Space Hardware

Laboratory P5

Integrating high level classes/functions into a completed application and debugging

Feras Yahya
Mark Lopez
Rajika Pati Arambage

Introduction

The purpose of P5 is to perform further testing and add any final changes and adjustments to the software. Specifically, the report will include any changes done after P4 as well as any additions and their purpose. The report will also determine whether the software is fully completed as well as any uncertainties within the software. **NOTE: a file containing all the modules will be attached to this document**

Changes after P4

Change	Location	Description
Adjusted function 1.13 (Validate)	Low-Level functions (dateAndTimeCalculations.py)	Alongside determining whether the format is valid, the adjustment allows validate to determine whether the date entered is valid itself. The requirement for each argument is as follows: Year must be between 2010 and 2020 Month must be between 1 and 12 Day must be between 1 and 31 Hour must be between 0 and 23 Minute must be between 0 and 59 Second must be between 0 and 59
Adjusted the Visibility Module such that it considers the link variables which can be inputted manually by the user	Low-Level functions (visibilityModule.py)	The purpose of the adjustment is to simplify the calculation process in case the link variables change
Adjusted the Tracking Module such that it considers the link variables which can be inputted manually by the user	Low-Level functions (TrackingData.py)	The purpose of the adjustment is to simplify the calculation process in case the link variables change
Few adjustments were made to the inputs for the Link calculations module	Low-Level functions (LinkCalculations.py)	These adjustments should provide a more accurate value for the power received
Adjusted the UI in the UserInputParser.py file. The adjustments are as follows: 1. Validating time step input 2. Prompting user to input link variables 3. Prompting user to choose coordinate system for ephemeris file	High-Level function (UserInputParser.py)	The UI was adjusted and made more user-friendly.

Low-Level functions

The following python files contain low-level functions that are used multiple times throughout the software. Such low-level functions are useful since they omit the need to re-write these calculations over and over (Changes and additions are highlighted).

Python:

1) dateAndTimeCalculations.py

- 1.1. frac(value)
  - *1.1.1. The function returns the fraction of the given value.*
- 1.2. isLeapYear(YR)
  - *1.2.1. The function determines whether a year is a leap year or not.*
- 1.3. daysInMonth(YR)
  - *1.3.1. The function adjusts the number of days in a month for a leap year and for a non-leap year.*
- 1.4. doy(YR,MO,D)
  - *1.4.1. The function determines the number of days for a given date.*
- 1.5. frcofd(HR,MI,SEC)
  - *1.5.1. The function calculates the fraction of day at the specified input time.*
- 1.6. ep2dat(JulainDate)
  - *1.6.1. The function converts the epoch date, given in Julians, to standard Gregorian date.*
- 1.7. curday()
  - *1.7.1. The function returns the current day in utc and date at the time the function is called.*
- 1.8. timeSinceEpoch(epoch,YR,MO,D,HR,MIN,SEC,MICS)
  - 1.8.1. The function returns the time (in seconds) since epoch given a specific tracking time
- 1.9. gregToJulian(YR,MO,D,HR,MIN,SEC)
  - *1.9.1. The function returns the Julian date given the Gregorian date*
- 1.10. TimeSinceJ2000(YR,MO,D,HR,MIN,SEC,MICS)
  - *1.10.1. The function returns the time (in seconds) since J2000 given a specific tracking time*
- 1.11. formatDate(date)
  - *1.11.1. The function returns the year, month, date, hours, minutes and seconds of a date of the following format: YYYY-MM-DD HH:MM:SS*
- 1.12. TimeBetweenTwoDates(date1, date2)
  - *1.12.1. The function returns the interval in seconds between two dates of the following format: YYYY-MM-DD HH:MM:SS*
- *1.13. validate(date)*
  - *1.13.1. The function reads the date the user inputs and validates it. If the user inputs a false date, the function prints out an error and allows the user to input the date again. The function returns the validated date. The function also determines whether the year, month, date, hour, minute or seconds entered are valid. The requirement for each argument is as follows: Year between 2010 and 2020, Month between 1 and 12, Day between 1 and 31, Hour between 0 and 23, Minute between 0 and 59 and Second between 0 and 59*

1.14. *validateStopTime(date,date2)*

1.14.1. The function determines whether the stopping date entered by the user is in a valid format or not as well as whether its before or after the starting date. The function also determines whether the year, month, date, hour, minute or seconds entered are valid. The requirement for each argument is as follows:

Year between 2010 and 2020 and after the year of the starting date

Month between 1 and 12 and after the month of the starting date

Day between 1 and 31 and after the day of the starting date

Hour between 0 and 23 and after the hour of the starting date

Minute between 0 and 59 and after the minute of the starting date

Second between 0 and 59 and after the second of the starting date

2. Fileio.py

2.1. banner()

2.1.1. return banner with group info

2.2. errmsg(ERRMSG)

2.2.1. returns an error message with a beep

2.3. ReadStationFile(STNFIL)

2.3.1. Reads the given station file and validates the file name. If the user inputs an invalid file name, the function returns an error and allows the user to enter the file name until a valid file name is inputted

2.4. ReadNoradTLE (line0, line1, line2)

2.4.1. Reads the NORAD TLE file and stores all the elements in a structure for easier access

2.5. STKout(outfile,EphemFile,StartString,EpochTimeofSAT,Coord,position,velocity)

2.5.1. The function generates an Ephermis file which can be read using STK for testing purposes

3. visibilityModule.py

3.1. AOSLOS(AOSLOSfile, startTime, stopTime, TimeStep, F_cnt, AE, Diameter, Bandwidth, RCV, RNT, Pt, Gt,Ltranspath,Latm)

3.1.1. Reads all satellites in a TLE file and generates an AOSLOS table. The function inputs are the file name, start time, stop time, time step and link variables.

4. LinkCalculationsModule.py

4.1. LinkDesign(F_cnt, AE, D, B, RG, RNT, Pt, Gt,Ltranspath, Latm, rangeTopo)

4.1.1. Reads in the RF characteristics of the GPS signal and outputs the power received

5. TrackingDataModule.py

5.1. ComputePositionAndVelocity(SatName, Px, Py, Pz, Vx, Vy, Vz, startTime, stopTime, TimeStep,F_cnt, AE, D, B, RG, RNT, Pt, Gt,Ltranspath, Latm)

5.1.1. The function computes the position and velocity of the satellite at the given time intervals. The user can specify which coordinates to compute (Perifocal, ECI, ECF and Topocentric). The generated coordinates can then be inputted into the STKout function to generate the ephemeris file. The function is also used to generate the tracking data table

6. PointingFile.py

6.1. PointingFile(PointingFileName, SatToTrack, startTime, stopTime, TimeStep)

6.1.1. Generates a pointing file for a specific satellite given the start and stop time of the tracking as well as the time step.

OpenModelica

7. SatellitePackage.mo

7.1. Vector (Record)

7.1.1. Assigns the x,y,z components of a vector to a variable for easier access

7.2. Satellite (Module)

7.2.1. Computes the perifocal coordinates of a satellite by using the information provided in a TLE. The inputs are all elements in the TLE and the output is the perifocal position and velocity as vectors

7.3. GndStn (Module)

7.3.1. Computes the ECF coordinates of the ground station. The inputs are the longitude, latitude and elevation of the ground station and the output is the position of the ground station in ECF coordinates as a vector

7.4. Sat_ECI (Function)

7.4.1. Computes the ECI coordinates of the satellite. The inputs are the perifocal coordinates, both position and velocity, and the output is the ECI position and velocity as vectors

7.5. Theta_t (Function)

7.5.1. Computes the GMST angle as a function of time. The inputs are the number of days between the tracking time and J2000 and the number of hours since midnight in UTC time. The output is the GMST angle as a function of time

7.6. Sat_ECF (Function)

7.6.1. Computes the ECF coordinates of the satellite. The inputs are the ECI coordinates, both position and velocity, and GMST angle. The output is the ECF position and velocity as vectors

7.7. Range_ECF2topo (Function)

7.7.1. Computes the Topocentric coordinates of the satellite. The inputs are the ECF coordinates, both position and velocity, ECF coordinates of the ground station (computed using GndStn) and longitude and latitude of ground station. The output is the Topocentric position and velocity as vectors

7.8. Range_topo2look_angles

7.8.1. Computes the look angles (azimuth and elevation) required to point towards a certain satellite from the ground station. The inputs are the Topocentric position and velocity of the satellite and azimuth and elevation velocity limits (used for doppler calculations). The output is the azimuth and elevation angles required to point towards the given satellite.

High-Level functions

The User Input Parser file is responsible for combining all the low-level function together. The file can read all the user inputs which will be used to perform all the required computations to produce the pointing file. It will also be implemented on python. The code will perform major the calculations in the following order (All changes are highlighted):

Step	Function	Purpose
Print the group banner	The banner is printed using function 2.1	N/A
Read the station file and print the station information	The file is read and printed using function 2.3. The user input is also validated using the same function	Print the file information of the station the user will be operating
Define the starting time of tracking	Inputted manually by the user	Start and stop time will define the duration for which the calculations will be performed for
Validate the starting time	Function 1.13 will be used to validate the user input	A false tracking time can cause the software to produce an incorrect output. Therefore, it is crucial that the software recognizes when a user has inputted a false date
Define the stopping time of tracking	Inputted manually by the user	Start and stop time will define the duration for which the calculations will be performed for
Validate the stopping time	Function 1.14 will be used to validate the user input	A false tracking time can cause the software to produce an incorrect output. Therefore, it is crucial that the software recognizes when a user has inputted a false date
Define the time step to be used	Inputted manually by the user	The time step determines how accurate the calculations are. Smaller time steps produce more accurate results. However, they require longer run times

Validate time step	Using a while loop	If an invalid time step is entered, a while loop will loop until a correct time step is entered
Define the link variables	Inputted manually by the user	The link variables are used in calculating the power received
Calculate perifocal coordinates (Calculations)	Satellite.mo	Perifocal coordinates are used to calculate ECI coordinates which are required for ECF and so on.
Calculate ECI coordinates (Calculations)	Sat_ECI.mo	ECI coordinates are required to compute the ECF coordinates which are required to compute Topocentric coordinates
Calculate GMST angle (Calculations)	Theta_t.mo	GMST angle is required to calculate the ECF coordinates
Calculate ECF coordinates (Calculations)	Sat_ECF.mo	ECF coordinates are required to compute Topocentric coordinates
Calculate ECF coordinates of Ground station (Calculations)	GndStn.mo	ECF coordinates of ground station are required to calculate the topocentric coordinates of the satellite
Calculate Topocentric coordinates (Calculations)	Range_ECF2topo.mo	Topocentric coordinates are required to compute the look angles
Calculate the look angles (Calculations)	Range_topo2look_angles.mo	Look angles are used to produce the pointing file which is used to point the satellite dish at the satellite being tracked
Generate the AOSLOS file	visiblityModule.py	The AOSLOS file provides information about the satellite visible in the chosen time interval
Choose the satellite to track	Inputted manually by the user after all the satellites are read from the TLE file	Once a specific satellite has been chosen to be tracked, an ephemeris file can be generated for testing purpose
Prompt user to input coordinate system for ephemeris file	Inputted manually by user	This allows the user to choose the coordinate system for their ephemeris file (can be used for testing). The user input is validated using a while loop. Inputting a false coordinate system will re-prompt the user to enter a coordinate system until a correct input is received

Generate ephemeris file based on coordinate system chosen by user	STKout(outfile,EphemFile, StartString,EpochTimeofSAT, Coord,position,velocity)	The ephemeris file can be used to validate the information generated by the code by comparing it to the actual satellite in STK
Generate the link budget	LinkCalculationsModule.py	The link budget will provide information about the power to be seen on the spectrum analyzer when tracking the satellite
Generate Tracking data	TracingDataModule.py	The tracking data module provides a table of the antenna state during the tracking interval
Generate sensor pointing file	SensorPointingFile.py	The sensor pointing file can be used as means of validating the software. The file can be read using STK.
Generate the pointing file	PointingFile.py	The pointing file will be used to point the satellite dish to the satellite

Source code

The following section provides the source code for the changes that were done after P4.

dateAndTimeCalculations.py

- Validate(date)

```
def validate(date):

    buffer = False
    tries = 0
    notValidated = True
    while buffer != True:
        if tries > 0:
            date = input("TRY AGAIN (Format: YYYY-MM-DD HR:MM:SS)\n")
            notValidated = True

        if notValidated != False:
            try:
                YR = int(date[0:4])
                MO = int(date[5:7])
                D  = int(date[8:10])
                HR = int(date[11:13])
                MIN= int(date[14:16])
```

```
      SEC= int(date[17:19])
      notValidated = False
    except:
      io.errmsg("Invalid format!")
      tries = tries+1
      buffer = False
      notValidated = True

  if notValidated != True:
    if YR > 2020 or YR < 2010:
      io.errmsg("Enter a valid year please! Year must be between 2010 and 2020")
      tries = tries+1
      buffer = False
    elif MO > 12 or MO < 1:
      io.errmsg("Enter a valid month please! Month must be between 1 and 12")
      tries = tries+1
      buffer = False
    elif D > 31 or D < 1:
      io.errmsg("Enter a valid day please! Day must be between 1 and 31")
      tries = tries+1
      buffer = False
    elif HR > 23 or HR < 0:
      io.errmsg("Enter a valid hour please! Hour must be between 0 and 23")
      tries = tries+1
      buffer = False
    elif MIN > 59 or MIN < 0:
      io.errmsg("Enter a valid minute please! Minute must be between 0 and 59")
      tries = tries+1
      buffer = False
    elif SEC > 59 or SEC < 0:
      io.errmsg("Enter a valid second please! Second must be between 0 and 59")
      tries = tries+1
      buffer = False
    else:
      buffer = True
      return date
```

- ValidateStopTime(date,date2)

```
def validateStopTime(date,date2):
  buffer = False
  tries = 0
  notValidated = True
  YR2 = int(date2[0:4])
  MO2 = int(date2[5:7])
  D2  = int(date2[8:10])
  HR2 = int(date2[11:13])
  MIN2= int(date2[14:16])
```

```
   SEC2= int(date2[17:19])

   while buffer != True:
      if tries > 0:
         date = input("TRY AGAIN (Format: YYYY-MM-DD HR:MM:SS)\n")
         notValidated = True

      if notValidated != False:
         try:
            YR = int(date[0:4])
            MO = int(date[5:7])
            D  = int(date[8:10])
            HR = int(date[11:13])
            MIN= int(date[14:16])
            SEC= int(date[17:19])
            notValidated = False
         except:
            io.errmsg("Invalid format!")
            tries = tries+1
            buffer = False
            notValidated = True

      if notValidated != True:
         if YR > 2020 or YR < 2010 or YR < YR2:
            io.errmsg("Enter a valid year please! Year must be between 2010 and 2020 and must be after starting date")
            tries = tries+1
            buffer = False
         elif MO > 12 or MO < 1 or MO < MO2:
            io.errmsg("Enter a valid month please! Month must be between 1 and 12 and must be after starting date")
            tries = tries+1
            buffer = False
         elif D > 31 or D < 1 or D < D2:
            io.errmsg("Enter a valid day please! Day must be between 1 and 31 and must be after starting date")
            tries = tries+1
            buffer = False
         elif HR > 23 or HR < 0 or HR < HR2:
            if HR > 23 or HR < 0:
               io.errmsg("Enter a valid hour please! Hour must be between 0 and 23")
               tries = tries+1
               buffer = False
            elif YR == YR2 and MO == MO2 and D == D2:
               io.errmsg("Enter a valid hour please! Hour must be after the hour of the starting date")
               tries = tries+1
               buffer = False
```

```
            else:
                buffer = True
                return date
        elif MIN > 59 or MIN < 0 or MIN < MIN2:
            if MIN > 59 or MIN < 0:
                io.errmsg("Enter a valid minute please! Minute must be between 0 and 59 and must
be after starting date")
                tries = tries+1
                buffer = False
            elif YR == YR2 and MO == MO2 and D == D2 and HR == HR2:
                io.errmsg("Enter a valid minute please! Minutes must be after the number of
minutes of the starting date")
                tries = tries+1
                buffer = False
            else:
                buffer = True
                return date
        elif SEC > 59 or SEC < 0 or SEC < SEC2:
            if SEC > 59 or SEC < 0:
                io.errmsg("Enter a valid second please! Second must be between 0 and 59 and must
be after starting date")
                tries = tries+1
                buffer = False
            elif YR == YR2 and MO == MO2 and D == D2 and HR == HR2 and MIN == MIN2:
                io.errmsg("Enter a valid number of seconds please! Seconds must be after the
number of seconds of the starting date")
                tries = tries+1
                buffer = False
            else:
                buffer = True
                return date
        else:
            buffer = True
            return date
```

LinkCalculationsModule.py

```
import numpy as np
import math

"""
The function calculates the link budget of the satellite

Arguments:
1. F_cnt - Frequency band center (MHz)
2. AE    - Antenna Efficieny
3. D     - Diameter (m)
4. B     - Bandwidth (MHz)
5. RG    - RCV gain (dB)
6. RNT   - Noise temperature (K)
7. Pt    - Power Transmitted
8. Gt    - Gain of transmitting antenna (Onboard satellite)
9. Ltranspath - RF Losses in trasmitter path
10.Latm - Atmospheric and polarization losses
11.rangeTopo - Topocentric range

Returns:
- Power received by satellite

Identification:
   author: Feras Yahya
"""
def LinkDesign(F_cnt, AE, D, B, RG, RNT, Pt, Gt,Ltranspath, Latm, rangeTopo):
   #Calculate power in decibels
   Ptdb = 10*np.log10(Pt)

   #Calculate antenna gain
   FGHz = F_cnt/1000
   Gr = 10*np.log10(110*AE*(FGHz)**2*(D)**2)

   #EIRP
   EIRP = Ptdb + Gt - Ltranspath

   #Find wavelength
   c = 299792458
   lam = c/(F_cnt*1000000)     #In meters


   #Find free space loss
   Lf = ((4*math.pi*(rangeTopo*1000))/lam)**2
   LfdBw = 10*np.log10(Lf)

   #Power received
```

```
    Pr = EIRP - LfdBw + Gr - Latm   #In dBW
    PrdBm = Pr + 30

    #Noise power density
    N0 = -228.6 + 10*np.log10(RNT)

    #Carrier to Noise Density ratio
    C_N0 = Pr - N0

    #Bandwidth
    B = 10*np.log10(B*1000000)

    #Carrier to noise temperature
    C_T = C_N0 - 228.6

    #Carrier to Noise ratio
    C_N = C_N0 - B


    return PrdBm
```

TrackingDataModule.py

```
import Fileio as stk
import dateAndTimeCalculations as dt
import datetime as datetime
from OMPython import ModelicaSystem
import LinkCalculation as LC
import math

def ComputePositionAndVelocity(SatName, Px, Py, Pz, Vx, Vy, Vz, startTime, stopTime,
TimeStep,F_cnt, AE, D, B, RG, RNT, Pt, Gt,Ltranspath, Latm):

  #Extract the year, month, date, hour, minute and second from the start time
  (YR, MO, D, HR, MIN, SEC) = dt.formatDate(startTime)

  #Compute the interval between the two dates to use for the OM simulation
  interval = dt.TimeBetweenTwoDates(startTime, stopTime)
  f = open('gps-ops.txt', 'r')
  num_lines = sum(1 for line in open('gps-ops.txt'))

  #To track a certain satellite, choose the required TLE file, then, input the satellite name in the
while loop below.
  #After choosing the satellite to track, choose the tracking time and then set the tracking
duration

  line0=''
  while SatName not in line0:
```

```
        line0=f.readline()
        line1=f.readline()
        line2=f.readline()

    f.close()

    sat = stk.ReadNoradTLE(line0,line1, line2)

    RefEpoch = sat.refepoch;
    MICS = 0
    epsec = dt.timeSinceEpoch(RefEpoch,YR,MO,D,HR,MIN,SEC,MICS);    #time since epoch in
seconds
    timeSinceJ2000 = dt.TimeSinceJ2000(YR,MO,D,HR,MIN,SEC,MICS);    #time since J2000 in days
    JulDay = float(dt.gregToJulian(YR,MO,D,HR,MIN,SEC));          #Julian date of current time in
days

    #Tracking hour for GMST
    hr0 = (HR + MIN/60 + SEC/3600);
    #extracting orbital elements from TLE
    eccn = float("0." + sat.eccn);
    incl = float(sat.incl);
    raan = float(sat.raan);
    argper = float(sat.argper);
    meanan = float(sat.meanan);
    meanmo = float(sat.meanmo);
    ndot = float(sat.ndot);
    ndot6 = 0;
    bstar = 0;
    orbitnum = float(sat.orbitnum);
    #Reading OM files from python

mod=ModelicaSystem("SatellitePackage.mo","SatellitePackage.test","Modelica.SIunits.Conversi
ons")

    #Obtain parameters from OM
    print(mod.getParameters())

    #Set the parameters in OM

mod.setParameters(**{"GPS_Test.ecc":eccn,"GPS_Test.M0":meanan,"GPS_Test.N0":meanmo,
                    "GPS_Test.Ndot2":ndot,"GPS_Test.Nddot6":ndot6,"GPS_Test.tstart":epsec,
                    "ARO.elevation":2604.2,"ARO.longitude":281.927,"ARO.latitude":45.9555,
                    "JulianDate":timeSinceJ2000, "hr0":hr0, "raan":raan, "inc":incl, "argper":argper})

    #Print new parameters
    print(mod.getParameters())
```

```
    #Simulate
    mod.setSimulationOptions(startTime=0, stopTime=interval, stepSize=TimeStep)
    mod.simulate()

    #Used for the STK out file
    (time,posx,posy,posz,velx,vely,velz) = mod.getSolutions("time",Px,Py,Pz,Vx,Vy,Vz)

    #Used for making the tracking file
    (posx1,posy1,posz1,velx1,vely1,velz1) =
mod.getSolutions('PositionTOPO.x','PositionTOPO.y','PositionTOPO.z','VelocityTOPO.x','Velocity
TOPO.y','VelocityTOPO.z')

    (Azimuth,Elevation) = mod.getSolutions("Azimuth","Elevation")

    Azimuth = Azimuth*(180/math.pi)
    Elevation = Elevation*(180/math.pi)

    #Used for ephemeris file
    time = time+epsec
    Position = [posx, posy, posz]
    Velocity = [velx,vely,velz]

    doy = dt.doy(YR,MO,D)
    simTime = datetime.datetime.utcnow()
    simTime = simTime.replace(year = YR, month=MO,
day=D,hour=HR,minute=MIN,second=SEC,microsecond=0)

    f = open('Tracking.txt', 'w')
    f.write("--------------------------------------------------------------------------------------------\n")
    f.write("# UTC\t\t\tAz\tEl\tAz-vel   El-vel   Range   Range Rate\tDoppler\tLevel\n")
    f.write("# UTC\t\t\tDeg\tdeg\tdeg/sec  deg/sec   km\t   km/sec\tkHz\tdBm\n")
    f.write("--------------------------------------------------------------------------------------------\n")

    for i in range (0,len(Azimuth)):
       YR = simTime.year
       HR = simTime.hour
       MIN = simTime.minute
       SEC = simTime.second
       D = simTime.day
       MO = simTime.month
       angle = (Azimuth[i])
       angleEl = (Elevation[i])
       Range = (posx1[i]**2 + posy1[i]**2 + posz1[i]**2)**(1/2)
       RangeRate = (velx1[i]**2 + vely1[i]**2 + velz1[i]**2)**(1/2)
       minPower = LC.LinkDesign(F_cnt, AE, D, B, RG, RNT, Pt, Gt,Ltranspath, Latm, Range) - 30
#dBm
       doppler = (RangeRate/299792.458)*(1575.42*1000)
```

```
      f.write(("{0:4d}.{1:03d}-{2:02d}:{3:02d}:{4:02d}\t{5:.2f}\t{6:.2f}\t{7:1.1f}\t {8:1.1f}\t{9:.2f}
{10:.2f}\t\t{11:.2f}\t{12:.2f}\n").format(YR,doy, HR, MIN, SEC, angle, angleEl,0,0,
Range,RangeRate,doppler,minPower))
      #f.write(("{0:4d}.{1:03d}-{2:02d}:{3:02d}:{4:02d}\t{5:.2f}  {6:.2f}  {7:.2f}  {8:.2f}  {9:.2f}
{10:.2f}\n").format(YR,doy, HR, MIN, SEC,posx[i],posy[i],posz[i],velx[i],vely[i],velz[i]))
      simTime = simTime + datetime.timedelta(seconds=TimeStep)
      doy = dt.doy(YR,MO,D)

   f.close()

   return RefEpoch, time, Position, Velocity, Azimuth, Elevation
```

UserInputParser.py

```
import dateAndTimeCalculations as dt
import VisibilityModule as stk
import PointingFile as point
import TrackingData as posvel
import Fileio as io
import SensorPointingFile as spf


io.banner()

print(io.ReadStationFile())

startTime = input("\nInput the start time of tracking (Format: YYYY-MM-DD HR:MM:SS)\n")
startTime = str(startTime)
startTime = dt.validate(startTime)

stopTime = input("Input the stop time of tracking (Format: YYYY-MM-DD HR:MM:SS)\n")
stopTime = str(stopTime)
stopTime = dt.validateStopTime(stopTime,startTime)

timeStep = -1
while timeStep < 0:
   timeStep = input("Input time step:")
   timeStep = float(timeStep)

print("\nEnter the following RF characteristics of the GPS signal\n")
F_cnt = input("Enter the Frequency band center (MHz): ")
F_cnt = float(F_cnt)
AE = input("\nEnter the Antenna efficiency: ")
AE = float(AE)
Diameter = input("\nEnter the Antenna diameter (m): ")
Diameter = float(Diameter)
Bandwidth = input("\nEnter the bandwidth (MHz): ")
Bandwidth = float(Bandwidth)
```

```
RCV = input("\nEnter the receiving antenna gain RCV (dB): ")
RCV = float(RCV)
RNT = input("\nEnter the receiving antenna noise temperature RNT (deg K): ")
RNT = float(RNT)
Pt = input("\nEnter the power transmitted (W): ")
Pt = float(Pt)
Gt = input("\nEnter the gain of the transmitting antenna (dBi): ")
Gt = float(Gt)
Ltranspath = input("\nEnter the RF Losses in trasmitter path (dB): ")
Ltranspath = float(Ltranspath)
Latm = input("\nEnter any Atmospheric and polarization losses (dB): ")
Latm = float(Latm)

#generate AOSLOS file
stk.AOSLOS('AOSLOS', startTime,stopTime, timeStep, F_cnt, AE, Diameter, Bandwidth, RCV, RNT,
Pt, Gt,Ltranspath,Latm)

#Prompt user to enter the satellite to track
satToTrack = input("Choose your satellite to Track from the list above(Format: PRN XX): ")

satToTrack = str(satToTrack)

ephemCoord = input("Choose your coordinate system for ephemeris file\n1) Perifocal\n2)
ECI\n3) ECF\n4)Topocentric\n ")
invalidInput = True
#Calculate the position and velocity to be used for ephem file

while invalidInput != False:
   if ephemCoord == 'Perifocal':
      (RefEpoch, time, Position, Velocity,Azimuth, Elevation) =
posvel.ComputePositionAndVelocity(satToTrack,
'GPS_Test.p_sat_pf.x','GPS_Test.p_sat_pf.y','GPS_Test.p_sat_pf.z','GPS_Test.v_sat_pf.x','GPS_Te
st.v_sat_pf.y','GPS_Test.v_sat_pf.z',startTime,stopTime,timeStep,F_cnt, AE, Diameter,
Bandwidth, RCV, RNT, Pt, Gt,Ltranspath, Latm)
      #generate ephem file
      io.STKout('ephemTOPOnew.e', 'empty', dt.FormatEpoch(RefEpoch),time,"Custom
Perifocal_2 CentralBody/Earth",Position, Velocity)
      invalidInput = False
   elif ephemCoord == 'ECI':
      (RefEpoch, time, Position, Velocity,Azimuth, Elevation) =
posvel.ComputePositionAndVelocity(satToTrack,
'Position.x','Position.y','Position.z','Velocity.x','Velocity.y','Velocity.z',startTime,stopTime,timeSte
p,F_cnt, AE, Diameter, Bandwidth, RCV, RNT, Pt, Gt,Ltranspath, Latm)
      io.STKout('ephemTOPOnew.e', 'empty', dt.FormatEpoch(RefEpoch),time,"J2000",Position,
Velocity)
      invalidInput = False
   elif ephemCoord == 'ECF':
```

```
        (RefEpoch, time, Position, Velocity,Azimuth, Elevation) =
posvel.ComputePositionAndVelocity(satToTrack,
'PositionECF.x','PositionECF.y','PositionECF.z','VelocityECF.x','VelocityECF.y','VelocityECF.z',startTime,stopTime,timeStep,F_cnt, AE, Diameter, Bandwidth, RCV, RNT, Pt, Gt,Ltranspath, Latm)
        #generate ephem file
        io.STKout('ephemTOPOnew.e', 'empty', dt.FormatEpoch(RefEpoch),time,"Fixed",Position, Velocity)
        invalidInput = False
    elif ephemCoord == 'Topocentric':
        (RefEpoch, time, Position, Velocity,Azimuth, Elevation) =
posvel.ComputePositionAndVelocity(satToTrack,
'PositionTOPO.x','PositionTOPO.y','PositionTOPO.z','VelocityTOPO.x','VelocityTOPO.y','VelocityTOPO.z',startTime,stopTime,timeStep,F_cnt, AE, Diameter, Bandwidth, RCV, RNT, Pt, Gt,Ltranspath, Latm)
        #generate ephem file
        io.STKout('ephemTOPOnew.e', 'empty', dt.FormatEpoch(RefEpoch),time,"Custom TOPOCENTRIC_1 Facility/Algonquin",Position, Velocity)
        invalidInput = False
    else:
        ephemCoord = input("Invalid coordinate system\n1) Perifocal\n2) ECI\n3) ECF\n4)Topocentric\n ")
        invalidInput = True


#generate pointing file
point.PointingFile('point', satToTrack, startTime,stopTime,timeStep,Azimuth, Elevation)

spf.SensorPointingFile('SensorPointing',time,Azimuth, Elevation)
```

Preliminary testing

Since few changes were made along with the addition of couple new functions, it is important that the software is tested again using the testing plan provided in P3. At first, each added function will be tested individually to ensure it works as intended. Starting with the validation process for the inputs. Running the UserInputParser module:

```
Group 2
 Rajika Pati Arambage
 Feras Yahya
 Mark Lopez
 March 6 2018
 Welcome!

Input the station file name:
STNFIL.txt
station_info(name='ARO\n', stnlat=45.95550333333333, stnlong=281.9269597222222, stnalt=260.42, utc_offset=-4.0, az_el_nlim=1.0, az_el_lim=[angle_limits(az='0.0', elmin=' 9.0', elmax=' 89.0\n')], st_az_speed_max=3.0, st_el_speed_max=3.0)

Input the start time of tracking (Format: YYYY-MM-DD HR:MM:SS)
2018-45-11 10:00:00
Enter a valid month please! Month must be between 1 and 12
TRY AGAIN (Format: YYYY-MM-DD HR:MM:SS)
```

As seen above, upon entering an invalid month (or year, or day or HMS), the software prints out an error and prompts the user to input the starting date again:

```
Input the start time of tracking (Format: YYYY-MM-DD HR:MM:SS)
2018-45-11 10:00:00
Enter a valid month please! Month must be between 1 and 12
TRY AGAIN (Format: YYYY-MM-DD HR:MM:SS)
2018-03-10 10:00:00
Input the stop time of tracking (Format: YYYY-MM-DD HR:MM:SS)
2018-03-10 09:00:00
Enter a valid hour please! Hour must be after the hour of the starting date
TRY AGAIN (Format: YYYY-MM-DD HR:MM:SS)
```

As seen above, if the stopping date entered is before the starting date, the software prints out an error and prompts the user to input the stopping date again:

```
Enter a valid hour please! Hour must be after the hour of the starting date
TRY AGAIN (Format: YYYY-MM-DD HR:MM:SS)
2018-03-10 10:30:00
Input time step:-4
Input time step:-50
Input time step:60
```

Enter the following RF characteristics of the GPS signal

Enter the Frequency band center (MHz):

Entering a valid stopping date will allow the software to move forward and prompt the user for the time step. As seen, entering a negative time step will simply re-prompt the user to input the time step again until a valid number is inputted. After the time step, the software prompts the user to input the link variables. For this test, the losses and power transmitted inputted are arbitraty:

Enter the following RF characteristics of the GPS signal

Enter the Frequency band center (MHz): 1575.42

Enter the Antenna efficiency: 0.5

Enter the Antenna diameter (m): 46

Enter the bandwidth (MHz): 2

Enter the receiving antenna gain RCV (dB): 56

Enter the receiving antenna noise temperature RNT (deg K): 200

Enter the power transmitted (W): 25

Enter the gain of the transmitting antenna (dBi): 5

Enter the RF Losses in trasmitter path (dB): 1.25

Enter any Atmospheric and polarization losses (dB): 0.5

Once the user inputs all the required link variables, the software begins reading the TLE file and prints out all the satellites along with their AOS/LOS times. After which it prompts the user to choose a specific satellite from the list of satellites:

2018-05-10 17:49:53,737 - OMPython - INFO - OMC Server is up and running at file:///C:/Users/Owner/AppData/Local/Temp/openmodelica.port.58b77f3e63d24d8780b40ca401c23752
PRN 13
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 267.4987, 'GPS_Test.N0': 2.00565266, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': 2.9e-07, 'GPS_Test.ecc': 0.0034018, 'GPS_Test.tstart': 138543.43392, 'JulianDate': 6642.916666666511, 'argper': 92.8576, 'hr0': 10.0, 'inc': 55.5039, 'raan': 216.7057}
AOS: 2018-03-10 09:59:59
LOS: none
PRN 11
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 309.9191, 'GPS_Test.N0': 2.00569625, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -3.8e-07,

'GPS_Test.ecc': 0.0164815, 'GPS_Test.tstart': 80388.87264, 'JulianDate': 6642.916666666511, 'argper': 99.9411, 'hr0': 10.0, 'inc': 51.8208, 'raan': 66.0352}
AOS: none
LOS: none
PRN 20
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 182.878, 'GPS_Test.N0': 2.00549093, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -4.3e-07, 'GPS_Test.ecc': 0.0041652, 'GPS_Test.tstart': 107164.96704, 'JulianDate': 6642.916666666511, 'argper': 104.0841, 'hr0': 10.0, 'inc': 53.1718, 'raan': 143.8336}
AOS: 2018-03-10 09:59:59
LOS: none
PRN 28
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 2.3795, 'GPS_Test.N0': 2.00563996, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -8.9e-07, 'GPS_Test.ecc': 0.019909, 'GPS_Test.tstart': 100041.31296, 'JulianDate': 6642.916666666511, 'argper': 272.6476, 'hr0': 10.0, 'inc': 56.5284, 'raan': 334.8013}
AOS: none
LOS: none
PRN 14
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 108.0323, 'GPS_Test.N0': 2.00555321, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': 2.2e-07, 'GPS_Test.ecc': 0.0096493, 'GPS_Test.tstart': 78935.2704, 'JulianDate': 6642.916666666511, 'argper': 248.3032, 'hr0': 10.0, 'inc': 55.0765, 'raan': 214.3089}
AOS: none
LOS: none
PRN 16
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 268.0594, 'GPS_Test.N0': 2.00573204, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -8.9e-07, 'GPS_Test.ecc': 0.0098392, 'GPS_Test.tstart': 111770.614081, 'JulianDate': 6642.916666666511, 'argper': 26.298, 'hr0': 10.0, 'inc': 56.594, 'raan': 334.5603}
AOS: none
LOS: none
PRN 21
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 240.7437, 'GPS_Test.N0': 2.00554562, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -2.2e-07, 'GPS_Test.ecc': 0.0246775, 'GPS_Test.tstart': 81524.160001, 'JulianDate': 6642.916666666511, 'argper': 269.2453, 'hr0': 10.0, 'inc': 54.0286, 'raan': 87.2156}
AOS: 2018-03-10 09:59:59
LOS: none
PRN 22
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 90.7324, 'GPS_Test.N0': 2.00574924, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -4.1e-07, 'GPS_Test.ecc': 0.0075843, 'GPS_Test.tstart': 165094.084801, 'JulianDate': 6642.916666666511, 'argper': 268.4437, 'hr0': 10.0, 'inc': 52.9835, 'raan': 146.7414}
AOS: none
LOS: none
PRN 19

{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 168.7108, 'GPS_Test.N0': 2.00560575, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -6.2e-07, 'GPS_Test.ecc': 0.0095457, 'GPS_Test.tstart': 94134.4128, 'JulianDate': 6642.916666666511, 'argper': 64.2841, 'hr0': 10.0, 'inc': 56.1346, 'raan': 35.4388}
AOS: none
LOS: none
PRN 23
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 199.9677, 'GPS_Test.N0': 2.00573744, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': 1.5e-07, 'GPS_Test.ecc': 0.0122515, 'GPS_Test.tstart': 104186.491201, 'JulianDate': 6642.916666666511, 'argper': 223.9, 'hr0': 10.0, 'inc': 54.0728, 'raan': 209.1629}
AOS: none
LOS: none
PRN 02
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 266.6541, 'GPS_Test.N0': 2.00561144, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -2.2e-07, 'GPS_Test.ecc': 0.0180497, 'GPS_Test.tstart': 93044.0448, 'JulianDate': 6642.916666666511, 'argper': 252.2562, 'hr0': 10.0, 'inc': 54.4003, 'raan': 86.796}
AOS: none
LOS: none
PRN 17
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 354.0648, 'GPS_Test.N0': 2.00552519, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -6.5e-07, 'GPS_Test.ecc': 0.0125673, 'GPS_Test.tstart': 94160.40192, 'JulianDate': 6642.916666666511, 'argper': 257.8502, 'hr0': 10.0, 'inc': 56.2802, 'raan': 32.7367}
AOS: none
LOS: none
PRN 31
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 254.2321, 'GPS_Test.N0': 2.00569808, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': 2.5e-07, 'GPS_Test.ecc': 0.0088516, 'GPS_Test.tstart': 86081.163841, 'JulianDate': 6642.916666666511, 'argper': 350.2472, 'hr0': 10.0, 'inc': 55.2371, 'raan': 271.6643}
AOS: none
LOS: none
PRN 12
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 308.3599, 'GPS_Test.N0': 2.00568829, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -8.9e-07, 'GPS_Test.ecc': 0.0068566, 'GPS_Test.tstart': 249219.846721, 'JulianDate': 6642.916666666511, 'argper': 52.2947, 'hr0': 10.0, 'inc': 56.5697, 'raan': 333.5398}
AOS: none
LOS: none
PRN 15
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 323.1977, 'GPS_Test.N0': 2.00559734, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': 7e-08, 'GPS_Test.ecc': 0.0100561, 'GPS_Test.tstart': 92901.92544, 'JulianDate': 6642.916666666511, 'argper': 37.4662, 'hr0': 10.0, 'inc': 53.1971, 'raan': 205.2692}
AOS: 2018-03-10 09:59:59
LOS: none

PRN 29
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 159.8997, 'GPS_Test.N0': 2.0055791, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -6.4e-07, 'GPS_Test.ecc': 0.0004357, 'GPS_Test.tstart': 87038.208001, 'JulianDate': 6642.916666666511, 'argper': 23.7111, 'hr0': 10.0, 'inc': 56.3514, 'raan': 33.3346}
AOS: 2018-03-10 09:59:59
LOS: 2018-03-10 10:24:59
PRN 07
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 144.0794, 'GPS_Test.N0': 2.00562307, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': 2.3e-07, 'GPS_Test.ecc': 0.0113609, 'GPS_Test.tstart': 100359.230401, 'JulianDate': 6642.916666666511, 'argper': 215.1859, 'hr0': 10.0, 'inc': 54.9694, 'raan': 270.974}
AOS: none
LOS: none
PRN 05
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 18.2394, 'GPS_Test.N0': 2.00565278, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -4.2e-07, 'GPS_Test.ecc': 0.0051481, 'GPS_Test.tstart': 97853.578561, 'JulianDate': 6642.916666666511, 'argper': 33.3563, 'hr0': 10.0, 'inc': 54.2954, 'raan': 148.7643}
AOS: 2018-03-10 09:59:59
LOS: 2018-03-10 10:24:59
PRN 25
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 250.3477, 'GPS_Test.N0': 2.00563669, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -8.5e-07, 'GPS_Test.ecc': 0.0071297, 'GPS_Test.tstart': 123837.523201, 'JulianDate': 6642.916666666511, 'argper': 45.8403, 'hr0': 10.0, 'inc': 55.8638, 'raan': 330.1156}
AOS: none
LOS: none
PRN 01
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 301.7821, 'GPS_Test.N0': 2.00563879, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -1.8e-07, 'GPS_Test.ecc': 0.0072988, 'GPS_Test.tstart': 86528.56896, 'JulianDate': 6642.916666666511, 'argper': 33.4002, 'hr0': 10.0, 'inc': 55.601, 'raan': 90.1994}
AOS: none
LOS: none
PRN 24
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 331.496, 'GPS_Test.N0': 2.0056548, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': 1.4e-07, 'GPS_Test.ecc': 0.0067499, 'GPS_Test.tstart': 213850.34784, 'JulianDate': 6642.916666666511, 'argper': 28.8899, 'hr0': 10.0, 'inc': 53.9996, 'raan': 267.6836}
AOS: 2018-03-10 09:59:59
LOS: none
PRN 27
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 341.4863, 'GPS_Test.N0': 2.00563359, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -7e-07, 'GPS_Test.ecc': 0.0055628, 'GPS_Test.tstart': 97148.148481, 'JulianDate': 6642.916666666511, 'argper': 18.7093, 'hr0': 10.0, 'inc': 55.9804, 'raan': 29.667}
AOS: 2018-03-10 09:59:59

LOS: none
PRN 30
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 175.3763, 'GPS_Test.N0': 2.00552781, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': 9e-08, 'GPS_Test.ecc': 0.0031037, 'GPS_Test.tstart': 183589.188481, 'JulianDate': 6642.916666666511, 'argper': 184.6161, 'hr0': 10.0, 'inc': 54.1467, 'raan': 273.0043}
AOS: none
LOS: none
PRN 06
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 72.0011, 'GPS_Test.N0': 2.00574156, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -2.1e-07, 'GPS_Test.ecc': 0.0013872, 'GPS_Test.tstart': 115797.674881, 'JulianDate': 6642.916666666511, 'argper': 287.8973, 'hr0': 10.0, 'inc': 55.5854, 'raan': 89.7354}
AOS: none
LOS: none
PRN 09
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 254.7599, 'GPS_Test.N0': 2.00564031, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': 1.7e-07, 'GPS_Test.ecc': 0.0009056, 'GPS_Test.tstart': 107927.593921, 'JulianDate': 6642.916666666511, 'argper': 105.308, 'hr0': 10.0, 'inc': 54.5979, 'raan': 209.2151}
AOS: none
LOS: none
PRN 03
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 345.6683, 'GPS_Test.N0': 2.0057872, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -4.2e-07, 'GPS_Test.ecc': 0.0011342, 'GPS_Test.tstart': 76526.766721, 'JulianDate': 6642.916666666511, 'argper': 14.4046, 'hr0': 10.0, 'inc': 55.0459, 'raan': 149.9423}
AOS: none
LOS: none
PRN 26
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 1.8976, 'GPS_Test.N0': 2.00569857, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -8.7e-07, 'GPS_Test.ecc': 0.0027325, 'GPS_Test.tstart': 193547.211841, 'JulianDate': 6642.916666666511, 'argper': 358.1304, 'hr0': 10.0, 'inc': 54.8092, 'raan': 328.9727}
AOS: none
LOS: none
PRN 08
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 32.5985, 'GPS_Test.N0': 2.00552197, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -7.2e-07, 'GPS_Test.ecc': 0.0035149, 'GPS_Test.tstart': 93125.37312, 'JulianDate': 6642.916666666511, 'argper': 327.1766, 'hr0': 10.0, 'inc': 55.512, 'raan': 29.2166}
AOS: none
LOS: none
PRN 10
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 156.6712, 'GPS_Test.N0': 2.00564267, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -4.1e-07, 'GPS_Test.ecc': 0.0034321, 'GPS_Test.tstart': 89031.12768, 'JulianDate': 6642.916666666511, 'argper': 203.2143, 'hr0': 10.0, 'inc': 55.0521, 'raan': 149.7475}

AOS: 2018-03-10 09:59:59
LOS: none
PRN 32
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 146.4831, 'GPS_Test.N0': 2.0056706, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': 1.9e-07, 'GPS_Test.ecc': 0.0019047, 'GPS_Test.tstart': 124512.41952, 'JulianDate': 6642.916666666511, 'argper': 213.3646, 'hr0': 10.0, 'inc': 54.8504, 'raan': 209.4852}
AOS: none
LOS: none
Choose your satellite to Track from the list above(Format: PRN XX):

For this test, satellite PRN 05 was chosen. Once the user chooses their satellite to track, the software prompts the user to input the coordinate system for which the ephemeris file will be generated:

Choose your satellite to Track from the list above(Format: PRN XX): PRN 05
Choose your coordinate system for ephemeris file
1) Perifocal
2) ECI
3) ECF
4)Topocentric

Inputting an invalid coordinate system will re-prompt the user:

Choose your coordinate system for ephemeris file
1) Perifocal
2) ECI
3) ECF
4)Topocentric
random
Invalid coordinate system
1) Perifocal
2) ECI
3) ECF
4)Topocentric
Perifocal
2018-05-10 17:54:51,622 - OMPython - INFO - OMC Server is up and running at file:///C:/Users/Owner/AppData/Local/Temp/openmodelica.port.a97c983f6d74482f883611add9ca4414
Expected end of text (at char 3248), (line:16, col:37)
{'ARO.elevation': None, 'ARO.latitude': None, 'ARO.longitude': None, 'GPS_Test.M0': None, 'GPS_Test.N0': None, 'GPS_Test.Nddot6': None, 'GPS_Test.Ndot2': None, 'GPS_Test.ecc': None, 'GPS_Test.tstart': None, 'JulianDate': None, 'argper': None, 'hr0': None, 'inc': None, 'raan': None}
{'ARO.elevation': 2604.2, 'ARO.latitude': 45.9555, 'ARO.longitude': 281.927, 'GPS_Test.M0': 18.2394, 'GPS_Test.N0': 2.00565278, 'GPS_Test.Nddot6': 0.0, 'GPS_Test.Ndot2': -4.2e-07, 'GPS_Test.ecc': 0.0051481, 'GPS_Test.tstart': 97853.578561, 'JulianDate': 6642.916666666511, 'argper': 33.3563, 'hr0': 10.0, 'inc': 54.2954, 'raan': 148.7643}

Once a correct coordinate system is chosen, the software begins generating the ephemeris file, tracking file, sensor pointing file and the pointing file. Since there were minor adjustments made to the AOSLOS file and tracking file, they will be displayed.

AOSLOS file:

Sat No.	Name	AOS	LOS	Min. Expected Level(dBm)
1	PRN 13	2018-03-10 09:59:59	none	-81.10
2	PRN 11	none	none	none
3	PRN 20	2018-03-10 09:59:59	none	-80.68
4	PRN 28	none	none	none
5	PRN 14	none	none	none
6	PRN 16	none	none	none
7	PRN 21	2018-03-10 09:59:59	none	-81.14
8	PRN 22	none	none	none
9	PRN 19	none	none	none
10	PRN 23	none	none	none
11	PRN 02	none	none	none
12	PRN 17	none	none	none
13	PRN 31	none	none	none
14	PRN 12	none	none	none
15	PRN 15	2018-03-10 09:59:59	none	-80.58
16	PRN 29	2018-03-10 09:59:59	2018-03-10 10:24:59	-82.09
17	PRN 07	none	none	none
18	PRN 05	2018-03-10 09:59:59	2018-03-10 10:24:59	-82.17
19	PRN 25	none	none	none
20	PRN 01	none	none	none
21	PRN 24	2018-03-10 09:59:59	none	-82.09
22	PRN 27	2018-03-10 09:59:59	none	-82.34
23	PRN 30	none	none	none
24	PRN 06	none	none	none
25	PRN 09	none	none	none
26	PRN 03	none	none	none
27	PRN 26	none	none	none
28	PRN 08	none	none	none
29	PRN 10	2018-03-10 09:59:59	none	-82.45
30	PRN 32	none	none	none

Tracking file

# UTC # UTC	Az Deg	El deg	Az-vel deg/sec	El-vel deg/sec	Range km	Range Rate km/sec	Doppler kHz	Level dBm
2018.069-10:00:00	85.90	17.34	0.0	0.0	24019.79	3.05	16.05	-125.43
2018.069-10:01:00	86.17	17.00	0.0	0.0	24054.53	3.06	16.06	-125.44
2018.069-10:02:00	86.44	16.66	0.0	0.0	24089.32	3.06	16.08	-125.45
2018.069-10:03:00	86.71	16.32	0.0	0.0	24124.16	3.06	16.09	-125.46
2018.069-10:04:00	86.98	15.98	0.0	0.0	24159.05	3.06	16.11	-125.48
2018.069-10:05:00	87.25	15.64	0.0	0.0	24194.00	3.07	16.12	-125.49
2018.069-10:06:00	87.52	15.30	0.0	0.0	24228.99	3.07	16.14	-125.50
2018.069-10:07:00	87.79	14.96	0.0	0.0	24264.02	3.07	16.15	-125.51
2018.069-10:08:00	88.06	14.62	0.0	0.0	24299.11	3.08	16.16	-125.53
2018.069-10:09:00	88.33	14.28	0.0	0.0	24334.24	3.08	16.18	-125.54
2018.069-10:10:00	88.60	13.94	0.0	0.0	24369.42	3.08	16.19	-125.55
2018.069-10:11:00	88.87	13.61	0.0	0.0	24404.63	3.08	16.20	-125.56
2018.069-10:12:00	89.14	13.27	0.0	0.0	24439.90	3.09	16.22	-125.58
2018.069-10:13:00	89.41	12.93	0.0	0.0	24475.20	3.09	16.23	-125.59
2018.069-10:14:00	89.68	12.60	0.0	0.0	24510.55	3.09	16.24	-125.60
2018.069-10:15:00	89.95	12.26	0.0	0.0	24545.94	3.09	16.26	-125.61
2018.069-10:16:00	90.21	11.93	0.0	0.0	24581.36	3.10	16.27	-125.63
2018.069-10:17:00	90.48	11.59	0.0	0.0	24616.83	3.10	16.28	-125.64
2018.069-10:18:00	90.75	11.26	0.0	0.0	24652.33	3.10	16.29	-125.65
2018.069-10:19:00	91.02	10.93	0.0	0.0	24687.87	3.10	16.30	-125.66
2018.069-10:20:00	91.28	10.59	0.0	0.0	24723.45	3.10	16.31	-125.68
2018.069-10:21:00	91.55	10.26	0.0	0.0	24759.06	3.11	16.32	-125.69
2018.069-10:22:00	91.82	9.93	0.0	0.0	24794.71	3.11	16.34	-125.70
2018.069-10:23:00	92.08	9.60	0.0	0.0	24830.39	3.11	16.35	-125.71
2018.069-10:24:00	92.35	9.27	0.0	0.0	24866.11	3.11	16.36	-125.73
2018.069-10:25:00	92.62	8.94	0.0	0.0	24901.85	3.11	16.37	-125.74
2018.069-10:26:00	92.88	8.61	0.0	0.0	24937.63	3.12	16.38	-125.75
2018.069-10:27:00	93.15	8.28	0.0	0.0	24973.44	3.12	16.39	-125.76
2018.069-10:28:00	93.41	7.95	0.0	0.0	25009.28	3.12	16.40	-125.78
2018.069-10:29:00	93.68	7.62	0.0	0.0	25045.15	3.12	16.40	-125.79
2018.069-10:30:00	93.94	7.29	0.0	0.0	25081.04	3.12	16.41	-125.80
2018.069-10:31:00	93.94	7.29	0.0	0.0	25081.04	3.12	16.41	-125.80

When comparing the values to the ones obtained from P3, the results are similar except for the Min. Expected Level(dBm) in AOSLOS and Level dBm in Tracking, which should be more accurate than the ones obtained in P3.

Printing the ephemeris file, we see that the software generated an ephemeris file for the coordinate system chosen by the user:

stk.v.4.3

BEGIN Ephemeris

NumberOfEphemerisPoints   32

ScenarioEpoch   9 Mar 2018 06:49:06.421439
InterpolationMethod   Lagrange
InterpolationOrder   7
CentralBody   Earth
CoordinateSystem   Custom Perifocal_2 CentralBody/Earth

EphemerisTimePosVel

9.78535785610000E+04 -1.18860885627090E+07 2.38197789350745E+07 0.00000000000000E+00 -3.46638850404423E+03 -1.70978690982399E+03 0.00000000000000E+00
.
.
.
9.96535785610000E+04 -1.76506954607578E+07 1.99673441511334E+07 0.00000000000000E+00 -2.90252471460698E+03 -2.54582465724139E+03 0.00000000000000E+00

END Ephemeris

Team contribution

Lab	Feras Yahya	Mark Lopez	Rajika Pati Arambage
P1	Entire dateAndTimeCalculations.py module	Satellite model and vector record	Entire Fileio.py module
P2	Added STKout() function to Fileio.py module	Added GndStn, Sat_ECI, theta_t and Sat_ECF to the OpenModelica model	Added station_ECF, range_ECF2Topo and range_topo2lookup_angles to the OpenModelica model
P3	Developed the comprehensive testing plan. Made few adjustments to all the OpenModelica models and added few functions to the dateAndTimeTalculations.py module. Made few adjustments to the Fileio.py module. Performed STK testing	Developed the visibility module.	Developed the pointing file module.
P4	Added the link calculations module, the sensor pointing file module, the tracking data module, link calculations module and developed the userInputParser.py Module. Added validation functions to the dateAndTimeCalculations.py module which are used to validate user inputs. Performed STK testing	Edited and proof-read the report. Made few adjustments to the AOS/LOS module	Edited and proof-read the report. Made few adjustments to the pointing file module
P5	Made few adjustments to the userInputParser.py module and added few more functions to the dateAndTimeCalculations.py modules. Performed STK testing	Edited and proof-read the report. Performed the final testing to conclude whether the software is complete or not	Edited and proof-read the report. Wrote the conclusion

Conclusion

In conclusion, the software is overall complete. The software included accepting the user inputs, validating them, performing any necessary calculations and producing the following files: Ephemeris file, AOS/LOS file, Tracking data file, sensor pointing file and pointing file. When compared to STK, the ephemeris file proved to be valid and accurate since the satellite it generated was very similar to the actual satellite imported from the same TLE file. In terms of the access times, they were relatively accurate since they matched the times provided by STK. However, it is important to note that the visibility module only provides the AOS and LOS times for the first access. The azimuth and elevation angles used in the sensor and pointing files were also relatively accurate since they matched the value produced by STK. Our biggest uncertainty is the link calculations module. This is because we were unable to compare our values to a verified source. However, using the web, we found typical values for the power received from GPS satellites and compared them to our case. We also used losses and power transmitted values found from the web to compute our received power. Although we are unsure about the validity of the link calculations module, we are certain that the values produced contain some level of accuracy within them.