АО «Российские космические системы»

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ МАКЕТА КОМПЛЕКСА ПЛАНИРОВАНИЯ ВЫПОЛНЕНИЯ РАДИОЛОКАЦИОННОЙ СЪЁМКИ

Авторы: Дудин Даниил Евгеньевич
Мирославлев Илья Антонович
Наумов Роман Евгеньевич
Филиппова Арина Евгеньевна
Яцук Анастасия Александровна

Руководители: Ширшова Вера Юрьевна, к.г.н.
Шувайникова Татьяна Петровна
Мекшун Игорь Андреевич

Сириус – 2025

ОГЛАВЛЕНИЕ

1. Расчёт зон визирования

1.1. Функция «get_spacetrack_tle»

Функция «get_spacetrack_tle» создана для получения с сайта «Space-track.org» файлов TLE.

На вход функции подаётся номер космического аппарата в NORAD, дата начала и окончания наблюдения за аппаратом, а также логин и пароль на сайте «Space-track.org».

Сначала осуществляется вход в аккаунт пользователя на сайте «Space-track.org» и запрашивается TLE для заданного спутника. Далее полученные данные записываются в новый файл в формате txt, который представляет собой 2 строчки информации.

1.2. Функции «create_orbital_track_shapefile_for_day» и «create_orbital_track_shapefile_north»

Функции «create_orbital_track_shapefile_for_day» и «create_orbital_track_shapefile_north» созданы для создания файлов в формате shp с геоданными о точках надира. Функции практически одинаковые. Их различие будет описано позже.

На вход функции получают дату наблюдения за космическим аппаратом, время начала и окончания наблюдения, шаг между точками надира в секундах, названия выходного файла в формате shp и входного файла в формате txt. Входной файл в формате txt – это файл, получаемый в конце функции «get_spacetrack_tle».

Первый шаг – считывание информации из входного файла. Каждая строчка файла записывается в новую переменную. Затем полученные переменные записываются в переменную класса Orbital.

Далее производится создание нового файла в формате shp: задаются названия столбцов и типы данных в каждом из них. В данных функциях создаются столбцы: ‘ID’ (целочисленный тип), ‘TIME’ (строка), ‘LAT’ (вещественное число), ‘LON’ (вещественное число).

Затем создаётся переменная start_seconds, в которой хранится время начала наблюдения, переведённое в секунды, и счётчик i с начальным значением 0.

Далее запускается цикл с параметром seconds, изменяющимся в пределах от start_seconds до времени окончания наблюдения в секундах с шагом, поданным на вход функции. В каждой итерации цикл сохраняет рассматриваемое время в отдельную строку в формате «YYYY-MM-DD HH:MM:SS», чтобы в будущем было возможным реализовать визуализацию в QGIS. Затем с помощью функции «get_lonlatalt» рассчитывается долгота и широта точки надира. Функция «point» позволяет с помощью полученных данных создать в shp-файле новый объект с геометрией точки и записать его долготу, широту и порядковый номер в файле, определяемый счётчиком i.

После выхода из цикла полученные данные функция сохраняет в файл формата shp, а данные о системе координат записывает в файл формата prj и выводит 'End of code!'. Если у пользователя нет прав на запись информации в файл, то программа выведет 'Unable to save shapefile'.

Различие данных функций состоит в сохранении точек. «create_orbital_track_shapefile_north» получает на вход максимальный и минимальный угол, ограничивающие зону интересов, точки внутри которой сохраняются, а остальные не рассматриваются.

1.3. Функции «polygon_to_coords», «point_to_coords», «poly_to_coords»

Функции «polygon_to_coords», «point_to_coords», «poly_to_coords» - функции считывание координат из геодатасета GeoData. Функция «polygon_to_coords» определяет координаты группы полигонов, «point_to_coords» - координаты группы точек, «poly_to_coords» - координаты отдельного одного полигона.

1.4. Функция «getWKT_PRJ»

Функция «getWKT_PRJ» получает данные о проекции Земли по EPSG. Она обращается к сайту «Spatialreference.org» для получения информации. Затем переводит эти данные в строку и возвращает их.

1.5. Функция получения точек съёмки по заданному углу визирования. Функция «ugol_visir»

Функция «ugol_visir» создана для расчёт координат точек съёмки по заданному углу визирования. На вход функции подаются названия выходных файлов формата shp, угол визирования и название входного файла формата shp, содержащего координаты точек надира. Такие файла создаются с помощью функций «create_orbital_track_shapefile_for_day» и «create_orbital_track_shapefile_north».

Для работы функции используются библиотеки GeoPandas, NumPy, Matplotlib и Math.

Первым шагом рассчитывается угол покрытия и перпендикулярное расстояние от точек надира до границ зон визирования. Расчёт осуществляется с помощью уравнений (3), (4).

$$\gamma = \arcsin\frac{(h+R)*\sin\alpha}{R} - \alpha, \quad (3)$$

где γ – угол покрытия;

h – высота орбиты спутника над поверхностью Земли, м;

R – радиус Земли;

α – угол визирования.

$$L = \frac{\pi R \gamma}{180}, \quad (4)$$

где L – перпендикулярное расстояние от точек надира до границы полосы визирования.

Функция переводит полученные координаты точек надира в EPSG 3043 и сохраняет полученные данные в отдельную переменную coords_nadir. С помощью цикла с параметром производится проверка корректности координат. Далее создаются два файла формата shp, в которые позже будут записываться координаты точек границ полос обзора.

С помощью цикла с параметром осуществляется расчёт и запись точек границ зон визирования. Программа учитывает положение аппарата в разных частях витка: восхождение, нисхождение и пики. Структура цикла представляет из себя большое количество условий, которые позволяют определить положение космического аппарата в каждой точке. Во всех условиях используются две точки: текущая и следующая. Сначала рассматриваются пики – случаи, в которых широта двух соседних точек одинакова. Вложенное условие определяет к восхождению или нисхождению относится рассматриваемый пик. Затем функция рассчитывает координаты точек границ, вычитая или добавляя L к широте.

Второе условие учитывает случай, при котором долгота двух точек одинакова. Алгоритм его работы такой же, как и у первого условия, но в нём L прибавляется к долготе или вычитается из неё.

Далее рассматриваются случаи, в которых изменяются и долгота, и широта. Для такого расчёт необходимо рассчитать угол отклонения от горизонтали. Для его расчёта используется формула (5).

$$\beta = arctg \frac{\Delta lat}{\Delta lon}, \quad (5)$$

где β – угол отклонения от горизонтали;
Δlat – разница широт двух соседних точек;
Δlon – разница долгот двух соседних точек.

Далее рассчитывается координата точек границ зон визирования. К значению широты прибавляется или из неё вычитается $L * \cos\beta$, к долготе - $L * \sin\beta$.

После всех расчётов координаты сохраняются в файлы формата shp отдельные для левого и правого бортов. Затем с помощью функции «getWKT_PRJ» функция запрашивает информации о системе координат, которую может вводить сам пользователь, и сохраняет его в файл формата prj. Далее программа сохраняет все файлы или выводит 'Unable to save shapefile', если нет прав на запись.

1.6. Функция получения полигонов полос визирования для отдельного витка. Функция «add_vitok»

Функция «add_vitok» служит для объединения точек границ полос обзора в полигоны и их сохранения в файлы с форматом shp. На вход функция получает файл с координатами точек границ зон визирования, минимальный и максимальный углы визирования, названия выходных файлов формата shp. В функции импортируется библиотека Warnings.

Первым шагом в «add_vitok» используется функция «ugol_visir» для создания точек границ зон визирования на минимальном и максимальном углах визирования. Полученные файлы записываются в новые переменные в формате Geo Data Frame и перепроецируются в метрическую систему координат EPSG 4326. Далее в новые переменные min_l_lat, min_l_lon, min_r_lat, max_r_lon (для полос визирования с минимальным углом визирования), max_l_lat, max_l_lon, max_r_lat, max_r_lon (для полос визирования с максимальным углом визирования) записываются координаты точек границ зон визирования. Запись осуществляется с помощью 4-ёх циклов с параметром, которые идут друг за другом. Далее начинается само объединение в полигоны.

При создании полигонов возникает проблема, связанная с пересечением спутником 180-го меридиана. При пересечении данного меридиана координата долготы обнуляется, что затрудняет моделирование. Для решения этой проблемы вводится индикатор ind20 изначально равный 0.

Само моделирование происходит с помощью циклов, которые с алгоритмической точки зрения одинаковы, поэтому будут рассмотрены 2 цикла по моделированию.

Сначала с помощью цикла с параметром определяется номер точки, на которой спутник пересекает 180 меридиан. Затем задаётся условие, проверяющее пересекает ли аппарата на данном витке 180 меридиан. Если $ind20 = 0$, то начинается создание полигона зоны визирования для каждого борта. Рассмотрим на примере правого борта. Запускается цикл с параметром, перебирающий все точки из min_r_lon. Если долгота больше -150, то в отдельный список добавляются координаты рассматриваемой точки. Аналогично происходит для max_r_lon, но перебор точек начинается с последней. После циклов создаётся словарь с полигоном, который записывается в файл формата shp. Аналогично происходит для левого борта. В выход программа выдаёт два файла формата shp.

Рассмотрим случай, когда аппарат пересекает 180-й меридиан. Сначала создаются полигоны полос визирования, которые находятся до 10 меридиана. Задаётся счётчик k равный 0. Запускается цикл с предусловием, который работает пока $k \neq ind20 - 1$. Цикл перебирает точки с границы минимального угла визирования. В цикле проверяется, что долгота текущей точки меньше или равна 160. Если условие верно, то координаты точки добавляются в список. Потом к счётчику прибавляется 1. После выхода из

цикла из счётчика вычитается 2, и запускается новый цикл с предусловием, работающий пока $k \neq -1$, перебирающий точки с границы максимального угла визирования. В цикле проверяется то же условие, что и в первом цикле, из счётчика вычитается 1. После выхода из цикла создаётся словарь с полигоном полосы визирования, который сохраняется в файле формата shp. Аналогично создаётся файл с полосой визирования для другого борта.

Для создания остальных полигонов используются циклы с предусловием, но с другими счётчиками и условиями. Счётчик k принимает значение $ind20$. Первый цикл с предусловием работает пока k не равен длине списка с точками границ для минимального угла визирования -1. Далее проверяется, что долгота текущей точки больше или равна -160. Если условие выполняется, то координаты точки записываются в список. После выхода из цикла k присваивается значение длины списка с точками границ для минимального угла визирования -1. Второй цикл с предусловием работает пока $k \neq ind20$. Проверяется то же условие, что и в первом цикле, только для точек границ для максимального угла визирования. В конце цикла из счётчика вычитается 1. После циклов создаётся словарь с полигонами, сохраняемый в файл формата shp. Аналогично создаётся полигон для другого борта.

2. Составление предполагаемой рабочей программы с учётом баллистических ограничений. Функция «nearest_subsatellite_point»

Функция «nearest_subsatellite_point» создана для поиска ближайшей к центроиду кадра точки надира. На вход функция получает файл формата shp с данными о точках съёмки и название папки, в которой хранятся файлы с точками надира на разных витках.

Функция представляет из себя три цикла с параметром, которые вложены друг в друга. Первый цикл перебирает все номера витков, которые есть во входном файле, второй - точки съёмки, третий – точки надира.

Сначала запускается первый цикл, счётчик которого принимает значение номера текущего витка. Затем запускается второй цикл, в котором проверяется производится ли текущая съёмка на рассматриваемом витке. Если условие выполняется, то запускается третий цикл, перебирающий все точки текущего витка и находящий ближайшую из них. Время прохождения ближайшей точки надира записывается в отдельный список, который позже конвертируется в таблицу формата csv.

На выход функция выводит итоговый файл формата csv, содержащий время съёмки каждого кадра. Данный файл далее используется при учёте ограничений.

3. Временные ограничения

В ходе работы над проектом создана программа для вывода пересечений ЦУПа с точками надира и корректной программой съемки. Из исходной программы съемки удаляются те съемки, которые попадают под

ограничения ЦУПа и производятся в течение определённого числа секунд после окончания предыдущей съемки. Время между съёмками пользователь может вводить самостоятельно.

3.1. Создание списков ограничений ЦУПа: ТЦУ и ОПД. Функция «create_table_cup»

Функция «create_table_cup» написана для создания списков с ограничениями от ЦУПа: технологическим циклом управления и определением параметров движения. На вход функция получает название файла формата txt, содержащего информацию об ограничениях.

Функция состоит из цикла с параметром, который перебирает все временные параметры из входного файла. Значения времени записываются в отдельные списки tcu и opd. На выход функция выводит эти списки с ограничениями.

3.2. Определение вхождения точки в некоторый интервал времени. Функция «is_time_between»

Функция «is_time_between» написана для определения вхождения точки в определённый интервал времени. На вход функция получает время начала и конца промежутка и время прохождения интересующей точки. Все временные параметры вводятся в формате datetime.

Функция проверяет принадлежность временного промежутка к одним суткам. На выход функция выдаёт логическое значение True или False.

3.3. Создание файла формата shp с пересечением точек надира с ограничениями. Функция «cup_intersections»

Функция «cup_intersections» написана для создания файла формата shp с пересечением точек надира с ограничениями ЦУПа. На вход функция получает название файла с точками надира, полученного с помощью функции «create_orbital_track_shapefile_for_day», и название файла, в который будут сохранены пересечения.

Функция считывает из входного файла время прохождения точек надира и принимает ограничения от ЦУПа. Далее запускается цикл с параметром, который проверяет накладывается ли ограничение на конкретную точку. Для каждого ограничения создаёт отдельный файл, состоящий из тех точек надира, которые попадают под ограничения. На выход функция выводит 2 файла с точками, попавшими под ограничения.

3.4. Создание корректного плана съёмок. Функция «success_shoting»

Функция «success_shoting» написана для создания корректного плана съёмок. На вход функция получает минимальный временной промежуток между двумя ближайшими съёмками, путь к файлу формата csv с программой

съёмок космического аппарата, название файла формата xlsx, в который будет сохраняться таблица с корректной программой съёмки.

С помощью цикла с параметром перебираются все съёмки. Функция создаёт файл формата xlsx, в который записывает данные о съёмках. По считанным данным заполняются 3 столбца таблицы: IS_VALID, TM_DLT_VLD и REASON. В столбец IS_VALID записываются логические значения: TRUE – точки не пересекают ограничения, FALSE – пересекают. В столбец TM_DLT_VLD тоже заносятся логические значения: TRUE – между съёмками прошло установленное время, FALSE – не прошло. Столбец REASON заполняется, если в столбце IS_VALID записано FALSE. В данный столбец заносится информация об ограничении, под которое попала точка.

На выход функция выводит файл формата xlsx, содержащий корректную программу съёмок.

4. Ограничения по памяти

4.1. Определение временных границ последующей сессии связи. Функция «intersect_sessions»

Функция «intersect_sessions» создана для определения временных границ следующей сессии связи. На вход функция получает временные промежутки, в течение которых осуществляется первая и вторая сессии связи, в формате datetime, время переключения между пунктами приёма информации в секундах, скорость передачи информации в Мбит/с, вес одного снимка в Мбайт.

Сначала функция проверяет, что между двумя сессиями меньше установленного времени. Если время больше установленного, то возвращаем время второй сессии. Если оно меньше, то происходит вторая проверка. Условный оператор проверяет совпадают ли временные промежутки. Если промежутки входят друг в друга, то удаляется меньший из них. При равенстве промежутков удаляется второй.

Если промежутки пересекаются, то начало второй сессии сдвигается на установленное время после завершения первой сессии. Далее проверяется возможность передачи во время второй сессии хотя бы одного снимка. Если времени недостаточно, то время второй сессии удаляется.

На выход функция подаёт время начала и конца второй сессии.

4.2. Расчёт временных границ сессии связи. Функция «intersect_capture_with_sessioons»

Функция «intersect_capture_with_sessioons» создана для определения временных границ сессии связи. На вход функция получает временной промежуток, в который выполняется съёмка, временной промежуток, отведенный на сессию связи, время на выполнение съёмки, время переключения между ППИ или ППИ и съёмкой, скорость передачи

информации, вес одного снимка и положение сессии связи относительно съёмки.

Логическая составляющая данной функции совпадает с предыдущей. Отличие функций заключается только во входных данных и условиях.

На выход функция выводит временной промежуток сессии связи в формате datetime.

4.3. Вывод итоговой таблицы записи съёмок, сброса информации и удаления сброшенных снимков. Функция «calculate_memory_restricts»

Функция «calculate_memory_restricts» создана для создания таблицы, содержащей информацию о записи съёмок, сбросе информации и удалении сброшенных файлов. На вход функция получает названия выходного файла формата xlsx, файла с корректным расписанием съёмок, полученного с помощью функции «success_shooting», файла с сессиями связи, полученного с помощью функции «intersect_capture_with_sessioons», счётчик ячеек памяти, скорость передачи информации, время съёмки, время переключения между ППИ или ППИ и съёмкой, вес одного снимка.

Сначала удаляются съемки, пересекающиеся с ограничениями ЦУПа или находящиеся в определённом интервале времени после предыдущей съемки. После этого составляется план сброса информации, план сессий связи без пересечений, общий план съемок со сбросом информации. Производится проверка на пересечения съемок с сессиями связи и симуляция записи, удаления или сброса информации из ячеек памяти запоминающего устройства.

На выход функция выводит таблицу с планом съёмок и сброса информации формата xlsx. Таблица имеет три колонки: PROCESS, START и END. В колонке PROCESS хранится название выполняемой операции: WRITE CELL (запись съёмки в ячейку памяти), DROP CELL (сброс информации из ячейки памяти), VALID DELETE CELL (удаление сброшенной съёмки из ячейки памяти) и INVALID DELETE CELL (удаление несброшенной съёмки из ячейки памяти).

5. Расчёт энергопотребления. Функция «energy_consumption»

Функция «energy_consumption» создана для расчёта энергопотребления космического аппарата и выявления превышения энергопотребления. Для работы используются методы из библиотеки «Pandas».

На вход функции подаётся установленная мощность, мощность обеспечивающих систем, мощность радиолокатора, период обращения вокруг Земли, время выполнения одной съёмки и файл в формате csv с временем выполнения съёмок. Синтаксис функции Вы можете видеть на рисунке ниже.

```
def energy_consumption(P_ust, P_obes, N_radio, T_KA, t_sem, input_csv):
```

где P_ust – установленная мощность, Вт;
P_obes – мощность обеспечивающих систем, Вт;
N_radio – мощность радиолокатора, Вт;
T_KA – период обращения космического аппарата, с;
t_sem – время, затрачиваемое на выполнение одной съёмки, с;
input_csv – название входного файла.

Особое внимание нужно уделять данным из входного файла. Время должно быть записано в следующем формате:

YYYY-MM-DD HH:MM:SS

Формат даты может быть любым другим, но часы минуты и секунды обязательно нужно указывать через двоеточие. При этом в строке двоеточие должно присутствовать только во времени.

Перед началом работы функция конвертирует исходный файл в Data Frame с помощью библиотеки «Pandas».

Функция состоит из 3-ёх частей. В первой части производится обработка файла в формате csv с помощью цикла с параметром. В цикле программа перебирает все значения времени, которые есть в файле, и распределяет их по виткам. Процесс распределения описывается формулой (1):

$$N_{vit} = 1 + \left[\frac{T_{sec}}{T_{ka}}\right], \quad (1)$$

где N_{vit} – номер витка;
T_{sec} – время, переведённое в секунды, с;
T_{ka} – период обращения космического аппарата вокруг Земли, с.

Далее в том же цикле полученные значения записываются в отдельный массив. После цикла массив с номерами витков с помощью библиотеки «Pandas» конвертируется в новый столбец таблицы Data Frame.

Вторая часть представляет из себя цикл с параметром, в который вложены ещё 2 цикла с параметром и 1 цикл с предусловием. Основной цикл перебирает все номера витков, благодаря чему производится расчёт энергопотребления для конкретного витка. Первый вложенный цикл с предусловием рассчитывает количество съёмок на рассматриваемом витке. Циклы с параметром полностью идентичны. Они необходимы для расчёта энергопотребления отменяемых и неотменяемых операций. Если такие операции на данном витке не выполняются, то программа не запускает циклы. На вход циклам подаётся количества операций, участвующих установок, время их работы и энергопотребление.

В третьей части производится сравнение полученного энергопотребления на витке с установленной мощностью, которую нельзя превышать. Энергопотребление на витке рассчитывается по формуле (2):

$$P_v = P_{sem} + P_{obes} + P_o + P_{no}, \quad (2)$$

где P_v – энергопотребление на рассматриваемом витке, Вт;
P_{sem} – энергопотребление съёмки на витке, Вт;
P_{obes} – энергопотребление обеспечивающих систем, Вт;
P_o – энергопотребление отменяемых операций, Вт;
P_{no} – энергопотребление неотменяемых операций, Вт.

Полученная величина сравнивается с установленной мощностью, подаваемой на вход функции. Если P_v > P_ust, то программа выведет предупреждение о превышении энергопотребления. Если $P_v \leq$ P_ust, то программа выведет «ОК».