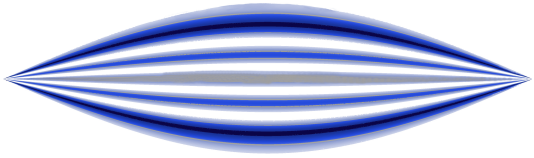
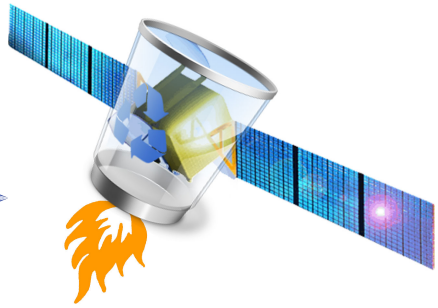
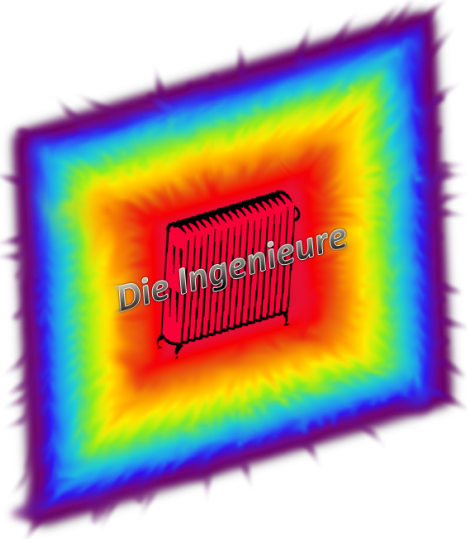
10. Woche der Modellierung mit Mathematik



Equazione delle onde





Dokumentationsbroschüre 8.2. – 14.2.2014

WOCHE DER MODELLIERUNG MIT MATHEMATIK

DEUTSCHLANDSBERG, 8.2.–14.2.2014

WEITERE INFORMATIONEN:

HTTP://MATH.UNI-GRAZ.AT/MODELLWOCHE/2014/

ORGANISATOREN UND SPONSOREN

Comfortplan
Online - Versicherungsmakler

KOORDINATION

Mag. Michaela Seiwald

Alexander Sekkas

Mag. Dr. Patrick-Michel Frühmann

Vorwort

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

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

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

Wir danken dem Landesschulrat für Steiermark, und hier insbesondere Frau Landesschulinspektorin Frau HR Mag. Marlies Liebscher, für die Hilfe bei der Organisation und ihre kontinuierliche Unterstützung der Idee einer

Modellierungswoche. Ohne den idealistischen, unentgeltlichen und engagierten Einsatz der direkten Projektbetreuer Univ.-Prof. Dipl.-Ing. Dr. Klemens Fellner, Dipl.-Math. Carl Philip Trautmann, Daniel Kraft, BSc BSc MSc und Mag. Dr. Martin Holler – Institut für Mathematik und Wissenschaftliches Rechnen – hätte diese Modellierungswoche nicht stattfinden können.

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

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

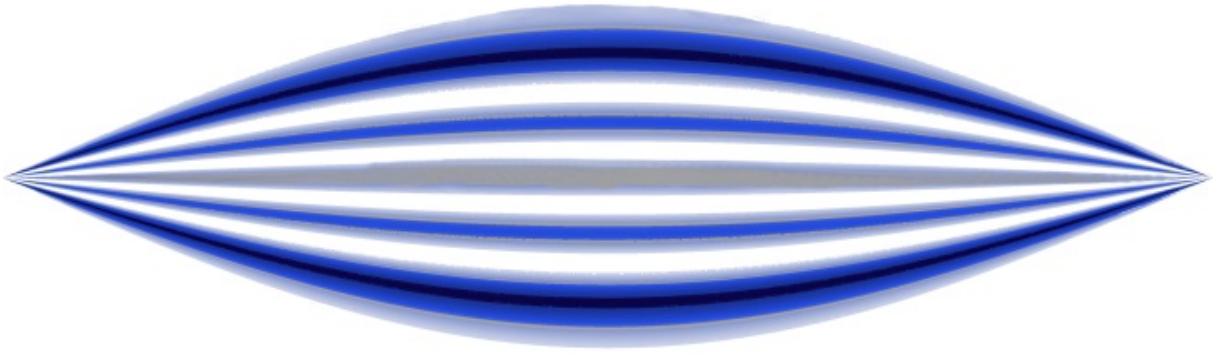
Deutschlandsberg, am 14. Februar 2014

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

AKUSTIK

Betreuer: Univ.-Prof. Dipl.-Ing. Dr. Klemens Fellner

Team: Doris Prach, Jasmin Pfleger, Jakob Prager, Daniel Ulrich Teichert, David Wörgötter, Valentin Zsilavecz



Equazione delle onde

8.2.-14.2.2014

Inhalt

Wellengleichung

Warum erzeugt eine Gitarrensaite einen Ton und kein Geräusch? Wovon hängt die Tonhöhe einer Saite ab? Was ist die Klangfarbe eines Tones und wie kann man sie beeinflussen? Warum erzeugt ein Blasinstrument einen Ton wie eine Saite? Wie groß sind die Unterschiede zwischen der Idealen Theorie und der Praxis?

Die Wellengleichung beschreibt mit der Formel $\partial_{tt}u = c^2 * \partial_{xx}u$⇔Beschleunigung = c^2 ∗Krümmung die Entwicklung einer Welle, ohne die Anfangsauslenkung bzw. die Anfangsgeschwindigkeit zu beachten, wobei c^2 die Wellengeschwindigkeit beschreibt, die sich aus Spannung/Dichte ergibt.

Die Formel enthält die Kräfte-Bilanz, also den Zusammenhang zwischen Zugkraft (Spannung) und Trägheitskraft und ist linear, da die vorausgesetzten Ableitungen lineare Operationen sind.

Die Elementarlösungen sind Sinus und Cosinus, man kann durch Überlagerung von Linearkombinationen (Superpositionsprinzip) beliebig viele Lösungen finden. Je mehr Lösungen man kombiniert, desto genauer kann man die Anfangsauslenkung darstellen.
Faktor $_1$* Lösung$_1$ + Faktor$_2$ * Lösung$_2$ = Lösung$_3$

Um eine Formel für die Amplitude zu finden, trennen und vereinfachen wir die zeit- und ortsabhängigen Komponenten mit Hilfe des Separationsansatzes:

- Eine Funktion $u(t,)$ kann als Produkt von zwei Funktionen $g(t) * f(x)$ dargestellt werden.
- bei Ableitungen: $\partial_{tt}u = \partial_{tt}g * f$ $\quad$ $\partial_{xx}u = g * \partial_{xx}f$

Wellengleichung: $\partial_{tt}g * f = c^2 * g * \partial_{xx}f$ $\quad$ $| * \frac{1}{f*g}$

$\Leftrightarrow \frac{\partial xxf}{f} = -\gamma^2$ $\Longrightarrow$ a) ortsabhängige Komponente

$\Leftrightarrow \frac{1}{c^2} * \frac{\partial_{tt}g}{g} = -\gamma^2$ $\Longrightarrow$ b) zeitabhängige Komponente

ad a)

Randbedingungen: Die Saite ist an beiden Enden eingespannt, darum muss die Funktion an den Stellen 0 und l (Länge der Saite) Null sein.

Die allgemeine Lösung ist $f(x) = A\sin(\gamma x) + B\cos(\gamma x)$, weil Sinus und Cosinus die Elementarlösungen der Gleichung sind.

Da $f(0) = 0$ sein muss, und $A\sin(\gamma x)$ sicher Null, aber $\cos(\gamma x) \neq 0$ ist, muss $B = 0$ sein.

Die Formel wird auf $f(x) = A\sin(\gamma x)$ gekürzt. Weil $f(l) = 0$ sein muss, muss entweder A oder $\sin(\gamma l)$ = 0 sein. Wenn A = 0 gibt es keine Schwingung, deswegen nehmen wir an dass $\sin(\gamma l) = 0$ ist.

Die Sinusschwingung hat den Durchmesser π, weshalb γl ein natürliches Vielfaches von π sein muss. $\quad \gamma l = \pi, 2\pi, 3\pi, \ldots => k\pi \quad (k \in \mathbb{N}) => \gamma_k = \frac{k\pi}{l}$

ad b)

Wir ersetzen γdurch $\frac{k\pi}{l}$: $\quad \frac{\partial_{tt} g}{g} = -c^2 \left(\frac{k\pi}{l}\right)^2 = -\left(\frac{ck\pi}{l}\right)^2$

Wir stellen die zwei Lösungen (Sinus und Cosinus) mit Hilfe einer Phasenverschiebung Φ_k dar, die vereinfachte Form der Zeitkomponente ist $A_k \cos\left(\frac{ck\pi}{l}(t-\Phi_k)\right)$ bzw. $A_k \cos\left(\frac{ck}{2l} 2\pi(t-\Phi_k)\right)$.

Wellenlänge: $\lambda_k = \frac{2l}{ck}$ => Frequenz: $\nu_k = \frac{1}{\lambda_k} = \frac{ck}{2l}$

Die Frequenz ist die Anzahl der Schwingungen pro Zeiteinheit.

Aus a) und b) ergibt sich nun die Formel:

$$u_k(t,x) = A_k \cos(\nu_k 2\pi(t-\Phi_k)) * \sin\left(\frac{k\pi}{l}x\right)$$

Für die Formel für die Amplitude brauchen wir zum Zeitpunkt Null

a) eine Anfangsgeschwindigkeit: $u_t(t=0,x) = 0$ und
b) eine Anfangsauslenkung: $u(t=0,x) = u_0(x)$.

} für Gitarre

Skalarprodukt (SKP): $\quad \langle \sin kx \,|\, \sin mx \rangle = 0 \quad \forall\, k \neq m; m, k \in \mathbb{N}$

Für alle u gilt:

$$u(t,x) = \sum_{k=1}^{\infty} A_k \cos(\nu_k 2\pi(t-\Phi_k)) \sin\left(\frac{k\pi}{l}x\right)$$

ad a)

$\partial_t u(t,x) = \partial_t [\sum_{k=1}^{\infty} u_k] = \sum_{k=1}^{\infty} \partial_t u_k$ gilt, weil die Funktion linear ist.

Wir bilden das skalare Produkt der Ableitung der Summen von u_k:

$$\left\langle \sum_{k=1}^{\infty} -A_k \sin\left(\nu_k 2\pi(t-\Phi_k)\right)\nu_k 2\pi \sin\left(\frac{k\pi}{l}x\right) \middle| \sin\left(\frac{m\pi}{l}x\right) \right\rangle = \left\langle 0 \middle| \sin\left(\frac{m\pi}{l}x\right) \right\rangle$$

Durch die Linearität kann man die ortsunabhängigen Komponenten aus dem SKP herausheben. Die rechte Seite der Gleichung ist Null, da das SKP mit Null immer Null ist.

Daraus ergibt sich folgende Gleichung:

$$\sum_{k=1}^{\infty} -A_k \sin\left(\nu_k 2\pi(t-\Phi_k)\right)\nu_k 2\pi \underbrace{\left\langle \sin\left(\frac{k\pi}{l}x\right) \middle| \sin\left(\frac{m\pi}{l}x\right) \right\rangle}_{=0 \quad \forall\, k \neq m} = 0$$

Wir betrachten nur noch den Fall bei dem $k = m$, bei dem das Skalarprodukt $\frac{l}{2}$ ist, also gibt es nur eine Möglichkeit und wir benötigen die Summe nicht mehr.

$$-A_k \sin\big(\nu_k 2\pi(t - \Phi_m)\big)\nu_k 2\pi \left(\frac{l}{2}\right) = 0$$

Daher sieht man, dass beim Zeitpunkt Null $-\Phi_m$auch Null sein muss, damit die Gleichung erfüllt ist.

Weil m alle natürlichen Zahlen abdeckt und $-\Phi_m$ damit immer Null ist, weiß man, dass Φ_k ebenfalls immer Null ist.

ad b)

$$u(t = 0, x) = \sum_{k=1}^{\infty} A_k \cos(\nu_k 2\pi * 0) * \sin\left(\frac{k\pi}{l} x\right)$$

Wie vorher bilden wir das SKP, heben die nicht ortsunabhängigen Komponenten heraus und betrachten den Fall $k = m$:

$$A_k \underbrace{\left\langle \sin\left(\frac{k\pi}{l} x\right) \middle| \sin\left(\frac{k\pi}{l} x\right) \right\rangle}_{\frac{l}{2}} = \left\langle u_0 \middle| \sin\left(\frac{k\pi}{l} x\right) \right\rangle \quad \Rightarrow \quad A_k = \frac{2}{l} \left\langle u_0 \middle| \sin\left(\frac{k\pi}{l} x\right) \right\rangle \quad \forall m \in \mathbb{N}$$

a) eine Anfangsgeschwindigkeit: u_t(t=0,x) = v_0 und
b) eine Anfangsauslenkung: u(t=0,x) = 0.

} für Klavier

Für alle v_0 gilt (da die Auslenkung = 0 können wir cos mit sin ersetzen):

$$u_t(t = 0, x) = \sum_{k=1}^{\infty} A_k \sin(\nu_k 2\pi(t - \Phi_k)) \sin\left(\frac{k\pi}{l} x\right)$$

Wir leiten in der Formel den Sinus ab und setzen t = 0:

$$v_0 = \sum_{k=1}^{\infty} A_k \cos(\nu_k 2\pi(0 - \Phi_k))\nu_k 2\pi \sin\left(\frac{k\pi}{l} x\right)$$

Wir bilden SKP, heben heraus und betrachten den Fall $k =$ m:

$$A_k \nu_k 2\pi \underbrace{\left\langle \sin\left(\frac{k\pi}{l} x\right) \middle| \sin\left(\frac{k\pi}{l} x\right) \right\rangle}_{\frac{l}{2}} = \left\langle u_0 \middle| \sin\left(\frac{k\pi}{l} x\right) \right\rangle \quad \Rightarrow \quad A_k = \frac{\frac{2}{l}\left\langle u_0 \middle| \sin\left(\frac{k\pi}{l} x\right) \right\rangle}{\nu_k 2\pi}$$

Skalarprodukt:

Ist eine positiv definite, symmetrische Bilinearform im Vektorraum V und wertet Vektoren in $\mathbb{R}$ aus.

1) Bilinearform $\langle \alpha\overrightarrow{v_1} + \beta\overrightarrow{v_2}|\vec{w}\rangle = \alpha\langle\overrightarrow{v_1}|\vec{w}\rangle + \beta\langle\overrightarrow{v_2}|\vec{w}\rangle \qquad \alpha, \beta \in \mathbb{R}$ -> linear in 1. Komponente

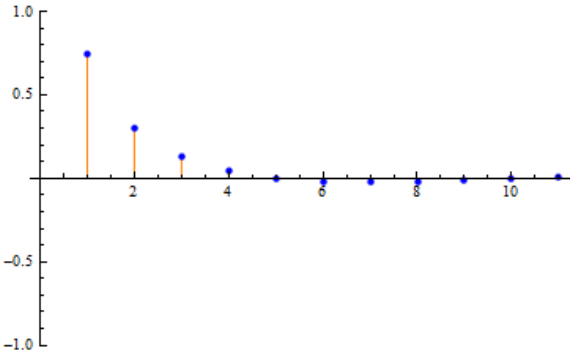
2) Symmetrie $\langle\vec{v}|\vec{w}\rangle = \langle\vec{w}|\vec{v}\rangle$

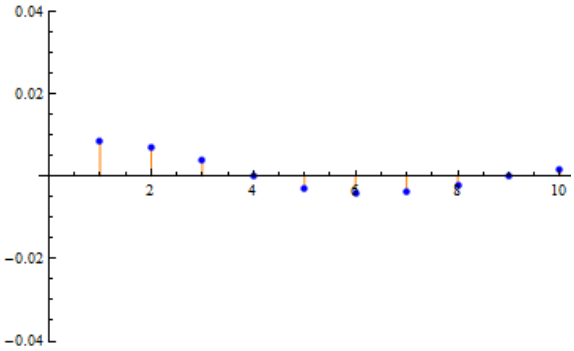
3) pos. def.: $\langle\vec{v}|\vec{v}\rangle = 0$, wenn $\vec{v} = \vec{0}$ und $\langle\vec{v}|\vec{v}\rangle > 0$, wenn $\vec{v} \neq \vec{0}$

Orthogonalität: $\langle\vec{v}|\vec{w}\rangle = 0 \Leftrightarrow \vec{v} \perp \vec{w}$

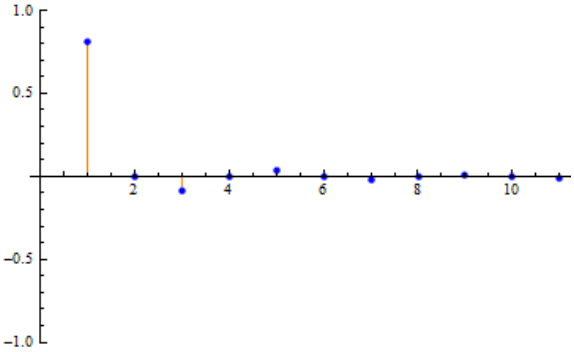
Approximation der Anfangsauslenkung

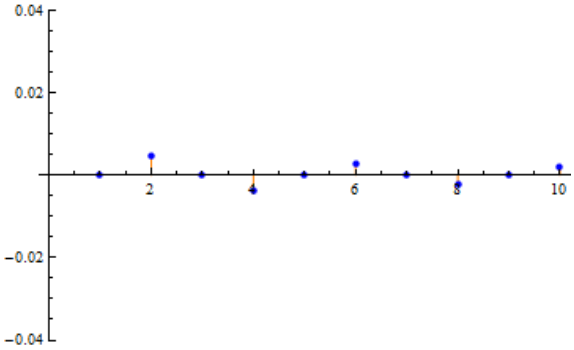
Zuerst werden mit Hilfe der oben errechneten Formel für die Amplitude A_k die Amplituden für beliebig viele Obertöne bestimmt:



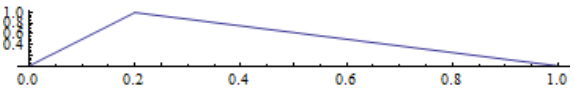


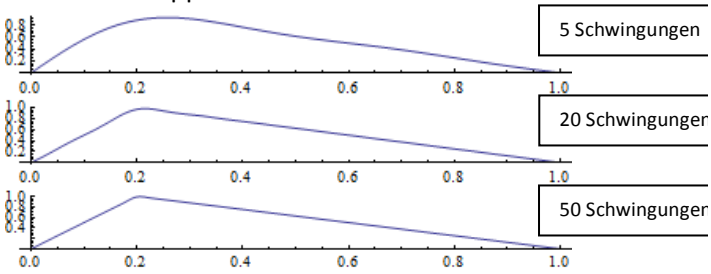
Dabei fällt auf, dass das Obertonspektrum, je nachdem wo man die Saite auslenkt, stark variiert. Die obigen Grafiken zeigen das Spektrum bei Auslenkung über dem Schallloch, während die unteren genau in der Mitte der Saite ausgelenkt wurden. Dadurch fällt jeder zweite Oberton weg und der Ton klingt eher hohl.

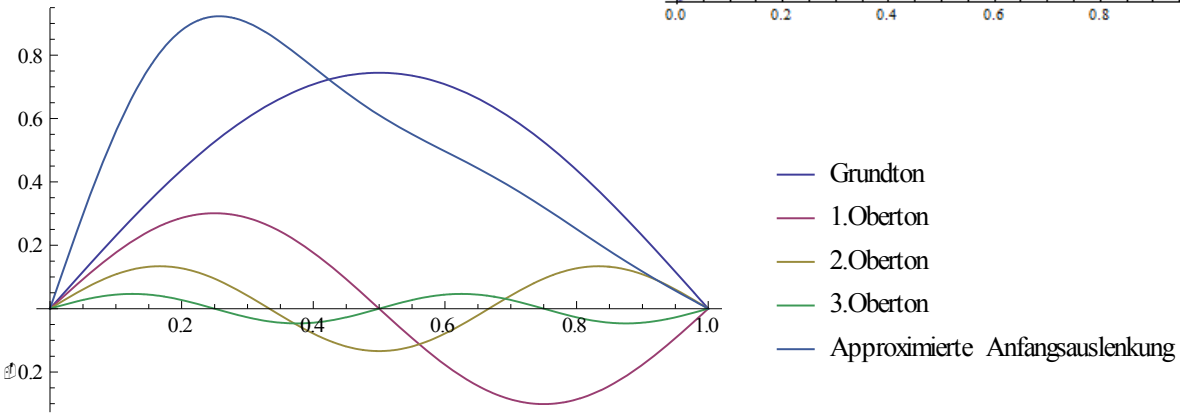




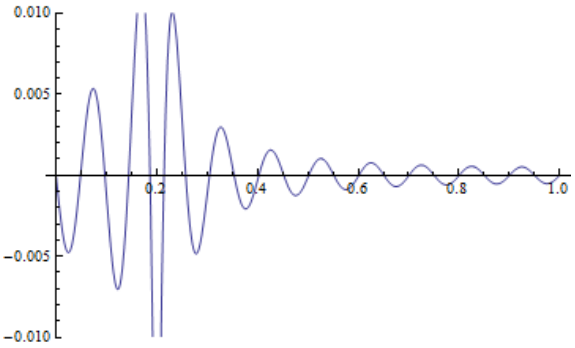
Addiert man die Funktionen der Obertöne, sollte man wieder zur Anfangsauslenkung zurück kommen. Je mehr Schwingungen man addiert, desto besser ist die Approximation:

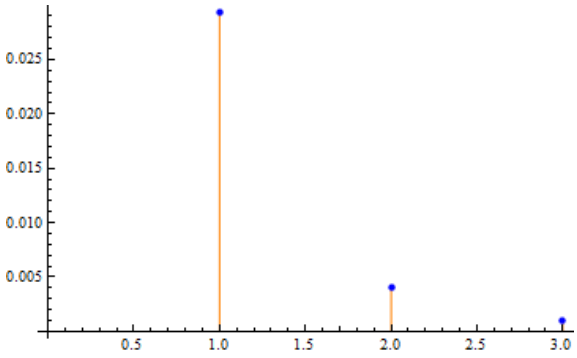






Wir haben den Fehler der Approximation auf zwei Arten bestimmt. Auf der linken Seite ist der absolute Fehler, also die Differenz zwischen u_0 und$\sum u_k$, wobei an der Stelle des größten Fehlers die Saite ausgelenkt wurde. Um diesen optischen Fehleindruck zu vermeiden kann man auch folgende Formel anwenden und bekommt so einen Fehler als Skalar für die ganze Funktion$\sqrt{\int_0^l (u_0 - \sum u_k)^2}$





Auswertungen der Messungsergebnisse

Alle folgenden Abbildungen sind wie folgt beschriftet:

x-Achse: Hertz

y-Achse: Dezibel

Die Achsen sind je nach Lautstärke unterschiedlich skaliert. Besonders zu beachten ist, dass sie logarithmisch gestuft sind. Dies bedeutet, dass der Amplitudenabfall in Wirklichkeit um einiges heftiger ablaufen würde.

Die Bildbezeichnung „ganz“ bedeutet, dass der gesamte Ton vom Anschlag bis zum Abklingen gemessen wurde. „Laut“ und „leise“ zeigt nur die entsprechenden Abschnitte.

Zusätzlich zu den Messungen wurden auch die Frequenzen analysiert. Diese sind den Abbildungen aber nicht ersichtlich.

Gitarre:

Bei diesen Messungen war es das Ziel, aus den Obertonstrukturen Schlüsse auf die Gründe für die unterschiedlichen Klangfarben zu ziehen, wenn man zum Beispiel am Schallloch oder am Steg den Ton auslöst.

- E
  Das Amplitudenverhältnis stimmt bei jeder Lautstärke, positionsunabhängig, kaum mit den Werten einer idealen Saite überein (Abbildung 1). Die Obertöne sind viel lauter als der Grundton. Der Grund liegt wahrscheinlich darin, dass die unteren Saiten der Gitarre gewickelte Metallsaiten sind, also inhomogen.
  In der Mitte ist die Amplitude jedes zweiten Obertons deutlich schwächer als am Steg und am Schallloch (Abbildung 2).

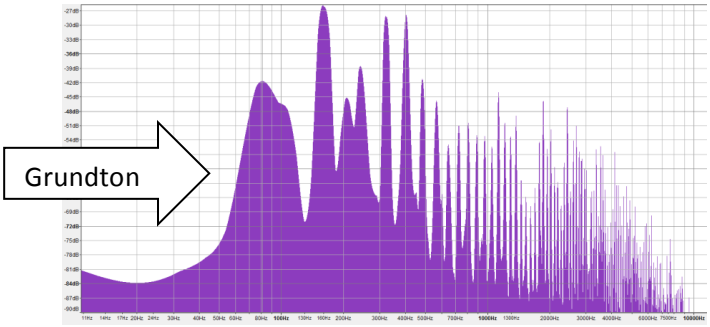


Abbildung 1 (E am Steg, ganz)

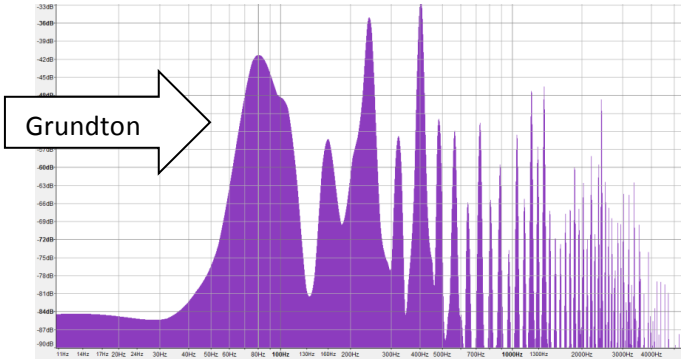


Abbildung 2 (E in der Mitte, ganz)

Die Frequenzen der Obertöne haben gegenüber den errechneten Werten kaum Unterschiede.

- g

  Hier stimmen die Messungen einigermaßen mit den errechneten Werten überein (Abbildung 3). Der Grundton ist generell am lautesten, nur während des Abklingens dominieren die Obertöne. (Abbildung 4) Am Steg sind die Obertöne lauter als an anderen Positionen. Die drei höheren Gitarrensaiten haben eher den Charakter einer idealen Saite, wahrscheinlich deswegen, weil sie aus Kunststoff und damit einem homogenen Material bestehen.
  Im Gegensatz zu den Frequenzen der tieferen Saiten sind jene der Obertöne der Kunststoffsaiten zu tief.

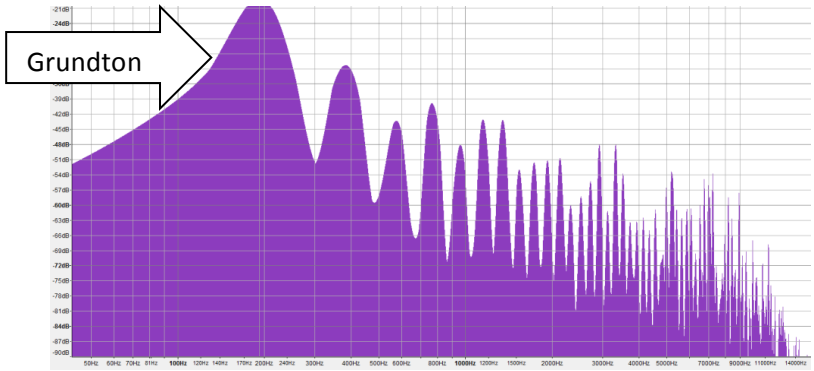


Abbildung 3 (g- Schallloch ganz)

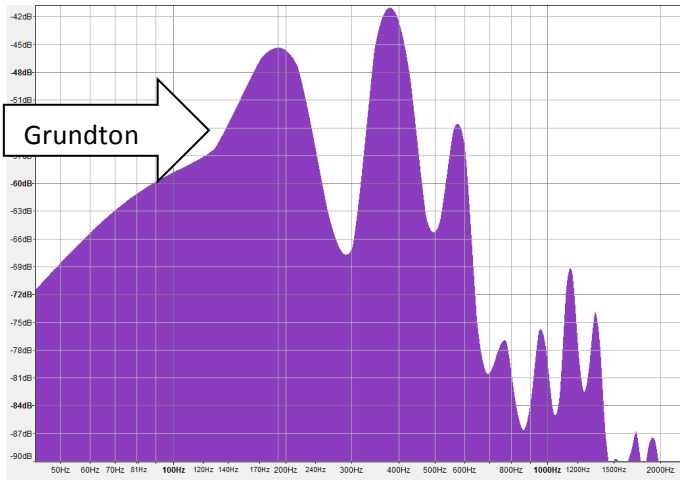


Abbildung 4 (g Schallloch leise)

Bei der Gitarre wurden die beiden Töne jeweils in der Mitte, am Schallloch und nahe dem Steg gemessen. Zusätzlich wurden der gesamte Ton, nur der laute Beginn sowie das leise Abklingen verglichen.
Der übliche, volle Gitarrenklang wurde nur über dem Schallloch erreicht, am Steg klingt der Klang etwas metallisch und in der Mitte nicht so imposant.

Klavier:

Bei diesen Messungen haben wir tiefe und hohe Töne am Klavier analysiert, verglichen und anhand der für ideale Saiten errechneten Werte ausgewertet.

- A_1

  Die ersten Obertöne sind deutlich lauter als der Grundton (Abbildung 5). Beim Abklingen ist der Grundton im Verhältnis lauter, aber trotzdem noch leiser als die Obertöne (Abbildung 6). Generell sind beim Nachklingen weniger klingende Frequenzen festzustellen.

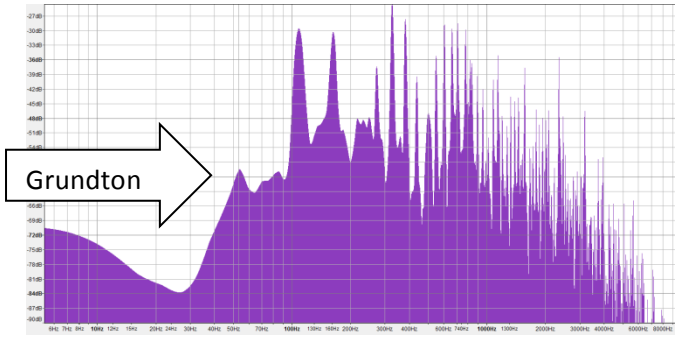


Abbildung 5 (A1 ganz)

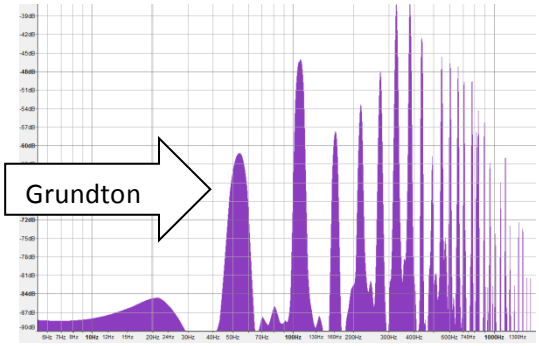


Abbildung 6 (A1 leise)

- A

  Der Grundton ist annähernd gleich laut wie die ersten Obertöne, beim Abklingen ist er wieder leiser. Generell wurden bei diesem Ton keine großen Unterschiede zwischen laut und leise gemessen. (Abbildung 7)

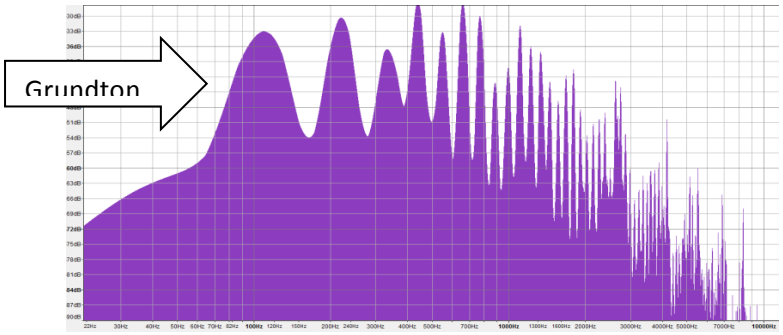


Abbildung 7 (A ganz)

Eigentlich sollte der Grundton immer die höchste Amplitude haben.
Eine mögliche Erklärung für das Schwingungsverhalten am Klavier von A_1 und A wäre, dass die dickeren Klavierseiten umwickelt, also inhomogen sind. Dadurch haben sie andere Eigenschaften als eine ideale Saite.
Außerdem wird die Frequenz mit jedem Oberton stetig höher als errechnet.

- a‘

  Der Grundton ist lauter als die Obertöne (nur beim Abklingen sind die ersten beiden Obertöne lauter) (Abbildungen 8 und 9). Auch hier nimmt mit der Lautstärke die Anzahl der Frequenzen ab.
  Das Amplitudenverhältnis nähert sich dem Errechneten für ideale Saiten an.

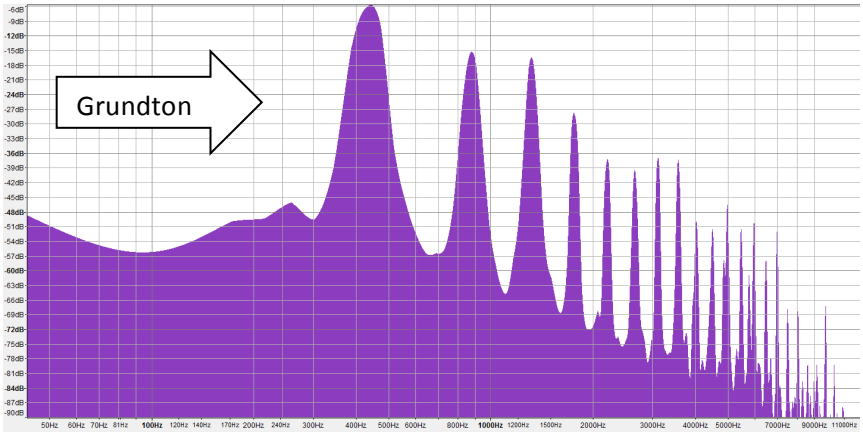


Abbildung 8 (a' laut)

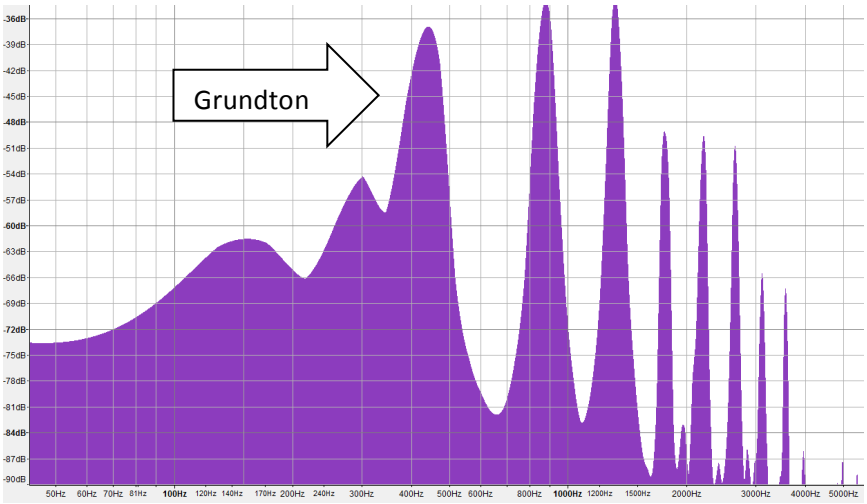


Abbildung 9 (a' leise)

- a‘‘

  Der Grundton ist deutlich höher als die Obertöne (wie errechnet), besonders beim Abklingen (Abbildung 10) dominiert nun der Grundton und es sind nur wenige Obertöne dabei. Die Messung des lauten Tonbeginns (Abbildung 11) zeigt eine schöne Abnahme der Amplitudengröße.
  Ab circa 2000 Hz wird die Frequenz der Obertöne viel zu hoch.

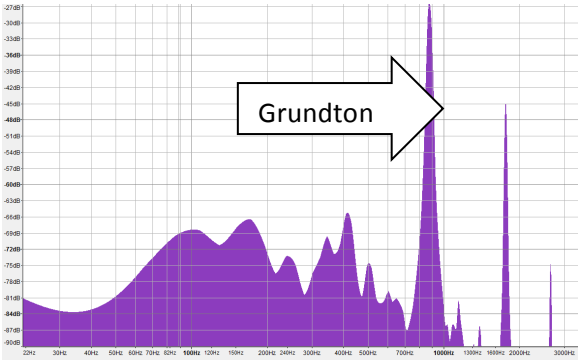


Abbildung 10 (a'' leise)

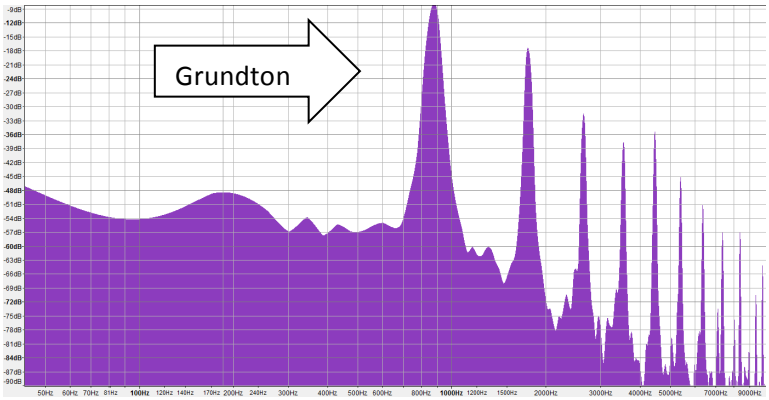


Abbildung 11 (a'' laut)

Die höheren Klaviersaiten entsprechen besser den idealen Saiten, da sie nicht mehr umwickelt sind. Trotzdem kommt es zu Abweichungen, unter anderem auch deswegen, weil ja immer drei Saiten nebeneinander angeschlagen werden.

Allgemein ist festzustellen, dass sowohl das Klavier, als auch die Gitarre und ebenso alle anderen Saiteninstrumente durch verschiedenste Faktoren beeinflusst werden. Ein entscheidender Faktor ist beispielsweise das Material der Saiten. Auch der Resonanzkörper spielt eine gewichtige Rolle: Er ist dafür verantwortlich, dass, wie oben erwähnt, im Abklingen die Obertöne mehr zur Geltung kommen. Aber eben diese Tatsache, dass jedes Instrument auf seine eigene Art und Weise unterschiedliche Obertöne produziert, sorgt für die verschiedenen Klangfarben der Saiteninstrumente.

Zu all diesen Messungen ist noch zu sagen, dass die Software und das Mikrofon keine hochwissenschaftlichen Geräte waren, weshalb man die Ergebnisse nicht überbewerten sollte.

Anwendung der Wellengleichung auf druckgesteuerte Blasinstrumente

Die erzeugten Schallwellen werden an der Öffnung des Instrumentes teilweise reflektiert. Die reflektierten Schallwellen wandern zurück zum Mundstück, wo bestimmte Frequenzen verstärkt werden. Diese wandern wieder zur Öffnung, werden teilweise reflektiert, etc. Daraus entsteht innerhalb von 50 ms eine stehende Welle im Instrument.

Impedanz

ist der lineare Zusammenhang zwischen Druck und Geschwindigkeit → Kräfte-Bilanz

Für die Berechnung wichtige Parameter:

- Frequenz
- Wellenzahl, daraus ergibt sich die Kreisfrequenz
- Schallgeschwindigkeit
- Referenzdichte
- Abschnittslänge
- Zylinderradius
- Länge des Zylinders

Die Ausgangsimpedanz an der Öffnung des Instrumentes wird mithilfe mehrerer zusammenhängender Formeln berechnet, die an dieser Stelle nicht erwähnt werden, da sie 1. zu kompliziert sind und 2. nicht zum Verständnis beitragen.

Anschließend wird eine andere Formel verwendet, um die Impedanz stückweise (z.B. in 5mm-Scheiben) hochzurechnen. Zum Schluss erhält man die Eingangsimpedanz, also den linearen Zusammenhang zwischen Druck und Geschwindigkeit am oberen Ende des Instrumentes. Bei dieser Formel werden die Tonlöcher sowie das Mundstück nicht berücksichtigt. Die Impedanz wird im Endeffekt für ein Rohr berechnet. Mithilfe von Matlab haben wir die Impedanz jeweils für den reibungsfreien Zylinder und den Zylinder unter Berücksichtigung der Viskosität modelliert.

Reibungsfreier Zylinder

Abbildung 1 zeigt eine Impedanzkurve für 100-1000 Hertz bei einem 65cm-langen Zylinder, wenn man die Reibung nicht berücksichtigt.

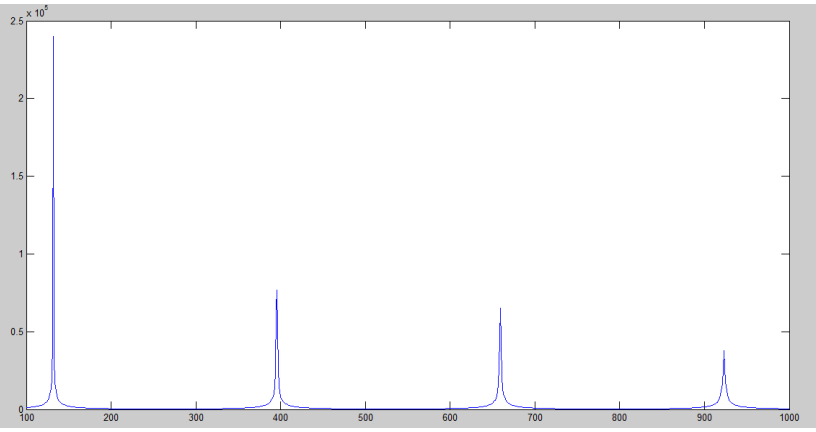


Abbildung 12: Impedanzkurve eines reibungsfreien Zylinders

Zylinder

Abbildung 2 zeigt die deutlich unterschiedlichere Impedanzkurve von 100-1000 Hertz bei einem 65cm-langen Zylinder, unter Berücksichtigung einer gewissen Viskosität der Luft.

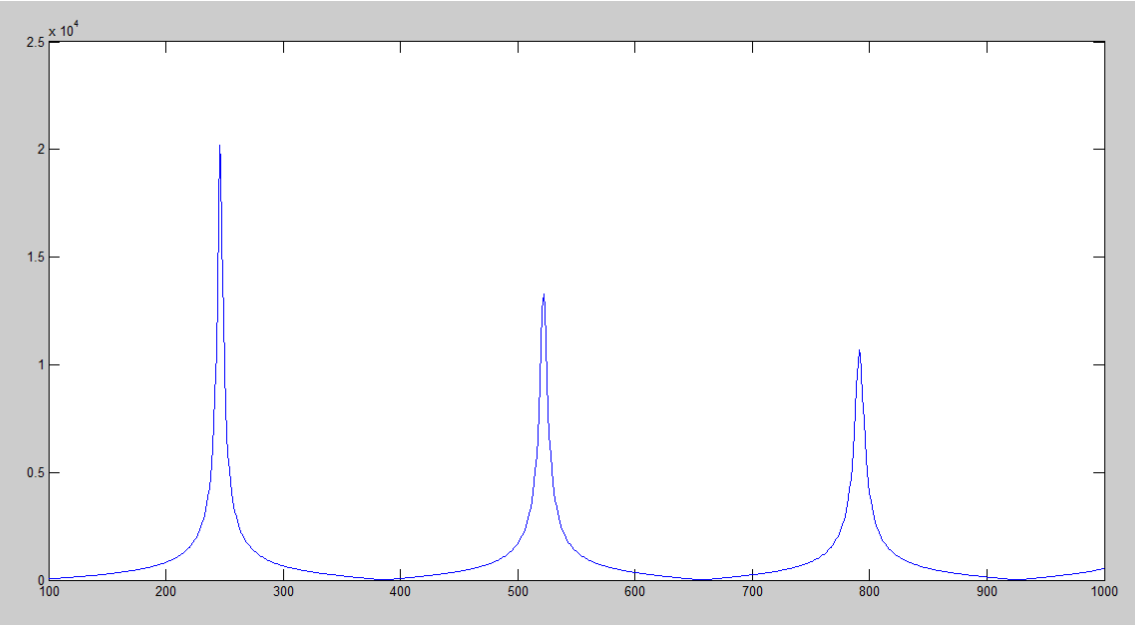


Abbildung 13: Impedanzkurve unter Berücksichtigung der Viskosität der Luft

Vergleich

In Abbildung 3 werden die Impedanzkurve des reibungsfreien Zylinders (blau) mit der Kurve des Zylinders unter Berücksichtigung der Viskosität (grün) verglichen. Man sieht, dass die Kurven sich stark unterscheiden: Die Amplituden des reibungsfreien Zylinders könnten theoretisch unendlich groß sein, da sie nicht von Reibung beeinflusst werden.

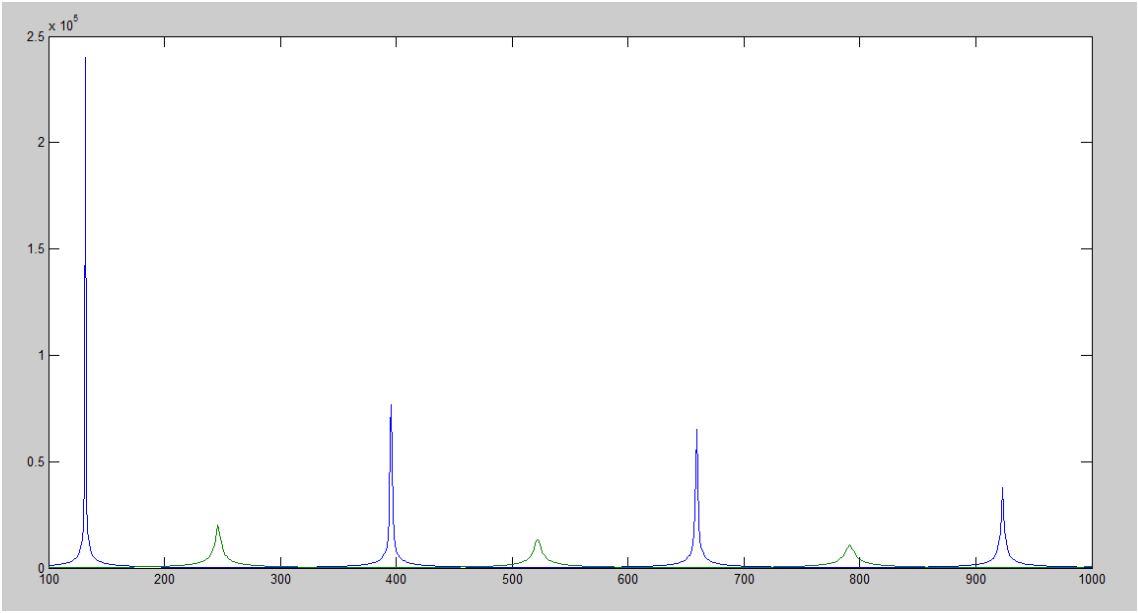


Abbildung 14: Vergleich reibungsfrei - unter Berücksichtigung der Viskosität

Antworten:

Die Gitarrensaite erzeugt einen Ton, weil nur ganzzahlige Vielfache der Grundschwingung verstärkt werden. Bis jetzt ist noch unklar, warum unser Gehirn dieses Zusammenspiel aus Grund- und Oberschwingungen als Ton wahrnimmt.

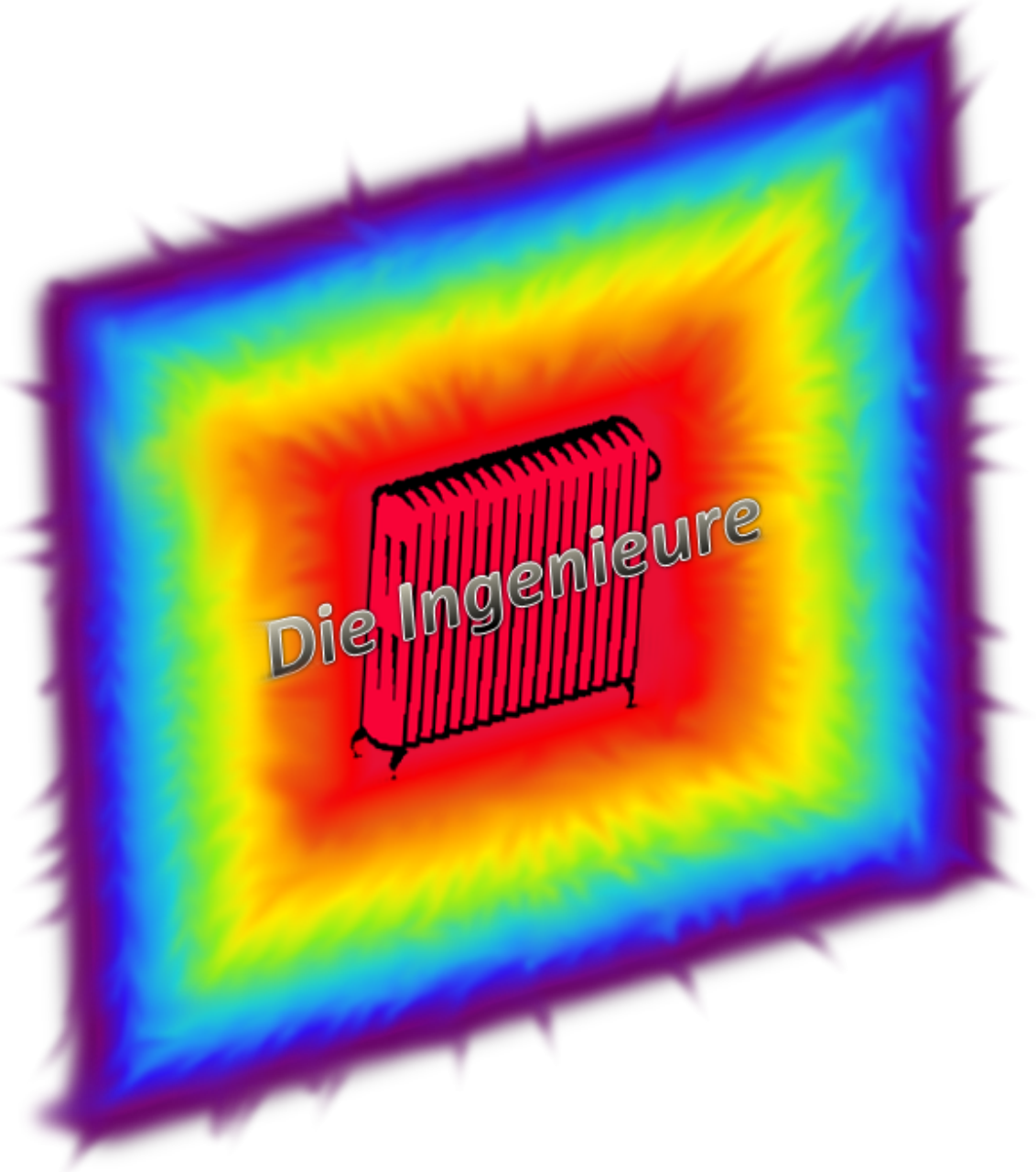
Die Tonhöhe einer Saite hängt von ihrer Dichte und Spannung ab.

Das Obertonspektrum wird als Klangfarbe wahrgenommen, bei einer Gitarre kann sie durch das Spielen an einer anderen Stelle der Saite beeinflusst werden. Je mehr Obertöne es gibt, desto voller ist der Klang.

Ein Blasinstrument erzeugt im Inneren eine stehende Welle, die dann, wie bei der Gitarre, als wandernde Welle an die Luft abgegeben wird.

Die Unterschiede zwischen der idealisierten Theorie und der Praxis sind sehr groß, da bei der Herleitung der Wellengleichung immer wieder Werte angenähert und Faktoren wie Reibung vernachlässigt werden.

Ingenieurtechnik



Betreuer: Daniel Kraft, BSc BSc MSc

8.2.-14.2.2014

Steuerung einer Heizung

Valentin Haberl, Benedikt Andritsch, Simon Danglmaier,

Thomas Birbacher, Stefan Krickl, Christoph Kremser

14. Februar 2014

Inhaltsverzeichnis

1 Aufgabenstellung

Die optimale Steuerung einer Heizung zählt sicher zu jenen Problemen, die für praktisch jeden Menschen von Interesse sind. Die zentrale Fragestellung bei diesem Projekt besteht darin, zu ergründen, wie man erreichen kann, dass trotz des Vorhandenseins von nur einem einzigen Thermostat in allen Räumen eine optimale Temperatur herrscht. Da ein Haus aus mehreren Räumen besteht, die sich durch ihre Größe, die unterschiedlichen Wärmeflüsse und die verwendeten Dämmmaterialien, sowie zahlreiche andere Faktoren unterscheiden, ist es notwendig, dass sich die Leistungen der Heizkörper den Umständen anpassen, um die Zieltemperatur zu erreichen.
Unser Projekt umfasste zusammengefasst folgende Ziele:

- Die Berechnung der Wärmeflüsse, die zwischen den Räumen und der Außenwelt bestehen und zwischen den Räumen untereinander, wobei besonderes Augenmerk auf die Wände gelegt wurde.
- Das Pendeln der Heizung um die Zieltemperatur zu verhindern oder zumindest möglichst einzuschränken.
- Die Bedingungen zu identifizieren, bei denen eine angenehme Temperatur in allen Zimmern herrscht.
- Sowie zu zeigen, wie sich verschiedene Dämmmodelle auf den Energiebedarf auswirken und wie groß die Kostenersparnis durch die Reduktion des Energieverbrauchs ist, die bei einer guten Dämmung im Vergleich zu einer schlechten auftritt.

2 Grundüberlegungen

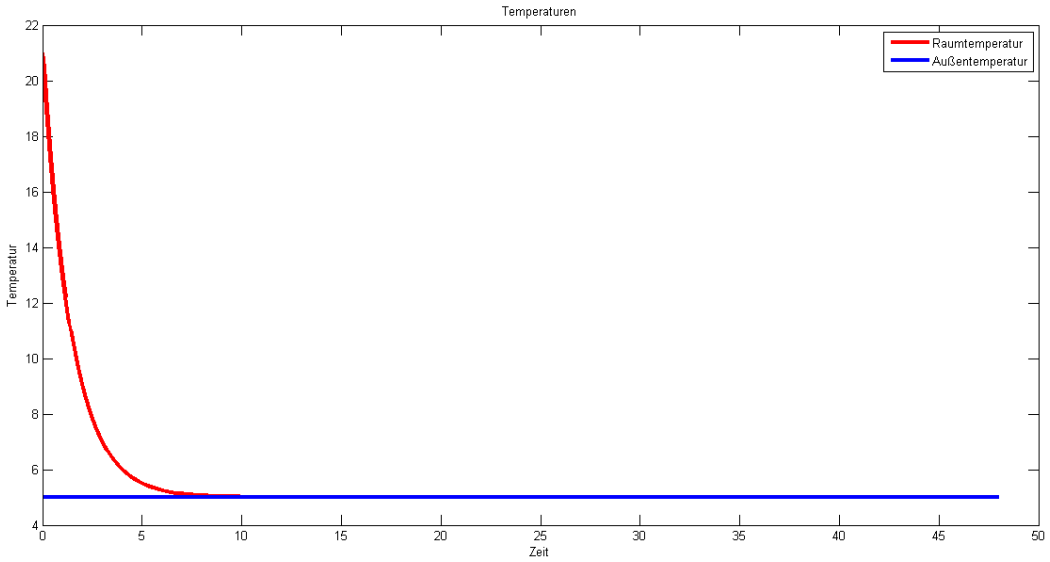
Die erste Überlegung bestand darin, dass wir den Wärmefluss eines einzigen Raumes in Bezug auf die Außentemperatur modellierten. Die Temperaturänderung $\dot{T}$ war hierbei umso größer, je stärker der Temperaturunterschied zwischen Raum und Außenwelt war.

$$\dot{T} = \frac{dT}{dt} = -a \cdot (T - T_a)$$

T repräsentiert hierbei die momentane Innentemperatur des Raumes, T_a die Außentemperatur, welche zu Beginn als fester Wert angenommen wurde, und a eine Konstante, die sich aus verschiedenen Faktoren zusammensetzt, welche im nächsten Kapitel näher erläutert werden. Diese gewöhnliche Differentialgleichung konnte noch sehr leicht durch Trennung der Variablen gelöst werden, sodass sich nach der fertigen Integration und Festlegung einer beliebigen Starttemperatur $T(0) = T_0$ folgende Funktion für T ergab:

$$T(t) = (T_0 - T_a)e^{-at} + T_a$$

Die Variable T_0 steht hierbei für die Starttemperatur im Innenraum. Diese Exponentialfunktion stellt eine natürliche Abnahme dar, was ein intuitiv richtiges Ergebnis ist. Die Annäherung an die Außentemperatur erfolgt zuerst schnell (aufgrund des hohen Temperaturunterschiedes) und stagniert schließlich.


Temperaturen
Raumtemperatur
Außentemperatur
Temperatur
Zeit
22
20
18
16
14
12
10
8
6
4
0
5
10
15
20
25
30
35
40
45
50

3 Der Wärmeflusskoeffizient a

Um in unserem bisherigen Modell zu realistischen Werten zu gelangen, war es nötig den von uns verwendeten Faktor a genau zu bestimmen. Hierfür zogen wir unsere Informationen aus dem Fourierschen Gesetz:

$$\dot{Q} = \frac{\lambda}{d} \cdot A \cdot (T_{W1} - T_{W2})$$

$\dot{Q}$... Änderung der Wärme

λ... Wärmeleitkoeffizient

d... Dicke der Wand

A... Fläche der Wand

T_{W1}... Temperatur der wärmeren Wandoberfläche

T_{W2}... Temperatur der kälteren Wandoberfläche

Um von dieser Formel auf a schließen zu können, mussten wir noch den Zusammenhang zwischen der Wärme Q und der Temperatur T nutzen:

$$\Delta Q = \Delta T \cdot c \cdot m \tag{3.1}$$

ΔT... Änderung der Temperatur

c... spezifische Wärmekapazität der Luft im Raum

m... Masse der Luft im Raum

Durch einfaches Umformen der Gleichungen erhält man a:

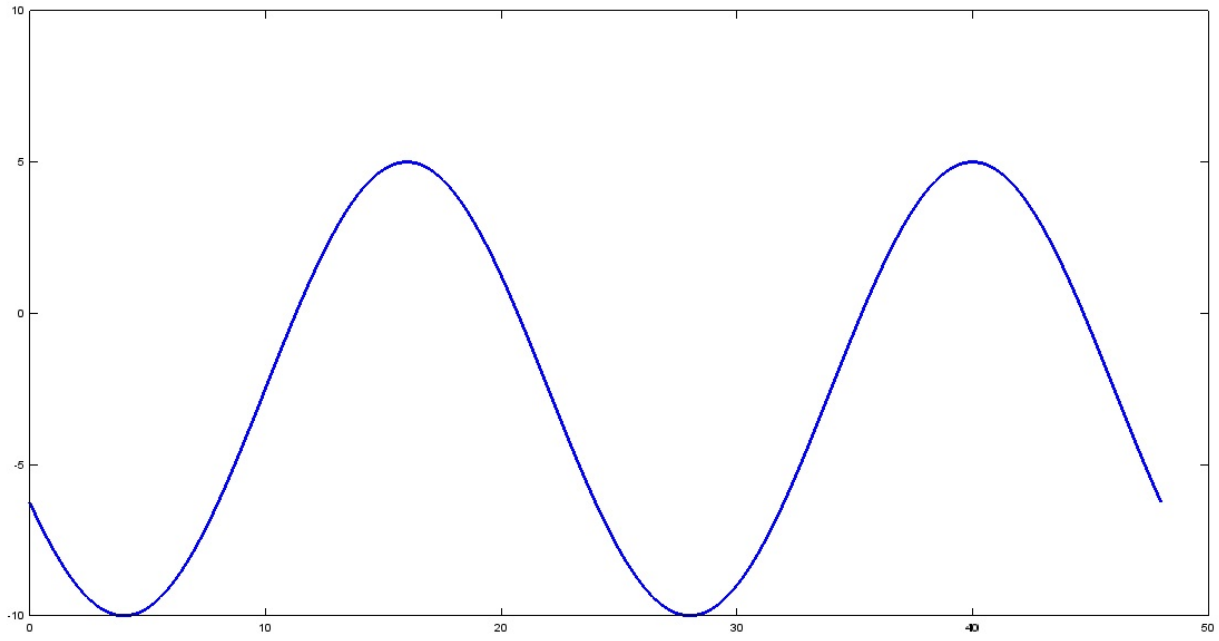
$$a = \frac{A \cdot \lambda}{d \cdot m \cdot c}$$

Die Differentialgleichung erhält daraufhin folgende Form:

$$\dot{T}(t) = -\frac{A \cdot \lambda}{d \cdot m \cdot c} \cdot (T(t) - T_a)$$

4 Die Außentemperatur T_a

Der nächste Schritt bestand darin, dass die Idee einer konstanten Außentemperatur verworfen und durch eine sich ändernde ersetzt wurde. Um dies zu bewerkstelligen wurde T_a als Sinuskurve modelliert. Diese sieht folgendermaßen aus:



$$T_a = \frac{T_{\max} - T_{\min}}{2} \cdot sin\left[\frac{\pi}{6} \cdot \left(\frac{t}{2} - 5 + \frac{h_{\text{Start}}}{2}\right)\right] + \frac{T_{\max} + T_{\min}}{2}$$

$T_{\max}$... tägliche Höchsttemperatur (um ca. 16 Uhr)

$T_{\min}$... tägliche Tiefsttemperatur (um ca. 4 Uhr)

h_{Start}... Uhrzeit, ab der die Simulation beginnt

Anhand von $T_{\max}$ und $T_{\min}$ können verschiedene Temperaturbereiche betrachtet werden, wodurch man den Temperaturverlauf auch in einer anderen Jahreszeit betrachten kann. Durch das Abhängigmachen der Außentemperatur T_a von der Zeit t wurden wir nun mit dem Problem konfrontiert, dass das Verfahren des Trennens der Variablen hier nicht mehr funktioniert. Daher wendeten wir ein numerisches Verfahren an.

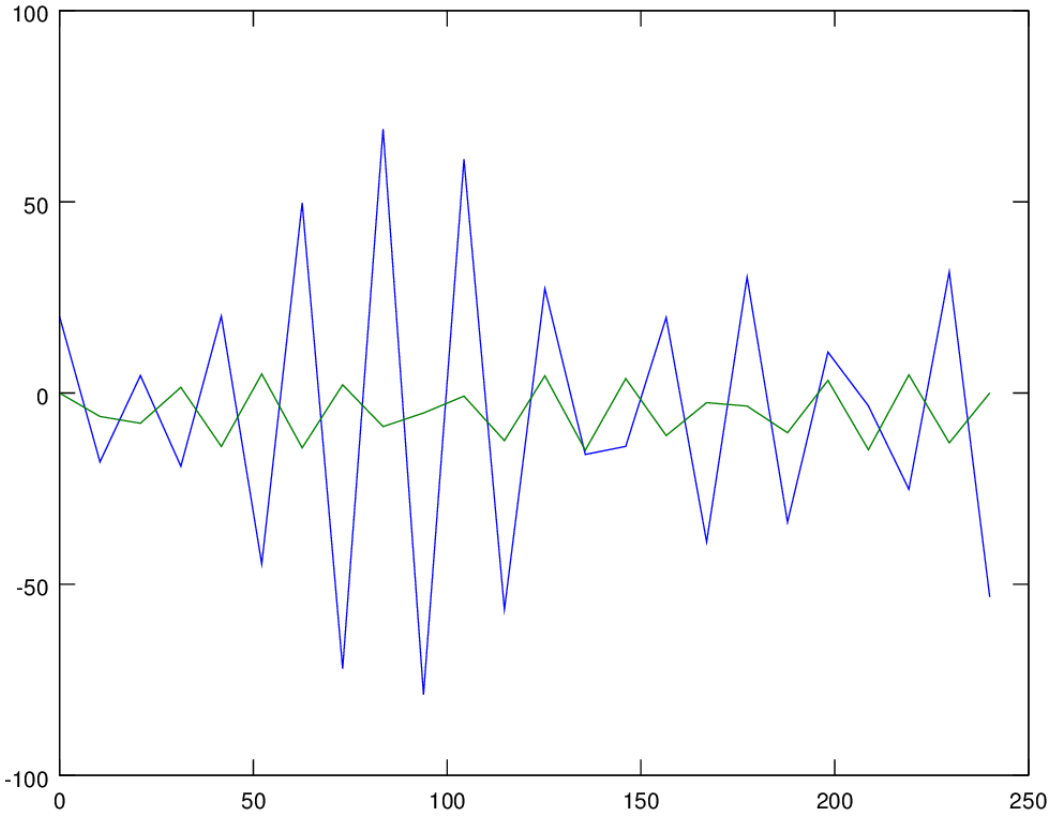
5 Numerische Lösung der Differentialgleichung

5.1 Explizites Euler-verfahren

Das explizite Euler-Verfahren zählt zu den einfachsten numerischen Verfahren zur Lösung von Differentialgleichungen, gilt als nicht besonders stabil und läuft folgendermaßen ab:

$$T_{t+\Delta} = T_t + \Delta t \cdot \dot{T}_t$$

Man legt einen Startwert T_t für $T(t)$ fest, errechnet dazu die erste Ableitung und addiert das Produkt aus dieser und der Zeitschrittweite Δt mit besagtem Startwert. So entsteht der approximierte Wert für $T_{t+\Delta t}$, der zum neuen Startwert T_t wird, mit dem wiederum $T_{t+\Delta t}$ approximiert wird. Durch stetiges Wiederholen dieses Verfahrens kann die Kurve $T(t)$ näherungsweise bestimmt werden. Allerdings hat das explizite Euler-Verfahren den Nachteil, dass bei größeren Werten für die Schrittweite Δt die Temperaturwerte in unrealistischer Weise explodieren.



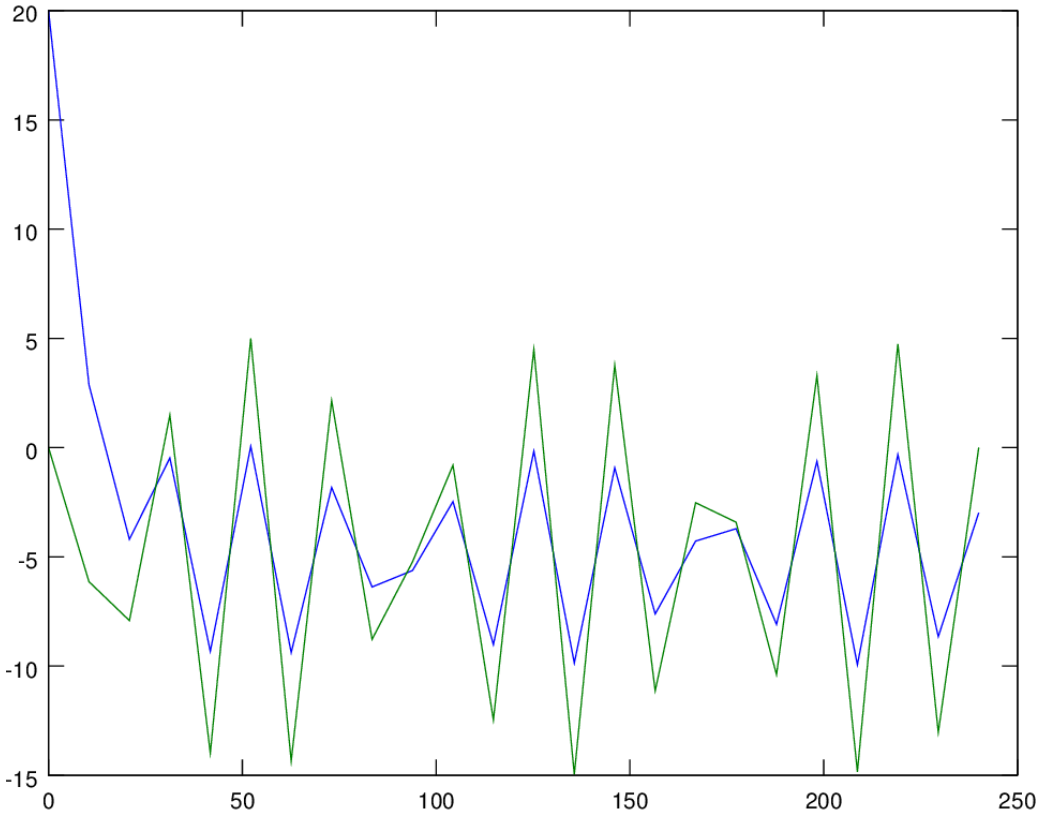
5.2 Implizites Euler-Verfahren

Das implizite Euler-Verfahren ist ein weiteres numerisches Verfahren, dass im Vergleich zum expliziten Euler-Verfahren deutlich stabiler ist und nach folgendem Schema funktioniert:

$$T_{t+\Delta t} = T_t + \Delta t \cdot \dot{T}_{t+\Delta t}$$

Der Unterschied zum expliziten Euler-Verfahren besteht darin, dass hier nicht die erste Ableitung des vorhandenen Punktes verwendet wird, sondern jene des zu errechnenden. Deswegen muss die Gleichung allerdings nach $T_{t+\Delta t}$ aufgelöst werden, da $T(t)$ auch in der Formel für $\dot{T}(t)$ auftaucht. Somit werden die Punkte von $T(t)$ auf folgende Weise approximiert:

$$T_{t+\Delta t} = \frac{T_t + \Delta t \cdot a \cdot T_{a_t}}{1 + a \cdot \Delta t}$$



5.3 Crank-Nicolson-Verfahren

Das Crank-Nicolson-Verfahren stellt eine Mischform der beiden vorherigen Verfahren dar. Hier wird das arithmetische Mittel der ersten Ableitung im vorhandenen Punkt T_t und der ersten Ableitung im Punkt $T_{t+\Delta t}$ gebildet, mit Δt multipliziert und das entstandene Produkt zu T_t hinzu addiert.

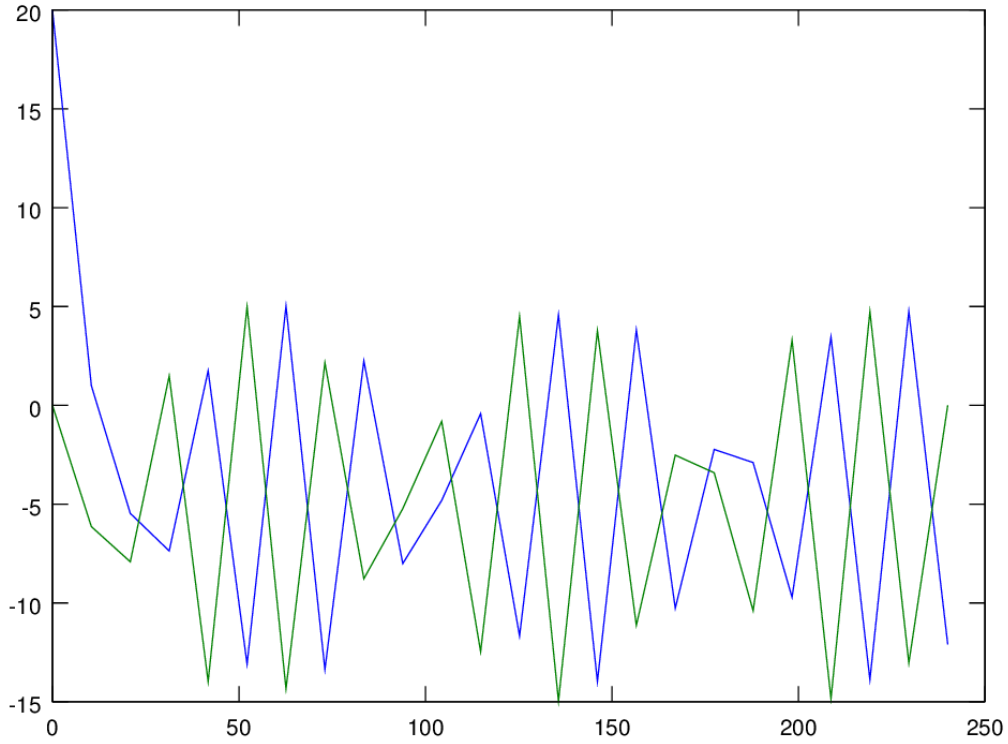
$$T_{t+\Delta t} = T_t + \Delta t \cdot \frac{\dot{T}_t + \dot{T}_{t+\Delta t}}{2}$$

Wenn man dieses Verfahren auf unsere Differentialgleichung anwendet, muss man die Gleichung natürlich wie beim impliziten Verfahren nach $T_{t+\Delta t}$ auflösen:

$$T_{t+\Delta t} = \frac{2 \cdot T_t - \Delta t \cdot a \cdot (T_t - T_{a_t}) + T_{a_{t+\Delta t}} \cdot \Delta t \cdot a}{2 + \Delta t \cdot a}$$

Bei diesem Verfahren bleiben die Werte auch bei größeren Werten für Δt stabil, allerdings sind sie deutlich unrealistischer als beim impliziten Euler-Verfahren, da dort die Werte eher

der Außentemperatur angeglichen werden und keine Temperaturen entstehen, die wie hier physikalisch nicht stimmen können.



5.4 Finales Verfahren

Das implizite Verfahren schnitt von allen drei numerischen bei Weitem am besten ab. Es blieb stets stabil und nahm realistische Werte auch für größere Δt an. Da wir im Folgenden auch die Wirkungen der Wärmeleitung zwischen mehreren Räumen betrachtet haben, entschieden wir uns für das explizite Euler-Verfahren. Denn bei Verwendung des impliziten Verfahrens würden wir in der Berechnung jedes neuen Schritts ein Gleichungssystem lösen müssen. Außerdem sind die Ergebnisse bei genügend kleinem Δt praktisch identisch bei allen drei Verfahren.

6 Wärmeflüsse zwischen mehreren Räumen

In allen bisherigen Simulationen wurde nicht berücksichtigt, dass ein Haus aus mehr als nur einem großen Raum besteht. Für eine realistische Berechnung ist es allerdings notwendig, auch Innenwände miteinzubeziehen. Deren Dämmung ist in der Regel schlechter, da sowohl die Mauern dünner als auch das verwendete Material durchlässiger ist. Der Temperaturausgleich im Haus findet schneller statt als jener zwischen Haus und Außenwelt. Problematisch ist dabei die Tatsache, dass die Formel verallgemeinert auf eine beliebige Anzahl Räume komplexer wird. Um die Berechnung der Temperaturänderung mit mehreren Räumen (und somit Wärmeflüssen) zu erleichtern, wurde eine Matrix erstellt, welche die Wärmeflusskoeffizienten von jedem Raum zu jedem Raum enthält. Wenn die Anzahl der Räume n beträgt, so wird der Fluss nach außen durch einen zusätzlichen "virtuellen Raum"mit der Nummer n+1 simuliert.

$$a = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & a_{1,n+1} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & a_{2,n+1} \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} & a_{n,n+1} \end{pmatrix}$$

Der Eintrag $a_(i, j)$ repräsentiert dabei den Wärmeflusskoeffizienten vom Raum i zum Raum j, und die Dimension der Matrix ist $n \times n + 1$. Befüllt wird sie entweder manuell oder durch einen halbautomatischen Rechner, der im nächsten Kapitel genauer erläutert wird. Um die Temperaturen aller Räume zu jedem Zeitpunkt speichern zu können, wird eine weitere Matrix erstellt.

$$T = \begin{pmatrix} T_{1,1} & T_{1,2} & \cdots & T_{1,t} \\ T_{2,1} & T_{2,2} & \cdots & T_{2,t} \\ \vdots & \vdots & & \vdots \\ T_{n,1} & T_{n,2} & \cdots & T_{n,t} \end{pmatrix}$$

Die Variable t gibt hier die Anzahl der Zeitschritte während der numerischen Berechnung an, n die Anzahl der Räume. Die Werte zum Zeitpunkt 1 befinden sich in der ersten Spalte und sind zu Beginn als Starttemperaturen bekannt. Um deren zeitliche Nachfolger zu berechnen verwendet man die bereits bekannt Formel

$$\dot{T}(t) = -a(T(t) - T_a(t))$$

und modifiziert sie indem man über alle Temperaturflüsse summiert , um eine Version zu erhalten, mit der es möglich ist das Verhalten von beliebig vielen Räumen zu simulieren. Eine zur numerischen Berechnung geeignet umgeformte Version lautet

$$T_{i,t+\Delta t} = \Delta t \sum_{r=1}^{n+1} [-a_{i,r}(T_{i,t} - T_{r,t})] + T_{i,t}$$

Durch diese Lösung können Temperaturflüsse zwischen beliebig vielen Räumen berechnet werden. Bei gleichmäßig verteilter Innentemperatur scheinen die Temperaturflüsse zwar vernachlässigbar, allerdings lassen sich somit zum Beispiel die Auswirkungen eines geöffneten Fensters oder generell unterschiedliche Starttemperaturen simulieren.

7 Genaue Berechnung des Wärmeflusskoeffizienten a

Um (vor alle im Zuge der Verallgemeinerung auf beliebig viele Räume im letzten Kapitel) die Werte für den Wärmeflusskoeffizienten a genau berechnen zu können, wurde ein Skript erstellt, welches deren Berechnung nach Eingabe einiger Parameter automatisiert. Außerdem werden Elemente wie zum Beispiel Fenster und Türen mit einbezogen, um möglichst genaue und wirklichkeitsgetreue Resultate zu erhalten.

$$a_{\text{gesamt}} = a_{\text{Wände}} + a_{\text{Fenster}} + a_{\text{Türen}}$$

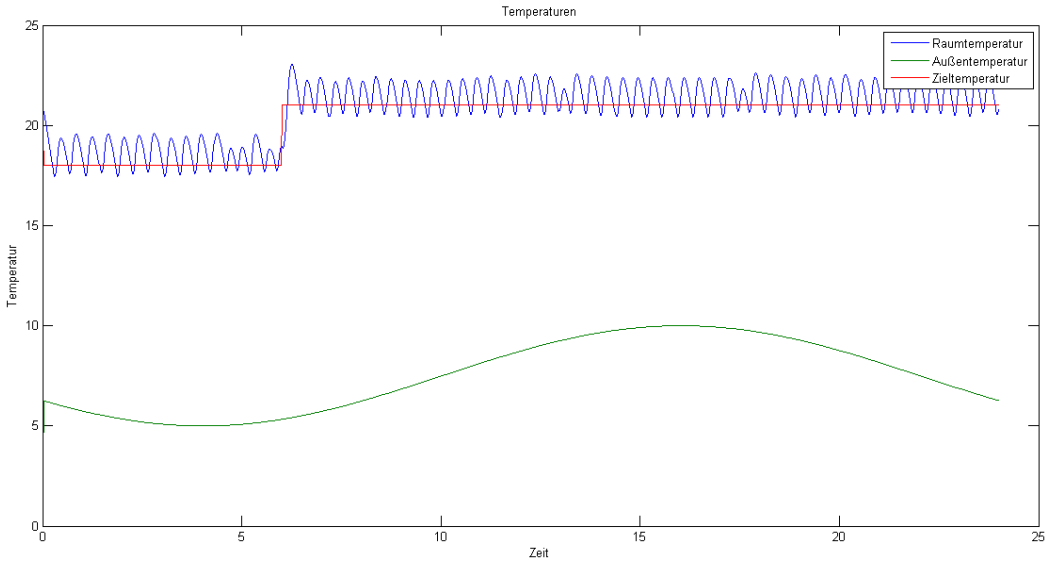
Hierbei wird für jeden Summanden die Formel

$$a = \frac{A \cdot U}{m \cdot c}$$

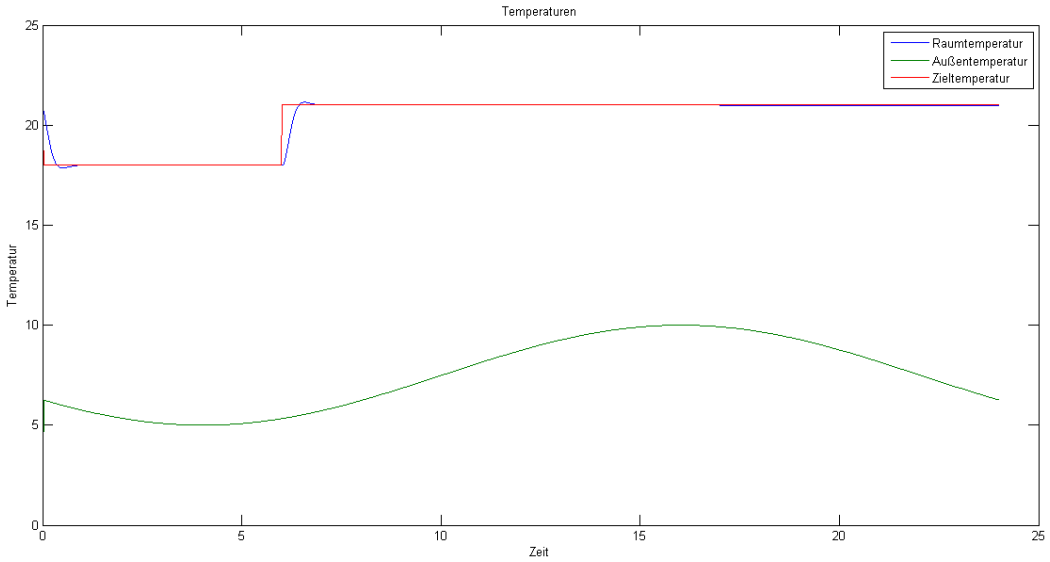
verwendet, wobei U für den Wärmedurchgangskoeffizienten steht, welcher in unserem Fall λ/d ersetzt. Der sogenannte „U-Wert“ wurde deshalb gewählt, weil er bereits den Wärmeleitkoeffizienten λ in Relation zur Dicke des jeweiligen Bauteils setzt. Da moderne Wände aus vielen, in ihren thermischen Eigenschaften unterschiedlichen Schichten bestehen, bietet die Verwendung des U-Werts eine große Gelegenheit zur Vereinfachung und verbessert gleichzeitig die Genauigkeit. Außerdem ziehen es Hersteller aus eben diesem Grund vor, nur diesen Wert anzugeben, um potentiellen Kunden oder Prüfern die Abschätzung der Dämmung zu erleichtern. Im finalen Skript wird aus Gründen der Einfachheit eine Konfigurationsdatei gestartet, welche zuvor mit Hilfe des Rechners erstellt wurde und die fertigen Wärmeflusskoeffizienten enthält. Somit können verschiedene Haus-Konfigurationen gespeichert und später aufgerufen werden.

8 Heizung für einen Raum

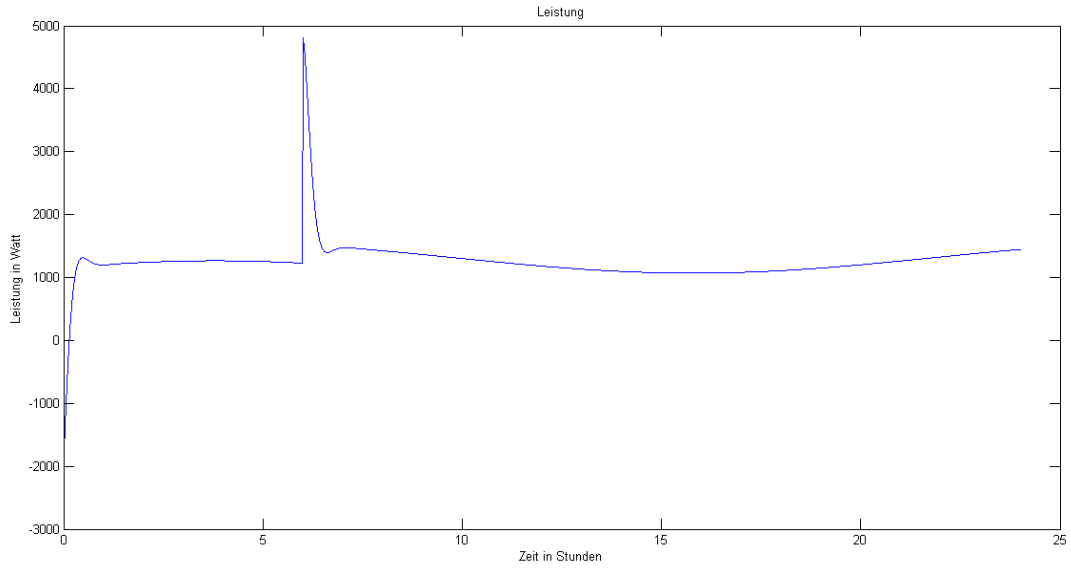
Um im Haus eine konstante, angenehme Innentemperatur zu ermöglichen, wurde ein System erstellt, welches Wärme zuführt und im Haus verteilt. Zu Beginn wurde eine Heizung für einen Raum programmiert, ohne dabei die Außentemperatur mit einzubeziehen. Man ging von einer angestrebten Raumtemperatur von 21 °C am Tag und 18 °C in der Nacht (von null bis sechs Uhr) aus. Die Berechnung der Heiztemperatur erfolgte anfangs durch eine Steuerung, welche die Heizung entweder mit 100% oder 0% Leistung (in zwei Stufen) aktiviert, woraus ein unerwünschtes pendeln der realen Temperatur um die Zieltemperatur entsteht.



Der Grund dafür ist, dass das System mit voller Leistung bis zum Erreichen des Zielwerts heizt, allerdings bei dessen Überschreiten sofort in den Ruhezustand zurück verfällt. Die Innentemperatur sinkt und steigt nun periodisch, ohne stabil zu sein. Korrigiert wird dies durch eine stufenlos einstellbare Heizung, deren Leistung davon abhängt, wie groß der Unterschied zwischen realer und angestrebter Temperatur ist.



Aus dieser Heiztemperatur ließ sich durch Multiplizieren mit der Wärmekapazität der Luft die Leistung der Heizung berechnen. Dabei entstanden (bezogen auf ein großes Zimmer) realistische Werte von circa einem Kilowatt. Die Spitze in der Grafik ist auf den kurzen Aufheizprozess von 18 auf 21 °C zurückzuführen.



Der nächste Schritt war das Hinzufügen einer zuerst konstanten,später durch eine Sinusschwingung angenäherte Außentemperatur. Zu dem Term, der die aktuelle Temperatur des Raumes beschrieb, kam jetzt ein Verlust an die Umgebung hinzu. Das führte dazu, dass die Raumtemperatur immer etwas unter der Zieltemperatur lag. In Folge dessen mussten wir eine neue Temperatur einführen, welche vom Steuerprogramm zwar angestrebt, von der Raumtemperatur allerdings nie erreicht wurde. Diese lag somit unter Einbezug des Verlusts nach außen genau bei der gewollten Temperatur. Das Verhältnis zwischen Raumtemperatur-

Außentemperatur und angestrebter Temperatur-Raumtemperatur war gleich wie jenes zwischen dem Proportionalitätsfaktor und dem Wärmefluss nach außen. Daraus ergab sich folgende Formel:

$$T_g(t) = T + \frac{a_R(T - T_a(t))}{K_p}$$

wobei T_g die neue angestrebte Temperatur, T die Zieltemperatur, T_a die Außentemperatur, K_p der Proportionalitätsfaktor und a_R der Wärmefluss nach außen ist. Um das Modell des Heizsystems realistischer zu gestalten, fügten wir zum bestehenden System noch Heizkörper hinzu. Diese beinhalteten eine gewisse Trägheit, mit der sie aufheizten bzw. abkühlten. Dazu ergänzten wir einen Trägheitsfaktor a_H. Außerdem beeinflusste im Gegensatz zum früheren Heizsystem nicht nur die Heizung die Raumtemperatur, sondern auch die Raumtemperatur die Heizkörpertemperatur. Dadurch geschah dasselbe wie durch das Hinzufügen einer Außentemperatur. Die Raumtemperatur wanderte leicht nach unten und erreichte das Ziel nicht mehr. Wir mussten daher unsere Formel für die angestrebte Temperatur erweitern. Zuerst fügten wir eine Heizkörpertemperatur T_h hinzu und mit dieser errechneten wir wiederum durch Nullsetzung der Differenzialgleichung eine Formel für die neue angestrebte Temperatur T_g.

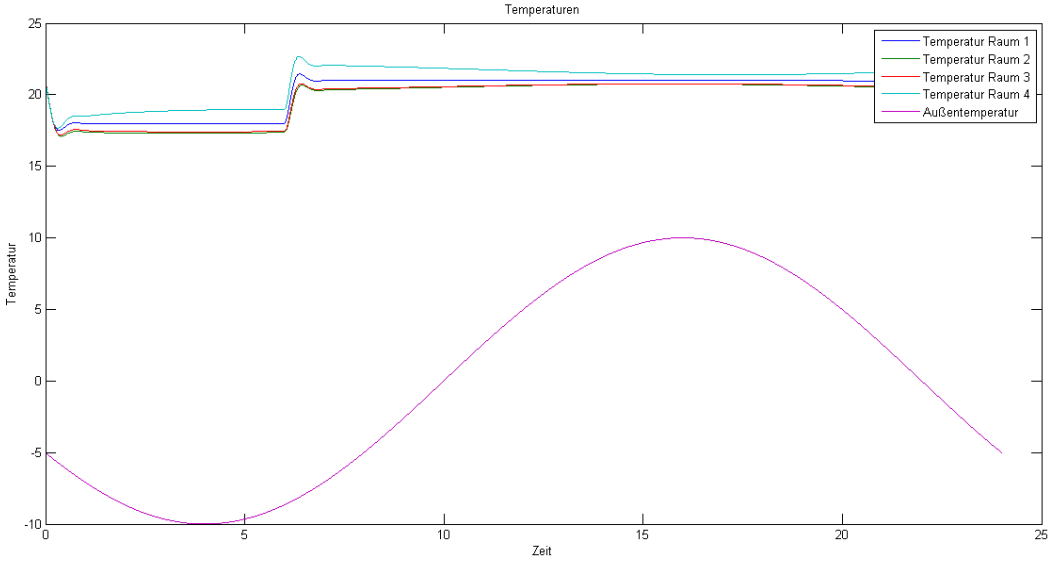
$$T_h(t) = \frac{T(t) \cdot (a_R + b_R) - a_R \cdot T_a(i)}{b_R}$$

$$T_g(t) = T(t) + \frac{b_H \cdot \frac{T_h(t) - T(t)}{a_H} + T_h(t)}{K_p}$$

wobei b_R den Wärmefluss vom Heizkörper zum Raum beschreibt und b_H jenen in die umgekehrte Richtung. Daraufhin implementierten wir unsere Heizungssteuerung mit Heizkörpern in das mehrräumige Matrixsystem. Zu beachten war, dass es nur eine zentrale Steuerung in einem Raum gab. Diese war, wie bereits erwähnt, abhängig vom Wärmefluss nach außen, welcher jedoch bei jedem Raum anders war. So müsste in jedem Raum unterschiedlich stark geheizt werden, aus Gründen der Realitätsnähe kann es nur eine zentrale Steuerung geben. Nun musste für jeden Raum ein vom Wärmefluss abhängiger Faktor gefunden werden, mit dem die Heiztemperatur des ersten Raumes multipliziert wird.

9 Heizung für mehrere Räume

Durch das installierte System, welches eine gleiche Leistung für jeden Raum bereitstellte, wurden bei Räumen mit unterschiedlicher Größe einige Bereiche zu stark beziehungsweise zu schwach erhitzt.



Um dieses Problem zu lösen wurde das jeweilige Output mit einem Faktor zwischen null und eins multipliziert. Der Faktor für den Raum i wurde als a_i bezeichnet, dessen Herleitung folgendermaßen erfolgte:

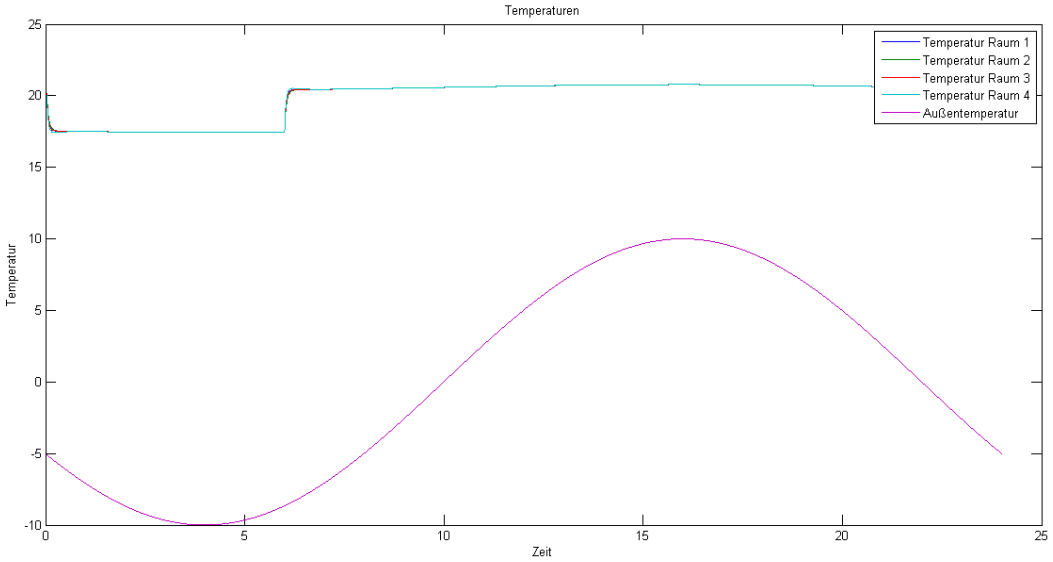
$$\dot{T}_i = \sum_{r=1}^{n+1} [-a_{i,r}(T_i - T_r)] + \alpha_i \cdot q_1$$

In der Praxis kann man die Temperaturflüsse zwischen zwei perfekt beheizten Räumen vernachlässigen, da alle Räume auf gleiche Zieltemperatur gebracht werden sollten und der Fluss in dieser Situation gegen null geht. Des Weiteren wurde vorausgesetzt, dass $\dot{T}_i$ (die Temperaturänderung) ebenfalls null ist, da die Temperatur stabil gehalten werden sollte.

$$\alpha_i = \frac{a_{i,a}(T_i - T_a)}{q_1} = \frac{a_{i,a}(T_i - T_a)}{a_{1,a}(T_1 - T_a)}$$

Unter Einbeziehung dieses Wertes konnte mithilfe von nur einem Thermostat im Raum

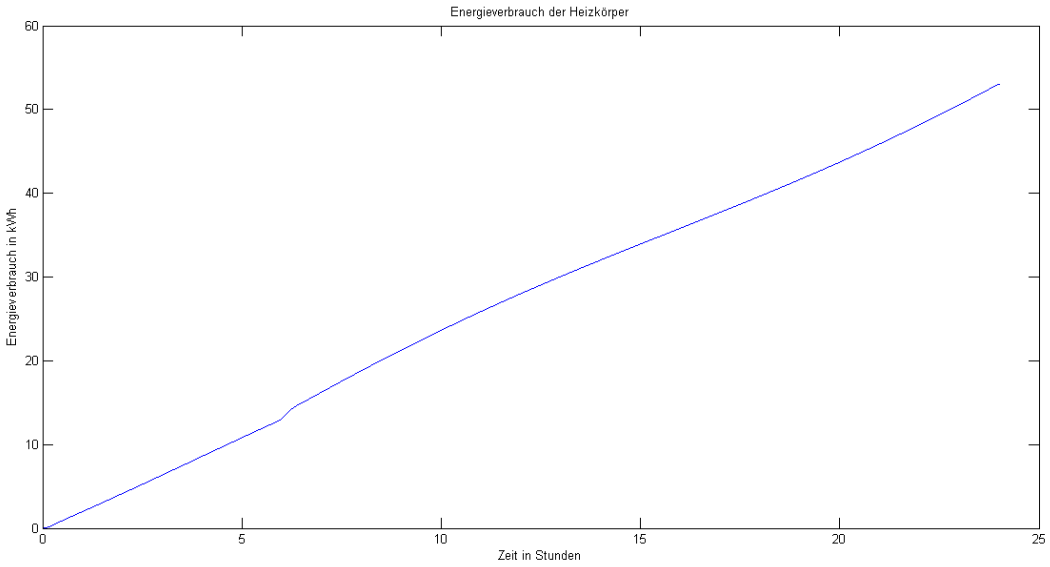
eins im ganzen Haus eine vorgegebene, stabile Temperatur realisiert werden. Im Plot liegen die Werte so nahe beieinander, dass sie nicht unterscheidbar sind.



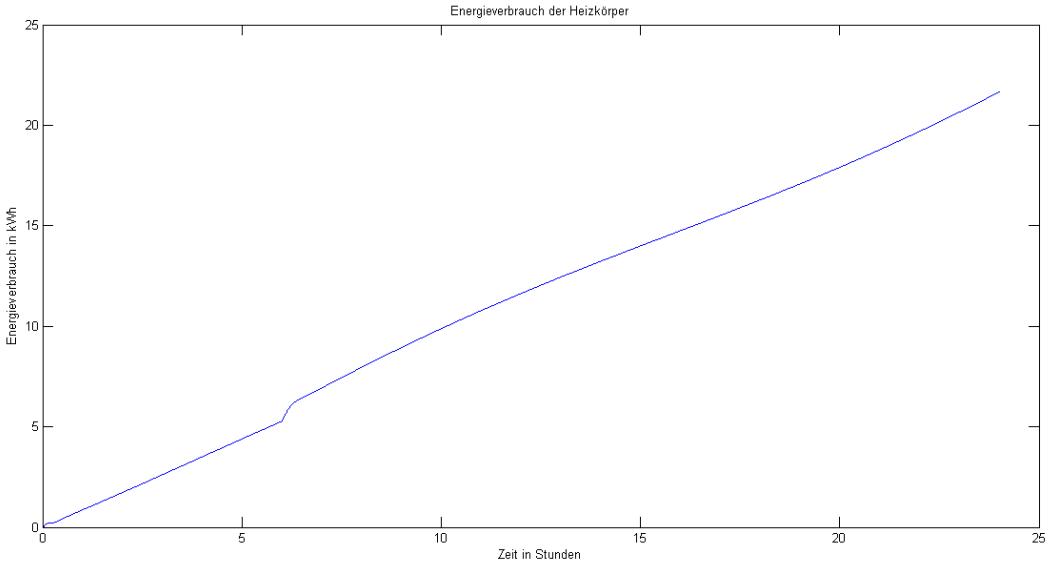
Auch der Wechsel zwischen 18 und 21 °C in der Früh beziehungsweise umgekehrt in der Nacht stellt keine Probleme dar, da die Zentralheizung unabhängig vom Alpha-Wert die Leistung für alle Räume erhöht

10 Energieverbrauch

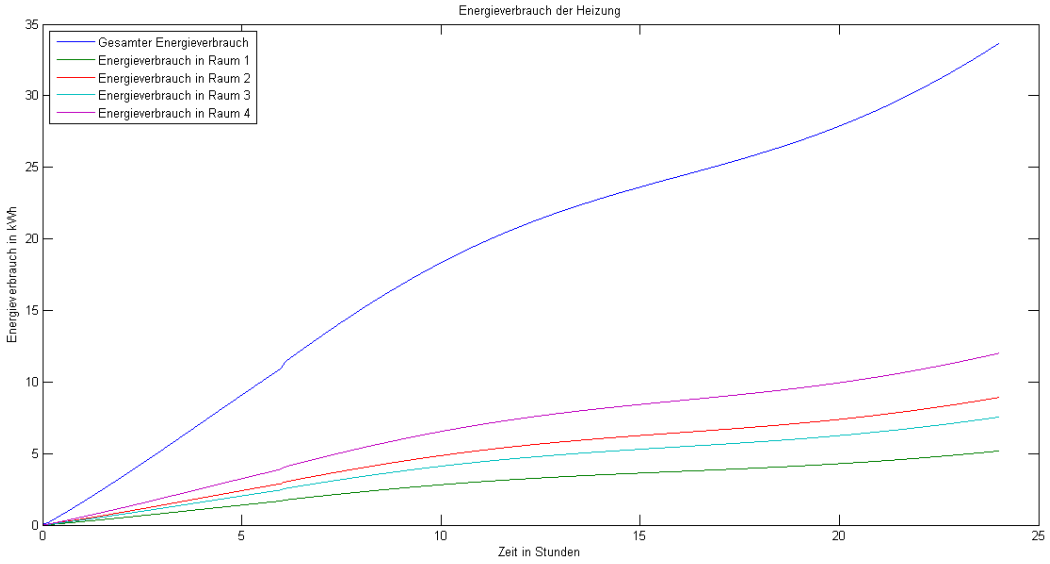
Um den Energieverbrauch des Gesamtsystems zu ermitteln, wurden alle Einträge des Leistungsvektors aufsummiert und mit dem Zeitschritt Δt multipliziert. Die Wahl des Dämmmaterials (und dadurch dessen Modifikation des Wärmeflusskoeffizienten) beeinflusst dabei maßgeblich den Energieverbrauch. In einer Beispielwohnung mit circa 120 Quadratmetern lag der Energieverbrauch bei schlechter Dämmung (vergleichbar mit einem Altbau) bei über 50 Kilowattstunden pro Tag, das entspricht 150 Kilowattstunden pro Jahr und Quadratmeter.



Das gleiche Haus erreichte im gut isolierten Zustand einen Verbrauch von nur 20 Kilowattstunden, weniger als die Hälfte des ursprünglichen Wertes.



Außerdem lässt sich erkennen, wie viel ein einzelner Raum zum Gesamtbedarf an Energie beiträgt. Abhängig ist dies nicht nur von der Größe des Raumes, sondern auch von der Fläche der Außenwand und deren Dämmung.




MISHUMO
J
&
J

Optimale Steuerung

Betreuer: Dipl.-Math. Carl PhilipTrautmann

Gruppe: Erwin Fink, Andreas Jan Geierhofer, Katharina Lachner
Markus Liebenwein, Hannah Lichtenegger, Peter Weiss

8.2.-14.2.2014

Optimale Steuerung eines Chaser-Satelliten

Inhalt

Einleitung

In den letzten Jahrzehnten wurde der Weltraum von den Menschen immer mehr erforscht, dadurch entwickelte sich jedoch auch eine beachtliche Menge an Weltraumschrott. Dieser Weltraumschrott stellt eine Bedrohung für Satelliten und die ISS dar und sollte deswegen entfernt werden. Dafür könnte ein spezieller Chaser-Satellit in die Erdumlaufbahn geschickt werden, der sich dem Müll annähert, diesen fasst und mit ihm zurück zur Erde fliegt, um in der Atmosphäre zu verglühen. Die durch Triebwerke gesteuerte Annäherung

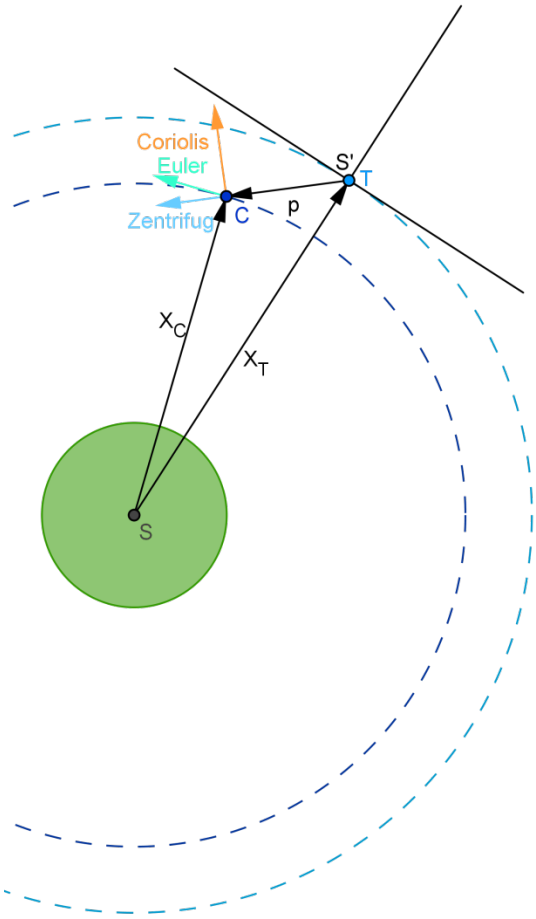
des Chaser-Satelliten sollte möglichst energiesparend und daher kostengünstig sein, aber auch möglichst schnell vonstatten gehen.

Um diesen Vorgang darstellen zu können, haben wir ein Modell erstellt, in dem die Bewegungen beider Satelliten berechnet wurden. Dazu wurde der Erdmittelpunkt als Ursprung eines Koordinatensystems (des Inertialsystems S) angenommen. Der Mittelpunkt des Schrottstückes (als Target bezeichnet) wird als Ursprung eines zweiten Koordinatensystems (S‘) verwendet, der sich auf einer Kreisbahn mit konstanter Geschwindigkeit um das Inertialsystem dreht.

Auf die Satelliten wirkt die Gravitation, dieser entgegen die Zentrifugalkraft. In einem bewegten System muss man zusätzlich berücksichtigen, dass Scheinkräfte wirken, wie die Eulerkraft und die Corioliskraft.

Für die Modellierung benötigten wir Differentialgleichungen und die Grundprinzipien der Optimalsteuerung.

Variablen



C … Chaser

T … Target

S … Inertialsystem (Erdmittelpunkt = Ursprung)

S‘ … rotierendes Bezugssystem mit Ursprung C

X_T bzw. X_C … Die Position von T bzw. C in S

p … Die Position von C in S‘

ω … Winkelgeschwindigkeit in S

u … kontrollierbare Kraft von C

m_1 … Masse des Satelliten

m_2 … Masse der Erde

G … Gravitationskonstante ($6{,}673*10^{-11}$ $m^3 * kg^{-1} * s^{-2}$)

μ … $G*m_2$

Dynamik in beschleunigten Bezugssystemen

S‘ bewegt sich relativ zu S beschleunigt. Es gilt:

$$X_C = X_T + p$$

Nun wollen wir $\dot{X}_C$ berechnen, indem wir X_C ableiten. Im Folgenden werden die Ableitungen mit $\dot{}|_I$ bzw. $\dot{}|_R$ versehen um zwischen Ableitungen im Inertialsystem S und im bewegten System S‘ zu unterscheiden.

$$\dot{X}_C|_I = \dot{X}_T|_I + \dot{p}|_I$$

Die Veränderung des Vektors p aus der Sicht von S setzt sich zusammen aus der relativen Veränderung aus der Sicht von S‘ und der Rotation von S‘ aus der Sicht von S.

$$\dot{p}|_I = \dot{p}|_R + \omega \; x \; p$$

Daraus ergibt sich

$$\dot{X}_T|_I = \dot{X}_T|_I + \dot{p}|_I$$

Als nächstes wird die Beschleunigung durch Ableiten der Geschwindigkeit berechnet. Zunächst leiten wir $\dot{p}|_I$ ab.

$$\frac{d\dot{p}|_I}{dt}\Big|_I = \frac{d\dot{p}|_I}{dt}\Big|_R + \omega \; x \; \dot{p}|_I$$

$$= \frac{d}{dt}\Big|_R (\dot{p}|_R + \omega \; x \; p) + \omega \; x \; (\dot{p}|_R + \omega \; x \; p)$$

$$= \frac{d\dot{p}|_R}{dt}\Big|_R + \dot{\omega} \; x \; p + 2\omega \; x \; \dot{p}|_R + \omega \; x \; (\omega \; x \; p)$$

Insgesamt ergibt sich für die Beschleunigung bezüglich des Inertialsystems

$$\ddot{X}_C|_I = \ddot{X}_T|_I + \ddot{p}|_R + \dot{\omega} \; x \; p + 2\omega \; x \; \dot{p}|_R \; + \; \omega \; x \; (\omega \; x \; p)$$

Kraftwirkung auf die Satelliten

Um die Gesamtkraft, die auf den Chaser-Satellit in S‘ wirkt, zu berechnen, muss man die relative Beschleunigung $\ddot{p}|_R$ mit der Masse multiplizieren.

$$F_{ges} = \ddot{p}|_R * m = \ddot{X}_C|_I * m - \ddot{X}_T|_I * m - m * \dot{\omega} \; x \; p - 2\mathrm{m} * \omega \; x \; \dot{p}|_R - \; m * \omega \; x \; (\omega \; x \; p)$$

F_{ges} sich zusammen aus der Gravitationskraft ($\ddot{X}_C|_I * m$), der Trägheit des Bezugssystems S‘ ($\ddot{X}_T|_I * m$), der Eulerkraft (m*$\dot{\omega} \; x \; p$), der Corioliskraft ($2 * \mathrm{m} * \omega \; x \; \dot{p}|_R$) und der Zentrifugalkraft ($m * \omega \; x \; (\omega \; x \; p)$).

Die Gravitationskraft (Erdanziehungskraft) zieht die Satelliten in Richtung Erdmittelpunkt. Die Zentrifugalkraft, auch Fliehkraft, wirkt der Gravitationskraft entgegen. Zusätzlich wirkt auf den Chaser-Satelliten auch noch die Corioliskraft, weil er sich in einem rotierenden Koordinatensystem bewegt, die seine Bewegungen ablenkt. Außerdem muss die Eulerkraft berücksichtigt werden. Das ist die Trägheitskraft, die aufgrund von Veränderung der Winkelgeschwindigkeit entsteht (vergleichbar mit der Kraft, die auf eine Person in einem Karussell wirkt, wenn dieses die Winkelgeschwindigkeit verändert). Dazu kommt noch die Kraft die auf C wirkt, wenn bei konstanter Position von C der Ursprung von S‘ bzw. T beschleunigt wird.

Herleitung der Gravitationskraft

Die Gravitationskraft lässt sich mit dem newtonschen Gravitationsgesetz berechnen.

$$|\vec{F}_G| = \frac{m_1 * m_2}{r^2} * G$$

wobei r = $|X_T|$ der Abstand zwischen m_1 und m_2 ist

bzw. in vektorieller Form

$$\vec{F}_G = \frac{m_1 * m_2}{r^2} * G * X_T$$

Der Vektor der Gravitationskraft ist gleich dem Betrag der Gravitationskraft multipliziert mit dem Einheitsvektor X_{T_0}. Der Einheitsvektor entsteht durch Division durch den Betrag des Vektors, daher der Nenner $|X_T|^3$. Weiter gilt

$$\vec{F}_G = \frac{m_1 * m_2}{|X_T|^3} * G * X_T = m_1 * \ddot{X}_T|_I$$

oder

$$\frac{G * m_2}{|X_T|^3} * X_T = \ddot{X}_T|_I$$

Im Folgenden ersetzen wir $G*m_2 = \mu$.

Das Gleiche wie für $\ddot{X}_T|_I$ gilt auch für $\ddot{X}_C|_I$ nur wird hier zusätzlich die Kontrolle $\frac{u}{m}$ dazu addiert. Hier bezeichnet m die Masse des Chasers.

$$\ddot{X}_C = \frac{\mu}{|X_C|^3} * (X_C) + \frac{u}{m}$$

Da der Vektor X_C aus $X_T + p$ entsteht, gilt

$$\ddot{X}_C|_I = \frac{\mu}{|X_T + p|^3} * (X_T + p) + \frac{u}{m}$$

Gleichungen zur Steuerungen des Chaser-Satelliten

Der Vektor p entsteht aus $X_c - X_T$, also

$$\ddot{p}|_I = \frac{-\mu}{|X_T|^3} * X_T + \frac{\mu}{|X_T + p|^3} * (X_T + p) + \frac{u}{m}$$

Es gilt $X_T = \begin{pmatrix} Tx \\ 0 \\ 0 \end{pmatrix}$, weil sich das Inertialsystems der Erde mit ω dreht. Weiter bezeichnen wir p mit

$$p = \begin{pmatrix} px \\ py \\ pz \end{pmatrix}$$

und

$$u = \begin{pmatrix} ux \\ uy \\ uz \end{pmatrix}$$

Ausgeschrieben in Koordinaten bedeutet das

$$\ddot{p}|_I = \frac{-\mu * \begin{pmatrix} Tx \\ 0 \\ 0 \end{pmatrix}}{Tx^3} + \frac{\mu * \begin{pmatrix} Tx + px \\ py \\ pz \end{pmatrix}}{((Tx + px)^2 + py^2 + pz^2)^{3/2}} + \frac{u}{m}$$

Außerdem kann $\ddot{p}|_I$ durch folgenden Ausdruck beschrieben werden:

$$\ddot{p}|_I = \ddot{p}|_R + \dot{\omega}\ x\ p + 2\omega\ x\ \dot{p}|_I + \ \omega\ x\ (\omega\ x\ p)$$

diese Formel wurde aus der Kraftwirkung auf die Satelliten (S. 4) hergeleitet (Anmerkung: $\dot{p}|_R \triangleq \dot{p}'$ und $\dot{p}|_I \triangleq \dot{p}$). Da wir ohne Beschränkung der Allgemeinheit annehmen können, dass die Rotationsebene gleich der X-Y-Ebene von S ist, gilt

$$\omega = \begin{pmatrix} 0 \\ 0 \\ \omega \end{pmatrix}$$

So gilt komponentenweise

$$\ddot{p} = \begin{pmatrix} \ddot{p}x' \\ \ddot{p}y' \\ \ddot{p}z' \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \dot{\omega} \end{pmatrix} x \begin{pmatrix} px \\ py \\ pz \end{pmatrix} + 2 * \begin{pmatrix} 0 \\ 0 \\ \omega \end{pmatrix} x \begin{pmatrix} \dot{p}x' \\ \dot{p}y' \\ \dot{p}z' \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \omega \end{pmatrix} x \left(\begin{pmatrix} 0 \\ 0 \\ \omega \end{pmatrix} x \begin{pmatrix} px \\ py \\ pz \end{pmatrix} \right)$$

bzw.

$$\ddot{p} = \begin{pmatrix} \ddot{p}x' \\ \ddot{p}y' \\ \ddot{p}z' \end{pmatrix} + \begin{pmatrix} -\dot{\omega} * py \\ \dot{\omega} * px \\ 0 \end{pmatrix} + 2 * \begin{pmatrix} -\omega * \dot{p}y' \\ \omega * \dot{p}x' \\ 0 \end{pmatrix} + \begin{pmatrix} -\omega^2 * px \\ -\omega^2 * py \\ 0 \end{pmatrix}$$

Diese Formel und die Formel für die Gravitationsbeschleunigung bzw. Kontrollbeschleunigung können nun gleichgesetzt werden:

$$\frac{-\mu * \begin{pmatrix} Tx \\ 0 \\ 0 \end{pmatrix}}{(Tx)^3} + \frac{\mu * \begin{pmatrix} Tx+px \\ py \\ pz \end{pmatrix}}{((Tx+px)^2+py^2+pz^2)^{3/2}} + \begin{pmatrix} ux/m \\ uy/m \\ uz/m \end{pmatrix} = \begin{pmatrix} \ddot{p}x' \\ \ddot{p}y' \\ \ddot{p}z' \end{pmatrix} + \begin{pmatrix} -\dot{\omega} * py \\ \dot{\omega} * px \\ 0 \end{pmatrix} + 2 * \begin{pmatrix} -\omega * \dot{p}y' \\ \omega * \dot{p}x' \\ 0 \end{pmatrix} + \begin{pmatrix} -\omega^2 * px \\ -\omega^2 * py \\ 0 \end{pmatrix}$$

$\begin{pmatrix} -\dot{\omega} * py \\ \dot{\omega} * px \\ 0 \end{pmatrix}$ = 0, weil die Winkelgeschwindigkeit als konstant angenommen wird (Kreisbahn).

Exemplarisch für die X-Komponente gilt:

$$\frac{-\mu * Tx}{Tx^3} + \frac{\mu * (Tx+px)}{((Tx+px)^2+py^2+pz^2)^{3/2}} + \frac{ux}{m} = \ddot{p}x' - 2 * \omega * \dot{p}y' - \omega^2 * px$$

Der Nenner des zweiten Bruch kann nun linearisiert werden:

$$((Tx + px)^2 + py^2 + pz^2)^{3/2} = (Tx^2 + 2 * Tx * px + px^2 + py^2 + pz^2)^{3/2}$$

$$= Tx^3 \left(1 + \frac{2*px}{Tx} + \frac{px^2+py^2+pz^2}{Tx^2}\right)^{3/2}$$

Hier kann der Term $\frac{px^2+py^2+pz^2}{Tx^2}$ weggelassen werden, da in diesem Modell der Abstand zwischen den beiden Satelliten $|p|$ sehr klein im Verhältnis zum Abstand zur Erde *Tx* ist.

ux/m = $\ddot{p}x' - 2 * \omega * \dot{p}y' + \frac{\mu * Tx}{Tx^3} - \frac{\mu * (Tx+px)}{Tx^3\left(1+\frac{2*px}{Tx}\right)^{3/2}} - \omega^2 * px$

Zum Auflösen der Wurzel im Term $\mu * (Tx + px) * Tx^{-3} \left(1 + \frac{2*px}{Tx}\right)^{-3/2}$ wird der Binomialsatz verwendet.

$$\left(1 + \frac{-2*px}{Tx}\right)^{-3/2} = \sum_{k=0}^{\infty} \binom{-\frac{3}{2}}{k} * 1^{-3/2-k} * \left(\frac{-2*px}{Tx}\right)^k$$

$$= \left(1 + \left(\frac{-3}{2} * \frac{-2px}{Tx}\right)^1 + \left(\frac{-3}{8} * \frac{-2px}{Tx}\right)^2 + \ldots\right)$$

Ab dem 3. Summanden darf man aus demselben Grund wie zuvor linearisieren, und erhält $Tx^{-3} * \left(1 + \left(\frac{-3}{2} * \frac{-2px}{Tx}\right)\right)$.

$\mathrm{ux/m} = \frac{\mu * Tx}{Tx^3} - \frac{\mu * (Tx - px) * (1 + \frac{3px}{Tx})}{Tx^3} + \ddot{p}x' - 2 * \omega * \dot{p}y' - \omega^2 * px$ … $\mu/\mathrm{Tx}^3 = \omega^2$

$\mathrm{ux/m} = \ddot{p}x' - 2 * \omega * \dot{p}y' + \omega^2 * Tx - \omega^2 * (Tx - px) * (1 + \frac{3px}{Tx}) - \omega^2 px$

$\mathrm{ux/m} = \ddot{p}x' - 2 * \omega * \dot{p}y' + \omega^2 * Tx - \omega^2 * (Tx + 3px - px - \frac{3px^2}{Tx}) - \omega^2 px$

Weil $\frac{3px^2}{Tx}$ sehr klein ist, kann der Term weggelassen werden.

$\mathrm{ux/m} = \ddot{p}x' - 2 * \omega * \dot{p}y' + \omega^2 Tx - \omega^2 * (Tx + 3px - px) - \omega^2 px$

$\mathrm{ux/m} = \ddot{p}x' - 2\omega\dot{p}y' - 3\omega^2 px$

Für die y- und z-Komponente von $\ddot{p}'$ kann analog verfahren werden :

$\mathrm{uy/m} = \omega^2 * py + 2 * \omega * \dot{p}x'$

$\mathrm{uz/m} = \omega^2 py + \ddot{p}z'$

Zusammengefasst erhält man also für den Vektor u

$\mathrm{ux/m} = \ddot{p}x' - 2\omega\dot{p}y' - 3\omega^2 px$

$\mathrm{uy/m} = \ddot{p}y' + 2\omega\dot{p}x'$

$\mathrm{uz/m} = \ddot{p}z' + \omega^2 py$

Das Optimalsteuerungsproblem

Das Ziel des Optimalsteuerungsproblems ist, das Target möglichst schnell oder unter möglichst geringem Kraftaufwand bzw. einem Kompromiss zwischen den beiden Komponenten (wobei beide Teile des Funktionalen mit c und d gewichtet werden können) zu erreichen.

$$\min_{u, t_f} \quad c * t_f + d * \int_0^{t_f} |u(t)|^2 dt$$

Dabei muss berücksichtigt werden, dass weder Zeit noch Kraft unendlich groß sein können. Das bedeutet, dass ein maximaler Wert für $|u|$ u_{max} und eine maximale Endzeit $t_f > 0$ vorgegeben wird.

$$ux(t)^2 + uy(t)^2 + uz(t)^2 \leq umax$$

$$t_f \leq tmax$$

Der relative Abstand zwischen Target und Chaser p und der Einfluß der Kontrolle u auf p ergibt sich aus den im letzten Abschnitt hergeleiteten Differentialgleichungen

$$\mathrm{ux/m} = \ddot{p}x' - 2\omega\dot{p}y' - 3\omega^2 px$$

$$\mathrm{uy/m} = \ddot{p}y' + 2\omega\dot{p}x'$$

$$\mathrm{uz/m} = \ddot{p}z' + \omega^2 py$$

Die Endbedingung ist, dass die relative Geschwindigkeit 0 ist, genau so wie der Abstand zwischen den beiden Satelliten.

$$p(t_f) = 0$$

$$\dot{p}'(t_f) = 0$$

Der Vorgang der Diskretisierung

Berechnung der Matrix

Um die unendliche Menge an Lösungen für den Computer, welcher nur über einen endlichen Speicherplatz verfügt, berechenbar zu machen, mussten wir unsere Funktion diskretisieren.

Eine Erläuterung des Vorgangs anhand einer einfachen Differentialgleichung:

$$\ddot{y}(x) = -\omega^2 y(x)$$

Wir führen eine neue Variable v ein, für die gilt:

$$\dot{y}(x) = v(x)$$

$$\dot{v}(x) = \ddot{y}(x) = -\omega^2 y$$

Wir führen den Vektor $Y = (y, v)$ ein. Dann lässt sich die letzte Gleichung als Produkt einer Matrix A mit dem Vektor Y darstellen:

$$\dot{Y} = \begin{pmatrix} a_{1/1} & a_{1/2} \\ a_{2/1} & a_{2/2} \end{pmatrix} * Y$$

Aus $\dot{Y} = \begin{pmatrix} \dot{y} \\ \dot{v} \end{pmatrix}$ kann man durch einsetzen alle a der Matrix bestimmen:

$$\begin{pmatrix} a_{1/1}y + a_{1/2}v \\ a_{2/1}y + a_{2/2}v \end{pmatrix} = \begin{pmatrix} v \\ \omega^2 y \end{pmatrix}$$

Es ergibt sich die Matrix A:

$$A = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix}$$

Verfahren zum Auflösen

Wenn man ẏ = A*y lösen will, verwendet man entweder den expliziten, den impliziten Euler oder das Crank-Nicolson Verfahren:

Expliziter Euler

$$\frac{y(k)-y(k-1)}{\tau} = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix} * y(k-1)$$

Durch umformen ergibt sich:

$$\boldsymbol{y(k) = y(k-1) * (A\tau + I) \quad I = Einheitsmatrix}$$

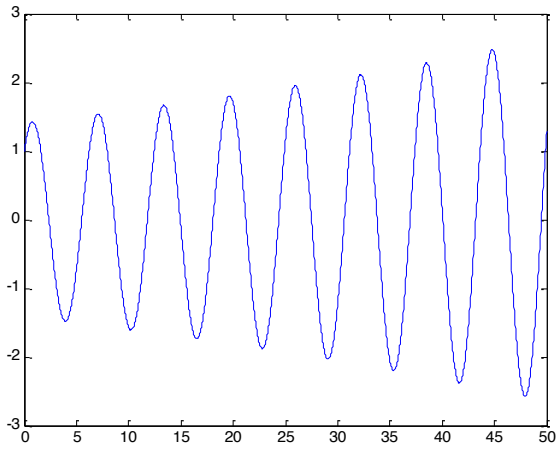
Die Diskretisierung selbst führen wir anschließend mit MatLab, welches ein Computerprogramm zur Lösung von mathematischen Problemen und zur graphischen Darstellung von den Ergebnissen ist, durch:

```
function [p] = diskretisierung ()

n=1000;
y = zeros(2,n);
tend=50;
T=linspace(0,tend,n);
omega=1;
A =[0,1;-omega^2,0];
I =[ 1,0;0,1];
y(:,1)=[1;1];
tau= T(2)-T(1);


for k=2:n
  y(:,k)= (I+(tau*A))*y(:,k-1);

end
```

Anschließend plotten wir die T und y(,:1) und T und y(,:2):

```
figure(1);
plot(T,y(1,:));
figure(2)
plot(T,y(2,:));
```

Es ergibt sich für beide Figuren eine aufschaukelnde Oszillation:



Impliziter Euler

$$\frac{y(k)-y(k-1)}{\tau} = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix} * y(k)$$

Durch Umformen ergibt sich:

$$\boldsymbol{y(k) = \frac{y(k-1)}{I - A\tau}}$$

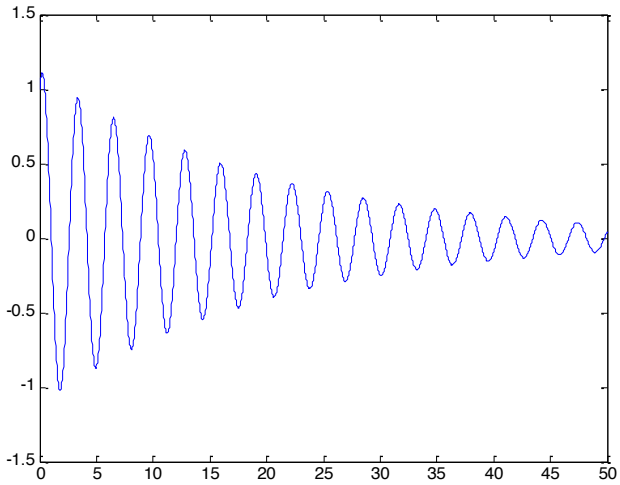
Wir diskretisieren wieder in MatLab:

```
function [p] = Diskretisierung4 ()

n=2000;
y = zeros(2,n);
tend=50;
T=linspace(0,tend,n);
omega=2;
A =[0,1;-omega^2,0];
I =[ 1,0;0,1];
y(:,1)=[1;1];
tau= T(2)-T(1);

for t=2:n
    y(:,t)= (I-(tau*A))\(y(:,t-1));
end
```

Wir plotten T und y(,:1) und T und y(,:2):

```
figure(1);
plot(T,y(1,:));
figure(2)
plot(T,y(2,:));
```

Es ergibt sich eine gedämpfte Schwingung:



Crank-Nicolson

Dieses Verfahren kann man als den Mittelwert des impliziten und expliziten Eulerverfahrens beschreiben.

Allgemeine Schreibweise

Man führt eine Variable (hier σ) ein, welche y(k) und y(k-1) miteinander verbindet:

$$\frac{y(k)-y(k-1)}{\tau} = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix} * ((\mathbf{1}-\boldsymbol{\sigma}) * \boldsymbol{y(k)} + \boldsymbol{\sigma} * \boldsymbol{y(k-1)})$$

Das Sigma verbindet das implizite Eulerverfahren mit dem expliziten Eulerverfahren

Durch festlegen des Sigma-Wertes legt man fest wie sehr implizit (gedämpft) bzw. explizit (verstärkt) die Berechnung ist.

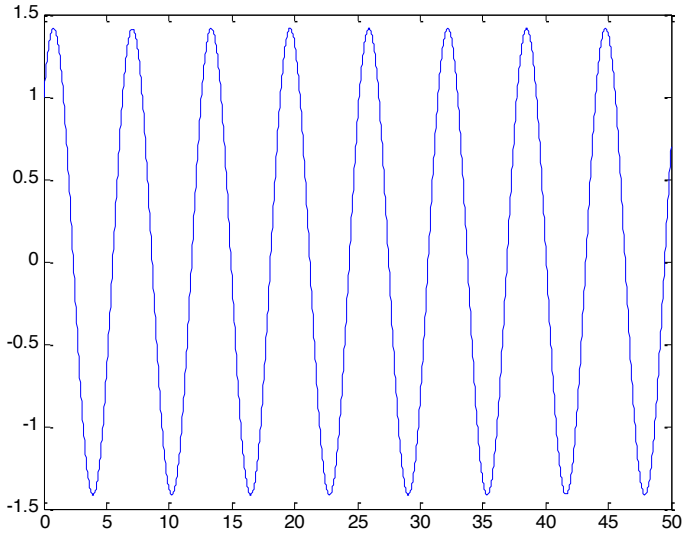
Für Sigma gilt: $\mathbf{0} \leq \boldsymbol{\sigma} \leq \mathbf{1}$ (0 = impliziter Euler; 1 = expliziter Euler; 0.5 = Crank-Nicolson)

Es ergibt sich:

$$\boldsymbol{y(k)} = \frac{(\boldsymbol{A\tau\sigma} + \boldsymbol{I}) * \boldsymbol{y(k-1)}}{\boldsymbol{I} - \left(\boldsymbol{A\tau} * (\mathbf{1}-\boldsymbol{\sigma})\right)}$$

Wir diskretisieren und plotteten äquivalent zu vorher in MatLab (mit σ=0,5):

Es ergibt sich eine Oszillation mit konstanter Amplitude:



Das eigentliche Problem

Ausgegangen sind wir von den folgenden drei bereits hergeleiteten Gleichungen:
(Anmerkung: statt ω wurde hier n verwendet, das p aus px, py und pz wurde der Einfachheit halber weggelassen, d.h. px ≙ x, py ≙ y und pz ≙ z)

$$\ddot{x} - 2n\dot{y} - 3n^2x = 0$$

$$\ddot{y} + 2n\dot{x} = 0$$

$$\ddot{z} + n^2z = 0$$

Diese haben wir mit den oben genannten Verfahren gelöst. Daraufhin haben wir die Gleichung vereinfacht und in MatLab eingegeben.

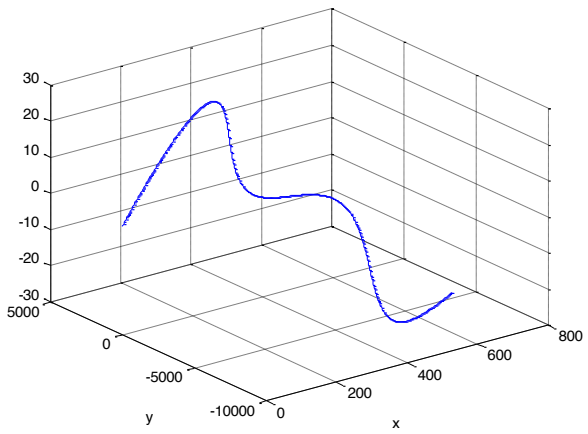
Außerdem haben wir eine Kontrolle u hinzugefügt. Mit dieser können wir Einfluss auf die Funktion nehmen, indem wir Kräfte hinzufügen oder diese gleich null setzen.

```
function [p] = diskretisierung7 ()
n=1000;
y=zeros(6,n);
u=zeros(6,n);
tend=5;
T=linspace(0,tend,n);
l = pi;
A = [0,1,0,0,0,0;
     (3*l^2),0,0,(2*l),0,0;
     0,0,0,1,0,0;
     0,-2*l,0,0,0,0;
     0,0,0,0,0,1;
     0,0,0,0,-l^2,0];
Sigma=0.5;
I = [1,0,0,0,0,0;
     0,1,0,0,0,0;
```

```
      0,0,1,0,0,0;
      0,0,0,1,0,0;
      0,0,0,0,1,0;
      0,0,0,0,0,1];
x0=0;
x1=0;
y0=0;
y1=0;
z0=0;
z1=0;
u(4,:) = sin(T);
u(4,:) = ones(1,n);
u(6,:) = sin(T);
y(:,1) = [x0;x1;y0;y1;z0;z1];

tau= T(2)-T(1);

for k = 2:n
y(:,k)=(I-((A*tau)*(1-Sigma)))\((((A*Sigma*tau)+I)*(y(:,k-
1)))+(Sigma*u(:,k))+(1-Sigma)*u(:,k-1));
end
```

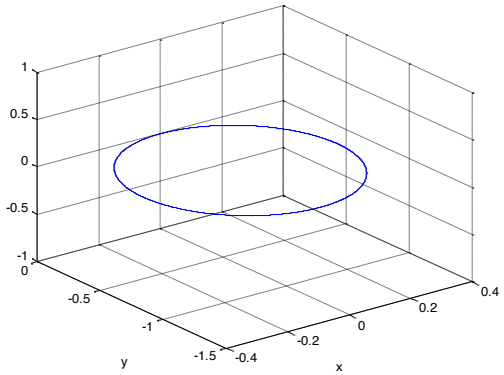
Durch die drei Gleichungen haben wir x-, y- und z-Werte erhalten. Somit ist diese Funktion dreidimensional. Unsere Funktion plotten wir dann.

```
figure(18);
plot3(y(1,:),y(3,:),y(5,:))
xlabel('x')
ylabel('y')
grid on
```



Beispiel 1

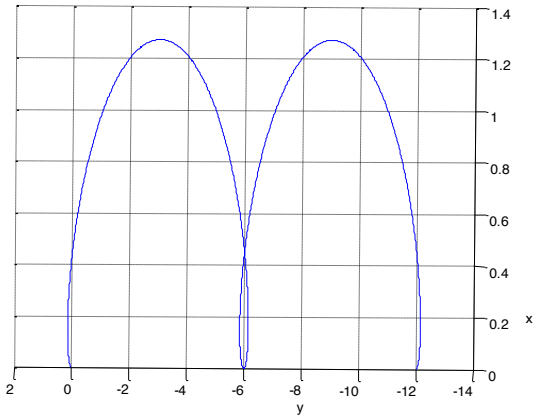
Wir betrachten zuerst das Beispiel, dass z.B. ein Astronaut einen Gegenstand hinunter Richtung Erde wirft und somit nur in eine Richtung beschleunigt. Somit schalten wir alle unsere Kontrollen aus und setzen für unsere Werte für x, y und z am Anfang null ein. Nur für die Geschwindigkeit in Richtung der x-Achse setzen wir einen Wert, in unserem Fall 1, ein. Dann plotten wir dies und erhalten einen Graphen, der die Bewegung des Gegenstands beschreibt.



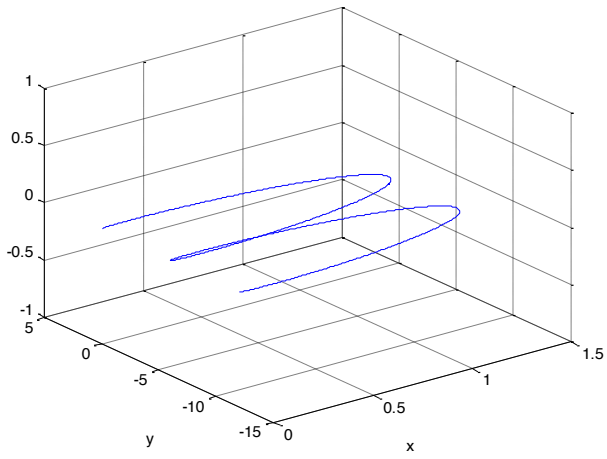
Beispiel 2

Ein weiteres Beispiel wäre eine alleinige Beschleunigung in die y-Richtung. Diese Kurve nennt man Zykloide.

Um dies zu simulieren, setzen wir alle Anfangswerte auf null, nur die Anfangsgeschwindigkeit in Richtung y-Achse setzen wir auf einen Wert, in unserem Fall auf eins.



Ansicht: x-,y-Achse



Ansicht: x-, y-, z-Achse

Optimierung

AMPL, NEOS-Server und IPOPT

Um unser Optimierungsproblem lösen zu können, brauchen wir nun ein Programm, welches uns Ergebnisse für die Variablen liefert, die es zu optimieren gilt. Hierfür programmieren wir die vorher gewonnenen Erkenntnisse in AMPL. AMPL ist allgemein eine mathematische Programmiersprache, die speziell für komplizierte Optimierungsprobleme geeignet ist. AMPL kann diese Probleme jedoch selbst nicht lösen und daher verwenden wir den NEOS-Server mit IPOPT um unser Optimierungsproblem zu lösen.

AMPL file

Unser file beginnen wir mit den Variablen, die optimiert werden sollen:

```
var tf >=0, default 1500;
```

Als nächstes werden die Zustandsvariablen definiert:

```
var x{0..N};
```

Nun kommt der wichtigste Teil in unserem file: die Gleichung, die es zu optimieren gilt:

```
minimize f: c*tf + d*(tf/N) * sum{i in 0..N-1}(ux[i]^2 + uy[i]^2 +
uz[i]^2);
```

Danach müssen wir noch einige wichtige Variablen definieren:

```
param tmax default 10800;
```

Die Bedingungen für die Optimierung fügen wir nun als nächstes ein, angefangen mit den Startbedingungen:

```
subject to Bereich_start1: x[0]=10;
```

Dann folgen die Begrenzungen für Schub und Zeit:

```
subject to Bereich_ux{i in 0..N-1}: ux[i]^2 + uy[i]^2 + uz[i]^2 <= umax^2;

subject to Bereich_tf: tf <= tmax;
```

An nächster Stelle stehen die Differentialgleichungen.

```
subject to Bereich_x{i in 1..N}: x[i]-x[i-1]= (tf/N)*((b*vx[i])+((1-
b)*vx[i-1]));
```

Um festzulegen, dass der Chaser am Ende beim Target angekommen sein soll, müssen wir die letzten Koordinaten 0 setzen:

```
subject to Bereich_end1: x[N]=0;
```

Zum Schluss geben wir `solve;` ein, damit das Problem gelöst wird und geben an, welche Variablen ausgegeben werden sollen: `display tf;`

NEOS-Server

Das fertige Dokument senden wir nun an den NEOS-Server, der uns die Ergebnisse per E-Mail sendet und im Browser anzeigt.

Beispiele

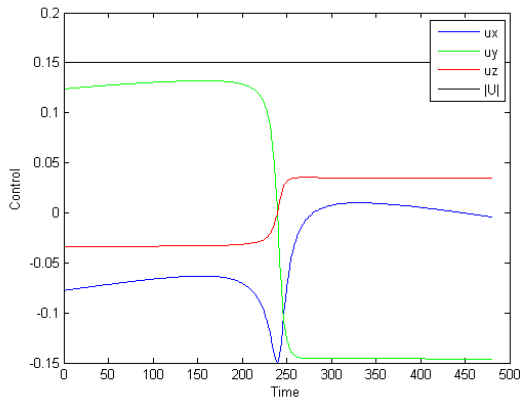
Im folgenden Abschnitt haben wir Beispiele mit einer Variation verschiedener Parameter dargestellt. Wir haben die Daten als Matrix in MatLab gespeichert, und als Graph ausgegeben.

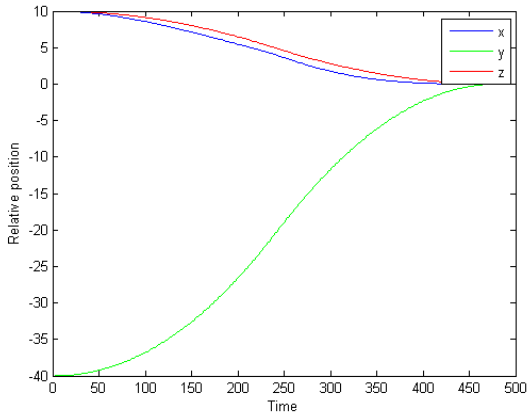
Gewichtung des Zielfunktionals:

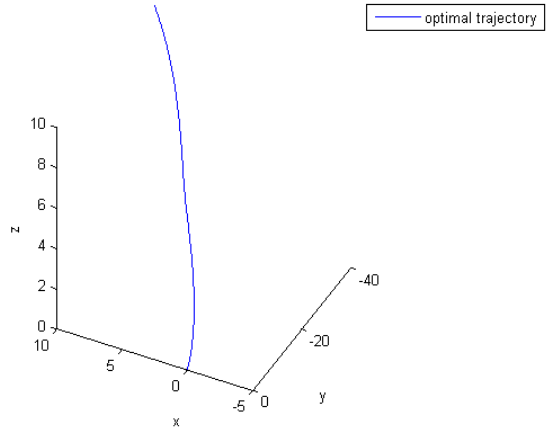
```
minimize f: c*tf + d*(tf/N) * sum{i in 0..N-1}(ux[i]^2 + uy[i]^2 +
uz[i]^2);
```

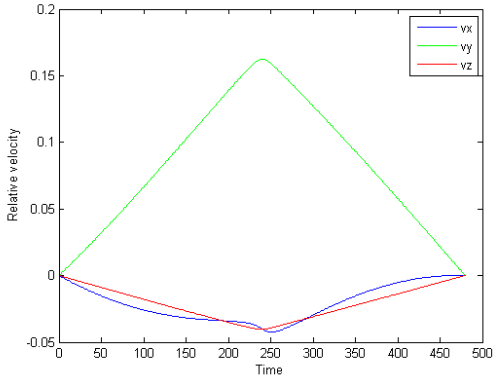
Bei den folgenden Beispielen haben wir eine Gewichtung der Teile in der Zielfunktion durchgeführt, durch die Veränderung der Parameter c und d.

Gewichtung c:1 d:0



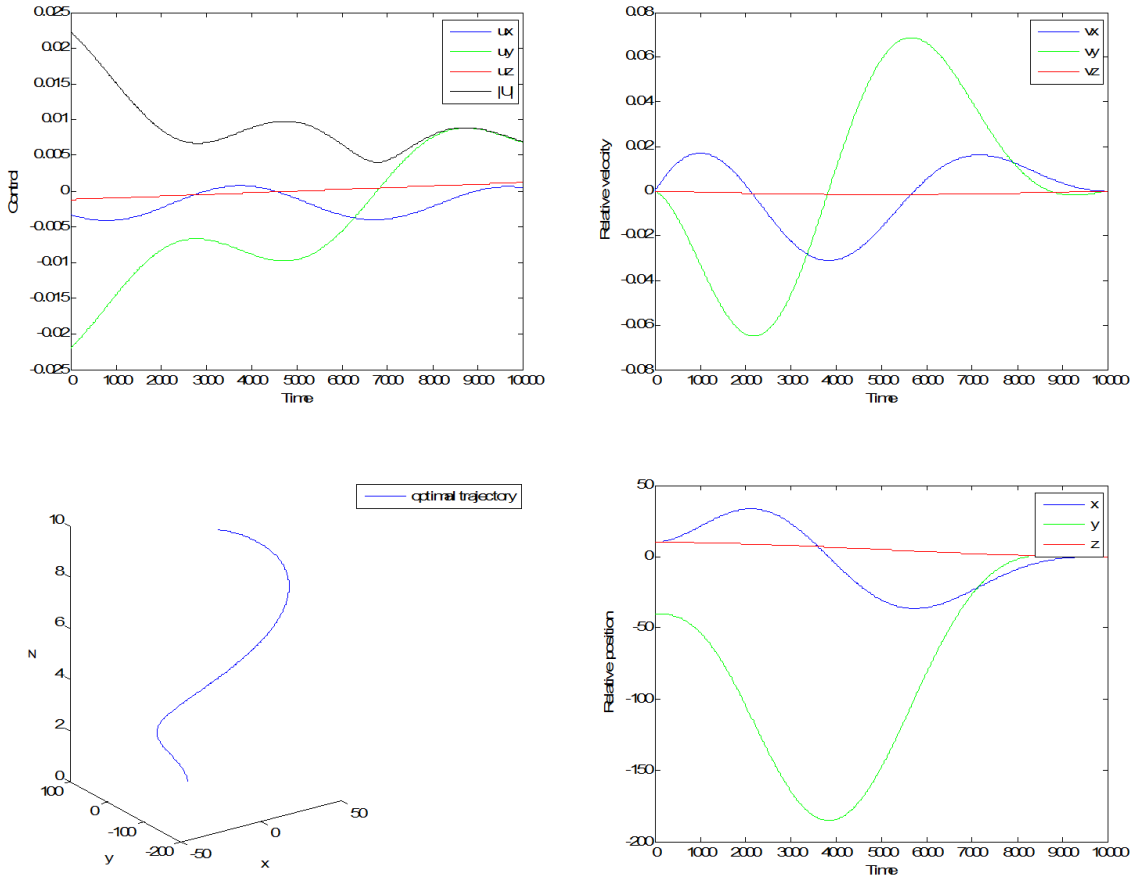






Bei dieser Variante wird nur die Zeit optimiert → unser Programm verwendet den maximal eingestellten Schub von 0,2 (siehe control). Daher wird zwar die benötigte Zeit sehr kurz ausfallen, aber der Energieverbrauch wird sehr hoch sein → nicht optimal.

Gewichtung c:0 d:1

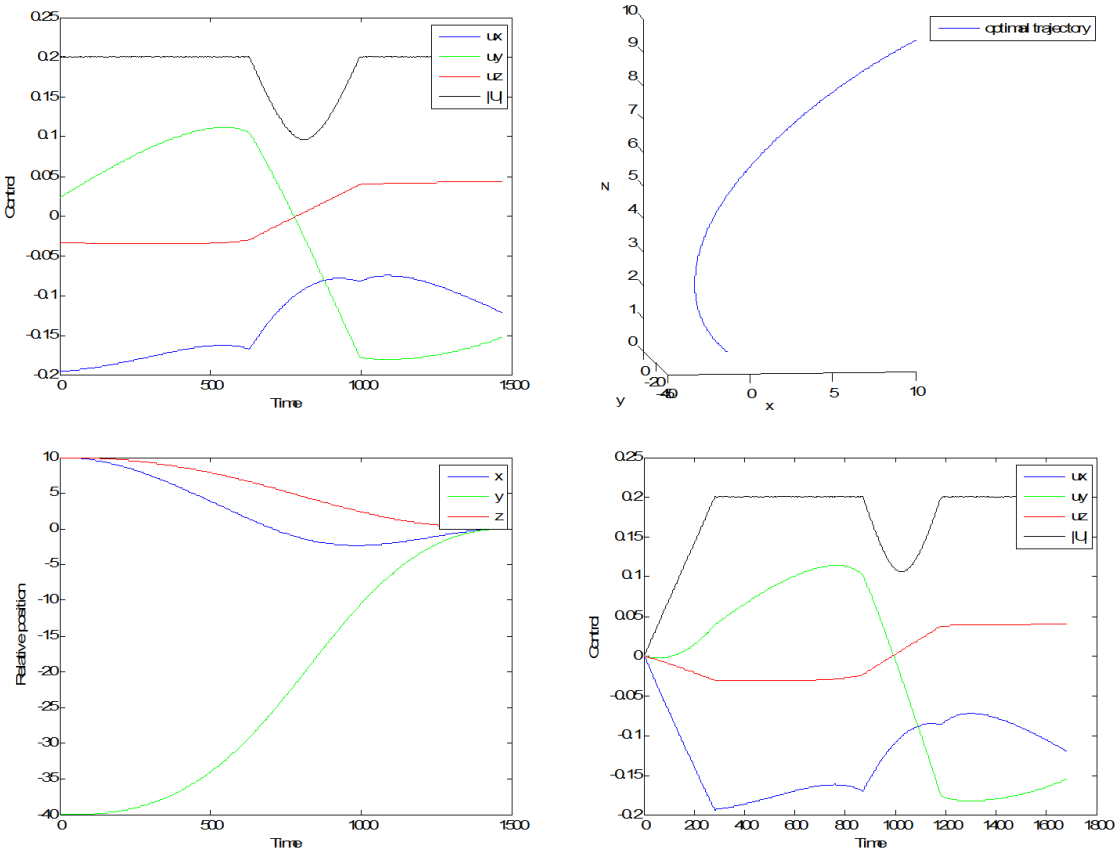


Bei dieser Variante verwendet der Satellit die Zeit, die als maximal eingestellt worden ist, in unserem Fall: 10000. Dabei wird versucht den Schub möglichst gering zu halten. Diese Variante ist zwar energieeffizienter, dauert aber lange. Man könnte nun die gewünschte Zeit als t_{max} einstellen, aber eine Berechnung mit gleicher Gewichtung bietet sich eher an.

Gewichtung c:1 d:1

Bei dieser Gewichtung erreicht man ein gleiches Ergebnis, wie bei c:1 d:0, da die Schubbegrenzung immer noch voll genutzt wird.

Gewichtung c:1 d:5



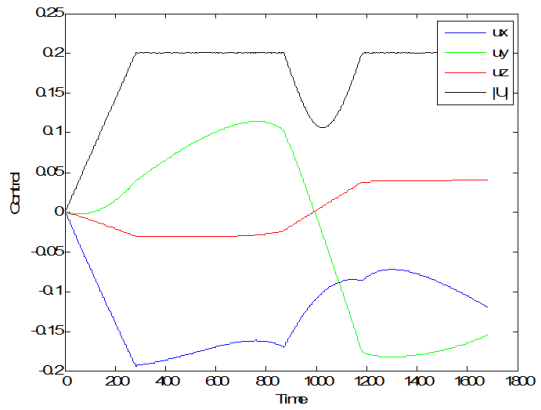
Nun lässt sich eine Veränderung im Schub erkennen, die etwa bei der Hälfte der Zeit entsteht: Die Schubkurve fällt in einer Kurve nach unten ab → es wird weniger Schub benötigt, um das Ziel zu erreichen auf Kosten der Zeit. Diese Variante ist aber Energiesparender.

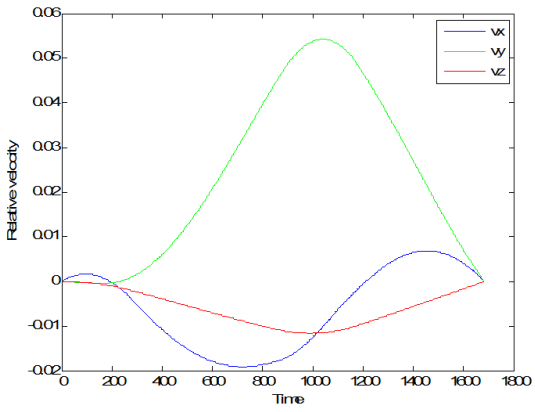
Veränderung der Bedingungen

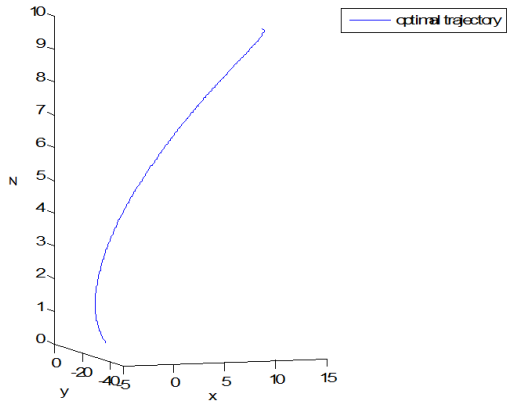
Da das Triebwerk unseres Satelliten eine gewisse Zeit zum Starten benötigt, führen wir eine weitere Bedingung ein, die den maximalen Schub in der Startphase des Triebwerks begrenzt:

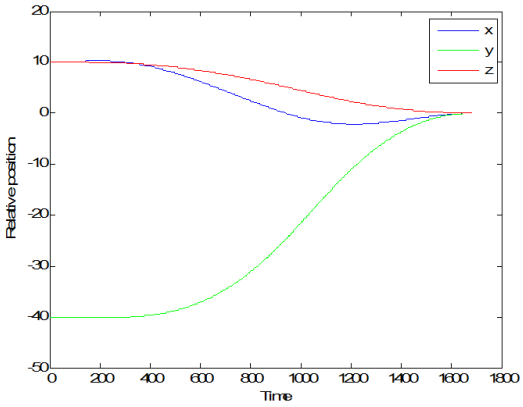
```
subject to Bereich_ux1{i in 0..100}: ux[i]^2 + uy[i]^2 + uz[i]^2 <=
(umax*0.01*i)^2;
```

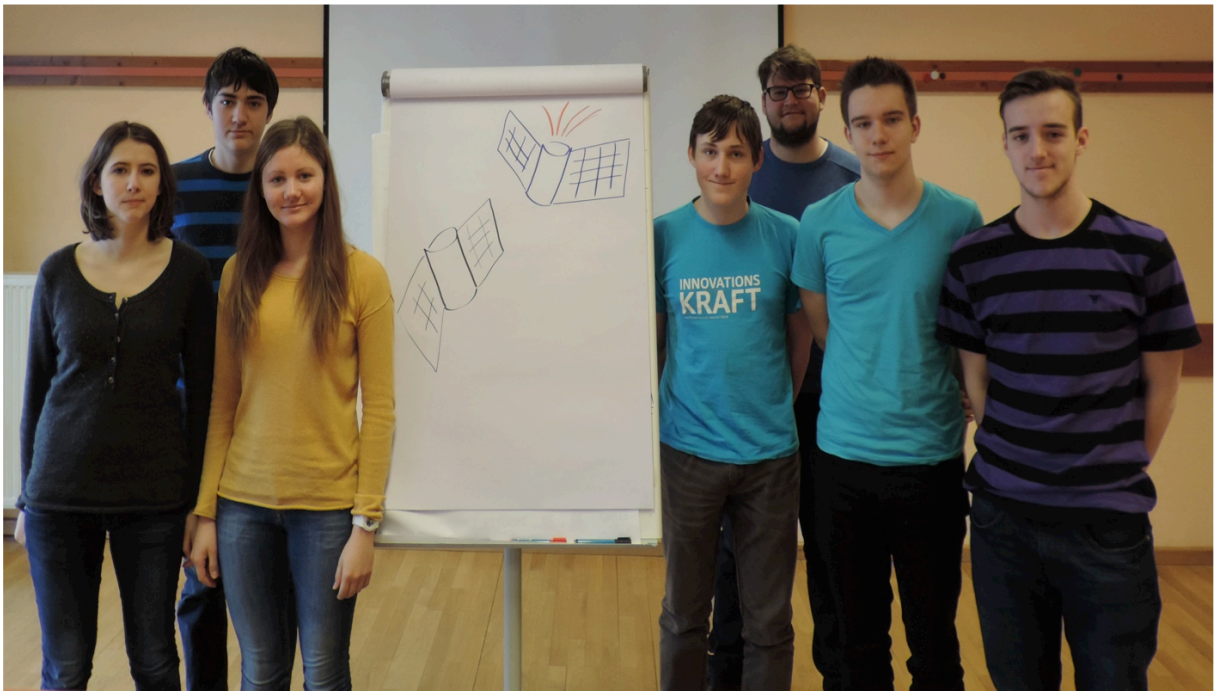
Mit dieser neuen Bedingung und dem Verhältnis c:1 d:5 entstehen folgende neue Graphen:











Digitale Bildverarbeitung- Steganographie und Bilder

Datum: 8-14 Februar 2014

Betreuer: Mag. Dr. Martin Holler

Gruppe:

Alexander Prutsch
Felix Breuß
Larissa Hammer
Lisa Kogler
Mira Höggerl
Roxana Trif

Inhaltsangabe

1 Einleitung

In der heutigen Zeit wird es immer wichtiger eigene Daten zu sichern und vor Raubkopien zu schützen. Spätestens seit der Enthüllung der Geheimorganisation NSA ist die Medienpräsens dieses Thema immens gestiegen. Eine Möglichkeit seine Daten zu verstecken ist mit Hilfe von Steganographie.

Aus dem altgriechischen übersetzt bedeutet Steganographie ‚bedecktes' oder ‚geheimes schreiben'. Es besteht eine gewisse Ähnlichkeit zur Kryptographie. Der Unterschied ist allerdings, dass ein Außenstehender bei der Kryptographie weiß, dass Informationen existieren, die aber aufgrund des fehlenden Schlüssels zur Entschlüsselung nicht decodieren kann - Bei der Steganographie weiß der Dritte nicht einmal, dass Informationen übertragen werden.

Es gab sogar schon in der Antike Verfahren die der Steganographie ähneln, zum Beispiel wurden zu dieser Zeit Sklaven der Kopf rasiert und dann die Kopfhaut tätowiert, nachdem die Haare nachgewachsen waren, wurde er weggeschickt. Ein anderes Beispiel sind Wachstafeln, bei deinen zuerst etwas in die Holzblatte geritzt worden ist und dies ist dann mit Wachs übergossen/versteckt worden. Das Ziel der Steganographie ist also die Vertraulichkeit und die Geheimhaltung von Information in einem Trägermedium. Ein gutes Beispiel dafür sind Bilder, da pro Sekunde tausende von Bildern über das Internet versendet werden, daher wird eher wenig Verdacht erschöpft, dass Informationen gerade in Bildern gespeichert werden.

Wir spezialisierten uns hauptsächlich auf das Bearbeiten von Bildern, aber weiteres auch auf Filme. Unser Hauptarbeitswerkzeug war das Programm MATLAB.

2 Einführung in unser Thema

Am Beginn unseres Projektes übten wir den Umgang mit dem Programm MATLAB, welches für uns alle fremd war und mit dem wir somit auch keine Erfahrung hatten.

Wir lernten zuerst essentielle Befehle, wie das Einlesen von Bildern, das Anzeigen von Bildern oder auch das Speichern. Weiterführend beschäftigten wir uns damit die Größe eines Bildes zu ändern, sowie gewisse Pixel beziehungswiese Ausschnitte einzufärben.

Außerdem lernten wir auch Theoretisches über Steganographie und die Dateistruktur von unterschiedlichen Bilddateiformaten. Wichtig dabei ist, dass die Bilder aus Pixel, welche wiederum in Bits unterteilt sind, bestehen. Dies erwies sich bereits am Beginn unserer Arbeit als äußerst wichtig, da es einen großen Unterschied macht in welchem Bit man seine Information codiert:

Wenn man das erste Bit verändert erkennt man im Bild selbst keinen Unterschied, allerdings ist die Gefahr die Information zu verlieren auch sehr hoch. Ein Bild im achten Bit-Level eines anderen Bildes zu verstecken ist gelinde gesagt nicht sehr sinnvoll, da dann das Bild, das versteckt werden soll das andere zur Gänze überschreibt.

Weiteres wurde auch der Unterschied zwischen Farbbildern und Schwarzweiß-Bildern besprochen, der darin besteht, dass Schwarzweiß-Bilder nur eine Farbkomponente haben und Farbbilder drei.

3 Bilder in Bilder verstecken

Nachdem wir mit den Grundfunktionen vertraut waren, gingen wir einen Schritt weiter: Wir entwickelten einen Code, der es uns ermöglichte Bilder in Bilder zu verstecken. Dazu nahmen wir die Bits, also die Nuller und Einser die die

Bildinformationen beinhalten und speicherten sie in die Bits des anderen Bilder hinein.

Zunächst passten wir die Größe immer manuell an und der Code funktionierte auch nur mit Schwarzweiß-Bilder. Wir verwendeten dafür for-Schleifen sowie die Funktionen „bitget“ und „bitset“, welche wir in unserem weiteren Umgang mit MATLAB noch sehr häufig verwendeten.

Simultan dazu erstellten wir außerdem noch eine Ausleseroutine, die die Informationen wieder aus dem Geheimbild auslas und das verstecke Bild ausgab. Das verwendete Dateiformat war PNG.

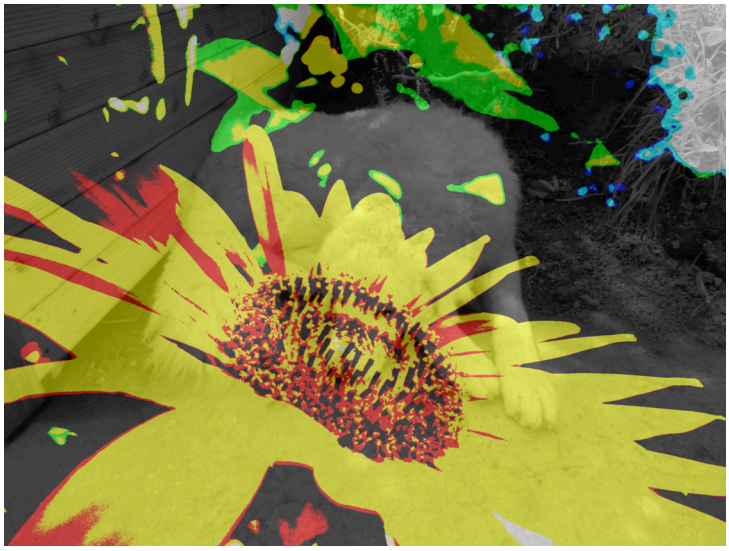
Graubild, in dem ein Graubild im 2. Bit-Level versteckt ist

Graubild in dem ein Graubild im 7. Bit-Level versteckt ist

Zweiter Schritt war es die Größe der Bilder nicht mehr mit der Hand anzupassen sondern diese Funktion in den Code zu integrieren.

Nachdem wir es erreicht hatten unterschiedlich große Schwarzweiß-Bilder in Schwarzweißbilder zu verstecken, begannen wir auch die anderen Kombinationen, also Farbbilder in Farbbilder, Schwarzweißbilder in Farbbilder sowie Farbbilder in Schwarzweißbilder zu verstecken, zu lösen. Wir erweiterten unseren Code daher zuerst um if-Abfragen, die überprüften, um welche Bild-Typen es sich handelt und dann dementsprechende Befehle ausführen. Grau in Grau sowie Farbe in Farbe waren kein Problem. Für Farbe in Grau fanden wir

nur eine suboptimale Lösung: Wir konvertierten einfach das farbige Bild in ein schwarzweißes. Für grau in farbig mussten wir den Code erweitern, so dass er die Informationen des schwarzweiß-Bildes in alle drei Komponenten des Farbbildes speichert.



Farbbild in dem ein Graubild im 7. Bit-Level versteckt ist

Farbbild in dem ein Graubild im 2. Bit-Level versteckt ist

Mit unserem vollständigen waren Code waren wir in der Lage von der Größe und der Farbe unabhängige Bilder in andere Bilder zu verstecken.

4 Text zu Binär Code und Binär Code zu Text

Nach diesen ersten gemeinsamen Entwicklungen und Erfolgen teilten wir uns in drei Gruppen auf, welche drei unterschiedliche Problemstellungen bearbeiteten.

Die erste Gruppe befasste sich mit zwei Funktionen zu Binärcode, welche essentiell für die weitere Arbeit waren und auch von den beiden anderen Gruppe mitverwendet , also in ihre Codes integriert, wurden.

Der Binärcode besteht nur aus Nullen und Einsen. Für jeden Buchstaben, sowie für jedes Zeichen gibt es eigenen achtstelligen Code.

Die Umrechnung von Zahlen in den Binärcode lässt sich leicht ohne Tabellen oder Auswendiglernen durchführen: Die erste Stelle des Binärcodes ist 2^0=1, die zweite 2^1=2, die dritte 2^2=4 usw. So besitzt beispielsweise die Zahl 19 einen Binärcode von 00010011 ($2^4+2^2+2^0$). Zahlen ab 256 erfordern eine Darstellung von mehr als nur 8-Bit, 578 hat beispielsweise einen Binärcode von 0000001001000010 ($2^9+2^6+2^1$).

4.1 Text2bin

Ziel war es eine Funktion in MATLAB zu entwickeln, die einen beliebigen Text in Binärcode umwandelt. Jeden Buchstaben und jedes Zeichen stellt der Computer mithilfe einer 8-stelligen Zahl, einem sogenannten Binärcode, dar. Alles, was der Computer darstellen kann, wird mithilfe des Binärsystems gerechnet.

In die Funktion muss man den Text eingeben, der dann als Rückgabewert der Funktion umgewandelt von einem Text in eine Binärzahl, zurückgegeben wird.

Beispiel: ‚a‘ -> 01100001

Probleme gab es hier leider auch. Bei unserem ersten Programm wurde die Zahlenfolge als Text ausgegeben und nicht als Zahlenreihe, wodurch wir die Zahlen nicht weiter verwenden konnten. Diesen Fehler haben wir jedoch behoben, indem wir mit einem Befehl die Ausgabe, ein String (Text), in eine Zahl (Integer) umgewandelt haben. Dadurch konnten wir die Zahlenfolge weiterhin im Programm verwenden.

4.2 Bin2text

Das Ziel beim Aufstellen dieser Funktion war, eine Funktion aufzustellen, die einen Binärcode in einen Text umgewandelt. Man gibt den Binärcode, eine 8-stellige Zahl aus Nullen und Einsern an, und aus jener Zahl wird dann ein Zeichen und der Rückgabewert der Funktion ist das dem Binärcode zugeordnete Zeichen. Jede solcher Zahlen ist einem Zeichen zugeordnet, Standardzeichen sind die Zeichen die den Binärcodes von 8-mal null bis 8-mal Einsern, das sind 256 Zeichen.

Beispiel: 01000001 -> „A“

Hier wird zuerst die Zahlenfolge von einem String in einen Integer umgewandelt (Text zu Zahl). Danach wird diese Reihe an jeweils 8 Zahlen in einen String umgewandelt und an das Letzte angehängt. Dadurch entsteht ein String mit allen Zeichen, die man als Binärzahl eingegeben hat.

Beispiel: Aus 01001000 („H“) und 01101001 („i“) wird „Hi“.

5 Digitale Signatur auf abfotografierten Bildern und Filmen

Ein weiteres Team behandelte die Problemstellung eine Information in einem Bild zu speichern und diese dann, wenn das Bild abfotografiert wurde aus dem abfotografierten Bild wieder herauszulesen.

Unsere Aufgabe bestand darin, in einem Bild einen Text zu verstecken, welcher aus einem Code von 1en und 0en besteht. Dieses Bild wurde dann abfotografiert. Das Foto des Bildes musste schlussendlich mittels Computer den Code wiederfinden und ihn in einen Text zurückverwandeln. Dabei bekamen wir die Funktionen für die Textverwandlung in den Binärcode und wieder zurück von unseren Gruppenmitgliedern.

Das Hauptziel war es, ein Bild aus einem Film herauszuholen, dieses zu codieren und wieder an dieselbe Stelle des Filmes einzufügen. Nachdem dieser Film wiederrum abgefilmt wurde, sollte der Computer das codierte Bild finden und die Bedeutung des Textes wiedergeben.

Unser erstes Problem erschien gleich nach dem Erfassen der Problemstellung. Wir mussten das codierte Bild schlussendlich abfotografieren, weshalb viele Werte verfälscht wurden. Wir mussten es schaffen, das Bild zu fixieren beziehungsweise zu erkennen, wie es verfälscht wurde und dies verbessern, um genauere Werte zu erzielen. Dabei versteckten wir alles, was wir veränderten nicht in den Bits des Bildes, da es für unser Endziel nicht relevant war, ob man den Code sieht oder nicht. Der Grund dafür ist, dass ein Film aus mehreren Bildern besteht, sodass dem menschlichen Auge die Veränderung eines einzigen Bildes bei einer durchschnittlichen Geschwindigkeit von 30 Bildern pro Sekunde nicht auffällt.

Um das Bild zu fixieren, haben wir ein Script erstellt, welcher Rechtecke an den Ecken des Bildes setzt. Damit wollten wir erzielen, dass der Computer die inneren Eckpunkte dieser Rechtecke ausrechnet, um eine fixe Stelle im Bild auswendig zu machen. Somit konnten wir den Code auch in Abhängigkeit von diesen Eckpunkten verstecken, sodass er wieder leicht zum decodieren war.

Hier sieht man ein Bild, in welchem die Quadrate an den Ecken des Bildes gesetzt wurden. Die roten Punkte sind die inneren Eckpunkte der Rechteckt, die zum Fixieren dienen.

Diese Rechtecke sind etwas Besonderes, weil nicht nur ihre Lage definiert ist. Auch die Größe der Rechteckt passt sich dem Bild an, da die Breite eines Rechtecks 1/16 der Gesamtbreite des Bildes entspricht, genauso wie auch deren Länge 1/16 der Bildlänge ausmacht. Zusätzlich kommt noch hinzu, dass sich die Farbe der Rechtecke der Farbe des Bildes in der Umgebung der Eckpunkte anpasst. Das bedeutet, dass die Rechtecke weiß werden, wenn die Fläche, in der sie sich platzieren und noch zusätzlich ein paar unmittelbare Pixel neben dieser Fläche eher schwarz sind. Dies funktioniert dadurch, dass aus allen Pixelwerten ein Mittelwert erstellt wird, welcher anzeigt, ob die Fläche eher schwarz oder weiß ist. Danach werden die Rechtecke in der entgegengesetzten Farbe erstellt. Dafür ist es aber wichtig, dass das Bild vorher in Schwarz-Weiß konvertiert wird.

Für das Abfotografieren ist es besser, wenn die Rechtecke möglichst groß sind, da das vor weiteren Verfälschungen bewahrt, weil sich die Rechtecke dann mit einer größeren Wahrscheinlichkeit auf dem fotografierten Bild befinden, sodass die weiteren Schritte überhaupt ausgeführt werden können.

Als nächstes mussten wir ein Script erstellen, welches die inneren Eckpunkte der Rechtecke berechnet, damit wir mit diesen Werten weiterarbeiten konnten. Dafür haben wir den Unterschied zwischen den Pixelwerten an den Rändern des Bildes verwendet, indem wir den Computer von den Eckpunkten des Bildes ausgehend zwei aufeinanderfolgende Pixelwerte miteinander vergleichen ließen, sodass er den ersten und zweiten Pixelwert, den zweiten und dritten Pixelwert und so weiter verglich. Die Pixelwerte werden so miteinander verglichen, dass die Differenz zwischen den zwei Pixelwerten hergestellt wird. Wenn diese Differenz groß ist, nimmt er entweder den x-Wert heraus, wenn er den x-Wert abläuft oder den y-Wert, wenn er die y-Werte abläuft. Diese werden anschließend zum inneren Eckpunkt zusammengefasst.

Um diese Funktion robuster zu machen, verglichen wir nicht nur zwei Pixelwerte, sondern gleich vier Pixelwerte pro Reihe. Das bedeutet, dass wir wiederrum die aufeinanderfolgenden Werte verglichen, jedoch verglichen wir nun zwei Reihen, die aus 4*1 Pixel bestanden. Diese vier Werte einer Reihe wurden zusätzlich vor dem Vergleichen addiert.

Da wir dadurch unsere inneren Eckpunkte berechnen konnten und sie auch gegen Verfälschungen robuster gemacht hatte, konnten wir nach einigen praktischen Versuchen zum Testen der Funktionalität mit dem codieren beginnen.

Um den Code wiederzufinden, musste seine Position in Abhängigkeit von den Punkten bestimmt werden. Wir lösten es dadurch, dass wir einen Vektor vom inneren Eckpunkt in der linken oberen Ecke (Punkt 1) bis zum inneren Eckpunkt in der linken unteren Ecke (Punkt 4) aufstellten. Auf diesem Vektor sollte unser Code in der Form von schwarzen und weißen Quadraten eingeschrieben werden, da ein schwarzes Quadrat einer Null und ein weißes Quadrat einer Eins entspricht. Es mussten aber auch noch wichtige Dinge definiert werden, wie zum Beispiel die Größe dieser Quadrate. Diese wurden variabel gestaltet, da sie immer von der Länge des Vektors abhängig sein sollen. Zuerst wurde der Vektor jedoch in mehreren kleineren Vektoren unterteilt, indem er durch die Anzahl der Quadrate, also die Anzahl der Einsen und Nullen, dividiert wurde. Die Quadrate sollten sich dabei in der Mitte dieser Vektoren befinden und 3/4 der Länge dieser kleineren Vektoren haben. Somit entstehen Quadrate, dessen Größe von der Länge und Breite des Bildes abhängt, wobei die Reihenfolge ihrer Farben einen Code darstellt.

Wenn man das Bild codiert, gibt man sowohl den Namen des Bildes mit den Rechtecken als auch den Text, der codiert werden soll, ein. Wenn man ‚Code' im Bild verschlüsselt, sieht das Bild wie Folgendes aus.

Wenn man statt ‚Code' ‚Testcode' codiert, ändert sich die Anzahl der Buchsstaben und somit auch die Anzahl der Nullen und Einsen. Die Veränderung kann man im folgenden Bild erkennen.

Die Quadrate werden wegen der größeren Anzahl der Buchsstaben und somit auch der Nullen und Einsen kleiner, da mehrere Nullen und Einsen auf die gleiche Länge aufgeteilt wird.

Da die Quadrate sich bei einem zu langem Text überlagern, kann man nur eine bestimmte Anzahl von Buchsstaben auf eine Länge codieren. Damit man einen längeren Text codieren kann, kann man das gleiche Prinzip auf die anderen Punkte anwenden, sodass ein Coderahmen entsteht. Nachdem man das Script erstellt hat, kann man beispielsweise ‚Dies' ‚ist' ‚ein' ‚Test' codieren. Das folgende Bild zeigt an, wie es dann aussieht.

Selbst wenn man nur einen kurzen Text hat, kann die Abspeicherung des gleichen Textes in alle vier Seiten Vorteile bringen, da dieser Code dann vier Mal im Bild enthalten ist. Somit ist die Wahrscheinlichkeit, dass mindestens einer dieser Texte richtig ist, größer als bei lediglich einem Bild.

Nachdem man den Code hineinschreibt, muss man ihn wieder entziffern. Dazu haben wir ein Script erstellt, das wiederrum die Vektoren zwischen den inneren Eckpunkten abläuft, den Code so findet und schlussendlich wieder entziffert. Dies funktioniert sowohl bei dem Originalbild, als auch beim fotografierten Bild. Das Einzige, das man wissen muss, ist die Buchstabenanzahl des Textes. Der Code wird schlussendlich gleich als Text ausgegeben.

Damit das Decodieren nicht so leicht verfälscht wird, kann man einstellen, dass nicht die Größe der Quadrate abgesucht wird, sondern nur die Hälfte ihrer Größe. Somit wir bei Verschiebungen eher nur der Teil abgesucht, der den Code enthält, ohne weitere Pixelwerte mitzunehmen. Das führt wiederrum zu einem genaueren Ergebnis.

Nachdem man den Code in ein Bild eingeschrieben hat und das Script zum Herausholen der Geheimnachricht fertiggestellt hat, kann man das codierte Bild abfotografieren. Das fotografierte Bild wird dann in den Script für das Entschlüsseln der Nachricht eingegeben. Normalerweise sollte das Bild nun decodiert werden und der Text in jeder Reihe angezeigt werden. Natürlich gibt es aufgrund des Abfotografierens manchmal Verfälschungen, diese können aber normalerweise durch ein erneutes Abfotografieren aufgehoben werden. Da uns diese Verfälschungen immer wieder Probleme machten, versuchten wir überall, wo uns etwas eingefallen ist, die Funktionen zu verbessern um ein genaues Ergebnis zu bekommen. Deswegen hielten wir beispielsweise unsere Rechtecke in einem großen Format und decodierten den Code in Form von Quadern mit Quadern einer kleineren Größe. Weitere Hilfsmittel gegen Verfälschungen sind im oberen Teil bereits erklärt worden.

5.1 Digitale Signatur in Videos

Um unser Hauptziel erfüllen zu können mussten wir ein neues Skript erstellen, welches das benötigte Bild aus dem Video herausliest und dieses mit den Blöcken versieht und codiert. Wir benutzten eine MATLAB- Funktion und unsere schon verfassten Skripten und erlangten somit das neue Skript.

Nachdem diese Hürde geschafft war überlegten wir uns eine Lösung für das neue Problem- Dadurch, dass wir das Video abfilmten, erkannte der Computer den Frame nicht, welchen er jedoch gebraucht hätte, um das Video zu entschlüsseln.

Die erste Idee war simpel das graue Bild im Video zu suchen, da das Bild mit dem Code grau ist. Das neue Skript funktionierte sehr gut, allerdings nur mit dem noch nicht abgefilmten Video. Im abgefilmte Video konnte das richtige Bild durch die starke Farbveränderung nicht gefunden werden.

Das neue Konzept war statt ein graues Bild, ein Bild mit Blöcken zu suchen, doch auch dies funktionierte nicht.

Wir fügten einen neuen Block in die Mitte des Bildes ein, welcher wiederum in der einen Hälfte schwarz und in der anderen weiß war. Diesen benutzten wir dann als Erkennungszeichen für das Bild mit dem Code. Somit konnte der Computer durch das Durchsuchen aller Frames an der Stelle des Blockes und durch das Anzeigen des Miniums aller Summen, das richtige Bild anzeigen und entschlüsseln.

6 Speicherung von Daten in JPEG-Dateien

Der Rest unserer Projektgruppe beschäftigte sich folglich damit, Informationen nicht mehr nur im verlustfreien, aber mit sehr großen Dateigrößen verbundenen PNG-Format abzuspeichern, sondern auch Informationen in das sehr beliebte und wahrscheinlich auch häufigste Bilddateiformat JPEG abzuspeichern.

Das große Problem dabei war, dass JPEG die gesamte Dateistruktur verändert und dabei die Bits, die wir bis jetzt zum Speichern der Daten verwendet haben, geändert werden. So mussten wir uns eine neue Methode überlegen wie wir die Informationen im Bild speichern können. Was für uns ein großes Problem darstellte, ist aber zugleich auch der Grund wieso JPEG so beliebt ist: Durch die Veränderung der Dateistruktur und die Komprimierung wird die Größte der Datei deutlich verringert.

6.1 JPEG-Komprimierung

Um eine Lösung zu finden und um einen Code zu schreiben, mit dem wir Informationen in JPEG Bilder können, mussten wir zunächst verstehen wie die JPEG-Kompression abläuft. Dafür halfen uns einerseits die Recherche im Internet und andererseits auch die Erklärung unseres Betreuers. Außerdem stellte er uns auch einen MATLAB Code bereit, der die Kompression von JPEG Bildern simuliert.

Bei der JPEG-Kompression wird das Bild, welches meistens als RGB-Bild vorliegt in ein Bild mit YCbCr-Farbraum umgerechnet. Außerdem wird das Bild in acht mal acht große Blöcke unterteilt. Diese werden dann einer diskreten Kosinustransformation (=DCT) unterzogen. Die eigentliche Dateigrößenreduktion passiert durch die Division der DCT-Koeffizienten durch die Quantisierungsmatrix.

$$Q = \begin{bmatrix} 10 & 15 & 25 & 37 & 51 & 66 & 82 & 100 \\ 15 & 19 & 28 & 39 & 52 & 67 & 83 & 101 \\ 25 & 28 & 35 & 45 & 58 & 72 & 88 & 105 \\ 37 & 39 & 45 & 54 & 66 & 79 & 94 & 111 \\ 51 & 52 & 58 & 66 & 76 & 89 & 103 & 119 \\ 66 & 67 & 72 & 79 & 89 & 101 & 114 & 130 \\ 82 & 83 & 88 & 94 & 103 & 114 & 127 & 142 \\ 100 & 101 & 105 & 111 & 119 & 130 & 142 & 156 \end{bmatrix}$$

Quantisierugnsmatrix

Danach werden diese auf Ganzzahlen gerundet. Wenn die Datei geöffnet wird, werden dann die Koeffizienten wieder multipliziert und die Inverse Kosinustransformation wird mit den Daten durchgeführt.

6.2 Informationsspeicherung in JPEG Bildern

Unsere Idee war es die Daten nicht mehr in die Bits einzelner Pixel sondern in die Bits der DCT-Koeffizienten abzuspeichern. Dies konnten wir erreichen, indem wir mit dem Bild im MATLAB zunächst eine Kosinustransformation durchführten, danach die Informationen in die Koeffizienten speicherten und dann wieder die Inverse-Kosinustranformation durchführten bevor wir das Bild im JPEG-Format speicherten.

Diese Methode ermöglichte es uns zwar Informationen in JPEG-Dateien zu speichern, allerdings gingen fast alle davon bei der Speicherung als JPEG-Dateien verloren. Grund hierfür ist die angesprochene Rundung der einzelnen DCT-Koeffizienten. Zusätzlich musste auch die Pixelanzahl der Breite und Höhe genau durch acht teilbar sein, sowie das Bild in schwarzweiß vorliegen.

6.3 Schutzbits

Wir grübelten lange Zeit über dieses mathematische Problem, bis wir auf die Lösung gekommen sind: Wir implantierten Schutzbits in den Code, die besagen dass der Bit im nächstniedrigerem Bit-Level als der Bit in dem wir unsere Information speicherten, auf 1 gesetzt wird und alle weiteren Bits in niedrigeren Bit-Level, auf 0 gesetzt werden. So konnten wir die Gefahr minimieren, dass unser Informationsbit durch die Rundung zerstört wird, dieses Vorgehen wird genauer im Abschnit 6.2.2 beschrieben.

Mit diesem, durch sogenannte Schutzbits erweiterten, Code erzielten wir sehr passable Ergebnisse und die Speicherung von Informationen in JPEG-Bilder mit einer Kompressionsrate von 50 erwies sich als durchaus a machbar.

Wirklich zuverlässig war unsere Methode bei einer Kompressionsrate von mindestens 60.

6.3.1 Beweis der Schutzbits

Problemstellung

Wir haben eine Zahl λ, von der wir wissen, dass sie durch JPEG-Komprimierung um einen Wert ≤N geändert wird. Die veränderte Zahl x̃ liegt also in [λ-N, λ+N]

Wir möchten ein Bit b in einem möglichst niedrigen Bitlevel n von λ speichern, um λ möglichst wenig zu verändern. Zugleich sollte eine Zerstörung unseres Bits (also eine Änderung) die Zahl λ möglichst stark verändern, um so den Erhalt des Bits b bis zu einer möglichst großen Änderung N zu garantieren.

Da bei der Speicherung von b in das Bitlevel n auch die Information in allen niedrigeren Bitlevel zerstört wird, können wir alle Bitlevel von n bis 1 beliebig verändern um die Information zu speichern.

Fragestellung

Gegeben eine Zahl λ, ein Bitlevel n, wie können wir die Information eines bIts b in den Bitlevel n bis 1 von λ Speichern so dass eine Zerstörung von 6 eine möglichst große Änderung von λ erzwingt.

Erklärung und Beweis

Als erstes betrachteten wir nur die Möglichkeit die Information b im Bitlevel n zu speichern und alle niedrigeren Bitlevel beliebig zu verändern (Schutzbits).

Angenommen wir speichern b im Bitlevel b und b wird verändert. Wie stark ändert sich der Wert der Zahl mindestens? Eine Änderung von b bedeutet eine Änderung der Zahl λ um 2^{n-1}.

Wie kann diese Änderung bestmöglich begrenzt werden in dem man die Bits ≥n+1 und n-1 bis 1 ändert. Eine Änderung des Bit ≥n+1 kann nur das Vorzeichen, nicht aber der Betrag der bisherigen Änderung beeinflussen. Um

also die Änderung zu verhindern muss die Zahl 2^{n-1} durch Änderung des Bits n-1 bis 1 möglichst gut auszugeglichen werden. Je nach vorheriger Setzung des Bits n-1 bis 1 kann dies durch Addition oder Subtraktion von 2^{n-1}, $k \in \{1,\ldots n-1\}$ geschehen. Durch eine solche Operation kann die Änderung bestenfalls auf ± 1 reduziert werden. Wenn nun das Bit n-1 auf 1 und alle niedrigeren auf 0 gesetzt werden, kann durch beliebige Änderung dieser Bits die Zahl λ maximal um 2^{n-2} oder $2^{n-2} – 1$ geändert werden. Die minimale Änderung die durch Zerstörung des Bits b bei dem Bitmuster b=1 0…0 verursacht wird, ist als also

$2^{n-1} – 2^{n-2} = 2^{n-2} (2-1)=2^{n-2}$

Jede ander Konstellation der Bits n-1 bis 1 erlaubt eine Änderung von mindestens 2^{n-2} also eine Zerstörung des Bits b bei gleichzeitiger Änderung der Zahl λ um weniger als 2^{n-2}.

Die Methode b 1 0…0 ist also optimal, falls die Information im Bitlevel n gespeichert wird.

Falls b in einem Bitlevel $k \in \{n-1,\ldots,1\}$ gespeichert wird, kann b immer durch Änderungen von um λ um 2^{k-1} zerstört werden- Da $2^{k-1} \leq 2$ n-2 gilt, wäre eine solche Methode nicht besser.

Wir fassen zusammen: Die Speicherung aus Bits b im Level m mit der Methode b 1 0…0 ist die allerbeste um den Bit b bei möglich großer Änderung von λ garantiert zu erhalten. Eine Änderung von b ist nur möglich wenn λ um mindestens 2^{n-2} geändert wird.

6.4 Fehlerbehebungen und Optimierung

Nach diesen Erfolgen erweiterten wir unseren Code, sodass wir auch Informationen in Farbbildern und in Bildern, deren Breite und Höhe nicht durch acht teilbar ist, verstecken konnten.

Bei ausführlichen Tests erkannten wir aber, dass es auch mit diesem jetzigen Code noch immer zu Fehler kommt. Als Fehlerquelle erkannten wir, dass teilweise der Farbwert über den erlaubten Wert von 255 steigt und dass das Programm diese Werte dann verändert. Auch für dieses Problem fanden wir

eine Lösung und zwar ergänzten wir den Code so, dass diese zu hohen Wert schon bevor sie beschrieben werden erkannt werden und gesenkt werden. Dies ist zwar theoretisch mit einer Verschlechterung der Bildqualität verbunden, allerdings war mit dem freien Auge der leicht eingeschränkte Farbraum nicht erkennbar.

Das Ergebnis war ein äußerst robuster Code, mit dem es möglich war Binärcodes in Bildern, fast ohne merkbarer optischer Veränderung zu verstecken. Es trat nur noch in seltenen Fällen auf das ein Buchstabe falsch gelesen wurde. Dieser Fehler, die nur bei gewissen Bildern, bei einem oder zwei Buchstaben auftraten haben ihren Ursprung in der Konvertierung der Bilder von RGB in YCbCr.

Originales Bild

JPEG Bild mit einer Geheimbotschaft, versteckt im 7. Bitlevel. Die Dateigröße wurde um mehr als die Hälfte

7 Hochladen versteckter Nachrichten auf Youtube

Da wir am Ende unseres Projektes noch Zeit zu verfügen hatten, entschieden wir uns auch noch ein optionales Ziel, ein Video mit einer Geheimbotschaft auf Youtube hoch zuladen, diesen wieder downzuloaden und dann die Botschaft auszulesen, in Angriff zu nehmen.

Grundlage hierfür waren der Code mit dem wir Informationen in JPEG-Bildern speichern konnten und der Code mit dem wir Frames in Videos speichern konnten. Wir kombinierten diese beiden dann so, dass zuerst ein Frame aus dem Video gespeichert wird und dann in diesen, nach der gleichen Methode wie in den JPEG Bildern eine Geheimbotschaft eingespeichert wird und. Dieser Frame mit dem Code wird dann wieder an die gleiche Stelle in das Video hineingestellt und das Video wird ausgegeben.

Danach luden wir das Video mit der Geheimbotschaft auf Youtube hoch und luden es es dann wieder herunter. Danach spielten wir es in einen Code ein, der einen von uns bestimmten Frame aus dem Video herausspeichert und aus diesen dann die versteckte Nachricht ausliest.

Wir hätten uns nicht erwartet, dass es überhaupt möglich ist versteckte Botschaften in einem Youtube Video hochzuladen und vor allem diesen dann wieder erfolgreich auszulesen. Auch ging die Erstellung des Codes sehr schnell von sich, woran man merkte, dass wir im Laufe dieser Woche einiges an Erfahrung hinzugewonnen haben und viel routiniert im Gebiet der Programmierung, was vorher für uns ein fast unbekanntes Metier war.

8 Resüme

Am Anfang dieser Woche stürzten wir uns in ein für uns vollkommen unbekanntes Abenteuer. Einerseits waren wir nicht mit dieser Arbeitsweise vertraut, die sich doch beträchtlich von Schule unterscheidet, anderseits hatten wir auch nicht wirklich Ahnung von Steganographie – Wir hielten es für einen Teil der Kryptografie und auch die umfassenden Möglichkeiten, die damit verbunden sind waren uns nicht bekannt.

Nun blicken wir auf eine sehr erfolgreiche Woche zurück und auch auch auf unser Flipchart, welches wir am Anfang unseres Projektes als Leitfaden erstellt haben. Wir haben nahezu alle Aufgaben erfüllt, obwohl es am Anfang nicht einmal klar war, ob wir die Speicherung in JPEG Bildern bewältigen können und ob Digitale Signatur in Filmen, die abgefilmt werden, sowie das Hochladen von geheimen Botschaften überhaupt funktioniert.

Zusammenfassend können wir sagen, dass diese Woche sicher in allen von uns einen bleibenden Eindruck hinterlassen hat. Wir haben nicht nur eine große Menge an neuem Wissen hinzugewonnen, wir haben was vor allem wichtig ist immens viel an Erfahrungen in wissenschaftlichen Arbeitsweisen dazugewonnen.

Woche der Modellierung mit Mathematik

Voraussage des Ergebnisses einer Cerclage Operation

Projekt: Medizin

Gruppenmitglieder:

Dreisiebner Lisa, Jost Etta, Maierhofer Thomas

Polz Fabian, Resch Nadine, Rusu Elena

Betreuer:

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

Jufa Deutschlandsberg 8. – 14. Februar 2014

Inhalt

1 Veränderung des Auges nach und während einer Cerclage Operation

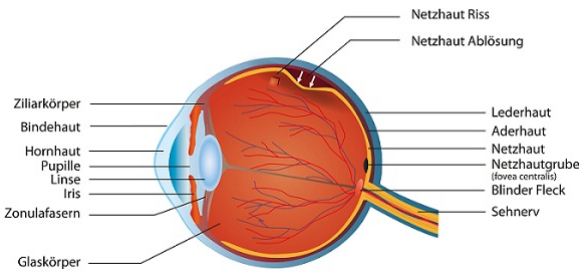
1.1 Definition des Problems

Netzhautriss und Netzhautablösung werden von Augenärzten durch eine Cerclage Operation behandelt, wobei ein Gummiband am Äquator des Augapfels angewendet wird, um das Auge zusammenzudrücken. Im präoperativen Zustand (d.h. vor der Operation) ist der Augapfel ungefähr kugelförmig mit einem normalen Augeninnendruck. Im intraoperativen Zustand (d.h. während und sofort nach der Operation) ist der Augeninnendruck erhöht, das Augeninnenvolumen bleibt gleich und der Augapfel ist deformiert. Im postoperativen Zustand (d.h. spätestens eine Woche nach der Operation) wird das Augeninnenvolumen durch Selbstregelung reduziert. Infolgedessen wird der Augeninnendruck reduziert, und die Deformation des Augapfels wird entsprechend entspannt. Wegen dieser Entspannung während der postoperativen Phase, muss die Cerclage so angewendet werden, dass der gezielte Zustand im Fließgleichgewicht erreicht wird. Das Ziel dieses Projektes ist, die Zustände des Augapfels während und nach der Cerclage Operation vorauszusagen, um die Operationsplanung zu erleichtern.

1.2 Grundlagen

Eine Cerclage Operation ist ein Operationsverfahren in der Augenheilkunde, das bei Netzhautablösungen angewandt wird.

1.2.1 Netzhautablösung



Als Netzhautablösung bezeichnet man die teilhafte oder komplette Ablösung der inneren Anteile der Netzhaut (Neuroretina) von der unteren Versorgungsschicht, dem retinalen Pigmentepithel, des Auges. In einem gesunden Auge liegt die Neuroretina direkt auf

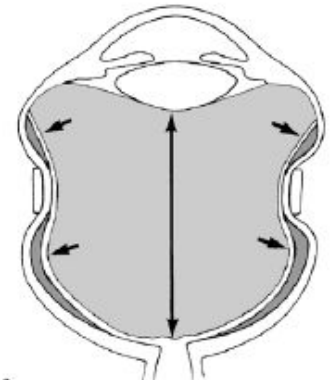
dem Pigmentepithel auf und ist nur im Teil der Netzhaut direkt mit ihm fest verbunden. Eine Netzhautablösung kann verschiedene Entstehungsursachen haben, rissbedingte, zugbedingte, durch Einlagerungen von Flüssigkeiten bedingte, tumorbedingte oder Druckwellen bedingte. In jedem Fall dieser Erkranken löst sich die Neuroretina vom retinalen Pigmentepithel und Flüssigkeit strömt zwischen diese beiden Schichten und eine anfangs örtliche Netzhautablösung kann sich durch weiteres Einströmen von Flüssigkeiten innerhalb weniger Stunden auf die gesamte Fläche der Netzhaut ausbreiten.

Die betroffene Person kann bei einer anfänglichen Netzhautablösung kleine Lichtblitze oder dunkle Flecken im Gesichtsfeld erkennen und später bei einer fortgeschrittenen oder bei Blutungen im inneren des Auges schwarze Balken, die sich wie Vorhänge um das Gesichtsfeld legen. Die Netzhautablösung kann jedoch auch ohne Symptome ablaufen, bis der gelbe Fleck, die Makula, davon betroffen ist, wo sich bekanntlich die größte Dichte an Sehzellen befindet.

Eine Netzhautablösung stellt einen medizinischen Notfall dar, der innerhalb weniger Tage oder gar Stunden behandelt werden muss, da ansonsten die Sehkraft, da die Netzhaut mit all den auf ihr liegenden Photorezeptoren bei einer Ablösung von dem Pigmentepithel nicht mehr versorgt werden kann, stark verringert wird beziehungsweise vollständig schwindet.

Bei einer Netzhautablösung gibt es mehrere Behandlungsmethoden, wie eine Laserbehandlung, eine pneumatische Retinopexie oder einen kryo-chirurgischen Eingriff. Eine heutzutage eher veraltete aber dennoch wirksame und oft angewandte Methode ist die Cerclage Operation.

1.2.2 Cerclage Operation



Bei einer Cerclage Operation wird ein Silikonband um den Äquator des Augapfels gelegt und mit einer dementsprechenden Spannung befestigt. Dadurch beult sich das Auge um das Silikonband ein, der innere Druck

steigt und die Netzhaut wird wieder an das Pigmentepithel gedrückt. Nach ungefähr einer Woche hat die Selbstregulierung stattgefunden und die Netzhaut hat sich von selbst wieder angelegt. Bei 80 Prozent der Fälle glückt eine Cerclage Operation und das Auge muss nicht weiter behandelt werden. Als Nebenwirkung wird die Verformung des Auges angegeben, da sich die Form des Auges mit dem Silikonband verändert und so das Sehvermögen verringert werden kann, was sich als Kurzsichtigkeit zeigt.

1.3 Darstellung des Problems

Anfangs gilt es, die Sachverhalte darzustellen. Man geht davon aus, dass sich das Auge in einem präoperativen Zustand vor der Operation befindet, in einem intraoperativen Zustand während und unmittelbar nach der Operation und dem postoperativen Zustand ungefähr eine Woche nach der Operation.

Ziel ist es das Verhalten, die Form, das Volumen, den Druck und die Größe des Auges im intraoperativen und im postoperativen Zustand vorherzusagen und zu berechnen.

1.3.1 Präoperativ

Das Auge hat das Volumen und den Innendruck eines normalen, gesunden Auges. Werte wie Größe, Radius und Druckunterschied innen und außen sind gegeben. Alle restlichen Werte können ohne weitere Probleme berechnet werden.

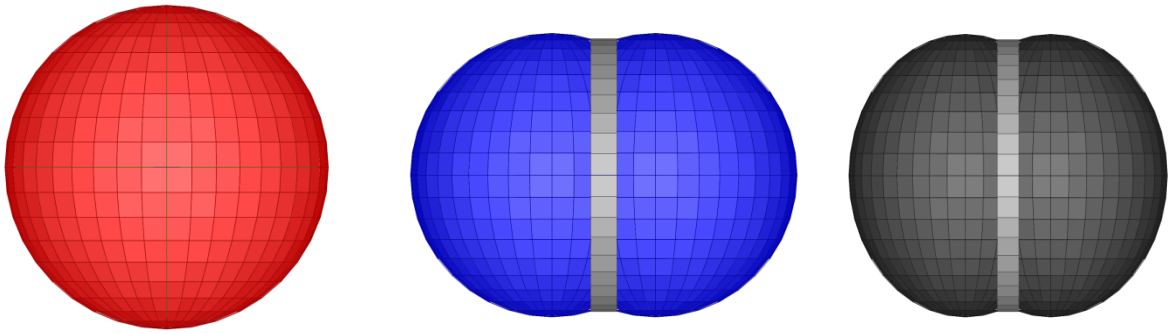
1.3.2 Intraoperativ

Das Silikonband übt Druck auf den Augapfel aus. Der Druck von außen verstärkt sich im Bereich um das Band und der Innendruck des Auges wird auch gleichmäßig erhöht. Der Innendruck muss selbst in einem gesunden Auge größer sein als der Außendruck. Durch die Krafteinwirkung des Gummibands verformt sich das Auge und wird sozusagen eingebeult und in die Länge gedrückt. Das Volumen des Auges im intraoperativen Zustand ist gleich dem Volumen des präoperativen Zustands, jedoch ist der Druck im intraoperativen Zustand wesentlich höher als der Druck im präoperativen Zustand.

1.3.3 Postoperativ

Eine Woche nach der Operation hat sich das Auge von selbst reguliert. Das Silikonband befindet sich immer noch am Auge, der Außen- und der Innendruck sind immer noch erhöht, jedoch gibt das Silikonband in seiner Spannung nach und das Auge strebt danach seine ursprüngliche, annähernde Kugelform zu erreichen. Im postoperativen Zustand ist das Volumen des Auges verringert und der Innendruck des Auges ist annähernd der des Auges im präoperativen Zustand.

1.3.4 Zusammenfassung



Auge präoperativ	Auge intraoperativ	Auge postoperativ
V_0	$V_0 = V_1$	$V_0 > V_2$
p_0	$p_0 < p_1$	$p_0 \approx p_2$

- V0 = Volumen präoperativ
- p0 = Innendruck präoperativ
- V1 = Volumen intraoperativ
- p1 = Innendruck intraoperativ
- V2 = Volumen postoperativ
- p2 = Innendruck postoperativ

1.4 Weg zur Problemlösung

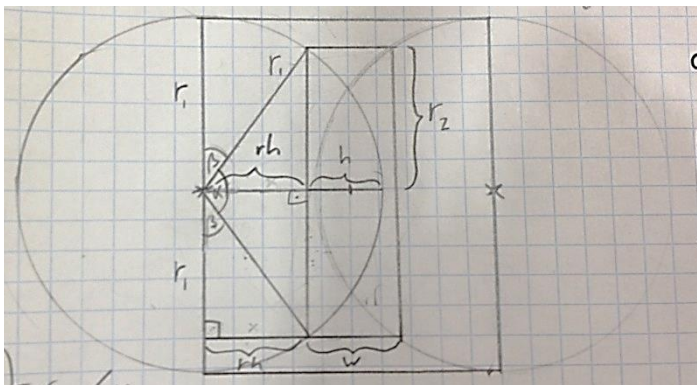
Durch anfängliche Internetrecherche ergaben sich diverse Formeln für Druck, Spannung, Elastizität etc. Es musste klargestellt werden, welche Umstände herrschen und von welchen Faktoren das Auge beeinflusst wird. Es wurde beschlossen mit zwei verschiedenen Lösungswegen ans Ziel zu gelangen. Die wichtigsten Werte, die es herauszufinden galt, waren die Radien und die Volumina der verschiedenen Zustände

des Auges. Es musste berücksichtigt werden, dass die Veränderung des Auges nicht linear abläuft. Viele weitere Faktoren mussten in Betracht gezogen werden, wie zum Beispiel die Elastizität des Auges und die verschiedenen Druckzustände durch das Silikonband verursacht. Einige der wichtigsten Formeln in diesem System waren das Laplace-Gesetz sowie das Hookesche Gesetz. Viele Formeln wurden neu kombiniert und zusammengesetzt.

1.4.1 Intraoperativ

Kreative Maßnahmen zu einer Verbesserung des Vorstellungsvermögens

1.4.1.1 Lösungsweg 1



Für den intraoperativen Zustand des Auges wurde von einem aus einem Zylinder und zwei Kugeln bestehenden Modell ausgegangen. Der Zylinder sollte das vom Silikonband eingedrückte Auge darstellen und die beiden Kugeln die in die Länge gezogenen Teile des Auges. Mit Hilfe dieses Modelles gelang es, mit einfacher Trigonometrie auf alle wichtigen Längen, Winkel und Radien zu schließen. So konnte der erste Teil eines Gleichungssystems

aufgestellt werden. Die Unbekannten in dieser Volumsformel waren der Radius der zwei Kugeln (r_1) des intraoperativen Augmodells und der Radius des Auges, wo das Silikonband aufliegt (r_2).

$$\frac{4}{3}r^3\pi = 2 * \left(\frac{4}{3}{r_1}^3\pi - \frac{h^2\pi}{3}(3r_1 - h)\right) + w{r_2}^2$$

- r = Radius des Auges im präoperativen Zustand
- r_1 = Radius der Kugel des intraoperativen Zustands
- r_2 = Radius des Zylinders
- h = Höhe des Segments
- w= Breite des Silikonbandes

Davon konnte ausgegangen werden, da das Volumen des Auges im präoperativen Zustand gleich dem des im intraoperativen Zustand ist. Da für ein Gleichungssystem mehrere Gleichungen benötigt werden, um die Unbekannten auszurechnen, mussten noch weitere Gleichungen gefunden werden. So ergaben sich die Formeln für die Spannungen der Oberfläche des Auges im intraoperativen Zustand.

$$T_1 = \frac{(p\alpha - p\beta + \lambda) * r_1}{2}$$

$$T_2 = ((p\alpha + \lambda) - (p\beta + p_B)$$

- T_1 = Spannung der Oberfläche der Kugel
- T_2 = Spannung der Oberfläche des Zylinders
- pα = Druck im inneren des Auges
- pβ = Druck von außen
- p_B = Druck des Silikonbandes
- λ = durch den Druck des Silikonbandes erhöhte Druck im Inneren

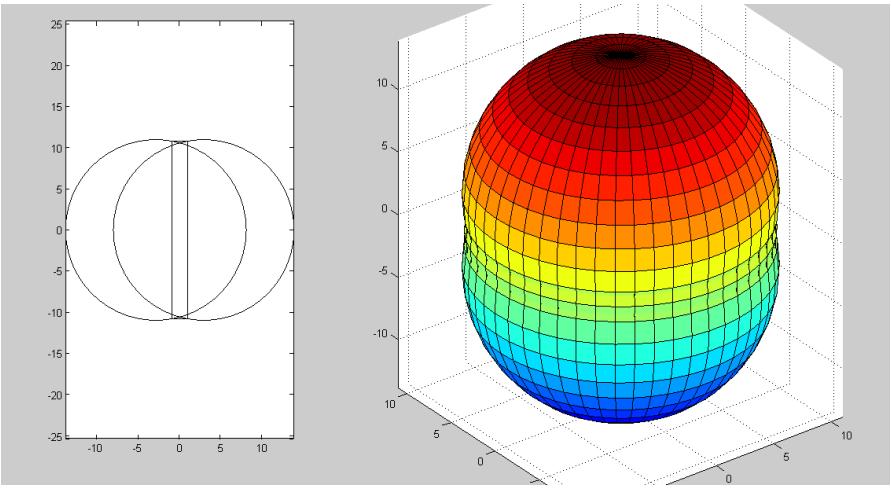
Davon konnte ausgegangen werden, da angenommen wurde, dass der Unterschied der Spannung in den beiden unterschiedlichen Körpern annähernd gleich ist. Eine weitere Formel für die Oberflächenspannung jedoch in Bezug auf die Fläche des Modells ist unsere vierte Gleichung unseres Gleichungssystems.

$$T_{1,2} = T_0 + Ed * \left(\frac{\Delta A}{A_0}\right)$$

- $T_{1,2}$ = Spannung der gesamten Oberfläche des Auges im intraoperativen Zustand.
- T_0 = Spannung des Auges im präoperativen Zustand
- E = Elastizität des Auges
- d = Dicke der Lederhaut
- ΔA = Differenz der Oberflächen des präoperativen und des intraoperativen Auges
- A_0 = Oberfläche des präoperativen Auges.

Durch Eingabe in das Programm „MATLAB“ konnten die Unbekannten (r_1, r_2, T_1, T_2, λ) berechnet werden.

r_1	Radius der Kugeln des Modells	10,4652 mm
r_2	Radius des Zylinders des Modells	10,4173 mm
T_1	Spannung der Oberfläche der Kugel	120,3347 mmHg
T_2	Spannung der Oberfläche des Zylinders	120,3347 mmHg
λ	Der durch das Silikonband erhöhte Innendruck	22,9972 mmHg



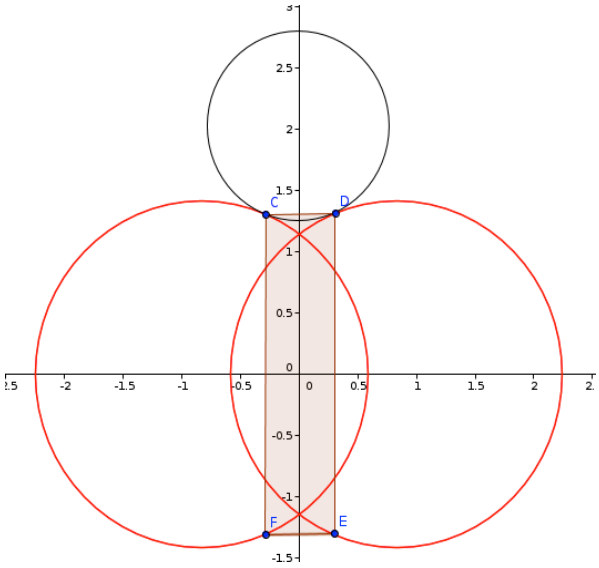
Darstellung des Auges im intraoperativen Zustand im Kugel-Zylinder-Kugel - Modell

1.4.1.2 Lösungsweg 2

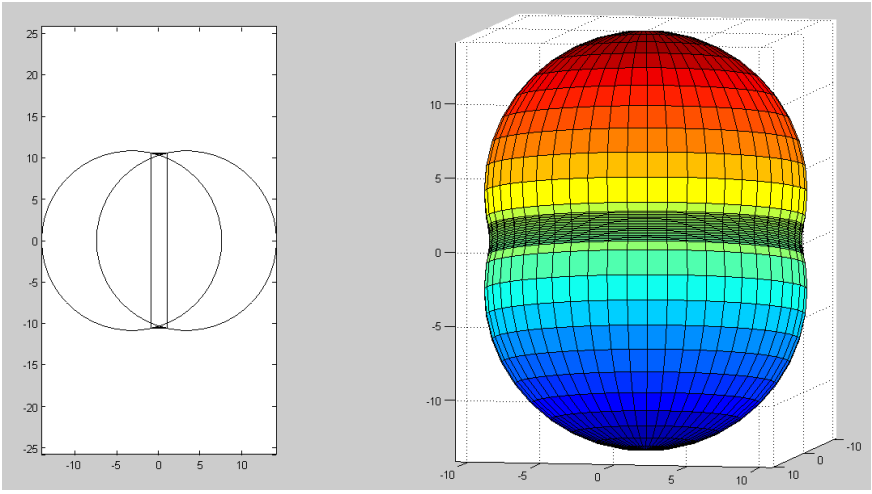
Ausgehend vom ersten Lösungsweg, wurde festgestellt, dass sich die Werte ändern, wenn man die Geometrie anpasst. Die Formeln der Spannungen wurden für das Gleichungssystem übernommen, jedoch wurden die Übergänge, die zwischen dem Zylinder und der Kugel in dem ersten Modell liegen, geglättet, um ein realistisches Modell zu erhalten. Anstatt des Zylinders wurde das Volumen unter einem Kreis mittels Integral bestimmt. Auch die Oberfläche des Körpers, die sich in weiterer Folge auf Spannung und Druck auswirkt, wurde an dieses Modell angepasst. Somit besteht das neue Modell aus zwei gleich großen Kugeln (k_1) mit Radius r_1 und einer kleinen Kugel (k_2) mit Radius r_2, die die k_1 in den Punkten C und D schneidet, wobei die Strecke CD der Bandbreite entspricht (siehe Grafik). Grafisch dargestellt wurden die Überlegungen, um zu den neuen Formeln zu kommen, mit GeoGebra.

$$V = 2\pi \int_0^{\frac{w}{2}} (k_1(x))^2 \, dx + 2[\frac{4}{3}r^3\pi - \frac{\pi h^2}{3}(3r - h)]$$

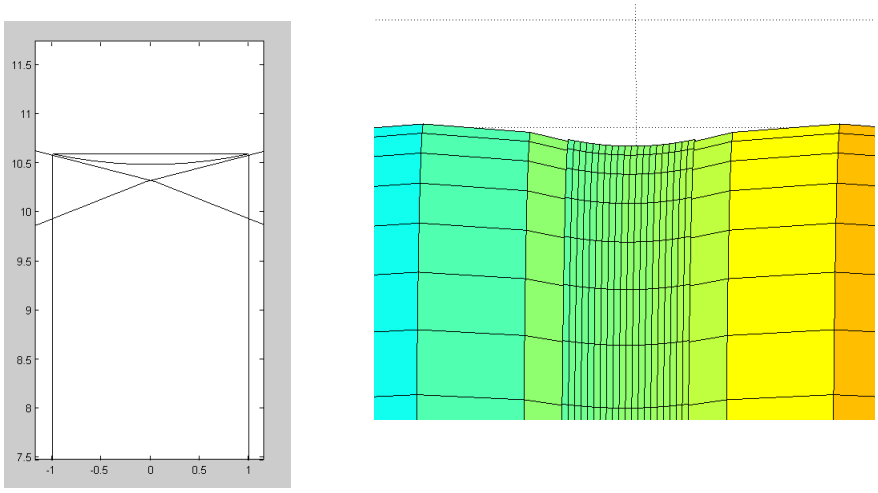
Auf Grund von Schwierigkeiten während der Rechenoperation mit dem Programm MATLAB, wurden der Radius des Bandes und dessen tiefste Eindrucksstelle gemittelt und das Volumen wiederum mit einem Zylinder ermittelt, wobei die Werte sehr nahe denen des mittels Integral errechneten Modell lagen.



- k_1 = Teil des Auges im intraoperativen Zustand
- k_2 = Kreis, der den Übergang zwischen dem Auge und dem Silikonband beschreibt
- Strecke CD = Breite des Silikonbandes



Darstellung des Auges im intraoperativen Zustand im 3Kugel - Modell



1.4.2 Postoperativ

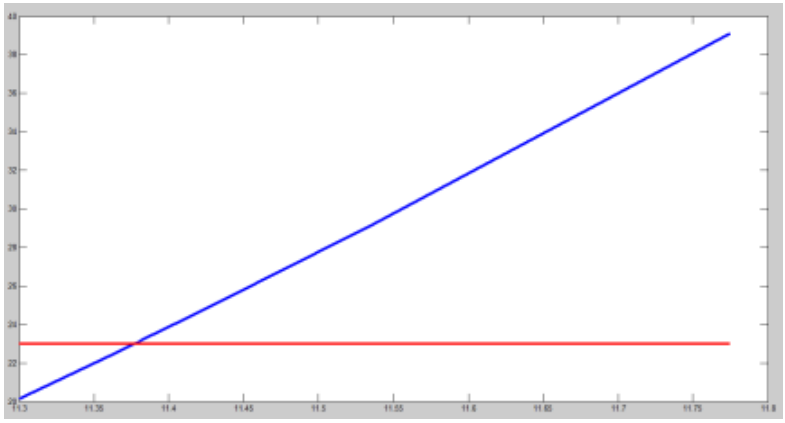
1.4.2.1 Lösungsweg

Es musste eine Funktion erstellt werden in der ein Zusammenhang zwischen den verschiedenen Radien und den verschiedenen Drucken in den unterschiedlichen Phasen berechnet wird. Die verwendete Funktion wurde aus der Logarithmus-Funktion $V(p) = \frac{ln*p}{K} + V_0$ gebildet. So ergab sich die Exponentialfunktion:

$$p(r) = p_{prä} * e^{kappa*(V_2 - V_0)}$$

- p = Druckunterschied
- $p_{prä}$ = Druckunterschied des Auges im präoperativen Zustand
- kappa= Steifigkeit eines gesunden Auges
- V_2 = Volumen des Auges im postoperativen Zustand
- V_0 = Volumen des Auges im präoperativen Zustand

Mit dieser Exponentialfunktion wurde durch das Bisektionsverfahren in „MATLAB“ der Radius des Auges im postoperativen Zustand berechnet, zu dem pα+λ gleich dem Druck des präoperativen Zustand des Auges entspricht.



Graphische Darstellung des Bisektionsverfahrens zur Ermittlung des Drucks im postoerativen Zustand

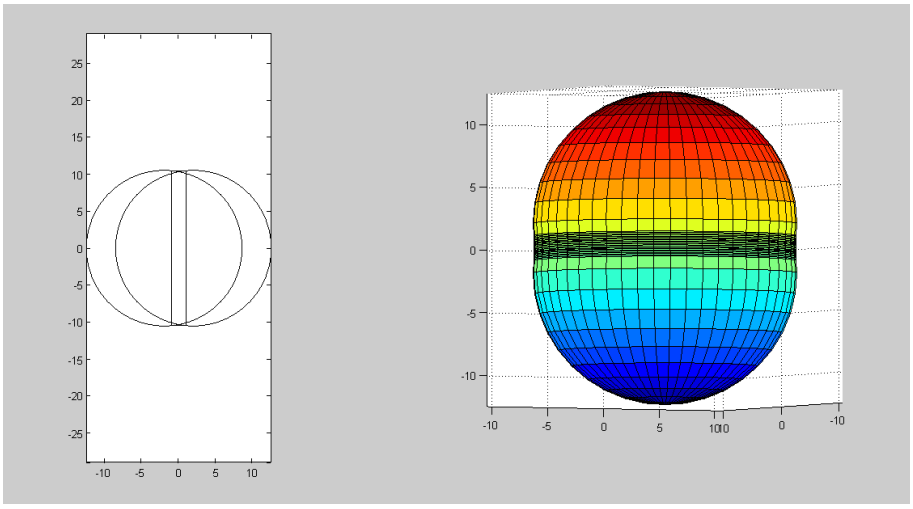
Prinzipiell verkleinert sich das Volumen, da sich das Auge durch Selbstregulierung dem Band anpasst. Der Augeninnendruck ist im postoperativen Zustand derselbe wie im präoperativen Zustand, jedoch ist der Innendruck im postoperativen Zustand der wegen des verringerten Volumens niedrigere Augeninnendruck addiert mit dem durch das Silikonband erhöhten Innendruck ($p_2\alpha+\lambda=p_0$).

Für den postoperativen Zustand des Auges ergaben sich folgende Werte:

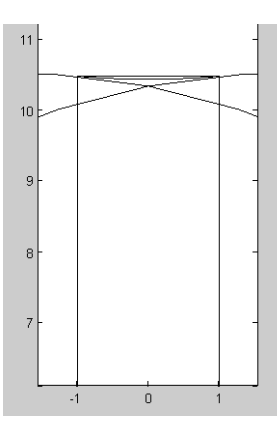
r_3	Radius des Auges im postoperativen Zustand	11,3772 mm
V_2	Volumen des Auges im postoperativen Zustand	6168,7161 mm³

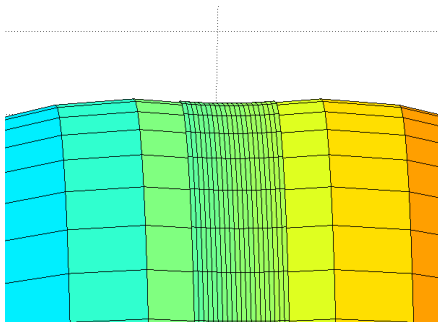
1.4.2.2 Optimierung

Die resultierenden Werte dieses Konzepts werden infolge im postoperativen Stadium weitergeführt und mit derselben Methode, wie im anderen Modell, gelöst.



Darstellung des Auges im postoperativen Zustand im 3Kugel - Modell



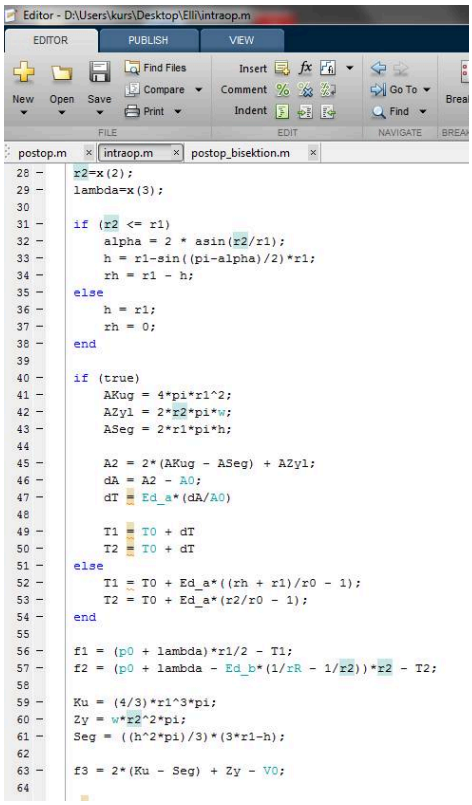


1.5 Odyssee

Anfangs gingen wir davon aus, dass man die verschiedenen Zustands- und Volumsänderungen des Auges durch die Zustandsgleichung eines idealen Gases berechnen könnte. Dies erwies sich nach ersten Überlegungen als falsch, da das Auge elastische Außenhäute besitzt und sich bei veränderten Druck- und Volumsverhältnissen in Form und Größe verändert.

Als wir versuchten den sich verändernden Radius des Silikonbandes zu berechnen, stießen wir auf ein erneutes Hindernis. Das Silikonband, das beim Übergang vom intraoperativen zum postoperativen Zustand an Elastizität zunimmt, ist einer der wenigen Feststoffe, der sich nicht nach dem Hookeschen Gesetz dehnt. Beim Dehnen des Silikonbandes verändert es sich nicht nur in Länge, sondern auch in Breite. Um die Veränderung des Querschnitts zu berechnen hätten wir zu viele unbekannte Komponenten in die Rechnungen miteinbezogen.

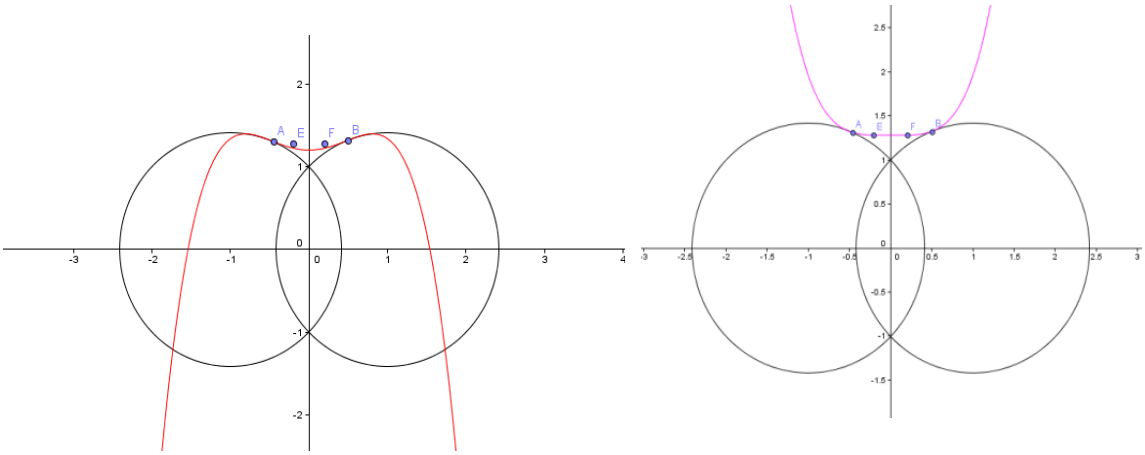
Bei beiden ersten Lösungswegen vereinfachten wir das intraoperative Modell des Auges auf einfache geometrische Formen und missachteten gewisse Krümmung, derer wir nicht im Stande waren auszurechnen.



Das uns die meisten Schwierigkeiten bereitende Problem wurde von dem ebenso essentiellen wie Nerv raubenden Mathematik Programm „MATLAB“ verursacht. Da wir anfangs der Woche noch Laien im Umgang damit waren, kamen wir nicht umhin etliche Fehlermeldungen zu erhalten. Selbst nach einigen Tagen intensiver Einschulung für Eingabecodes und Tipps und Tricks das Programm zu meistern, gelang uns die die Eingabe noch immer nicht problemlos. Die Tatsache, dass es sichtbare und versteckte Fehler gibt, womit uns Stephen am Anfang der Woche warnte, lernten wir schnell. Auch wenn die Eingabe einmal geglückt sein sollte und das

Programm keine Fehlermeldungen zeigte, so ergaben dennoch etliche Male die Ergebnisse für den logisch denkenden Verstand keinen Sinn und man musste erneut alle Codes nach einem Gedanken- oder Formelfehler durchsuchen. Letztendlich war es uns dank Stephens Geduld und Hilfe jedoch möglich, exakte und hilfreiche Ergebnisse zu erhalten.

Am Weg zur richtigen geometrischen Darstellung dieser Lösungsvariante gingen wir zuerst von verschiedenen Funktionen aus, um die Krümmung darzustellen, die den Übergang von dem mittleren Teilkörper zu den äußeren Kugeln ausglätten sollte. Diese Funktionen stellten wir ebenfalls mit GeoGebra da, damit wir eine bessere Vorstellung hatten die Formeln zu formulieren. Wir kamen dann zum Entschluss, einen Kreis anstatt Polynome verschiedenen Grades zu benutzen, wobei der Rand des Bandes nicht auf den Schnittpunkten der Kreise lag, wie bei dem aktuellen Modell sondern innerhalb. Dieses Modell stellte sich jedoch als nicht zielführend da und wir bestimmten die Schnittpunkte der Kreise so, dass sie genau am Rand des Bandes lagen.



- Roter Graph = biquadratische Gleichung
- Rosafarbener Graph = Polynomfunktion 4. Grades

1.6 Resultate

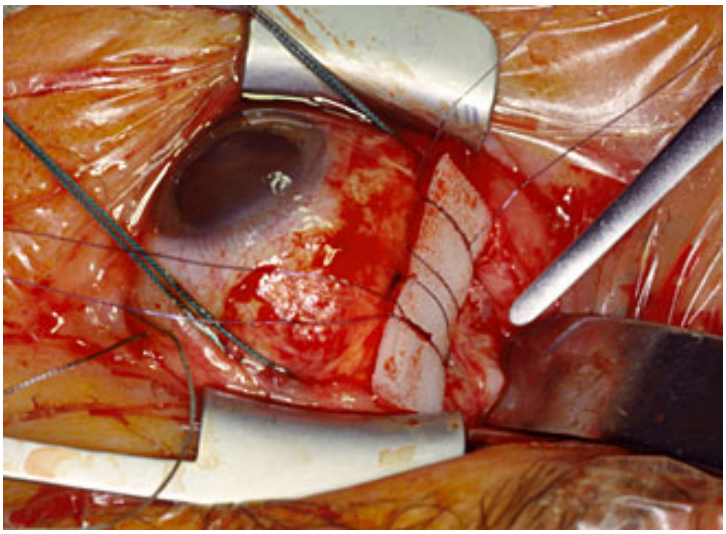
Das Resultat dieses Projektes stellt - ausgehend von einem durchschnittlichen Auge mit durchschnittlichen Maßen und Werten – eine Funktion dar, deren Werte in Relation zu den Werten eines jeden beliebigen Auges stehen. So kann man – vorausgesetzt man ist in der Lage MATLAB zu bedienen – Form, Volumen, Größe,

Spannung, Oberfläche und Druck eines jeden Auges in den verschiedenen Stadien der Cerclage Operation berechnen.

1.7 Anwendung

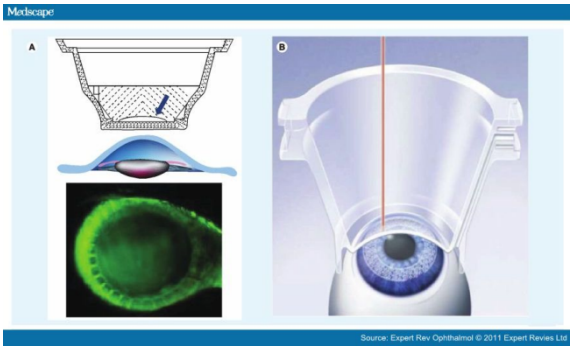
Durch diese Berechnungen und Formeln kann man genau vorhersagen, wie sich das Auge in Form und Größe verändern wird. Das Programm, in welches die Formeln und Funktionen eingetragen wurden, kann weiterhin verwendet werden, indem man die Werte und Maße durch die eines jeden beliebigen Auges vertauscht. So kann bei jedem beliebigen Menschen vorhergesagt werden, wie sich sein Auge verändern wird.

Augenärzte können davon profitieren, da man nun genauer die operativen Phasen vorherbestimmen kann. Es können auch verschiedene Druckzustände angenommen werden und an das Auge angepasst werden. Man erhält Informationen über die Deformierung und in wie weit das Silikonband das Auge eindrückt. Somit sollen Risiken und Nebenwirkungen der Cerclage Operation gemindert und eventuelle Komplikationen vorbeugend vermieden werden.



2 Fixierung des Auges bei einer Laserbehandlung

2.1 Darstellung des Problems

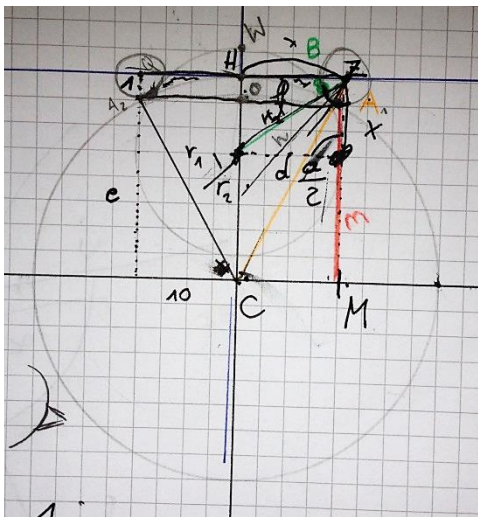


Eine weitere Behandlungsmethode bei einer noch nicht großräumig fortgeschrittenen Netzhautablösung ist die Laserbehandlung, bei der rund um den abgelösten Teil der Neuroretina mit einem Laser Narben erzeugt werden, um so die Neuroretina nach einigen Tagen erneut mit dem Pigmentepithel - der darunterliegenden Versorgungsschicht - verwachsen zu lassen. Dabei wird ein Kunststoffrohr, mit einem Torus als Endstück, zur Fixierung des Auges während der Behandlung auf das Auge gedrückt und das Auge wird mit einem Sauger vorsichtig angesaugt.

Es gilt herauszufinden, wie sich das Auge bei angelegtem Augfixierer und erzeugtem Vakuum verformt und sich der innere Druck des Auges verändert.

2.2 Weg zur Problemlösung

Wir stellten uns die grundlegende Frage, was passiert, wenn man einen Holahop-Reifen auf einen Gymnastikballs drückt und wie verformt sich dieser dabei.



Es wurde ein Modell erstellt; bei dem davon ausgegangen wird, dass sich das Auge nach Anlegen der Zange und Anschalten eines Saugers, zu zwei verschieden großen Kugelsegmenten verändert. Zur Vereinfachung der Berechnungen wurde nicht auf die kleinen Lücken geachtet, die nach dem Modell rund um die Zange entstehen.

Es wurde jeder wichtige Punkt der Skizze mit einem Namen bezeichnet, damit mit Hilfe der

Vektorrechnung Spannung, Größe der Kugeln, Druck und einwirkende Kräfte berechnet werden konnten. Auch bei diesem Problem wurde ein Gleichungssystem mit vier Unbekannten (r_1, r_2, λ, T_2) aufgestellt, die mit Hilfe des Computerprogrammes „MatLab“ berechnet werden können.

$$f_1 = (p_0 + \lambda) * \frac{r_1}{2} - T_1$$

$$f_2 = (p_0 + \lambda + I) * \frac{r_2}{2} - T_2$$

$$f_3 = (p_0 + \lambda - p_R) - 2T_3H_3$$

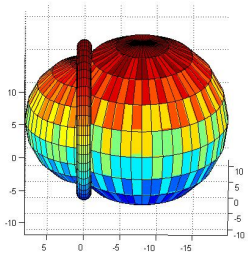
$$f_4 = V - V_0$$

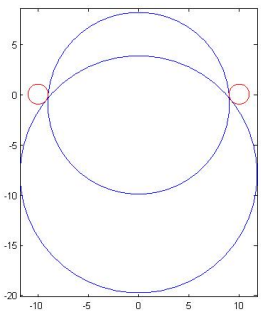
- p_0 = Druckdifferenz des Auges im präoperativen Zustand
- λ = durch den Druck des Silikonbandes erhöhte Druck im Inneren
- r_1 = Radius des großen Kugelsegmentes
- T_1 = Spannung des großen Kugelsegmentes
- p_R = Druck des Kunststofftorus auf das Auge
- T_3 = Spannung am Auflagepunkt des Toruses
- H_3 = Krümmung am Auflagepunkt des Kunststofftorus
- V = Volumen des Auges im intraoperativen Zustand
- V_0 = Volumen des Auges im präoperativen Zustand
- I = Änderung der Spannung an der Krümmung

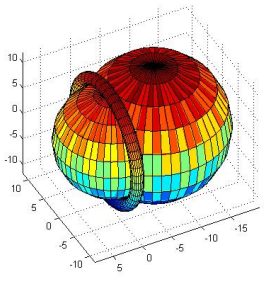
Errechnete Werte bei einem durchschnittlichen Auge:

r_1	Radius des großen Kugelsegments	11,7939 mm
r_2	Radius kleinen Kugelsegments	9,0217 mm
λ	Der durch das Silikonband erhöhte Druck im Inneren	33,7661 mmHg
T_1	Spannung des großen Kugelsegments	334,7472 mmHg
T_2	Spannung des kleinen Kugelsegments	3357,9336 mmHg
T_3	Spannung in der Krümmung	2,7729 mmHg

Auge im intraoperativen Zustand während der Laserbehandlung







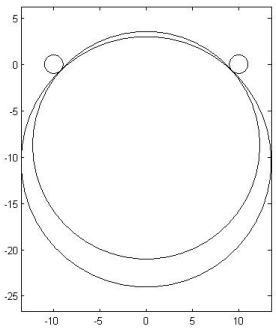
2.3 Odyssee

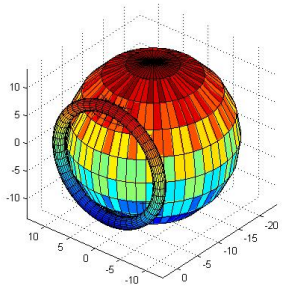
Bei unserem ersten Versuch gingen wir von einem Modell aus bei dem sich die untere, von uns erstellte Halbkugel keineswegs verändert. Auf diese Halbkugel wurde ein Art Torus ohne Loch in der Mitte gesetzt und zu Oberst platzierten wir eine kleine Halbkugel, die von der Zange hinausgedrückt wurde.

Wir konnten dieses Modell allerdings nicht berechnen, da Berechnungen mit einem Torus ohne Loch nicht möglich waren und sich später herausstellte, dass prinzipiell nicht davon ausgegangen werden konnte, dass sich die untere Halbkugel überhaupt nicht veränderte.

Danach stellten wir das Modell mit den zwei Kreissegmenten auf, aber auch hier rechneten wir zuerst im Kreis, da wir Formeln ohne Vektoren aufstellen wollten und sich dann viel zu viele Unbekannte ergaben.

Als wir unsere Gleichungen schon in „MATLAB“ eingegeben hatten, stellte sich erst heraus, dass wir vergessen hatten, die Saugkraft des Saugers (Abb.) miteinzubeziehen.





Eine der größten Schwierigkeiten, war das Programmieren des Systems im Programm „MATLAB“, da das angewendete System „f-solve“ Nullstellen sucht und das hängt von den angegebene Startwerten ab, die nicht leicht zu finden waren.

2.4 Resultat

Als Resultat wurde eine Grafik erhalten, die die Veränderung des Auges mit der Zange und des Saugers darstellte. Mit Hilfe unseres Codes ist man nun auch in der Lage, ausgehend von jedem beliebigen Auge, die Veränderung des Auges während des Lasereingriffs vorauszubestimmen und mögliche Schäden wegen des zu hohen Druckes oder zu großer Kraft können vermieden werden. Der Code in „MATLAB“ könnte noch

dahingehend verbessert werden, dass ein genauerer Übergang vom einem Segment ins andere modelliert wird. Aber prinzipiell könnte unser Code bereits in der Medizin angewendet werden.