

Albrecht-Dürer-Gymnasium Hagen

Wie bestimmt man die Umlaufbahn und die Position auf der Internationalen Raumstation?

Ein Versuch im Rahmen der European Astro Pi Challenge 2020

Projektarbeit im L^AT_EX-Kurs
der Q1 bei Hr. Dr. Müller,
vorgelegt von Lucas Schleuß,
Fleyer Str. 83, 58097 Hagen,
am 25. Mai 2020.

Vorwort

Im Herbst 2019 schrieb die Europäische Weltraumorganisation (ESA) einen Wettbewerb aus, bei dem man mit der Programmiersprache Python Experimente auf einem Minicomputer mit dem Namen RaspberryPi einreichen konnte. Die besten Programme wurden auf die Internationale Raumstation (ISS) gesendet und durften dort drei Stunden lang ausgeführt werden. Die Ergebnisse aus dem erdnahen Weltraum konnten dann ausgewertet werden. Da ich gerne mit Python programmiere und mich zudem sehr für die Raumfahrt interessiere, beschloss ich, mich mit Jannik Mänzer und meinem Onkel Torben Heßling zusammen als Team MILLIWAYS REGULARS¹ einzuschreiben und als Experiment die Position der ISS zu bestimmen. Dafür haben wir Ende Oktober unsere Experimentidee abgeschickt und dann von November bis Februar das Programm entwickelt. Am 02.04.20 gab es endlich die Rückmeldung. Unser Experiment durfte auf der Internationalen Raumstation durchgeführt werden! Das Programm lief am 21.04.20 und umrundete dabei etwa zweimal die Erde. Am 18.05.20 bekamen wir die Daten zurück und konnten diese auswerten. In dieser Projektarbeit möchte ich auf die Idee und die Entwicklung dieses Projektes zurückblicken, um Rückschlüsse aus dem Experiment zu ziehen und Verbesserungen zu erarbeiten. Weitere Erläuterungen und Gedanken werde ich vereinzelt in den Anmerkungen² ausführen, damit die Projektarbeit nicht zu lang wird. Zudem sind sämtliche Projektdaten, sowie das Programm auf <https://github.com/lucas56098/astropi> einsehbar.

Inhaltsverzeichnis

1 Einleitung	2
2 Orbit- und Positionsbestimmung	3
2.1 Die Internationale Raumstation und der Astro Pi	4
2.2 Was messen wir?	5
2.3 Analyse der Daten	6
3 Auswertung	12
3.1 Probleme bei der Datenaufzeichnung	12
3.2 Ergebnisse	12
3.3 Fazit	15

1 Einleitung

Um an der EUROPEAN ASTRO PI CHALLENGE [3] teilzunehmen, braucht man eine gute Experimentidee. Unsere Idee war, zu überprüfen, ob man mit einem einfachen Programm die Position der Internationalen Raumstation (ISS) anhand von Messdaten bestimmen kann. Warum ist diese Idee im Weltraum sicherheitsrelevant?

Raumstationen sind voller Technik. Computer melden Unregelmäßigkeiten und berechnen passend dazu die aktuelle Position der ISS. Doch im Weltall herrscht eine hohe Strahlenbelastung. Diese wird durch Sonneneruptionen ausgelöst und kann für Probleme sorgen. Trifft Strahlung zum Beispiel auf den Speicher eines Computers, so können dabei Teile des Speichers verändert werden. Dieses kann zum Absturz oder sogar zum Verlust des Betriebssystems eines Computers führen. In diesem Moment weiß man nicht mehr, wo man ist, und das ist gefährlich.³ [13] Würde man nun, unwissend wo man ist, sein Orbit abändern, würde man Gefahr laufen, mit Weltraumschrott zu kollidieren. In dem erdnahen Weltraum gibt es schätzungsweise 330 Millionen Objekte in Umlaufbahnen um die Erde [4]. Diese bewegen sich mit mehreren Kilometern pro Sekunde durch das All, was sie zu gefährlichen Geschossen macht. Ein Stück Weltraumschrott, das nur 10g wiegt, hat eine Sprengkraft von etwa 1.000.000 Joule⁴, weshalb eine Kollision mit Weltraumschrott nicht unterschätzt werden sollte (siehe Abb. 2⁵). Ein Kompletausfall aller Systeme ist zwar sehr unwahrscheinlich, aber dennoch möglich. Für diesen Fall versuchten wir mit unserem Projekt nun den Orbit und die Position der ISS zu bestimmen. Das Experiment wurde dabei speziell für die ISS ausgelegt, ist aber theoretisch mit kleinen Abänderungen im gesamten erdnahen Orbit anwendbar. In der ersten Phase wurde unsere Programmidee angenommen und wir bekamen einen Astro Pi (siehe 2.1) zugesandt, um in der zweiten Phase unser Programm zu entwickeln und Messideen vorab zu Hause zu testen. So mussten wir das gesamte Programm in mehreren Testläufen optimieren (z.B. diente die Fleyer Str. als Messstelle für Tag-/Nachtübergänge).

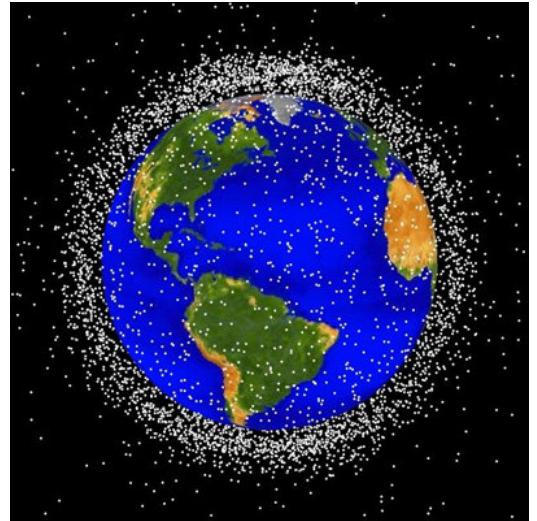


Abbildung 1: Weltraumschrott in Erdnähe



Abbildung 2: Einschlag von Weltraumschrott

2 Orbit- und Positionsbestimmung

Was ist ein Orbit und welche Parameter definieren diesen? Als Orbit bezeichnet man die Umlaufbahn eines Satelliten (zum Beispiel die ISS) um einen Himmelskörper (zum Beispiel die Erde). Ein Orbit stellt also eine ellipsenförmige Bewegung um einen Planeten dar, die sich durch die Keplerschen Gesetze beschreiben lässt. Um den Orbit zu bestimmen, müssen wir dafür verschiedene Parameter kennen. Die sogenannten TLE-Daten⁶ sind eine aktuelle Variante, um einen Orbit zu definieren. Dafür benötigt man die in Abb. 3 aufgeführten Parameter. Da das unsere Berechnungen viel zu kompliziert gemacht hätte, versuchten wir den Orbit möglichst einfach darzustellen, ohne dabei viele Informationen zu verlieren. Dadurch sparten wir uns auch die Einführung vieler Variablen, die für unsere Messgenauigkeit ohnehin irrelevant waren. So entstanden folgende Überlegungen: Wir bestimmen die Position senkrecht über der Erde und die Höhe des Orbits getrennt. Zudem nehmen wir an, dass der Orbit keine Ellipse ist, sondern ein Kreis, da dieses bei der ISS näherungsweise stimmt und für viele andere Orbits von Satelliten auch zutrifft. Ist der Orbit eines Satelliten sehr ellipsenförmig, so kann man unser Modell leider nicht mehr anwenden. Durch diese Annahme fallen alle Parameter bis auf die Inklination, also die Bahnneigung und der Winkel des Knotens⁷, also die Rotation des Orbits, weg. Mathematisch können wir jetzt das Ganze mit einer Ebene an der Einheitskugel⁸ darstellen. Diese stellt nun die Erde dar und die Ebene den Orbit. Dort wo sich die Einheitskugel und die Ebene schneiden, entsteht eine Kreisbahn, die die Position des Satelliten über der Erde darstellt. Nun können wir mit Hilfe eines Punktes auf der Kreisbahn zu einem bestimmten Zeitpunkt und dem Radius des Orbits die Position des Satelliten jederzeit berechnen. Dank der Vektorrechnung können wir die Ebene dadurch bestimmen, dass wir einen Vektor \vec{n} (den sogenannten Normalenvektor) auf die Ebene stellen. Alle Vektoren, die im rechten Winkel zu dem Normalenvektor stehen, bilden die Ebene. Dieses lässt sich durch folgende Gleichung darstellen:

Bahnelement	Bezeichnung
Mittlere Bewegung	n
Numerische Exzentrizität	ϵ
Inklination	i
Winkel des Knotens	Ω
Argument der Periapsis	ω
Mittlere Anomalie	M

Abbildung 3: TLE-Daten Parameter

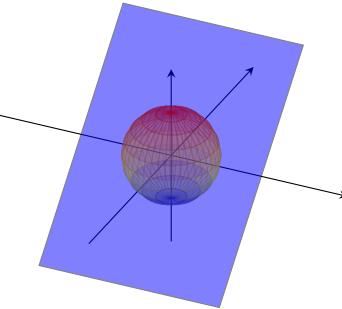


Abbildung 4: Orbit durch eine Ebene

$$E : \vec{x} \cdot \vec{n} = 0 \quad (1)$$

Um den Vektor zu bestimmen, benötigen wir mit Hilfe von Polarkoordinaten⁹ an der Einheitskugel nur die zwei Parameter Phi (ϕ) und Theta (θ). Diese sind ähnlich wie die Inklination und der Winkel des Knotens. Dabei entspricht ϕ dem Längengrad des Tiefpunktes und θ dem Winkel zwischen Normalenvektor und z-Achse.

2.1 Die Internationale Raumstation und der Astro Pi

Die Internationale Raumstation ist mit 109 m Spannweite das größte von Menschen erbaute Objekt, das nicht auf der Erde ist. Seit dem 2. November 2000 ist die ISS dauerhaft von Menschen bewohnt, die dort täglich viele Experimente in verschiedenen Forschungsgebieten ausführen. Dabei umkreist die ISS die Erde in einer Höhe von rund 400 km und braucht für eine Umrundung rund 90 Minuten. Sie hat eine Inklination von rund 52 Grad. An der ISS beteiligte Länder sind die USA, Russland, Großbritannien, Frankreich, Dänemark, Spanien, Italien, die Niederlande, Schweden, Kanada, Deutschland, die Schweiz, Belgien, Japan und Norwegen [11]. Aktuell gelangt man nur mit Hilfe der russischen Sojus-Kapsel zur ISS. Dieses soll sich allerdings in naher Zukunft ändern, da Boeing und SpaceX kurz vor ihren ersten bemannten Flügen zur Internationalen Raumstation stehen. Früher wurden auch Space Shuttles zur Versorgung der ISS genutzt. Möchte man die ISS beobachten, kann man dieses am besten kurz nach Sonnenuntergang tun. Zu diesem Zeitpunkt kann man die ISS noch gut sehen, da sie von der Sonne angestrahlt wird, es aber schon dunkel ist. Die ISS sieht dann aus wie ein großer, heller Stern, der sich relativ langsam bewegt.

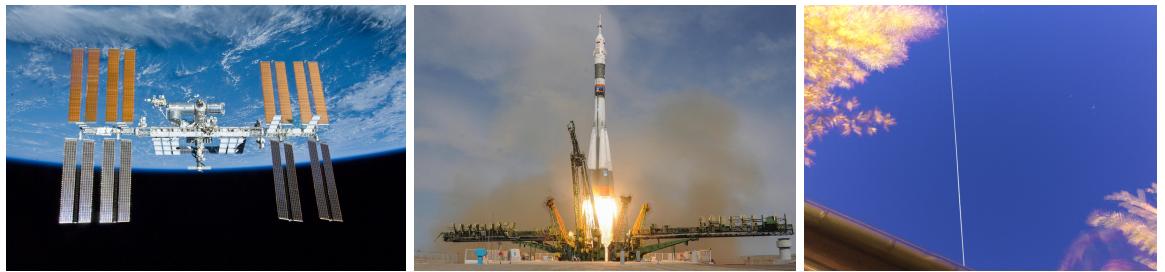


Abbildung 5: Internationale Raumstation, Sojus-Rakete und ISS am Nachthimmel

Der Astro Pi ist ein von der Europäischen Weltraumorganisation (ESA) entwickelter Minicomputer, der sich aus einem Raspberry Pi, einem Sense HAT und einer Infrarotkamera zusammensetzt [2]. Der Raspberry Pi ist ein bekannter Minicomputer, der aus dem Grund entwickelt wurde, dass die Zahl der Informatikstudenten an der Universität Cambridge gesunken ist und die Neubewerber immer weniger von Informatik verstanden. Man vermutete, dass die Computer zu teuer und zu komplex geworden sind, sodass Eltern ihren Kindern nicht mehr erlauben würden, am Heim-PC zu experimentieren. Daher entwickelte man einen preisgünstigen und vergleichsweise sehr einfachen Minicomputer, den man selber benutzerfreundlich programmieren kann. Der Sense HAT ist eine Erweiterung des Raspberry Pi, er kann aufgesteckt werden und liefert viele verschiedene Sensoren: unter anderem einen Magnetsensor, einen Temperatursensor und ein Gyroskop. Die Infrarotkamera, die man ebenfalls mit dem Raspberry Pi verwenden kann, ist eine „near infrared camera“.[6] Das Infrarotspektrum ist in nahe Infrarot, mittleres Infrarot und langwelliges Infrarot unterteilt. Das bedeutet,

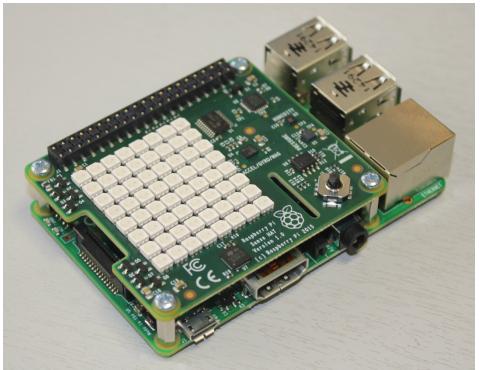


Abbildung 6: Astro Pi ohne Hülle

dass die Infrarotkamera zwar mehr als normales Licht, aber zum Beispiel keine Temperaturen sehen kann, wohingegen Wärmebildkameras das langwellige Infrarot aufnehmen.[8] Zwei Astro Pi Minicomputer sind auf der ISS stationiert und einer davon ist mit seiner Kamera auf den Erdboden gerichtet. Auf diesen beiden kann man im Rahmen des Wettbewerbes Python3-Programme laufen lassen. Da der Astro Pi vergleichsweise billig ist und mit USB betrieben werden kann, ist er ideal, um unser Projekt auszuführen.

2.2 Was messen wir?

Um herauszufinden, wo wir sind, mussten wir zunächst klären, was wir messen können und was uns hilft. Wir begannen mit dem Magnetsensor. Die Erde hat ein einigermaßen regelmäßiges Erdmagnetfeld, bei dem die Magnetpole näherungsweise dem Nord- und Südpol entsprechen. Da die Internationale Raumstation sich einmal pro Erdumdrehung um sich selbst dreht, und damit die Kamera immer auf den Boden zeigt, können wir davon ausgehen, dass die Z-Komponente des Magnetfeldes sinusförmig ist und die Amplitude mit der Inklination zunimmt (siehe Abb. 7). Bei einem Orbit, der über beide Pole geht (sprich Inklination = 90 Grad) ist die Amplitude dieses Sinus maximal. Bei einem Orbit, der die Inklination 0 hat, ist hingegen die Amplitude gleich 0. Deshalb ist es sinnvoll, das Magnetfeld aufzuzeichnen, um die Inklination herauszufinden.

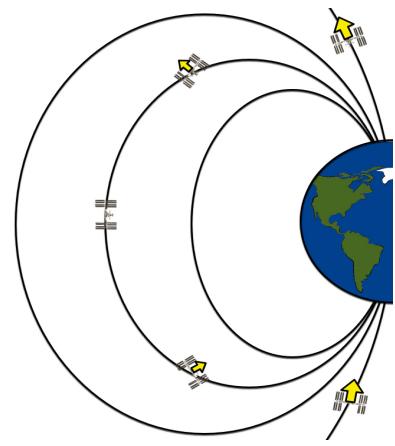


Abbildung 7: Magnet-Z

Mit der Kamera und einem kleinen Programm konnten wir die Helligkeit aufzeichnen. Diese konnte uns dann mit der Uhrzeit den Zeitpunkt des Sonnenaufgangs und des Sonnenuntergangs liefern. Dieses half uns später, die Bahndrehung ϕ zu berechnen (siehe 2.3, Abb. 11).

```

29      #naming imagepath
30      image_path = "image" + str(self.__photo_number) + ".jpg"
31      #captures image and saves image to file
32      self.__camera.capture(image_path)
33      #opens imagefile and converts image to greyscale
34      image = Image.open(image_path).convert("L")
35      #calculates average pixel level in the image
36      image_stat = ImageStat.Stat(image)
37      brightness = image_stat.mean[0]

```

Listing 1: Programm zur Helligkeitsbestimmung

In den ersten 175 Minuten des Experiments zeichneten wir alle Daten auf, die wir benötigten. Die restlichen 5 Minuten wurden die Daten ausgewertet. Wir zeichneten hierbei die Unixzeit, also die Zeit in Sekunden seit dem 1.1.1970 00:00 UTC, den Magnetsensor, der eine x-, y- und z-Komponente hat, die Kamera-Helligkeit und die echte Position der ISS (diese diente nur als späterer Vergleich und ließ sich ganz einfach mit einem vorprogrammierten Befehl bestimmen) auf. Um die Daten möglichst platzsparend zu speichern, verwendeten wir csv-Dateien. Eine Comma-Separated-Value-Datei (CSV) speichert die Daten nur durch ein Komma getrennt hintereinander. Um unsere Daten in einer csv-Datei zu speichern, mussten wir also jede Messung erneut an die csv-Datei anhängen. So konnten wir 461 Datenreihen aufzeichnen, um den Orbit zu berechnen. Die Internationale Raumstation umlief in dieser Zeit zweimal die Erde.

sense_time	magnet_x	magnet_y	magnet_z	brightness	sublat	sublong	image_path
1587473196.2013657	10.448518753051758	-10.219980239868164	62.623348236083984	120.5181884765625	49.73814787570653	-63.59438189018215	
1587473218.4810102	10.299482345581055	-10.017217636108398	62.35128402709961	112.2004508972168	50.118732846448815	-61.52908977952121	
1587473241.9158564	10.114873886108398	-9.920600891113281	62.74197006225586	106.05514526367188	50.475142362623835	-59.32142237561134	
1587473264.2156506	9.817997932434082	-10.054533958435059	62.6287956237793	136.58805084228516	50.77135400654895	-57.19019844889227	
1587473286.8467422	9.455493927001953	-10.137681007385254	62.1744384765625	112.0606079105625	51.0281383349016	-55.000395404756574	
1587473309.7821057	9.05334758758545	-10.246517181396484	62.07954406738281	134.32860565185547	51.242404932848045	-52.75746714925652	
1587473332.3597174	8.661995887756348	-10.32711124420166	62.17489242553711	119.77761459350586	51.40736756168219	-50.530511291130544	
1587473355.3923137	7.753302574157715	-10.481472969055176	62.11799621582031	133.81427001953125	51.52804675903256	-48.243647843794385	
1587473377.9128404	7.258775234222412	-10.598686218261719	61.80044937133789	120.16498184204102	51.599090971684454	-45.99789188997794	
1587473400.162123	7.027459621429443	-10.683283805847168	61.743080139160156	127.41163635253906	51.62344059607555	-43.77424867359942	
1587473424.096354	6.834428787231445	-11.004539489746094	61.55152130126953	137.1222267150879	51.598664084861326	-41.38221397342341	
1587473446.3969882	6.620405673980713	-11.044910430908203	61.27192306518555	121.88994598388672	51.5281187601885	-39.1582941343835	
1587473469.0314047	6.429504871368408	-10.96745777130127	60.75057983984375	109.19729995727539	51.40996986375597	-36.910799897423296	
1587473491.942964	6.065385818481445	-11.142842292785645	60.6717414855957	137.71399307250977	51.243057115913786	-34.65044099916137	
1587473514.4402068	5.820157527923584	-11.075047492980957	60.41108322143555	125.5045394897461	51.03346593104277	-32.44998969010963	
1587473537.5449064	5.718739986419678	-11.208395004272461	59.9158821105957	124.28379821777344	50.771838949980015	-30.21380283132038	

Abbildung 8: Die data.csv

2.3 Analyse der Daten

Um den Orbit zu kennen, mussten wir seine Inklination θ , seine Bahndrehung ϕ , die Umlaufzeit und die Position zu einem Zeitpunkt kennen. Wir begannen damit, dass wir möglichst viele Informationen aus der Magnet-Z-Komponente herausfilterten, da diese, wie wir im vorherigen Kapitel gesehen haben, abhängig von der Umlaufbahn der Internationalen Raumstation sind. Dafür legten wir eine Funktion über unsere Messwerte. Da es sich bei der Erdumrundung um eine kreisförmige Bewegung handelt, ist es sinnvoll, eine Kosinus-Funktion¹⁰ über die Messwerte zu legen. Dabei muss man jedoch annehmen, dass die Erde sich während des Experiments nicht dreht, da sonst streng genommen keine Kosinus-Funktion vorliegt. Verallgemeinert gilt dabei:

$$f(t) = A \cdot \cos(\omega t + p) + c. \quad (2)$$

Wir versuchten also, A, ω , p und c zu bestimmen. Da man Python in Methoden und Klassen schreibt, konnten wir aus dem Pythonpaket SciPy¹¹ eine sogenannte Fit-Methode verwenden, die die bestmögliche Kurve, also die mit dem kleinsten Fehler, über die Messwerte legt. Da diese Methode einigermaßen gute Startwerte braucht, damit sie funktioniert, mussten wir diese vorher „raten“. Dafür nutzten wir eine spezielle Fourier-Transformation¹² aus der Dokumentation [12]. Mit diesen Werten und den Messdaten führten wir die Methode aus und erhielten die gesuchten Werte A, ω , p und c.

```

83     # getting the start values with fft
84     f = np.fft.fftfreq(len(t), ((t[5]-t[1])/4.0))
85     fy = abs(np.fft.fft(y))
86     start_freq = abs(f[np.argmax(fy[1:])+1])
87     start_amp = np.std(y) * 2.**0.5
88     start_offset = np.mean(y)
89     start = np.array([start_amp, 2.*np.pi*start_freq, 0., start_offset])
90
91     def cosfunc(t, A, w, p, c): return A * np.cos(w*t + p) + c
92
93
94     #fitting the curve
95     try:
96         popt, pcov = optimize.curve_fit(cosfunc, t, y, p0 = start)
97     except Exception as err:
98         raise Exception("Fitting impossible: is allowed during testing with no real data")
99     A, w, p, c = popt

```

Listing 2: Fouriertransformation und SciPy-Methode

Da die Fit-Methode auch eine negative Amplitude und ein $\phi > 2\pi$ zulässt, mussten wir die Funktion noch anpassen. So kann man zum Beispiel die Amplitude umdrehen, indem man die Funktion um eine halbe Periodendauer verschiebt. Nun kann man die Kurve analysieren. Da man ω kennt, kann man die Umlaufzeit T unmittelbar durch

$$T = \frac{2\pi}{\omega} \quad (3)$$

bestimmen. Zudem weiß man, dass bei einem Kosinus der Hochpunkt immer bei 0 bzw. bei der Umlaufzeit, der Tiefpunkt immer bei der halben Umlaufzeit und die Wendepunkte immer bei 1/4 der Umlaufzeit bzw. 3/4 der Umlaufzeit liegen. Da das Magnetfeld der Erde aber vor allem im Süden unregelmäßig ist [9], verwenden wir für den Hoch- und den Tiefpunkt eine separate Methode, die den höchsten bzw. niedrigsten Wert sucht. Man kann nun aus den Hoch- und Tiefpunkten bestimmen, wo Norden ist. Normalerweise wird der Astro Pi so aufgehängt, dass die Magnet-X-Komponente am Hochpunkt in Richtung Norden zeigt. Wenn der Astro Pi perfekt aufgehängt wäre, müsste die Magnet-Y-Komponente gleich null sein. Doch da wir nicht wussten, wie genau der Astro Pi aufgehängt ist, überprüften wir dieses, indem wir die Magnetfelddaten kalibrierten. Der wirkliche Norden setzt sich dabei aus der Magnet-X- und der Magnet-Y-Komponente zusammen, und man muss danach die Daten um diesen Winkel korrigieren.

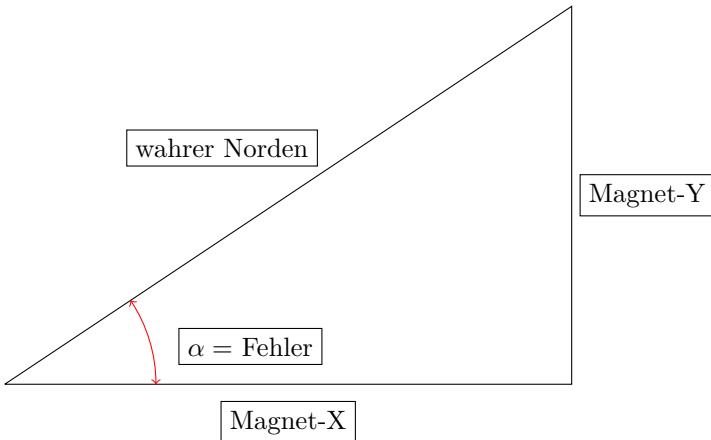


Abbildung 9: Berechnung des Nordversatzes

Der Winkel des Dreiecks aus Abb. 9 entspricht dem Fehler, der beim Aufhängen des Astro Pis entsteht und lässt sich folgendermaßen berechnen:

$$\alpha = \arctan\left(\frac{Y_{hp} - Y_{tp}}{X_{hp} - X_{tp}}\right), \quad (4)$$

wobei Y_{hp} und X_{hp} die entsprechenden Magnetwerte am Hochpunkt und Y_{tp} und X_{tp} die entsprechenden Magnetwerte am Tiefpunkt sind. Man kann nun mit Hilfe einer Drehmatrix [7] den Fehler bei jedem einzelnen Magnetwert korrigieren und so aus den Magnetwerten (x_{mag} und y_{mag}) die kalibrierten Magnetwerte (x_{cal} und y_{cal}) bestimmen.

$$\begin{pmatrix} x_{cal} \\ y_{cal} \end{pmatrix} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} x_{mag} \\ y_{mag} \end{pmatrix} \quad (5)$$

Diese Kalibrierung lässt sich in Python relativ einfach mit einer List-Comprehension darstellen, bei der jedes Element in den Magnet-Listen durch die Rotationsmatrix kalibriert wird. Das Ergebnis wird dann in neuen Listen abgespeichert, die sich `cal_magnet_x` und `cal_magnet_y` nennen.

```

130     #calibrates magnet data
131     cal_magnet_x = [self.__magnet_x[i] * np.cos(north_offset) - self.__magnet_y[i] *
132                     np.sin(north_offset) for i in range(len(self.__magnet_x))]
133     cal_magnet_y = [self.__magnet_x[i] * np.sin(north_offset) + self.__magnet_y[i] *
134                     np.cos(north_offset) for i in range(len(self.__magnet_y))]
```

Listing 3: List-Comprehension zur Magnetdatenkalibrierung

Da unsere Magnetdaten jetzt korrigiert sind, können wir diese verwenden, um weitere Werte zu berechnen. Man beginnt mit der Inklination, also der Bahnneigung. Da wir dank der Kalibrierung jetzt den Norden festgelegt haben, entspricht die Inklination nur noch der Neigung α' an den Wendepunkten. Dafür kann man nun wieder den Arkustangens wie in Gleichung 4 verwenden.

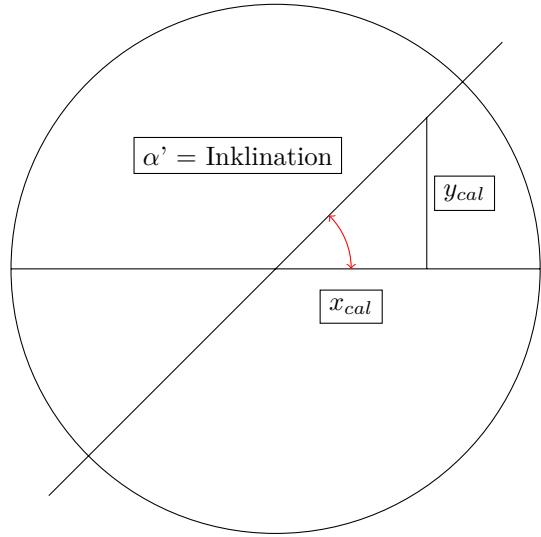


Abbildung 10: Inklinationsberechnung an den Wendepunkten

Um Abweichungen zu minimieren, verwendet man die Magnetwerte an den Wendepunkten und ihre benachbarten Werte, um diese jeweils zu mitteln. Daraus berechnet man für beide Wendepunkte jeweils die Neigung. Dieses Ergebnis mittelt man erneut und erhält so die Inklination.

Wir kennen nun also schon die Umlaufzeit und die Inklination. Damit fehlt uns noch die Bahndrehung und die Position zu einem bestimmten Zeitpunkt, weshalb wir als nächstes versuchen, die Bahndrehung herauszufinden. Dieses ist jedoch schwieriger als gedacht. Zu Beginn wollten wir die Bahndrehung mit Hilfe der Kosinusfunktion bestimmen, bis uns auffiel, dass das gar nicht geht. Den Großteil der Projektarbeit verbrachten wir damit, dieses Problem zu lösen. Dazu machten wir uns die Kamera zunutze und bestimmten mit dieser den Sonnenaufgang und den Sonnenuntergang. In einem kreisförmigen Orbit ist die Mitte zwischen Sonnenaufgang und -untergang immer der Zeitpunkt, an dem die Sonne am höchsten steht. Dafür kann man den Zeitpunkt berechnen. Dadurch wissen wir auch, auf welchem Längengrad wir uns befinden. Daraus bestimmen wir dann die Bahndrehung. Wenn ein Normalenvek-

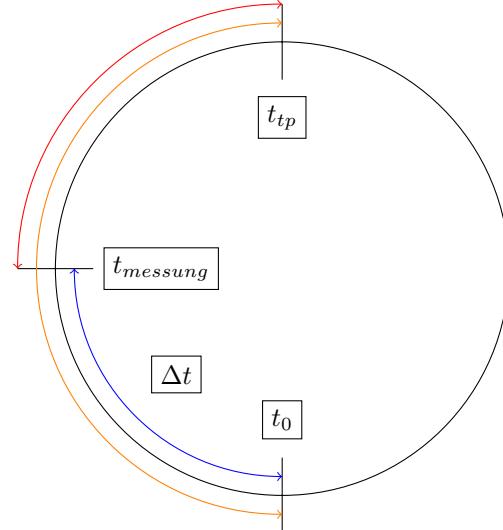


Abbildung 11: Δt bestimmen

tor die Ebene definiert, ist der Tiefpunkt t_{tp} immer direkt unter ϕ (vorausgesetzt θ ist positiv, was es aber ist). Wir nehmen nun an, dass $\phi = 0$ gilt und somit der Tiefpunkt an $\phi = 0$ ist¹³(siehe Abb. 11). Die Zeitdifferenz zwischen dem Tiefpunkt t_{tp} und dem gemessenen Sonnenhöchststand $t_{messung}$ kann bestimmt werden. Mit einer weiteren Methode, die den Längengrad L des echten Sonnenhöchststands berechnet, kann man nun die Zeit t_0 bestimmen, an der die ISS über den Sonnenhöchststand fliegen würde, wenn der Orbit die Bahndrehung = 0 hätte.

$$t_0 = t_{tp} + \left(T \cdot \frac{L}{2\pi} \right) \quad (6)$$

Dafür multipliziert man den Bruchteil des bisherigen Umlaufs $\frac{L}{2\pi}$ mit der Umlaufzeit T und addiert danach die „Startzeit“ t_{tp} , also wenn die ISS am Tiefpunkt ist. Man kann nun die Differenz Δt aus t_0 (also der hypothetischen Zeit, wenn $\phi = 0$ wäre) und aus der auf der ISS gemessenen Zeit $t_{messung}$ bilden.

$$\Delta t = t_0 - t_{messung} \quad (7)$$

Diese formt man mit Hilfe der Winkelgeschwindigkeit in einen Winkel um.

$$\beta = \omega \cdot \Delta t \quad (8)$$

Dieser Winkel β ist nun der Fehler, der entsteht, wenn wir $\phi = 0$ setzen und entspricht somit der gesuchten Bahndrehung.

Man kennt nun den Orbit der ISS und es fehlt nur noch die Position zu einem Zeitpunkt. Dafür verwendet man eine ähnliche Methode wie die, die den Längengrad des Sonnenhöchststands berechnet hat. Die hier verwendete Methode gibt

passend zur Zeit einen Vektor zurück, der vom Erdmittelpunkt auf die Sonne zeigt. Dieser definiert nun die Tag-Nacht-Ebene. Alle Vektoren, die senkrecht zu dieser Ebene sind, liegen auf dem Tag-Nacht-Übergang. Nun hat man die Tag-Nacht-Ebene und die Orbit-Ebene. Dort, wo sich diese beiden Ebenen an der Einheitskugel schneiden, entstehen zwei Punkte (P_1 und P_2). Diese Punkte sind die Position der ISS, wenn sie über einen Sonnenaufgang bzw. Sonnenuntergang

fliegt. Die Zeiten zu diesen Punkten hat man bereits weiter oben mit Hilfe der Kamera berechnet. Mathematisch lassen sich die Schnittpunkte mit dem normierten Kreuzprodukt der beiden Normalenvektoren der Ebenen berechnen.

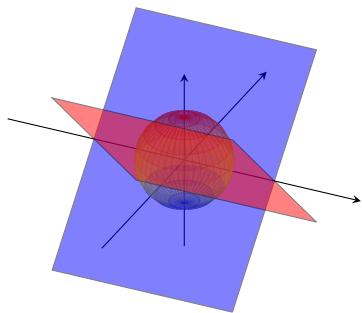


Abbildung 12: Tag-Nacht- und Orbit-Ebene

$$P_{1/2} = \pm \frac{\overrightarrow{\text{tag_nacht}} \times \overrightarrow{\text{orbit}}}{|\overrightarrow{\text{tag_nacht}} \times \overrightarrow{\text{orbit}}|} \quad (9)$$

Dasselbe sieht in Python implementiert so aus:

```
231     vector_sunrise = np.cross(vector_day, vector_orbit)
232     vector_sunrise = vector_sunrise/np.linalg.norm(vector_sunrise)
```

Listing 4: Schnittpunktberechnung

Man hat nun also den Orbit und die Position vollständig bestimmt. Damit ist das Hauptziel der Projektarbeit erreicht. Doch eine theoretische Überlegung ist nichts wert, wenn sie nicht in der realen Welt funktioniert. Deshalb führten wir im nächsten Abschnitt unser Experiment aus und testeten ob es funktioniert. Zunächst mussten wir uns noch mit dem Fall beschäftigen, was passiert, wenn irgendetwas nicht klappt. Es muss dabei gesichert werden, dass das Programm niemals abstürzt, auch wenn die Positionsbestimmung oder ander Methoden nicht klappen. Hauptziel ist es, dass die Daten immer vollständig gesichert werden, um eine spätere Fehleranalyse zu ermöglichen. Dafür verwenden wir sogenannte try/except-Anweisungen.

```
95     try:
96         popt, pcov = optimize.curve_fit(cosfunc, t, y, p0 = start)
97     except Exception as err:
98         raise Exception("Fitting impossible: is allowed during testing with no real data")
```

Listing 5: try/except-Anweisung

Diese versucht den in try angegebenen Befehl und meldet einen Error, wenn er nicht funktioniert. Im Hauptprogramm werden dann die gefundenen Fehler in eine .txt Datei gespeichert und das Programm bis zum Ende hin ausgeführt. Das Programm war nun fertig und konnte auf der Internationalen Raumstation im Weltall ausgeführt werden.

3 Auswertung

3.1 Probleme bei der Datenaufzeichnung

Als die Versuchsdaten nach dem Experiment auf der ISS vom Astro Pi Team in Cambridge heruntergeladen wurden, fiel ihnen auf, dass der Astro Pi eine falsche Kameraeinstellung hatte, weshalb alle Bilder einen Pinkstich hatten. Außerdem hatte die NASA den sogenannten CANADARM vor der Kamera vergessen. Der CANADARM ist ein fest auf der ISS stationierter Roboterarm, der verschiedenste Arbeiten auf der ISS ausführt. Da das für die meisten Experimente problematisch war, mussten alle Programme ein zweites Mal laufen. Am 18.05.20 erhielten wir dann die neuen Experimentdaten, um diese auszuwerten. Auf Abb. 13 kann man ein Bild mit der falschen Kameraeinstellung und dem CANADARM sehen. Die Internationale Raumstation flog zu diesem Zeitpunkt über die Hawaii-Inseln, die man auch im Bild erkennen kann.

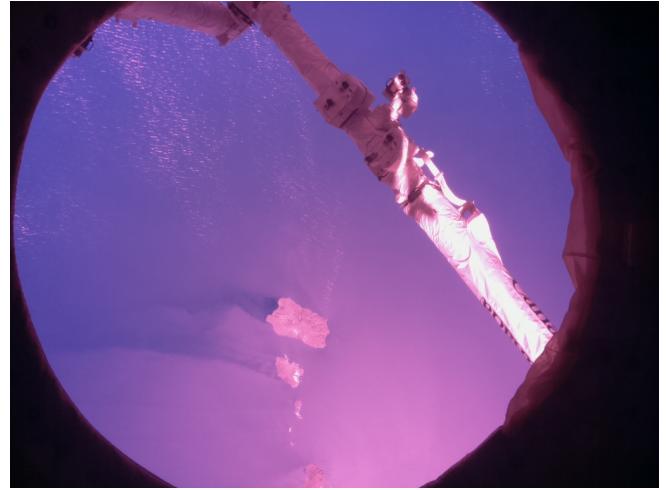


Abbildung 13: CANADARM und Hawaii

3.2 Ergebnisse

In den drei Stunden Rechenzeit auf der ISS wurden für unser Experiment 461 Datenreihen aufgezeichnet und während des Flugs ausgewertet. Es wurde kein Fehler gemeldet und ein Ergebnis wurde in der result.txt gespeichert, was bedeutet, dass die Tag-/Nachtübergänge erkannt wurden und dass die Fit-Methode funktioniert hat. Die Daten haben wir mit Pythons Bibliothek Matplotlib¹⁴ weiter ausgewertet. Bevor wir jedoch damit begannen, betrachten wir den Fehler, der durch die Zeitintervalle der Messung entstand. So kann man in der Abb. 14 erkennen, dass die Datenreihen recht unregelmäßig aufgezeichnet wurden. Im Mittel ergibt sich ein Intervall von 22,8 Sekunden.

Dieses bestimmt die maximale Genauigkeit der Berechnungen. Die ISS legt zum Vergleich in dieser Zeit rund 200km zurück.

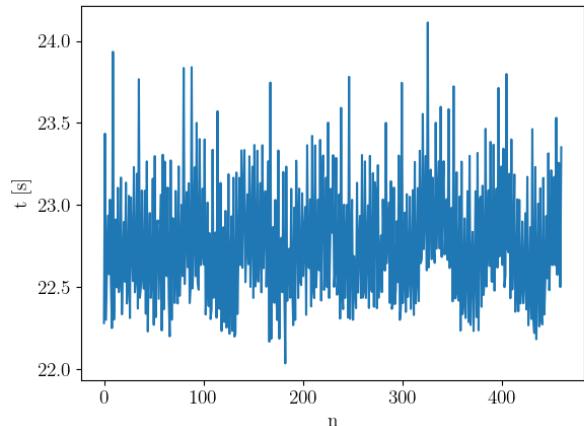


Abbildung 14: Zeitintervalle der Messungen

Als nächstes haben wir den Magnet-Z-Komponente und die Fit-Methode ausgewertet. Auf der x-Achse ist dabei die Zeit in Sekunden zu sehen und auf der y-Achse zum einen der Magnet-Z-Wert in Mikrotesla und zum anderen der Breitengrad der ISS. In blau kann man nun die Magnetdaten sehen, in orange sieht man die durch die Fit-Methode bestimmte Kosinusfunktion und in grün den Referenzwert des Breitengrades der ISS. Deutlich kann man hier erkennen, dass es den in 2.2 beschriebenen Zusammenhang zwischen Inklination und Magnet-Z-Komponente gibt. Was man auch erkennen kann ist, dass es Ungenauigkeiten im Magnetfeld gibt. Daher stimmt die Annahme, dass das Erdmagnetfeld einem Stabmagneten gleicht, nur näherungsweise. Die Inklination, die sich nun über die Wendepunkte berechnen lässt, beträgt 52,26 Grad. Zudem kann man aus dieser Kurve die Umlaufzeit ablesen, die 94,1 Minuten beträgt. Daraus ergibt sich über das Newtonsche Gravitationsgesetz, gleichgesetzt mit der Zentripetalkraft, eine Flughöhe der ISS von 475,7 Kilometern.

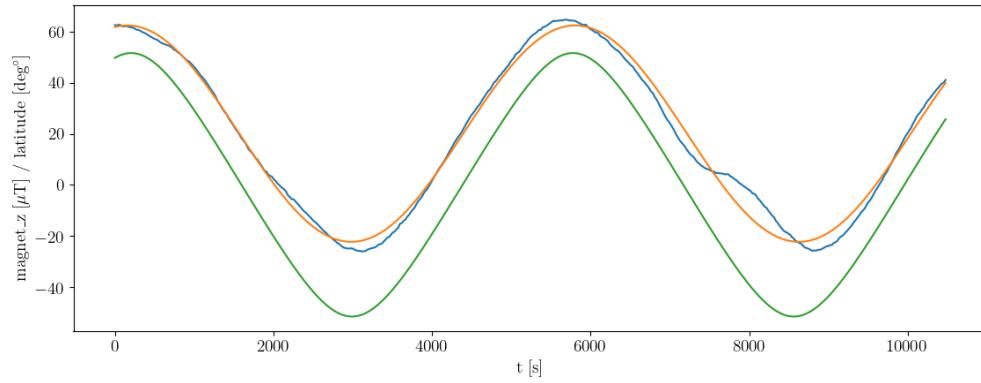


Abbildung 15: Magnet-Z und Fit-Methode

Als nächstes haben wir die Helligkeitsdaten ausgewertet, die in blau dargestellt sind. Die orangene Linie gibt zudem an, ob unser Programm denkt, dass es Tag oder Nacht ist. So kann man gut sehen, dass die Tag-/Nachttübergänge relativ präzise bestimmt wurden. Damit konnten wir auch den hier in rot eingezzeichneten Zeitpunkt des Sonnenhöchststands bestimmen, der ja für die Bahndrehung entscheidend ist.

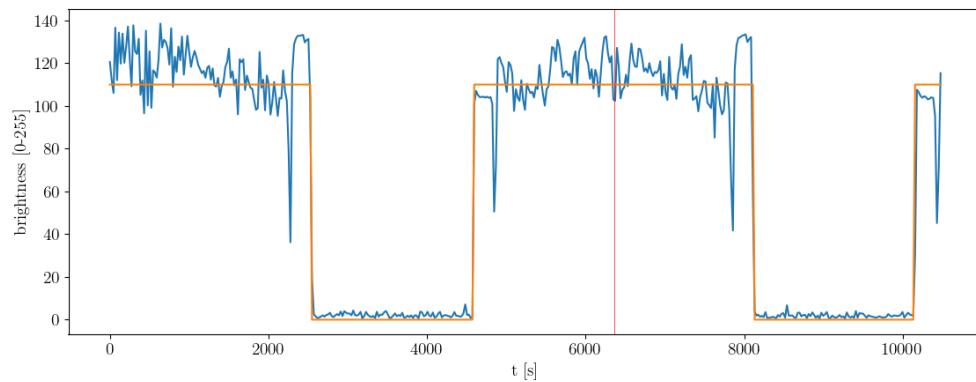


Abbildung 16: Tag-/Nachtmethode angewandt

In Abb. 17 sieht man ein Beispiel eines Bildes, aus dem die Helligkeit ermittelt wurde. Zu sehen sind die Ausläufer des Yellowstone Nationalparks in den USA. Bei der Auswertung der Bahndrehung stellten wir fest, dass die berechnete Bahndrehung von 96,25 Grad im erwarteten Bereich zwischen 92 und 136 Grad liegt. Jedoch war der erwartete Bereich ziemlich groß, da sich die Erde während den drei Stunden Messzeit um 45 Grad dreht.

Das Orbit ist nun bestimmt. Wenn man das auf der Internationalen Raumstation berechnete Orbit in rot auf einer Plaktkarte¹⁵ einzeichnet und zudem in grau die tatsächliche Position der ISS unterlegt, kann man sehen, dass man mit einem Astro Pi die Position der Internationalen Raumstation recht gut bestimmen kann. Erkennbar ist jedoch auch, dass sich die Erde beim zweiten Umlauf der ISS schon ein Stück weiter gedreht hat, weshalb die zweite graue Punktreihe versetzt erscheint.

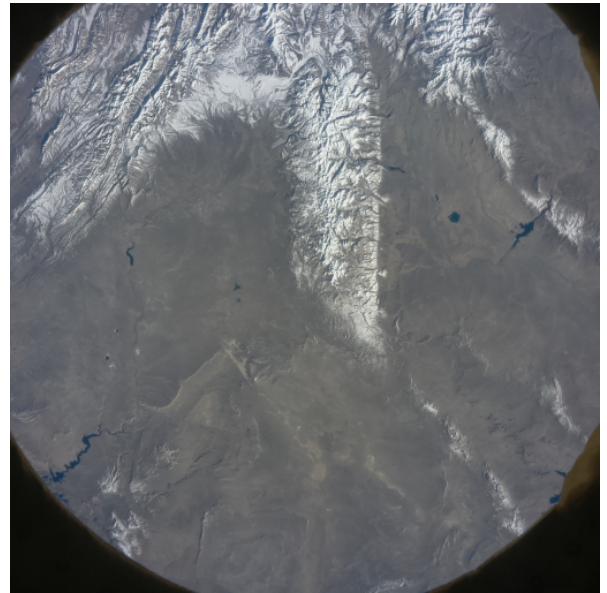


Abbildung 17: Yellowstone Nationalpark

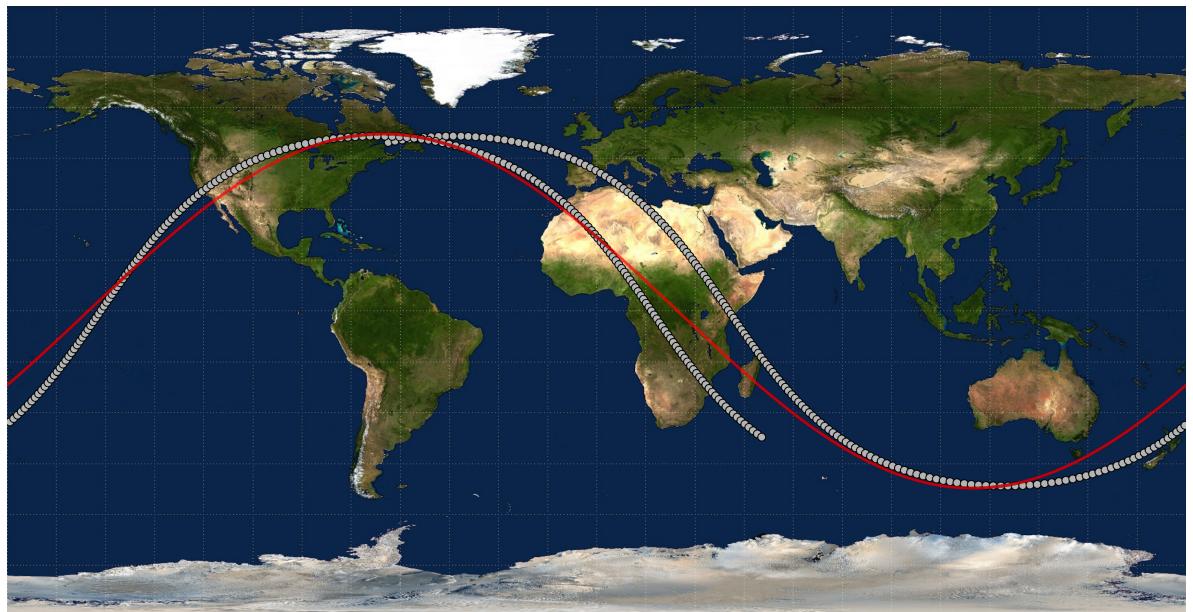


Abbildung 18: Berechneter Orbit im Vergleich zu echten Positionen

3.3 Fazit

Das Programm ist am 21.04.2020 über die 180 Minuten erfolgreich und fehlerfrei auf dem Astro Pi der ISS (Abb. 20) gelaufen. Während dieser Zeit wurden alle erwarteten Daten aufgezeichnet und in die csv-Datei gespeichert. Die Daten konnten vom Programm verwendet werden und wurden erfolgreich während des Flugs ausgewertet. Die Fit-Methode hat funktioniert und so konnten Umlaufzeit und Höhe näherungsweise bestimmt werden. Des Weiteren ließen sich folgende Annahmen bestätigen: Die Inklination hängt direkt mit der Magnet-Z-Komponente zusammen. Durch die Kamera kann man erstaunlich genau Tag und Nacht bestimmen. Es gab aber auch Faktoren, die für Ungenauigkeiten verantwortlich waren. So waren die Zeitintervalle der Messungen unerwartet groß. Zudem ist das Erdmagnetfeld in manchen Bereichen doch mit einer deutlich höhern Missweisung belastet, weshalb man in einem verbesserten Experiment das Erdmagnetfeld nicht als Stabmagneten annehmen sollte, sondern besser einen Datensatz inklusive Magnetfeldanomalien verwenden sollte. Zudem ist eine Erdrotation von 45 Grad während des Experiments nicht unbedeutend und sollte daher in weiteren Experimenten nicht vernachlässigt werden. Dieses würde das mathematische Modell allerdings deutlich verkomplizieren, da die Umlaufbahn dann keine Trigonometrische Funktion mehr ist. Auch die gewählten Parameter könnten noch um weitere ergänzt werden, um die Genauigkeit und Flexibilität zu erhöhen (siehe Abb. 3). Wir konnten zusammenfassend zeigen, dass man mit so einfachen Mitteln (Raspberry Pi mit Sense HAT und Kamera = 80 Euro) den Orbit und die Position der ISS erstaunlich gut bestimmen kann. Jedoch führen im Weltall aufgrund der hohen Geschwindigkeiten kleinste Ungenauigkeiten bereits zu deutlichen Positionsabweichungen, weshalb unser Programm das Problem aus der Einleitung nicht löst. Es ist aber sicherlich „ein kleiner Schritt ...“[10].

Parameter	Tatsächlich	Berechnet
Umlaufzeit	93min	94,1min
Höhe	320-410km	475,7km
Inklination	51,5°	52,26°
Bahndrehung	92-136°	96,25°

Abbildung 19: Ergebnis

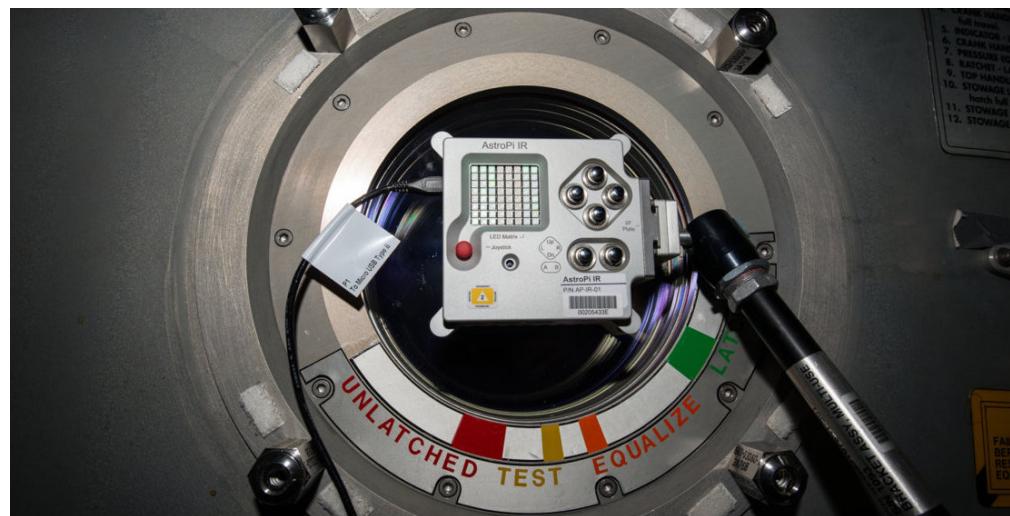


Abbildung 20: Der Astro Pi in der Internationalen Raumstation

Anmerkungen

¹ Der Teamname Milliways Regulars bedeutet Milliways Stammkunden und ist eine Anspielung an die Per Anhalter durch die Galaxis Werke von Douglas Adams. Milliways ist in diesen Büchern das Restaurant am Ende des Universums. „Das Restaurant am Ende des Universums ist eine der außergewöhnlichsten Unternehmungen in der gesamten Geschichte des Gaststättengewerbes. Es steht auf den zertrümmerten Überresten eines möglicherweise zerstörten Planeten, der in eine riesige Zeitblase eingeschlossen und in die zukünftige Zeit genau an den Moment projiziert ist, an dem das Universum endet. Das würden viele sagen ist unmöglich. In dem Restaurant nehmen die Gäste an Tischen Platz und essen kostspielige Menüs, während sie zusehen, wie die ganze Schöpfung um sie herum explodiert.“[1](Kapitel 15)

² Don't Panic! Das hier ist nur eine Anmerkung!

³ Natürlich gibt es mittlerweile Lösungen, die dieses Problem minimieren. Die sogenannten redundanten Systeme sind im Prinzip einfach das selbe System mehrfach. Dabei gleichen sich die Festplatten immer ab und so werden Störungen gut beseitigt. Ein kleines Restrisiko besteht jedoch immer. [13]

⁴ Angenommen wird hier, dass die ISS und der Weltraumschrott frontal mit jeweils einer Geschwindigkeit von 7km/s kollidieren: $E_{kin} = \frac{1}{2}mv^2 = \frac{1}{2} \cdot 0,001 \cdot 14000^2 = 1.000.000$

⁵ Das Bild zeigt einen Einschlag von Weltraumschrott mit ca. 6,8 km/s in einen 18cm dicken Aluminiumblock. Die Kugel war dabei nur 1,2cm breit und wog nur ganze 1.7 g. Die Temperaturen, die bei diesem Einschlag entstanden sind waren größer als die, die im Erdmittelpunkt sind.

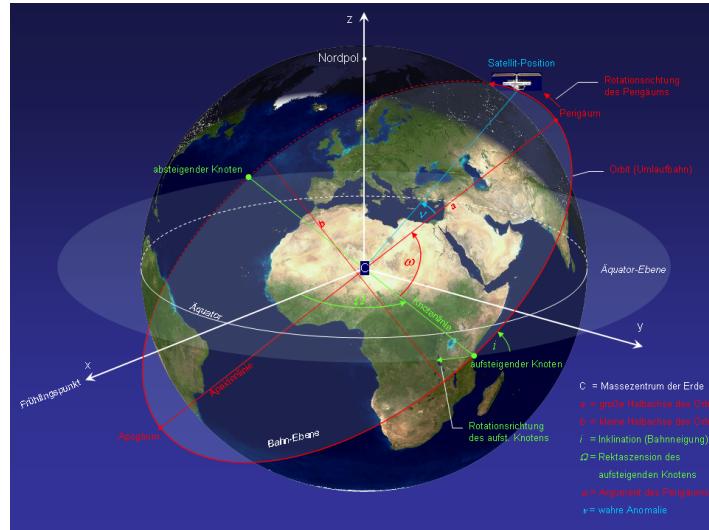
⁶ Das Two-line element set ist eine Variante um ein Orbit anzugeben:

1 ISS (ZARYA)

2 25544U 98067A 08264.51782528 -.00002182 00000-0 -11606-4 0 2927

3 2 25544 51.6416 247.4627 0006703 130.5360 325.0288 15.72125391563537

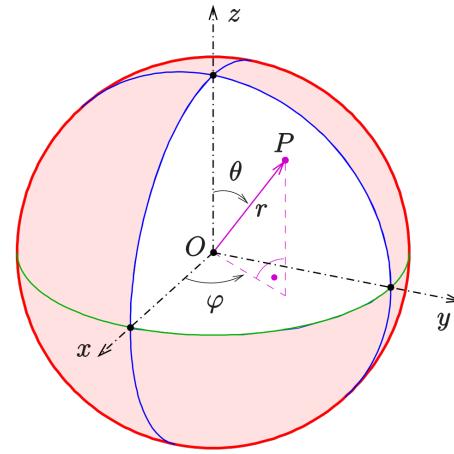
Auf dem Bild (Wikimedia Commons) kann man noch einmal alle Parameter sehen.



⁷ Der Knoten ist in diesem Zusammenhang der Name für die Stelle, an dem der Satellit den Äquator in Richtung Norden überschreitet. Der Winkel des Knotens gibt also an, um wieviel Grad das Orbit gedreht ist. Wir verwenden für unsere Berechnungen jedoch ϕ , was dem Winkel des Tiefpunktes entspricht und somit ähnlich dem Winkel des Knotens ist.

⁸ Eine Kugel mit dem Radius eins um den Nullpunkt eines Vektorraums.

⁹ Polarkoordinaten, insbesondere Kugelkoordinaten, sind eine alternative Variante Punkte im Raum darzustellen. Anstelle von den 3 Koordinaten (x,y,z) gibt es bei den Kugelkoordinaten den Radius r (der bei unseren Berechnungen wegfällt, da er immer eins ist, weil wir an der Einheitskugel arbeiten), den Winkel ϕ und den Winkel θ . Ihre Anordnung kann man auf dem Bild (Wikimedia Commons) erkennen.



¹⁰ Da die Trigonometrischen Funktionen (Sinus, Kosinus) am Einheitskreis definiert sind, ist es sinnvoll, bei kreisförmigen Bewegungen auf diese zurückzugreifen. Ob man den Sinus oder Kosinus verwendet, ist aufgrund von

$$\sin(\alpha + 90^\circ) = \cos(\alpha)$$

egal.

¹¹ Eine Dokumentation für SciPy findet sich unter: <https://www.scipy.org>

¹² Eine zeitabhängige Funktion $f(t)$ und ihre Fourier-Transformierte $F(\omega)$ sind so miteinander verknüpft [5]:

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} d\omega F(\omega) e^{i\omega t}$$

mit

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dt f(t) e^{-i\omega t}$$

¹³ Phi ist nicht gleich Null. Nur der Fehler, der entstand, wenn wir Phi = 0 einsetzen, entsprach dem echten Phi.

¹⁴ Eine Dokumentation von Matplotlib findet sich unter: <https://matplotlib.org>

¹⁵ Eine Kugel perfekt auf eine zweidimensionale Karte abzubilden ist unmöglich. Daher versuchen seit Jahrhunderten Wissenschaftler verschiedene Karten zu entwickeln, die die Realität möglichst gut wiederspiegeln. Die bekannteste Projektion ist wohl die Mercatorprojektion. Diese hat den Vorteil, dass Strecken erhalten bleiben. Die hier verwendete quadratische Plattkarte (Rektangularprojektion) hat hingegen den Vorteil, dass das Gradnetz einem Kartesischen Koordinatensystem entspricht und wir somit das Orbit und die Referenzpunkte direkt abbilden können.

Literatur

- [1] Douglas Adams. *Das Restaurant am Ende des Universums.* Roger und Bernhard bei Zweitausendeins, Ulm, 2013.
- [2] ESA Education. Astro Pi Hardware. www.astro-pi.org (11.05.20).
- [3] ESA Education. European Astro Pi Challenge (Mission SpaceLab). www.astro-pi.org (10.05.20).
- [4] ESA. Space Debris: Gefahren für die Raumfahrt beherrschen lernen. www.esa.int (10.05.20).
- [5] Bartelmann et al. *Theoretische Physik.* Springer Spektrum, Heidelberg, Berlin, 2015.
- [6] RaspberryPi Foundation. About Us. www.raspberrypi.org (22.05.20).
- [7] Peter Furlan. *Das gelbe Rechenbuch 1 - für Ingenieure, Naturwissenschaftler und Mathematiker.* Verlag Martina Furlan, Erbstollen 12, Dortmund, 2012.
- [8] Deutsches Institut für Normung. *Strahlungsphysik im optischen Bereich und Lichttechnik, Benennung der Wellenlängenbereiche: DIN 5031.* (Hrsg.), 7 edition, 1984.
- [9] gfz potsdam. Entwicklung des geomagnetischen Dipolmoments und der südatlantischen Anomalie. www.gfz-potsdam.de (11.05.20).
- [10] NASA. Apollo 11 Lunar Surface Journal - One Small Step. history.nasa.gov (23.05.20).
- [11] NASA. Partners Sign ISS Agreements. www.nasa.gov (22.05.20).
- [12] unsym. How do I fit a sine curve to my data with pylab and numpy? www.stackoverflow.com (11.05.20).
- [13] Frank Wunderlich-Pfeiffer. Hardware im Weltall: Unendliche Weiten voller Strahlung. www.golem.de (10.05.20).

Abbildungsverzeichnis

1	Weltraumschrott in Erdnähe: www.dlr.de (10.05.20)	2
2	Einschlag von Weltraumschrott: www.esa.int (10.05.20)	2
3	TLE-Daten Parameter	3
4	Orbit durch eine Ebene	3
5	ISS: www.esa.int(10.05.20), Sojus Rakete: www.spiegel.de (10.05.20)	4
6	RaspberryPi: www.astro-pi.org (22.05.20)	4
7	Magnet-Z	5
8	Die data.csv	6
9	Berechnung des Nordversatzes	8
10	Inklinationsberechnung an den Wendepunkten	9
11	Δt bestimmung	9
12	Tag-Nacht- und Orbit-Ebene	10
13	CANADARM und Hawaii: Astro Pi Team, ESA (via Email)	12
14	Zeitintervalle der Messungen	12
15	Magnet-Z und Fit-Methode	13
16	Tag-/Nachtmethode angewandt	13
17	Yellowstone Nationalpark	14
18	Berechneter Orbit im Vergleich zu echten Positionen, Plakarte: Wikimedia Commons	14
19	Ergebnis	15
20	Astro Pi: www.esa.int (22.05.20)	15

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die im Literaturverzeichnis angegebenen Hilfsmittel verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen uns sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.