

# Spezielle Relativität

Carl Jonas Mikaelsson Berggren

February 27, 2020

## Contents

<b>1</b>	<b>Relativität nach Newtonscher Physik</b>	<b>3</b>
1.1	Bezugssystem . . . . .	3
1.2	Wechsel von Bezugssystemen . . . . .	3
1.3	Minkoswsky Raumzeit diargamme . . . . .	4
<b>2</b>	<b>spezielle Relativität</b>	<b>4</b>
2.1	Herleitung . . . . .	4
2.2	Transformation zwischen Bezugssytmen . . . . .	5
2.3	Raumzeititnerval als erhaltene Größe . . . . .	6
2.4	Vierervektoren und Vierergeschwindigkeiten . . . . .	6
2.5	Implikationen . . . . .	6
2.6	Einchränkungen dieses Modells . . . . .	6
<b>3</b>	<b>Programm</b>	<b>7</b>
3.1	Nutzung . . . . .	7
3.2	Button . . . . .	8
3.2.1	Methoden . . . . .	8
3.3	Input . . . . .	8
3.3.1	Methoden . . . . .	9
3.4	Box . . . . .	9
3.4.1	Methoden . . . . .	9
3.5	Obj . . . . .	9
3.5.1	Methoden . . . . .	10
3.6	Funktionen . . . . .	10

4	Quellen	11
5	Notizen	11

## Abstract

In diesem Dokument erkläre ich, wie ich ein Programm entwickelt habe, was Albert Einsteins spezielle Relativität visualisiert. Das Programm arbeitet anhand von Minkowski-Raumzeitdiagrammen, und nutzt Objektorientierte Programmierung.

Außerdem erkläre ich die darunter liegende Physik und leite Lorentztransformation her. Dabei werde ich auf Relativität vor Einstein, auf Minkowski-Raumzeit-Diagramme, auf Transformationen zwischen Bezugssystemen, auf die Implikation Spezieller Relativität, und dessen Einschränkungen. Dabei beziehe ich mich nur auf spezielle Relativität und nehme inertielle Bezugssysteme in einer flachen Raumzeit an.

# 1 Relativität nach Newtonscher Physik

Bevor wir über Einsteins spezielle Relativität reden können, müssen wir das Konzept von Raum, Zeit und Bewegung klarstellen

## 1.1 Bezugssystem

Zunächst muss klar gestellt werden wie Position, Zeit und Geschwindigkeit gemessen werden. Dazu muss ein Koordinatensystem räumlich und zeitlich definiert werden. Das Koordinatensystem hat einen Ursprung mit  $x = 0, y = 0, z = 0$  und  $t = 0$ . Hierbei liegt der Ursprung des Koordinatensystems in der Regel auf ein Objekt zu Beginn des Beobachtungszeitraums bezogen. Durch die Tatsache, dass in allen Inertialen Bezugssystemen die gleichen physikalischen Gesetze gelten, sind alle Bezugssysteme gleich gültig. Es ist keine universal-gültige Aussage über die Position oder Geschwindigkeit eines Körpers, oder Zeitpunkt eines Ereignisses möglich. Demnach ist es bedeutungslos zu sagen, man hätte zum Zeitpunkt  $t$  die Position  $x, y, z$  und bewege sich mit einer Geschwindigkeit  $\vec{v}$ . Es muss immer ein Bezugspunkt gewählt werden z.B. Erdmittelpunkt, Upload des ersten http Dokuments. Bezugssysteme können sich also relativ zu eine ander bewegen und dennoch gleichermaßen gültig, das selbe Ereignis beschreiben.

## 1.2 Wechsel von Bezugssystemen

Ich werde mich im folgenden auf eine Raumdimension beschränke. Weitere Raumdimension können durch ersetzen der gerichteten Größen durch Vek-

toren hinzugefügt werden.  $x$  wird demnach zu  $0\vec{p}$ ,  $v$  zu  $\vec{v}$ . Dabei ist zu beachten, dass die gerichteten relativistischen Effekte nur entlang der Bewegungsrichtung auftreten.

Es wird zunächst ein Bezugssystem gewählt mit den Größen  $x, t$  und  $v$ . Anschließend wird ein gestrichenes Bezugssystem gewählt mit den Größen  $x', t'$  und  $v'$ , wobei  $t = t'$  gilt. Aus unserer alltäglichen Erfahrung geht hervor, dass für die Position eines, zu dem ungestrichenen System statischen Objekts gilt:  $x' = x - vt$  Genau so gilt für Geschwindigkeiten:

$$u' = u - v \quad (1)$$

Hierbei ist  $v$  die Geschwindigkeit des gestrichenen Bezugssystems und  $u$  Die Geschwindigkeit des betrachteten Objekts.

### 1.3 Minkowsky Raumzeit Diagramme

Das Minkowski Raumzeit Diagramm betrachtet, in seiner üblichen Form, Objekte in einer Raumdimension und Zeit. Hierzu wird die Zeit auf die vertikale Achse gelegt und die Position auf die horizontale. Für die Betrachtung von spezieller Relativität werden die Einheiten einfachheitshalber so gewählt, dass die Lichtgeschwindigkeit  $c = 1$  und  $x = t$ .

## 2 spezielle Relativität

Die Spezielle Relativität fügt ein entscheidendes Postulat hinzu:

- Die Lichtgeschwindigkeit ist eine universelle Konstante

Dies wirft direkt eine Frage auf: Wenn jemand mich mit einer Taschenlampe anleuchtet während ich mich auf ihn zu bewege, wie kann es dann sein, dass wir beiden den exakt gleichen Wert für die Geschwindigkeit dieses Lichts messen? Wenn eine einzelne Geschwindigkeit immer identisch sein kann, es nicht sein, dass Raum und Zeit erhaltene Größen sein.

### 2.1 Herleitung

Es werden zwei Bezugssysteme betrachtet, die sich relativ zu einander mit der Geschwindigkeit  $v \neq 0$  bewegen. In dem ungestrichenen Bezugssystem fliegt ein Photon zwischen zwei Spiegeln hin und her, die  $l$  voneinander entfernt

sind. Um von einem Spiegel zum anderen zu gelangen muss das Licht vom gestrichen System aus gesehen eine Größere Strecke  $d$  zurücklegen, nämlich:

$$d^2 = l^2 + (vt')^2 \quad (2)$$

Da  $d \neq l$  gilt, jedoch beide Strecken mit der gleichen Geschwindigkeit zurückgelegt werden, muss  $\Delta t' \neq \Delta t$  gelten. Drückt man  $d$  und  $l$  in  $c$  aus so gilt:

$$c^2 \Delta t'^2 = c^2 \Delta t^2 + v^2 \Delta t'^2 \quad (3)$$

Dies kann dann durch subtrahieren von  $v^2 \Delta t'^2$  um ausklammern von  $t'^2$  nach  $t'$  aufgelöst werden. Dann gilt:

$$\Delta t' = \frac{\Delta t}{\sqrt{1 - \frac{v^2}{c^2}}} \quad (4)$$

Teilt man nochmal durch  $\Delta t$ , ergibt sich der Relativitätsfaktor  $\gamma$ :

$$\gamma = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}} \quad (5)$$

## 2.2 Transformation zwischen Bezugssystemen

Dies löst aber nicht direkt die Frage der Geschwindigkeitsaddition. Das Postulat ist nämlich nicht mit der Formel 1 vereinbar. Geschwindigkeitsaddition muss daher neu definiert werden und zwar als:

$$v' = \frac{u + v}{1 + \frac{uv}{c^2}} \quad (6)$$

Es gelten außerdem:

$$t' = \gamma(t - \frac{vx}{c^2}) \quad (7)$$

$$x' = \gamma(x - vt) \quad (8)$$

$t$  und  $x$  beschreiben die Position und den Zeitpunkt eines Ereignisses. Hingegen beschreibt  $\Delta t$  aus Abschnitt 2.1 einen Zeitabstand.

## 2.3 Raumzeitintervall als erhaltene Größe

Nach der speziellen Relativität sind Raum, Zeit und sogar die Reihenfolge von Ereignissen relativ. Es wirkt so als nur die Lichtgeschwindigkeit konstant. Es gibt aber eine weitere Größe die unter Lorenztransformation invariant bleibt: Die sogenannte Eigenzeit

$$\tau^2 = (\Delta ct)^2 - \Delta x^2 - \Delta y^2 - \Delta z^2 \quad (9)$$

Wobei  $\Delta t$  der Zeitleihen und  $\Delta x, y, z$  den räumlichen Abstand zwischen zwei Ereignissen darstellen.  $\tau$  kann im Minkowski-Diagramm als Vektor dargestellt werden der die Ereignisse verbindet.

Diese Größe kann durch die Kombination aus negativen Raumkomponenten und positiver Zeitkomponenten im gesamten Reellen Bereich definiert werden. Was für ein Wert diese Größe hat hat auf die Wirkung darauf worüber sich Beobachter uneinig sein können.

Ist  $\tau$  negativ können sich Beobachter über die Reihenfolge der Ereignisse uneinig sein. Außerdem können sich die beiden Ereignisse nicht beeinflussen. Dies ist im Diagramm daran zu erkennen, dass der Winkel zwischen  $\tau$  und der  $ct$ -Achse kleiner als  $45^\circ$  ist. Somit sind auch Gleichzeitigkeitslinien zulässig die Steiler sind als  $\tau$ . Weltlinien die flacher sind als  $\tau$  sind jedoch unzulässig.

Gilt  $\tau = 0$  so kann das eine Ereignis das andere durch ein Lichtteilchen beeinflussen, jedoch nicht durch ein massebehaftetes Signal.

Ist  $\tau$  positiv ist die Reihenfolge der Ereignisse universell. In dem Fall entspricht  $\tau$  der Zeit die eine Uhr zwischen den Ereignissen messen würde, die sich von dem ersten zu den zweiten Ereignis bewegt.

## 2.4 Vierervektoren und Vierergeschwindigkeiten

## 2.5 Implikationen

## 2.6 Einschränkungen dieses Modells

Spezielle Relativität beschreibt nur inertielle Bezugssysteme in einer flachen Raumzeit.

Inertielle Bezugssysteme sind unbeschleunigt. Ein Ball der mit einer Geschwindigkeit gleich null losgelassen wird behält seine Position bei.

Dies löst zum Beispiel das bekannte Zwillingsparadoxon: Ein Zwilling fährt auf eine Raummission bei der er mit beispielsweise halber Lichtgeschwindigkeit

durch das All fliegt, während sein Bruder auf der Erde verbleibt. Als er zur Erde zurückkehrt stellt sich die Frage wer nun älter ist. Nach der Speziellen Relativitätstheorie können sich beide Zwillinge für den Zeitraum der Mission als statisch betrachten und den anderen als bewegt. Demnach kommen beide zudem zum Schluss, dass sie selbst jünger sein sollten.

Dabei wird jedoch vernachlässigt, dass der Astronaut beispielsweise zu Alpha Centauri fliegt, dort umkehrt und wiederkommt. Dabei erfährt er eine Beschleunigung, die die Spezielle Relativität nicht vorsieht.

Nehme man an, der Astronaut führe bei Alpha Centauri ein Swingby-Manöver durch und ändere so seine Richtung, so würde er sich nicht in einer flachen Raumzeitbewegung befinden.

Diese Überlegung führt uns in die Allgemeine Relativität, die sich maßgeblich von Newtons Gravitation unterscheidet. Gravitation wird nicht mehr als Kraftfeld beschrieben, sondern als Raumzeitkrümmung, die die inertielle Laufbahn von Objekten verändert.

## 3 Programm

Das Programm basiert auf objektorientierter Programmierung und nutzt das Pygame Modul für die Grafik. Ich habe insgesamt vier Klassen entwickelt: Eine für die Darstellung der Bezugssysteme im Diagramm, eine für die Darstellung von Buttons und zwei für die Darstellung der Eingabefelder. Außerdem habe ich jeweils eine Funktion zum Wechseln von Bezugssystem, und zum Handhaben von unzulässigen Nutzereingaben.

### 3.1 Nutzung

Dieses Programm kann in Python 2.7 ausgeführt werden. In dem rechten Feld können Geschwindigkeit und Startposition von Objekten eingegeben werden. Beim Klicken der Send-Taste wird die Weltlinie, und die dazu gehörige Gleichzeitigkeitslinie des Objektes in das Koordinatensystem eingezeichnet. Es können nach Belieben Objekte hinzugefügt oder entfernt werden. Alle sichtbaren Weltlinien können angeklickt werden, um die Lorentz-Transformation in das jeweilige Bezugssystem beobachten zu können.

## 3.2 Button

Die Klasse `Button` dient zur Darstellung und Handhabung aller buttons. Diese Klasse nimmt einen Typ und optional, einen parent als Argument.

### 3.2.1 Methoden

In dem Konstruktor wird die Box in Abhängigkeit von Typ definiert. Anschließend werden weitere Instanzvariablen, für Farbe, Aktivität, Text, Typ, Mutterobjekt definiert.

Die Methode `handle` wird in der Hauptschleife aufgerufen und nimmt pygame ereignisse als Argumente. In Abhängigkeit vom Typ wird der Ortsvektor der Maus angepasst. Wenn sich die Maus über ein dem button befindet wird der Aktivitätszustand auf True gesetzt, andernfalls auf False. Je nach Aktivitätszustand wird in der Methode `draw` die Farbe geändert, was ein Hoverfunktion darstellt. Befindet sich die Maus über `self`, wird dann bei einem Klick, zwischen den Typen unterschieden. Ist `self.type` gleich add, wird der Liste der `objs` eine Instanz von `Obj` hinzugefügt. Außerdem wird die Position von `self.rect` angepasst.

Ist `self.type` gleich send wird der index von `self` ermittelt, und der Methode `enter` übergeben, die in Abschnitt 3.3.1 näher erläutert wird.

Ist `self.type` gleich ok wird die globale Variable `err` None gesetzt.

Ist `self.type` gleich x wird der index von `self` ermittelt. Anschließend wird dieser Index genutzt, um die jeweiligen Einträge in den Listen `objs`, `delbuttons`, `inputs` und `sendbuttons`. anschließend werden die Position von den verbleibenden Elementen angepasst um das gelöschte Element aufzufüllen.

Die Methode `draw` wird in der Hauptschleife aufgerufen und zeichnet je nach Typ `self.rect` und `self.txt` auf die unterschiedlichen Flächen.

## 3.3 Input

Die Klasse `Input` erstellt Eingabefelder indem sie drei Instanzen der Klasse Box erstellt. Der Sinn dieser Klasse ist, die Liste der Eingabefelder besser zu organisieren damit Indizes leichter gehandhabt werden können.



### 3.3.1 Methoden

Diese Klasse hat bis auf den Konstruktor nur die Methode `enter`. Sie beim Klicken des 'send' Buttons aufgerufen. Sie prüft die Eingaben in dem jeweiligen Feldern auf Zulässigkeit. Es wird überprüft ob Parameter die Zahlen sein sollen Zahlen sind, und das keine Geschwindigkeit über der Lichtgeschwindigkeit eingegeben wird. Wird eine unzulässige Eingabe erkannt wird die Funktion `err` aufgerufen, die eine fallspezifische Fehlermeldung anzeigt. Sind Felder leer, werden Standardwerte ergänzt. Anschließend wird mit den ermittelten Parametern eine neue Instanz von `Obj` erstellt.

## 3.4 Box

Die Klasse `Box` ist die eigentliche Maschinerie hinter den Eingabefeldern. Je nachdem was für Parameter übergeben werden, wird `self` zum Namens-, Positions- oder Geschwindigkeitsfeld. Demnach werden Position, und Standardinhalt angepasst.

### 3.4.1 Methoden

Die Methode `handle` wird in der Hauptscheife aufgerufen, und nimmt pygameevents als Parameter. Wird das Feld angeklickt, wird der Standardinhalt entfernt und `self.active` gleich True gesetzt. Wird woanders hingeklickt, werden die Veränderungen wieder rückgängig gemacht. Demnach wird auch die Farbe des Feldes angepasst. Ist anschließend `self.active` True werden die Tastatureingaben gespeichert.

`draw` Zeigt dann den passenden Text und das Feld in der richtigen Farbe.

## 3.5 Obj

`Obj` ist das Herzstück des Programms. Es handhabt alle Einträge in das Diagramm. Es argumentiert, definiert den Namen, den Index, die Position, die Geschwindigkeit, den Farbindex und  $t = 0$  definieren. Es werden zunächst alle Argumente als Instanzvariablen gespeichert. Anschließend werden der Winkel der Weltlinien und die Weltlinie definiert. Anschließend wird `pos` definiert was das Errechnen von Punkten entlang der Weltlinie massiv vereinfacht. Falls der Betrag von `self.v` kleiner als 1 ist werden zusätzlich graphische Features definiert. Diese sind die Gleichzeitigkeitslinie für  $t=0$ , kleine Einheitmarker

für  $x$  und  $t$  und das Gesamte Koordinatensystem für das Jeweilige Bezugssystem.

Für die Gleichzeitigkeitslinie wird eine neue Linie erstellt die um  $90 - \text{self.angle}$  rotiert ist. Für die Einheitsmarker wird der Schnittpunkt zwischen den Beiden Geraden ermittelt. Von dort ausgehend werden iterativ neue Positionen für entlang der Geraden ermittelt die um eine Konstante  $/cdot\gamma$  von einander entfernt sind.

Ähnlich wird auch das Koordinatensystem definiert. Hier wird jedoch der Referenzpunkt der Geraden als Startpunkt gewählt.

Zuletzt wird die Position des Namens definiert. Der Name steht immer rechts neben der Weltlinie soweit oben wie es möglich ist ohne das zwei namen kollidieren.

### 3.5.1 Methoden

`draw` prüft ob `self.v` gleich 1 ist. Wenn das der Fall ist wird nur die Weltlinie gezeichnet. Andernfall werden mindestens die Einheitsmarker, und die Gleichzeitigkeitslinie hinzugefügt. Ist `self.active` wird zusätzlich auch das Gesamte Koordinatensystem gezeichnet.

`handle` handhabt das Nutzerverhalten in Bezug auf das Diagramm. befindet sich die Maus über der Weltlinie, wird `self.active` zu True. Wird die Weltlinie angeklickt, wird die Funktion `change` aufgerufen, und somit das wechseln des Bezugssystem initiiert. Wie das funktioniert ist in Abschnitt 3.6 näher erklärt.

## 3.6 Funktionen

Die Funktion `error` wird von der Funktion `enter` bei jedem Fehlerfall aufgerufen. Es wird ein fallspezifischer Fehlerwert übergeben. Die Funktion `error` erstellt ein Fenster auf dem eine fehlerspezifische Nachricht angezeigt wird. Außerdem wird ein 'ok' button erstellt mit dem das Fenster wieder geschlossen werden kann.

Die Funktion `change` wird aufgerufen wenn der Nutzer eine Weltlinie anklickt und danach in jeder Iteration des Programms, bis die Lorentztransformation fertig ist. Sie nimmt den Index des gewünschten Bezugssystems

als Argument. Die Änderung der Bezugssysteme wird der Anschaulichkeit halber für jede Größe nacheinander gemacht. Zuerst wird entlang der x-Achse verschoben, dann wird die Geschwindigkeit angepasst und zuletzt  $t$  justiert. Dies alles wird aber nicht in Schleifen in der Funktion gemacht, da so die Zwischenschritte zur fertigen Transformation nicht gezeichnet werden würde.

## 4 Quellen

Loedel-Minkowski-Diagramm; zweidimensionale Raumzeit (Zugriff: 06.09.2019):

<https://stackoverflow.com/questions/46390231/how-to-create-a-text-input-box-with-pygame>  
<https://www.youtube.com/playlist?list=PLD9DDFBDC338226CA>

## 5 Notizen

$x = ct$   $x' = ct - vt = (c-v)t'$  speed addition Raum, Zeit, Gleichzeitigkeit, Reihenfolge Invariante Proper time propertime, if  $\Delta x = 0$  they can be connected by a lightbeam spacelike, timelike lightcone Fourvector Fourvelocity Gleichzeitigkeit  $t = vx$  Linie der Gleichzeitigkeit ist die Spiegelung der Weltlinie entlang der Lichtlinie