Kollegium Spiritus Sanctus Brig

Maturitätsarbeit 2025/2026

Invarianten in der Knotentheorie

Von:

Grosjean Annaëlle Sophie Cumba, 5C

Eingereicht im Fachbereich Mathematik

Betreut durch:

Imboden Ingemar David

Brig, den 26.11.2025 Nummer 2025/72

Inhaltsverzeichnis

1 Abstract **1**

2 Einleitung **2**
2.1 Geschichtlicher Hintergrund 2
2.2 Idee eines mathematischen Knotens 2

3 Grundlagen **4**
3.1 Definition eines Knotens 4
3.2 Primknoten 5
3.3 Äquivalenz von Knoten 5
3.4 Diagramme und Projektionen 6
3.5 Orientierung 8
3.6 Reidemeister-Bewegungen 8

4 Invarianten **9**
4.1 Grundfragen der Knotentheorie 9
4.2 Kreuzungszahl 10
4.3 Entknotungszahl 11
4.4 Dreifärbbarkeit 12
4.5 p-Färbbarkeit 16
4.6 Anzahl der p-Färbbarkeiten 29
4.7 Alexander-Polynom 31

5 Python-Programm **34**
5.1 Gitter-Diagramm 34
5.2 Struktur des Programms 35
5.3 Tabelle der p-Färbbarkeiten 38

Fazit **39**

Danksagung **40**

Anhang **41**

Erklärung zu generativer KI **62**

Erklärung zu online-tools zur sprachlichen Unterstützung **63**

Abbildungsverzeichnis **64**

Literaturverzeichnis **66**

Authentizitätserklärung **68**

Kapitel 1

Abstract

Diese Maturitätsarbeit bietet eine Einführung in die Knotentheorie, beginnend mit ihrer historischen Entwicklung und den mathematischen Grundlagen. Die Arbeit stellt das Konzept der Invarianten vor und stellt die am leichtesten zugänglichen Invarianten konkret vor. Die Invarianten in dieser Arbeit sind: die Kreuzungszahl, die Entknotungszahl, die 3-Färbbarkeit, die p-Färbbarkeit, die Anzahl p-Färbbarkeiten und das Alexander-Polynom. Als praktische Anwendung wird ein Python-Programm entwickelt, das die p-Färbbarkeit beliebiger eingegebener Knoten berechnet. Das Programm führt anschliessend einen Vergleich der p-Färbbarkeit mit einer Knotentabelle durch, um entsprechende Knoten vorzuschlagen.
Die Inhalte sind für Schülerinnen und Schüler der Mittelstufe mit Schwerpunktfach oder Ergänzungsfach *Anwendungen der Mathematik* (AdM) oder mehr mathematischem Wissen zugänglich.

Kapitel 2

Einleitung

Das Thema Knoten begegnet uns im Alltag auf unterschiedliche Art und Weise, sei es einfach beim Schnüren unserer Schuhe. Wer Sportarten wie Bergsteigen, Klettern oder Segeln betreibt, entwickelt eine noch engere Beziehung zu Knoten, da deren sichere Anwendung dort oft lebenswichtig ist. Viele könnten einen Kreuzknoten, einen Achterknoten oder einen Palstek-Knoten sogar mit geschlossenen Augen knüpfen.
Ich persönlich fand Knoten und die Art, wie sie geknüpft werden, schon immer faszinierend. Sie sind ein fester Bestandteil unseres Alltags und werden sowohl aufgrund ästhetischer als auch technischer Eigenschaften geschätzt. Mit den in dieser Arbeit behandelten mathematischen Knoten, die sich grundlegend von den “klassischen” Knoten unterscheiden, bin ich erstmals durch das Video *The Insane Math of Knot Theory* des Wissenschaftskanals *Veritasium* in Kontakt gekommen. Beim Durchsehen der Themenliste für die Maturitätsarbeit erkannte ich sofort, dass dies die perfekte Gelegenheit wäre, dieses kurz zuvor entdeckte und faszinierende Thema vertieft zu untersuchen.

2.1 Geschichtlicher Hintergrund

Das Studium der mathematischen Knoten, die sogenannte Knotentheorie, begann im 19. Jahrhundert mit dem englischen Mathematiker und Physiker William Tomson, später Lord Kelvin genannt. Er stellte die Hypothese auf, dass Atome stabile, verknotete Ätherwirbel seien [19]. Sein Vorschlag war, dass die verschiedenen Elemente unterschiedlichen Knoten entsprechen. Ein Versuch Knoten systematisch zu klassifizieren wurde daher vom englischen Mathematiker Peter Guthrie Tait unternommen [19]. Seine erste Klassifikation sortierte Knoten nach Anzahl ihrer Kreuzungen. Obwohl Tomsons Theorie zusammen mit der Existenz des Äthers 1887 mit dem Michelson-Morley-Experiment zum luminiferen Äther verworfen wurde, blieb das Interesse an Knoten und die Knotentheorie als reine mathematische Kuriosität bestehen [12].

2.2 Idee eines mathematischen Knotens

Zunächst muss klargestellt werden, dass ein mathematischer Knoten nicht gleichbedeutend einem gewöhnlichen Knoten ist, wie beispielsweise einem Achterknoten beim Klettern. Intuitiv muss man sich einen mathematischen Knoten wie ein Seil vorstellen, das geknüpft wurde und dessen beide Enden zusammen-

geklebt wurden. Der so entstandene Knoten kann nicht gelöst werden, es sei denn, man schneidet das Seil durch. Der Knoten ist anders gesagt eine einfache geschlossene Kurve im $\mathbb{R}^3$, dem dreidimensionalen Raum ([7], Seite 11).

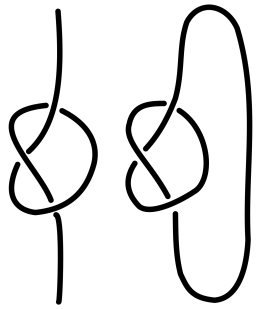


Abbildung 1: Gewöhnlicher Knoten (links) und mathematischer Knoten (rechts).

Kapitel 3

Grundlagen

Um Knoten besser zu verstehen und eingehend analysieren zu können, werden sie formal definiert. Der gesamte Abschnitt basiert auf den Seiten 11 bis 29 des Buches “Knotentheorie für Einsteiger” von Charles Livingston [7].

3.1 Definition eines Knotens

Analytisch lässt sich ein Knoten als stetige Funktion im Intervall $[0, 1]$ mit $f(0) = f(1)$ und $f(x) = f(y)$ bestimmen. Diese Definition führt jedoch zu Problemen wie dem in Abbildung 2 sichtbaren “wilden Knoten”. Ein solcher “wilder Knoten” ist eine unendlich oft verknotete Schlinge.

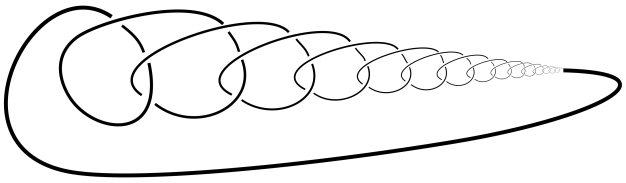


Abbildung 2: Ein “wilder Knoten”.

Dieses Problem kann gelöst werden, wenn ein Knoten als eine differenzierbare Funktion definiert wird. Für diese Arbeit genügt jedoch eine einfachere, geometrische Definition.
Zwei Punkte p und q im $\mathbb{R}^3$ seien durch die Strecke $[p, q]$ verbunden. Es sei $(p_1, ..., p_n)$ eine geordnete Menge voneinander verschiedener Punkte. Die Vereinigung der Strecken $[p_1, p_2]$,$[p_2, p_3]$,...,$[p_{n-1}, p_n]$, $[p_n, p_1]$ bildet einen *geschlossenen Polygonzug*.
Die Elemente der Menge p_i heissen *Eckpunkte* des Knotens. Wenn jede Strecke genau zwei andere schneidet, und zwar jede an einem Endpunkt, dann heisst die Kurve *einfach*.

Definition 3.1.1. Ein *Knoten* ist ein einfacher geschlossener Polygonzug im $\mathbb{R}^3$.

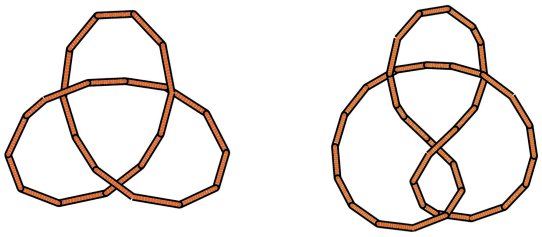


Abbildung 3: Kleeblattknoten 3_1 (links) und Achterknoten 4_1 (rechts) als Polygonzüge.

3.2 Primknoten

Definition 3.2.1. Ein Knoten heisst *Primknoten*, wenn er nicht in zwei nichttriviale Knoten zerlegbar ist.

Bemerkung 3.2.2. Der triviale Knoten auf Bild 4 zu sehen wird auch *Unknoten* genannt. Er ist der einfachste Knoten, bestehend aus einer einfachen geschlossenen Schlaufe. Es wird mit 0_1 bezeichnet.

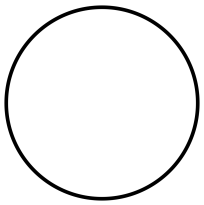


Abbildung 4: Der Unknoten 0_1.

Umgekehrt ist ein zusammengesetzter Knoten die zusammenhängende Summe zweier nichttrivialen Knoten K und J. Diese Summe wird als $K\#J$ geschrieben.
In dieser Arbeit werden ausschliesslich Primknoten betrachtet.

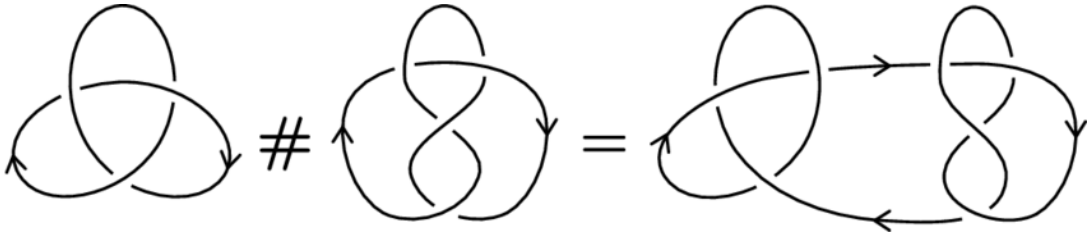


Abbildung 5: Zusammenhängende Summe $K\#J$ von zwei Primknoten.

3.3 Äquivalenz von Knoten

Die Äquivalenz von Knoten mag intuitiv offensichtlich sein, muss aber vor einer weiteren Untersuchung von Knoten mathematisch definiert werden.

Definition 3.3.1. Ein Knoten J, der durch die geordnete Punktemenge $(p_1, p_2, ..., p_n)$ bestimmt wird, heisst *elementare Deformation* des Knotens K, der durch die geordnete Punktemenge $(p_0, p_1, ..., p_n)$ bestimmt ist, wenn:

(i) Die Punkte p_0, p_1 und p_2 nicht kollinear sind. Das heisst, dass sie nicht auf einer Geraden liegen.

(ii) Das von (p_0, p_1, p_n) aufgespannte Dreieck schneidet den Knoten J nur in der Strecke $[p_1, p_n]$.

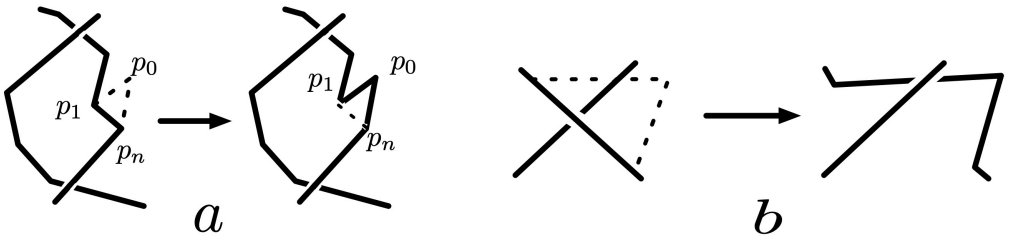


Abbildung 6: Erlaubte elementare Deformation (rechts) und nicht erlaubte elementare Deformation (links).

Bildlich gesprochen darf das Seil, in das der Knoten geknüpft ist, nur auf “klassische” Weise gehandhabt werden. Es darf weder durchgeschnitten werden, noch darf es durch sich selbst hindurchlaufen.

Definition 3.3.2. Zwei Knoten K und J heissen *äquivalent*, wenn es eine Folge von Knoten $K = K_0, K_1, ..., K_n = J$ gibt, sodass K_i für $i > 0$ eine elementare Deformation von K_{i-1} ist.

Zwei Knoten K und J sind also genau dann äquivalent, wenn sich K durch eine Folge von elementaren Deformationen in J überführen lässt.

3.4 Diagramme und Projektionen

Knoten sind Teil des $\mathbb{R}^3$, werden aber sehr oft auf dem $\mathbb{R}^2$ projiziert. Tatsächlich erfolgt die Arbeit auf Papier oder Bildschirmen.
Wird eine Projektion nicht sorgfältig gewählt, kommt es zu einem Informationsverlust. Bevor dieser beschrieben werden kann, gilt es zunächst, eine Terminologie festzulegen. Das Bild eines Knotens K heisst die *Projektion* von K. Die *Kreuzungen* sind doppelte Punkte in der Projektion.
Der erste Informationsverlust ist die räumliche Anordnung der Knotenabschnitte an den Kreuzungen. Um zu erkennen, welche Teile des Knotens über andere Teile laufen, lässt man Lücken in der Zeichnung der Projektion. Eine solche Zeichnung heisst *Knotendiagramm*. Das Knotendiagramm besteht aus *Bögen*. Es

gibt gleich viele Bögen wie Kreuzungen. An jeder Kreuzung gibt es eine *Überführung* oder *Überkreuzung* und eine *Unterführung* oder *Unterkreuzung*, die mit einer Lücke gekennzeichnet ist.

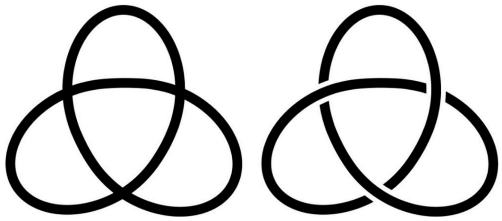


Abbildung 7: Projektion (links) und Diagramm (rechts) eines Kleeblattknotens.

Der andere Informationsverlust ist rechts auf der Abbildung 8 zu sehen. Die Lage der Eckpunkte in Bezug auf die Bögen lässt sich nicht klar unterscheiden. Es gibt beispielsweise zwei überlappende Bögen, oder auch einen Eckpunkt, der mit einem Bogen zusammenfällt.

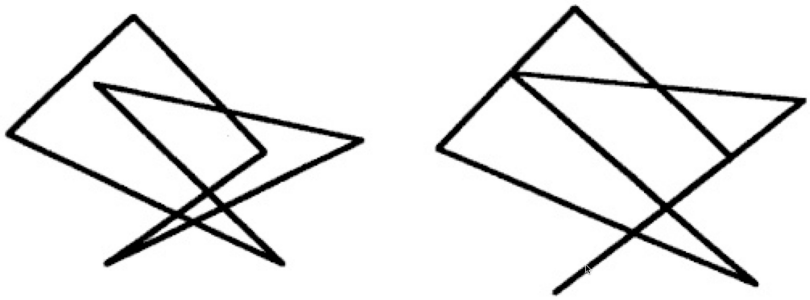


Abbildung 8: Reguläre Projektion (links) und Projektion mit Informationsverlust (rechts).

Definition 3.4.1. Eine Knotenprojektion heisst *reguläre Projektion*, wenn nie drei Punkte des Knotens auf denselben Punkt projiziert werden, und kein Eckpunkt auf denselben Punkt projiziert wird, wie irgendein anderer Punkt des Knotens.

Die Knoten und ihre Diagramme sind eng miteinander verbunden.

Satz 3.4.2. Wenn zwei Knoten K und J reguläre Projektionen und identische Diagramme besitzen sind sie äquivalent.

3.5 Orientierung

Knoten können mit einer Orientierung versehen werden, das heisst, mit einer festgelegten Laufrichtung.

Definition 3.5.1. Ein *orientierter* Knoten besteht aus einem Knoten und einer Anordnung seiner Eckpunkte. Zwei Anordnungen sind *äquivalent,* wenn sie sich nur durch eine zyklische Permutation unterscheiden.

Die Orientierung eines Knotens wird durch einen Pfeil in seinem Diagramm wie im Bild 9 zu sehen dargestellt.

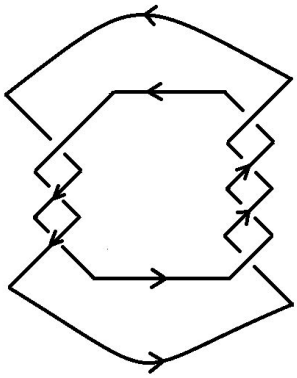


Abbildung 9: Diagramm eines orientierten Knotens.

3.6 Reidemeister-Bewegungen

Wenn ein Knoten deformiert wird, gibt es drei einfache Veränderungen, siehe Abbildung 10, die in einem Diagramm auftreten. Diese drei Bewegungen, die nach dem deutschen Mathematiker Kurt Reidemeister benannt wurden, sind mit ihren Inversen gepaart. Es gibt also insgesamt sechs mögliche Reidemeister-Bewegungen.

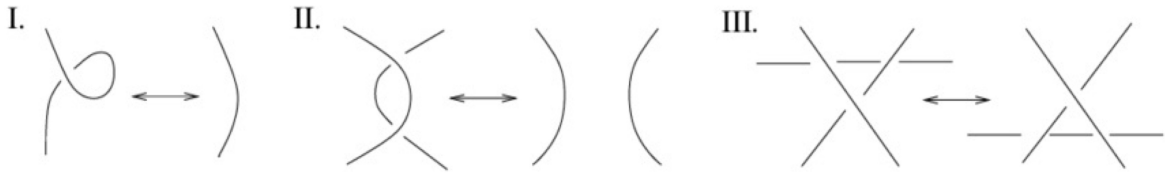


Abbildung 10: Die 3 Reidemeister-Bewegungen.

Satz 3.6.1. Sind zwei Knoten äquivalent, dann lassen sich ihre Diagramme durch eine endliche Folge von Reidemeister-Bewegungen ineinander überführen.

Kapitel 4

Invarianten

4.1 Grundfragen der Knotentheorie

Die wichtigste und immer noch aktuelle Frage der Knotentheorie lautet: Wie kann man zwei Knoten voneinander unterscheiden? Auch das Gegenteil steht zur Diskussion: Wie kann man sicher sein, dass zwei Knoten äquivalent sind?
In der Abbildung 11 ist ein Knotendiagramm dargestellt, das auf den ersten Blick sehr komplex erscheint. Kaum jemand würde vermuten, dass es sich dabei um eine Projektion des trivialen Knotens handelt.

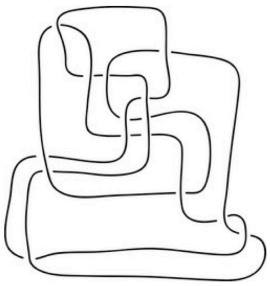


Abbildung 11: Unknoten von Thistlethwaite.

Dieses Beispiel verdeutlicht, dass die Beantwortung der Fragen zur Unterscheidung von Knoten selten einfach ist. Zur Feststellung der Äquivalenz von Knoten spielt das Konzept der Knoteninvariante eine zentrale Rolle.

Definition 4.1.1. Eine Knoteninvariante ist eine Menge oder Eigenschaft eines Knotens, die durch Reidemeister-Bewegungen, also über alle möglichen Knotendiagramme, unverändert bleibt [9].

Eine Invariante kann eine numerische Grösse, eine Eigenschaft oder sogar ein Polynom sein. Die Invarianten schaffen somit eine Verbindung zwischen der geometrischen Knotentheorie und der Algebra.
Wenn zwei Knoten bei der Berechnung derselben Invariante ein unterschiedliches Ergebnis aufweisen,

kann mit Sicherheit bestimmt werden, dass es sich dabei um zwei verschiedene oder nicht äquivalente Knoten handelt. Umgekehrt ist es unmöglich zu behaupten, dass zwei Knoten äquivalent sind, wenn sie die gleiche Invariante haben. Das ultimative Ziel der Knotentheorie ist daher die Suche nach einer vollständigen Invariante, die jeden Knoten eindeutig charakterisiert [15].
Eine Invariante muss zwei Merkmale erfüllen. Sie muss:

(i) *wohl definiert* sein. Das bedeutet, dass die Art und Weise wie sie berechnet wird klar und unmissverständlich sein muss.

(ii) von den Reidemeister-Bewegungen nicht beeinflusst werden. Um *invariant* zu sein, kann dieses Merkmal natürlich nicht durch eine unterschiedliche Darstellung desselben Knotens verändert werden.

Nun werden einige numerische Invarianten eingeführt.

4.2 Kreuzungszahl

Jede reguläre Projektion eines Knotens besitzt eine endliche Zahl von Doppelpunkten, die als Kreuzungen bezeichnet werden (siehe 3.4).

Definition 4.2.1. Die *Kreuzungszahl* $c(K)$ ist die kleinstmögliche Anzahl von Doppelpunkten in einer Projektion eines Knotens ([7], Seite 117).

Es geht hier um “kleinstmöglich”, weil die Reidemeister-Bewegungen vom Typ I und II diese Anzahl beeinflussen. Zum Beispiel fügt das Ausführen einer Reidemeister-Bewegung I, also das Erstellen einer Schleife, dem Diagramm dieses Knotens eine Kreuzung hinzu.

Beispiel 4.2.2. Der Unknoten hat eine $c(K) = 0$. Der Kleeblattknoten hat eine $c(K) = 3$.

Bemerkung 4.2.3. Alle Werte von $c(K) = x$ sind möglich für $x \in \mathbb{N}_0$ ausser $x = 1$ und $x = 2$. Der Grund dafür ist, dass ein Knoten mit einer Kreuzung der triviale Knoten ist, nachdem eine Reidemeister-Bewegung vom Typ I ausgeführt wurde.

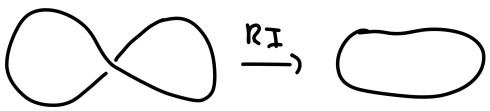


Abbildung 12: Knoten mit $c(K) = 1$.

Ein Knoten mit zwei Kreuzungen ist nach der Ausführung einer Reidemeister-Bewegung vom Typ II oder von zwei Reidemeister-Bewegungen vom Typ I ebenfalls der triviale Knoten.

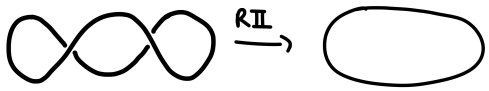


Abbildung 13: Knoten mit $c(K) = 2$, Fall 1.

Abbildung 14: Knoten mit $c(K) = 2$, Fall 2.

4.3 Entknotungszahl

Jeder Knoten kann trivial werden, wenn man eine bestimmte Anzahl seiner Kreuzungen verändert. Eine Kreuzung zu ändern bedeutet, dass die Überkreuzungen zu Unterkreuzungen werden und umgekehrt.

Definition 4.3.1. Die *Entknotungszahl* $e(K)$ bezeichnet die kleinste Anzahl von Kreuzungsänderungen, die in einem beliebigen Diagramm des Knotens K erforderlich sind, um diesen in den trivialen Knoten zu überführen ([7], Seite 118).

Ein einfacher Algorithmus erlaubt eine Bestimmung der Anzahl der Änderungen: Zunächst wird ein neues Diagramm dieses Knotens gezeichnet, ausgehend von einem willkürlichen Punkt im ursprünglichen Diagramm. Anschliessend wird die Projektion durchgelaufen. Jede Kreuzung wird also zweimal passiert. Beim zweiten Passieren wird sichergestellt, dass jede Kreuzung eine Unterkreuzung ist. Das resultierende Knotendiagramm wird unweigerlich mit dem trivialen Knoten übereinstimmen.
Der Anfangspunkt zum Durchlaufen des Diagramms kann eine Rolle spielen. Ergibt sich eine Anzahl von zu ändernden Kreuzungen, die grösser als die Hälfte der Gesamtanzahl der Kreuzungen $c(K)$ ist, müssen stattdessen alle anderen Kreuzungen geändert werden.

Beispiel 4.3.2. Der Knoten in Abbildung 15 besitzt eine Entknotungszahl von zwei. Das heisst, wenn die beiden markierten Kreuzungen auf dem Bild geändert werden, wird dieser Knoten zu einer anderen Projektion des trivialen Knotens.

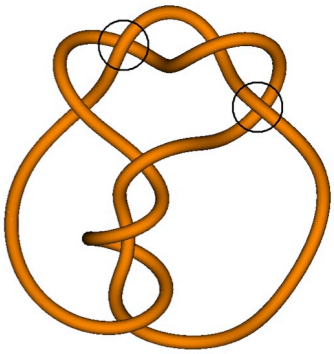


Abbildung 15: Knoten mit einer Entknotungszahl $e(K) = 2$.

Bemerkung 4.3.3. Der triviale Knoten hat offensichtlich die Entknotungszahl $e(K) = 0$, da er keine Kreuzungen besitzt, die verändert werden könnten, und daher äquivalent mit sich selbst ist.

Anschliessend werden zwei Invarianten behandelt, bei denen es sich um Eigenschaften von Knoten handelt.

4.4 Dreifärbbarkeit

Wie der Name schon erkennen lässt, bedeutet es, wenn ein Knoten dreifärbbar ist, dass man jeden Bogen des Knotens mit einer von drei möglichen Farben einfärben kann. Diese Einfärbung ist jedoch nicht zufällig, denn es gibt Regeln, die eingehalten werden müssen.

Definition 4.4.1. Ein Knoten ist dreifärbbar, wenn jeder Bogen mit einer von drei Farben so eingefärbt werden kann, dass:

(i) mindestens zwei verschiedene Farben benutzt werden.

(ii) an jeder Kreuzung, an der zwei Farben vorkommen, müssen alle drei Farben vorhanden sein ([7], Seite 30).

Beispiel 4.4.2. Auf der Abbildung 16 ist ein Beispiel eines dreifärbbaren Knotens zu sehen. Die drei verwendeten Farben Rot, Blau und Grün sind aus Gründen der Lesbarkeit auch mit den Buchstaben R, B und V markiert.

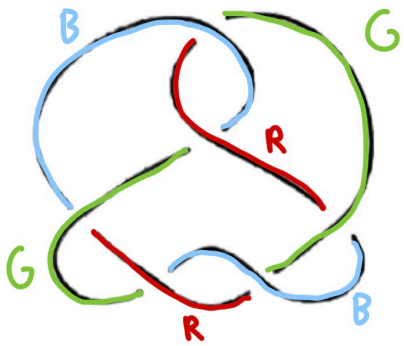


Abbildung 16: Dreifärbung des Knotens 6_1.

Beispiel 4.4.3. Der Achterknoten 4_1 ist im Gegensatz zum Knoten 6_1 nicht dreifärbbar. Die Abbildung 17 zeigt eine Einfärbung. Sei der Bogen x_1 blau (o.B.d.A.) und die Kreuzung k_1 einfarbig. Nach 4.4.1 müssen auch die Bögen x_2 und x_4 blau sein. An der Kreuzung k_4 treffen sich die Bögen x_2 und x_1, die beide blau sind. Nach 4.4.1 muss der Bogen x_3 ebenfalls blau sein. Alle vier Bögen sind somit blau gefärbt, was gegen die zweite Regel der Dreifärbbarkeit verstösst (siehe 4.4.1). Eine solche Einfärbung, bei der alle Bögen die gleiche Farbe haben, wird als *trivial* bezeichnet.

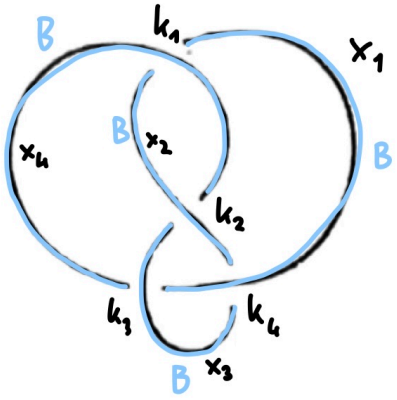


Abbildung 17: Unerlaubte oder triviale Färbung des Knotens 4_1.

Die Abbildung 18 zeigt eine Einfärbung. Sei die Kreuzung k_1 dreifarbig und der Bogen x_1 blau. Dann hat der Bogen x_2 eine andere Farbe, beispielsweise rot. Hier hätte man stattdessen auch grün problemlos wählen können. An der Kreuzung k_4 treffen sich x_2 (rot) und x_1 (blau). Da zwei verschiedene Farben vorhanden sind, muss k_4 dreifarbig sein und der Bogen x_3 grün gefärbt werden. An der Kreuzung k_2 treffen sich x_3 (grün) und x_2 (rot). Der nächste Bogen x_4 müsste daher blau gefärbt werden. Dies führt jedoch zu einem Widerspruch, da die dreifarbige Kreuzung k_1 verlangt, dass x_4 grün ist.
Daraus lässt sich schliessen, dass der Achterknoten 4_1 nicht dreifärbbar ist. Die einzigen möglichen Färbungen sind die drei trivialen Färbungen.

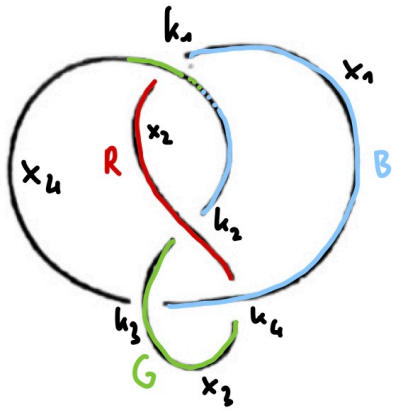


Abbildung 18: Unmögliche Dreifärbbung des Knotens 4_1.

Satz 4.4.4. Ist irgendein Diagramm eines Knotens färbbar, dann sind alle Diagramme dieses Knotens färbbar ([7], Seite 30).

Diesen Satz 4.4.4 kann man beweisen, indem man zeigt, dass die Dreifärbbarkeit unter allen Reidemeister-Bewegungen erhalten bleibt ([7], Seite 31). Dieser Beweis lässt sich am einfachsten grafisch darstellen.

Beweis. 4.4.4 Es ist wichtig zu beachten, dass die Farben jedes Mal willkürlich ausgewählt werden. Es gibt also viel mehr Möglichkeiten, wenn die Farbpermutationen berücksichtigt werden.

Reidemeister I : Es gibt nur eine Möglichkeit: Ein Bogen wird zu einer einfarbigen Kreuzung.

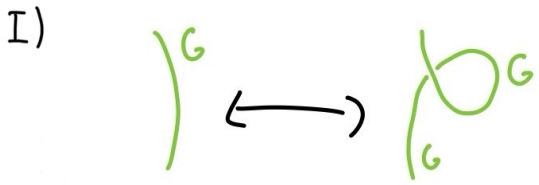


Abbildung 19: Fall 1.

Reidemeister II: Es gibt zwei Möglichkeiten:

(i) Beide Stränge haben dieselbe Farbe. Es entstehen dann zwei einfarbige Kreuzungen.

(ii) Die beide Stränge haben unterschiedliche Farben. Es entstehen zwei dreifarbige Kreuzungen.

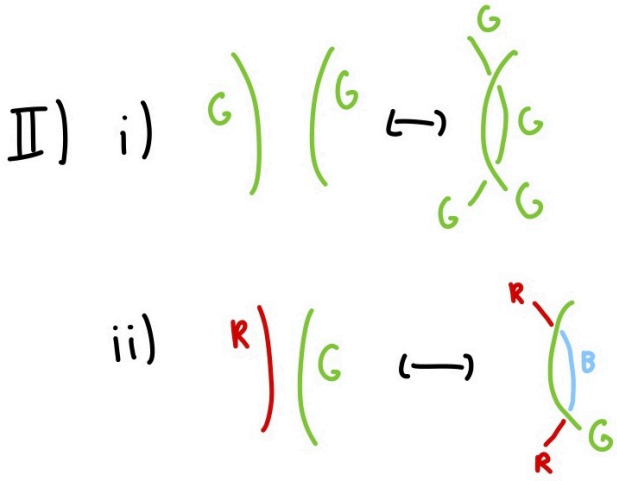


Abbildung 20: Fall 2.i (oben) und Fall 2.ii (unten).

Reidemeister III: Es gibt deutlich mehr Möglichkeiten:

(i) Alle Stränge weisen die selbe Farbe auf. Alle Kreuzungen bleiben dann einfarbig.

(ii) Alle Kreuzungen sind dreifarbig. Eine dieser Kreuzungen wird dann einfarbig.

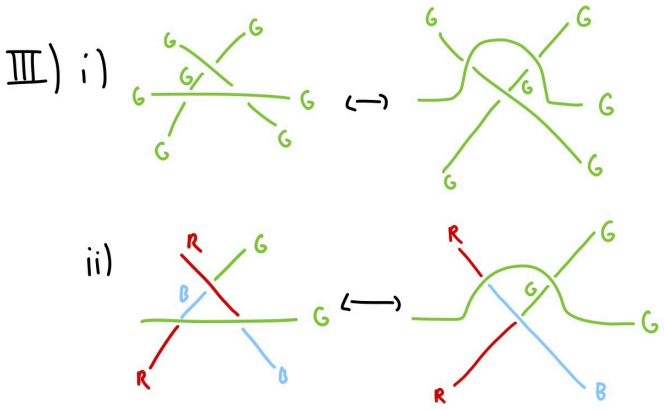


Abbildung 21: Fall 3.i (oben) und Fall 3.ii (unten).

(iii) Es gibt eine einfarbige Kreuzung. In diesem Fall gibt es wiederum drei Möglichkeiten:

(a) Die einfarbige Kreuzung bleibt einfarbig.

(b) Eine andere Kreuzung wird einfarbig.

(c) Alle Kreuzungen werden dreifarbig.

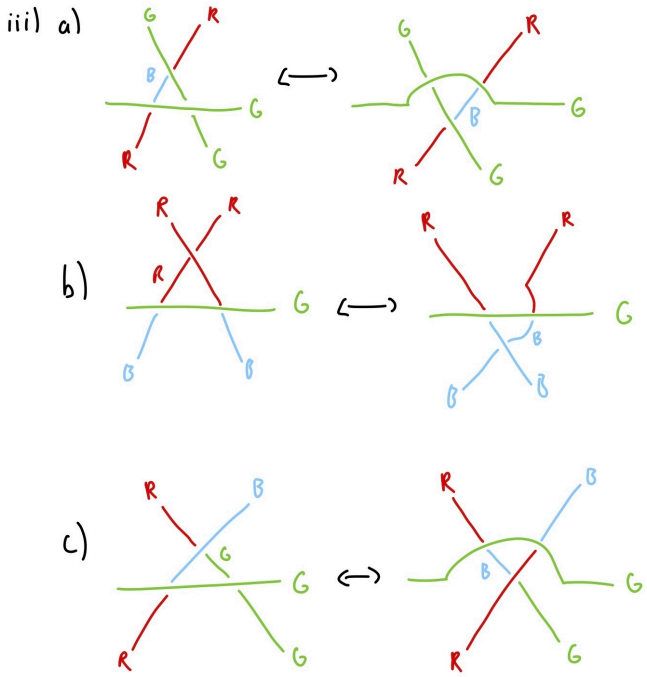


Abbildung 22: Fall 3.iii.a (oben), Fall 3.iii.b (Mitte) und Fall 3.iii.c (unten).

□

Aus dem Satz 4.4.4 wird die Färbbarkeit eines Knotens selbst definiert:

Definition 4.4.5. Ein Knoten ist *färbbar*, wenn seine Diagramme färbbar sind.

Die Dreifärbbarkeit erfüllt die Bedingungen einer Invariante: Sie ist wohl definiert, mit klaren und anwendbaren Regeln, und sie ist auch, wie gerade gezeigt, von den Reidemeister-Bewegungen unberührt.

Beispiel 4.4.6. Diese Invariante erlaubt also die Behauptung, dass es Knoten gibt, die sich vom Unknoten unterscheiden. Der Kleeblattknoten ist dreifärbbar, während der Unknoten nicht dreifärbbar ist. Es kann also sogar gesagt werden, dass alle dreifärbbaren Knoten nicht trivial sind. In der Abbildung 23 sind der Unknoten, der dreifärbbare Kleeblattknoten und der Achterknoten zu sehen. Letzterer ist nicht dreifärbbar, es kann also mit dieser Invariante nicht widerlegt werden, dass er nicht mit dem trivialen Knoten äquivalent ist.

Abbildung 23: Unknoten (links), gefärbter Kleeblattknoten (mitte) und Achterknoten (rechts).

4.5 p-Färbbarkeit

Die Dreifärbbarkeit ist ein Werkzeug, das ermöglicht Knoten eine bestimmte Eigenschaft zu verleihen. Dieses Werkzeug wäre jedoch mächtiger, wenn zum Einfärben eine andere Anzahl von Farben als drei verwendet werden könnte. Genau das leistet die p-Färbbarkeit, die somit eine Verallgemeinerung der Dreifärbbarkeit ist.
Die Dreifärbbarkeit lässt sich mit Zahlen definieren. Die drei Farben werden durch die Zahlen 0, 1 und 2 ersetzt. An einer Kreuzung hat der überkreuzende Bogen den Wert x und die zwei unterkreuzenden Bögen die Werte y und z. Die Bedingungen der Dreifärbbarkeit lassen sich also folgendermassen formulieren:

(i) Die Differenz $2 \cdot x - y - z = 0 \mod 3$.

(ii) Es werden mindestens zwei der Zahlen 0, 1 und 2 verwendet.

Die Differenz entspricht der Bedingung, dass jede Kreuzung entweder einfarbig oder dreifarbig ist. Wenn alle Farben identisch sind, ist die Differenz: $2 \cdot x - x - x = 0 \mod 3$. Wenn sich alle Farben unterscheiden, gibt es drei Fälle:

(i) $2 \cdot 2 - 1 - 0 = 3 = 0 \mod 3$

(ii) $2 \cdot 1 - 2 - 0 = 0 \mod 3$

(iii) $2 \cdot 0 - 2 - 1 = -3 = 0 \mod 3$

Die Voraussetzung mindestens zwei Zahlen zu verwenden, ist gleichbedeutend mit der Forderung mindestens zwei Farben einzusetzen.

Beispiel 4.5.1. Der Kleeblattknoten ist vertraut aus der vorherigen Betrachtung 23, bei welcher seine Dreifärbarkeit gezeigt wurde. Auf Bild 24 ist der numerische Ausdruck dieser Tatsache zu sehen.
An den Kreuzungen gilt:

$$\begin{aligned} k_1: &\quad 2 \cdot 2 - 0 - 1 = 3 = 0 \mod 3 \\ k_2: &\quad 2 \cdot 1 - 0 - 2 = 0 \mod 3 \\ k_3: &\quad 2 \cdot 0 - 2 - 1 = -3 = 0 \mod 3 \end{aligned}$$

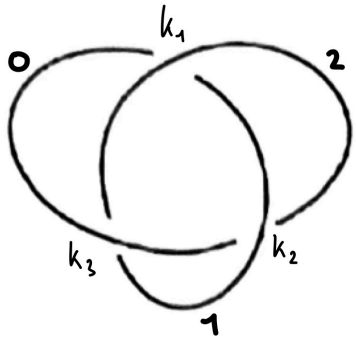


Abbildung 24: Der Kleeblattknoten 3_1 mit 0, 1 und 2 etikettiert.

Definition 4.5.2. Ein Knotendiagramm lässt sich $\mod p$ *etikettieren*, wenn jeder Bogen mit einer ganzen Zahl zwischen 0 und $p-1$ (die *Etikette*) versehen werden kann, so dass:

(i) an alle Kreuzungen die Beziehung $2x - y - z = 0 \mod p$ erfüllt ist, wobei x das Etikett der Überkreuzung ist und y und z die beiden anderen Etiketten sind.

(ii) mindestens zwei verschiedene Etiketten verwendet werden.

Satz 4.5.3. Erlaubt irgendein Diagramm eines Knotens eine Etikettierung $\mod p$, dann besitzen alle Diagramme dieses Knotens eine Etikettierung $\mod p$.

Beweis. 4.5.3 Da der Beweis des Etikettierungssatzes (4.5.3) dem der Dreifärbbarkeit (4.4.4) sehr ähnlich ist, wird nur eine Reidemeister-Bewegung explizit gezeigt. Der Rest des Beweises verläuft ähnlich.

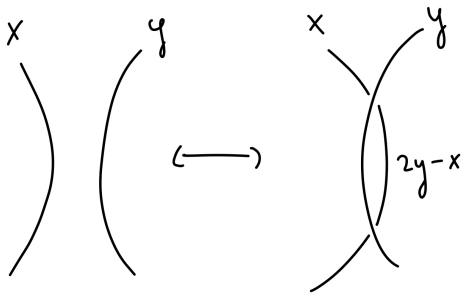


Abbildung 25: Reidemeister-Bewegung II auf einem etikettierten Diagramm.

□

Beispiel 4.5.4. Der Knoten 4_1 lässt sich $\mod 5$ etikettieren (siehe 26) aber nicht $\mod 7$. Der Knoten 7_1 kann im Gegenteil $\mod 7$ etikettiert (siehe 27) werden aber nicht $\mod 5$. Daraus kann also geschlossen werden, dass diese beiden Knoten nicht äquivalent sind.
An den Kreuzungen des Knotens 4_1 gilt:

$$
\begin{aligned}
k_1: &\quad 2\cdot 4-0-3=5=0 \mod 5\\
k_2: &\quad 2\cdot 2-4-0=0 \mod 5\\
k_3: &\quad 2\cdot 3-4-2=0 \mod 5\\
k_4: &\quad 2\cdot 0-2-3=-5=0 \mod 5
\end{aligned}
$$

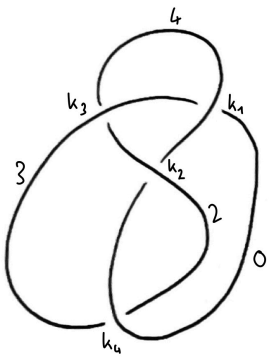


Abbildung 26: Etikettierung $\mod 5$ des Knotens 4_1.

An den Kreuzungen des Knotens 7_1 gilt:

$$
\begin{aligned}
k_1: &\quad 2\cdot 3-1-5=0 \mod 7\\
k_2: &\quad 2\cdot 1-6-3=-7=0 \mod 7\\
k_3: &\quad 2\cdot 6-1-4=7=0 \mod 7\\
k_4: &\quad 2\cdot 4-6-2=0 \mod 7\\
k_5: &\quad 2\cdot 2-0-4=0 \mod 7\\
k_6: &\quad 2\cdot 0-5-2=-7=0 \mod 7\\
k_7: &\quad 2\cdot 5-0-3=7=0 \mod 7
\end{aligned}
$$

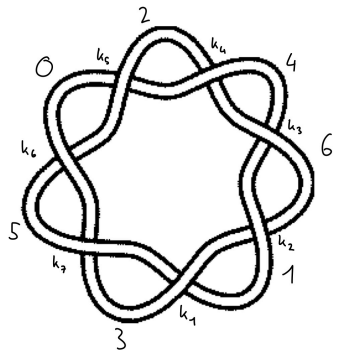


Abbildung 27: Etikettierung $\mod 7$ des Knotens 7_1.

Bemerkung 4.5.5. Eine andere Art auszudrücken, dass einen Knoten $\mod p$ etikettiert werden kann, ist zu sagen, er sei *p-färbbar*.

Bemerkung 4.5.6. Es wird $\mod p$ nur mit $p \in \mathbb{P}$ etikettiert. Wenn n, eine Zahl, die nicht prim ist oder eine zusammengesetzte Zahl verwendet wird, um einen Knoten zu etikettieren, werden keine neuen Informationen erhalten. Daraus resultiert, dass ein Knoten, der $\mod 5$ etikettiert werden kann, und ein Knoten, der $\mod 3$ etikettiert werden kann, dann beide $\mod 15$ etikettiert werden können. Zudem kann 2, die einzige gerade Primzahl, nicht zur Etikettierung von Knoten verwendet werden, da die einzige mögliche Färbung in diesem Fall die triviale Färbung ist [14].

Durch die p-Färbbarkeit lässt sich ein Zusammenhang zwischen der linearen Algebra und den Knoteninvarianten herstellen. Matrizen erlauben eine systematische Lösung des p-Färbbarkeitsproblems und ersetzen dadurch das Vorgehen nach dem Prinzip Versuch und Irrtum.
Die Algebraisierung der p-Färbbarkeit erfolgt folgendermassen: Jeder Bogen eines Diagrammes wird mit der Variable x_i etikettiert. Angenommen der Bogen x_i überkreuzt die Bögen x_j und x_k soll an jeder Kreuzung die folgende Beziehung zwischen den Variablen gelten: $2 \cdot x_i - x_j - x_k = 0 \mod p$. Mit anderen Worten, an jeder Kreuzung wird dem oberführenden Bogen der Wert 2 zugewiesen und den beiden unterführenden Bögen der Wert -1 zugewiesen. Wenn derselbe Bogen zweimal in einer Kreuzung vorkommt, beispielsweise an einer Schleife, werden die Werte addiert. Ein Knotendiagramm ist p-färbbar, wenn alle erhaltenen Gleichungen eine Lösung $\mod p$ besitzen. Der Fall, dass alle x_i gleich sind, wird ausgeschlossen, da er einer trivialen Färbung entspricht. Das Gleichungssystem wird in Form einer Matrix dargestellt.
Herauszufinden ob ein Knoten p-etikettierbar ist, ist daher nun eine Frage der linearen Algebra.

Beispiel 4.5.7. Auf der Abbildung 28 ist der Knoten 5_2 mit nummerierten Kreuzungen und mit den Variablen etikettierten Bögen dargestellt. Die Matrix unten entspricht dem Knotendiagramm, indem die Kreuzungen den Zeilen und die Variablen den Spalten zugeordnet sind. Die $(x_1, ..., x_5)$ entsprechen den Etikettierungen der Bögen. Das Ergebnis muss $0 \mod p$ sein.

$$\begin{bmatrix} 2 & 0 & 0 & -1 & -1 \\ -1 & -1 & 0 & 2 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & 0 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \mod p$$

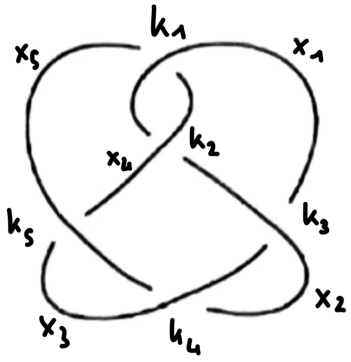


Abbildung 28: Knoten 5_2 mit den Variablen etikettierten Bögen.

Die Lösung $x_i = 1$ für alle i erfüllt das Gleichungssystem, ist jedoch gemäss der Definition 4.5.2 ungültig. Ausserdem ist die Summe zweier Lösungen wieder eine Lösung. Wenn es eine Lösung gibt, bei der nicht

alle x_i gleich sind, dann gibt es - aufgrund der beiden Feststellungen - auch eine Lösung, mit $x_n = 0$ (4.5.8). Das gewählte x_i ist beliebig.

Lemma 4.5.8. Sei $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}$ eine Lösung des Gleichungssystems, also sei $A \cdot \vec{x} = \vec{0}$ mit der Ausgangsmatrix A (siehe 4.5.7), dann gibt es eine Lösung $\vec{x'} = \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_{n-1} \\ 0 \end{pmatrix}$.

Beweis. 4.5.8 Sei $\vec{x}$ eine Lösung und $\begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix}$ eine Lösung (4.5.7), dann ist $\vec{x} - \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix}$ auch eine Lösung.

$$A(\vec{x} - \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix}) = A \cdot \vec{x} - A \cdot \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} = \vec{0} \Leftrightarrow \vec{x'} = \vec{x} - \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix}$$

□

Deshalb ist $\vec{x'}$ auch eine Lösung des Gleichungssystems. Eine *nichttriviale Lösung* ist also eine Lösung der ursprünglichen Matrix ohne die letzte Spalte des Gleichungssystems.

Satz 4.5.9. Die p-Färbbarkeit für ein Knotendiagramm mit n Bögen führt zu einem linearen Gleichungssystem, das mit einer $n \times n$-Matrix ausgedrückt sein kann. Diese Matrix wird nun betrachtet, wobei eine beliebige Zeile und eine beliebige Spalte gestrichen wurden. Der Knoten kann $\mod p$ etikettiert werden, wenn das Gleichungssystem, das dieser $(n-1) \times (n-1)$-Matrix entspricht, eine nichttriviale Lösung $\mod p$ besitzt.

Die Frage der p-Färbbarkeit fällt nun in den Bereich der linearen Algebra. Eine solche Lösung existiert, wenn die Determinante der $(n-1) \times (n-1)$-Matrix gleich 0, bzw. gleich $0 \mod p$ ist.

Definition 4.5.10. Der Betrag der Determinante der oben konstruierten $(n-1) \times (n-1)$-Matrix nennt man *Determinante* des Knotens.

Ein Knoten ist somit für jede Primzahl p, die ein Teiler seiner Determinante ist, p-färbbar.

Beispiel 4.5.11. Es ist nun möglich das Beispiel 4.5.7 zu beenden und die p-Färbbarkeit des Knotens 5_2 mit der linearen Algebra zu berechnen.

$$A = \begin{bmatrix} 2 & 0 & 0 & -1 & -1 \\ -1 & -1 & 0 & 2 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & 0 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$

$$A_{5,5} = \begin{bmatrix} 2 & 0 & 0 & -1 \\ -1 & -1 & 0 & 2 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & 0 \end{bmatrix}$$

A ist die $n \times n$-Matrix, die aus der Etikettierung des Knotens gemäss 4.5 erstellt wurde. $A_{5,5}$ ist die $(n-1) \times (n-1)$-Matrix mit einer gestrichenen Zeile und Spalte, hier willkürlich $i = 5$ und $j = 5$.

$$\det A_{5,5} = 7$$

Die Determinante des Knotens ist 7. Der einzige Primteiler der Determinante ist offensichtlich 7. Ohne überhaupt nach Etikettierung zu suchen, ergibt sich, dass der Knoten 5_2 7-färbbar und nur 7-färbbar ist.

Satz 4.5.12. Die Determinante eines Knotens hängt nicht von der Wahl des Diagramms und der Etikettierung ab.

Es wird zunächst bewiesen, dass der Betrag dieser Determinante nicht davon abhängt, welche Zeile und Spalte gestrichen werden, vorausgesetzt, die Summe der Zeilen und Spalten dieser quadratischen Matrix ist gleich null.
Der erste Schritt besteht in der Sicherstellung, dass die Summe der Zeilen und Spalten einer solchen Knoten-Matrix null sein muss.
Zuerst werden die Zeilen angeschaut: Entweder gibt es eine “klassische” Kreuzung mit drei verschiedenen Bögen, von denen einer überkreuzend ist. Die Einträge in dieser Zeile sind 2,-1 und -1. Die Summe ist also null. Ansonsten gibt es im Diagramm eine Schleife mit einem Bogen, der zugleich überkreuzend und unterkreuzend ist. Die Einträge sind $(2 + (-1))$ und -1, die Summe ist wieder null.
Jetzt werden die Spalten betrachtet: Jeder Bogen beginnt und endet an einer Unterkreuzung. Zwischen diesen beiden Kreuzungen kann es keine, eine oder mehrere Oberkreuzungen geben. Die Spaltensumme ist daher nicht immer gleich null. Glücklicherweise gibt es eine Möglichkeit dieses Problem zu beheben. Um die Matrix in eine Form zu bringen, bei der die Spaltensummen null sind, werden einige Zeilen der Matrix mit -1 multipliziert, wodurch sich einige Zeilen zu null addieren.

Bemerkung 4.5.13. Die Auswahl der Zeilen, die mit -1 multipliziert werden sollen, erfolgt folgendermassen:
Der Knoten wird orientiert. Vor jeder Kreuzung wird ein Punkt rechts neben dem überkreuzenden Bogen gezeichnet. Jeder dieser Punkte wird dann mit einem Punkt weit im Unendlichen verbunden. Wenn die Anzahl der Bögen, die er kreuzt, ungerade ist, muss die Zeile, die dieser Kreuzung in der Matrix entspricht,

ε	$-\varepsilon$	ε	$-\varepsilon$
$-\varepsilon$	ε	$-\varepsilon$	ε
ε	$-\varepsilon$	ε	$-\varepsilon$

Abbildung 29: “Schachbrettmuster” der ε-Zeichen.

mit -1 multipliziert werden. Dieses Verfahren führt zu einem “Schachbrettmuster” der Zeichen. Dieses Muster ist auf der Abbildung 29 zu sehen. Das ε-Zeichen steht hier für das Zeichen einer Region und nicht für den Eintrag selbst.

Satz 4.5.14. Wenn die Vorzeichen der Bögen eines Knotens dem “Schachbrettmuster” entsprechen und jede Kreuzung mit dem ε-Vorzeichen der Region vor der Oberkreuzung rechts multipliziert wird, ist die Spaltensumme gleich Null.

Beweis. 4.5.14 Um dies vollständig zu beweisen, müssten alle möglichen Fälle untersucht werden. Anhand der folgenden vier Hauptfälle lässt sich das Prinzip jedoch nachvollziehen. Die übrigen Fälle lassen sich analog untersuchen.
Ein Bogen, der über n anderen Bögen verläuft, hat Enden, die durch zwei überkreuzende Bögen markiert sind. Diese können in die gleiche Richtung oder in die entgegengesetzte Richtung verlaufen.
Der rote Punkt auf den Bildern zeigt die Regionen an, deren ε-Zeichen die Einträge in die Matrix multiplizieren werden.

(i) Fall 1 mit $n = 0$ gerade und beiden überkreuzenden Bögen, die in die gleiche Richtung verlaufen.

$$\varepsilon \cdot (-1) + (-\varepsilon) \cdot (-1) = -\varepsilon + \varepsilon = 0$$

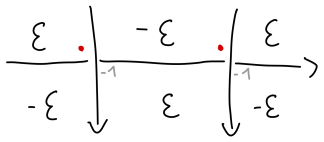


Abbildung 30: Fall 1 mit $n = 0$ gerade und überkreuzende Bögen in die gleiche Richtung.

(ii) Fall 2 mit $n = 0$ gerade und beide überkreuzenden Bögen, die in entgegengesetzte Richtung verlaufen.

$$\varepsilon \cdot (-1) + (-\varepsilon) \cdot (-1) = -\varepsilon + \varepsilon = 0$$

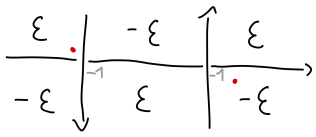


Abbildung 31: Fall 2 mit $n = 0$ gerade und überkreuzende Bögen in die entgegengesetzte Richtung.

(iii) Fall 3 mit $n = 1$ ungerade und beide überkreuzende Bögen, die in die gleiche Richtung verlaufen.

$$\varepsilon \cdot (-1) + \varepsilon \cdot (2) + \varepsilon \cdot (-1) = -\varepsilon + 2\varepsilon - \varepsilon = 0$$

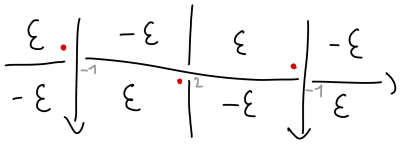


Abbildung 32: Fall 3 mit $n = 1$ ungerade und überkreuzende Bögen in die gleiche Richtung.

(iv) Fall 4 mit $n = 3$ ungerade und beide überkreuzende Bögen, die in die entgegengesetzte Richtung verlaufen.

$$\varepsilon \cdot (-1) + \varepsilon \cdot 2 + (-\varepsilon) \cdot 2 + \varepsilon \cdot 2 + \varepsilon \cdot (-1) = -\varepsilon + 2\varepsilon - 2\varepsilon + 2\varepsilon - \varepsilon = 0$$

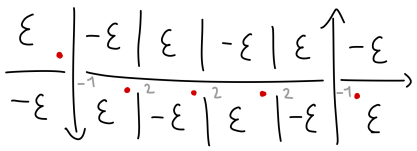


Abbildung 33: Fall 4 mit $n = 3$ ungerade und überkreuzende Bögen in die entgegengesetzte Richtung.

□

Lemma 4.5.15. Sei A eine $n \times n$ Matrix mit einer Zeilen- und einer Spaltensumme, die beide gleich null sind. Wenn eine Zeile und eine Spalte dieser Matrix gestrichen werden, ist die Determinante der resultierenden Matrix unabhängig von der Wahl der Spalte und der Zeile.

Beweis. 4.5.15

Sei B eine $n \times n$ Matrix mit allen Einträgen gleich 1.

Gegeben: Matrix A mit Zeilen- und Spaltensumme gleich null

$$\sum_{i=1}^{n} a_{ki} = 0 \quad \forall k \in \{1, 2, ..., n-1, n\} \tag{4.1}$$

$$\sum_{k=1}^{n} a_{ki} = 0 \quad \forall i \in \{1, 2, ..., n-1, n\} \tag{4.2}$$

und Matrix B mit $b_{ij} = 1 \quad \forall i, j \in \{1, 2, ..., n-1, n\}$.

A und B werden addiert. Die Determinante dieser neuen Matrix wird berechnet.

$$det(A+B) = det \begin{bmatrix} a_{1,1}+1 & a_{1,2}+1 & \cdots & a_{1,n}+1 \\ \vdots & \vdots & & \vdots \\ a_{i,1}+1 & a_{i,2}+1 & \cdots & a_{i,n}+1 \\ \vdots & \vdots & & \vdots \\ a_{n,1}+1 & a_{n,2}+1 & \cdots & a_{n,n}+1 \end{bmatrix}$$

Bevor die Determinante berechnet wird, muss die Matrix wie folgt geändert werden:

(i) Alle anderen Zeilen von $A+B$ werden zur i-ten Zeile addiert. Da die Summe der Spalten null ist (4.1), sind alle Einträge der i-ten Zeile gleich n und der Rest der Matrix bleibt unverändert.

$$= det \begin{bmatrix} a_{1,1}+1 & \cdots & a_{1,j}+1 & \cdots & a_{1,n}+1 \\ \vdots & & \vdots & & \vdots \\ n & \cdots & n & \cdots & n \\ \vdots & & \vdots & & \vdots \\ a_{n,1}+1 & \cdots & a_{n,j}+1 & \cdots & a_{n,n}+1 \end{bmatrix}$$

(ii) Alle anderen Spalten von $A+B$ werden zur j-ten Spalte addiert. Da die Summe der Zeilen null ergibt (4.1), ist der Eintrage (i,j) gleich n^2, alle andere Einträge der j-ten Spalte gleich n und der Rest der Matrix unverändert.

$$= det \begin{bmatrix} a_{1,1}+1 & \cdots & n & \cdots & a_{1,n}+1 \\ \vdots & & \vdots & & \vdots \\ n & \cdots & n^2 & \cdots & n \\ \vdots & & \vdots & & \vdots \\ a_{n,1}+1 & \cdots & n & \cdots & a_{n,n}+1 \end{bmatrix}$$

(iii) Ein Faktor n wird von der i-ten Zeile ausgeklammert.

$$= n \cdot det \begin{bmatrix} a_{1,1}+1 & \cdots & n & \cdots & a_{1,n}+1 \\ \vdots & & \vdots & & \vdots \\ 1 & \cdots & n & \cdots & 1 \\ \vdots & & \vdots & & \vdots \\ a_{n,1}+1 & \cdots & n & \cdots & a_{n,n}+1 \end{bmatrix}$$

(iv) Die i-te Zeile wird von den anderen Zeilen subtrahiert. Dabei wird der Eintrag (i, j) gleich n sein, die anderen Einträge der j-ten Spalte sind gleich null und der Rest der Matrix ist gleich die Matrix A.

$$= n \cdot det \begin{bmatrix} a_{1,1} & \cdots & a_{1,j-1} & 0 & a_{1,j+1} & \cdots & a_{1,n} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{i-1,1} & \cdots & a_{i-1,j-1} & 0 & a_{i-1,j+1} & \cdots & a_{i-1,n} \\ 1 & \cdots & 1 & n & 1 & \cdots & 1 \\ a_{i+1,1} & \cdots & a_{i+1,j-1} & 0 & a_{i+1,j+1} & \cdots & a_{i+1,n} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,j-1} & 0 & a_{n,j+1} & \cdots & a_{n,n} \end{bmatrix}$$

Jetzt wird die j-te Spalte entwickelt. Somit ist $det(A + B) = n^2 \cdot det(A_{i,j})$, wobei $A_{i,j}$ der Matrix A mit der i-ten Zeile und der j-ten Spalte gestrichen entspricht.

$$det(A_{ij}) = \frac{1}{n^2} \, det(A + \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix})$$

Die rechte Seite dieser Gleichung ist unabhängig von der Wahl der i-ten Zeile und der j-ten Spalte, daher ist auch die linke Seite der Gleichung, also die Determinante der Matrix A_{ij}, unabhängig von diesen beiden Wahlen [2].

□

Jetzt kann gezeigt werden, dass die Ausführung von Reidemeister-Bewegungen die Determinante der $(n-1) \times (n-1)$-Matrix A nicht beeinflusst.

Beweis. Hier ist die Beweisidee skizziert: Es wird gezeigt, dass $\det A$ unter allen drei Reidemeister-Bewegungen invariant bleibt, indem die Matrixoperationen explizit durchgeführt werden. Jede Bewegung wird o.B.d.A nur in eine Richtung analysiert, das heisst ohne ihre Umkehrung.

- **Reidemeister I:** Die Reidemeister-Bewegung vom Typ I fügt eine neue Kreuzung (Zeile) und einen neuen Bogen (Spalte) hinzu (siehe 34).

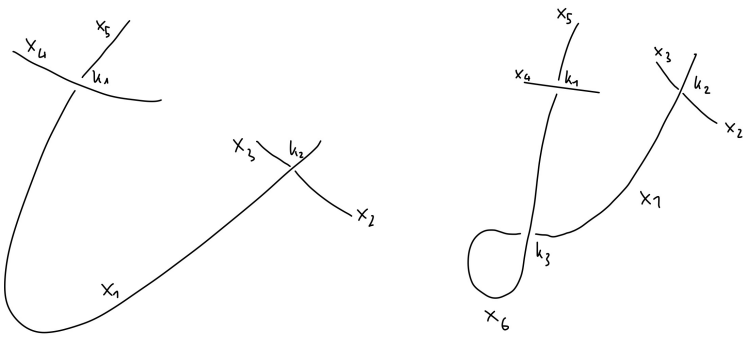


Abbildung 34: Ausschnitt aus einem Knoten vor (links) und nach (rechts) einer Reidemeister-Bewegung I.

Teil der Ausgangsmatrix A:

$$A = \begin{bmatrix} \ddots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & -1 & 0 & 0 & 2 & -1 \\ \cdots & \cdots & 2 & -1 & -1 & 0 & 0 \end{bmatrix}$$

Teil der Matrix A' nach der Ausführung von Reidemeister I:

$$A' = \begin{array}{c} \\ \\ \\ I \\ II \\ III \end{array} \begin{array}{c} \begin{array}{ccccccc} & & i & ii & iii & iv & v & vi \end{array} \\ \begin{bmatrix} \ddots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & 0 & 0 & 0 & 2 & -1 & -1 \\ \cdots & \cdots & 2 & -1 & -1 & 0 & 0 & 0 \\ \cdots & \cdots & -1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Die folgenden Umformungen werden durchgeführt: $(I) + (III)$ und $(vi) + (i)$. Dies liefert:

$$A' = \begin{array}{c} \\ \\ \\ I \\ II \\ III \end{array} \begin{array}{c} \begin{array}{ccccccc} & & i & ii & iii & iv & v & vi \end{array} \\ \begin{bmatrix} \ddots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & -1 & 0 & 0 & 2 & -1 & 0 \\ \cdots & \cdots & 2 & -1 & -1 & 0 & 0 & 0 \\ \cdots & \cdots & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Bei der Determinantenberechnung durch Entwicklung nach Zeile III hat der Faktor 1 in der Diagonale keinen Einfluss auf die Determinante, während der verbleibende Teil der Matrix der ursprünglichen Matrix entspricht. Daher gilt: $\det A' = \det A$.

- **Reidemeister II** Die Reidemeister-Bewegung vom Typ II fügt zwei neue Kreuzungen (Zeilen) und zwei neue Bögen (Spalten) ein (siehe 35).

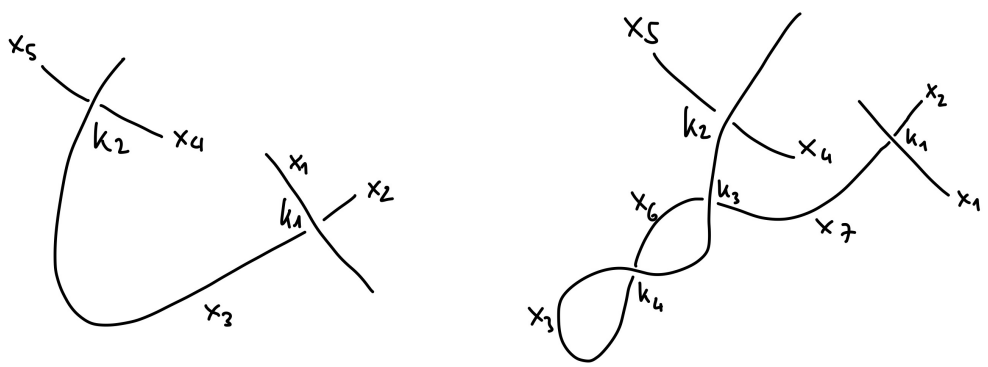


Abbildung 35: Ausschnitt aus einem Knoten vor (links) und nach (rechts) einer Reidemeister-Bewegung II.

Teil der Ausgangsmatrix A:

$$A = \begin{bmatrix} \ddots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & 2 & -1 & -1 & 0 & 0 \\ \cdots & \cdots & 0 & 0 & 2 & -1 & -1 \end{bmatrix}$$

Teil der Matrix A' nach der Ausführung von Reidemeister II:

$$A' = \begin{array}{c} \\ \\ I \\ II \\ III \\ IV \end{array} \begin{array}{c} \begin{array}{ccccccc} i & ii & iii & iv & v & vi & vii \end{array} \\ \begin{bmatrix} \ddots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & 2 & 1 & 0 & 0 & 0 & 0 & -1 \\ \cdots & \cdots & 0 & 0 & 2 & -1 & -1 & 0 & 0 \\ \cdots & \cdots & 0 & 0 & 2 & 0 & 0 & -1 & -1 \\ \cdots & \cdots & 0 & 0 & 1 & 0 & 0 & -1 & 0 \end{bmatrix} \end{array}$$

Die folgenden Umformungen werden durchgeführt: $(I) - (IV)$, $(vi) + (vii)$, $(III) - 2 \cdot (IV)$, $(iii) + (vi)$, $(III) \cdot (-1)$, $(IV) \cdot (-1)$, $(I) + (III)$ und $(III) \Leftrightarrow (IV)$. Der Teil der Matrix sieht so aus:

$$A' = \begin{array}{c} \\ \\ I \\ II \\ III \\ IV \end{array} \begin{array}{c} \begin{array}{ccccccc} i & ii & iii & iv & v & vi & vii \end{array} \\ \begin{bmatrix} \ddots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ \cdots & \cdots & 0 & 0 & 2 & -1 & -1 & 0 & 0 \\ \cdots & \cdots & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \cdots & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Die Zeilen III und IV enthalten beide nur eine 1 in der Diagonale. Der Rest der Teilmatrix ist derselbe wie vor der Reidemeister-Bewegung II. Bei der Berechnung der Determinante erfolgt zunächst eine Entwicklung nach der IV-ten Zeile und anschliessend nach der III-ten Zeile. Daher ist $\det A' = \det A$.

- **Reidemeister III**

  Da keine neuen Kreuzungen oder Bögen entstehen, entspricht die Bewegung einem Austausch von Zeilen und Spalten (siehe 36).

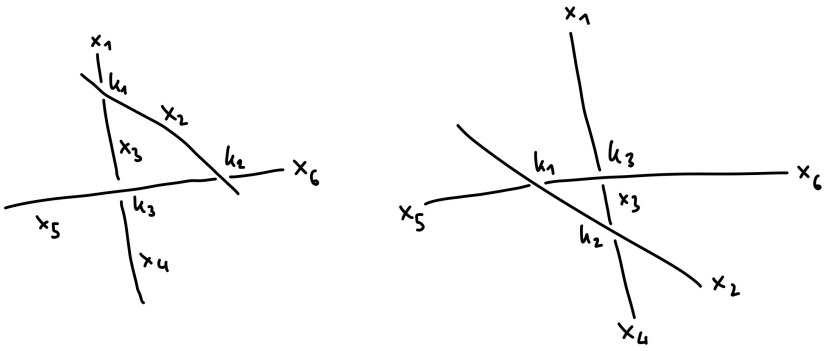


Abbildung 36: Ausschnitt aus einem Knoten vor (links) und nach (rechts) einer Reidemeister-Bewegung III.

Teil der Ausgangsmatrix A:

$$A = \begin{bmatrix} \ddots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & -1 & 2 & -1 & 0 & 0 & 0 \\ \cdots & \cdots & 0 & 2 & 0 & 0 & -1 & -1 \\ \cdots & \cdots & 0 & 0 & -1 & -1 & 2 & 0 \end{bmatrix}$$

Teil der Matrix A' nach der Ausführung von Reidemeister III:

$$A' = \begin{array}{c} \\ \\ I \\ II \\ III \end{array} \begin{array}{c} \begin{array}{cccccccc} & & i & ii & iii & iv & v & vi \end{array} \\ \begin{bmatrix} \ddots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & 0 & 2 & 0 & 0 & -1 & -1 \\ \cdots & \cdots & 0 & 2 & -1 & -1 & 0 & 0 \\ \cdots & \cdots & -1 & 0 & -1 & 0 & 0 & 2 \end{bmatrix} \end{array}$$

Folgende Umformungen werden ausgeführt: $(I) \Leftrightarrow (III)$ und $(i) \Leftrightarrow (iv)$. Der Matrixteil sieht so aus:

$$A' = \begin{array}{c} \\ \\ I \\ II \\ III \end{array} \begin{array}{c} \begin{array}{cccccccc} & & i & ii & iii & iv & v & vi \end{array} \\ \begin{bmatrix} \ddots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & -1 & 2 & -1 & 0 & 0 & 0 \\ \cdots & \cdots & 0 & 2 & 0 & 0 & -1 & -1 \\ \cdots & \cdots & 0 & 0 & -1 & -1 & 2 & 0 \end{bmatrix} \end{array}$$

Dieser Teil der Matrix ist identisch mit dem Teil vor der Reidemeister-Bewegung III. Also ist

$$A' = A \Rightarrow \det A' = \det A$$

□

4.6 Anzahl der p-Färbbarkeiten

Die Anzahl der p-Färbbarkeiten ist ebenfalls eine Invariante. Wenn ein Knoten für eine Primzahl p etikettierbar ist, gibt es für alle Diagramme dieses Knotens eine feste Anzahl an Arten diesen Knoten einzufärben.

Definition 4.6.1. Der Rangdefekt $\mod p$ der $(n-1) \times (n-1)$-Matrix, wie in 4.5.9 konstruiert wurde, heisst *Rang mod p* des Knotens.

In der linearen Algebra ist der Rangdefekt gleich der Anzahl Zeilen der Matrix, die linear abhängig sind.

Satz 4.6.2. Sei $(a_1, ..., a_{n-1})$ die Matrix A, ist die Anzahl der p-Färbbarkeiten, also die Anzahl der Elemente in

$$\{\vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} \mid (a_1, ..., a_{n-1}) \cdot \vec{y} = \vec{0}, \vec{y} \neq \vec{0}\}$$

gleich

$$p \cdot (p^{n-1-\text{rang}(a_1,...,a_{n-1})} - 1).$$

Satz 4.6.3. Sei $\varphi : V \to W$ eine lineare Abbildung. Dann gilt

$$\dim V = \dim(\text{Bild}(\varphi)) + \dim(\text{Kern}(\varphi)).$$

Dies kann auch so ausgedrückt werden:

$$\dim V = \text{Rang } \varphi + \text{Defekt } \varphi.$$

([6], Seite 123)

Definition 4.6.4. Kern $\varphi = \{v \in V \mid \varphi(v) = 0\}$ ([6], Seite 123)

Beweis. 4.6.2:

Ausgangspunkt: $A \cdot \vec{y} = \vec{0}$. Hierbei ist $\vec{y} = \begin{pmatrix} y_1 \\ ... \\ y_{n-1} \end{pmatrix}$.

A ist die Matrix mit einer gestrichenen Zeile und Spalte.

Sei die lineare Abbildung $\varphi : V \to W$ durch die p-Färbbarkeitsmatrix $A = (a_1, ..., a_{n-1})$ dargestellt, dann ist $\dim V = n - 1$. Der Rangdefekt dieser Abbildung, also der Kern, ist Kern $A = \{\vec{y} \mid A \cdot \vec{y} = \vec{0}\}$. Es sei daran erinnert, dass man die Anzahl der Kombinationen von Elementen in der Matrix A erhalten möchte, die zu einem Ergebnis von Null führen. Dieser Rangdefekt ist aufgrund von 4.6.3 gleich der

$\dim V - \dim Bild(\varphi)$. $\Rightarrow$ Defekt$(A) = n - 1 -$ Rang$A$.
Da bei $a_i \in \{0, ..., p-1\}$, jedes Mal p Elemente pro Eintrag möglich sind, gibt es pro Dimension p Möglichkeiten.
Der Kern A hat dann $p^{n-1-\text{Rang}A}$ Elemente. Der Term $n-1-$Rang$A$ ist gleich dem Rangdefekt oder dem *Rang mod p* des Knotens.
Es soll nicht vergessen werden, dass die triviale Lösung nicht erwünscht ist ($\vec{y} \neq \vec{0}$). Deshalb wird 1 vom vorherigen Ergebnis abgezogen, was $p^{n-1-\text{rang}A} - 1$ Elemente ergibt.
Schliesslich war bisher $\vec{y} = \begin{pmatrix} y_1 \\ ... \\ y_{n-1} \end{pmatrix}$, also wurde einer (der letzte) Bogen fix $\vec{0}$ gehalten. Für den letzten Eintrag gibt es nun auch noch p Möglichkeiten, womit die Anzahl der p-Färbbarkeiten gegeben ist durch die Zahl:

$$p \cdot (p^{n-1-\text{rang}A} - 1).$$

□

Interessanterweise wurde gerade ein Matrizenproblem mithilfe von linearen Abbildungen gelöst.

Beispiel 4.6.5. Das Beispiel 4.5.11 wird übernommen und die Anzahl der 7-Färbbarkeiten berechnet.

$$A = \begin{bmatrix} 2 & 0 & 0 & -1 \\ -1 & -1 & 0 & 2 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & 0 \end{bmatrix}$$

A ist die $(n-1) \times (n-1)$-Matrix. Da ihre Dimensionen 4×4 sind, ist $n = 5$. Als letztes muss also der Rang $\mod 7$ des Knotens ermittelt werden.
Der Rang von A modulo 7 wird mittels Gauss-Elimination berechnet.
Zunächst werden die negativen Einträge konvertiert ($-1 \equiv 6 \pmod 7$):

$$A \equiv \begin{bmatrix} 2 & 0 & 0 & 6 \\ 6 & 6 & 0 & 2 \\ 6 & 2 & 6 & 0 \\ 0 & 6 & 2 & 0 \end{bmatrix} \pmod 7$$

Die Matrix wird in die Zeilenstufenform umgewandelt:

- $4 \cdot (I)$ weil $2 \cdot 4 \equiv 1 \pmod 7$:

$$\begin{matrix} I \\ II \\ III \\ IV \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 3 \\ 6 & 6 & 0 & 2 \\ 6 & 2 & 6 & 0 \\ 0 & 6 & 2 & 0 \end{bmatrix}$$

- $(II) - 6 \cdot (I)$ und $(III) - 6 \cdot (I)$:

$$\begin{matrix} I \\ II \\ III \\ IV \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 6 & 0 & 5 \\ 0 & 2 & 6 & 3 \\ 0 & 6 & 2 & 0 \end{bmatrix}$$

- $6 \cdot (II)$ weil $6 \cdot 6 \equiv 1 \pmod 7$:

$$\begin{matrix} I \\ II \\ III \\ IV \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 2 & 6 & 3 \\ 0 & 6 & 2 & 0 \end{bmatrix}$$

- $(III) - 2 \cdot (II)$ und $(IV) - 6 \cdot (II)$:

$$\begin{matrix} I \\ II \\ III \\ IV \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 6 & 6 \\ 0 & 0 & 2 & 2 \end{bmatrix}$$

- $6 \cdot (III)$ weil $6 \cdot 6 \equiv 1 \pmod 7$:

$$\begin{matrix} I \\ II \\ III \\ IV \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 2 \end{bmatrix}$$

- $(IV) - 2 \cdot (III)$:

$$\begin{matrix} I \\ II \\ III \\ IV \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Die Matrix besitzt drei von Null verschiedene Zeilen in Zeilenstufenform. Diese drei Zeilen sind linear unabhängig. Der Rang der Matrix $A \mod 7$ ist also gleich 3. Der Rangdefekt oder Rang $\mod 7$ des Knotens ist dabei gleich 1.
Die Anzahl der 7-Färbbarkeiten des Knotens 5_2 ist somit gleich:

$$7 \cdot (7^{5-1-3} - 1) = 7 \cdot 6 = 42.$$

4.7 Alexander-Polynom

Nach mehreren numerischen Invarianten wird nun eine polynomiale Invariante eingeführt. Dieser gesamte Abschnitt basiert, sofern nicht anders angegeben, auf den Seiten 44 bis 48 des Buches “Knotentheorie für Einsteiger” von Livingston [7].
Es wird ein orientiertes Knotendiagramm gewählt. Die Kreuzungen k_i und Bögen x_i werden mit $i \in \{1, ..., n\}$ nummeriert. Eine $n \times n$-Matrix, in der jede Kreuzung eine Zeile hat, und jeder Bogen eine Spalte hat, wird wie folgt aufgestellt:

Bei rechtshändigen Kreuzungen, wie im Bild 37 links gezeigt, bekommt der überkreuzende Bogen den Eintrag $1-t$. Der ankommende Bogen bekommt den Eintrag -1 und der abgehende den Eintrag t. Bei linkshändigen Kreuzungen, wie im Bild 37 rechts gezeigt, bekommt der überkreuzende Bogen den Eintrag $1-t$. Der ankommende Bogen bekommt den Eintrag t und der abgehende den Eintrag -1.
Wenn ein Bogen sich selbst schneidet, werden seine verschiedenen Einträge addiert.

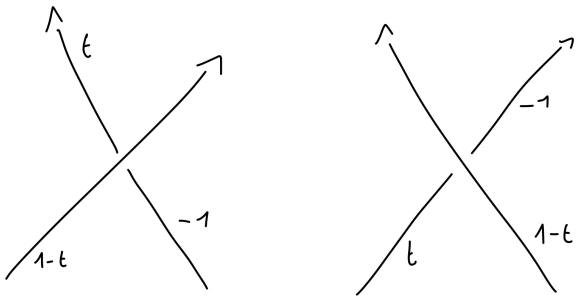


Abbildung 37: rechtshändige Kreuzung (links) und linkshändige Kreuzung (rechts).

Definition 4.7.1. Die $(n-1)\times(n-1)$- *Alexander-Matrix* des Knotens K ist die gerade aufgestellte Matrix, in der eine beliebige Zeile und Spalte gestrichen wurden. Die Determinante der Alexander-Matrix heisst *Alexander-Polynom* $A_K(t)$ des Knotens.

Satz 4.7.2. Bei der Berechnung des Alexander-Polynoms aus zwei verschiedenen Diagrammen oder Beschriftungen eines Knotens unterscheiden sich die Polynome um den Faktor $\pm t^k$, wobei k eine ganze Zahl ist.

Der Beweis von der Invarianzeigenschaft vom Alexander-Polynom (4.7.2) ähnelt dem des Satzes 4.5.12, ist jedoch komplexer. Er wird in dieser Arbeit nicht näher behandelt.

Beispiel 4.7.3. Der Knoten 5_2 wird wie im Bild 38 orientiert und nummeriert.

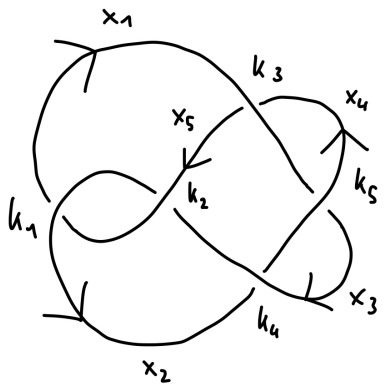


Abbildung 38: Orientierter und durchnummerierter Knoten 5_2.

Die zugehörige $n \times n$-Matrix lautet:

$$\begin{bmatrix} -1 & 1-t & 0 & 0 & t \\ 0 & -1 & t & 0 & 1-t \\ 1-t & 0 & 0 & t & -1 \\ 0 & t & 1-t & -1 & 0 \\ t & 0 & -1 & 1-t & t \end{bmatrix}$$

Die Zeile $i = 5$ und die Spalte $j = 5$ werden gestrichen. Diese sind willkürlich gewählt. Die $(n-1)\times(n-1)$-Alexander-Matrix lautet:

$$\begin{bmatrix} -1 & 1-t & 0 & 0 \\ 0 & -1 & t & 0 \\ 1-t & 0 & 0 & t \\ 0 & t & 1-t & -1 \end{bmatrix}$$

Die Determinante dieser Matrix, also das Alexander-Polynom ist:

$$A_K(t) = -2t^3 + 3t^2 - 2t = (-t)(2t^2 - 3t + 2)$$

.

Das Alexander-Polynom hat eine sehr interessante Eigenschaft, die eine Verbindung zum vorherigen Abschnitt über die p-Färbbarkeit ermöglicht.

Satz 4.7.4. Ein Knoten ist genau dann p-färbbar, wenn sein Alexander-Polynom von -1, also $A_K(-1)$ durch p teilbar ist. Dies lässt sich auch wie folgt ausdrücken: $\mid A_K(-1) \mid$ ist gleich der Determinante (4.5.10) des Knotens ([8], Seite 213).

Der Beweis dieses Satzes wird in dieser Arbeit nicht näher betrachtet.

Beispiel 4.7.5. Die obigen Beispiele 4.5.11 und 4.7.3 können ergänzt werden durch:

$$A_K(-1) = 2 \cdot (-1)^2 - 3 \cdot (-1) + 2 = 2 + 3 + 2 = 7.$$

Da $7 \mid 7$, ist der Knoten 5_2 7-färbbar und nur 7-färbbar.

Kapitel 5

Python-Programm

Diese Arbeit soll nicht nur theoretische Begriffe der Invarianten in der Knotentheorie erklären, sondern auch die Gelegenheit für eine praktische Arbeit bieten. Ich habe daher ein Python-Programm auf der Software PyCharm codiert, dessen Hauptfunktion darin besteht, die p-Färbbarkeiten eines von einem Benutzer eingegebenen Knotens zu berechnen. Dieses Programm, dessen Code im Anhang zu finden ist, ermittelt darüber hinaus die Kreuzungen und Bögen des Knotens. Ausserdem erstellt es eine grafische Darstellung der Eingabe des Knotens durch den Benutzer. Das Programm ermöglicht es schliesslich, Knoten vorzuschlagen, die dieselbe p-Färbbarkeit wie der eingegebenen Knoten haben. Das Programm ist auch im Internet unter folgender Adresse verfügbar: `https://knoten-farbbarkeit-rechner.streamlit.app`. Weitere Informationen hierzu finden Sie im Anhang.

5.1 Gitter-Diagramm

Die Knoten werden als Gitterdiagramme eingegeben, vom englischen “grid diagram”. Diese Art der Darstellung eines Knotens stammt aus dem 19. Jahrhundert [1]. Ein solches auf einem $N \times N$ Gitter gezeichnetes Gitterdiagramm ist durch eine Menge von $2 \cdot N$ Punkten (a, b) definiert, wobei a und b ganze Zahlen zwischen 1 und N sind. Die Spalten sind mit a und die Zeilen mit b nummeriert. Der Punkt $(2, 4)$ befindet sich beispielsweise am Schnittpunkt der zweiten Spalte und der vierten Zeile. Jede Zeile und jede Spalte des Gitters muss genau zwei Punkte enthalten. Der Knoten wird gebildet, indem die Punkte durch vertikale und horizontale Segmente verbunden werden, und zwar so, dass die vertikalen Segmente immer über den horizontalen Segmenten laufen. Die Reihenfolge, in der die Punkte angegeben werden, spielt keine Rolle, da die Konstruktion des Knotens ausschliesslich von ihrer Position abhängt.

Definition 5.1.1. Die kleinste Zahl N, für die ein Knoten K ein Gitterdiagramm der Grösse $N \times N$ hat, wird als *Gitterzahl* bezeichnet, vom englischen “grid number” [17]. Diese Gitterzahl, auch “arc index” genannt, ist eine Knoteninvariante [10] .

Beispiel 5.1.2. Für den Knoten 7_5 ist die Menge der Gitterpunkte [18]:

$[(1,1),(1,3),(2,2),(2,4),(3,3),(3,5),(4,4),(4,7),(5,1),(5,6),(6,5),(6,8),(7,7),(7,9),(8,2),(8,8),(9,6),(9,9)]$

. Das Gitterdiagramm dieses Knotens sieht folglich so aus:

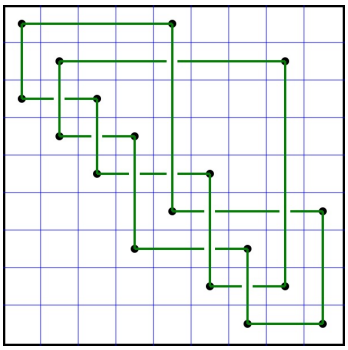


Abbildung 39: Gitterdiagramm des Knotens 7_5.

Die Gitterzahl N des Knotens 7_5 ist 9, denn die Gitterpunkte befinden sich auf einem 9×9-Gitter.

5.2 Struktur des Programms

Dieser Abschnitt der Arbeit erklärt in Worten, was bei der Verwendung des Programms passiert und welche Überlegungen hinter der Struktur des Rechenalgorithmus stehen.

- **Benutzereingabe**: Die Funktion im Programm heisst *eingabe_points*. Der Benutzer gibt über die Tastatur zuerst die Gitterzahl N ein. Danach gibt er die $2 \cdot N$ Punkte in Form von Paaren (a, b) ein. Es gibt eine weitere Eingabemöglichkeit, bei der der Benutzer einfach einen der im Programm vorab eingegebenen Knoten auswählt.

- **Diagramm zeichnen**: Die Funktion *diagramm_zeichnen* formatiert die eingegebenen Punkte. Sie erzeugt das Gitterdiagramm im Textformat mit den Koordinaten a und b und den Punkten.

- **Vorläufige Analyse**: Ausgehend von den eingegebenen Punkten findet die Funktion *punkte_zu_kreuzungen* die verschiedenen vertikalen und horizontalen Segmente, die sie verbinden. Aus diesen Segmenten werden die Kreuzungen im Diagramm ermittelt. Es ist zu beachten, dass die Anzahl dieser Kreuzungen nicht unbedingt der Kreuzungszahl des betreffenden Knotens entspricht. Dies ist daher der Fall, weil das Gitterdiagramm nicht immer mit der minimalen Darstellung des Knotens übereinstimmt.

- **Erstellung der p-Färbbarkeitsmatrix**: Die Funktion *feed_matrix* imitiert die menschliche Vorgehensweise, die p-Färbbarkeitsmatrix zu erstellen, indem sie den Knoten durchläuft. Die p-Färbbarkeitsmatrix wird im Folgenden aus Gründen der besseren Lesbarkeit einfach als Matrix bezeichnet. Die Matrix wird aus den eingegebenen Punkten und den zuvor gefundenen Kreuzungen erstellt und

ist wie folgt strukturiert: Die Zeilen entsprechen den Kreuzungen und die Spalten den Bögen des Knotens. Die Einträge sind standardmässig null, bei einer Unterkreuzung ist der Eintrag -1 und bei einer Oberkreuzung ist der Eintrag 2.

Um die Analyse zu beginnen, wird eine erste Kreuzung willkürlich genommen. Seine Koordinaten werden unter den Variablen a und b gespeichert. Eine Schleife durchläuft alle Bögen des Knotens. Die Anzahl der Bögen ist bekannt, da sie gleich der Anzahl der Kreuzungen ist. In der Matrix wird für den Bogen unter Analyse und die entsprechende Kreuzung eine -1 eingetragen, weil der Beginn eines Bogens eine Unterkreuzung ist.

Horizontale Suche: Die Funktion *hori_search* durchläuft den Bogen horizontal, bis sie entweder eine Unterkreuzung oder eine Richtungsänderung findet, das heisst einen der vom Benutzer eingegebenen Punkte. Die Laufrichtung (rechts oder links) wird von der Funktion *test_h_rechts* bestimmt, mit Ausnahme des ersten Bogens, in dem die Suche willkürlich nach rechts geht.

Wenn *hori_search* eine Unterkreuzung findet, ist dies das Ende des aktuell analysierten Bogens, weil vertikale Segmente immer über horizontale Segmente verlaufen. Es wird also eine -1 in die Matrix geschrieben, und die Koordinaten a und b werden an die Schleife in *feed_matrix* zurückgegeben, die anhand dieser Koordinaten einen neuen Bogen analysiert.

Wenn eine Richtungsänderung gefunden wird, werden die Koordinaten a und b des analysierten Punktes an die vertikale Suchfunktion weitergeleitet.

Vertikale Suche: Zuerst testet die Funktion *verti_search*, ob das vertikale Segment nach oben oder nach unten verläuft. Dann durchläuft sie das Segment, bis sie eine weitere Richtungsänderung findet, also das Ende des vertikalen Segments, welches einem der vom Nutzer eingegebenen Punkte entspricht. Nach einer Richtungsänderung werden die Koordinaten a und b an die horizontale Suchfunktion zurückgegeben, um den weiteren Verlauf dieses Bogens zu analysieren. Bei diesem Durchlauf prüft *verti_search*, ob die untersuchten Punkte auch Oberkreuzungen sind. Eine gefundene Kreuzung kann nämlich aufgrund der Definition des Gitter-Diagramms nur eine Oberkreuzung sein. Wird eine solche Kreuzung gefunden, wird eine 2 in die Matrix eingetragen.

- **Berechnung der möglichen p**: Die Funktion *p_farbbarkeit* erhält die vervollständigte p-Färbbarkeits-

matrix. Sie streicht darin eine Zeile und eine Spalte und berechnet die Determinante der resultierenden Matrix. Anschliessend werden die Primfaktoren dieser Determinante gefunden. Diese Primfaktoren sind die verschiedenen p, für welche der eingegebene Knoten $\mod p$ etikettierbar ist.

- **Vergleich der p-Färbbarkeiten mit der Knotentabelle**: Die Funktion *vergleich_tabelle_mit_p* vergleicht die gerade berechnete p-Färbbarkeiten mit denen in der Tabelle der p-Färbbarkeiten aller Knoten bis zu 8 Kreuzungen. Sie schlägt somit Knoten vor, die dem vom Benutzer eingegebenen Knoten entsprechen könnten. Wenn es keine Übereinstimmung gibt, erläutert die Funktion, dass der eingegebene Knoten kein Knoten mit 8 oder weniger Kreuzungen ist.

- **Benutzerausgabe**: Die Funktion *generate_pdf* erstellt ein PDF-Dokument, das den Programmablauf dokumentiert. In diesem Dokument sind die wichtigsten Elemente der Knotenanalyse zusammengefasst. Es enthält also die “grid points”, das Gitterdiagramm, die p-Färbbarkeitsmatrix, deren Determinante, die p-Färbbarkeiten des Knotens und schliesslich den Vergleich der p-Färbbarkeiten des eingegebenen Knotens mit den Knoten der Tabelle (siehe 5.3).

Die Abbildungen 40 und 41 geben einen visuellen Überblick über den Aufbau des Programms im Allgemeinen beziehungsweise der Funktion *feed_matrix*.

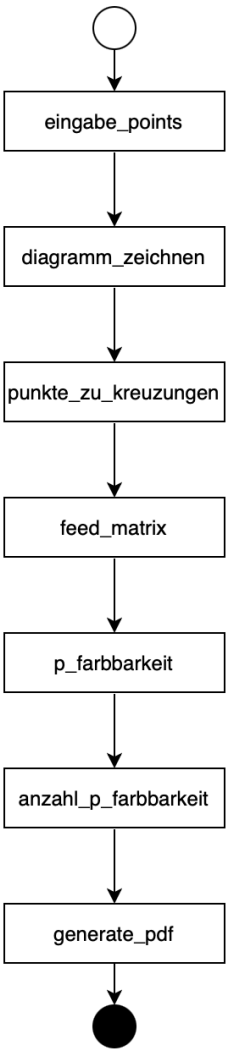


Abbildung 40: Aufbau des Programms.

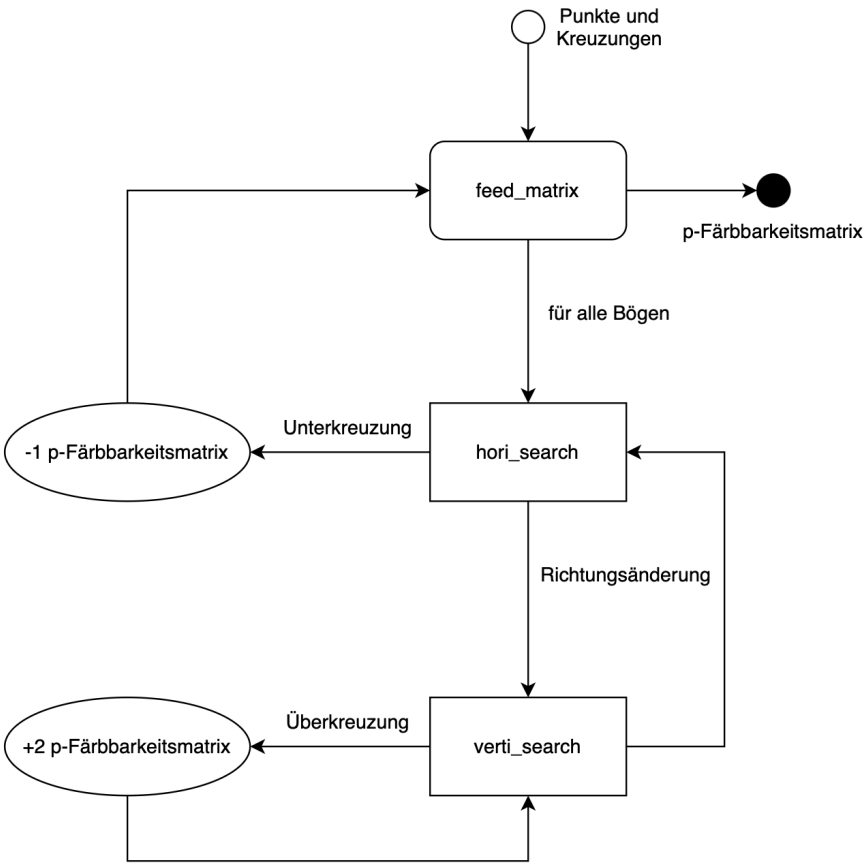


Abbildung 41: Aufbau der Funktion *feed_matrix*.

5.3 Tabelle der p-Färbbarkeiten

Die Eigenschaft des Alexander-Polynoms (4.7.4), die es mit der p-Färbbarkeit verbindet, erlaubt die relativ einfache Erstellung einer Tabelle der p-Färbbarkeiten, da Alexander-Polynome bekannt und gut erforscht sind. Im Rahmen dieser Arbeit wird eine Tabelle erstellt, in der alle Knoten bis zu 8 Kreuzungen, ihr Alexander-Polynom und ihre p-Färbbarkeiten aufgeführt sind. Diese Werte werden dann im Python-Programm verwendet. Die Ergebnisse des Programms werden mit den Werten dieser Tabelle verglichen. Dieser Vergleich ermöglicht es dem Programm, verschiedene Knoten vorzuschlagen, die in das Programm eingegeben werden konnten, entsprechend ihrer p-Färbbarkeit. Das Programm hat dadurch nicht nur eine Funktion zur Berechnung der p-Färbbarkeit, sondern auch eine Vergleichsfunktion, anhand welche die potenziell eingegebenen Knoten reduziert werden können. Die betreffende Tabelle befindet sich im Anhang dieser Arbeit.

Fazit

Diese Maturitätsarbeit über die Knotentheorie nähert sich hier ihrem Ende. Ihr Ziel war es, nicht nur eine allgemeine Einführung in die Knotentheorie zu geben und verschiedene Invarianten zu behandeln, sondern auch selbst etwas Konkretes und Neues zu schaffen. Dieses Ziel wurde erreicht, aber es gibt noch einige Themen oder Fragen, die es verdienen würden, vertieft oder in einer späteren Arbeit behandelt zu werden. Diese Arbeit besteht zunächst aus einer historischen und mathematischen Einführung. Anschliessend enthält sie eine Erläuterung des Begriffs der Invariante sowie eine Erklärung mit Beispielen für sechs Invarianten: die Kreuzungszahl, die Entknotungszahl, die Dreifärbbarkeit, die p-Färbbarkeit, die Anzahl der p-Färbbarkeiten und das Alexander-Polynom. Die wichtigsten Beweise sind entweder enthalten oder skizziert, mit Ausnahme des Beweises der Invarianz des Alexander-Polynoms, auf den zugunsten anderer Themen an dieser Stelle verzichtet wurde. Es gibt eine Vielzahl weiterer Invarianten, von denen die meisten “stärker” sind als die in dieser Arbeit behandelten, die aus mathematischen Gründen nicht zugänglich sind. Invarianten wie Knotengruppen [16], solche, die Seifert-Flächen [13] verwenden, oder das HOMFLY-Polynom [3] erfordern zum Beispiel Kenntnisse und Beherrschung der algebraischen Topologie.
Der praktische Teil dieser Arbeit besteht aus einem Computerprogramm, das die p-Färbbarkeit eines Knotens berechnet. Zwar liefert diese Berechnung keine neuen Erkenntnisse für die Knotentheorie, da bereits andere Verfahren zur Bestimmung der p-Färbbarkeit existieren. Nach meinen Recherchen stellt diese Implementierung jedoch die erste Automatisierung des algebraischen Verfahrens dar, mit dem die p-Färbbarkeit aus dem Knotendiagramm bestimmt werden kann.
Ein Aspekt, der in dieser Arbeit leider nicht vertieft werden konnte, ist die Vielzahl aktueller und praxisnaher Anwendungen der Knotentheorie. Dieses Teilgebiet der Mathematik findet heute unter anderem in der Molekularbiologie, der Chemie und der Quantenphysik Verwendung. Knotenkonzepte spielen beispielsweise beim Verständnis von DNA-Strukturen in der DNA-Replikation [5], bei der Entwicklung des molekularen Designs mit molekularen Knoten [4] oder beim Verständnis der Quantenfeldtheorie [20] und der Stringtheorie [11] dank Knoten als physikalische Objekte in vier Raumzeitdimensionen eine wichtige Rolle.

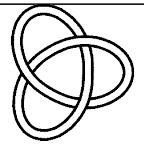
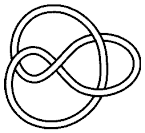
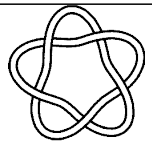
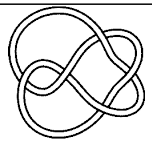
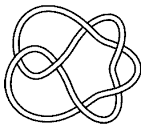
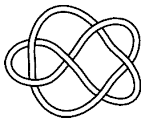
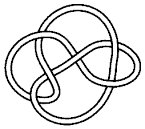
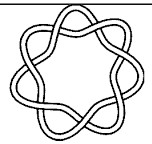
Danksagung

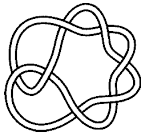
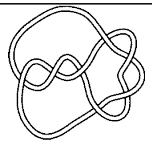
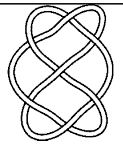
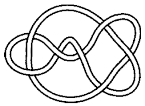
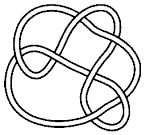
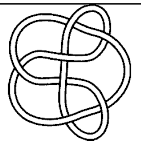
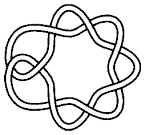
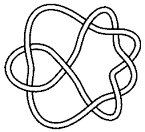
Schlussendlich möchte ich allen Personen danken, die mich bei der Verfassung dieser Arbeit auf die eine oder andere Weise unterstützt haben. Zuerst danke ich meinem Betreuer, Herrn Ingemar Imboden. Er war stets für mich da, wenn ich Fragen oder Unsicherheiten hatte, und er hat mir auch in mathematischer Hinsicht sehr geholfen. Weiter möchte ich meinen Eltern und meinem Freund danken, die mich in Momenten fehlender Motivation unterstützt haben. Ganz besonders danke ich meinem Vater, der sein wertvolles Wissen über Programmierung mit mir geteilt hat. Abschliessend möchte ich allen danken, die diese Arbeit Korrektur gelesen und mir formale sowie sprachliche Rückmeldungen gegeben haben.

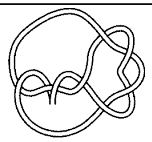
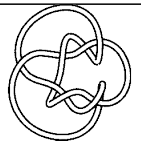
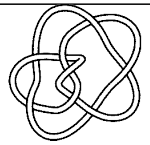
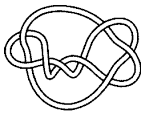
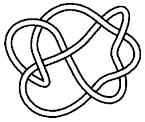
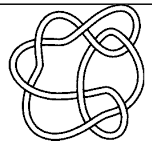
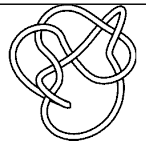
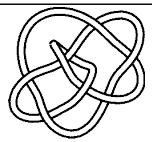
Anhang

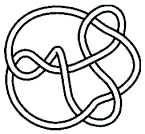
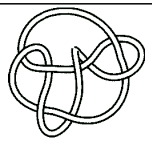
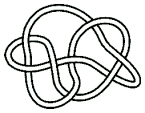
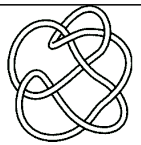
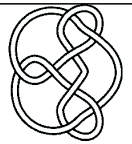
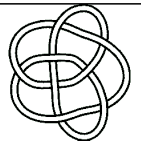
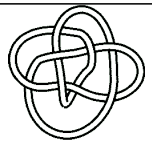
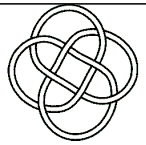
Tabelle der p-Färbbarkeiten

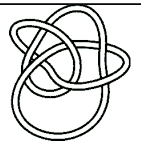
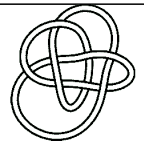
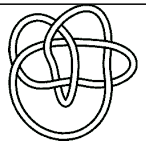
Die Tabelle der p-Färbbarkeiten aller Knoten bis zu 8 Kreuzungen ist das Ergebnis meiner eigenen Arbeit. Die Bilder der Diagramme stammen von der Website: `https://katlas.org/wiki/The_Rolfsen_Knot_Table`, Abruf: 27.09.2025 und die Alexander-Polynome von der Website: `https://knotinfo.org`, Abruf: 27.09.2025.

Knoten	Diagramm	Alexander-Polynom	p=3	p=5	p=7	p=11	p=13	p=17	p=19	p=23	p=29	p=31	p=37
3_1		$1 - t + t^2$	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein	nein
4_1		$1 - 3t + t^2$	nein	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein
5_1		$1 - t + t^2 - t^3 + t^4$	nein	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein
5_2		$2 - 3t + 2t^2$	nein	nein	ja	nein	nein	nein	nein	nein	nein	nein	nein
6_1		$2 - 5t + 2t^2$	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein	nein
6_2		$1 - 3t + 3t^2 - 3t^3 + t^4$	nein	nein	nein	ja	nein	nein	nein	nein	nein	nein	nein
6_3		$1 - 3t + 5t^2 - 3t^3 + t^4$	nein	nein	nein	nein	ja	nein	nein	nein	nein	nein	nein
7_1		$1 - t + t^2 - t^3 + t^4 - t^5 + t^6$	nein	nein	ja	nein	nein	nein	nein	nein	nein	nein	nein

Knoten	Diagramm	Alexander-Polynom	p=3	p=5	p=7	p=11	p=13	p=17	p=19	p=23	p=29	p=31	p=37
7_2		$3 - 5t + 3t^2$	nein	nein	nein	ja	nein	nein	nein	nein	nein	nein	nein
7_3		$2 - 3t + 3t^2 - 3t^3 + 2t^4$	nein	nein	nein	nein	ja	nein	nein	nein	nein	nein	nein
7_4		$4 - 7t + 4t^2$	ja	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein
7_5		$2 - 4t + 5t^2 - 4t^3 + 2t^4$	nein	nein	nein	nein	nein	ja	nein	nein	nein	nein	nein
7_6		$1 - 5t + 7t^2 - 5t^3 + t^4$	nein	nein	nein	nein	nein	nein	ja	nein	nein	nein	nein
7_7		$1 - 5t + 9t^2 - 5t^3 + t^4$	ja	nein	ja	nein	nein	nein	nein	nein	nein	nein	nein
8_1		$3 - 7t + 3t^2$	nein	nein	nein	nein	ja	nein	nein	nein	nein	nein	nein
8_2		$1 - 3t + 3t^2 - 3t^3 + 3t^4 - 3t^5 + t^6$	nein	nein	nein	nein	nein	ja	nein	nein	nein	nein	nein

Knoten	Diagramm	Alexander-Polynom	p=3	p=5	p=7	p=11	p=13	p=17	p=19	p=23	p=29	p=31	p=37
8_3		$4 - 9t + 4t^2$	nein	nein	nein	nein	nein	ja	nein	nein	nein	nein	nein
8_4		$2 - 5t + 5t^2 - 5t^3 + 2t^4$	nein	nein	nein	nein	nein	nein	ja	nein	nein	nein	nein
8_5		$1 - 3t + 4t^2 - 5t^3 + 4t^4 - 3t^5 + t^6$	ja	nein	ja	nein	nein	nein	nein	nein	nein	nein	nein
8_6		$2 - 6t + 7t^2 - 6t^3 + 2t^4$	nein	nein	nein	nein	nein	nein	nein	ja	nein	nein	nein
8_7		$1 - 3t + 5t^2 - 5t^3 + 5t^4 - 3t^5 + t^6$	nein	nein	nein	nein	nein	nein	nein	ja	nein	nein	nein
8_8		$2 - 6t + 9t^2 - 6t^3 + 2t^4$	nein	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein
8_9		$1 - 3t + 5t^2 - 7t^3 + 5t^4 - 3t^5 + t^6$	nein	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein
8_{10}		$1 - 3t + 6t^2 - 7t^3 + 6t^4 - 3t^5 + t^6$	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein	nein

Knoten	Diagramm	Alexander-Polynom	p=3	p=5	p=7	p=11	p=13	p=17	p=19	p=23	p=29	p=31	p=37
8_{11}		$2-7t+9t^2-7t^3+2t^4$	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein	nein
8_{12}		$1-7t+13t^2-7t^3+t^4$	nein	nein	nein	nein	nein	nein	nein	nein	ja	nein	nein
8_{13}		$2-7t+11t^2-7t^3+2t^4$	nein	nein	nein	nein	nein	nein	nein	nein	ja	nein	nein
8_{14}		$2-8t+11t^2-8t^3+2t^4$	nein	nein	nein	nein	nein	nein	nein	nein	nein	ja	nein
8_{15}		$3-8t+11t^2-8t^3+3t^4$	ja	nein	nein	ja	nein	nein	nein	nein	nein	nein	nein
8_{16}		$1-4t+8t^2-9t^3+8t^4-4t^5+t^6$	nein	ja	ja	nein	nein	nein	nein	nein	nein	nein	nein
8_{17}		$1-4t+8t^2-11t^3+8t^4-4t^5+t^6$	nein	nein	nein	nein	nein	nein	nein	nein	nein	nein	ja
8_{18}		$1-5t+10t^2-13t^3+10t^4-5t^5+t^6$	ja	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein

Knoten	Diagramm	Alexander-Polynom	p=3	p=5	p=7	p=11	p=13	p=17	p=19	p=23	p=29	p=31	p=37
8_{19}		$1 - t + t^3 - t^5 + t^6$	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein	nein
8_{20}		$1 - 2t + 3t^2 - 2t^3 + t^4$	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein	nein
8_{21}		$1 - 4t + 5t^2 - 4t^3 + t^4$	ja	ja	nein	nein	nein	nein	nein	nein	nein	nein	nein

Programmcode

Der Python-Programmcode (siehe 5), um den es in dieser Arbeit geht, ist hier zu finden. Er ist das Ergebnis meiner persönlichen Arbeit mit der Entwicklungssoftware PyCharm. Gelegentlich habe ich auf IT-Hilfeforen wie `https://stackoverflow.com/questions` oder `https://www.geeksforgeeks.org` zurückgegriffen. Diese Websites wurden ausschliesslich dazu genutzt, um mir die Lösungen anzusehen, die von Nutzern verwendet wurden, die beim Schreiben des Codes auf die gleichen Probleme gestossen sind wie ich. Das Programm ist im Internet unter folgender Adresse verfügbar: `https://knoten-farbbarkeit-rechner.streamlit.app`. Der Link kann in allen gängigen Browsern (Safari, Chrome, Firefox, Edge, Brave usw.) geöffnet werden. Die Web-App muss manchmal aktiviert sein; eine gewisse Latenzzeit ist dabei normal. Die Anweisungen werden dann auf dem Bildschirm angezeigt. Diese Web-App wurde von einer künstlichen Intelligenz erstellt: Claude, Sonnet 4.5 (Gratisversion). Diese künstliche Intelligenz hat meinen Code in einen für das Webformat formatierten Code umgewandelt, aber nicht zum Entstehungsprozess des Programms beigetragen. Das Programm wurde anschliessend auf der Website GitHub und dann auf dem Python-Server Streamlit online gestellt.

```
import numpy as np
import sympy as sp
from sympy import primefactors
import sys
from reportlab.lib.pagesizes import letter, A4
from reportlab.lib import colors
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph,
↪  Spacer,
Image, PageBreak
import datetime
from reportlab.lib.units import inch
import os
import subprocess

points_by_col = {}
points_by_row = {}

tabelle = {
      '3_1': [3],
      '4_1': [5],
      '5_1': [5],
      '5_2': [7],
      '6_1': [3],
      '6_2': [11],
      '6_3': [13],
      '7_1': [7],
      '7_2': [11],
```

```
        '7_3': [13],
        '7_4': [3, 5],
        '7_5': [17],
        '7_6': [19],
        '7_7': [3, 7],
        '8_1': [13],
        '8_2': [17],
        '8_3': [17],
        '8_4': [19],
        '8_5': [3, 7],
        '8_6': [23],
        '8_7': [23],
        '8_8': [5],
        '8_9': [5],
        '8_10': [3],
        '8_11': [3],
        '8_12': [29],
        '8_13': [29],
        '8_14': [31],
        '8_15': [3, 7],
        '8_16': [5, 7],
        '8_17': [37],
        '8_18': [3, 5],
        '8_19': [3],
        '8_20': [3],
        '8_21': [3, 5]
    }


points= []
crossings = []
vertical_lines = []  # (spalte, start_zeile, end_zeile)
horizontal_lines = []  # (zeile, start_spalte, end_spalte)
bögen = []  # für jeder bogen i: alle Kreuzungswerte -1 oder 2 oder 0 wenn keine
↪  Kreuzung
p = []
matches = []


num_crossings = 0
N = 0

# Matrix mit Leerzeichen initialisieren, type weil es kein Integer ist
diagramm = np.full((N, N), ' ', dtype=str)
bögen_gestrichen= np.full((N, N), 0)


h_test_zu_machen= False
richtung_rechts= True

determinante=0
```

```
#Punkte schon eingegeben für eine einfachere User-Eingabe
Nkleeblatt=5
pointskleeblatt = [[1, 1], [1, 4], [2, 3], [2, 5], [3, 2], [3, 4], [4, 1], [4, 3],
↪  [5, 2], [5, 5]]
Nachter=6
pointsachter=[[1,1],[1,3],[2,2],[2,4],[3,3],[3,6],[4,1],[4,5],[5,4],[5,6],[6,2],[6,5
↪  ]]
N8_21=8
points8_21=[[1,1],[1,3],[2,2],[2,5],[3,4],[3,6],[4,5],[4,8],[5,3],[5,7],[6,1],[6,6],
↪  [7,4],[7,8],[8,2],[8,7]]



def eingabe_user():
    global points
    global N

    print("================================================================================
    ↪  ============")
    print("Willkommen zum Programm zur Berechnung der p-Färbbarkeit von
    ↪  mathematischen Knoten.")
    print("================================================================================
    ↪  ============")
    print("Bitte befolgen Sie die Anweisungen auf der Konsole.")
    print("Nach Ausführung des Programms können Sie ein Dokument mit allen
    ↪  Informationen zur Analyse des Knotens abrufen.")
    wahl1=str(input("\nWenn Sie einen bereits im Programm gespeicherten Knoten
    ↪  auswählen möchten, drücken Sie die Taste [p].\nWenn Sie einen Knoten selbst
    ↪  eingeben möchten, drücken Sie die Taste [s].\n--> "))

    if wahl1=="p":
        wahl2=str(input("Für den Kleeblattknoten 3_1 drücken Sie bitte die Taste
        ↪  [1].\nFür den Achterknoten 4_2 drücken Sie bitte die Taste [2].\nFür den
        ↪  Knoten 8_21 drücken Sie bitte die Taste [3].\n--> "))

        if wahl2=="1":
            points=pointskleeblatt
            N=Nkleeblatt

        elif wahl2=="2":
            points=pointsachter
            N=Nachter

        elif wahl2=="3":
            points=points8_21
            N=N8_21

        else:
            print("Eingegebener Wert nicht gültig. Bitte versuchen Sie es erneut.")
    elif wahl1=="s":
```

```
        print("Bitte geben Sie den Knoten als \"Grid diagram\" ein.")
        print("Weitere Informationen und Quellen zu diesem Thema finden Sie auf der
        ↪  ausgezeichneten Website:")
        print("https://knotinfo.org/descriptions/grid_notation.html")
        print("\nDie \"Grid points\" können ebenfalls auf der Webseite:
        ↪  https://knotinfo.org gefunden werden.")
        N=int(input("\nWas ist die Grösse des Grid Diagramms (Gitterzahl N)? \n-->
        ↪  "))
        eingabe_points(N)

    else:
        print("Eingegebener Wert nicht gültig. Bitte versuchen Sie es erneut.")

def eingabe_points(N):

    global points

    for i in range(2 * N):
        print("Punkt", i+1)
        print("_________")
        a = int(input("Spaltenkoordinate des Punktes:"))
        b = int(input("Zeilenkoordinate des Punktes:"))
        points.append([a, b])

def punkte_zu_kreuzungen(points):

    global bögen
    global num_crossings
    global crossings
    global vertical_lines
    global horizontal_lines

    if len(points) % 2 != 0:
        raise ValueError("Anzahl der Punkte muss gerade sein (2 Punkte pro Spalte und
        ↪  Zeile im Diagramm)")

    # Gruppiere Punkte nach Spalten und Zeilen
    for col, row in points:  # punkte so gegeben--> [Spalte, Zeile], durchgehen
        if col not in points_by_col:  # wenn diese Spalte noch nicht da ist
            points_by_col[col] = []  # neue Liste als Item bei dieser Spalte im
            ↪  Dictionary
        points_by_col[col].append(row)  # Zeilenzahl dieses Punktes in der Liste->
        ↪  welche Punkte in dieser Spalte

        if row not in points_by_row:
            points_by_row[row] = []
        points_by_row[row].append(col)

    # Punkte in jeder Spalte/Zeile sortieren
    for col in points_by_col:
        points_by_col[col].sort()
```

```
for row in points_by_row:
    points_by_row[row].sort()

print("Punkte in Spalten", points_by_col)
print("Punkte in Zeilen", points_by_row)

# Erstelle vertikale und horizontale Linien
# Vertikale Linien (2 Punkte pro Spalte)
for col, rows in points_by_col.items():  # col=in welcher Spalte, rows=
↪  Zeilenzahl der 2 Punkte in dieser Spalte
    if len(rows) == 2:
        vertical_lines.append(
            (col, min(rows), max(rows)))  # zu vertikaler Linie: welche Spalte,
            ↪  Anfang->minimum, Ende->maximum

# Horizontale Linien (2 Punkte pro Zeile)
for row, cols in points_by_row.items():
    if len(cols) == 2:
        horizontal_lines.append(
            (row, min(cols), max(cols)))  # zu horizontaler Linie: welche Zeile,
            ↪  Anfang-> minimum, Ende->maximum

print("Vertikale Segmente:", vertical_lines)
print("Horizontale Segmente:", horizontal_lines)

# Kreuzungen finden

for v_col, v_start, v_end in vertical_lines:  # v_col=in welche Spalte ist der
↪  vertikaler Segment, v_start und v_end=Start- und Endpunkte dieses Segmentes
    for h_row, h_start, h_end in horizontal_lines:
        # ob die vertikale und horizontale Segmente sich kreuzen, wenn Spalte vom
        ↪  vertikalen Segment zwischen start und end vom horizontalen ist und
        ↪  umgk-> Kreuzung
        if (h_start < v_col < h_end) and (v_start < h_row < v_end):
            crossings.append((v_col, h_row))

# Sortiere Kreuzungen
crossings.sort(key=lambda x: (x[1], x[0]))  # sortiert zuerst nach Zeile(zweites
↪  Element), dann Spalte (erstes Element)

num_crossings = len(crossings)  # Anzahl Kreuzungen

if num_crossings == 0:
    sys.exit("Fehler: Keine Kreuzungen gefunden! Keine Analyse ist möglich, bitte
    ↪  versuchen Sie es erneut.")

bögen = np.zeros((num_crossings, num_crossings))  # zuerst Zeile dann Spalte,
↪  Zeile=Kreuzung, Spalte=Bogen

print("Es gibt", num_crossings, "Kreuzungen:", crossings)
```

```
    print("Das ist nicht unbedingt gleich die minimale Anzahl Kreuzungen dieses
    ↪ Knotens!")

def diagramm_zeichnen(points):
    global diagramm
    print("Punkte:", points)

    # Matrix mit Leerzeichen initialisieren, type weil es kein Integer ist
    diagramm = np.full((N, N), ' ', dtype=str)

    # Punkte als Sterne markieren-> alle eingegebene Punkte werden als Stern gegeben
    for point in points:  # durch Punkte gehen
        diagramm_spalte = point[0]  # erste Koordinate Spalte
        diagramm_zeile = point[1] # zweite Koordinate Zeile
        diagramm[diagramm_zeile-1, diagramm_spalte-1] = '*' #-1 wegen Indexierung auf
        ↪ 0 basiert

    print("\nDiagramm (", N, "x", N, "mit", 2 * N, "Punkten):")

    print("  " + " ".join(str(i + 1) for i in range(N)))  # Spalten-Bezeichnung,
    ↪ join->Konkatenierung von Indexen (0+1 bis N+1) mit Leerzeichen

    for i in range(N):
        print(i + 1, ' '.join(diagramm[i]))  # Zeilen: Bezeichnung i+1 und join->
        ↪ Konkatenierung von Elementen der i-ten Zeile des Diagramms mit
        ↪ Leerzeichen

def test_h_rechts(a,b):
#test verifiziert für hori_search ob man rechts oder links geht
#a ist die Spaltekoordinate und b ist die Zeilenkoordinate

    global horizontal_lines

    ligne=-1 #ligne wird mit einem ungültigen Wert definiert aber nachdem einen neuen
    ↪ Wert gegeben

    for i in range(len(horizontal_lines)): #geht durch alle horizontale Zeilen
        if b == horizontal_lines[i][0]: #wenn die betroffene Zeile gefunden ist, wird
        ↪ das unter ligne gespeichert
          ligne = i
          break

    if ligne==-1:
        print("Fehler: Keine entsprechende Zeile für:", b, "gefunden!")

    #in horizontal_lines wurden die Spaltenkoordinate geordnet
    if a== horizontal_lines[ligne][1]: #äusserster linker Punkt -> nach rechts gehen
        return True
    elif a== horizontal_lines[ligne][2]:#äusserster rechter Punkt -> nach links gehen
        return False
    else:
```

```
        print("Fehler: Punkt",a,b,"nicht am Endpunkt einer Zeile")

def test_v_oben(a,b):
#Test, ob man nach oben oder nach unten bei vertikaler Suche geht

    global vertical_lines

    colonne = -1 #colonne wird mit einem ungültigen Wert definiert aber nachdem einen
    ↪  neuen Wert gegeben

    for i in range(len(vertical_lines)): #in welche Spalte findet man sich befindet
        if a == vertical_lines[i][0]:
            colonne= i
            break

    if colonne==-1:
        print("Fehler: keine entsprechende Spalte für", a, "gefunden!")

    if b == vertical_lines[colonne][1]: #wenn es vom ersten Punkt handelt, liegt
    ↪  dieser Punkt oben, deshalb muss man nach unten weitersuchen -> oben=false
        return False
    elif b == vertical_lines[colonne][2]:#wenn es vom zweiten Punkt handelt, liegt
    ↪  dieser Punkt unten, deshalb muss man nach oben weitersuchen -> oben=true
        return True
    else:
        print("Fehler: Punkt",a,b,"nicht am Endpunkt einer Spalte")

def verti_search(s,a,b, points, crossings, v_test_zu_machen, richtung_oben=False):
    # s : Bogen
    # a : Spaltenkoordinate des Punktes zu analysieren
    # b : Zeilenkoordinate des Punktes zu analysieren
    # v_test_zu_machen : boolean um Richtungstest zu zwingen, test nur machen wenn
    ↪  vertisearch von horisearch angerufen wird
    # richtung_oben=False fakultativer Argument um Richtungssuche zu holen wenn Test
    ↪  nicht nötig ist

    # wenn Kreuzung-> uberkreuzung, bögen mit +2, ->verti-search
    # sonst weiter suchen-> inkrementieren
    # wenn Richtungsänderung-> hori_search & Koordinaten a und b übermitteln

    global bögen
    global h_test_zu_machen

    if v_test_zu_machen == True: #d.h. wenn verti_search von hori_search aufgerufen
    ↪  wird
        oben = test_v_oben(a,b) #nur wenn verti_search von hori_search aufgerufen
        ↪  wird -> Richtungsänderung
    else: #d.h. wenn verti_search von sich selbst aufgerufen wird
        oben=richtung_oben

    if oben == True:    # test --> nach oben suchen
```

```
        b = b-1
    elif oben == False:  # test --> nach unten suchen
        b = b+1

    if b < 1 or b > N:
        print("Fehler: vertikale Suche ist ausserhalb der Grenzen gegangen!")
        return

    target = [a,b]


    # wenn punkt a b eine Richtungsänderung ist, wird die horizontale Suche angerufen
    for point in points:  # point= index in Punktenliste (user eingabe), die die
    ↪  Richtungsänderungen entsprechen
            if target == point:  # ist eine Richtungsänderung
                h_test_zu_machen = True
                target=hori_search(s,a,b, points, crossings)
                return target #verlässt verti_search, erreicht es nicht (hori_search
                ↪  stattdessen angerufen)


    for n, crossing in enumerate(crossings): #n ist Kreuzungindex und crossing ist
    ↪  inhalt von crossings
            if target == list(crossing): #tuple (Kreuzung) zu einer Liste (wie
            ↪  Target) für den Vergleich umgewandelt
                if n<len(bögen) and s<len(bögen[n]): #ist der Kreuzungsindex in den
                ↪  Matrix-Grenzen und ist der Bogenindex in den Matrix-Grenzen (n-te
                ↪  Spalte)
                    bögen[n][s] = 2 #Kreuzung ist überführung-> +2 in der Matrix
                break

    #ruft sich selbst wieder: vertikales Segment entlang gehen, solange man keine
    ↪  Richtungsänderung findet
    # test zu machen als False: weil man wiederum in der gleichen Richtung suchen
    ↪  soll (gleiches vertikales Segment)
    target=verti_search(s, a, b, points, crossings, False, oben)
    return target

def hori_search(s,a,b, points, crossings):
#a und b Koordinaten von entweder Anfangskreuzung oder Richtungsänderung am Ende von
↪  verti_search; s ist Index des Bogens

    global bögen
    global h_test_zu_machen
    global richtung_rechts

    if h_test_zu_machen == True:
        rechts = test_h_rechts (a,b)
    elif h_test_zu_machen == False: #wenn hori_search nicht von verti_search
    ↪  abgerufen ist
        rechts = richtung_rechts
```

```
    for j in range(N): #Durchlaufschleife solange keine Kreuzung und keine
    ↪  Richtungsänderung

        if rechts == True:
            a=a+1   #nach rechts suchen
        elif rechts == False:
            a=a-1     # nach links suchen

        if a<1 or a>N+1:
            print("Fehler: horizontale Suche ist ausserhalb der Grenzen gegangen!")

        target=[a,b]

        #ob target eine Kreuzung ist
        for n, crossing in enumerate(crossings): #n = index in Kreuzungsliste
            if target== list(crossing): #Ende des Bogens s-> Wert -1 in Matrix
                    if n<len(bögen) and s<len(bögen[n]):
                            bögen[n][s]= -1 #Ende Bogen, Unterkreuzung i und Bogen s
                            h_test_zu_machen = False
                            richtung_rechts=rechts #übermittlung von der Suchrichtung
                            ↪  für den neuen Bogen
                    return a, b #return gibt die Werte des Bogenendes an feedmatrix
                    ↪  weiter

        #ob target eine Richtungsänderung ist->verti_search
        for n, point in enumerate(points): #n = index in Punktenliste (User-Eingabe),
        ↪  Punkte entsprechen den Richtungsänderungen
            if target==point: #Richtungsänderung
                    v_test_zu_machen = True #Richtungstest nötig wenn verti_search
                    ↪  abgerufen wird
                    target=verti_search(s, a, b, points, crossings,True)
                    ↪  #verti_search abrufen
                    return target
def feed_matrix(crossings, num_crossings):
    #feed_matrix füllt die p-Färbbarkeitsmatrix, indem sie alle Bögen nacheinander
    ↪  analysiert
    global bögen
    global h_test_zu_machen
    global richtung_rechts
    k=0 #erste Kreuzung willkürlich gewählt
    h_test_zu_machen = False
    richtung_rechts =True #willkürlich zuerst nach rechts gesucht

    a = crossings[k][0]  # a ist die Spaltenkoordinate der Kreuzung k
    b = crossings[k][1]  # b ist die Zeilenkoordinate der Kreuzung k


    for s in range(num_crossings): #Anzahl Kreuzungen ist gleich Anzahl Bögen

            print("\n--- Analyse vom Bogen", s, "beginnt in: (",a,",",b,").---")
```

```
            # Kreuzungsindex finden
            crossing_found = False
            for i, crossing in enumerate(crossings):
                if crossing[0] == a and crossing[1] == b:
                    k = i
                    crossing_found = True
                    break

            if not crossing_found:
                print("Fehler: Kreuzung (", a, ",", "b) nicht gefunden!")
                break

            # Anfang eines Bogens-> Wert -1 in p-Färbbarkeitsmatrix
            if k < len(bögen) and s < len(bögen[k]):
                bögen[k][s] = -1

            if s == 0:
                h_test_zu_machen = False
                richtung_rechts = True


            # Suche nach Ende Bogen
            try:
                result = hori_search(s, a, b, points, crossings)

                if result is not None:
                    a,b = result
                    print("Ende vom Bogen", s, "in: (", a, ",", b, ").")
                else:
                    print(f"Fehler: die horizontale Suche hat keine Ergebnisse für
                    ↪  den Bogen {s} gefunden!")
                    break
            except Exception as e:
                print(f"Fehler im Bogen {s}: {e}")
                break

            print(f"Bogen {s} endet in ({a},{b})")
            print("Aktuelle p-Färbbarkeitsmatrix:")
            for row in bögen:
                print(row)

    print("=== Analyse der Bögen ist fertig ===\n")
    print("Endgültige p-Färbbarkeitsmatrix :\n")
    print(bögen)

def diagramm_zeichnen(points):
    global diagramm
    print("Punkte:", points)
```

```
    # Matrix mit Leerzeichen initialisieren, type weil es kein integer ist
    diagramm = np.full((N, N), ' ', dtype=str)

    # Punkte als Sterne markieren-> alle eingegebene Punkte werden als Stern gegeben
    for point in points:  # durch Punkte gehen
        diagramm_spalte = point[0]  # erste Koordinate Spalte
        diagramm_zeile = point[1] # zweite Koordinate Zeile
        diagramm[diagramm_zeile-1, diagramm_spalte-1] = '*' #-1 wegen Indexierung auf
        ↪  0 basiert

    print("\nDiagramm (", N, "x", N, "mit", 2 * N, "Punkten):")

    print("  " + " ".join(str(i + 1) for i in range(N)))  # Spalten-Bezeichnung,
    ↪  join->Konkatenierung von Indexen (0+1 bis N+1) mit Leerzeichen

    for i in range(N):
        print(i + 1, ' '.join(diagramm[i]))  # Zeilen: Bezeichnung i+1 und join->
        ↪  Konkatenierung von Elementen der i-ten Zeile des Diagramms mit
        ↪  Leerzeichen

def p_farbbarkeit(bögen):

    global determinante
    global p
    global bögen_gestrichen

    bögen_gestrichen=bögen[:-1, :-1] #streicht eine Zeile und eine Spalte,
    ↪  willkürlich, hier die letzten
    print("\n(n-1)*(n-1)-Matrix:\n", bögen_gestrichen)

    determinante= int(sp.Matrix(bögen_gestrichen).det()) #Berechnung der Determinante
    ↪  der (n-1)*(n-1)-Matrix
    print("\nDeterminante der (n-1)*(n-1)-Matrix:", determinante)

    determinante_abs = abs(determinante) #Betrag der Determinante
    primfaktoren=primefactors(determinante_abs) #Primfaktorzerlegung des Betrags der
    ↪  Determinante

    p.extend([factor for factor in primfaktoren])

    print("Der eingegebene Knoten ist für diese p=", p, "etikettierbar.")

def vergleich_tabelle_mit_p(p, tabelle):
    global matches

    p_set = set(p)
    matches = [nom for nom, valeurs in tabelle.items() if set(valeurs) == p_set]
    if len(matches) == 1:
        print("\nDer eingegebene Knoten könnte dem Knoten:", matches, "entsprechen.")
    elif len(matches) !=0:
```

```
        print("\nDer eingegebene Knoten könnte einem der Knoten:", matches,
        ↪  "entsprechen.")
    elif len(matches) == 0:
        print("\nDer eingegebene Knoten ist kein Knoten mit 8 Kreuzungen oder
        ↪  weniger!")

    print("Das Dokument, das die Analyse des Knotens zusammenfasst, sollte sich in
    ↪  einem Pop-up-Fenster öffnen, "
          "wenn Sie ein MacBook verwenden."
          "\nAndernfalls finden Sie es als PDF-Datei unter dem Namen
          ↪  \"knoten_resultate.pdf\" "
          "im Ordner des Programms.\n")
    print("\n===ENDE===")

class KnotReportGenerator:
    def __init__(self, points, diagramm, bögen_gestrichen, determinante, p, matches):

        self.points = points
        self.diagramm = diagramm
        self.bögen_gestrichen = bögen_gestrichen
        self.determinante = determinante
        self.p = p
        self.matches = matches
        self.timestamp=datetime.datetime.now().strftime("%d/%m/%Y")
        self.timestamp2 = datetime.datetime.now().strftime("%H:%M:%S")

    def generate_pdf(self):
        doc = SimpleDocTemplate("knoten_resultate.pdf", pagesize=A4)
        story = []
        styles = getSampleStyleSheet()

        title_style = ParagraphStyle(
            'Resultate der Knotenanalyse',
            parent=styles['Heading1'],
            fontSize=24,
            textColor=colors.cadetblue,
            spaceAfter=40,
            alignment=1)

        story.append(Paragraph("Resultate der Knotenanalyse", title_style))
        story.append(Paragraph(f"Datum : {self.timestamp}", styles['Normal']))
        story.append(Paragraph(f"Zeit : {self.timestamp2}", styles['Normal']))
        story.append(Spacer(1, 15))


        story.append(Paragraph("Punkte des Gitterdiagramms :", styles['Heading2']))
        points_text = ", ".join([f"({int(x)}, {int(y)})" for x, y in self.points])
        story.append(Paragraph(points_text, styles['Normal']))
        story.append(Spacer(1, 20))

        story.append(Paragraph("Gitterdiagramm :", styles['Heading2']))
```

```
N = len(self.diagramm)
labeled_diagramm = []

header_row = [''] + [str(i + 1) for i in range(N)]
labeled_diagramm.append(header_row)

for i, row in enumerate(self.diagramm.tolist()):
    labeled_row = [str(i + 1)] + row
    labeled_diagramm.append(labeled_row)

grid_table = Table(labeled_diagramm)
grid_table.setStyle(TableStyle([
        ('BACKGROUND', (0, 0), (-1, -1), colors.white),
        ('TEXTCOLOR', (0, 0), (-1, -1), colors.black),
        ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
        ('FONTNAME', (0, 0), (-1, -1), 'Courier'),
        ('FONTSIZE', (0, 0), (-1, -1), 10),
        ('GRID', (0, 0), (-1, -1), 0.8, colors.black),

        ('BACKGROUND', (0, 0), (-1, 0), colors.cadetblue),
        ('BACKGROUND', (0, 0), (0, -1), colors.cadetblue),
        ('FONTNAME', (0, 0), (-1, 0), 'Courier-Bold'),
        ('FONTNAME', (0, 0), (0, -1), 'Courier-Bold'),
    ]))
story.append(grid_table)
story.append(Spacer(1, 20))

story.append(Paragraph("p-Färbbarkeitsmatrix :", styles['Heading2']))
matrix_data_als_int= [[int(val) for val in row] for row in
↪  self.bögen_gestrichen.tolist()]
matrix_table=Table(matrix_data_als_int,
↪  colWidths=[0.4*inch]*len(self.bögen_gestrichen[0]))
matrix_table.setStyle(TableStyle([
        ('BACKGROUND', (0, 0), (-1, -1), colors.white),
        ('TEXTCOLOR', (0, 0), (-1, -1), colors.black),
        ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
        ('VALIGN', (0,0), (-1, -1), 'MIDDLE'),
        ('FONTNAME', (0, 0), (-1, -1), 'Helvetica'),
        ('FONTSIZE', (0, 0), (-1, -1), 10),
        ('BOTTOMPADDING', (0, 0), (-1, -1), 8),
        ('TOPPADDING', (0, 0), (-1, -1), 8),
        ('LEFTPADDING', (0, 0), (-1, -1), 6),
        ('RIGHTPADDING', (0, 0), (-1, -1), 6),
        ('BOX', (0,0), (-1, -1), 1, colors.black),
        ('LINEABOVE', (0, 0), (-1, 0), 1, colors.white),
        ('LINEBELOW', (0, -1), (-1, -1), 1, colors.white)
]))

display_table=Table([["A =", matrix_table]], colWidths=[0.8*inch, None])
display_table.setStyle(TableStyle([
        ('ALIGN', (0, 0), (0, 0), 'RIGHT'),
```

```
        ('VALIGN', (0, 0), (-1, -1), 'MIDDLE'),
        ('FONTNAME', (0, 0), (0, 0), 'Helvetica'),
        ('FONTSIZE', (0, 0), (0, 0), 12),
        ('RIGHTPADDING', (0, 0), (0, 0), 12),
]))

story.append(display_table)
story.append(Spacer(1, 20))

story.append(Paragraph("Determinante der p-Färbbarkeitsmatrix :",
↪  styles['Heading2']))
story.append(Paragraph(f"det(A) = {self.determinante}", styles['Normal']))
story.append(Spacer(1, 20))

story.append(Paragraph("Primzahlen p, die eine p-Färbung des Knotens erlauben
↪  :", styles['Heading2']))
story.append(Paragraph(f"p = {self.p}", styles['Normal']))
story.append(Spacer(1, 20))



story.append(Paragraph("Möglicherweise übereinstimmende Knoten :",
↪  styles['Heading2']))
story.append(Paragraph("Das Programm schlägt Knoten vor, "
                       "die entsprechend ihrer p-Färbbarkeit mit dem
                       ↪  eingegebenen "
                       "Knoten übereinstimmen könnten. Die p-Färbbarkeit ist
                       ↪  eine Invariante in der "
                       "Knotentheorie, das heisst derselbe Knoten ist immer
                       ↪  für diesselbe Primzahl p färbbar, "
                       "unabhängig von seiner Projektion. Dieser Vorschlag
                       ↪  ist jedoch keineswegs "
                       "eine Gewissheit, da es keine absolute Invariante
                       ↪  gibt. Um eine grössere "
                       "Sicherheit hinsichtlich des eingegebenen Knotens zu
                       ↪  erhalten, müssten weitere "
                       "Invarianten berechnet werden. Zum Vergleich verwendet
                       ↪  das Programm eine Tabelle "
                       "mit den p-Färbbarkeiten aller Knoten bis zu 8
                       ↪  Kreuzungen. Diese Tabelle wurde "
                       "erstellt, indem man sich zunutze machte, dass das
                       ↪  Alexander-Polynom von -1 "
                       "eines Knotens gleich der Determinante der
                       ↪  p-Färbbarkeitsmatrix ist.", styles['Normal']))
if self.matches:
    matched_heading_styles = ParagraphStyle('MatchedKnots',
                                            parent=styles['Heading3'],
                                            fontSize=10,
                                            spaceAfter=5,
                                            spaceBefore=10)
```

```
            story.append(Paragraph("Knoten mit übereinstimmenden p-Färbbarkeiten:",
            ↪  matched_heading_styles))
            for match in self.matches:
                story.append(Paragraph(match, styles['Normal']))
        else:
            story.append(Paragraph("Der eingegebene Knoten hat eine minimale
            ↪  Kreuzungszahl von mehr als 8, da das oder die p nicht mit einem der
            ↪  Einträge in der Tabelle übereinstimmen.", styles['Normal']))

        doc.build(story)
        print("PDF Resultate generiert: knoten_resultate.pdf")

        if os.path.exists("knoten_resultate.pdf"):
            subprocess.Popen(["open", "knoten_resultate.pdf"])

if __name__ == "__main__":

    eingabe_user()

    diagramm_zeichnen(points)

    punkte_zu_kreuzungen(points)

    feed_matrix(crossings, num_crossings)

    p_farbbarkeit(bögen)

    vergleich_tabelle_mit_p(p, tabelle)

    report= KnotReportGenerator(
        points=points,
        diagramm=diagramm,
        bögen_gestrichen=bögen_gestrichen,
        determinante=determinante,
        p=p,
        matches=matches
    )

    report.generate\_pdf()
```

Erklärung zu generativer KI und KI-gestützten Technologien im Schreibprozess

Nach dem Programmieren benutzte die Autorin Claude Sonnet 4.5 (Gratisversion) zur online-Publikation des Programms, um die Web-App mit ihrem Python-Programm zu erstellen. Der Inhalt des Programms wurde in keiner Weise mit diesem Tool erstellt oder beeinflusst. Nach der Nutzung dieses Tools hat die Autorin den Inhalt überprüft und bearbeitet und übernimmt die volle Verantwortung für den Inhalt der veröffentlichten Arbeit.

Erklärung zu online-tools zur sprachlichen Unterstützung

Während der Vorbereitung dieser Arbeit benutzte die Autorin DeepL Translate (Gratisversion 25.13), um bestimmte Wörter oder Wendungen aus dem Französischen, ihrer Muttersprache, ins Deutsche zu übersetzen. Nach der Verwendung dieser Software hat die Autorin den Inhalt der Übersetzungen überprüft. Somit übernimmt sie die volle Verantwortung für den Inhalt dieser Maturitätsarbeit.

Abbildungsverzeichnis

Abb. 1: Gewöhnlicher Knoten und mathematischer Knoten., https://commons.wikimedia.org/wiki/File:Example_of_Knots.svg?uselang=de, Abruf: 29.06.2025 3
Abb. 2: Ein “wilder Knoten”., https://de.wikipedia.org/wiki/Knotentheorie, Abruf: 28.06.2025. 4
Abb. 3: Der Kleeblattknoten und der Achterknoten als Polygonzüge., https://www.uni-math.gwdg.de/schick/publ/KnotentheorieInDerSchule.pdf, Abruf: 29.06.2025 5
Abb. 4: Der Unknoten 0_1., https://fr.wikipedia.org/wiki/Circonf%C3%A9rence, Abruf: 02.09.25 5
Abb. 5: Zusammenhängende Summen von zwei Primknoten., https://www.researchgate.net/figure/Connected-sum-of-knots-A-knot-K-is-prime-if-K-K-1-K-2_-implies-K-1-O-or-K-2fig2339015556, Abruf: 29.06.2025 5
Abb. 6: Erlaubte und nicht erlaubte elementare Deformation., https://www.thi.uni-hannover.de/fileadmin/thi/abschlussarbeiten/heidari-ba.pdf, Abruf: 29.06.2025 . . . 6
Abb. 7: Projektion und Diagramm eines Kleeblattknotens., https://alphaknot.cent.uw.edu.pl/knot_detection, Abruf: 29.06.2025 7
Abb. 8: Reguläre Projektion eines Knotens., https://www.math.tecnico.ulisboa.pt/~ggranja/manuela.pdf, Abruf: 29.06.2025 7
Abb. 9: Orientierter Knoten., https://www.researchgate.net/figure/An-oriented-knot-diagram-D-1_fig5_336742408, Abruf: 29.06.2025 8
Abb. 10: Reidemeister-Bewegungen., https://www.researchgate.net/figure/Reidemeister-moves_fig1_2114003, Abruf: 01.07.2025 8
Abb. 11: Unknoten von Thistlethwaite., https://fr.m.wikipedia.org/wiki/Fichier:Thistlethwaite_unknot.svg, Abruf: 01.07.2025 9
Abb. 12: Knoten mit $c(K) = 1$., Eigenarbeit 10
Abb. 13: Knoten mit $c(K) = 2$, Fall 1., Eigenarbeit 11
Abb. 14: Knoten mit $c(K) = 2$, Fall 2., Eigenarbeit 11
Abb. 15: Knoten mit einer Entknotungszahl von 2., https://www.uni-math.gwdg.de/schick/publ/KnotentheorieInDerSchule.pdf, Abruf: 01.07.2025 11
Abb. 16: Dreifärbung des Knotens 6_1., Eigenarbeit 12
Abb. 17: Unerlaubte oder triviale Färbung des Achterknotens 4_1., Eigenarbeit 13
Abb. 18: Unmögliche Dreifärbung des Achterknotens 4_1., Eigenarbeit 13

Abb. 19: Einfluss von Reidemeister I auf der Dreifärbbarkeit., Eigenarbeit 14
Abb. 20: Einfluss von Reidemeister II auf der Dreifärbbarkeit., Eigenarbeit 14
Abb. 21: Einfluss von Reidemeister III auf der Dreifärbbarkeit 1., Eigenarbeit 15
Abb. 22: Einfluss von Reidemeister III auf der Dreifärbbarkeit 2., Eigenarbeit 15
Abb. 23: Vergleich der Dreifärbbarkeit mit dem Unknoten, dem Kleeblattknoten und dem Achterknoten., https://www.researchgate.net/figure/From-left-to-right-the-unknot-the-trefoil-knot-and-the-figure-eight-knot-The_fig6_220692408, Abruf: 01.07.2025 16
Abb. 24: Etikettierung $\mod 3$ des Kleeblattknotens., Eigenarbeit 17
Abb. 25: Reidemeister-Bewegung II auf einem etikettierten Diagramm., Eigenarbeit 18
Abb. 26: Etikettierung $\mod 5$ des Knotens 4_1., Eigenarbeit 18
Abb. 27: Etikettierung $\mod 7$ des Knotens 7_1., Eigenarbeit 18
Abb. 28: Knoten 5_2 mit den Variablen etikettierten Bögen., Eigenarbeit 19
Abb. 29: “Schachbrettmuster” der ε-Zeichen., Eigenarbeit 22
Abb. 30: Beweis der Spaltensumme gleich null: Fall 1 mit $n = 0$ gerade und überkreuzende Bögen in die gleiche Richtung., Eigenarbeit 22
Abb. 31: Beweis der Spaltensumme gleich null: Fall 2 mit $n = 0$ gerade und überkreuzende Bögen in die entgegengesetzte Richtung., Eigenarbeit 23
Abb. 32: Beweis der Spaltensumme gleich null: Fall 3 mit $n = 1$ ungerade und überkreuzende Bögen in die gleiche Richtung., Eigenarbeit 23
Abb. 33: Beweis der Spaltensumme gleich null: Fall 4 mit $n = 3$ ungerade und überkreuzende Bögen in die entgegengesetzte Richtung., Eigenarbeit 23
Abb. 34: Ausschnitt aus einem Knoten vor und nach der Ausführung einer Reidemeister - Bewegung I., Eigenarbeit 25
Abb. 35: Ausschnitt aus einem Knoten vor und nach der Ausführung einer Reidemeister - Bewegung II., Eigenarbeit 26
Abb. 36: Ausschnitt aus einem Knoten vor und nach der Ausführung einer Reidemeister - Bewegung III., Eigenarbeit 28
Abb. 37: Rechts- und linkshändige Kreuzung für das Alexander-Polynom., Eigenarbeit 32
Abb. 38: Orientierter und durchnummerierter Knoten 5_2., Eigenarbeit 32
Abb. 39: Gitterdiagramm des Knotens 7_5., https://knotinfo.org/diagram_display.php?7_5, Abruf: 13.09.2025 35
Abb. 40: Gesamter Aufbau des Programms., Eigenarbeit, Erstellt am 14.09.2025 auf der Website https://app.diagrams.net 37
Abb. 41: Aufbau der Funktion *feed_matrix*., Eigenarbeit, Erstellt am 13.09.2025 auf der Website https://app.diagrams.net 37

Literaturverzeichnis

Artikel

[1] Hermann Brunn, „Über verknotete Kurven,“ *Verhandlungen des Internationalen Math. Kongresses (Zürich 1897)*, Jg. 1, S. 256–259, 1897.

[2] Samantha Dixon, „Composite Knot Determinants,“ *University of Chicago, Chicago, Illinois*, 2010.

[3] Peter Freyd, David Yetter, Jim Hoste, W.B.R. Lickorish, Kenneth Millett und Adrian Ocneanu, „A new polynomial invariant of knots and links,“ *Bull. Amer. Math. Soc.*, Jg. 12, Apr. 1985. DOI: 10.1090/S0273-0979-1985-15361-3.

[4] Kate Horner, Mark Miller, Jonathan Steed und Paul Sutcliffe, „Knot Theory in Modern Chemistry,“ *The Royal Society of Chemistry*, 2013.

[5] Danielle O'Donnol, Andrzej Stasiak und Dorothy Buck, „Two convergent pathways of DNA knotting in replicating DNA molecules as revealed by θ-curve analysis,“ *Nucleic Acids Research*, 2018.

Bücher

[6] Michael Artin, *Algebra*. Birkhäuser Verlag, 1993.

[7] Charles Livingston, *Knotentheorie für Einsteiger*. Vieweg & Teubner, 1995.

[8] Dale Rolfsen, *Knots and Links*. AMS Chelsea Publishing, 2003.

Internetquellen

[9] Isha Agarwal und Victoria Li. „Knot Theory. “Adresse: https://math.mit.edu/research/highschool/primes/materials/2023/May/1-2%20Agarwal-Li.pdf. Abgerufen am 01.07.2025.

[10] Peter Cromwell. „Arc Index. “Adresse: https://knotinfo.org/descriptions/arc_index.html. Abgerufen am 13.09.2025.

[11] Philip Gibbs. „Is String Theory in Knots? “Adresse: https://arxiv.org/pdf/hep-th/9510042. Abgerufen am 27.10.2025.

[12] Erik Gregersen. „Michelson-Morley experiment. “Adresse: https://www.britannica.com/science/Michelson-Morley-experiment. Abgerufen am 28.06.2025.

[13] Bhavika Kalia, Aleah He und Laurelyn Newsome. „Seifert Surfaces. “Adresse: https://math.mit.edu/research/highschool/primes/materials/2024/May/6-3%20Carlos.pdf. Abgerufen am 27.10.2025.

[14] Mischa Lavrov. „Why in p-colorability, p must be a prime? “Adresse: https://math.stackexchange.com/questions/4669723/why-in-p-colorability-p-must-be-a-prime-knot-theory. Abgerufen am 02.07.2025.

[15] Ho Hon Leung. „Fundamental concepts of knot theory. “Adresse: https://pi.math.cornell.edu/~mec/2008-2009/HoHonLeung/page3_knots.htm. Abgerufen am 01.07.2025.

[16] Larsen Linov. „AN INTRODUCTION TO KNOT THEORY AND THE KNOT GROUP. “Adresse: https://math.uchicago.edu/~may/REU2014/REUPapers/Linov.pdf. Abgerufen am 27.10.2025.

[17] Charles Livingston und Allison H. Moore. „Grid Diagrams. “Adresse: https://knotinfo.org/descriptions/grid_notation.html. Abgerufen am 13.09.2025.

[18] Charles Livingston und Allison H. Moore. „KnotInfo: Table of Knot Invariants. “Adresse: https://knotinfo.org/results.php?searchmode=selectknot&desktopmode=0&mobilemode=0&singleknotprev=&submittype=selectknot&category%5B%5D=le8&name=%3D1&grid_notation=%3D1&arc_index=%3D1. Abgerufen am 13.09.2025.

[19] Robert Osserman. „Knot Theory. “Adresse: https://www.britannica.com/science/knot-theory. Abgerufen am 28.06.2025.

[20] Edward Witten. „Knots and Quantum Theory. “Adresse: https://www.ias.edu/ideas/2011/witten-knots-quantum-theory. Abegerufen am 27.10.2025.

Authentizitätserklärung

Ich bezeuge mit meiner Unterschrift, dass ich diese Arbeit selbstständig verfasst und erlaubte Hilfen von Drittpersonen korrekt veranschaulicht habe. Informationen, die ich wörtlich oder sinngemäss von anderen Publikationen oder Quellen, unter anderem aus dem Internet oder mithilfe künstlicher Intelligenz (KI), verfasst habe, sind klar, wiederauffindbar zitiert und machen nur einen geringen Anteil der Arbeit aus. Die in meiner Arbeit benützten Hilfsmittel entsprechen der Wahrheit, sind vollständig aufgelistet und auf diese Weise noch nicht veröffentlicht worden. Mir ist bewusst, dass ich unter Umständen zu den von mir gebrauchten Hilfsmitteln Stellung nehmen muss.

Ort und Datum: _____

Unterschrift der Schülerin: _____