Wolfgang Erlitz vom Landesschulrat für Steiermark und Vizerektor Martin Polaschek und Dekan Karl Crailsheim von der Universität Graz für die finanzielle Unterstützung.

Schloss Seggau, am 19. 1. 2008

F. Kappel
(Leiter des Institutes für Mathematik und Wissenschaftliches Rechnen)

REFLEXION VON BÄLLEN

Projekt: *Sportwissenschaften*
Betreuerin: *Dr. Sigrid Thaller*
Modellierer: *Lisa Dittmer, Tanja Koch, Hans Körner, Michael Lasnik, Alexander Leitner, Patrick Reicher*

In diesem Projekt wird das Reflexionsverhalten von Bällen unterschiedlichster Art untersucht. Wie hängt der Reflexionswinkel von der Rotation, der Elastizität oder der Geschwindigkeit des Balles ab? Gibt es noch weitere Einflussgrößen? Was passiert, wenn man einen rotierenden Hartgummiball in eine Ecke wirft?

FREIER FALL

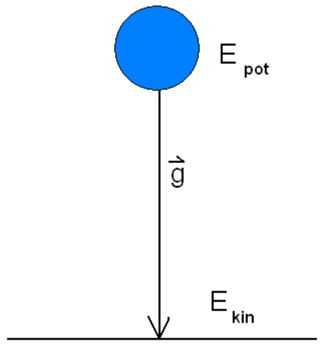
Stellen wir uns vor, wir sitzen auf einem Baum und lassen von dort einen Gummiball fallen. Wir beobachten dabei, dass der Ball immer schneller fällt bis er am Boden aufkommt. Aus physikalischer Sicht bezeichnet man dies als „Freien Fall" (wobei wir hier den Luftwiderstand vernachlässigen). Der freie Fall ist eine gleichförmig beschleunigte Bewegung, in der ausschließlich die Fallbeschleunigung wirksam wird.

Auf der Erde ist in der Regel ein vollkommener freier Fall nicht möglich, da er durch andere Kräfte, wie den Luftwiderstand, beeinflusst wird. Ein vollkommener freier Fall gibt es nur im Vakuum, da sich nur dort eine lineare Beschleunigung möglich ist. Auf der Erde schwankt der Betrag der Fallbeschleunigung je nach geographischer Lage. Der Wert für die Erdbeschleunigung wird durchschnittlich mit g = 9,81 m/s^2 angegeben. Beim Freien Fall in Erdnähe vergrößert sich, wenn der Luftwiderstand nicht berücksichtigt wird, die Geschwindigkeit v eines fallenden Körpers um 9,81 m/s pro Sekunde.

Welchen Kräften ist der Ball ausgesetzt?
Bevor wir den Ball loslassen, besitzt der Ball eine bestimmte **potentielle Energie**. Sie wird auch als Energie der Lage bezeichnet. Es handelt sich dabei um diejenige Energie, welche ein Körper durch seine Position oder Lage in einem Kraftfeld (etwa einem Gravitationsfeld oder elektrischen Feld) enthält. Während des Falles wird diese potentielle Energie in kinetische Energie umgewandelt. **Kinetische Energie** ist jene Bewegungsenergie, die in der bewegten Masse eines Körpers enthalten ist.

$$E_{kin} = \frac{m \cdot v^2}{2}$$

$$E_{pot} = m \cdot g \cdot h$$



ELASTIZITÄTSKOEFFIZIENT

Um später praktische Experimente durchführen zu können, mussten wir die notwendigen Parameter ebenfalls experimentell überprüfen. Zu Beginn unserer Arbeit begannen wir den Elastizitätskoeffizienten *e* zu messen. Der Elastizitätskoeffizient gibt das Verhältnis der Geschwindigkeit vor und nach dem ersten Aufprall an. Da wir die Geschwindigkeiten nicht exakt messen konnten, haben wir den Koeffizient wie folgt hergeleitet:
Allgemein ist bekannt, dass sich die Geschwindigkeit des freien Falls mit der Höhe folgendermaßen zusammen hängt:

$$v_{\text{Freier Fall}} = \sqrt{2 \cdot g \cdot h}$$

Aus dem Verhältnis der Geschwindigkeiten vor und nach dem Aufprall erhält man also

$$v_2 : v_1 = \frac{\sqrt{2 \cdot g \cdot h_2}}{\sqrt{2 \cdot g \cdot h_1}}$$

Durch Kürzen erhält man folgende Endformel

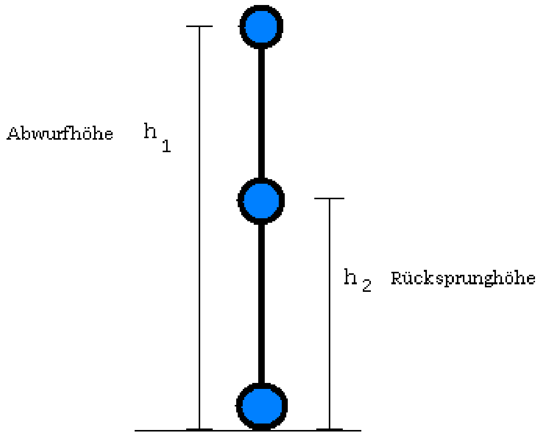
$$e = \sqrt{\frac{\text{Rücksprunghöhe}}{\text{Abwurfhöhe}}}$$

Wir definieren: h_1 sei die Abwurfhöhe
h_2 sei die Rückspringhöhe

$$e = \sqrt{\frac{h_2}{h_1}}$$

Da h_1 stets größer sein muss als h_2 und beide natürlich nicht negativ werden können, ist ein Wert zwischen 0 und 1 zu erwarten. Hierbei gilt, je höher der Wert, desto elastischer ist der Ball und daraus folgt auch ein größerer Wert der Rücksprunghöhe.

Nach der theoretischen Herleitung müssen diese Werte für jeden Ball individuell ermittelt werden, indem von einer bestimmten Abwurfhöhe die Rücksprunghöhe nach dem Aufprall gemessen wurde. Dabei wurden die einzelnen Sprünge gefilmt und in Zeitlupe analysiert, um möglichst genaue Werte zu erhalten. Weiters wurden pro Ball fünf Versuche durchgeführt und als konkreter Wert der Durchschnitt aus diesen Versuchen genommen, um Messungenauigkeiten möglichst zu unterbinden.



Diese Versuche ergaben folgende Werte:

Ball	Elastizitäts-koeffizient
Minifußball	0,845
Tennisball	0,779
Gummiball (groß)	0,887
Tischtennisball	0,846
Gummiball (klein)	0,891

Wie man hier sieht, ist der kleine Gummiball der elastischste Ball, da er den größten Koeffizienten hat. Der etwas in die Jahre gekommene Tennisball weist hingegen den kleinsten Wert auf und hat somit die geringste Rücksprunghöhe.

1. Modell: Die gleitende Kugel bzw. der gleitende Kontakt

In unserem ersten Modell nahmen wir an, dass der Ball bei Kontakt mit dem Boden gleitet oder besser gesagt durchrutscht.

Warum gleitet ein Ball?
Wenn der Ball mit sehr hoher Geschwindigkeit und starker Rotation auf dem Boden auftrifft, und dadurch die Reibung zu gering ist um den Ball abzubremsen, gleitet der Ball.

Physikalische Grundlagen:

Trägheitsmoment *I*: Das Trägheitsmoment, ist eine physikalische Größe, die in der Mechanik die Trägheit eines starren Körpers gegenüber einer Änderung seiner Rotationsbewegung angibt.

Das Trägheitsmoment einer vollen Kugel ist:

$$I = \frac{2}{5} \cdot m \cdot r^2$$

(m = Masse, r = Radius)

Drehimpuls *L*: Der Drehimpuls ist eine physikalische Größe, welche Richtung und Geschwindigkeit einer Drehbewegung um einen Punkt beschreibt. Der Drehimpuls einer Kugel um ihren Auflagepunkt ist:

$$L = I \cdot \omega - m \cdot r \cdot v$$

(ω = Winkelgeschwindigkeit, v = Translationsgeschwindigkeit)

Elastizitätskoeffizient *e*: Der Elastizitätskoeffizient bestimmt die Rückspringhöhe h_2 aus der Ausgangshöhe h_1:

$$e = \sqrt{\frac{h_2}{h_1}}$$

Unelastischer Stoß: Beim unelastischen Stoß wird ein Teil der kinetischen Energie in innere Energie (U) umgewandelt. Nach jedem Aufprall eines hüpfenden Balles springt er weniger hoch zurück, sprich die kinetische Energie des Balles wird mit jedem Aufprall geringer. Auch aufgrund von Reibung und anderen Faktoren geht so einem System an kinetischer Energie verloren.

$$|u_y| = |e \cdot v_y|$$

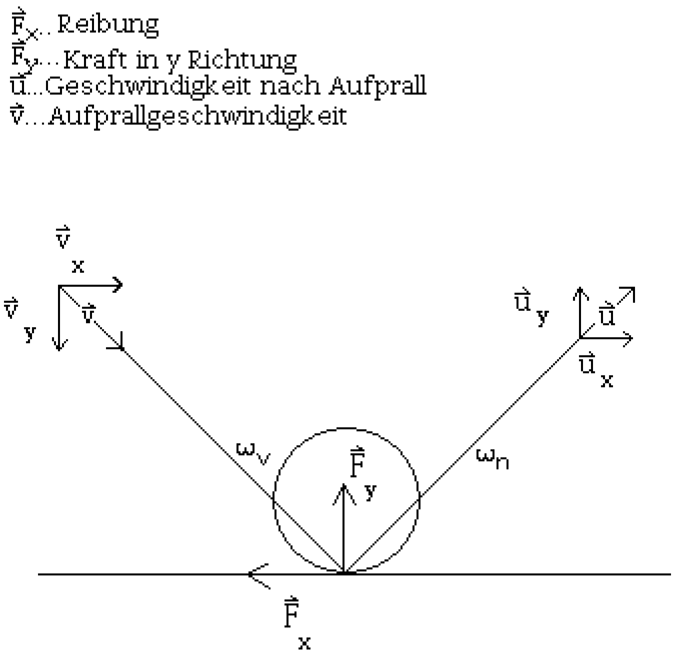
(v_y = Geschwindigkeit vor dem Aufprall, u_y = Geschwindigkeit nach dem Aufprall)

Winkelgeschwindigkeit *ω*: Die Winkelgeschwindigkeit ist die Ableitung des Winkels nach der Zeit. Sie gibt an, wie schnell sich etwas dreht. Man könnte auch „Winkeländerung pro Zeitspanne“ sagen. Sie ist unabhängig vom Radius.

Reibungskoeffizient *μ*: Der Reibungskoeffizient gibt an, wie die Reibungskraft F_x (z.B. zwischen Ball und Boden) von der Normalkraft F_y abhängt.

Modellierung der Reflexion:

Skizze des Aufpralls eines Balles:



Mit Hilfe der bereits vorgegeben Formeln der Impulsänderung beim Stoß, der Drehimpulsänderung und der Formel des unelastischen Stoßes konnten wir durch Umformen die Bedingungen eines jeden Balles nach dem Aufprall auf dem Boden darstellen.

Formeln:

- Impulsänderung *M* beim Stoß: $m \cdot (v_x - u_x) = M_x$

  $m \cdot (v_y - u_y) = M_y$

- Drehimpulsänderung: $I \cdot (\vec{\omega}_v - \vec{\omega}_n) = M_x \cdot r$

- Unelastischer Stoß: $|u_y| = |e \cdot v_y|$

- Reibung: $F_x = \mu \cdot |F_y|$

Diese Formeln werden nun so umgeformt, dass wir mit Hilfe von den „Eingangsgrößen“(v_x, v_y, ω_v, e, r und μ) die „Ausgangsgrößen“ (u_x, u_y, ω_n) berechnen können. Somit können wir aufgrund des angegebenen Eintrittswinkels den Austrittswinkel und die Rotationsgeschwindigkeit des Balles vorhersagen.

Umformung:

1. Wir bringen die Impulsänderung M_x in Verbindung mit dem Reibungskoeffizienten und der Impulsänderung M_y.

$$\left|M_x\right| = \left|\int F_x \, dt\right| = \left|\int \mu \cdot F_y \, dt\right| = \left|\mu \cdot M_y\right|$$

2. Aus 1. ersetzen wir M_y durch $m \cdot (v_y - u_y)$ *und bauen den Elastizitätskoeffizienten ein.*

$$\begin{aligned} m \cdot (v_x - u_x) &= \left|M_x\right| \\ &= \mu \cdot m \cdot (v_y - u_y) \\ &= \mu \cdot m \cdot \left(v_y - e \cdot v_y\right) \\ &= -\mu \cdot m \cdot v_y \cdot (1 + e) \end{aligned}$$

3. Nun setzen wir in die umgeformte Drehimpulsänderung 2. ein.

$$\begin{aligned} \omega_v - \omega_n &= \frac{M_x \cdot r}{I} \\ &= \frac{-\mu \cdot m \cdot v_y \cdot (1 + e) \cdot r}{\frac{2}{5} \cdot m \cdot r^2} \\ &= \frac{-5 \cdot \mu \cdot v_y \cdot (1 + e)}{2r} \end{aligned}$$

4. Jetzt wird auf u_x, u_y und ω_n umgeformt.

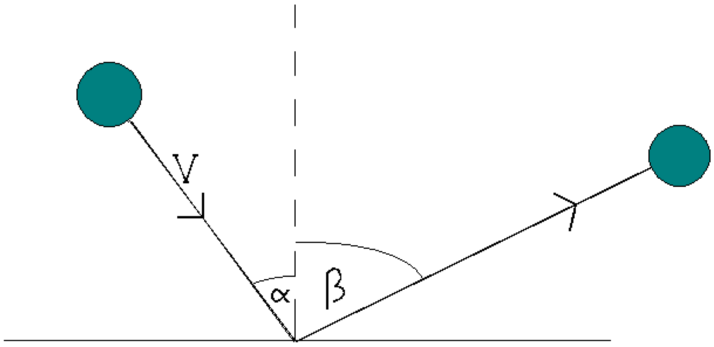
$$\begin{aligned} u_x &= v_x + v_y \cdot \mu \cdot (1 + e) \\ u_y &= -e \cdot v_y \\ \omega_n &= \omega_v + \frac{5 \cdot \mu \cdot v_y \cdot (1 + e)}{2r} \end{aligned}$$

<u>Beispiel:</u> Cricketball

Daten: Durchmesser: 7 cm= 0,07 m
Anfangsgeschwindigkeit: 5 m/s
Eintrittswinkel zum Lot: 30°
Reibungskoeffizient: 0,6

Elastizitätskoeffizient: 0,7
Winkelgeschwindigkeit (ω): 400 m/s

Skizze:

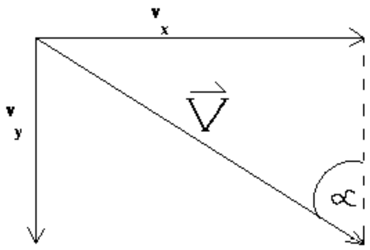


Formeln:

$$u_x = v_x + v_y \cdot \mu \cdot (1 + e)$$

$$u_y = -e \cdot v_y$$

$$\omega_n = \omega_v + \frac{5 \cdot \mu \cdot v_y \cdot (1 + e)}{2r}$$



$$\sin\alpha = \frac{v_x}{v}$$

$$\sin\alpha \cdot v = v_x$$

$$v_x = 2,5\ \frac{m}{s}$$

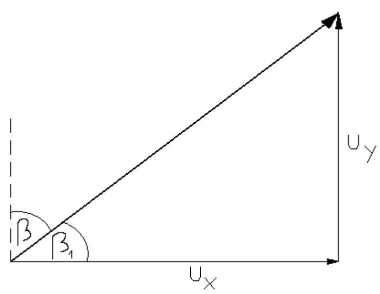
$$\cos\alpha \cdot v = v_y$$

$$v_y = -4,33\ \frac{m}{s}$$

$$u_x = 6,92\ \frac{m}{s}$$

$$u_y = 3,03\ \frac{m}{s}$$

$$\omega_n = 242,26\ \frac{rad}{s}$$



$$\tan\beta_1 = \frac{u_y}{u_x} = 0,4382$$

$$\beta_1 = 23,66^\circ$$

$$\beta = 90^\circ - 23,66^\circ = 66,34^\circ$$

Der Cricketball hüpft in einem Winkel β von 66,34° mit einer Winkelgeschwindigkeit von 242,26 rad/s (positive Rotation= Drehung gegen den Uhrzeigersinn) vom Boden ab.

Schlussfolgerungen:

- Reibung hängt nicht von der Rotation ab, weil die Reibung nur von der Normalkraft und nicht von der Geschwindigkeit abhängt.
- Die Geschwindigkeit u_x hängt nicht von der Winkelgeschwindigkeit ab, sondern nur von *e* und *μ.*
- Die Geschwindigkeit u_y hängt nur vom Elastizitätskoeffizienten ab.
  Die Winkelgeschwindigkeit ω_n ist kleiner als die Winkelgeschwindigkeit $\omega_{v,}$ da v_y negativ ist. Wenn das ω_n zu klein ist, geht der gleitende Ball in eine Rollbewegung über (siehe Rollender Ball).

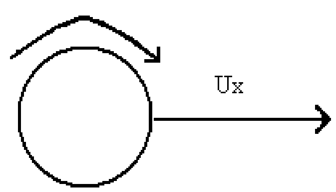
2.MODELL: DIE ROLLENDE KUGEL

Wenn ein Ball bestimmte Bedingungen erfüllt, gleitet er nicht, sondern beginnt auch zu rollen. Somit ergibt sich eine neue Formel.

Warum rollt eine Kugel?
Wenn die Kugel rollt, erfüllt sie die Rollbedingung.
Rollbedingung: Wenn Geschwindigkeit und Winkelgeschwindigkeit gering sind, reicht die Reibung zwischen Kugel und Boden aus, um die Kugel in eine Rollbewegung zu versetzen.

Mathematische Herleitung:

Rollbedingung: Zuerst müssen wir die Winkelgeschwindigkeit ω_n und die Geschwindigkeit u_x in Verbindung bringen.



$$\omega_n,\ u_x = ?$$
$$t = 1\ s$$

$$u_x = \frac{2 \cdot \pi \cdot r}{t} = \frac{2 \cdot \pi \cdot r}{1}$$

$$\omega_n = \frac{2\pi}{t} = \frac{2\pi}{1}$$

$$\omega_n = \frac{u_x}{r}$$

$$u_x = \omega_n \cdot r$$

Formeln:

- Impulsänderung M beim Stoß: $m \cdot (v_x - u_x) = M_x$
- Drehimpulsänderung: $I \cdot (\vec{\omega}_v - \vec{\omega}_n) = M_x \cdot r$
- Unelastischer Stoß: $\left|u_y\right| = \left|e \cdot v_y\right|$
- Trägheitsmoment: $I = \frac{2}{5} \cdot m \cdot r^2$
- Rollbedingung: $\omega_n = \frac{u_x}{r}$

Diese Formeln werden nun so umgeformt, dass wir mit Hilfe von den „Eingangsgrößen“(v_x, v_y, ω_v, e, und r) die „Ausgangsgrößen“ (u_x, u_y, ω_n) berechnen können. Somit können wir aufgrund des angegebenen Eintrittswinkels den Austritts-winkel und die Rotationsgeschwindigkeit des Balles vorhersagen.

Umformung:

1. Wir bauen in die umgeformte Drehimpulsänderungsformel I und M_x ein.

$$\omega_v - \omega_n = \frac{M_x \cdot r}{I}$$

$$\omega_v - \omega_n = \frac{m \cdot (v_x - u_x) \cdot r}{\frac{2}{5} \cdot m \cdot r^2}$$

$$\omega_v - \omega_n = \frac{5 \cdot (v_x - u_x)}{2 \cdot r}$$

2. Jetzt ersetzen wir ω_n.

$$\omega_v - \frac{u_x}{r} = \frac{5 \cdot (v_x - u_x)}{2 \cdot r}$$

3. Nun formen wir auf u_x um.

$$2 \cdot r \cdot \omega_v - u_x \cdot 2 = 5 \cdot (v_x - u_x)$$
$$3 \cdot u_x = 5 \cdot v_x - 2 \cdot r \cdot \omega_v$$
$$u_x = \frac{5 \cdot v_x - 2 \cdot r \cdot \omega_v}{3}$$

4. Durch Umformen erhalten wir:

$$u_x = \frac{5 \cdot v_x - 2 \cdot r \cdot \omega_v}{3}$$
$$v_y = -e \cdot u_y$$
$$\omega_n = \frac{5 \cdot v_x - 2 \cdot r \cdot \omega_v}{3r}$$

Schlussfolgerungen:

- Im Gegensatz zur gleitenden Kugel hängt die Geschwindigkeit u_x nicht mehr von µ und *e* ab, sondern zusätzlich von *r*.
- Die Geschwindigkeit u_y wird gleich definiert wie beim gleitenden Kontakt.

3. Modell: Reflexion von Gummibällen

Um Hartgummibälle zu beschreiben wurde weiteres Modell aufgestellt, das davon ausgeht, dass bei der Reflexion Energie und Impuls erhalten bleiben.

Physikalische Grundlagen:

Energieerhaltungssatz: Der Energieerhaltungssatz sagt aus, dass die Gesamtenergie eines abgeschlossenen konservativen Systems nicht verändert werden kann.

Dabei bleibt die Gesamtenergie konstant. Die Bewegungsenergie setzt sich aus zwei Komponenten zusammen: 1. der Kinetischen Translationsenergie und 2. der Rotationsenergie.

$$E = \frac{m \cdot v^2}{2} + \frac{I \cdot \omega^2}{2}$$

Impulserhaltung L: Der Impulserhaltungssatz ist einer der wichtigsten Erhaltungssätze der Physik und besagt, dass der Gesamtimpuls in einem abgeschlossenen System konstant ist. „Abgeschlossenes System“ bedeutet, dass keine Kräfte von außen auf Teile des Systems einwirken. Der Drehimpuls setzt sich aus dem Drehimpuls um den Schwerpunkt und dem Drehimpuls des Schwerpunkts um den Aufprallpunkt zusammen.

$$L = I \cdot \omega - m \cdot v \cdot r$$

Bahngeschwindigkeit s: Bahngeschwindigkeit ist die Geschwindigkeit, die ein Körper auf der Kreisbahn für einen bestimmten Weg in einer bestimmten Zeit zurücklegt. (= Geschwindigkeit der Oberfläche)

$$\omega \cdot r = s$$

Trägheitsmoment: Das Trägheitsmoment, ist eine physikalische Größe, die in der Mechanik die Trägheit eines starren Körpers gegenüber einer Änderung seiner Rotationsbewegung angibt. Das Trägheitsmoment einer vollen Kugel ist:

$$I = \frac{2}{5} \cdot m \cdot r^2$$

Mathematische Herleitung des Modells:

E_v=Energie$_{vorher}$
E_n=Energie$_{nachher}$
I_v=Impuls$_{vorher}$
I_n=Impuls$_{nachher}$

Da Energie und Impuls vor und nach dem Aufprall erhalten bleiben müssen, setzen wir $E_v = E_n$ und $I_v = I_n$ gleich.

v_v= Geschwindigkeit$_{vorher}$
v_n= Geschwindigkeit$_{nachher}$
ω_v= Winkelgeschwindigkeit$_{vorher}$
ω_n= Winkelgeschwindigkeit$_{nachher}$
s_v= Bahngeschwindigkeit$_{vorher}$
s_n= Bahngeschwindigkeit$_{nachher}$

Formeln:

- Impuls L: $L = I \cdot \omega - m \cdot v \cdot r$
- Energie E: $E = \frac{m \cdot v^2}{2} + \frac{I \cdot \omega^2}{2}$
- Trägheitsmoment I: $I = \frac{2}{5} \cdot m \cdot r^2$
- Bahngeschwindigkeit s: $\omega \cdot r = s$

Umformung:

1) *Energieerhaltung*
a) *Zuerst formen wir die Energie um und bauen I und s ein.*

$$E = \frac{m \cdot v^2}{2} + \frac{I \cdot \omega^2}{2} = \frac{m}{2} \cdot \left(v^2 + \frac{2}{5} \cdot s^2 \right)$$

b) *Jetzt setzen wir* $E_v = E_n$

$$\frac{m}{2} \cdot \left(v_v^2 + \frac{2}{5} \cdot s_v^2 \right) = \frac{m}{2} \cdot \left(v_n^2 + \frac{2}{5} \cdot s_n^2 \right)$$

$$v_v^2 + \frac{2}{5} \cdot s_v^2 = v_n^2 + \frac{2}{5} \cdot s_n^2$$

$$v_v^2 - v_n^2 = \frac{2}{5} \cdot \left(s_n^2 - s_v^2 \right)$$

$$(v_v - v_n) \cdot (v_v + v_n) = \frac{2}{5} \cdot (s_n - s_v) \cdot (s_n + s_v)$$

2) *Impulserhaltung*
a) *Zuerst formen wir den Impuls um und bauen I und s ein.*

$$L = I \cdot \omega - m \cdot v_x \cdot r = m \cdot r \left(\frac{2}{5} \cdot s - v \right)$$

b) *Jetzt setzen wir* $I_v = I_n$

$$m \cdot r \cdot \left(\frac{2}{5} \cdot s_v - v_v \right) = m \cdot r \left(\frac{2}{5} \cdot s_n - v_n \right)$$

$$\frac{2}{5} \cdot s_v - v_v = \frac{2}{5} \cdot s_n - v_n$$

$$v_n - v_v = \frac{2}{5} \cdot (s_n - s_v)$$

3) *Nun setzen wir* 1)b) *und* 2)b) *in ein Gleichungssystem ein und kürzen*

1. $(v_v - v_n) \cdot (v_v + v_n) = \frac{2}{5} \cdot (s_n - s_v) \cdot (s_n + s_v)$

2. $v_n - v_v = \frac{2}{5} \cdot (s_n - s_v)$

$$-(v_v + v_n) = s_n + s_v$$

4) *Jetzt wird v_n in* 2)b) *eingesetzt und es wird auf s_n umgeformt*

$$\left(\frac{2}{5} - 1\right) \cdot s_v - \left(\frac{2}{5} + 1\right) \cdot s_n = 2 \cdot v_v$$

$$s_n = -\frac{3}{7} \cdot s_v - \frac{10}{7} \cdot v_v$$

5) *Durch Umformen erhalten wir:*

$$s_n = -\frac{3}{7} \cdot s_v - \frac{10}{7} \cdot v_v$$

$$v_n = -\frac{4}{7} \cdot s_v + \frac{3}{7} \cdot v_v$$

$$v_{\perp n} = -v_{\perp v}$$

Mit Hilfe dieser Formel können wir nun das Sprungverhalten von Bällen ausrechnen, unter der Annahme, dass die Bewegungsenergie und der Impuls erhalten bleiben.

Folgerung:

Anhand des vorigen Beispieles haben wir gesehen, dass der Ball, im Falle der Energie- und Impulserhaltung, nach jedem zweiten Aufprall mit gleichem Spin und gleichem Winkel wegspringt.

Beweis:

1. Zuerst für setzten wir für s_v= a und für v_v=b:

$$v_n = -\frac{4}{7} \cdot a + \frac{3}{7} \cdot b$$

$$s_n = -\frac{3}{7} \cdot a - \frac{10}{7} \cdot b$$

2. Jetzt wird v_n und s_n in die Formel eingesetzt uns ausgerechnet.

$$v_n = -\frac{4}{7} \cdot \left(-\frac{3}{7} \cdot a - \frac{10}{7} \cdot b\right) + \frac{3}{7} \cdot \left(-\frac{4}{7} \cdot a + \frac{3}{7} \cdot b\right)$$

$$v_n = \frac{12}{49} \cdot a + \frac{40}{49} \cdot b - \frac{12}{49} \cdot a + \frac{9}{49} \cdot b$$

$$v_n = \frac{49}{49} \cdot b = b$$

3. Nun machen wir das gleiche wie in 2., nur dass wir in s_n einsetzen.

$$s_n = -\frac{3}{7} \cdot \left(-\frac{3}{7} \cdot a - \frac{10}{7} \cdot b\right) - \frac{10}{7} \cdot \left(-\frac{4}{7} \cdot a + \frac{3}{7} \cdot b\right)$$
$$s_n = \frac{9}{49} \cdot a + \frac{30}{49} \cdot b + \frac{40}{49} \cdot a - \frac{30}{49} \cdot b$$
$$s_n = \frac{9}{49} \cdot a + \frac{40}{49} \cdot a$$
$$s_n = \frac{9}{49} \cdot a + \frac{40}{49} \cdot a = a$$

4. Jetzt setzen wir die Werte wieder in die Formel ein und können sehen, dass wir wieder das Ursprüngliche herausbekommen.

$$v_n = b$$
$$s_n = a$$
$$v_{yn} = -c$$
$$v_n = -\frac{4}{7} \cdot a + \frac{3}{7} \cdot b$$
$$s_n = -\frac{3}{7} \cdot a - \frac{10}{7} \cdot b$$

Somit ist also bewiesen, dass diese Formel auf jede(n) Ball/Kugel anwendbar ist.

MATRIZEN

Matrizen werden unter anderem dazu benutzt, lineare Gleichungssysteme zu beschreiben. Matrizen stellen Zusammenhänge, in denen insbesondere Linear-kombinationen eine Rolle spielen, übersichtlich dar und erleichtern damit Rechen- und Gedankenvorgänge. Eine Matrix besteht aus einer öffnenden und einer schließenden Klammer, Spalten und Zeilen. Die Spalten bzw. Zeilen bilden Spalten- bzw. Zeilenvektoren. Die Elemente, die in einer Matrix angeordnet sind, nennt man Einträge oder Komponenten der Matrix.

Anwendung einer Matrix auf unser Beispiel:

Wir haben ein Gleichungssystem bestehend aus drei Gleichungen:

$$v_{x_n} = -\left(\frac{4}{7}\right) \cdot s_v + \left(\frac{3}{7}\right) \cdot v_{x_v}$$
$$s_n = -\left(\frac{3}{7}\right) \cdot s_v - \left(\frac{10}{7}\right) \cdot v_v$$
$$v_{y_n} = -v_{y_v}$$

Dieses Gleichungssystem kann man mit Hilfe folgender Matrix beschreiben:

$$\begin{pmatrix} \frac{3}{7} & -\frac{4}{7} & 0 \\ -\frac{10}{7} & -\frac{3}{7} & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

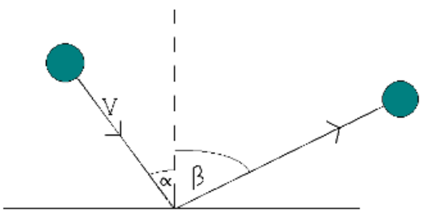
Diese Matrix erhält man, indem man die entsprechenden Koeffizienten der Gleichung in die Matrix einfügt. Größen, die in der Gleichung nicht vorkommen, besitzen in der Matrix den Koeffizienten 0.

Die erhaltene Matrix nennen wir „Stoßmatrix“, da sie das Gleichungssystem eines Stoßes beschreibt.

Multipliziert man den Ausgangsvektor (verallgemeinerte Koordinaten: Geschwindigkeit in x-Richtung, Bahngeschwindigkeit, Normalgeschwindigkeit) mit der Stoßmatrix, erhält man einen neuen Vektor, der die neuen Eigenschaften des Balles in Bezug auf Geschwindigkeit, Drehsinn und Winkel angibt:

$$\begin{pmatrix} v_{x_n} \\ s_n \\ v_{y_n} \end{pmatrix} = \begin{pmatrix} \frac{3}{7} & -\frac{4}{7} & 0 \\ -\frac{10}{7} & -\frac{3}{7} & 0 \\ 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} v_{x_v} \\ s_v \\ v_{y_v} \end{pmatrix}$$

Konkret haben wir dieses Beispiel auf die Reflexion eines Gummiballes angewandt. Wir konnten dadurch zum Beispiel die Geschwindigkeitsänderung, den Drehsinn und auch den Reflexionswinkel des Balles nach dem Aufprall auf dem Boden berechnen.



Will man den Ball anschließend auch noch an der Wand abprallen lassen, so müssen beim neuen Vektor die Geschwindigkeitskoordinaten v_x und v_y vertauscht werden (das Koordinatensystem wird um 90° gedreht) und die neue v_y-Koordinate wird negativ gesetzt, da diese Geschwindigkeit in die negative y-Richtung gerichtet ist. Eine zweite Möglichkeit wäre die Multiplikation der Stoßmatrix mit einer weiteren Matrix, die die Drehung der Koordinaten beschreibt:

$$\begin{pmatrix} \frac{3}{7} & -\frac{4}{7} & 0 \\ -\frac{10}{7} & -\frac{3}{7} & 0 \\ 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\frac{4}{7} & \frac{3}{7} \\ 0 & -\frac{3}{7} & -\frac{10}{7} \\ 1 & 0 & 0 \end{pmatrix}$$

Hat man diese neue Stoßmatrix erstellt, kann man mit dem ursprünglichen Vektor rechnen, ohne diesen zu verändern.

Beispiel 1:

Matrix: $\begin{pmatrix} \frac{3}{7} & -\frac{4}{7} & 0 \\ -\frac{10}{7} & -\frac{3}{7} & 0 \\ 0 & 0 & -1 \end{pmatrix}$

Anfangsbedingungen eines Gummiballes: $\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$ …Geschwindigkeit in x-Richtung
…Bahngeschwindigkeit
…Geschwindigkeit in y-Richtung

Anfangszustand: Aufprallwinkel = 45°
Geschwindigkeit = $\sqrt{2}$ m/s
Bahngeschwindigkeit = 0 m/s

Reflexion am Boden:

Vektor multipliziert mit Matrix: $\begin{pmatrix} 0.4286 \\ -1.4286 \\ 1 \end{pmatrix}$

Eigenschaften des neuen Vektors: Abprallwinkel = 66,8°
Geschwindigkeit = 1,088 m/s
Bahngeschwindigkeit = -1,4286 m/s
(im Uhrzeigersinn!)

Reflexion an der Wand:

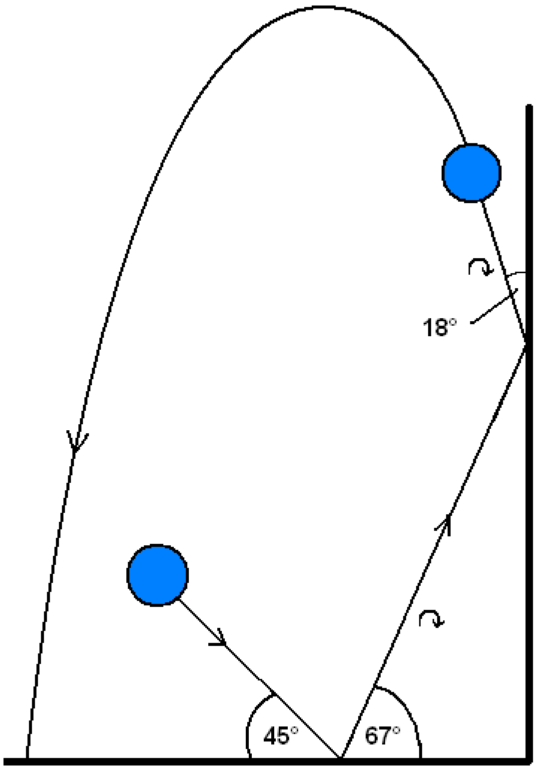
Matrix zur Berechnung der Reflexion an der Wand: $\begin{pmatrix} 0 & -0.5714 & 0.4286 \\ 0 & -0.4286 & -1.4286 \\ 1 & 0 & 0 \end{pmatrix}$

Neuer Vektor multipliziert mit neuer Matrix: $\begin{pmatrix} -0.4286 \\ -0.8163 \\ 1.2449 \end{pmatrix}$

Eigenschaften des von der Wand reflektierten Vektors:

Abprallwinkel = 18,64°
Geschwindigkeit = 1,314 m/s
Bahngeschwindigkeit = -0,8163 m/s
(im Uhrzeigersinn!)

Skizze:



Beispiel 2:

Reflexion des Tennisballes am Boden:

Da der Tennisball eine Hohlkugel ist, ist das Trägheitsmoment größer.

Für eine Hohlkugel gilt: Trägheitsmoment: $I = \frac{2}{3} \cdot mr^2$

Dadurch erhalten wir das Gleichungssystem:

$$v_{x_n} = v_{x_v} + v_{y_v} \cdot \mu \cdot (1 + e)$$

$$\omega_n = \omega_v + \frac{5 \cdot \mu \cdot v_{y_v} \cdot (1 + e)}{2r}$$

$$v_{y_n} = -e \cdot v_{y_v}$$

Für den Tennisball (r = 3,25 cm;   μ= 0,5; e = 0,75) erhält man demnach folgende Matrix:

$$\begin{pmatrix} v_{x_n} \\ \omega_n \\ v_{y_n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0.875 \\ 0 & 1 & 40.32 \\ 0 & 0 & -0.75 \end{pmatrix} \cdot \begin{pmatrix} v_{x_v} \\ \omega_v \\ v_{y_v} \end{pmatrix}$$

Nehmen wir zum Beispiel folgenden Anfangsvektor $\begin{pmatrix} 10 \\ 100 \\ -7 \end{pmatrix}$

Eigenschaften des Vektors:

Aufprallwinkel = 35°
Geschwindigkeit = 12,20 m/s
Bahngeschwindigkeit = 32,5 m/s

Reflexion am Boden:

Vektor multipliziert mit Matrix: $\begin{pmatrix} 3.875 \\ -182.24 \\ 5.25 \end{pmatrix}$

Eigenschaften des neuen Vektors:

Abprallwinkel = 49,8°
Geschwindigkeit = 6,5 m/s
Bahngeschwindigkeit = -59.228 m/s

4. Modell: Reflexion von Gummibällen mit Energieverlust

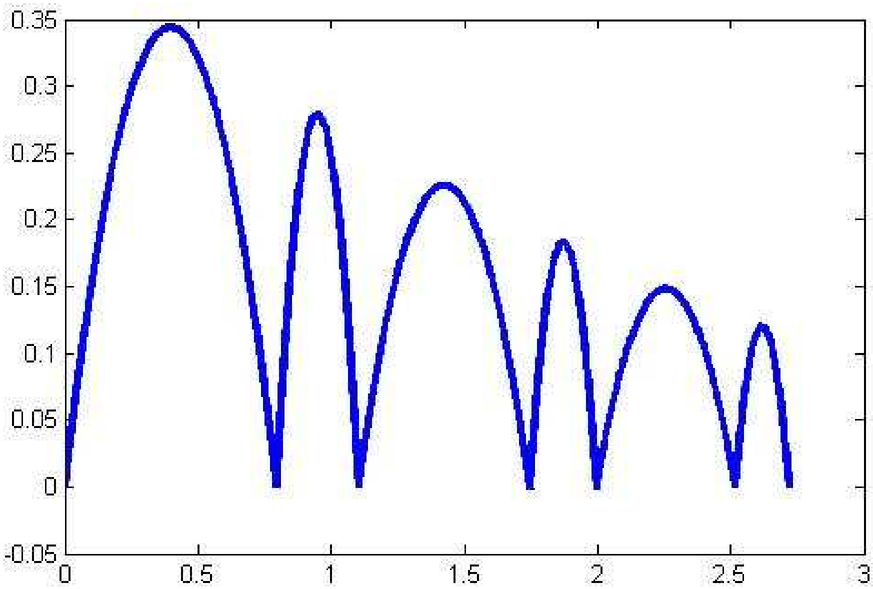
In diesem Modell fügen wir zum Modell 3 noch den Elastizitätskoeffizienten hinzu, um den Energieverlust zu berücksichtigen. Dadurch verändert sich die Stoßmatrix nur gering: die senkrechte Geschwindigkeit nach dem Stoß ist, wie im ersten Modell, Elastizitätskoeffizient mal Geschwindigkeit vor dem Stoß.

Matlab

Mittels des Mathematikprogramms MATLAB konnte ein Programm erstellt werden, mit welchem sich Bewegungsabläufe von Bällen einfach darstellen lassen. Da sich die Wurfparabeln allerdings nur als bijektive Funktionen darstellen lassen, können Bewegungen in negative x-Richtung nicht dargestellt werden. Für alle Bewegungen eines Balles, die sich nur in positive x-Richtung ausbreiten, kann dieses Programm den Bewegungsablauf graphisch darstellen. Es werden grundsätzlich drei Fälle unterschieden:

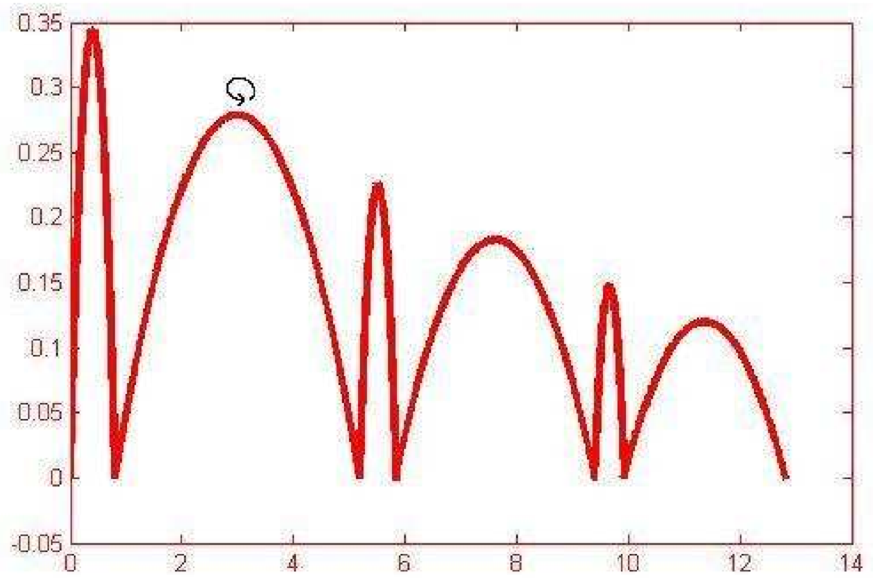
1.) *Der Ball besitzt keine Drehung beim Abwurf:*

Daten: Abwurfwinkel: 60°
Drehgeschw.: 0 m/s
vAnfang = 3 m/s
e= 0.9



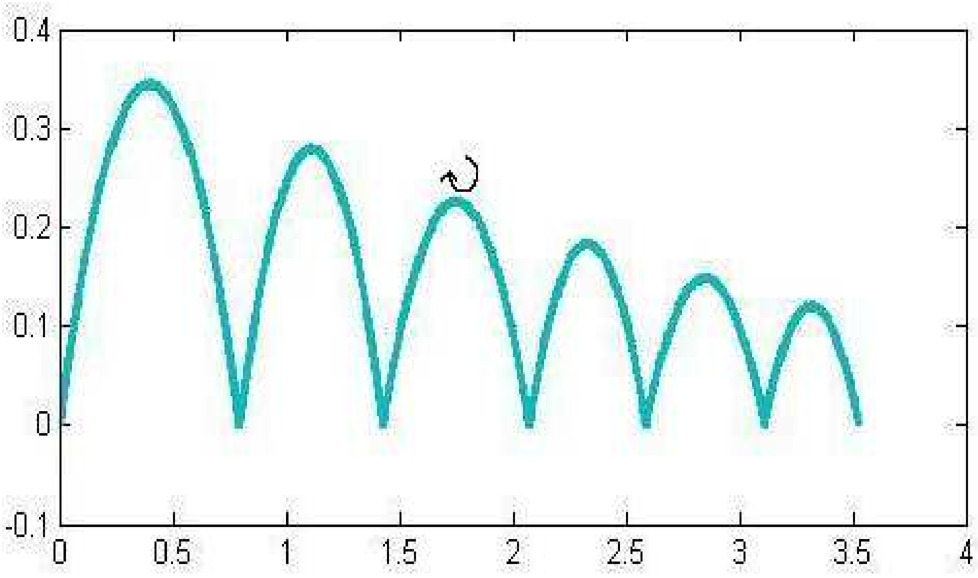
2.) Der Ball hat einen negativen Drehsinn beim Abwurf

Daten: Abwurfwinkel: 60°
Drehgeschw.: -1 m/s
vAnfang = 3 m/s
e= 0.9



3.) Der Ball hat einen positiven Drehsinn beim Abwurf

Daten: Abwurfwinkel: 60°
Drehgeschw.: 1 m/s
vAnfang = 3 m/s
e= 0.9



Wie man sieht, spielen der Drehsinn und die Drehgeschwindigkeit eine wesentliche Rolle beim Bewegungsablauf eines Balles. Auffallend dabei ist, dass es immer einen Zweier-Zyklus gibt. Der Ball beschreibt beispielsweise bei Fall 1 zuerst eine breitere dann eine dünnere Parabel. Das wiederholt dich solange, bis der Ball ins Rollen kommt. Auch in Fall 2 tritt der gleiche Effekt auf, nur dass der Unterschied zwischen den Parabelbreiten etwas größer ist. Wird bei diesem Fall die Drehgeschwindigkeit erhöht, so tritt das Phänomen auf, dass sich der Ball in entgegen gesetzte Richtung bewegt. Fall 3 beschreibt einen Spezialfall, denn durch leichte Drehung im Uhrzeigersinn können die Bewegungsparabeln die gleiche Breite erreichen.

Die Veränderung der Parabeln lässt sich vereinfacht so erklären, dass ein Ball für die Rotation Energie benötigt und diese Energie somit der Bewegung in die x-Richtung fehlt, die Parabelweite wird somit also kleiner.

Schloss Seggau, 18.1.2008

Geometrische Optimierung: Endbericht

Aufgabenstellung:

Die Minimierung von Herstellungskosten ist ein zentrales Problem in der Wirtschaft.

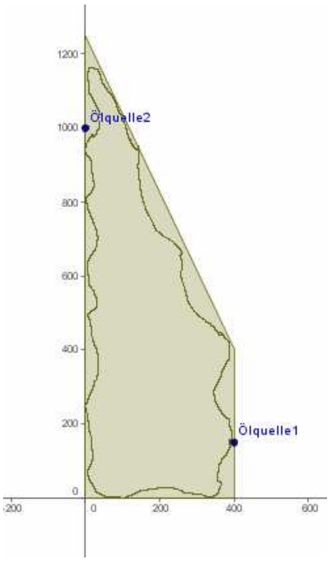
Bei der Verlegung von Rohrleitungen durch verschiedene Geländearten, wie z.B. Trockenland und Sumpfgebiet, sind die Herstellungskosten pro Meter im Trockengebiet geringer als pro Meter im Sumpfgebiet, wo man eigene Maschinen benötigt.

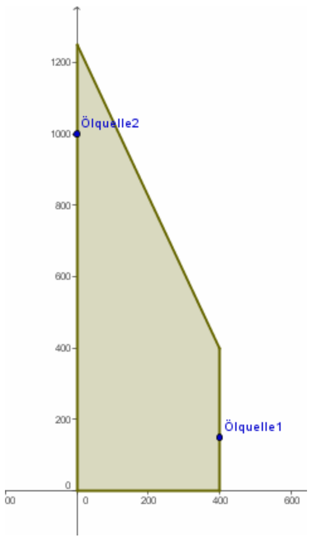
Ist in einer Landkarte ein Sumpfgebiet abgegrenzt gegen das Trockengebiet dargestellt und sind *A* und *B* zwei Punkte im Trockengebiet, so erhebt sich die Frage nach der billigsten Rohrverbindung von *A* nach *B*.

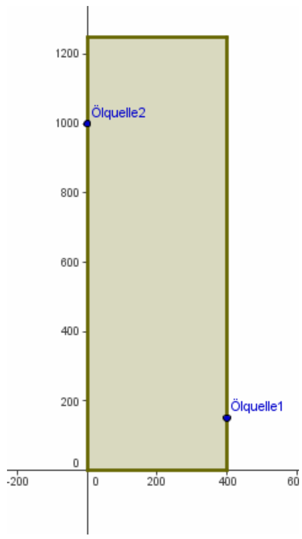
Die Rohrleitung soll dabei aus geraden Rohrstücken mit einer vorgegebenen Höchstzahl von dazwischenliegenden Rohrknien (beliebiger Winkel) bestehen. Bereits die Bestimmung kostenminimaler Rohrleitungen mit höchstens einem Knie bietet bei den allereinfachsten Sumpfgebieten mit Rechtecks- oder Trapezgestalt überraschende Schwierigkeiten. Die schrittweise Entdeckung, Formulierung und Überwindung dieser Schwierigkeiten ist das Hauptanliegen der Aufgabenstellung zu diesem Thema.

Die optimale Lage eines Knies an einem Rechteck

Um dieses Problem zu lösen, mussten wir zuallererst die komplizierte Form eines realen Sumpfes vereinfachen.



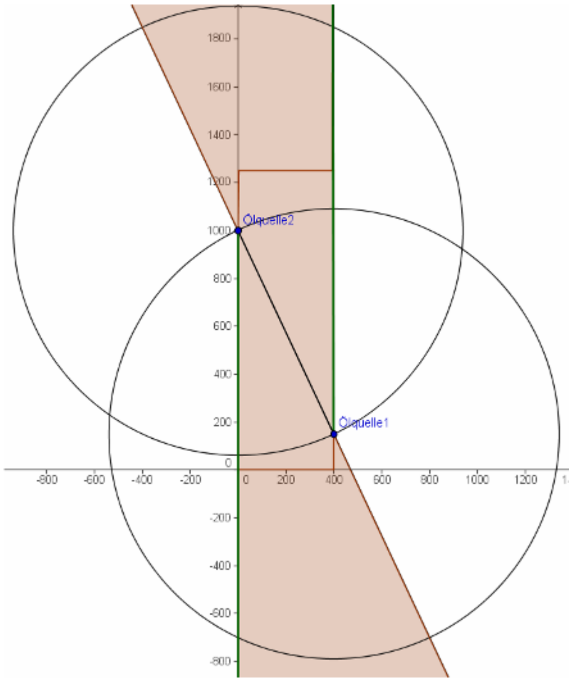




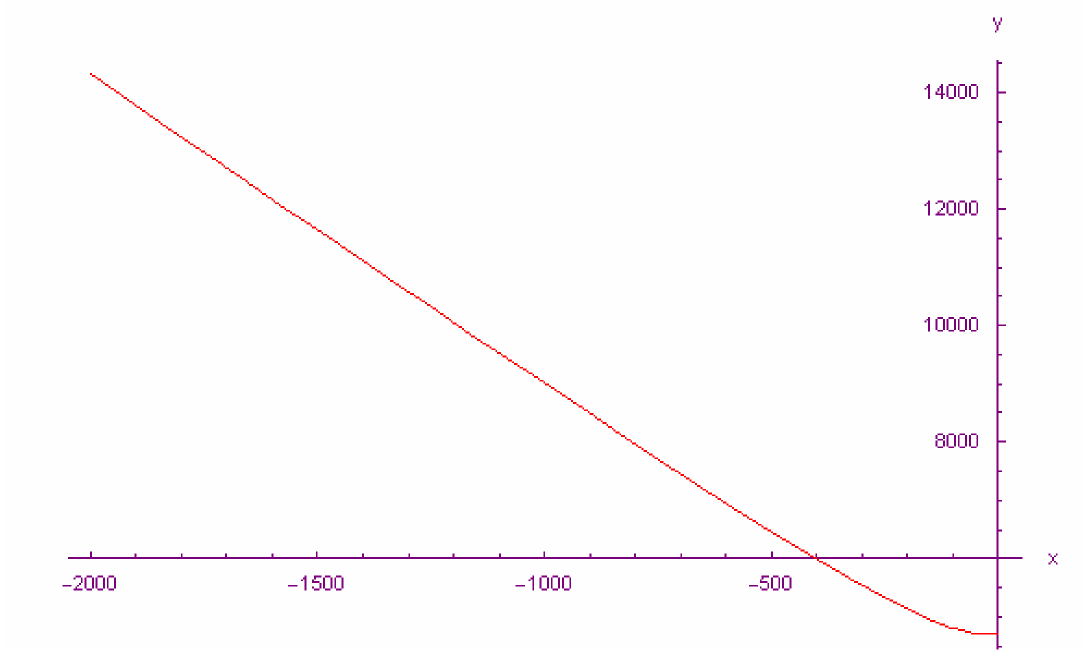
Die Höhe unseres idealisierten Sumpfes war 1250, die Breite 400 und die Ölquellen befanden sich an den Punkten *A* (400, 150) und *B* (0, 1000). Wir einigten uns darauf, die Ränder des Sumpfes zum Trockengebiet zu zählen und definierten für den Trockenpreis t=2,7$/m und für den Sumpfpreis s=4,7$/m.

Zuerst überlegten wir, welche Gebiete wir für die Lage des günstigsten Knies von vornherein ausschließen könnten. Diese sind:

- Der Sumpf selbst; liegt der Knick in ihm, kann die daraus entstehende Strecke niemals kürzer sein als die direkte Verbindung der beiden Punkte *(in der Graphik rot markiert)*
- Alle Richtungen, in denen die Sumpfstrecke zum Knick länger ist als die direkte Verbindung der Punkte *(ebenfalls rot markiert)*
- Der ganze verbleibende weiße Bereich bis auf die Geraden *x=0* und *x=400*, weil die Trockenstrecke dann unnötig lange ist.

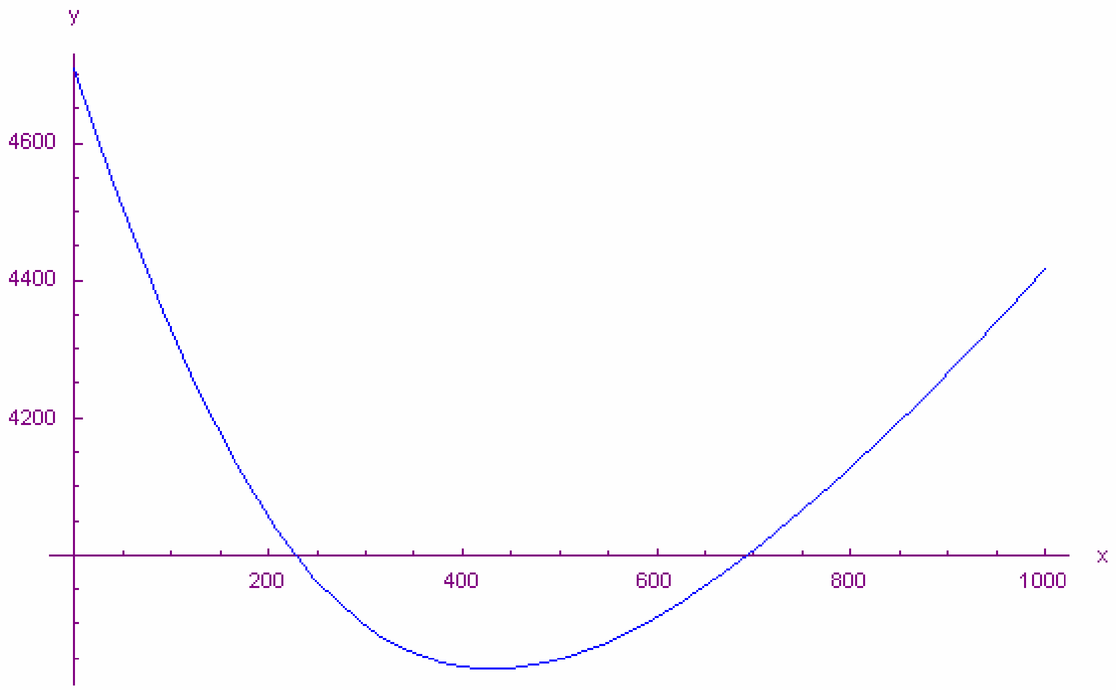


Also blieben uns nur mehr die hier grün dargestellten Strecken für die Lage des optimalen Knickes. Dabei beschränkten wir uns auf die Gerade *x=0* und stellten vier Funktionen auf, die diese beschreiben.



Kostenfunktion 1: $y \leq 0$

$$f1[t\_, y\_] = t * \left((1000 - y) - y * \sqrt{400^2 + (150 - y)^2} \Big/ (150 - y)\right) + 4.7 * \left(\sqrt{(150 - y)^2 + 400^2} + y * \sqrt{(150 - y)^2 + 400^2} \Big/ (150 - y)\right);$$

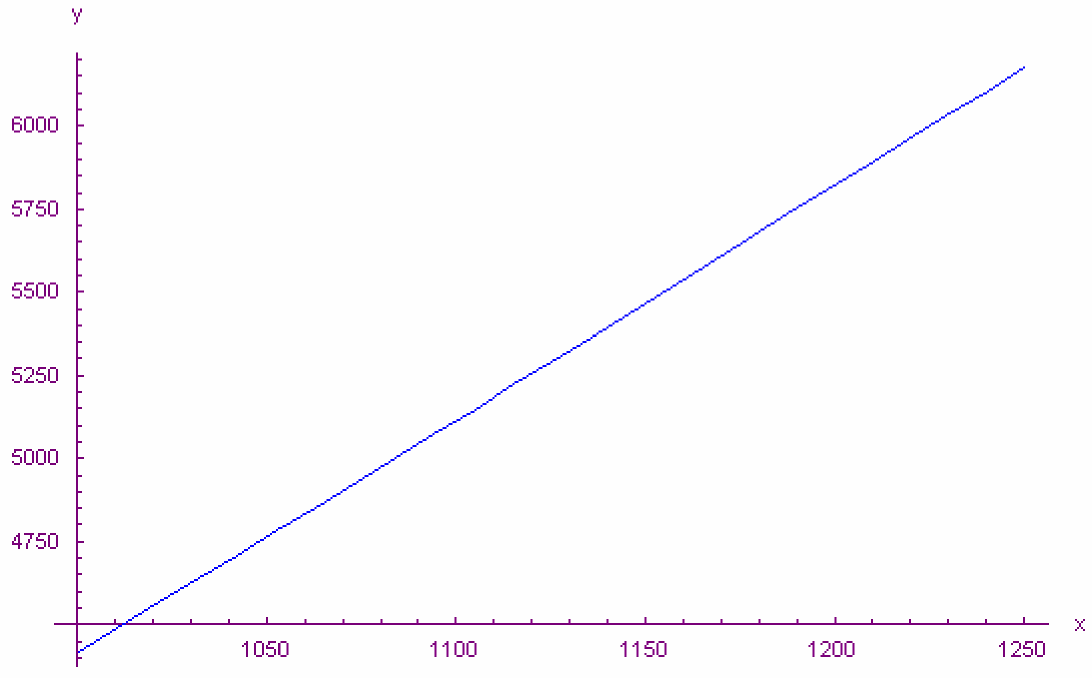


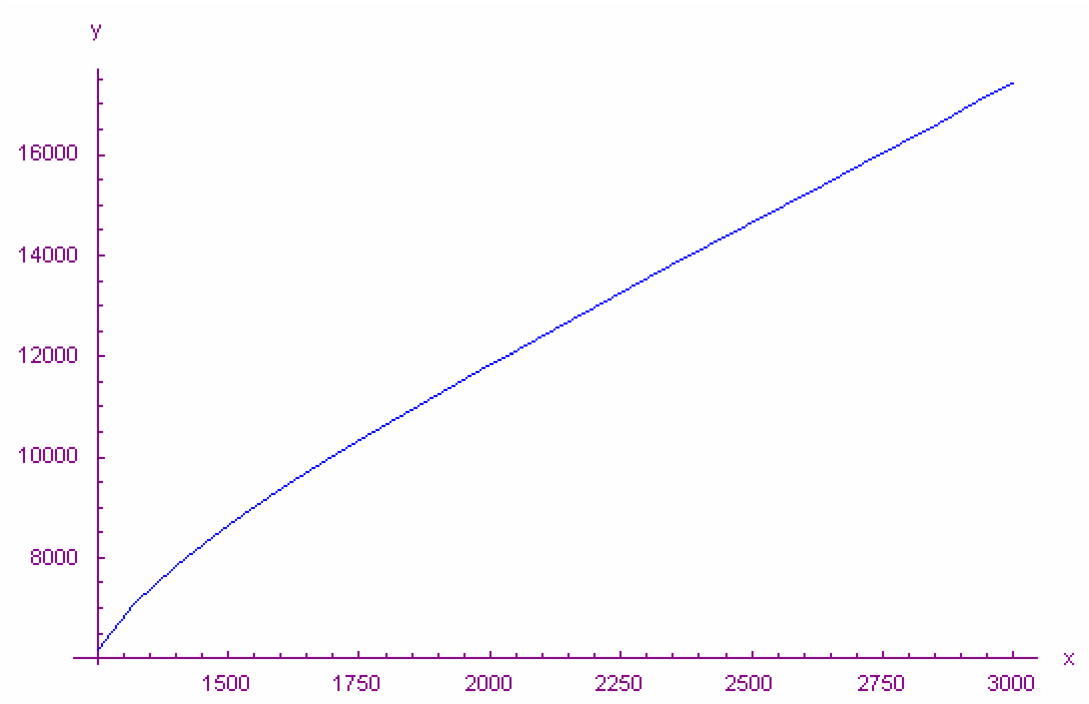
Kostenfunktion 2: $0 \leq y \leq 1000$

```
f2[t_, y_] = t * (1000 - y) + 4.7 * Sqrt[400^2 + (150 - y)^2];
```

Kostenfunktion 3: $1000 \leq y \leq 1250$ Sie ist nicht, wie es auf den ersten Blick scheint, linear!

```
f3[t_, y_] = t * (y - 1000) + 4.7 * Sqrt[400^2 + (150 - y)^2];
```

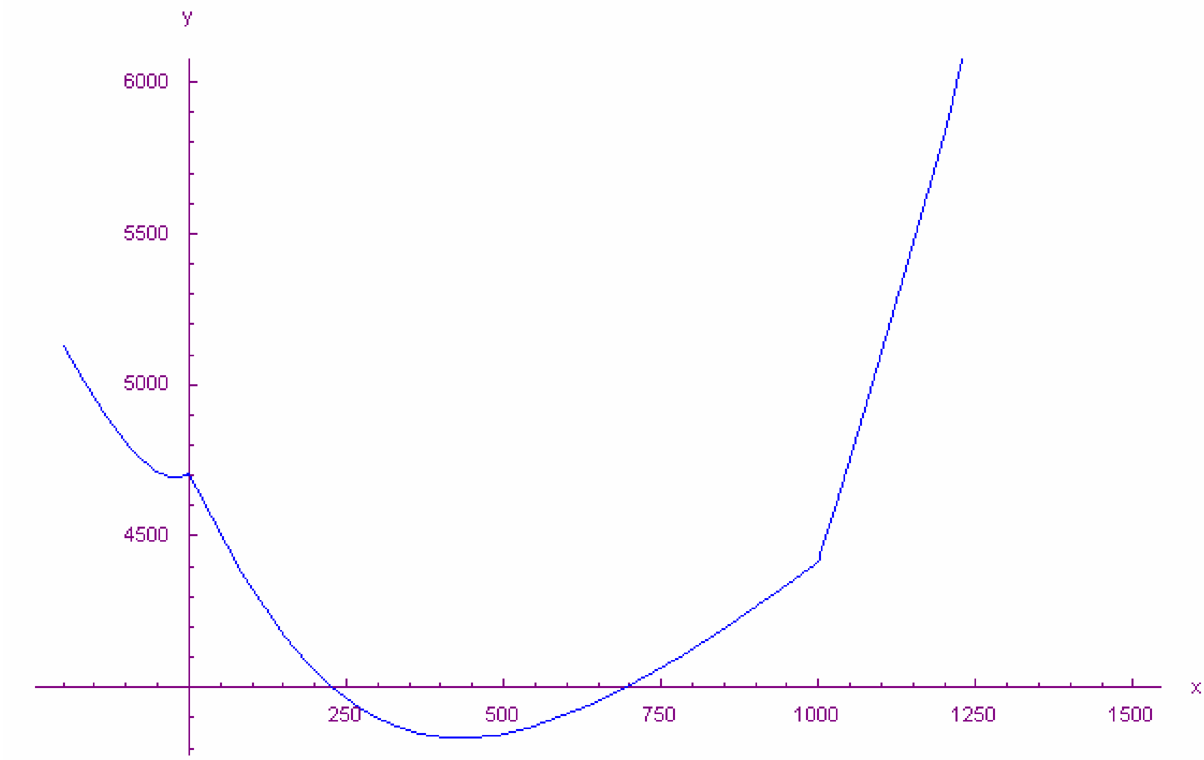


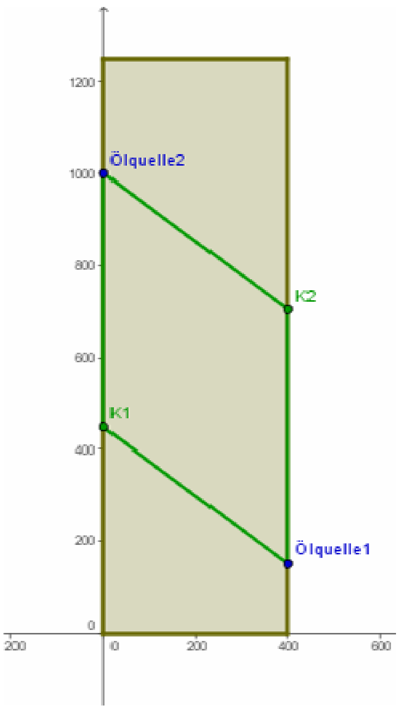


Kostenfunktion 4: $y \geq 1250$

$$f4[t\_, y\_] = t * \left((y - 1000) + (y - 1250) * \sqrt{400^2 + (y - 150)^2} \Big/ (y - 1000)\right) + 4.7 * \left(\sqrt{(y - 150)^2 + 400^2} - (y - 1250) * \sqrt{(y - 150)^2 + 400^2} \Big/ (y - 150)\right);$$

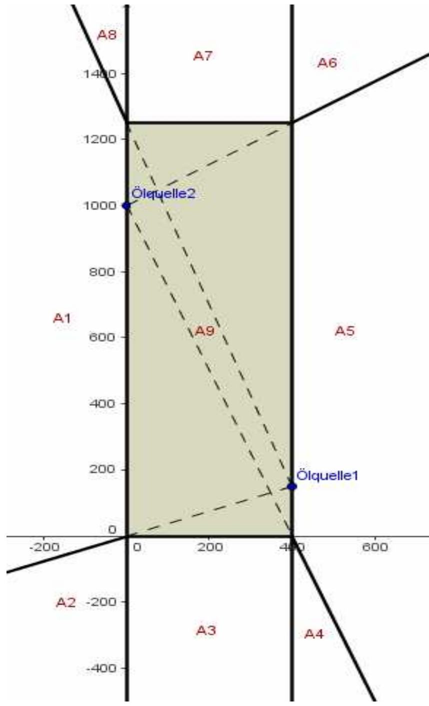
Aus diesen vier Stücken erstellten wir dann folgende Funktion:





Nun mussten wir nur noch die erste Ableitung der Kostenfunktion 2 bilden. Damit errechneten wir den optimalen Punkt *P* (0/ 431) mit den Kosten von 3834$. Allerdings gibt es natürlich eine zweite Möglichkeit, die in folgender Zeichnung dargestellt ist.

So weit, so gut. Im Folgenden wollten wir aber nicht nur die Kosten für diese beiden Geraden berechnen, sondern auch den Preis für jede mögliche Lage des Knies. Dabei wurde es notwendig, weitere Formeln zu finden, die jetzt aber 2 Unbekannte enthielten.



Außerdem mussten wir die Definitionsbereiche für jede Formel festlegen und teilten somit die Fläche in neun verschiedene Bereiche, wie hier rechts zu sehen ist.

Das Aufstellen der Formeln verkomplizierten wir selbst, indem wir erst versuchten, sie mit Hilfe des Satzes von Pythagoras und der Vektorrechnung aufzustellen, wo es doch mit dem Strahlensatz sehr viel einfacher gewesen wäre. Schließlich gelangten wir zu folgendem Ergebnis:

$$\text{lAK}[x\_, y\_] = \sqrt{(400-x)^2+(y-150)^2};$$

$$\text{lBK}[x\_, y\_] = \sqrt{x^2+(y-1000)^2};$$

$$\text{kf1}[x\_, y\_] := 2.7*\left(\sqrt{x^2+(y-1000)^2}+\sqrt{x^2+(y-150+400*(y-150)/(x-400))^2}\right)+4.7*\sqrt{400^2+(400*(y-150)/(x-400))^2};$$

$$\text{kf2}[x\_, y\_] := 2.7*\left(\sqrt{x^2+(y-1000)^2}+\sqrt{((x-400-(x-400)*(-150)/(y-150))^2+y^2)}\right)+4.7*\sqrt{((x-400)*150/(y-150))^2+150^2};$$

$$\text{kf3}[x\_, y\_] := 4.7*\left(\sqrt{((400-x)*(-150)/(150-y))^2+150^2}+\sqrt{(x*(-1000)/(y-1000))^2+1000^2}\right)+$$
$$2.7*\left(\sqrt{((400-x)*(-150)/(150-y)+400-x)^2+y^2}+\sqrt{(x*(-1000)/(y-1000)-x)^2+y^2}\right);$$

$$\text{kf4}[x\_, y\_] := 2.7*(\text{lAK}[x, y]+(-y)/(1000-y)*\text{lBK}[x, y])+4.7*(1000/(1000-y)*\text{lBK}[x, y]);$$

$$\text{kf5}[x\_, y\_] := 2.7*\left(\sqrt{(x-400)^2+(y-150)^2}+\sqrt{(400-x)^2+(1000+400*(y-1000)/(x)-y)^2}\right)+4.7*\sqrt{400^2+(400*(y-1000)/x)^2};$$

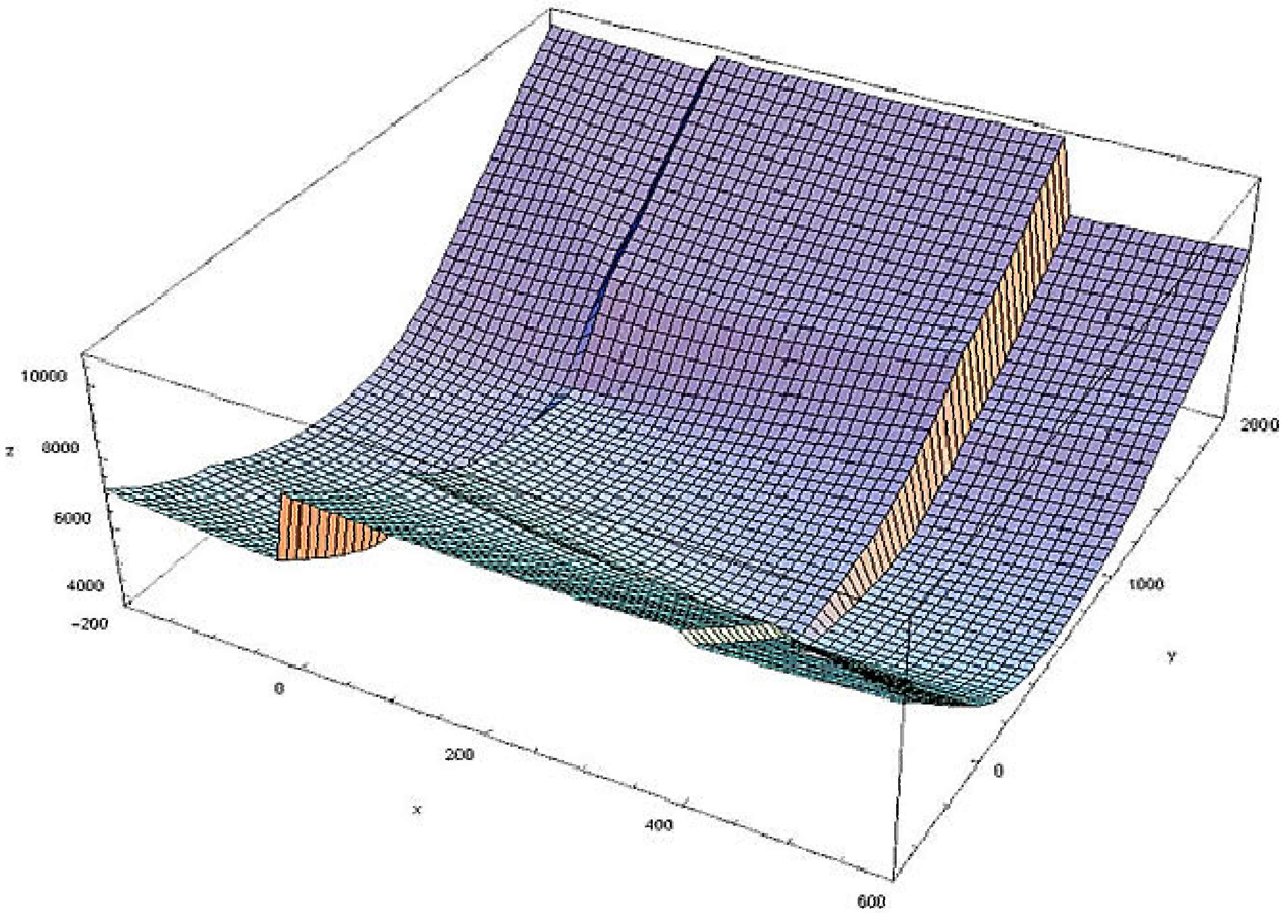
$$\text{kf6}[x\_, y\_] := 2.7*(\text{lAK}[x, y]+(y-1250)/(y-1000)*\text{lBK}[x, y])+4.7*(250/(y-1000)*\text{lBK}[x, y])$$

$$\text{kf7}[x\_, y\_] := 2.7*((y-1250)*\text{lBK}[x, y]/(y-1000)+(y-1250)*\text{lAK}[x, y]/(y-150))+$$
$$4.7*((\text{lBK}[x, y]*250/(y-1000))+1100*\text{lAK}[x, y]/(y-150));$$

$$\text{kf8}[x\_, y\_] := 2.7*(\text{lBK}[x, y]+(y-1250)/(y-150)*\text{lAK}[x, y])+4.7*(1100/(y-150)*\text{lAK}[x, y]);$$

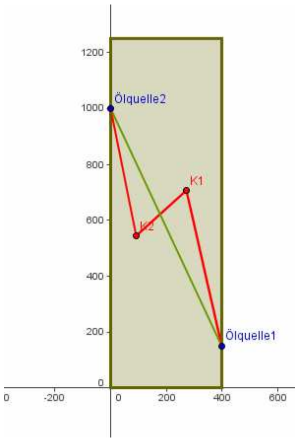
$$\text{kf9}[x\_, y\_] := 4.7*\left(\sqrt{(400-x)^2+(150-y)^2}+\sqrt{x^2+(y-1000)^2}\right);$$

Verständlicherweise ist es sehr schwer, sich solche Formeln bildlich vorzustellen, deswegen versuchten wir mit dem Programm Mathematica einen dreidimensionalen Graphen zu erstellen, der diese Funktionen veranschaulichen sollte. Außerdem sollte dieser Graph uns selbst als Kontrolle für unsere Funktionen dienen.



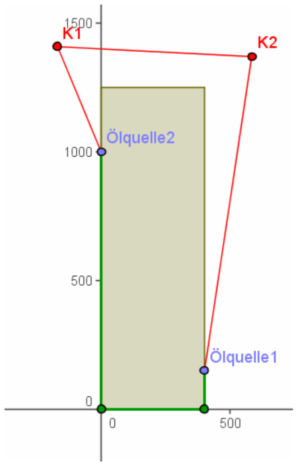
Lage von zwei Knicken an einem Rechteck

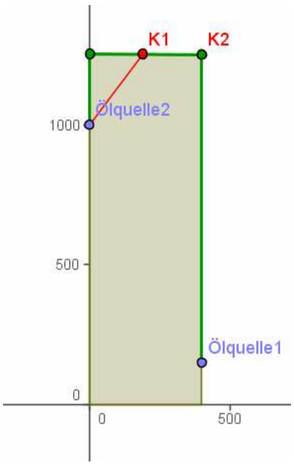
Durch die Verwendung von zwei Knicken eröffnen sich zahlreiche neue Optionen, unter anderem solche, die den Sumpf komplett umgehen. Die meisten dieser Optionen sind allerdings sehr einfach auszuschließen, wie in folgenden Graphiken dargestellt ist.



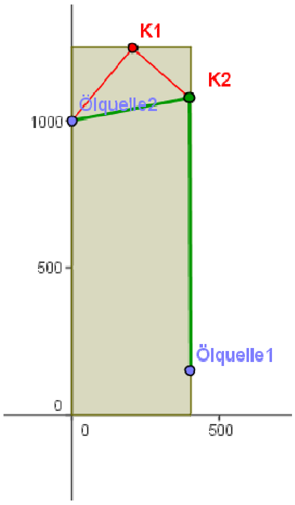
Links: Liegen beide Knie im Sumpf, kann die Strecke nicht kürzer sein als die direkte Verbindung.

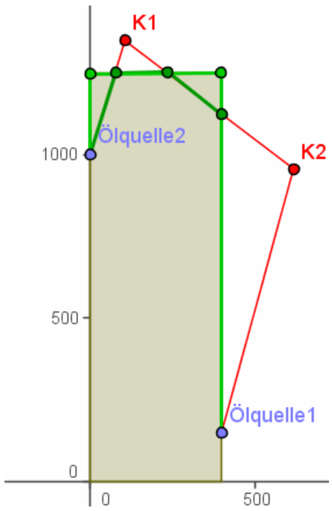
Rechts: Die Strecke vom Eintritt bis zum Austritt der Pipeline in den Sumpf (hier die beiden Ölquellen) kann immer durch die direkte Verbindung abgekürzt werden





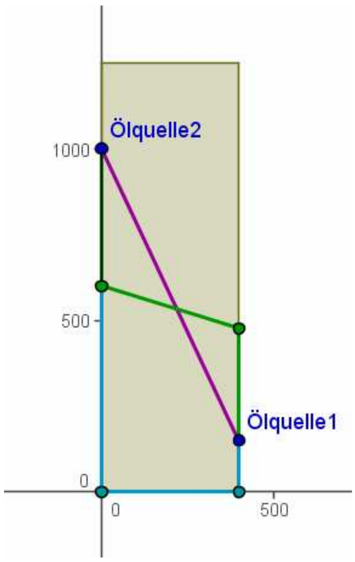
Links: Das Streckenverhältnis Abkürzung: Umgehung ist bestenfalls (beide Winkel betragen 45°) 0,707, das Preisverhältnis aber nur 0,574. Aber wenn sich eine Abkürzung lohnen würde, müsste sie so groß wie möglich sein (**rechts**), dann könnte man sie aber wieder durch eine direkte Verbindung verkürzen.

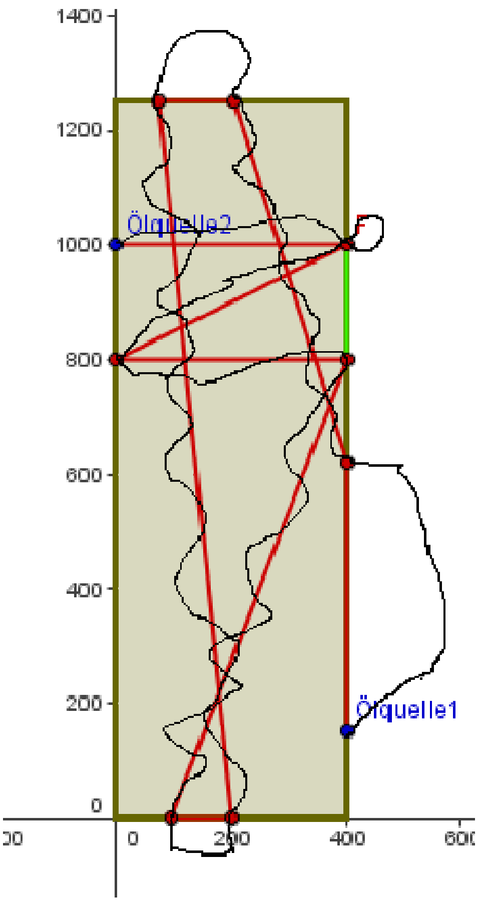




Links: Bei diesem Sonderfall muss man ein drittes Knie annehmen, um die Trockenstrecke abzukürzen.
Die Sumpfstrecken können allerdings – wie eben beschrieben – durch eine Umgehung ersetzt werden, womit das dritte Knie wieder unnötig wird.

Folglich verbleiben nur mehr wenige Lösungen, die abhängig vom Preisverhältnis, unterschiedliche Formen annehmen – hier **rechts** abgebildet. Anzumerken ist, dass zu der grünen Version unendlich viele Parallele existieren.





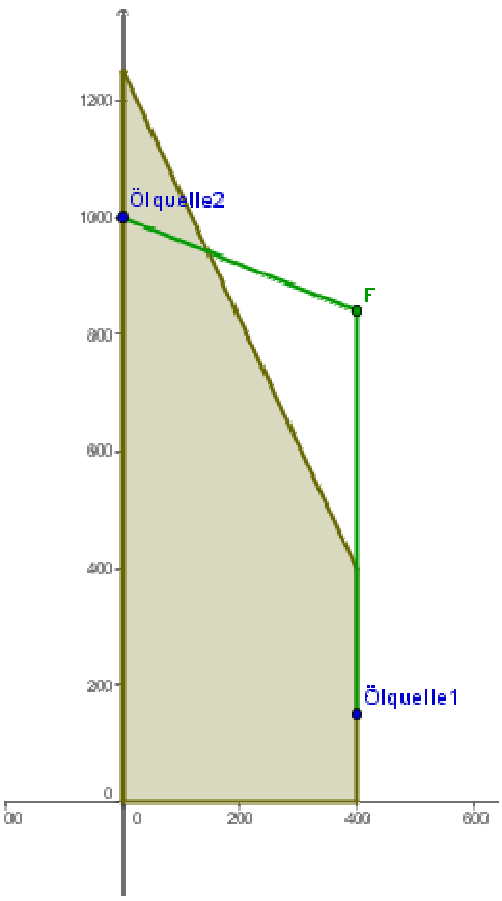
Lage von beliebig vielen Knien in einem Rechteck

Wenn man von beliebig vielen Knien ausgeht, ist es am einfachsten, nicht mehr mit geraden Verbindungslinien und Knicken zu denken, sondern eine Strecke mit beliebigen Biegungen und Kurven anzunehmen (Schwarze Linie). Auf den ersten Blick ist es klar, dass eine solche Strecke keine optimale Lösung ist. (siehe Grafik) Um zu beweißen, dass es nicht sinnvoll ist mehr als 2 Knie zu verwenden, haben wir zunächst alle Ein- und Austrittspunkte direkt miteinander verbunden. (Rote Linie). Hierbei treten drei verschiedene Fälle auf:

1. Die Punkte liegen auf der gleichen Seite
2. Die Punkte liegen auf zwei aneinander grenzenden Seiten
3. Die Punkte liegen auf gegenüberliegenden Seiten

Im ersten Fall läuft die Verbindungsstrecke entlang der Seite des Rechtecks. Die billigste Verbindung entlang des Randes kann aber bereits mit zwei Knien verwirklicht werden. Im zweiten Fall wird ein Dreieck aus dem Rechteck ausgeschnitten; dass das nicht sinnvoll ist wurde bereits weiter oben erklärt. Im dritten Fall durchquert die Verbindung entweder die Länge oder die Breite des Rechtecks. Jede Strecke, die längs durch den Sumpf läuft, kann ausgeschlossen werden, da sie bereits teurer ist als die direkte Verbindung der beiden Punkte. Die Durchquerung der Breite des Rechtecks kann auch höchstens einmal sinnvoll sein, da man sonst wieder zu der Ausgangsseite zurückkommt und somit eine billigere Verbindung entlang der Kante des Rechtecks wählen könnte. (*Grüne Linie*) Eine Strecke, die den Sumpf nur einmal der Breite nach durchläuft, kann auch mit zwei Knien verwirklicht werden.

Das Trapez



Danach wollten wir die optimale Lage eines Knies in einem Trapez berechnen, das ja der wahren Form des Sumpfes schon deutlich näher kommt als das Rechteck. Wir nahmen dabei für den neuen Eckpunkt die Koordinaten (400, 400) an. Ein sehr günstiger Wert, wie sich später herausstellen sollte…

Bei der Berechnung dieses optimalen Knies kamen uns die Erkenntnisse zu gute, die wir aus dem Rechteck gewonnen hatten. So wussten wir nun sofort den Bereich einzugrenzen und die Funktion für die Gerade $x = 400$ aufzustellen. Diese bestand wie beim Rechteck aus drei Formeln und es ergab sich das Problem, dass offensichtlich weder der Taschenrechner noch der Computer die Nullstelle der ersten Ableitung finden konnten. Warum wir keine Lust hatten, es auf herkömmliche Weise auszurechnen, sieht man hier:

$$
\frac{2.35\left(-\frac{625}{2\left(\frac{17}{8}+\frac{1}{400}(-1000+y)\right)^3}+\frac{25(-1000+y)}{32\left(\frac{17}{8}+\frac{1}{400}(-1000+y)\right)^2}-\frac{(-1000+y)^2}{512\left(\frac{17}{8}+\frac{1}{400}(-1000+y)\right)^3}\right)}{\sqrt{\frac{62500}{\left(\frac{17}{8}+\frac{1}{400}(-1000+y)\right)^2}+\frac{25(-1000+y)^2}{64\left(\frac{17}{8}+\frac{1}{400}(-1000+y)\right)^2}}}+
$$

$$
2.7\left(\frac{-150+y}{\sqrt{(-150-y)^2}}-\frac{\frac{5\left(400-\frac{250}{\frac{17}{8}+\frac{1}{400}(-1000+y)}\right)}{4\left(\frac{17}{8}+\frac{1}{400}(-1000+y)\right)^2}+2\left(1-\frac{5}{8\left(\frac{17}{8}+\frac{1}{400}(-1000+y)\right)}+\frac{-1000+y}{640\left(\frac{17}{8}-\frac{1}{400}(-1000+y)\right)^2}\right)\left(-1000-\frac{5(-1000+y)}{8\left(\frac{17}{8}+\frac{1}{400}(-1000+y)\right)}+y\right)}{2\sqrt{\left(400-\frac{250}{\frac{17}{8}+\frac{1}{400}(-1000+y)}\right)^2+\left(-1000-\frac{5(-1000+y)}{8\left(\frac{17}{8}+\frac{1}{400}(-1000+y)\right)}-y\right)^2}}\right)
$$

Schließlich fanden wir numerisch die Koordinaten des Punktes (400, 840), der die Preiskosten auf nur 3356$ senkt. Wie beim Rechteck gingen wir nun wieder dazu über, die Flächenformeln für sämtliche Bereiche aufzustellen. Glücklicherweise mussten wir aber nicht alle Rechtecksformeln auch tatsächlich verändern, wir konnten sechs davon weiterverwenden. Nachdem wir die übrigen drei aufgestellt hatten und die Definitionsbereiche neu zusammengestellt hatten, konnten wir nach einigem Ärger mit den Details endlich den 3D Graphen modellieren.

Diesem liegen - wie dem Rechtecksgraphen - neun Funktionen zugrunde. Diese konnten wir wegen des glücklich gewählten Eckpunktes sowohl mit der Vektorrechnung als auch mit dem Strahlensatz berechnen, was die Kontrolle erheblich erleichterte.

$$
\mathrm{lAK}[x\_, y\_] = \sqrt{(400-x)^2+(y-150)^2};
$$

$$
\mathrm{lBK}[x\_, y\_] = \sqrt{x^2+(y-1000)^2};
$$

$$
\mathrm{Px}[x\_, y\_] = 250 / ((y-1000)/x + (850/400));
$$

$$
\mathrm{Py}[x\_, y\_] = 1000 + (y-1000)/x * \mathrm{Px}[x, y];
$$

$$
\mathrm{Qx}[x\_, y\_] = (400-x) * 250 * ((150-y) + 850 * (400-x)/400)^{1} + 400;
$$

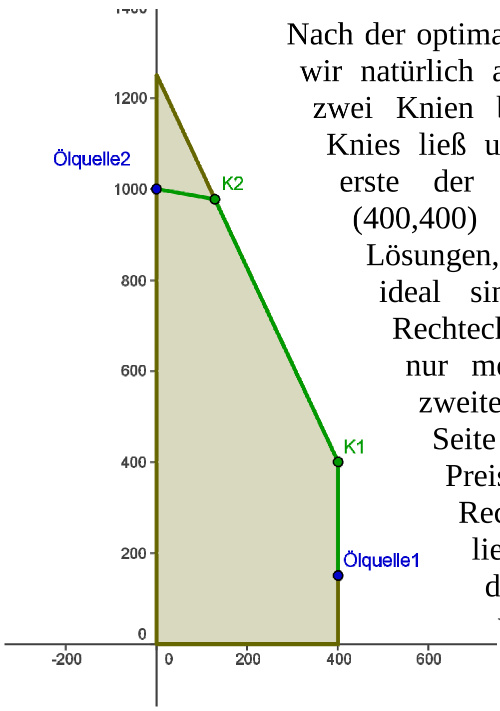
$$
\mathrm{Qy}[x\_, y\_] - (150-y) * 250 * ((150-y) + 050 * (400-x)/400)^{-1} + 150;
$$

$$
\mathrm{kf1}[x\_, y\_] := 2.7 * \left(\sqrt{x^2+(y-1000)^2} + \sqrt{x^2+(y-150+400*(y-150)/(x-400))^2}\right) + 4.7 * \sqrt{400^2+(400*(y-150)/(x-400))^2};
$$

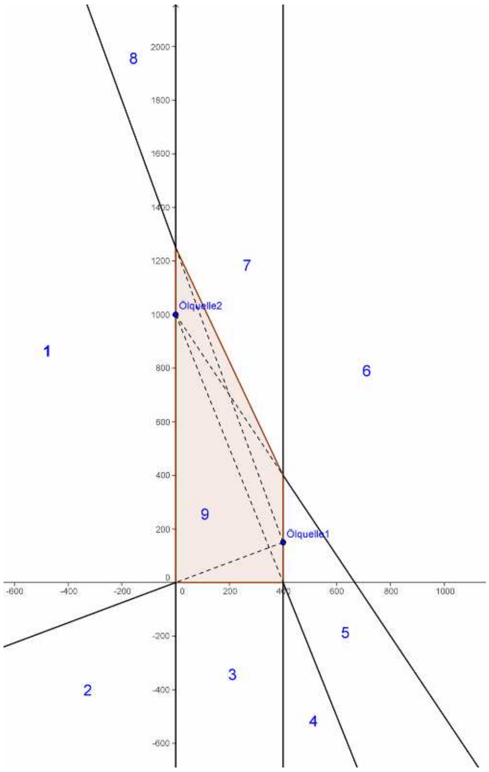
$$
\mathrm{kf2}[x\_, y\_] := 2.7 * \left(\sqrt{x^2+(y-1000)^2} + \sqrt{((x-400-(x-400)*(-150)/(y-150))^2+y^2)}\right) + 4.7 * \sqrt{((x-400)*150/(y-150))^2+150^2};
$$

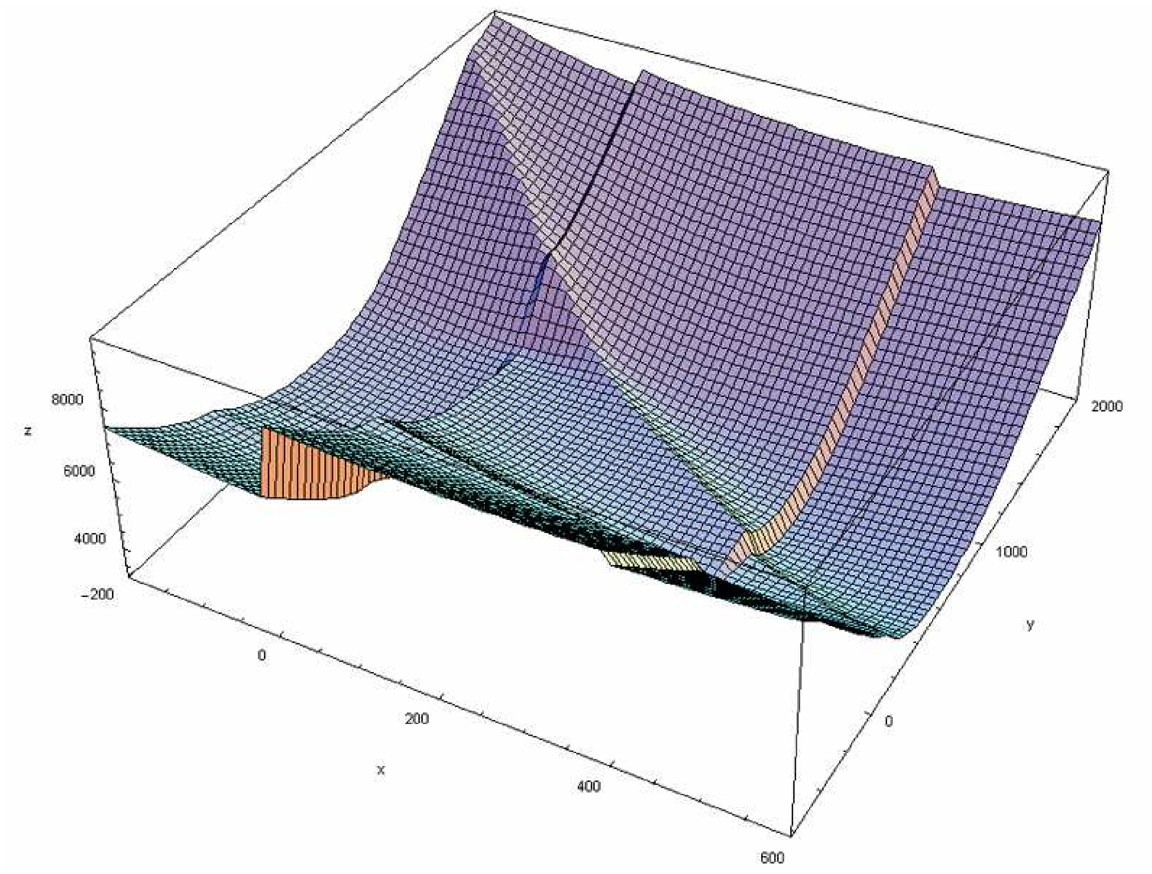
$$
\mathrm{kf3}[x\_, y\_] := 4.7 * \left(\sqrt{((400-x)*(-150)/(150-y))^2+150^2} + \sqrt{(x*(-1000)/(y-1000))^2+1000^2}\right) + 2.7 * \left(\sqrt{((400-x)*(-150)/(150-y)+400-x)^2+y^2} + \sqrt{(x*(-1000)/(y-1000)-x)^2+y^2}\right);
$$

```
kf4[x_, y_] := 2.7 * (lAK[x, y] + (-y) / (1000 - y) * lBK[x, y]) + 4.7 * (1000 / (1000 - y) * lBK[x, y]);

kf5[x_, y_] := 2.7 * (lAK[x, y] + (x - 400) * lBK[x, y] / x) + 4.7 * 400 * lBK[x, y] / x;

kf6[x_, y_] := 2.7 * (lAK[x, y] + √((x - Px[x, y])^2 + (y - Py[x, y])²)) + 4.7 * (√((0 - Px[x, y])² + (1000 - Py[x, y])²));
kf7[x_, y_] := 2.7 * (√((x - Px[x, y])^2 + (y - Py[x, y])^2) + √((x - Qx[x, y])^2 + (y - Qy[x, y])^2)) +
    4.7 * (√((0 - Px[x, y])² + (1000 - Py[x, y])²) + √((400 - Qx[x, y])^2 + (150 - Qy[x, y])^2));

kf8[x_, y_] :=
  2.7 * (√((0 - x)^2 + (1000 - y)^2) + √((x - Qx[x, y])^2 + (y - Qy[x, y])^2)) + 4.7 * (√((400 - Qx[x, y])^2 + (150 - Qy[x, y])^2));

kf9[x_, y_] := 4.7 * (√((400 - x)² + (150 - y)²) + √(x² + (y - 1000)²));
```



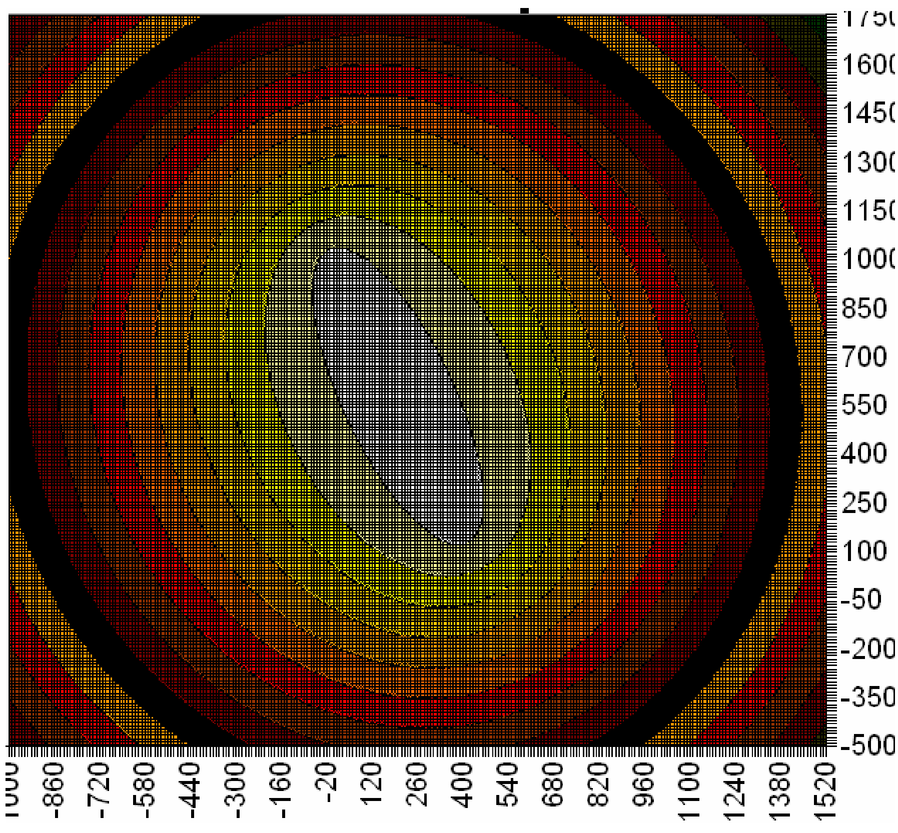
Nach der optimalen Lage eines Knies wollten wir natürlich auch die optimale Lage von zwei Knien bestimmen. Die Lage eines Knies ließ uns annehmen, dass sich das erste der zwei Knie im Eckpunkt (400,400) liegen muss. Alle anderen Lösungen, die möglicherweise auch ideal sind, hatten wir bereits im Rechteck berechnet. Wir mussten also nur mehr die optimale Lage des zweiten Knies auf der schrägen Seite des Trapezes finden und den Preis mit dem optimalen Preis des Rechteckes vergleichen. Dieser liegt an der Stelle (128, 977), die Lösung war mit 3010$ wesentlich billiger als im Rechteck.

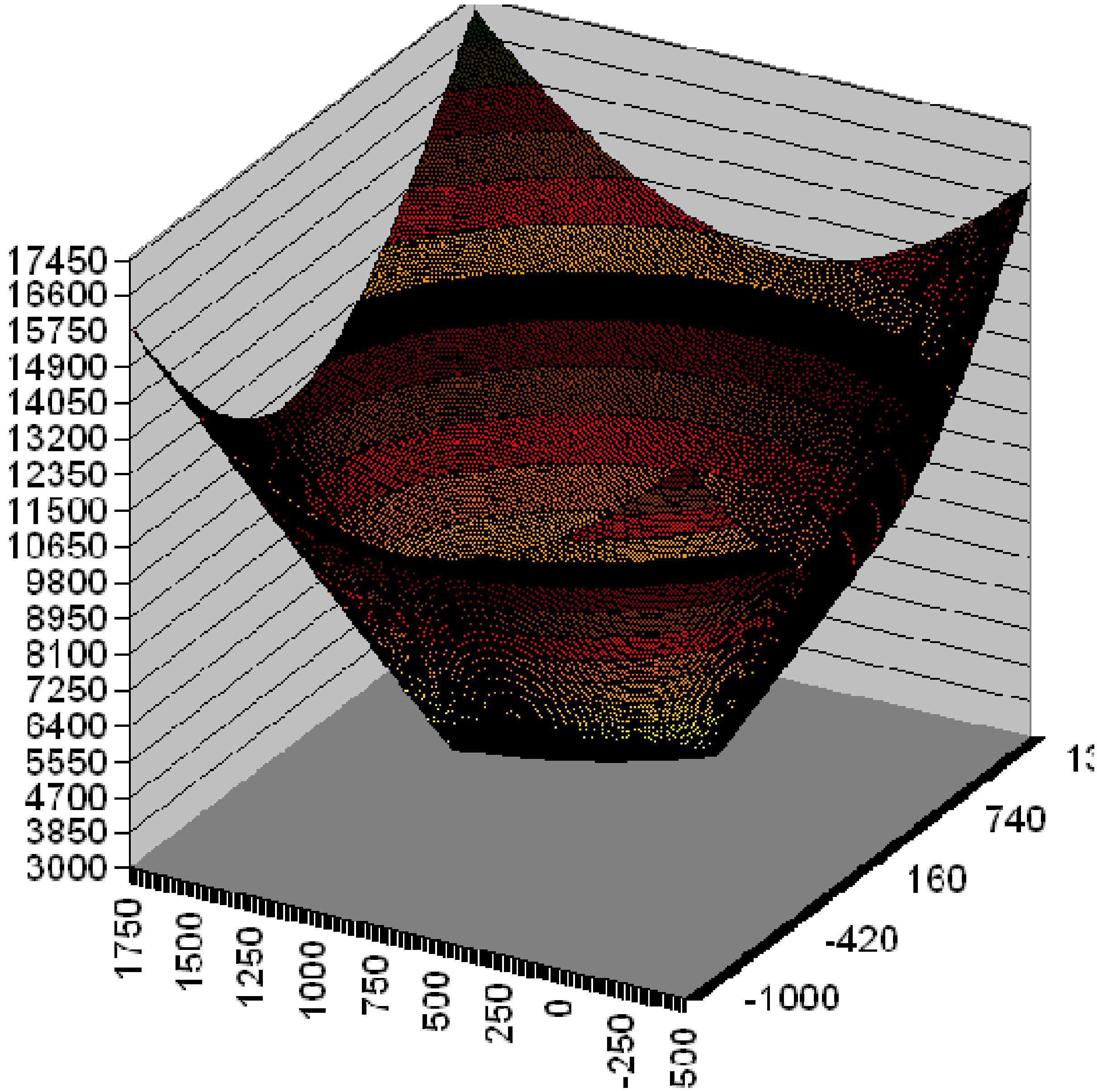




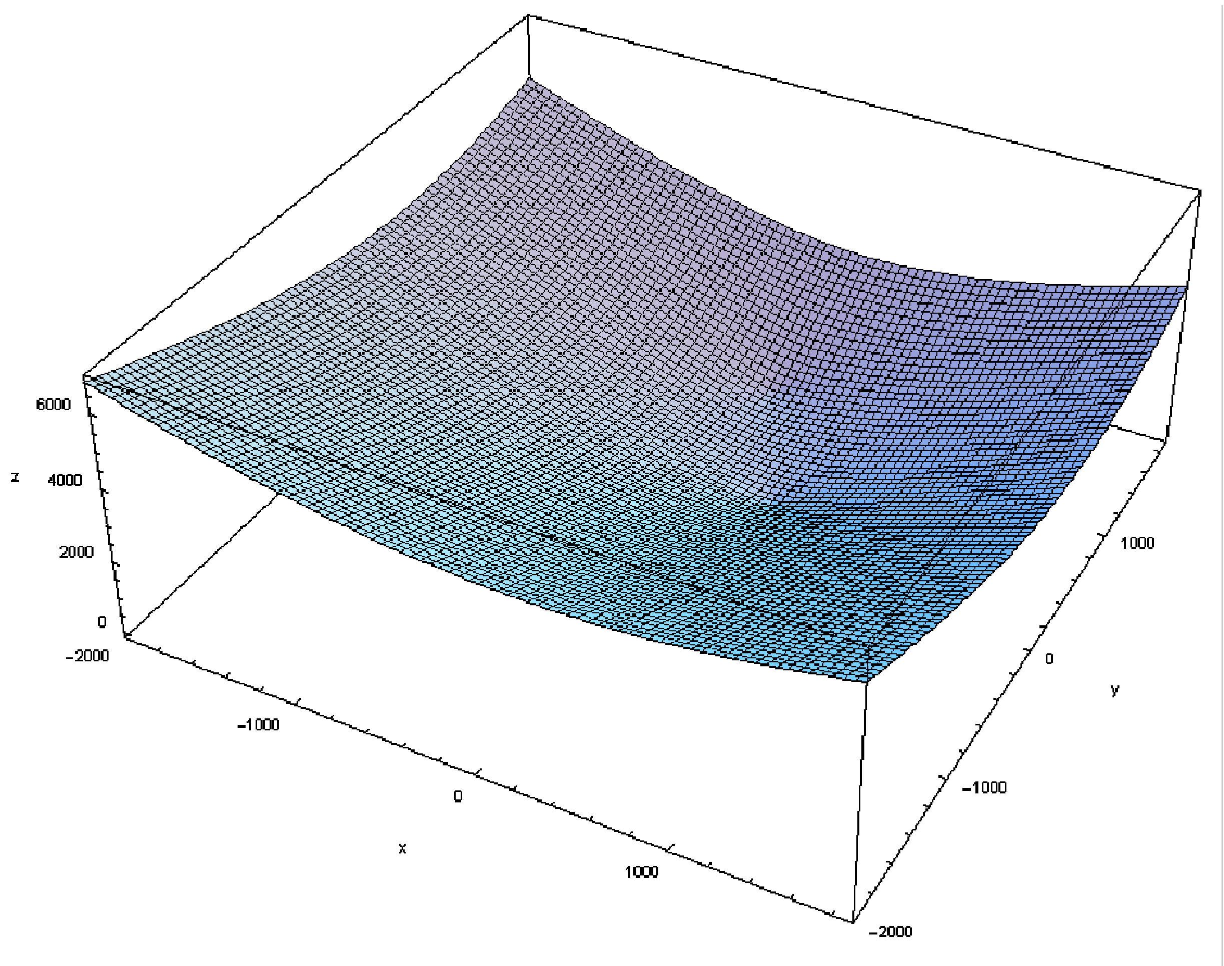
Probe

Um unsere Ergebnis zu überprüfen, versuchten wir, den Trockenpreis dem Sumpfpreis gleichzusetzen. Wenn die Formeln und die Definitionsbereiche des 3D Graphen stimmen, müsste dieser eine Ellipse mit den Punkten A und B als Brennpunkten zeigen. Und siehe da, das ist die Excel-Graphik von oben gesehen:





Hier ist die gleiche Graphik aus einem anderen Blickwinkel betrachtet. Man sieht dabei sehr gut die Gerade, die die beiden Ölquellen verbindet.



Noch einmal die Mathematica - Darstellung

Schloss Seggau, 18.1.2008

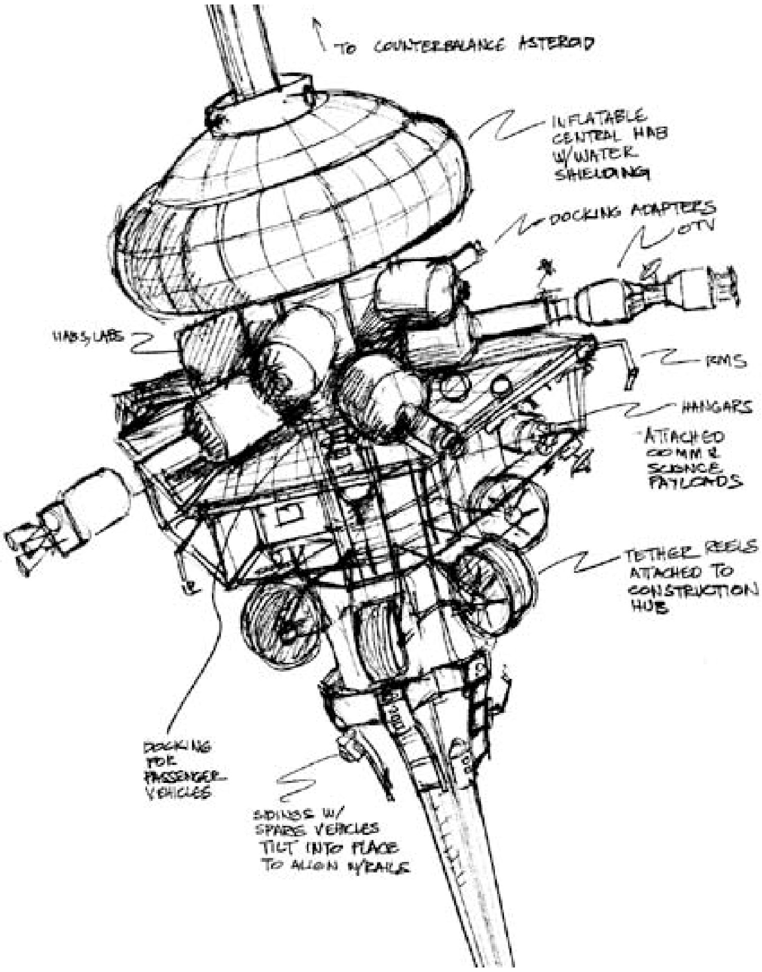
Teilnehmer:

Sebastian Hofer
Lukas Schafzahl
Mattias Schönberger
Harald Fitzek
Michael Draxler
Rudolf Krainer

Betreuer:

Prof. Peter Schöpf

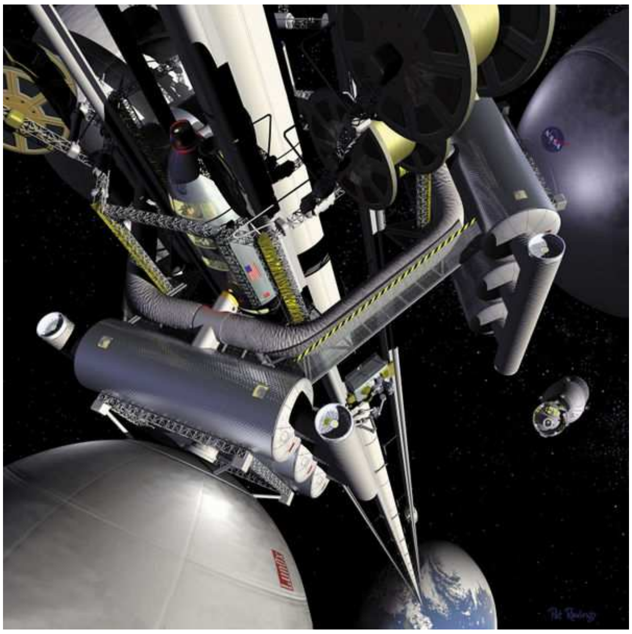
Abschließend wollen wir uns bei allen Organisatoren, Sponsoren und Mitwirkenden bedanken, die uns dieses einmalige Erlebnis ermöglicht haben.
Dank gilt auch den Angestellten des Schlosses. Besonderen Dank verdient unser Gruppenleiter, der uns mit viel Humor und Allgemeinwissen entgegenkam. Danke Herr Prof. Peter Schöpf!!
Natürlich bedanken wir uns auch bei den Schülern, die uns um 1 Uhr in der Früh mit ihren Klavierstücken bei Laune hielten und uns erfolgreich vom Schlaf bewahrt haben.



Project Space Elevator

Team:
Leitung: Prof. Bernd Thaller
Regina Seidl, Gernot Haselmann, Georg Groß, Robert Triebl, Lukas von Berg, Jakob Kleinschuster

Einführung in das Problem



Die Idee, einen Lift ausgehend von einem bestimmten Punkt des Äquators zu einem geostationären Orbit zu konstruieren, stammt ursprünglich von Konstantin Tsiolkovsky. Arthur C. Clarke greift dieses Modell in seinem Roman „The Fountains of Paradise" auf. Ihm schwebt vor, mit Hilfe dieses Lifts Materialien und sogar Menschen günstiger als mit Stufenraketen ins Weltall zu befördern.
Gleich zu Beginn des Projekts stellten sich für uns sehr viele interessante Fragen. Aufgrund der uns zur Verfügung stehenden Zeit konnten wir aber nur auf einige dieser Fragen näher eingehen. Es ist unserer Ansicht nach nicht möglich, alle Aspekte des Problems „Space Elevator" mathematisch darzustellen, weshalb wir uns nur mit für uns interessanten und spannenden Teilaspekten auseinandersetzten.
Viele Vorgänge, mit denen wir uns beschäftigten, hängen mit physikalischen Gesetzen zusammen. Ohne die Hilfe von unserem Betreuer Prof. Thaller hätten wir sicher nur einen Bruchteil der Ansätze, die unten näher beschrieben werden, genauer betrachten können.
Eine weitere Erschwernis für dieses Projekt des Lifts ist jene, dass viele Materialien und andere benötigten Technologien noch gar nicht erfunden bzw. noch nicht weit genug ausgereift sind. Wir konnten also, was beispielsweise das Kabel aus Carbonnanotubes betrifft, auf fast keine Erfahrungswerte mit diesem Material zurückgreifen.
Im ersten Moment scheint ein solches Projekt sowieso völlig absurd und unrealisierbar. Die Tatsache, dass die NASA jedoch teure Machbarkeitsstudien finanziert, zeigt aber doch, dass diese Idee nicht ganz als Nonsens abgetan werden kann.
Es ist natürlich klar, dass es keine Formel geben kann, die beschreibt, wie man so einen Lift am besten baut. Es spielen so viele Vorgänge und Aspekte eine Rolle, dass dies einfach nicht möglich ist. Auch ein Zusammenfügen der Teilbereiche des Projekts ist nicht möglich, da die einzelnen Systeme nicht wirklich im Zusammenhang stehen. Es spielt beispielsweise für das Antriebssystem keinerlei Rolle, aus welchem Material das Kabel besteht.

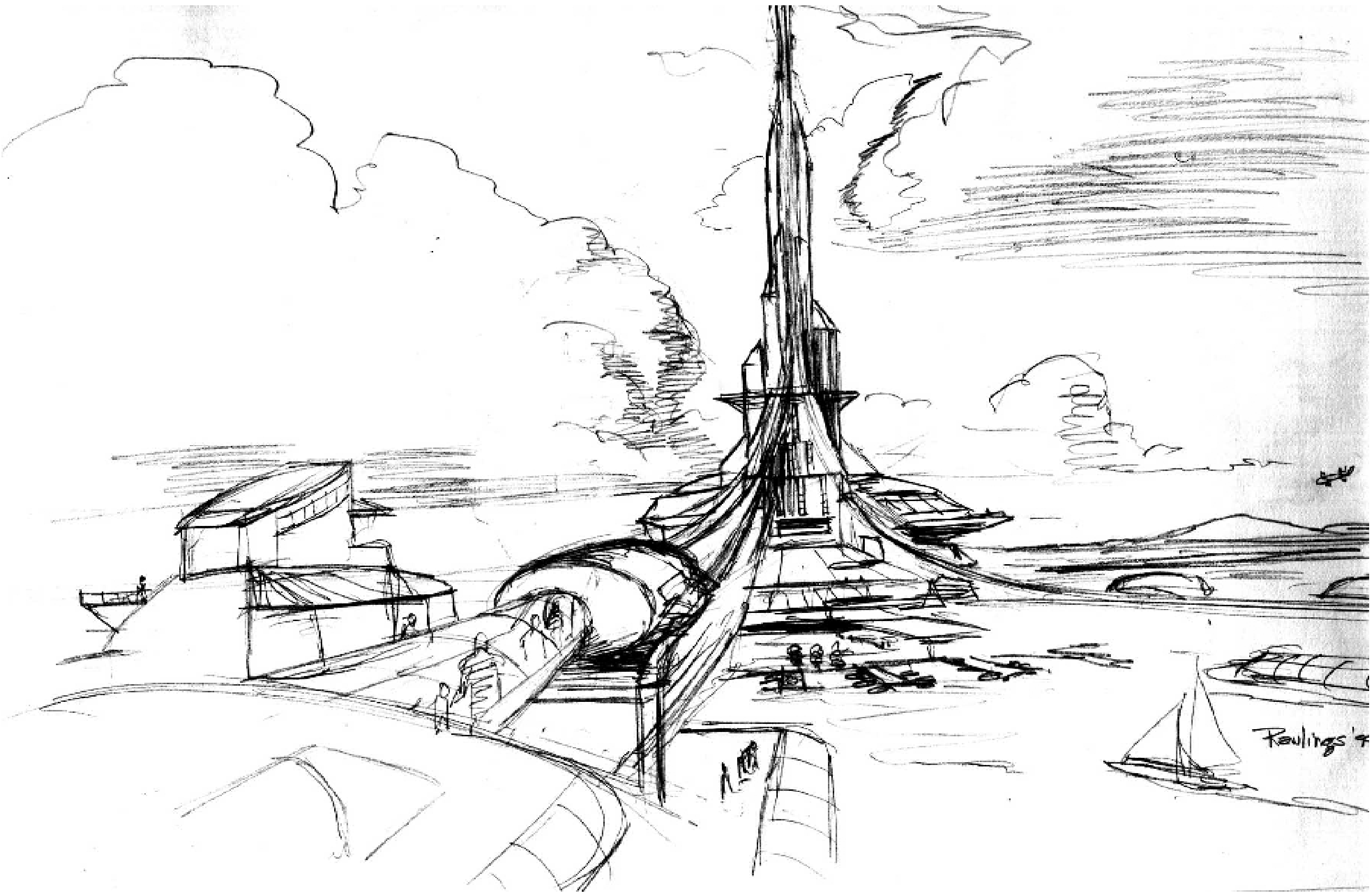
Ansätze

I. Höhe des geostationären Orbits

II. Ausstieg aus der Liftkabine

III. Seil

IV. Energie



I. Höhe des geostationären Orbits

Anfänglich muss man sich die Frage stellen, was allgemein als geostationärer Orbit bezeichnet wird: Ein Satellit befindet sich genau dann in einem solchen Orbit, wenn Zentripetalkraft (Gravitationskraft) und Zentrifugalkraft (Fliehkraft) genau gleich groß sind, das heißt in weiterer Folge, dass sich die beiden Kräfte, die auf diesen Satellit wirken, genau ausgleichen.
Im Folgenden wird nun die Berechnung der Höhe dieses Orbits anhand einfacher physikalischer Gesetze erläutert:

$$F_z = \frac{mv^2}{r}$$

Beschreibt die Zentrifugalkraft
m Masse des Satelliten
r der gesuchte Radius der Kreisbahn
v Tangentialgeschwindigkeit

$$F_g = G\frac{Mm}{r^2}$$

Beschreibt die Gravitationskraft
G Gravitationskonstante
($6{,}67428 \cdot 10^{-11} kg^{-1} m^3 s^{-2}$)
M Masse der Erde ($5{,}974 \cdot 10^{24}$ kg)
m Masse des Satelliten
r der gesuchte Radius der Kreisbahn

$$G\frac{Mm}{r^2} = \frac{mv^2}{r}$$

Nun setzt man diese beiden Kräfte gleich. (1)

$$G\frac{M}{r} = v^2$$

Durch umformen fällt m weg. (2)

$$v = \sqrt{G\frac{M}{r}}$$

v wird nun durch G, M und r beschrieben. (3)

$$\omega \cdot r = \sqrt{G\frac{M}{r}}$$

Es gilt $v = \omega \cdot r$. (4)

$$\omega = \sqrt{G\frac{M}{r^3}}$$

Nach ω umformen und r unter die Wurzel bringen. (5)

$$\frac{2\pi}{T} = \sqrt{G\frac{M}{r^3}}$$

Für ω wird nun $\frac{2\pi}{T}$ eingesetzt. (6)

$$\boxed{r = \sqrt[3]{\frac{T^2 GM}{4\pi^2}}}$$

Durch umformen kann nun r durch T, G und m ausgedrückt werden. T entspricht der Länge eines siderischen Tages (86.164 Sekunden). (7)

Bei richtiger Durchführung aller Operationen erhält man nun r= 42.168,44 km. Dieser Abstand ist allerdings inklusive des Erdradius (ca. 6365 km), dh. dass sich der geostationäre Orbit in ca. 35.803 km Höhe, gemessen von der Erdoberfläche, befindet.

<u>Erläuterungen:</u>

(4) v ist die Tangentialgeschwindigkeit, die man auch als Winkelgeschwindigkeit multipliziert mit dem Radius ausdrücken kann. (Beweis folgt)

Zur Kreisbahn: Man kann jeden Punkt x der sich auf der Kreisbahn befindet, durch 2 Koordinaten darstellen, die vom Radius r, sowie der Winkelgeschwindigkeit ω und der Zeit t abhängig sind:

$$\vec{x}(t) = (r \cdot \cos(\omega \cdot t), r \cdot \sin(\omega \cdot t))$$

$$\vec{x}'(t) = (-r\omega \cdot \sin(\omega \cdot t), r\omega \cdot \cos(\omega \cdot t)) = \vec{v}(t)$$

$$\left|\vec{v}(t)\right| = \sqrt{r^2\omega^2 \cdot \underbrace{(\sin^2(\omega \cdot t) + \cos^2(\omega \cdot t))}_{1}}$$

Pythagoras im Einheitskreis

$$\boxed{\left|\vec{v}(t)\right| = r\omega}$$

---

(5) $\omega = \frac{2\pi}{T}$

Die Winkelgeschwindigkeit beschreibt den pro Zeiteinheit zurückgelegten Winkel (also z.B.: der volle Winkel 2π pro Umlaufszeit T). Der Winkel wird hier durch die Bogenlänge am Einheitskreis gemessen. Der Umfang des Kreises ist $2r\pi$, im Einheitskreis wird r = 1 gesetzt, das heißt, dass der Umfang des Einheitskreises (also der volle Winkel) gleich 2π ist.

II. Ausstieg aus der Liftkabine

Nach dem ersten Keplerschen Gesetz sind die Umlaufbahnen eines Körpers im Gravitationsfeld eines Planeten Ellipsen. Weiters befindet sich der Schwerpunkt des Systems in einem der Brennpunkte der Ellipse. Wir gehen also davon aus, dass ein Mensch, der aus dem Lift aussteigt, bevor er den geostationären Orbit erreicht, eine elliptische Bahn um die Erde beschreiben würde.
Als Vorarbeit stellten wir einige grundsätzliche Überlegungen zur Ellipse an:

a	Große Halbachse der Ellipse
b	Kleine Halbachse der Ellipse
c	Abstand der Brennpunkte zum Mittelpunkt (M) = lineare Exzentrizität
B1, B2	Brennpunkte der Ellipse
M	Mittelpunkt der Ellipse

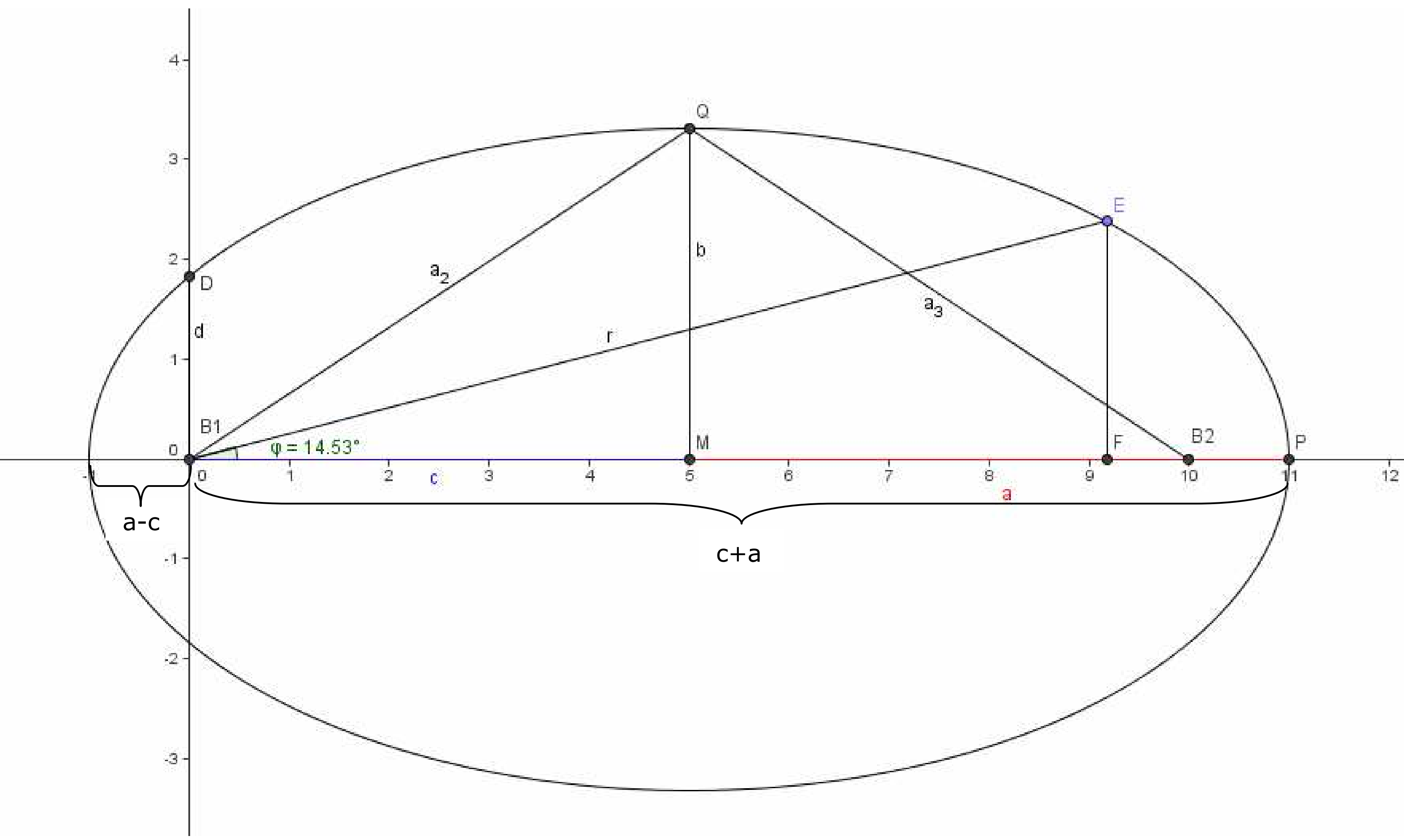
<u>Algebraische Darstellungen einer Ellipse:</u>

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

kartesische Koordinaten (der Mittelpunkt der Ellipse muss im Ursprung liegen)

$$r(\varphi)=\frac{d}{1-\varepsilon\cdot\cos\varphi} \qquad (2)$$

Polarkoordinaten (der linke Brennpunkt liegt im Ursprung)



Zusammenhänge:

$$\varepsilon = \frac{c}{a}$$

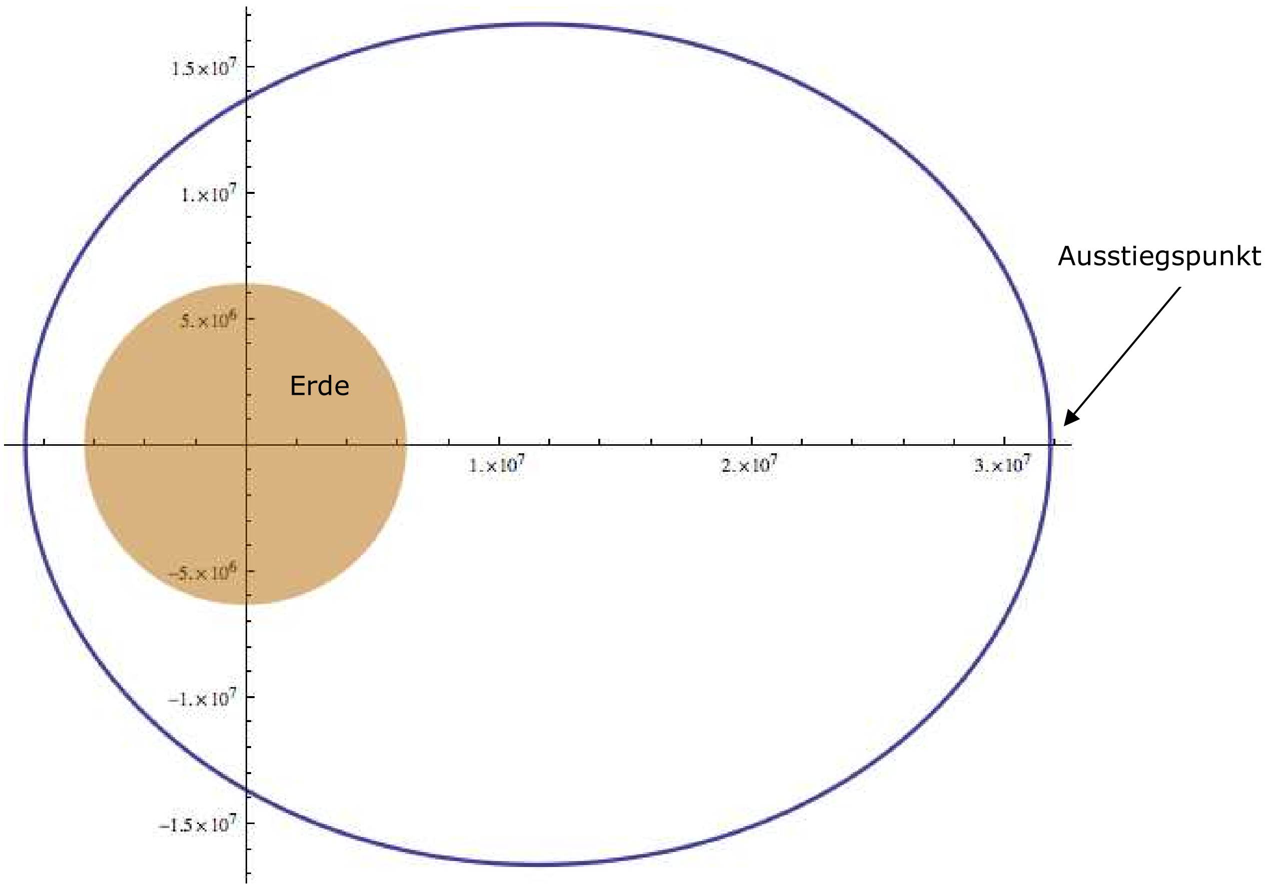
ε....Exzentrizität

Wenn ε gegen 0 geht, nähert sich die Form der Ellipse immer mehr der Form eines Kreises, da die Brennpunkte immer weiter in Richtung des Mittelpunkts wandern.

$$\frac{d}{1+\varepsilon} = a - c \qquad (3) \qquad\qquad \frac{d}{1-\varepsilon} = c + a \qquad (4)$$

Diese Formeln ergeben sich aus der Polarkoordinatendarstellung einer Ellipse (siehe 2), wenn man für φ entweder 0° oder 180° einsetzt. Cos 180° ist -1 und somit erhält man $1+\varepsilon$ (siehe 3). Cos 0° ist 1, daraus erhält man $1-\varepsilon$ (siehe 4).

Idee: Bahn, die beschrieben wird, wenn man aussteigt.



Folgende Überlegungen beweisen einerseits, dass unsere Umlaufbahn elliptisch ist, und ermöglichen uns in weiterer Folge die Berechnung der Umlaufbahn bei gegebenem Radius.

$$E = \frac{mv^2}{2} + V(x)$$

Dieser Ausdruck beschreibt die Gesamtenergie, die sich aus kinetischer und potentieller Energie berechnet.
E ist eine Erhaltungsgröße, dh. sie bleibt konstant.
V(x) = potentielle Energie

$$E = \frac{p^2}{2m} - \frac{\gamma}{r}$$

Der Impuls ist definiert als Masse mal Geschwindigkeit. Es gilt also: $p = m \cdot v$

$$\vec{L} = \vec{x} \times \vec{p}$$

Eine weitere Erhaltungsgröße ist der Drehimpuls $\vec{L}$. Dieser Vektor ist das Kreuzprodukt aus dem Ortsvektor des Punktes x und dem Vektor des Impulses p.

$$\left|\vec{L}\right| = |\vec{x}| \cdot |\vec{p}| \sin\alpha$$

Um den Betrag des Vektors $\vec{L}$ zu erhalten, muss der Winkel α zwischen den beiden Vektoren $\vec{x}$ und $\vec{p}$ miteinbezogen werden.

$$\vec{K} = \vec{L} \times \vec{p} + \frac{m\gamma\vec{x}}{|\vec{x}|}$$

Die dritte konstante Größe ist der Vektor $\vec{K}$, der so genannte Runge – Lenz Vektor (Skizze siehe unten).
Dieser Vektor liegt genau in der Bahnebene der Ellipse.

$$\vec{K} \cdot \vec{x} = \left|\vec{K}\right| \cdot |\vec{x}| \cos\varphi$$

$$\vec{K} \cdot \vec{x} = (\vec{L} \times \vec{p})\vec{x} + m\gamma \frac{\vec{x} \cdot \vec{x}}{|\vec{x}|}$$

Für $\vec{K}$ setzt man $\vec{L} \times \vec{p} + \frac{m\gamma\vec{x}}{|\vec{x}|}$ ein.

$$\vec{K} \cdot \vec{x} = (\vec{p} \times \vec{x})\vec{L} + m\gamma \frac{\vec{x} \cdot \vec{x}}{|\vec{x}|}$$

$\vec{L}, \vec{p}$ und $\vec{x}$ werden zyklisch vertauscht.

$$\vec{K} \cdot \vec{x} = -\vec{L} \cdot \vec{L} + m\gamma \frac{\vec{x} \cdot \vec{x}}{|\vec{x}|}$$

Für $\vec{x} \times \vec{p}$ setzt man $\vec{L}$ ein.

$$\vec{K} \cdot \vec{x} = -\vec{L}^2 + m \cdot \gamma \cdot r$$

Aus $\vec{x} \cdot \vec{x}$ wird $|\vec{x}|^2$. Dividiert durch $|\vec{x}|$ erhält man wieder $|\vec{x}|$, was wiederum r ist.

$$\left|\vec{K}\right| \cdot |\vec{x}| \cos\varphi = -\vec{L}^2 + m \cdot \gamma \cdot r$$

Für $\vec{K} \cdot \vec{x}$ wird $\left|\vec{K}\right| \cdot |\vec{x}| \cos\varphi$ eingesetzt.

$$r \cdot (m\gamma - \left|\vec{K}\right| \cdot \cos\varphi) = \vec{L}^2$$

Zwischenschritt zur Umformung nach r.

$$r = \frac{\vec{L}^2}{(m\gamma - |\vec{K}| \cdot \cos\varphi)}$$

$$r = \frac{\vec{L}^2 / m\gamma}{(1 - |\vec{K}| / m\gamma \cdot \cos\varphi)}$$

Der ganze Bruch wurde mit $1/m\gamma$ erweitert.

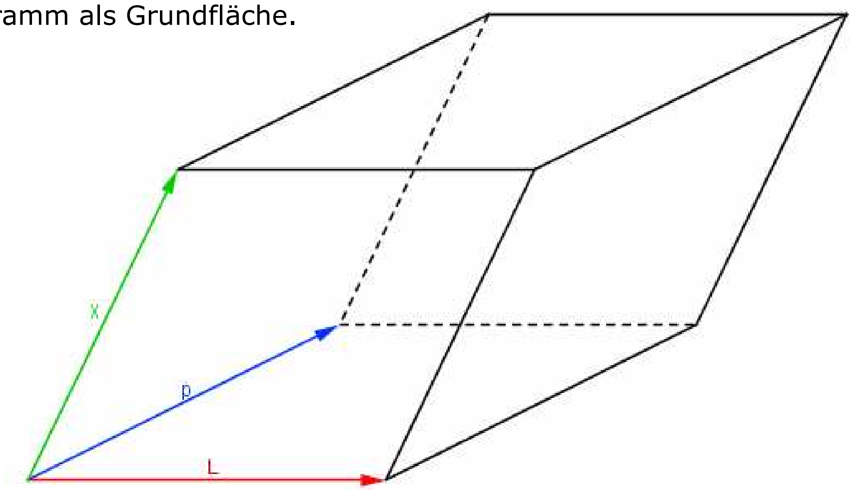
$$\boxed{r = \frac{d}{(1 - \varepsilon \cdot \cos\varphi)}}$$

Für $\vec{L}^2 / m\gamma$ setzt man d ein, für $|\vec{K}| / m\gamma$ setzt man ε ein und erhält eine Möglichkeit, eine Ellipse darzustellen.

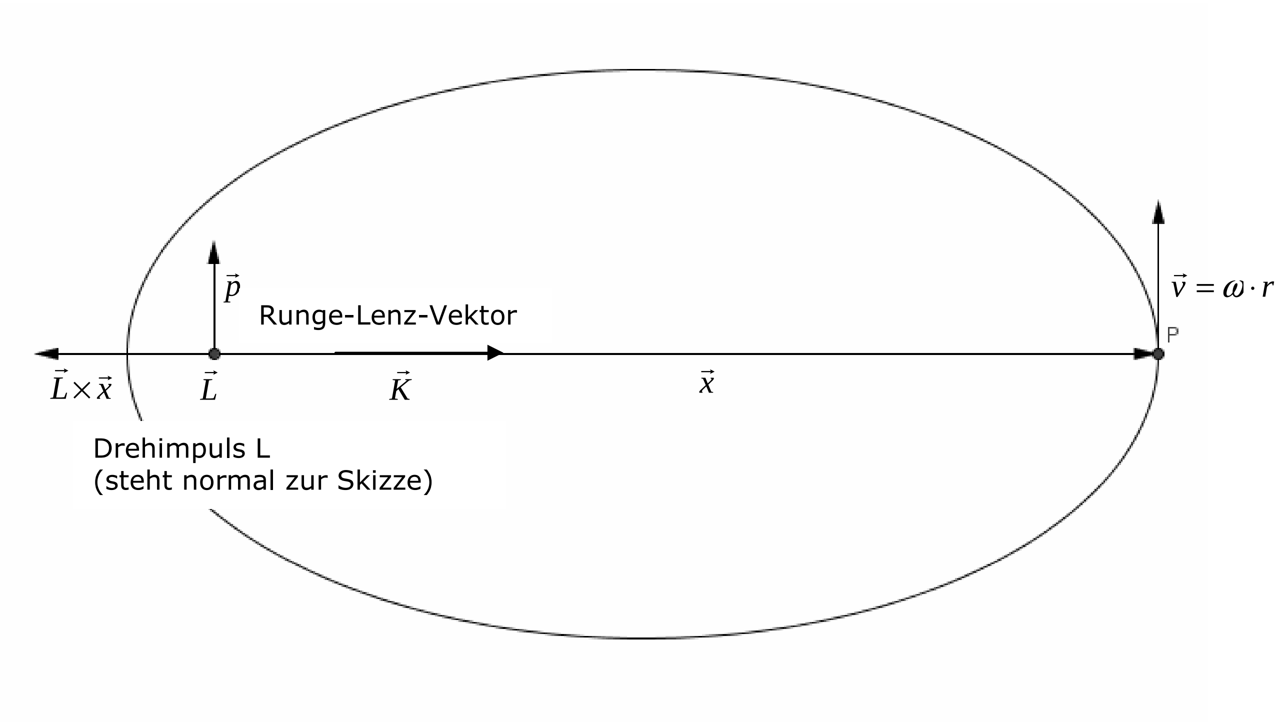
Anmerkung:

$$(\vec{L} \times \vec{p}) \cdot \vec{x} = V$$

Das „Triple product" beschreibt das Volumen eines Parallelepipedes mit einem Parallelogramm als Grundfläche.



Skizze:



Tatsächliche Berechnung der Absprunghöhe

$\vec{L} = \vec{x} \times \vec{p}$

Der Vektor $\vec{L}$ steht senkrecht zur von $\vec{x}$ und $\vec{p}$ aufgestellten Ebene

$\left|\vec{L}\right| = |\vec{x}| \cdot |\vec{p}| \cdot \sin \varphi$

Da $\vec{x}$ und $\vec{p}$ normal aufeinander stehen beträgt der eingeschlossene Winkel 90°, sin 90° gleich 1 also gilt, $\left|\vec{L}\right| = |\vec{x}| \cdot |\vec{p}|$

$\boxed{\left|\vec{L}\right| = m \cdot \omega \cdot r^2}$

$|\vec{p}| = m \cdot \omega \cdot r$, $|\vec{x}| = r$

$\vec{K} = \vec{L} \times \vec{p} + m\gamma \frac{\vec{x}}{|\vec{x}|}$

$\left|\vec{K}\right| = \left|\left|\vec{L} \times \vec{p}\right| - m\gamma\right|$

$\left|\vec{K}\right| = \left|m^2 \cdot \omega^2 \cdot r^3 - GMm^2\right|$

$\gamma = GMm$, $\left|\vec{L}\right| = m \cdot \omega \cdot r^2$, $|\vec{p}| = m \cdot \omega \cdot r$

$\boxed{\left|\vec{K}\right| = m^2 \cdot \left|\omega^2 \cdot r^3 - GM\right|}$

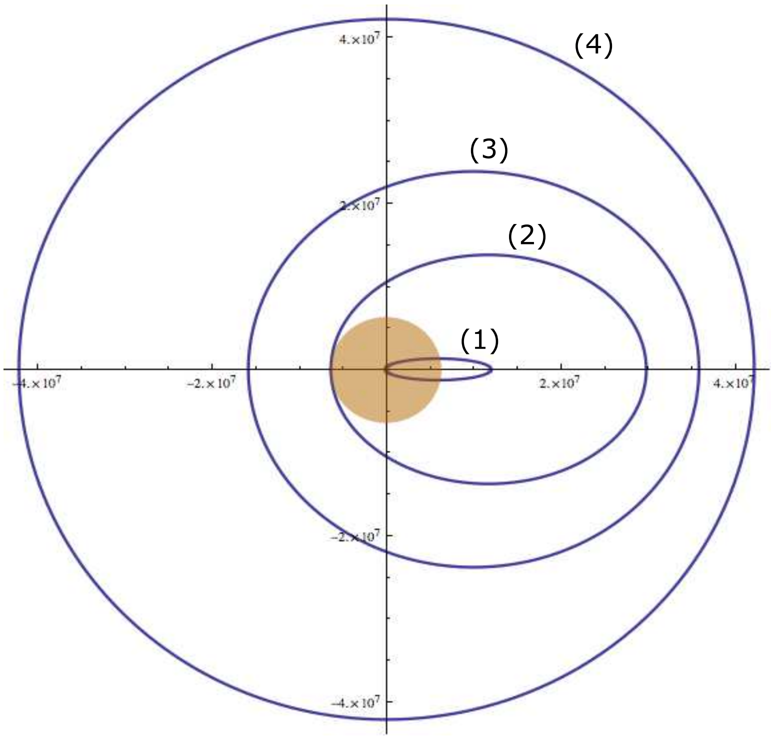
$$\varepsilon = \frac{\left|\vec{K}\right|}{m\gamma} \qquad d = \frac{\left|\vec{L}\right|^2}{m\gamma} \qquad \gamma = GMm$$

$$\varepsilon = \frac{\left|\vec{K}\right|}{GMm^2} \qquad d = \frac{\left|\vec{L}\right|^2}{GMm^2}$$

$$\varepsilon = \frac{m^2 \cdot \left|\omega^2 \cdot r^3 - GM\right|}{GMm^2} \qquad d = \frac{m^2 \cdot \omega^2 \cdot r^4}{GMm^2}$$

$\left|\vec{K}\right|$ und $\left|\vec{L}\right|$ werden eingesetzt (siehe oben)

$$\boxed{\varepsilon = \left|\frac{\omega^2 \cdot r^3}{GM} - 1\right|} \qquad \boxed{d = \frac{\omega^2 \cdot r^4}{GM}}$$



(1) Absprung aus ca. 12.000 km Höhe: Fall wird durch Erdoberfläche gestoppt

(2) Absprung aus ca. 30.000 km Höhe: man kommt knapp an der Erde und deren Atmosphäre vorbei

(3) Absprung aus ca. 36.000 km Höhe: nach zwei Umrundungen der Erde ist man wieder mit dem Lift an gleicher Stelle (theoretischer Wiedereinstieg möglich)

(4) Absprung aus 42.000 km Höhe: man befindet sich im geostationären Orbit, die Umlaufbahn ist ein Kreis, man schwebt immer neben der Station im Orbit

Angaben in km vom Erdmittelpunkt gemessen.

Bei einem Absprung vor dem geostationären Orbit, gerät man in eine Bahn in Form einer Ellipse, wobei der Erdmittelpunkt im linken Brennpunkt liegt. Diese Bahn wird jedoch bei einer niedrigen Absprunghöhe von der Erde gestoppt, man fällt also auf die Erdoberfläche.

Um die Grenze zu finden, wann man an der Erde „vorbei“ fällt, muss man den Abstand vom Perigäum (erdnächster Punkt) zum Brennpunkt (Erdmittelpunkt) mit dem Erdradius vergleichen:

$$\frac{d}{1+\varepsilon}=a-c$$

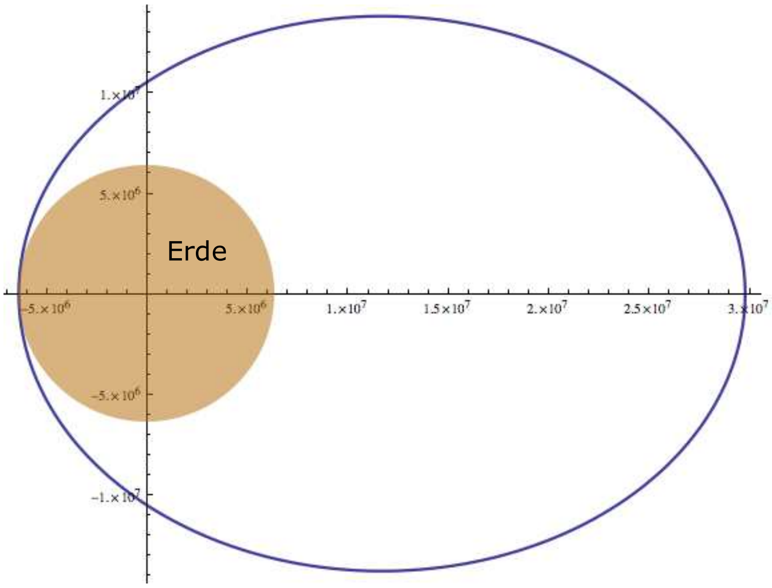
$$\frac{\frac{\omega^2\cdot r^4}{GM}}{1+\left|\frac{\omega^2\cdot r^3}{GM}-1\right|}=a-c$$

Für d und ε werden nun die oben errechneten Terme eingesetzt.

$$\frac{\frac{\omega^2\cdot r^4}{GM}}{1+\left|\frac{\omega^2\cdot r^3}{GM}-1\right|}=6{,}578*10^6$$

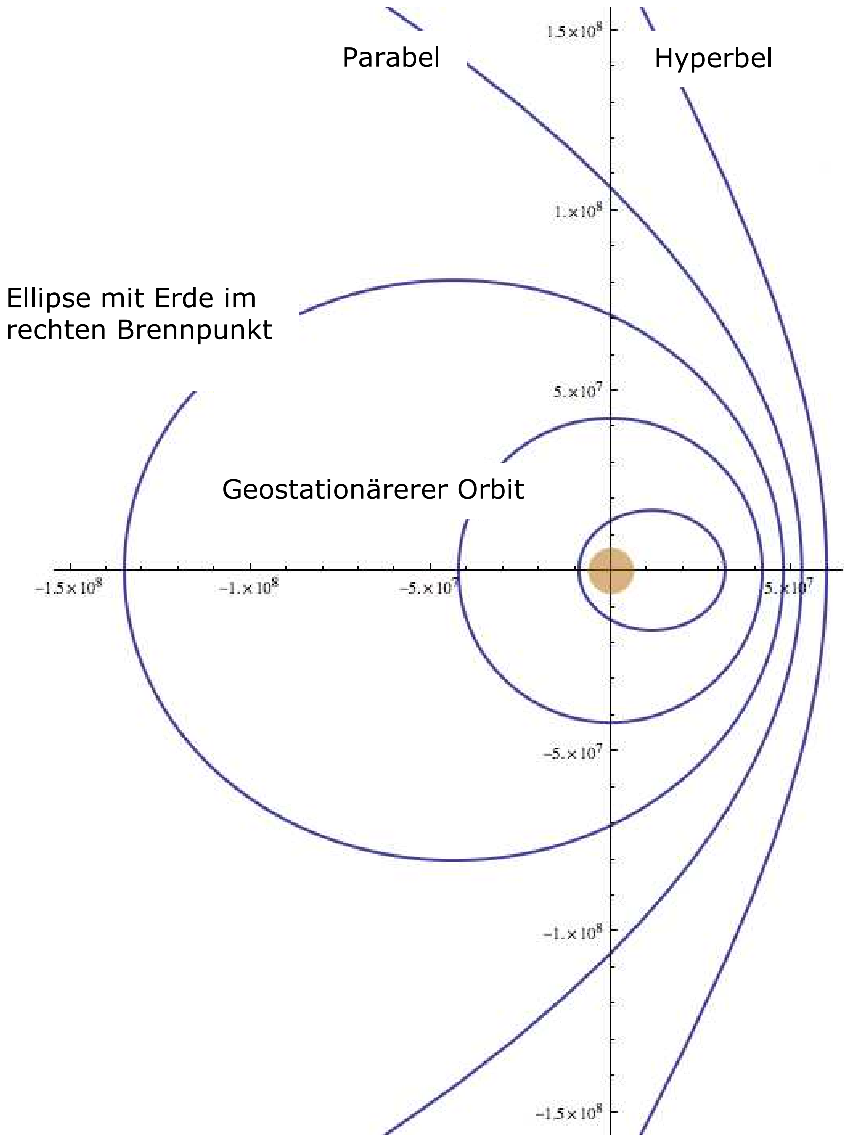
a-c ist das Perigäum (Erdradius + 200km, da man sonst in der Atmosphäre verglühen würde), also setzt man für a-c 6578 km ein.

Wenn man diese Gleichung richtig löst erhält man für den Radius <u>$r=2.9991*10^7$</u>



Es stellt sich natürlich auch die Frage, ob man den Erdorbit auch verlassen kann, wenn man sich außerhalb des geostationären Orbits fallen lässt. Hier entsteht zunächst wiederum eine Ellipse, wobei der Erdmittelpunkt diesmal der rechte Brennpunkt ist, da ε kleiner als 0 ist. Ab einer gewissen Entfernung zum geostationären Orbit ist die Flugbahn jedoch keine Ellipse mehr, sonder eine Parabel, wodurch der Körper nie mehr zur Erde zurück kommt. Wenn man sich noch weiter von diesem Punkt entfernt und das Seil los lässt, beschreibt die Flugbahn eine Hyperbel. Mathematisch gesehen nähert sich ε -1, was eine Division durch 0 zur Folge hätte. Diese Parabel entsteht beim Loslassen in einer Entfernung von $5{,}312*10^7$m. Wenn ε kleiner als -1 ist (Entfernung weiter als $5{,}312*10^7$m), erhält man eine Hyperbel, was ebenfalls ein Verlassen des Orbits zur Folge hat.

Mit diesem Effekt könnte man Raumschiffe für weitere Missionen im Weltall auf sehr hohe Geschwindigkeiten beschleunigen und damit viel Energie sparen.



Umlaufzeit:

Wir stellten uns natürlich auch die Frage, wie lange man um die Erde kreist, falls man aus dem Lift aussteigt. Zur Berechnung der Umlaufzeit einer elliptischen Bahn verwendet man das 3. Keplersche Gesetz.

$$\frac{T_1^2}{T_2^2} = \frac{a_1^3}{a_2^3}$$

Das 3. Keplersche Gesetz beschreibt den Zusammenhang zwischen Umlaufzeit und großer Halbachse.

$$\frac{d}{1+\varepsilon} + \frac{d}{1-\varepsilon} = 2a$$

Zur Berechnung der großen Halbachse muss man die beiden Teilstücke der Hauptachse addieren (siehe Skizze).

$$a = \frac{d}{1-\varepsilon^2}$$

Durch Umformen erhält man a.

$$T_2^2 = \frac{T_1^2 a_2^3}{a_1^3}$$

$$T_2^2 = \frac{T_1^2 \left(\frac{d}{1-\varepsilon^2}\right)^3}{a_1^3}$$

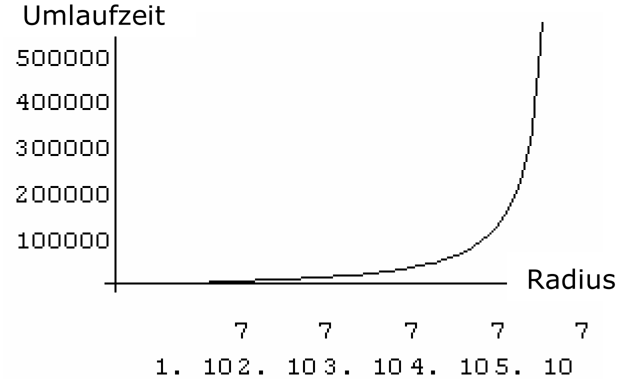
Durch Umformen erhält man diese Gleichung, wobei T_1 der siderische Tag und a_1 der geostationäre Orbit ist.

$$T_2 = \sqrt{\frac{T_1^2 \left(\frac{d}{1-\varepsilon^2}\right)^3}{a_1^3}}$$

$$T_2(r) = \sqrt{\frac{T_1^2 \left(\frac{\frac{\omega^2 \cdot r^4}{GM}}{1-\left|\frac{\omega^2 \cdot r^3}{GM}-1\right|^2}\right)^3}{a_1^3}}$$

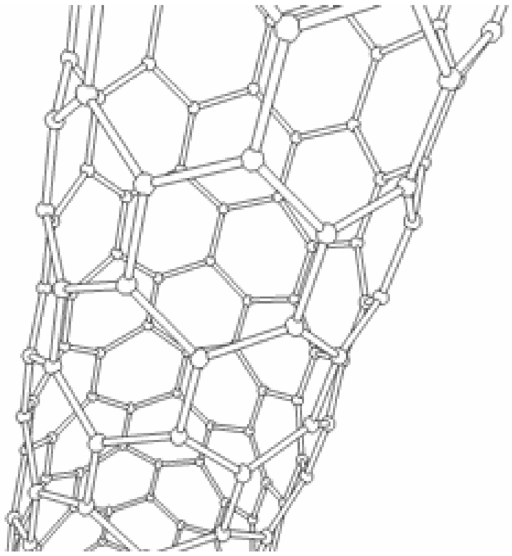
Einsetzen von d und ε erhält man diese Gleichung in r, alle anderen Buchstaben sind Konstante.

Mit dieser Formel kann man die Umlaufzeiten berechnen. Außerdem kann man den Abstand vom Erdmittelpunkt ermitteln, bei dem man abspringen muss, um nach zwei Umrundungen wieder in die Liftkabine einzusteigen ($3{,}6*10^7$m). Bei $4{,}6*10^7$m kann man bereits nach einer Umrundung der Erde einsteigen, wobei sich diese in der Zwischenzeit zweimal um die eigene Achse gedreht hat.



III. Seil

Das Seil, welches bis über den geostationären Orbit führen sollte, ist immensen Kräften ausgesetzt. Das stellte uns (und auch die Wissenschaftler der NASA) vor ein großes Problem: So ein Seil gibt es nicht! Ein homogenes Stahlseil würde allein durch sein Eigengewicht schon nach einem Bruchteil der Strecke reißen. Allerdings wird an einem neuen Material gearbeitet, welches dieser Belastung gewachsen wäre. Diese Errungenschaft wird Carbonnanotube genannt, die Struktur besteht aus mikroskopisch kleinen röhrenförmigen Gebilden, welche aus Kohlenstoff hergestellt werden. Die mechanischen Eigenschaften der zu Fasern zusammengefassten Nanoröhrchen sind überragend: Sie haben eine Dichte von 1,3-1,4 g / cm^3 und eine Zugfestigkeit von 45 GigaPascal. (Zum Vergleich: Stahl hat eine Dichte von mind. 7,8 g / cm^3 und eine max. Zugfestigkeit von 2 GPa. Folglich haben Carbonnanotubes ein 135-mal besseres Verhältnis von Zugfestigkeit zu Dichte als Eisen)

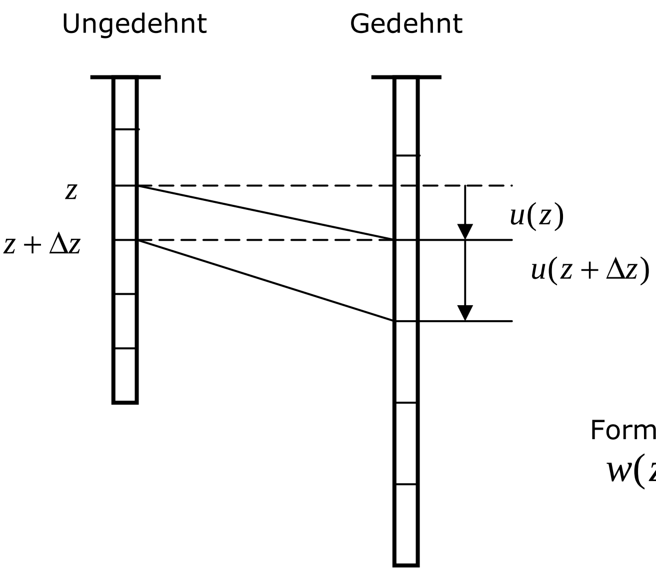


Struktur eines Carbonnanotubes

Dehnung:

Wir gehen von 2 Seilmodellen aus:

1) Gleichbleibende Dicke und homogenes Gravitationsfeld

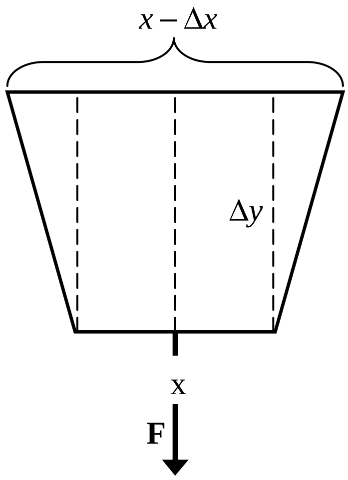


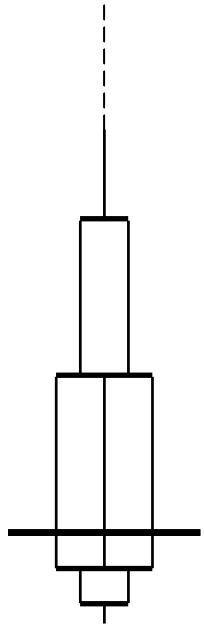
$$\Rightarrow \quad \frac{u(z+\Delta z)-u}{\Delta z} = \frac{du(z)}{dz} = u'(z)$$

Formel für die Dehnung:

$$w(z) = c(z)u'(z)$$

2) Verbreiterung nach Oben





Wir wollen erreichen, dass die Dehnung an allen Stellen gleich groß ist. Daher muss auch der Zug gleich groß sein.
Dieser Zug berechnet sich aus Kraft und Fläche:

$$p = \frac{F}{x}$$

p = Zug
F = Kraft
x = Breite an der Stelle r

$$p = \frac{F + g(r)\Delta A \rho}{x + \Delta x}$$

p=Zug
g(r)=Kraft, welche auf das Trapez wirkt.

$$\Delta A = \left(x + \frac{\Delta x}{2} \right) \cdot \Delta y$$

Fläche des Trapezes

$$g(r) = \frac{GMm}{r^2} - m\omega^2 r$$

g(r)=Kraft, welche auf das Trapez wirkt
($g(R_{gst})=0$)

Nach mehreren Umformungen erhält man:

$$\frac{\Delta y}{\Delta x} = \frac{p}{g(r)\left(x + \frac{\Delta x}{2} \right)\rho}$$

Diese Gleichungen erhält man nach mehreren Umformungen der oben beschriebenen Gleichungen.

$$\frac{dr}{dx} = \frac{p}{g(r)x\rho}$$

Wenn Δx nach 0 strebt (x wird als Funktion von r betrachtet) und r = R+y ist, wobei die Konstante R beim Streben nach 0 vernachlässigbar ist.

$$\frac{dr}{dx} = \frac{1}{x(r)} x'(r) = (\ln x(r))'$$

Durch Umformungen erhält man die Ableitung einer Funktion.

$$\ln x(r) = \frac{p}{\rho}\int g(r) = \frac{p}{\rho}\left(-\frac{GMm}{r}-\frac{m\omega^2 r^2}{2}\right)+c$$

Dieser Term wird durch Integrieren erreicht. c ist eine Integrationskonstante.

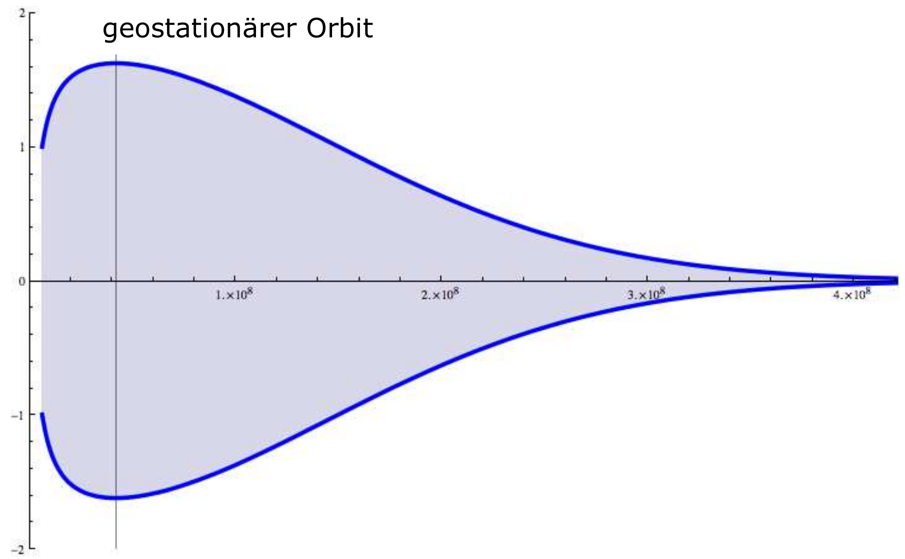
$$\boxed{x(r) = D\cdot e^{\frac{p}{\rho}\left(-\frac{GMm}{r}-\frac{m\omega^2 r^2}{2}\right)}}$$

Durch Potenzieren erhält man die Funktion x(r), welche die Dicke des Seiles in Abhängigkeit des Abstandes von der Erde angibt. $D = e^c$.

Diese Funktion gibt die Dicke eines Bandes an. Bei einem räumlichen Seil (statt ΔA des Trapezes verwendet man ΔV des Zylinderstumpfes) erhält man das gleiche Ergebnis, mit dem einzigen Unterschied, dass sich die Konstante geringfügig ändert.

Dicke des Seils (in m) abhängig vom Abstand zum geostationären Orbit

Das Seil kann aber nicht so lang sein, da es bereits bis zum Mond reichen würde. Man könnte jedoch das Seil nur bis zu einem gewissen Punkt hinter dem geostationären Orbit anbringen und an dessen Ende ein entsprechend schweres Gegengewicht befestigen. Diese Möglichkeit würde außerdem sehr viel Material sparen.



Zitat: www.zadar.net/space-elevator/

Probleme

- Das Seil könnte durch verschiedene Strahlungen beschädigt werden.
- „Weltraummüll", welcher das Seidl beschädigen könnte.
- Was würde passieren, wenn das Seil reißt?

Wenn das Seil reißt, würde der herabhängende Teil in einen niedrigeren Orbit versetzt werden und somit eine elliptische Bahn um die Erde beschreiben. Wir denken daher, dass sich das Seil wegen der Drehbewegung um die Erde wickeln würde. Die Station und das restliche Seil würden sich aufgrund der Fliehkraft von der Erde wegbewegen, in einen höheren Orbit gelangen und in einer wesentlich größeren elliptischen Bahn die Erde umkreisen.

IV. Energie

Um die Energieersparnis durch den Lift gegenüber der Rakete zu berechnen befassten wir uns mit dem Zuwachs an potenzieller Energie. Wie so oft gingen wir von der grundlegenden Formel für das Kraftfeld aus, welches sich aus der Differenz von Gravitationskraft und Fliehkraft berechnet.

$F_{(r)}$ ... Gravitationskraft

$F_{(z)}$ ... Fliehkraft

G ... Gravitationskonstante

M ... Erdmasse

m ... Masse eines Körpers im Gravitationsfeld

r ... Der Abstand vom Erdmittelpunkt zur Masse eines zweiten Körpers

R ... Der Abstand vom Erdmittelpunkt zur Erdoberfläche

ω ... Winkelgeschwindigkeit

$$F_G = -\frac{GMm}{r^2}$$

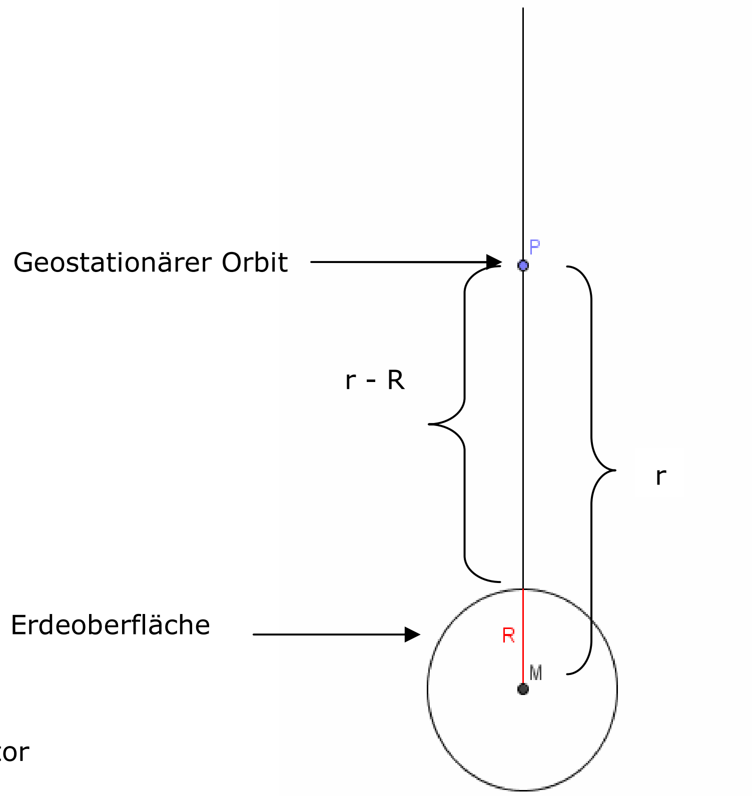
wir integrieren die Funktion F_G um die Arbeit zu erhalten.

$$V_G = -\frac{GMm}{r}$$

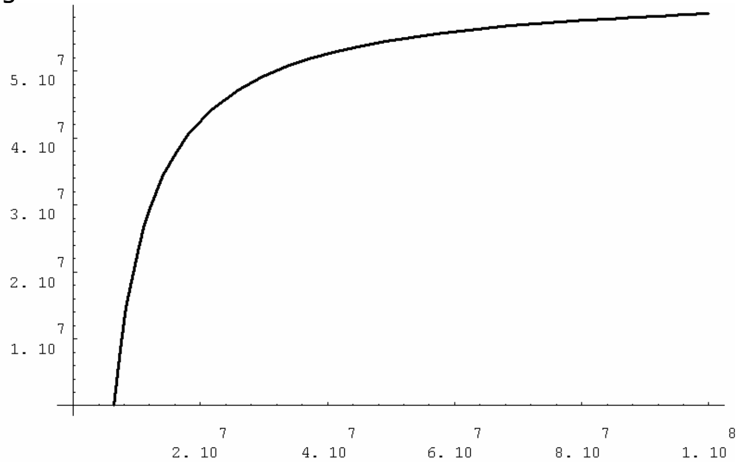
$$F_z = m\omega^2 r$$

weiters integrieren wir die Funktion F_z und erhalten die geleistete Arbeit.

$$V_z = -\frac{m\omega^2 r^2}{2}$$

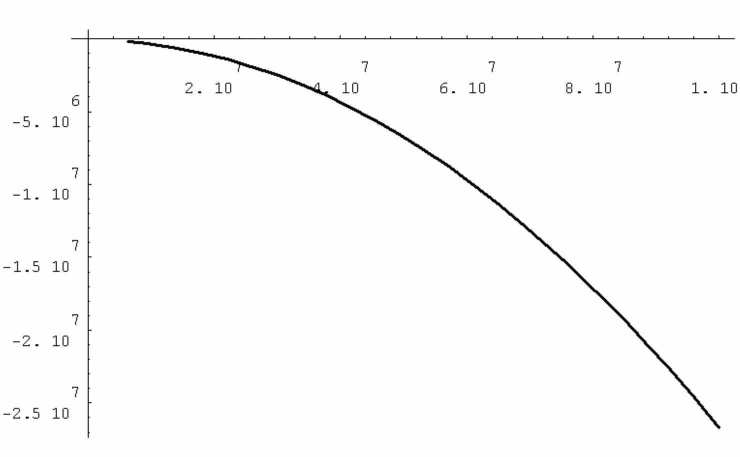


Die Arbeit, die aufgewendet werden muss, um einen Körper der Masse m gegen die Gravitationskraft von der Erdoberfläche bis in den geostationären Orbit zu befördern, berechnet sich aus der Differenz der potentiellen Energien von Erdradius und Radius des geostationären Orbits.



$$E_r = \frac{GMm}{R} - \frac{GMm}{r}$$

Da uns die Fliehkraft, beim Verlassen der Erde mit Hilfe des Liftes, behilflich ist, ist die von der Fliehkraft geleistete Arbeit von der Gravitationsarbeit abzuziehen. Bei dieser muss jetzt ebenfalls die Differenz gebildet werden.

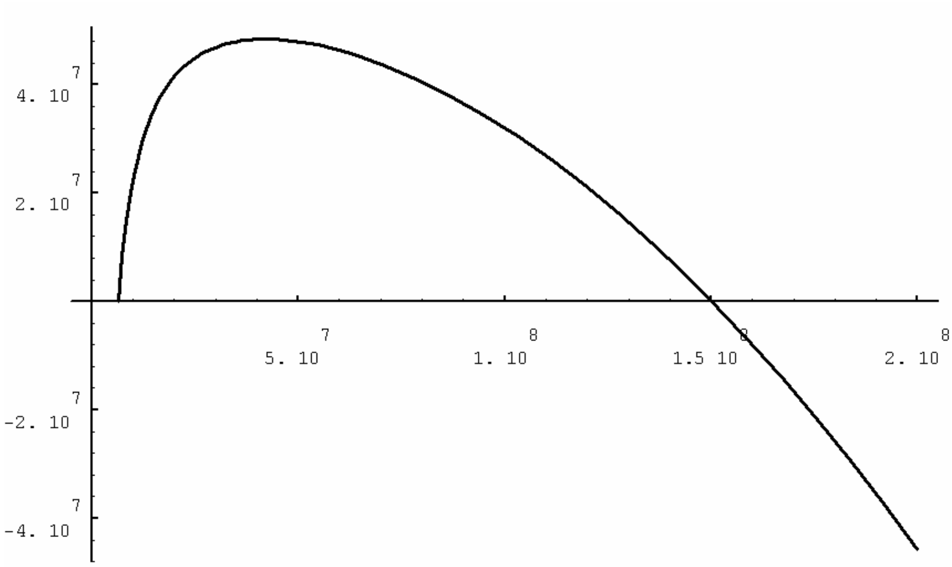


$$E_z = -\frac{m\omega^2 r^2}{2} + \frac{m\omega^2 R^2}{2}$$

Weiters gilt: $E_{pot} = E_r + E_z$

$$E_{pot} = \frac{GMm}{R} - \frac{GMm}{r} - \frac{m\omega^2 r^2}{2} + \frac{m\omega^2 R^2}{2}$$

Mit Hilfe dieser Formel können wir die potenzielle Energiedifferenz zwischen einem Punkt im Orbit und der Erdoberfläche, die auf einer rotierenden Geraden durch den Erdmittelpunkt liegen, unter Berücksichtigung der Fliehkraft berechnen.
Wenn wir die Gravitation, die Erdmasse und die Radien als gegeben betrachten und m gleich 1 setzen, erhalten wir die Energie, die wir aufwenden müssen um 1 kg von der Erdoberfläche zu einem Punkt im Abstand r im Orbit zu bringen. Um 1 kg von der Erde zum geostationären Orbit zu befördern benötigt man $4{,}84401 \cdot 10^7$ Nm (ca. 13,466 kWh).



Der Graph verdeutlicht, dass die potentielle Energie an der Erdoberfläche gleich Null ist. Die höchste potenzielle Energie steckt in der Masse die sich im Hochpunkt, dem geostationären Orbit, befindet.

Zusammenfassung

Wir stellten sehr bald fest, dass die meisten Aspekte, mit denen wir uns befassten, mit Physik zusammenhingen. Eigentlich konnte niemand von uns behaupten, Physik sei eines seiner Lieblingsfächer. Aus diesem Grund stießen wir sehr bald an unsere Grenzen. Wir mussten also oft auf die Hilfe unseres Betreuers Prof. Bernd Thaller zurückgreifen. Er schaffte es, uns den nötigen physikalischen Background so zu vermitteln, dass das Ganze auch noch interessant und spannend erschien. An dieser Stelle sei ihm auch ein großes Dankeschön ausgesprochen, vor allem für seine scheinbar endlose Geduld mit uns und seine immer freundliche, humorvolle Art.

Das Arbeiten in der Gruppe funktionierte bei uns sehr gut, wir teilten uns die Arbeit gut und gerecht auf, und alle bearbeiteten die Dinge, die sie am meisten interessierten. Das Zusammenfügen der gesammelten Ergebnisse gestaltete sich dann allerdings als aufwendig, da dann irgendwie niemand mehr wusste, was zu was gehörte oder woraus sich jene spezielle Formel noch einmal schnell ergab. Aber auch diese abschließende, etwas chaotische Phase meisterten wir. In unserer Gruppe gab es kaum Spannungen und alle beteiligten sich interessiert an den Diskussionen und Überlegungen zum Thema.

Was uns sehr zusagte war die Tatsache, dass wir uns unsere Arbeitszeiten eigentlich sehr frei einteilen durften. Dies war eine angenehme Weise des Arbeitens im Vergleich zum Schulalltag, der ja doch sehr durch die zeitliche Einschränkung beeinflusst wird.

Wir fanden es schade, dass der Tagesablauf sehr eintönig war und wir wenig Möglichkeit auf Abwechslung hatten.

Am Ende dieses Berichts fragten wir uns natürlich, ob solch ein Lift in den Weltraum wirklich realisierbar ist. Unsere Berechnungen und Studien der NASA zeigen, dass es theoretisch möglich wäre, die benötigten Materialien jedoch noch nicht vollständig entwickelt sind. Außerdem muss man bedenken, dass die vielen von uns genannten Probleme (z.B.: Weltraummüll, der das Kabel beschädigen könnte; die gefährlich hohe Teilchenstrahlung im Van-Allen-Gürtel) die Realisierung des Projekts weiter erschweren würden. Viele Wissenschaftler sind davon überzeugt, dass dieses Projekt nie umgesetzt wird. Auf der anderen Seite die teuren Studien der NASA und viele Physiker, die sich mit dem Problem genauestens auseinandergesetzt haben. Bei Recherchen im Internet findet man auf google.com ungefähr 599.000 Suchergebnisse für „space elevator".

<u>Bildnachweis:</u> Illustrationen aus dem NASA Bericht CP-2000-210429 und wikipedia.org

KRYPTOGRAPHIE

Florian Andritsch
Alexander Bors
Wolfgang Hrauda
Stefanie Kaiser
Paul Laufer
Florian Mikulik
unter der Betreuung von Prof. Günter Lettl

19. Januar 2008

Inhaltsverzeichnis

1 Einleitung

Florian Andritsch

Kryptogarphie ist im klassischen Sinn die Wissenschaft vom Verschlüsseln von Informationen. Heutzutage ist dieses Gebiet von höchster Wichtigkeit, um einen sicheren Transfer von Daten über das Internet zu ermöglichen. Egal ob man einen EInkauf über das Internet tätigt, das Angebot *eBanking* nutzt, oder sich lediglich auf einer Seite mit Benutzernamen und Passwort anmeldet, muss Information vom eigenen PC zum Webserver übertragen werden.

Gibt es nun einen sog. Angreifer, der versucht, die übermittelte Information herauszufinden, hätte dieser im Erfolgsfall die Möglichkeit, den Account ungehindert für den Eigengebrauch zu missbrauchen. Damit dies nur unter äußerst erschwerten Bedingungen möglich ist, bedient man sich bei der Übertragung verschiedener Verschlüsselungsmethoden, um im Idealfall eine 100% sichere Authentifikation eines Benutzers zu ermöglichen.

Dabei reicht es jedoch nicht das lediglich die zur Übermittlung bestimmten Daten zu verschlüsseln, denn ein Angreifer könnte trotzdem die übertragene Information "mithorchen", und diese zu seinem Nutzen verwenden. Man muss also auf eine bestimmte Weise verschlüsseln, und danach wieder entschlüsseln können.

Während unserer Arbeit im Laufe der Modellierungswoche 2008 analysierten wir verschiedenste Verschlüsselungs- und Übermittlungsvarianten auf ihre Funktion, Nutzbarkeit und Sicherheit. In vielen Stunden Arbeit, machten wir uns mit Zahlentheoretischen Hintergründen wie dem Rechnen mit Kongruenzen, der Eulerschen φ-Funktion, Primitivwurzeln sowie zahlreichen Schemata zum sicheren Informationsaustausch vertraut, um auf dem gewonnenen Wissen basierend eine eigene Methode zur sicheren Übertragun mit einem eigenen Verschlüsselungsalgorithmus zu entwickeln.

Durch die Arbeit an diesem Thema konnten wir reichlich Erfahrung im Umgang mit kryptographischen Methoden sammeln, und unseren mathematischen Horizont erweitern.

Seggauberg, Jänner 2008
Florian Andritsch im Namen der Gruppe "Kryptographie"

2 Über die Wahl eines sicheren Passwortes

Stefanie Kaiser

- Regeln für den sicheren Umgang mit Passwörtern:
  - Die Passwörter sollte man unbedingt für sich behalten!
  - Keine benutzerbezogenen Daten in Passwörtern.
  - Passwörter nicht an den Monitor kleben oder in Word Dateien speichern.
  - Keine trivialen Tastaturkombinationen verwenden.
  - Das Passwort sollte man nie für zwei Accounts benutzen.
  - Ein sicheres Passwort besteht sinnvollerweise aus Groß- und Kleinbuchstaben sowie aus Ziffern. Es sollte auch regelmäßig gewechselt werden.
  - Außerdem soll es keine Systematik beinhalten.
- Wie lang ein sicheres Passwort sein sollte:
  - Je nach Qualität eines Passworts variiert auch die Zeitdauer, die ein Hacker benötigt, um das Passwort zu entschlüsseln. Einige Beispiele:
    * Bei einem Passwort von 3 Zeichen und einer Zeichenklasse von 62 Zeichen ergeben sich 62^3 mögliche Kombinationen. Um die benötigte Zeit zu berechnen, dividiert man dies durch die Anzahl der Kombinationen, welche der Computer pro Sekunde entschlüsselt, und man erhält die benötigte Zeit in Sekunden.
    * Ein Passwort sollte aus mindestens 8 Stellen bestehen. Es ergeben sich 62^8 mögliche Kombinationen, was einer Rechenzeit von 6,9 Jahren entspricht.
    * Bei höheren Sicherheitsbereichen (z.B.: Firmen-Netze) sollte man die Passwörter auf mindestens 10 Zeichen erhöhen. Dies ergibt 62^{10} mögliche Kombinationen, was einer Rechenzeit von 26614 Jahren entspricht.

Mindestlänge	maximal benötigte Zeit (bei einer Zeichenklasse von 62 Zeichen und 1 Million Kombinationen pro Sekunde)
3 Zeichen	ca. 0,2 Sekunden
5 Zeichen	ca. 15 Minuten
8 Zeichen	ca. 7 Jahre
10 Zeichen	ca. 26600 Jahre
12 Zeichen	ca. 102 300 000 Jahre
15 Zeichen	ca. $2.4 * 10^{13}$ Jahre

Genaue Auflistung von der Berechnung der Dauer wie lange ein Hacker braucht, um ein Passwort zu entschlüsseln:

1. Die Anzahl der möglichen Kombinationen dividiert man: erstens durch 1000 000 (Anzahl der Kombinationen, welche der Computer pro Sekunde entschlüsselt), so erhält man die Dauer in Sekunden.
2. Zweitens dividiert man das Ergebnis durch 60, so erhält man die Dauer in Minute.
3. Drittens dividiert man das Ergebnis wiederum durch 60, so erhält man die Dauer in Stunden.

4. Als viertes dividiert man das Ergebnis durch 24, so erhält man die Dauer in Tagen und letztlich
5. dividiert man das Ergebnis durch 365, um die Dauer des Passwortentschlüsselns in Jahren zu erhalten.

Beispiel:
Die Berechnung der Dauer des Passwortentschlüsselns eines Passwortes, welches aus 10 Zeichen besteht:
→ Mögliche Kombinationen bei einer Zeichenklasse von 62 Zeichen:

$$\begin{aligned} 62^{10} &= 8.4 \cdot 10^{17} \\ \\ \frac{8.4 \cdot 10^{17}}{10^6} &= 8.4 \cdot 10^{11}\, Sekunden \\ \frac{8.4 \cdot 10^{11}}{60} &= 1.4 \cdot 10^{10}\, Minuten \\ \frac{1.4 \cdot 10^{10}}{60} &= 2.331 \cdot 10^{8}\, Stunden \\ \frac{2.331 \cdot 10^{8}}{24} &= 9714113\, Tage \\ \frac{9714113}{365} &= 26614\, Jahre \end{aligned}$$

Doch bei diesen Angaben wird die Maximalzeit berechnet, d.h. dass der Hacker das Passwort erst bei der letzten Zeichenkombination errät, was theoretisch auch früher möglich sein könnte. Außerdem wurden diese Zahlen mit 1 Million Kombinationen pro Sekunden berechnet, besser gebaute Rechner schaffen vielleicht das Millionenfache.

– Wie Hacker versuchen, Ihre Passwörter herauszufinden
    * Damit der Computer überprüfen kann, ob ein eingegebenes Passwort das Richtige ist, werden diese gespeichert. Um die Gefahr zu umgehen, dass jemand die Passwörter einfach ausliest, werden die Passwörter mit speziellen Kryptografiealgorithmen verschlüsselt.
    * Passwörter zu knacken, bedeutet in diesem Zusammenhang also Passwörter zu entschlüsseln. Die von Angreifern zu diesem Zweck verwendeten Programme heißen Passwort-Cracker und sind ohne Schwierigkeiten über das Internet zu beziehen. Die einfachste Art, ein Passwort zu entschlüsseln, ist alle möglichen Kombinationen von Buchstaben, Ziffern und Sonderzeichen durchzuprobieren.
    * Solch ein Angriff wird Brute-Force-Angriff genannt. Wegen der großen Menge von Zeichenkombinationen brauchen solche Programme aber viel Zeit, um ein Passwort zu entschlüsseln.
    * Deutlich effektiver sind dagegen Passwort-Cracker, die Wortlisten verwenden. Das Programm probiert nacheinander Begriffe aus der Wortliste, bis es das gesuchte Passwort gefunden hat.
    * Aus diesem Grund sollten auf keinen Fall Passwörter verwendet werden, die nur aus Kleinbuchstaben bestehen oder in Wortlisten auftauchen können.
– Die Gefahr durch knackbare Passwörter

  * Ist einmal ein Passwort entschlüsselt, eröffnet sich die Möglichkeit für den Cracker, sich unbefugt bei fremden Benutzerberechtigungen einzuloggen.
  * Der Cracker kann:
    - fremde Daten betrachten, verändern oder zerstören und auch in Ihrem Namen agieren. Z.B.: Beleidigungen oder Erpresserbriefe an Kollegen oder Freunde schicken, deren Adresse sich ihrer Mailbox entnehmen lässt.
    - Waren auf ihren Namen bestellen.
    - In Firmen-Netze eindringen.
    - Usw.

Benutzerauthentifikation

Grundsätzlich kann man einen Menschen an folgenden Merkmalen erkennen:

- Charakteristische Eigenschaften
- Besitz
- Wissen

Durch diese Punkte kann man Menschen identifizieren.

Diesen Mechanismus verwenden wir unbewusst sehr häufig in unserem täglichen Leben: Wir erkennen unsere Freunde an ihrem Aussehen, ihrer Stimme, ihrem Gang, usw.

In anderen Situationen werden auch Fingerabdrücke oder Ausweise (Grenzübergang) zur Identifikation herangezogen.

Bei der Authentifikation zwischen Mensch und Rechner sieht die Lage anders aus: Authentifikation durch Wissen ist der am einfachsten zu realisierende Mechanismus, Authentifikation durch Besitz ist ebenfalls möglich. Doch die Authentifikation durch Wissen ist ziemlich komplex und wird nur bei Anwendungen, die extrem hohe Sicherheit erfordern, eingesetzt.

Folgende Methoden bauen auf Wissen oder auf Besitz der zu authentifizierenden Person auf: Z.B.: beim Computer: Bei dieser Methode hat ein Benutzer ein Geheimnis, und der Rechner will sich davon überzeugen, dass die Person das ihr entsprechende Geheimnis hat.

Bei jeder Authentifikation durch Wissen geht es darum, dem Gegenüber nachzuweisen, ein bestimmtes Geheimnis zu kennen. Die verschiedenen Verfahren unterscheiden sich nur darin, wie dieser Nachweis erfolgt. Das Passwortverfahren ist das primitivste Authentikifationsverfahren.

→ Passwörter: Ein Passwort ist eine beliebige Folge von Zeichen (Buchstaben, Ziffern, Sonderzeichen), die das gemeinsame Geheimnis einer Person und des Rechners ist, d.h. im Idealfall kennen nur der Rechner und der jeweilige Benutzer das Passwort:

Der Rechner hat ein „Referenzpasswort" P0 gespeichert, das dem Benutzer bekannt ist. Der Benutzer tippt zur Authentifikation sein Passwort P ein, und der Rechner überprüft, ob P und P0 übereinstimmen. Wenn es eine Übereinstimmung gibt, wird der Benutzer zugelassen.

→ Erzeugen eines Passwortes: Zwischen Verifizierer und dem Benutzer wird ein geheimes Passwort vereinbart. → Verifizieren eines Passworts: Der Benutzer gibt seine Identität an und überträgt das Passwort. Der Verifizierer überprüft, ob dies das dem Benutzer zugeordnete Passwort ist.

Sicherheit beim Handy: → Challenge and Response: Der Verifizierer schickt dem Benutzer eine Zufallszahl. Dieser wendet darauf einen Authentifikationsalgorithmus unter einem geheimen Schlüssel an und schickt das Ergebnis RES (Response) zurück. Der Empfänger erhält dies und vergleicht den erhaltenen Wert mit dem von ihm berechneten. Er akzeptiert den Benutzer genau dann, wenn die beiden Werte übereinstimmen.

- Authentifikation: Der Netzbetreiber möchte sicher gehen, dass er seine Rechnungen an den richtigen Teilnehmer schickt. Mit anderen Worten: Er muss den anrufenden Teilnehmer zweifelsfrei identifizieren.

- Verschlüsselung: Die Teilnehmer möchten sich sicher sein, dass ihre Gespräche geheim bleiben. Diese Anforderung wurde beim GSM-System (Global System for Mobile Communication) gelöst: Der Netzbetreiber authentifiziert den Teilnehmer durch ein Challenge – and – Response – Protokoll. Ein Problem tritt auf, wenn der Teilnehmer in einem Fremdnetz eingebucht ist. Der Betreiber schickt mehrere Paare der Form (RES) auf Anforderung an den Betreiber des Fremdnetzes. Dieser führt mehrere Authentifikationsprotokolle mit dem Teilnehmer durch, kann aber keine neuen Paare (RES) erzeugen.

- ABER: die Authentifizierung ist nur einseitig: Der Netzbetreiber kann sich vor betrügerischen Teilnehmern schützen, ein Teilnehmer steht aber einem vorgespielten, betrügerischen Netz machtlos gegenüber. Dieses Problem wurde von UMTS (Universal Mobile Telecommunication Systems) behoben: dort ist eine zweiseitige Authentifikation vorgesehen.

- Alle Schlüssel, die für die kryptographischen Operationen verwendet werden, sind in der SIM (Subscriber Identity Module) gespeichert.

3 Kollisionswahrscheinlichkeit unter standardisierten Bedingungen als Maß für die Qualität von Hash-Funktionen

Alexander Bors, Stefanie Kaiser

Unter einer Hash-Funktion versteht man ganz allgemein eine Funktion, welche für eine relativ große Definitionsmenge über eine relativ kleine Wertemenge verfügt. Hash-Funktionen spielen v. a. in der Kryptologie eine große Rolle, speziell im Bereich der Benutzeridentifikation. Eine Person erhält dabei ein Passwort p aus einer vorgegebenen Menge (der Definitionsmenge der Hash-Funktion), über die Hash-Funktion wird deren Funktionswert $f(p)$ berechnet und einzig dieser Wert dem Computer im Zuge der Identifikation präsentiert.

Das Prinzip beherbergt Vor- und Nachteile: Durch die vielen Argumente mit selbem Funktionswert ist die Notwendigkeit umgangen, dem Computer die essentiellen Passwörter selbst in einer Liste mitzuteilen, welche von Hackern leicht geraubt und in weiterer Folge missbraucht werden können. Andererseits wird es Datendieben, welche prinzipiell ohne Diebstahl solcher Listen arbeiten, leichter gemacht, Zugang zu erhalten, da ja jeder gültige Funktionswert mehrere (eventuell sehr viele) entsprechende Argumente besitzt und womöglich noch mehrere solcher Funktionswerte (für verschiedene Benutzer) als gültig anerkannt werden.

Im Zuge eines Teils unseres Projektes haben wir verschiedene gängige Hash-Funktionen auf deren „Qualität", d. h. Sicherheit vor möglichem Eindringen eines Unbefugten in das System untersucht.

Dazu haben wir uns standardisierter Kriterien bedient: Die Definitionsmenge der Hash-Funktion bestehe stets aus allen natürlichen Zahlen von 0 bis 1000 (jeweils einschließlich), als Maßzahl für die Qualität der Funktion diene die Wahrscheinlichkeit, dass ein Unberechtigter, welcher zufällig einen gültigen Funktionswert kennt (wobei sich die Wahrscheinlichkeit für jeden der gültigen Funktionswerte an der Häufigkeit ihres Auftretens unter den Funktionswerten für alle möglichen Argumente orientieren soll) bei einmaligem Raten ein Argument „erwischt", das den jeweiligen Funktionswert liefert. Je geringer diese Wahrscheinlichkeit ist, desto besser ist die Funktion.

Beispiel

Eine Hashfunktion ordne jedem $x \in \mathbb{N}_{1000}$ eine letzte Ziffer zu. Z.B.: $153 \mapsto 3$, $115 \mapsto 5$, $g \mapsto 6, \ldots$

Man berechne die genau definierte Wahrscheinlichkeit zur Qualitätskontrolle!

Lösung:

1. Zunächst muss man überhaupt feststellen, welche Möglichkeiten für unterschiedliche Funktionswerte es gibt: Da es sich beim Funktionswert um eine einzige Ziffer handelt, kann er nur einen Wert $a \in \mathbb{N}_9$ annehmen (0,1,2,... 9).

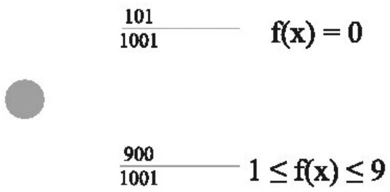
2. Nun beschäftigt man sich zunächst mit der Wahrscheinlichkeit, mit der jeder der zehn Fälle „Täter erfährt 0 als gültig.", „Täter erfährt 1 als gültig" usw. Dies entspricht dem Wert

$$\frac{Zahl\,der\,Argumente\,mit\,dem\,jeweiligen\,Funktionswert}{Gesamtzahl\,der\,Argumente (= 1001)}$$

Für 0 gibt es $\forall = 100x + 10y \in \mathbb{N}_{1000} : x, y \in \mathbb{N}_9$ und 1000 insgesamt 101 Argumente. Für $1 - 9$ gibt es jeweils $\forall = 100x + 10y + 1/2/3 \ldots : x, y \in \mathbb{N}_9$

insgesamt 100 Argumente, womit alle 1001 Elemente von $\mathbb{N}_{1000}$ abgedeckt werden (Probe-Betrachtung)

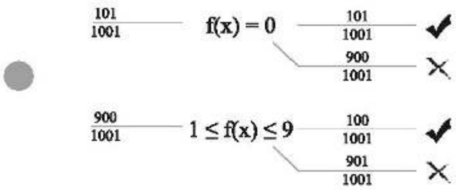
3. Als Wahrscheinlichkeitsbaum lassen sich die bisherigen Ergebnisse so zusammenfassen:



4. Und nun berechnet man für jeden Fall jeweils die Wahrscheinlichkeit, dass der Täter aus der Gesamtmenge von 1001 Argumenten eines zufällig aussucht, das den jeweiligen Funktionswert liefert.

   - Für 0: $\frac{101}{1001}$
   - Für 1-9: $\frac{100}{1001}$

5. Wahrscheinlichkeitsbaum:



   „Erfolg“ ist aus unserer Sicht negativ, da es ja bedeutet, dass der Unbefugte Zugang erhalten hat. Nun interessiert einzig die Gesamt-Wahrscheinlichkeit für einen Erfolg, nicht nur für die jeweiligen Funktionswerte.

$$p = \frac{101^2}{1001^2} + \frac{900 \cdot 100}{1001^2} = \frac{100201}{1002001} \approx 0.10000090 \approx 10\%$$

Erweiterung der Kennzahl

Die alleinige Angabe der K-Wahrscheinlichkeit bei einer Hashfunktion scheint nicht auszureichen, und sei vollständig zu charakterisieren. Besser ist es, noch eine weitere Zahl dazu zu nehmen, die ebenfalls ein wichtiges Kriterium für die „Qualität“ von Hashfunktionen ist: die Varianz für jene Urliste, die sich aus der Anzahl an Argumenten pro Funktionswert zusammensetzt; sie steht als Maßzahl für die gleichmäßige Aufteilung der Argumente auf jeweilige f-Werte und sollte auch möglichst klein sein. Der ideale Fall für pK tritt ein, wenn für $\mathbb{N}_{1000}$ jedes der 1001 Argumente einen anderen f-Wert erhält, dann ist

$$pK = 1001 * \frac{1}{1001}^{2} = \frac{1}{1001}$$

Kleiner kann pK unter den gegebenen Standard-Bedingungen nicht werden. Interessanterweise liefert dieser Fall auch für die Varianz V den bestmöglichen Wert, nämlich D, da ja dann jedes $Argument - Anzahl$ gleich dem Durchschnitt aller Argumente Zahlen (1) ist, und damit $V = 0$ wird.

V: Urliste mit 9x100, 1x101

$$\begin{aligned}
\bar{\mathrm{x}} &= \frac{9 \cdot 100 + 1 \cdot 101}{10} = \frac{1001}{10} \\
v &= \frac{(101 - \frac{1001}{10})^2 + 9 \cdot (100 - \frac{1001}{10})^2}{10} = 0.09
\end{aligned}$$

Weiters haben wir folgende Beispiele untersucht

- f ordne jeder Zahl eine zweistellige Zahl zu, deren erste Ziffer die erste Ziffer des Arguments, die zweite Ziffer die letzte Ziffer des Arguments sei. Einstellige Argumente erhalten beim Funktionswert als erste Ziffer 0 und als zweite Ziffer sich selbst zugeordnet.
- Quersummenfunktion
- Divisions-Rest-Methode
- Multiplikative Methode
- Mittquadratmethode

Dabei sind wir, in Anlehnung an das erste Beispiel, stets nach folgendem Schema vorgegangen:

1. Bestimmung aller möglichen Funktionswerte für die vorgegebene Definitionsmenge.
2. Berechnung der jeweiligen Wahrscheinlichkeit als Quotient aus Zahl der Argumente des jeweiligen Funktionswertes und Gesamtzahl der Argumente (1001)
3. Eventuell Zusammenfassung ähnlicher Äste
4. Berechnung der Wahrscheinlichkeit, dass ein Täter zufällig ein Argument „erwischt“, das den jeweiligen Funktionswert liefert.
5. Berechnung der gekoppelten Wahrscheinlichkeit.
6. Aufsummierungen der einzelnen Wahrscheinlichkeiten für Erfolg des Täters, als Ergebnis Gesamt-Wahrscheinlichkeit (Kennzahl)

ad Beispiel 1

1. mögliche Funktionswerte: Es handelt sich um zweiziffrige Zahlen, welche von $0 - 99$ alle Werte annehmen können (100 mögliche Funktionswerte)
2. Gewichtung der einzelnen Funktionswerte: Gemäß der Definition der Funktion entspricht $0 \leq f(x) \leq 9$ der f-Wert selbst (eindeutige Zuordnung, 1 Möglichkeit) Dem f-Wert 10 entspricht 10 selbst, andererseits auch 1000 und alle Zahlen der Form 1x0 ($x \in \mathbb{N}_9, 10$ Möglichkeiten), insgesamt also 12 Möglichkeiten. Alle anderen f-Werte ($11 \leq f(x) \leq 99$) besitzen je 11 Möglichkeiten.

   z.B.: 56: 56 selbst (1 Möglichkeit), sowie alle Zahlen der Form $5x6$ ($x \in \mathbb{N}_9, 10$ Möglichkeiten)

3. Einzelne Wahrscheinlichkeiten für zufälliges Finden:

   $0 \leq f(x) \leq 9 : p_1 = \frac{1}{1001}$

   $f(x) = 10 : p_2 = \frac{12}{1001}$

   $11 \leq f(x) \leq 99 : p_3 = \frac{11}{1001}$

4. Varianzberechnung:

   Urliste (Argumentzahl): 10x1, 1x12, 89x11 (10 f-Werte haben ein Argument, ein f-Wert hat 12 Argumente, 89 f-Werte haben 11 Argumente)

$$\begin{aligned} \bar{\mathrm{x}} &= \frac{10 \cdot 1 + 1 \cdot 12 + 89 \cdot 11}{100} \\ \bar{\mathrm{x}} &= 10.01 \end{aligned}$$

$$\begin{aligned} v &= \frac{10 \cdot (1 - 10.01)^2 + 1 \cdot (12 - 10.01)^2 + 89 \cdot (11 - 10.01)^2}{100} \\ v &= 9.0299 \end{aligned}$$

$$\begin{array}{llll} 1.Beispiel: & pK = 0.0109 & 2.Vorbeispiel(letzte\,Ziffer): & pK = 0.1000 \\ & V = 9.0299 & & v = 0.09 \end{array}$$

ad Beispiel 2

Die Funktionswerte können zwischen 0 (Argument 0) und 27 (Argument 999) liegen. Das Auffinden der jeweiligen Anzahl an Argumenten gestaltet sich bei rein empirischem Arbeiten schwierig. Um dieses Problem zu umgehen, haben wir uns des Programms Excel (Tabellenkalkulation) bedient. Zuerst haben wir dazu eine Formel für die Quersumme von höchstens vierstelligen natürlichen Zahlen entwickelt. Es gilt:

Tausenderziffer $t(x) = \lfloor \frac{x}{1000} \rfloor$

Hunderterziffer $h(x) = \lfloor \frac{x}{100} \rfloor - \lfloor \frac{x}{1000} \rfloor \cdot 10$

Zehnerziffer $z(x) = \lfloor \frac{x}{10} \rfloor - \lfloor \frac{x}{100} \rfloor \cdot 10$

Einerziffer $e(x) = x - \lfloor \frac{x}{10} \rfloor \cdot 10$

Dadurch ergibt sich für die Ziffernsumme:

$$\begin{aligned} \sum Z(x) &= t(x) + h(x) + z(x) + e(x) = \\ &= \left\lfloor \frac{x}{1000} \right\rfloor + \left\lfloor \frac{x}{100} \right\rfloor - \left\lfloor \frac{x}{1000} \right\rfloor \cdot 10 + \left\lfloor \frac{x}{10} \right\rfloor - \left\lfloor \frac{x}{100} \right\rfloor \cdot 10 + x - 10 \cdot \left\lfloor \frac{x}{10} \right\rfloor = \\ &= -9 \cdot \left(\left\lfloor \frac{x}{1000} \right\rfloor + \left\lfloor \frac{x}{100} \right\rfloor + \left\lfloor \frac{x}{10} \right\rfloor \right) + x \end{aligned}$$

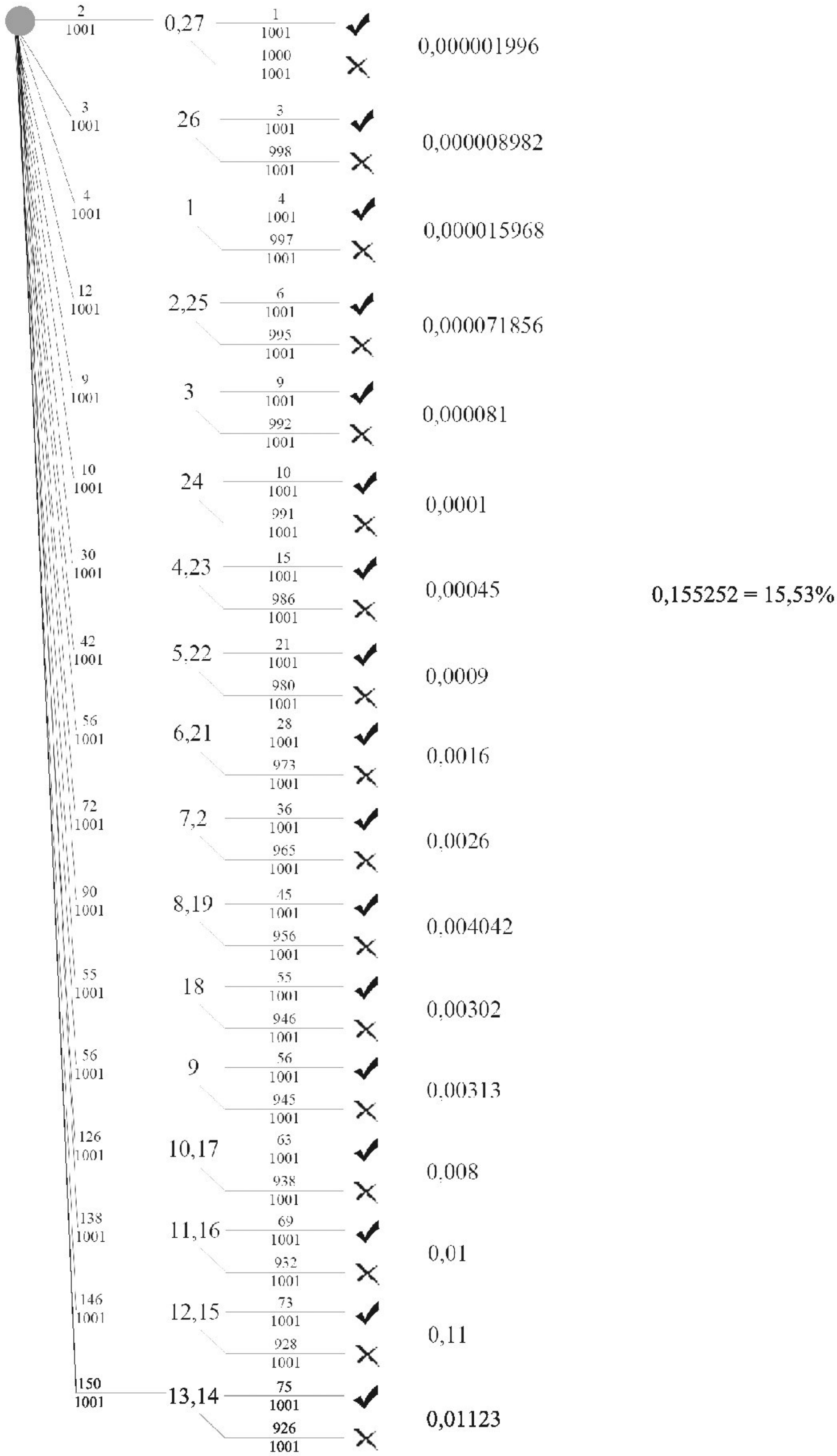
Diese Formel haben wir in Excel eingegeben und auf alle natürlichen Zahlen zwischen 0 und 1000 angewendet. Anschließend haben wir die so entstehende Liste von Funktionswerten der Größe nach geordnet und die jeweilige Zahl an Argumenten abgewählt.

Für Excel ist jeweils $\lfloor argument \rfloor$ durch GANZZAHL(argument) zu schreiben.

Folgendes Ergebnis:

0 :	1
1 :	4
2 :	6
3 :	9
4 :	15
5 :	21
6 :	28
7 :	36
8 :	45
9 :	56
10 :	63
11 :	69
12 :	73
13 :	75
14 :	75
15 :	73
16 :	69
17 :	63
18 :	55
19 :	45
20 :	36
21 :	28
22 :	21
23 :	15
24 :	10
25 :	6
26 :	3
27 :	1
	–
	1001

Wahrscheinlichkeitsbaum:



Varianz-Berechnung: → Argumentanzahl:
2 mit 1 Argument
1 mit 3 Argumenten
1 mit 4 Argumenten
2 mit 6 Argumenten
1 mit 9 Argumenten

1 mit 10 Argumenten
2 mit 15 Argumenten
2 mit 21 Argumenten
2 mit 28 Argumenten
2 mit 36 Argumenten
2 mit 45 Argumenten
1 mit 55 Argumenten
1 mit 56 Argumenten
2 mit 63 Argumenten
2 mit 69 Argumenten
2 mit 73 Argumenten
2 mit 75 Argumenten

$$\begin{aligned}
\bar{\mathrm{x}} &= \frac{2\cdot 1 + 1\cdot 3 + 1\cdot 4 + \cdots + 2\cdot 75}{28} = \frac{1001}{28} = 35.75\% \\
v &= \frac{2\cdot(1-35.75)^2 + 1\cdot(3-35.75)^2 + 1\cdot(4-35.75)^2 + \cdots + 2\cdot(75-35.75)^2}{28} \\
v &= 698.7589
\end{aligned}$$

ad Beispiel 3

$f : \mathbb{N}_{1000} \mapsto ?$

$f : x \mapsto x \, mod \, m \ \ (m \in \mathbb{N}\backslash\{0\})$

Fallunterscheidung: $m \geq 1001$: Jeder Zahl wird sie selbst zugeordnet. $m \leq 1000$: $\mathrm{W}_f = \mathbb{N}_{(m-1)}$

bis $(m-1)$ wird jeder Zahl sie selbst zugeordnet, ab m beginnt die Zuordnung wieder von 0 weg.

z.B.: bei $m = 13$ gibt es insgesamt $\lfloor\frac{1001}{12}\rfloor = 83$ "volle" Durchgänge, $1001 \equiv 5 \, mod \, 12$ zusätzliche Zahlen (996-1000) bleiben übrig, sie liefern die Werte 1 bis 5.

Verallgemeinerung: $\lfloor\frac{1001}{m}\rfloor$ volle Durchgänge. $1001 - m \cdot \lfloor\frac{1001}{m}\rfloor$ bleiben an Zahlen übrig (bis zu dieser Zahl, die zwischen 0 und $(m-1)$ liegt, bei 0 beginnend, kommt zu den f-Werten noch ein Argument dazu).

d.h. wenn ein n-Wert bei gegebenem $m < 1001 - m \cdot \lfloor\frac{1001}{m}\rfloor$ ist, besitzt es $\lfloor\frac{1001}{m}\rfloor + 1$ Argumente, liegt er zwischen $1001 - m \cdot \lfloor\frac{1001}{m}\rfloor$ und m (der erste ist hiebei eingeschlossen, der zweite Wert ausgeschlossen), besitzt er $\lfloor\frac{1001}{m}\rfloor$ Argumente, ist er dagegen $\geq m$, so besitzt er 0 Argumente (da bei der Ganzzahldivision durch m der Rest nicht den Wert $(m-1)$ übersteigen kann.) Diese Argumente (günstige Fälle) werden für die Wahrscheinlichkeit, dass ein Täter den jeweiligen Funktionswert erfährt, durch die Gesamtzahl an Argumenten (stets 1001) dividiert. Um die Kollisionswahrscheinlichkeit im jeweiligen Fall zu erhalten, wird diese Wahrscheinlichkeit mit sich selbst multipliziert (quadriert), diese Summe dieser einzelnen K-Wahrscheinlichkeiten ergibt die Gesamt-K- Wahrscheinlichkeit, die gesucht ist.

Da die Durchführung händisch sehr lange dauert (es gibt im Bereich $m \leq 1000$ 1000 verschiedene m-Werte und damit m k-Wahrscheinlichkeiten mit jeweils 1001 Summanden, sofern Nullen dazugezählt werden), bietet sich zur Lösung wieder Excel an:

Man bildet eine Tabelle, deren äußerste linke Spalte die 1000 möglichen m-Werte aufzählt, die oberste Spalte zählt die 1001 n-Werte, die als f-Werte in Frage kommen, auf. Der Krenzungspunkt bzw. die (Kreuzungszelle) einer m-Zeile und n-Spalte enthält die jeweilige K-Wahrscheinlichkeit des f-Wertes n mit dem Modul m. Diese Werte werden über eine doppelte WENN Funktion allgemein beschrieben.

WENN (jeweiliger n-Spaltenwert $\leq 1001-$ jeweiliger m-Zeilenwert $\cdot \lfloor \frac{1001}{m-zw} \rfloor$; Siehe Zettel

Nun werden alle einzelnen K-Wahrscheinlichkeiten einer Zeile summiert, um die Gesamt- K-Wahrscheinlichkeit der jeweiligen Zeile (des jeweiligen m-Wertes) verhalten. Auch hier lässt sich eine allgemeine Summenformel verfassen und auf alle Zeilen kopieren. Es fällt auf, dass die K-Wahrscheinlichkeit umso kleiner (die Hashfunktion umso effektiver) wird, je größer m wird. Betrachtet man allerdings das (einheitliche) Schema aller m-Werte ≥ 1001, so zeigt dieses die geringste Wahrscheinlichkeit mit 0.000999 entspricht 0.0999001%.

Divisionsrest – Varianz ,

Da beim Divisionsrest erstmals ein Parameter m vorkommt, muss auch die Varianz stets in Bezug auf einen Wert für m angegeben werden.

Zuerst stellt man fest, wie die Verteilung der Argumente pro f-Wert für die einzelnen m-Werte aussieht. Hierzu kommen sich der Formeln $\lfloor \frac{1001}{m} \rfloor + 1$ für $< 1001 - m \cdot \lfloor \frac{1001}{m} \rfloor$, $\lfloor \frac{1001}{m} \rfloor$ für $1001 - m \cdot \lfloor \frac{1001}{m} \rfloor \leq n < m$ und 0 für $n \geq m$ bedienen, wenn man mit n den jeweiligen Funktionswert bezeichnet ($n \in \mathbb{N}_{1000}$). Anschließend ist die Zahl der verschiedenen f-Werte, die Argumente (mindestens 1) besitzen, zu ermitteln. Sie entspricht jeweils dem m-Wert, da bei der Division durch m die m Reste $0, 1, 2 \ldots (m-1)$ auftreten. Erst ab $m = 1001$ gilt einheitlich der Wert 1001 (also auch für $m = 1002, 1003 \ldots$).

Zur Berechnung der Varianz benötigt man noch den Durchschnitt der jeweiligen Argumentzahl - Urlisten, den man berechnet, indem man 1001 (die stets konstante Summe der Urlisten) durch m (die Zahl der Werte) dividiert.

Anschließend berechnet man die Varianz wie folgt: Man legt eine zweite Tabelle auf demselben Blatt an, die aus gleich vielen Zeilen und Spalten besteht wie die erste. In jede Zelle dieser Tabelle kommt entweder 0 (wenn die entsprechende Zelle in der ersten Tabelle 0 enthält, es sich also um einen f-Wert ohne Argumente handelt) oder der Wert der entsprechenden Zelle der ersten Tabelle abzüglich des Durchschnittswertes für die jeweilige Zeile, diese Differenz zum Quadrat und auch die Zahl m als Anzahl von f-Werten mit mindestens einem Argument dividiert. Addiert man nun die Werte einer Zeile, so erhält man die Varianz für diese Zeile.

Es fällt auf, dass die Varianz weder monoton steigt noch fällt, sondern mehrere Zyklen durchläuft.

Für folgende m-Werte wird sie dabei 0: $1, 7, 11, 13, 77, 91, 143$ Sowie $\forall m \geq 1001$ Nichtsdestotrotz bleibt wohl letzterer Fall der beste, da er zusätzlich zur minimalen Varianz auch über minimale K-Wahrscheinlichkeit $\frac{1}{1001}$ verfügt.

ad Beispiel 4: Multiplikative Methode

Die entsprechende Hash-Funktion ist definiert als

$$f(x) = \lfloor m \cdot (x \cdot \varphi - \lfloor x \cdot \varphi \rfloor) \rfloor$$

dabei ist die Zahl φ beliebig wählbar, empfohlen wird aber, um eine möglichst gleichmäßige Verteilung der 1001 Argumente auf die möglichen Funktionswerte zu erreichen,
mäßige Verteilung der 1001 Argumente auf die möglichen Funktionswerte zu erreichen,
(Formel siehe Zettel Multiplikative Methode) zu wählen. m bezeichnet die (ebenfalls frei wählbare, im praktischen Gebrauch aber auf eine bestimmte Menge beschränkte) Zahl an Hashtabellenadressen.

Beispiel: $m = 25$, $\varphi = \frac{\sqrt{5}-1}{2}$,

$$f(2) = \left\lfloor 25 \cdot \left(2 \cdot \frac{\sqrt{5}-1}{2} - \left\lfloor 2 \cdot \frac{\sqrt{5}-1}{2} \right\rfloor \right) \right\rfloor = 5$$

Wie man sieht, werden die einzelnen Berechnungen sehr aufwändig, und da man nicht nur die f-Werte aller $x \in \mathbb{N}_{1000}$ für eine konkrete Funktion, sondern auch für möglichst viele m-Werte berechnen will, bedient man sich - wieder einmal - des geliebten Excels! Als φ wählt man (siehe Zettel), m lässt man bis 1000 zu. Das Tabellenblatt ist dabei in zwei Teile unterteilt. Im ersten werden, wie auch schon bei der Divisionsrestmethode, die einzelnen f-Werte für die zwei Parameter (m und x) berechnet (dazu wird einfach der Funktionsterm in eine Zelle als Formel eingegeben und die Formel dann kopiert - absolute Zellbezüge nicht vergessen!) Anschließend ermittelt man mit der „Zählenwenn“ - Funktion die absoluten Häufigkeiten der Argumente für die jeweiligen f-Werte (die theoretisch bis 1000 wachsen können), was den zweiten Teil des Tabellenblattes ausmacht. Anschließend ermittelt man die einzelnen K-Wahrscheinlichkeiten, indem man den mit der „Zählenwenn“ – Funktion gefundenen Wert an günstigen Argumenten durch 1001 (die Gesamtzahl an Argumenten) dividiert und diese Zahl dann quadriert (=mit sich selbst multipliziert). Wieder ergibt die Summe der einzelnen K-Wahrscheinlichkeiten für einen m-Fall die Gesamt - K-Wahrscheinlichkeit. Lässt man - wie vereinbart - nur $m \leq 1000$ zu, so wird der minimale - und damit beste - Gesamt - K-Wahrscheinlichkeitswert für $m = 1000$ erreicht, und zwar mit 0.001218562 entspricht 0.122%. Dennoch verhält sich die K-Wahrscheinlichkeit hinsichtlich des m-Wertes nicht streng monoton fallend, wie manche Werte zeigen, wenngleich auch die allgemeine Tendenz nach unten geht.

Multiplikative - Varianz

Auch hier wird sehr ähnlich vorgegangen wie bei den bisherigen Varianz - Berechnungen in Excel. Zuerst wird die Zahl der Argumente pro potentiellen F-Wert mit der „Zählenwenn“-Funktion ermittelt, anschließend die Zahl der f-Werte mit mindestens einem Argument und aus diesen der Durchschnitt der sich aus den Argumentzahlen zusammensetzenden Urlisten. Dann wird wieder der „Beitrag“ jedes f-Wertes für die einzelnen m-Werte zur Varianz berechnet (wie zuvor) und diese Zahlen aufsummiert.

Es fällt auf, dass sich die Varianz bei der multiplikativen Methode noch chaotischer verhält als die K-Wahrscheinlichkeit: Mal steigt sie, dann fällt sie wieder stark, was es natürlich sehr schwer macht, den besten sprich minimalen v-Wert zu finden. Tatsächlich besitzen $m = 0$ und $m = 1$ des optimalen v-Wert, dafür aber zugleich die schlechteste K-Wahrscheinlichkeit mit jeweils 1.

ad Beispiel 5: Mittquadrat – Methode

Bei der Mittquadrat- Methode wird eine Zahl $x \in \mathbb{N}$ quadriert und von der Quadratzahl die erste und letzte Ziffer entfernt. So erhält man den f-Wert.

Beispiel: $36 \mapsto 29$, da $36^2 = 1296$, nach Entfernung der Ziffern 1 und 6 bleibt 29 übrig. Wählt man wie gewohnt $\mathbb{N}_{1000}$ als D, so kommen als f-Werte alle natürlichen Zahlen bis 9999 in Frage (1000 besitzt zwar ein siebenstelliges Quadrat, doch wird ihm nur 0 zugeordnet, ab 317 werden die Quadrate und die f-Werte damit im Allgemeinen vierstellig). Tatsächlich aber stellen viele Zahlen aus $\mathbb{N}_{9999}$ keine f-Werte der gegebenen Funktion dar, wie Gesamtbetrachtungen zeigen.

Dies führt man wieder mit Hilfe von Excel durch. Wir sind dabei so vorgegangen:

Zuerst haben wir die Argumente 0-1000 in eine Spalte eingetragen, in die Zelle rechts daneben kann jeweils das Quadrat des Arguments. Anschließend haben wir Formeln für die Gewinnung des Funktionswertes aus den verschieden- stelligen Quadraten der Argumente entwickelt:

- Für 1-, und 2-stellige Quadrate: 0
- Für 3-stellige Quadrate: $\lfloor\frac{x}{10}\rfloor - 10 \cdot \lfloor\frac{x}{100}\rfloor$ (x bezeichnet das Quadrat des jeweiligen Argumentes)
- Für 4-stellige Quadrate: $\lfloor\frac{x}{10}\rfloor - 100 \cdot \lfloor\frac{x}{1000}\rfloor$
- Für 5-stellige Quadrate: $\lfloor\frac{x}{10}\rfloor - 1000 \cdot \lfloor\frac{x}{10000}\rfloor$

Die Formeln haben wir jeweils auf alle Zeilen mit entsprechenden Quadraten übertragen.

Dann haben wir über die „Zählenwenn“ – Funktion des Programms die Anzahl von Argumenten pro Funktionswert ermittelt und daraus die jeweiligen K-Wahrscheinlichkeiten berechnet. Die Aufsummierung aller K-Wahrscheinlichkeiten ergibt die Gesamt-K-Wahrscheinlichkeit: 0.001767463 entspricht 0.176746331%.

Mittquadrat-Varianz

Die „Zählenwenn“ - Funktion wurde bereits zuvor weingesetzt, d.h. man kann die entsprechenden Informationen über Argumente pro f-Wert gleich verwenden. Es wird sehr ähnlich wie zuvor vorgegangen: Zuerst bestimmt man überhaupt die Zahl an f-Werten, die über Argumente verfügen, errechnet den Durchschnitt als Quotient aus der Summe der Werte der „Zählenwenn“ - Funktion und der Zahl an f-Werten mit mindestens einem Argument. Anschließend bestimmt man den „Betrag“ jedes potentiellen f-Wertes zur Varianz, der entweder 0 ist, wenn der f-Wert über keine Argumente verfügt, oder sich aus der Formel (siehe Zettel Mittquadrat-Varianz) errechnet. Die Summe dieser Werte liefert die Varianz: 712025.096 was einen sehr hohen Wert darstellt.

Grenz-Wahrscheinlichkeit

Lässt man bei einer Hashfunktion, deren Definitionsmenge $Df \subset \mathbb{N}$, genauer $Df = \mathbb{N}_n = \{0, 1, 2, \ldots, n-1; n\}$, n als variabel stehen, so wird auch die Wahrscheinlichkeit des Täters, einen bestimmten f-Wert $\in Wf$ zu erhalten, zu einer Funktion von n $p(n)$. Lässt man nun $n \to \infty$ gehen und erhält dabei einen Grenzwert $\lim_{n\to\infty} p(n) \in [0, 1]$, so spricht man auch von der Grenzwahrscheinlichkeit p_{lim}.

Beispiel:

f:$\mathbb{N}_n \to \mathbb{N}_9$

f: $x \to x \; mod \, 10$ („Letzte-Stellen-Hashfunktion“)

Hier lässt sich für alle Ziffern (Wertemenge) ein einheitliches Schema zur Berechnung erkennen. Es gilt: Die Zahl „ohne“ ihre Einerziffer, d.h. der Wert $\lfloor\frac{x}{10}\rfloor$ liefert ein ungefähres Maß dafür, wie viele Argumente jede Ziffer hat. Zusätzlich ist die Einerziffer von n zu betrachten: Alle Ziffern, welche kleiner oder gleich dieser Ziffer sind, haben noch ein zusätzliches Argument, also $\lfloor\frac{n}{10}\rfloor + 1$ Argumente.

D.h., die Wahrscheinlichkeit im Abhängigkeit von n schwankt für jede Ziffer zwischen (siehe Zettel „Grenz-Wahrscheinlichkeit“). Betrachtet man den Verlauf der „Wahrscheinlichkeits-Funktion“ genau, so fällt auf, dass sie an bestimmten Stellen nach oben „spring“ (Sprungstellen), und zwar für jede Argumente, bei denen die Einerziffer mit der jeweiligen Ziffer, die betrachtet wird, übereinstimmt.

Die erste „Sprungstelle“ von 0 liegt also schon bei $n = 0$, die zweite bei n=10, die dritte bei $n = 20$ usw. Allgemein gilt für die Ziffer 2, dass ihre k-te Sprungstelle ($k = 1, 2, 3, \dots$) bei $n = 10 \cdot (k - 1) + z$ liegt.

Zwischen diesen Sprungstellen verhält sie sich wie eine Funktion der Form $\frac{c}{x}$, $c \in \mathbb{N}$, und zwar für das Intervall $[0; z[$ wie $\frac{0}{x} = 0$, für das Intervall $[z; 10 + z[$ wie $\frac{1}{x}$ usw. Das k-te solche Intervall besitzt also auch die k-te Sprungstelle als obere (ausgeschlossene) Intervallgrenze. An direkten Sprungstelle hätte sie (nach der “alten”, im Intervall zuvor gültigen Funktion) theoretisch den Wert. (siehe Rückseite „Grenz-Wahrscheinlichkeit“)

Der theoretische Wert liegt stets unter $\frac{1}{10}$: siehe Zettel

Der praktische Wert liegt stets über $\frac{1}{10}$:

Dabei nähern sich aber sowohl theoretische, als auch praktische Werte immer mehr $\frac{1}{10}$ an: (siehe Zettel)

Der Funktionswert weicht “nach unten” also maximal um die kleiner werdende Differenz aus $\frac{1}{10}$ und theoretischem Wert ab, nach den maximal um die (ebenfalls gegen 0 stehende) Differenz aus praktischem Wert und $\frac{1}{10}$, und das ab der k-ten Sprungstelle. Für jedes noch so kleine $\epsilon \in \mathbb{R}^+$ liegt der Funktionswert also ab einem gewissen Argument n (das ja indirekt das k bedingt) immer in der ϵ-Umgebung um $\frac{1}{10}$, was der Definition des Grenzwertes entspricht. Also: Für alle Ziffern gilt: $plim = \frac{1}{10}$

ZUSAMMENFASSUNG

letzte Ziffer:	$p = 0.1000$ $v = 0.09$
erste und	$p = 0.0109$
letzte Ziffer:	$v = 9.0299$
Quersumme:	$p = 0.1553$ $v = 698.7589$
Divisionsrest:	bester Fall für $m \geq 1001$ mit $p = 0.000999$ v=0
Mittquadrat Methode:	$p = 0.0018$ $v = 712025.096$
Multiplikative Methode:	sowohl p als auch v verhalten sich stark chaotisch, Lösung ist deswegen schwer zu finden!

4 Rechnen mit Kongruenzen

Florian Andritsch

4.1 Rechnen modulo 37

zu beachten: $37 \in \mathbf{P}$, 37 ist eine Primzahl

Es werden nur die Reste jeder Zahl bei Division durch 37 behandelt. Zahlen die den selben Rest bei Division durch 37 lassen, werden als *kongruent* bezeichnet, befinden sich also in der selben *Restklasse*.

Bsp1:

$$\begin{aligned} ((17+10)\cdot 2 - 3)\cdot 11 &\equiv \ldots (37) \\ 51 \cdot 11 &\equiv 561\,(37) \\ 561 &\equiv 6\,(37) \end{aligned}$$

Bsp2:

$$\begin{aligned} 2 \cdot X &\equiv 19\,(37) \\ 2 \cdot X &\equiv -18\,(37) \\ X &\equiv -9\,(37) \end{aligned}$$

Satz: Für $a \ldots$ Zahl mit $37 \nmid a : \exists$ Zahl b mit $a \cdot b \equiv 1(37)$

Beweis : $\mathrm{ggT}(a, 37) = 1$

Euklidscher Algorithmus:

$\exists x, y \in \mathbf{Z} : a \cdot x + 37 \cdot y = 1$

$\square$

4.2 Modulo 22

Restklassen: 0, 1, 2, ..., 21

Bei Multiplikation mit einem Teiler von 22 treten nicht mehr alle Reste auf, sondern nur mehr ein paar Reste, diese aber mehrfach.

Satz: $\forall a \, mit \, \mathrm{ggT}(a, 22) = 1$ gilt:

$$\exists b : a \cdot b \equiv 1\,(37)$$

Es gibt also zu jedem Modul ein Multiplikativ-Inverses zu jeder Zahl, die teilerfremd zum betrachtetetn Modul ist.

5 $\oplus$-XOR-exklusives oder

Florian Andritsch

Diese logische Verknüpfung ist eine besonders wichtige im Bereich der Kryptographie. Besonders häufig wird sie überall dort verwendet, wo in Binärdarstellung gerechnet wird. Dieses Kapitel soll die Fukntionsweise erläutern, damit in den darauf folgenden Kapiteln keine Verwirrung auftkommt.

Wir wollen die Funktion anhand zweier 8 Bit langer Zahlen erläutern

$$01000111 \oplus 10101101 =?$$

es stellt sich als sinnvoll heraus, die Schreibweise von nebeneinander auf übereinander zu ändern:

$$\begin{array}{ll} 01000111 & \\ 10101101 & \oplus \\ \hline & \\ 11101010 & \end{array}$$

Die Funktion lässt nur entweder das eine oder das andere Bit 1 sein. Sind beide gleich 0, oder beide gleich 1, so wird das dazugehörige Bit auf 0 gesetzt, ist nur ein Bit gleich 1, das andere gleich 0, wird das entsprechende im Ergebnis auf 1 gesetzt.

Es ist klar, dass diese Funktion auf Bitketten beliebiger Länge angewendet werden kann.

6 Bitshift

Unter dem Begriff “Bitshift” ist ganz einfach eine Verschiebung alles Bis um einen gewissen Wert. Jene Bits, die an STellen verschoben werden, die über die Länge der Kette hinausgehen, werden am anderen Ende angestückelt:

$$Bitshift(10101101, 3) = 01101 - 101$$

Die ersten drei Bits (1-0-1) werden hinten angehängt, die anderen Rücken jeweils 3 Stellen nach vorne.

Wir werden in weiterer Folge Bitshifts desöfteren verwenden, um verschiedene Schlüssel aus ein und der selben Ausgangs-Bitkette zu erzeugen, wie es auch in öffentlich verwendeten Algorithmen durchgeführt wird.

7 DES - Data Encryption Standard

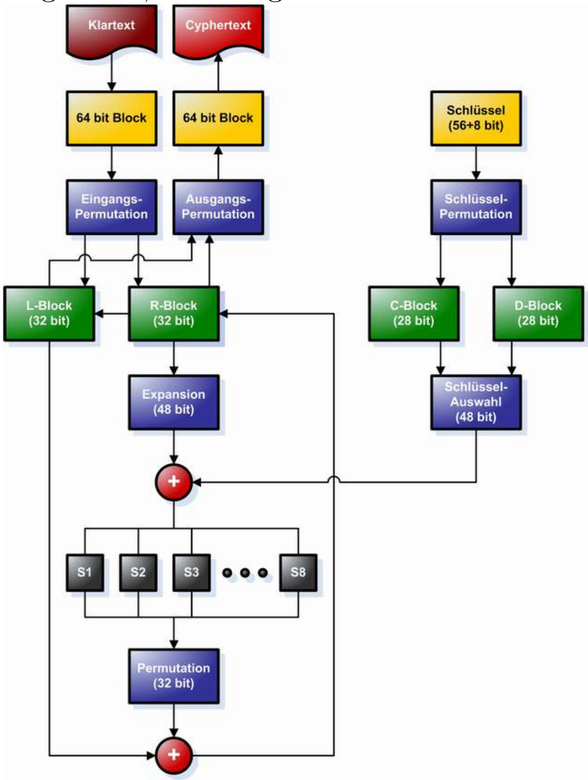
Florian Anritsch

7.1 Über DES

Der Data Encryption Standard (Abkürzung: DES) ist ein weit verbreiteter symmetrischer Verschlüsselungsalgorithmus. Im Jahr 1976 legte die US-Regierung DES als Standard fest. Seither findet DES weltweit Einsatz, wobei Kritiker meinen, dass aufgrund der Beteiligung der NSA am Design des Algorithmus dessen Sicherheit in Frage zu stellen ist. Heutzutage gilt DES aufgrund seiner relativ kurzen Schlüssellänge von 56bit nicht mehr als ausreichend sicher. Bei dreifacher Anwendung von DES, dem sog. Triple-DES, 3DES oder TDES, herrscht allerdings nochimmer eine extrem hohe Sicherheit vor. Erst vor wenigen Jahren wurde 3DES als offizieller Standard für die USA durch den Advanced Encryption Standard (AES) abgelöst.

7.2 Funktionsweise

Dass DES ein sog. symmetrischer Verschlüsselungsalorithmus ist, bedeutet, dass zum Ver- und Entschlüsseln der selbe Schlüssel verwendet wird. Die Verarbeitung läuft in 64Bit Blöcken, die einzeln codiert und am Schluss wieder zusammen gesetzt werden. Der Schlüssel weist ebenfalls eine Länge von 64Bit auf, effektiv werden allerdings "nur" 56Bit genutzt, die übrigen sind für den Paritätscheck reserviert.



Schematische Darstellung von DES

DES besteht aus einer Reihe von Permutationen, sowie einer Verrechnung mit 16 Schlüssel in 16 Runden.

7.3 Schwachstellen

Wie erwähnt ist DES aufgrund der Schlüssellänge (56Bit) nicht mehr 100% sicher, und wurde bereits durch Brute-Force-Angriffe überwunden, indem nacheinander alle möglichen ($2^{56} = 72 * 10^{15}$ Möglichkeiten) ausprobiert wurden. Es gibt die Vermutung, dass diese kleine Schlüssellänge absichtlich gewählt wurde, weil die NSA bereits in den 1970er-Jahren genug Rechnerkapazität besaß, um diese Verschlüsselung in brauchbarer Zeit zu brechen.

7.3.1 Deep Crack

1998 wurde ein 250000$ teurer Supercomputer namens *Deep-Crack* gebaut. er enthielt 1536 speziell auf den DES abgestimmte Krypto-Chips, die pro Sekunde über 88 Milliarden Schlüssel testen konnte. In nur 56 Stunden konnte so ein DES-Code entschlüsselt werden. 1999 schaffte es die gleiche Maschine, mit dem weltweiten Netzwerk von distributed.net, bestehend aus etwa 100.000 Rechnern, zusammen arbeitend, den DES Schlüssel in 22 Stunden und 15 Minuten zu finden.

7.4 COPACOBANA

Die einzige andere öffentlich bekannte Maschine zum Brechen von DES ist COPACOBANA. Sie wurde 2006 von zwei Arbeitsgruppen an den Universitäten Bochum und Kiel gebaut. Im Gegensatz zu Deep Crack besteht eine COPACOBANA aus rekonfigurierbaren Hardware-Bausteinen, so genannten FPGAs. 120 FPGAs vom Typ XILINX Spartan3-1000 sind in einer Maschine auf 20 DIMM Modulen zusammen gefasst, wobei jedes DIMM Modul sechs FPGAs enthält. COPACOBANA kann 65 Milliarden DES-Schlüssel pro Sekunde testen, woraus sich eine durchschnittliche Suchzeit von 6,4 Tagen für eine DES-Attacke ergibt. Durch den Einsatz rekonfigurierbarer Hardware kann COPACOBANA auch zum Brechen anderer Chiffren wie A5 eingesetzt werden. Die Material- und Herstellungskosten von COPACOBANA belaufen sich auf etwa 10.000 Dollar. Der Kostenvorteil gegenüber Deep Crack um einen Faktor 25.

7.5 Formale Algorithmische Beschreibung

Im Grobaufbau finden laufen folgende Schritte ab:

- $r(b)$ bezeichnet die rechte Hälfte einer 64-Bit-Folge b,
- $l(b)$ bezeichnet die linke Hälfte,
- $b1b2$ bezeichnet das Aneinanderhängen zweier Bit-Folgen $b1$ und $b2$.

  Die Zuordnung
  $$b := r(b)l(b)$$
  beschreibt also das Vertauschen von linker und rechter Hälfte.
- Mit F(b,i) bezeichnen wir das Ergebnis des F-Moduls bei Anwendung auf die 32-Bit-Folge b in Runde i.

Algorithmus DES Verschlüsselung

Eingabe: zu verschlüsselnde 64-Bit-Folge b

Ausgabe: 64-Bit-Folge c, die DES-Verschlüsselung von b

$$
\begin{aligned}
c &:= IP(b)\ \{Anfangspermutation\} \\
i &= 0, 1, \ldots, 15\ \{Runden\} \\
&\quad c1 := r(c) \\
&\quad c2 := l(c)\, XOR\, F(c1, i)\ \{F - Modul\} \\
&\quad\ c := c1c2 \\
c &:= r(c)l(c)\ \{erneute Vertauschung\} \\
c &:= PI(c)\ \{Endpermutation\}
\end{aligned}
$$

7.5.1 Permutationen in DES

Hier findet sich eine Übersicht über alle Permutationen, die in DES verwendet werden. Sie werden in den darauffolgenden Kapiteln noch genauer erläutert.

	IP	PI	P	E	PC1	PC2		IP	PI	P	E	PC1	PC2
1	58	40	9	2,48	8	5	33	57	36			4	35
2	50	8	17	3	16	24	34	49	4			12	41
3	42	48	23	4	24	7	35	41	44			20	
4	34	16	31	5,7	56	16	36	33	12			28	46
5	26	56	13	6,8	52	6	37	25	52			48	28
6	18	24	28	9	44	10	38	17	20			40	
7	10	64	2	10	36	20	39	9	60			32	39
8	2	32	18	11,13		18	40	1	28				32
9	60	39	24	12,14	7		41	59	35			3	25
10	52	7	16	15	15	12	42	51	3			11	44
11	44	47	30	16	23	3	43	43	43			19	
12	36	15	6	17,19	55	15	44	35	11			27	37
13	28	55	26	18,20	51	23	45	27	51			47	34
14	20	23	20	21	43	1	46	19	19			39	43
15	12	63	10	22	35	9	47	11	59			31	29
16	4	31	1	23,25		19	48	3	27				36
17	62	38	8	24,26	6	2	49	61	34			2	38
18	54	6	14	27	14		50	53	2			10	45
19	46	46	25	28	22	14	51	45	42			18	33
20	38	14	3	29,31	54	22	52	37	10			26	26
21	30	54	4	30,32	50	11	53	29	50			46	42
22	22	22	29	33	42		54	21	18			38	
23	14	62	11	34	34	13	55	13	58			30	30
24	6	30	19	35,37		4	56	5	26				40
25	64	37	32	36,38	5		57	63	33			1	
26	56	5	12	39	13	17	58	55	1			9	
27	48	45	22	40	21	21	59	47	41			17	
28	40	13	7	41,43	53	8	60	39	9			25	
29	32	53	5	42,44	49	47	61	31	49			45	
30	24	21	27	45	41	31	62	23	17			37	
31	16	61	15	46	33	27	63	15	57			29	
32	8	29	21	47,1		48	64	7	25				

7.5.2 IP-Permutation

Unter einer Permutation versteht man das Vertauschen aller Elemente nach einer bestimmten Vorgabe. eine Übersicht über die einzelnen Permutationen, von denen in DES Gebrauch gemacht wird, findet sich in obiger Ansicht. Die 64 Bit lange Eingabekette wird in angegebener Weise vertauscht. Es folgen die 16 Runden, in denen die Bit-Kette mit den 16 Schlüsseln codiert werden.

7.6 F-Modul

Das F Modul besteht aus einer am Beginn durchgeführten erweiternden Permutation-E (siehe Tabelle), einer anschließenden Codierung mit dem Schlüssel.

Die erweiternde Permutation bildet eine 32 Bit lange Hälfte es Inputs auf eine 48 Bit lange Kette ab. Das Funktioniert, indem bestimmte Elemente an mehrere Stellen geschrieben. Anschließend folg eine Verknüpfung mittels exklusivem Oder zwischen der Eingabe und dem Schlüssel.

Abschließend wird der 48Bit lange Code mittels der S-Boxen auf 32Bit zusammengerechnet.

Die acht Substitutionsmodule

S0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
01	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
10	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
11	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D

S1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	F	1	8	E	6	B	3	4	9	7	2	D	C	0	5	A
01	3	D	4	7	F	2	8	E	C	0	1	A	6	9	B	5
10	0	E	7	B	A	4	D	1	5	8	C	6	9	3	2	F
11	D	8	A	1	3	F	4	2	B	6	7	C	0	5	E	9

S2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	A	0	9	E	6	3	F	5	1	D	C	7	B	4	2	8
01	D	7	0	9	3	4	6	A	2	8	5	E	C	B	F	1
10	D	6	4	9	8	F	3	0	B	1	2	C	5	A	E	7
11	1	A	D	0	6	9	8	7	4	F	E	3	B	5	2	C

S3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	7	D	E	3	0	6	9	A	1	2	8	5	B	C	4	F
01	D	8	B	5	6	F	0	3	4	7	2	C	1	A	E	9
10	A	6	9	0	C	B	7	D	F	1	3	E	5	2	8	4
11	3	F	0	6	A	1	D	8	9	4	5	B	C	7	2	E

S4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	2	C	4	1	7	A	B	6	8	5	3	F	D	0	E	9
01	E	B	2	C	4	7	D	1	5	0	F	A	3	9	8	6
10	4	2	1	B	A	D	7	8	F	9	C	5	6	3	0	F
11	B	8	C	7	1	E	2	D	6	F	0	9	A	4	5	3

S5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	C	1	A	F	9	2	6	8	0	D	3	4	E	7	5	B
01	A	F	4	2	7	C	9	5	6	1	D	E	0	B	3	8
10	9	E	F	5	2	8	C	3	7	0	4	A	1	D	B	6
11	4	3	2	C	9	5	F	A	B	E	1	7	6	0	8	D

S6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	4	B	2	E	F	0	8	D	3	C	9	7	5	A	6	1
01	D	0	B	7	4	9	1	A	E	3	5	C	2	F	8	6
10	1	4	B	D	C	3	7	E	A	F	6	8	0	5	9	2
11	6	B	D	8	1	4	A	7	9	5	0	F	E	2	3	C

S7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	D	2	8	4	6	F	B	1	A	8	3	E	5	0	C	7
01	1	F	D	8	A	3	7	4	C	5	6	B	0	E	9	2
10	7	B	4	1	9	C	E	2	0	6	A	D	F	3	5	8
11	2	1	E	7	4	A	8	D	F	C	9	0	3	5	6	B

Es werden jeweils 6 Bit hergenommen. Das erste und das letzte Bit ergeben die Zeile, die mittleren 4 Bit zusammengezählt ergeben die Spalte. Das Element das an der gefragten Stellen steht wird dann ein Binärdarstellung angeschrieben. Jede der 8 6Bit langen Ketten wird mit einem eigenen S-Modul (S0-S7) Codiert. Es Resultiert wieder eine 32Bit lange Kette von Nullen und Einsen.

Am Ende wird die PI-Permutation Permutation durchgeführt. Die PC1 und PC2 Permutation, welche ebenfalls in der Liste aufgeführt sind dienen in Zwischenschritten zur Schlüsselerzeugung, auf welche an dieser Stelle nicht weiter eingegangen werden soll.

7.7 Implementierung des DES Algorithmus

Es gelang, den DES Algorithmus in C nachzuprogrammieren. Die Umsetzung der willkürlich erscheinenden Permutationen stellte sich als ausgesprochen schwierig heraus. Die übrigen durchgeführten Operationen weisen einzeln keine hohe Schwierigkeit auf, im Summe jedoch ist die Vielzahl der nacheinander durchgeführeten Codierungen verantwortlich für die hohe Komplexität von DES.

8 Vorbereitung für die Hash-Funktion

Florian Mikulik

8.1 Anfang

Als großes Ziel haben wir uns gesetzt, eine eigene Hash-Funktion zu erstellen, mit der wir dann von einem PC zum anderen verschlüsselte Daten senden können. Wir haben uns dazu entschlossen, alle unsere Programme in C zu programmieren.

Leider ist es in C nicht ganz einfach, alle mathematischen Methoden anzuwenden, die in der Kryptographie von Nöten sind. Deshalb mussten wir teilweise Algorithmen entwickeln, die diese mathematischen Methoden nachahmen.

8.2 Primzahlberechnung

Als Vorbereitung für die Hash-Funktion, und für die Diffie-Hellman-Verschlüsselung der gesendeten Hash-Werte benötigten wir einige Funktionen und Programme, die uns zum Beispiel alle Primzahlen bis zu einem gewissen Grenzwert berechnen können. Dies passiert, indem eine Schleife alle ungeraden Zahlen bis zu dem Grenzwert durchläuft, und diese mit allen kleineren Zahlen auf die Teilbarkeit überprüft.

```
int i=3;
for(i; i<=stop;i=i+2){ //----Schleife von 3 - zur obersten Grenze, die Primzahl 2 ist bereits im Array vordefiniert und wird nicht berechnet ---//
    int j=2;
    for(j;j<i;j++)//----Schleife von 2 bis zur Zahl die gerade ueberprueft wird---//
    {
      if(i%j==0){//----Wenn teilbar, dann keine Primzahl---//
                    pruef=1;
                    j=i;//----Abbruch der Schleife---//
                   }
    }
   if(pruef==0){//----Wenn nicht teilbar, wird die Zahl in als Primzahl gespeichert---//
                     zarray[c]=i;
                     c=c+1;
                     }
    pruef=0;
}
```

Das Programm zur Prinzahlberechnung läuft problemlos mit bis zu 8-stelligen Zahlen, nur leider ist der Rechenaufwand sehr groß, da alle kleineren Zahlen überprüft werden, was zur Folge hat, dass die Rechenzeit des Programms ab 6-stelligen Zahlen über 30 Sekunden beträgt.

8.3 Primfaktorisierung

Des Weiteren brauchen wir für die Diffie-Hellman-Verschlüsselung ein kleines Script, welches Zahlen in ihre Primfaktoren zerlegt. Bei diesem Programm wird eine Zahl eingegeben. Dann werden alle kleineren Zahlen ab 2 auf die teilende Eigenschaft überprüft.

```
while (x%a!=0)//----Erster Primfaktor wird ermittelt. Wenn wenn a x nicht ohne Rest teilt, wird a um 1 erhoeht ---//
{
```

```
a=a+1;
}
while (x%a==0)//----Wenn wenn a x ohne Rest teilt, wird die dividiert, der Faktor ausgegeben und die Schleife solange wiederholt, bis a x nicht mehr ohne Rest teilen kann ---//
{
x=x/a;
printf("%d,", a);
while (x%a!=0 && x!=1)
{
a=a+1;
}
}
```

Das Script zur Primfaktorisierung zerlegt bis zu 9-stellige Zahlen in unter einer Sekunde, größere Zahlen jedoch erhöhen den Rechenaufwand so erheblich, dass die Dauer-Nutzen-Relation nicht mehr im sinnvollen Bereich ist.

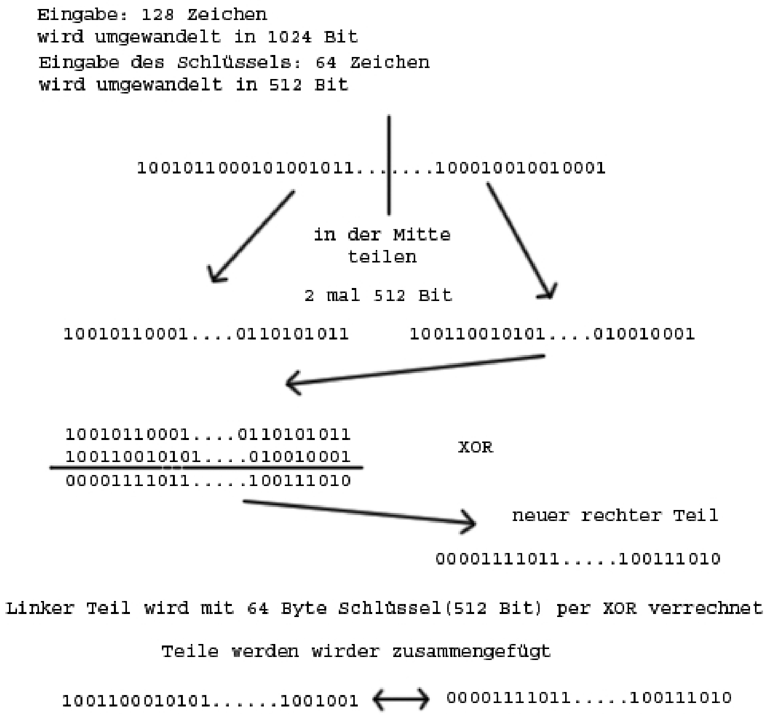
8.4 Verschlüsseln in Bit und Byte

Danach haben wir uns noch dazu entschlossen, dass wir den bereits verschlüsselten Hash-Code, den wir aus der Hash-Funktion erhalten, vor dem Senden noch einmal verschlüsseln, und statt in hexadezimaler Schreibweise in Bit-Schreibweise(nur „0“ und „1“) versenden. Zuerst werden die Zahlen und Buchstaben des Hashes eingelesen, und über den ASCII-Code(zB. $A = 65 = 2^0 + 2^6 = 01000001$) in 1024 Bit umgewandelt. Danach wird der Binärcode in 2 Hälften unterteilt. Die *rechte* Hälfte wird per bitweisem *exklusivem ODER* mit der *linken* Hälfte verrechnet($1+1 = 0; 0+0 = 0; 1+0 = 1; 0+1 = 1$), danach wird das Ergebnis als rechte Hälfte verwendet, somit ist die Rechte Hälfte von der Linken abhängig. Nun werden die linken 512 Bit mit einem 64 Zeichen langen Schlüssel, der bereits vom User bestimmt wurde und in Binärdarstellung vorliegt, per *exklusivem ODER* verrechnet. Nun ist die 1024 Bit lange Kette aus Nullen und Einsen so vermischt, dass diese ohne den Schlüssel nur schwer wiederhergestellt werden kann.

Als Problem bei der Programmierung stellte sich heraus, das die Typenkonvertierung der char-Variablen(String der Eingabe), in Integer-Arrays nicht ganz einfach vor sich geht. Wenn man in der Char-Variable ein „1“ einliest, wird in der Variable der ASCII Wert des Zeichens gespeichert. Für „1“ ist dieser „49“. Das heißt, bei bei der Übertragung eines Char-Arrays mit Binärcode, wie es zum Beispiel bei der Entschlüsselung eingegeben wird, in ein Int-Array muss der Wert „49“ im Char-Array in den Wert „1“ im Int-Array konvertiert werden.

```
for(i=0;i<1024;i=i+1){
if(char1[i] == 49)char3[i]=1;
if(char1[i] == 48)char3[i]=0;
}
```

Leider haben sich beim Angleichen des Codes der Bitverschlüsselung an das Programm zum Senden der Daten einige kleine Fehler eingeschlichen, die wir bis zum Schluss teils nicht finden und teils nicht beheben konnten, und deshalb haben wir uns entschlossen, den Hash-Code direkt zu übermitteln.



Diese Verschlüsselungsmethode wurde dem DES-Schlüssel nachempfunden, der fast die gleiche Methode zur Verschlüsselung verwendet, jedoch wird der Code nicht in einer Runde, sondern in 16 Runden verschlüsselt.

9 Diffie-Hellman-Verschlüsselung

Paul Laufer

Eine der ersten Methoden, die wir im Laufe der diesjährigen Modellierungswoche besprochen haben, ist das Diffie-Hellman-Verfahren zum sicheren Austausch von Schlüsseln. Unser Vorhaben war es, ein solches System zu programmieren um einen sicheren Datenaustausch zwischen zwei Systemen zu erlauben. In weiterer Folge sollte auch einfacher Text durch einen Algorithmus verschlüsselt, übertragen, und wieder entschlüsselt werden. Doch aufgrund von Kompatibilitätsproblemen konnte dieses Ziel nicht erreicht werden. Bevor wir uns dem eigentlichen Problem zuwenden, werden hier vorab noch einige Begriffe erläutert:

9.1 Informatik

9.1.1 TCP/IP

Das "Transmission Control Protocol / Internet Protocol" wurde in den 60er Jahren vom amerikanischen Militär zum übermitteln von Nachrichten zwischen dezentralen Knotenpunkten entwickelt. Diese Technologie ist für die Identifizierung mehrerer Computer in einem Netzwerk und den anschließenden Aufbau von sogenannten Socket-Verbindungen zuständig. Über diese Sockets können die Systeme anschließend in beide Richtungen kommunizieren.

9.1.2 Programmierung in C

Die Programmiersprache C ist eine der erfolgreichsten in der Geschichte der elektronischen Datenverarbeitung. In den frühen 70ern für UNIX-Systeme entwickelt, wird die Sprache heute auf allen Systemen verwendet. Dank des relativ einfachen, direkten Zugriffs auf Speicherplatz wird C häufig zur Systemprogrammierung verwendet. Diese Tatsache war auch für uns ausschlaggebend, denn kryptographische Funktionen sollten möglichst direkt mit den Komponenten verbunden sein, um eine fehlerlose Kommunikation zu ermöglichen. Des Weiteren war es mit C im Gegensatz zu dessen Weiterentwicklungen C++ und C# einfacher möglich, beinahe gleichen Code auf Linux- und Windows-Rechnern auszuführen und somit alle Laptops unserer Gruppe zu verwenden.

> When I find my code in tons of trouble, Friends and colleagues come to me, Speaking words of wisdom: "Write in C."
>
> As the deadline fast approaches, And bugs are all that I can see, Somewhere, someone whispers: "Write in C."
>
> Write in C, Write in C, Write in C, oh, Write in C. LOGO's dead and buried, Write in C.
>
> I used to write a lot of FORTRAN, For science it worked flawlessly. Try using it for graphics! Write in C.
>
> If you've just spent nearly 30 hours, Debugging some assembly, Soon you will be glad to Write in C.
>
> Write in C, Write in C, Write in C, yeah, Write in C. BASIC's not the answer. Write in C.
>
> Write in C, Write in C Write in C, oh, Write in C. Pascal won't quite cut it. Write in C.

9.2 Mathematik

9.2.1 Eulersche φ-Funktion

Florian Andritsch

$$\varphi(n) := \left|\{1 \leq a \leq n \mid \text{ggT}(a,n) = 1\}\right|$$

Gibt die Anzahl der positiven, zu n teilerfremden Zahlen a mit $a \leq n$ an.

$$\varphi(n) = \begin{cases} (n-1) & \text{für } n \in \mathbf{P} \\ (p_1 - 1)p_1^{e^1-1} \cdots (p_n - 1)p_n^{e^n-1} & \text{für } n = p_1^{e^1} \cdots p_n^{e^n} \end{cases}$$

9.2.2 Primitivwurzel

Florian Andritsch

Modulo n - Restklassen: $0, 1, \ldots (n-1)$
relativ prime Reste: $(\mathbb{Z}/_{(n)})^x$
git es ein $g \in \mathbb{Z}$ mit $\left\{g, g^2, \ldots, g^{\varphi(n)} = 1\right\} mod(n)$
$\text{ggT}(g,n) = 1$

modulo 11 :

$$\varphi(11) = 10$$

$g = 2$:

$$\begin{aligned}
2^1 &\equiv 2\,(11) \\
2^2 &\equiv 4\,(11) \\
2^3 &\equiv 8\,(11) \\
2^4 &\equiv 16 \equiv 5\,(11) \\
2^5 &\equiv 32 \equiv 10\,(11) \\
2^6 &\equiv 64 \equiv 9\,(11) \\
2^7 &\equiv 128 \equiv 7\,(11) \\
2^8 &\equiv 256 \equiv 3\,(11) \\
2^9 &\equiv 512 \equiv 6\,(11) \\
2^{10} &\equiv 1024 \equiv 1\,(11)
\end{aligned}$$

Satz: Zu einem Modul n gibt es Primitivwurzeln
$\Longleftrightarrow n \in \{(1), 2, 4, p^e, 2p^e | p : \text{ungerade Primzahl}, e > 1\}$

9.3 Wie findet man Primitivwurzeln?

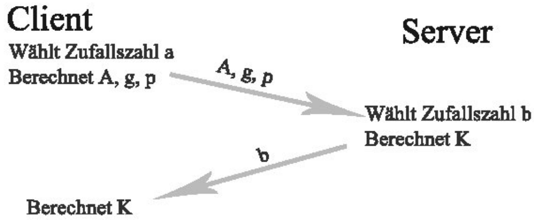
Zum einen ist es möglich alle Reste durchzuprobieren, der Zahlentheoretisch gebildetere verwendet aber folgende Tatsache:
n Modul,$a \in \mathbb{Z}$ mit $\text{ggT}(a,n) = 1$

$$\exists t : a^t \equiv 1\,(11) \text{ und wenn } 1 \leq t \leq \varphi(n) \Rightarrow t \mid \varphi(n)$$

9.4 Zahlentheorie in der Verschlüsselung

Paul Laufer

Beim Diffie-Hellman-Verfahren[1]senden sich zwei Kommunikationspartner über eine unsichere Leitung jeweils eine Nachricht zu. Allein mit diesen beiden Nachrichten ist es praktisch unmöglich, den Schlüssel zu errechnen, während die beiden Kommunikationsteilnehmer diese Aufgabe sofort durchführen können, da die übermittelten Nachrichten von gespeicherten Zufallsvariablen abhängen.



Zwei Kommunikationspartner wollen über ein unsicheres Medium, etwa eine Kabel- oder Funkleitung, verschlüsselt kommunizieren. Dazu soll ein symmetrisches Kryptosystem eingesetzt werden, für das beide jedoch zunächst einen gemeinsamen geheimen Schlüssel benötigen. Indem sie den Diffie-Hellman-Schlüsselaustausch durchführen, gelangen sie beide in den Besitz eines solchen Schlüssels. 1)Die Kommunikationspartner einigen sich zunächst auf eine Primzahl p und eine Primitivwurzel $g\,mod\,p$. Diese $2 \leq g \leq p-2$ Parameter müssen nicht geheim bleiben, können also insbesondere auch über ein unsicheres Medium übertragen werden. 2)Beide Kommunikationspartner erzeugen jeweils eine geheim zu haltende Zufallszahl a bzw. b aus der Menge $\{1, \ldots, p-2\}$ a und b werden nicht übertragen, bleiben also dem jeweiligen Kommunikationspartner, aber auch potenziellen Lauschern, unbekannt. 3)Die Kommunikationspartner berechnen $A = g^a\,mod\,p$ bzw. $B = g^b\,mod\,p$. Nun werden A und B über das unsichere Medium übertragen. 4)Die Kommunikationspartner berechnen nun $K = B^a\,mod\,p$ bzw. $K = A^b\,mod\,p$. Das Ergebnis K ist für beide Partner gleich und kann als Schlüssel für die weitere Kommunikation verwendet werden. Dass beide Kommunikationspartner den selben Wert für K berechnen, zeigen die folgenden beiden Gleichungen:

$$K = B^a\,mod\,p = (g^b\,mod\,p)^a\,mod\,p = g^{b^a}\,mod\,p = g^{a^b}\,mod\,p$$

$$K = A^b\,mod\,p = (g^a\,mod\,p)^b\,mod\,p = g^{a^b}\,mod\,p$$

9.5 Client-Teil

In unserem Programm wurde dieses System anschließende folgendermaßen umgesetzt:

```
Prime = getRandomPrime(); // Get random prime
PrimeRoot = getPrimeRoot(Prime); // Get Primeroot to given Prime
Random = getRandom(Prime); // Get Random Number smaller than Prime
X = getAB(Prime, PrimeRoot, Random); // get Number to pass to partner
```

[1]Diffie-Hellman-Schlüsselaustausch. Online im Internet: URL: http://de.wikipedia.org/wiki/Diffie-Hellman-Schl%C3%BCsselaustausch [Stand: 2008-01-18]

Zuerst werden die einzelnen Werte über eigene Funktionen ermittelt, wobei die Variablen in diesem Fall folgendermaßen den oben im Beispiel angeführten gleichzusetzen sind: Prime entspricht p PrimeRoot entspricht g Random entspricht a X entspricht A

In der Funktion getRandomPrime() wird eine Primzahl zufällig ermittelt, da hier aber kein mathematischer Hintergrund vorhanden ist, wird auf diese Funktion nicht mehr genauer eingegangen. Die Funktion getPrimeRoot() wiederum ermittelt eine passende Primitivwurzel zur Primzahl Prime (p). Diese einzelnen Funktionen haben folgenden Aufbau:

```
int length = Phi(P); // Get Length of numbers that are no factor of P
PrimeFactors = getPrimeFactors(length); // Get Prime Factors of length
for(i = 2; i <= (P-2); i++) // Loop P-2 times to use all possible
{
    ok = 1; // Set default value of boolean-used var ok to 1 (true)
    for(j = 0; j < 200; j++) // Loop 200 times to be sure to get all
        Prime Factors
    {
        if(PrimeFactors[j] != 0)  // If PrimeFactor-Item is set
        {
            if((int)pow(i, (length/PrimeFactors[j])) % P == 1) // If i to
                the power of
 length divived by the current PrimeFactor modulu P is 1 set ok to 0 (
    false)
                ok = 0;
        }
    }
    if(ok == 1) // Only if this is a valid Primeroot
    {
        result = i; // Set the result-var
        goto end; // Go to the end section without continuing the loop
    }
}
end:
    return result;
```

Zu Beginn wird in die Variable length das Ergebnis der Eulerschen φ-Funktion geschrieben, die in diesem Fall immer P-1 zurückgibt, da eine Primzahl durch alle natürlichen Zahlen, die kleiner sind als sie selbst (außer 1) nicht teilbar ist. Von dieser Variable length werden anschließend mit Hilfe der Funktion getPrimeFactors die einzelnen Primfaktoren ermittelt. Falls keiner der Werte von 1 bis P -2 hoch der length dividiert durch den aktuellen Primfaktor von length eins ergibt, ist die Basis i ein gültiges Resultat als Primitivwurzel und kann von der Funktion zurückgegeben werden. (Die Funktion pow wird zum Hochrechnen benutzt, das erste Argument ist die Basis, das zweite der Exponent.)

Nachdem alle Werte ermittelt wurden, wird der zu übermittelnde String sowie die Werte an die Funktion HandleTCPClient übergeben, die für die Kommunikation mit dem Partner zuständig ist.

```
HandleTCPClient(string, test); // Initialize the TCP-Client with the
    values
```

In dieser Funktion wird das Modul zur TCP/IP-Verbindung erstellt und die Werte IP und Port aus den globalen Variablen eingelesen. Windows stellt zu diesem Zweck eine eigene Bibliothek namens winsock.h zur Verfügung. Nachdem die Verbindung über die Funktion connect dieser Bibliothek hergestellt wurde, kann mit Hilfe der Funktion send ein Datenstring an den Kommunikationspartner übermittelt werden:

```
if (send(sock, Code, strlen(Code), 0) != strlen(Code))
    DieWithError("send() sent a different number of bytes than expected");
printf("SEND: %s\n", Code); // Print the char-string that was send
```

Der String, welcher übermittelt wird, setzt sich folgendermaßen zusammen: Alle Werte sind durch einen Strichpunkt getrennt, zu Beginn wird das Schlüsselwort „data“ dessen Gegenpart „end“ den String wieder abschließt. Die Zeichenkette hat daher folgenden Aufbau:

data;[Primzahl];[Primitivwurzel];[Errechnete Zahl (A bzw. B);end

Nach dem gleichen Schema wird anschließend der Code ausgeführt, der für den Empfang der vom Kommunikationspartner errechneten Zahl zuständig ist. Diese Zahl wird in die Variable echoBuffer geschrieben. Mit Hilfe der Funktion atoi wird die Variable, die als Zeichenkette ankommt in einen Integer-Wert verwandelt, der rechentechnisch verwendet werden kann.

```
if ((bytesRcvd = recv(sock, echoBuffer, RCVBUFSIZE - 1, 0)) <= 0)
        DieWithError("recv() failed or connection closed prematurely");

OtherX = atoi (echoBuffer); // Convert the received number (B') to int
```

Da nun alle Werte bekannt sind, kann über die Funtion getK() der Schlüssel errechnet werden:

```
int getK(int p, int AB, int ab)
{
    int K = 0; // Set the return value
    int temp = pow(AB,ab); // Set temp = AB to the power of ab
    K = temp%p; // K = temp modulu p (Prime)
    return K;
}
```

9.6 Server-Teil

Nachdem gleich wie beim Client die TCP-Verbindung hergestellt wurde, wird die erhaltene Zeichenkette in ihre Einzelwerte aufgespalten, um die Primzahl, die Primitivwurzel und die vom Client errechnete Zahl B getrennt handhaben zu können. Auch die Ermittlung einer Zufallszahl b und des zugehörigen Wertes B läuft synchron ab. Dieser Wert B wird anschließend wieder an den Client zurückgesandt. Durch dieselbe Funktion getK() kann nun derselbe Schlüssel errechnet werden.

10 MD5 Hash-Algorithmus

Wolfgang Hrauda

10.1 Allgemeines

Der MessageDigest Algorithm 5, kurz MD5, stellt eine oft verwendete Möglichkeit einer Hash-Funktion dar. Sie wurde 1991 von Ronald L. Rivest am Massachusetts Institute of Technology entwickelt, da Analysen des Vorgängers MD4 Schwächen offenbart hatten. Nachrichten, welche mit MD5 verschlüsselt wurden, sind 128 Bit lang und werden normalerweise als 32-stellige Hexadezimalzahl notiert. Die Länge der Originalnachricht ist dabei nicht von Bedeutung. Die MD5 - Summe für den String „Ich bin ein Freak!" beträgt zum Beispiel „d5f8426b80f7babe672ab8701a01e83c". Selbst eine „leere Nachricht", also eine Zeichenkette der Länge null, erzeugt folgende MD5 - Summe: d41d8cd98f00b204e9800998ecf8427e

10.2 Anwendungen

MD5 - Summen werden zum Beispiel zur Integritätsprüfung von Dateien verwendet, wobei die momentane MD5 - Summe einer Datei mit einer bereits von früher bekannten Summe verglichen wird. Hat sich diese nicht verändert, so ist auch die Datei unverändert; damit kann zum Beispiel sichergestellt werden, dass sie nicht von Viren oder Trojanern befallen ist. Ebenso kann am Ende eines Downloads überprüft werden, ob eine Datei vollständig heruntergeladen wurde, indem ihre Prüfsumme mit einer vom Server mitgeteilten Prüfsumme verglichen wird. In den meisten gängigen Betriebssystemen ist eine Möglichkeit, MD5 - Summen zu berechnen bereits standardmäßig integriert (z.B. md5sum unter Linux). Die einzige Ausnahme bildet das Betriebssystem „Windows" der Firma Microsoft, welches kein Programm zur Berechnung von MD5 - Hashes bereitstellt.

10.3 Sicherheit

10.3.1 Brute-Force Angriff

Anfangs wurde MD5 als kryptographisch sicher beurteilt, doch bereits 1994 wurden erste Pseudo-Kollisionen entdeckt. Zur selben Zeit entwickelten Michael J. Wiener und Paul C. van Oorschot das Konzept eines Angriffs auf MD5, welches sich der sogenannten Brute-Force-Methode bedient. Diese Methode beruht im Wesentlichen auf dem erschöpfenden Durchprobieren aller bzw. möglichst vieler möglichen Fällen. Das Projekt war jedoch auf einen 10 Millionen Dollar teuren fiktiven Rechner aufgebaut, mit dessen Hilfe man innerhalb von 24 Tagen eine Kollision in MD5 zu finden hoffte. Aufgrund der hohen Kosten begann die praktische Umsetzung erst 2004, wobei man die notewendigen Operationen auf mehrere Rechner verteilte um zusätzliche Kosten zu sparen. Da die Ausgangsdaten dabei nur sehr begrenzt kontrolliert werden können, lag der Hauptzweck dieses Projekts eher darin, Unternehmen, welche noch immer auf MD5 verwenden, zum Nachdenken zu bringen.

10.3.2 Rainbow - Tables

Rainbow - Tables sind Tabellen, in denen eine sehr große Anzahl von kurzen Zeichenketten mit ihren dazugehörigen MD5 - Summen gespeichert ist, womit sogar das

Entschlüsseln von MD5 möglich wird. Passwörter, welche in ihrer verschlüsselten Form in einer Datenbank gespeichert sind, können also auf diese Art und Weise zum Beispiel von Hackern, die sich Zugang zur Datenbank verschaffen haben, entschlüsselt werden. Die Nachteile dieser Methode liegen allerdings auf der Hand, denn aufgrund von beschränktem Speicherplatz und zu großem Zeitaufwand kann eine solche Tabelle nur für Strings bis zu einer bestimmten Länge erstellt werden. Es zeigt sich aber, wie wichtig es ist, bei Passwörtern eine gewisse Mindestlänge einzuhalten (diese wird meist mit 8 Zeichen angegeben). Im Internet finden sich sogar frei zugängliche MD5 - Decrypter, welche sich solcher Rainbow - Tables bedienen Unter diesem Link[2] findet sich zum Beispiel ein Decrypter, dessen Tabellen bis zu 4-stellige Zahlen-Buchstabenkombinationen, bis zu 6-stellige Zahlenkombinationen und zusätzlich mehr als 720.000 häufig verwendete Passwörter enthalten. Es bestehen jedoch sogar fertige Listen, welche bis zu einer Länge von acht Zeichen gehen.

10.3.3 Analyse

Bei dieser Methode werden aufwändige Analysen des Algorithmus durchgeführt, womit systematisch Kollisionen erzeugt werden können. Im August 2004 fand ein chinesisches Wissenschaftlerteam mithilfe eines IBM-P690-Clusters[3] nach einer Stunde Rechenzeit eine Kollision in MD5; weitere Kollisionen wurden innerhalb von maximal fünf Minuten gefunden.

10.3.4 Zusammenfassung

Es wurden also bereits mehrfach Kollisionen in MD5 festgestellt, die allerdings auf die praktische Fälschungssicherheit von mit MD5 erzeugten Zertifikaten keine Auswirkung haben. Es ist nämlich bislang nicht möglich gewesen, in sinnvoller Zeit einen sogenannten „Preimage-Angriff“ erfolgreich durchzuführen. Dies bedeutet, dass zu einer gegebenen Nachricht eine weitere Nachricht gefunden wird, die exakt denselben Hashwert erzeugt. Erst damit können richtige Fälschungen vorgenommen werden.

10.4 Algorithmus

10.4.1 Vorbereitende Schritte

Zunächst wird die zu verschlüsselnde Nachricht für die Verarbeitung durch den Hauptalgorithmus vorbereitet. Hierfür wird an die Binärdarstellung der Nachricht eine Eins angehängt; danach folgen Nullen, welche so lange hinzugefügt werden, bis die Länge der gesamten Nachricht 64 Bits davon entfernt ist, ein Vielfaches von 512 zu sein. Mathematisch ausgedrückt bedeutet dies:

$$l \equiv 448 \, mod \, 512$$

Wobei l die Länge der Nachricht ist.

Schließlich wird noch eine 64-bit Binärrepräsentation der Länge der Originalnachricht angehängt. Damit ist die Länge der resultierenden Bitkette durch 512 und damit auch durch 16 teilbar. Dies wird bei der Unterteilung in 16 32-bit „Wörter“ genützt.

[2] http://md5.xpzone.de/

[3] Siehe: http://www.fz-juelich.de/nic/Supercomputer/computer-d.html

10.4.2 Die Komponenten der Hauptverarbeitung

Als Basis der Verschlüsselung werden vier 32-bit Wörter (A, B, C, D) verwendet, deren Startwerte wie folgt standardisiert sind (in Hexadezimaldarstellung):

Wort A:	01 23 45 67
Wort B:	89 ab cd ef
Wort C:	fe dc ba 98
Wort D:	76 54 32 10

Weiters werden vier Hilfsfunktionen verwendet, welche drei 32-bit Wörter (X,Y,Z) mittels logischer Verknüpfungen zu einem 32-bit Wort verarbeiten:

$$F(X,Y,Z) = (X \wedge Y) \vee (\neq X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (Xv\neg Z)$$

Schließlich wird ein Array $T[i]$ mit 64 Elementen definiert, dessen Werte mithilfe der Sinusfunktion berechnet werden:

$$T[i] = \lfloor |sin(i+1)| \cdot 2^{32} \rfloor$$

Die Werte, welche der Sinusfunktion übergeben werden, sind im Bogenmaß angegeben, sodass - bedingt durch den ganzzahligen Index i - mehr oder weniger willkürliche Werte des Sinus verwendet werden. Durch den Absolutbetrag wird der negative Wertebereich des Graphen der Sinusfunktion sozusagen an der x-Achse in den positiven Bereich gespiegelt. Der Faktor 2^{32} sieht zunächst sehr willkürlich aus, beim Rechnen mit binären Zahlen nimmt jedoch die Zahl 2 denselben Stellenwert wie die Zahl 10 im Dezimalsystem ein. So wie bei der Multiplikation mit 10^{32} im Dezimalsystem das Komma um 32 Stellen nach rechts „verschoben" wird, geschieht dasselbe bei der Multiplikation mit 2^{32} im Binärsystem. Die Nachkommastellen des Ergebnisses werden mit der Gaussklammer $\lfloor x \rfloor$ „abgeschnitten" und das Endergebnis in das Feld i des Arrays T geschrieben.

10.4.3 Verschlüsselung

Nun werden jeweils 16 32-bit Wörter in einem Block verarbeitet. Als erstes werden die einzelnen Wörter im Array X gepuffert, wobei jeder Index ein gesamtes Wort anspricht. Nun folgt die eigentliche Verschlüsselung, welche in vier Runden geschieht, wobei jede Runde 16 Operationen beinhaltet. Alle Operationen sind im Allgemeinen wie folgt aufgebaut:

$$a^{'} = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s)$$

Da die Operationen von ihrer Struktur her ähnlich sind, werde ich exemplarisch die erste Operation der ersten Runde herausgreifen:

$$A^{'} = B + ((A + F(B,C,D) + X[0] + T[1]) <<< 7)$$

Der Wert A' setzt sich aus der Addition von unveränderten Wörtern mit dem aktuell zu verarbeitenden Wort der Originalbotschaft ($X[0]$), einem Wert aus dem „Sinusarray" ($T[1]$) und einem weiteren 32-bit Wort, welches durch logische Verknüpfungen von drei Wörtern entsteht (F(B,C,D)), zusammen. Auf Teile der Summe wird zusätzlich ein Bit Shift angewandt («< 7); in diesem Fall werden die Bits also um sieben Stellen nach links verschoben. Alle anderen Operationen sind ähnlich aufgebaut; die Unterschiede bestehen in vertauschten Rollen der einzelnen Wörter (A, B, C, D), Verwendung der Funktionen $F()$, $G()$, $H()$ und $I()$ (je nach Runde), sowie veränderten Werten für die Variablen k, i und s. Dabei werden die Werte für k und i (Startwerte bei der ersten Operation: 0 bzw. 1) nach jeder Operation um eins erhöht. Zu beachten ist, dass der Wert a' der vorhergehenden Operation bei der aktuellen Operation weiterverwendet wird, im Endeffekt gilt also $a = a'$. Nach Beendigung der vier Runden werden die neu erhaltenen Werte der vier Wörter A, B, C, D mit ihren alten Werten von vor der Verarbeitung des 16 Wörter (512-bit) - Blocks addiert. Dies gilt auch für die Verarbeitung der weiteren 512-bit Blöcke, sodass das Endprodukt ein 512-bit Block ist, auf den sich jedes einzelne Bit des Inputs auswirkt.

11 Fiat-Shamir-Protokoll

Florian Mikulik

Das Fiat-Shamir-Protokoll ist ein sogenanntes Zero-Knowledge Protokoll, bei dem der Kontrollinstanz der Schlüssel des Übermittlers völlig unbekannt ist.

Der Übermittler wählt eine Zahl

$$n = p * q$$

wobei p und q sehr hohe Primzahlen sind. Die Zahlen p und q hält der Übermittler geheim, nur die Zahl n wird veröffentlicht.

Nun wählt der Übermittler eine Zahl

$$s \in \{2, 3, \ldots, n-2\}\, mit\, ggT(s, n) = 1$$

und berechnet daraus

$$v \equiv s^2\, mod\, n$$

Die Zahl v wird auch bekanntgegeben. Das "Geheimnis" des Übermittlers ist jetzt die Zahl s, welche eine Quadratwurzel aus v modulo (n) ist.

Bei der Übertragung an die Kontrollinstanz wählt der Übermittler nun eine Zufallszahl r mit

$$r \in \{2, \ldots, n-2\}$$

und berechnet damit

$$x \equiv r^2\, mod\, n$$

und übermittelt x der Kontrollinstanz. Diese wählt nun zufällig eine Zahl

$$e \in \{0, 1\}$$

und sendet diese zurück zum Übermittler. Je nachdem ob $e = 1$ oder $e = 0$ sendet der Übermittler die Zahl y mit

$$y^2 \equiv x\, mod\, n$$

oder

$$y^2 \equiv v * x\, mod\, n$$

Der Übrmittler schickt als Antwort nun entweder r oder $r * s\, mod\, n$.

So wurde der Schlüssel s des Übermittlers nie bekanntgegeben, lediglich das Vorzeichen des Schlüssels ist nach der Übermittlung bekannt.

Bei dem Fiat-Shamir-Protokoll beträgt die Wahrscheinlichkeit eines Betrugs bei einer Runde 50%. Der Betrüger kann entweder ein r kennen, oder ein $r * s\, mod\, n$, aber ohne den wirklichen Schlüssel kann er nie beide Werte kennen. Wiederholt man nun dieses Verfahren oft genug, sinkt die Wahrscheinlichkeit eines Betrugs mit der Funktion

$$f(x) = 2^{-x}$$

nach 20 Runden beträgt die Wahrscheinlichkeit eines Betrugs daher nur mehr

$$f(20) = 2^{-20} \Leftrightarrow f(20) = 9,5 * 10^{-7}\%$$

12 |MEGATRON| - Designen eines eigenen Hash-Algorithmus

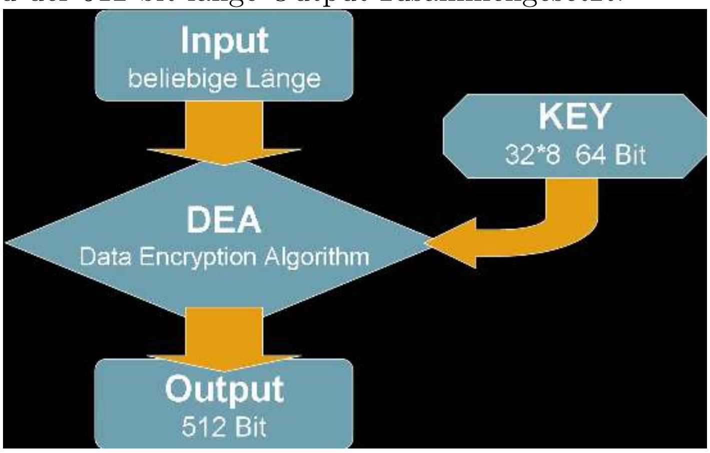
Wolfgang Hrauda, Florian Andritsch

12.1 Einleitung

Bereits am Beginn der Woche beschäftigte sich jeder von uns mit einem bestimmten Verschlüsselungsalgorithmus, dessen Funktionsweise wir anschließend innerhalb der Gruppe präsentierten. Auf diese Weise stiegen wir direkt in die Materie ein und entwickelten schnell ein Gefühl für die Prinzipien und Methoden von Verschlüsselungsalgorithmen. Auf diesen Erkenntnissen und Anregungen basierend begannen wir erste Ideen für das Design einer eigenen Hash - Funktion zu entwickeln. Innerhalb kürzester Zeit schafften wir es, diese zu konkretisieren und begannen parallel dazu sofort mit der programmiertechnischen Umsetzung, da unser Endziel war, eine Konsolenanwendung zur Verschlüsselung von Daten mit unserem Algorithmus zu schaffen. Die sofortige Realisierung unserer Ideen im Rahmen der Programmierung erwies sich innerhalb kürzester Zeit als optimale Vorgehensweise für die Entwicklung unserer Hash - Funktion. Damit war es uns in jedem Entwicklungsstadium möglich, den Algorithmus zu testen, womit wir die dadurch gewonnene praktische Erfahrung und gezielte Tests direkt in die Entwicklung einfließen lassen konnten. Durch die Verwendung der Programmiersprache C ist hundertprozentige Kompatibilität zu unseren anderen Entwicklungen gewährleistet. Außerdem konnte der Code beinahe plattformunabhängig geschrieben werden.

12.2 Grundstruktur

Der Input besteht aus einer Zeichenkette beliebiger Länge, von der die |MEGATRON| - Summe gebildet werden soll. Zunächst wird der Input in 512-bit Blöcke unterteilt, die getrennt verarbeitet, danach aber wieder zusammengefügt werden. Als erstes wird ein 8*64-bit Schlüssel 32-mal angewendet. Danach folgt die weitere Verarbeitung im eigentlichen Herz von |MEGATRON|, dem DEA (Data Encryption Algorithm). Nach einer abschließenden paarweisen Multiplikation aller einzelnen Bits in Pseudo-Hexadezimaldarstellung, wird der 512-bit lange Output zusammengesetzt.



12.3 Algorithmus

12.3.1 Input - Vorbearbeitung

Die zu verschlüsselnde Nachricht wird von der Konsoleneingabe eingelesen und anschließend mittels folgender for - Schleife in Binärdarstellung konvertiert.

```
for(i=0; i<64; i++)
{
        for(j=7;j>=0;j--)
        {
                if(inputchar[i]>=pow(2,j))
                {
                        inputbin[i][7-j]=1;
                        inputchar[i]=inputchar[i]-pow(2,j);
                }
        }
}
```

Wir rechnen von ASCII- in Binärdarstellung um. Dies geschieht, indem wir für jedes Bit überprüfen, ob der Wert größer oder gleich der dazugehörigen Zweierpotenz ist. Wenn dies der Fall ist, wird das dazu gehörige Bit auf 1 gesetzt, ansonsten auf 0. Wurde ein Bit auf 1 gesetzt, wird diese Zweierpotenz vom ASCII-Wert abgezogen.

12.4 Schlüssel

Es zu Beginn stehen 32*8 verschiedee 64Bit lange Schlüssel generiert. Als Basis dient die Binärentwicklung der Nachkommastellen von π. Mittels Verschiebungen werden 8 unterschiedliche Schlüssel für jede der 32 Runden erzeugt.

Anschließend wird die Bitkette in 8*64 Bit lange Blöcke (den sogenannten sBlöcken) aufgesplittet. Man kann sich das als 8*64 Tabelle vorstellen. Nun werden die zuvor generierten Schlüssel Runde für Runde zur Verfügung gestellt. Zeilenweise werden Schlüssel und Input per eklusivem oder logisch miteinander verknüpft.

12.5 Weiterverarbeitung im 8x8x8 |MEGATRON| Würfel

Die oben erwähnte 8*64 Bit Tabelle wird alle 8 Bits zerschnitten und in einen Würfel mit Kantenlänge 8 geschrieben. In Würfelform lassen sich die 512 Bit sehr kompakt verarbeiten. Bitshifts und logische Verknüpfungen können äußerst einfach durchgeführt werden. Die geometrische Form eines Würfels für die komplexe Datenverschlüsselung zu verwendetn tritt mit |MEGATRON| das erste Mal innerhalb einer Hashfunktion auf.

12.5.1 Cryptographic Rearrangement Array of Numbered Cubes

Der |MEGATRON|-DEA-Würfel wird Softwaretechnisch über ein 4 Dimensionales Array (*cranc*[4][8][8][8] cranc (Cryptographic Rearrangement Array of Numbered Cubes) realisiert. 3 Stufen sind als Koordinaten im Würfel zu verstehen, die vierte Stufe deutet mehrere (4) Würfel an, in welche unterschiedliche Varianten der Daten geschrieben werden. Würfel 0 und 1 werden, ebenfalls wie Würfel 2 und 3, nach Durchführung eines Bitshifts verknüpft.

```
for(i=0;i<8;i++)
{
        for(j=0;j<8;j++)
        {
                for(k=0; k<8; k++)
                {
                        cranc[0][i][j][k]=sblock[i][8*j+k];
                        cranc[1][i][7-k][7-j]=cranc[0][i][j][k]; //cranc
                            [1] gegenueber cranc[0] gespiegelt
```

```
                    cranc[2][i][k][j]=cranc[0][i][j][k]; //cranc[2] gegenueber cranc[0] um andere Achse gespiegelt
                    cranc[3][i][7-j][7-k]=cranc[0][i][j][k]; //cranc[3] gegenueber cranc[0] gespiegelt Oben-Unten verkehrt
            }
      }
}
```

12.5.2 cranc[ ][ ][ ][ ]-Bitshift und ⊕

Es werden nun die einzelnen Elemente des *cranc*[0] Würfels gegenüber jenen des *cranc*[1] Würfels verschoben und anschließend per exklusivem oder codiert. Der Zweck, den diese Operation erfüllt, ist nicht allzu absurd. Durch dieses Verfahren verschlüsselt man die eigegebenen Daten nicht nur mit einem Schlüssel, sondern ebenfalls mit sich selbst, was zu einer höheren Effizienz des Algorithmus führt.

```
for(i=0;i<8;i++)
{
      for(j=0;j<8;j++)
      {
            for(k=0; k<8; k++)
            {
                  if(cranc[0][i][(j+1)%8][k]==cranc[1][i][j][k]) cranc[0][i][j][k]=0;
                  else cranc[0][i][j][k]=1;
                  if(cranc[2][i][j][(k+1)%8]==cranc[3][i][j][k]) cranc[1][i][j][k]=0;
                  else cranc[1][i][j][k]=1;
            }
      }
}
```

Wir haben sozusagen von 4 Würfeln durch Verknüpfen von jeweils zweien nur mehr 2 übrig gebliebene Würfel. welche nocheinmal nach folgendem Schema mittels exklusivem oder codiert werden:

```
for(i=0;i<8;i++)
{
      for(j=0; j<8; j++)
      {
            if(cranc[0][0][i][j]==cranc[1][1][i][j]) cranc[0][0][i][j]=0;
//Wuerfel[0][0] mit Wuerfel[1][1]
            else cranc[0][0][i][j]=1;

            if(cranc[0][1][i][j]==cranc[1][2][i][j]) cranc[0][1][i][j]=0;
//Wuerfel[0][1] mit Wuerfel[1][2]
            else cranc[0][1][i][j]=1;

            if(cranc[0][2][i][j]==cranc[1][3][i][j]) cranc[0][2][i][j]=0;
//Wuerfel[0][2] mit Wuerfel[1][3]
            else cranc[0][2][i][j]=1;

            if(cranc[0][3][i][j]==cranc[1][0][i][j]) cranc[0][3][i][j]=0;
//Wuerfel[0][3] mit Wuerfel[1][4]
```

```
                    else cranc[0][3][i][j]=1;

                    if(cranc[0][4][i][j]==cranc[1][5][i][j]) cranc[0][4][i][j]=0;
//Wuerfel[0][4] mit Wuerfel[1][5]
                    else cranc[0][4][i][j]=1;

                    if(cranc[0][5][i][j]==cranc[1][6][i][j]) cranc[0][5][i][j]=0;
//Wuerfel[0][5] mit Wuerfel[1][6]
                    else cranc[0][5][i][j]=1;

                    if(cranc[0][6][i][j]==cranc[1][7][i][j]) cranc[0][6][i][j]=0;
//Wuerfel[0][6] mit Wuerfel[1][7]
                    else cranc[0][6][i][j]=1;

                    if(cranc[0][7][i][j]==cranc[1][4][i][j]) cranc[0][7][i][j]=0;
//Wuerfel[0][7] mit Wuerfel[1][4]
                    else cranc[0][7][i][j]=1;
          }
}
```

12.5.3 Abschließendes Zusammenfügen der Ebenen des Würfels

Als nächster Schritt werden die einzelnen Ebenen des Würfels wieder in eine Tabelle geschrieben. wir reduzieren um eine Dimension, und gehen zurück zum *sBlock*-Array. Die diffizile Verarbeitungs-Methode wird derart fortgesetzt, dass die einzelnen Ebenen zueinander um je 90° verdreht werden, bevor sie aneinandergefügt werden.

```
for(i=0;i<8;i++)
{
          for(j=0;j<8;j++)
          {
                    for(k=0;k<8;k++)
                    {
                              if((i==0)||(i==5)) sblock[j][8*i+k]=cranc[0][i][k][7-j];
//0 Grad
                              if((i==3)||(i==4)) sblock[j][8*i+k]=cranc[0][i][7-k][7-j];
//90 Grad
                              if((i==1)||(i==7)) sblock[j][8*i+k]=cranc[0][i][7-k][j];
//180 Grad
                              if((i==2)||(i==6)) sblock[j][8*i+k]=cranc[0][i][k][j];
//270 Grad
                    }
          }
}
```

Bis hier her wird alles Runde für Runde durchgeführt. 32 Mal werden die einzelnen Zeilen der 8*64 Tabelle mit verschiedenen Schlüsseln codiert, in einen Würfel geschrieben, zueinander gespiegelt, verdreht und verschoben, und mehrmals mit exklusivem oder verknüpft.

12.5.4 Rückführen der Tabelle in eine Kette

Nun werden die einzelnen sBlöcke in verkehrter Reihenfolge zurück in das Array bitchain[ ] geschrieben, wobei sie wieder zu einer 512 Bit langen Bitkette zusammengehängt werden.

12.5.5 Die Pseudo-Hexadezimalmultiplikationen

Zur Vorbereitung der Ausgabe wird diese Bitkette in Pseudo-Hexadezimaldarstellung umgewandelt und in das Array hexmultiply[ ] geschrieben. Diese beruht auf dem Prinzip der konventionellen Hexadezimaldarstellung, statt der Buchstaben a-f wird für die Zahlen von 10 bis 15 aber die gewöhnliche Dezimalschreibweise verwendet. Diese eigens für den |MEGATRON| - Algorithmus von uns entwickelte Schreibweise stellt sich bei den finalen Multiplikationen als äußerst vorteilhaft heraus. Ziel dieser Multiplikationen ist es, dass jedes einzelne Zeichen direkten Einfluss auf die |MEGATRON| - Summe hat. Um den Effekt zu maximieren, wird der gesamte Operationsblock 20-mal ausgeführt.

```
for(i=0;i<20;i++)
{
for(j=0;j<128;j++)
{
for(k=0;k<128;k++)
{
if (j==0) hexmultiply[1][k]=((hexmultiply[0][j]+1)*(hexmultiply[0][k]+1))%19;
else hexmultiply[1][k]=((hexmultiply[0][j]+1)*(hexmultiply[1][k]+1))%19;
}
}
}
```

Zwei for - Schleifen durchlaufen über ihre Indizes j und k das gesamte Array hexmultiply[ ] und sprechen dabei jedes einzelne Zeichen an. Das Zeichen, welches über j angesprochen wird, wird dabei mit allen anderen Zeichen - welche über k angesprochen werden - paarweise multipliziert. Beim ersten Zeichen wird ein Puffer gefüllt, der zur weiteren Berechnung dient. Vom Ergebnis wird aufgrund innermathematischer, zahlentheoretischer Überlegungen nur die Kongruenz *mod* 19 betrachtet. Es muss beachtet werden, dass sich - sobald das Ergebnis einer Multiplikation 0 ist - dieses Ergebnis fortpflanzt, was im Endeffekt dazu führt, dass der Wert sämtlicher Zeichen auf 0 gesetzt wird. Um dies zu verhindern, wird jeder Wert vor der Multiplikation um eins vergrößert.

12.5.6 Output

Bei der Ausgabe wird das Array deztohex[ ] verwendet, welches auf 16 Feldern die char - Zeichen von 0 bis f für die Hexadezimalausgabe beinhaltet. Zur Konvertierung der Pseudo-Hexadezimalwerte in die konventionelle Schreibweise, wird das Array hexmultiply[ ], welches die Endergebnisse der Multiplikationen beinhaltet, als Index für deztohex[ ] verwendet, womit die entsprechenden Zeichen der konventionellen Hexadezimaldarstellung angesprochen werden. Die Werte aus hexmultipl[ ] werden dabei modulo 15 verwendet, da einzelne Hexadezimalzeichen dargestellt werden sollen. Schließlich befindet sich die verschlüsselte |MEGATRON| - Summe im Array hexout[], welches abschließend mittels einer for - Schleife ausgegeben wird.

12.6 Testphase

Wie bereits erwähnt, wurden schon während der Entwicklung laufend Tests durchgeführt, deren Erfahrungswerte wir sofort verarbeiteten. Als wir den Algorithmus vorläufig zu unserer Zufriedenheit optimiert hatten, gingen wir in eine intensive Testphase, bei der wir vor allem untersuchten, inwieweit |MEGATRON| der Forderung, dass selbst kleine Differenzen im Input große Differenzen beim Ausgabewert erzeugen, nachkommen kann. Die Pseudo-Hexmulitplikationen hatten hier bereits eine sehr gute Basis geschaffen, doch durch systematisches Suchen nach Kollisionen fanden wir einige Ausnahmefälle, in denen die verschlüsselten Nachrichten identisch waren oder zumindest einander ähnelten. Durch aufwändige Analysen, bei denen wir im Endeffekt den Weg von einzelnen Bits durch den Algorithmus verfolgten, entdeckten wir schließlich einige Fehler in der Verarbeitung, welche wir durch Modifikation der Funktion beheben konnten. So zieht in der aktuellen Version 1.1 bereits die Änderung eines einzelnen Zeichens eine komplette Änderung der verschlüsselten Nachricht nach sich.

12.7 Zusammenfassung

Im Laufe der Woche konnten wir die einzelnen Schritte von Planung, Design des Algorithmus und dessen programmiertechnischer Umsetzung bis hin zu intensiven Tests erfolgreich abschließen. Zum jetzigen Zeitpunkt sind uns keine Schwachpunkte, wie zum Beispiel Kollisionen, bekannt, weshalb wir die Entwicklung mit Version 1.1 vorläufig als abgeschlossen betrachten. Dennoch bestehen einige noch nicht realisierte Ideen zum weiteren Ausbau des Algorithmus. Alles in allem hat die Arbeit an diesem Projekt im Rahmen der Modellierungswoche unseren Horizont in vielerlei Hinsicht erweitert. Problemstellungen innerhalb der Kryptographie sind uns ebenso auf sehr praxisorientierte Art und Weise nähergebracht worden, wie uns der mathematische Hintergrund - vor allem in Bezug auf Zahlentheorie und das binäre Zahlensystem - vertraut gemacht wurden. Die Kombination mit praktischen Anwendungen aus dem Bereich der Informatik stellte für uns dabei einen besonderen Reiz dar.

13 Quellen

Florian Andrisch, Alexander Bors, Wolfgang Hrauda, Stefanie Kaiser, Paul Laufer, Florian Mikulik

Da auch wir keinesfalls alles wissen konnten, bevor wir unsere Arbeit an den bisher beschriebenen Themen begannen, bedienten wir uns zwecks geistiger Weiterbildung folgender Quellen:

- Prof. Günter Lettl
- Kryptographie; Albrecht Beutelspacher, vieweg 2005
- Moderne Verfahren der Kryptographie; Albrecht Beutelspacher, Jörg Schwenk, KLaus Dieter Wolfenstetter, vieweg 2004
- Einführung in die Kryptographie; Johannes Buchman, Springer 2001
- Kryptographie in Theorie und Praxis; Albrecht Beutelspacher, Heike B. Neumann, Thomal Schwarzpaul, vieweg 2005
- Cryptography: An Introduction; V.V.
- Yaschenko, Americam Methematical Society 2002
- Public Key Cryptography; Arto Salomaa, Springer 1996
- Aus dem Internet:
  - http://www.faqs.org/rfcs/rfc1321.html
  - http://de.wikipedia.org/wiki/Md5
  - http://de.wikipedia.org/wiki/DES
  - http://www.matheprisma.de/Module/DES/index.htm

Peak Oil

Projektleiter: Stephen Keeling

Katharina Albert, Lukas Andritsch, Teodora Corduneanu,
Elisabeth Gaar, Lisa-Maria Sommer, Daniel Steuber

Inhaltsverzeichnis:

Problemstellung: Peak Oil

Peak Oil (=Ölfördermaximum) bezeichnet den Zeitpunkt, ab dem die Ölproduktion ihren Höhepunkt erreicht.

In vielen Regionen der Welt wurde der Höhepunkt der Erdölforderung bereits erreicht in den 90er Jahren erreicht. Mittlerweile wurde die weltweite Spitze der Erdölproduktion erreicht und unser Erdölvorrat könnte in naher Zukunft erschöpft sein.
Die Entdeckung der Erdölreserven erreichte ihren Höhepunkt schon in den sechziger Jahren. Heutzutage finden wir ein Barrel Öl für vier, die wir konsumieren.

Der Großteil der modernen Welt ist von der Verfügbarkeit der Ressource Erdöl abhängig, denn jeder von uns hat umgerechnet 300 Energiesklaven. Allerdings lässt sich das Erdöl nicht leicht durch alternative Energieträger ersetzen. Das Angebot wird die steigende Nachfrage, die sich aus dem ständigen Bevölkerungswachstum ergibt, nicht mehr decken können.
Dementsprechend werden Wirtschaftsbereiche, die auf Rohöl angewiesen sind, auf Dauer höhere Preise in Kauf nehmen, ihren Verbrauch verringern oder effizientere Technologien und andere Rohstoffquellen einsetzen müssen.

Das Ziel unseres Projekts ist es, ein Modell zu entwickeln, das möglichst viele Einflüsse auf den Erdölpreis zeigt, und sowohl die Entwicklung der Bevölkerung in Abhängigkeit des Öls beinhaltet erklärt, als auch die Entdeckung und Produktion beinhaltet.

Logistische Funktion

Als allgemeine Formel gilt:

$$f(t) = \frac{K}{1 + e^{-\frac{t - t_0}{\tau}}}$$

Hierbei gilt

K…Maximale Kapazität

e…Euler'sche Zahl (2,718281828459)

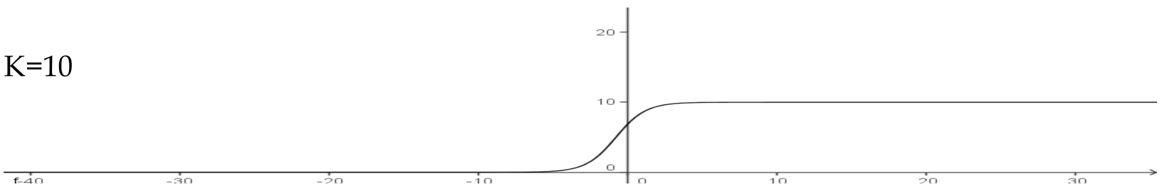
t_0…Wendepunkt der Kurve

τ…Parameter

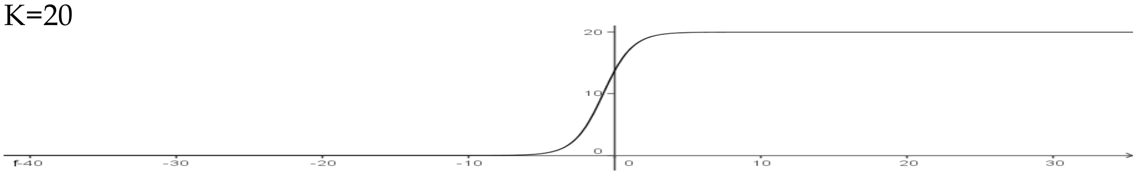
K wirkt sich auf die maximale Ausdehnung des Graphen in Y-Richtung aus

Beispiele dafür sind:

K=10



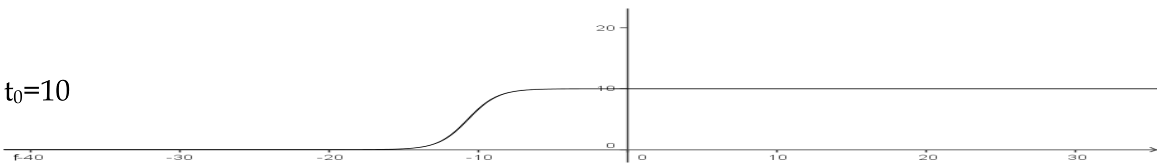
K=20



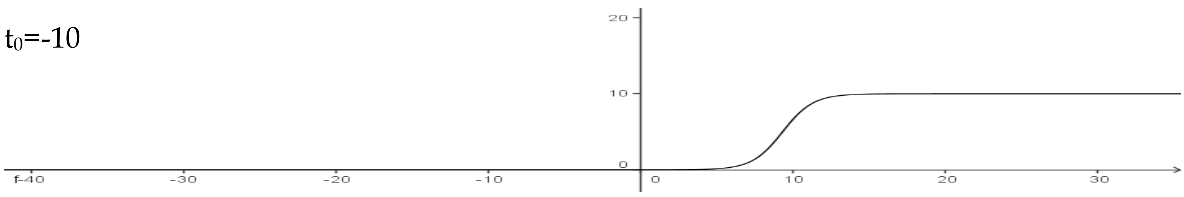
t_0 wirkt sich auf den Wendepunkt der Kurve aus.

Beispiele dafür sind:

t_0=10



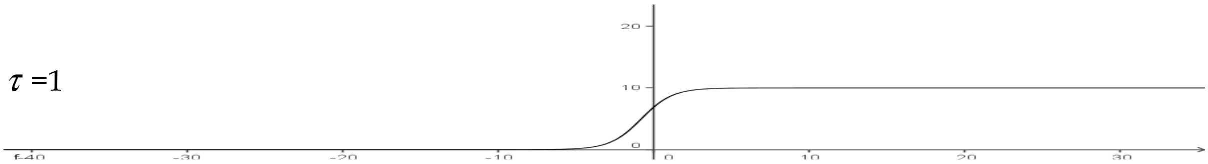
t_0=-10



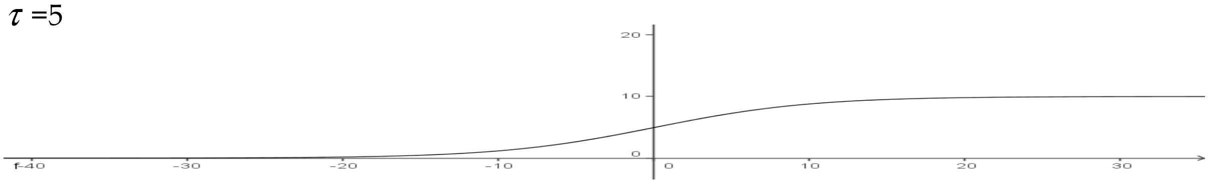
τ wirkt sich auf die Steigung der Kurve aus

Beispiele dafür sind:

τ =1



τ =5



Die logistische Funktion wird sehr häufig im Zusammenhang mit Wachstumsprozessen verwendet. Diese verbindet 2 wichtige Eigenschaften in sich.

- Anfangs($-\infty$ bis t_0) steigt sie wie eine Exponentialfunktion
- Danach (t_0 bis ∞) wird die Steigung wieder geringer und die Kurve nähert sich dem Wert K immer mehr an, wobei dieser aber erst im Unendlichen erreicht wird.

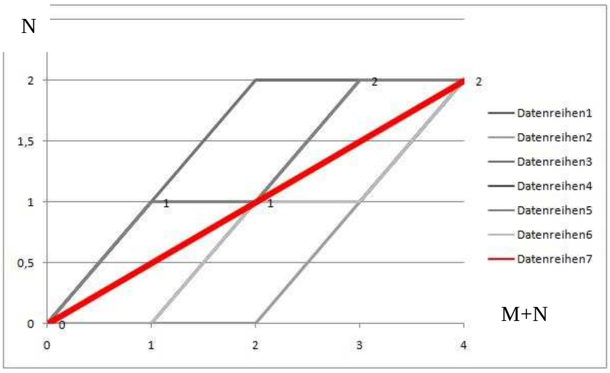
Diese Eigenschaften werden bei einem Bevölkerungswachstumsmodell folgendermaßen interpretiert:

- Anfangs wächst die Bevölkerung annähernd exponentiell.(z.B: um 3%)
- Danach kann aufgrund der beginnenden Knappheit von Ressourcen die Bevölkerung nicht mehr uneingeschränkt wachsen und das Wachstum nimmt ab. Dadurch kann die maximale Kapazität K(Nahrung, Lebensraum, Sauerstoff…) in einer endlichen Zeitspanne nicht überschritten werden. Die Kurve nähert sich aber immer mehr an K an

Man sieht schon, dass dieses Modell, das anfangs für das Wachstum von Bakterienkulturen verwendet wurde, auch für viele andere Zwecke verwendet werden kann. Auch in unseren Modellen spielt diese Funktion eine wesentliche Rolle.

Entdeckung

Wenn man sich die Welt als ein riesiges Gitter voller Felder vorstellt, die entweder Öl enthalten oder nicht, und man zufällig eines nach dem anderen aufbohrt, so lässt sich der Graph der Funktion der Entdeckung immer als **Gerade** darstellen: Die Wahrscheinlichkeit für jeden auftretenden Fall (in welcher Reihenfolge man die vollen bzw. leeren Felder aufbohrt) ist gleich groß, d.h. jeder Fall tritt nach oftmaligem Probieren gleich oft auf und somit ist der Durchschnitt eine Gerade. (Dies gilt für jegliche Anzahl von Öl- und Antiölfeldern).



N….Anzahl der Felder mit Öl
M…Anzahl der Felder ohne Öl

X	
X	

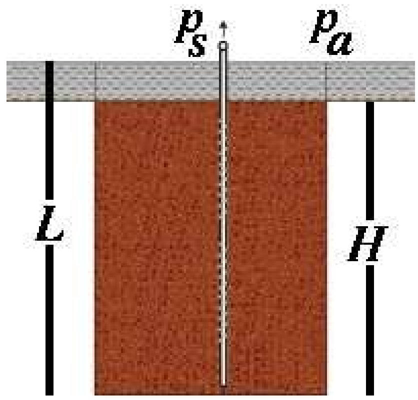
In dieser Grafik ist das Beispiel von einem 2x2 Feld dargestellt (die Felder mit Öl sind mit X markiert), das Modell gilt aber auch für jegliche Anzahl an Feldern.

Doch aufgrund der Erfahrungen die der Mensch macht (niemand bohrt freiwillig ein zweites Mal in einen Berg wenn er beim ersten Mal nichts findet!!!), ist der Graph der Funktion eine **logistische Kurve**: nach anfänglichen Startschwierigkeiten, bei denen der Suchende willkürlich die Gegend absucht, kann er nach einiger Zeit die Eigenschaften des Bodens immer genauer bestimmen und aufgrund des Auswahlverfahrens steigt die Wahrscheinlichkeit des Erfolges. Nachdem ein Großteil der Ölfelder gefunden wurde, fällt die Häufigkeit des Entdeckens und die Kurve flacht ab.

Außerdem zeigen die Daten dass es eindeutig eine logistische Kurve ist.

Physikalischer Ansatz: Das Brunnenmodell - Ölproduktion

Hierbei wird die Ölproduktion anhand eines einfachen Brunnenmodells gezeigt.
Der Brunnen ist bis oben hin mit Öl gefüllt (H), dieses Öl wird mit einer Pumpe, deren Rohr bis an den Grund des Brunnens reicht, gefördert. Wir gingen davon aus, dass die Pumpe mit einer konstanten Pumpenleistung arbeitet.



h(t)	Höhe zum Zeitpunkt t
H = h(0)	Ausgangshöhe des Öls
A	Grundfläche des Brunnens (konstant)
P(t)	Produktion zum Zeitpunkt t
F	Flussrate
V(t)	Volumen zum Zeitpunkt t
β	Flussrate/Volumen
O	Konzentration des Öls zum Zeitpunkt t
l	Ölspiegel zu Zeit t
L	Höhe des Brunnens
Ps	Pumpendruck
Pa	Atmosphärischer Druck

Damit Öl gefördert werden kann, muss ein zusätzlicher Druck ausgeübt werden, der Pumpendruck, dieser muss gleich sein wie die Flussrate und der Widerstand der auf das Öl im Rohr wirkt.

$$\Delta P(l) = F \cdot W(l)$$

Durch das Miteinbeziehen von dem Erhaltungsgesetz, stellten wir folgende Formel auf:

$$Ps + \rho gL + FW(L) = Pa + \rho gh$$

Da sich der atmosphärische Druck mit dem Pumpendruck ($Ps = Pa - \rho gL$) und dem Pumpendruck aufhebt, ergibt sich die Flussrate, die gleich ist wie $-V' \to ah'$.

$$F = \frac{Pa - Ps + \rho g(L)}{W(L)} = -V' = -Ah'$$

Da $F = -V' = ah'$, erhalten wir daher:

$$h' = -\frac{\rho g}{AW(L)} h$$

$\frac{\rho g}{AW(L)}$ wird zu β zusammengefasst, daher:

$$h'(t) = -\beta \cdot h(t)$$

$$-\beta = \frac{h'(t)}{h(t)}$$

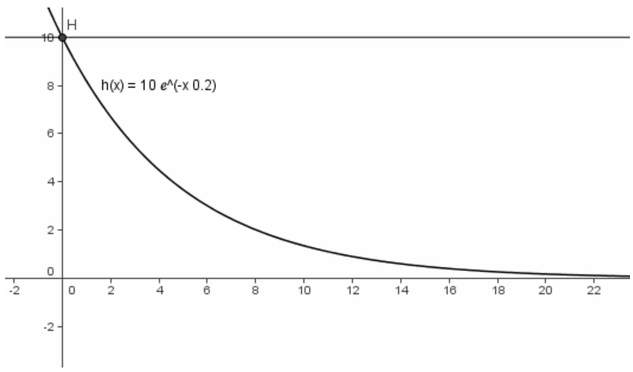
$$\int_0^t \frac{h'(t)}{h(t)} dt = \ln|h(t)| \Big|_0^t$$

$$-\beta t = \ln\left|\frac{h(t)}{H}\right|$$

$$e^{-\beta t} = e^{\ln\left|\frac{h(t)}{H}\right|}$$

$$e^{-\beta t} = \frac{h(t)}{H}$$

$$h(t) = He^{-\beta t} = He^{-\frac{F}{V}t} \qquad \beta = \frac{F}{V}$$



Die Kurve stellt den Ölspiegel im Brunnen dar.

Da die Menge an Öl die wir aus dem Brunnen produziert haben, gleich dem Volumen vom Anfang minus dem Volumen dass sich noch im Brunnen befindet, ist, können wir die errechnete Formel für *h(t)* in folgende Formel einsetzen:

$$P(t) = V_0 - V(t)$$
$$P(t) = AH - Ah(t)$$
$$P(t) = A\left(H - He^{-\beta t}\right)$$
$$P(t) = V_0\left(1 - e^{-\beta t}\right) = V_0\left(1 - e^{-\beta t}\right)$$

Durch Ableitung von P ergibt sich

$$P' = \beta(V_0 - P) = \beta(E - P)$$

Da unser Brunnen die einzige Ölquelle ist, können wir annehmen, dass $V(0) = E$

Durch Nachforschungen im Internet haben wir herausgefunden, dass viele Ölquellen während dem Herauspumpen vom Erdöl mit Salzwasser aufgefüllt werden, um den Druck auszugleichen. Dabei ist allerdings darauf zu achten, dass die Konzentration von Erdöl im Geförderten sinkt.
Durch das Erhaltungsgesetzt kommen wir auf folgende Formel:

$$V_0 O' = -FO \qquad O = \frac{V(t)}{V_0}$$

$$V_0 \frac{V(t)}{V_0 t} = \frac{V}{t} \frac{V(t)}{V_0}$$

$$\frac{V(t)}{t} = \frac{V(t)}{t}$$

$$P(t) = V_0 - V(t)$$

$$P(t) = V_0 - V_0 O$$

$$P(t) = V_0 (1 - O)$$

Um unsere Formel für die Produktion nicht nur auf einen Brunnen anwenden zu können, sondern auf beliebig viele Ölquellen, passten wir die Formel für mehrere Brunnen an, allerdings unter der Annahme, dass alle Brunnen die gleichen Eigenschaften haben.

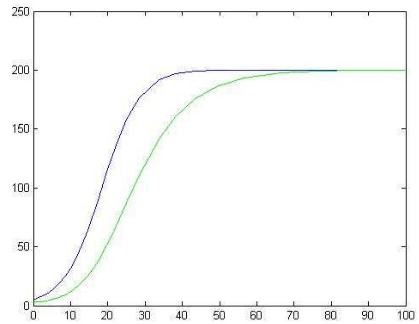
$$P_i' = \beta_i \left(P_i^{\max} - P_i\right)$$

$$P' = \sum_{i=1}^{N} \cdot P_i'$$

$$P' = \beta\left(\sum_{i=1}^{N} P_i^{\max}\right) - \beta\left(\sum_{i=1}^{N} P_i\right)$$

$$P' = \beta(E - P)$$

Kopplung von Entdeckung und Produktion



$$E' = \alpha E(M - E)$$

$$P' = \beta(E - P)$$

Wenn man Entdeckung und Produktion koppelt sehen beide logistisch aus.

Wirtschaftlicher Ansatz: Wirtschaftssystem

Bei unserem Wirtschaftssystem versuchen wir, die wichtigsten Einflüsse auf den Ölpreis zu bestimmen.
Um es zu vereinfachen nehmen wir an die einzige Ölfirma zu sein (ein richtiges Monopol!).
Laut den Gesetzten des freien Marktes Bestimmen somit Angebot und Nachfrage den Preis.

C	Preis
Co	Mindestpreis
Cm	Maximalpreis
V	Vorrat
B	Bevölkerung
κ	Verbrauch/Person

Um zur Funktion des Angebotes zu kommen, berücksichtigten wir, dass wir nicht mehr als den Vorrat anbieten können und dass unser Preis nicht kleiner als unserer Mindestreis (beinhaltet alle Kosten) sein soll.

Das Angebot in einer gewissen Zeitspanne (dt) ist somit: $A(t+dt)-A(t)=V\cdot\left(1-\left(\frac{c_0}{c}\right)^{dt}\right)$

Wobei A(t) dass gesamte Angebot von -∞ bis jetzt ist.
Danach dividiert man durch dt und berechnet den Limes um zu Veränderung von A zu kommen.

$$A'(t)=\lim_{dt\to 0}\frac{V\cdot\left(1-\left(\frac{C_0}{C}\right)^{dt}\right)}{dt}$$

Das Ergebnis lautet dann: $A'(t)=V\cdot\ln\left(\frac{C}{C_0}\right)$

Die Nachfrage ist wiederum direkt von der Bevölkerung und indirekt vom Preis abhängig, die Konstante κ bestimmt den Verbrauch an Öl pro Person.

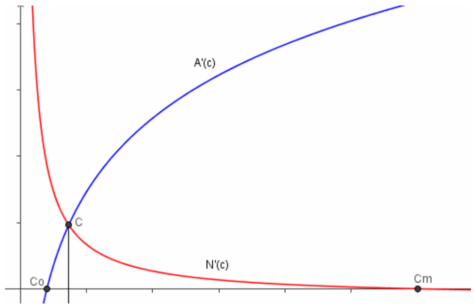
In einer gewissen Zeitspanne also: $N(t+dt)-N(t)=B\cdot\kappa\cdot dt\cdot\left(\frac{1}{C}-\frac{1}{C_m}\right)$

N(t) ist hier wieder die Nachfrage von -∞ bis jetzt.

$$A'(t)=\lim_{dt\to 0}\frac{B\cdot\kappa\cdot dt\cdot\left(\frac{1}{C}-\frac{1}{C_m}\right)}{dt}$$

Das heißt: $N'(t)=B\cdot\kappa\cdot\left(\frac{1}{C}-\frac{1}{C_m}\right)$

Der Schnittpunkt von N und A ist somit C, der Ölpreis.



$$N' = A'$$

$$B \cdot \kappa \cdot \left(\frac{1}{C} - \frac{1}{C_m}\right) = V \cdot \ln\left(\frac{C}{C_0}\right)$$

$$\frac{B \cdot \kappa}{V} - \frac{\ln\left(\frac{C}{C_0}\right)}{\frac{1}{C} - \frac{1}{C_m}} = f(c) = 0$$

Der Preis ergibt sich aus Gleichsetzung von Angebot und Nachfrage.

Mit Hilfe des Newton - Verfahrens wurde die Nullstelle der Funktion berechnet, denn an dieser Stelle befindet sich der Preis C.

Das Kapital unserer Firma berechnet sich aus Einnahmen und Ausgaben.
Die Einnahmen sind C mal das verkaufte Öl oder auch C mal die Nachfrage.
Die Ausgaben sind Co mal das Angebot plus etwaige Fixkosten (γ).

→ $K' = CN' - C_0 A' - \gamma$

Um Gewinnmaximierung zu betreiben (nicht vergessen - wir sind ein böses Ölmonopol) hören wir sofort zum Produzieren auf, wenn wir keinen Gewinn mehr machen, das heißt wenn $K' \leq 0$ ist.

Bevölkerungsformel:

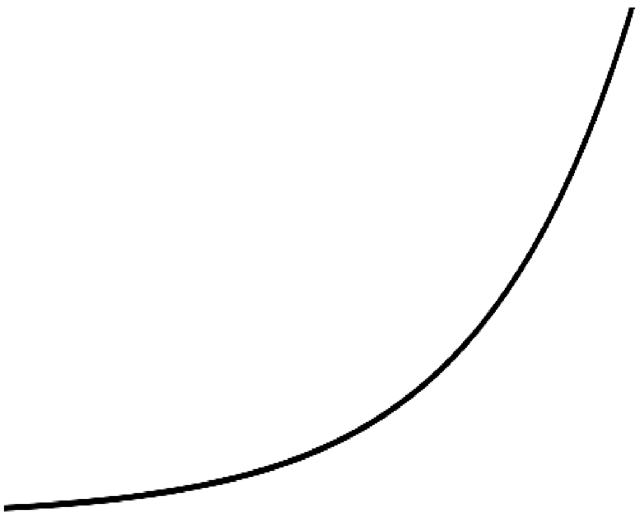
Da die Bevölkerung nicht gleich bleibt, muss auch für diese eine Formel gefunden werden. Daher überlegten wir uns, dass wir einen Zusammenhang zwischen vorhandenem Erdöl und der Bevölkerung herstellen sollten. Wir nahmen an, dass die Bevölkerung jedes Jahr um einen konstanten Wert wächst. Daraus ergab sich unser erstes, sehr leicht zu behandelndes Modell, eine Exponentialfunktion. Die allgemeine Formel dafür lautet:

$$f(x) = f_0 * k^x$$

Hierbei gilt

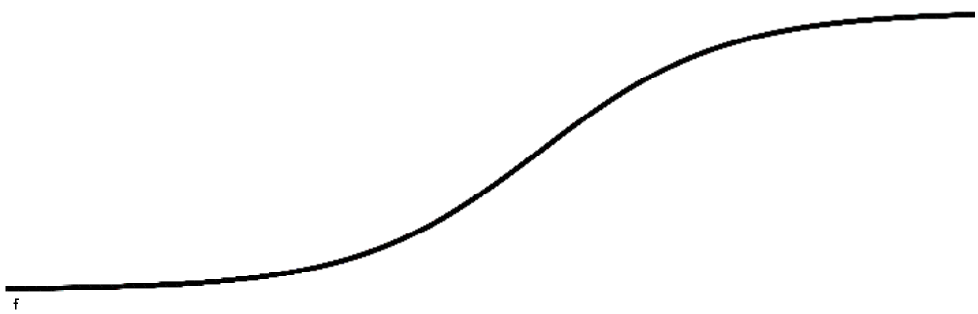
f_0...Startwert für die Funktion

k…Wachstumsparameter (bei Zunahme >1 bei Abnahme <1)



Bei dieser Variante geht man von unbegrenzten Ressourcen aus. Als wir allerdings realistische Daten gesucht haben, mussten wir einsehen, dass die Realität anders aussieht, und unsere Formel noch musste stark verändert werden. Nach längerem Nachdenken kamen wir zu

dem Schluss, eine logistische Funktion zu verwenden. Diese hat den Vorteil, dass das Wachstum nicht unbegrenzt ist, sondern nach einer bestimmten Zeitspanne wieder abnimmt. Dieses Modell sieht dann so aus:



f

Leider entspricht auch dieses Modell unserer Meinung nach nicht der Realität, denn wir glauben, dass die Bevölkerung vom aktuellen Angebot von Erdöl abhängig ist. Wenn am Markt kein Erdöl mehr angeboten wird, darf auch die Bevölkerung nicht mehr wachsen. Stattdessen sollte sie, solange kein Öl angeboten wird, sinken. Außerdem sollte noch bedacht werden, dass man mit alternativer Energie einen gewissen Bedarf an Energie decken kann, unabhängig davon ob die Erdölressourcen aufgebraucht sind oder nicht. Unser neuentwickeltes Modell lautet dann:

$$B' = \nu * (F - B) * B + \rho * A'$$

ν …Gewichtung für (F-B)*B

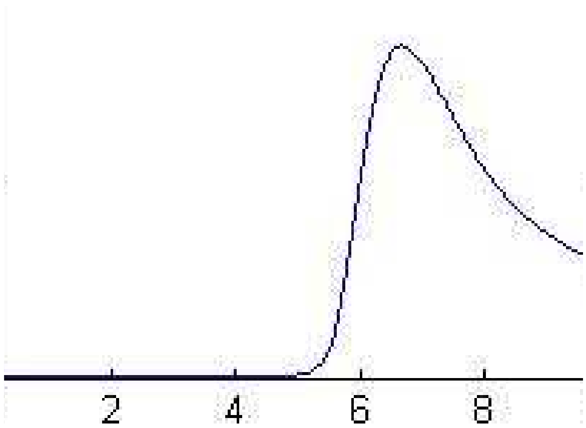
F… Maximalbevölkerung ohne Erdöl(nur Alternativen)

B…Momentane Weltbevölkerung

ρ …Gewichtung von A'

A'…Momentanes Angebot an Erdöl

Ein möglicher Graph dafür könnte so aussehen:



Man sieht nun folgendes:

- Solange genug Erdöl angeboten wird, steigt die Bevölkerung(bis ca. 6)
- Wenn Erdölknappheit herrscht, beginnt die Bevölkerung zu sinken
- Wenn kein Erdöl mehr angeboten wird, sinkt die Weltbevölkerung auf den Wert F

Das Modell in MatLab

Da wir unser Modell im Programm „MatLab“ erstellen wollten, sich aber niemand von uns damit wirklich auskannte, war es unsere erste Aufgabe, uns mit MatLab vertraut zu machen. Das geschah durch einige Übungsbeispiele, die uns Steve Keeling vorrechnete und als Modell grafisch darstellte. Nach 2 solchen Beispielen probierten wir das erste Mal, MatLab für eine Simulation von unseren hergeleiteten Formeln einzusetzen, doch bald mussten wir erfahren, dass wir noch weit zu wenig Überblick über MatLab hatten und wir scheiterten schon am Zeichnen der Graphen von Entdeckung und Produktion von Erdöl. Deshalb holten wir einmal mehr unseren Steve Keeling zur Hilfe und er erklärte uns den Aufbau und die Funktionsweise von MatLab anhand unseres Brunnenmodells in Zusammenhang mit der Entdeckung. Da das Modell und die Grafik schon weiter oben erklärt wurden, wird an dieser Stelle nicht mehr weiter darauf eingegangen.

Unser nächster Schritt war der wichtigste im ganzen Projekt, denn jetzt begann das eigentliche Modellieren. Wir haben alle Formeln zusammengefasst und versucht ein zusammenhängendes Modell zu erstellen, in dem man jeweils die Graphen von Entdeckung, Produktion, Vorrat, Angebot, Nachfrage, Preis und Kapital in Abhängigkeit von Zeit sieht.
Unsere ersten Schritte waren noch nicht so schwer, und es dauerte auch nicht außergewöhnlich lange, bis wir die ersten Graphen auf dem Bildschirm hatten, doch zu diesem Zeitpunkt fingen unsere Probleme erst an.
Die Funktionen nahmen manchmal die wildesten und unmöglichsten Formen an, darunter Spritzen, Flaschen, Bücher und Wurzelzeichen. Manchmal bestanden die Funktionen nicht aus einer Linie sondern aus vielen Dreiecken oder kleinen Pünktchen, doch der Höhepunkt wurde erreicht, als man in den Funktionsgraphen sogar Buchstaben erkennen konnte und so schienen sie uns „MAUS“ oder „SAUF“ mitzuteilen.
Durch ein paar Änderungen im Quellcode bekamen wir dann Grafiken, die halbwegs annehmbar aussahen, doch wir hatten weiterhin einige Deutungsschwierigkeiten der Kurven. Durch die Veranschaulichung unserer Funktionen kamen wir hinter ein paar kleine Verbesserungsmöglichkeiten für unsere Formeln, so wurde die Bevölkerungsformel mindestens 4 Mal überarbeitet. Ein weiteres Problem war die Division durch Null, denn beim Integrieren war es einmal nötig durch den Vorrat zu dividieren, und so stürzte das Programm oft ab, wenn der Vorrat verbraucht wurde.
Ein wesentlicher Bestandteil unseres Modells ist das Newton Verfahren, das dazu dient, den Schnittpunkt zwischen Angebot und Nachfrage zu berechnen. Zwei weitere Dateien dienen hauptsächlich dazu die Formeln und die Werte der Parameter zu bestimmen. Um besser zu veranschaulichen, wie unser Modell funktioniert, folgt der Quelltext und als Kommentar die Erklärung:

Ölsim.m:

```
global a b c0 cm d g imax k m p r tol %Definiert Variablen, die in mehreren
      Dateien vorkommen
a=0.5e-2; %Gibt die Werte der Variablen für die Entdeckung an
m=1.5e2;
b=1e-2; %Gibt die Werte der Variablen für die Produktion an
k=0.05; %Gibt die Werte der Variablen für die Nachfrage an
c0=2.0e0; %Gibt die Werte der Variablen für das Angebot an
cm=9.1e1;
g=31.4352; %Gibt die Werte der Variablen für das Kapital an
r=5.0e2; %Gibt die Werte der Variablen für die Bevölkerung an
d=1.0e-3;
p=1.0e2;
tol = 1.e-3; %Gibt die Werte der Variablen für numerische Funktionen an
imax=100;
E0=9.0e-1; %Gibt den Anfangswerte von E an
P0=7.0e-12;
V0=3.0e1;
K0=2.5e1;
B0=50;
N0=P0-V0;
A0=N0;
x0=[E0;P0;N0;A0;V0;K0;B0]; %Anfangswerte zur Lösung des Modells
tspan=[0,20]; %Gibt das Zeitintervall für die Lösung des Modells an
[t,x] = ode45(@Ölrates,tspan,x0); %ode45 ist eine Funktion von MatLab zur
      Lösung von Differentialgleichungen
E = x(:,1); %Definiert E als die erste Spalte der Matrix

B = x(:,7);
c=zeros(size(t));
for i=1:length(t)
    c(i)=Newton(B(i),V(i));
end
subplot(4,2,1); %Definiert die Einteilung der Unterbilder
plot(t,E); %Befehl E(t) zu zeichnen
title('Entdeckung'); %Gibt die Beschriftung an

subplot(4,2,8);
plot(t,K);
title('Kapital');
```

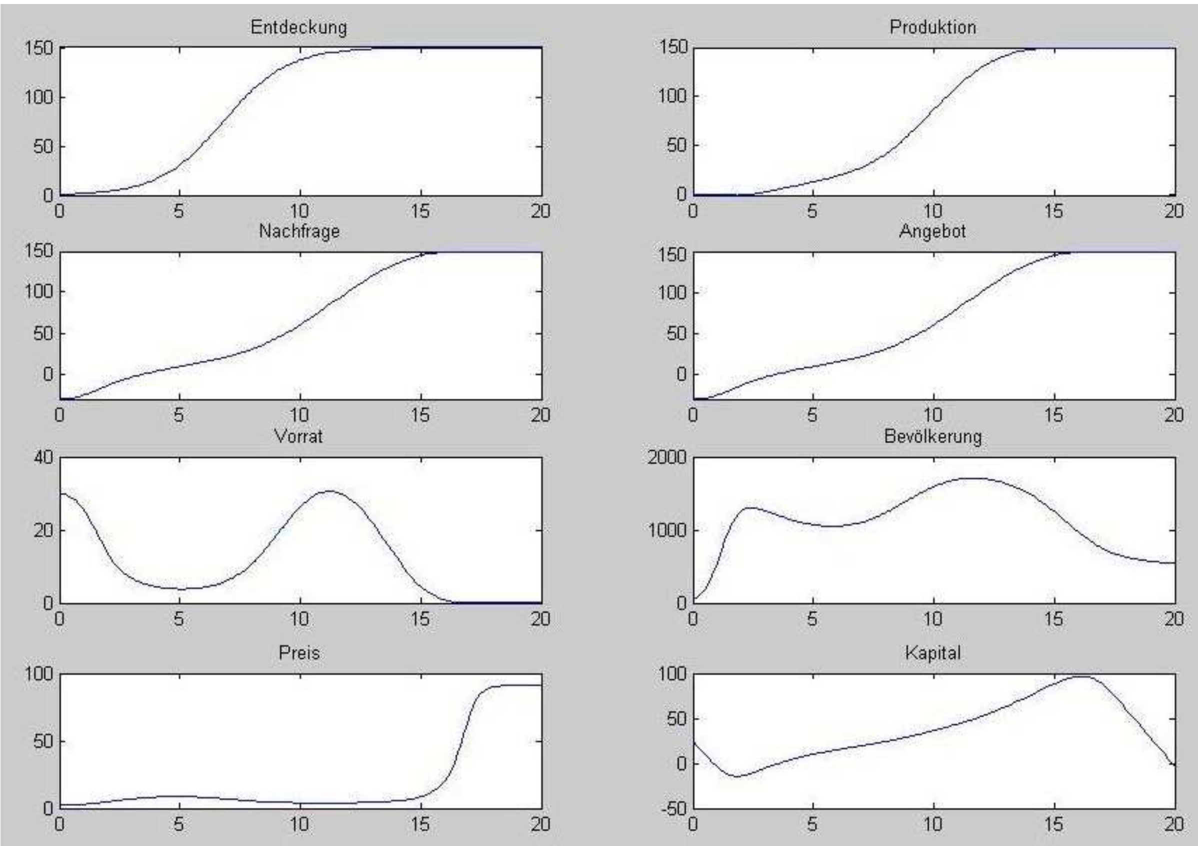
Ölrates.m:

```
function f = Ölrates(t,x) %Gibt den Namen der Funktion f an
global a b c0 cm d g k m p r %Definiert Variablen, die in mehreren Dateien
      vorkommen
E=x(1); %Definiert, dass E der erste Vektor ist
B=x(7);
c=Newton(B,V); %Gibt die Formel für c an, die von der Datei Newton bezogen
      wird
Es=a*E*(m-E);
Ns=k*B*((1/c)-(1/cm));
As=V*log(c/c0);
Ks=c*Ns-c0*As-g;
Ps=b*(E-P)*(Ks>0)*Ks^2;
Vs=Ps-Ns;
Bs=d*(r-B)*B+p*As;
f=zeros(7,1); %Gibt an, dass die Matrix 7 Spalten und 1 Reihe hat
f(1)=Es; %Definiert, dass gilt Es=E
f(7)=Bs;
```

Newton.m:

```
function c = Newton(B,V) %Gibt den Namen der Funktion an
global c0 cm k tol imax %Definiert Variablen, die in mehreren Dateien
      vorkommen
c=2*c0; %Gibt an, dass unser Preis doppelt so hoch wie unser Mindestpreis
      sein soll
ca = (1+2*tol)*abs(c);
i=0;
while ((abs(c-ca) > tol*abs(c)) && (i <= imax)) %Gibt unsere Newton-
      Funktion an, um den Schnittpunkt von N und A, also c zu berechnen
    ca=c;
    f  = V*(log(c/c0))-k*B*(1/c-1/cm);
    fs = V/c+k*B/c^2;

    if (fs==0) %Definiert den Fall, dass durch Null dividiert wird
         disp('fs = 0!!!');
    end

    if (fs~=0) %Definiert den Fall, dass nicht durch Null dividiert wird
        fs=1/fs;
    end

    c=c - f*fs;
    c=max([c*(1+tol),c0]);
    i=i+1;
end
```

Mit diesem Quelltext kamen wir auf folgendes Modell:



Entdeckung: Da die Entdeckung in unserem System nur von einem Parameter und der maximalen Ölfördermenge und von sonst nichts abhängig ist, sieht die Kurve immer gleich aus, nämlich wie eine Logistische Funktion, von der sich nur die Steigung verändert. Die Entdeckung beeinflusst aber in weiterer Folge die höchstens mögliche Geschwindigkeit der Produktion.

Produktion: Bei diesem Graph sieht man deutlich, dass es eine gewisse Zeit dauert, bis das entdeckte Öl auch produziert werden kann, was in der Realität durch statistische Daten bewiesen ist. Weiteres sieht man deutlich, dass die Produktion noch nicht einsetzt, wenn das Kapital sinkt, sie aber anfängt, wenn das Kapital anfängt zu steigen. Da das Kapital steigt bis alle Funde produziert worden sind, ist es logisch, dass die Produktion nie aussetzt.

Nachfrage und Angebot: Da wir den Preis als den Punkt gewählt haben, in dem sich Angebot und Nachfrage treffen, ist es logisch, dass die Graphen der Nachfrage und des Angebots gleich verlaufen. Man erkennt, dass die Nachfrage mit der Entdeckung und Produktion des Öls steigt, aber kurzzeitig weniger ansteigt, wenn die Bevölkerung sinkt.

Vorrat: Den Anfangswert vom Vorrat haben wir definiert. Da eine Nachfrage herrscht, wir aber noch nicht produzieren können, da noch nichts entdeckt worden ist, sinkt unser Vorrat. In weiterer Folge wird die Produktion gesteigert, die Nachfrage steigt aber nicht in gleichem Maß an, deshalb steigt unser Vorrat wieder. Da dann aber die Bevölkerung wächst und dadurch mehr Erdöl verbracht wird, wird der Vorrat immer leerer und ist schlussendlich aufgebraucht, wenn man alles Entdeckte schon produziert und verkauft hat.

Bevölkerung: Die Bevölkerung hat einen von uns bestimmten Startwert. Dadurch, dass unser Vorrat mit Erdöl gefüllt ist, können wir den Menschen genug Erdöl verkaufen und die Bevölkerung steigt. Dann nimmt sie kurzzeitig ab, da unser Vorrat sinkt und wir nicht mehr so viel Öl anbieten können. Da allerdings unser Vorrat bald wieder ansteigt, hat die Bevölkerung wieder eine gesättigte Nachfrage und die Bevölkerung steigt weiter. Trotzdem ist das maximale Erdölvorkommen bald abgebaut und verbraucht. Dadurch sinkt die Bevölkerung auf die Maximalbevölkerung ohne Erdöl.

Preis: Der Preis wird nach einiger Zeit etwas teurer, da unser Vorrat zur Neige geht. Da dieser aber bald wieder voller wird, sinkt der Preis wieder. Wenn aber dann der Zeitpunkt kommt, an dem die Erdölreserven langsam aber sicher ausgehen, steigt der Preis gewaltig an, da jeder dieses Erdöl haben möchte.

Kapital: Am Anfang machen wir mit unserer Firma Verluste, da wir einen großen Vorrat haben, der Preis dadurch niedrig ist und wir wenige Einnahmen haben, aber wir trotzdem für die Fixkosten aufkommen müssen. Sobald der Vorrat gering ist, können wir einen höheren Preis einnehmen und machen daher Gewinn, außerdem können wir später auch das, was neu produziert wurde, verkaufen. Unseren größten Gewinn machen wir bei fast leerem Vorrat da der Preis extrem hoch ist. Doch sobald unser Vorrat leer ist und das ganze Erdöl verbraucht worden ist, verkaufen wir nichts mehr, da wir aber leider noch die Hypothek für unsere Raffinerie bezahlen müssen, machen wir Verluste.

Zusammenfassung

Was passiert, wenn man mathematikbegeisterte Schüler und Schülerinnen vor ein fast unlösbares Problem stellt, nämlich Prognosen über die Zukunft des Erdöls zu entwickeln?

Trotz ursprünglicher Begeisterung, fühlten wir uns in der Anfangsphase ziemlich überfordert, fanden keinen Ansatz oder brauchten viele zusätzliche Erklärungen, was unter anderem auch auf unser geringes Vorwissen über wirtschaftliche und physikalische Erdölaspekte zurückzuführen ist.

Doch bereits nach kurzem Einlesen - wobei uns Wikipedia sehr behilflich war – gelangten wir auf den richtigen Weg und unsere Begeisterung für mathematische Modellierung war nicht mehr zu hemmen.

Wir teilten uns vorerst in zwei Gruppen. Während die eine Gruppe sich den physikalischen Seiten, wie z.B. Erdölförderung widmete, beschäftigte sich die andere mit dem wirtschaftlichen Part, der u.a. Preise, Kapital und Angebot beinhaltet. Binnen kürzester Zeit stellten wir eine Reihe von Überlegungen, Formeln und Funktionen auf.

Das Schwierige war nun, ein Modell zu entwickeln, das die Arbeit beider Gruppen vereinte und unsere Ergebnisse auch graphisch darstellte, und wir experimentierten viel mit MatLab herum, um auf zufriedenstellende Ergebnisse zu kommen, weshalb wir auch manchmal bis 11 in der Nacht (oder noch länger) arbeiteten.

Jedoch – die Arbeit hat sich gelohnt, und wir blicken auf eine gelungene Woche zurück.

An dieser Stelle wollen wir ein herzliches Dankeschön an unseren „Master Steve“ aussprechen, der uns in jeder freien Minute zur Seite stand und selber schlaflose Nächte mit unserem Modell verbrachte.