Projektarbeit

Stratosphärenballon

Entwicklung einer Aufstiegssimulation und eines Programms zur Auswertung der Messdaten eines Stratosphärenballonflugs

Lorenzo Fahr

Projektkurs Physik

Schuljahr 2020/21

Inhaltsverzeichnis

1. Einleitung

Diese Arbeit ist im Rahmen eines Stratosphärenballonflugs entstanden und besteht aus zwei Blöcken. Im Ersten wird der Ballonaufstieg simuliert und mit den gemessenen Daten des Fluges verglichen, im Zweiten wird ein Programm entwickelt, mit dem die während des Fluges aufgezeichneten Messwerte grafisch dargestellt werden können.

1.1. Was ist ein Stratosphärenballon?

Ein Stratosphärenballon ist ein meist aus sehr elastischem Naturkautschuk bestehender Ballon, der mit Helium, Wasserstoff oder Ballongas gefüllt wird und so bis in mehr als 30.000 Metern Höhe steigen kann. Anbei trägt er meist eine Sonde mit verschiedenen Messgeräten, die während des Aufstiegs und Falls die Daten der Sensoren speichern, sodass sie später ausgewertet werden können. Zusätzlich können Kameras mitfliegen, die die Aussicht mit Fotos oder Videos dokumentieren.

Verwendung finden Stratosphärenballons häufig in der Meteorologie, um atmosphärische Messungen in Abhängigkeit der Höhe zu erstellen[1].

1.2. Konfiguration des Ballons

Der bei dieser Mission verwendete Ballon ist der *Wetterballon 1600* mit einer Nutzlast von bis zu 1.600 Gramm[2]. In der Sonde befinden sich der STRATO3-Datenlogger mit einem Außen- und Innentemperatursensor, einem Luftdruck- und Luftfeuchtigkeitssensor und Positionsbestimmung per GPS, ein Mikrocontroller mit Temperatur-, Neigungs- und Beschleunigungssensor, ein Geigerzähler, zwei GPS-Sender zur Ermittlung des Landeortes und zwei Kameras.

Als Füllmedium für den Ballon wird Ballongas verwendet, da Wasserstoff sehr reaktiv ist und deshalb gefährlich sein kann und Helium deutlich teurer ist. Da Ballongas nur mindestens 95% Helium[3] beinhaltet, hat es zwar eine 11% höhere Dichte, der Unterschied ist jedoch so gering, dass Ballongas hier ausreicht und der hohe Preisaufschlag keinen ausreichenden Nutzen hat. Die Füllmenge von 4,232 Kubikmeter wurde zuvor mit dem Heliumrechner[4] berechnet, um die notwendige Aufstiegsgeschwindigkeit von 5 Metern

[1] SCHÜTTLER et al, Leitfaden zum Aufstieg von Stratosphärenballons, 2017, S. 6
[2] STRATOFLIGHTS, Wetterballon 1600
[3] https://www.hochhinaus.de/wp-content/uploads/Produktdatenblatt_Ballongas.pdf
[4] https://www.stratoflights.com/tutorial/helium-rechner/

pro Sekunde zu erreichen, damit der Flugverkehr im Luftraum nicht unnötig gestört wird und zusätzlich die größtmögliche Flughöhe erreicht werden kann.

Teil I.

Simulation des Aufstiegs

2. Berechnung der Aufstiegsgeschwindigkeit

2.1. Erklärung der Formeln

Um die Steiggeschwindigkeit in Abhängigkeit von der Höhe zu berechnen, wird die Formel zur Geschwindigkeitsberechnung bei gleichmäßig beschleunigten Bewegungen genutzt. Da der Ballonaufstieg nicht einer gleichmäßig beschleunigten Bewegung entspricht, wird die Zeitdifferenz pro Intervall Δt so klein gewählt, dass die Bewegung darin annähernd gleichmäßig beschleunigt ist.

$$v = a \cdot \Delta t + v_0$$

$\Delta t =$ Zeitdifferenz
$a =$ Beschleunigung
$v_0 =$ Geschwindigkeit zum letzten Zeitpunkt

Zum Zeitpunkt $t = 0s$ beträgt die Geschwindigkeit $v = 0\frac{m}{s}$, da sich der Ballon noch auf dem Erdboden befindet und noch nicht fliegt. Die Beschleunigung a wird mit dem Newton´schen Gesetz berechnet, indem man dieses nach a umstellt:

$$\begin{aligned} & F = m \cdot a \\ \Leftrightarrow \quad & a = \frac{F}{m} \end{aligned}$$

Die Gesamtmasse des Ballons berechnet sich mit der Formel

$$m_{ges} = m_{Nutzlast} + m_{Ballon} + m_{Helium}$$

und beträgt bei diesem Flug $m = 3.807kg$. Dieser Wert bleibt bis zum Platzen konstant. Dabei berechnet sich die Masse des Heliums über die Zustandsgleichung idealer Gase und der Gleichung für die Teilchenanzahl:

$$\begin{aligned} & p \cdot V = N \cdot k \cdot T \\ \Leftrightarrow \quad & N_{Ballon} = \frac{V_{Helium} \cdot p_{Helium}}{k \cdot T} \\ & N_{Helium} = \frac{m_{Helium}}{M_{Helium}} \\ \Leftrightarrow \quad & m_{Helium} = N_{Helium} \cdot M_{Helium} \end{aligned}$$

$p_{Helium} = 101325Pa$
$k =$ Boltzmannkonstante
$M_{Helium} =$ Masse eines Heliumatoms $= 6.6464731 \cdot 10^{-27}kg$[1]

Die Kraft F entspricht der Gesamtkraft F_{ges}, die sich aus der Auftriebskraft F_A, der Gewichtskraft F_G und der Luftwiderstandskraft F_L mit folgender Formel berechnet:

$$F_{ges} = F_A - F_G - F_L$$

Da die zu erwartende Bewegung vertikal nach oben verläuft, werden die Kräfte nach oben hin mit positivem Vorzeichen dargestellt. Bewegungen in horizontaler Richtung beispielsweise durch Winde werden in dieser Berechnung vernachlässigt. Die Gewichts- und Luftwiderstandskraft, welche beide in Richtung Erdboden wirken, werden von der nach oben wirkenden Auftriebskraft subtrahiert. Die Gewichtskraft berechnet sich aus dem Produkt der Gesamtmasse m und des Ortsfaktors g.

$$F_G = m \cdot g \qquad \Big| m = 3.807kg$$

Mit steigender Höhe nimmt der Ortsfaktor leicht ab, da sich der Ballon vom Erdmittelpunkt entfernt und sich somit der Einfluss des Gravitationsfeldes auf den Ballon verringert. Den Ortsfaktor in Abhängigkeit von der Höhe berechnet man wie folgt:

$$g = \frac{G \cdot M_{Erde}}{r_{Erde}^2} = \frac{G \cdot M_{Erde}}{(r_{Erde} + h)^2}$$

$$\begin{aligned} G &= \text{Gravitationskonstante} \\ &= 6.673 \cdot 10^{-11} \frac{m^3}{kg \cdot s^2} \\ M_{Erde} &= \text{Masse Erde} \\ &= 5.97 \cdot 10^{24} kg \\ r_{Erde} &= \text{mittlerer Radius Erde} \\ &= 6371000m \end{aligned}$$

Die auf dem Gesetz des Archimedes beruhende Auftriebskraft F_A wirkt in Fluiden, so auch in Luft. Der Betrag dieser Kraft entspricht der Gewichtskraft des verdrängten Mediums[2]. Daher gilt folgender Zusammenhang:

$$F_A = F_{G,Medium} = \rho_{Medium} \cdot V_K \cdot g$$

$$\begin{aligned} F_{G,Medium} &= \text{Gewichtskraft des verdrängten Mediums} \\ \rho_{Medium} &= \text{Dichte des verdrängten Mediums} \\ V_K &= \text{Volumen des in das Medium eingetauchten Körpers} \end{aligned}$$

Die dritte, auf den Ballon wirkende Kraft ist die Luftwiderstandskraft F_L, die mithilfe des Widerstandsbeiwertes C_W berechnet wird. Dieser ist ein dimensionsloser Faktor, der das Verhältnis zwischen der Widerstandskraft und dem Staudruck des Körpers beschreibt, wenn er sich durch ein Fluid bewegt[3]. Für den verwendeten Wetterballon gilt $C_W =$

[2] https://www.leifiphysik.de/mechanik/druck-und-auftrieb/grundwissen/auftriebskraft
[3] https://www.grc.nasa.gov/www/k-12/airplane/dragco.html

0.25[4].

$$F_L = \frac{1}{2} \cdot C_W \cdot \rho_{Luft} \cdot A \cdot v^2 \qquad \left| \begin{array}{l} A = \text{Stirnfläche des Ballons} \\ \rho_{Luft} = \text{Dichte von Luft} \end{array} \right.$$

Mit steigender Höhe fällt der Luftdruck, was bedeutet, dass sich das Volumen des Ballons vergrößert und sich auch dessen Form verändert. Die zu Anfang tropfenförmige Form des Ballons verändert sich immer mehr in Richtung einer Kugel, wodurch sich auch der C_W-Wert verändert. Beim Platzen gilt also der Widerstandsbeiwert einer Kugel $C_W = 0.45$[5]. Der Verlauf lässt sich in Abhängigkeit von dem Volumen des Ballons in einem linearen Zusammenhang wie folgt modellieren:

$$C_W = \frac{1}{3} \cdot 10^{-3} \cdot V + 0.25$$

Das Volumen des Ballons wird berechnet, indem man die Zustandsgleichung idealer Gase nach V umstellt. Sie besagt, dass das Produkt aus Druck und Volumen dem Produkt aus der Teilchenanzahl N, der Boltzmannkonstante k und der Temperatur T in Kelvin entspricht.

$$\begin{array}{rl} & p \cdot V = N \cdot k \cdot T \\ \Leftrightarrow & V = \dfrac{N \cdot k \cdot T}{p} \end{array} \qquad \left| \begin{array}{l} k = 1.3806503 \cdot 10^{-23} \dfrac{J}{K} \\ [p] = 1Pa \end{array} \right.$$

Die Stirnfläche A des Ballons erhält man, wenn man mit dem Volumen des Ballons den Radius ausrechnet (2.1) und daraus die Querschnittsfläche (2.2) berechnet. Da die Form des Ballons einer Kugel ähnelt, gelten folgende Formeln:

$$V = \frac{4}{3} \cdot \pi \cdot r^3 \; ; \qquad \Leftrightarrow \qquad r = \sqrt[3]{\frac{3 \cdot V}{4 \cdot \pi}} \tag{2.1}$$

$$A = \pi \cdot r^2 \tag{2.2}$$

Die Dichte der Luft ρ_{Luft} wird mit dem Luftdruck p in Pascal und der Temperatur T in Kelvin berechnet[6]:

$$\rho_{Luft} = \frac{1000 \cdot p}{0.2869 \cdot T} \qquad \left| \begin{array}{l} [p] = 1Pa \\ [T] = 1K \end{array} \right.$$

Diese Formel ergibt sich aus der allgemeinen Formel der Dichte, die den Quotienten aus der Masse eines Stoffes und seines Volumens beschreibt[7]. Mit der Formel für die Teil-

[4] Dieser Wert lässt sich aus dem Quellcode des Heliumrechners von der Firma Stratoflights ablesen: siehe Anhang B

[5] http://www.dieter-heidorn.de/Physik/VS/Mechanik/K05_FallMitFL/K05_FallMitFL.html

[6] https://www.grc.nasa.gov/www/k-12/airplane/atmosmet.html

[7] IUPAC. Compendium of Chemical Terminology, 2. Version (the "Gold Book"). Zusammengestellt von A. D. McNaught und A. Wilkinson. Blackwell Scientific Publications, Oxford (1997). Online Version (2019-) erstellt von S. J. Chalk. ISBN 0-9678550-9-8. https://doi.org/10.1351/goldbook

chenanzahl ergibt sich folgender Zusammenhang:

$$\begin{aligned} \rho Luft &= \frac{m}{V} \\ \not{N}\Leftrightarrow &= \frac{m}{M}\rho Luft \sim \frac{N}{V} \end{aligned}$$

Die Zustandsgleichung idealer Gase nach $\frac{N}{V}$ umgestellt ergibt:

$$\begin{aligned} & p \cdot V = N \cdot k \cdot T \\ \Leftrightarrow \quad & \frac{N}{V} = \frac{p}{k \cdot T} \\ \Leftrightarrow \quad & \frac{N}{V} \sim \frac{p}{T} \end{aligned}$$

Daraus folgt:

$$\begin{aligned} & \rho Luft \sim \frac{N}{V} \sim \frac{p}{T} \\ \Leftrightarrow \quad & \rho Luft \sim \frac{p}{T} \end{aligned}$$

Der Druck und die Temperatur in Grad Celsius vom Erdboden bis an den oberen Rand der Stratosphäre können nicht in einer Funktion beschrieben werden, weshalb diese Formeln in drei Intervalle eingeteilt sind: die Troposphäre vom Erdboden bis in $11000m$ Höhe, die untere Stratosphäre von $11000m$ bis $25000m$ über dem Erdboden und ab $25000m$ die obere Stratosphäre[5].
Für $h < 11.000m$ gilt:

$$\begin{aligned} p &= 101290 \cdot \left(\frac{T + 273.15}{288.08}\right)^{5.256} \\ T &= 15.04 - 0.00649 \cdot h \end{aligned}$$

Für $11.000m < h < 25.000m$ gilt:

$$\begin{aligned} p &= 22650 \cdot \mathrm{e}^{(1.73 - 0.000157 \cdot h)} \\ T &= -56.46 \end{aligned}$$

Für $h > 25.000m$ gilt:

$$\begin{aligned} p &= 2488 \cdot \left(\frac{T + 273.15}{216.6}\right)^{-11.388} \\ T &= -131.21 + 0.00299 \cdot h \end{aligned}$$

Sowohl die Temperatur als auch der Luftdruck in der Atmosphäre lassen sich nicht ohne komplexe Rechenmodelle erklären, weshalb diese Formeln auf Erfahrungswerte und Messungen der nationalen Luft- und Raumfahrtagentur der Vereinigten Staaten von Ame-

rika (NASA) beruhen. Dennoch ist die Aussagekraft dieser Formeln als hoch einzustufen.

Nun kann die Höhe berechnet werden, indem die Höhe des letzten Zeitintervalls h_0 mit der zurückgelegten Strecke dieses Zeitintervalls summiert wird:

$$h = h_0 + v_0 \cdot \Delta t + \frac{1}{2} a \cdot \Delta t^2 \qquad \left| \begin{array}{l} v_0 = \text{Geschwindigkeit des letzten} \\ \quad \text{Zeitintervalls} \end{array} \right.$$

Für die Höhe zum Zeitpunkt $t = 0s$ gilt die Höhe des Startortes Lützenkirchen $h = 110m$.

2.2. Berechnung der Simulation

Für die Berechnung dieser Formeln wird – anstatt das Tabellenkalkulationsprogramm Excel zu verwenden – ein Pythonscript erstellt, um Änderungen in den Formeln unkompliziert auf die gesamte Berechnung anwenden zu können. Zusätzlich ist Python im Gegensatz zu Excel sehr ressourcenschonend und schnell in der Ausführung.

Zuerst werden die einzelnen Werte in eine sinnvolle Reihenfolge gebracht, um sie nacheinander berechnen zu können. Dafür müssen die Werte, die in der zu berechnenden Formel benötigt werden, bereits existieren. Die endgültige Reihenfolge ist im Anhang (A.1) zu finden. Um eine möglichst genaue Berechnung zu erhalten, sollte das Zeitintervall sehr klein gewählt werden. Tests zufolge eignet sich $\Delta t = 0.05s$ am besten, da in diesem Zeitraum die Bewegung als gleichmäßig beschleunigt betrachtet werden kann und die Menge an Daten vom Computer auch ohne Probleme verarbeitet werden kann. Zusätzlich werden mögliche Rundungsfehler stärker, je kleiner δt gewählt wird, was dafür sorgt, dass die Berechnung weniger genau wird.

Die Berechnung wird so lange ausgeführt, bis der Radius des Ballons $r = 5.25m$ überschritten hat, da das der maximalen Ausdehnung des Ballons entspricht[8]. Wird dieser Wert überschritten, platzt der Ballon und beginnt zu sinken.

[8] Dieser Wert lässt sich aus dem Quellcode des Heliumrechners von der Firma Stratoflights ablesen: siehe Anhang B

3. Datenanalyse

3.1. Erläuterung der berechneten Werte

Die Berechnung[1] zeigt eine anwachsende Steiggeschwindigkeit, die bei $v = 5.56\frac{m}{s}$ startet und bis zu einer Höhe von $31500m$ auf $9.6\frac{m}{s}$ durchgängig steigt. Danach sinkt die Geschwindigkeit leicht auf $9.47\frac{m}{s}$, bis der Ballon in einer Höhe von $34638m$ platzt.

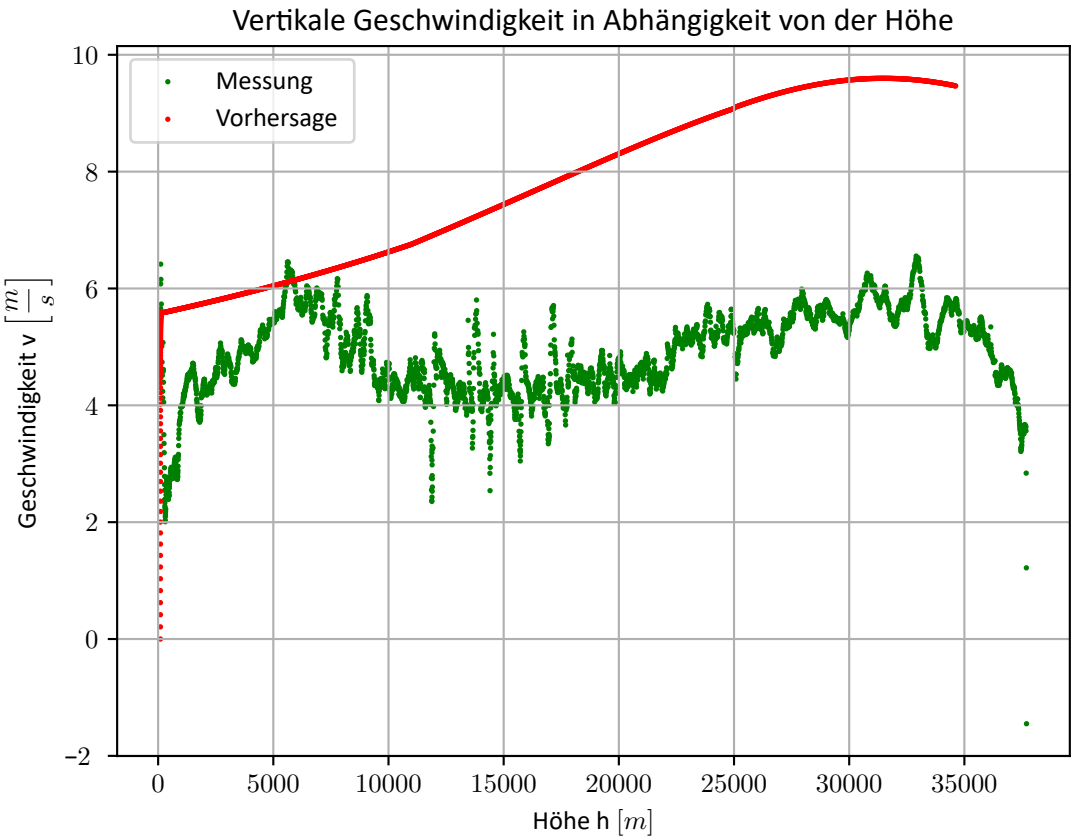

Abbildung 3.1.: Aufstiegsgeschwindigkeiten im Vergleich

Zu Beginn wird der Ballon stark beschleunigt, da für die Luftwiderstandskraft $F_L = 0N$ gilt, da die Geschwindigkeit $0\frac{m}{s}$ groß ist. Wird der Ballon nun schneller, wächst die Luftwiderstandskraft und der Ballon nimmt weniger schnell an Geschwindigkeit zu. Bis zu einer Höhe von $h = 150m$ fällt die Beschleunigung auf $a = 4.9 \cdot 10^{-4}\frac{m}{s^2}$ exponentiell ab und nähert sich asymptotisch $a = 4.8 \cdot 10^{-4}\frac{m}{s^2}$ an (A.2).

[1] Die Datei ist auf dem USB-Stick zu finden: `F_Datensätze/Simulation.csv`

Ab einer Höhe von $510m$ nimmt die Beschleunigung wieder zu und steigt annähernd linear auf $a = 9 \cdot 100^{-4} \frac{m}{s^2}$ in einer Höhe von $11000m$. Da die Temperatur aufgrund des Intervallübergangs zwischen den Formeln in der Berechnung (siehe Kapitel 2.1) einen 55-mal höheren Zuwachs pro Zeiteinheit hat, wenn $h = 11000m$ überschritten wird, steigt die Beschleunigung sprunghaft auf $12.3 \cdot 10^{-4} \frac{m}{s^2}$ an. Daraufhin steigt die Beschleunigung bis zum Maximum von $14.8 \cdot 10^{-4} \frac{m}{s^2}$ in einer Höhe von $19782m$. Mit steigender Höhe nimmt nun die Beschleunigung zügig ab und ab $h = 30600m$ wird die Beschleunigung negativ, sodass der Ballon an Geschwindigkeit verliert, bis dieser platzt (A.3).

3.2. Vergleich zu den gemessenen Daten

Zuerst fällt an den gemessenen Daten des *STRATO3*-Datenloggers[2] auf, dass der Ballon in einer Höhe von $37742m$ geplatzt ist, was 10.9% mehr ist als die Vorhersage berechnet hat. Zusätzlich ist auffällig, dass der Ballon mit einer geringeren Geschwindigkeit von maximal $5.9 \frac{m}{s}$ steigt[3], die bei $h = 6100m$ erreicht wird, bis $h = 14400m$ wieder auf $v = 4 \frac{m}{s}$ abnimmt und in einer Höhe von $32000m$ dieselbe Maximalgeschwindigkeit von $5.9 \frac{m}{s}$ wieder erreicht. Bis zum Platzen nimmt die Geschwindigkeit daraufhin deutlich ab (Abbildung 3.1).

Durch die zu hohe Geschwindigkeit in der Berechnung platzt der Ballon bereits nach $t = 4759s$, in der Wirklichkeit ist er allerdings erst nach $8258s$ geplatzt, was einer Abweichung von 80.3% entspricht.

Dennoch ist sowohl in der Berechnung als auch in der Aufzeichnung des Flugverlaufs eine Verringerung der Geschwindigkeit ab einer Höhe von $32000m$ erkennbar.

3.3. Wie lassen sich die Unterschiede erklären?

3.3.1. C_W-Wert

Eine mögliche Fehlerquelle bei der Berechnung der Aufstiegsgeschwindigkeit des Ballons könnte der C_W-Wert sein, da sich die Form des Ballons mit steigender Ausdehnung verändert. Der anfangs noch sehr stark wabernde Ballon, dessen Form der eines gestauchten Tropfens ähnelt, verändert seine Form durch Ausdehnung des Ballongases so, dass diese einer Kugel ähnelt, dessen breiteste Stelle allerdings etwas höher liegt, als bei einer Kugel. Dadurch ist die Stirnfläche weniger abgerundet und die aufprallende Luft stößt auf eine größere Fläche fast senkrecht auf den Ballon, sodass ein größerer Teil der Auftriebskraft des Ballons an die aufprallende Luftmasse abgegeben wird. Außerdem können aus zuvor laminaren Strömungen teilweise turbulente entstehen, die den Strömungswiderstand des Ballons erhöhen.

[2] Die Datei ist auf dem USB-Stick zu finden: `F_Datensätze/Messung_Strato3.LOG`

[3] Da der *STRATO3* keine Steiggeschwindigkeit aufzeichnet, werden diese aus der Höhenzunahme pro Zeiteinheit berechnet. Um starke Schwankungen zu vermeiden, wird die Geschwindigkeit über ein Zeitintervall von $\Delta t = 40s$ berechnet.

Für die Simulation bedeutet das, dass die Luftwiderstandskraft des Ballons größer ist als angenommen und somit die Gesamtkraft F_{ges} einen geringeren Wert besitzt. Die Beschleunigung ist also geringer und der Ballon steigt dadurch langsamer, die Geschwindigkeit im Flugverlauf ist also geringer und der Ballon benötigt mehr Zeit, bis er die Höhe erreicht, in der er schlussendlich platzt.

3.3.2. Temperatur

Eine weitere Fehlerquelle kann aus den Temperaturdaten entstanden sein. Durch die Aufteilung der Berechnung in drei Intervalle gibt es an den Intervallgrenzen einen geringfügigen Sprung zwischen dem letzten Temperaturwert eines Intervalls und des ersten des neuen Intervalls.

Der Intervallübergang bei $h = 11000m$ wird nach einer Flugzeit von $t = 1786.35s$ überschritten. Dabei steigt die Temperaturdifferenz pro Zeitintervall von zuvor $\Delta T = -0.002193°C$ auf $\Delta T = -0.110406°C$. Diese abrupte Temperaturänderung spiegelt sich sehr stark in der Beschleunigung des Ballons wider. Am Intervallübergang erhöht sich die Beschleunigung des Ballons um 37.5%, was um ein Vielfaches höher ist, als die übliche Veränderung von unter einem Prozent und Schwankungen von maximal 3% (A.4). Die Anpassung der Temperatur verursacht allerdings nur äußerst geringe, zu vernachlässigende Änderungen (A.5[4]).

3.3.3. Verwendung gemessener Daten des *Strato3* zur Simulation

Um möglichst genaue Temperatur- und Luftdruckwerte zur Berechnung heranzuziehen, ist es sinnvoll, die während des Fluges aufgezeichneten Messwerte zu verwenden. Da der Datenlogger *STRATO3* nur im Zwei-Sekunden-Rhythmus Werte aufzeichnet, werden diese Werte auf den 0.05-Sekunden-Rhythmus der Berechnung migriert. Dafür wird die Temperatur beziehungsweise der Luftdruck durch lineare Interpolation anteilig an der Höhendifferenz ermittelt.

Durch die an den Extrema etwa 20° Celsius höhere Temperatur (A.6) und den kaum veränderten, gemessenen Luftdruck (A.7) im Vergleich zu den berechneten Temperatur- und Luftdruckwerten ist der Ballon in der Spitze leicht schneller und platzt 1.5 Kilometer früher (Abbildung 3.2). Im Ganzen lässt sich sagen, dass sich der Kurvenverlauf nicht groß verändert hat. Die Kurve wurde lediglich etwas an der Höhenachse gestaucht.

Somit hat sich die Simulation weiter von der gemessenen Aufstiegsgeschwindigkeit entfernt und ist insofern kontraproduktiv.

[4] Die Änderung ist so gering, dass in dem Diagramm beide Kurven übereinander liegen, sodass nur eine erkennbar ist.

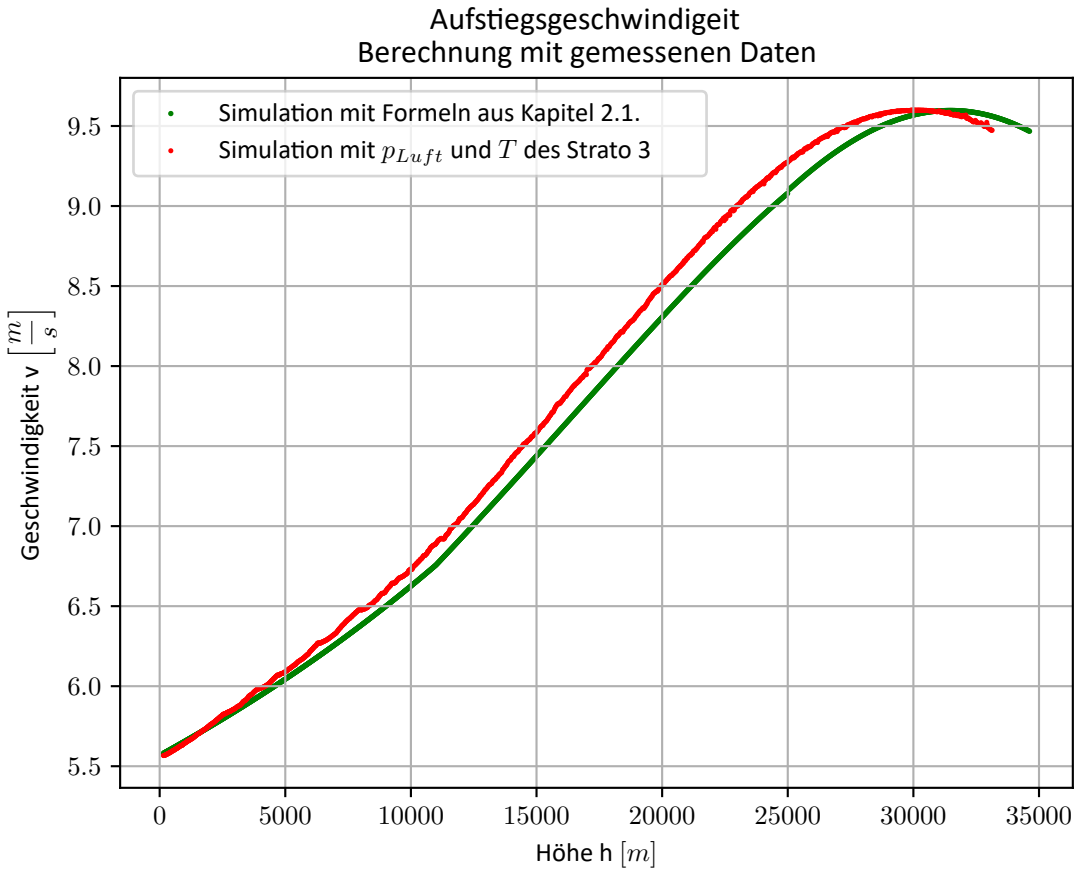

Abbildung 3.2.: Aufstiegsgeschwindigkeit berechnet mit Luftdruck und Temperatur vom *STRATO3*

3.3.4. Einfluss durch Winde

Durch vertikale Winde kann die relativ geringe Aufstiegsgeschwindigkeit des Ballons stark beeinflusst werden. Diese Winde können durchaus Geschwindigkeiten von mehreren Metern pro Sekunde erreichen, sodass deren Einfluss nicht vernachlässigt werden darf. Sichtbar ist der Wind auch in den Schwankungen der Aufstiegsgeschwindigkeit, die vom *STRATO3* aufgezeichnet wurde. So kann man davon ausgehen, dass zum Beispiel zwischen $9000m$ und $22000m$ Höhe Fallwinde denn Ballon abgebremst haben (3.1). Trotzdem kann dies nur schwer in der Berechnung berücksichtigt werden, da sowohl die Geschwindigkeit als auch die Richtung nur schwer vorhersagbar sind und sich innerhalb kurzer Zeit stark ändern können, da Faktoren wie Höhe des Ballons, Temperatur und Thermik durch Hoch- und Tiefdruckgebiete großen Einfluss auf die horizontale Bewegung großer Luftmassen haben.

4. Verlässlichkeit der GPS-Daten des *STRATO3*

Vor der Verwendung der GPS-Daten des *STRATO3*-Datenlogger ist es sinnvoll, die Verlässlichkeit dieser Daten zu überprüfen. Dafür können die Aufzeichnungen des *Spot Trace* GPS-Senders verwendet werden, die die Koordinaten mit den zugehörigen Höhenmessungen im 5-Minuten-Rhythmus beinhalten. Um dies zu prüfen, werden die Koordinaten und Höhenangaben beider Messgeräte zusammen in einem dreidimensionalen Koordinatensystem mit Längen- und Breitengrad auf der x-/y-Ebene und die Höhe auf der z-Achse dargestellt.

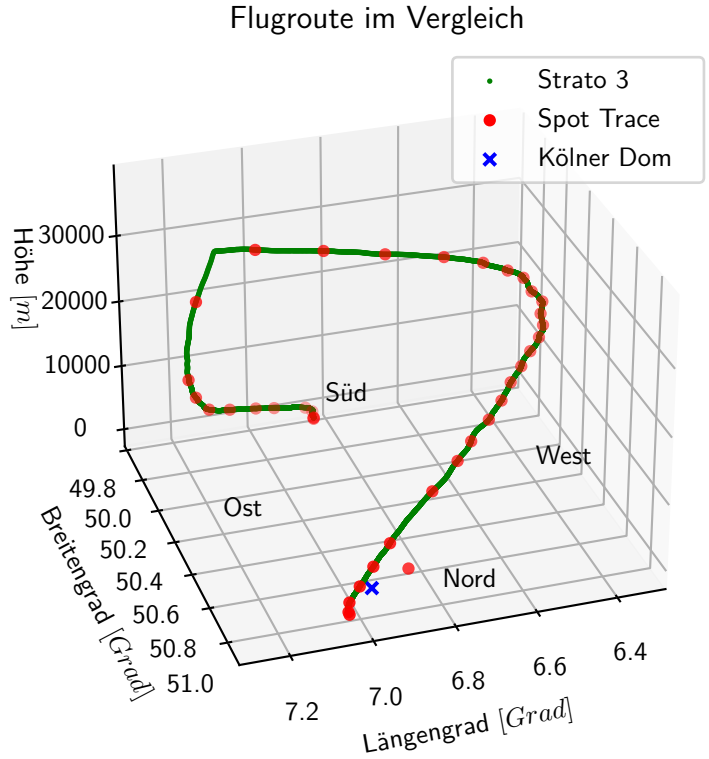

Abbildung 4.1.: Flugrouten vom *STRATO3* und *Spot Trace* im Vergleich - 3D Ansicht

Bei dieser Darstellung ist sehr gut erkennbar, dass beide GPS-Sender stets sehr ähnliche Daten aufgezeichnet haben. Ein Koordinatenpunkt des *Spot Trace* ist auf in der x-/y-Ebene mit $h = 0m$ aufgezeichnet, da dort keine Höhenwerte vorliegen. Infolgedessen kann den aufgezeichneten Koordinaten- und Höhendaten des *STRATO3* hohe Zuverlässigkeit zugesprochen werden.

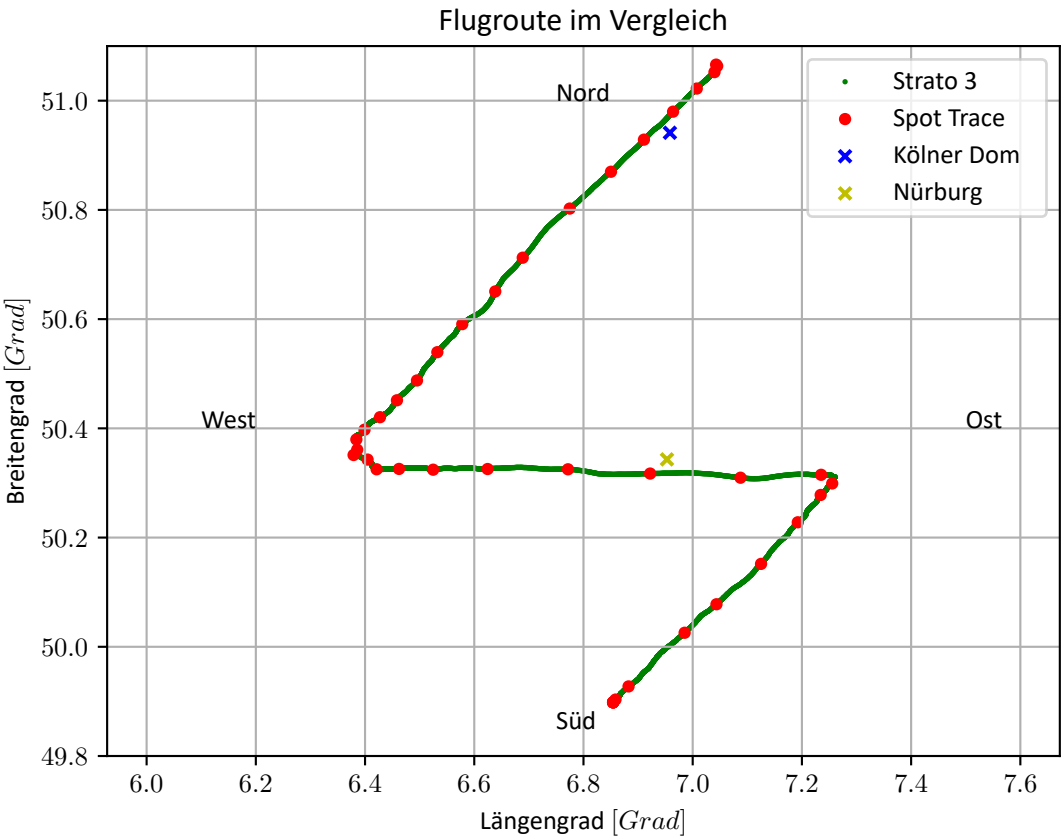

Abbildung 4.2.: Flugrouten vom *STRATO3* und *Spot Trace* im Vergleich - 2D Ansicht

Teil II.

Entwicklung eines Programms zur Auswertung der Flugdaten

5. Vorbereitung für weitere Flüge

5.1. Programm zur Auswertung der Messwerte

Um die Messwerte zu formatieren, die Simulation des Aufstiegs zu berechnen und diese Daten visuell in interaktiven Graphen darzustellen gibt es kein passendes Programm. Deshalb wurden von mir Skripte in der Programmiersprache Python geschrieben, die durch Eingabe einiger Parameter wie beispielsweise die Heliummenge, die Masse der Nutzlast und den Radius, bei dem der Wetterballon platzt, den Aufstieg simulieren.

Damit es möglich ist, diese Skripte auch bei weiteren Flügen verwenden zu können, ist es sinnvoll, die Skripte soweit zu verallgemeinern, dass man alle Parameter problemlos verändern kann. Dies ist besonders wichtig, da man davon ausgehen kann, dass auch Schüler dieses Tool verwenden möchten, die wenig bis keine Erfahrung mit Programmieren haben. Zudem wird die Verwendung übersichtlicher, wenn die Parameter gruppiert werden und vom Benutzer in nur eine Datei nach einem bestimmten Muster eingetragen werden und nicht in jedem einzelnen Skript an vielen verschiedenen Orten verändert werden müssen.

Dafür bietet sich die Struktur einer Library an, da dort keine Skripte selbst verändert werden, sondern nur Parameter an einzelne Funktionen übergeben werden. Eine Datei aus der Library enthält lediglich Funktionen, die aus einem externen Skript aufgerufen werden und selber nie verändert wird.

Durch die manuelle Eingabe der Parameter können sehr schnell Fehler auftreten, beispielsweise durch Verschreiben oder fehlerhafte Syntax. Zusätzlich ist das Anpassen der Skripte und die Fehlersuche bei nicht erfolgreich ausgeführtem Skript für Benutzer, die mit Python nur wenig vertraut sind, häufig schwierig und frustrierend.

Um dieser Fehlerquelle vorzubeugen, wird eine grafische Benutzeroberfläche[1] entwickelt, in der der Benutzer eine vorgefertigte Struktur ausfüllen muss und direkt geschaut wird, ob die Eingabe dem richtigen Format entspricht. So werden die meisten Fehler vermieden und es ist möglich, auf einer visuellen Ebene mit komplexen Systemen zu arbeiten, ohne Vorerfahrung zu benötigen. Zusätzlich ist man nur durch die Rechenleistung des eigenen Computers begrenzt, nicht durch Einschränkungen von Tabellenkalkulationsprogrammen wie Microsoft Excel. Mit dem selbst erstellten Programm kann man bis zu drei Datenreihen und so viele Datenpunkte in ein Diagramm einfügen, wie der Computer berechnen kann. Die Limitierung auf drei Datenreihen dient lediglich der Übersichtlichkeit im Programm. Sind mehr Datenreihen erwünscht, kann das sehr einfach geändert wer-

[1] Im Folgenden mit UI (*user interface*) abgekürzt

den.

5.2. Die Benutzeroberfläche

In der Benutzeroberfläche stehen drei Funktionen zur Verfügung: Formatierung der Rohdateien, Durchführung der Simulation und das grafische Darstellen von Messdaten und Simulationen.

5.2.1. Formatierung

Für die Formatierung muss der Name der Rohdatei, Speicherort und Name der formatierten Datei und das Trennzeichen, mit dem die Spalten getrennt werden sollen, angegeben werden. Soll die Datei in diesem Programm wieder eingelesen werden, um Graphen zu erstellen, muss das Trennzeichen ein Komma sein, für Excel empfiehlt sich ein Semikolon. Wird eine Datei vom *STRATO3*-Datenlogger formatiert, kann optional eine Start- und Landeuhrzeit angegeben werden, beim Arduino muss diese angegeben werden. Durch die gespeicherten Höhendaten vom *STRATO3* kann der Start- und Landezeitpunkt automatisch berechnet werden.

Wird nun auf *Run* geklickt, wird die Uhrzeit in Stunden umgerechnet, die Flugzeit in Sekunden und Koordinaten in Dezimalgrad, damit sie ein einheitliches Format in Form einer Dezimalzahl haben und so problemlos vom Computer verarbeitet werden können. Zusätzlich wird berechnet, wann der Ballon gestartet bzw. gelandet ist und alle Werte, bei denen der Ballon nicht in der Luft ist, entfernt. Daraufhin wird die Flugzeit angepasst, sodass der Ballon bei Sekunde 0 startet. Um kein Problem mit Messungen zu bekommen, bei denen der Datenlogger keinen GPS-Empfang hatte, werden diese Messungen entfernt.

Bei Messungen vom Arduino werden ebenfalls Uhrzeiten formatiert und Messfehler des Onboard-Temperatursensors entfernt, da dieser in seltenen Fällen Temperaturen größer als $500°C$ angibt. Zusätzlich werden auch hier alle Werte, bei denen der Ballon nicht fliegt, entfernt und die Zeit seit Ballonstart zu Anfang auf 0 gesetzt.

5.2.2. Simulation

Unter dem Reiter *Simulation* kann der Aufstieg simuliert werden. Hierfür wird die Berechnung aus Kapitel 2 verwendet. Wie in der Analyse dieser Simulation in Kapitel 3 verdeutlicht, ist der Verlauf der Geschwindigkeit zwar passend, der Ballon erhöht seine Geschwindigkeit im Verlauf jedoch deutlich stärker als er es in Wirklichkeit tut.

Aus diesem Grund gibt es hier die Möglichkeit, die Simulation rein physikalisch durchzuführen oder zusätzlich auf Erfahrungswerte zurückzugreifen, die die Simulation an die Werte des vorherigen Flugs anpassen.

Um die Simulation auf die individuelle Konfiguration des Stratosphärenballons anzupassen, müssen die verwendete Heliummenge und die Masse der Nutzlast inklusive Fall-

schirm und Schnur angegeben werden. Zuletzt muss angegeben werden, nach welchem Kriterium die Simulation beendet werden soll, ob dies vom Radius des Ballons oder der Höhe abhängig sein soll.

Wird die Auswahl mit *Run* bestätigt, wird die Simulation mit $\Delta t = 0.05s$ berechnet. Erfahrungen haben gezeigt, dass eine Berechnung mit $\Delta t < 0.05s$ deutlich mehr Zeit benötigt und keine genaueren Daten errechnen. Bei $\Delta t > 0.05s$ kann es sein, dass die Geschwindigkeit so stark schwankt, dass keine brauchbaren Werte berechnet werden.

Aus Sicherheitsgründen wird eine Berechnung, die mehr als 180.000 Zeilen beinhaltet, abgebrochen, um eine mögliche Endlosschleife zu verhindern. Ist der Ballon nach 150 Minuten noch nicht geplatzt oder hat er die eingegebene Höhe noch nicht erreicht, müssen die Eingaben noch einmal überprüft werden, da sie nicht sinnvoll sind.

5.2.3. Visuell darstellen

Hier kann der Benutzer die gemessenen und berechneten Daten in Graphen darstellen. Es können sowohl zweidimensionale als auch dreidimensionale Graphen erzeugt werden, die den gesamten Flug darstellen oder nur den Aufstieg.

Jede Datenreihe, die angezeigt werden soll, muss nun aktiviert und eingetragen werden. Dafür muss der Datentyp ausgewählt werden – ob vom *Strato3*, Arduino oder der Simulation, daraufhin die Datei und die Werte der verschiedenen Achsen. Es gilt zu beachten, dass die Datei mit diesem Programm vorher formatiert werden muss, damit sie das richtige Format besitzt. Nun werden visuelle Einstellungen getroffen und angegeben, ob man nur jeden x-ten Wert haben möchte. Als Letztes kann entschieden werden, ob die Werte verbunden oder einzeln dargestellt werden sollen.

6. Die Library

6.1. Grundstruktur der Library

Die für die Berechnung der Simulation wichtigen Konstanten[1] und die Eigenschaften der verschiedenen Datensätze von der Simulation, vom *STRATO3* und vom Arduino sind in einer eigenen Datei[2] gespeichert. Die Eigenschaften einer jeden Spalte der Datensätze sind in Dictionaries gespeichert. Diese beinhalten die Indices der einzelnen Spalten zusammen mit dem Namen, der beschreibt, was die Spalte beinhaltet und bei der Simulation die Formel und gegebenenfalls einen Startwert. Die Informationen aus diesen Dateien werden dann dort, wo sie benötigt werden, importiert.

In den Dateien mit dem Namen `typings.py` werden `Dataclasses` erstellt, die alle benötigten Parameter als ein Objekt übersichtlich übergeben.

Im Folgenden werden nur ausgewählte Teile des Codes der Library erklärt, da vieles lediglich auf Mathematik basiert. Die gezeigten Ausschnitte zeigen die Anwendung externer Libraries oder strukturelle Grundlagen zum Verständnis des Codes.

6.1.1. Lesen und schreiben von Dateien

Um den Inhalt von Dateien verwenden zu können, müssen diese erst einmal eingelesen werden. Damit diese dabei nicht beschädigt werden, ist es wichtig, sie nach dem Öffnen wieder zu schließen.

```
1 def readData(inputfilename: str):
2     with open(inputfilename, "r") as f:
3         raw = f.read()
4         return raw
```

Abbildung 6.1.: Lesen von Dateien

Dies wird automatisch mit der `with` Funktion gemacht. Der erste Parameter, der der Funktion `open()` übergeben wird, ist der Dateiname der einzulesenden Datei, der zweite gibt an, ob gelesen oder geschrieben werden soll. `"r"` zusammen mit `read()` steht für Lesen, `"w"` mit `write()` für Schreiben. Ist die Datei bereits beschrieben, wird der Inhalt überschrieben.

Um mit der eingelesenen Datei nun arbeiten zu können, muss der zurückgegebene String in eine zweidimensionale Liste aufgetrennt werden.

[1] `konstanten.py`

[2] `values.py`

```
1 def splitRaw(raw: str, delimiter: str):
2     data = []
3     for row in raw.split("\n"):
4         data.append(row.split(delimiter))
5     return data
```

Abbildung 6.2.: Auftrennen der eingelesenen Datei in eine Liste

Einzelne Zeilen einer CSV-Datei werden mit dem allgemeinen Trennzeichen `"\n "` getrennt, sodass dort mit `split()` getrennt wird, um die Zeilen als einzelne Strings zu bekommen. Diese wird daraufhin am Trennzeichen, meist `","`, ebenfalls getrennt, um in einer Zeile einzeln die Spalten auslesen zu können. Um in einer Tabelle t die vierte Zelle in der zweiten Zeile auszulesen, muss `t[1][3]` aufgerufen werden, da immer bei null angefangen wird zu zählen.

Bevor diese Liste wieder in eine Datei geschrieben werden kann, muss diese mit `join()` wieder zu einem String zusammengefügt werden.

6.1.2. Erstellung von Graphen

Zur grafischen Darstellung der Messwerte wird die Library `matplotlib` [3] verwendet. Mit dieser lassen sich Punktdiagramme mit `pyplot.scatter()` oder Liniendiagramme mit `pyplot.plot()` erstellen. Diese sind sowohl im zwei- als auch im dreidimensionalen darstellbar, indem die x, y und gegebenenfalls auch z Koordinaten als einzelne Listen gleicher Länge übergeben werden. Diese Listen werden in der `extractCoords()`-Funktion erstellt. Zusätzlich können Elemente wie ein Titel mit `pyplot.title()`, Achsenbezeichnungen mit beispielsweise `pyplot.xlabel()` oder eine Legende mit `pyplot.legend()` eingefügt werden. Dreidimensionale Graphen werden mit `pyplot.axes(projection='3d').plot()` beziehungsweise `pyplot.axes(projection='3d').scatter()` erstellt. Mit `pyplot.autoscale()` kann die automatische Skalierung der Achsen ein- beziehungsweise ausgeschaltet werden und `pyplot.figure().canvas.set_window_title()` legt den Titel des Fensters fest, indem der Graph dargestellt wird.

Sollen die Diagramme später in mit LaTeX erstellte Dokumente eingefügt werden, empfiehlt es sich, den Boolean `makePgf` auf `True` zu setzen. Dadurch wird der Graph in einer `pgf`-Datei gespeichert, die in ein LaTeX-Dokument integriert werden kann und automatisch intelligent angepasst und skaliert wird. Da dieses Feature aber nur für sehr wenige Personen von Interesse sein wird, ist diese Funktion, genauso wie die Änderung des Fenstertitels, nicht in der UI implementiert. Bei Bedarf kann dies jedoch problemlos hinzugefügt werden.

[3] Die Dokumentation dieser Library, aus der die Informationen entnommen werden, ist hier zu finden: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html

```python
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [4,2.4,1.6,5,3]

plt.scatter(x, y, label="Beispiel", color="r", s=[20 for _ in x])
plt.title("Beispieldiagramm")
plt.legend()
plt.grid(True)
plt.show()
```

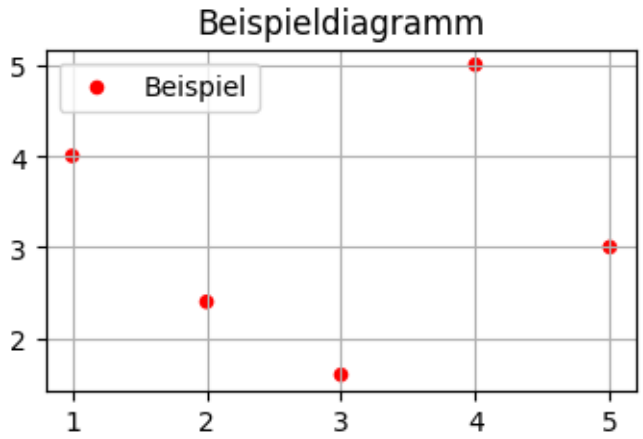

Abbildung 6.3.: Beispiel zur Erstellung von Graphen

6.2. Ausführen der Pythonskripte

Um ein Skript auszuführen, das in Python geschrieben ist, wird der Pythoninterpreter benötigt. Dieser Interpreter muss extra heruntergeladen und zu den Umgebungsvariablen hinzugefügt werden. Da das Programm zur Auswertung der Flugdaten allerdings unabhängig von anderen Programmen funktionieren soll, ist es notwendig, die Pythonskripte in eine Anwendung mit der Dateinamenserweiterung `.exe` zu konvertieren, was mithilfe des Package `pyinstaller` durch einen Befehl[4] im Terminal durchgeführt wird.

Sollen die Pythonskripte in Zukunft angepasst werden, ist es jedoch notwendig, den Pythoninterpreter[5] zu installieren, da eine Anwendung nicht bearbeitet werden kann. Änderungen müssen in den Pythonskripten durchgeführt werden, woraufhin die Anwendung neu kompiliert werden muss.

[4] `pyinstaller --onefile -w <filename.py>`
[5] Zu finden unter *www.python.org/downloads/*

7. Programmierung der Benutzeroberfläche

Zur Programmierung der Benutzeroberfläche wird die Programmiersprache Dart mit dem UI-Entwicklungskit Flutter verwendet. Dieses ermöglicht die Erstellung von Benutzeroberflächen in kurzer Zeit auf allen Plattformen, egal ob Windows, Mac, iOS oder Android. Dieses Programm läuft aktuell jedoch nur auf Windows, bei Bedarf kann es ohne viel Arbeit auf beispielsweise Mac umgeschrieben werden.

Aufgebaut ist die UI in drei verschiedene Tabs, die jeweils eine Funktion abdecken – Simulieren, Formatieren oder grafisch Darstellen. Jeder einzelne Tab ist in einer `ListView()`, also einer Liste dargestellt, die die Zeilen, die mit `ListTile()` erstellt wurden, enthält. Diesen Zeilen werden dann als `children` beispielsweise Textboxen, Dropdowns oder Schaltflächen zum Auswählen von Datei oder Ausführen der Aktion übergeben. Außerdem gibt es spezielle Zeilen mit Checkboxes oder Radio-Auswahlbuttons, mit denen man entweder mehrere oder nur eine Auswahl tätigen kann. An einigen Stellen finden sich Tooltips, die Hilfen bei der Auswahl geben, wenn mit der Maus darüber gefahren wird.

Um Fehler vorzubeugen, ist der Button, mit dem die Auswahl bestätigt und das Programm ausgeführt wird erst klickbar, wenn alle notwendigen Eingaben getätigt wurden. Außerdem sind bei einigen Textfeldern nur Eingaben aus Zahlen möglich oder die Anzahl der einzugebenden Zeichen begrenzt. So beispielsweise bei dem Trennzeichen, das nur ein Zeichen betragen darf. Standardmäßig ist dort ein Komma ausgewählt, da dies bei weiterer Verarbeitung mit diesem Programm notwendig ist.

Die gesamte Dateistruktur der Benutzeroberfläche befindet sich auf dem USB-Stick im Ordner `E_Flutter_UI`. Der selbstgeschriebene Code befindet sich hier: `E_Flutter_UI/gui/lib/`

8. Kommunikation zwischen Front- und Backend

Damit beim Drücken des *Run*-Butttons die `main.exe` ausgeführt wird und alle notwendigen Parameter übergeben werden, kommuniziert die Flutter UI über die Standarddatenströme mit der `main.exe`. Diese sind Kommunikationswege über die Kommandozeile, die über drei verschieden Datenströme verlaufen, die Standardeingabe, Standardausgabe und die Standardfehlerausgabe.

Die UI führt die `main.exe` aus, welche daraufhin auf die Eingabe `command` wartet, die über die Standardeingabe `stdin` übergeben wird. Diese Eingabe ist ein String, der alle Parameter durch einen Separator getrennt beinhaltet. Die Executable nimmt diesen String an und trennt ihn wieder am Separator, sodass die einzelnen Elemente dieser Liste der auszuführenden Funktionen übergeben werden. Das erste Element dieser Liste gibt an, ob der Aufstieg simuliert, eine Datei formatiert oder ein Graph erstellt werden soll, die restlichen Elemente sind Argumente für die aufgerufene Funktion.

```
1 void run() async {
2     var res = await Process.start("main.exe", []);
3     res.stdin.writeln(command);
4     res.stdout.transform(utf8.decoder).forEach(standardout);
5     res.stderr.transform(utf8.decoder).forEach(standarterror);
6     }
```

Abbildung 8.1.: Kommunikation auf der Seite der UI

Über die Standardausgabe `stdout` gibt die `main.exe` über die Funktion `print()` Informationen an die UI zum Beispiel über den Status der Aktion zurück. Diese werden dem Benutzer in der UI unterhalb des *Run*-Buttons dargestellt, um beispielsweise falsche Eingaben erkennen zu können. Gravierende Fehler, die das Programm zum Abstürzen bringen, werden über die Standardfehlerausgabe `stderr` an die UI weitergegeben, sodass der Benutzer die Möglichkeit hat, die Fehler zu erkennen und das Programm zu debuggen.

Anhang

A. Diagramme

Die Diagramme sind auch als interaktive Graphen auf dem USB-Stick zu finden.

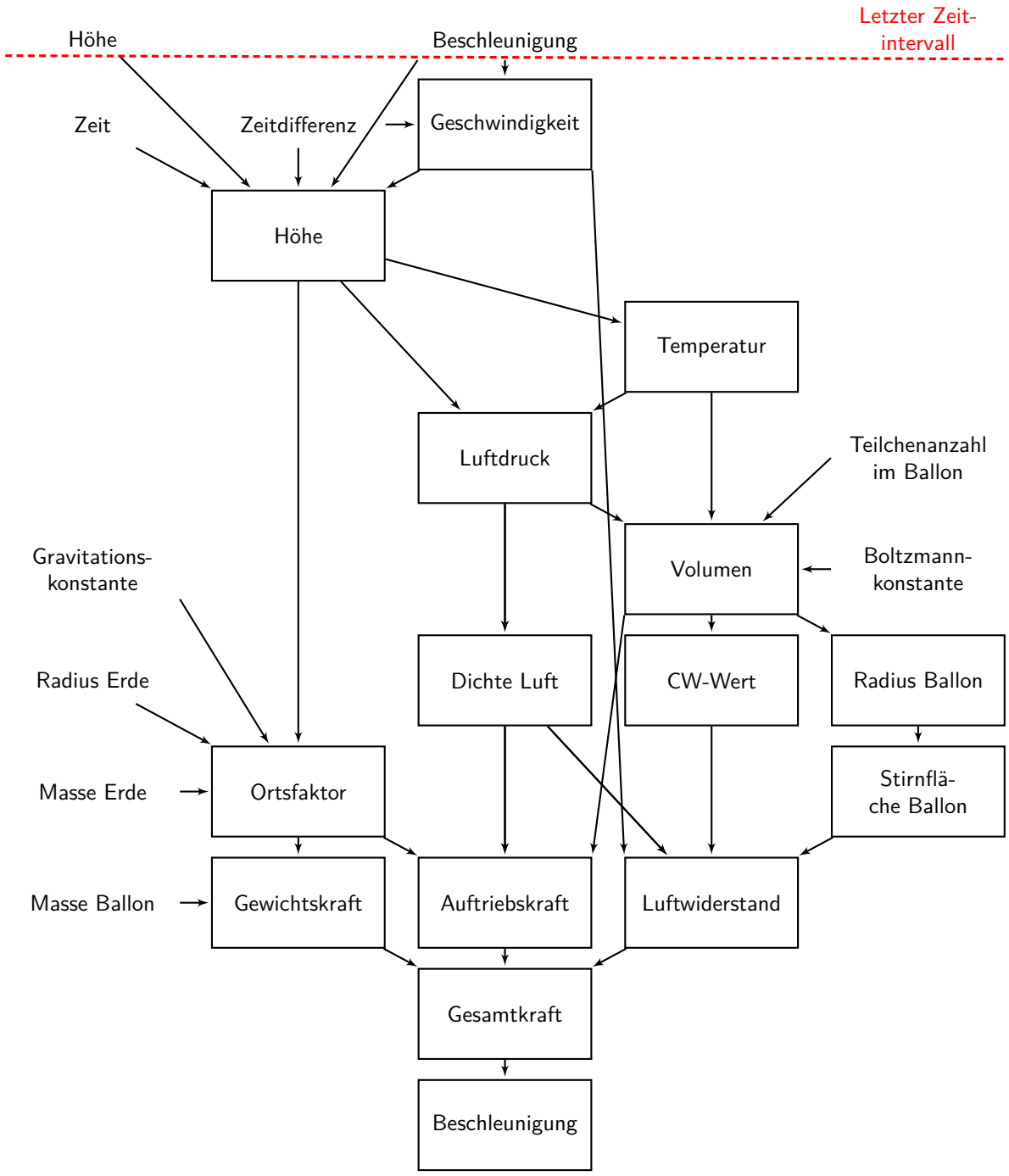

Abbildung A.1.: Reihenfolge der Berechnung in der Aufstiegssimulation

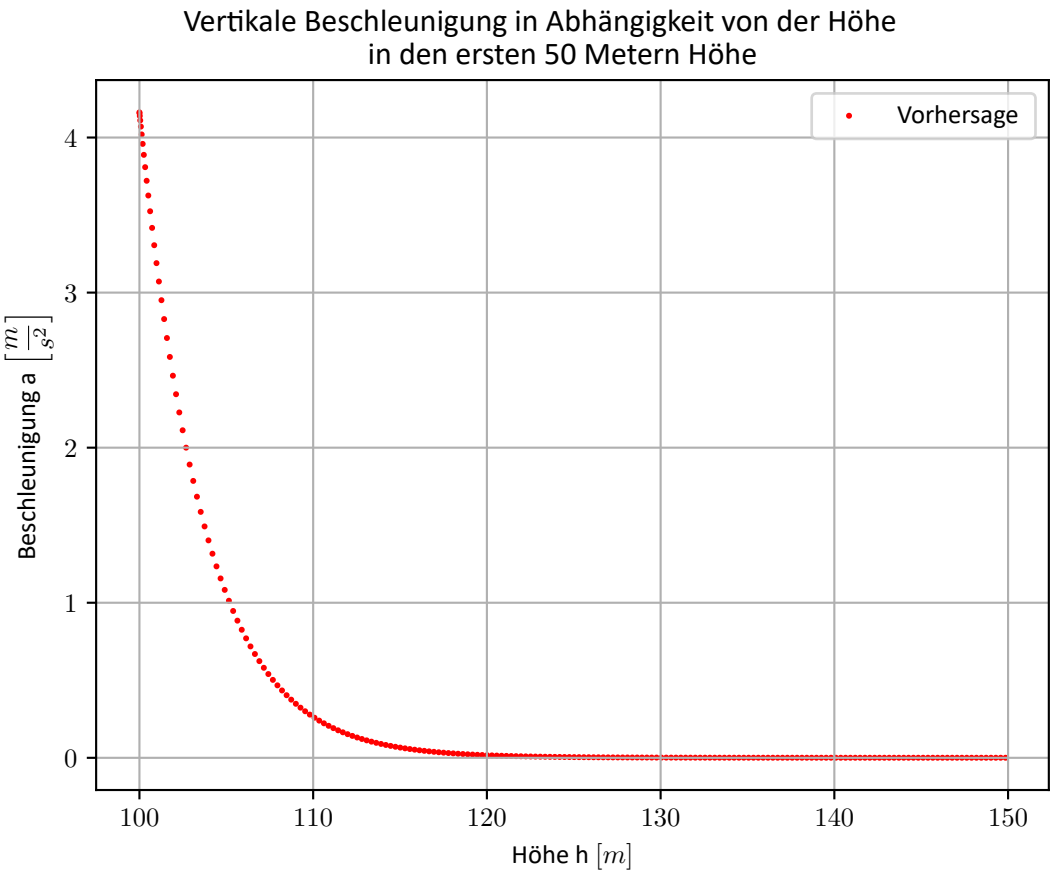

Abbildung A.2.: Beschleunigung des Ballons in den ersten $50m$ Höhe

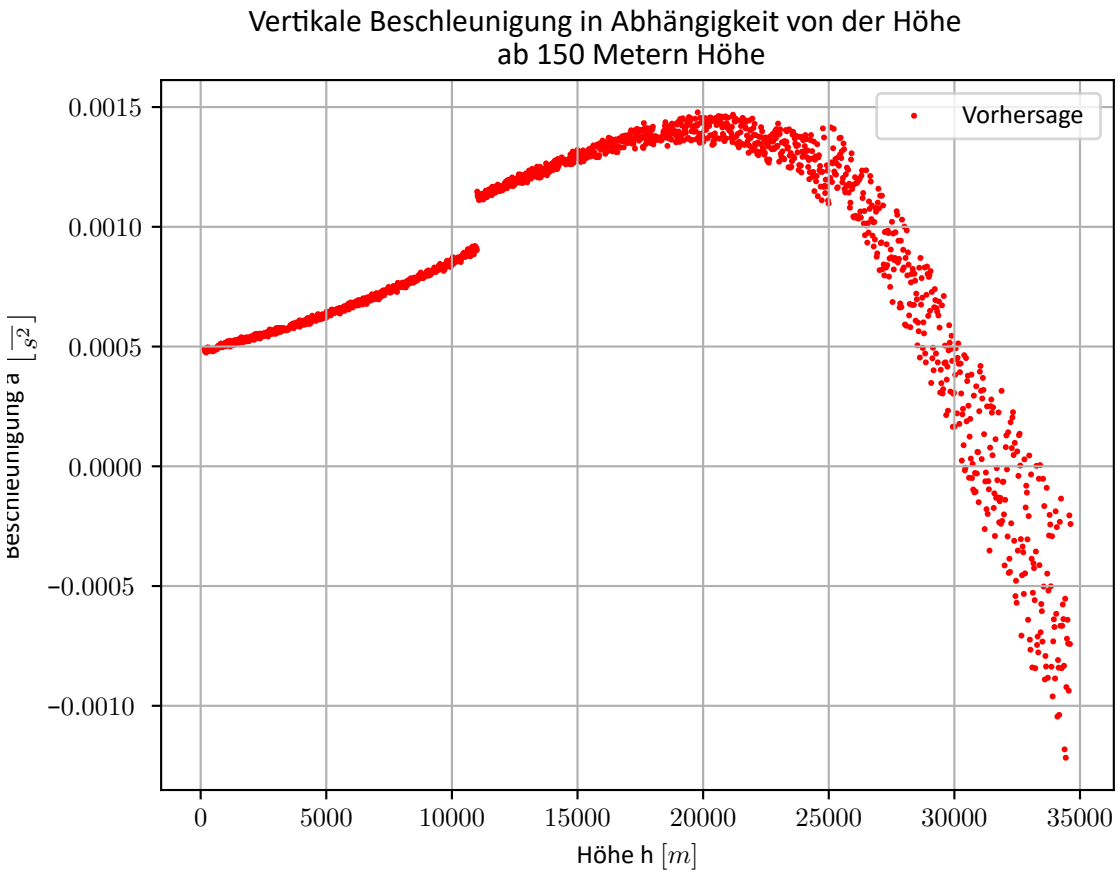

Abbildung A.3.: Beschleunigung des Ballons ab $150m$ Höhe

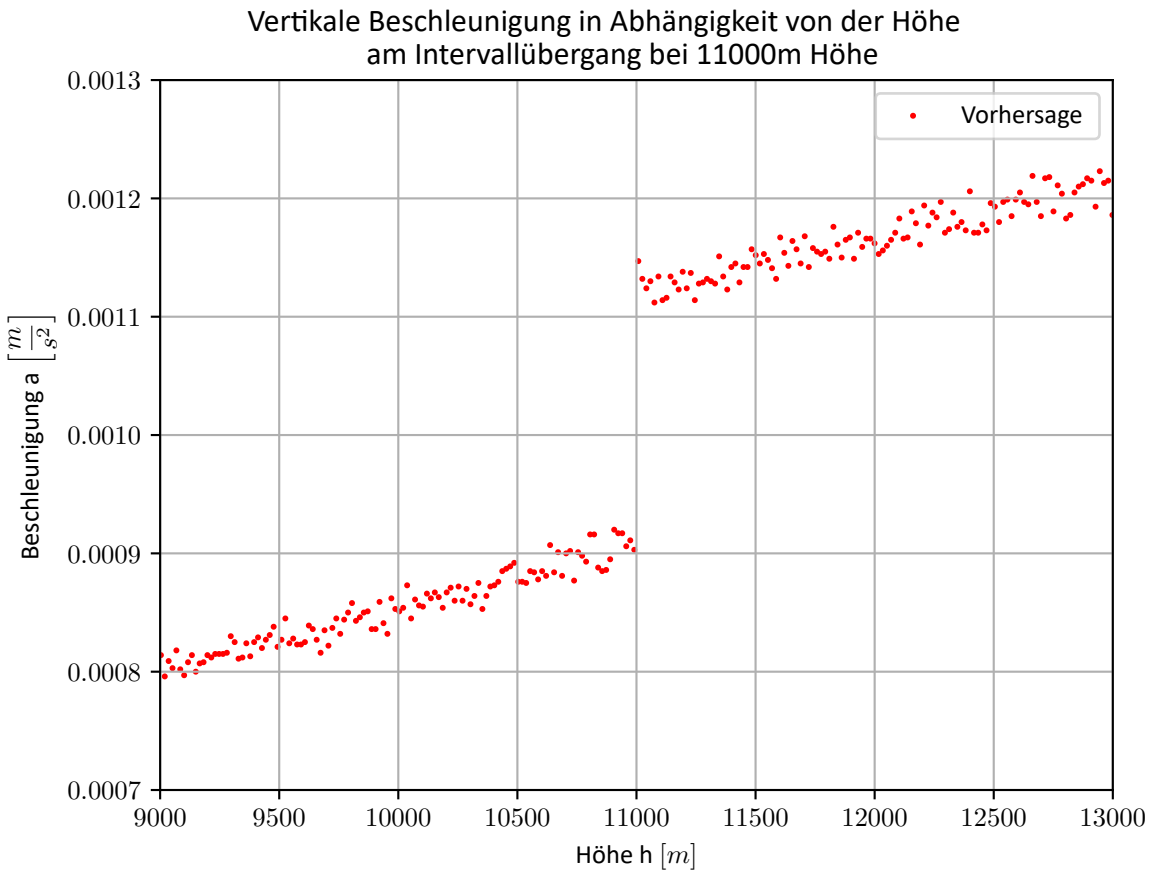

Abbildung A.4.: Beschleunigung des Ballons am Intervallübergang $h = 11000m$

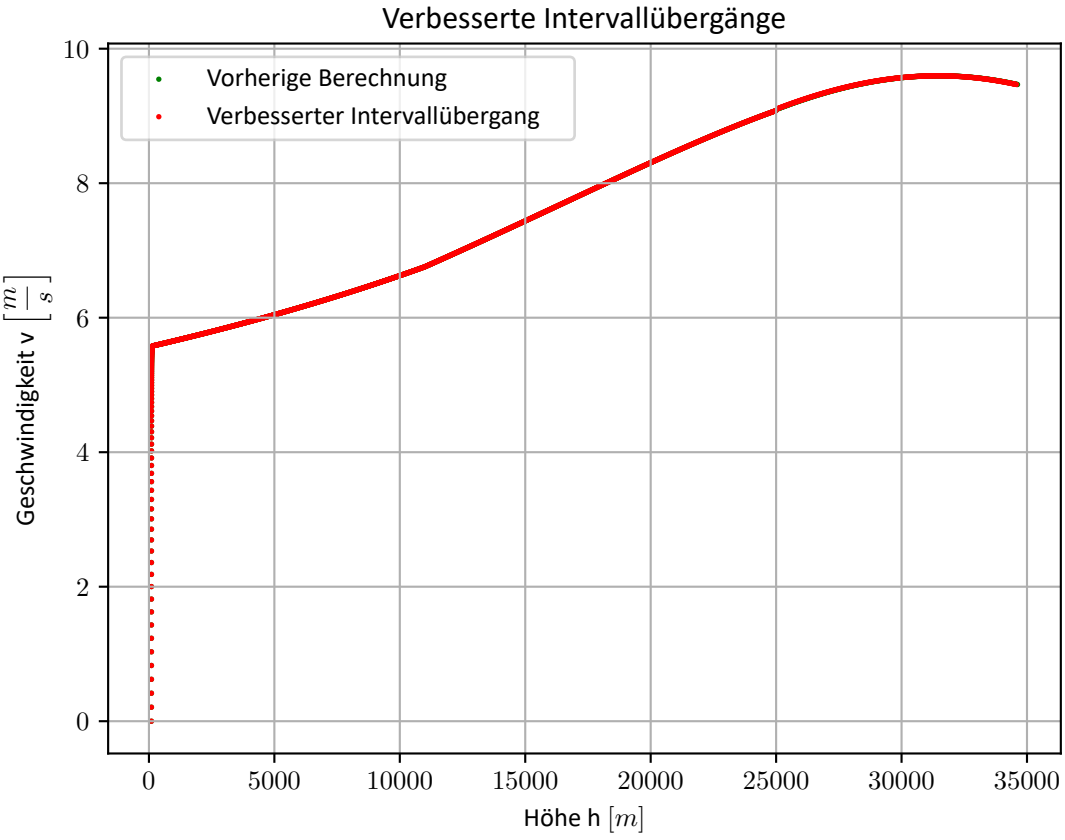

Abbildung A.5.: Geschwindigkeitsverlauf bei angepasster Temperatur im Vergleich. Die Änderung ist so gering, dass in dem Diagramm beide Kurven übereinander liegen, sodass nur eine erkennbar ist.

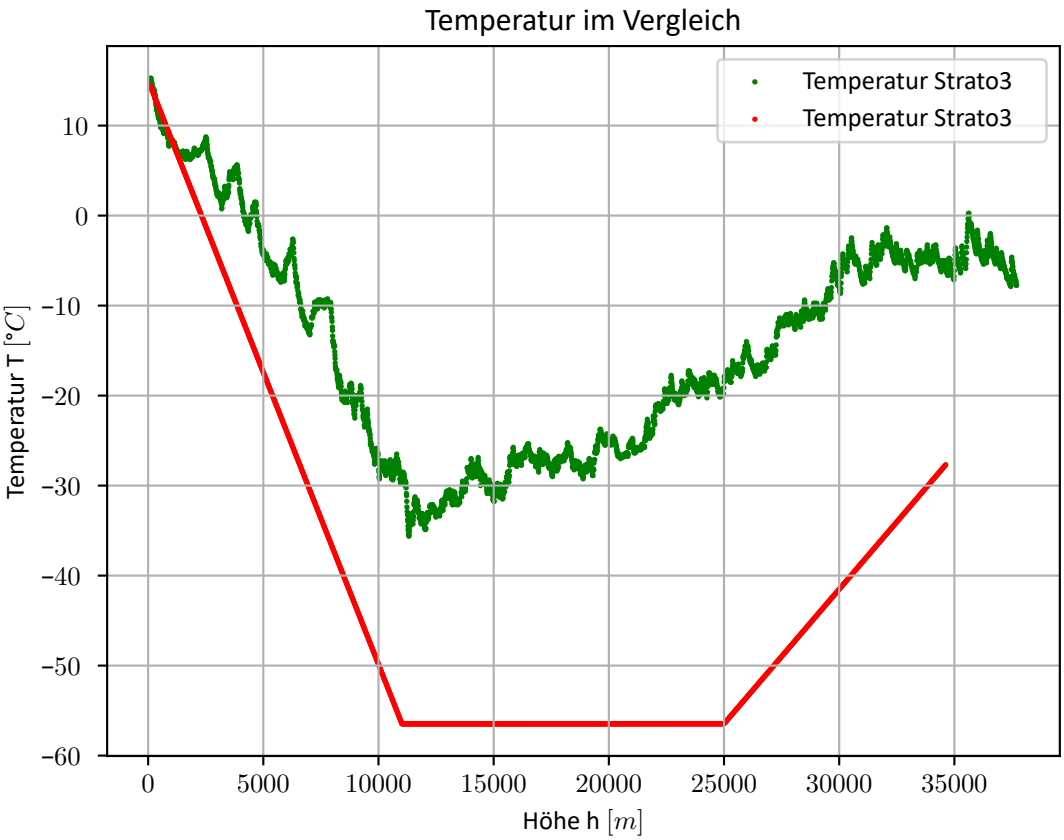

Abbildung A.6.: Gemessene und berechnete Temperatur im Vergleich

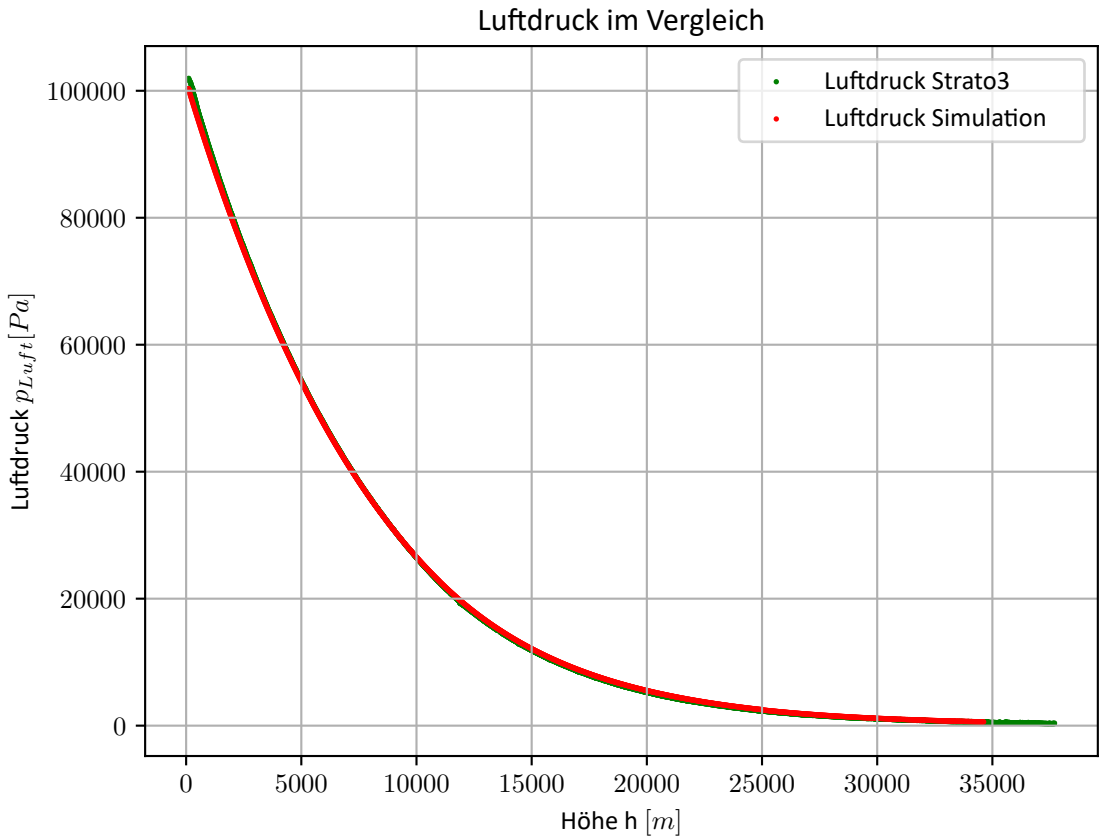

Abbildung A.7.: Gemessener und berechneter Luftdruck im Vergleich

B. Quellcode Heliumrechner

Der folgende Code ist ein stark gekürzter Auszug aus dem Quellcode des Heliumrechners[1] der Firma *Stratoflights*. Es werden nur für diese Arbeit notwendige Codezeilen kommentiert gezeigt. Besonderes Augenmerk wird auf die Daten des Wetterballons gelegt.

Der ungekürzte Code ist auf dem USB-Speicherstick zu finden.

```
function find_gas(gas) {
    var gas = new Array();
    gas["he"] = 0.1786;
    gas["bg"] = 0.1855; // Dichte von Ballongas
    var g;
    g = gas[jQuery('#gas_rho').val()];
    return g;
}

// Hier wird der Durchmesser beim Platzen für den ausgewählten Wetterballon
↪  festgesetzt. Der Wetterballon 1600 ist hier mit "h1600" gekennzeichnet.
function find_bd(mb) {
    var bds = new Array();
    bds["h200"] = 3.00;
    bds["h800"] = 6.80;
    bds["h1600"] = 10.50;
    bds["h2000"] = 11.00;
    bds["h3000"] = 12.50;
    var bd;
    bd = bds[jQuery('#mb').val()];
    return bd;
}

// Hier wird der C_W-Wert für den ausgewählten Wetterballon festgesetzt.
function find_cd(mb) {
    var cds = new Array();
    cds["h200"] = 0.25;
    cds["h800"] = 0.30;
    cds["h1600"] = 0.25;
    cds["h2000"] = 0.25;
    cds["h3000"] = 0.25;
    var cd;
    cd = cds[jQuery('#mb').val()];
    return cd;
}
```

[1] https://www.stratoflights.com/tutorial/helium-rechner/

C. Datenauswertungsprogramm

Auf dem USB-Stick befindet sich im Ordner `C_Datenauswertungsprogramm` das entwickelte Programm. Hierbei gilt es zu beachten, dass die Ordner- und Dateistruktur in diesem Ordner nicht verändert werden darf, da ansonsten benötigte Dateien zur Ausführung des Programms nicht gefunden werden können.

Zur Ausführung des Programms muss die Datei `Datenauswertung.exe` gestartet werden. Gegebenenfalls wird vom Windows Defender oder einem anderen Antivirusprogramm davor gewarnt die Datei auszuführen, da das Programm unbekannt ist. Um es dennoch auszuführen, muss man auf *Weitere Informationen* und dann auf *Trotzdem ausführen* klicken.

Erklärung

Ich erkläre, dass ich die Projektarbeit ohne fremde Hilfe angefertigt und nur die angeführten Quellen und Hilfsmittel benutzt habe.

.....	.....
Ort, Datum	Lorenzo Fahr