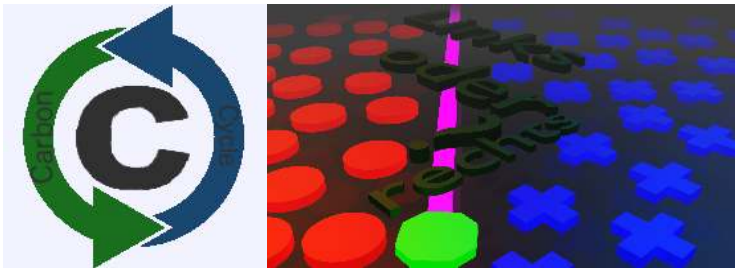
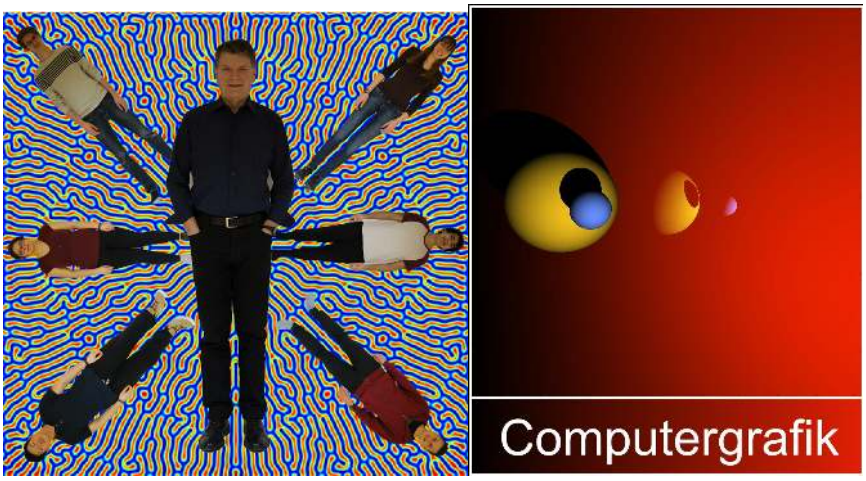
13. WOCHE
DER MODELLIERUNG MIT MATHEMATIK





Dokumentationsbroschüre 4.2. – 10.2.2017

WOCHE DER MODELLIERUNG MIT MATHEMATIK

PÖLLAU, 4.2.–10.2.2017

WEITERE INFORMATIONEN:

HTTP://MATH.UNI-GRAZ.AT/MODELLWOCHE/2017/

ORGANISATOREN UND SPONSOREN

KOORDINATION

Thomas Russold, BA MA

Alexander Sekkas

Mag. DDr. 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 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 13. Mal diese inzwischen zur Institution gewordene Veranstaltung. Ihm sei an dieser Stelle noch einmal ausdrücklich und herzlich gedankt. Besonders wichtig war in den vergangenen Jahren auch die Unterstützung durch den langjährigen Mentor der Modellierungswoche, Herrn o.Univ.-Prof. Dr. Franz Kappel, der oft auch eine eigene Gruppe mit interessanten Problemstellungen betreut hat.

Wir danken dem Landesschulrat für Steiermark, und hier insbesondere Herrn Landesschulinspektor Mag. Gerhard Sihorsch und Frau Fachinspektorin Mag. Michaela Kraker, für die Hilfe bei der Organisation und die kontinuierliche Unterstützung der Idee einer Modellierungswoche.

Finanzielle Unterstützung erhielten wir von der Karl-Franzens-Universität Graz durch Vizerektor Prof. Dr. Martin Polaschek und Dekan Prof. Dr. Christof Gattringer, vom regionalen Fachdidaktikzentrum für Mathematik und von Comfortplan.

Ohne den idealistischen, unentgeltlichen und engagierten Einsatz der direkten Projektbetreuer Florian Thaler, BSc, Sebastian Engel, MSc, Richard Huber, BSc und Dr. Laurent Pfeiffer – Institut für Mathematik und Wissenschaftliches Rechnen – hätte diese Modellierungswoche nicht stattfinden können.

Besonderer Dank gebührt ferner Herrn Mag. DDr. Patrick-Michel Frühmann, der die ganze Veranstaltung betreut und auch die Gestaltung dieses Berichtes übernommen hat, Herrn Thomas Russold, BA MA, für die tatkräftige Hilfe bei der organisatorischen Vorbereitung und Herrn Alexander Sekkas für die Hilfe bei der Betreuung der Hard- und Software.

Pöllau, am 10. Februar 2017

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

Klimaphysik

Der (globale) Kohlenstoffkreislauf

Konrad Eisenberger, Nina Lampl, Jakob Pertl
Lena Wurzinger, Martina Wutti, Victoria Zeiler

Betreuer: Florian Thaler, BSc

Inhaltsverzeichnis

1 Einleitung

Der (globale) Kohlenstoffkreislauf ist ein in den Medien sehr präsentes Thema, wobei man meistens das Gefühl hat, nicht besonders tiefgehende Informationen zu erhalten. Normalerweise spricht man vom Klimawandel beziehungsweise der Klimaerwärmung, im Zusammenhang mit der Kohlenstoffdioxidkonzentration in der Luft. Um ehrlich zu sein, machen kohlenstoffhaltige Stoffe aber nur ungefähr 25 Prozent des Treibhauseffekts aus. Wasserdampf hat mit rund 60 Prozent den höchsten Anteil, dennoch ist der Wasserdampf hauptsächlich für den natürlichen Treibhauseffekt verantwortlich, ohne den es auf der Erde $-18°$C hätte. Der anthropogene Treibhauseffekt wird vor allem durch CO_2 (Kohlenstoffdioxid) und CH_4(Methan) verursacht. Dies kann verheerende Folgen haben, wie zum Beispiel den Anstieg des Meeresspiegels durch das Abschmelzen der Gletscher oder die zunehmende Desertifikation in weiten Gebieten der Erde.
Allen aus unserer Gruppe liegt die Umwelt sehr am Herzen, aber vor diesem Projekt hat sich kaum jemand von uns tiefer mit dem Kohlenstoffkreislauf beschäftigt. Es ist sehr wichtig, sich um unseren Planeten zu kümmern, denn Nachhaltigkeit ist ein wichtiges Prinzip, das immer mehr Anklang in unserer Gesellschaft findet und auch finden muss.

2 Annahmen

Bevor wir unser Modell aufgestellt haben, mussten wir ein paar Annahmen formulieren. Vor allem da wir nur eine Woche Zeit hatten, mussten wir uns mehr beschränken als normale Klimaforscher das tun. Schulussendlich geht es ja beim Modellieren genau darum. Unsere erste Annahme ist, dass die Erde ein geschlossenes System in Bezug auf die Kohlenstoffmenge ist. Lange Rede kurzer Sinn, es kommt kein zusätzlicher Kohlenstoff aus dem All und es geht kein Kohlenstoff verloren. Der Kohlenstoff wird immer nur verschoben. Zweitens haben wir die Kryosphäre (Eis) einfach der Hydrosphäre untergeordnet. Es wäre zwar kein großer Aufwand, die Kryosphäre als eigenes Compartment darzustellen, aber es wäre ein Mehraufwand für keine besseren Ergebnisse gewesen. Weiteres wird von uns angenommen, dass alle Sphären gleichmäßig auf der Erde verteilt sind. Diese Annahme ist natürlich unrealistisch, da es mit den Kontinenten riesige Bereiche gibt, in denen es eine relative kleine Hydrosphäre gibt. Eine weitere Annahme ist, dass unsere Parameter alle konstant sind. Wenn heutzutage 10% der Atmosphäre in die Biosphäre wechseln, dann nehmen wir an, dass das in 1000 Jahren auch noch so sein wird. Natürlich ist streitbar, ob das realistisch ist. Das Problem ist jedoch, dass wir nicht schätzen können, wie sich unsere Parameter verändern könnten.

3 Das Compartment-Modell

3.1 Mengenbilanz

Abbildung 1: Zufluss und Abfluss

Sei z die zeitabhängige Zustandsgröße des Compartments, sei $f_{in}(t)$ der Zufluss in das Compartment und $f_{out}(t)$ der Abfluss aus dem Compartment, dann ergibt sich die zeitliche Änderungsrate von z, sprich die Ableitung (z'(t)) gerade durch:

$$z'(t) = f_{in}(t) - f_{out}(t)$$

Dieses Prinzip führt uns in Abschnitt 3.3 zu einem gekoppelten linearen Differentialgleichungssystem.

3.2 Kohlenstoffzyklus

Als Compartment-Modell bezeichnet man im Allgemeinen ein Modell, das aus mehreren Bereichen besteht. Von diesen Bereichen kann es Zuflüsse oder Abflüsse geben, aber es kann nichts Neues entstehen und nichts verschwinden, was bedeutet, dass man es hierbei mit einem geschlossenen Kreislauf zu tun hat. Eine grundlegende Annahme betreffend eines Compartments ist dessen Homogenität. Im vorliegenden Fall handelt es sich um einen geschlossenen Kohlenstoffkreislauf. Die verschiedenen Compartments sind Atmosphäre, Biosphäre, Hydrosphäre und Lithosphäre.

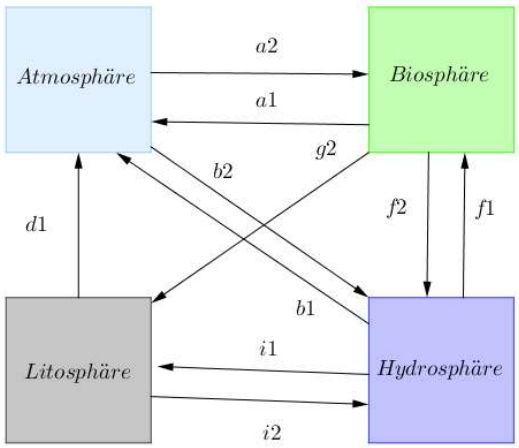


Abbildung 2: Compartment Modell ohne menschlichen Einfluss

In der Atmosphäre befinden sich 7,2·10^2 Gigatonnen Kohlenstoff, zum Beispiel als Anteil bei der Zusammensetzung der Luft als CO_2. In der Biosphäre findet man $4 \cdot 10^3$ Gigatonnen, vor allem im menschlichen Körper. Speicherformen in der Biosphäre sind Carbonate und Skelette. In der Hydrosphäre befinden sich $3,84015 \cdot 10^4$, hauptsächlich in gelöster Form. Die Mehrheit des Kohlenstoffes auf der Erde ist mit 7,500413·10^7 Gigatonnen in der Lithosphäre vorhanden. Dort gibt es vor allem in Sedimenten in Form von Calciumcarbonaten eine große Menge an Kohlenstoff.

3.3 Kohlenstoffflüsse

Aus der Atmosphäre kommt Kohlenstoff in die Biosphäre, indem Pflanzen Photosynthese betreiben, da sie bei diesem Vorgang aus CO_2 energiehaltige Produkte – zumeist Kohlenhydrate – erzeugen. Jedoch gibt es auch entgegengesetzte Kohlenstoffflüsse, weil Menschen, Tiere und Pflanzen atmen. Aus der Hydrosphäre gelangen kohlenstoffhaltige Moleküle in die Biosphäre, indem Tiere Wasser trinken. Dies macht allerdings nur einen minimalen Bereich des globalen Kohlenstoffkreislaufs aus, weswegen man es außer Acht lassen kann. Umgekehrt kommt Kohlenstoff von der Biosphäre in die Hydrosphäre, zum Beispiel, wenn abgestorbene Meerestiere zersetzt werden. Wenn die Skelette der abgestorbenen Meerestiere in die Gesteinsschicht eingelagert werden, kommt es zu einem Kohlenstofftransfer von der Hydrosphäre in die Lithosphäre. Es gibt auch weitere Flüsse, die man der oben zu sehenden Grafik entnehmen kann.

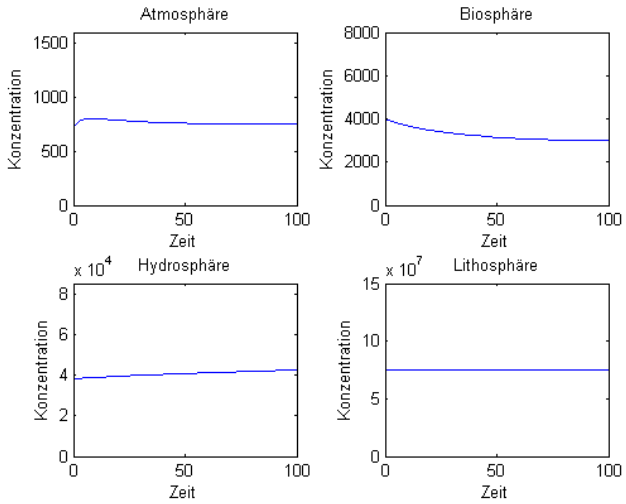


Abbildung 3: Kohlenstoffkonzentration in den verschiedenen Compartments

Interessanterweise greift der Mensch immer mehr in den natürlichen Kohlenstoffkreislauf ein, da das Verbrennen der fossilen Brennstoffe zu einem Anstieg des Kohlenstoffgehalts in der Luft führt. Dieser Kohlenstoff stammt aus der Lithosphäre, da man zum Beispiel Erdöl, Erdgas und Kohle. Aber auch durch das Verbrennen von Holz gelangt mehr Kohlenstoff in die Luft. Deswegen sollte man auf erneuerbare Energieträger setzen und klimafreundliche Politik forcieren.

3.4 Das gekoppelte, lineare Differentialgleichungssystem

$$
\begin{aligned}
C'_A &= a_1 \cdot C_B + b_1 \cdot C_H + d_1 \cdot C_L + e_1 \cdot C_L - (a_2 \cdot C_A + b_2 \cdot C_A + k \cdot C_A) \\
C'_B &= a_2 \cdot C_A + f_1 \cdot C_H - (a_1 \cdot C_B + f_2 \cdot C_B + g_2 \cdot C_B) + k \cdot C_A \\
C'_H &= b_2 \cdot C_A + f_2 \cdot C_B + i_1 \cdot C_L - (i_2 \cdot C_H + b_1 \cdot C_H + f_1 \cdot C_H) \\
C'_L &= i_2 \cdot C_H + g_2 \cdot C_B - (i_1 \cdot C_L + d_1 \cdot C_L + e_1 \cdot C_L)
\end{aligned}
$$

Der Parameter k beschreibt die Idee des Carbon storage, welche später erläutert wird, während der Parameter e_1 den negativen menschlichen Einfluss widerspiegelt. Dieser resultiert hauptsächlich aus der Verbrennung fossiler Ressourcen. Nun folgt eine Darstellung des obenstehenden Gleichungssystems mithilfe einer Matrix-Vektor-Multiplikation. Zuerst wird diese Methode aber noch an einem einfachen Beispiel erklärt:

$$
\begin{bmatrix} 1 & 0 & 1 \\ 3 & 2 & 1 \\ 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 + 0 \cdot 2 + 1 \cdot 3 \\ 3 \cdot 1 + 2 \cdot 2 + 1 \cdot 3 \\ -1 \cdot 1 + 0 \cdot 2 + 0 \cdot 3 \end{bmatrix} = \begin{bmatrix} 4 \\ 10 \\ -1 \end{bmatrix}
$$

Als Ergebnis erhält man einen Vektor. Am Beispiel erkennt man, dass die Dimensionen der Matrix 3x3 und die des Vektors 3x1 sind. Man kann Matrizen nur mit Vektoren multiplizieren, wenn die inneren Dimensionen übereinstimmen. Dann erhält man als Ergebnis eine Matrix bzw. in diesem Fall einen Vektor, deren bzw. dessen Dimensionen den äußeren Dimensionen entsprechen.

$$
\begin{bmatrix} c'_A \\ c'_B \\ c'_H \\ c'_L \end{bmatrix} = \begin{bmatrix} -(a_2 + b_2 + k) & a_1 & b_1 & d_1 + e_1 \\ a_2 + k & -(a_1 + f_2 + g_2) & f_1 & 0 \\ b_2 & f_2 & -(i_2 + b_1 + f_1) & i_1 \\ 0 & g_2 & i_2 & -(i_1 + d_1 + e_1) \end{bmatrix} \cdot \begin{bmatrix} c_A \\ c_B \\ c_H \\ c_L \end{bmatrix}
$$

Da wir annehmen, dass es sich um einen geschlossenen Kohlenstoffkreislauf handelt, muss die Summe der Ableitungen 0 ergeben. Wenn das der Fall ist, gibt es keine Änderung

der Stammfunktion, also keinen Zufluss oder Abfluss von Kohlenstoff. Deswegen gilt:

$$\begin{aligned} C'_A + C'_B + C'_H + C'_L = {} & a_1 \cdot C_B + b_1 \cdot C_H + d_1 \cdot C_L + e_1 \cdot C_L - a_2 \cdot C_A - b_2 \cdot C_A - k \cdot C_A + \\ & + a_2 \cdot C_A + k \cdot C_A + f_1 \cdot C_H - a_1 \cdot C_B - f_2 \cdot C_B - g_2 \cdot C_B + b_2 \cdot C_A + \\ & + f_2 \cdot C_B + i_1 \cdot C_L - i_2 \cdot C_H - b_1 \cdot C_B - f_1 \cdot C_H + i_2 \cdot C_H + g_2 \cdot C_B - \\ & - i_1 \cdot C_L - e_1 \cdot C_L - d_1 \cdot C_L = 0 \end{aligned}$$

3.5 Exakte Lösung einer Bilanz-Differentialgleichung

In der nachfolgenden Rechnung soll zur Anschauung eine einfache Bilanz-Differentialgleichung analytisch gelöst werden. Dazu betrachten wir folgende Gleichung:

$$y'(t) = a \cdot y(t)$$

Dieses Beispiel stellt ein Anfangswertproblem dar. Zum Anfangszeitpunkt t_0 wurde der Anfangswert $y(t_0) = y_0$ gewählt. Für $t \geq 0$: gilt also:

$$\frac{y'(t)}{y(t)} = a$$

Man führt hier eine Trennung der Variablen durch. Integriert man diesen Ausdruck, erhält man:

$$\int_{t_0}^{s} \frac{y'(t)}{y(t)} dt = a \cdot \int_{t_0}^{s} dt$$

Nach geltenden Ableitungsregeln ist $\ln'(x) = \frac{1}{x}$. Auf diese Gleichung angewandt, bedeutet das:

$$(\ln[y(t)])' = \frac{1}{y(t)} \cdot y'(t)$$

Integriert man nun die linke Seite der Gleichung, erhält man:

$$\int_{t_0}^{s} \frac{y'(t)}{y(t)} dt = \ln[y(t)]\Big|_{t_0}^{s}$$

Außerdem integriert man natürlich auch noch die rechte Seite der Gleichung:

$$\ln[y(t)]\Big|_{t_0}^{s} = a \cdot (s - t_0)$$

Durch Äquivalenzumformungen gelangt man schließlich zu der Stammfunktion y(s).

$$\ln[y(s)] - \ln[y(t)] = a \cdot (s - t_0)$$

$$\ln[\frac{y(s)}{y(t_0)}] = a \cdot (s - t_0)$$

$$y(s) = y(t_0) \cdot e^{a \cdot (s - t_0)}$$

4 Strahlungsbilanz

4.1 Modell 1

Die Strahlungsbilanz ist die Differenz von Strahlungsflüssen, wenn keine interne Strahlungsquelle existiert. Das bedeutet, dass man die zurückgesendete Strahlung von der einfallenden Strahlung abzieht. Das kann man mit einem Zufluss-Abfluss-Modell darstellen. Die Sonne wird als punktuelle externe Strahlungsquelle angenommen, von der $340\,W/m^2$ bei uns an der Erdoberfläche ankommen. Diesen Wert erhält man, indem man die Solarkonstante (S) 1360 durch 4 dividiert. In die Erde gelangt der Anteil, der mit folgender Formel beschrieben werden kann:

$$\sigma \cdot \pi \cdot r^2 \cdot (1 - \alpha)$$

Die Albedo(α) besagt, wie viel Prozent der Sonneneinstrahlung abgeschirmt werden. Diese Abschirmung kann zum Beispiel durch Wolken oder Schnee erfolgen. Als gemittelten Durchschnittswert nimmt man 0,3 an. Daras folgt, dass 70 Prozent der Sonneneinstrahlung auf die Erde gelangen. Die von der Erde im infraroten Spektrum abgegebene Strahlung wird durch den folgenden Ausdruck mit Hilfe des Stefan-Boltzmann-Gesetzes beschrieben:

$$s \cdot T^4 \cdot \pi \cdot r^2 \cdot \varepsilon$$

Dabei bezeichnet σ die Stefan-Boltzmann Konstante. T entspricht der Durchschnittstemperatur der Erde und ε ist eine Zahl zwischen 0 und 1, die die Emissivität widerspiegelt. Je kleiner ε ist, desto stärker ist der Effekt der globalen Klimaerwärmung.

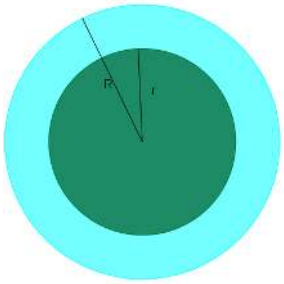


Abbildung 5: Vereinfachte Darstellung Erde und Atmosphäre

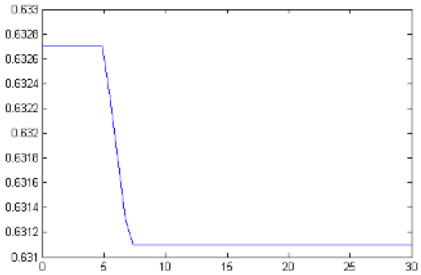


Abbildung 4: Emissivität

Die einfallende Wärmestrahlung, genauer gesagt Infrarotstrahlung, ist nichts anderes als Energie. Deswegen kann man sagen, dass gilt:

$$\Delta E = m \cdot C \cdot \Delta T = \rho \cdot V \cdot C \cdot \Delta T$$

Das Volumen V einer Kugelschale kann man folgendermaßen berechnen:

$$V = \frac{4 \cdot \pi \cdot (R^3 - r^3)}{3}$$

Hierbei ist R der Radius der großen Kugel – also der Erde inklusive Atmosphäre – und r der Radius der kleinen Kugel – also der Erde exklusive Atmosphäre. Außerdem gilt der Zusammenhang:

$$R - r = h$$

Daraus folgt:

$$T'(t) = \frac{S \cdot \pi \cdot r^2 \cdot (1 - \alpha) - \sigma \cdot T(t)^4 \cdot 4 \cdot \pi \cdot r^2 \cdot \varepsilon}{\rho \cdot \frac{4}{3} \cdot \pi \cdot (R^3 - r^3) \cdot C}$$

C ist die spezifische Wärme für Luft.

4.2 Modell 2

Nachdem die Strahlungsbilanz uns nach langem Bearbeiten unrealistische Daten geliefert hat, war es Zeit, ein anderes Modell zu entwickeln. Das neue Modell sollte nun zu „richtigen“ Daten führen, nur über einen anderen Lösungsweg. Die Einstrahlung der Sonne wird dabei vollkommen außer Acht gelassen. Die Erde gibt Infrarotstrahlung ab. Ein Teil dieser Strahlung verlässt die Erde ohne schwerwiegende Interaktion mit der Atmosphäre und wird oft als Epsilon abgekürzt. Der andere Teil, den man betrachten sollte, ist folglich (1-ε). Die Strahlung wird in Hitze umgewandelt, da sie auf atomarer Ebene die Teilchen anregt. Daher darf man diese Strahlung als Energie ansehen.

$$E = (1 - \varepsilon) \cdot \sigma \cdot A \cdot (T_s^4 - T_a^4)$$

Wobei Sigma die Stefan-Boltzmann Konstante darstellt, A die Oberfläche des Strahlers und T_s und T_a die Temperatur der Oberfläche und der Umgebung. Man muss aber bedenken, dass die Einheit pro Sekunde ist, also dividiert man durch die Zeit. Jetzt fehlt noch eine Formel, die Energie zu dem Unterschied der Temperatur verlinkt. Die Formel der spezifischen Wärmekapazität eignet sich perfekt dafür. Sie beschreibt, wie sich Hitze auf die Temperatur auswirkt:

$$\Delta Q = c \cdot m \cdot \Delta T$$

Q-Hitze, c-spezifische Wärmekapazität, m-Masse und δ T, der Unterschied der Temperatur. Die zwei obigen Formeln lassen sich jetzt gleichsetzen, da beide Energie durch Zeit sind. Wenn man dies tut und δ T isoliert kommt man zu dieser Formel:

$$\Delta T_a = \frac{(1 - \varepsilon) \cdot \sigma \cdot A \cdot (T_s^4 - T_a^4)}{c \cdot m} \cdot \Delta t$$

Nun, da man eine fertige Formel hat, muss man nur mehr die Werte für unsere Erde einsetzen. Der Großteil der Werte lässt sich schnell finden, zum Beispiel die Temperaturen und die Fläche. Auch die Masse der Atmosphäre lässt sich durch die Dichte der einzelnen Gase gut berechnen. Schwierig erweist sich die spezifische Wärmekapazität der Atmosphäre, da unsere Atmosphäre sich aus verschiedenen Gasen zusammensetzt. Um die durchschnittliche Kapazität herauszufinden sollte man jedoch zwischen den Gasen gewichten, da zum Beispiel Stickstoff viel häufiger vorkommt als Kohlendioxid. Den Wert den man schließlich ermittelt, sollte die durchschnittliche Wärmekapazität unserer Erde entsprechen. Schlussendlich setzt man alle Daten bei MATLAB ein. Der Graph, den das Programm darstellt, ist ironischerweise fast ident zu dem unserer vorherigen Strahlungsbilanz. Ob das zufällig passiert, ist streitbar, aber es ist durchaus seltsam, dass man über zwei verschiedene Lösungswege zu dem gleichen Ergebnis kommt. Wahrscheinlich haben beide Modelle einen wichtigen Faktor außer Acht gelassen, da wir es uns nicht anmaßen

wollen, die Erderwärmung zu leugnen.

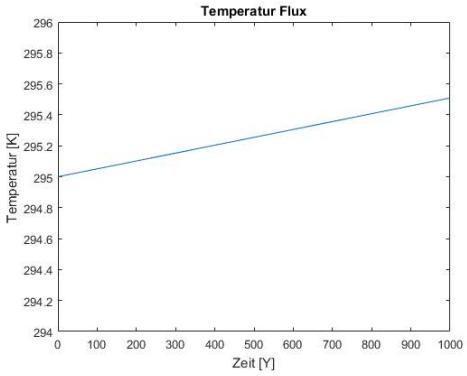


Abbildung 6: Strahlungsbilanzmodell 2

5 Szenarien und Ergebnisse

5.1 Vergleich: Kohlenstoffgehalt in den Sphären mit und ohne menschlichen Einfluss

Ausgehend von einem Zyklus ohne menschlichen Einfluss wurden Graphen bezüglich des Kohlenstoffgehalts in den verschiedenen Sphären erstellt. Zu sehen ist, dass der Gehalt der Lithosphäre annähernd konstant bleibt. Der Graph der Hydrosphäre steigt leicht an, wobei bei Atmos- und Hydrosphäre ein Minimum zu vermerken ist.

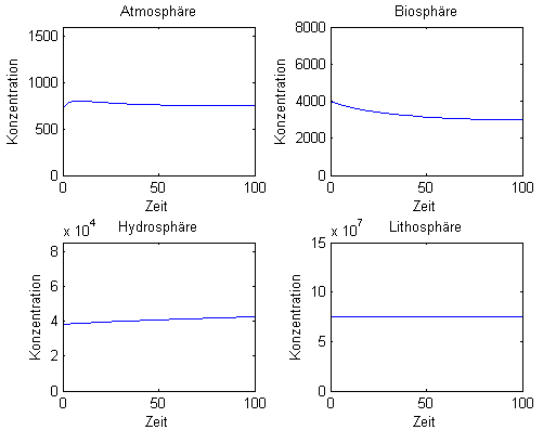


Abbildung 7: Kohlenstoffzyklus ohne menschlichen Einfluss

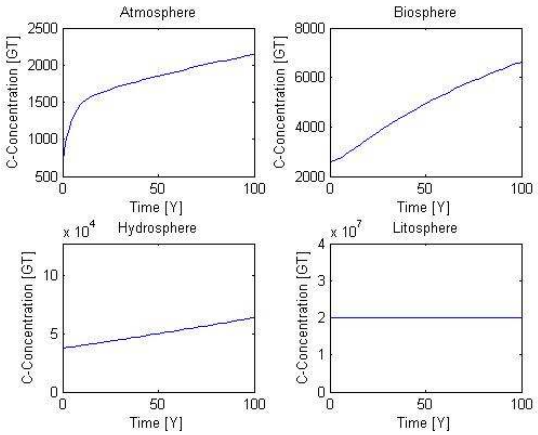


Abbildung 8: Kohlenstoffzyklus mit menschlichem Einfluss

Bezüglich der Graphen des Kohlenstoffgehalts mit einberechnetem menschlichem Einfluss sind Veränderungen in der Atmos- und Biosphäre zu vermerken. In der Biosphäre steigt der Gehalt stetig an. Ebenfalls ist die Atmosphäre verändert und der Kohlenstoffgehalt nimmt vor allem in den ersten Jahren stark zu. Die Graphen der Hydro- und Lithosphäre bleiben gleich. Jedoch ist hier zu beachten, dass der Kohlenstoffgehalt in der Lithosphäre, aufgrund des Verbrauchs von fossilen Brennstoffen, enorm sinken könnte. Deshalb muss erwähnt werden, dass sich das Modell der Realität nur annähert.

5.2 Delay Model

Die Idee hinter dem Delay Model ist, den Übergang zwischen den verschiedenen Departments nicht unmittelbar darzustellen, sondern mit einer gewissen Verzögerung. Dies kommt der Realität näher, da in der Natur Prozesse prinzipiell langsam verlaufen. Das Problem, das mit diesem Model auftritt, ist, dass man nicht mehr automatische Solver wie Ode45 verwenden kann. Um seinen eigenen Solver zu programmieren, muss man vorerst Punkte generieren und diese mit Nullen reservieren. Daraufhin schreibt man eine Schleife mit „If clauses“. Jede „If clause“ beinhaltet sowohl unseren Delay Parameter als auch eine unserer Kohlenstoffkonzentrationsgleichungen. Die Schleife ordnet jetzt Werte bis zu dem vorher festgelegten Zeitpunkt T zu. Je nachdem, wie man nun die Delay Parameter bestimmt, beeinflusst der jetzige Wert nicht den nächsten, sondern einen späteren Wert. Jetzt nimmt man die Daten von vorher und setzt diese ein. Die Ergebnisse sind den vorherigen ähnlich, wirken aber kantiger. Der einzige große Unterschied ist bei der Atmosphäre zu finden, bei der der Graph wilde Zickzack-Linien projiziert. Generell kommt dieses Modell näher an die Werte einiger Klimaforscher heran.

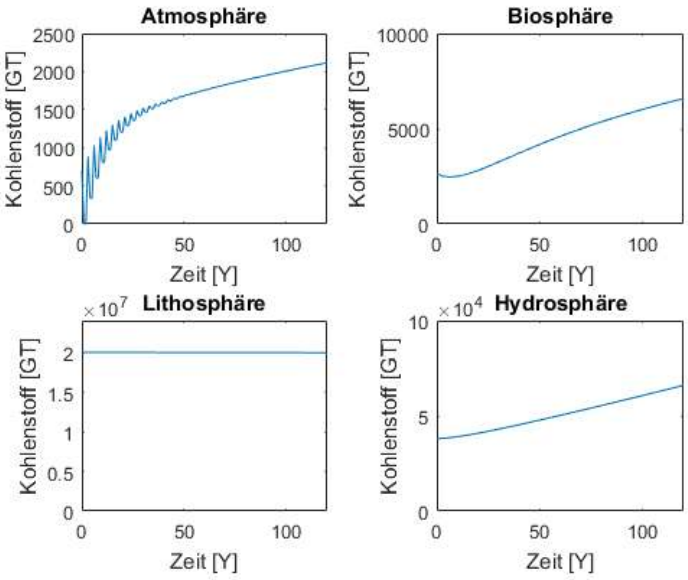


Abbildung 9: Delay Model

5.3 Der Vulkanausbruch

Für jedes Experiment muss man auch Extremfälle in Betracht ziehen. Einer dieser Fälle ist ein Vulkanausbruch. Wir wussten nicht, wie sich ein solches Ereignis auf die verschiedenen Sphären, also Atmo-, Bio-, Hydro- und Lithosphäre, auswirkt. Um zu beginnen, braucht man eine spezielle Funktion, die einen hohen Wert an Kohlenstoff generiert. Dazu verwendeten wir zuerst eine skalierte Gauß'sche Glockenfunktion, oder wie sie auch genannt wird, eine Hutfunktion.

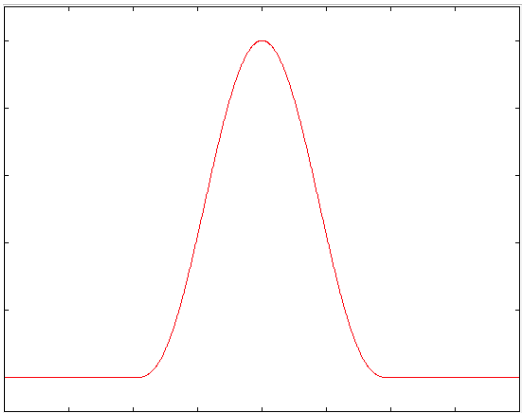


Abbildung 10: Gauß'sche Glocke

Man stellt die Parameter so ein, dass man eine extrem hohe Glocke generiert, um einen großen Ausstoß an Kohlenstoff zu simulieren. Danach fügt man die neue Funktion in das schon vorhandene Modell des Kohlenstoffstroms zwischen den Sphären. Wir wollten mit dem Vulkanausbruch den Anstieg am Kohlenstoffffluss von der Lithosphäre in die Atmosphäre zeigen, also setzten wir die Hutfunktion für die Variable „d_1“ ein. Dafür verwendet man ein „If“ Statement.

```
if 300<t<320
    t0=310;
    m=0.000004;
    n=0.001;
    d1=m*exp(-n*(t-t0).^2);
else
    d1=0.0000004;
end;
```

Abbildung 11: If Statement

Die Auswirkung der Simulation konnte man in den Graphen deutlich erkennen. Nicht nur in der Atmosphäre, sondern auch in der in den anderen drei Bereichen. Die Kurven der Graphen haben die gleiche Anfangssteigung, doch bei T=330, also nach 330 Jahren, ist der Vulkanausbruch deutlich zu sehen.

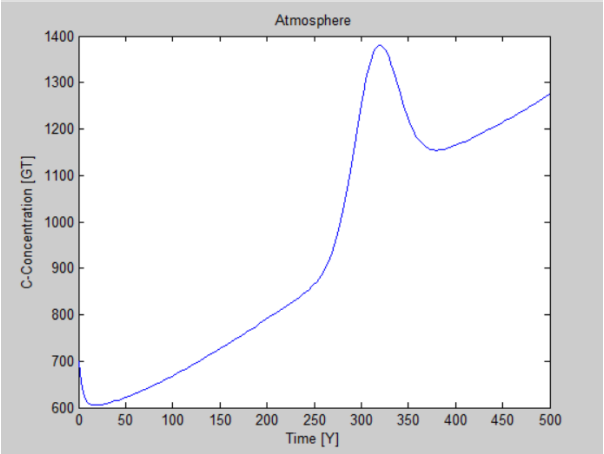


Abbildung 12: Vulkanausbruch (Atmosphäre)

Man sieht also, dass der Kohlenstoffgehalt stark ansteigt und dadurch, dass alle Abteile miteinander verbunden sind, hat es auch große Auswirkungen auf die anderen Sphären, wie zum Beispiel die Biosphäre.

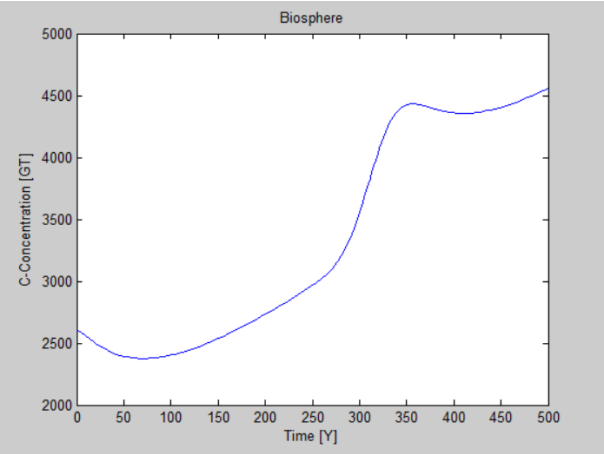


Abbildung 13: Vulkanausbruch (Biosphäre)

5.4 Carbon Storage

Es gibt verschiedene Methoden, Kohlenstoff aus der Atmosphäre zu entfernen und der Biosphäre zuzuführen, beispielsweise durch eine dauerhafte unterirdische Einlagerung. Um herauszufinden, ob diese Maßnahmen tatsächlich bewirken können, dass sich langfristig weniger Kohlenstoff in der Atmosphäre befindet, wurde in das vorgefertigte Modell betreffend des Kohlenstoffgehalts der verschiedenen Sphären ein Parameter von 0.3 eingefügt, der ausdrücken soll, dass 30% des Kohlenstoffgehalts der Atmosphäre durch ebensolche Methoden jedes Jahr in die Biosphäre verschoben werden. Aus diesem Modell kann man nun ablesen, dass sich der Gehalt an Kohlenstoff in der Biosphäre wie zu erwarten steigern würde. Interessant ist die Tatsache, dass der Gehalt an Kohlenstoff zwar kurzfristig nach ca. 2 Jahren einen Tiefpunkt von nur 625 GT Kohlenstoff in der Atmosphäre (im Vergleich zu über 700 GT zum Zeitpunkt 0) erreichen, danach aber wieder stetig steigen würde. Wenn man beispielsweise „nur“ 20% des Kohlenstoffs pro Jahr aus der Atmosphäre entfernen würde, würde dies zwar den Anstieg des Kohlenstoffgehalts etwas verringern, allerdings würde dieser nie abfallen.
Zunächst war ich zwar überrascht, wie gering der Einfluss der Maßnahmen, Kohlenstoff aus der Atmosphäre zu entfernen, tatsächlich wäre, allerdings halte ich dies nach längerer Überlegung für durchaus plausibel: Die Menschen pumpen unter Anderem durch die Verbrennung fossiler Brennstoffe so viel Kohlenstoff in die Atmosphäre, dass sie dies nicht durch solche Maßnahmen ausgleichen können. Dies zeigt das Modell sehr deutlich. Um langfristig den Gehalt an Kohlenstoff in der Atmosphäre zu verkleinern, müssten die Menschen meines Ermessens nach ihren Kohlenstoffausstoß deutlich verringern.

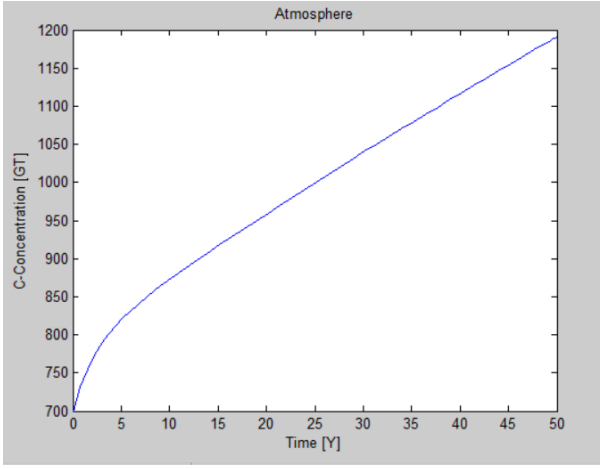


Abbildung 14: Entfernung von 20% Kohlenstoffs (Atmosphäre)

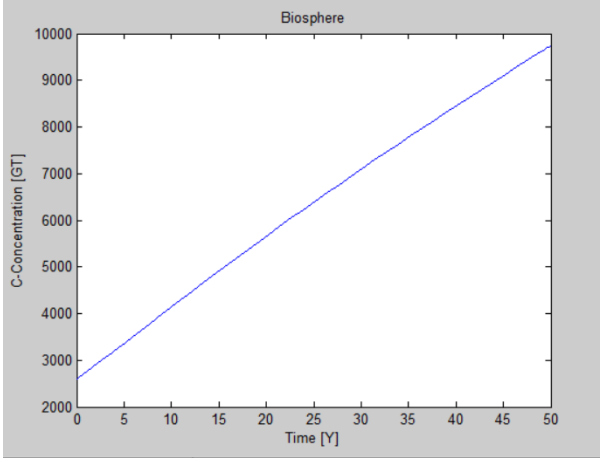


Abbildung 15: Entfernung von 20% Kohlenstoffs (Biosphäre)

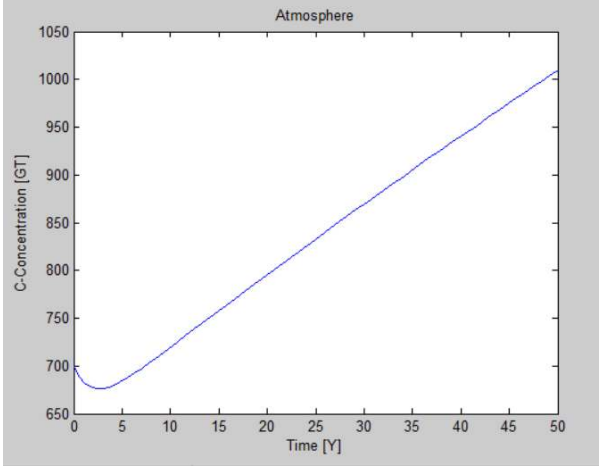


Abbildung 16: Entfernung von 30% Kohlenstoffs (Atmosphäre)

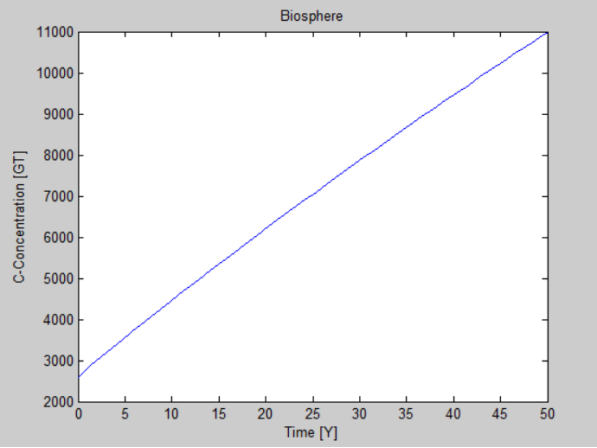


Abbildung 17: Entfernung von 30% Kohlenstoffs (Biosphäre)

5.5 Kohlenstoffausstoß-Modelle in Hinblick auf die Gesellschaft

Es ist offensichtlich, dass der Kohlenstoffausstoß der Menschen stark von Entwicklungen der Gesellschaft abhängig ist. Aus diesem Grund wurde versucht, die Entwicklung von Umweltschützern bzw. Nicht-Umweltschützern mithilfe der sogenannten Lotka-Volterra-Gleichungen (auch als „Predator-Prey-Gleichungen“ bekannt) dazustellen:

$$\frac{dx}{dt} = \alpha x - \beta xy$$

$$\frac{dy}{dt} = \delta xy - y\gamma$$

Die Variable x steht immer für „Prey“, die Beute (oder in diesem Fall die Nicht-Umweltschützer), und die Variable y steht für „Predators“, die Jäger (oder in diesem Fall die Umweltschützer). Die erste Gleichung gibt die Entwicklung der Population an Nicht-Umweltschützern im Verlauf der Zeit an. Mit dem Parameter α, der mit der Anzahl an Nicht-Umweltschützern multipliziert wird, berechnet man das exponentielle Wachstum der Nicht-Umweltschützer. Davon wird die Anzahl an Nicht-Umweltschützern, die von den Umweltschützern überzeugt wurden, ihr Umweltverschmutzertum aufzugeben (dadurch werden sie allerdings nicht zu Umweltschützern!) abgezogen. Dieser Ausdruck wird mithilfe des Parameters β, der sowohl mit der Anzahl an Nicht-Umweltschützern als auch Umweltschützern multipliziert wird, dargestellt
Die zweite Gleichung gibt die Entwicklung der Population der Umweltschützer im Hinblick auf die Zeit an. Der Parameter δ wird mit der Anzahl an Nicht-Umweltschützern und Umweltschützern multipliziert, so stellt dies das Wachstum der Umweltschützer dar. Hiervon wird die Verlustrate an Umweltschützern (Zahl der Umweltschützer multipliziert mit γ) abgezogen.
Die Darstellung von Umweltschützern als „Predators“ und Nicht-Umweltschützern als „Prey“ liegt der Überlegung zugrunde, dass die Zahl an Umweltschützern mit der Zahl an Nicht-Umweltschützern steigt bzw. sinkt. Schließlich engagieren sich mehr Menschen für die Umwelt, wenn die Verschmutzung gerade wirklich stark ist, also besonders viel Kohlenstoff von den Nicht-Umweltschützern ausgestoßen wird. Die Menge an Kohlenstoff, die ausgestoßen wird, steigt direkt proportional mit der Zahl an Nicht-Umweltschützern. Außerdem machen die Umweltschützer „Jagd“ auf die Nicht-Umweltschützer, indem sie diese davon überzeugen, die Umwelt weniger zu verschmutzen. Die Nicht-Umweltschützer werden dadurch aber, wie bereits schon erwähnt, nicht zu Umweltschützern.

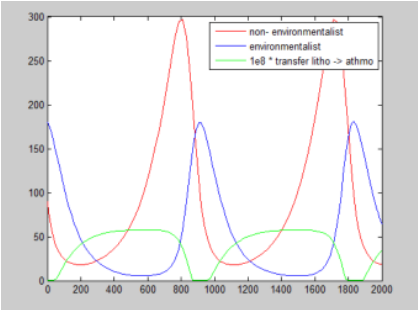


Abbildung 18: Verlauf von e_1

Die Schwachstelle dieser Überlegungen ist, dass die „Predator-Prey-Gleichungen“ eigentlich für die Entwicklung der Populationen von Raubtieren und deren Beute konzipiert sind. Wenn es zu einem Zeitpunkt keine Raubtiere mehr gibt, diese also aussterben, können diese zu einem späteren Zeitpunkt nie wieder entstehen. Anders verhält es sich mit Umweltschützern: Selbst wenn es zu einem Zeitpunkt keine Umweltschützer gäbe, könnten diese trotzdem zu einem späteren Zeitpunkt wieder auftreten. Da in dem ausgeführten Modell die Umweltschützer ohnehin nicht ausster-

ben, ist diese Schwachstelle allerdings vernachlässigbar.

In einem ersten Schritt wird nun der Parameter, der den Kohlenstoffausstoß der Menschen darstellt, mit der Anzahl an Nicht-Umweltschützer multipliziert (je mehr Nicht-Umweltschützer, desto mehr Ausstoß). Davon wird die Anzahl der Umweltschützer multipliziert mit demselben Parameter abgezogen, da diese sich ja für weniger Ausstoß einsetzen. Des Weiteren wird ein unterer Grenzwert für den Kohlenstoffausstoß festgelegt, da selbst von sehr vielen Umweltschützern ein totales Ende des Kohlenstoffausstoßes nicht erreicht werden kann. Auf dieser Grafik kann man nun in Grün den Kohlenstoffausstoß, in Rot die Population an Nicht-Umweltschützern und in Blau die an Umweltschützern im Verlauf der Zeit erkennen.
In weiterer Folge ist es möglich, diesen Einfluss des Menschen, der im vorherigen Schritt ja von der Entwicklung der Gesellschaft abhängig gemacht wurde, in unser Basis-Modell, das den Fluss des Kohlenstoffs zwischen den diversen Sphären zeigt, einzubauen. Hierbei kann man nun ablesen, wie sich der Verlauf der Gesellschaft auf die Kohlenstoffkonzentration in den Sphären auswirkt. In diesem Modell zeigt sich, dass der Gehalt an Kohlenstoff in der Atmosphäre sich auf einem niedrigen Niveau einpendeln würde, wenn abwechselnd Umweltschützer und Nicht-Umweltschützer die Kontrolle über den Kohlenstoffausstoß der Menschen (e_1) hätten. Der Kohlenstoffanteil in der Biosphäre würde abfallen (anstatt wie im Basismodell zu steigen), Hydro- und Lithosphäre würden von diesem Gesellschaftsmodell nicht merklich beeinflusst werden.

Es wäre möglich, das Modell weiter zu verfeinern, indem man die Zahl an Umweltschützern vom Kohlenstoffgehalt in der Atmosphäre abhängig macht, da sich vermutlich mehr Menschen für den Umweltschutz engagieren würden, wenn der Gehalt an Kohlenstoff besonders hoch ist.

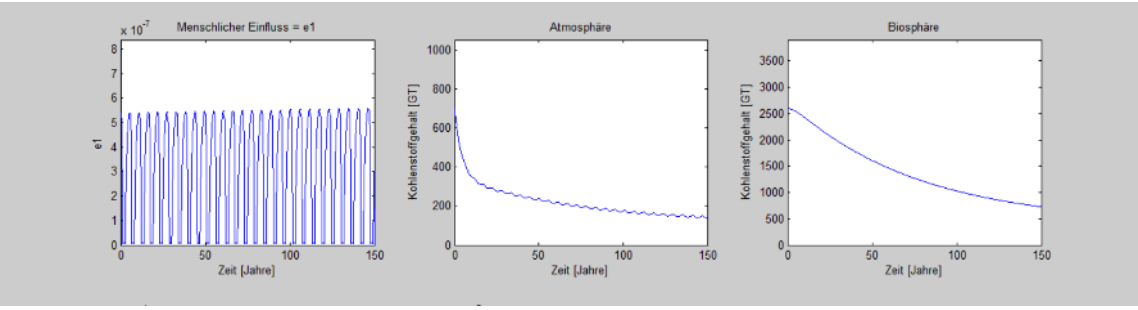


Abbildung 19: Atmosphäre und Biosphäre im Anbängigkeit von e_1

5.6 Nordhalbkugel/Südhalbkugel

In diesem Modell wurden die Nord- und Südhalbkugel der Erde als abgeschlossene Systeme angenommen, so soll dargestellt werden wie viel Kohlenstoff in den jeweiligen Sphären der beiden Erdhälften ist. Besonders groß ist der Unterschied zwischen den Atmosphären der beiden Hälften: Auf der Nordhalbkugel leben 90% der Menschen, deshalb ist der Kohlenstoffausstoß hier viel größer als auf der Südhalbkugel. Der Kohlenstoffanteil der Atmosphäre der Nordhalbkugel ist schon nach ca. 2 Jahren ungefähr gleich hoch wie der der Südhalbkugel nach 100 Jahren. Auch die Biosphäre der Nordhalbkugel hat einen höheren Kohlenstoffgehalt als die Südhalbkugel, dies liegt neben dem stärkeren Einfluss auch daran, dass 67% des Landes der Erde auf der Nordhalbkugel befinden und die Ausgangswerte so höher sind. Auf der Südhalbkugel befinden sich 57% des Wassers der Erde, daher gibt es dort zunächst mehr Kohlenstoff in der Hydrosphäre, allerdings ist nach 100 Jahren mehr Kohlenstoff in der Hydrosphäre der Nordhalbkugel aufgrund des großen Einfluss der Menschen. Natürlich ist dieses Modell nur theoretisch zu betrachten, da sich in Wirklichkeit durch Diffusionsprozesse der Kohlenstoff gleichmäßig in Atmosphäre und auch bedingt in Bio- und Hydrosphäre verteilt.

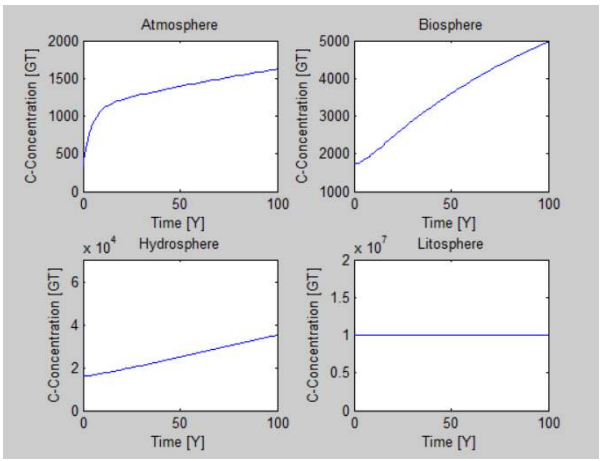


Abbildung 20: Nordhalbkugel

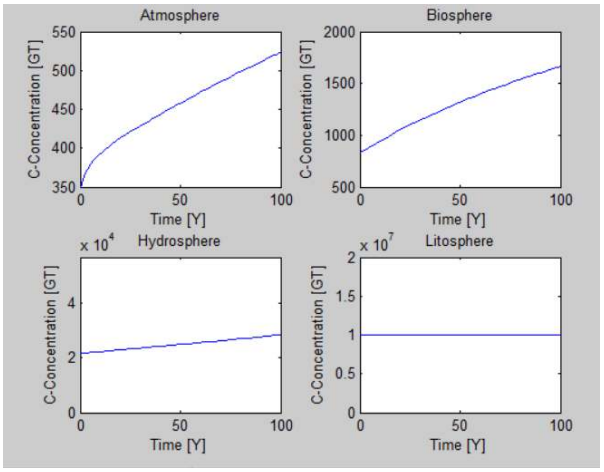


Abbildung 21: Südhalbkugel

6 Fazit

Zusammenfassend lässt sich sagen, dass die Ergebnisse des Projektes zwar nachvollziehbar sind, jedoch eher ungenau und nicht zur Prognose von zukünftigen Umweltereignissen verwendet werden können. Durch die Vereinfachung wurden sehr viele äußere Einflüsse außer Acht gelassen. Grund dafür war die mangelnde Zeit, die zur Verfügung stand, sowie das fehlende Fachwissen im Bereich der Modellierung.
Durch das Erstellen von Graphen mithilfe von MATLAB wurden am Anfang erstellte Vermutungen bestätigt und mathematisch dargestellt. Von den Annahmen wich nur das Strahlungsbilanzmodell ab, da der Temperaturanstieg laut Berechnungen sehr gering ausfällt. Des Weiteren konnten auch nicht alle Ideen umgesetzt werden, aufgrund von fehlenden Informationsquellen. Abschließend ist jedoch zu erwähnen, dass die anfangs gesetzten Ziele größtenteils mit Erfolg erreicht wurden. Schlussendlich stand nicht nur das Modellieren im Vordergrund, sondern auch das gemeinsame Interesse an Mathematik und das gruppendynamische Arbeiten.

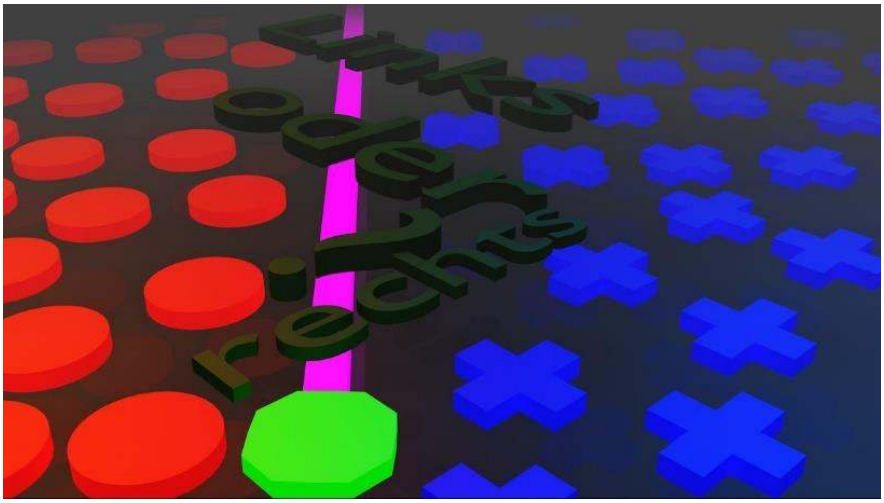
Woche der Modellierung mit Mathematik

Projekt: Informatik

Betreuer: BSc Richard Huber

Titel: Klassifikation, das Erkennen von Mustern und Zusammenhängen

Das menschliche Gehirn ist ein Meister im Erkennen von Strukturen und Zusammenhängen. So können wir beispielsweise ein Foto ansehen, und sofort wissen wir, was darauf zu sehen ist, obwohl wir dieses spezifische Foto noch nie gesehen haben. Das Ziel dieses Projekts ist es, grundlegende Methoden und Ansätze der Klassifikation zu finden und auszuarbeiten. Des Weiteren sollen diese Methoden im Matlab implementiert und an kleinen Beispielen getestet werden.



Teilnehmer:

Konstantin Andritsch
Franziska Harich
Florian Hübler
Max Schüßler
Fabian Tritthart
Simon Vasold

Inhaltsverzeichnis

1 Klassifikation

Was ist Klassifikation? Das ist eine Frage die sich die meisten wahrscheinlich noch nie gestellt haben, obwohl wir theoretisch tagtäglich damit konfrontiert werden.

Das Wort Klassifikation kommt vom Lateinischen. Dort entspricht das Wort *classis* dem deutschen Wort *Klasse*, während *facere* zu Deutsch *machen* bedeutet. Klassifikation beschreibt die Fähigkeit, Dinge unter Anwendung von Erfahrung in unterschiedliche Kategorien zu unterteilen. Wir Menschen sind Meister der Klassifikation. Aufgrund unserer Erfahrung ist es uns zum Beispiel möglich, ein Auto als solches zu erkennen, ohne genau dieses eine je zuvor gesehen zu haben. Wir erkennen ein Auto da wir die vier Reifen sehen, die Größe, möglicherweise hören wir das Auto sogar oder sehen dessen Bewegung. All dies weist uns darauf hin, dass wir es mit einem Auto zu tun haben und nicht etwa mit einem Fahrrad. Diese Fähigkeit ist allerdings nicht einzigartig auf unserem Planeten. Fast alle höheren Lebewesen können ebenfalls klassifizieren. Sie unterscheiden Artgenossen von Fremden, potenzielle Beute von Gefahren oder auch ihre Jungen von denen der anderen. Diese Kompetenz ist überlebenswichtig für sie. Jedoch ist anzumerken, dass immer sehr viel mehr Anfangsinformation als nachher abgefragte benötigt wird, um zuverlässige Ergebnisse zu liefern. So hört ein kleines Kind zum Beispiel lange Zeit seine gesamte Muttersprache, bevor es selbst beginnt, vollständige Wörter zu artikulieren.

In weiterer Folge ist Klassifikation allerdings auch ein Bereich der Informatik, da immer mehr und mehr versucht wird, dem Computer ebendiese Fähigkeiten anzutrainieren. In vielen Bereichen waren diese Bemühungen schon ziemlich erfolgreich, sodass nunmehr viele Geräte eine eingebaute Gesichtserkennung haben, ohne dass der Endnutzer die dahinterstehende Theorie auch nur annähernd erahnen kann. In anderen Bereichen wie zum Beispiel der Wildtierfotografie im Bereich der Naturforscher steht die Entwicklung solcher Mechanismen noch nahezu am Anfang und so beginnt die Maschine gerade erst, in diesen Fällen den Mensch zu ersetzen. Ein Beispiel in diesem Bereich ist jenes bei der Fotografie von seltenen Tieren. Bisher mussten Forscher tagelang im Unterholz sitzen, in der Hoffnung nicht genau in dem Moment eingeschlafen zu sein, in dem das entsprechende Tier vorbeikommt. Mittlerweile gibt es jedoch auch Mechanismen, bei denen eine Kamera mithilfe eines Bewegungsmelders ausgelöst wird, und dann auswertet, ob auf dem Foto das entsprechende Tier zu sehen ist, sollte dies der Fall sein, so werden weitere Fotos erstellt.

Dies läuft auf die heutzutage vielzitierte künstliche Intelligenz hinaus. Allerdings können sich Computer nichts von Grund auf selbst beibringen. Wenn sie mit genug Daten gespeist werden, können sie sich natürlich aufgrund von eigenen Erfahrungen verbessern, doch ganz ohne Informationen von außen, also vom Programmierer, funktionieren sie nicht.

Klassifikation kann bedeuten, dass die Gesamtmenge auf zwei Klassen aufgeteilt wird, jedoch kann es genauso meinen, dass viele verschiedene Klassen existieren oder aber auch, dass die vorhandenen Informationen über ein hierarchisches System gegliedert werden. In Grunde genommen ist also Klassifikation einfach das Einteilen von Informationen, Daten und (Verhaltens-)Mustern.

2 Zielsetzung und Ideen

Ziel dieses Projektes war es in erster Instanz, aufgrund von eigenen Ideen Methoden zu entwickeln, mithilfe derer der Computer in der Lage sein soll, aufgrund von vorgegebenen Punkten weitere Punkte einer von zwei Kategorien zuzuordnen. In weiterer Folge konnte auf bereits existierende Methoden zurückgegriffen werden, diese mussten jedoch erst selbstständig in Matlab programmiert werden, wodurch die Ergebnisse immer mehr verschiedene Punktemengen abdecken konnten. Als weitere Zwischenetappe war das Ziel gesteckt, die Punkte in mehr als zwei Klassen einzuteilen. Das Endziel war es, den Inhalten von Bildern gewisse Klassen zuzuordnen. Dazu wurden beispielsweise die Bilder von Tieren und Pflanzen verglichen und versucht den entsprechenden Klassen zuzuordnen.

Es gab folgende Ideen:

- Nähe (dem/den nächsten Punkt/Punkten entsprechend einteilen)
- Muster (erkennen von Kreisen, Dreiecken und anderen geometrischen Formen)
- Grenzen (in Form einer Geraden oder Kurve)
- Flächen (innerhalb welcher alle Punkte dieselbe Klasse besitzen)
- konvexe Hülle (das Finden der konvexen Hülle aller Punkte einer Klasse)

3 Erste Versuche

3.1 Nächster Punkt

Eine der einfachsten Umsetzungsmöglichkeiten einer Klassifikation besteht darin, einem zufälligen Punkt die Gruppe des am nächsten liegenden Ausgangspunktes zuzuweisen. Hierzu werden zuerst die Distanzen zwischen dem Punkt und allen Ausgangspunkten berechnet. Die Gruppe wird nun von jenem Ausgangspunkt bestimmt, welcher die kleinste Distanz zum neuen Punkt aufweist. Diese Methode weißt mit ihrer leichten Umsetzung einen großen Vorteil auf. Auch in ihrer Effizienz unterscheidet sie sich nicht grob von komplexeren Methoden. Der größte Nachteil besteht allerdings darin, dass bereits eine kleine Anzahl an Punkten das Ergebnis des Algorithmus stark beeinflussen kann. So unterscheiden sich die beiden unten aufgeführten Beispiele um nur vier Punkte, weisen aber einen großen Unterschied in Hinsicht auf blaue und rote Flächen auf.

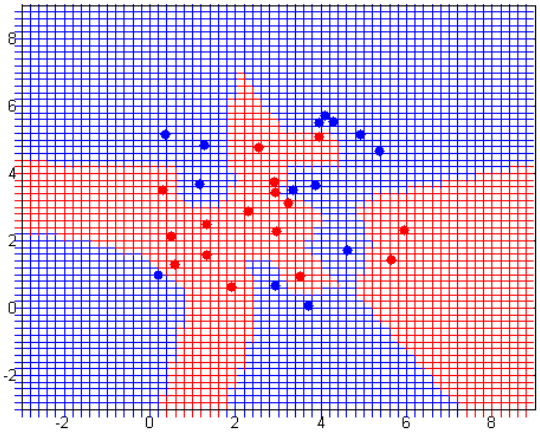


Abbildung 1: Beispiel 1

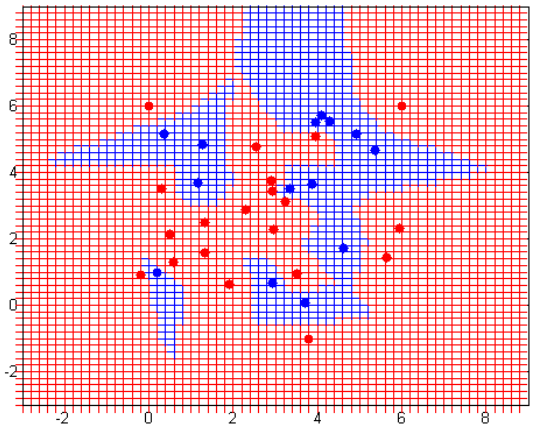


Abbildung 2: Beispiel 1 mit neuen Punkten

3.2 Schwerpunkt/Radius

Eine Idee war es, sich jenen Punkt auszurechnen, der die Durchschnittlichen X- und Y-Koordinaten aller blauen und roten Punkte als Koordinaten hat. Von diesem Mittelpunkt aus wurden alle Punkte in einem gewissen Radius eingefärbt, bevor rund um alle nicht in diesem Radius liegenden vorgegebenen Punkte wieder eingefärbt wurde. In diesem zweiten Schritt jedoch war der Radius nur ein Fünftel dessen, der beim Kreis um den Mittelpunkt verwendet wurde.

In Abb.1 sieht man die Punkte von denen ich ausging. In Abb.2 dagegen die Ergebnisse die ich mit meiner Methode erzielte. Es ist an dieser Stelle notwendig zu erwähnen, dass die Grafiken unterschiedlich formatiert sind.

Abb.1

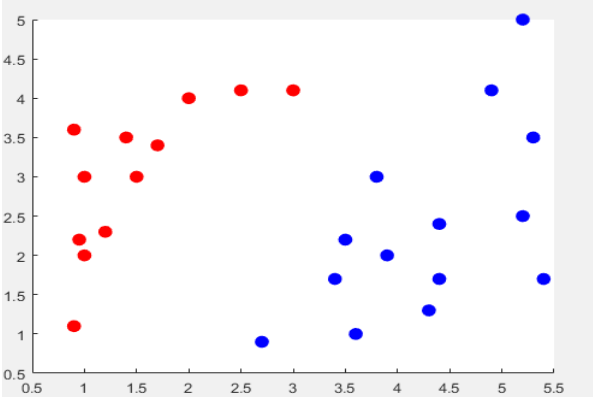
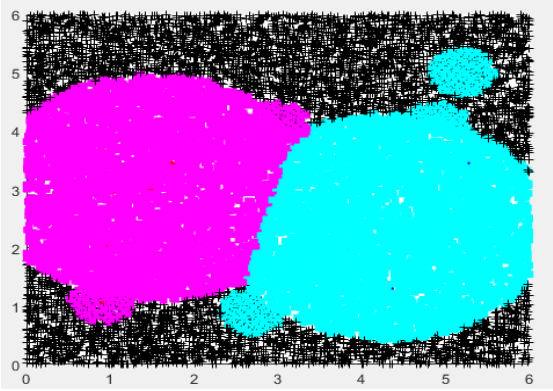


Abb.2



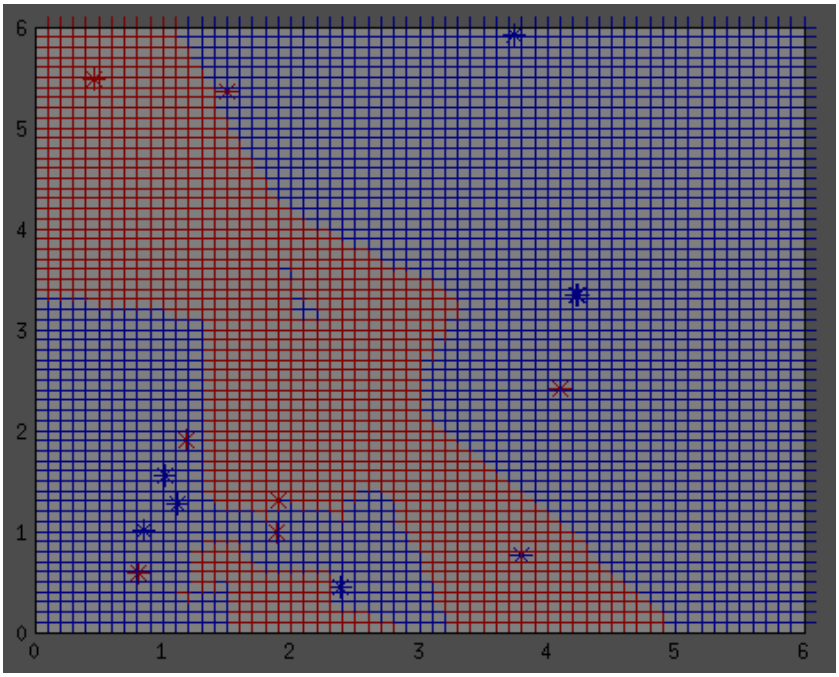
Wie in der Abbildung ersichtlich werden nicht alle Punkte einer Klasse zugeordnet. Dies soll insofern sinnvoll sein, als dass es möglichen Fehlern vorbeugen soll. Das geschieht, da ein vollständiges Ausfüllen des gesamten dargestellten Bereichs nicht zielführend wäre, denn mit steigender Entfernung sinkt die Präzision dieser Methode. Damit wurde eine Maßnahme gesetzt, um diese Methode resistent gegen Ausreißer zu machen. Die Idee für den Versuch mit dem Schwerpunkt entstand eben auch aufgrund der möglichen Ausreißer.

3.3 Die Nächsten Fünf Punkte

Der nächste Schritt nach der Sortierung bezüglich des nächstgelegenen Punktes war laut unseren Ideen eine Einteilung anhand der fünf am nächsten gelegenen Punkte. Das heißt der zu klassifizierende Punkt gehört zu der Menge, zu der die Mehrheit der fünf nächstgelegenen gehört. Dadurch zählt, zum Beispiel, ein Punkt, obwohl der nächste Punkt blau ist, zur roten Menge, da vier der nächsten fünf rot sind.

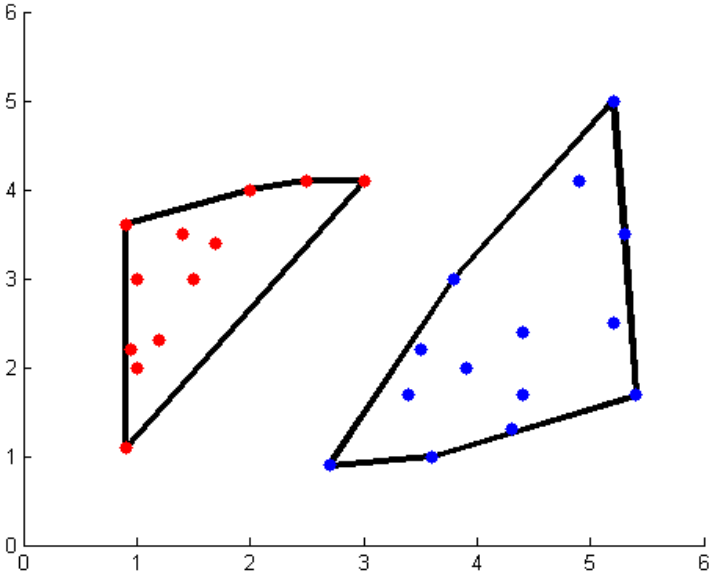
Diese Methode ist der vorherigen vorzuziehen, wenn in einer Region des Bildes, in der es allgemein deutlich mehr rote Punkte gibt, ein blauer Ausreißer existieren. Dieser würde bei

der ersten Variante das Bild stark beeinflussen, da sich alle Punkte in näherer Umgebung ebenfalls blau färben würden, obwohl es eigentlich ein „von rot bestimmter“ Bereich ist.



3.4 Konvexe Hüllen

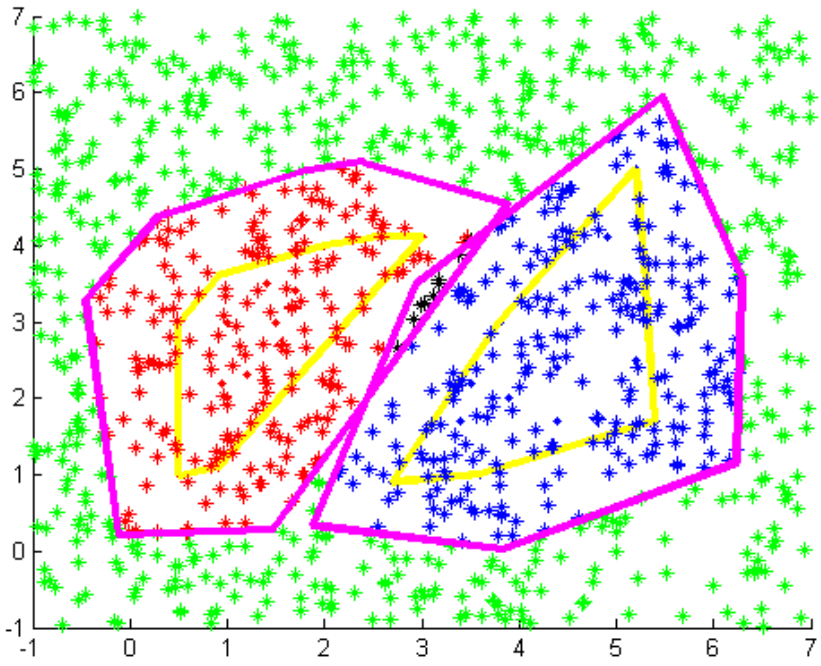
Eine weitere Idee zur Einteilung der Punkte in der Ebene ist das Erstellen der konvexen Hülle aller Punkte einer Klasse. Die Definition der konvexen Hülle ist die kleinste konvexe Menge, welche die gesamte Ausgangsmenge enthält. Einfach gesagt die kleinste Menge, die jede Verbindungsstrecke zweier Punkte dieser Menge enthält. So lässt sich beispielsweise die Form einer Punktemenge gut darstellen, und allen Punkten die innerhalb dieser Hülle die entsprechende Klasse zuweisen.



Beispielbild zweier konvexer Hüllen für die rote, sowie die blaue Punktemenge.

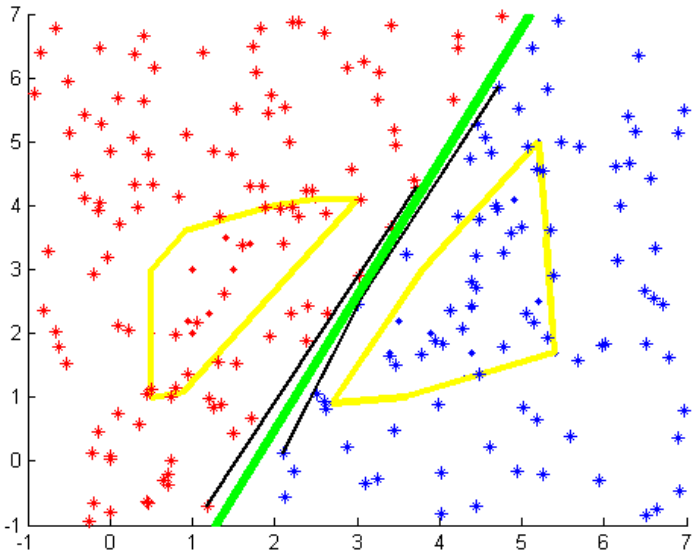
Somit ist sowohl die beiden Punktemengen völlig vom restlichen Raum abgetrennt. Ein mögliches Entscheidungsproblem entsteht in dem Fall, dass sich die Beiden, oder möglicherweise mehr als zwei konvexe Hüllen überlappen, und damit Punkte innerhalb beiden Mengen liegen.

Da die vorhandenen Daten nicht alle Fälle abdecken, gibt es keinen Grund anzunehmen, dass die konvexe Hülle der Daten der konvexen Hülle der Grundmenge entspricht. Daher wollen wir die von uns berechneten Hüllen vergrößern. Dazu wurde jeder Eckpunkt der konvexen Hülle auf der Winkelsymmetrale des Winkels, den er mit den beiden nächststehenden Eckpunkten der konvexen Hülle einschließt, um einen konstanten Faktor verschoben. Dadurch entsteht eine gestreckte, vergrößerte Version der vorherigen konvexen Hülle, wie in der folgenden Abbildung zu erkennen ist. Die gelben Linien stellen die normale konvexe Hülle dar, und die in Magenta gehaltenen Linien die vergrößerte.



Die Grafik zeigt 1000 getestete Daten, welche als rote Sterne dargestellt werden, sollten sie innerhalb der vergrößerten konvexen Hülle der roten Punkte liegen, und analog für die blauen Punkte. Alle restlichen Punkte die außerhalb liegen sind unbestimmt und daher grün gefärbt. In der Mitte des Bildes gibt es einige überbestimmte Sterne, also Sterne die aufgrund ihrer speziellen Position zu mehr als einer Klasse gezählt werden können. Diese überbestimmte Datenpunkte sind schwarz dargestellt. Das zeigt auch die Schwierigkeit dieser Methode, dass nicht für jeden Punkt eine eindeutig bestimmbare Klasse existiert.

Eine weitere durchgeführte Methode der Klassifikation mithilfe der konvexen Hülle zeigt die folgende Abbildung. Bisher wurde immer nur ein Teilbereich der Ebene klassifiziert und viele Punkte hatten keine bestimmte Klasse. Mit dieser Methode gelingt es den gesamten Raum in zwei Klassen einzuordnen.



In obiger Grafik, zeigt die grüne Linie eine Gerade, die die Ebene in zwei Bereiche trennt. Wie zu sehen, gehören alle Punkte, die rechts von ihr liegen zur blauen Klasse, und die anderen zur roten. Die grüne Trennlinie entsteht aus der Lage der schwarzen Linien. Diese sind jene Linien, die Teil der konvexen Hülle der Klassen sind und ihrerseits den Raum in blaue und rote Punkte trennen. Somit liegen für die schwarze Linie auf der linken Seite der grünen Linie alle roten Punkte links und alle blauen rechts von ihr. Um die grüne Linie zu bestimmen wird der Mittelpunkt aller Punkte, die Anfangs- oder Eckpunkte einer schwarzen Linie sind, gebildet und die mittlere Steigung aller schwarzen Linien errechnet. Somit ergibt sich ein Punkt und eine Steigung, dadurch lässt sich eine Gerade eindeutig festlegen.

3.5 Angepasste Konvexe Hüllen

Wenn mithilfe der Konvexen Hüllen Formen erkannt werden sollen, können Punkte, welche eine leichte Abweichung aufweisen, ein Problem darstellen. Anstatt eines Dreiecks würde ein Vier- oder gar Fünfeck erkannt werden, wenn nur ein oder zwei Punkte geringste Abweichungen zur Konvexen Hülle aufweisen. Um dieses Problem zu beheben, wäre es eine mögliche Methode, solche Punkte als Ausreißer zu sehen und somit für die Konvexe Hülle ignorieren. Hierzu wird zuerst die Konvexe Hülle aller Punkte gebildet. Anschließend werden die Entfernungen der Hüllenpunkte zu den Hüllengeraden berechnet. Unterschreitet eine dieser Entfernungen einen gewissen Schwellwert, so wird er aus der Konvexen Hülle ausgeschlossen und der Vorgang beginnt erneut. Dies wird wiederholt, bis es keinen Punkt mehr gibt, dessen Entfernung zu einer Gerade der Hülle kleiner als der Schwellwert ist.

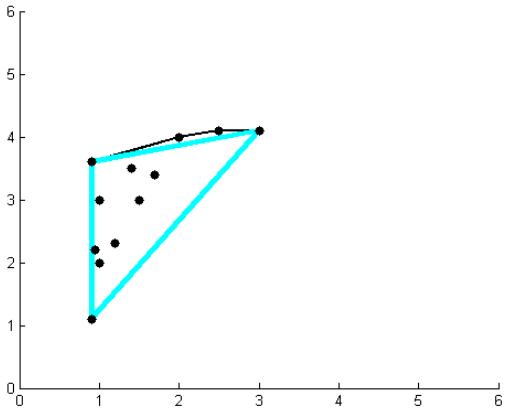


Abbildung 3: Beispiel adjustierte Konvexe Hülle

Anhand dieser Punkte ist es nun einfacher und auch sicherer, Formen zu bestimmen. Hierzu kann das Verhältnis zwischen Seitenlängen und Umfang, Fläche und Seitenlängen etc. hergenommen werden.

3.6 Mittelpunkte

Eine der ersten Ideen um zur Bestimmung der trennenden Gerade zu ermitteln war jene, dass man zu jedem der Punkte jenen Punkt der anderen Klasse sucht, der am wenigsten weit entfernt ist. Wenn man nun diese Verbindungen hat, kann man deren Mittelpunkte verbinden. Hat man nun diese Mittelpunkte so kann man sie auf verschiedene Arten zu verbinden versuchen. Aufgrund einer Skizze war ein Ansatz, den dem Ursprung nächsten und den am weitesten entfernten dieser Mittelpunkte zu verbinden, wobei diese Idee sogleich wieder verworfen werden musste, da bei genauerer Betrachtung klar war, dass sie nur für diese besondere Punktemenge funktionieren würde. Als nächstes entwickelte sich die Idee, dass die Verbindung der beiden kürzesten dieser Verbindungsgeraden eine Gerade aufspannt, die alle Punkte trennt. Leider stellte sich diese Idee als Irrweg heraus, da die beiden kürzesten Geraden sehr nah zueinander lagen, und selbst wenn es für diese eine Punktemenge funktioniert hätte, wäre es mit höchster Wahrscheinlichkeit nicht schwer gewesen, eine Punktemenge zu finden, für die diese Methode zu keiner Lösung führt.

4 Gradientenverfahren

4.1 Gradient

Der Gradient ist ein Vektor, welcher aus den partiellen Ableitungen einer Funktion in mehreren Variablen besteht. Man definiert den Gradienten von f(x,y) folgender Maßen:

$$von\ der\ Funktion\ f(x,y) ist\ der\ Gradient\ \nabla f = \begin{pmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{pmatrix}$$

Eine wichtige Eigenschaft dieses Gradienten, welche wir uns zu nutzen machen, ist, dass er immer in die Richtung des stärksten Anstieges zeigt.

4.2 Verfahren

Man nimmt an, es ist die Funktion $f(x,y)$ gegeben, welche konvexe Eigenschaften besitzt. Nun möchte man das Minimum herausfinden, also die Stelle an der $\frac{\delta f}{\delta x} = 0$ und $\frac{\delta f}{\delta y} = 0$ ist (Extremstelle), um die minimalen Werte von f zu finden. Das Gradientenverfahren wird nun zur Annäherung an diese Stelle genutzt. Dabei macht man davon Gebrauch, dass der Gradient in Richtung des maximalen Anstiegs zeigt, was heißt, dass $(\nabla f * (-1))$ eine Abstiegsrichtung ist, und damit zum Minimum führt.

Mit dieser Funktion kann man sich dann annähern.

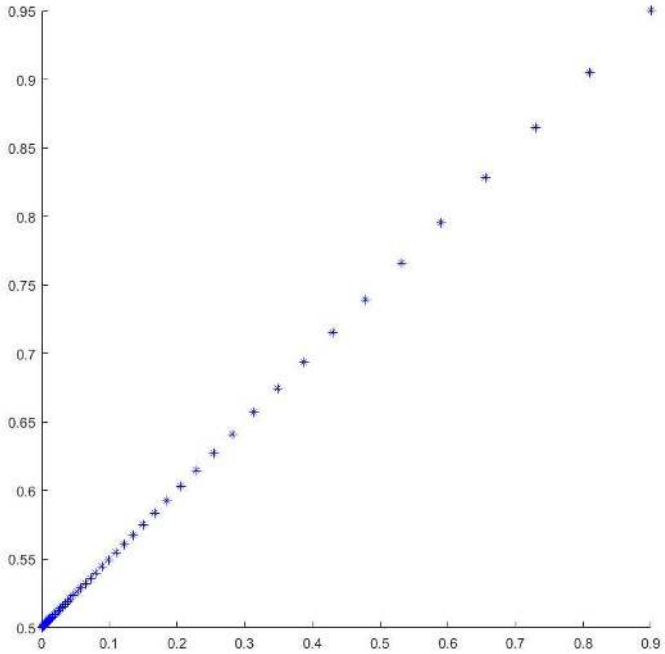


Abbildung 1, Darstellung der Annäherung an das Minimum von einer f(x,y) Funktion mit dem Verfahren

$$\mathrm{X} = \mathrm{X} + \alpha * \nabla f^T$$

α…Schrittkonstante, gibt die Stärke an, mit der man sich an das Minimum annähert.

Man sollte bei der Auswahl von α entweder eine Funktion zum Berechnen verwenden, oder unbedingt zumindest beachten, dass der Wert nicht zu groß gewählt wird. Denn sonst kann der Fall eintreten, dass man den gegenteiligen Effekt einer Annäherung erzielt: Man kann sich vorstellen, man geht, ohne dass man weiß wo das Minimum ist und ab wann die Funktion wieder Steigt die Funktion entlang. Wenn nun α zu groß ist, gelangt man nicht näher an das

Minimum, sondern „wandert“ auf der anderen Seite wieder hinauf. Dies „schaukelt“ sich dann jede Iteration weiter hinauf, wodurch es für unsere Verwendungen nicht sinnvoll wäre.

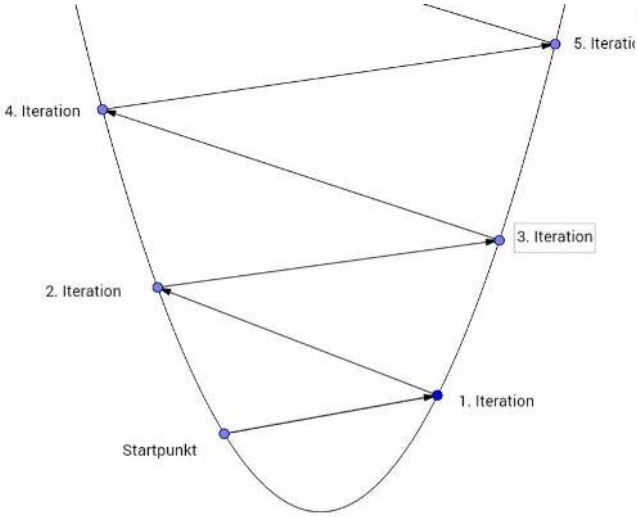


Abbildung 2, Skizze einer Annäherung an das Minimum auf einer Funktion mit einem zu hohen a-Wert

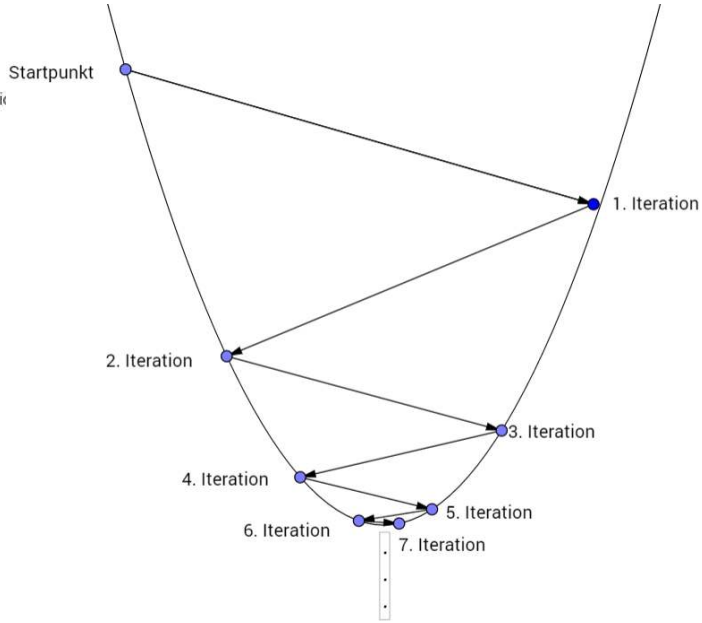


Abbildung 3, Skizze einer Annäherung an das Minimum auf einer Funktion mit einer richtig gewählten a-Konstanten

4.3 Armijo-Goldstein-Ungleichung

Man möchte nun aber immer eine Konvergenz zum Minimum bekommen, das ist allerdings mit einem, für α jeder Iteration konstant bleibenden Wert nicht garantiert. Daher kann man mit dieser Ungleichung eine Menge definieren, welche α nicht erlaubt zu klein, oder zu groß sein. Zu groß ist es dann, wenn es zwischen beiden Seiten von der Funktion „hin und her“ hüpft und so sich nicht mehr annähert, zu klein dann, wenn man zu kleine Sprünge macht und so der Algorithmus sehr langsam wird.

Der Algorithmus funktioniert, in dem am für jede Iteration des Gradientenverfahren einen neuen Wert für α definiert, welcher abhängig vom x ist.

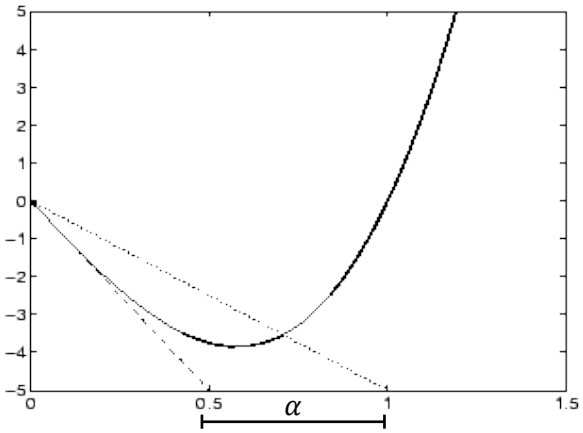


Abbildung 2, grafische Darstellung der Ungleichung.
Der markierte Bereich auf der x-Achse ist die Menge, in welcher sich a befinden muss.

Ungleichung:

$$f(x_k) + (1-c)\alpha_k \nabla f_k^T p_k \leq f(x_k \alpha_k p_k) \leq f(x_k) + c\alpha_k \nabla f_k^T p_k$$

$f(x_k)$…Funktion, an welcher das Gradientenverfahren angewendet wird
c…Eine wählbare Konstante. Bedingung: $0 < c < \frac{1}{2}$
p_k…Gibt die Richtung zur Annäherung an, im Fall des Gradientenverfahrens ist dies $p = -\nabla f_k(x_k)$

5 Support Vector Machine

Eine Support Vector Machine dient als eine Methode der Klassifizierung dazu, eine Menge an Objekten in Klassen zu teilen. Durch die Eigenschaft, dass zwischen den Grenzen und den Klassen ein möglichst breiter leerer Bereich vorliegt, wird die Methode auch als ein *Large Margin Classifier* klassifiziert. Im Gegensatz zu ihrem Namen ist die Methode jedoch keine physikalische Maschine, sondern ein rein mathematisches Verfahren.

5.1 Linear

Als Ausgangsposition für den Algorithmus dient eine Menge an n – dimensionalen Daten, für welche die Klasse bekannt ist. Aufgabe der Methode ist es nun, eine Hyperebene zu finden, welcher die beiden Klassen mit einem möglichst großen Abstand voneinander abgrenzt. In der Umsetzung dient hierzu die Bedingung, dass γ, der minimale Abstand einer Klasse zur Hyperebene, sein Maximum anstrebt. Daten welche weiter als γ von der Hyperebene entfernt sind, spielen hierbei keine Rolle. Sie wird lediglich von den beiden, ihr am nahe liebendsten, Daten definiert. Durch die Auffassungsmöglichkeit als Vektoren werden sie daher auch Stützvektoren genannt. Den Stützvektoren – oder auch support vectors – verdankt die Methode ihren Namen. Da es jedoch nicht möglich ist, eine Hyperebene zu „verbiegen“, ist die Methode nur dann erfolgreich, wenn eine lineare Trennung möglich ist. Um eine Non – Lineare Trennung durchzuführen, sind weiter mathematische Vorgänge nötig. Diese werden in späteren Kapiteln der Dokumentation bearbeitet.

Zu Beginn müssen nun die jeweiligen Ausgangsdaten – oder auch Trainingsdaten genannt – für den Algorithmus angepasst werden. Hierzu wird jedem Datensatz je nach der zugehörigen Klasse -1 oder 1 zugeschrieben. Um die Lesbarkeit der Dokumentation nicht negativ zu beeinflussen, werden die beiden Klassen von nun an als Klasse 1 und Klasse 2 bezeichnet. Die Trainingsdaten sind wie folgt aufgebaut:

$$Daten_i = \{(x_i, y_i) | i = 1, \dots, m;\ y_i \in \{-1,1\}\}$$

Wobei y_i die jeweilige Klasse der Daten ist.

Anhand dieser Trainingsdaten wird nun jene Hyperebene berechnet, welche beiden Daten mit einen möglichst großem Margin trennt. Sie wird durch den Normalvektor ***w*** und dem Bias ***b*** definiert. Anschließend kann aus diesen Parametern eine Entscheidungsfunktion $y_i =$

$sgn(\langle w, x_i \rangle + b)$ definiert werden, mit welcher für jeden Datensatz x_i bestimmt werden kann, zu welcher Klasse er gehört. Insgesamt gilt also:

$$\min(\|w\|_2^2)$$

$$y_i(\langle w, x_i \rangle + b) \geq 1 \text{ für alle } i \in \{1, \dots, m\}$$

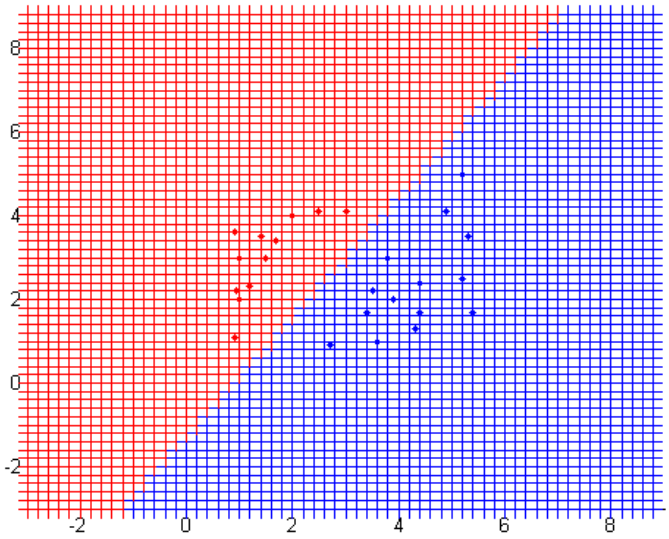


Abbildung 4: Beispiel Lineare Support Vector Machine

5.2 Nonlineare Support Vector Machine

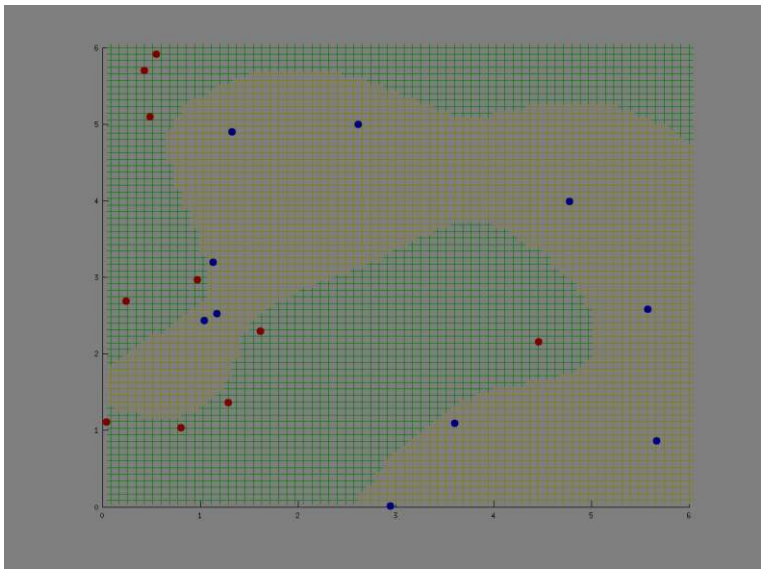
Das lineare Teilen funktioniert bei zwei Mengen, die mit einer Gerade getrennt werden können sehr gut, doch sobald sie sich überlappen ist es nicht mehr möglich.

Um sich überlappende Mengen zu trennen muss man die Daten in einem mehrdimensionalen Raum abbilden. Wobei der neue Raum immer mehr Dimensionen haben muss, als der anfängliche. Bei Support Vector Machines ist es sehr einfach diese Erweiterung einzubauen, da alle Datenpunkte nur in Skalarprodukten vorkommen. Dadurch kann man diese Skalarprodukt mit einer sogenannten Kernelfunktion ersetzen. Kernels werden in diesem Bereich von Algorithmen verwendet um ihre Berechnungen in einem höherdimensionalen Raum auszuführen. Das heißt man kann die Hyperebene in einem solchen Raum berechnen und dies dann später auf den, zum Beispiel, zweidimensionalen Raum anwenden.

$f_{pred} = \frac{1}{\gamma} \cdot (\sum \mu_i \cdot y_i \cdot k(x_i, x)) - b$ …ist die Funktion mit deren Werte man beurteilen kann zu welcher Klasse der Punkt x gehört. Wobei k die Kernelfunktion ist.

Durch die Verwendung der Kernelfunktionen ist es möglich die Support Vector Machine auf allgemeine Strukturen, wie später in dieser Dokumentation beschrieben, anwenden.

Da mit bestimmten Kernelfunktionen (Bsp. Gauß'sche Kernelfunktion) der Transfer in einen unendlichdimensionalen Raum möglich ist, lässt sich die Ebene mit jeder Anordnung an eindeutig klassifizierten Punkten in ihre jeweiligen Klassen unterteilen. Die von Matlab erhaltenen Ergebnisse sind teilweise nicht richtig, sollte die Berechnungen, aufgrund ihrer Komplexität zu lange dauern.



Die obige Abbildung zeigt die Unterteilung der Ebene, wobei alle Punkte im gelben Bereich zur Klasse der blauen Punkte gehört bzw. alle im grünen Bereich zur der Klasse der roten Punkte.

5.3 Anwendung bei mehreren Klassen

Zum Einteilen bei mehreren Klassen haben kann man sozusagen die „One vs All“-Strategie anwenden. Dabei wird die Position des Punktes ergründet, indem man seine Lage relativ zu jeder Klasse einzeln mit den anderen Klassen zusammengefasst vergleicht. Das heißt, wenn man die Klassen 1,2 und 3 hat, wird zuerst die Lage des Punktes relativ zur Klasse 1 und den anderen zwei, die sozusagen zusammen die Klasse -1 bilden, bewertet, danach zur Klasse 2 bzw. den anderen zwei zusammen und so weiter. Danach kann man die Werte für alle drei Vergleiche wiederum miteinander vergleichen und demnach beurteilen zu welcher Klasse der Punkt gehört

5.4 Bildererkennung

Mit der oben beschriebenen Methode der Non-Linear Support Vector Machine ist es auch möglich viel komplexere Elemente in verschiedene Klassen einzuteilen. Ein Beispiel dafür sind Bilder von Zahlen. Allerdings in sehr vereinfachter Form von 50x50px bis 200x200 Pixel. Für die Einteilung wurden einige Bilder von Nullen, Einsen, Zweien, Dreien und Vieren in den beiden Bildbearbeitungsprogrammen Gimp und Paint erstellt. Außer an den Grenzzonen, bestanden diese Grafiken nur aus schwarzen und weißen Pixeln, Farbe spielte also keinerlei Rolle. In den Vorherigen Beispielen wurde lediglich die Klassifikation im zweidimensionalen Raum betrachtet, doch steht nun jedes einzelne Bild für einen Punkt, so hat dieser Punkt keine X- und Y-Koordinate, sondern hat so viele Dimensionen wie er Pixel besitzt (wobei ein Pixel

eben nur als Graustufe angesehen wurde, und daher nur einen Wert anstatt der üblichen drei Farbwerte hat). Somit handelt es sich bei Bildern um einen Raum, der viel mehr Dimensionen besitzt. So wie die Ausgangspunkte im Vorhinein in verschiedene Klassen eingeteilt wurden müssen auch die Referenzbilder in Klassen eingeteilt werden. Es bietet sich an, allen Bildern die eine 0 darstellen der Klasse Null zuzuordnen und allen anderen Bildern die demnach entsprechenden Klassen.

Beispiele zweier verwendeter Bilder

Die Ergebnisse zwischen den beiden Klassen 0 und 1, sowie 1 und 2 sind sehr zufriedenstellend. Das liegt wohl vor allem an der großen Differenz zwischen den beiden Zahlen. Getestet wurde mit der Referenz von jeweils 40 Beispielbildern und 15 zu testenden Bildern pro Klasse. Jedes zu testende Bild wurde mit jedem Bild verglichen, dessen Klasse bekannt war. Der Testgrafik wurde die Klasse zugewiesen die den besten Übereinstimmungsgrad lieferte. Unter den 30 getesteten Bildern befanden sich lediglich drei Bilder, denen die falsche Klasse zugeordnet wurde. Die Fehlerquote liegt also bei diesem Versuch bei 10%, jedoch ließe sich dieses Ergebnis mit einer Vielzahl an Referenz Bildern um einiges verbessern. Die Klassifikation zwischen Zweiern, Dreiern und Vierern stellt sich schwieriger dar. Der Grund dafür ist einerseits die zu große Ähnlichkeit der Form der Zahlen (Sowohl 2 und die 3 besitzen beide eine Rundung in der oberen Hälfte.), andererseits spielt auch hier die geringe Anzahl an Vergleichsbildern eine große Rolle.

Als Steigerung dazu ist interessant, wie sich das Programm bei noch komplexeren Bildern(Objekten) verhält. Dafür ist allerdings von Nöten die Anzahl an Ausgangsbildern zu erhöhen. Diesbezüglich wurden dem Algorithmus diesmal 60 Bilder von einer Rose und 60 Bilder von einem Elefanten verschiedenster Größen und Arten zur Verfügung gestellt, wovon jeweils 5 zum Testen waren. Hierbei ist das Ergebnis leider nicht gut, nur etwas mehr als die Hälfte der Bilder wurden deren richtiger Klasse zugeordnet. Vermutlich liegt das daran, dass die Bilder bevor sie von der Non-Linear Support Vector Machine analysiert werden, in Graustufen umgewandelt werden, um die Dauer des Programms zu verkürzen. Dadurch geht aber

einiges an Information verloren, und die Bilder sind teilweise sogar für den Menschen nicht mehr zwischen den Objekten „Rose“ und „Elefant“ differenzierbar.

6 Graphical User Interface

In Matlab hat man auch die Möglichkeit mithilfe des Graphical User Interface Development, Enviroment, Graphical User Interfaces (GUI) zu erstellen. Dadurch ist es möglich die Verwendung von Programmen zu erleichtern, man kann z.B. bestimmte auftretende Phänomene besser untersuchen.

Wir implementierten mehrere unserer Programme in Kombination mit einer GUI.

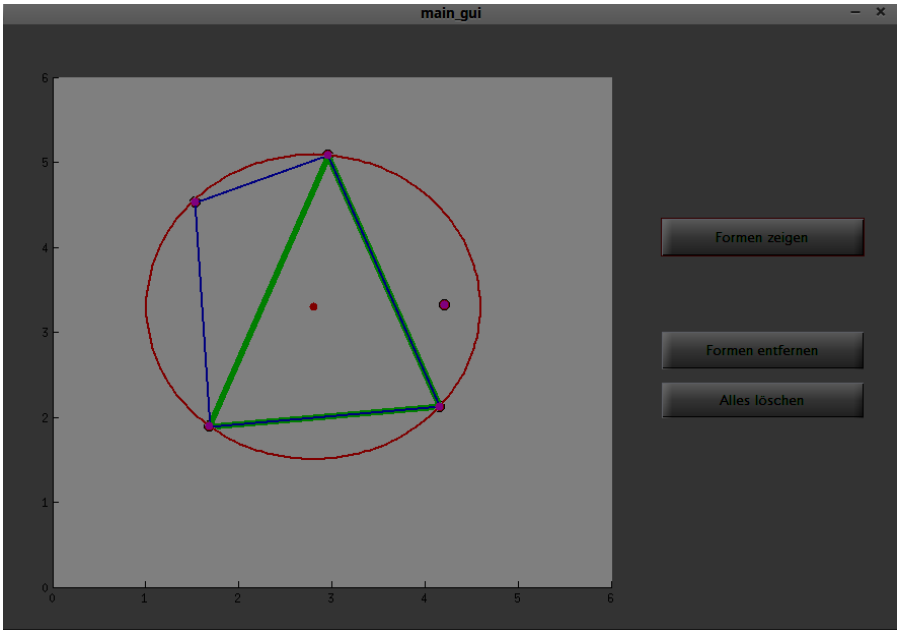
Die ursprüngliche Motivation etwas mit einer GUI umzusetzen basierte darauf ganz einfach Punkte auf einem Graphen zu setzen und mit diesen verschiedene Berechnungen durchzuführen.

Anfangs ergaben sich ein paar Probleme beim Verständnis der Logik, die dahinter steckt, da es teilweise natürlich abweicht von der normalen Verwendung Matlabs.

Zuerst schrieben wir ein Programm zur Formenerkennung in den gesetzten Punkten. Der Hauptteil bestand schon davor, da wir ein sehr ähnliches für zufällige Punkte schrieben. Doch durch die Möglichkeit die Punkte händisch mit einem Mausklick zu setzen erleichterte die Analyse der Implementierung immens.

Die Funktionen beschränken sich auf das Punkte einzeichnen, Formen anzeigen, Formen entfernen und das Löschen aller Variablen und Einzeichnungen in der Anzeige.

Andererseits konnten wir auch die Support Vektor Machine zusammen mit einer graphischen Oberfläche umsetzen. Dabei kann man ebenfalls die Punkte selber setzen und die Farbe dieser in einem Drop-Down-Menü wählen, wenn man alle gesetzt hat kann man ein Gitter darüber zeichnen lassen, welches die jeweiligen Farben annimmt. Man hat auch die Möglichkeit im



Nachhinein noch Punkte hinzuzufügen um so die Struktur zu verändern.

Risikomanagement

Betreuer:

Sebastian Engel, M. Sc.

Gruppenmitglieder:

Jan Stalzer
Sebastian Negovec
Anna-Sophie Posch
Elias Götz
Viktoria Haberl
Simon Nitsch

Inhaltsverzeichnis

Einleitung

Die 13. Woche der Modellierung mit Mathematik fand vom 04.02.2017 bis zum 10.02.2017 in der Jugendherberge Pöllau statt. Bearbeitet wurden die Themengebiete Kohlenstoffkreislauf, Klassifikation, Beobachtungsvoraussage durch Strahlenverfolgung, Musterbildung durch Reaktion und Diffusion sowie Risikomanagement.

Wir, das sind Jan Stalzer, Sebastian Negovec, Anna-Sophie Posch, Elias Götz, Simon Nitsch und Viktoria Haberl, befassten uns unter der Betreuung von M.Sc. Sebastian Engel mit dem Thema Risikomanagement von Versicherungsgesellschaften. Eine Hauptaufgabe von Versicherungsgesellschaft ist die Bereitstellung von genügend Kapital zur Deckung von Aufwendungen durch eingetretene Schäden. Die hierbei möglichen Schadensarten sind vielseitig und können von Naturkatastrophen, Cyberkriminalität oder unternehmensinterner Kriminalität bis hin zu fremd- oder eigenverschuldeten Einbrüchen des Versicherungsportfolios am Finanzmarkt reichen.

Im Laufe der Woche war es unser Ziel, ein mathematisches Versicherungsmodell vorzustellen, welches zur Prognostizierung von fest definierten Schadenskategorien verwendet werden sollte. Die Verwendung von Matlab und den dort integrierten statistischen Funktionen halfen uns, Simulationen und daraus folgende modellabhängige Prognostizierungen von zukünftigen Schäden zu erhalten. Unsere Modellparameter resultierten dabei aus vergangenen Schadensdatenbanken, die einer realistischen Risikoeinschätzung zukünftiger Schäden dienten. Das Ziel war es, nicht nur unabhängige Schäden, sondern auch korrelierende Schadenskategorien und Schadenhöhen zu simulieren.

1 Problemstellungen

Zuallererst galt es, Erkenntnisse über die Arbeitsweise einer Versicherung zu gewinnen und die Bedeutung hinter dem Begriff Risk Management zu verstehen. Eine Versicherung hat verschiedene Schadenstypen einzuschätzen und entsprechend Geldmittel zurückzustellen. Die Hauptaufgabe eines Risk Managers ist es, mittels Daten aus der Vergangenheit zukünftige Schadenshäufigkeiten und Schadenshöhen zu prognostizieren.

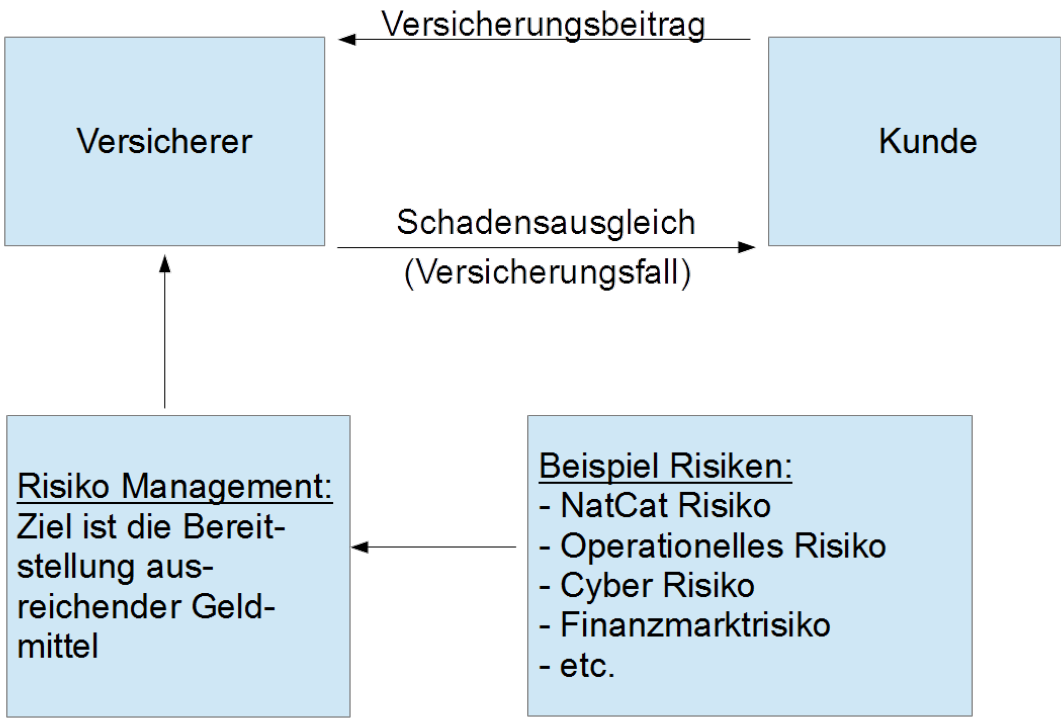


Abbildung 1: Arbeitsweise einer Versicherung

Ziel der Versicherung ist es folglich, die vorbereiteten Datensätze einer jeweiligen Kategorie für die Simulation zukünftiger Schäden zu verwenden. Einfach formuliert:

$$\left(\begin{matrix}\text{Kategorie spezifischer}\\ \text{Datensatz}\end{matrix}\right) + \left(\begin{matrix}\text{Modell Festlegung für}\\ \text{Schadenshöhe und Schadensanzahl}\end{matrix}\right) \Rightarrow \underbrace{\left(\begin{matrix}\text{Modell Parameter Justierung}\\ \text{via Datensatz}\end{matrix}\right)}_{\Rightarrow \text{Simulation und Rückstellung}}$$

Es ist mit einigen, im weiteren Verlauf dieses Berichts erwähnten, Rechenoperationen und Befehlen in Matlab möglich, Schadensanzahl und Schadenshöhe zu simulieren. Interessant wird es allerdings, wenn man beachtet, dass einige Schadenskategorien miteinander korrelieren, sich also gegenseitig beeinflussen. So kann beispielsweise durch einen Schaden in der Kategorie Erdbeben ein weiterer Schaden in der Kategorie Tsunami entstehen oder ein Finanzmarktschaden durch einen Schaden im Bereich der Cyberkriminalität ausgelöst werden.

Die zweite Problemstellung beinhaltete also das Schreiben eines Algorithmus, der ebenjene Korrelation berücksichtigt, um damit realistischere Datensimulationen durchzuführen. Hierzu ist es notwendig, die Datensätze anzugleichen, um eine gewisse Vergleichbarkeit zu schaffen. Funktioniert der Algorithmus für die Korrelation zweier Schadenskategorien, so stellte sich die Frage, wie man eine Simulation von Schäden für beliebig viele Kategorien entwickeln könnte. Das zu bewerkstelligen, war die dritte Problemstellung dieser Woche.

Letztendlich war unser Ziel, die entwickelten Algorithmen an einer eigenen Versicherung zu testen, um so eine Simulation in der wirklichen Versicherungswelt nachzustellen um einen Einblick in diese Welt zu bekommen und das theoretische Wissen auch in der Praxis anzuwenden und zu überprüfen.

2 Arbeitsweisen

Am Anfang programmieren wir mithilfe von MATLAB einen Algorithmus für die Simulation von Versicherungsschäden pro Jahr. Zur Modellierung dieser Jahresschäden ziehen wir einerseits die Poisson-Verteilung, ein Modell für die Darstellung der Schadenshäufigkeit, und andererseits die logarithmische Normalverteilung, welche die Schadenshöhe der Daten veranschaulicht, zu Hilfe.

Eingangs legen wir einen fixen Zeithorizont fest, um somit unsere Schadenskategorie näher zu definieren. Wir simulieren 10 000 (=n) beliebige Jahre, sodass jeder Datenpunkt einem Jahr entspricht. Nun werden wir die Schadenshäufigkeit mit einer diskreten Verteilung darstellen. Dabei ist die Anzahl der Schäden im Intervall [0;∞) festgelegt und unsere Schadensuntergrenze (=SUG) liegt z.B. bei 5000€. Folgendes soll einen Teil unseres Algorithmus von Matlab darstellen:

```
Daten=Datenladen(Daten);
Daten=Daten(Daten(:,2)>=SUG,:);
Daten_SchadenAnzahl_Perioden=JahresSchadenAnzahl(Daten);
lambda=mean(Daten_SchadenAnzahl_Perioden);
SimSchadenHauf=poissrnd(lambda,n,1);
```

Abbildung 1: Ausschnitt des programmierten Algorithmus

In den ersten drei Zeilen des Codes nehmen wir Bezug auf die Schadenshäufigkeit der zur Verfügung gestellten, vergangenen Datensätze „Daten“ und filtern alle Werte unter der Schadensuntergrenze hinaus. Weiters berechnen wir in der nächsten Zeile den Erwartungswert der Schadenshäufigkeit, der hierbei auch als λ bezeichnet wird. Die folgende Formel beweist die Aussage, dass der Erwartungswert von X mit λ verglichen werden kann:

$$E(X) = \sum_{k=0}^{\infty} kP(X=k) = \sum_{k=0}^{\infty} k\frac{\lambda^k}{k!}e^{-\lambda} = e^{-\lambda}\sum_{k=1}^{\infty}\frac{\lambda^k}{(k-1)!} =$$

$$= e^{-\lambda}\sum_{k=0}^{\infty}\frac{\lambda^{k+1}}{k!} = e^{-\lambda}\underbrace{\left(\sum_{k=0}^{\infty}\frac{\lambda^k}{k!}\right)}_{=e^{\lambda}}\lambda = \lambda$$

Abbildung 2: Formel zur Herleitung von E(X)= λ

2.1 Poisson-Verteilung: Schadenshäufigkeitsverteilung

Die Poisson Verteilung *Poi(λ)* (diskret) kann in einem festen Zeitraum das Auftreten eines Ereignisses modellieren, wobei hier die x-Achse die Schadenshäufigkeit pro Jahr und die y-Achse die Gesamtjahre darstellt. λ, also unser Erwartungswert, beträgt 17.
Aus unseren simulierten 10 000 Jahren gibt es zum Beispiel 1800 Jahre, in denen 17 Schadensfälle auftraten.

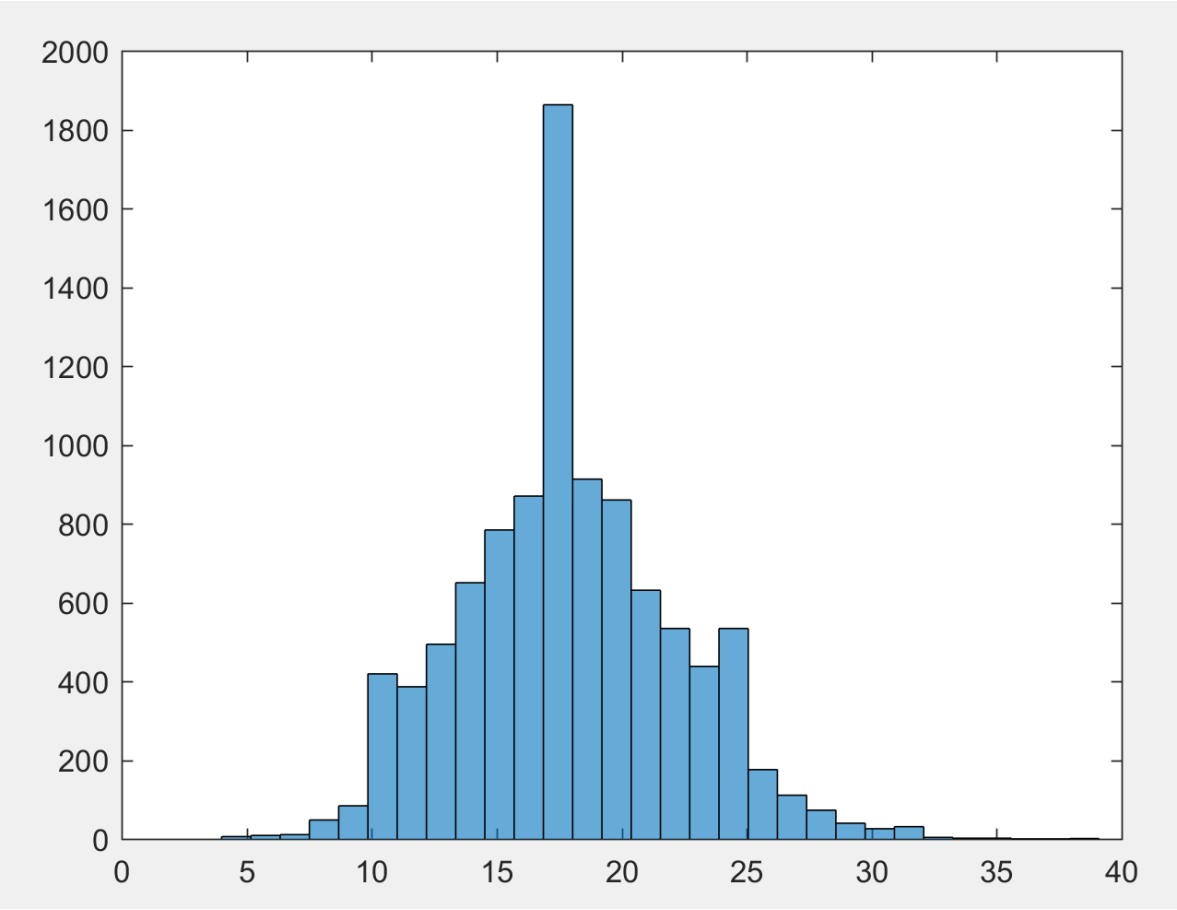


Abbildung 3: Poisson-Verteilung der Schadenshäufigkeit

2.2 Log-Normalverteilung: Schadenshöhenverteilung

Als nächstes erstellen wir mit der Log-Normalverteilung ein Modell, welches berücksichtigt, dass kleine Schäden häufiger und große Schäden seltener auftreten: $logNorm(\mu,\sigma^2)$ mit $\mu=\mathbb{R}$, $\sigma > 0$.

Dabei können jedoch viele kleine Schäden die erwartete Schadenshöhe verfälschen. Aus diesem Grund ist eine Versicherung dazu verleiten, zu wenig Geld für eine mögliche Schadensauszahlung zurückzulegen. Deshalb eliminieren wir kleine, häufige Schäden, um eine realistischere Log-Normalverteilung zu erhalten.

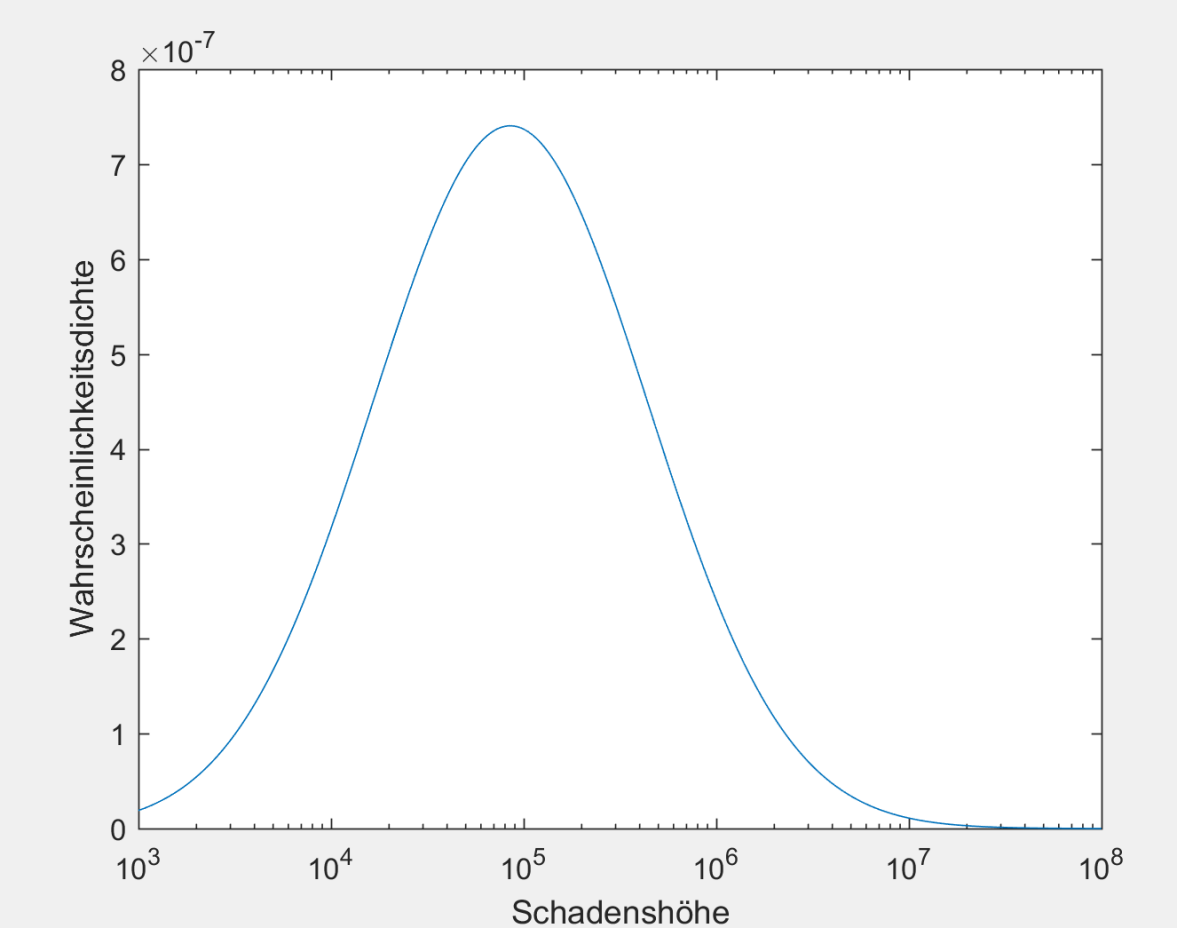


Abbildung 4: Log-Normalveteilung

Diese kontinuierliche Verteilungsdichte kann wie folgt beschrieben werden:

$$f(x) = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma x} e^{-\frac{(ln(x)-\mu)^2}{2\sigma^2}} & , x > 0 \\ 0 \end{cases}$$

Zudem gilt für den Erwartungswert und die Varianz:

$$E(X) = e^{\mu + \frac{\sigma^2}{2}}$$

$$Var(X) = e^{2\mu+\sigma^2}\left(e^{\sigma^2} - 1\right)$$

Durch Umformung der vorhin genannten Formeln bestimmen wir nun die datenabhängigen Parameter μ und σ für die Poisson- und die Log-Normalverteilung, die auch für die Simulation zur Rückstellung der Geldmittel verwendet werden. Wie das bewerkstelligt wurde, zeigt folgender Code.

```
E=(mean(Daten(:,2)));
V=var(Daten(:,2));
sigma_logNormal=sqrt(log(exp(log(V))-2*log(E))+1);
mu_logNormal=(log(E)-sigma_logNormal^2/2);
```

Abbildung 5: Ausschnitt Code Algorithmus

Wie im folgenden Code ersichtlich, berechnen wir mithilfe der zuvor hergeleiteten Parameter (sigma_logNormal und mu_logNormal) die Gesamtschadenssumme pro simuliertem Jahr, siehe auch Abbildung 6.

```
for n=1:1:length(SimSchadenHauf)
    for k=1:1:SimSchadenHauf(n)

        SchadenZufall=lognrnd(mu_logNormal,sigma_logNormal);
        SimulationsSchadenSummePeriode(n)=SimulationsSchadenSummePeriode(n)+...
        SchadenZufall*(SchadenZufall>=SUG);

    end
end
```

2.3 Rückstellung von Geldmitteln

Diese beiden Modelle (Poisson- und Logarithmus-Normalverteilung) nutzen wir nun, um die Schadenskategorie zu modellieren. Ein mögliches Modell wäre das Folgende.

$$\begin{gathered}\text{Schadenshäufigkeit pro Periode als } Poi(\lambda) = X \\ \text{Schadenshöhe pro Ereignis als } logNormal(\mu, \sigma^2) = Y \\ \hline \Rightarrow \text{Gesamtschaden pro Periode} = \sum_{i=1}^{X} Y_i = Y_1 + \cdots + Y_X\end{gathered}$$

Abbildung 6: Modell Gesamtschaden

Mit dem Gesamtschadenmodell simulieren wir nun die Schäden einer Kategorie n-mal (hier z.B. 10 000-mal) für einen Durchlauf pro Jahr.
Als nächstes stellt sich die Frage wie viele Geldmittel zurückgelegt werden sollen. Dafür bestimmen wir das Alpha Quantil des prognostizierten Gesamtschadens. Das Alpha Quantil ($\alpha \in [0,1]$) unseres Quantil beträgt 0.95, das heißt, dass 95% der simulierten Gesamtschäden gedeckt werden.

Im nächsten Schritt sortieren wir die Gesamtschadenssumme pro simuliertem Jahr (SimulationSchadenSummePeriode).

In der zweiten und dritten Zeile wenden wir das Alpha Quantil auf unsere sortierte Gesamtschadenssumme pro simuliertem Jahr an und erhalten so die Rückstellung, siehe Abbildung 7. Abschließend berechnen wir den Erwartungswert der Schadenshöhe und lassen uns die sortierte Gesamtschadenssumme pro prognostiziertem Jahr in drei verschiedenen Formen Gesamtschadensimulation, Box Plot, Balkendiagramm) darstellen.

```
SortSimulationsSchadenSummePeriode=sort(SimulationsSchadenSummePeriode);
AlphaQuantil=ceil(alpha*n);
Ruckstellung=SortSimulationsSchadenSummePeriode(AlphaQuantil);
Erwartungswert_Simulation=mean(SortSimS);
PlotDaten(sort(SimulationsSchadenSummePeriode),AlphaQuantil,Ruckstellung);
```

Abbildung 7: Ende des Algorithmus

2.4 Ergebnis

In unserem programmierten Algorithmus erzielen wir eine Rückstellung von 90 474 000€ für den vorgegebenen Datensatz.

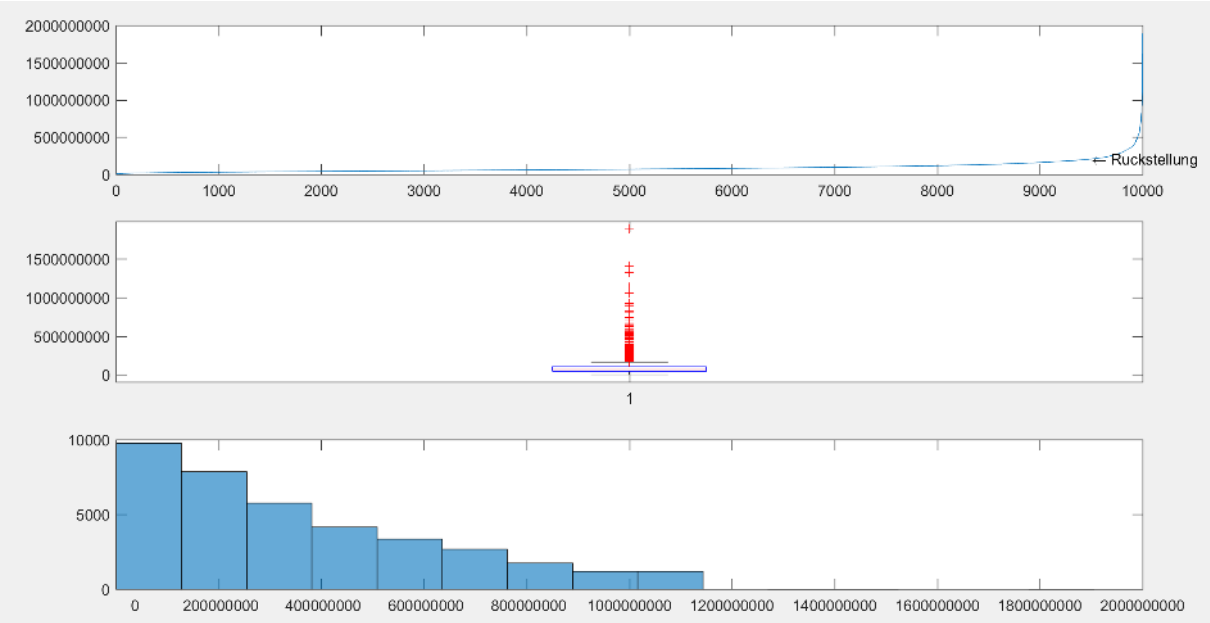


Abbildung 8: Ergebnis

2.5 Korrelation zweier Kategorien

Anders verhält es sich bei der Simulation abhängiger Schadenskategorien. Hierzu müssen allerdings mehrere Kategorien auf einmal ins Auge gefasst werden, da viele davon miteinander korrelieren. Beispiele hierfür ist die Abhängigkeit von natürlichen und menschengemachten Naturkatastrophen wie etwa der Super-Gau in Fukushima (menschengemacht) infolge eines Tsunamis (natürlich).

Trotz individueller Simulation der einzelnen Schadenskategorien beeinflussen sich manche Datensätze gegenseitig, sie korrelieren. Zur Ermittlung der (Rang-)Korrelation von zwei verschiedenen Datensätzen ist es unbedingt nötig, diese in ihren Strukturen anzupassen:

- Eichung der Zeiträume zweier oder mehrerer Datensätze (D_1 , D_2 , ... , D_n).

    - Eine Möglichkeit wäre die Schnittmenge der beiden Datensätze. (Zr = Zeitraum $D_1 \cap$ Zeitraum D_2)
    - Ist an einem Tag kein Schaden vorhanden, setzt man 0€ für den Schaden ein, damit beide Datensätze gleich lang sind. Um Verzögerungseffekte in Daten besser zu berücksichtigen, kann man die Tagesschäden auf Jahre aufsummieren.

- Anschließend ordnen wir jedem bestimmten Schaden einen Rang zu
    - Rang 1 → 0€
    - Rang 2 → 2. geringster Schaden

      .

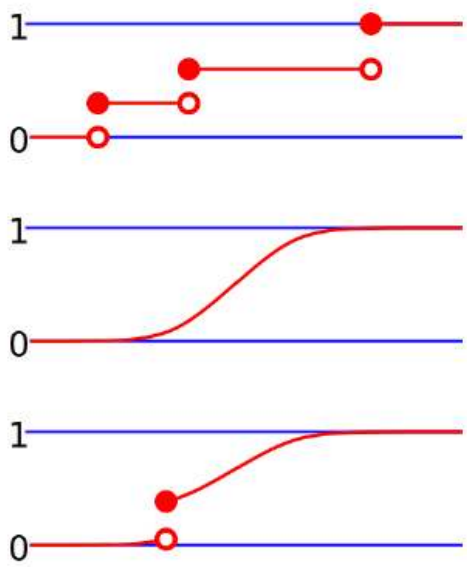
      .

      .
    - Rang n → höchster Schaden

Nun können wir die sogenannte Rang-Korrelation ermitteln. $\overline{D}_1$ und $\overline{D}_2$ sind die Erwartungswerte der jeweiligen Datensätze.

$$Corr(D_1, D_2) = \frac{\sum_{i=1}^{n}(D_{1i} - \overline{D}_1)(D_{2i} - \overline{D}_2)}{\sqrt{\sum_{i=1}^{n}(D_{1i} - \overline{D}_1)^2 \sum_{j=1}^{n}(D_{2j} - \overline{D}_2)^2}}$$

Das Ergebnis ist ein Wert von -1 bis 1. Je weiter die Zahl von 0 abweicht, desto höher die Korrelation. Ist das Ergebnis 0, korrelieren die Kategorien nicht. Mithilfe dieses Korrelationswertes ist es uns möglich, zwei korrelierte Schäden zu simulieren. Dafür nutzen wir eine Verteilungsfunktion.



Sei X eine beliebige Verteilung, dann bezeichnet Fx(x) = P(X ≤ x) die Verteilungsfunktion von X.

In unserem Fall gilt $F_{Kat}(x)$ = P(Kat ≤ x) ist die Wahrscheinlichkeit, dass der Gesamtschaden kleiner gleich x ist. Kat ist die Gesamtschadensverteilung einer Kategorie.

Für einen simulierten Gesamtschadensvektor Kat (mit n-Simulationen) haben wir:

$$F_{Kat}(x) = \frac{1}{n}(\text{Anzahl Schäden } \leqslant x)$$

Hierbei bedeutet „Anzahl Schäden ≤ x“, dass wir die Anzahl der simulierten Gesamtschäden betrachten, die unterhalb des Wertes x liegen.

Die Verteilungsfunktion der Standardnormalverteilung wird mit φ bezeichnet.

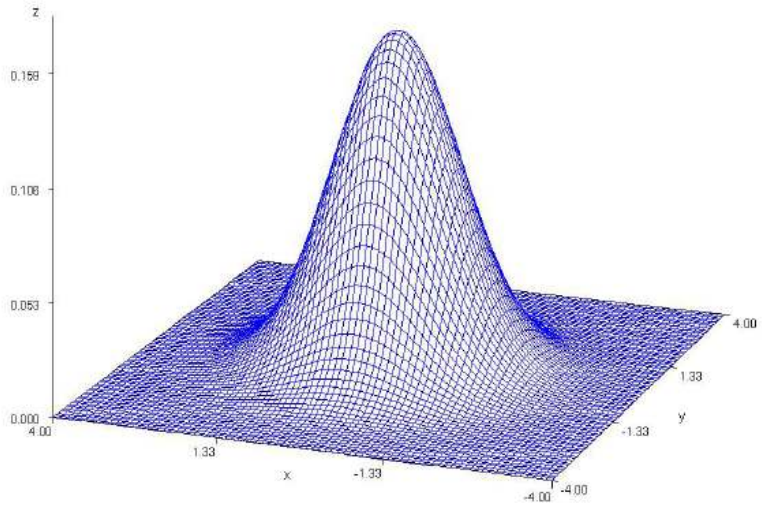
$$\varphi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt = P(N(0,1) \leqslant x) = F_{N(0,1)}(x)$$

Die zweidimensionale Standardnormalverteilung benötigt einen Erwartungswert μ und eine Varianz σ^2. Der Erwartungswert ist jedoch ein Element aus $\mathbb{R}^2$ und die Varianz eine Matrix mit 2 Zeilen und 2 Spalten. Die Parameter können also beispielsweise $\mu = \binom{0}{0}$ und $\sigma^2 = \binom{1\ 0}{0\ 1}$ sein, die Standardnormalverteilung wird dann mit $N_2(\mu, \sigma^2)$ bezeichnet. Die Verteilungsdichte von $N_2(\mu, \sigma^2)$ sieht wie folgt aus:

$$f(x_1, x_2) = \frac{(2\pi)^{-\frac{d}{2}}}{\sqrt{\det(\sigma^2)}} e^{-\frac{1}{2}(x_1-\mu_1, x_2-\mu_2)\left((\sigma^2)^{-1}\right)\begin{pmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{pmatrix}}$$

Wichtig ist, dass $\det(\sigma^2) > 0$ mit $w_{21} = w_{12}$ und $w_{11}, w_{22} > 0$ für für $\sigma^2 = \binom{w11\ w12}{w21\ w22}$ gilt. Weiters bezeichnet d = 2 die Dimension 2 und $(\sigma^2)^{-1}$ die inverse Matrix.

Dichtefunktion einer 2-dimensionalen Normalverteilung



Die zweidimensionale Standardnormalverteilung X kann von zwei unabhängigen eindimensionalen Standardnormalverteilungen X_1 und X_2 mit $\binom{X}{X}$ dargestellt werden.

Eindimensionale Normalverteilungen kann man von einer Standardnormalverteilung herleiten: $Y = \mu + \sigma X$

Dasselbe funktioniert bei einer 2-dimensionalen Verteilung, $N_2(\mu, \sigma^2)$ bezeichnet man mit Y. X sei zweidimensional standartnormalverteilt, es gibt eine Matrix A, sodass $\sigma^2 = AA^T$ gilt. A = $\binom{a11\ a12}{a21\ a22}$ und $A^T = \binom{a11\ a21}{a12\ a22}$. Dann gilt $Y = \mu + AX$.
A kann auch als die Wurzel der Matrix σ^2 betrachtet werden. Um so eine Matrix zu erhalten, wenn ich nur σ^2 kenne, benutze ich die Cholesky Zerlegung von σ^2.

Konkret erhalten wir $a_{12} = 0$, $a_{11} = \sqrt{w11}$, $a_{22} = \sqrt{w22 - a21\text{^}2}$, $a_{21} = \frac{w21}{a11}$

1) $a_{ij} = 0$ falls $i < j$

2) $a_{ii} = \sqrt{w_{ii} - \sum_{k=1}^{i-1} a_{ik}^2}$

3) $a_{ij} = \frac{1}{a_{jj}} \left(w_{ij} - \sum_{k=1}^{j-1} a_{ik} a_{jk} \right)$ für $i > k$

3 Die Simulation zweier korrelierter Datensätze anhand des Beispiels „Versicherung“

3.1 Die Problemstellung

Um im Schadensfall ausreichend abgesichert zu sein, möchte eine Versicherung aus 2 Datensätzen, die aus verschiedenen Kategorien stammen, eine Rückstellung von 95% berechnen. Jedoch muss hierbei auch die Korrelation der beiden Kategorien berücksichtigt werden. Aufgrund von Verzögerungseffekten zweier Schadenskategorien, kann z.B. zwischen Tages- und Jahresschadenswerten unterschieden werden.

3.2 Der Algorithmus

```
% vgl. 1. Beispiel zur Rückstellung (Daten wurden bereits einmal simuliert)
    addpath 'RiskCodes'

    Daten1='FinanzMarktVerluste_gut_zu_sortieren.mat';
    Daten2='CyberCrimeVerluste_gut_zu_sortieren.mat';

% 1 oder 0 : wahr oder falsch  (Zeitverzögerungseffekte)
    Jahre_Hochsummiert=1;
    Tage_Hochsummiert=0;

% vgl. 1. Beispiel zur Rückstellung
    SchadenSimulation1='FinanzMarktVerluste_gut_zu_sortierenSchadenSimulation.mat';
    SchadenSimulation2='CyberCrimeVerluste_gut_zu_sortierenSchadenSimulation.mat';

    N=15000;
    alpha1=0.95;
    alpha2=0.95;

    Daten1=Datenladen(Daten1);
    Daten2=Datenladen(Daten2);
    SchadenSimulation1=Datenladen(SchadenSimulation1);
    SchadenSimulation2=Datenladen(SchadenSimulation2);
```

Abbildung 9: Daten hereinladen

Im ersten Abschnitt werden die 2 Datensätze hereingeladen und mit Variablen benannt. Danach kann man angeben, ob Tages- oder Jahresschadenswerten berechnet werden sollen. Dies ist insofern wichtig, da nicht nur Schäden, die am selben Tag passieren, zusammenhängen, sondern auch aus demselben Grund mit einigen Tagen oder Monaten Verzögerung auftreten können.

Die folgenden Absätze sind für das Hereinladen bereits simulierter Schäden sowie die Festlegung der Anzahl der zu erzeugenden Datenpunkte und des Rückstellungsquantils zuständig.

```
24      % aus den Daten werden jeweils das älteste und das neueste Datum
25      % heausgesucht...
26 -        JahrAlt1=min(Daten1(:,1));
27 -        JahrNeu1=max(Daten1(:,1));
28
29 -        JahrAlt2=min(Daten2(:,1));
30 -        JahrNeu2=max(Daten2(:,1));
31
32       % und unter einer Variable zusammmengefasst (Durchschnittszeitraum)
33 -        JahrAlt=max(JahrAlt1,JahrAlt2);
34 -        JahrNeu=min(JahrNeu1,JahrNeu2);
35
36       % Platzhalter für Schadenswerte zu Datum zugehörig
37 -        Daten1Neu=zeros(JahrNeu-JahrAlt,1);
38 -        Daten2Neu=zeros(JahrNeu-JahrAlt,1);
```

Abbildung 10: Die Zeitspanne vom ältesten zum neuesten Datum

Die Schadensvektoren, die produziert werden sollen, müssen dieselbe Läng haben, damit ihre Korrelation berechnet werden kann. Deshalb wird hier aus beiden Datensätzen das jeweils jüngste und jeweils älteste Datum gesucht und danach werden diese Zeitspannen miteinander geschnitten um einheitliche Zeitvektoren produzieren zu können. Der jüngste und älteste Datumswert werden nun als zeitliche Begrenzung für die Platzhaltervektoren *Daten1Neu* und *Daten2Neu* verwendet. In diese werden nachher die Schadenswerte in Eurobeträgen eingetragen.

3.3 Die Tages-if-Schleife

```
40      % if-Schleife um Werte der selben Tage zusammenzuzählen
41 -        if Tage_Hochsummiert
42 -        Step=1;
43 -        for n=JahrAlt:1:(JahrNeu-1)
44 -            StepDaten=1;
45 -            for k=Daten1(:,1)'
46 -                if k==n          % wenn für ein Datum mehrere Werte -> zusammenzählen
47 -                Daten1Neu(Step)=Daten1(StepDaten,2)+Daten1Neu(Step);
48 -                end
49 -                StepDaten=StepDaten+1;
50 -            end
51 -            Step=Step+1;
52 -        end
53
54        % auch für den 2. Datenvektor
55 -         Step=1;
56 -        for n=JahrAlt:1:(JahrNeu-1)
57 -            StepDaten=1;
58 -            for k=Daten2(:,1)'
59 -                if k==n
60 -                Daten2Neu(Step)=Daten2(StepDaten,2)+Daten2Neu(Step);
61 -                end
62 -                StepDaten=StepDaten+1;
63 -            end
64 -            Step=Step+1;
65 -        end
66 -        end
```

Abbildung 11: Tagesschadenwerte zusammenzählen

Wenn am Anfang *Tage_Hochsummiert=1* definiert wurde, so tritt diese Schleife nun in Aktion. Sie zählt in Einserschritten vom ältesten bis zum jüngsten Datum durch und addiert Schadenswerte, die am selben Tag passiert sind. Dies passiert so lange, bis alle Daten behandelt werden. Dann wird aus der Schleife ausgetreten. Dasselbe passiert auch mit dem 2. Datensatz. Wenn auch hier alle Werte betrachtet wurden, wird die komplette Schleife beendet und das Programm behandelt die nächste Zeile.

3.4 Die Jahres-if-Schleife

```
68
69      % if-Schleife um Werte der selben Jahre zusammenzuzählen
70 -        if Jahre_Hochsummiert
71
72       % die Datumswerte werden  nach Jahren sortiert
73 -        JahrAltHochsummiert=str2num(datestr(JahrAlt,'yyyy'));
74 -        JahrNeuHochsummiert=str2num(datestr(JahrNeu,'yyyy'));
75 -        PeriodenGesamtHochsummiert=JahrNeuHochsummiert-JahrAltHochsummiert;
76
77       % Platzhalter für Schadenswerte zu Datum zugehörig
78 -        JahresSchaden1=zeros(PeriodenGesamtHochsummiert+1,1);
79 -        JahresSchaden2=zeros(PeriodenGesamtHochsummiert+1,1);
80
```

Abbildung 12: Die Zeitspanne vom ältesten bis zum neuesten Jahr

Dieser Teil des Codes wird ausgeführt wenn für *Jahre_Hochsummiert* die Bedingung *Wahr* (=1) erfüllt ist. Doch bevor es, wie vorher, ans Aufsummieren der Schadenswerte geht müssen noch einige wichtige Definitionen gemacht werden. Dafür ist es wichtig, zu wissen, dass der Computer Datumswerte nicht so schreibt, wie wir es tun. Er arbeite nicht mit dem Schema TT.MM.JJJJ, sondern rechnet ein Datum in einen Zahlenwert um. Auch darf nicht vom erstbesten Schnitt-Datum zu zählen begonnen werden, was in unserem Fall ein Datum mitten im Jahr ist. Hierbei müssen wir dem Computer sagen, dass er in Jahren zählen muss. Jetzt werden Platzhaltervektoren mit der Möglichkeit für *n* Eintragungen, also so viele wie es Jahre in der Schnittmenge gibt, gemacht.

```
96
97 -      Step_JahresSchaden_Vektor=1;
98
99 -      for n=JahrAltHochsummiert:1:JahrNeuHochsummiert
100 -         Step=1;
101 -         for k=Daten2(:,1)'
102 -             if str2num(datestr(k,'yyyy'))==n
103 -                 JahresSchaden2(Step_JahresSchaden_Vektor)=JahresSchaden2(Step_JahresSchaden_Vektor)+Daten2(Step,2);
104 -             end
105 -                 Step=Step+1;
106 -         end
107
108 -      Step_JahresSchaden_Vektor=Step_JahresSchaden_Vektor+1;
109 -      end
110
111
112       % Daten werden auf den Platzhalter-Vektor geladen
113 -      Daten1Neu=JahresSchaden1;
114 -      Daten2Neu=JahresSchaden2;
115 -      end
```

Abbildung 13: Jahresschadenswerte zusammenzählen

Dieser Teil des Algorithmus ist der *Tage_Hochsummieren* Schleife sehr ähnlich, mit dem Unterschied, dass in Jahren und nicht in Tagen gezählt wird. Diese Schleife wird für *Daten1* und *Daten2* durchlaufen. Wichtig ist, dass die erzeugten Daten auch auf den *Daten1Neu* und *Daten2Neu* Vektor geladen werden, damit die nächsten Schritte funktionieren.

3.5 Korrelation

```
116
117     % den Daten werden Ränge zugeordnet und deren Korrelation wird ermittelt (Rangkorrelation)
118 -       CorData12=corr(Daten1Neu,Daten2Neu,'Type','Spearman');
119
120
121     % 2D-Standardnormalverteilung
122 -       X1=normrnd(0,1,N,1);
123 -       X2=normrnd(0,1,N,1);
124
125
126     % Matrix der Korrelation
127 -       Korr=[[1,CorData12];[CorData12,1]];
128
129
130     % Wurzel der Korrelationsmatrix
131 -       A=chol(Korr, 'lower');
132
```

Abbildung 14: Korrelation und Standardnormalverteilung

Ab hier kommt die Korrelation ins Spiel. Vorweg muss man wissen, dass es nicht zielführend ist, die reinen Eurobeträge miteinander zu vergleichen. Denn auch, wenn in der einen Schadenskategorie 1000€ ein hoher Wert ist, heißt das nicht, dass dies auch für die andere Kategorie zutrifft. Deshalb werden Schadenswerten Ränge zugeordnet und diese verglichen. Daraus wird die Korrelation berechnet.

Im nächsten Absatz werden 2 Standardnormalverteilungen simuliert, auf die im nächsten Screenshot eingegangen wird.

Im 3. Abschnitt wird aus den Korrelationswerten die Korrelationsmatrix erstellt, aus der im nächsten Schritt durch Cholesky-Zerlegung die Wurzel gezogen wird. Diese Matrix wird mit *A* bezeichnet.

3.6 Die Verknüpfung von Simulationswerten und Korrelation

```
134     % 2D-Standardnormalverteilung unter einer Variable zusammengefasst
135 -       X=[X1,X2]';
136
137     %Platzhalter
138 -       Y=0*X;
139
140      % die Verzerrung, die durch die Korllation zustande kommt wird auf die Standardnormalverteilung angewandt
141 -       for n=1:1:length(X(1,:))
142 -           Y(:,n)=A*X(:,n);
143 -       end
144
145     % Simulationswerten werden Zahlen von 0 bis 1 zugeordnet
146     % (1. Übersetzungsschritt)
147 -       GaussCopula=normcdf(Y,0,1)';
148     % 1 Zeile, 2 Spalten, 1. Grafik
149 -       subplot(1,2,1)
150 -       scatter(GaussCopula(:,1),GaussCopula(:,2))
151
152     % Abhängigkeitsgitter von Y wird auf vorher simulierte Schäden übertragen
153 -       for n=1:1:length(GaussCopula(:,1))
154 -       Schaden1(n)=SchadenSimulation1(ceil(length(SchadenSimulation1)*GaussCopula(n,1)));
155 -       Schaden2(n)=SchadenSimulation2(ceil(length(SchadenSimulation2)*GaussCopula(n,2)));
156 -       end
157
158     % Simulationspunktewolke
159 -       subplot(1,2,2)
160 -       scatter(Schaden1,Schaden2)
161
```

Abbildung 15: Die simulierten Daten werden mit Korrelationen versehen

Im ersten Absatz wird die 2D-Normalverteilung mit der Variable *X* bezeichnet. Y ist die mit A multiplizierte und dadurch verzerrten, Standardnormalverteilung. Nun werden den Zahlenwerten von *Y* und den simulierten Schäden, mit der Standardnormalverteilungsfunktion φ, Zahlen von 0 bis 1, zugeordnet über die sie dann einander zugewiesen werden können. Mit der Funktion *scatter* werden die mit der Standardnormalverteilungsfunktion φ geänderten, simulierten Y-werte, bezeichnet mit GaussCopula, in einer 2D-Grafik dargestellt.

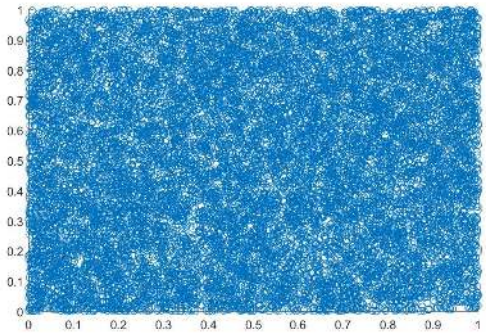


Abbildung 16: Übersetzungsschritt

Im vorletzten Absatz werden die Schadenswerte und die Korrelation durch die GaussCopula und den jeweiligen Quantilsfunktionen der einzelnen Schadenssimulationen zusammengebracht und mit *Schaden1* und *Schaden2* benannt. Dann werden auch die korrelierten Werte in einer Grafik dargestellt.

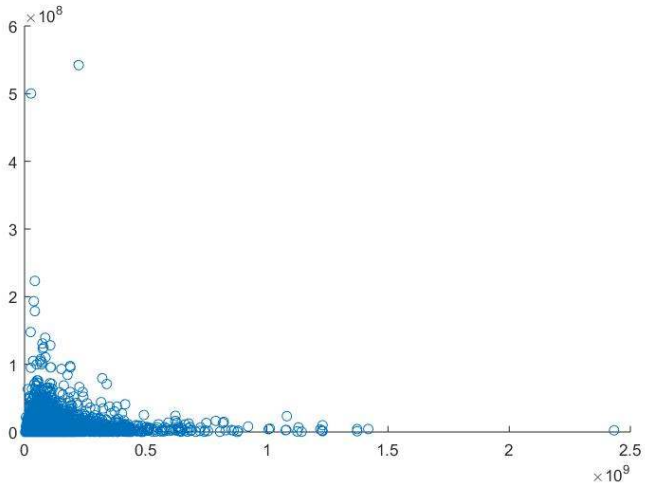


Abbildung 17: korrelierte Verteilung vom Beispiel „Versicherung“

3.7 Die Rückstellung

```
162
163     % hier wird die Rückstellung berechnet
164 -       Schaden1=sort(Schaden1);
165 -       Rueckstellung1=Schaden1(ceil(alpha1*length(Schaden1)));
166
167 -       Schaden2=sort(Schaden2);
168 -       Rueckstellung2=Schaden2(ceil(alpha2*length(Schaden2)));
169
```

Abbildung 18: Algorithmus 8

Hier wird für beide Schadensvektoren die jeweilige Rückstellung berechnet. Dafür werden die korrelierten Schäden ihrer Größe nach sortiert und der Wert an dem, in diesem Fall, aufgerundeten 95. Prozent berechnet.

3.8 Dreidimensionale Darstellungen

Die folgenden 4 Abbildungen beinhalten den Code der 3D-Glockenkurve sowie die Grafik selbst von „Versicherung“ und 2 Vergleichskurven um zu zeige, wie sich die Korrelation auswirkt. Bei allen 3 Kurven liegt der Erwartungswert in [0;0] und die Achsenskalierungen sind ident.

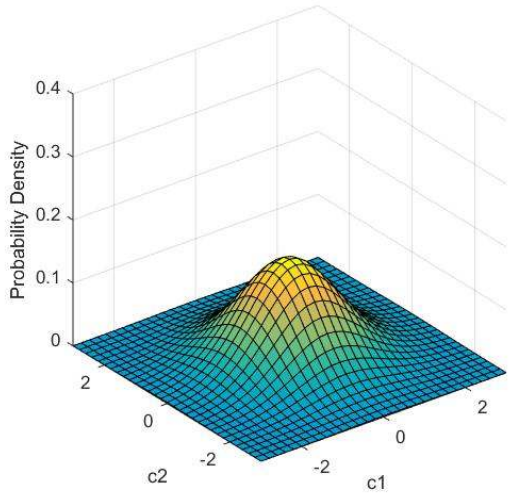


Abbildung 8: Standardnormalverteilung mit Korrelation 0

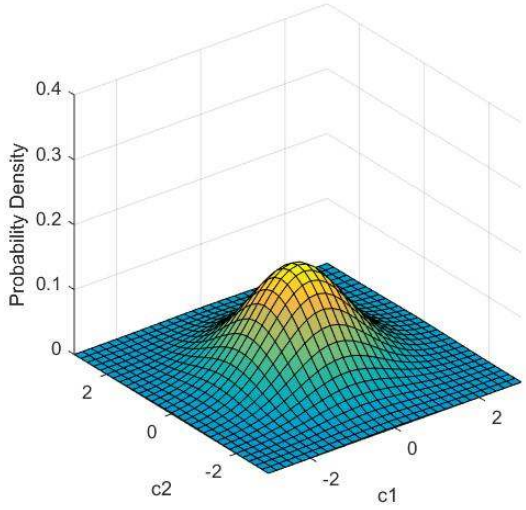


Abbildung 9: korrelierte Verteilung von „Versicherung“

```
 3
 4 -    subplot(1,3,2)
 5
 6 -    mu = [0 0];
 7 -    Sigma =Korr;
 8 -    c1 = -3:.2:3;
 9 -    c2 = -3:.2:3;
10 -    [C1,C2] = meshgrid(c1,c2);
11 -    F = mvnpdf([C1(:) C2(:)],mu,Sigma);
12 -    F = reshape(F,length(c2),length(c1));
13 -    surf(c1,c2,F);
14 -    caxis([min(F(:))-.5*range(F(:)),max(F(:))]);
15 -    axis([-3 3 -3 3 0 .4])
16 -    xlabel('c1'); ylabel('c2'); zlabel('Probability Density');
17
```

Abbildung 10: Code der 3D-Glockenkurve von "Versicherung"

Die Simulation von 3 korrelierten Datensätzen anhand des Beispiels „Versicherung“

Das Prinzip der Korrelation funktioniert nicht nur mit 2 sondern auch mit beliebig vielen Kategorien. Hier werden 3 voneinander abhängige Simulationen durchgeführt. Folgendes muss erweitert/geändert werden damit 3 Datensätze miteinander verknüpft werden können:

- 3. Datensatz laden
- 3 Quantile
- Zeitintervalle auf die 3 Datensätze abstimmen
- Tags- und Jahresschleifen jeweils noch einmal ausführen
- Die Datensätze müssen jeweils einzeln miteinander korreliert werden (vgl. Formel) und jeder dieser Korrelationen muss eine Variable zugewiesen werden

```
160     % den Daten werden Ränge zugeordnet und deren Korrelation wird ermittelt (Rangkorrelation)
161 -       CorData12=corr(Daten1Neu,Daten2Neu,'Type','Spearman');
162 -       CorData13=corr(Daten1Neu,Daten3Neu,'Type','Spearman');
163 -       CorData23=corr(Daten2Neu,Daten3Neu,'Type','Spearman');
164
```

Abbildung 11: Code für die Korrelation dreier Datensätze

- 3 Standardnormalverteilungen simulieren
- Eine 3x3 Korrelationsmatrix erstellen
- Zum Ausgeben der Grafiken *scatter* verwenden

```
195     % 1 Zeile, 2 Spalten, 1. Grafik
196 -       figure
197 -       subplot(1,2,1)
198 -       scatter3(GaussCopula(:,1),GaussCopula(:,2),GaussCopula(:,3))
199
200     % Abhängigkeitsgitter von Y wird auf vorher simulierte Schäden übertragen
201 -       for n=1:1:length(GaussCopula(:,1))
202 -       Schaden1(n)=SchadenSimulation1(ceil(length(SchadenSimulation1)*GaussCopula(n,1)));
203 -       Schaden2(n)=SchadenSimulation2(ceil(length(SchadenSimulation2)*GaussCopula(n,2)));
204 -       Schaden3(n)=SchadenSimulation3(ceil(length(SchadenSimulation3)*GaussCopula(n,3)));
205 -       end
206
207     % Simulationspunktewolke
208 -       subplot(1,2,2)
209 -       scatter3(Schaden1,Schaden2,Schaden3)
210
```

- Die 3. Rückstellung berechnen

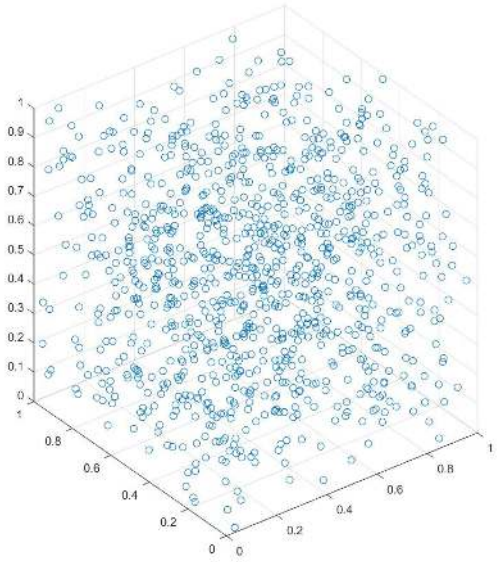


Abbildung 12: korrelierte Verteilung von „Versicherung“

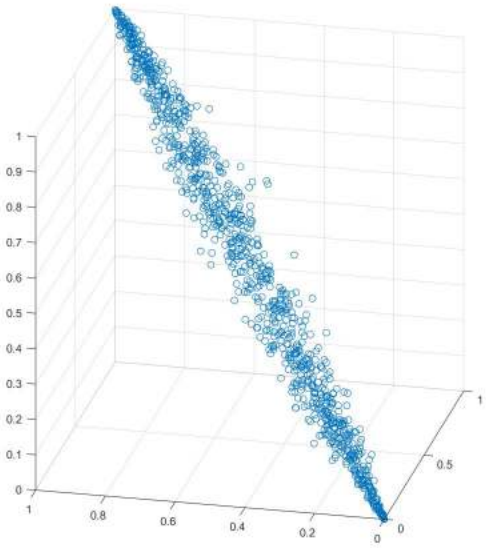


Abbildung 13: Zum Vergleich eine Verteilung mit einer beinahe 1 Korrelation

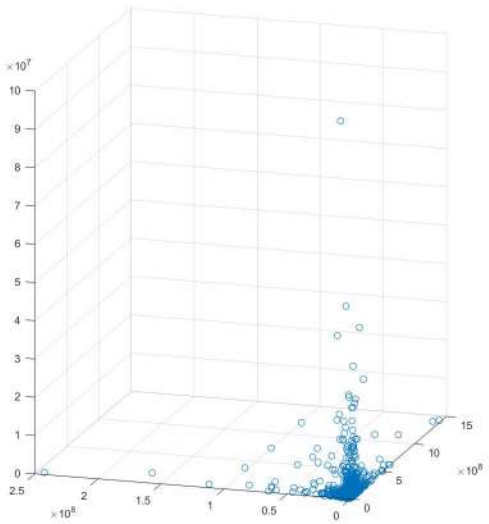


Abbildung 14: korrelierte Schadenswerte vom Beispiel „Versicherung“

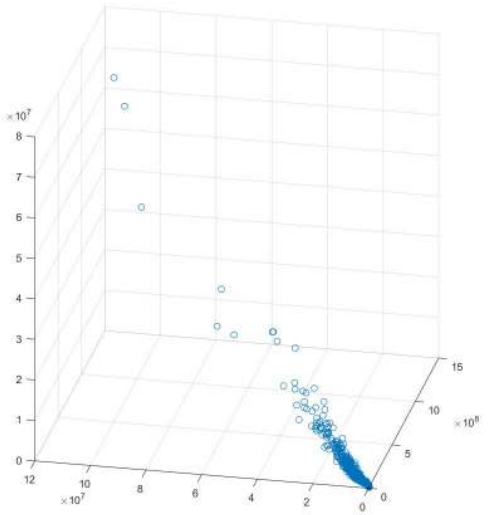
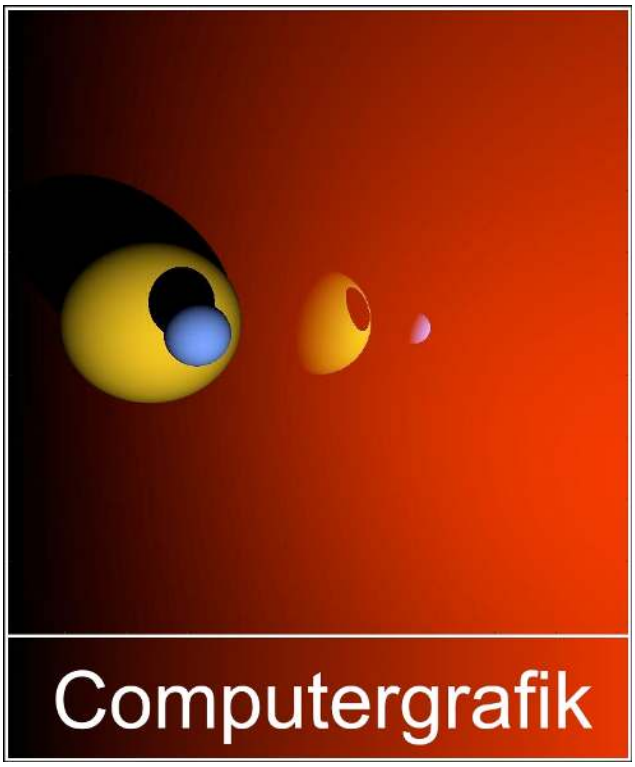


Abbildung 15: Zum Vergleich Schadenswerte mit einer beinahe 1 Korrelation

Abbildungsverzeichnis

Woche der Modellierung mit Mathematik
von 04.02.2017 bis 09.02.2017

Voraussage einer Beobachtung durch Strahlenverfolgung



Gruppenmitglieder: Carmen Kahl,
Fabian Pototschnik, Jakob Hernler,
Leon Pehmer, Miriam Mayer und
Sabrina Unterreiner

Betreuer: Ass. Prof. Dr. Laurent Pfeiffer

Inhaltsverzeichnis

1 Einleitung

Ob in Computerspielen, in Special Effects bei Filmen oder sonstigen Medien – ein Leben ohne Computergrafik wäre heutzutage undenkbar. Ohne explizites Nachdenken nehmen wir die Tatsache der virtuellen Bilderstellung hin und sind uns der Komplexität dieser schon lange nicht mehr bewusst.

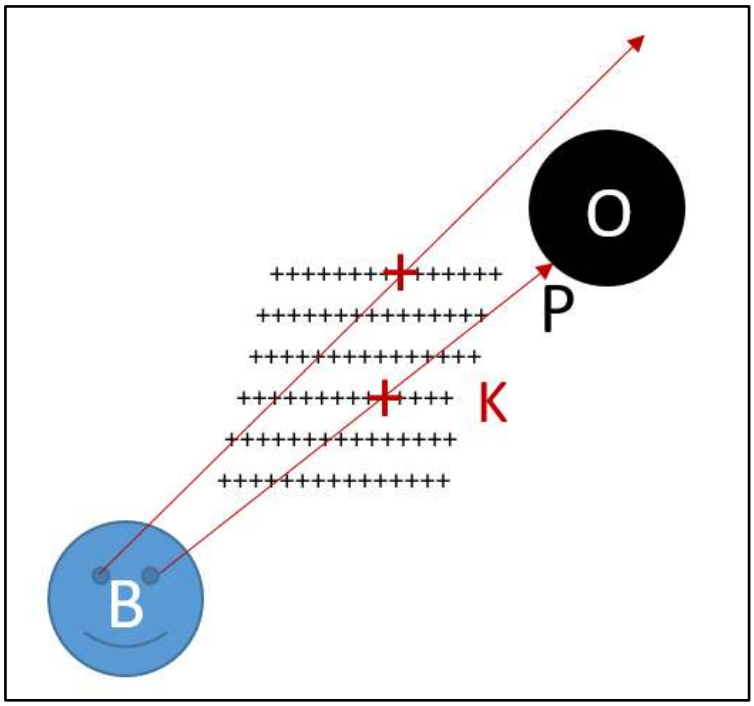
Die Computergrafik ist eine Teildisziplin zwischen den beiden Bereichen Informatik und angewandter Mathematik. Sie befasst sich mit der Erzeugung von Bildern bzw. Grafiken, die durch Strahlenverfolgungsalgorithmen beschrieben und durch gezielte Spezifikation eines Programmes immer weiter verfeinert werden. Dies ermöglicht eine möglichst realitätsnahe Darstellung von mathematisch definierbaren Körpern und Flächen wie beispielsweise Kugeln, Ebenen oder Würfeln.

Zur Illustration und Berechnung dieser Bilder wurde die Software „MATLAB" herangezogen, wobei jedes Gruppenmitglied das Programm individuell erstellt und weiterentwickelt hat. Von einfachsten zweidimensionalen Kreisen mit niedriger Auflösung kamen wir über dreidimensionale Bildnisse bis hin zu komplexen Figurenkompositionen. Um schnelles und problemloses Verändern von Input-Werten zu gewährleisten, wurde der ganze Prozess anhand von Variablen generalisiert.

2 Allgemeine Erklärung und Einführung

2.1 Strahlenverfolgungsalgorithmus

Zu Beginn existiert im dreidimensionalen Raum ein Beobachter B, eine Lichtquelle L und ein Objekt O. Der Beobachter sendet seine „Seh-Strahlen" durch die „Kreuze" K eines imaginären Bildschirms (siehe Abschnitt: Screen), welche gegebenenfalls das Objekt schneiden. Zur Vereinfachung wird momentan in der Skizze nur der Weg zweier Strahlen dargestellt. Trifft der Strahl auf das Objekt, so nimmt das „Kreuz" K die Farbe des Objektes im Punkt P an. Dieses Bild kann durch den Computer grafisch dargestellt werden.



2.2 Screen

Wie bereits erklärt, wird der Screen als „Punkteebene" in Form einer Matrix dargestellt, wobei die Anzahl der Pixel der Anzahl der Punkte entspricht (n_U ≙ Anzahl der horizontalen Pixel/Punkte bzw. Spalten der Matrix; n_O ≙ Anzahl der vertikalen Pixel/Punkte bzw. Zeilen der Matrix). Mathematisch wurde unsere Screen_Matrix mit drei Eckpunkten F_1, F_2 und F_3 eindeutig bestimmt. Damit jeder Punkt K auf dem Screen vom Strahl des Beobachters „besucht"

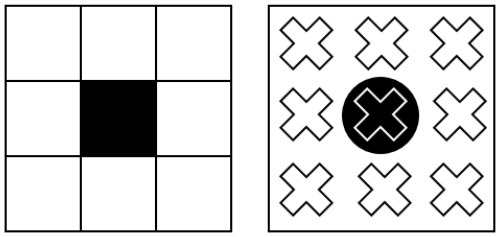
werden kann, werden am Eckpunkt F_1 zwei Richtungsvektoren angehängt ($\overrightarrow{Fo}$[1] für vertikale Richtung, $\overrightarrow{Fu}$[2] für horizontale Richtung).

$$\overrightarrow{Fu} = \frac{F_2 - F_1}{n_U} \; ; \qquad \overrightarrow{Fo} = \frac{F_3 - F_1}{n_O}$$

$$K = F + (i-1) * Fo + (j-1) * Fu$$

$i = 1, \dots, n_O$ und $j = 1, \dots, n_U$ sind Parameter

Trifft der Strahl des Beobachters ($P = B + s * \overrightarrow{BK}$), ein Objekt, so wird diese Information als 1 in der Matrix gespeichert, im anderen Fall ist diesem Punkt auf der Matrix der Wert 0 zugeordnet.



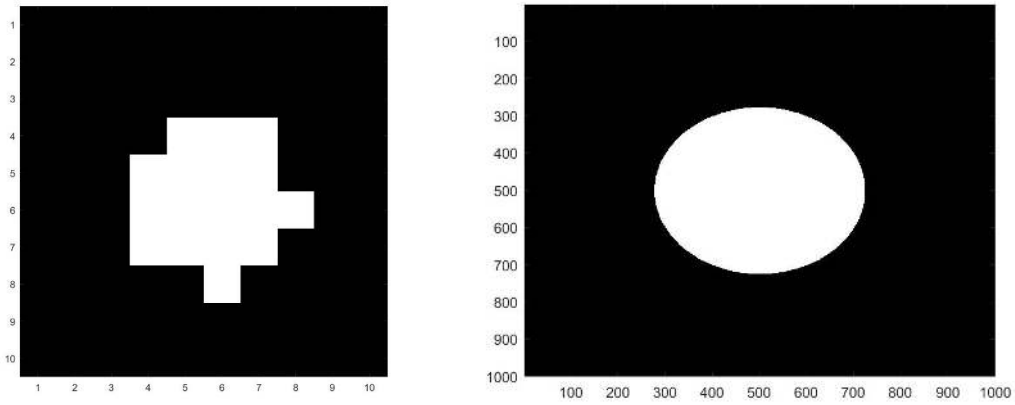
$$Matrix = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

3 Kugel 2D- Strahl & Bildschirm

Der erste Schritt bei der Programmierung unseres Programmes war die Darstellung einer einfachen Kugel in einem zweidimensionalen Raum. Um dies zu erreichen, haben wir einen Beobachter, eine Kugel und einen Screen verwendet. Nachdem die ersten Zeilen des Codes geschrieben worden waren, konnten wir schon erste Ergebnisse betrachten: unsere erste Abbildung einer Kugel.

[1] Fo, o steht für „owi". Aufgrund der zahlreichen Variablen muss auf kreative Namensgebung geachtet werden, damit die verschieden Variablen im Gedächtnis eindeutig zuzuordnen sind und vor allem bleiben.

[2] Fu, u steht für „ummi"



Aber schon nach kurzem Betrachten wurde uns klar, dass wir bei unserem nächsten Versuch eine höhere Auflösung verwenden werden müssten. Sobald wir also die Pixelanzahl des Screens erhöht hatten, sah die daraus resultierende Kugel schon erheblich klarer aus, wie in der folgenden Grafik zu erkennen ist.

4 Darstellung einer Kugel

Aus mathematischer Sicht ist die Kugel ein dreidimensionaler Drehkörper, bei welchem jeder Punkt auf der Oberfläche die gleiche Entfernung zum Mittelpunkt aufweist. Anders gesagt, liegt der Punkt P genau am Rande der Kugel mit dem Mittelpunkt M und den Radius r, wenn $\left|\overrightarrow{PM}\right| = r$ beziehungsweise wenn $\left|\overrightarrow{PM}\right|^2 = r^2$. Daraus ergibt sich:

$$(x_M - x_P)^2 + (y_M - y_P)^2 + (z_M - z_P)^2 = r^2$$

Um des Weiteren zu prüfen, ob ein Strahl (BK) die Kugel schneidet, arbeiten wir mit folgender Voraussetzung:

$$P = B + s * \overrightarrow{BK} \leftrightarrow \begin{cases} x_P - x_B = s * (x_K - x_B) \\ y_P - y_B = s * (y_K - y_B) \\ z_P - z_B = s * (z_K - z_B) \end{cases} \leftrightarrow \begin{cases} x_P = x_B + s * (x_K - x_B) \\ y_P = y_B + s * (y_K - y_B) \\ z_P = z_B + s * (z_K - z_B) \end{cases}$$

Es steht fest, dass der Punkt P von dem Strahl geschnitten wird, falls P die Summe von B und einem Vielfachen des Vektors BK ist.

Nun werden diese Formeln für x_P, y_P, z_P in die weiter oben eingeblendete Gleichung eingefügt:

$$(x_B + s(x_K - x_B) - x_M)^2 + (y_B + s(y_K - y_B) - y_M)^2 + (z_B + s(z_K - z_B) - z_M)^2 = r^2.$$

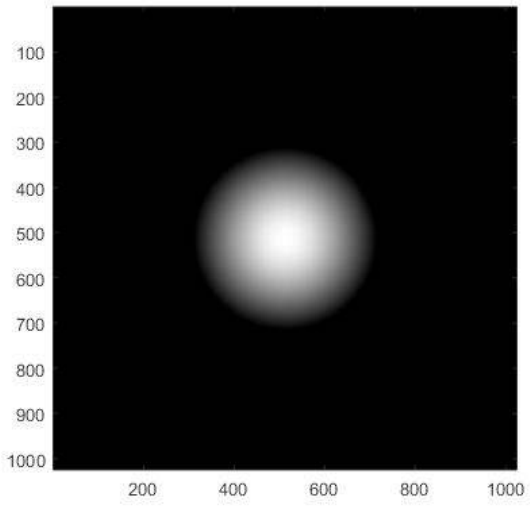
In Matlab wurde diese quadratische Gleichung mit bestimmten Variablen vereinfacht und eine Lösungsformel programmiert.

Um den Punkt P schließlich finden zu können, wird die Unbekannte s benötigt, die dann ebenfalls in die Formel $P = B + s * \overrightarrow{BK}$ eingesetzt wird.

5 Beleuchtung in 3D

Der nächste Schritt war nun, die Kugel dreidimensional darzustellen, indem man Schatten und Beleuchtung in der Programmierung auf Matlab berücksichtigt. Zunächst haben wir die Beleuchtung programmiert:

Als Input-Variablen werden hier B (Beobachter), M (Mittelpunkt der Kugel), r (Radius der Kugel), L (Lampe), F_1, F_2, F_3 (die Koordinaten des Screens) und n_O, n_U (Variblen, die die Auflösung bestimmen) verwendet.

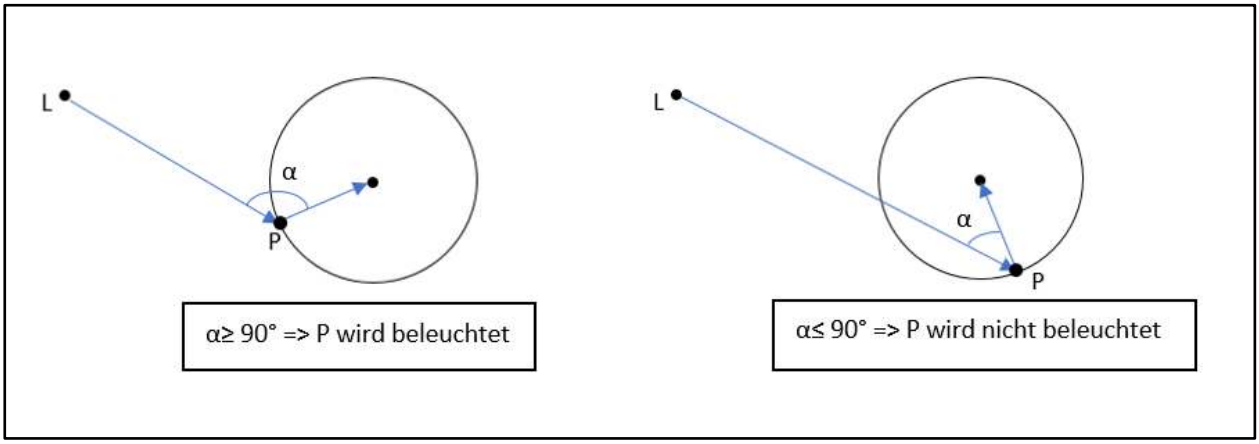


Um zu prüfen, ob der vom Betrachter ausgehende Strahl ein Objekt, in unserem Fall eine Kugel, trifft, werden sogenannte „For- Schleifen" in MatLab verwendet. Damit wird berechnet, ob und in welchem Winkel der Strahl auf die Kugel, trifft. Der Winkel bestimmt, welche Seiten beleuchtet werden.

$$\cos(\alpha) = \frac{\overrightarrow{PL} * \overrightarrow{PM}}{|PL| * |PM|}$$

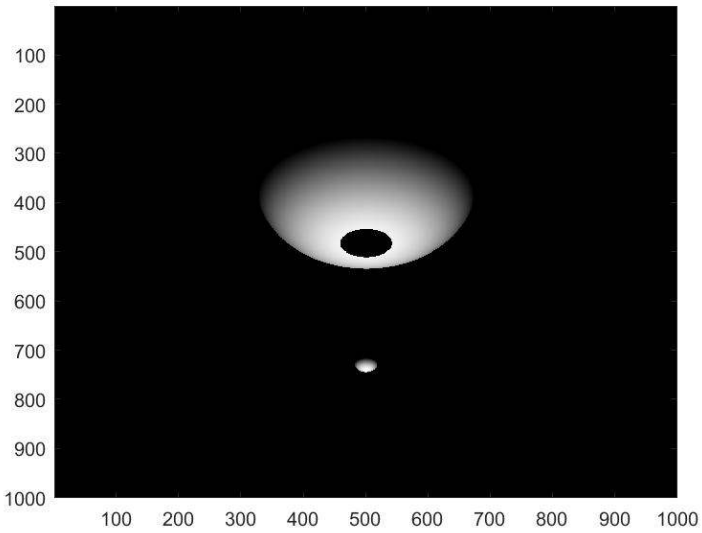
Da das Licht der Lampe die Kugel nur im Bereich von $90° \leq \alpha \leq 180°$ erreicht, unterscheidet man hier zwischen zwei Fällen, die mit einer If-Abfrage beschrieben werden:

- Wenn $cos\,\alpha \leq 0$ zutrifft, wird der Punkt P auf der Kugel beleuchtet.
- Wenn allerdings $cos\,\alpha > 0$ zutrifft, wird dieser Bereich nicht beleuchtet beziehungsweise schwarz.



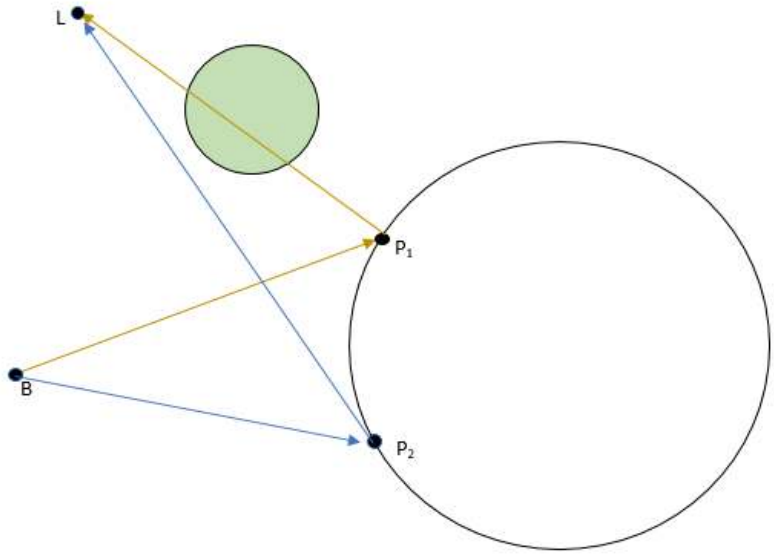
6 Schatten und Eklipsen

Um Objekte in einem dreidimensionalen Raum noch realistischer darzustellen, arbeitet man mit Schatten. Es gibt die Möglichkeit, dass der Schatten eines Objekts auf ein anderes Objekt oder auf eine Ebene fällt. Bei der Programmierung von Schatten geht es eigentlich nur darum, zu prüfen, wo der Strahl am frühesten, genauer gesagt mit dem kleinsten Abstand, auf ein Objekt trifft.



Die Programmierung wird dabei so verändert, dass wenn der Strahl der Lichtquelle schon einmal einen Schnittpunkt mit einem Objekt hatte, das dahinterliegende Objekt nicht mehr beleuchtet wird.

Beim darauffolgenden Beispiel wird Punkt P_1 nicht beleuchtet, da eine weitere Kugel den zweiten Strahl, der von P_1 zur Lampe L vom Algorithmus ausgesendet wird, verdeckt. Punkt P_2 ist jedoch für den Beobachter B sichtbar, weil beide Strahlen nicht auf ein anderes Objekt treffen.

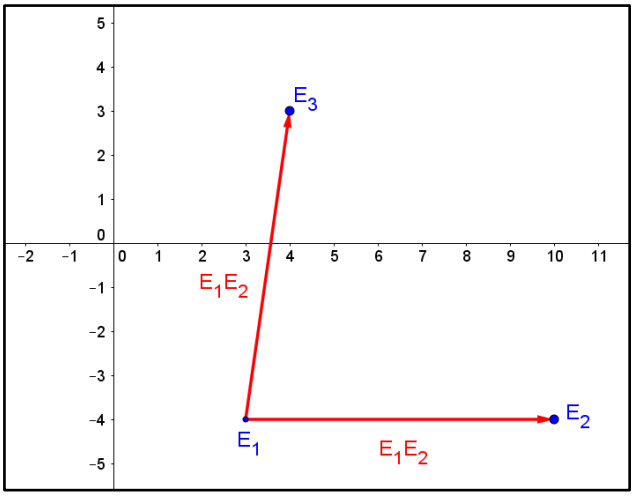


7 Ebenen

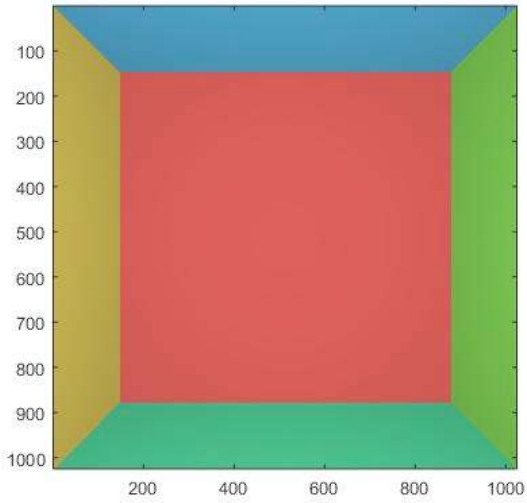
Um eine Ebene mathematisch zu definieren, benötigt man im Grunde nur 3 Punkte. Wir nennen sie E_1, E_2, und E_3. Ein Punkt, wir nennen ihn hier P, liegt genau dann auf der Ebene, wenn folgendes gilt:

$$P = E_1 + t_1 * \overrightarrow{E_1E_2} + t_2 * \overrightarrow{E_1E_3},$$

wobei t_1 und t_2 zwei reelle Zahlen sind.



Die Strecke $\overrightarrow{E_1E_2}$ ist der eine Richtungsvektor. Der andere Richtungsvektor wird durch die Strecke $\overrightarrow{E_1E_3}$ beschrieben. Die Zahlen t_1 und t_2 sind in unserem Fall die Parameter. Mit diesen Informationen können wir alle Punkte auf dieser Ebene beschreiben.



Als nächstes versuchen wir, den Schnittpunkt zwischen einer Ebene und einer Geraden zu berechnen. Die Gerade wird später in unserem Fall ein Strahl sein. Um einen Schnittpunkt zu berechnen, benötigen wir also auf jeden Fall eine Gerade und eine Ebene. Beide stellen wir in Parameterform dar. Bevor wir den Schnittpunkt berechnen, zeigen wir noch einmal die beiden Definitionen.

Gerade: $$P = B + s * \overrightarrow{BK}$$

Ebene: $$P = E_1 + t_1 * \overrightarrow{E_1E_2} + t_2 * \overrightarrow{E_1E_3}$$

Nun setzen wir die beiden Gleichungen gleich:

$$B + s * \overrightarrow{BK} = E_1 + t_1 * \overrightarrow{E_1E_2} + t_2 * \overrightarrow{E_1E_3}$$

$$B - E_1 = t_1 * \overrightarrow{E_1E_2} + t_2 * \overrightarrow{E_1E_3} - s * \overrightarrow{BK}$$

Es entsteht ein Gleichungssystem mit den 3 Unbekannten s, t_1, t_2. Da sowohl die beiden Ausganspunkte B und E_1 als auch die Vektoren jeweils 3 Koordinaten besitzen, bekommen wir 3 Gleichungssysteme und wir können das Gleichungssystem mit 3 unbekannten lösen. Nun erstellen wir für Vektoren $\overrightarrow{E_1E_2}$, $\overrightarrow{E_1E_3}$ und $\overrightarrow{BK}$ eine Matrix, mat, und können die Gleichung folgendermaßen umformen:

$$mat * \begin{pmatrix} t_1 \\ t_2 \\ s \end{pmatrix} = B - E1,$$

wobei : $$mat = \begin{pmatrix} \overrightarrow{E_1E_2} & \overrightarrow{E_1E_3} & -\overrightarrow{BK} \end{pmatrix}.$$

Dann multiplizieren wir die ganze Gleichung mit der inversen Matrix, mat^{-1}. Dadurch können wir die 3 Parameter berechnen. Wir müssen die inverse Matrix nicht selber berechnen, da Matlab das selbst lösen kann.

$$mat^{-1} * mat * \begin{pmatrix} t_1 \\ t_2 \\ s \end{pmatrix} = mat^{-1} * (B - E1)$$

$$\begin{pmatrix} t_1 \\ t_2 \\ s \end{pmatrix} = mat^{-1} * (B - E1)$$

Allerdings kann man diese Methode in Sonderfällen nicht anwenden. Es könnte sein, dass die Gerade parallel zur Ebene ist. In diesem Fall gibt es

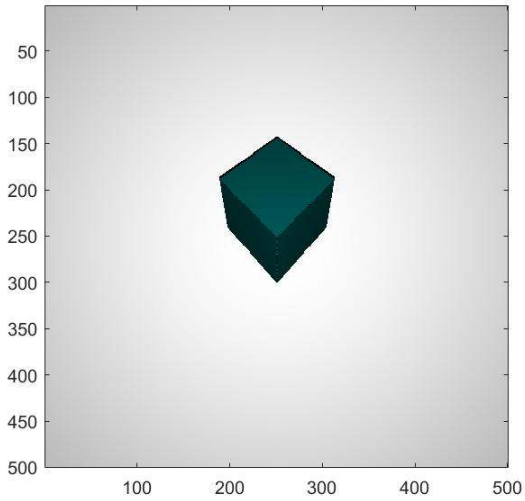
keine inverse Matrix und somit auch keine Lösung für das Gleichungssystem. Der zweite Sonderfall tritt dann auf, wenn die Gerade in der Ebene liegt. Hierbei gibt es dann unendlich viele Lösungen. In der Praxis haben wir die Positionen des Beobachters und der Lichtquelle so gewählt, dass die Sonderfälle nicht auftreten können.

8 Würfel

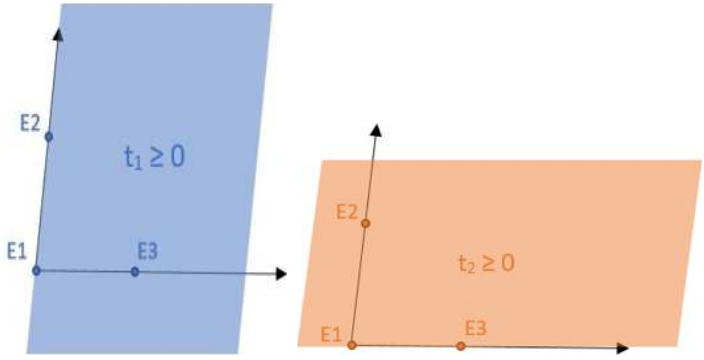
Jeder Punkt der Ebene, auf der das Quadrat liegt, lässt sich durch eine einfache Formel berechnen:

$$P = E_1 + t_1 * \overrightarrow{E_1E_2} + t_2 * \overrightarrow{E_1E_3}$$

wobei E_1, E_2 und E_3 die gegebenen Eckpunkte des Quadrats sind. Wird für t_1 und t_2 1 eingesetzt, erhält man den vierten Eckpunkt E_4, bei t_1 und t_2 gleich 0, ergibt sich der Punkt E_1.

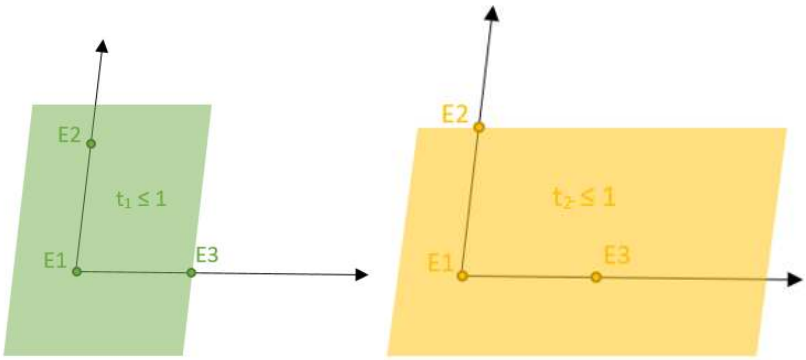


In der Graphik unten sieht man die Halbebenen die durch die Veränderung der Parameter t_1 und t_2 entstehen.



Ein Punkt liegt genau dann im Quadrat, wenn er sich in der Schnittmenge von allen vier Halbebenen befindet. Daraus lässt sich schließen, dass jeder

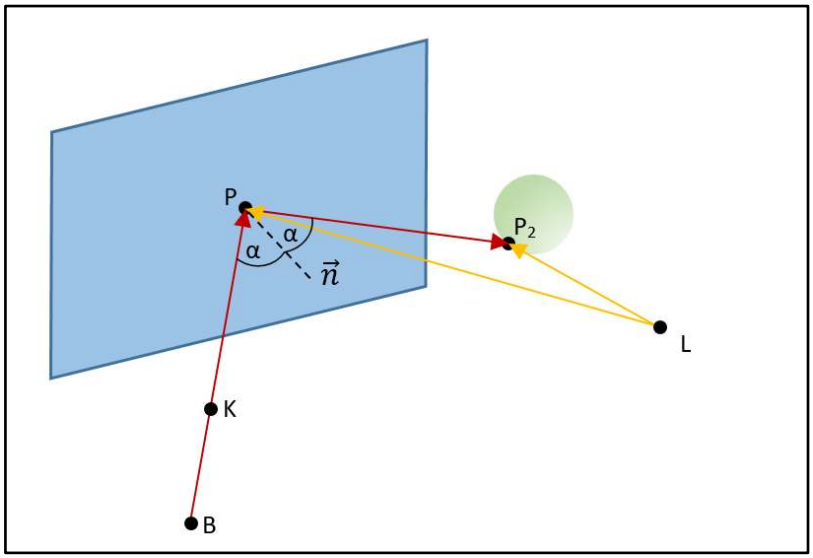
Punkt P, bei dem t_1 und t_2 sowohl größer als null, als auch kleiner als eins sind, auf der Fläche des Quadrats liegt.



Um daraufhin einen Würfel zu erstellen, mussten nur noch sechs Quadrate zusammengefügt werden. Die Beleuchtung für diese wurde auf die gleiche Weise berechnet wie bei Ebenen.

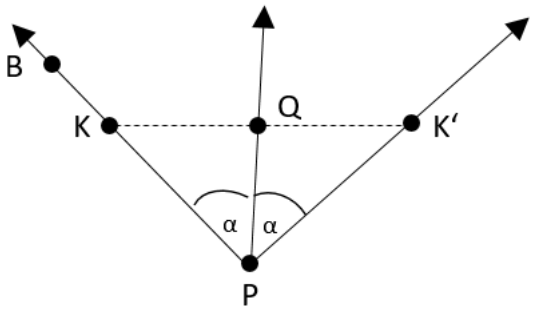
9 Spiegelung

Damit der Eindruck einer Spiegelung besteht, müssen die Strahlen eines Objektes auf einer Spiegelfläche reflektiert werden. Zur Vereinfachung gehen wir nicht den natürlichen Weg der Strahlen, also vom Licht zum Objekt und anschließend zum Beobachter, sondern in umgekehrter Reihenfolge.



Unsere Ebene wird dabei als spiegelnde Oberfläche definiert, deren Spiegelung auf den Gesetzen der Optik basiert. Der Einfallswinkel des Vektors $\overrightarrow{BP}$ fällt im Winkel α auf die Ebene ein und wird auch im selben

Winkel wieder reflektiert. Dabei wird errechnet, ob der reflektierte Vektor einen Schnittpunkt P_2 mit einem Objekt hat. Ist dies der Fall, so entsteht ein Spiegelbild auf der Ebene im Punkt P. Dafür muss aber die Voraussetzung gegeben sein, dass der Punkt P_2 überhaupt erst von einem Lichtstrahl beleuchtet werden kann.



Zur Berechnung des Vektors $\overrightarrow{PK'}$ und somit des Strahles von P zu P_2 müssen mehrere Rechenschritte erfolgen.

Zuerst wird $\cos(\alpha)$ durch folgende Ausdrücke definiert:

$$\cos(\alpha) = \frac{\left|\overrightarrow{PQ}\right|}{\left|\overrightarrow{PK}\right|} \quad ; \quad \cos(\alpha) = \frac{\overrightarrow{PK} * \vec{n}}{|\overrightarrow{PK}|}$$

Danach wird $\cos(\alpha)$ durch den zweiten Ausdruck ersetzt, um eine unbekannte Variable zu eliminieren.

$$\left|\overrightarrow{PQ}\right| = \frac{\overrightarrow{PK} * \vec{n}}{\overrightarrow{PK}} * \left|\overrightarrow{PK}\right|$$

Da nun die Länge vom Vektor $\overrightarrow{PQ}$ bekannt ist, lassen sich nun die Koordinaten des Punktes Q berechnen.

$$Q = P + \left|\overrightarrow{PQ}\right| * \vec{n}$$

Um zum Punkt K' zu gelangen, wird am Punkt Q der Vektor $\overrightarrow{KQ}$ angehängt.

$$K' = Q + \overrightarrow{KQ}$$

Nun kann durch den Punkt K' ein Strahl gelegt und somit eruiert werden, ob dieser Strahl auf ein Objekt trifft. Damit die Spiegelung realistisch wirkt, muss noch die genaue Farbkombination errechnet werden. Dies wird im Kapitel Farben erläutert.

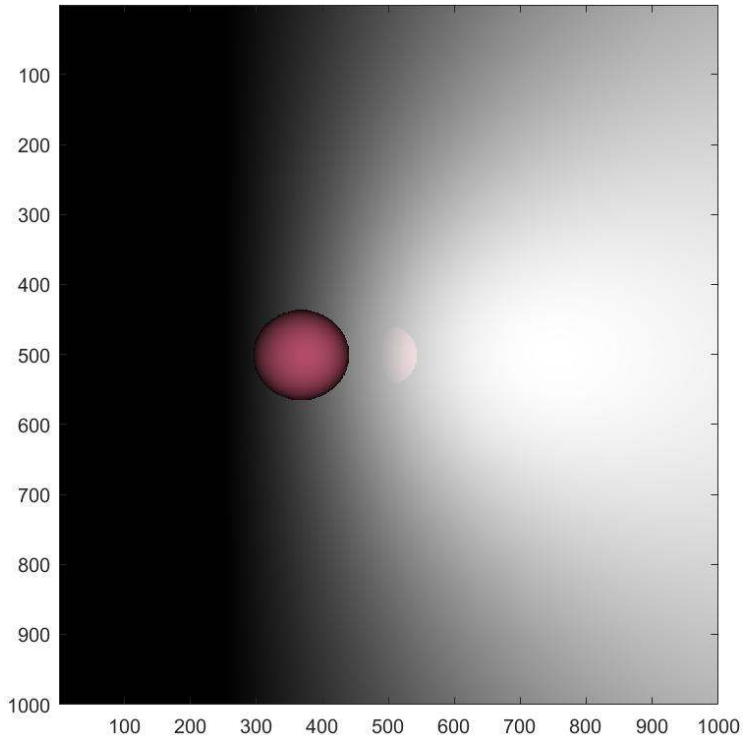


Abbildung 1: Spiegelung einer Kugel in einer Ebene

10 Farben

10.1Farbwahrnehmung in der Natur

Die Sonne sendet Lichtstrahlen aus, die z.B. auf ein grünes Blatt eines Baumes aufprallen. Das Blatt absorbiert alle Wellenlängen außer die grünen Wellenlängen, die mit einen Ausgangswinkel, der gleich wie der Eingangswinkel ist, reflektiert werden. Bestimmte Rezeptoren in den Augen können die Wellenlänge visuell wahrnehmen, wobei die Stäbchen für den Hell/Dunkel-Kontrast verantwortlich sind. Die Farben entstehen letztlich durch die Interpretation der Farbreize im Gehirn.

10.2Darstellung am Computer

Eine Möglichkeit, wie der Computer Farben am Bildschirm darstellt, ist mit den RGB Farbkreis. Die Abkürzung RGB steht für Red, Green und Blue. Jeder dieser drei Farben wird durch eine Zahl beschrieben, die von 0 bis 255 reicht. Der RGB Farbkreis findet vor allen Verwendung bei den Computerbildschirmen und Fernsehermonitoren. Daher wird diese Farbabbildung nicht nur von Matlab verwendet, sondern wird auch von CSS benutzt, um die verschiedensten Farben auf Webseiten verkörpern zu können.

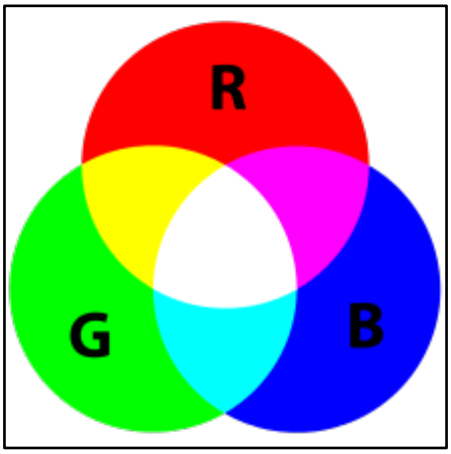


Abbildung 2: RGB Farbkreis[3]

Die beiden Kontraste Schwarz und Weiß kann man mit (0, 0, 0) und (255, 255, 255) beschreiben. Die Farben Rot, Grün und Blau können durch folgende Schreibweise formuliert werden: (255, 0, 0), (0, 255, 0) und (0, 0, 255). Möchte der Benutzer auf Farben wie Gelb, Magenta und Türkis zurückgreifen, muss der User die Farben Rot und Grün so vermischen, dass diese gleichmäßig verteilt sind. So ergibt sich für Gelb (255, 255, 0), Türkis (0, 255, 255) und Magenta (255, 0, 255).

10.3Verwendung in Matlab

Matlab übernimmt die Gestaltung der Farbe vom RGB Farbkreis, verzichtet jedoch auf die ursprüngliche Beschreibung der Farbe von 0 bis 255 und definiert seinen Farbkreis im Intervall von 0 und 1. Folglich muss jede RGB

[3] https://en.wikipedia.org/wiki/File:AdditiveColor.svg

Farbe durch 255 dividiert werden, um seine demensprechende Farbe im Matlab zu erhalten. In der folgenden Tabelle sind die wichtigsten Farben und Kontraste in Matlab zusammengefasst:

Weiß	(1, 1, 1)
Schwarz	(0, 0, 0)
Rot	(1, 0, 0)
Grün	(0, 1, 0)
Blau	(0, 0, 1)
Gelb	(1, 1, 0)
Türkis	(0, 1, 1)
Magenta	(1, 0, 1)

10.4 Färben von Objekten in Matlab

Wie bereits erwähnt, wird ein Bildschirm durch drei gegebene Punkte erzeugt. Dieser Bildschirm setzt sich aus Pixeln zusammen, diese beschreiben auch die Auflösung des entstehenden Bildes. Wenn ein Schnittpunkt mit einen Objekt gefunden wird, welches nicht im Schatten liegt, so wird dieses Objekt in seiner zuvor definierten Farbe im ausgewählten Pixel gefärbt. Anschließend wird die Farbe des Pixels in drei Matrizen red_matrix, green_matrix und blue_matrix in der Zeile „i" und in der Spalte „j" gespeichert, diese werden zum Schluss verwendet, um die Farbe an den bestimmten Pixel zu erzeugen.

Wie bereits im Kapitel „Beleuchtung in 3D" angesprochen wurde, müssen die drei verschieden Matrizen mit $\cos\alpha$ multipliziert werden, um Farbabstufungen, die einem Schatten ähneln, zu erzeugen.

Wenn auf eine Fläche zwei Beleuchtungen aufeinandertreffen, müssen die beiden Lichtstrahlfarben so berechnet werden, sodass die Beleuchtung die zwei Lichtstrahlen miteinbezieht. Mit 1 werden die jeweiligen Lichtstrahlen einzeln subtrahiert und abschließend miteinander multipliziert. Zu guter Letzt wird 1 mit dem vorherigen Ergebnis subtrahiert und so haben wir zwei

verschiedene Lichtquellen berücksichtigt. Diese Berechnung muss für alle 3 Farben durchgeführt werden und wird in einer Matrix gespeichert.

$$finalColor = 1 - (1 - colorSource1) * (1 - colorSource2)$$

Es soll vermerkt werden, dass $colorSource1$ und $colorSource2$ größer gleich 0 und kleiner gleich 1 sein muss. Diese Formel erfüllt zwei Eigenschaften:

- Das Ergebnis von $finalColor$ ist größer als 0 und kleiner als 1.
- Das Ergebnis $finalColor$ ist größer als $colorSource1$ und größer als $colorSource2$.

Um das Verständnis dieser Formel zu gewährleisten, wird sie anhand eines Beispiels verdeutlicht:

$$colorSource1 = 0.8 \; und \; colorSource2 = 0.35$$

$$finalColor = 1 - (1 - 0.8) * (1 - 0.35)$$

$$finalColor = 1 - 0.2 * 0.65$$

$$finalColor = 1 - 0.13$$

$$finalColor = 0.87$$

11 Unser Matlab Programm

Anschließend beginnt in Zeile 8 die for-Schleife, welche die Normalvektoren aller Ebenen berechnet und anschließend im Vektor plane_normal_vec in Form eines Einheitsvektors. Dies wird später wichtig für die Berechnung des Schattens und der Spiegelung. Dies wird in Zeile 20 analog bei Vierecken durchgeführt.

```
1 -   red_Matrix = zeros(nO, nU);
2 -   green_Matrix = zeros(nO, nU);
3 -   blue_Matrix = zeros(nO, nU);
4
5 -   m_Ebenen = size(plane_point1_list, 2);
6 -   plane_normal_vec = zeros(3, m_Ebenen);
7
8 -   for i=1:m_Ebenen
9 -       plane_normal_vec(:, i) = ...
10            cross(plane_point2_list(:, i)...
11            -plane_point1_list(:, i),...
12            plane_point3_list(:, i)-plane_point1_list(:, i));
13 -       plane_normal_vec(:, i)=...
14            plane_normal_vec(:, i)/norm(plane_normal_vec(:, i));
15 -   end
16
17 -   m_square = size(square_point1_list, 2);
18 -   square_normal_vec = zeros(3, m_square);
19
20 -   for i=1:m_square
21 -       square_normal_vec(:, i) = ...
22            cross(square_point2_list(:, i) ...
23            - square_point1_list(:, i), ...
24            square_point3_list(:, i) - square_point1_list(:, i));
25 -       square_normal_vec(:, i)= ...
26            square_normal_vec(:, i)/norm(square_normal_vec(:, i));
27 -   end
28
```

In Zeile 29 beginnt der Hauptteil unserer Programmierung in Form einer for-Schleife, wobei die Werte für jedes Pixel, das ja durch die waagrechten und senkrechten Koordinatenangaben genau bestimmt ist, einzeln berechnet werden. Durch den Punkt K wird ein Strahl vom Beobachter ausgesandt und im Anschluss eruiert, ob eine Schnittstelle mit einem Objekt besteht. Die genaue Programmierung findet sich in der Function intersect_line_objects(args).

Wenn ein Schnittpunkt besteht (also für e == 1), muss die Farbe des Punktes P auf dem Objekt neu bestimmt werden. Die Werte wurden in der Funktion lightning(args) genau errechnet und in die Matrizen der Farbe für das ganze Bild übertragen.

Nun wird der Sonderfall einer spiegelnden Ebene betrachtet: Wenn die Funktion intersect_line_object(args) einen Schnittpunkt mit einer Ebene erkennt, so wird der einfallende Strahl im gleichen Winkel reflektiert. Die

exakte Berechnung wird von der Funktion Symmetry(args) übernommen. Wenn dieser Strahl wiederum auf ein Objekt fällt, so entsteht die Spiegelung und die Farben jener Punkte werden mit der speziellen Funktion.

```
for i=1:nO
    for j=1:nU
        K = F1 + (i-1)*FO + (j-1)*FU;
        [P,e,k,form] = intersect_line_objects(B,K,ball_center_list,ball_radius_list,...
            plane_point1_list,plane_point2_list,plane_point3_list, ...
            square_point1_list, square_point2_list, square_point3_list);
        if e == 1
            color_output = lightning(P,form,k,L,ball_center_list,ball_radius_list, ...
                ball_color_list, plane_point1_list,plane_point2_list,plane_point3_list, ...
                plane_normal_vec, plane_color_list, ...
                square_point1_list, square_point2_list, square_point3_list, ...
                square_normal_vec, square_color_list);


            if strcmp (form, 'Ebene')


                K2 = symetry(P,K,plane_normal_vec(:,k));
                [P2,e2,k2,form2] = intersect_line_objects(P,K2,ball_center_list,ball_radius_list,...
                    plane_point1_list,plane_point2_list,plane_point3_list, ...
                    square_point1_list, square_point2_list, square_point3_list);
                if e2==1
                    color_output2 = lightning(P2,form2,k2,L,ball_center_list,ball_radius_list, ...
                        ball_color_list, plane_point1_list,plane_point2_list,plane_point3_list, ...
                        plane_normal_vec, plane_color_list, ...
                        square_point1_list, square_point2_list, square_point3_list, ...
                        square_normal_vec, square_color_list);
                    color_output = sum_color(color_output,color_output2);
                end
            end
```

sum_colors(args) berechnet und anschließend in die Hauptfarbmatrix übernommen, da ja die Farbe der Spiegelung und die Farbe der Spiegelfläche berücksichtigt werden müssen.

Zu guter Letzt bekommt das Programm die Anweisung, die berechneten Zahlenwerte richtig als Bild darzustellen und die Achsen richtig zu skalieren.

```

            red_Matrix(i, j) = color_output(1);
            green_Matrix(i, j) = color_output(2);
            blue_Matrix(i, j) = color_output(3);

        end
    end
end

picture = cat(3, red_Matrix, green_Matrix, blue_Matrix);
im = image(picture);
axis image


```

12 Fazit

Der siebentägige Exkurs in die Computergrafik hat uns gezeigt, wie man die Vektorrechnung, welche wir in der Schule gelernt haben, praktisch anwenden kann. Des Weiteren haben wir uns intensiv mit Vektorrechnungen in R^3 auseinandergesetzt und wir haben verständnisvoll das Kreuzprodukt und Skalarprodukt eingesetzt. Darüber hinaus haben wir den Betrag von Vektoren, den Einheitsvektor und den Winkel zwischen zwei Vektoren berechnet. Das Erstellen von dreidimensionalen Objekten hat uns vor Augen geführt, wie kompliziert und anspruchsvoll das Rendern dieser Objekte ist.

Moderne Darstellungen von dreidimensionalen Objekten haben sich unseren Respekt verdient, da wir die Komplexität und Arbeit, die hinter solchen Aufgaben steckt, jetzt nachvollziehen können. Außerdem haben wir die Grundlagen des Programmierens kennengelernt, denn wir verstehen nun das Prinzip von Variablen, Schleifen, Funktionen und If-Else Abfragen. Nicht zu vergessen ist, dass wir uns oft auf Fehlersuche in unseren Programmen begeben haben müssen und auch großteils selbstständig ausgebessert haben. Zum Schluss haben wir das Rechnen mit Matrizen kennengelernt: Wie man eine Matrix erstellt und welche Regeln man beachten muss, wenn man mit Matrizen arbeitet.

Unser Verständnis von Mathematik hat sich in dieser Zeit grundlegend verändert und uns wurden viele neue Möglichkeiten der praktischen Anwendung offenbart. Je tiefer wir in diese Welt eingetaucht sind, desto mehr erkannten wir die eigentliche Komplexität, die dahinter steckt und uns fasziniert: „Scio me nihil scire“[4].

Zu guter Letzt möchten wir jenen Personen danken, die unser Projekt erst möglich gemacht haben, allem voran unserem Betreuer Laurent. Merci beaucoup! Nous avons vraiment apprecié le temps avec toi!

[4] Lat. Ich weiß, was ich nicht weiß.

Biologie

Problemstellung

Das erste Ziel des Projekts ist, die Dynamik einerseits von Reaktion und andererseits Diffusion zu verstehen und zu simulieren. Erst dann kann man versuchen, sich vorzustellen, wie diese Mechanismen miteinander konkurrieren können, um den Zustand eines Systems zu bestimmen. Erstaunlicherweise können die jeweiligen stabilen Fließgleichgewichtszustände mit reiner Reaktion oder mit reiner Diffusion besonders einfach sein, während die Konkurrenz zwischen den Mechanismen dazu führen kann, dass ein komplexes Muster im System entsteht. Die letzte größte Herausforderung des Projekts ist dann, die spontane Entwicklung von Fasern, wie z.B. Nerven oder Blutgefäße, durch die natürlichen Mechanismen von Reaktion und Diffusion zu simulieren.

Ziele des Projekts:

- Grundlagen von Matlab-Programmierung
- Diffusion eindimensional darzustellen
- mit Matlab verschiedene Muster zu erstellen
- Musterbildung zu beeinflussen
- Musterbildung zu kontrollieren
- Vertiefung in verschiedene Bereiche

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

Team: Ekam Felizia
Farajov Fuad
Guo Xiang-He
Kölbl Sebastian
Schuhmann Liv
Tscheppe Niklas

1. Grundlagen

1.1. Wie man Matrizen miteinander multipliziert

Man berechnet das Skalarprodukt der jeweiligen Zeilen- und Spaltenvektoren und erhält so einen weiteren Vektor.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e \\ f \end{bmatrix} = \begin{matrix} a*e+b*f=g \\ c*e+d*f=h \end{matrix} = \begin{bmatrix} g \\ h \end{bmatrix}$$

1.2. Diffusion und Reaktion

Wieso funktioniert das überhaupt?

Wir haben eine Formel von Francis Crick verwendet. Diese Formel zeigt Diffusion:

$$\frac{\partial C(x,t)}{\partial t} = D\frac{\partial^2 C(x,t)}{\partial x^2},$$

wobei zutreffende Parameter sind:
L … die Länge des Intervalls
t … die Zeit in Sekunden
n… die Anzahl der Zellen im besagten Intervall
l… die Länge der einzelnen Zellen in Intervall
D… die Diffusionskonstante in cm^2s^{-1}
τ… die Degradationsrate
ρ… die Syntheserate

C(x,t) ist die Konzentration der Chemikalie an Position x zur Zeit t .
Er hat folgende Randbedingungen aufgestellt:

$C(0,t)= C_0$
$C(L,t)= 0$
Crick hat sich überlegt wie man eine nicht triviale Lösung mit dieser Formel erreicht und hat daher die linke Seite 0 gleichgesetzt. Es ergibt sich dann die Differentialgleichung, mit Crick´s Randbedingungen, $C(0,t) = C_0 > 0$, $C(L, t)= 0$.

$$\frac{\partial^2 C(x,t)}{\partial x^2} = 0,$$

Dadurch wird das Konzentrationsprofil eine zeitunabhängige gerade Linie. Crick hat abgeschätzt, dass die Zeit die es braucht um einen solchen, wie oben gegebenen, Gradienten aufzustellen gegeben ist durch:

$$t - \frac{A(nl)^2}{D},$$

Eines der wichtigsten Chemikalien eines Embryos ist das Biocid. Anhand dieser Chemikalie und deren Wachstum und Zerfall haben wir modelliert, dass eben diese voneinander abhängig sind.

$$\frac{\partial C(x,t)}{\partial t} = D(t)\frac{\partial^2 C(x,t)}{\partial x^2} - \frac{1}{\tau}C(x,t) + \rho(x,t),$$

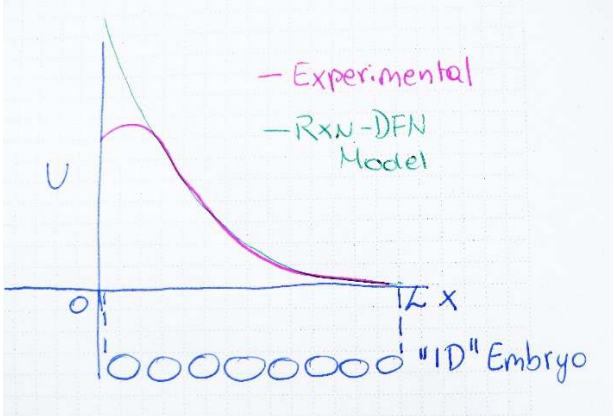
und eine einfachere Version davon wäre dann

$$\frac{\partial C(x,t)}{\partial t} = D\frac{\partial^2 C(x,t)}{\partial x^2} - \frac{1}{\tau}C(x,t),$$

und davon können wir dann einen stabilen stationären Gradient herleiten

$$C(x,t) = C_0 e^{-x/\lambda}, \quad \lambda = \sqrt{D\tau}$$

welcher realistischer ist als das obige lineare Profil.
Dieses Beispiel zeigt, dass Reaktion und Diffusion wichtig sind, um biologische Modelle realistischer darstellen zu können.



1.3. Räumliche Darstellung von Reaktion und Diffusion

Diese braucht man um z.B. die Population von Plankton welche nur unter

bestimmten Bedingungen wie der adequaten Wassertemperatur- und Menge überlebensfähig ist zu berechnen wann die Anzahl an Plankton steigen wird, senken wird oder es gleichbleiben wird.
Wieder verwenden wir Randbedingungen, diesmal um jeden Plankton, der über den Rand hinauswächst, abzutöten, damit es nicht zu einer Überpopulation kommt und die Konzentration konstant auf t = 0 bleibt

$$c(0,t) = 0 = c(L,t)$$
$$c(x,0) = c_0.$$

Die Diffusionsgleichung beschreibt die Konzentration ohne Wachstum oder Zerfall

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}.$$

Weil jetzt Plankton aber lebende Organismen sind, reproduzieren sie in einer Rate, proportional zu ihrer Konzentration. Deswegen gibt es eine Konstante in diesen Term.

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2} + Kc,$$

Wir können diese Formel aber vereinfachen, in dem wir den diffusionslosen Exponenzialwachstum ausklammert

$$c(x,t) = f(x,t)e^{Kt},$$

und das dann in die Reaktions- und Diffusionsgleichung davor einsetzen. Dadurch finden wir, dass f die folgende Diffusionsgleichung erfüllt.

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2}.$$

Wenn wir die Anfangsbedingungen C(x,0) = c_0(x) beachten, finden wir mit der sogenannte Fourierdarstellung der Anfangswerten heraus, dass

$$f = \sum_{n=1}^{\infty} B_n \sin\left(\frac{n\pi x}{L}\right) e^{-n^2\pi^2 D/L^2 t},$$

Wobei B_n

$$B_n = \frac{2}{L}\int_0^L c_0 \sin\left(\frac{n\pi x}{L}\right) \mathrm{d}x,$$

n = 1,2,…, welche dann für die Konzentration C_0 errechnen kann.

$$c(x,t) = \sum_{n=1}^{\infty} B_n \sin\left(\frac{n\pi x}{L}\right) e^{(K - n^2\pi^2 D/L^2)t}.$$

Dieser Zeit-Term steuert das Verhalten des Systems

$$e^{(K - n^2\pi^2 D/L^2)t}.$$

Das Vorzeichen des Exponenten zeigt wie sich die Population entwickeln wird.

Es gibt 3 Möglichkeiten wie sich das ganze entwickeln wird.

Wenn L = Lc die Population bleibt gleich

Nur wenn diese Bedingungen erfüllt sind, entsteht ein Muster mit Matlab

Wenn $L > L_c$ die Population vergrößert sich drastisch

Wenn $L < L_c$. die Population stirbt schnell aus

2. Randbedingungen

Randbedingungen sind bei einer Differenzialgleichung die Werte, die auf dem Rand des Definitionsbereiches für die Normalableitung der Lösung vorgegeben werden. Die drei Rahmenbedingungen sind die periodische, die Dirichlet'sche und die Neumann'sche Randbedingungen.

- Bei der periodischen Randbedingung wird angenommen, dass es einen Fluss zwischen den Enden der gesuchten Zellen gibt. Daher wird immer die Zelle mit ihrem Vorgänger subtrahiert und bei der ersten Zelle ist der Vorgänger der letzte.

  u_1-u_4, u_2-u_1, u_3-u_2, u_4-u_3

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{vmatrix} \otimes \begin{vmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{vmatrix} \otimes \frac{1}{dx} = Dx$$

- Bei der Dirichlet'schen Randbedingung wird angenommen, dass die nicht gesuchten Werte 0 sind. Das bedeutet, dass die erste und die letzte Zelle mit 0 subtrahiert wird und so gleichbleibt.

u_1-0, u_2-u_1, u_3-u_2, u_4-u_3, 0-u_4,

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{vmatrix} \times \begin{vmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{vmatrix} \times \frac{1}{dx} = Dx$$

- Die Neumann'sche Randbedingung besagt, dass es keinen Fluss zwischen den Zellen gibt und somit werden nur die innen liegenden Werte definiert.

u_2-u_1, u_3-u_2, u_4-u_3

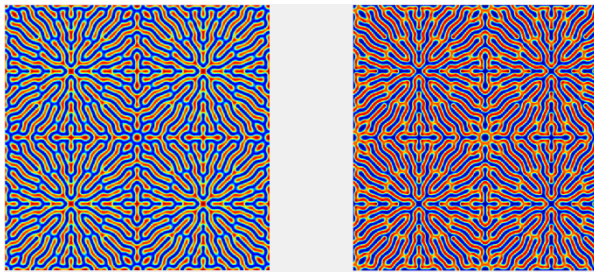
$$\begin{vmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{vmatrix} \times \begin{vmatrix} u_1 \\ u_2 \\ u_3 \end{vmatrix} \times \frac{1}{dx} = Dx$$

Man kann die Randbedingungen z.B. auch beim Swift Hohenberg Modell und im Gray Scott Modell anwenden. Die Einfügungen der Randbedingung in diese Modelle haben verschiedene Auswirkungen auf das Ergebnis.

<u>Gray Scott Modell</u>

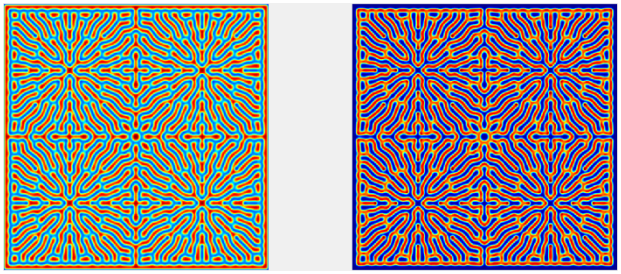
Periodische Randbedingung:

- die Korallen breiten sich immer weiter aus und gehen irgendwann in den Rand über



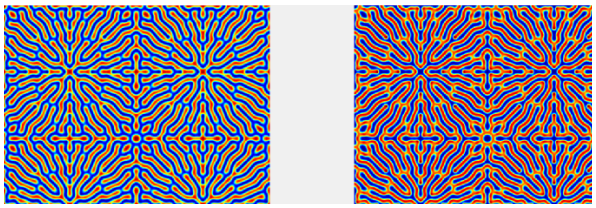
Dirichlet'sche Randbedingung:

- die Korallen breiten sich bis zum Rand aus, aber gehen nicht in ihn über



Neumann'sche Randbedingung:

- die Korallen breiten sich soweit aus, bis sie gerade in den Rand übergehen



3. Bifikationsmodell

$u = u*(1-u^2)$

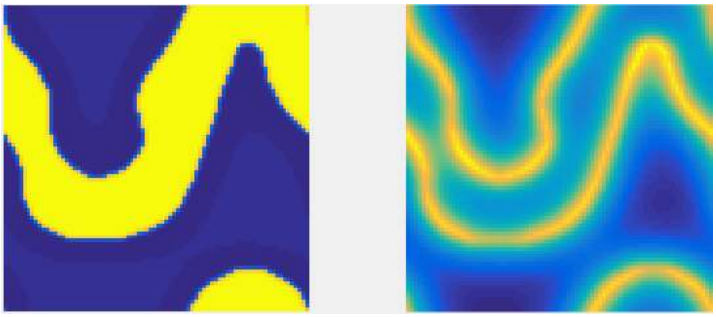
$u = u + d_t* \delta *(I - L*Du)^{2}* u + d_t*\gamma*u.*(1-u^2)$
gelb und blau sind stabil

$u = u - d_t* \delta* L* v + d_t* \alpha*(v.^2-u.^2)$
$v = v – d_t* \delta *L*v + d_t* \alpha *(E^2-u.^2)$

$d_t = 0.5$ $\delta = 0.1/max(I(:))$ $\gamma = 0.5$ $E = 0.5$ $\alpha = 0.001$

$u.^2$… jede Komponente der Matrix quadriert wird
dt… t(2)-t(1), Änderung von der Zeit t

Nach dem Aufstellen dieser Formel haben wir versucht, die Parameter so einzustellen, dass wir ein stabiles Bild herausbekommen. Mit den oberen Parametern



bekamen wir folgendes Bild:

Wir nehmen an, dass es sich dabei um einen stabilen gelben und blauen Bereich handelt, aber am Rand zwischen den zwei Beriechen ist u ungefähr 0 und das führt dazu, dass v größer ist, wie man an den gelben Kurven im linken Bild sieht.
Wenn u also ca. 0 ist, wird $(ep^2-u.^2)$ positiv, was eine Vergrößerung von v bewirkt. Je größer also v ist, desto größer wird auch u.

>wenn die Steigung größer als 0 ist, ist die Funktion unstabil
>wenn die Steigung kleiner als 0 ist, wirkt an der Nullstelle ein starkes Gleichgewicht

4. Glättung von Kurven

Damit eine Funktion stabil wird, wird die Graphenkurve geglättet. Die Glättung selbst beschreibt den Vorgang, eine Kurve in eine Kurve mit geringerer Krümmung hinüberzuführen, die aber trotzdem nicht zu weit vom Original abweicht.

Wenn der Punkt über der Geraden liegt, ist die zweite Ableitung von u_t negativ und der Graph glättet sich nach unten. Bei einem unter der Geraden liegenden Punkt ist die zweite Ableitung von u_t positiv du der Graph glättet sich nach oben.

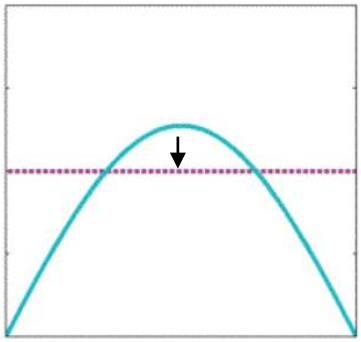


Abb.[1]

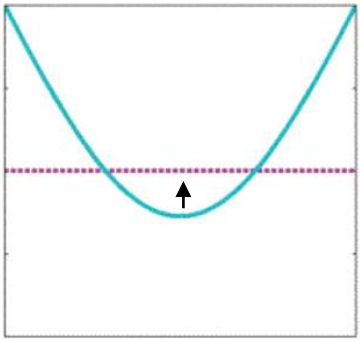
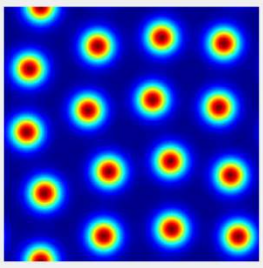


Abb.[2]

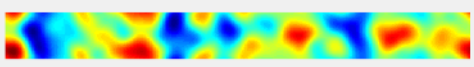
5. Die Mathematik der Punkten- und Streifenbildung

Wenn nur wenig Platz zum Ausbreiten zur Verfügung steht, wird das Muster nur in eine Länge dargestellt und man erkennt das dann als Streifen.
Wenn mehr Platz zum Ausbreiten zur Verfügung steht, erkennt man Punkte, weil sich das Muster in beide Richtungen ausbreitet.
Dieses Phänomen kann man auch bei Tieren erkennen. Es schlägt vor, dass aufgrund der Reaktion zwischen zwei oder mehr chemischen Spezies sich verschiedene Muster entwickeln können. Deshalb kann z.B. ein Jaguar komplett gestreiften Schwanz oder einen halb-gestreiften und halb-gepunkteten Schwanz haben, weil der Schwanz in den Entwicklungsstufen unterschiedliche Größen hat.

Leoparden Flecken



Leopard Streifen



[1] www.theshapeofmath/princeton.com
[2] www.theshapeofmath/princeton.com

Aber nicht alle Reaktion-Diffusions-Modelle erzeugen Muster durch die gleichen Mechanismen. Schon kleine Veränderungen der Parameter können zu verschiedenen Mustern führen.

$$\frac{\partial u}{\partial t} = D_u \frac{\partial^2 u}{\partial x^2} - uv^2 + F(1-u),$$
$$\frac{\partial v}{\partial t} = D_v \frac{\partial^2 v}{\partial x^2} + uv^2 - (F+c)v.$$

D_u ,D_v… Diffusions-Konstanten
F ,c… Konstanten
dt… Veränderung der Zeit
u… Ausgangsstoff

6. Swift Hohenberg Gleichung

6.1. Allgemein

Die Swift-Hohenberg Gleichung wurde nach den amerikanischen Wissenschaftlern Jack B. Swift und Pierre C. Hohenburg benannt. Sie ist eine mathematische Modellgleichung die zur Untersuchung von Musterbildungsprozessen benutzt wird. Mit ihr kann zum Beispiel das Wachstum der Papillarleisten an den Fingern beschrieben werden, welche das Muster von Fingerabdrücken bestimmen. Auch kann man mit ihr thermische Konvektion beschreiben, also das Verhalten von Flüssigkeiten, wenn diese zum Beispiel erhitzt werden.

6.1.1. Gleichung

$$\delta p/ \delta t = E * p - (\nabla^2+1)^2 * p + \gamma * p^2 - p^3$$

Die Gleichung beschreibt die Änderung von p mit der Zeit, wobei p zum Beispiel für die Größe von Papillaren oder die Temperatur einer Flüssigkeit stehen kann. E*p lässt p exponentiell wachsen. Das Wachstum wird von der Diffusionsgleichung $(\nabla)^2+1$ gehemmt. $\gamma * p^2$ und p^3 sind dabei nicht Linearitäten und γ kontrolliert die quadratische Nichtlinearität von p.

Um ein Muster zu erzeugen, muss man E so wählen, dass es größer als Null ist. Erst dann bilden sich Amplituden mit überkritischen Wellenzahlen, welche dann Muster bilden. Wählt man E größer als 0 reicht der Temperaturunterschied jedoch und die Flüssigkeit beginnt zu fließen. Die Fließgeschwindigkeit würde exponential steigen, wenn man sie nicht mit der Diffusionsgleichung bremsen würde. Diese versucht Krümmungen zu glätten, damit sich alle Werte durch Diffusion angleichen. Wenn man E so wählt, dass die Diffusion und die Konvektion im Gleichgewicht sind, ensteht ein stabiles Muster.

Wählt man für E eine negative Zahl, so konvergieren die Werte von p gegen Null und es formt sich kein Muster. Das liegt daran, dass man E auch als den Unterschied zwischen der kleinsten Temperatur und der Temperatur im betrachteten Punkt ist. Wenn E kleiner als 0 ist, reicht die Temperatur nicht aus um eine Konvektion zu erzeugen.

6.2. Umsetzung

Diese Gleichung haben wir in Matlab umgesetzt. Dazu mussten wir zunächst einmal die Ableitungsmatrix L von der Einheitsmatrix I abziehen. Im Gegensatz zu den vorherigen Gleichungen muss man für die Swift-Hohenberg Gleichung diesen Ausdruck, (I - L), quadrieren. Diese neue Matrix haben wir we genannt.

$we = (I - L)^2;$

Danach haben wir eine for-Schleife mit der Gleichung geschrieben. Auch hier kann man die Gleichung sowohl explizit als auch implizit lösen.

```
p = p - dt*we*p + dt*f  - explizit
p = (I+dt*we)\(p+f*dt);  - implizit
```

Im Code haben wir hierbei p als Vektor definiert, der alle Punkte unserer Fläche enthält. dt steht für die Änderung der Zeit pro Iteration und f ist der fehlende Teil der Gleichung, also:

$f = (es*p) + ge*(p.^2) - (p.^3)$

In die for Schleife haben wir dann noch zwei Zeilen geschrieben die den p-Vektor als Fläche graphisch darstellen.

```
for k = 1 : (Nt-1)
   f = (es*p) + ge*(p.^2) - (p.^3)

   p = p - dt*we*p + dt*f   - explizit
   p = (I+dt*we)\(p+f*dt) – implizit
```

```
    imagesc(reshape(p,Nx,Ny)) - Graphische Darstellung
    colormap jet; axis equal; axis off; drawnow;

end
```

6.2.1. Ableitung

Eine Überlegung unserer Seite war es, dass die Hoch- und Tiefpunkte der Swift Hohenberg Gleichung stabile Werte für p sind, da an jenen Stellen die Steigung 0 ist und somit die Konvektion und Diffusion im Gleichgewicht sind. Nun haben wir die Gleichung abgeleitet. Unsere Ableitung sah dann also in etwa so aus:

$f'(p) = E+2* \gamma*p-3*p^2$

Nun haben wir die selben Parameter, die wir bei Matlab verwendet haben in die Ableitung eingesetzt (E=0.3 und γ=2) und die Extremstellen (-0.071 und 1.404) erhalten.

$p_t = E*p-(I+ \delta)^2*p+ \gamma*p^2-p^3$

$p_t = E*p-p-2\Delta*p-2\Delta*p+ \gamma*p^2-p^3$

Jedoch haben die Werte der Hoch- und Tiefpunkte nicht mit den dargestellten Werten von Matlab übereingestimmt. Nach langen Überlegungen, sind wir auf eine Approximierung zur Bestimmung der Hoch- und Tiefpunkte gekommen.

$\Delta^2*p = (\delta*p)\_xx+(\delta*p)_{yy}$

$\Delta*p = p_{xx}+p_{yy}$

$p = \sin(\omega*x)*e^(\lambda*t)$

1. Ableitung nach t

   p_t = Λ*sin(ω*x)*e^(λ*t)=λ*p

2. Ableitung nach x

   p_xx = - ω^2*sin(ω*x)*e^(λ*t)=- ω^2*p

3. Ableitung von p_t nach x

   p_xxxx = ω^4*p

Das Δ von der ersten Gleichung wird umgeformt nach ω^2

Λ*p = E*p-(I+Δ)^2*p+ γ*p^2-p^3

Danach haben wir durch p dividiert und den Ausdruck (p^2-p^3) weggelassen, da dieser für die Approximation von Λ vernachlässigbar ist.

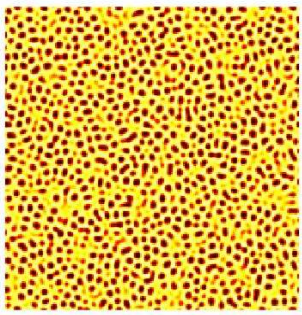
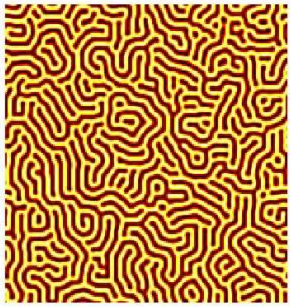
$\Lambda \sim E-(1- \omega^2)^2$

Hierbei können wir E so ändern, dass der gesamte Ausdruck für Λ 0 ist, weil nur dann das Muster stabil bleibt.

6.3. Ergebnisse

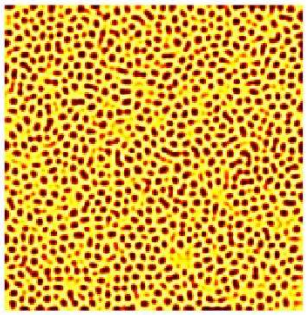
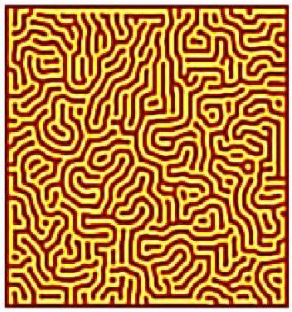
6.3.1. periodische Randbedingungen

Das erste Bild wurde mit zufällig gesetzten Startwerten und periodischen Randbedingungen erzeugt. Nach einiger Zeit bilden sich durch die Diffusion und die Konvektion die abgebildeten Papillarleisten.

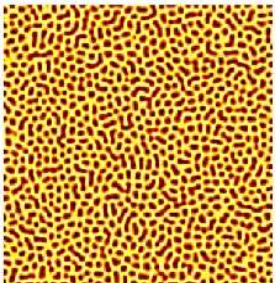
6.3.2. Dirichlet

Das linke Bild zeigt wiederum zufällige Startwerte, diesmal jedoch mit den Randbedingungen von Dirichlet und beim rechten Bild haben sich die Punkte schon zu Papillarleisten verbunden. Auffällig dabei ist der durchgehende Rahmen der dadurch entsteht, dass alle Randwerte gleich 0 sind.

6.3.3. Neumann

Die unteren zwei Bilder zeigen das gleiche wie die vorherigen Bilder, nur mit der Neumann'schen Randbedingung. Wenn man genau hinsiehts kann man erkenne, dass die Papillarleisten gerade werden, bevor sie in den Bildrand übergehen, weil es in der Neumann’schen Randbedingung keinen Fluss zwischen den Zellen gibt.




6.3.4. Bestimmte Werte als Startwerte

Nun haben wir anstatt zufälligen Startwerten, einen bestimmten Punkt in der Mitte des Bildes als Startwert festgelegt. Man kann beobachten, wie sich Ringe um den Kreis bilden.

Wählt man drei Punkte als Startwerte entstehen erneut Ringe um diese Punkte, die

sich jedoch wie unten dargestellt verbinden.

Wählt man einen Strich als Startwert, so bilden sich um diesen längliche Kreise.

6.3.5. Papillarleisten

Schließlich haben wir versucht mit dem Swift Hohenberg Modell einen individuellen Fingerabdruck zu erzeugen. Dabei haben wir zuerst eine kleine Ellipse erstellt, in dem zufällige Startwerte sich ellipsenförmig weiterentwickelten.
Für die Begrenzung haben wir dann eine größere Ellipse erstellt. Das Ergebnis war

ein ziemlich individueller Fingerabdruck, bei dem sich bei jedem Durchlauf einzigartige Merkmale bildeten.

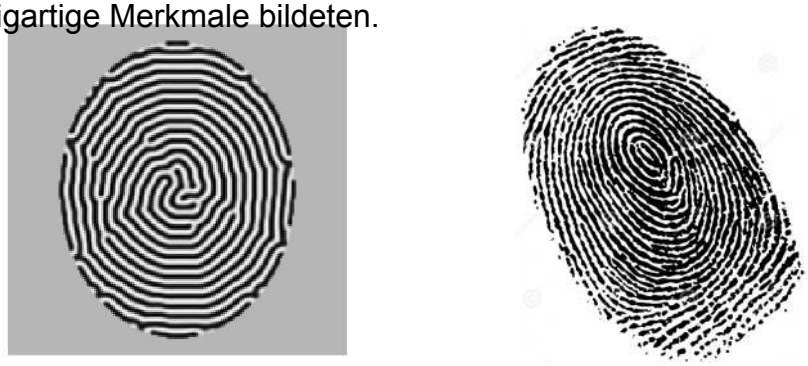


Abb.[3]

7. Coral Growth

Gray-Scott Reaction Diffusion Model

Wir gehen von der Situation, in der Chemikalie A mit zwei Chemikalien B zu insgesamt drei Chemikalien B reagiert, aus:

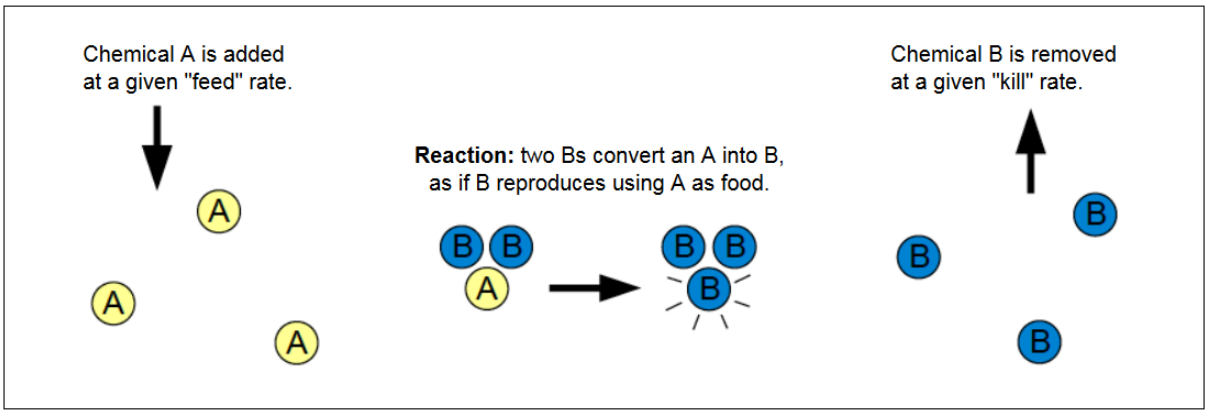


Abb.[4]

Chemikalie A wird mit einer bestimmten „feed-rate“ (im Folgenden als F bezeichnet) zugeführt.

Chemikalie B wird mit einer bestimmten „kill-rate“ (im Folgenden als k bezeichnet) abgeführt/vernichtet.

Du und Dv sind die Diffusionraten

D_u, D_v, F und k sind Konstanten, von ihnen hängt die Funktionstüchtigkeit der Funktion ab.

∇-Operator (siehe umgedrehtes Dreieck[2])

$d_u/d_v/d_t$ sind die Änderung von u/v/t

[3] https://thumbs.dreamstime.com/z/fingerabdruck-1595020.jpg
[4] www.karlsims.com/rd.html

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u - uv^2 + F(1-u),$$
$$\frac{\partial v}{\partial t} = D_v \nabla^2 v + uv^2 - (F+k)v.$$

Die Terme $-uv^2+F(1-u)$ und $+uv^2-(F+k)*v$ haben wir in unserer Gleichung durch r und g definiert.

In diesem Fall ist Chemikalie A als u bezeichnet und Chemikalie B als v bezeichnet worden.

b stellt F und a stell k dar

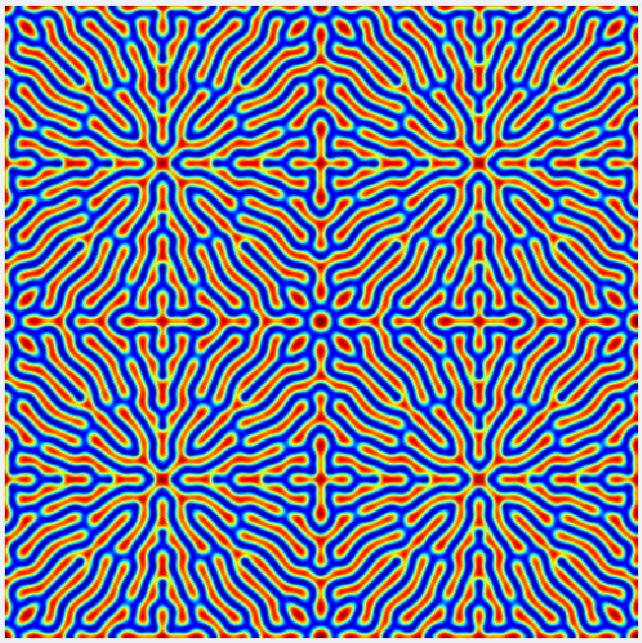
```
r = -u.*v.^2 + b*(1-u);
g = u.*v.^2 - (b+a)*v;

u = (I-Du*dt*L)*u + dt*r;
v = (I-Dv*dt*L)*v + dt*g;
```

```
Du = 1;
Dv = 0.5;
b = 0.0545; a = 0.062;
```

Da die Chemikalie u zu v reagieren will, wird sich der Reaktionsverlauf in die Richtung mit dem meisten u, daher stoßen sich die Korallen ab.

Jedoch bei geringer Entfernung fusionieren die Korallen.

Dies erkennt man am folgenden Bild besonders gut, wenn die Korallen (blau) sich in der Mitte der x- und y-Achse verbinden.



Der Code zum Programmieren dieses Bildes ist auf dem Folgeblatt:

Ändert man die Parameter…

- dt, so ist die Grafik nicht stabil und „explodiert“
- Du/Dv, so diffundieren die Chemikalien anders schnell und man erkennt keine schönen Übergänge
- b/a, so überwiegt die eine Chemikalie und vernichtet die andere (explodiert)

```
T  = 1000;
Nt = 20001;
t = linspace(0,T,Nt);
dt = t(2)-t(1);
dt = 0.89;

Nx = 500;
xmin = 0;
xmax = 1;
x = linspace(xmin, xmax, Nx);
dx = x(2)-x(1);
dx = 1;
Ix = speye(Nx);

Ny = 500;
ymin = 0;
ymax = 1;
y = linspace(ymin, ymax, Ny);
dy = y(2)-y(1);
dy = 1;
Iy = speye(Ny);

%     D1x = spdiags(ones(Nx,1),0,Nx,Nx) ...
%     -spdiags(ones(Nx,1),-1,Nx,Nx);
% D1x(1,Nx)=-1;
%
% Dx = kron(speye(Ny),D1x/dx);
%
%     D1y = spdiags(ones(Ny,1),0,Ny,Ny) ...
%     -spdiags(ones(Ny,1),-1,Ny,Ny);
% D1y(1,Ny)=-1;

% Dy = kron((D1y/dy), speye(Nx));
% L = Dx'*Dx+Dy'*Dy;
% L = L/4;

I = speye(Nx*Ny);

Du = 1;
Dv = 0.5;
b = 0.0545; a = 0.062;

% bcs=1 => per, bcs=2 => Dir, bcs=3 => Neu
bcs=1;

u = ones(Nx,Ny);
v = zeros(Nx,Ny);
v(Nx/2-125,Ny/2-125) = 1;
v(Nx/2+125,Ny/2+125) = 1;
v(Nx/2-125,Ny/2+125) = 1;
v(Nx/2+125,Ny/2-125) = 1;
u = u(:);
v = v(:);

if (true)
     if (bcs==1)
    Dx = spdiags(ones(Nx,1),0,Nx,Nx)...
        -spdiags(ones(Nx,1),-1,Nx,Nx);
    Dx(1,Nx) = -1;
    Dx = kron(Iy,Dx);

    Dy = spdiags(ones(Ny,1),0,Ny,Ny)...
        -spdiags(ones(Ny,1),-1,Ny,Ny);
    Dy(1,Ny) = -1;
    Dy = kron(Dy,Ix);
     elseif (bcs==2)
    Dx = spdiags(ones(Nx,1),0,Nx+1,Nx)...
        -spdiags(ones(Nx,1),-1,Nx+1,Nx);
    Dx = kron(Iy,Dx);

    Dy = spdiags(ones(Ny,1),0,Ny+1,Ny)...
        -spdiags(ones(Ny,1),-1,Ny+1,Ny);
    Dy = kron(Dy,Ix);
     elseif (bcs==3)
    Dx = spdiags(ones(Nx,1),1,Nx-1,Nx)...
        -spdiags(ones(Nx,1),0,Nx-1,Nx);
    Dx = kron(Iy,Dx);

    Dy = spdiags(ones(Ny,1),1,Ny-1,Ny)...
        -spdiags(ones(Ny,1),0,Ny-1,Ny);
    Dy = kron(Dy,Ix);
     else
     end
end
L = Dx'*Dx+Dy'*Dy;
L = L/4;

for k=1:Nt-1
    r = -u.*v.^2 + b*(1-u);
    g = u.*v.^2 - (b+a)*v;

    u = (I-Du*dt*L)*u + dt*r;
    v = (I-Dv*dt*L)*v + dt*g;

    %u = Du*u\-u*v^2+r(1-u*dt);
    %v = Dv*v\+uv^2-(r+g)*v*dt;

    %u = u+dt*b*(I-L*Du)^2*u+dt*a*u.*(1-u.^2);

    Dx = Dx/dx;
    Dy = Dy/dy;

    max(abs(u))
    subplot(1,2,1)
    imagesc(reshape(u,Nx,Ny));
    axis image; axis off; colormap('jet');
    title(['t=',num2str(k*dt)]);
    xlabel('x'); ylabel('u');
    subplot(1,2,2)
    imagesc(reshape(v,Nx,Ny));
    axis image; axis off; colormap('jet');
    title(['t=',num2str(k*dt)]);
    xlabel('x'); ylabel('u');
    drawnow;
end
```

8. Chemische Kinetik

Ein allgemeines Gleichgewicht hat die Form:

$aA + bB \rightleftarrows cC + dD$,

wobei *a,b,c,d* die Anzahl der beteiligten Moleküle darstellt und *A,B,C,D* für bestimmte Chemikalien stehen.

Die Hinreaktion hat die Gleichgewichtskonstante κ_1 und die Rückreaktion κ_2.

Der Ablauf der Reaktion kann folgendermaßen modelliert werden, wobei x(t) die Menge angibt, wie viel mol/l von der Anfangssubstanz sich umwandeln.

- $[A](t) = [A]_0 - a*x(t)$
- $[B](t) = [B]_0 - b*x(t)$
- $[C](t) = [C]_0 + c*x(t)$
- $[D](t) = [D]_0 + d*x(t)$

[A](t), [B](t), [C](t) und [D](t) beschreiben die Konzentrationen zum Zeitpunkt t.

Wenn man x nach t ableitet, erhält man den Reaktionsablauf.

$$\frac{dx}{dt} = -\frac{1}{a}\frac{d[A]}{dt} = -\frac{1}{b}\frac{d[B]}{dt} = \frac{1}{c}\frac{d[C]}{dt} = \frac{1}{d}\frac{d[D]}{dt}$$

Mit den Gleichgewichtskonstanten erhält man

$$= \kappa_1[A]^a[B]^b - \kappa_2[C]^c[D]^d$$

$$= \kappa_1(A_0 - a*x(t))^a(B_0 - b*x(t))^b - \kappa_2\big(C_0 - c*x(t)\big)^c(D_0 - d*x(t))^d$$

Man gehe von dieser einfachen Reaktion aus

$$A + B \rightleftarrows C + D$$

Der Reaktionsablauf wird folgendermaßen beschrieben

$$\frac{dx}{dt} = \kappa_1(\alpha - x)(\beta - x) - \kappa_2(\gamma + x)(\delta + x)$$

wobei $\alpha, \beta, \gamma, \delta$ die Anfangskonzentrationen darstellen.

Man kann dies als Polynom zweiten Grades darstellen:

$$\frac{dx}{dt} = (\kappa_1 - \kappa_2)(x - x_1)(x - x_2)$$

Annahme: $x_1 = 1$, $x_2 = 2$ → dies sind die beiden Gleichgewichtzustände, wobei nur der Zustand 1 stabil ist, wenn man annimmt, dass $\kappa_1 > \kappa_2$ ist.

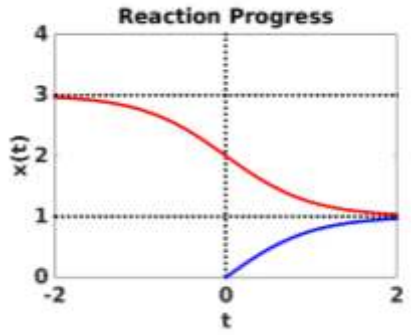


Abb.[5]

Ein anderes Beispiel für eine chemische Reaktion wäre:

$$2A + B \rightleftarrows 2C + D$$

$$x' = \kappa_1(a - 2x)^2(b - x) - \kappa_2(c + 2x)^2(d + x)$$

Man kann auch diesen Term als Polynom dritten Grades darstellen

$$x' = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3$$

Für $\alpha_0 = 0$, $\alpha_1 = r$, $\alpha_2 = 0$, $\alpha_3 = -1$ erhält man:

$$x' = x(r - x^2) = f(x)$$

$$f'(x) = r - 3x^2$$

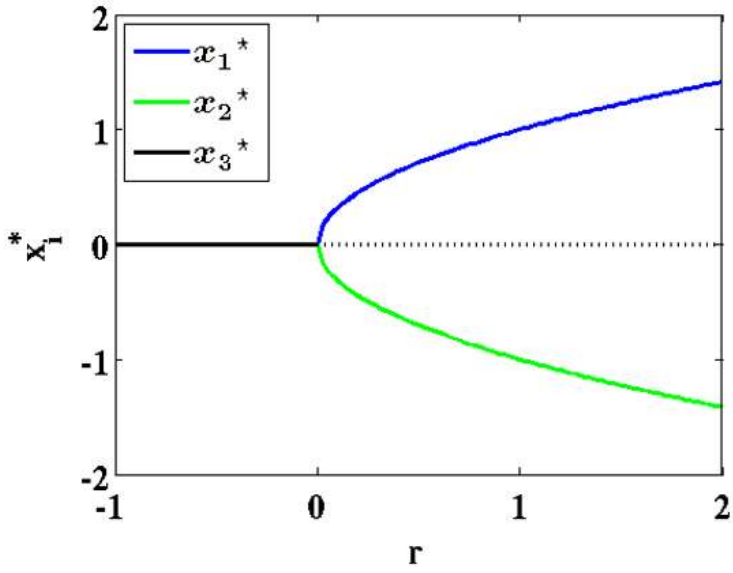


Abb. [6]

Beispiel für Gleichgewicht:

$X_1 = \sqrt{r}$ → f'($\sqrt{r}$) = -2r < 0 → stabil durch Asymptote

$X_2 = -\sqrt{r}$ → f'($-\sqrt{r}$) = -2r < 0 → stabiler Zustand

---

[5] http://imsc.uni-graz.at/keeling/modII_ss16/modnsc.pdf
[6] http://imsc.uni-graz.at/keeling/modII_ss16/modnsc.pdf

$X_3 = 0$ → $f'(0) = r$

- $r < 0$ → stabiler Zustand durch Asymptote
- $r = 0$ → stabiler Zustand durch Asymptote
- $r > 0$ → instabil durch Aufspaltung der Gleichgewichtszustände

8.1. Chemische Hysterese

Man kann für das Polynom dritten Grades auch folgendes annehmen:

$\alpha_0 = r$, $\alpha_1 = 1$, $\alpha_2 = 0$, $\alpha_3 = -1$

$x' = f(x)$

$f(x) = r + x(1-x^2)$

$f'(x) = 1 - 3x^2$

Nullstellen berechnen:

$f'(x) = 1 - 3x^2 > 0$

$$-\frac{1}{\sqrt{3}} < x < \frac{1}{\sqrt{3}}$$

Beim Gleichgewichtszustand ist $f'(t) = 0$.

Nach Umformungen erhält man

$r = x_i(r)[x_i(r)^2 - 1]$, $-\frac{1}{\sqrt{3}} < x_2(r) < \frac{1}{\sqrt{3}}$

A: $r < \frac{-2}{3\sqrt{3}}$ → x_1

B: $r = -\frac{2}{3\sqrt{3}}$ → $x_1 < x_2 = x_3$

C: $-\frac{2}{3\sqrt{3}} < r < \frac{2}{3\sqrt{3}}$ → $x_1 < x_2 < x_3$

D: $r = \frac{2}{3\sqrt{3}}$ → $x_1 = x_2 < x_3$

E: $r > \frac{2}{3\sqrt{3}}$ → x_3

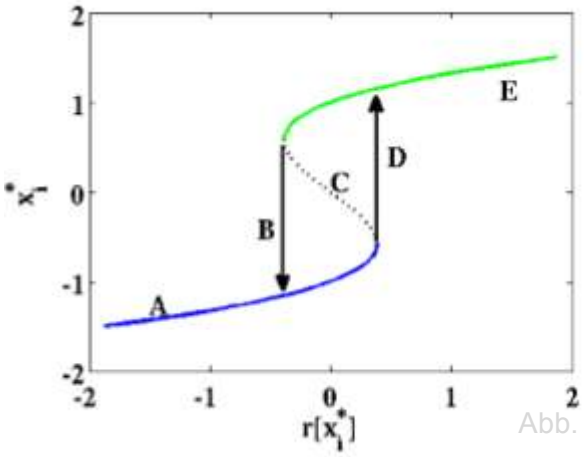


Abb. [7]

[7] http://imsc.uni-graz.at/keeling/modII_ss16/modnsc.pdf

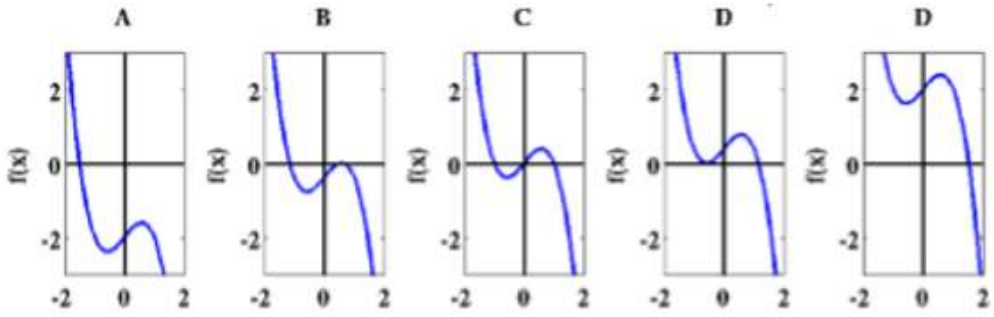


Abb. [8]

8.2. Hysterese

Hysterese beschreibt die Abhängigkeit eines bestimmten Status von seiner Vergangenheit. Ein Beispiel für eine Hysterese wäre die chemische Hysterese. Bei diesem Beispiel hängt der Gleichgewichtzustand einer Reaktion von der Temperatur im System ab. Die Ausgangsgröße bei diesem Systemverhalten hängt sowohl von ihrem vorherigen Zustand, als auch von ihrer Eingangsgröße ab. Es ist also ein System, das fähig ist, je nach Vorgeschichte, einen von mehreren möglichen Zuständen einzunehmen, die alle die gleiche Eingangsgröße haben.

- Beispiel:

Man betrachte ein chemisches Gleichgewicht in welchem zwei Reagenzien beteiligt sind, die entweder miteinander wechselwirken (blaue Striche) oder voneinander getrennt vorliegen.

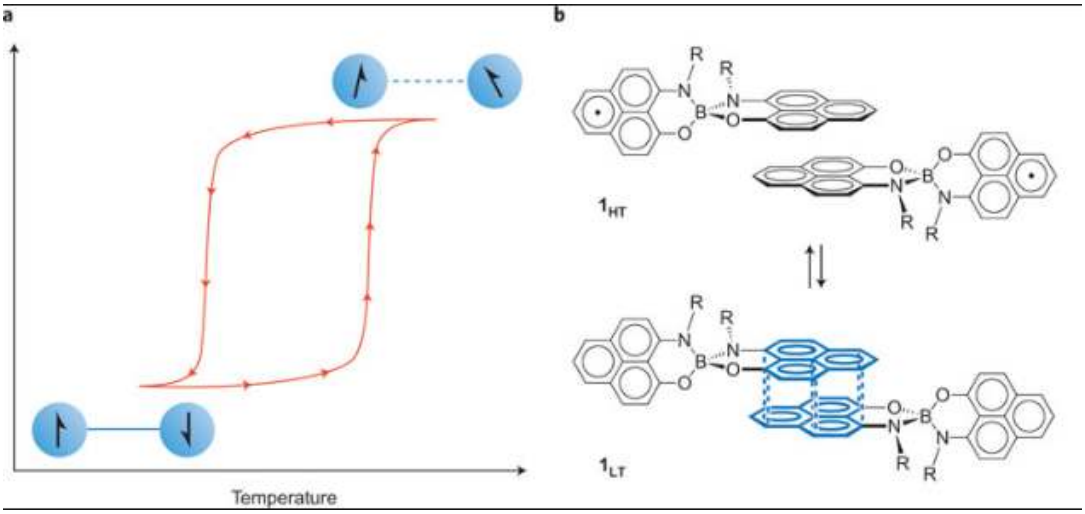


Abb [9]

Bei einer niedrigen Temperatur kann eine Wechselwirkung zwischen den beiden Reagenzien stattfinden, sobald aber die Temperatur zu hoch ist, wird die Wechselwirkung zerstört (Hohe Temperatur → Teilchen haben eine größere Schwingung → Wechselwirkung ist nicht mehr möglich).

Wenn aber die Temperatur nicht zu klein oder zu hoch ist, kann man nicht genau sagen, welcher Zustand im Gemisch vorliegt.

---

[8] http://imsc.uni-graz.at/keeling/modII_ss16/modnsc.pdf

[9] http://www.nature.com/nchem/journal/v3/n3/images/nchem.997-f1.jpg

Dabei ist die Vorgeschichte des Gemisches signifikant: Wenn das Gemisch am Anfang eine niedrige Temperatur hat, kann eine Wechselwirkung aufrechterhalten werden. Bei einem Temperaturanstieg wird die Wechselwirkung langsam zerstört. Aber da man eine bestimmte Aktivierungsenergie braucht, um die Wechselwirkung vollständig zu zerstören, kann man mit einer Sicherheit sagen, dass das Gemisch im mittleren Temperaturbereich eher die Wechselwirkung aufrechterhalten wird. Man spricht von Hysterese.

8.3. Hysterese in Matlab

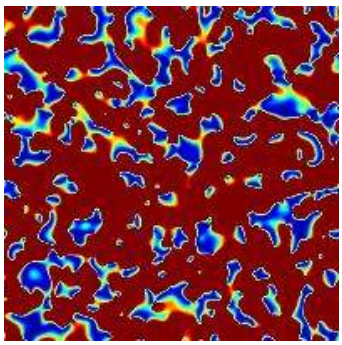
Das Hysterese-Verhalten kann man auch sehr gut in Matlab darstellen:

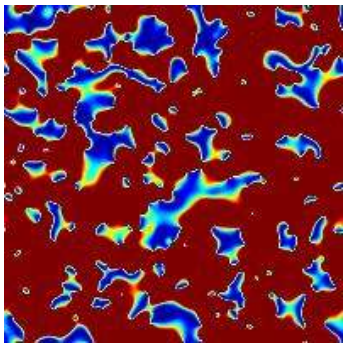
```
for k = 1:(Nt-1)
    r = ga*p.*(1-p.*p)+v;
    g =                -p;


    %p = (I + de*dt*L)\(p + dt*r); %implizit
    %v = (I + ga*dt*L)\(v + dt*g); %implizit
p = (I - de*dt*L)*p + dt*r; %explizit
v = (I - de*dt*L)*v + dt*g; %explizit
```

Dieser Code beschreibt das Reaktions- und Diffusionsverhalten einer Chemikalie p. Der Gleichgewichtszustand wird durch die Temperatur v gesteuert. Man erkennt diesen Verhalt an den Variablen r und g.

Zu r wird v hinzugezählt und bei g wird p abgezogen. Das heißt wenn die Temperatur v erhöht wird auch p erhöht. Durch die Erhöhung von p wird aber v wieder erniedrigt. Das heißt, dass auch p wieder niedrig wird und das resultiert wieder in einer Temperaturerhöhung.

So wechseln die Zustände immer hin und her und die Werte werden drastisch immer hin und her geschoben. Das Resultat ist ein Muster, das immer seine Farben ändert.





8.4. Gray Scott-Modell: Blutbahnen

Wie schon in den vorigen Kapiteln beschrieben, verwendet das Gray-Scott-Modell ein Fließgleichgewichtsystem, in dem 2 verschiedene Systeme bzw. Chemikalien, in welchem Diffusion und Reaktion miteinander konkurrieren. Bei bestimmten

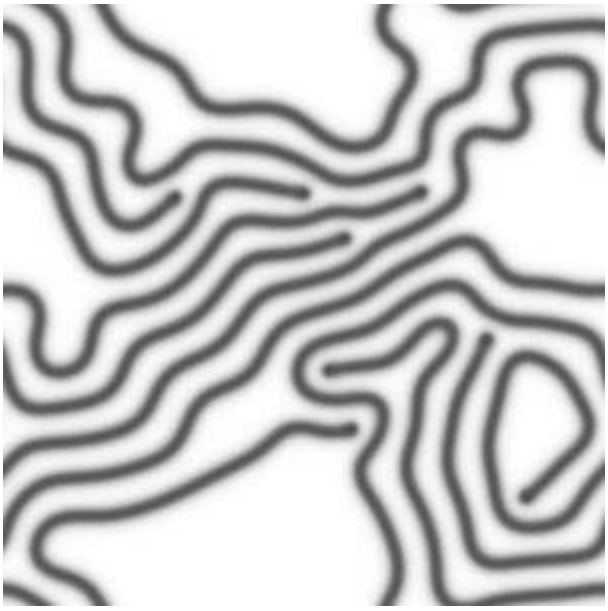
Parametern und Startbedingungen, kann aus einer Ansammlung von Punkten ein „Gefäß“ entstehen oder anders ausgedrückt, eine Blutbahn. Dieses „Gefäß“ wächst immer weiter und wird immer länger.

1)

2)

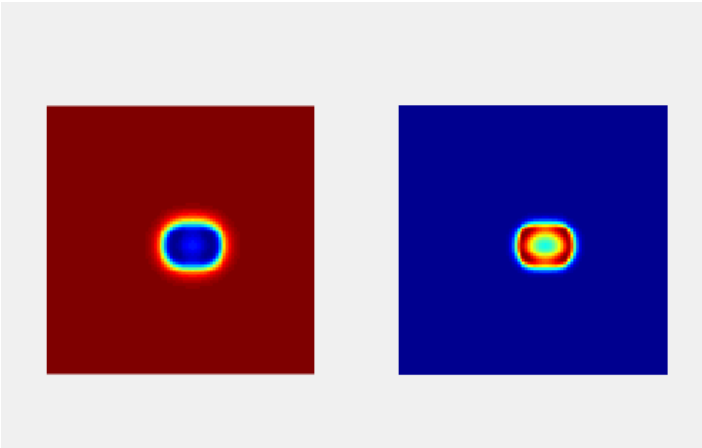
Die Parameter lauten: $D_u = 1$, $D_v = 0.5$, $F = 0.062$, $K = 0.065$

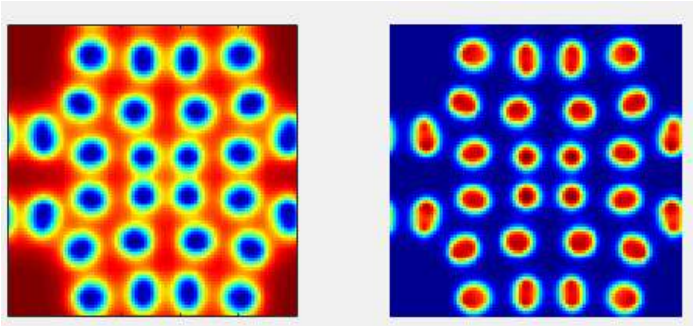
Durch die erhöhte Killing-Rate wird verhindert, dass sich die Blutbahnen verbinden. Sie weichen sich gegenseitig aus und wenn es keine Ausweichmöglichkeit gibt, werden die Teile der Blutbahnen weggedrückt.



8.5. Gray-Scott-Modell: Mitose

Man kann mit dem Gray-Scott-Modell auch sehr gut den Vorgang der Mitose nachstellen. Dabei verwendet man beim Gray-Scott-Modell eine sehr kleine Feeding-Rate, dafür aber eine große Killing-Rate, damit sich der Punkt gut teilen kann.





F = 0.026, k = 0.061, D_u = 1, D_v = 0.5

In Weiz steht ein Candle-Light-Dinner am Programm FOTOLIA

Angebote für Paare der Region

Segensfeiern für Liebende gibt es anlässlich des Valentinstags am **Samstag, 11. Februar,** in der Pfarrkirche St. Margarethen/Raab (18 Uhr) und **Samstag, 18. Februar,** in der Taborkirche Weiz (18 Uhr), anschließend Candle-Light-Dinner beim Gasthof Allmer in Weiz. Anmeldung dafür bei Elisabeth Hartinger unter Tel. 0664 21 55 678.

ST. MARGARETHEN/RAAB, WEIZ

Die Partnerschaft pflegen

Die Kirche wird rund um Valentinstag für Paare aktiv.

Elisabeth Hartinger weiß: „Eine Beziehung muss gepflegt werden, wie eine Pflanze." Darum hat sie gemeinsam mit dem Arbeitskreis „Ehe und Familie" der Pfarre Weiz auch heuer wieder anlässlich des Valentinstages ein Programm zusammengestellt, „das einen einfach wieder zum Nachdenken bringen soll", wie Hartinger sagt.

In Weiz ist das eine Segensfeier für Liebende am Samstag, 18. Februar, in der Taborkirche mit anschließender Agape inklusive Herzbrezeln und einem Candle-Light-Dinner im Gasthof Allmer (siehe Infokasten). Auch in den vergangenen Jahren sei die Taborkirche bei diesem besonderen Wortgottesdienst immer gut gefüllt gewesen, sagt Hartinger. „Und das Schöne war, dass sowohl ältere, als auch jüngere Paare dabei waren", sagt sie. Es werden Liebeslieder gespielt und zum Thema passende Texte gelesen, im Mittelpunkt stehe aber die Segensfeier, bei der die Partner einander segnen.

Schon heute um 18 Uhr wird ein solcher Gottesdienst für Liebende in der Pfarrkirche St. Margarethen/Raab gefeiert. Ein pfarrübergreifender Jugendchor sorgt für Musik.

Von 14. bis 19. Februar startet der katholische Familienverband übrigens auch eine Aktionswoche. Wer sich unter info-stmk@familie.at oder unter Tel. (0316) 80 41-398 anmeldet, erhält per Mail Anregungen für kleine Liebesbekundungen und weitere Impulse für eine gelungene Partnerschaft.

Sonja Berger

Wo Rechenkünstler nicht schief angeschaut werden

Junge Mathematik-Talente aus der ganzen Steiermark trafen sich eine Woche lang bei der 13. Modellierungswoche in Pöllau.

Von Franz Brugner

Die Feuermelder im Pöllauer Jugend- und Familiengästehaus blieben stumm. Und das trotz der rauchenden Köpfe bei insgesamt 30 Schülern, die an der 13. Modellierungswoche teilgenommen hatten. Auf die Oberstufenschüler im Alter von 16 bis 18 Jahren warteten nämlich sehr komplexe Aufgabenstellungen. Jede der fünf Gruppen bestand aus sechs Teilnehmern, die sich aus themenspezifischem Interesse zusammengefunden hatten.

Sebastian Engel aus München, der am mathematischen Institut der Universität Graz gerade seine Doktorarbeit schreibt, ist mit seiner Gruppe in die Materie der Versicherungsmathematik eingetaucht. Er schüttelt Begriffe wie Simulationsvektoren, Exponentialfunktionen oder Wurzeldefinition bei seinen mündlichen Erklärungen aus dem Ärmel und kritzelt dabei Gleichungen an die Pinnwand. Die Schüler hängen an seinen Lippen. Was sich in der Theorie äußerst abstrakt anhört, hat in der Realität einen großen Nutzen: Es geht um die Berechnung des nötigen Kapitals für Versicherungsgesellschaften zur Deckung von finanziellen Aufwendungen durch eingetretene Schäden. Da werden emsig Modelle für die Prognostizierung zukünftiger Schadensfälle berechnet.

Bei Cyberattacken und terroristischen Anschlägen stößt die von Engel offerierte Modellpalette aber an ihre Grenzen. „Diese Faktoren sind unberechenbar", muss er eingestehen. Das tut dem Eifer seiner Schüler bei ihrer rechnerischen Tüftelei aber keinen Abbruch.

Dem Kohlenstoffkreislauf ist die Gruppe rund um Florian Thaler auf der Spur. Man kämpft sich durch einen Wulst von Formeln, um klimarelevante Auswirkungen durch erhöhte Kohlendioxidkonzentration in der Atmosphäre möglichst exakt zu erforschen. Das Thema scheint angekommen zu sein, wie der hoch konzentrierte Arbeitseifer der Schüler verrät.

Jakob und Konrad, beide Schüler vom Grazer GIPS-Gymnasium, haben sich damit intensiv auseinander gesetzt, wie lange die Erde brauchen würde, den Treibhauseffekt wieder völlig abzubauen, würde die Spezies Mensch vom Planeten verschwinden. Jakob dazu: „In ein paar hundert Jahren hätte sich die Erde wieder völlig regeneriert."

Computergrafik steht am Lehrplan von Laurent Pfeiffer, Assistenzprofessor am Grazer Mathe-Institut. Die zwei BORG-Schülerinnen Miriam aus Feldbach und Sabrina aus Gleisdorf strahlen übers ganze Gesicht. Als völlige Neueinsteigerinnen in die Welt des Programmierens haben sie es schon nach zwei Tagen geschafft, eine plastische Kugel auf den Bildschirm ihres Laptops zu zaubern. Die ersten Ergebnisse sahen alles andere als einer Kugel ähnlich. „Das Schöne an der Modellierungswoche ist, hier Gleichgesinnte zu treffen mit denen man sich fachlich austauschen kann", meint Miriam. Sabrina pflichtet ihr bei: „In der Schule wird man, wenn man sich für Mathematik interessiert, ein wenig schief angeschaut."

Florian vom Gymnasium Knittelfeld, der sich unter der Anleitung von Richard Huber intensiv mit der künstlichen Intelligenz von Computern auseinandersetzt, sagt über seine Erfahrungen: „Das Schulwissen kann man hier vergessen. Wichtig ist, ein logisches Denkvermögen zu haben."

Stephen Keeling und Patrick-Michel Frühmann, die beiden Organisatoren der Modellierungswoche, können ein erfreuliches Resümee ziehen. „Ich glaube sagen zu können, dass alle Teilnehmer mit großen Lernfortschritten nach Hause fahren", betont Frühmann. Die Jugendlichen kehren als glühende Mathematik-Botschafter an ihre Schulen zurück.

Uni-Professor Laurent Pfeiffer und seine Schüler erweckten Computergrafiken zum Leben BRUGNER

KOMMENTAR

Von Susanne Rauschenbach
susanne.rauschenbach@kleinezeitung.at

Rauchende Köpfe

Zahlen, Formeln, Rechnungen, Funktionen oder Gleichungen – schon allein der Gedanke an die Mathematik lässt manchem die Haare zu Berge stehen und weckt da wie dort unschöne Erinnerungen an seine Schulzeit.

Das können die jungen Rechenkünstler, die sich in Pöllau eine Woche lang austauschten, nicht nachvollziehen. Für sie ist die Welt der Zahlen und Zeichen nicht nur eine verständliche, sondern auch faszinierende, für die sie gerne ihre Köpfe rauchen lassen.

In null Komma nichts haben sie mathematische (Schul)Probleme gelöst, im Rahmen der Modellierungswoche tüfteln sie an weit komplexeren Aufgaben.

Und auch wenn die jungen Mathematik-Fans mit ihren Ergebnissen wohl nicht immer auf einen gemeinsamen Nenner kommen, ist unterm Strich etwas Wichtiges entstanden: Kontakte zu Gleichgesinnten, mit denen man sich (fachlich) austauschen kann.

IN WEIZ FÜR SIE DA

Regionalredaktion Weiz, Birkfelder Straße 25, 8160 Weiz Tel. (03172) 65 20 weizred@kleinezeitung.at;

Redaktion: Sonja Berger (DW 18), Robert Breitler (DW 14), Raimund Heigl (DW 16), Ilka Patz (DW 11);

Sekretariat: Elfriede Brunnader, Doris Haider; Mo. – Do. 8 bis 16 Uhr, Fr. 8 bis 13 Uhr, weiz@kleinezeitung.at;

Werbeberater: Christoph Burböck (DW 12);

Abo-Service: (0316) 875 3200

MEISTGELESEN IM NETZ

Die Top 3 in der Kleine-App und auf kleinezeitung.at/weiz:

1 **Eine Kollision und ein Auto am Dach.** In Gleisdorf und in Birkfeld kam es am Donnerstagabend zu zwei Verkehrsunfällen - alle Beteiligten blieben unverletzt.

2 **Hohe Berge, kalte Duschen.** Der Alpenvereins Anger feierte 25-Jahr-Jubiläum und steigende Mitgliederzahlen.

3 **Schwestern und Mutter als Ersthelfer.** Beim Unfall auf der A2 bei Gleisdorf Süd half Familie dem Unfallauto.

AUS DEM ARCHIV

Vor zehn Jahren haben wir über die immer höher werdenden Kosten des Sozialwesens für die Gemeinden berichtet. 10,71 Millionen Euro oder 15,53 Prozent der Budgets mussten die damals 54 Gemeinden im Jahr 2007 an den Sozialhilfeverband abliefern. Ein weiteres Problem entstand dadurch, dass die Paragrafen des Sozialhilfegesetzes, das die Organisation der Sozialhilfe regelt, mit 1. Jänner außer Kraft gesetzt wurden. Die neuen Gesetze wurden zwar schon am 16. Jänner beschlossen, waren aber im Februar noch nicht gültig.

WAS HEUTE LOS IST

11. 2. Samstag

Tag der offenen Baustelle ist heute von von 10 bis 14 Uhr beim neuen Weizer Wirtschaftshof in der Energiestraße – neben der Elin-Motoren-Arena in Krottendorf – angesagt. Vertreter der Baufirmen und der Weizer Politik führen durch den modernen Bau und informieren über dessen Angebote.

R. HEIGL

STEIRER DES TAGES

Wenn Köpfe rauchen

Die Elite der steirischen Jungmathematiker weilt auf Trainingslager in Pöllau, betreut von Stephen Keeling und Patrick-Michel Frühmann.

Von Franz Brugner

Als der US-Amerikaner Stephen Keeling vor 19 Jahren als Professor an das Institut für Mathematik in Graz berufen wurde, war er verwundert, dass es außerhalb des schulischen Betriebssystems kein Zusatzangebot für Jugendliche mit ausgeprägter mathematischer Schlagseite gab. „Just do it", sagte sich der Wahlgrazer. Keeling rief eine Modellierungswoche für Mathematik ins Leben, zu der Schüler im Alter zwischen 16 und 18 Jahren für einen intensiven wie stressfreien Lernprozess ohne Prüfungen jeglicher Art eingeladen wurden.

Mittlerweile ist man bei der 13. Veranstaltung in Folge angekommen. Maximal 30 Schüler, die sich aus persönlichem Interesse gemeldet haben, werden in fünf Gruppen von Keeling & Co. zu herausfordernden mathematischen Gipfelbesteigungen mitgenommen.

Mit dem Gleisdorfer Mittelschullehrer Patrick-Michel Frühmann – der zweifache Doktor unterrichtet am BG/BRG Leibnitz Mathematik und hat zusätzlich einen Lehrauftrag an der Uni Graz – hat Keeling einen engagierten Mitstreiter gefunden. Frühmann selbst agiert als Promotor, Koordinator und Mädchen für alles vor Ort. Beide machen es faktisch ehrenamtlich. „Unser größter Lohn ist, zu sehen, welche Fortschritte die Schüler gemacht haben. Die Diskussionen und der Erfahrungsaustausch untereinander sind dabei ein immens wichtiger Lernprozess", konstatiert Frühmann. Am Ende der Woche sei jeder der hoch motivierten Rechenkünstler in der Welt der Logik ein paar Riesenschritte vorangekommen.

Stephen Keeling und Patrick-Michel Frühmann fördern mathematisches Potenzial zutage BRUGNER

Stolz erzählt Keeling von einem früheren Teilnehmer, der infolge zwei preisgekrönte Fachbereichsarbeiten geschrieben und sein Mathematikstudium abgeschlossen habe und jetzt an seinem Doktorat in Cambridge arbeite. „So etwas erfüllt einen mit einem gewissen Stolz."

Es sei aber ein Irrtum, zu glauben, so Frühmann, dass alle Teilnehmer beruflich einmal die Feder der Mathematik am Hut tragen. Deshalb sind auch die Aufgabenstellungen, die im Vorfeld streng geheim bleiben, sehr heterogen – von der präzisen Fixierung des Auges bei Operationen bis zur Optimierung des Verkehrsflusses.

Auf das schlechte Abschneiden der österreichischen Schüler bei den Naturwissenschaften beim letzten PISA-Test angesprochen, meint Frühmann: „In unserer Jugend steckt sehr viel Potenzial. Es ist die Aufgabe von uns Lehrern, ihnen die Ängste vor dem Versagen zu nehmen und sie für Mathematik zu begeistern." In Pöllau geschieht das.