Das Sofaproblem

Konstruktion eines Sofas durch Drehung des Raums

Eine Betrachtung des Sofaproblems durch Wechsel der Perspektive, daraus folgende Bedingungen und computergestützte Konstruktion

Autor: Patrick F. H. Sonnentag

Betreuer: Sandor Spiess

Mathematik / Informatik

Schülerforschungszentrum Friedrichshafen

25. Februar 2022

KARL MAYBACH GYMNASIUM

Kurzzusammenfassung

Das Sofaproblem ist die Frage nach der Fläche mit dem größten Flächeninhalt, welche durch einen Korridor mit Breite 1 und einer 90° -Ecke bewegt werden kann. Wir gehen das Problem durch Konstruktion dieser Fläche, dem Sofa, an, indem ein Wechsel des Referenzrahmen eingeführt wird. Diese Betrachtung wird näherungsweise in Java implementiert und liefert für bestimmte, ausgewählte Parameter ein Ergebnis von ungefähr 2,2079 für die Fläche des größten Sofas. Ebenfalls können Aussagen über die Verteilung der Werte in Abhängigkeit der Parameter gemacht werden, um zukünftige Untersuchungen zu erleichtern. Dabei finden sich Hinweise auf die Idealität von Gervers Sofa, dem bis jetzt größten gefundenen Sofa.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Das sogenannte *Sofaproblem* wurde zuerst formell 1966 von *Leo Moser* beschrieben. Es stellt die Frage welche zweidimensionale Form, die eine 90° Ecke in einem Korridor der Breite 1 passieren kann, den größten Flächeninhalt A hat. Es existiert eine untere Schranke für den größten Flächeninhalt mit $A \approx 2,2195$ durch *Joseph Gerver*[2] und eine obere Schranke mit $A = 2,37$ von *Yoav Kallus* und *Dan Romik*[4]. Es wird vermutet, dass Gervers Konstruktion tatsächlich optimal ist, aber ein formeller Beweis steht noch aus. Ebenso gibt es diverse Abänderungen des Problems, die bekannteste entstammt Romik, welcher versuchte eine Sofa um eine Links- und Rechtsecke zu bewegen, anstelle nur einer Ecke. In diesem Fall ist die größte untere Schranke $A \approx 1,645$, wobei auch hier kein Beweis der Idealität existiert.[4]

Wir wollen dieses Problem allgemein durch Berechnungen am Computer betrachten. Dafür definieren wir eine Konstruktionsmethode, um Sofas anhand gegebener Kurven zu konstruieren: Anstelle des Sofas bewegen wir den gesamten Korridor entlang einer Kurve. Alle Punkte, die während der gesamten Bewegung des Korridors diesen nicht verlassen haben, müssen Element des Sofas sein.
Anhand einiger Beobachtungen und Vermutungen über das Sofa und damit die Kurve untersuchen wir die Konstruktion anhand von Ellipsen, Hyperbeln, Polynomfunktionen und dem Sekans. Um möglichst viele und komplexere Kurven zu untersuchen, erarbeiten wir davor einen Suchalgorithmus, der zur Findung eines möglichst großen Sofas in Java implementiert wird.

2 Beschreibung der Konstruktionsmethode

Wie bereits in der Einleitung erwähnt, wechseln wir für die Konstruktion des Sofas unsere Sichtweise. Wir fixieren das Sofa so im $\mathbb{R}^2$, dass es genau zwischen $y = 0$ und $y = 1$ liegt. Zudem soll die y-Achse das Sofa in zwei Teile mit identischem Flächeninhalt teilen. Den Korridor bewegen wir damit um das Sofa entlang einer Kurve f.
Eine zusammenfassende Visualisierung unserer Konstruktionsmethode ist als Animation unter `https://www.geogebra.org/m/mu3bnvxe` zu finden.

2.1 Translation und Drehung des Korridors

Im Korridor bezeichnen wir die Begrenzungen als *Innen-* und *Außenwände*. Innenwände sind die Begrenzungen, welche stets näher am Ursprung sind. Damit sind dann die Außenwände die, die weiter vom Ursprung entfernt sind. Zudem sei $M\,(M_x|M_y)$ der Schnittpunkt der beiden Innenwände und dementsprechend $M'\left(M'_x|M'_y\right)$ der Schnittpunkt der Außenwände. Den Korridor bewegen wir entlang von, beziehungsweise M auf einer, Kurve f. Da alle bisherigen Sofas in der Größenordnung von Gervers Sofa eine gleichzeitige Rotation und Translation erfahren haben, bewegen und rotieren auch wir unseren Korridor auch gleichzeitig. Zu diesen Sofas gehört neben Gervers Sofa noch Hammersleys Sofa mit $A = \frac{\pi}{2} + \frac{2}{\pi} \approx 2,2074$.[3] Während der Bewegung entlang von f dreht sich der Korridor um den Punkt M, um 90° beziehungsweise $\frac{\pi}{2}$. Dies ist aber nichts selbstverständlich, da der Winkel kleiner sein könnte oder sich das Sofa, wie ein 1×1 Quadrat, überhaupt nicht drehen müsste.

Mit

$$\theta_M := \arctan\left(\frac{M_y}{M_x}\right)$$

definieren wir die Drehung des Korridors in Abhängigkeit der Position auf f, sodass f zwei Nullstellen haben muss, da θ_M sonst nicht jeden Wert aus $[0;\pi]$ annehmen wird. Die Innen- und Außenwände stehen per Definition jeweils im Winkel von $\frac{\pi}{2}$ zueinander. Somit können wir die Innenwände folgendermaßen als Funktion beschreiben:

$$W_I(x) = \begin{cases} \tan\left(\frac{\theta_M}{2}\right) \cdot (x - M_x) + M_y & \text{für } x \leq M_x \\ \tan\left(\frac{\theta_M + \pi}{2}\right) \cdot (x - M_x) + M_y & \text{für } x > M_x. \end{cases} \tag{1}$$

Die x- und y-Verschiebung um M_x und M_y bewirken, dass sich die Innenwände, wie von der Definition gefordert in M schneiden. Die Winkel der Geraden zur x-Achse sind so gewählt, dass sich diese um $\frac{\pi}{2}$ drehen und im rechten Winkel zueinander stehen. Damit können wir leicht die Funktion zur Beschreibung der Außenwände herleiten, da diese einfach immer einen Abstand von 1 zueinander haben. Es gilt die bekannte Formel

$$\sin^2\left(\frac{\theta_M}{2}\right) + \cos^2\left(\frac{\theta_M}{2}\right) = 1.$$

Somit müssen wir die beiden Summanden nur noch jeweils der x- und y-Verschiebung zuordnen.
Da wir den Korridor von der „linken“ zur „rechten“ Nullstelle bewegen, wollen wir, dass für $\theta_M = 0$ die rechte Wand ($x > M_x$) um 0 in y-Richtung und die linke Wand ($x \leq M_x$) um 1 in y-Richtung verschoben wird. Somit müssen wir $\cos\left(\frac{\theta_M}{2}\right)$ der y-Verschiebung zuordnen, da $\cos(0) = 1$ und $\cos\left(0 + \frac{\pi}{2}\right) = 0$. Dementsprechend wird $\sin\left(\frac{\theta_M}{2}\right)$ der x-Verschiebung zugeordnet. Es folgt damit für die Außenwände

$$W_A(x) = \begin{cases} \tan\left(\frac{\theta_M}{2}\right) \cdot \left(x - M_x + \sin\left(\frac{\theta_M}{2}\right)\right) + M_y + \cos\left(\frac{\theta_M}{2}\right) & \text{für } x \leq M'_x \\ \tan\left(\frac{\theta_M + \pi}{2}\right) \cdot \left(x - M_x - \sin\left(\frac{\theta_M + \pi}{2}\right)\right) + M_y - \cos\left(\frac{\theta_M + \pi}{2}\right) & \text{für } x > M'_x. \end{cases} \tag{2}$$

2.2 Ausschneiden des Sofas aus $\mathbb{R}^2$

Nun können wir mithilfe der beiden Funktionen, die zusammen den Korridor beschreiben, das Sofa konstruieren. Jedem M auf f ordnen wir ein θ_M zu, um für jeden Punkt in der Bewegung den Korridor zu bestimmen. So wird zu jedem Moment geprüft, welche Elemente von $\mathbb{R}^2$ im Korridor liegen. Dies geht einfach durch Einsetzen des x-Wertes x_0 des zu überprüfenden Elementes in $W_I(x)$ und $W_A(x)$. Damit muss $W_I(x_0) \geq y_0$ und $W_A(x_0) \leq y_0$ gelten, dass dieser Punkt innerhalb des Ganges liegt. Zu jedem Zeitpunkt sei also S_{θ_M} die Menge der Punkte, die innerhalb des Korridors liegen. So definiert die Menge

$$S = \bigcap_{M \in f} S_{\theta_M} \tag{3}$$

das Sofa, welches anhand von f ausgeschnitten wurde.

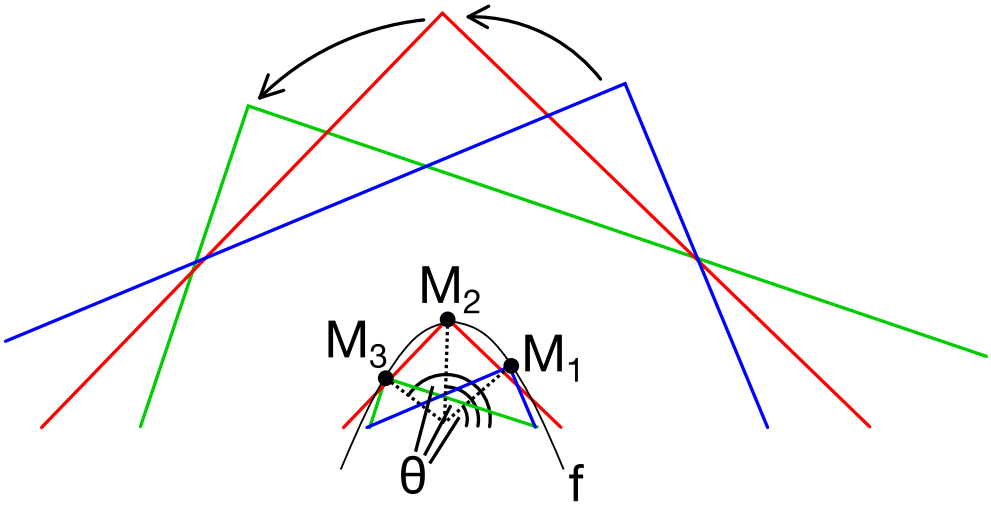

Abbildung 1: Bewegung des Korridors entlang von f mit θ_M.

3 Bedingungen der Idealität

In diesem Abschnitt wollen wir Eigenschaften behandeln, welche ein ideales Sofa erfüllen muss. Damit wollen wir die Wahl der Kurve f einschränken. Zum großen Teil folgen jedoch keine formellen Beweise, sondern viel eher Beweisideen, -skizzen oder Hinweise.

Lemma 1. *Es gibt ein ideales, symmetrisches Sofa.*

Beweis. Wir nehmen an das ideale Sofa sei nicht achsensymmetrisch. So kann das Sofa entlang der y-Achse in zwei Teile geteilt werden. Haben die Teile verschiedene Flächen, so kann das Teil mit der größeren Fläche gespiegelt werden, um ein größeres Sofa zu konstruieren, welches damit achsensymmetrisch sein muss. Im anderen Fall haben wir haben beide Flächen denselben Flächeninhalt. Auch hier kann wieder ein beliebiges Teil gespiegelt werden, um ein gleich großes, aber symmetrisches Sofa zu erhalten. So kann aus jedem nicht-achsensymmetrischen Sofa ein größeres oder gleich großes achsensymmetrisches Sofa konstruiert werden. □

Zusätzlich wurde mit der Konstruktion verschiedener Sofas statistisch recht klar, dass f, die teilweise linksgekrümmt waren, deutlich kleinere Flächen ergaben. So ist zum Beispiel für $f(x) = -10x^4 + x^2 + 0,64$ die Fläche $A \approx 2,09$ oder für $f(x) = 9x^4 - 5x^2 + 0,64$ ist $A \approx 1,97$. All dies sind recht kleine Flächen, wobei sich diese allgemeine Tendenz für alle, auch nur teilweise, linksgekrümmten Funktionen beobachten ließ.

4 Computergestützte Konstruktion

Wir implementieren anhand der soeben vorgestellten theoretischen Beschreibung in Java ein Programm, welches Sofas anhand beliebiger Kurven mit Parametern konstruiert und diese Parameter optimiert, um möglichst große Sofas zu erhalten. Dabei achten wir darauf, dass die gegebenen Kurven die Bedingungen in Abschnitt 3 erfüllen. Der gesamte Code zur Konstruktion der Sofas und Optimierung der Parameter

kann unter `https://github.com/PatrickSonnentag/SofaProblemApproximation` gefunden werden.

4.1 Implementation

Die Grundstruktur besteht aus den Klassen `Curve.java`, `Corridor.java`, `R2.java`, `Search.java` und `Sofa.java`. All diese Klassen bis auf `Search.java` sind zuständig für die Konstruktion einzelner Sofas. Diese Klasse wird dann genutzt, um für einen Typ von Kurve das größtmögliche Sofa zu konstruieren.

4.1.1 Konstruktion des Sofas

`R2.java` Zuerst wollen wir einschränken welche Teilmenge von $\mathbb{R}^2$ überhaupt relevant für ein einfach zusammenhängendes Sofa ist. Einmal muss offensichtlicher Weise für alle $(x, y) \in \mathbb{R}^2$ gelten, dass $y \in [0; 1]$. Nun ist die Frage wie groß die maximale Ausdehnung in x-Richtung sein kann.

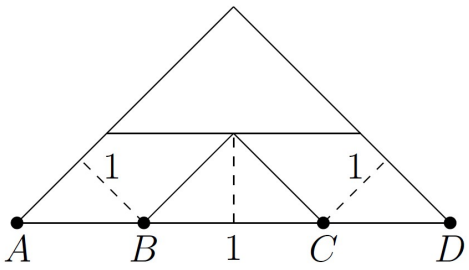

Abbildung 2: Maximale Länge des Sofas bei $\theta = \frac{\pi}{2}$.

Damit ist hier die gesuchte maximale Länge $|AD|$. Da das Sofa hier gerade so nicht zusammenhängend ist, muss die maximale Länge kleiner als das hiesige Resultat sein. $|BC| = 2$, da $\sqrt{\sqrt{2}^2 + \sqrt{2}^2} = \sqrt{4} = 2$. Genauso kann man über den Satz des Pythagoras $|AB| = |CD|$ berechnen: $|AB| = \sqrt{1^2 + 1^2} = \sqrt{2}$. Somit gilt in Summe

$$|AD| = |AB| + |BC| + |CD| = 2 + 2\sqrt{2}.$$

Da das Sofa erwiesenermaßen symmetrisch sein muss, reicht es die Hälfte dieser Punkte zu betrachten. Es muss also gelten

$$\forall (x, y) \in \mathbb{R}^2 : x \in \left[0; 1 + \sqrt{2}\right] \wedge y \in [0; 1]. \tag{4}$$

So implementieren wir diese Teilmenge in unserem Programm durch zwei einfache `for`-Schleifen, welche in beliebig kleinen Schritten hoch zählen, bis die soeben angegebenen Grenzen erreicht sind.
Gespeichert werden diese Werte so, dass die Punkte jeweils in Zeilen oder Spalten gespeichert sind, welche zusammen in einer Liste alle Punkte bilden. So kann das Sofa durch möglichst wenig Trapeze angenähert werden.

`Curve.java` Diese Klasse bietet einmal im Konstruktor die Möglichkeit Koeffizienten in Form eines `double`-Arrays anzugeben. Die Punkte M auf f finden wir, indem wir bei $x = 0$ anfangen und in kleinen Schritten so lange die x-Werte erhöhen, bis wir auf eine Nullstelle stoßen, womit wir alle relevanten Punkte gefunden haben. Da f achsensymmetrisch ist, können wir die Punkte mit negativen x-Wert einfach erzeugen, indem wir einmal alle gefundenen Punkte an der y-Achse spiegeln. Dabei wird auch jedes M mit θ_M identifiziert, indem

$$\theta_M = \arccos\left(\frac{M_x}{\sqrt{M_x^2 + M_y^2}}\right)$$

bestimmt wird. Überschreitet x aber den Wert $x = 1+\sqrt{2}$, so können wir alle folgenden Schritte terminieren, da dann die maximale Länge eins optimalen Sofas überschritten ist.
Ein Problem bei dieser Methode ist eine größere Ungenauigkeit bei großer Steigung in f, da wir nur den x Wert immer um denselben Wert erhöhen womit dort die verschiedenen M einen größeren Abstand haben.

`Corridor.java` Der Konstruktor dieser Klasse verlangt M_x, M_y und θ_M, um für jedes M die Funktion der Korridorwände zu bestimmen zu können. Sonst bietet diese Klasse noch eine Methode, um zu überprüfen, ob ein gegebener Punkt $P\,(x_0|y_0)$ im Korridor ist oder nicht. Zuerst wird aber überprüft, ob $x_0 \leq M_x$ oder $x > M_x$ ist, um die trigonometrischen Teile der Funktion nur für die relevanten Winkel berechnet. Die eigentliche Überprüfung erfolgt dann durch einfaches Einsetzen in $W_I(x)$ und $W_A(x)$.

`Sofa.java` Abschließend führt `Sofa.java` die letzten drei Klassen zusammen, um die Fläche A zu bestimmen. Zuerst werden die Punkte der Näherung an f genommen und zu jedem wird ein Korridor initialisiert. Nun wird für jeden Korridor überprüft, welche Punkte im Korridor liegen. Um zu vermeiden jeden Punkt einzeln überprüfen zu müssen, wird folgendermaßen für jede Zeile oder Spalte an Punkten vorgegangen: Von beiden Seiten schauen wir nach dem ersten Punkt, der in allen Korridoren ist. Dafür werden drei `for`-Schleifen benötigt; eine zum durchlaufen aller Zeilen oder Spalten, eine um alle Punkte in einer Zeile oder Spalte zu überprüfen und die innerste überprüft für jeden Punkt jeden zuvor initialisierten Korridor. So können wir durch diese getrennte Suche von oben und unten auch sofort die untere und obere Einhüllende des Sofas bestimmen:

```
for (ArrayList<double[]> doubles : r2) {
    boolean isInAllCorridors;     // Must be true after all iterations
    for (double[] aDouble : doubles) {     // Check lower points
        isInAllCorridors = true;
        // Check for every corridor every row of point with every point
        for (Corridor corridor : corridors) {
            // If lower point is not in every corridor we move on
            if (!(corridor.isInCorridor(aDouble[0], aDouble[1]))) {
                isInAllCorridors = false;
            }
        }
        if (isInAllCorridors) {
            lowerEnvelope.add(new double[]{aDouble[0], aDouble[1]});
            break;
        }
```

```
    }
    for (int j = doubles.size() - 1;
            j >= 0; j--) {  // Check upper points
        isInAllCorridors = true;
        // Check for every corridor every row of point with every point
        for (Corridor corridor : corridors) {
            // If lower point is not in every corridor we move on
            if (!(corridor.isInCorridor(doubles.get(j)[0],
                    doubles.get(j)[1]))) {
                isInAllCorridors = false;
            }
        }
        if (isInAllCorridors) {
            upperEnvelope.add(new double[]{doubles.get(j)[0],
                doubles.get(j)[1]});
            break;
        }
    }
}
```

Durch diese Vorgehensweise müssen im Schnitt weniger als die Hälfte der Punkte untersucht werden, da die ersten gefunden Punkte Teil der Einhüllenden sein müssen und für große Sofas sind diese auch sehr nah an den Punkten, an denen die Suche anfängt.
Dank dieses Verfahrens haben wir nun auch gleich die untere und obere Einhüllende getrennt. Nun können wir die Fläche A_O unter der oberen und A_U unter der unteren Einhüllenden berechnen. Zusammen gilt so $A = A_O - A_U$. Diese Flächen berechnen wir, indem wir jede Nachbarschaft zweier Punkte als Trapez bis zur x-Achse behandeln. Haben wir damit die Fläche berechnet ist es wichtig, dass wir aufgrund der Symmetrie nur die Hälfte des Sofas bestimmt, haben, womit das Ergebnis noch mit 2 multipliziert werden muss, um die tatsächliche Fläche des Sofas zu erhalten.

4.1.2 Optimierung der Parameter einer Kurve

Mit der obigen Implementation müssen wir nur noch verschiedene Kurven angeben, um den Flächeninhalt des aus f entstandenen Sofas zu bestimmen. f ist dabei eine Funktion mit n Koeffizienten. Wir wollen nun das Maximum dieser Funktion finden, beziehungsweise die Koeffizienten, die das größte Sofa ermöglichen. Dabei ist es wichtig zu beachten, dass wir möglichst wenig Funktionswerte von A bestimmen wollen, da dies sehr aufwändig ist. Zudem konnten wir beobachten, dass es (kleine) lokale Maxima gibt. Zur Vereinfachung gehen wir nur von $n = 1$ oder $n = 2$ in der Erklärung aus.

Grundsätzlich wird die Änderung der Fläche des Sofas untersucht, wenn man einen einzelnen Koeffizienten ändert, um tendenziell zu wissen, ob dieser größer oder eventuell kleiner werden sollte. Gibt es nur ein wirklich „großes“ Maximum, so kann man so dieses wahrscheinlich finden. Nichtsdestotrotz besteht das Problem, auf lokalen Maxima zu landen. Um sich in diesem Fall trotzdem von diesen entfernen zu können, muss der Koeffizient immer geändert werden, auch wenn A damit ein bisschen kleiner wird. So wird geschaut, welche Änderung des Koeffizienten die Fläche am wenigsten verringert, bevor der nächste Koeffizient geändert wird.

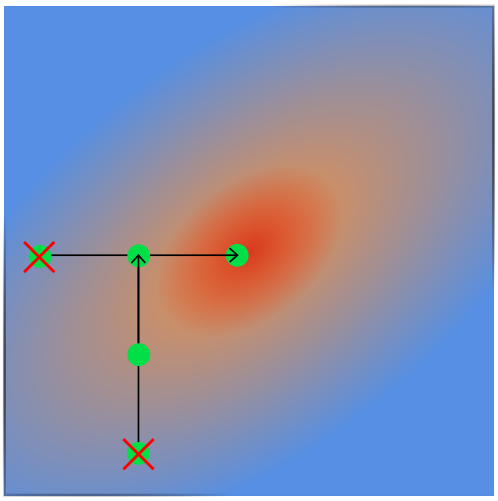

Abbildung 3: Bewegung auf einer dreidimensionelen Fläche, um das globale Maximum zu finden

Auch können damit nicht nur tatsächliche lokale Maxima umgangen werden, sondern auch Situationen, in denen es so wirkt, als wäre man auf einem Maximum, obwohl das nicht der Fall ist. Hier in der obigen Abbildung wird die Fläche im ersten Schritt in $-x$- und x-Richtung kleiner. Mit einer forcierten Bewegung kann man noch eher auf einer höheren Stelle landen, wie mit den Pfeilen verdeutlicht wird.
Aber auch noch mit diesem Vorgehen kann es zu weiteren Problemen kommen. Ist die Schrittgröße der Koeffizienten fest, so passiert es schnell, dass man in einer Art Schleife landet oder ein lokales Maximum trotzdem nicht überwinden kann.

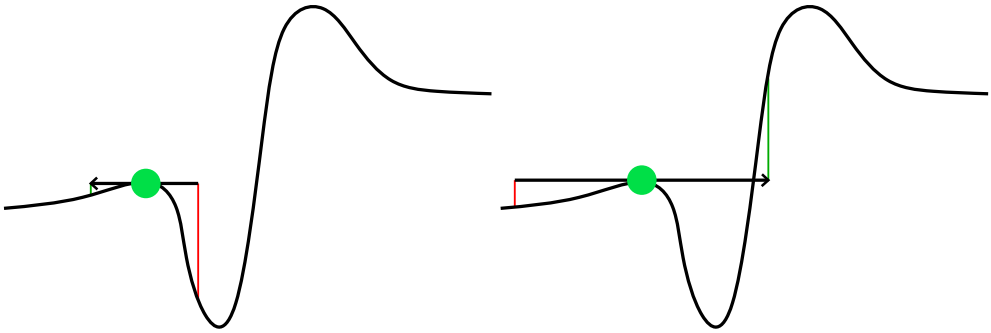

Abbildung 4: Wichtigkeit des Wählens verschiedener Schrittgrößen in der Optimierung

Um ebenfalls dieses Phänomen zu unterdrücken, wird für die Schrittgröße nur ein Intervall angeben, aus welchem ein Zufallswert als Schrittgröße genommen wird. Dieser wird immer neu generiert, um maximale Variation zu erreichen. Ist die Grenze des Intervall groß genug, können Extremfälle per Zufall wie rechts überwunden werden.

Natürlich ist aber auch dieses Verfahren nicht ideal. Neue Koeffizienten bedeuten so wohl nur eine lineare Zunahme der Laufzeitkomplexität, aber läuft das Programm seit einer Weile im selben Zufallsintervall, so werden die Werte kaum noch besser und

man muss manuell die Koeffizienten der bis jetzt größten Fläche übernehmen und ein feineres Intervall für die hinteren Nachkommastellen wählen, um wieder eine Verbesserung zu sehen. Auch ist der Algorithmus aufgrund der Zufallszahlen nicht determinant, man hat also nicht dasselbe Ergebnis, obwohl man dieselben Startparameter hatte. Auch ist ein Finden des globalen Maximums auch mit den obigen Maßnahmen nicht garantiert, da es dafür neben Glück auch nicht zu extreme Ausnahmen braucht.

4.2 Ungenauigkeiten

Da wir überall nur mit einfachen Fließkommazahlen rechnen, entstehen so überall im Prozess kleine Fließkommafehler. Diese sind aber so klein, dass sie von anderen, größeren Ungenauigkeiten überschattet werden und so nicht relevant sind. In diesem Abschnitt wollen wir auf genau diese Fehler eingehen und diese in ihrer Tragweite so weit wie möglich einschränken.
So gibt es insbesondere zwei große Fehlerquellen: Den Abstand der Punkte in `R2.java` aus denen das Sofa ausgeschnitten wird und die Annäherung an f durch einzelne Punkte.

Wie bereits erläutert, nähern wir f an, indem wir von $x = 0$ bis zur Nullstelle beliebig viele Funktionswerte bestimmen, wobei die eingesetzten x-Werte immer um dieselbe Zahl erhöht werden. Ähnlich verhält es sich in unserer Teilmenge von $\mathbb{R}^2$, da hier im Bereich mit $x \in \left[0; 1+\sqrt{2}\right] \wedge y \in [0;1]$ auch alle Punkte zum Nachbarn desselben x-Abstand δx und y-Abstand δy haben.

Zu kleine Flächen Dieser Fehler wird größer für größere δx und δy. Dadurch kommt es dazu, dass im richtigen $\mathbb{R}^2$ zu wenig Punkte dem Sofa zugeordnet werden würden.

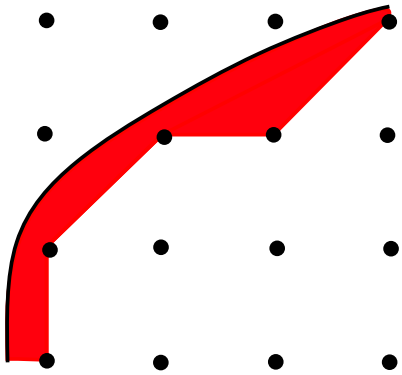

Abbildung 5: Fehler der zu zu kleinen Flächen führt

Hier in diesem Beispiel werden Punkte unter der Kurve dem Sofa zugeordnet. Wie man sieht, würde man in diesem Bereich zu kleine Flächen berechnen, da die hier rote Fläche noch zum Sofa gehören würde, aber nicht beachtet wird, da es keine weiteren Punkte gibt, die näher an der Kurve sind und damit die Einhüllende bilden würde. So kann es im extremen und sehr unwahrscheinlichen Fall dazu kommen, dass für jede zwei Nachbarn ein $\delta x \cdot \delta y$-Rechteck zu wenig beachtet wird. Es sei nochmal betont, dass dieser Fall eine extreme Übertreibung ist. Wir sagen l_I und l_A sind jeweils die Abstände der Nullstellen der jeweils inneren und äußeren Einhüllenden. Dann ist der

maximale Flächenverlust durch diesen Fehler

$$(\delta x \delta y)(l_i + l_A).$$

Zu große Flächen Hierzu kommt es hingegen, wenn die Näherung an f zu ungenau ist, beziehungsweise, wenn es zu wenig M oder Korridore gibt.

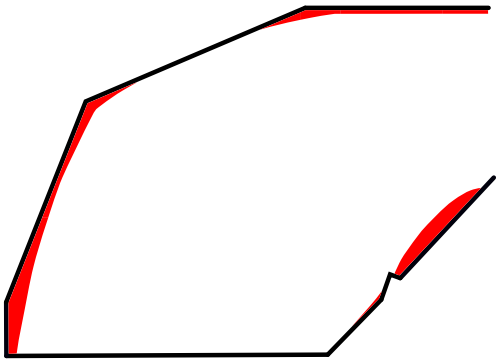

Abbildung 6: Fehler der zu zu großen Flächen führt

Wir sehen in schwarz die Begrenzungen durch vier Korridore, welche alle durch einen anderen Punkt M definiert wurden. In rot sieht man die Fläche, die das eigentliche Sofa, welches mit mehreren hundert Korridoren gebildet wurde, nicht mehr hat. Hier kann keine allgemeine Fehlerabschätzung gemacht werden, da dieser Fehler bei nur einem Korridor fast beliebig groß werden könnte. Jedoch kann dieser Fehler durch eine bessere Näherung an f leicht minimiert werden, aber wie bereits erwähnt, ist das für steilere Kurven aufwändiger.

Insgesamt konnten wir diese beiden Fehlerquellen soweit verringern, dass wir Hammersleys Sofa auf vier Nachkommastellen anhand von $f(x) = \sqrt{\frac{4}{\pi^2} - x^2}$ genau konstruieren konnten, womit wir allgemein von vier korrekten Nachkommastellen ausgehen.

4.3 Suche

Durch das Erben von der Klasse `Curve.java` haben wir die Funktionen

$$f_1(x) = \frac{b}{a}\sqrt{a^2 - x^2} \tag{5}$$

$$f_2(x) = a\frac{e^{cx} + e^{-cx}}{b} + d \tag{6}$$

$$f_3(x) = a \sec(bx) + c \tag{7}$$

$$f_4(x) = \sum_{i_0}^{7} a_{2i} x^{2i} \tag{8}$$

untersucht und die Parameter optimiert. Dabei war das erste gewählte Zufallsintervall $[1 \times 10^{-1}; 1 \times 10^{-2}]$, nachdem dies keine weiteren Verbesserungen brachte, wurde das Intervall zu $[1 \times 10^{-2}; 1 \times 10^{-3}]$ verfeinert. Nur bei der Hyperbel war dies anders, da

sich dort mit nur kleinen Änderungen sehr viel ändern kann. Weitere Verfeinerungen brachten keinen weiteren merklichen Vorteil. Wobei die Polynomfunktionen der mit Abstand größte Rechenaufwand ist, hat sich schnell abgezeichnet, dass es eine riesige Erhöhung des Grades bedarf, um größere Sofas zu finden, womit wir hier nur bis zum 14-ten Grad gesucht haben. Zusätzlich haben wir das Ergebnis der Polynomfunktion mit einer Brute-Force-Suche bestätigt, wobei unser oben beschriebenes Verfahren schneller einen leicht besseren Wert geliefert hat.
Auch haben wir alle resultierenden Flächen von $f(x) = -ax^2 + c$ errechnet, um die Vermutung über ein großes Maximum zu untersuchen, da so unser Algorithmus sehr gut funktionieren würde.

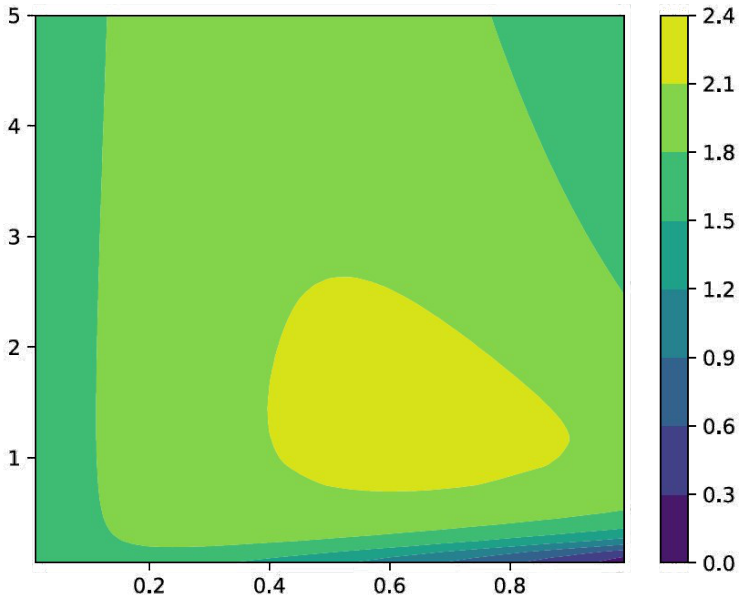

Abbildung 7: Graph der Flächeninhalte durch Konstruktion mit $f(x) = -ax^2 + c$. a auf der y- und c auf der x-Achse.

So ist um $a \approx 1,5$ und $c \approx 0,6$ ein klares Maximum zu sehen, welches zu allen anderen Seiten abfällt. Lokale Maxima könnten hier immer noch existieren, sind aber relativ klein und können so von unserer zufälligen Schrittgröße leicht übergangen werden.

5 Ergebnisse

In der untenstehenden Tabelle sind alle Ergebnisse tabellarisch aufgelistet. Unsere größtes Sofa konnten wir mit einer Ellipse konstruieren, womit $A \approx 2,2079$.

Funktion	**Parameter gerundet**	**A**
Ellipse	$a \approx 0,6301$ $b \approx 0,6450$	2,2079
Hyperbel	$a \approx -0,7065$ $b \approx 2,2219$ $c \approx -1,9119$ $d \approx 1,2886$	2,1869
Sekans	$a \approx -0,4192$ $b \approx 1,7840$ $c \approx 1,0605$	2,20485
Polynom	$a_0 \approx 0,6449$ $a_2 \approx -0,8044$ $a_4 \approx -0,1755$ $a_6 \approx -0,9479$ $a_8 \approx -2,9591$ $a_{10} \approx -4,4303$ $a_{12} \approx -6,4655$ $a_{14} \approx -8,5473$	2,2061

Tabelle 1: Ergebnisse & Laufzeit für alle untersuchten Kurven

6 Zusammenfassung

Wir eine Beweisskizzen zur Symmetrie des Sofas aufgestellt. Zudem gibt es noch eine kleine statistische Beobachtung über die Krümmung der Kurve f, anhand welcher das Sofa konstruiert wird. Ebenso haben wir in Java die zuvor beschrieben Konstruktion anhand von f für verschiedene Kurven implementiert. Dadurch stellen wir eine Suche an, die sich dadurch auszeichnet, dass ein weiterer Koeffizient nur dazu führt, dass zwei Sofas mehr pro Schnitt berechnet werden müssen; die Laufzeitkomplexität unserer Suche ist also $\mathcal{O}(n)$. Als Kurven für f wählen wir solche, die zu den Nullstellen möglichst große Steigung aufweisen, da dies wieder zu größeren Flächen zu führen scheint. Eine Ausnahme dazu ist in gewissermaßen die Polynomfunktion, welche wir nur wählen, da diese theoretisch jede andere Kurve beliebig gut annähern können. Jedoch werden diese schnell vergleichsweise sehr rechenintensiv, da wir so die meisten Koeffizienten haben. Damit lässt sich unser größtes Sofa anhand von

$$f(x) \approx \frac{0,6450}{0,6301}\sqrt{0,6301^2 - x^2} \tag{9}$$

konstruieren. Die dazugehörige Fläche ist $A \approx 2,2079$. Damit ist dieses Sofa um ungefähr 0,0116 Flächeneinheiten kleiner als Gervers Sofa. Auch rein optisch bestehen Ähnlichkeiten.
Auch unsere Konstruktion besitzt an der inneren Einhüllenden leicht abgerundete Ecken, jedoch weniger stark ausgeprägt, welche auf Gervers Sofa hinweisen, da dieses sich durch diese Ecken auszeichnet.

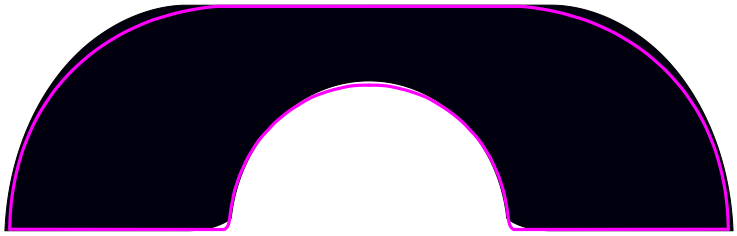

Abbildung 8: Gervers Sofa (schwarz) im Vergleich zu unserer Konstruktion (magenta)[1]

7 Ausblick

Unsere computergestützte Konstruktion war wohl in der Lage Sofas zu finden, die etwas größer sind als Hammersleys Konstruktion, aber dennoch sind wir noch ein Stück von Gervers Ergebnis entfernt. Somit ist es in der Zukunft sinnvoll weitere Kurven zu untersuchen, welche zu den Nullstellen eine hohe Steigung aufweisen, höhergradige Polynomfunktionen für bessere Näherungen oder solche, die aus mehreren Kurven zusammengesetzt ist, wie es auch bei Gerver der Fall ist. Der Algorithmus könnte ebenfalls an ein paar Stellen optimiert werden, wobei wahrscheinlich größere Geschwindigkeit-Gewinne durch eine Implementierung in Rust oder C++ gemacht werden können, da Java nicht geeignet für wissenschaftliche Berechnungen ist. Damit könnten auch mit höherer Präzision gerechnet werden. Zuletzt ist eine Näherung von f in Abhängigkeit von f'' wichtig, um Fehler zu vermeiden, da momentan Steile Stücke schlecht angenähert werden, was insbesondere ein Problem ist, da es so scheint, als würden genau diese großen Steigungswerte bei den Nullstellen zu größeren Sofas führen.

Literatur

[1] Wikimedia Commons. *File:Gerver.svg — Wikimedia Commons, the free media repository*. [Online; Zugriff am 25. Februar 2022]. 2020. URL: `https://commons.wikimedia.org/w/index.php?title=File:Gerver.svg&oldid=512271264`.

[2] Joseph L Gerver. „On moving a sofa around a corner“. In: *Geometriae Dedicata* 42.3 (1992), S. 267–283.

[3] J. M. Hammersley. „On the enfeeblement of mathematical skills by modern mathematics and by similar soft intellectual trash in schools and Universities“. In: *Educational Studies in Mathematics* 1.1-2 (1968), S. 17–17. DOI: `10.1007/bf00426226`.

[4] Yoav Kallus und Dan Romik. „Improved upper bounds in the moving sofa problem“. In: *Advances in Mathematics* 340 (2018), S. 960–982.

Unterstützungsleistungen

Dieses Projekt wurde durch das *Schülerforschungszentrum Friedrichshafen* begleitet und finanziell wie auch ideell unterstützt. Darunter fällt die Betreuung durch Herrn *Sandor Spieß*, welcher stets zur Seite stand, insofern Hilfe benötigt wurde. Ebenso

stellte das Schülerforschungszentrum Friedrichshafen die Gelder zum Drucken des Plakates für die Ausstellung. Zuletzt möchte ich auch das Design des Deckblattes durch meinen Mitschüler *Felix Kunze* anmerken.