Politechnika Warszawska

WYDZIAŁ MECHANICZNY ENERGETYKI I LOTNICTWA

Orbital Mechanics

Project

Program for International Space Station observations

Author

Daria Pączkowska

Supervisor

dr Mateusz Żbikowski

WARSZAWA 2018

Contents

1. Introduction

1.1. Objectives

The aim of this project was to create a program for determination of the International Space Station (ISS) visibility above defined observer in a wide range of time. Inspiration for this project was author's passion for sky objects observations. There are available many online applications for computing the ISS and artificial satellites position and visibility, but they have limited functionality. Creating one's program is a great opportunity to learn more about satellites observations, extend typical functions of web applications and adapt program to one's interests.

The project ISSobserver.py is created in Python programming language with an open-source package PyEphem, which provides basic astronomical computations. As the object of observation was chosen the International Space Station, because it can be easily observed passing overhead. In the future program operation may be extended for bright artificial satellites. The next step is to create an application with Graphical User Interface.

Fig.1 The long-exposure photograph of the ISS pass. [1]

1.2. International Space Station

The International Space Station (ISS) is a space station, or a habitable artificial satellite, in low Earth orbit. Its first component - module Zarya was launched into orbit in 1998. The station was expanded with subsequent modules until 2011. The ISS is expected to be used

until 2028. A six-person expedition crew typically stays four to six months aboard the ISS, which serves as a microgravity and space environment research laboratory in which crew members conduct experiments in various fields. The station is suited for the testing of spacecraft systems and equipment required for missions to the Moon and Mars.

The station is the largest human-made body in low Earth. The ISS reflect sunlight and with the best configuration can be seen from Earth at magnitude from -4.5, which is as bright as planet Venus to -6, which is 16 times brighter than Sirius the brightest star in the night sky. [2]

1.2.1. The ISS structure

The ISS was built in orbit during more than 40 missions. It consists of modules and connecting nodes that contain living quarters and laboratories, as well as exterior trusses that provide structural support, and solar panels that provide power. Robotic arms are mounted outside the space station. The robot arms earlier were used to help build the space station and now can move astronauts around when they go on spacewalks or operate science experiments. The ISS weighs almost 400 tonnes. Its total length is 58.5 meters and total height is 30.5 meters. The integrated truss length is 109 meters. Eight solar arrays with length of 73 meters generate 84 kilowatts of power. [3]

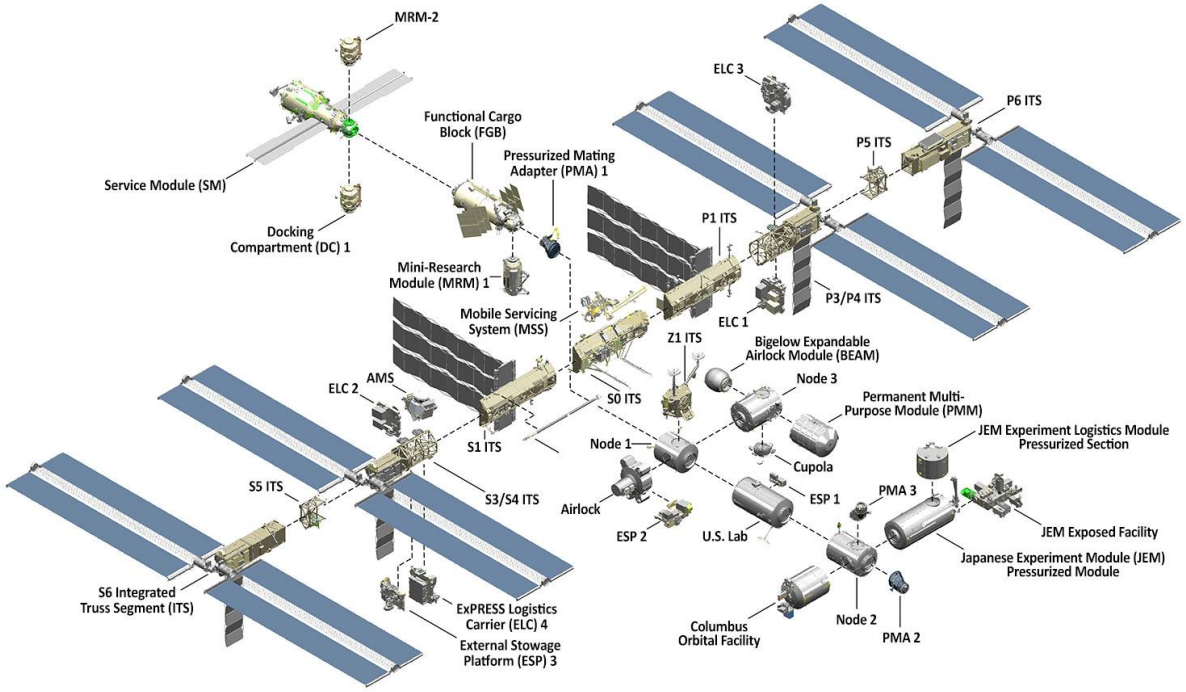


Fig. 2 The ISS modular structure. [2]

1.2.2. The ISS orbit

The ISS is maintained in a nearly circular orbit with a minimum mean altitude of 330 km and a maximum of 410 km, at an inclination of 51.6 degrees to Earth's equator. It travels at an average speed of 27,724 kilometres per hour and completes 15.54 orbits per day - 93 minutes per orbit. Height of the ISS orbit is changing due to atmospheric drag, which must be corrected with usage of engines. The rate of the ISS descending is not steady during time but depend on density of external atmosphere layers, which depend on Sun activity.

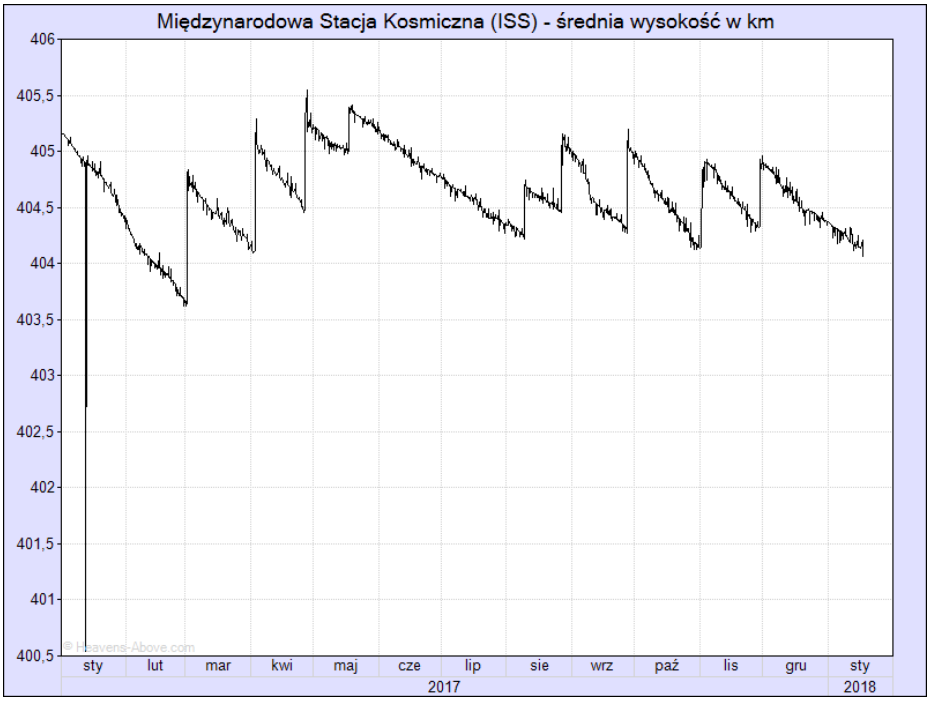


Fig. 3 Changes of the ISS orbit during last year. [4]

2. Artificial objects observations

2.1. Position determination

The position of artificial satellites and space stations can be determined based on general access data created by the United States Air Force, which tracks all detectable objects in Earth orbit, creating a corresponding files with data known as TLE - a two-line element set. TLEs are text files with each set of elements written to two 70-column ASCII lines. TLE contains a list of orbital elements of an Earth-orbiting object for a given point in time, the *epoch*. Using suitable prediction formula, the state (position and velocity) at any point in

the past or future can be estimated to some accuracy. A data set corresponding to one object include in fact three lines of data: a title line preceding the element data and two lines of data. Below is presented example of satellite TLE data set with all data explained. TLE data are available online, for example at website CelesTrack. [5]

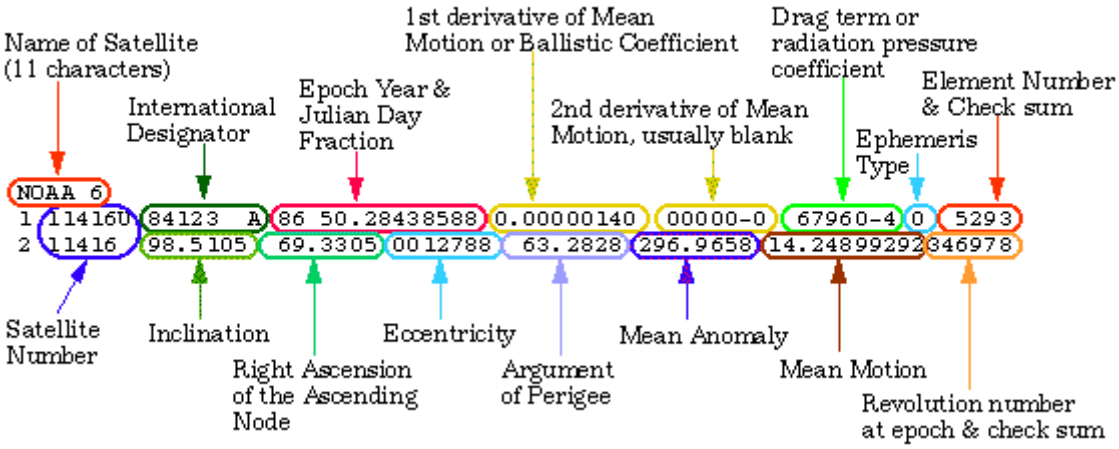


Fig. Example of satellite NOAA 6 TLE set and explanation of numbers. [6]

2.2.Visibility determination

If satellite position is known from TLE data the visibility of satellite must be specified. The requirements which must be met are:

- the satellite is above the observer's horizon,
- the sun is below the observer's horizon enough to darken the sky,
- the satellite is illuminated by the sun.

To determine whether a satellite is above the observer horizon it is necessary to compute satellite look angles in local horizontal coordinate system:

- elevation angle or altitude angle - is the angle between the object and the observer's local horizon. For visible objects it is an angle between 0 degrees and 90 degrees.
- azimuth angle - is the angle of the object around the horizon, usually measured from the north increasing towards the east.

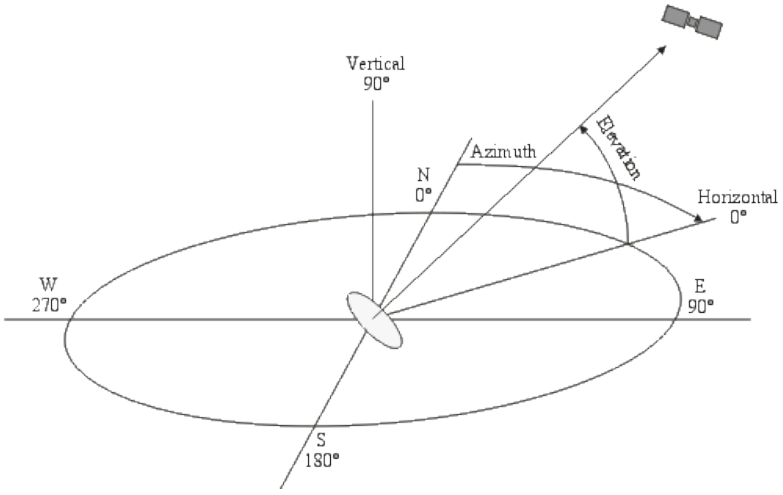


Fig. Elevation and azimuth angle in local horizontal plane.

Because of terrestrial obstacles satellites are not visible when elevation angle is equal to zero but usually when it reaches an elevation around 10 degrees. The condition for satellite visibility can be described as:

$$10° < elevation\ angle/altitude < 80°$$

The next requirement for observing artificial objects is Sun position related to the observer. The point when the sun's centre is 50 arcminutes below the horizon is considered to be the official time of sunset or sunrise and is the beginning or end of civil twilight. However, the sky is still quite bright at this point and no satellites can be seen. The point when satellites begin to become visible marks the beginning of nautical twilight—when the sun's centre is 6 degrees below the observer's horizon. The sun's effect on the night sky continues until the end of astronomical twilight when it reaches 18 degrees below the horizon and the indirect light from the sun is less than the contribution from starlight. However the sky brightness and atmospheric refraction are influenced by actual meteorological conditions, so these angles may vary. Approximated requirement for satellite visibility due to sun - observer position:

$$-18° < sun\ elevation < -6°$$

The last important requirement, which must be met is that satellite must be illuminated by the sun. There are two areas of shadow that trail the earth in the anti-solar direction. That cone where no portion of the sun's surface can be seen is known as the umbra. The angle of this cone is somewhat affected by atmospheric refraction and may vary. The shadow cone

where only part of the sun's disk is obscured by the earth is referred as the penumbra. This shadow is not a sharp transition, but rather a gradual transition from full sunlight to full darkness. For near-earth satellites, the transition across the penumbra occurs fairly quickly due to the smaller size and faster speed of the satellites.

To determine whether a satellite is in umbral or penumbral eclipse the distances from the satellite to the earth (ρ_E), the satellite to the sun (ρ_S), and the earth to the sun (r_S) in ECI coordinates must be known.

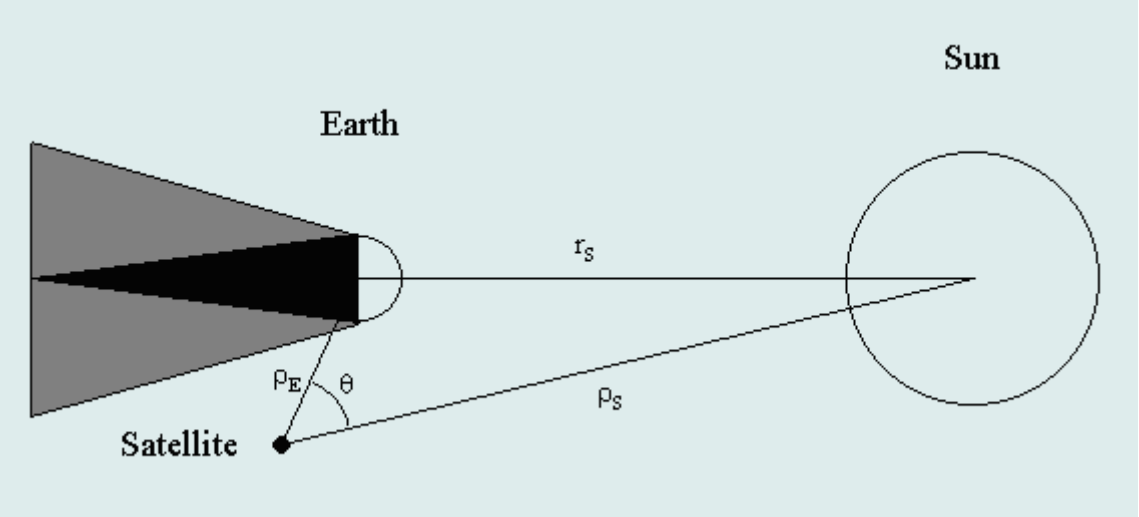


Fig. Satellite eclipse geometry. [5]

The next step is to determine the semidiameters of the earth and the sun (θ_E and θ_S, respectively) and the angle between the centre of the earth and the sun (θ), all from the vantage point of the satellite.

$$\theta_E = \sin^{-1}(R_E/\rho_E)$$

$$\theta_S = \sin^{-1}(R_S/\rho_S)$$

where R_E and R_S are the radii of the earth and the sun, respectively.

The angle between the centre of the earth and the sun:

$$\theta = \cos^{-1}(\boldsymbol{\rho}_E \bullet \boldsymbol{\rho}_S/\rho_E\rho_S)$$

where the numerator is the vector dot product of the two distance vectors.

The condition for an umbral eclipse:

$$\theta_E > \theta_S \qquad \theta < \theta_E - \theta_S$$

The conditions for a penumbral eclipse:

$$|\theta_E - \theta_S| < \theta < \theta_E + \theta_S$$

If the semidiameter of the earth is larger than the sun under these conditions, a partial eclipse occurs.

3. Project ISSobserver.py

3.1.Programming language

The program for ISS observations was created in Python 3.6 programming language with usage of editor Spyder. For orbital mechanics computations was used Python package PyEphem. According to [7] :

PyEphem provides basic astronomical computations for the Python programming language. Given a date and location on the Earth's surface, it can compute the positions of the Sun and Moon, of the planets and their moons, and of any asteroids, comets, or earth satellites whose orbital elements the user can provide. Additional functions are provided to compute the angular separation between two objects in the sky, to determine the constellation in which an object lies, and to find the times at which an object rises, transits, and sets on a particular day.

3.2.Program structure

The ISSobserver.py has a typical simple structure. The program tasks are divided into functions:

- input_location() - function enables user to choose location for ISS observation, there are three default locations and an option for user defined coordinates: longitude, latitude and elevation. Another input parameter is number of ISS passes to calculate and

parameter defining if all passes or only visible passes will be shown. In the figure below is presented structure of input data;

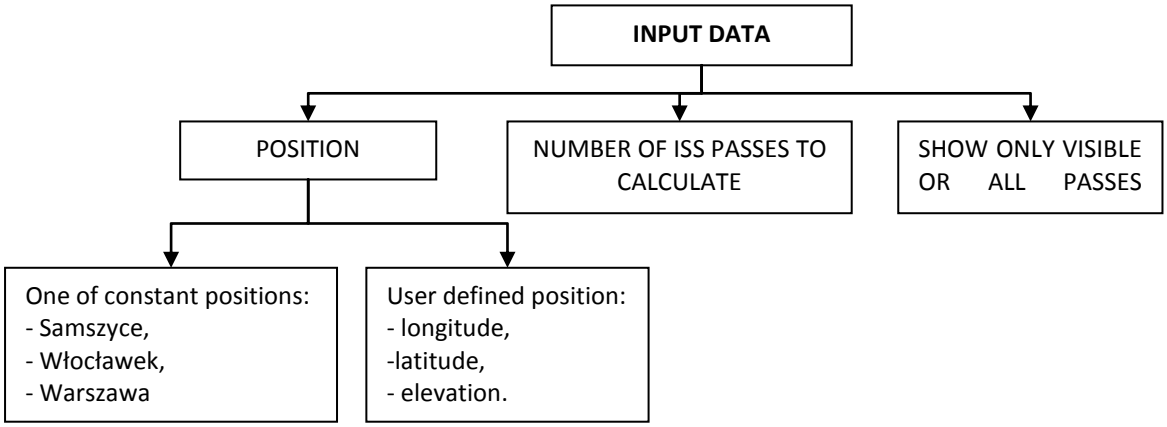


Fig. Input data structure.

- iss_tle() - function to import tle data form [5] , save it to program directory and retrieve ISS data for subsequent calculations;
- iss_position() - function, which calculated ISS passes above specified observer with PyEhpem's package module observer.next_pass();
- iss_visibility() - function consists of conditions described in chapter 2.2 to define if ISS pass is visible or why it is invisible, for eclipse calculations a PyEhpem's package module .eclipsed is used;
- iss_in_the _sky() - function, which creates polar plot representing sky map with ISS transition;
- strfdelta() - function which transforms data format to separate components, which allows to specify in output() function suitable, better readable data format;
- output() - function which shows user calculated data and information in a nice form, to make output nicer the module ansicolors [8] was installed to color console's output.

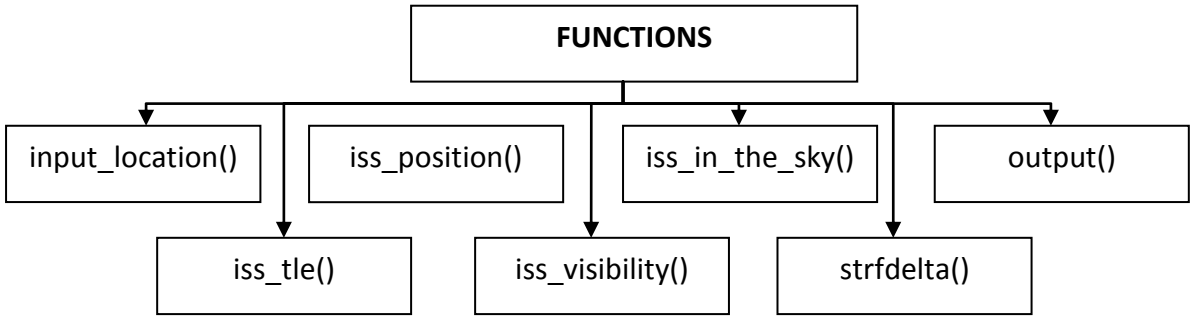


Fig. Scheme with program functions.

4. The ISS observations examples

4.1. Case 1 - First visible pass

The program calculates the first visible ISS pass for observer in Warsaw. The ISS is not visible above Warsaw until 27-01-2018 at 18:00.

```
====================================
ISS pass is visible
------------------------------------
Rise time: 2018-01-27 18:00:25
Set time: 2018-01-27 18:10:23
Max altitude: 18.54 deg
Rise azimuth: 210.48 deg
Set azimuth: 79.02 deg
Duration: 9 min 0 s
------------------------------------
```

Fig. First part of ISSobserver.py program output.

Maximum altitude is 18,54^{o} - the ISS is low above the horizon so this pass will not be good for observations.

The second part of program output is a sky map - the ISS pass projection in the polar plot with reversed West and East. Below is presented comparison of ISSobserver.py output and sky map from online website Heavens Above [4].

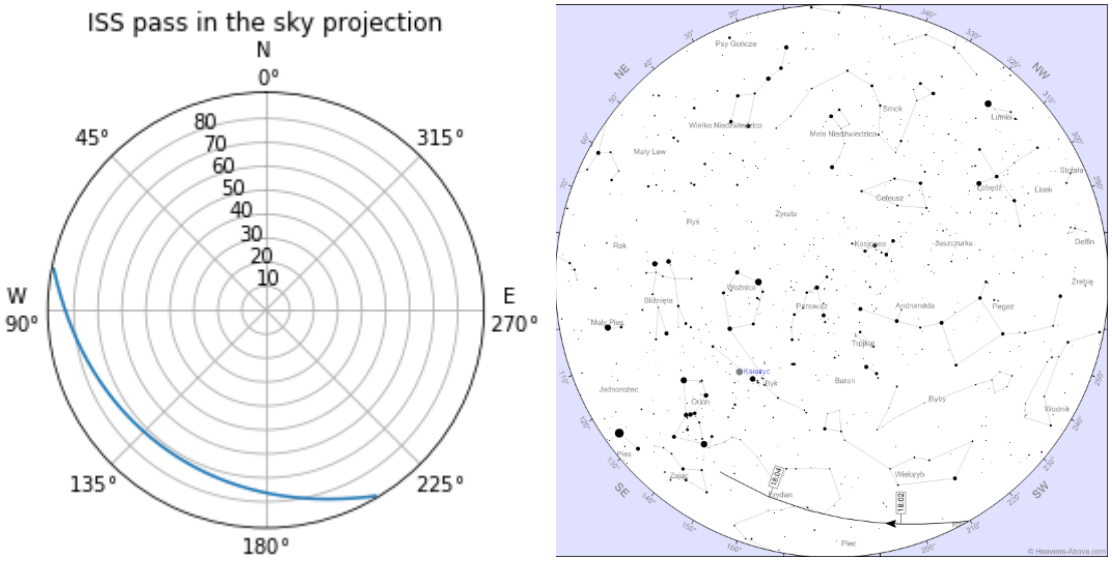


Fig. The ISS pass on 27-01-2018 at 18:00 projected on the sky map, left: output from ISSobserver.py, right: output from website [3]

4.2.Case 2 - First occurred pass

The first pass which occurred from date of starting program was 18-01-2018 at 1:07 am. This pass is not visible due to the Sun position. Sun is two low below horizon.

```
------------------------------------------------------------
ISS pass is not visible because of Sun position
Sun altitude = -52.2762598317
------------------------------------------------------------
Rise time: 2018-01-18 01:07:20
Set time: 2018-01-18 01:18:20
Max altitude: 81.67 deg
Rise azimuth: 271.00 deg
Set azimuth: 95.86 deg
Duration: 11 min 0 s
------------------------------------------------------------
```

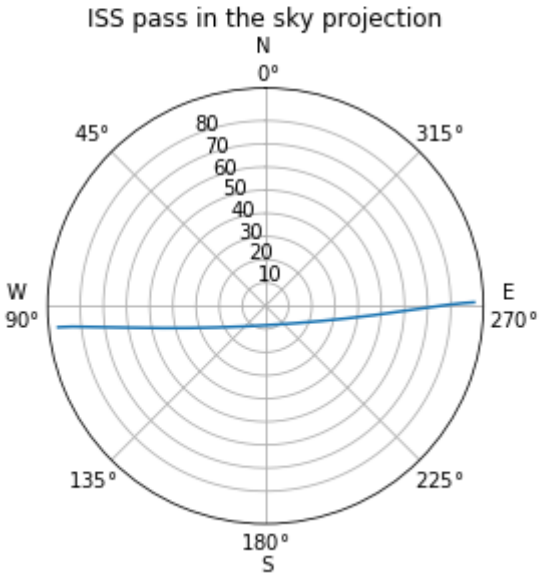
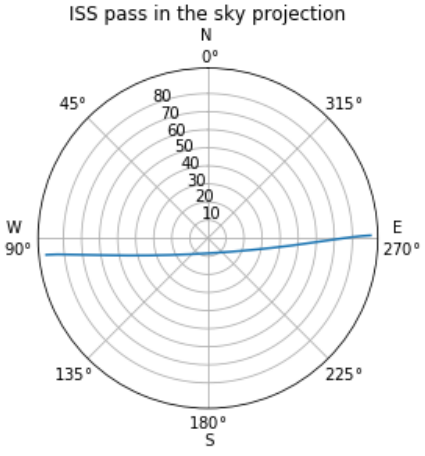


Fig. ISSobserver.py program output.



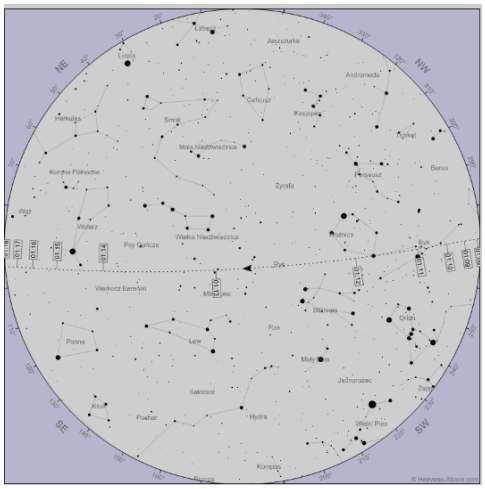


Fig. The ISS pass on 18-01-2018 at 1:07 am projected on the sky map, left: output from ISSobserver.py, right: output from website [3]

5. Summary

Created program ISSobserver.py allows user to check when ISS pass will occur above and if it will be visible or which requirement for pass visibility is not meet. The advantage of this program over many website applications is that shows all passes which user want to check with all information and sky map. Few applications also can tell why the pass is not visible which is very interesting for many astronomical observations amateurs. Creating of this program was a great opportunity to learn python programming and extend knowledge in orbital mechanics. The next step to evolve the program should be creation of graphical user interface with for example TKInter [9].

Bibliography

[1] “Observing satellites – Satellite Observation.” [Online]. Available: https://satelliteobservation.wordpress.com/2017/04/20/observing-satellites/. [Accessed: 17-Jan-2018].

[2] M. Garcia, “Facts and Figures ISS,” 2016. [Online]. Available: https://www.nasa.gov/feature/facts-and-figures. [Accessed: 17-Jan-2018].

[3] “Boeing: International Space Station.” [Online]. Available: http://www.boeing.com/space/international-space-station/. [Accessed: 16-Jan-2018].

[4] “Heavens-Above.” [Online]. Available: http://www.heavens-above.com/?lat=52.6483&lng=19.0677&loc=Włocławek&alt=60&tz=CET. [Accessed: 16-Jan-2018].

[5] “CelesTrak: NORAD Two-Line Element Set Format.” [Online]. Available: https://www.celestrak.com/NORAD/documentation/tle-fmt.asp. [Accessed: 17-Jan-2018].

[6] “Human Space Flight (HSF) - Realtime Data.” [Online]. Available: https://spaceflight.nasa.gov/realdata/sightings/SSapplications/Post/JavaSSOP/SSOP_Help/tle_def.html. [Accessed: 16-Jan-2018].

[7] “Welcome! — PyEphem home page.” [Online]. Available: http://rhodesmill.org/pyephem/. [Accessed: 17-Jan-2018].

[8] “ansicolors 1.1.8 : Python Package Index.” [Online]. Available: https://pypi.python.org/pypi/ansicolors. [Accessed: 23-Jan-2018].

[9] “TkInter - Python Wiki.” [Online]. Available: https://wiki.python.org/moin/TkInter. [Accessed: 23-Jan-2018].

1. Annex I

```
"""
Created 28-12-2017
Author: Daria Pączkowska
Program contains functions:
- iss_tle() - loads ISS TLE data
- input_location() - user input of observer location, number of calculated ISS
    passes, output only visible or all passes
- iss_position() - calculating of ISS position realtive to defined observer
- iss_visibility() - return if ISS pass is visible
- iss_in_the_sky() - return ISS pass projection on schematic sky map
- strfdelta() - return defined date format
- output() - output data presentation
"""

import sys
import ephem
import requests
import numpy as np
import matplotlib.pyplot as plt
import datetime
from string import Formatter
from colors import blue, cyan, green, red, magenta

#iss tle data loading
def iss_tle():
    tle_url = 'http://www.celestrak.com/NORAD/elements/stations.txt'
    satellite_tle = requests.get(tle_url).text
    tle_file = open('TLE_data.txt','w')
    tle_file.write(satellite_tle)
    tle_file.close()
    tle_data = open('TLE_data.txt', 'r').readlines()

    tle_data = [a for a in tle_data if a != '\n']
    tle_data = ''.join(tle_data)
    tle_data = tle_data.split('\n')
    names = tle_data[0::3]
    for i,j in enumerate(names[:-1], 1):
        x = str(i).strip()
        j= str(j).strip()
```

```
        if j=='ISS (ZARYA)':
            number=x

    sat_number = int(number) * 3 - 3

    return tle_data[sat_number:sat_number+3]

# user input of iss observer position
def input_location():

    locationKeyword = input(cyan('Location keyword: ')) #type first few letters of t
he location keywords included in the entries below
    locationName = ''

    if locationKeyword.lower() == 'samszyce'[0:len(locationKeyword)].lower():
        locationName = 'Samszyce'
        print()
        print('Observer in Samszyce')
        obs = ephem.Observer()
        obs.lat = '52.5997'
        obs.long = '18.6986'
        obs.elev = 92
    elif locationKeyword.lower() == 'włocławek'[0:len(locationKeyword)].lower():
        locationName = 'Włocławek'
        print()
        print('Observer in Włocławek')
        obs = ephem.Observer()
        obs.lat = '52.6483'
        obs.long = '19.0677'
        obs.elev = 60
    elif locationKeyword.lower() == 'warszawa'[0:len(locationKeyword)].lower():
        locationName = 'Warszawa'
        print()
        print('Observer in Warszawa')
        obs = ephem.Observer()
        obs.lat = '52.2297'
        obs.long = '21.0122'
        obs.elev = 113
    elif locationKeyword.lower() == 'userdefined'[0:len(locationKeyword)].lower():
        locationName = 'UserDefined'
        print()
        print(green('Observer position defined by user:'))
        obs = ephem.Observer()
        check = True
        while check:
            obs.lat = input(cyan('Observer latitude [xx] N or [-xx] S: '))
            if int(obs.lat) in range(-90,90):
                check = False
            else:
                print(red('Value is not in range (-90,90)'))
        check = True
        while check:
            obs.long = input(cyan('Observer longitude [xx] E or [-xx] W: '))
            if int(obs.long) in range(-180,180):
                check = False
            else:
                print(red('Value is not in range (-180,180)'))
        obs.elev = int(input(cyan('Observer elevation: ')))
    else:
        print()
        print(red('Location keyword not found.'))
            sys.exit()

        print()
        print(green('Enter number of ISS passes'))
        nr = int(input(cyan('Number of passes: ')))
```

```
105.           print()
106.           print(green('Show all passes or just visible?'))
107.           check = True
108.           while check:
109.               pas = input(cyan('Type "all" or "v": '))
110.               if pas in ['all', 'v']:
111.                   check = False
112.               else:
113.                   print(red('Error. Enter "all" or "v"'))
114.
115.           return locationName, obs.lat, obs.long, obs.elev, nr, pas
116.
117.       # calculations of iss position relative to the observer
118.       def iss_position(iss, obs):
119.
120.           tr, azr, tt, altt, ts, azs = obs.next_pass(iss)
121.           rise_time = tr.datetime()
122.           set_time = ts.datetime()
123.           duration = abs(set_time-rise_time)
124.           eclip = 0
125.           iss_alt, iss_az = [], []
126.           dt = datetime.timedelta(seconds=1)
127.           date = rise_time
128.           eclip_check = True
129.           while (date < set_time):
130.               obs.date=date
131.               iss.compute(obs)
132.               iss_alt.append(np.rad2deg(iss.alt))
133.               iss_az.append(np.rad2deg(iss.az))
134.               eclip = iss.eclipsed
135.               if eclip == False:
136.                   eclip_check = False
137.               date = date + dt
138.
139.           obs.date = tt.datetime()
140.           obs.date = tr + (25 * ephem.minute)
141.
142.           return tr, azr, altt, ts, azs, duration, iss_alt, iss_az, eclip_check
143.
144.       # checking if iss pass is visible
145.       def iss_visibility(obs, iss, altt, nr, pas, i, eclip_check):
146.
147.           sun = ephem.Sun(obs)
148.           sun_alt = np.rad2deg(sun.alt)
149.           visible = False
150.           if eclip_check is False and -20 < sun_alt < -
    5 and np.rad2deg(altt) >= 10:
151.               visible = True
152.               print()
153.               print(blue('=' * 60))
154.               print(blue('ISS pass is visible'))
155.               print(cyan('-' * 60))
156.
157.           if pas == 'all':
158.               if eclip is True:
159.                   print(magenta('=' * 60))
160.                   print('ISS pass is not visible because ISS is eclipsed')
161.                   print(cyan('-' * 60))
162.               elif np.rad2deg(altt) < 10:
163.                   print(magenta('=' * 60))
164.                   print('ISS pass is not visible because its altitude is below 10 d
    eg')
165.                   print(cyan('-' * 60))
166.               elif sun_alt < -20 or sun_alt > -5:
167.                   print(magenta('=' * 60))
168.                   print('ISS pass is not visible because of Sun position')
```

```
                    print('Sun altitude = %s ' % sun_alt)
                    print(cyan('-' * 60))

            return(visible)

        # plotting iss pas in the sky projection
        def iss_in_the_sky(iss_alt, iss_az):

            r = 90-np.array(iss_alt)
            theta = np.deg2rad(iss_az)
            sky = plt.subplot(111, projection='polar')

            sky.plot(theta, r)
            sky.set_theta_zero_location("N")
            s = np.arange(10,90,10)
            sky.set_rticks(s)
            sky.set_xticklabels([ 'N \n $0\degree$', '$45\degree$', 'W \n $90\degree$', '$135\degree$', '$180\degree$ \n S', '$225\degree$', 'E \n $270\degree$', '$315\degree$'])
            ttl = sky.set_title("ISS pass in the sky projection", va="bottom")
            ttl.set_position([0.5, 1.1])
            plt.show()

            return()

        # date formating
        def strfdelta(tdelta, fmt):
            f = Formatter()
            d = {}
            l = {'D': 86400, 'H': 3600, 'M': 60, 'S': 1}
            k = list(map( lambda x: x[1], list(f.parse(fmt))))
            rem = int(tdelta.total_seconds())

            for i in ('D', 'H', 'M', 'S'):
                if i in k and i in l.keys():
                    d[i], rem = divmod(rem, l[i])

            return f.format(fmt, **d)

        # output data presentation
        def output(tr,ts, altt, azr, azs, duration):
            tr_local = ephem.localtime(tr)
            ts_local = ephem.localtime(ts)

            print('Rise time: %s' % tr_local.isoformat(sep=' ', timespec='seconds'))

            print('Set time: %s' % ts_local.isoformat(sep=' ', timespec='seconds'))
            print('Max altitude: %.2f deg' % np.rad2deg(altt))
            print('Rise azimuth: %.2f deg' % np.rad2deg(azr))
            print('Set azimuth: %.2f deg' % np.rad2deg(azs))
            print(strfdelta(duration, "Duration: {M} min {D} s"))
            print(cyan('-' * 60))

            iss_in_the_sky(iss_alt, iss_az)
            return()

        # Main program

        print(blue('='*60))
        print(blue('Program for ISS observations'))
        print(blue('='*60))
        print()
        print('-'*60)
        print(blue('Program shows parameters of ISS passes above defined observer'))
```

```
230.       print(blue('ISS position is calculated based on TLE data from: \nhttps://www.
   celestrak.com/NORAD/elements/stations.txt'))
231.       print('-'*60)
232.
233.       # loading tle data
234.       tle_data=iss_tle()
235.       tle_data1 = " ".join(tle_data[0].split())
236.       tle_data2 = tle_data[1]
237.       tle_data3 = tle_data[2]
238.       iss = ephem.readtle(tle_data1, tle_data2,tle_data3)
239.
240.       # observer calculations
241.       obs = ephem.Observer()
242.
243.       print()
244.       print(green('Choose your position from defined positions or enter your coordi
   nates.'))
245.       print()
246.       print(green('Type first few letters of selected option:'))
247.       print()
248.       print(green('Samszyce, Włocławek, Warszawa, UserDefined'))
249.
250.       locationName, obs.lat, obs.long, obs.elev, nr, pas = input_location()
251.
252.       # setting as starting date for observation 'now'
253.       now = datetime.datetime.utcnow()
254.       obs.date= now
255.
256.       for i in range(nr):
257.
258.           tr, azr, altt, ts, azs, duration, iss_alt, iss_az, eclip_check = iss_posi
   tion(iss, obs)
259.
260.           if pas == 'v':
261.               visible = iss_visibility(obs, iss, altt, nr, pas, i, eclip_check)
262.               if visible == True:
263.                   output(tr,ts, altt, azr, azs, duration)
264.               else:
265.                   if i == nr-1:
266.                       print()
267.                       print(magenta('No visible ISS passes occure'))
268.           else:
269.               visible = iss_visibility(obs, iss, altt, nr, pas, i, eclip_check)
270.               output(tr,ts, altt, azr, azs, duration)
271.
272.       # if any iss pass is visible calculate until visible pass will occure
273.       if pas == 'v':
274.           if visible == False:
275.               print()
276.               print("You have to wait for visible ISS pass until:")
277.               check = True
278.               while check:
279.                   tr, azr, altt, ts, azs, duration, iss_alt, iss_az, eclip_check =
   iss_position(iss, obs)
280.                   visible = iss_visibility(obs, iss, altt, nr, pas, i, eclip_check)
281.                   if visible == True:
282.                       check = False
283.                   else:
284.                       check = True
285.               output(tr,ts, altt, azr, azs, duration)
```