Спутниковые системы

2022/23 учебный год

Второй отборочный этап

Задача IV.1. (20 баллов)

Темы: элементы орбиты, энергобаланс, ориентация и стабилизация КА.

Условие

Задачей космического аппарата является сбор научных данных на полюсах Земли и передача их на Землю.

Напишите код управления спутником, который обеспечивает разворот солнечных батарей на Солнце, а также включает и выключает устройства аппарата, обеспечивая положительный энергобаланс и сбор данных. Ориентация на Солнце осуществляется с помощью маховика на оси Z. Балл за сбор данных начисляется при работающих полезной нагрузке и передатчике. Помимо кода управления укажите также элементы орбиты аппарата.

Максимально возможная высота: 10000 км от центра Земли.

Время начала симуляции: 1 ноября 2022 года 00:00 по UTC.

Длительность симуляции: 24 часа.

Если заряд аккумулятора опустится до нуля, то аппарат считается вышедшим из строя и потом не включается.

Программу управления спутником необходимо написать на языке Java Script, используя API. API управления аппаратом: https://disk.yandex.ru/i/MZivlCuogSD_Fw.

Потребление устройств аппарата приведено в таблице ниже.

Детектор радиации (полезная нагрузка)	10 Вт
Маховик	5 Вт
Передатчик	1 Вт
Датчики и прочее оборудование	1 Вт

Подсказка

При решении задачи ориентации можно использовать алгоритм ПИД-регулирования. Однако на его отладку может понадобиться несколько попыток. Чтобы проще было разобраться с принципом работы ПИД- регулирования, можно воспользоваться задачей из первого этапа: https://nti.orbitagame.ru/events/190/2798.

Коэффициенты ПИД-регулятора из тренировочной задачи ***не*** *подойдут для данной задачи, так как масса аппаратов различна. Однако в первом приближении подобранные коэффициенты можно уменьшить в 10–20 раз.*

Площадь солнечных батарей: 0,05 м^2.

Солнечная постоянная: 1368 Вт/м^2.

КПД СБ: 20%.

Емкость аккумулятора: 10000 Вт·с.

Начальный заряд аккумулятора: 70%.

Начисляется 0,004 балла за 1 секунду симуляции при соблюдении следующих условий:

- аппарат находится над зоной полюса;
- полезная нагрузка включена;
- передатчик включен;
- заряд аккумулятора больше 0;
- высота орбиты меньше 10000 км от центра Земли.

Максимально возможный балл за задачу — 20.

Решение

Задача состоит из двух частей: подбор орбиты и написание кода управления спутником.

Подбор орбиты

Так как необходимо произвести сбор данных на полюсах, то наклонение орбиты лучше всего брать равным 90°. Это обеспечит гарантированный проход спутника над полюсами.

В задаче имеется ограничение по высоте, однако при наибольшей высоте окно видимости полюсов будет длиться дольше. Поэтому самым оптимальным вариантом будет взять максимально допустимую высоту.

Третьим важным моментом, который необходимо учесть, является обеспечение максимально возможного энергобаланса аппарата. Для этого желательно подобрать долготу восходящего узла так, чтобы аппарат на текущее время моделирования всегда находился на Солнце. Это облегчит последующее написание кода управления спутником. При значении, равном 90°, обеспечивается именно такое положение. Данное утверждение можно проверить с помощью ПО GMAT, используя метод `EclipseLocator`.

Код управления спутником

От кода управления требуется обеспечить такое положение солнечных батарей, при котором они будут направлены на Солнце. Сделать это можно, например, с помощью алгоритма ПД-регулирования (то же, что и ПИД-регулирование, но без интегральной составляющей):

```
let e = gyro;
wheels_bus.enable(); // Включение управления маховиками
var data = new Float32Array([0,0,((e) * 0.0002 + (e - e_prev) * 0.0008 +
↪  I*0.00000001)]); // ПД-регулят.
var byteView = new Uint8Array(data.buffer);  // Формирование требуемого типа данных
↪  (см. API)
wheels_bus.transmit(byteView);  // Передача управления на маховик
```

Данная часть алгоритма регулирует угловую скорость спутника по оси Z, стремясь свести ее к нулю. Здесь строка «`let e = gyro;`» задает рассогласование для алгоритма ПД-регулирования. Так как в конечном итоге необходимо свести скорость к нулю, то рассогласование в данном случае просто равно угловой скорости аппарата.

Коэффициенты алгоритма подобраны экспериментально, с учетом наличия тренировочной задачи и достаточно большого количества попыток без дисконта. Остается определить, в какой момент времени данная часть кода должна работать.

Так как полезная нагрузка (метод `producer`) потребляет 10 Вт, а маховик — 5 Вт, следует избежать одновременного включения двух устройств. Для определения факта включения полезной нагрузки можно использовать флаг:

`producer.functions[0].enabled`.

Помимо этого, алгоритм ПИД-регулирования будет эффективен лишь тогда, когда угловая скорость аппарата еще не сведена к нулю. Исходя из этого, можно дополнить алгоритм логическим условием:

```
if (Math.abs(gyros.functions[2].angular_velocity) > 0.0000001 && flag &&
↪  !producer.functions[0].enabled){
    wheels_bus.enable();
    var data = new Float32Array([0,0,((e) * 0.0002 + (e - e_prev) * 0.0008 +
    ↪  I*0.00000001)]);
    var byteView = new Uint8Array(data.buffer);
    wheels_bus.transmit(byteView)
}
else{
    wheels_bus.disable();
}
```

Также логично отключать маховики при достижении заданной скорости, чтобы снизить потребление аппарата.

Однако если не учитывать поворот к Солнцу, то аппарат может застабилизироваться в невыгодном положении, отвернув солнечные батареи от него. Чтобы понять, повернут ли аппарат к Солнцу, можно использовать следующий метод аккумулятора:

```
let charge = accumulator.functions[0].charge;
```

В переменную будет записываться текущий заряд аккумулятора. Чтобы понять, заряжается или разряжается аккумулятор, необходимо убедиться, что на начало работы отключены энергоемкие устройства, а также сравнить заряд аккумулятора на текущей и предыдущей итерации работы программы. Для этого понадобится еще одна переменная. В примере она обозначается как «`old_charge`» и задается как глобальная (см. приведенный полный код программы ниже).

```
if (Math.abs(gyros.functions[2].angular_velocity) > 0.0000001 && flag &&
↪  !producer.functions[0].enabled){
    if (charge - old_charge > 0){
        wheels_bus.enable();
        var data = new Float32Array([0,0,((e) * 0.0002 + (e - e_prev) * 0.0008 +
        ↪  I*0.00000001)]);
        var byteView = new Uint8Array(data.buffer);
        wheels_bus.transmit(byteView)
    }
}
else{
    wheels_bus.disable();
}
```

Добавленное условие будет запускать работу маховиков тогда, когда спутник уже повернут на Солнце, чтобы минимизировать его разрядку.

Еще один момент, который необходимо учесть, это то, что на момент начала работы спутника аккумулятор не до конца заряжен. Поэтому лучше дождаться, когда он зарядится.

```
if (Math.abs(gyros.functions[2].angular_velocity) > 0.0000001 && flag &&
↪  !producer.functions[0].enabled){
    if (charge - old_charge > 0){
        wheels_bus.enable();
        var data = new Float32Array([0,0,((e) * 0.0002 + (e - e_prev) * 0.0008 +
        ↪  I*0.00000001)]);
        var byteView = new Uint8Array(data.buffer);
        wheels_bus.transmit(byteView)
    }
}
else{
    wheels_bus.disable();
}

if (charge - old_charge < 0 && !flag){
    flag = true;
}
```

Осталось реализовать работу полезной нагрузки. Для сбора данных необходимо собрать данные именно при пролете над полюсом, а также лучше не включать одновременно полезную нагрузку и маховики:

```
if (Math.abs(nav.functions[0].location[0] - 90) < 15 && charge > 1000){
    producer.functions[0].enable();
    transmitter.functions[0].enable();
}
else{
    producer.functions[0].disable();
    transmitter.functions[0].disable();
}
```

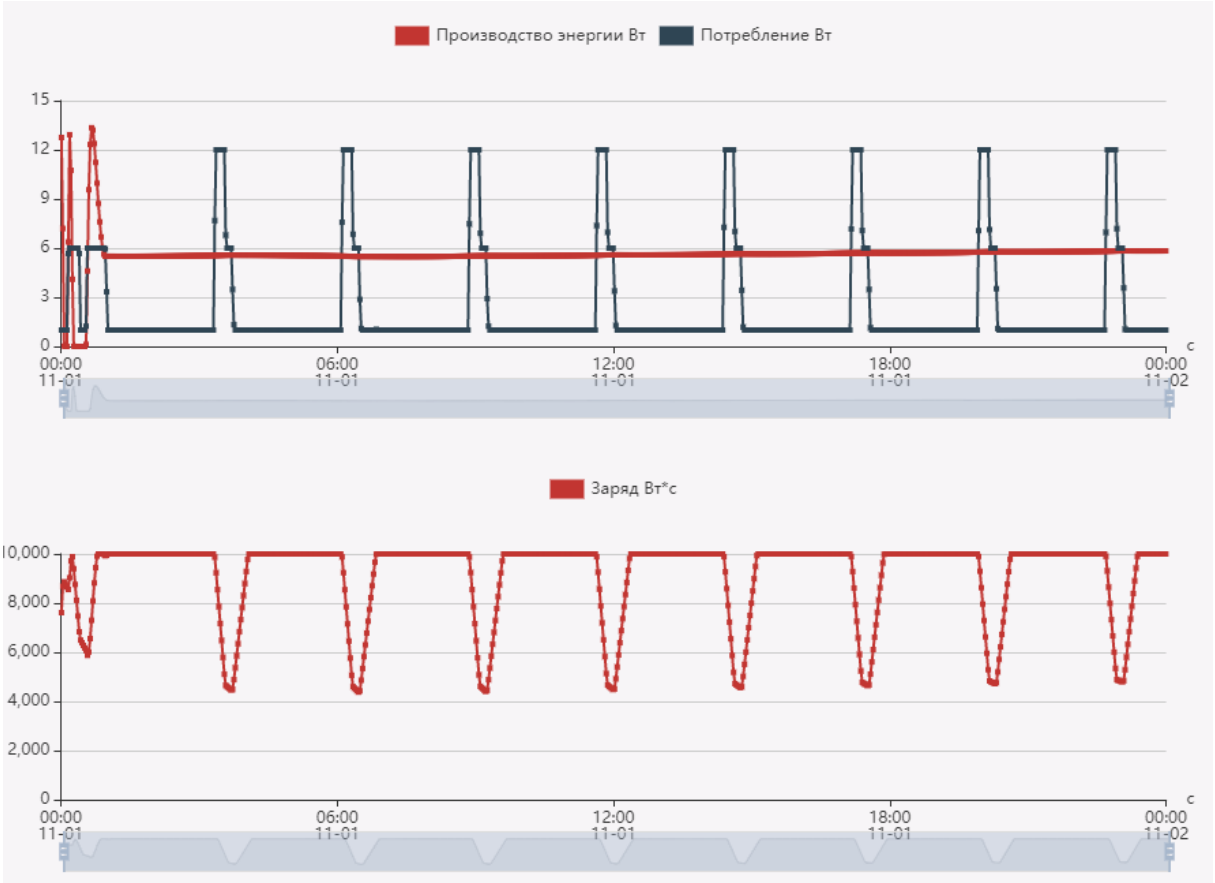
Пример программы-решения

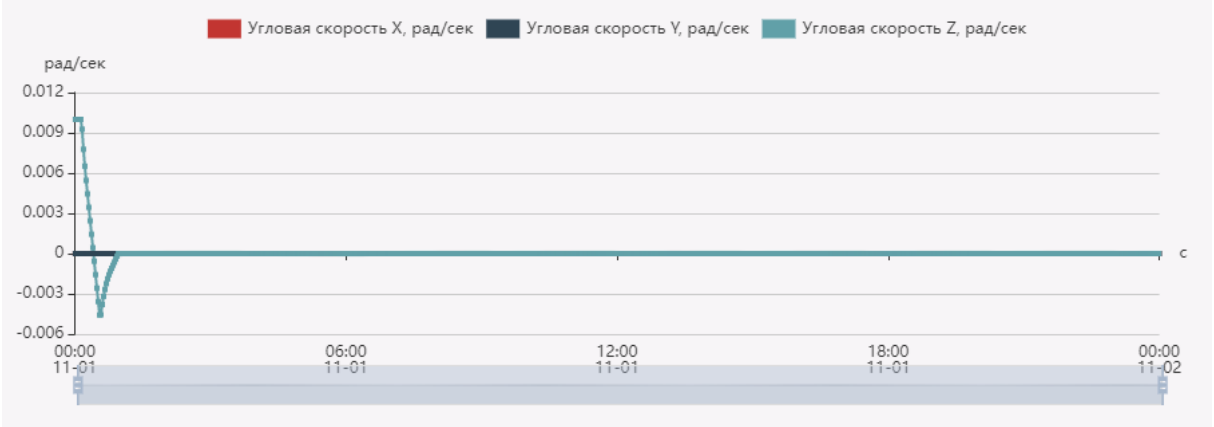
Ниже представлено решение на языке JavaScript.

```
1  'use strict';
2
3  var wheels;
4  var wheels_bus;
5  var gyros;
6  var nav;
7  var transmitter;
8  var producer;
9  var accumulator;
10
11 var e_prev = 0;
12 var old_charge = 0;
13 var flag = false;
14
15 function setup() {
16   wheels_bus = spacecraft.devices[0].functions[0];
17   gyros = spacecraft.devices[1];
18     nav = spacecraft.devices[2];
```

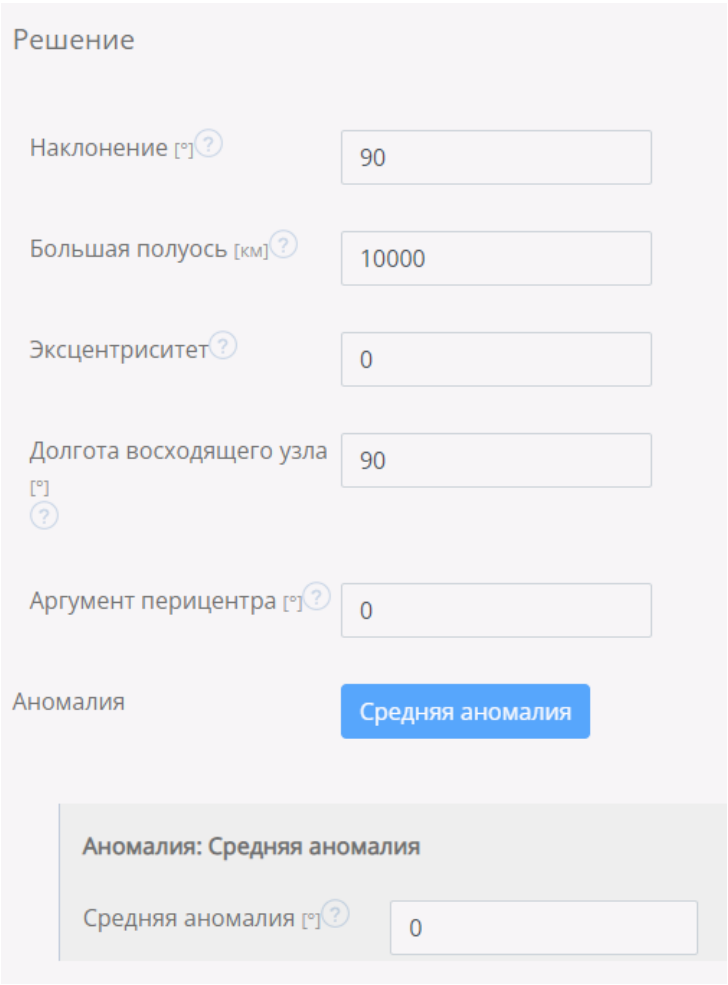
```
    transmitter = spacecraft.devices[3];
    producer = spacecraft.devices[4];
    accumulator = spacecraft.devices[5];
    wheels_bus.disable();
}

function loop() {
    transmitter.functions[0].disable();
    producer.functions[0].disable();
    nav.functions[0].enable();

    let gyro = gyros.functions[2].angular_velocity;

    let charge = accumulator.functions[0].charge;
    let e = gyro;
    if (Math.abs(gyros.functions[2].angular_velocity) > 0.0000001 && flag &&
    ↪  !producer.functions[0].enabled){
        if (charge - old_charge > 0){
            wheels_bus.enable();
            var data = new Float32Array([0,0,((e) * 0.0002 + (e - e_prev) * 0.0008
            ↪  + I*0.00000001)]);
            var byteView = new Uint8Array(data.buffer);
            wheels_bus.transmit(byteView)
        }
    }
    else{
            wheels_bus.disable();
    }

    if (charge - old_charge < 0 && !flag){
        flag = true;
    }

    old_charge = charge;
    e_prev = gyro;

    if (Math.abs(nav.functions[0].location[0] - 90) < 15 && charge > 1000){
        producer.functions[0].enable();
        transmitter.functions[0].enable();
    }
    else{
        producer.functions[0].disable();
        transmitter.functions[0].disable();
    }
}
```

Результат работы программы





Ответ.



Задача IV.2. (25 баллов)

Темы: элементы орбиты, энергобаланс, ориентация и стабилизация КА.

Условие

Группа студентов планирует запуск экспериментального роя из 3х спутников, двух наноспутников и одного мини спутника. В условиях роя наноспутники запущены на солнечно-синхронную орбиту со следующими параметрами:

- большая полуось: 6921 км;
- наклонение: 98°;
- эксцентриситет: 0;
- ДВУ: 50°;
- аргумент перицентра: 0°;
- истинная аномалия: 30°и 210°.

Задачей наноспутников является сбор данных над точками интереса с координатами 60,77, 295,3 и $-60,77$, 295,3. Проходя над заданными точками, каждый из наноспутников должен передать пакет данных на миниспутник, содержащих следующие данные:

- отметка бортового времени;

- текущие координаты наноспутника;
- байт данных от полезной нагрузки.

Данные на наноспутнике не могут храниться больше минуты. С вероятностью 0,012 при передаче между наноспутником и миниспутником могут быть повреждены (любой передаваемый бит может измениться на противоположный).

Получая данные от наноспутников, материнский спутник должен переформировать их, представив каждую запись в виде строки с отметкой времени, указанием длины байт показаний широты и долготы, широтой и долготой наноспутника в момент снятия показаний и 1 байтом данных:

```
YYYY-MM-DDT00:00:00.000Z+byte_len_cords+float_latitude+float_longitude+byte
```

Строка должна быть преобразована в тип Uint8Array. Координаты широты и долготы необходимо округлить до 2 знака после запятой. Передача данных между спутниками возможна при наличии прямой видимости между ними.

Напишите код управления наноспутниками и миниспутником, а также задайте орбиту миниспутника.

Время начала моделирования: 1 декабря 2022 года 00:00 по UTC. Длительность моделирования: 8 часов.

Ссылка на API: https://disk.yandex.ru/i/vtOinkLGddjI-A.

Пример данных, которые должны получиться на миниспутнике для отправки на Землю:

```
[50,48,50,50,45,49,50,45,48,49,84,48,55,58,53,50,58,51,54,46,49,48,48,90,12,49,
49,46,51,56,45,49,51,57,46,57,51,77]
```

Если декодировать его в строку, можно увидеть как он был собран:

```
'2022-12-01T07:52:36.100Z\x0c11.38-139.93M'
```

В данном пакете присутствует:

1. `'2022-12-01T07:52:36.100Z'` — дата и время в формате ISO.
2. `'\x0c'` — соответствует числу 12, что равно количеству байт координат.
3. `'11.38-139.93'` — координаты наноспутника, снятые в указанный момент времени.
4. `'M'` — байт, считанный с полезной нагрузки наноспутника.

Радиус Земли, м: 6371008,8.

Гравитационный параметр, $\mu = G \cdot M$, м^3/с^2: $3,986004418 \cdot 10^{14}$.

Угол видимости над точками интереса: 30°.

Скорость передачи между наноспутником и миниспутником: 20 байт/сек.

Скорость передачи между Землей и миниспутником: 3600 байт/сек.

Такт моделирования: 0,1 сек.

Максимально возможная высота орбиты: 16000 км от центра Земли.

Начисляется 0,03125 балла за каждый переданный в правильном формате пакет с миниспутника при соблюдении следующих условий:

- координаты соответствуют времени пролета над точкой интереса;
- при пролете над точкой интереса присутствует прямая видимость между наноспутником и миниспутником;
- высота орбиты меньше 16000 км от центра Земли.

Решение

Задача состоит из двух частей: подбор орбиты и написание кода управления спутниками.

Подбор орбиты

Чтобы обеспечить видимость наноспутников, оптимально использовать эллиптическую орбиту с апогеем, расположенным между точками интереса. Так как точки равноудалены от экватора, то аргумент перицентра можно выбрать равным либо 0, либо 180.

Учитывая ограничение по высоте, можно подобрать соотношение большой полуоси и эксцентриситета. Формулы для нахождение перицентра и апоцентра через большую полуось и эксцентриситет выглядят следующим образом:

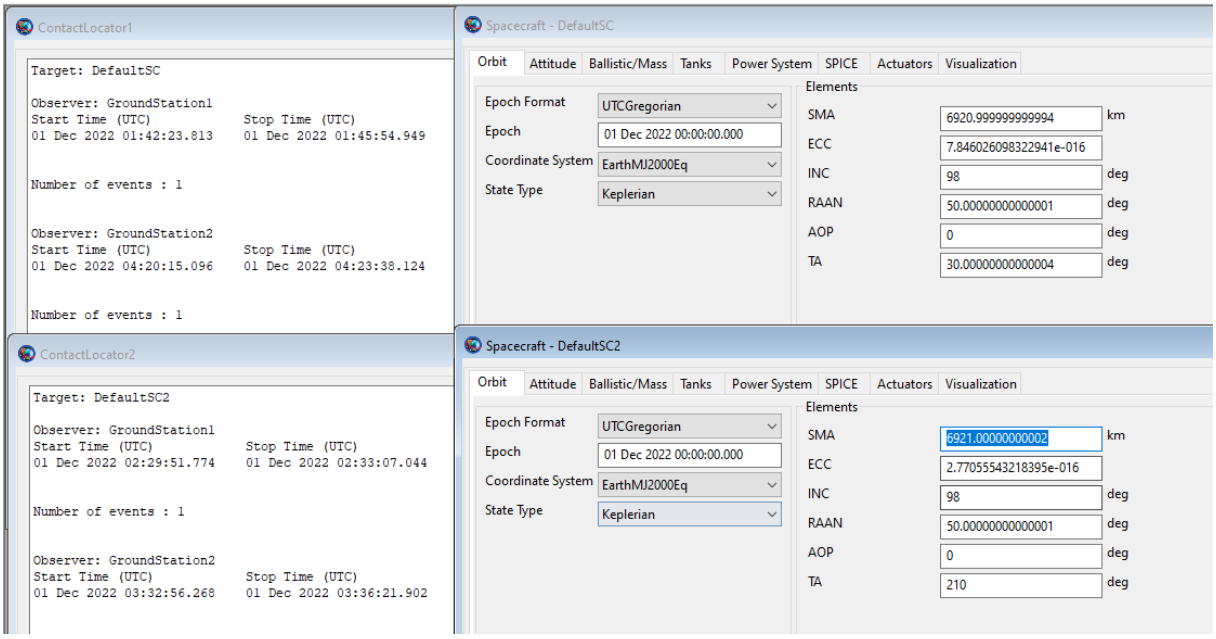
$$r_p = (1 - e)a$$

$$r_a = (1 + e)a$$

Используя данные формулы, можно подобрать подходящее соотношение большой полуоси и эксцентриситета. При этом чем больше будет эксцентриситет, тем дольше миниспутник будет находиться в апогее, что может обеспечить максимальное время видимости наноспутников. Пусть высота перигея будет равна примерно 6471 км (100 км высоты от поверхности Земли — линия Кармана).

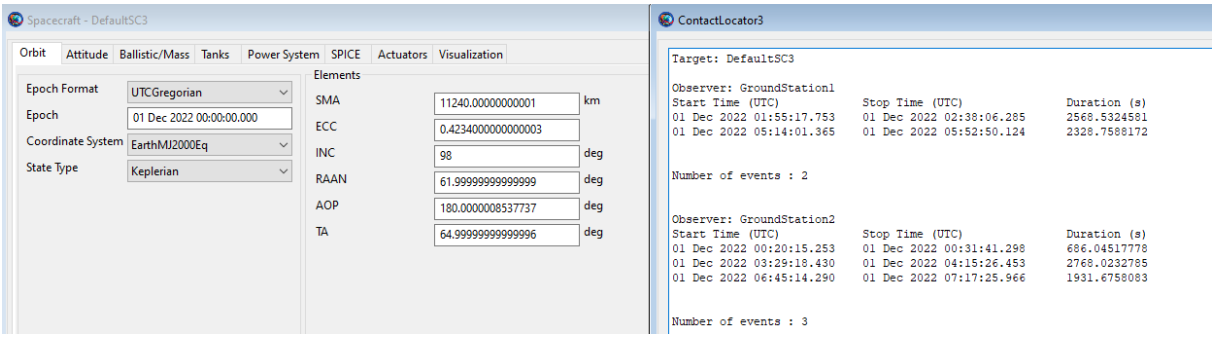
Большая полуось	**Эксцентриситет**	**Высота апогея**	**Высота перигея**
12000 км	0,4606	17527,191 км	6472,71 км
11000 км	0,4116	15527,591 км	6472,391 км
11240 км	0,4234	15999,007 км	6480,975 км

Также стоит определить время прохождения наноспутников над точками интереса. Для этого удобно использовать ПО GMAT и метод `ContactLocator`.



Получается, что наноспутники 4 раза пройдут над точками интереса. Если орбита миниспутника также сможет пройти примерно в эти же периоды времени над ними, то наноспутники будут в зоне видимости миниспутника.

Необходимо с помощью эллиптической орбиты обеспечить проход над точкой интереса с координатами (60,77, 295,3) в период с 01:42:43 до 02:23:07, а надо другой — с 03:32:56 до 04:23:38. Каждое окно связи длится около 50 минут.



Код управления спутником

Основным ограничивающим фактором при передачи данных является низкая скорость передачи между наноспутником и миниспутником. Для этого пакет данных вначале эффективно упаковывается, а потом, для защиты от помех, кодируется кодом Хемминга.

Пакет данных для передачи на наноспутнике стоит формировать с минимально необходимым количеством данных, чтобы уложиться в ограничение по скорости обмена. Как вариант, можно составить пакет из 2 байт преамбулы, 2 байт времени с начала запуска, байта данных с полезной нагрузки, $2 \cdot 2$ байт координат с точностью до 2 знаков после запятой. Пакет займет 10 байт, следовательно затем можно будет добавить помехоустойчивое кодирование, реализованное, например, с помощью кода Хэмминга. Для начала разберем пример программы без помехоустойчивого кодирования.

```
let producer = spacecraft.devices[1].functions[0];
let nav = spacecraft.devices[2].functions[0];
```

```
let flight_time = spacecraft.flight_time;

let produced_queue = []; // Задаем переменную для формирования пакета
let produced_packet = producer.read(1); // Считываем байт данных с полезной нагрузки

// Продолжаем формирование пакета только если удалось считать байт данных
if (produced_packet.length > 0) {
    let time_string = spacecraft.flight_time; // Создаем переменную для хранения
    ↪  времени

    // Преобразуем формат времени из Float в Uint8Array
    let u8a = new Uint8Array(2);
    u8a[0] = time_string & 0xff;
    u8a[1] = time_string >> 8;

    // Задаем преамбулу 'AB' для синхронизации начала пакетов
    var str = 'AB';
    // Разбиваем строку на символы и преобразуем каждый из них в ASCII
    var preamble = str.split('').map(function(x) {return x.charCodeAt(0)});
    // Преобразуем полученные ASCII-символы в Uint8Array
    let preamb_send = new Uint8Array(preamble);
    // Формируем пакет
    produced_queue.push(preamb_send[0]);  // Добавляем первый байт преамбулы
    produced_queue.push(preamb_send[1]);  // Добавляем второй байт преамбулы
    produced_queue.push(u8a[0]);  // Добавляем первый байт времени
    produced_queue.push(u8a[1]);  // Добавляем второй байт времени
    produced_queue.push(produced_packet[0]);  // Добавляем байт данных

    // Считываем координаты и преобразуем в необходимый формат
    let loc = nav.location;
    let cord = loc[0].toFixed(2)+loc[1].toFixed(2)
    let cord_bytes = new Uint8Array(new Int16Array([(loc[0]*100).toFixed(0),
    ↪  (loc[1]*100).toFixed(0)]).buffer);

        // Добавляем байты с координатами к формируемому пакету
    for (i of cord_bytes){
        produced_queue.push(i);
    }
}
    // Преобразуем пакет в Uint8Array
let packet = new Uint8Array(produced_queue);
if (packet.length == 0) {
    return;
}
// Отправляем пакет
let transmitter = spacecraft.devices[0].functions[0];
transmitter.transmit(packet);
```

Программа миниспутника ждёт приёма байт пакета в достаточном количестве, после чего пытается найти преамбулу. Если преамбула найдена, программа разбирает промежуточный пакет, собирает пакет заданный в условии и отсылает его на Землю.

```
let PACKET_SIZE = 9; // Задаем ожидаемый размер пакета

function setup() {
}

let resived_packet = []; // Задаем переменную для сохранения пакета

function loop() {
```

```
let flight_time = spacecraft.flight_time;
let resiver = spacecraft.devices[1].functions[0];
let resived = resiver.receive(PACKET_SIZE); // Принимаем пакет

if (resived.length == 0) {
    return;
}

resived_packet.push(...resived); // Добавляем принятые байты к тем, что были
↪  приняты ранее

 // Прерываем выполнение итерации, если данных недостаточно
if (resived_packet.length < PACKET_SIZE) {
    return;
}

// Определяем наличие преамбулы в принятом пакете
var flag = false;
for (var i = 0; i < resived_packet.length; i++) {
    if (String.fromCharCode(resived_packet[i]) == 'A'){
        if(String.fromCharCode(resived_packet[i + 1]) == 'B'){
            flag = true;
            resived_packet = resived_packet.slice(i)
            break;
        }
    }
}

var pr = 0;
let resived_pack = [];
var pico_time = 0;
if (flag){
     // Копируем из принятого пакета 3 и 4 байты с данными о времени и
     ↪  восстанавливаем их
    pico_time = resived_packet.slice((pr + 2), (pr + 4));
    let time = (pico_time[1] << 8);
    time = time | pico_time[0];
    // Преобразуем байты с данными о времени в формат ISO
    let date_string = (new Date((1638316800 + time +0.1)*1000)).toISOString()
    let u8a = new Uint8Array(date_string.length);
    // Преобразуем побайтово дату и время и добавляем в массив Uint8Array
    for (var i = 0; i < date_string.length; i++) {
        u8a[i] = date_string.charCodeAt(i)
        resived_pack.push(u8a[i]);
    }
    // Копируем байт данных из принятого пакета
    let data = resived_packet.slice(pr + 4);
    // Копируем данные с координатами
    let cord = resived_packet.slice(pr + 5);
    // Преобразуем и добавляем к формируемому пакету координаты с наноспутника
    let cord_arr = new Int16Array(new Uint8Array(cord).buffer,0,2);
    let cord_byte = (cord_arr[0]/100).toFixed(2)+(cord_arr[1]/100).toFixed(2)
    resived_pack.push(cord_byte.length);
    for (i in cord_byte) {
        resived_pack.push(cord_byte.charCodeAt(i));
    }
    // Добавляем байт данных с наноспутника
    resived_pack.push(data[0]);

    // Преобразуем и отправляем пакет на Землю
```

```
        let packet = new Uint8Array(resived_pack);
        let transmitter = spacecraft.devices[0].functions[0];
        transmitter.transmit(packet);

        resived_packet = resived_packet.slice(pr+4)
    }
}
```

Дополненные кодом Хэмминга программы представлены ниже.

Код управления миниспутником

```
let PACKET_SIZE = 9;
let resived_packet = [];
let resived_stream = [];

function setup() {
}


function loop() {
    let flight_time = spacecraft.flight_time;
    let resiver = spacecraft.devices[1].functions[0];
    let resived = resiver.receive(PACKET_SIZE);

    if (resived.length == 0) {
        return;
    }

    resived_stream.push(...resived);

    if (resived_stream.length >= PACKET_SIZE*2) {
        return;
    }

    // TODO check length % 2
    let decoded = converter(decode(resived_stream))
    resived_stream = []

    resived_packet.push(...decoded);


    runtime.debug('' + String(resived_packet));

    var flag = false;
    for (var i = 0; i < resived_packet.length; i++) {
        if (String.fromCharCode(resived_packet[i]) == 'A'){
            if(String.fromCharCode(resived_packet[i + 1]) == 'B'){
                flag = true;
                resived_packet = resived_packet.slice(i)
                break;
            }
        }
    }
    if (resived_packet.length < PACKET_SIZE) {
        return;
    }
    var pr = 0;
    let resived_pack = [];
```

```
48      var pico_time = 0;
49      if (flag){
50          pico_time = resived_packet.slice((pr + 2), (pr + 4));
51          let time = (pico_time[1] << 8);
52          time = time | pico_time[0];
53          let date_string = (new Date((1669852800 + time +0.1)*1000)).toISOString()
54          let u8a = new Uint8Array(date_string.length);
55          for (var i = 0; i < date_string.length; i++) {
56              u8a[i] = date_string.charCodeAt(i)
57              resived_pack.push(u8a[i]);
58          }
59
60          let data = resived_packet.slice(pr + 4);
61
62          let cord = resived_packet.slice(pr + 5);
63          let cord_arr = new Int16Array(new Uint8Array(cord).buffer,0,2);
64          let cord_byte = (cord_arr[0]/100).toFixed(2)+(cord_arr[1]/100).toFixed(2)
65          resived_pack.push(cord_byte.length);
66          for (i in cord_byte) {
67              resived_pack.push(cord_byte.charCodeAt(i));
68          }
69
70          resived_pack.push(data[0]);
71          let packet = new Uint8Array(resived_pack);
72          let transmitter = spacecraft.devices[0].functions[0];
73
74          transmitter.transmit(packet);
75
76          resived_packet = resived_packet.slice(pr+4)
77      }
78  }
79
80  var receiver;
81  var writer;
82
83  var createString = (len, char) => new Array(len + 1).join(char);
84  var createCode = (number, len = 8) => createString(len -
    ↪  number.toString(2).length, '0') + number.toString(2);
85  let isExponentTwo = (num) => !(num & (num - 1)) && num !== 0;
86  var getBit = (number, index) => (number >> index) % 2;
87
88  function decode(data) {
89      let out = [];
90      for (let number of data)
91          out.push(hammingDecode(number));
92      return out;
93  }
94
95  function hammingDecode(number) {
96      if (number >= 128)  // Ошибка выпала на левый бит
97          number -= 128;
98      let code = createCode(number, 7).split('').map(Number);
99      let bits = Array(7);
100     for (let i = 0; i < 7; i++)
101         bits[i] = isExponentTwo(i + 1) ? 0 : code[i];
102     for (let i = 0; i < 7; i++)
103         if (isExponentTwo(i + 1))
104             for (let j = 0; j < 7; j++)
105                 if (i !== j && getBit(j + 1, Math.log2(i + 1)))
106                     bits[i] ^= bits[j];
```

```
    if (code.join('') === bits.join(''))
        return parseInt(getUsefulBits(bits).join(''), 2);
    let error = 0;
    for (let i = 0; i < 7; i++)
        if (isExponentTwo(i + 1))
            if (bits[i] !== code[i])
                error += i + 1;
    bits = removeError(bits, error - 1);
    return parseInt(getUsefulBits(bits).join(''), 2);
}

function getUsefulBits(bits) {
    let useful = [];
    for (let i = 0; i < 7; i++)
        if (!isExponentTwo(i + 1))
            useful.push(bits[i]);
    return useful;
}

function removeError(array, index) {
    array[index] = +!(array[index] === 1);
    return array;
}

function converter(data) {
    let out = [];
    for (let i = 0; i < data.length; i += 2)
        out.push(parseInt(createCode(data[i], 4) + createCode(data[i + 1], 4),
        ↪  2));
    return out;
}
```

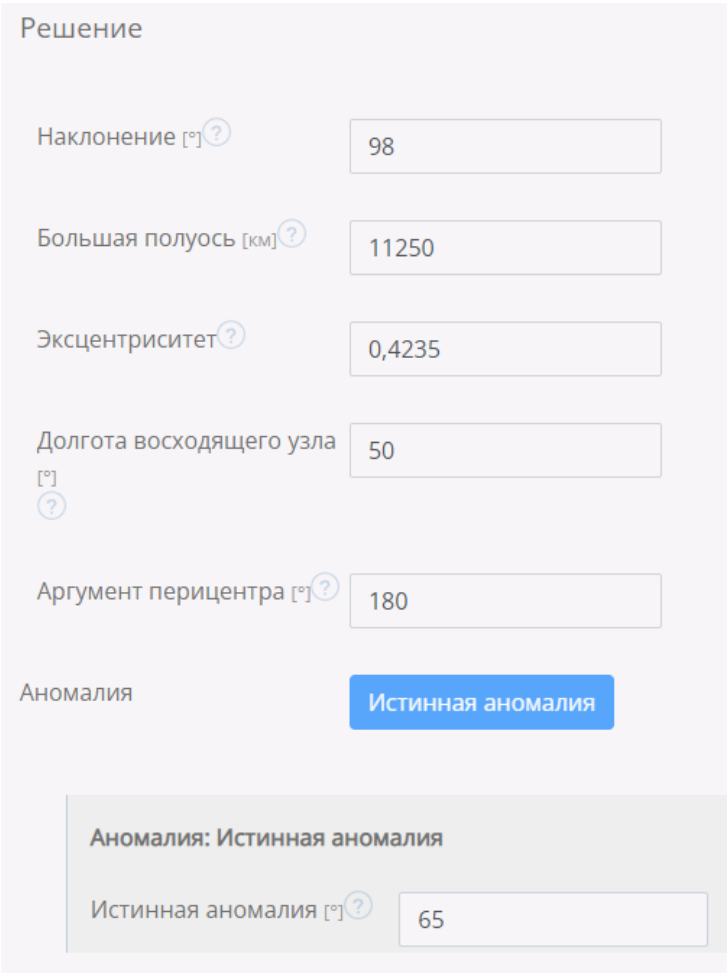
Код управления наноспутниками

```
let PACKET_SIZE = 9;
let resived_packet = [];
let resived_stream = [];

function setup() {
}


function loop() {
    let flight_time = spacecraft.flight_time;
    let resiver = spacecraft.devices[1].functions[0];
    let resived = resiver.receive(PACKET_SIZE);

    if (resived.length == 0) {
        return;
    }

    resived_stream.push(...resived);

    if (resived_stream.length >= PACKET_SIZE*2) {
        return;
    }

    // TODO check length % 2
    let decoded = converter(decode(resived_stream))
```

```
    resived_stream = []

    resived_packet.push(...decoded);


    runtime.debug('' + String(resived_packet));

    var flag = false;
    for (var i = 0; i < resived_packet.length; i++) {
        if (String.fromCharCode(resived_packet[i]) == 'A'){
            if(String.fromCharCode(resived_packet[i + 1]) == 'B'){
                flag = true;
                resived_packet = resived_packet.slice(i)
                break;
            }
        }
    }
    if (resived_packet.length < PACKET_SIZE) {
        return;
    }
    var pr = 0;
    let resived_pack = [];
    var pico_time = 0;
    if (flag){
        pico_time = resived_packet.slice((pr + 2), (pr + 4));
        let time = (pico_time[1] << 8);
        time = time | pico_time[0];
        let date_string = (new Date((1669852800 + time +0.1)*1000)).toISOString()
        let u8a = new Uint8Array(date_string.length);
        for (var i = 0; i < date_string.length; i++) {
            u8a[i] = date_string.charCodeAt(i)
            resived_pack.push(u8a[i]);
        }

        let data = resived_packet.slice(pr + 4);

        let cord = resived_packet.slice(pr + 5);
        let cord_arr = new Int16Array(new Uint8Array(cord).buffer,0,2);
        let cord_byte = (cord_arr[0]/100).toFixed(2)+(cord_arr[1]/100).toFixed(2)
        resived_pack.push(cord_byte.length);
        for (i in cord_byte) {
            resived_pack.push(cord_byte.charCodeAt(i));
        }

        resived_pack.push(data[0]);
        let packet = new Uint8Array(resived_pack);
        let transmitter = spacecraft.devices[0].functions[0];

        transmitter.transmit(packet);

        resived_packet = resived_packet.slice(pr+4)
    }
}

var receiver;
var writer;

var createString = (len, char) => new Array(len + 1).join(char);
var createCode = (number, len = 8) => createString(len -
↪  number.toString(2).length, '0') + number.toString(2);
```

```
let isExponentTwo = (num) => !(num & (num - 1)) && num !== 0;
var getBit = (number, index) => (number >> index) % 2;

function decode(data) {
    let out = [];
    for (let number of data)
        out.push(hammingDecode(number));
    return out;
}

function hammingDecode(number) {
    if (number >= 128)  // Ошибка выпала на левый бит
        number -= 128;
    let code = createCode(number, 7).split('').map(Number);
    let bits = Array(7);
    for (let i = 0; i < 7; i++)
        bits[i] = isExponentTwo(i + 1) ? 0 : code[i];
    for (let i = 0; i < 7; i++)
        if (isExponentTwo(i + 1))
            for (let j = 0; j < 7; j++)
                if (i !== j && getBit(j + 1, Math.log2(i + 1)))
                    bits[i] ^= bits[j];
    if (code.join('') === bits.join(''))
        return parseInt(getUsefulBits(bits).join(''), 2);
    let error = 0;
    for (let i = 0; i < 7; i++)
        if (isExponentTwo(i + 1))
            if (bits[i] !== code[i])
                error += i + 1;
    bits = removeError(bits, error - 1);
    return parseInt(getUsefulBits(bits).join(''), 2);
}

function getUsefulBits(bits) {
    let useful = [];
    for (let i = 0; i < 7; i++)
        if (!isExponentTwo(i + 1))
            useful.push(bits[i]);
    return useful;
}

function removeError(array, index) {
    array[index] = +!(array[index] === 1);
    return array;
}

function converter(data) {
    let out = [];
    for (let i = 0; i < data.length; i += 2)
        out.push(parseInt(createCode(data[i], 4) + createCode(data[i + 1], 4),
        ↪  2));
    return out;
}
```

Ответ.



Задача IV.3. (30 баллов)

Темы: элементы орбиты, ориентация и стабилизация КА.

Условие

Задачей спутника является проведение биологического эксперимента на высотах свыше пояса Ван Аллена (свыше 4000 км от поверхности Земли). Для эксперимента требуется, чтобы скорость вращения по оси Z была постоянной и равнялась 0,1 рад/сек. Начальная скорость вращения аппарата по всем осям равна нулю.

Накопленные данные необходимо сбрасывать в город Йоханненсбург ($-26,202$; 28,044). Однако передатчик на спутнике стоит маломощный, и для успешной передачи высота над городом должна быть менее 300 км.

Пакет полезных данных на спутнике генерируется раз в минуту при условии, что высота больше 4000 км и скорость вращение по оси Z соответствует требованиям.

Предложите программу управления спутником и элементы его орбиты. Данные собираются и сохраняются автоматически. Отправка данных также происходит автоматически при наличии окна связи.

Время начала моделирования: 1 декабря 2022 года в 00:00 по UTC. Длительность моделирования: 24 часа.

API для управления спутником: https://disk.yandex.ru/i/BsPMt8BKqukeGQ.

Радиус Земли, м: 6371008,8.

Гравитационный параметр, $\mu = G \cdot M$, м^3/с^2: $3,986004418 \cdot 10^{14}$.

Угол видимости над городом: 60°.

Начисляется 0,03 балла за каждый переданный пакет при соблюдении следующих условий:

- пакет был сгенерирован при соблюдении условий по высоте и скорости вращения;
- скорость вращения отличается от целевой не более чем на 0,01°/сек (0,0001745 рад/сек), а ускорение не более 0,01°/сек^2;
- при наличии окна связи высота над городом не более 300 км;
- по осям X и Y скорость равна нулю.

Решение

Задача состоит из двух частей: подбор орбиты и написание кода управления спутниками.

Подбор орбиты

Так как по условиям задачи необходимо обеспечить нахождение спутника на высотах 4000 км и выше и на высоте 300 км и ниже, то для данной задачи подойдет лишь эллиптическая орбита. При этом перигей орбиты необходимо расположить над городом Йоханненсбург.

Также оптимальным будет обеспечить проход над городом более одного раза, чтобы увеличить окно связи. Логично подобрать орбиту так, чтобы проход над городом был около 12 часов и 24 часов по UTC, так как спутник успеет собрать больше данных (хотя в текущей постановке задачи это требование не строго обязательно).

Для начала необходимо подобрать такое соотношение большой полуоси и эксцентриситета, чтобы соответствовать условиям задачи по требованиям к высоте. Формулы для нахождение перицентра и апоцентра через большую полуось и эксцентриситет выглядят следующим образом:

$$r_p = (1 - e)a$$

$$r_a = (1 + e)a$$

Используя данные формулы, можно подобрать несколько подходящих соотношений большой полуоси и эксцентриситета. Учитывая, что высота пролета над городом должна быть не более 300 км, стоит брать значение перигея, приблизительно равное 100 км, чтобы увеличить окно связи и снизить требования к точности прохода строго над городом.

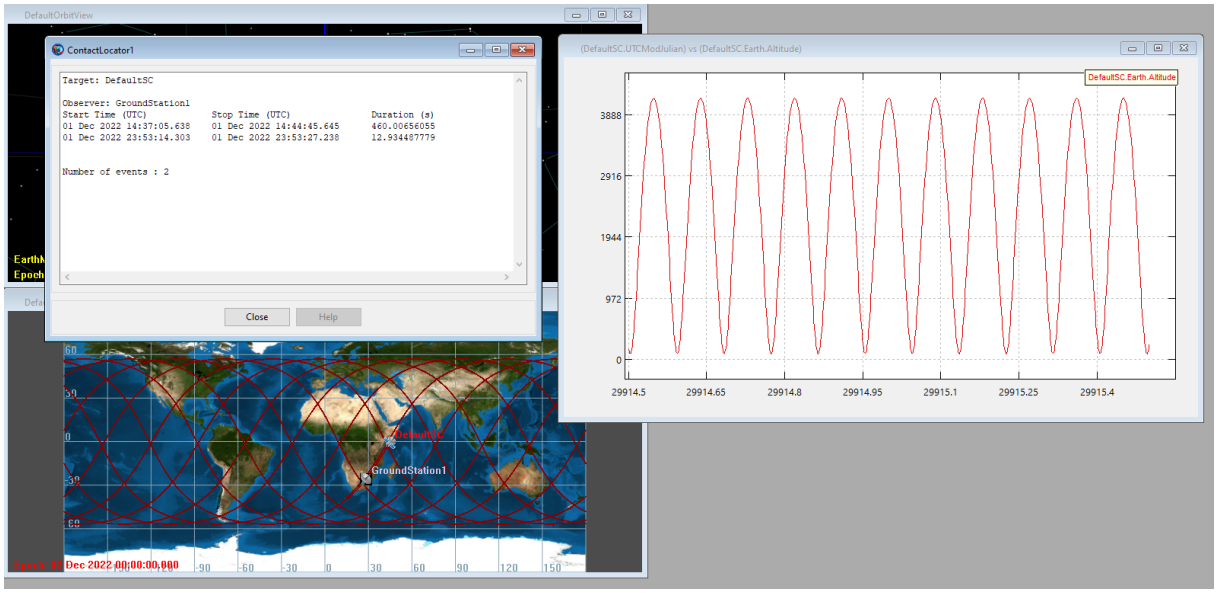
Большая полуось	**Эксцентриситет**	**Высота апогея**	**Высота перигея**	**Период**
11000 км	0,4115	9155,491 км	102,491 км	≈ 3 ч 11 мин
8500 км	0,2385	4156,241 км	101,741 км	≈ 2 ч 10 мин
8000 км	0,191	3156,991 км	100,991 км	≈ 2 ч

Высота апогея и перигея указаны без учета радиуса Земли. Из приведенного выше расчета следует, что значение большой полуоси ниже 8500 км не будут удовлетворять требованию по апогею при стремлении обеспечить 100 км в перигее. Помимо

этого важно учитывать и орбитальный период: для прохода спутника над городом порядка двух раз с разницей по времени в 12 часов необходимо подобрать период орбиты, обеспечивающий практически целое количество витков спутника вокруг Земли в сутки. В данном случае у второй орбиты более подходящий период, так что можно рассмотреть ее.

Ориентировочное значение аргумент перицентра можно рассчитать из широты Йоханнесбурга: $360 - 26,202 = 333,798$.

Оставшиеся параметры могут варьироваться в зависимости от времени прохода над городом. Их можно рассчитать аналитически или подобрать в ПО GMAT, используя метод `ContactLocator`.



Код управления спутником

В данном коде аналогично первой задаче целесообразно использовать алгоритм ПИД-регулирования. Отличием будет прежде всего формирование другого рассогласования, а также добавление интегральной составляющей.

```
let gyro = gyros.functions[2].angular_velocity;
    var e = gyro - 0.1 // Формирование рассогласования
    var speed = ((e) * 0.00009 + (e - e_prev) * 0.0003 + I*0.00000002) //
    ↪  ПИД-регулирование
        var data = new Float32Array([0,0,speed]);
        var byteView = new Uint8Array(data.buffer);
        wheels_bus.transmit(byteView)

    e_prev = e
```

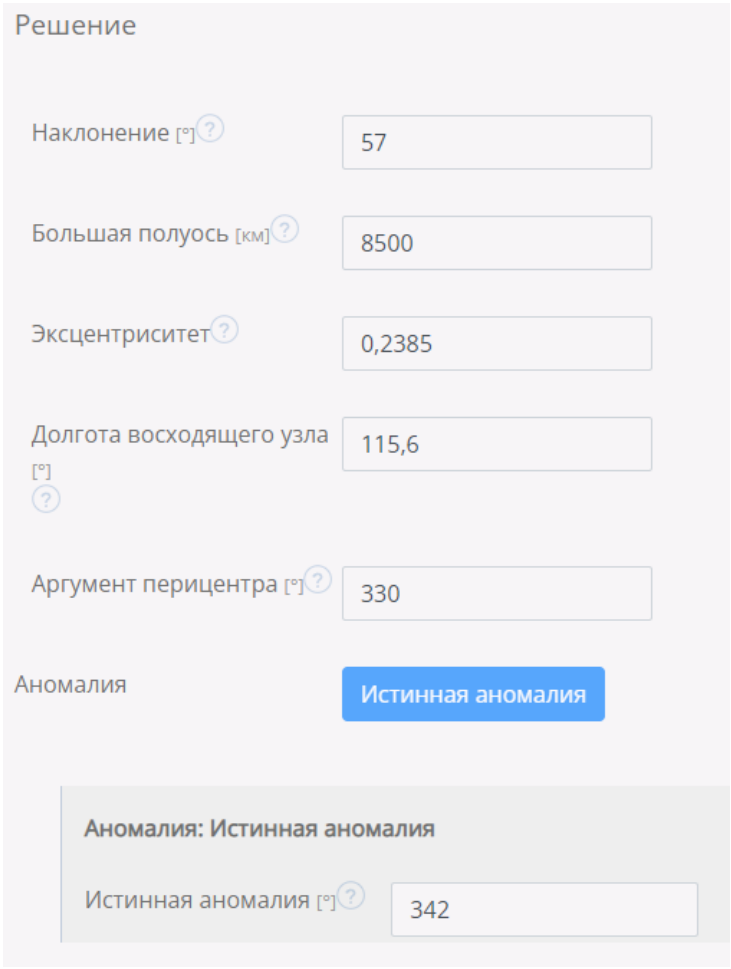
Так как дополнительных условий в данной задаче нет, то следует просто отладить работу ПИД-регулирования, скорректировав коэффициенты. Ниже приведен полный код программы. Результат работы программы:

Пример программы-решения

Ниже представлено решение на языке JavaScript.

```
'use strict';

var wheels;
var wheels_bus;
var gyros;
var nav;
var transmitter;
var producer;
var accumulator;

var I = 0;
var e_prev = 0;

function setup() {
  wheels_bus = spacecraft.devices[0].functions[0];
  gyros = spacecraft.devices[1];
    nav = spacecraft.devices[2];
    transmitter = spacecraft.devices[3];
    producer = spacecraft.devices[4];
}
function loop() {

    let gyro = gyros.functions[2].angular_velocity;
    var e = gyro - 0.1
    var speed = ((e) * 0.00009 + (e - e_prev) * 0.0003 + I*0.00000002)
        var data = new Float32Array([0,0,speed]);
        var byteView = new Uint8Array(data.buffer);
        wheels_bus.transmit(byteView)
        I+=e;

    e_prev = e
}
```

Ответ.



Задача IV.4. (6 баллов)

Темы: программирование микроконтроллеров, ШИМ, PWM, STM32.

Условие

Задачей программно реализованного пропорционального регулятора на плате управления является ограничение потребления двигателя-маховика в зависимости от внешнего сигнала.

Двигатель потребляет в диапазоне от 0 до 5,5 В в зависимости от заданного количества оборотов. Управление двигателем происходит по ШИМ-сигналу с диапазоном регулирования от 0 до 65535. Результатом работы регулятора является необходимое значение скважности ШИМ, которое нужно задать для двигателя-маховика.

Через некоторое время потребовалось узнать значение пропорционального коэффициента. Однако у пользователей не оказалось доступа к прошивке платы управления. Тогда было решено попробовать установить значение коэффициента экспериментально.

Для этого были измерены данные телеметрии.

Внешний сигнал, В	2,8	3,1	2,9	2,5	2,4	2,0
Потребление двигателя, В	3,67	4,06	3,8	3,27	3,14	2,62

В измерениях присутствует определенная погрешность.

Чему может быть равен коэффициент пропорционального регулятора? Ответ указать в целых. Потребление двигателя в зависимости от ШИМ-сигнала считать линейным.

Решение

Значение k определяется по формуле: $k = U_{\text{мах}}/5,5 \cdot 65535/U_{\text{ист}}$. Определим значение k для каждого значения напряжений.

Внешний сигнал, В	2,8	3,1	2,9	2,5	2,4	2,0
Потребление двигателя, В	3,67	4,06	3,8	3,27	3,14	2,62
ШИМ	15617,757	15605,4	15613,354	15585,416	15589,386	15609,246

Среднее значение коэффициента — 15603.

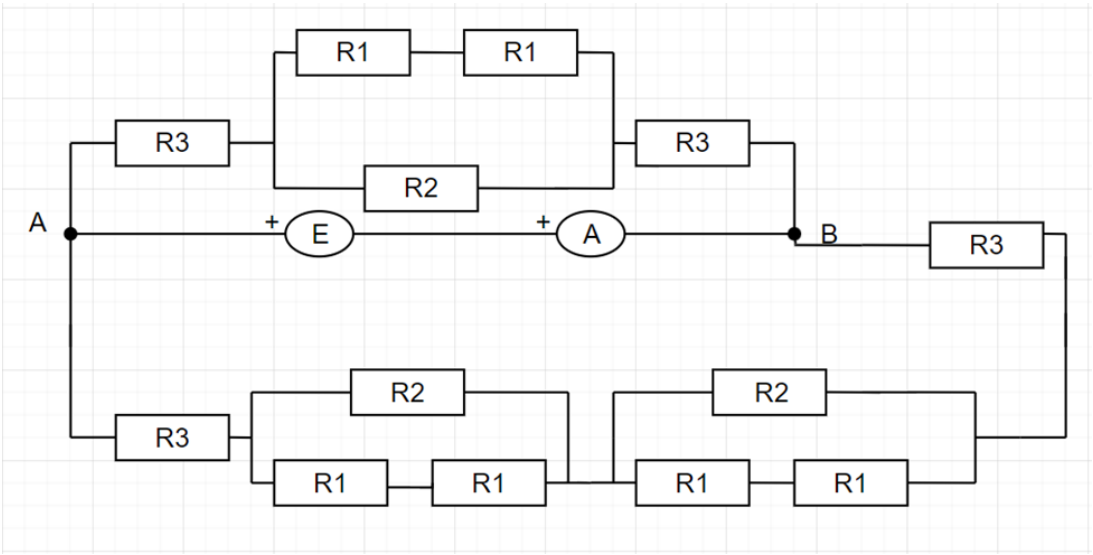
Ответ: 15603.

Задача IV.5. (6 баллов)

Темы: схемотехника, электрические цепи.

Условие

Была создана схема из резисторов, как показано на схеме.



Номиналы резисторов: $R_1 = 400$ Ом, $R_2 = 800$ Ом, $R_3 = 1800$ Ом. Схема была подключена к реальному источнику питания $E = 24$ В. Внутреннее сопротивление $R = 20$ Ом. Внутреннее сопротивление амперметра $R_a = 10$ Ом.

Определите:

1. эквивалентное сопротивление между точками A и B в кОм (ответ округлите до двух знаков после запятой);
2. показание амперметра I_a в мА (ответ округлите до целых).

Решение

Находим общее сопротивление резисторов 1 и 2:

$$R_{12} = \frac{2 \cdot R_1 \cdot R_2}{2R_1 + R_2} = 400.$$

Находим сопротивление верхней части цепи:

$$R_{up} = R_3 + R_3 + R_{12} = 4 \times 10^3.$$

Находим сопротивление нижней части цепи:

$$R_d = R_3 + R_3 + R_{12} + R_{12} = 4,4 \times 10^3.$$

Находим сопротивление между точками A–B:

$$R_{ab} = \frac{R_{up} \cdot R_d}{R_{up} + R_d} = 2,095238 \times 10^3.$$

Находим показание амперметра:

$$R_{un} = R_{ab} + R_e + R_a = 2,125238 \times 10^3.$$

$$I_a = \frac{E}{R_{un}} = 0,011293.$$

Ответ: 1. 2,095 кОм; 2. 11 мА.

Задача IV.6. (6 баллов)

Темы: помехоустойчивое кодирование.

Условие

Для проверки помехоустойчивости разрабатываемого канала связи сообщение отправили через среду Houston на конструктор спутника. Изначальное сообщение перед отправкой запаковали в пакет, содержащий в себе адрес устройства отправителя, устройства получателя, длину пакета в байтах и `msg id` сообщения.

Для обозначения адресов и длины сообщения отводится по одному двухразрядному шестнадцатеричному числу, а для `msg id` — четырехразрядное.

Адрес отправителя		**Адрес получателя**		**Длина**		**MSG_ID**				**Данные**		
0	1	1	F	0	9	0	1	1	9	9	...	1

Полученный пакет перед отправкой закодировали кодом Хэмминга. Полученное в двоичном виде сообщение выглядит следующим образом:

```
00000000 01101001 00000000 10011001 00000000 01100110 00000000 00110011
01010101 00000000 00001101 10011001 00000000 00000000 11000011 01100110
00000000 01101001 00000000 01101001 00000000 00000000
```

Учитывая набор возможных команд, определите название поля, в котором возникла ошибка в результате помех.

Примечание: рекомендуется использовать ПО Houston для определения команды (http://orbicraft3d.sputnix.ru/doku.php?id=les_01_01).

Решение

Посмотрев на формат предоставления данных, можно определить, что длина одного пакета 8 бит. Отсюда логично предположить, что сообщение закодировано кодом Хемминга (8,4), или так называемый расширенный код Хэмминга.

Декодировать код можно вручную или используя программную реализацию декодирования кода Хэмминга. Например, может подойти код Хэмминга из второй задачи:

```
var createString = (len, char) => new Array(len + 1).join(char);
var createCode = (number, len = 8) => createString(len - number.toString(2).length,
↪  '0') + number.toString(2);
let isExponentTwo = (num) => !(num & (num - 1)) && num !== 0;
var getBit = (number, index) => (number >> index) % 2;

var data = new Uint8Array([0, 105, 0, 153, 0, 102, 0, 51, 85, 0, 13, 153, 0, 0, 195,
↪  102, 0, 105, 0, 105, 0, 0])// Преобразованное исходное сообщение

function decode(data) {
    let out = [];
    for (let number of data){
        out.push(hammingDecode(number));
    }
    return out;
}

function hammingDecode(number) {
    if (number >= 128)
        number -= 128;
    let code = createCode(number, 7).split('').map(Number);
    let bits = Array(7);
    for (let i = 0; i < 7; i++)
        bits[i] = isExponentTwo(i + 1) ? 0 : code[i];
    for (let i = 0; i < 7; i++)
        if (isExponentTwo(i + 1))
            for (let j = 0; j < 7; j++)
                if (i !== j && getBit(j + 1, Math.log2(i + 1)))
                    bits[i] ^= bits[j];
    if (code.join('') === bits.join(''))
        return parseInt(getUsefulBits(bits).join(''), 2);
    let error = 0;
    for (let i = 0; i < 7; i++)
        if (isExponentTwo(i + 1))
            if (bits[i] !== code[i])
                error += i + 1;
    bits = removeError(bits, error - 1);
    console.log(number) // Вывод байта с ошибкой
```

```
    console.log(error - 1) // Вывод номера бита с ошибкой
    return parseInt(getUsefulBits(bits).join(''), 2);
}

function getUsefulBits(bits) {
    let useful = [];
    for (let i = 0; i < 7; i++)
        if (!isExponentTwo(i + 1))
            useful.push(bits[i]);
    return useful;
}

function removeError(array, index) {
    array[index] = +!(array[index] === 1);
    return array;
}

function converter(data) {
    let out = [];
    for (let i = 0; i < data.length; i += 2)
        out.push(parseInt(createCode(data[i], 4) + createCode(data[i + 1], 4), 2));
    return out;
}

console.log(converter(decode(data)))
```

```
13
5
▶ (22) [0, 1, 0, 9, 0, 6, 0, 11, 13, 0, 7, 9, 0, 0, 3, 6, 0, 1, 0, 1, 0, 0]
```

Рис. 1. Результат работы кода

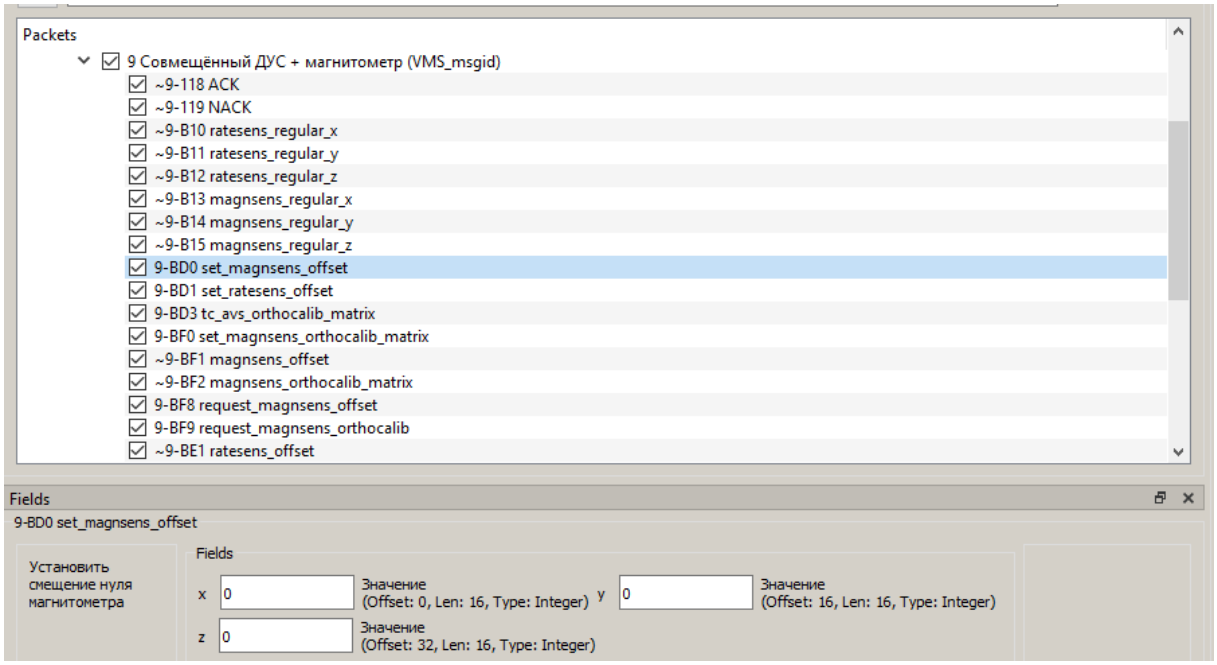
Байт с изначальным значением «13» содержит ошибку в 5-ом бите. Этот бит в сообщении является 11 по счету.

Исправив бит с ошибкой и преобразовав сообщение, получится исходное сообщение: 0109060BD0790036010100

При сопоставлении со структурой пакета, получим:

Адрес отправителя		**Адрес получателя**		**Длина**		**MSG_ID**				**Данные**
0	1	0	9	0	6	0	B	D	0	790036010100

Используя ПО Хьюстон, можно определить, что получателем был ДУС, а также найти среди доступных команд устройства ту, что использовалась в пакете данных.



11 по счету байт соответствует первому байту данных, а следовательно, ошибка была допущена в поле X.

Ответ: 1. Поле X; 2. 11.

Задача IV.7. (7 баллов)

Темы: элементы орбиты, межспутниковая связь.

Условие

Спутник передает данные на соединенную сеть любительских радиостанций различного формата. При получении пакетов одной станцией данные автоматически распространяются на остальные по средствам сети интернет. Радиостанции работают не постоянно, а только в местное рабочее время (с 8:00 до 17:00). Всего в составе сети, на данный момент 4 станции: в Москве (55,755864, 37,617698), Хабаровске (48,480229, 135,071917), Афинах (37,975534, 23,734855) и Мехико (19,4194, 260,8544). Набранный спутником битовый массив данных требуется передать за сутки. Для выполнения задания требуется подобрать орбиту космического аппарата, которая позволит передать всю необходимую информацию в срок.

Начисляется 0,00025 балла за каждую 0,1 секунду при наличии окна связи над городом.

Углы видимости спутника над станциями: 60°.

Максимально допустимая высота: 10000 км от центра Земли.

Время начала моделирования: 1 декабря 2022 года в 00:00 по UTC. Длительность моделирования: 24 часа.

Решение

Первоначально стоит перевести местное время станций в UTC, чтобы стандартизировать все окна связи по единому времени:

- Хабаровск (UTC+10): 22:00–07:00 по UTC;
- Москва (UTC+3): 05:00–14:00 по UTC;
- Афины (UTC+2): 06:00–16:00 по UTC;
- Мехико (UTC-6): 14:00–23:00 по UTC.

Так как необходимо за сутки охватить 4 точки, то имеет смысл рассматривать относительно низкие околоземные орбиты. Находясь на такой орбите, спутник может пролетать над городом 2 раза в сутки.

Максимальная широта у г. Москва — 55,755864, что накладывает ограничение на наклонение орбиты. Чтобы гипотетически проходить над Москвой, необходимо выбирать наклонение примерно от 50°и выше.

При этом Афины, Мехико и Хабаровск относительно равноудалены друг от друга по долготе:

- Мехико-Афины: 122,880455;
- Хабаровск-Мехико: 125,782483;
- Хабаровск-Афины: 111,337062.

Тогда число витков за сутки должно быть кратно трем, чтобы равномерно покрыть 3 города. Москва, в свою очередь находится недалеко от Афин, и может быть покрыта с ними одним витком.

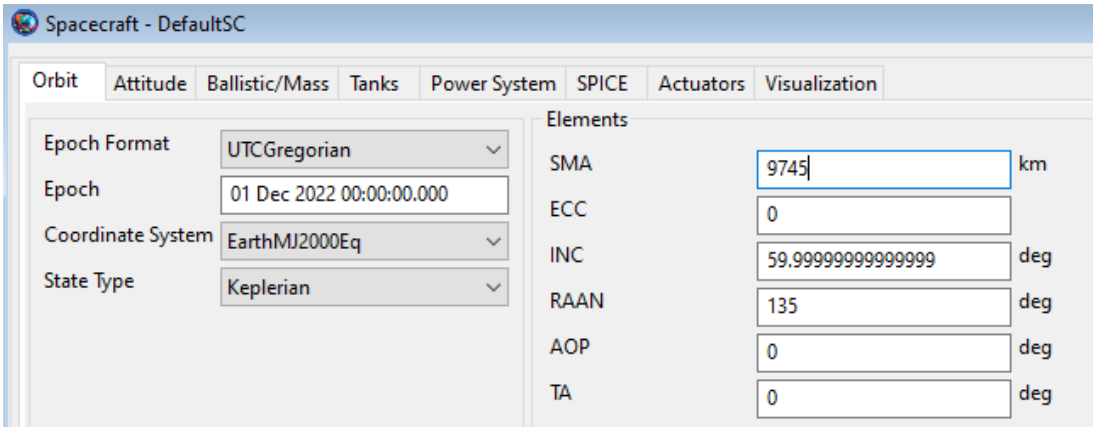
Учитывая, что максимально допустимая высота равно 10000 км, логично предположить, что максимальное количество витков, которое удовлетворяет условию — 9. Для 9 витко период обращения равен:

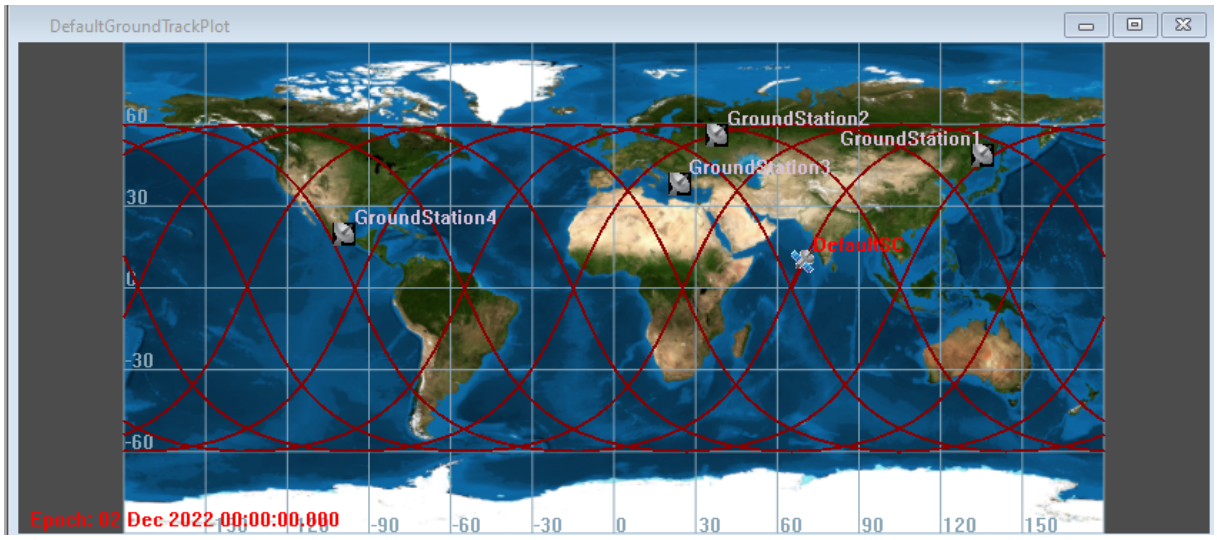
$$T = 86344/9 = 9593,78 \text{ сек} \approx 2 \text{ часа } 39 \text{ минут } 54 \text{ сек}.$$

Тогда высота орбиты:

$$a = \sqrt[3]{\frac{GMT^2}{4\pi^2}} = 9745 \text{ км}.$$

Так как первым городом, над которым необходимо пройти, является Хабаровск, то грубым приближением к решению будет взять долготу восходящего узла, равную долготе Хабаровска. Зафиксировав наклонение, равное 60°, и для простоты взяв эксцентриситет, равный 0, можно получить приблизительное решение в GMAT.



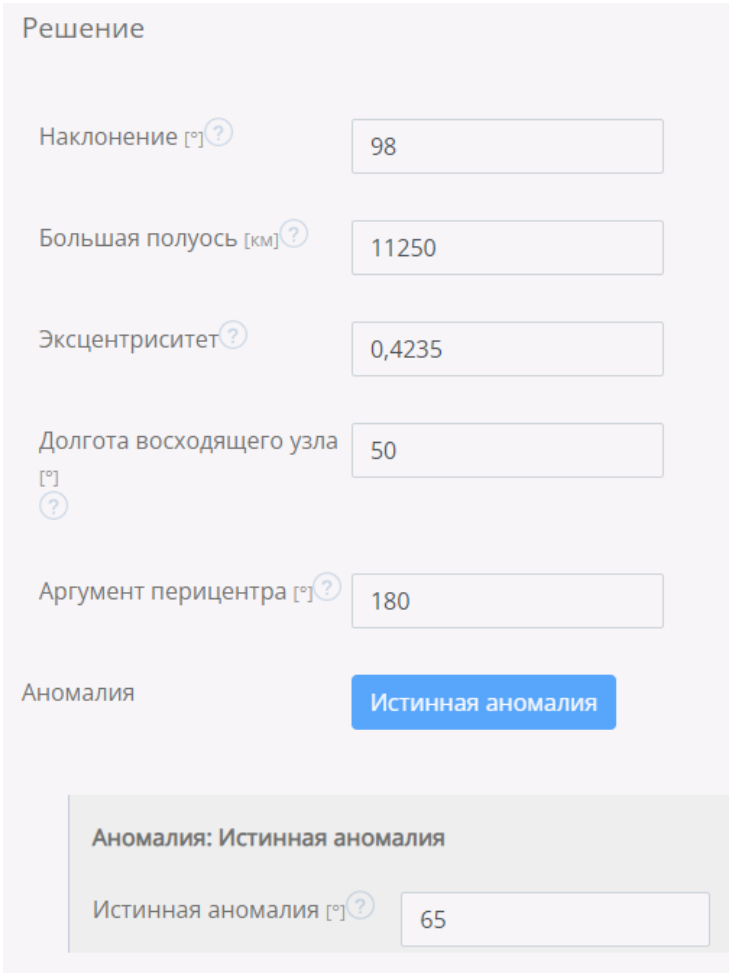


ContactLocator1

```
Target: DefaultSC

Observer: GroundStation1
Start Time (UTC)            Stop Time (UTC)             Duration (s)
01 Dec 2022 00:35:32.018    01 Dec 2022 00:37:29.791    117.77273300
01 Dec 2022 03:24:24.099    01 Dec 2022 03:32:50.653    506.55359075
01 Dec 2022 21:36:17.165    01 Dec 2022 21:46:18.783    601.61868094


Number of events : 3


Observer: GroundStation2
Start Time (UTC)            Stop Time (UTC)             Duration (s)
01 Dec 2022 05:46:35.420    01 Dec 2022 05:56:48.297    612.87667440
01 Dec 2022 08:36:39.034    01 Dec 2022 08:46:26.533    587.49907367
01 Dec 2022 11:26:29.561    01 Dec 2022 11:33:15.050    405.48937706


Number of events : 3


Observer: GroundStation3
Start Time (UTC)            Stop Time (UTC)             Duration (s)
01 Dec 2022 05:40:49.473    01 Dec 2022 05:45:30.654    281.18080739


Number of events : 1


Observer: GroundStation4
Start Time (UTC)            Stop Time (UTC)             Duration (s)
01 Dec 2022 22:21:42.987    01 Dec 2022 22:30:52.939    549.95285558


Number of events : 1
```

Данный результат можно улучшить, более точно рассчитав или подобрав в GMAT долготу восходящего узла и истинную аномалию.

Ответ.


Решение
Наклонение [°]
98
Большая полуось [км]
11250
Эксцентриситет
0,4235
Долгота восходящего узла [°]
50
Аргумент перицентра [°]
180
Аномалия
Истинная аномалия
Аномалия: Истинная аномалия
Истинная аномалия [°]
65