Albrecht Dürer High School Hagen

How to determine orbit and position on the International Space Station?

An experiment as part of the European Astro Pi Challenge 19/20

Project report in the LaTeX-course
of Q1 by Mr. Dr. Müller,
presented by Lucas Schleuß,
Fleyer Str. 83, 58097 Hagen,
on May 25, 2020.

Foreword

In autumn 2019, the European Space Agency (ESA) announced a competition in which experiments could be submitted using the Python programming language on a mini computer called RasperryPi. The best programs were uploaded on the International Space Station (ISS) and were allowed to run there for three hours. The results from Geospace could then be evaluated. Since I like programming with Python and am also very interested in space travel, I decided to team up with Jannik Mänzer and my uncle Torben Heßling as Milliways Regulars[1] and determine the position of the ISS as an experiment. We submitted our experiment idea at the end of October and then developed the program from November to February. On April 2nd, 2020 there was finally the feedback. Our experiment was achieved flight status on the International Space Station! The program ran on April 21st, 2020 and rounded the earth twice. On May 18, 2020 we got the data back and were able to evaluate it. In this report I would like to look back on the idea and the development of this project in order to draw conclusions from the experiment and to develop improvements. I will occasionally give further explanations and thoughts in the notes[2], so that the project work doesn't get too long. In addition, all project data and the program can be viewed at: `https://github.com/lucas56098/astropi`.

Contents

1 Introduction

To take part in the European Astro Pi Challenge [3] you need a good idea for an experiment. Our idea was to check whether the position of the International Space Station (ISS) could be determined using measurement data and a simple program. But, why might this idea be relevant to security in space?

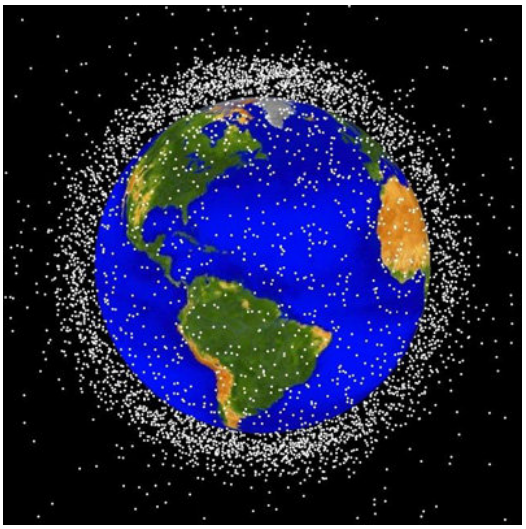


Figure 1: Space junk close to earth

Figure 2: Impact of space junk

Space stations are full of technology. Computers report irregularities and calculate the current position of the ISS regularly. But there is a high level of radiation in space. This is triggered by solar eruptions and can cause problems. If radiation hits a computer, parts of the memory can be changed. This can lead to mistakes or even loss of a computer's operating system. At this moment you don't know where you are anymore, and that's dangerous.[3] [13] If you were to change your orbit, ignorant of where you were, you would run the risk of colliding with space debris. In near-Earth space there are an estimated 330 million objects in orbits around the earth [4]. These move through space at several kilometers per second, which makes them dangerous projectiles. A piece of space junk weighing only 10g has an explosive power of around 1,000,000 joules[4], which is why a collision with space junk should not be underestimated (siehe Abb. 2[5]). A complete failure of all systems is very unlikely, but is still possible. In this case we tried to determine the orbit and the position of the ISS with our project. The experiment was specially designed for the ISS, but is theoretically applicable with small changes in the entire Geospace. In the first phase our program idea was accepted and we received an Astro Pi (see 2.1) to develop our program in the second phase and to test measurement ideas in advance at home. We had to optimize the entire program in several test runs (e.g. Fleyer Str. served as a measuring point for day / night transitions).

2 Orbit and position determination

What is an orbit and which parameters define it? An orbit is the orbit of a satellite (for example the ISS) around a celestial body (for example the earth). An orbit therefore represents an elliptical movement around a planet, which can be described by Kepler's laws. To determine the orbit, we need to know various parameters. The so-called TLE data[6] is a current variant for defining an orbit. For this you need the parameters listed in Fig. 3. Since this would have made our calculations far too complicated, we tried to determine the orbit as simply as possible without losing much information. This saved us the introduction of many variables that were irrelevant to our measuring accuracy anyway. This led to the following considerations: We determine the position perpendicular to the earth and the height of the orbit separately. In addition, we assume that the orbit is not an ellipse, but a circle, since this is approximately correct for the ISS and also applies to many other orbits from satellites. If the orbit of a satellite is very elliptical, our model can unfortunately no longer be used. This assumption eliminates all parameters except for the inclination and the angle of the node[7], i.e. the rotation of the orbit. We can now represent the orbit thing mathematically with one plane and the unit sphere[8]. The unit sphere now represents the earth and the plane the orbit. Where the unit sphere and the plane intersect, a circular path is created that represents the position of the satellite above the earth. Now we can calculate the position of the satellite at any time with the help of a point on the circular orbit at a specific time and the radius of the orbit. Thanks to vector calculation, we can determine the plane by placing a vector $\vec{n}$ (the so-called normal vector) on the plane. All vectors that are perpendicular to the normal vector form the plane. This can be represented by the following equation:

term	**variable**
Medium movement	n
Numerical eccentricity	ϵ
Inclination	i
Angle of the knot	Ω
Argument of periapsis	ω
Medium anomaly	M

Figure 3: TLE data

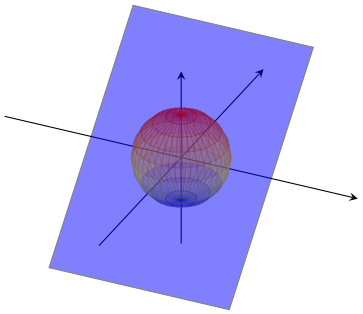


Figure 4: Orbit through a plane

$$E : \vec{x} \cdot \vec{n} = 0 \tag{1}$$

To determine the vector, we only need the two parameters Phi (ϕ) and Theta (θ) using polar coordinates[9]on the unit sphere. These are similar to the inclination and angle of the knot. ϕ corresponds to the longitude of the low point and θ is the angle between the normal vector and the z-axis.

2.1 The International Space Station and the Astro Pi

With a span of 109 m, the International Space Station is the largest man-made object that is not on Earth. Since November 2, 2000, the ISS has been permanently inhabited by people who carry out many experiments there every day in various research areas. The ISS orbits the earth at an altitude of around 400 km and takes around 90 minutes to circle. It has an inclination of around 52 degrees. Countries participating in the ISS are the USA, Russia, Great Britain, France, Denmark, Spain, Italy, the Netherlands, Sweden, Canada, Germany, Switzerland, Belgium, Japan and Norway [11]. Currently you can only get to the ISS with the help of the Russian Soyuz capsule. However, this is set to change in the near future as Boeing and SpaceX are about to begin their first manned flights to the International Space Station. In the past, space shuttles were also used to supply the ISS. If you want to watch the ISS, it is best to do this shortly after sunset. At this point you can still see the ISS well, because it is illuminated by the sun, but it is already dark. The ISS then looks like a large, bright star that moves relatively slowly.

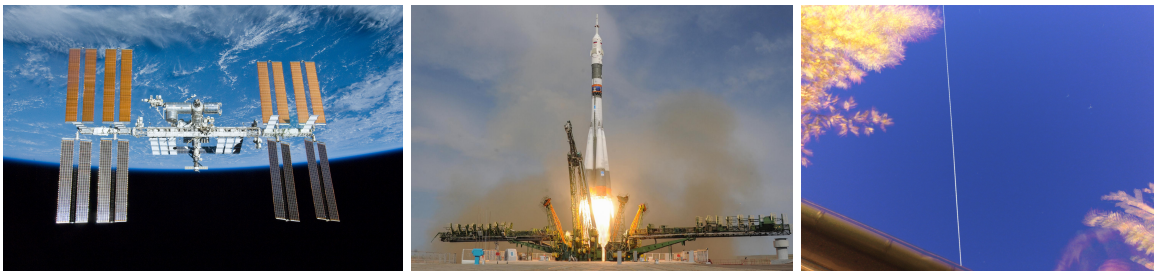


Figure 5: International Space Station, Soyuz-Rocket and ISS in the night sky

The Astro Pi is a mini computer developed by the European Space Agency (ESA), which is composed of a Raspberry Pi, a Sense HAT and an infrared camera [2]. The Raspberry Pi is a well-known mini computer that was developed for the reason that the number of computer science students at Cambridge University has decreased and that new applicants have become less aware of computer science. It was suspected that the computers had become too expensive and too complex, so that parents would no longer allow their children to experiment on their home PC. Therefore, an inexpensive and comparatively very simple mini-computer was developed, which you can program yourself in a user-friendly manner. The Sense HAT is an extension of the Raspberry Pi, it can be plugged on and supplies many different sensors: among others a magnetic sensor, a temperature sensor and a gyroscope. The infrared camera that can also be used with the Raspberry Pi is a "near infrared camera". [6] The infrared spectrum is divided into near infrared, medium infrared and long-wave infrared. This means that the infrared camera can see more than normal light, but for example no temperatures, whereas thermal imaging cameras record the long-wave

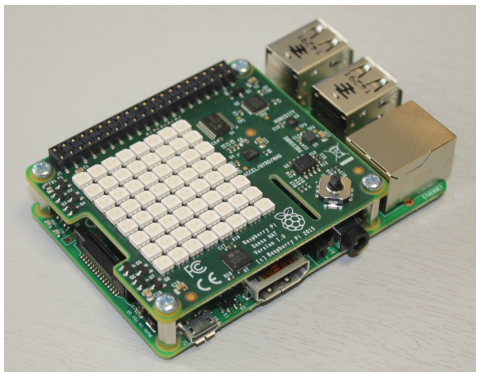


Figure 6: Astro Pi without case

infrared. [8] Two Astro Pi minicomputers are stationed on the ISS and one of them is directed on the Earth with its camera. You can run Python3 programs on these two as part of the competition. Since the Astro Pi is comparatively cheap and can be operated with USB, it is ideal to carry out our project.

2.2 What do we measure?

To find out where we are, we first had to clarify what we can measure and what helps us. We started with the magnetic sensor. Earth has a reasonably regular earth magnetic field, in which the magnetic poles correspond approximately to the north and south poles. Since the International Space Station rotates once per revolution of the earth, and thus the camera always points to the ground, we can assume that the Z component of the magnetic field is sinusoidal and that the amplitude increases with inclination (see Fig. 7). With an orbit that goes over both poles (i.e. inclination = 90 degrees), the amplitude of this sine is maximum. In the case of an orbit that has an inclination of 0, on the other hand, the amplitude is 0. Therefore it makes sense to record the magnetic field to find out the inclination.

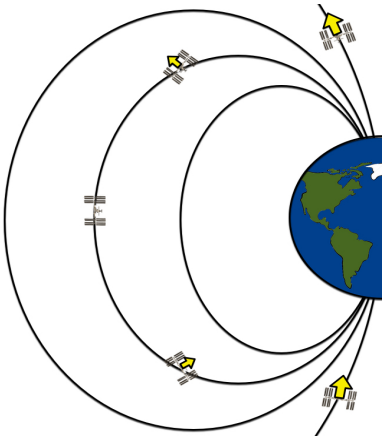


Figure 7: Magnet-Z

We were able to record the brightness with the camera and a small program. This could then give us the time of sunrise and sunset with the time, wich later helped us to calculate the longitude of the lowpoint ϕ (see 2.3, Fig. 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: program for brightness determination

In the first 175 minutes of the experiment, we recorded all the data we needed. The data was evaluated for the remaining 5 minutes. We plotted the Unix time, i.e. the time in seconds since 1/1/1970 00:00 UTC, the magnetic sensor, which has an x, y and z component, the camera brightness and the real position of the ISS (this served only as a later comparison and could easily be determined with a preprogrammed command). We used csv files to save the data as space-saving as possible. A comma separated value file (CSV) saves the data one after the other only separated by a comma. In order to save our data in a csv file, we had to append each measurement to the csv file again. So we could record 461 rows of data to calculate the orbit. The International Space Station orbited the earth twice during this time.

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	112.06060791015625	51.02813838349016	-55.000395404756574	
1587473309.7821057	9.05334758758545	-10.246517181396484	62.07954406738281	134.32860565185547	51.242404832848045	-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	51.52811847601885	-39.1582941343835	
1587473469.0314047	6.429504871368408	-10.96745777130127	60.750579833984375	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	

Figure 8: the data.csv

2.3 Analysis of the data

To know the orbit, we had to know its inclination θ, its orbit rotation ϕ, the orbital period and the position at a time. We started by filtering out as much information as possible from the Magnet-Z component because, as we saw in the previous chapter, it depends on the orbit of the International Space Station. For this we put a function over our measured values. Since the orbit around the earth is a circular movement, it makes sense to put a cosine function[10] over the measured values. However, one must assume that the earth does not rotate during the experiment, because strictly speaking there is no cosine function otherwise. The general rule is:

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

So we tried to find A, ω, p and c. Since Python is written in methods and classes, we were able to use a so-called fit method from the Python package SciPy[11],which overlays the best possible curve, i.e. the one with the smallest error, over the measured values. Since this method needs reasonably good starting values to work, we had to "guess" these beforehand. For this we used a special Fourier transformation[12] from the documentation [12]. With these values and the measurement data, we carried out the method and received the searched values A, ω, p and c.

```
        # getting the start values with fft
        f = np.fft.fftfreq(len(t), ((t[5]-t[1])/4.0))
        fy = abs(np.fft.fft(y))
        start_freq = abs(f[np.argmax(fy[1:])+1])
        start_amp = np.std(y) * 2.**0.5
        start_offset = np.mean(y)
        start = np.array([start_amp, 2.*np.pi*start_freq, 0., start_offset])

        def cosfunc(t, A, w, p, c): return A * np.cos(w*t + p) + c


        #fitting the curve
        try:
            popt, pcov = optimize.curve_fit(cosfunc, t, y, p0 = start)
        except Exception as err:
            raise Exception("Fitting impossible: is allowed during testing with no real data")
        A, w, p, c = popt
```

Listing 2: Fourier transform and SciPy method

Since the fit method also allows a negative amplitude and a $\phi > 2\pi$, we had to adjust the function. For example, you can reverse the amplitude by shifting the function by half a period. Now you can analyze the curve. Since ω is known, the orbital period T can be determined directly using

$$T = \frac{2\pi}{\omega}. \tag{3}$$

In addition, we know that the high point of a cosine is always zero or the orbital period, the low point is always half the orbital period and the turning points are always 1/4 of the round trip time or 3/4 of the orbital period. However, since the geomagnetic field is irregular especially in the southern hemisphere [9], we use a separate method for the high and low points, which searches for the highest and lowest value. You can now determine where north is from the high and low points. Usually the Astro Pi is suspended so that the Magnet-X component points north at the high point. If the Astro Pi were perfectly suspended, the magnet Y component would have to be zero. But since we didn't know how exactly the Astro Pi was suspended, we checked it by calibrating the magnetic field data. The real north is made up of the Magnet-X and Magnet-Y components, and you then have to correct the data by this angle.

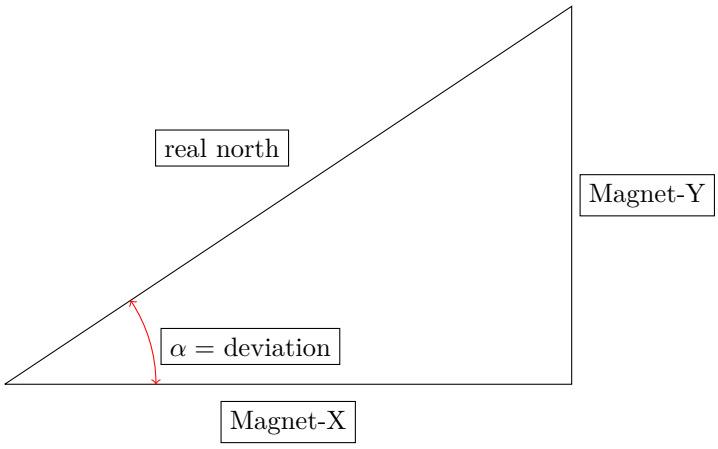


Figure 9: Calculation of the north offset

The angle of the triangle from Fig. 9 corresponds to the error that occurs when the Astro Pi is hung up and can be calculated as follows:

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

where Y_{hp} and X_{hp} are the corresponding magnet values at the high point and Y_{tp} and X_{tp} are the corresponding magnet values at the low point. With the help of a rotation matrix [7] you can now correct the error for each individual magnet value and thus from the magnet values (x_{mag} and y_{mag}) the calibrated magnet values (x_{cal} and y_{cal}).

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

This calibration can be represented relatively easily in Python with a list comprehension, in which every element in the magnet lists is calibrated by the rotation matrix. The result is then saved in new lists called cal _magnet _x and cal _magnet _y.

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

Listing 3: List comprehension for magnetic data calibration

Now that our magnetic data has been corrected, we can use it to calculate further values. You start with the inclination, i.e. the inclination. Since we have now determined the north thanks to the calibration, the inclination only corresponds to the inclination α' at the turning points. Now you can use the arctangent as in equation 4.

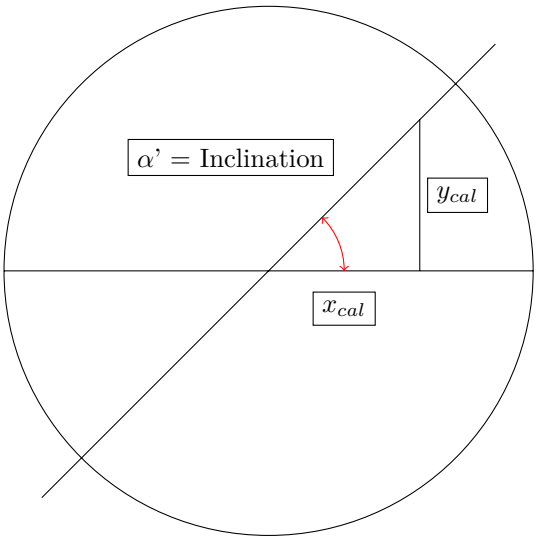


Figure 10: Inclination calculation at the turning points

In order to minimize deviations, the magnet values at the turning points and their neighboring values are used to average them. From this, the inclination is calculated for both turning points. This result is averaged again and the inclination is thus obtained.

So we already know the orbital period and the inclination. So we still lack the orbit rotation and the position at a certain time, which is why we will try to find out the orbit rotation next. However, this is more difficult than expected. At the beginning, we wanted to determine the orbit rotation using the cosine function until we realized that this was not possible. We spent most of the project work solving this problem. We took advantage of the camera and used it to determine the sunrise and sunset. In a circular orbit, the middle between sunrise and sunset is always the time when the sun is at its highest. You can calculate the time for this. As a result, we also know what longitude we are on. From this we then determine the orbit rotation. When a normal vector defines the plane, the low point t_{tp} is always directly below ϕ (provided θ is positive, which it is). We now assume that $\phi = 0$ applies and the lowest

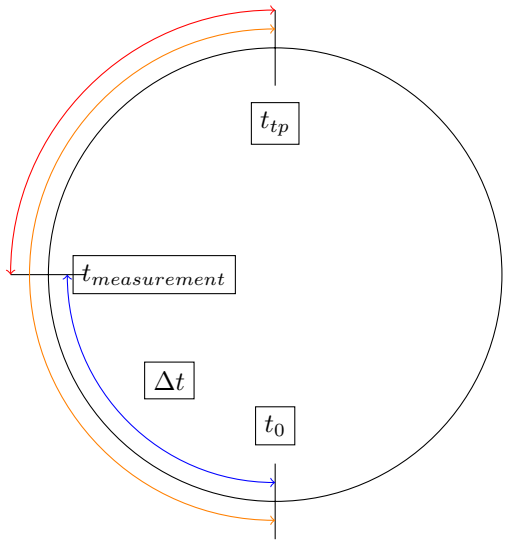


Figure 11: Δt determination

point is $\phi = 0$[13](see Fig. 11). The time difference between the low point t_{tp} and the measured high of the sun $t_{measurement}$ can be determined. Another method, which calculates the longitude L of the real sun peak, can now determine the time t_0 at which the ISS would fly above the sun peak if the orbit had the orbit rotation = 0.

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

To do this, multiply the fraction of the previous round $\frac{L}{2\pi}$ by the round time T and then add the "start time" t_{tp}, so when the ISS is at its lowest point. The difference Δt can now be calculated from t_0 (i.e. the hypothetical time if $\phi = 0$) and from the time $t_{measurement}$ measured on the ISS.

$$\Delta t = t_0 - t_{measurement} \tag{7}$$

These are transformed into an angle using the angular velocity.

$$\beta = \omega \cdot \Delta t \tag{8}$$

This angle β is now the error that arises when we set $\phi = 0$ and thus corresponds to the path rotation sought.

The orbit of the ISS is now known and only one position at a time is missing. To do this, a method similar to that used to calculate the longitude of the highest sun is used. The method used here returns a vector that points from the center of the earth to the sun. This now defines the day-night level. All vectors that are perpendicular to this plane lie on the day-night transition. Now you have the day-night level and the orbit level. Where these two levels intersect on the unit sphere, two points are created (P_1 and P_2). These points are the position of the ISS when it flies over a sunrise or sunset. The times at these points have already been calculated using the camera above. The intersection points can be calculated mathematically with the normalized cross product of the two normal vectors of the planes.

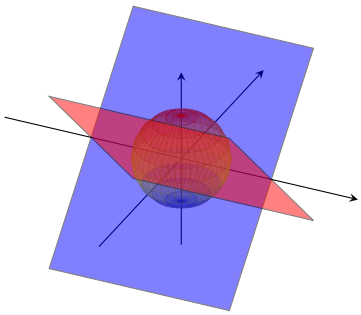


Figure 12: Day-night and orbit plane

$$P_{1/2} = \pm \frac{\overrightarrow{day\_night} \times \overrightarrow{orbit}}{|\overrightarrow{day\_night} \times \overrightarrow{orbit}|} \tag{9}$$

The same thing implemented in Python looks like:

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

Listing 4: Intersection calculation

So you have now completely determined the orbit and the position. This is the main goal of the project work. But a theoretical consideration is worth nothing if it doesn't work in the real world. Therefore, in the next section, we carried out our experiment and tested whether it worked. First of all, we had to deal with the case of what happens when something doesn't work. It must be ensured that the program never crashes, even if the positioning or other methods do not work. The main goal is to ensure that the data is always completely backed up in order to enable later error analysis. For this we use so-called try / except statements.

```
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/exept-statement

This tries the command specified in try and reports an error if it does not work. The errors found are then saved in a .txt file in the main program and the program is executed until the end. The program was now ready and could be carried out on the International Space Station in space.

3 Evaluation

3.1 Data recording problems

When the trial data was downloaded from the Astro Pi team in Cambridge after the experiment on the ISS, they noticed that the Astro Pi had the wrong camera setting, which is why all the pictures had a pink tinge. In addition, NASA had forgotten the so-called CANADARM in front of the camera. The CANADARM is a robot arm that is permanently stationed on the ISS and performs various tasks on the ISS. Since this was problematic for most experiments, all programs had to run a second time. On May 18, 20 we received the new experiment data to evaluate them. Fig. 13 shows an image with the wrong camera setting and the CANADARM. At that time, the International Space Station was flying over the Hawaiian Islands, which can also be seen in the picture.

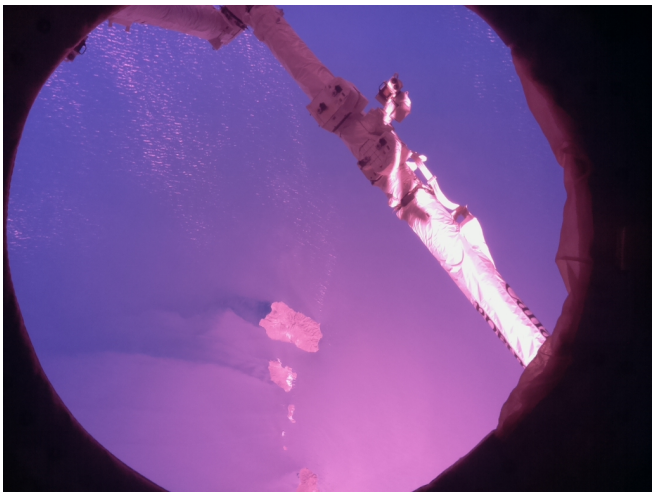


Figure 13: CANADARM and Hawaii

3.2 Results

In the three hours of computing time on the ISS, 461 data series were recorded for our experiment and evaluated during the flight. No error was reported and a result was saved in the result.txt, which means that the day / night transitions were recognized and that the fit method worked. We further evaluated the data with Pyhton's Matplotlib library[14]. Before we start, however, let's look at the error caused by the time intervals of the measurement. You can see in Fig. 14 that the data series were recorded quite irregularly. The average interval is 22.8 seconds. This determines the maximum accuracy of the calculations. The ISS covers around 200km during this time.

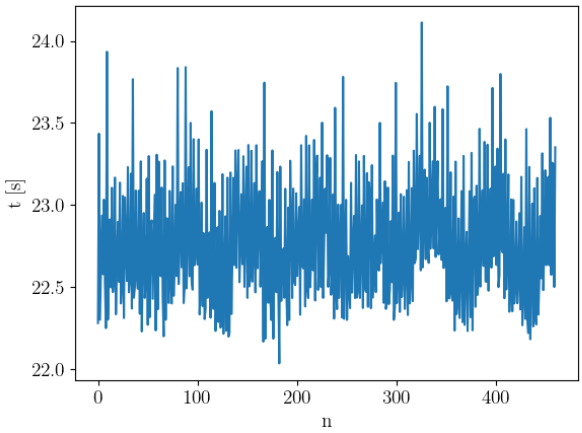


Figure 14: Time intervals of the measurements

Next we evaluated the Magnet-Z component and the fit method. The time in seconds can be seen on the x-axis and on the y-axis the Magnet-Z value in Microtesla and the latitude of the ISS. The magnetic data can now be seen in blue, the cosine function determined by the fit method in orange and the reference value for the latitude of the ISS in green. It can be clearly seen here that there is a relationship between inclination and Magnet-Z component wich is described in 2.2. What can also be seen is that there are inaccuracies in the magnetic field. Therefore, the assumption that the earth's magnetic field is like a bar magnet is only approximate. The inclination, which can now be calculated using the turning points, is 52.26 degrees. You can also measure the orbital period from this curve, which is 94.1 minutes. Based on Newton's law of gravity, equated with the centripetal force, this results in an ISS flight altitude of 475.7 kilometers.

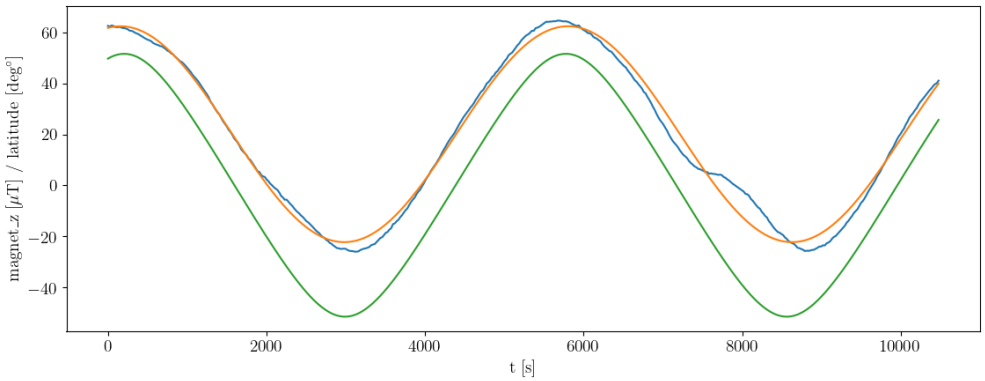


Figure 15: Magnet-Z und fit method

Next we evaluated the brightness data, which are shown in blue. The orange line also indicates whether our program thinks it is day or night. So you can see that the day / night transitions were determined relatively precisely. With this we were also able to determine the time of the sun's peak, which is shown here in red, which is decisive for the orbit rotation.

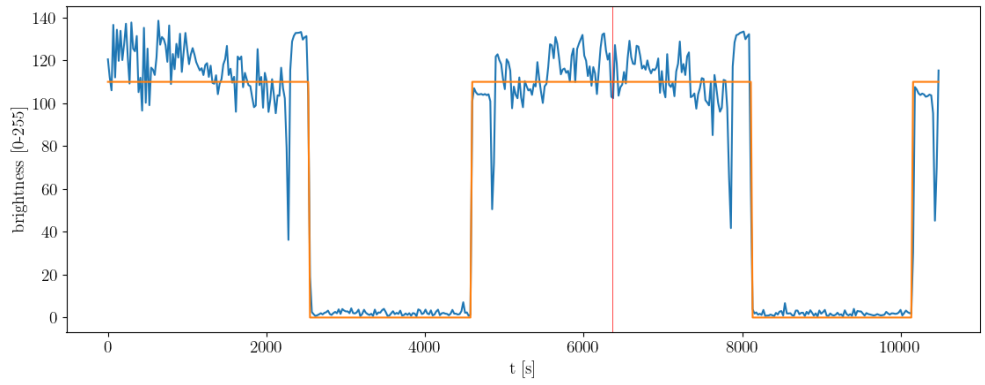


Figure 16: Day / night method applied

Fig. 17 shows an example of an image from which the brightness was determined. You can see the foothills of the Yellowstone National Park in the USA. When evaluating, we also found that the calculated angle of the minimum of 96.25 degrees is in the expected range between 92 and 136 degrees. However, the expected range was quite large as the Earth rotates 45 degrees during the three hour measurement period.

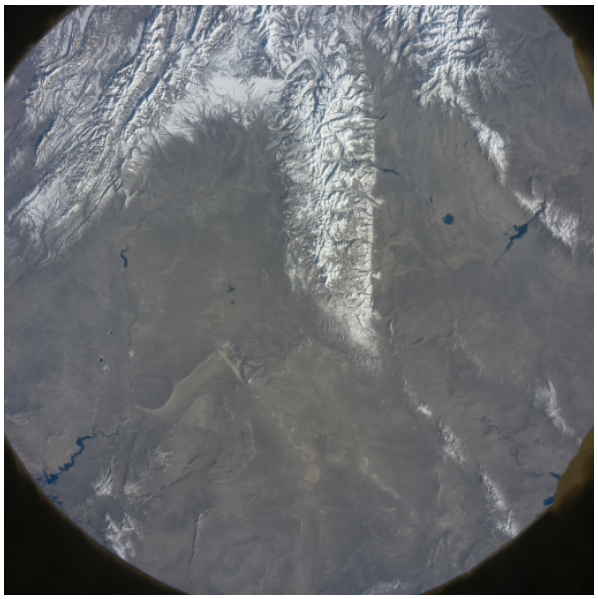


Figure 17: Yellowstone Nationalpark

The orbit is now determined. If you draw the calculated orbit of the International Space Station in red on a equirectangular projection of the earth [15] and also highlight the actual position of the ISS in gray, you can see that the position of the International Space Station can be determined quite well with an Astro Pi. However, it can also be seen that the Earth has already turned a little further during the second orbit of the ISS, which is why the second gray row of dots appears offset.

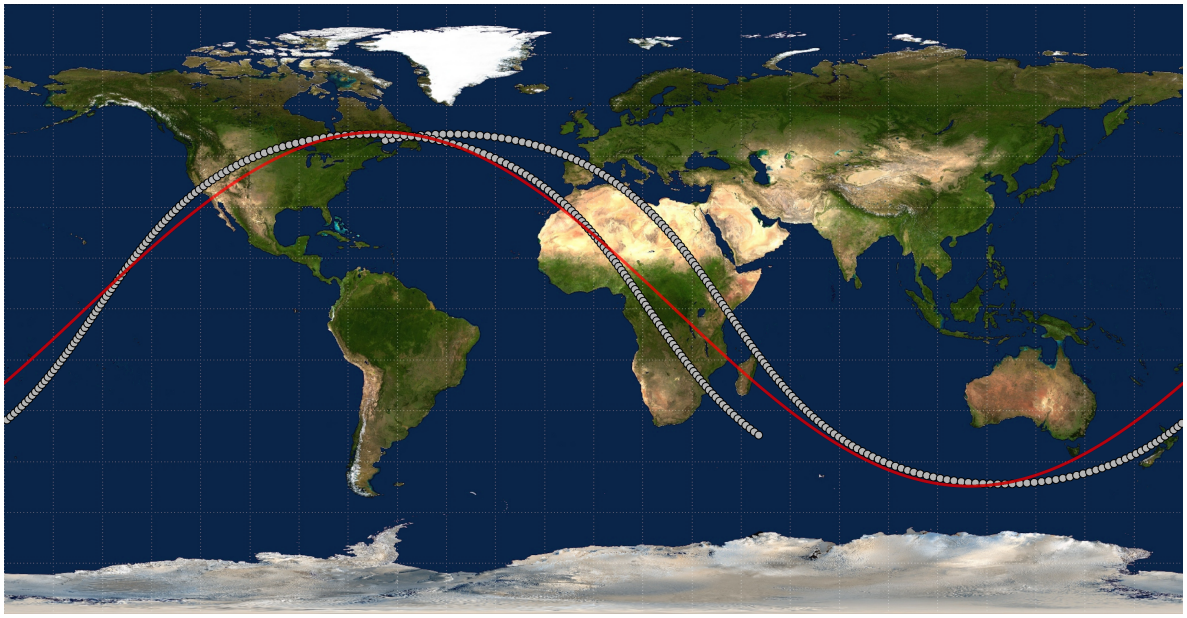


Figure 18: Calculated orbit compared to real positions

3.3 Conclusion

The program ran successfully on April 21st, 2020 over the 180 minutes on the Astro Pi of the ISS (Fig. 20). During this time, all expected data was recorded and saved in the csv file. The data could be used by the program and was successfully evaluated during the flight. The fit method worked and it was possible to roughly determine the orbital period and height. Furthermore, the following assumptions could be confirmed: The inclination is directly related to the Magnet-Z component. With the camera you can determine day and night with amazing accuracy. But there were also factors that were responsible for inaccuracies. The time intervals of the measurements were unexpectedly long. In addition, the geomagnetic field is burdened with a significantly higher degree of deflection in some areas, which is why, in an improved experiment, the geomagnetic field should not be assumed to be a bar magnet, but rather a data set including magnetic field anomalies should be used. In addition, an earth rotation of 45 degrees during the experiment is not insignificant and should therefore not be neglected in further experiments. However, this would complicate the mathematical model considerably, since the orbit is then no longer a trigonometric function. The selected parameters could also be supplemented to increase the accuracy and flexibility (see Fig. 3). In summary, we were able to show that with such simple means (Raspberry Pi with Sense HAT and camera = 80 euros) the orbit and the position of the ISS can be determined amazingly well. However, the smallest inaccuracies lead to significant position deviations in space due to the high speeds, which is why our program does not solve the problem from the introduction. But it is certainly „a small step ... “[10].

parameter	**true**	**calculated**
orbital period	93min	94,1min
height	320-410km	475,7km
inclination	51,5°	52,26°
longitude of the minimum	92-136°	96,25°

Figure 19: result

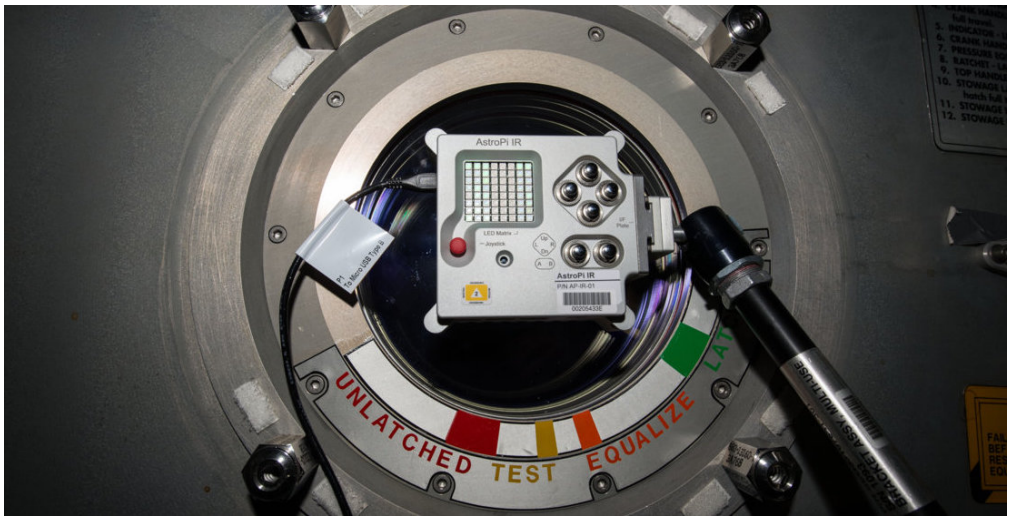


Figure 20: The Astro Pi in the International Space Station

Notes

[1] The team name Milliways Regulars means Milliways regular customers and is an allusion to the hitchhiker's guide to the galaxy books by Douglas Adams. Millyways is the restaurant at the end of the universe in these. „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! This is just a note!

[3] Of course there are now solutions that minimize this problem. In principle, the so-called redundant systems are simply the same system several times. The hard disks are always aligned and faults are eliminated well. However, there is always a small residual risk. [13]

[4] It is assumed here that the ISS and the space debris collide head-on at a speed of 7 km / s each:

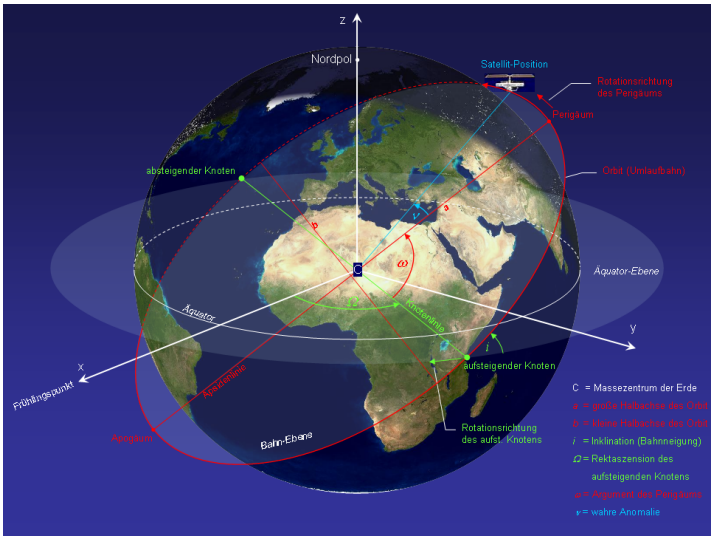
$$E_{kin} = \frac{1}{2}mv^2 = \frac{1}{2} \cdot 0,001 \cdot 14000^2 = 1.000.000$$

[5] The picture shows an impact of space junk at about 6.8 km / s in an 18cm thick aluminum block. The ball was only 1.2 cm wide and weighed only 1.7 g. The temperatures that occurred during this impact were higher than those that are in the center of the earth.

[6] The two-line element set is a variant for defining an orbit:

```
1 ISS (ZARYA)
2 1 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
```

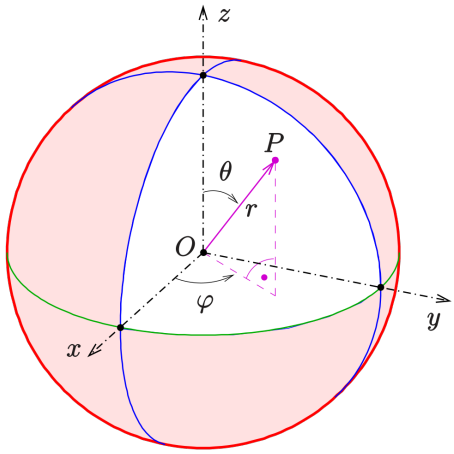
On the picture (Wikimedia Cummons) you can see all parameters again.



[7] In this context, the node is the name for the point at which the satellite crosses the equator to the north. The angle of the node therefore indicates how many degrees the orbit is rotated. However, we use ϕ for our calculations, which corresponds to the angle of the low point and is therefore similar to the angle of the node.

[8] A sphere with a radius of one around the zero point of a vector space.

[9] Polar coordinates, in particular spherical coordinates, are an alternative variant of representing points in space. Instead of the 3 coordinates (x, y, z), the sphere coordinates have the radius r (which is not included in our calculations since it is always one because we are working on the unit sphere), the angle ϕ and the angle θ. Their arrangement can be seen in the picture. (Wikimedia Cummons)



[10] Since the trigonometric functions (sine, cosine) are defined on the unit circle, it makes sense to use them for circular movements. It doesn't matter whether you use the sine or cosine because of

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

.

[11] Documentation for SciPy can be found at: `https://www.scipy.org`

[12] A time-dependent function $f(t)$ and its Fourier transform $F(\ omega)$ are thus linked [5]:

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

with

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

[13] Phi is not zero. Only the error that occurred when we used Phi = 0 corresponded to the real Phi.

[14] Documentation from Matplotlib can be found at: `https://matplotlib.org`

[15] It is impossible to reproduce a sphere perfectly on a two-dimensional map. For centuries, scientists have therefore tried to develop various maps that reflect reality as closely as possible. The best known projection is probably the Mercator projection. This has the advantage that routes are preserved. The square plate map used here (rectangular projection), on the other hand, has the advantage that the graticule corresponds to a Cartesian coordinate system and we can therefore map the orbit and the reference points directly.

References

[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] RasperryPi 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).

List of Figures