Projekt1

RISE Calculations

Projektteam:
Sebastian Wertheman unterstützt durch Matthias Keller,
betreut durch Prof. Marcel Pfahrer

10. Juni 2019

Berner Fachhochschule
Departement für Technik und Informatik
Informatik

Contents

1 Zweck dieses Dokuments

Dieses Dokument soll im ersten Teil das Projektumfeld anhand von Rise und dessen Ziele, sowie schliesslich das Projektscope und Projektziele zeigen.

Im zweiten Teil sollen die Herausforderungen und wie diese angegangen wurden erläutert, sowie schliesslich die gewählten, relativ rechnungslastigen Lösungswege möglichst Schritt für Schritt verständlich erklärt werden.

Im letzten Teil steht das Fazit zu den gefundenen Lösungen, zu deren Ausbaumöglichkeiten sowie zu der geleisteten Arbeit, deren Schwierigkeiten, Herausforderungen und entnommenen Lerneffekte.

2 Beschreibung, Ausgangslage und Ziele von Rise

2.1 Beschreibung Rise

Ziel von RISE (Response-Inducing Sustainability Evaluation) ist es, die Nachhaltigkeit von Landwirtschaftlichen Betrieben weltweit zu analysieren und aufgrund dieser Analyse Empfehlungen an Betriebe abzugeben, wie sie in diversen Bereichen abschneiden, beziehungsweise, wie sie sich verbessern können. Das folgende Spinnennetzdiagram zeigt, wie ein Beispielbetrieb in den diversen Kategorien abschneidet, das heisst, nachhaltig ist.

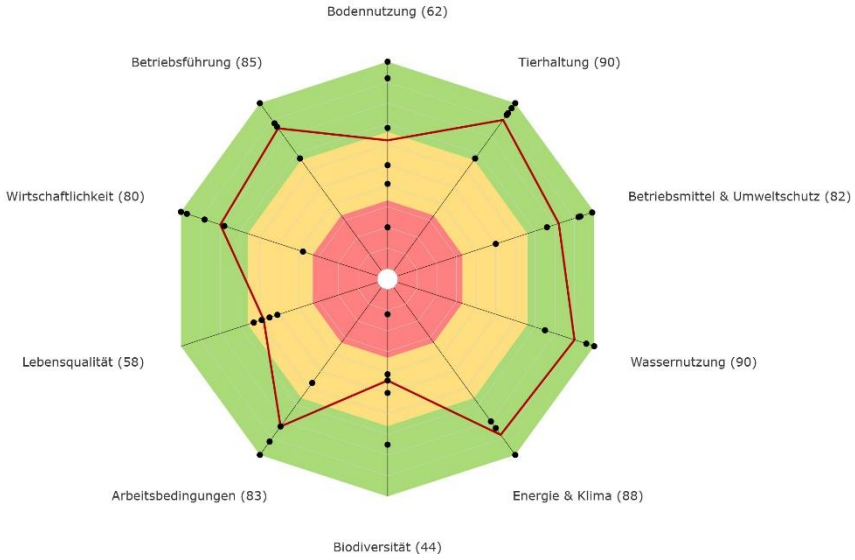


Abbildung 1: Spinnennetzdiagram einer Evaluation durch Rise

Die Auswertung geschieht aufgrund eines Fragebogens, welcher auf die relevanten Aspekte eines Betriebes zugeschneidert werden kann. Die Resultate der Befragung werden dann zusammen mit den bereits vorhandenen Referenzdaten aufgrund von festgelegten Algorithmen verrechnet und schrittweise über diverse Unterkategorien zu einem der im Spinnennetzdiagram ersichtlichen Hauptkategorien zusammengefasst. Im Spinnendiagram werden die Werte von null (rot) bis hundert (grün) auf die verschiedenen Farben und somit auf die Distanz zum Mittelpunkt gemappt. So ist für einen Betrieb, beziehungsweise die zuständigen Berater gut ersichtlich, in welchen Bereichen wie starkes Verbesserungspotential besteht. Anhand der Referenzdaten, welche zum Beispiel auch für regionale Faktoren differenziert vorhanden sind, können dann die Empfehlungen abgegeben werden.

2.2 Ausgangslage und Ziele von Rise

Ein Hauptziel von Rise ist es, die bereits funktionierende Webapplikation zur Analyse und Bewertung der Betriebe zu lokalisieren, das heisst, dass auch z.B. mit dem Laptop unterwegs in der Pampa, wenn eine günstige Internetverbindung nicht gegeben ist, eine Möglichkeit zur Durchführung der Betriebsanalyse von a bis z zur Verfügung stehen soll. Für die Offlineversion sollen Daten und Formeln zur Berechnung der Kennzahlen, wie auch zur Abgabe von Empfehlungen für den Betrieb lokal gespeichert und mit dem Server, sobald ein Internetzugang zur Verfügung steht, synchronisiert werden. Zur Bedienung soll ein Client zur Verfügung gestellt werden.

Ein weiteres Ziel ist es, die momentan sehr langsam implementierten Berechnungen zu beschleunigen. Mit den Zugriffen auf die Datenbank dauert eine Berechnung einer Kennzahl eines Auswertungsbereiches mehrere Sekunden, was klar optimierungspotential bietet.

3 Ziele und Scope des Projekts

3.1 Systemumfeld

Im Systemumfeld soll das Zusammenspiel mit anderen Systemen und/oder Lösungen aufgezeigt werden. Dazu wird die nachfolgende Grafik zur Visualisierung der Prozess- und Systemumfeld - Architektur, ohne technische Spezifikationen/Details genutzt.

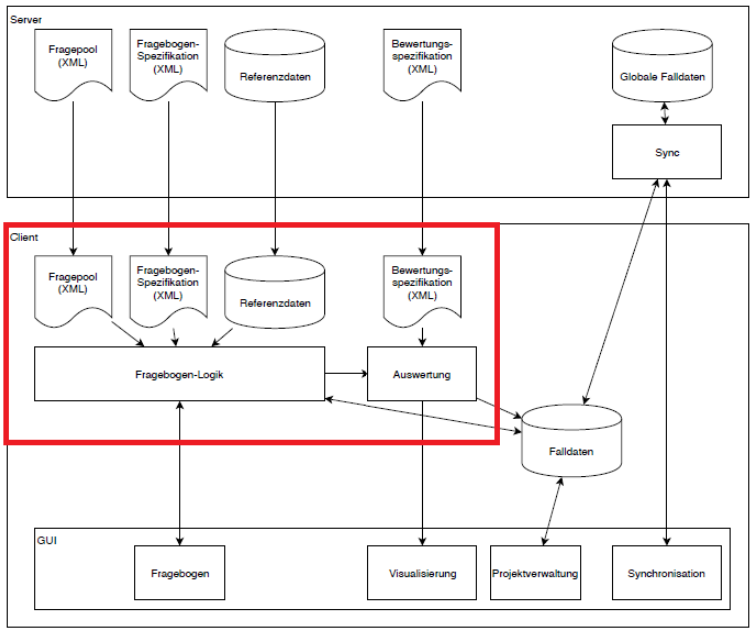


Abbildung 2: Project Scope in Rot. UML Diagramm von Prof. Marcel Pfahrer

Für das Projekt steht vorallem die Lokalisierung und Performanceoptimierung im Vordergrund. Der Fragebogen an sich ist bereits vorhanden, die Betriebsspezifische Auswahl und Anpassung der gestellten Fragen, beziehungsweise deren Menge ist für die Gesamtlogik und Performance relevant, aber nicht Teil des Projekts. Weiter ist die Optimierung der Database Queries für die Berechnungen, sowie für die Synchronisation des Clients wichtig, aber auch nicht Teil des Projekts. Letztlich ist auch das GUI für den lokalen Client kein Projektziel.

Im Projekt steht an erster Stelle die Berechnungsgeschwindigkeit. Dafür sollen die benötigten Falldaten aus dem Fragebogen, sowie den Referenzdaten aus der Datenbank als Input gegeben sein. Formeln zur Berechnung der Resultate sind ebenfalls gegeben, sollen aber analysiert und vor Allem in Bezug auf die Berechnungsperformance angepasst werden und für einen zukünftigen lokalen Klienten zur Verfügung stehen.

4 Development Environment

Abbildung 3: Project Environment

Der Hauptteil des Projekts wurde in Typescript unter der Nutzung des Angularframeworks erfasst.

Für die Organisation der Dependencies, für das Projektbuilding, wie auch zur Ausgabe des Programs, bzw. Anzeige im Browser, wurde der Packagemanager npm von Node verwendet.

Für die grafische Darstellung der verschiedenen Funktionskurven wurde D3js genutzt. Die drei D's stehen für Data Driven Documents.

Als IDE für das Projekt habe ich mich für Microsoft's Visual Studio Code entschieden, weiter war Excel zur schnellen, schrittweisen Berechnung der diversen Funktionen sehr hilfreich.

5 Herangehensweise an das Projekt

In diesem Teil des Dokuments soll gezeigt werden wie und begründet werden wieso welche Wege zur Lösung der Problemstellungen eingeschlagen wurden. Die Lösungswege werden hiermit eingeleitet, aber erst im nächsten Teil des Dokuments detailliert erläutert.

Als erstes stand eine Analyse der bereits vorhandenen Berechnungen im Vordergrund. Dazu stand ein relativ komplexes Excel-Sheet zur Verfügung, welches eine Auswertung eines Beispielbetriebes im Teilbereich Energie und Klima zeigt, sowie eine Dokumentation zur Berechnung der diversen Kennzahlen.

5.1 Lineare Funktionen

Aus der Analyse ging hervor, dass die meisten Berechnungen einen linearen oder zumindest teilweise linearen Charakter aufweisen.

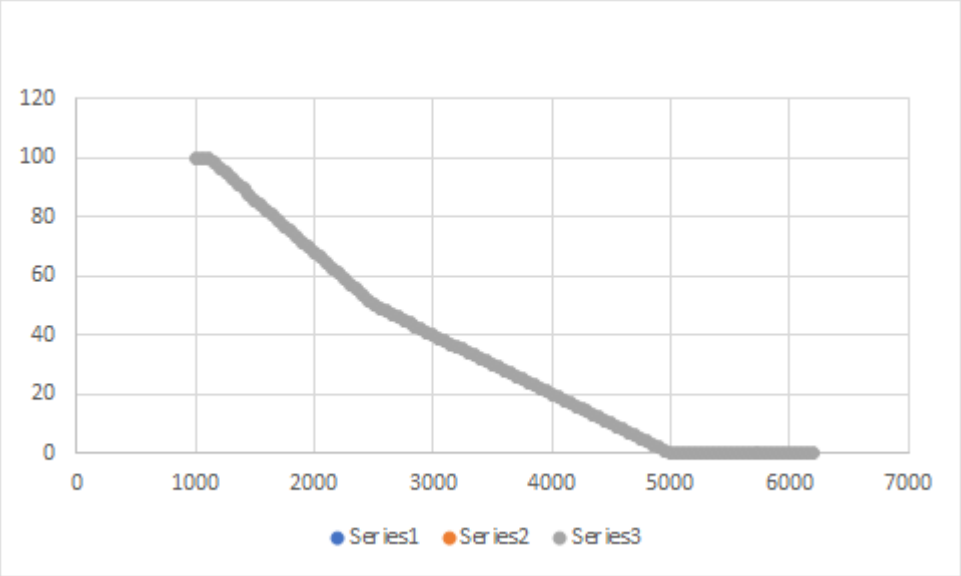


Abbildung 4: Excel Plot der Grünhausgas- Funktion von Prof. M. Pfahrer

Das Bild zeigt den Graphen einer der komplizierteren Berechnungen aus dem Excel-Sheet, welche durch viele, teilweise unnötige, if-Statements eine Bewertung zwischen 0 und 100 (Y-Achse) für den CO2 Ausstoss-Umfang eines Unternehmens (X-Achse) abgibt.

Als erster Lösungsansatz stand der Versuch, diese linearen und teilweise linearen Funktionen generalisierbar nachzubauen, sodass mit Angaben zur Berechnung, d.h. Verlauf der Funktion über die Intervalle, in welchen die Funktion linear verläuft, jede Funktion nachgebaut werden kann. Streng genommen kann durch eine gegen unendlich gehende Anzahl Intervalle jede, auch nichtlineare, Funktion approximiert werden.

5.2 Sigmoid Funktionen

Im weiteren Verlauf des Projekts stellte sich die Tendenz heraus, dass die meisten Funktionen, zumindest vom Output her relativ simpel sind und die Steigungen, beziehungsweise Resultatwertanpassungen relativ willkürlich aufgrund von Erfahrungswerten definiert wurden. Allgemein liess sich ein S-förmiger Verlauf bei den meisten Funktionen feststellen. Aufgrund dieser beiden Tatsachen ergab sich die Möglichkeit, die Funktionen durch einen Sigmoiden zu approximieren.

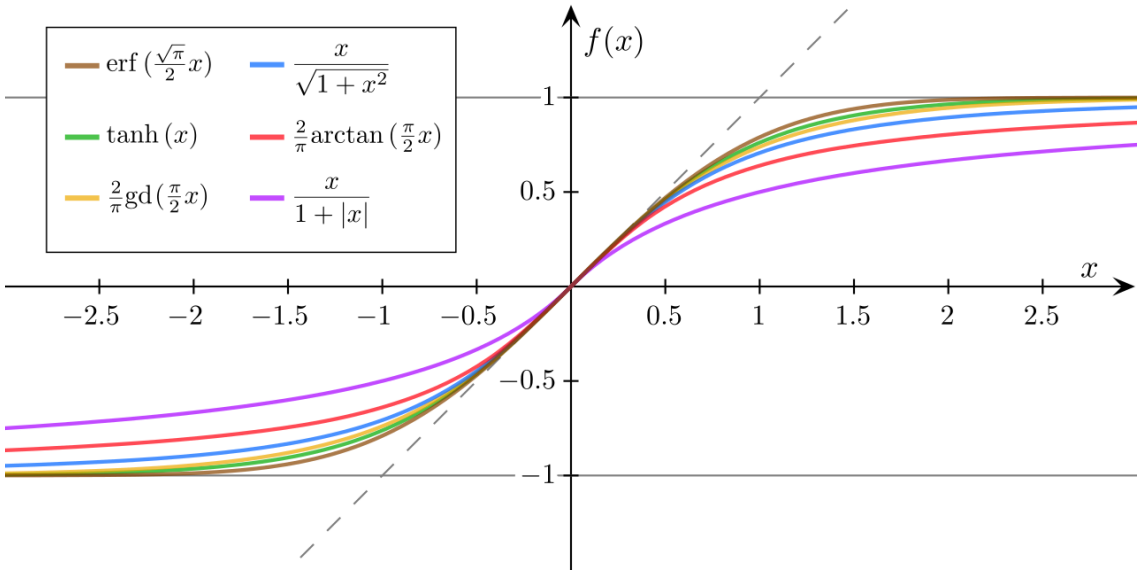


Abbildung 5 zeigt den graphischen Verlauf verschiedener Sigmoid-Funktionen.

Da der Verlauf je nach Funktion verschieden ausfällt, würde es allerdings nicht reichen, einen einzelnen, unskalierbaren Sigmoid für alle Berechnungen zur Verfügung zu stellen. Herausforderung war es nun, verschiedene Funktionen durch einen jeweils passenden Sigmoidverlauf nachzubilden.

Durch die Anpassung der Input- bzw. Output-Wertebereiche würde die Funktion in die Länge oder Breite gezogen, was wiederum entweder zu einem extrem steilen oder flachen Verlauf um den Mittelpunkt herum oder zu einer verlangsamten respektive beschleunigten Annäherung an die Grenzwerte führen würde. Gesucht war aber ein Weg, die Sigmoidkurve so zu skalieren, dass durch eine Parameterübergabe die Ausprägung der «Funktionsbäuche» angepasst werden kann.

5.3 Exponentielle Annäherung an den Sigmoidverlauf

Durch die skalierbare Sigmoidfunktion stand eigentlich bereits eine relativ breite Palette an Möglichkeiten zur Funktionsapproximation zur Verfügung, dennoch stellte sich unter anderem die Frage, wie effizient der, zur Berechnung der Sigmoide verwendete, Tangens Hyperbolicus ist, beziehungsweise, ob es eine simplere und effizientere Möglichkeit gibt, eine Sigmoidkurve zu simulieren. (Tangens Hyperbolicus – tanh(x): grüne Kurve in der Abbildung oben)
Weiter war es wünschenswert, ein paar verschiedene Approximationsformen auf ihre Eignung für die Rise-Berechnungen zu testen.

6 Lösungswege

6.1 Lineare Funktionen

Grundsätzlich ist eine allgemeine lineare Funktion folgendermassen definiert: y = f(x) = a * x + b. Wobei y der Output-Wert der Funktion, x der Input Wert, a die Steigung und b die Verschiebung auf der y-Achse ist. D.h., dass alle Output-Werte um den Wert von b verschoben werden.

Eine lineare Funktion mit konstantem a und b auf dem ganzen Funktionsverlauf ist folgendermassen als lineare Funktion mit einem einzigen Intervall zu verstehen.

6.2 Lineare Funktionen mit beliebiger Anzahl Intervalle

Die Implementation zur Berechnung der Linearen Funktionen wurde direkt in dieser allgemein anwendbaren Version durchgeführt.

Um eine beliebige Kurve durch lineare Schritte zu approximieren kann, durch die Wahl der Anzahl Schritte die Approximationsgenauigkeit festgelegt werden. Für jeden Schritt, d.h. für jedes Intervall wird laut der Definition der linearen Gleichung also ein Parameter a und b benötigt, sowie ein Weg, festzulegen von wo bis wo die Berechnung Anhand dieser Parameter durchgeführt werden soll, sprich wo dieses Intervall auf der x-Achse liegt.

In der Implementation werden die Input-Werte von 0 bis 100 ausgewertet, weiter gibt es in den Berechnungsformeln keine Lücken. Die Funktionen sind injektiv.
Es genügt also entweder den Start- oder Endpunkt des Intervalls zu definieren. Für die aufsteigende Auswertung macht es Sinn, jeweils die obere Grenze des Intervalls anzugeben und dann zu prüfen, ob der zu berechnende Wert kleiner als der Grenzwert des Intervalls ist und sich somit im Intervall befindet. Nun kann dieser Wert anhand der linearen Formel zusammen mit den Werten für a und b des Intervalls berechnet werden.

Die lineare Funktion ist nun in Angular-Typescript als «linear Service> implementiert und nimmt als Input ein «lower limit», «upper limit» und ein Model, welches aus drei Arrays besteht : einem Array für die oberen Grenzen der Intervalle und je einem für die jeweiligen a und b Werte.

Beim bauen des Models, sprich der Klasse, welche alle Intervalle für eine Funktion zur Verfügung stellt, muss darauf geachtet werden, dass für jedes Intervall alle drei Werte in geordneter Reihenfolge angegeben werden müssen.

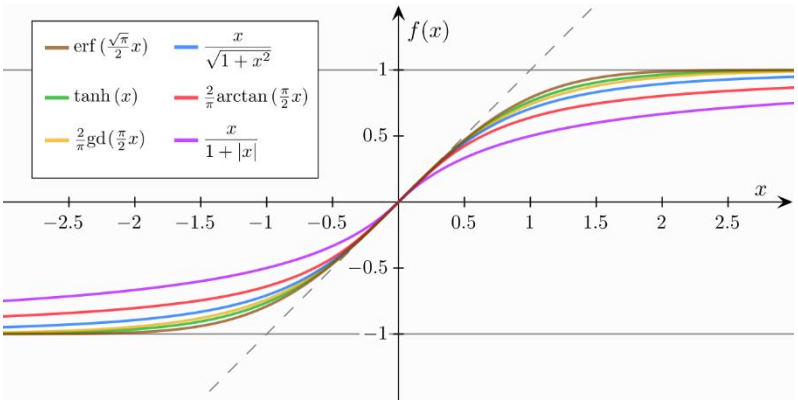
Um die benötigte Funktion für einen Input-Wert x zu verwenden, wird nun die Evaluate Funktion des Services mit dem Wert x und dem Model für die Funktion aufgerufen, wobei geprüft wird, ob der Input-Wert innerhalb des definierten Wertebereichs liegt, bzw. an den jeweiligen Grenzen davon. Ist eines davon der Fall, wird der jeweilige Grenzwert zurückgeliefert. Ist der Wert innerhalb des Wertebereichs wird für jedes Intervall aufsteigend geprüft, ob der x Wert kleiner als dessen Grenzwert ist. Trifft dies zu, wird der a und b Wert des Intervalls aus dem Model gelesen und mit damit die Berechnung der linearen Form a * x + b zurückgegeben, falls das Resultat dieser Berechnung nicht grösser als der maximale oder minimale Output-Wert ist. Sonst wird wiederum einer der beiden Grenzwerte retourniert.

6.3 Sigmoid Funktion

Wie bereits im Kapitel zu den Lösungsansätzen und Vorgehensweisen erwähnt, war beim Sigmoid die Hauptherausforderung, diesen skalierbar zu machen, ohne die Funktion einfach auszudehnen oder einen aussagenschwachen Kurvenverlauf zu erhalten.

Umgesetzt wurde dies, indem, Anstatt die Input-Werte auseinanderzuziehen, der Bereich der angenommenen Werte anpassbar gemacht wurde.
Indem man nur einen spezifischen Bereich, zentriert um den Nullpunkt als Input zulässt, erhält man einen bestimmten Ausschnitt der Sigmoidkurve, welcher je nach Breite vom Mittelpunkt mehr oder weniger stark an die Grenzwerte von -1 und 1angenähert ist – je breiter, desto näher am Grenzwert, welcher bei einem Input von jeweils unendlich oder minus unendlich erreicht werden würde.
Je nach Breite nähert sich der Wert also nur bis zu einem reduzierten Grenzwert an.

Dies ist fürs Verständnis am einfachsten mithilfe eines erneuten Betrachtens der Sigmoidkurven zu erklären.



Wählt man zum Beispiel einen Bereich zwischen -0.5 und 0.5 aus, erhält man von der Funktion fast nur den Ausschnitt der Geraden in der Mitte der Funktion und Output-Werte, welche zwischen ca. -0.6 und 0.6 liegen.
Je breiter man den Bereich also wählt, desto mehr von der Kurve, bzw. dem Teil der Kurve, wo sich dieser an die Grenzwerte annähert, erhält man. Die meisten Sigmoide sind ab einem Input-Wert von ungefähr -3, respektive 3 and die jeweiligen Grenzwerte von -1 und 1genügend angenähert, dass sich schlussfolgern lässt, dass für die Wahl des Bereiches nur bestimmte Werte Sinn machen. Nämlich Bereiche welche im kleinsten Fall von -0.5 bis 0.5 und im grössten fall von -3 bis 3 ragen.

Weiter sind für den sigmoidalen Verlauf Bereiche erwünscht, die um den Nullpunkt gespiegelt sind. Durchaus könnte man sich hier vorstellen, einen asymmetrischen Bereich auswählbar zu machen um verschiedenste Verläufe der S-Kurve als Output-Funktionsgraphen zu erhalten. Dies war aber nicht Ziel der Implementation, allerdings mit der fertigen Implementation des Sigmoids nicht allzu schwierig nachträglich umzusetzen.

Die Sigmoidfunktion ist nun in der «sigmoid service» Klasse implementiert und erwartet zwei Parameter: Den Input-Wert x, an welchem die Funktion ausgewertet werden soll und einen Skalierungswert s, welcher die Grösse des Inputbereiches festlegt. Bei einem Skalierungswert von 3 werden zum Beispiel Inputs zwischen -3 und 3 akzeptiert.

Die Input-Werte x müssen bereits auf einen Wertebereich von 0 bis 100 normiert sein, wofür ein «normaliser service» zur Verfügung steht.

Als erster Schritt werden die Input-Werte x nun auf den skalierbaren Bereich normiert, d.h. auf den Bereich -s bis s, sodass 0 zu -s und 100 zu s wird.
Um herauszufinden, um wieviel der Input Wert verkleinert werden muss, berechnet man am besten den Koeffizienten, welcher bestimmt, wie man von 100 auf einen Bereich kommt, der von der Breite her zwei Mal dem Skalierungsfaktor s entspricht: 100/(2*s). Dividiere ich die Input-Werte nun durch diesen Koeffizienten (x/(100/(2*s)), werden alle Input-Werte auf den Bereich 0 bis 2s gemappt. Nun muss ich also noch den Skalierungswert davon subtrahieren um den für die Sigmoidfunktion notwendigen Bereich von -s bis s zu erhalten.

Zusammengefasst wird, bevor der Input an die Sigmoidfunktion übergeben wird Folgende Formatierung vorgenommen: x = x/(100/(2*s))-s

Auf diesem Wert wird nun die Hyperbolische Tangens Funktion tanh(x) aufgerufen.

Wie bereits erwähnt, liegen die Output-Werte dieser Funktion tanh(x) je nach Input-Wertebereich variabel zwischen z.B. -0.7 und 0.7. Daher müssen die Werte noch auf eine Breite von -1 bis 1 gemappt werden, bevor Sie dann auf den erwünschten Bereich von 0 bis 100 gelegt werden.

Dafür wird der Output-Wert der tanh(x) Funktion durch Berechnung des maximalen Wertes skaliert. In anderen Worten, rufe ich die tanh(x) Funktion mit dem Skalierungswert s auf (tanh(s)), erhalte ich den maximalen Output-Wert, anhand dessen ich berechnen kann, um wieviel die x Werte am Schluss skaliert werden müssen um auf den Bereich -1 bis 1 zu liegen. Ich dividiere also die x werte noch durch den tanh von s. Anschliessend retourniere ich die Werte normiert und gerundet auf 0 bis 100 indem ich diese +1 und dann mal 50 rechne.

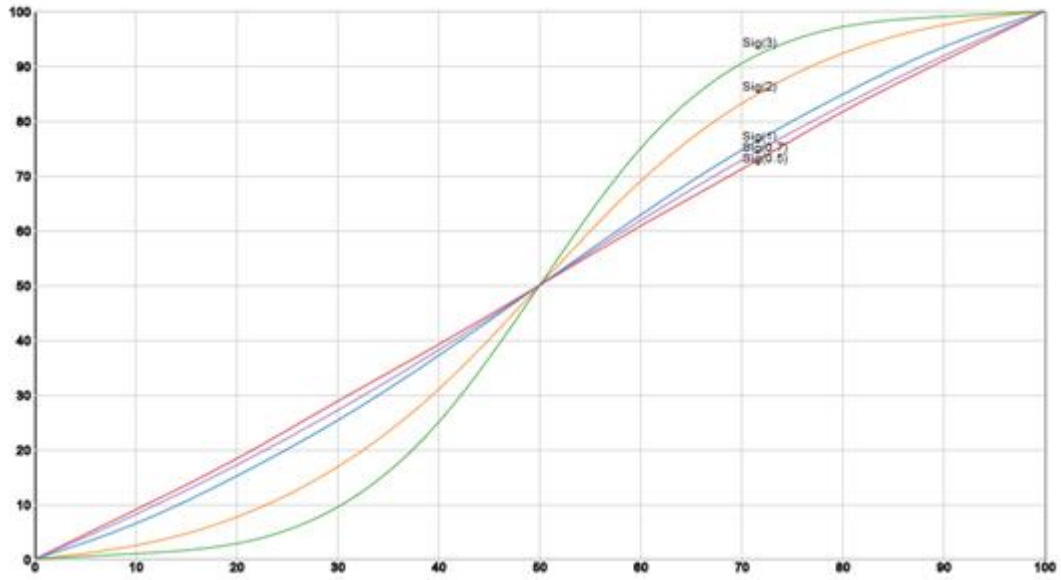


Abbildung 6 ist ein Screenshot von den Berechnungen, dargestellt mit d3js und zeigt fünf verschiedene Sigmoidkurven mit den jeweilig übergebenen Skalierungswerten als Zahl in Klammer.

6.4 Exponentieller Sigmoid

Im Grundgedanken dieses Lösungsansatzes liegt der kurvenförmige Verlauf der exponentiellen Funktionen. Als erster Ansatz, welcher im laufenden auch als Beispiel zur Erläuterung verwendet werden soll, war die Reihe von quadrierten Basiszahlen von null bis sieben. Grund dafür liegt in der einfachen Überlegung, dass sieben quadriert 49 ergibt, also durch Addition, respektive Subtraktion zu 50 den erwünschten Bewertungswertelimits von 0 und 100 sehr nahe liegt.

Der exponentielle Sigmoid ist nun variabel durch die Angabe der Basis, bis zu welcher die Werte exponiert werden sollen. Eine alternative Lösung wäre hier z.B. gewesen, eine fixe Basis durch einen wachsenden Exponenten zu exponieren. Dies wäre sicher auch ein spannender, jedoch eventuell etwas zu schnell wachsender Verlauf, was wiederum mit den richtigen Parametern kontrollierbar wäre.

Im ersten Versuch wurden also die auf 0 bis 100 normierten Werte, durch denselben Prozess wie bei der Sigmoidfunktion auf eine um den Nullpunkt gespiegelte Breite gespiegelt. Im Beispiel mit sieben also von -7 bis 7.
Dabei ist aufgefallen, dass Werte welche auf den Bereich -2 bis 2 gemappt wurden, relativ uninteressant sind, weshalb dieser mittlere Bereich ausgeschnitten wurde und somit im Beispiel mit 7 die Werte von 0 bis 100 neu auf den Bereich von -7 bis -2 und von 2 bis 7 gelegt wurden.

Die Berechnungen des exponentiellen Sigmoids wurden in der «exponential service» Klasse implementiert, wobei die Evaluierungsfunktion zur Berechnung folgende zwei Werte verlangt:
Den Input-Wert x für welchen die Funktion evaluiert werden soll, sowie die Basis base, welche wie oben beschrieben den Wertebereich festlegt. Base kann einen beliebigen Wert grösser ungleich zwei annehmen.

Die Evaluierungsfunktion läuft folgendermassen:

Der x-Wert wird darauf überprüft, ob er im erwünschten Wertebereich zwischen 0 und 100 liegt. Ist dem nicht der Fall, wird entsprechend 0 oder 100 retourniert. Falls x 50 ist, wird 50 retourniert.
Ist die Basis nicht grösser als zwei, wird wiederum 0 zurückgegeben.

Die Zuweisung der Inputwerte zwischen 0 und 100 auf den Wertebereich zwischen der Nullpunktgespiegelten Basis ohne den Bereich von -2 bis 2 erfolgt folgendermassen:
x/(100/((base-2)*2))-(base-2))
Dies liefert am Beispiel mit 7 einen Wertebereich von -5 bis 5. Da für die Exponentialrechnung die negativen Werte die gleichen Ergebnisse liefern wie Ihre jeweiligen positiven Gegenüber, kann für die Verschiebung, beziehungsweise Lückenbildung zwischen -2 und 2 von der obigen Berechnung der absolute Wert genommen und um zwei nach oben verschoben werden (+2).

Der Resultierende Wert kann nun quadriert werden, muss aber vor der Rückgabe noch anhand des Quadrats des maximalen Inputs, welches dem Quadrat der übergebenen Basis entspricht, normiert werden, sodass die Output-Werte zwischen 0 und 50 liegen. Dieses Resultat wird nun je nachdem ob der Input-Wert für die Evaluierungsfunktion kleiner oder grösser als 50 war zu 50 addiert oder subtrahiert zurückgegeben.

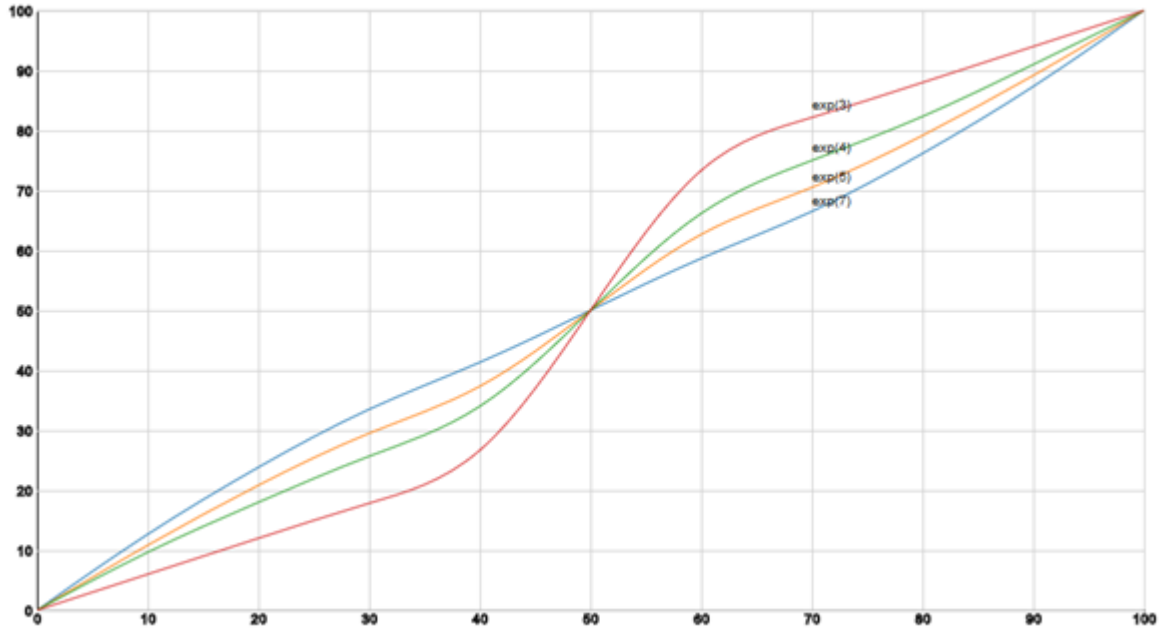


Abbildung 7 : Screenshot der d3js-Darstellung von vier exponentiellen Sigmoidfunktionen evaluiert für Zahlen zwischen 0 und 100 und jeweilig angegebener Nummer in Klammer als Parameter für die maximale Basis.

7 Auswertung der Lösung

Das Hauptziel, die Berechnungsgeschwindigkeit zu erhöhen wurde mit allen Lösungsansätzen erreicht. Was vorhin teilweise über zehn Sekunden gedauert hat, ist nun zum Beispiel auch noch bei tausend Berechnungen zur selben Zeit ausführbar, ohne als Mensch dabei eine Verzögerung zu merken.
Die Visualisierung in d3js im Browser via Start-Befehl des Node Package Managers wird im Test für mehrere Funktionskurven gleichzeitig, jeweils für ganzzahlige Werte zwischen 0 und 100 aufgerufen und kann zur Laufzeit angepasst werden, z.B. durch ein ein- oder ausblenden einer Funktion.

Die Arbeit zeigt recht gut, dass die momentanen Berechnungen von Rise neben der starken Beschleunigung auch relativ einfach abstrahiert, beziehungsweise vereinfacht werden können.
Durch den Sigmoid und den exponentiellen Sigmoid, respektive deren Anpassung durch die verschiedenen Skalierungmöglichkeiten, besteht bereits eine gewisse Palette an Funktionen, welche für die Approximation der bisherigen Berechnungen verwendet werden können.
Es ist durchaus vorstellbar, dass in Zukunft, zum Beispiel zur Berechnung neuer Kennzahlen, eine neue Art von Funktionskurven erwünscht ist, welche durch die beiden Sigmoide nicht optimal annäherbar ist.
Da es aber, wie bereits bei der Beschreibung der Sigmoide erwähnt, Möglichkeiten gibt, diese weiter zu variieren – wie zum Beispiel durch eine Verschiebung um den Mittelpunkt, oder durch eine Exponentialrechnung mit variablem Exponenten und es weiter auch denkbar ist, ganz andere Funktionen umzusetzen, besteht durchaus die Möglichkeit sehr viele Funktionen zu verallgemeinern.

Neben dem Annäherungsversuch durch die SIgmoidkurven besteht allerdings auch die Möglichkeit, eine beliebige Funktion genau abzubilden, beziehungsweise auf einen bestimmten Fall den Anforderungen entsprechend haargenau zu designen, indem diese durch den implementierten «linear service» konstruiert wird.
Von der Performance her ist auch dieser Weg problemlos wählbar, es werden aber je nach Wunsch zur detailliertheit der Funktion relativ viele Angaben für die Intervalle benötigt, was somit durchaus für den Versuch spricht, die ganze Funktionspallette angenähert zu designen.

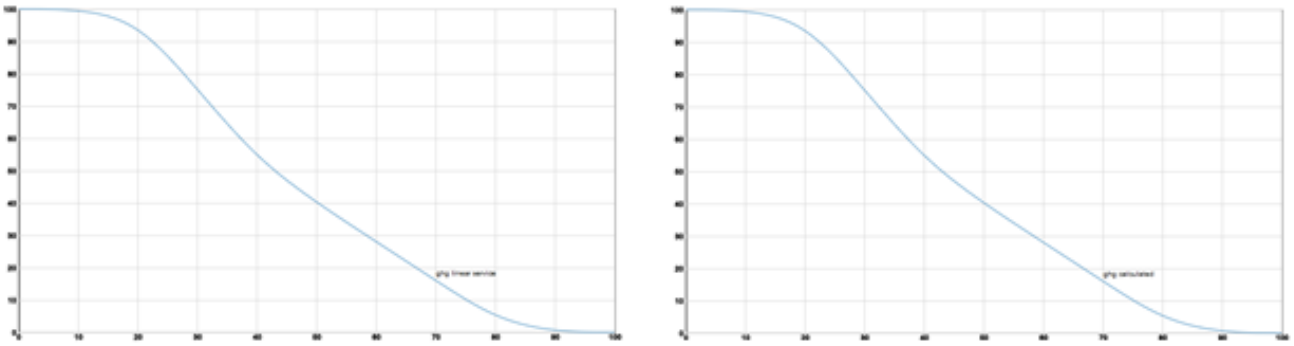


Abbildung 8 zeigt, dass die Originalfunktion zur Berechnung der Greenhouse Gas Bilanz (links) identisch durch den linearen Service (rechts) nachgeformt werden konnte.

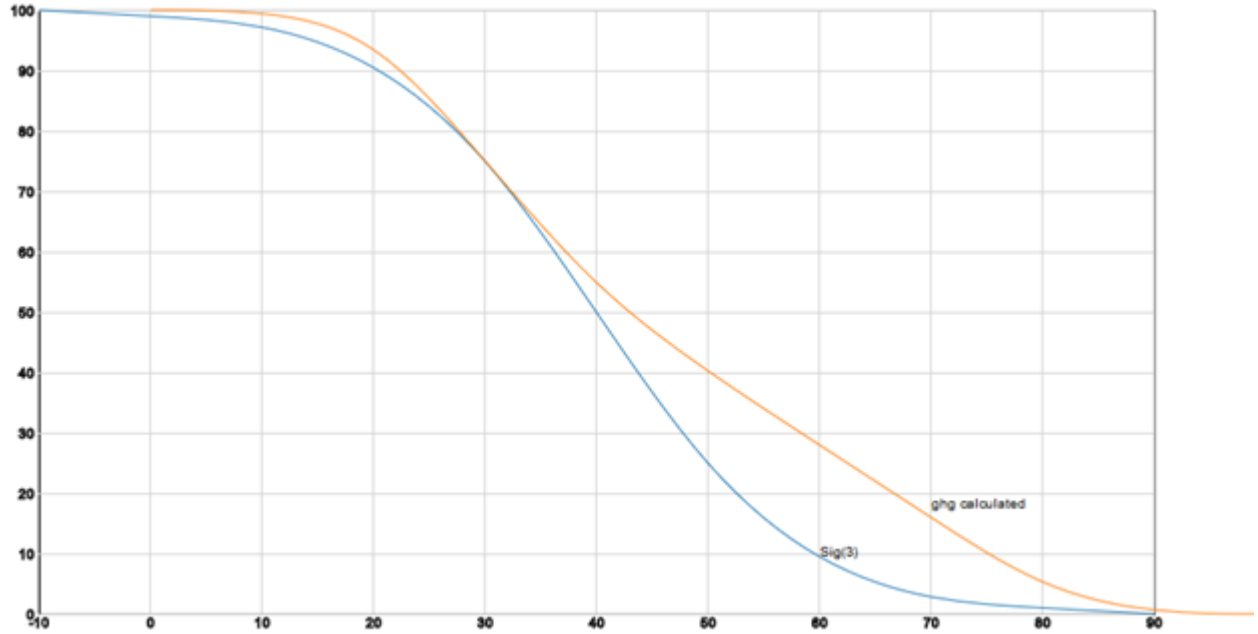


Abbildung 9 zeigt den Versuch, die Grünhausgas – Funktion (gelb) durch einen Sigmoiden (blau) zu approximieren.

Wie man sieht ist die «out oft he box»-gespiegelte Sigmoidfunktion mit dem Skalierungsfaktor drei der Originalfunktion auf einigen Bereichen bereits sehr nahe. Dies könnte durch weitere Verschiebung und Anpassung der Skalierung sicher noch präzisiert werden.
Da der Funktionsverlauf der Grünhausfunktion relativ beliebig gewählt ist, ist eine genaue Abbildung aber kaum möglich, wobei sich aufgrund der relativ willkürlich gewählten Konstruktion der Formel zur CO2-Ausstossbewertung, aber durchaus die Frage stellt, ob der Sigmoid dem eigentlichen Ziel der Wertevergabe nicht sogar näher kommt. Nämlich zeigt dieser einen schönen Verlauf über die ganze Breite der Evaluation. Die Originalfunktion versucht eigentlich durch eine relativ geringe Anzahl linearer Intervalle unterschiedlicher Steigung eine Kurvenform zu approximieren.

Die Entscheidung, wie präzis, beziehungsweise wie spezifisch eine Funktion für eine bestimmte Berechnung auszusehen hat, ist aber sicherlich von Fall zu Fall verschieden, sodass sich sicher die Frage stellt, ob z.B. der Sigmoid in den verschiedenen Skalierungen für alle Fälle, oder nur bestimmte Fälle verwendet werden kann.

8 Fazit

8.1 Erreichen der Zielsetzung

Der Fokus im Projekt ist schlussendlich vor allem auf die Berechnungsoptimierung und die verschiedenen Funktionen zur Vereinfachung dieser Berechnungen gefallen.
Die Berechnungen laufen zwar lokal, haben aber noch keine Schnittstelle zur Datenbank, welche an sich auch out of Scope für das Projekt waren, aber für eine genauere Analyse zu den Berechnungsgeschwindigkeiten Anhand eines Beispiels aus der Praxis sicher spannend gewesen wäre.

Wie bereits bei der Zielsetzung erwähnt, war auch die Realisation eines Clients nicht Teil der Projektarbeit, somit sind die Darstellungen in D3js mehr als Visualisierung der Berechnungen zu betrachten, nicht als endgültige Lösung. Dennoch besteht die Möglichkeit für einen zukünftigen Klienten durch D3js – Integration die bildliche Evaluation, eventuell bis und mit der finalen Anzeige eines navigierbaren Spinnennetzdiagrams, die Funktionalität der Data Driven Documents zu nutzen.

Wie bereits bei der Auswertung der Lösung beschrieben, war die Berechnungsoptimierung und Vereinfachung allerdings sehr erfolgreich.

8.2 Schwierigkeiten / Herausforderungen

Neben der recht zeitintensiven und teilweise herausfordernden, aber dadurch auch spannenden Arbeit zur Erstellung der Formeln für die Berechnung der schlussendlich drei verschiedenen Berechnungsmöglichkeiten (Linear, Sigmoid, Exponentieller Sigmoid), welche in einem ersten Schritt mit Hilfe von Excel umgesetzt wurden, musste ziemlich viel Zeit in die Einarbeitung in das Development Environment gesteckt werden.
Da zu Projektbeginn noch keine Erfahrung bezüglich Typescript vorhanden war, galt es zuerst, sich in dieser neuen Programmiersprache vertraut zu machen, was durch ein Videotutorial von Maximilian Schwarzmüller geschah.
Dieses Tutorial ist durchaus hilfreich und gut organisiert, aber es stellte sich später in der Projektarbeit heraus, dass die eigentlichen Herausforderungen weniger Typescript als Angular – Kenntnisse bedingten, welche schlicht fehlten und spätestens bei der Integration der D3js – Library ins Angular-Framework bemerkbar wurden.

Da ich dem Projekt vom Modul her alleine zugeteilt war, hatte ich oft die Vorstellung, dass es an mir liegt, die Herausforderungen, so gut möglich, alleine zu bewältigen und habe deswegen zu lange versucht die nötigen Informationen zu finden.
Für die Integration von D3js konnte ich dann schlussendlich nicht mehr auf die Hilfe meines modulextern zugeteilten Projektpartners, Matthias Keller, verzichten. Zusammen haben wir dann ein funktionierendes D3js – Beispiel, welches in Angular Typescript umgesetzt war als neue Basis verwendet.

8.3 Lerneffekt für die Zukunft

Vor Allem aus den Herausforderungen gegen Ende der Projektarbeit und der Zusammenarbeit mit meinem Projektpartner lege ich mir in Zukunft ans Herz, wenn ich bei etwas nicht weiterkomme, früher auf meinen Projektpartner oder Betreuungsperson zuzugehen. Im Vergleich zum Versuch das benötigte Wissen aus dem Internet zu filtern, war der Lerneffekt und somit auch Projektfortschritt durch die Unterstützung von Matthias Keller um Welten grösser.

9 Abbildungsverzeichnis

Titelbild:
https://fasw.de/nachhaltig

Abbildung 1:
Spinnennetz Diagramm Rise:
https://www.bfh.ch/hafl/en/research/reference-projects/rise/

Abbildung 2:
UML Diagram Project Environment:
Prof. M. Pfahrer

Abbildung 3:
Development environment:
D3js: https://d3js.org/
Visual Studio :https://visualstudio.microsoft.com/
Angular: https://angular.io/
Node: https://nodejs.org/en/
Typescript: https://www.typescriptlang.org/

Abbildung 4:
Excel Plot der Linearen Funktion:
Prof. M. Pfahrer

Abbildung 5:
Überblick der Sigmoidfunktionen:
https://en.wikipedia.org/wiki/Sigmoid_function

Abbildung 6-9:
Screenshots der Projektarbeit