НИУ «МЭИ»

Кафедра Радиотехнических систем

Расчетно-пояснительная записка к курсовому проекту по дисциплине «Аппаратура потребителей СРНС»

Выполнил: Чепелев И.И.

Группа: ЭР-15-17

Проверил: Корогодин И.В.

Москва 2022

РЕФЕРАТ

Цель курсового проекта – разработка модулей разбора навигационного сообщения GPS и расчета положения спутника, предназначенных для использования в составе навигационного приемника.

Требования к разрабатываемому программному модулю:

- требования назначения;
- отсутствие утечек памяти;
- малое время выполнения;
- низкий расход памяти;
- корректное выполнение при аномальных входных данных.

Для достижения цели курсового проекта в работе выполняется ряд задач, соответствующих этапам проекта и контрольным мероприятиям:

- разработка модуля разбора символов навигационного сообщения;
- расчет положения КА в Matlab/Python и его проверка сторонними сервисами;
- реализация модуля расчета положения КА на C/C++ и его тестирование.

На первом этапе был реализован модуль разбора навигационного сообщения до структуры эфемерид и проведено сравнение результатов со сторонней программой.

На втором этапе требуется реализовать на языке Matlab или Python функцию расчета положения спутника GPS на заданный момент по шкале 3 времени UTC. В качестве эфемерид использовать данные, полученные на предыдущем этапе.

На третьем этапе требуется разработать на языке C/C++ функцию расчета положения спутника GPS на заданное время по шкале UTC, минимизируя время её исполнения и количество затрачиваемой оперативной памяти. Вызов функции не должен приводить к выбросу исключений или утечкам памяти при любом наборе входных данных.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Спутниковая система навигации — система, предназначенная для определения местоположения (географических координат) наземных, водных и воздушных объектов, а также низкоорбитальных космических аппаратов. Спутниковые системы навигации также позволяют получить скорость и направление движения приёмника сигнала. Кроме того, могут использоваться для получения точного времени. Такие системы состоят из космического оборудования и наземного сегмента (систем управления).

Принцип работы спутниковых систем навигации основан на измерении расстояния от антенны на объекте (координаты которого необходимо получить) до спутников, положение которых известно с большой точностью. Таблица положений всех спутников называется *альманахом*, которым должен располагать любой спутниковый приёмник до начала измерений. Обычно приёмник сохраняет альманах в памяти со времени последнего выключения и если он не устарел — мгновенно использует его. Каждый спутник передаёт в своём сигнале весь альманах. Таким образом, зная расстояния до нескольких спутников системы, с помощью обычных геометрических построений, на основе альманаха, можно вычислить положение объекта в пространстве.

Метод измерения расстояния от спутника до антенны приёмника основан на том, что скорость распространения радиоволн предполагается известной (на самом деле этот вопрос крайне сложный, на скорость влияет множество слабопредсказуемых факторов, таких как характеристики ионосферного слоя и пр.). Для осуществления возможности измерения времени распространяемого радиосигнала каждый спутник навигационной системы излучает сигналы точного времени, используя точно синхронизированные с системным временем атомные часы. При работе спутникового приёмника его часы синхронизируются с системным временем, и при дальнейшем приёме сигналов вычисляется задержка между временем излучения, содержащимся в самом сигнале, и временем приёма сигнала. Располагая этой информацией, навигационный приёмник вычисляет координаты антенны. Все остальные параметры движения (скорость, курс, пройденное расстояние) вычисляются на основе измерения времени, которое объект затратил на перемещение между двумя или более точками с определёнными координатами.

ЭТАП 1. ОБРАБОТКА ЛОГОВ НАВИГАЦИОННОГО ПРИЕМНИКА

На первом этапе необходимо создать программу в среде C/ C++, выполняющую функции аналогичные модулю разбора навигационного сообщения. Листинг созданной части программы приведен в приложении.

На данном этапе реализован расчет всех целочисленных параметров спутника. В дальнейшем будет дописана программа для расчета остальных параметров (с плавающей точкой).

```
 Hello, World
LNAV Ephemeris (slot = 221473810) =
        Crs      = -4.031250e+00
        Dn       = 1.606622e-09  [deg/s]
        M0       = -1.300067e+01         [deg]
        Cuc      = -1.061708e-07
```

```
        e        = 6.875264e-03
        Cus      = 8.167699e-06
        sqrtA    = 5.153699e+03
        toe      = 93600
        Cic      = 8.009374e-08
        Omega0   = 1.268903e+02  [deg]
        Cis      = 6.705523e-08
        i0       = 5.377037e+01  [deg]
        Crc      = 2.095312e+02
        omega    = 2.118201e+01  [deg]
        OmegaDot= -4.743470e-07         [deg/s]
        iDot     = -2.506795e-08        [deg/s]
        Tgd      = 6.984919e-09
        toc      = 93600
        af2      = 0.000000e+00
        af1      = 3.183231e-12
        af0      = 1.830263e-04
        WN       = 149
        IODC     = 8
        URA      = 0
        Health   = 0
        IODE2    = 9
        IODE3    = 9
        codeL2   = 1
        L2P      = 0

Process finished with exit code 0
```

Рисунок 1 – Данные, полученные в ходе компиляции кода через командную строку

SAT	PRN	Statu	IODE	IODC	Accu	Hea	Toe	Toc	Ttrans	A (m)	e	i0 (deg)	OMEGA0 (d	omega (deg	M0 (deg)	deltan (deg/s	OMEGAdot (d	IDOT (deg/s)	af0 (ns)
G20	20	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00	0.0000E+00	0.0000E+00	0.0
G21	21	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00	0.0000E+00	0.0000E+00	0.0
G22	22	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00	0.0000E+00	0.0000E+00	0.0
G23	23	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00	0.0000E+00	0.0000E+00	0.0
G24	24	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00	0.0000E+00	0.0000E+00	0.0
G25	25	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00	0.0000E+00	0.0000E+00	0.0
G26	26	OK	2313	9	0	00	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/02/14 03:05:11	26560614.536	0.00687526	53.77037	126.89030	21.18201	-13.00067	2.8919E-07	-4.7435E-07	-2.5068E-08	183026.3

af1 (ns/s	af2 (ns/s	TGD (ns)	BGD5a(ns	BGD5b(ns	Cuc(rad)	Cus(rad)	Crc(m)	Crs(m)	Cic(rad)	Cis(rad)	Code	Flag
0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0
0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0
0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0
0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0
0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0
0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0
0.0032	0.0000	7.0	0.0	0.0	-1.0617E-07	8.1677E-06	2.0953E+02	-4.0312E+00	8.0094E-08	6.7055E-08	1	0

Рисунок 2 – Данные, полученные в RTKNAVI

Из рисунка 2 видно, что рассчитанные параметры с программы в C++ совпадают с параметрами RTKNAVI.

ЭТАП 2. МОДЕЛИРОВАНИЕ ТРАЕКТОРИИ ДВИЖЕНИЯ

На данном этапе требуется реализовать функцию расчета положения спутника GPS на заданный момент по шкале времени UTC на языке Matlab или Python. Значения, полученные на предыдущем этапе, нужны нам в качестве эфемерид для моделирования.

Построить трехмерные графики множества положений спутника GPS с системным номером, соответствующим номеру студента по списку. Графики в двух вариантах: в СК ECEF WGS84 и соответствующей ей инерциальной СК. Положения должны соответствовать временному интервалу с 00:00 МСК 14 февраля до 00:00 МСК 15 февраля 2022 года. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал. Для выполнения данного этапа я использовал пакет математического моделирования Matlab.

Исходные данные:

Значения эфемерид, полученные в исходном сигнале и обработанные в первом пункте. Уточнение и проверка данных выполнено в RTKNAVI.

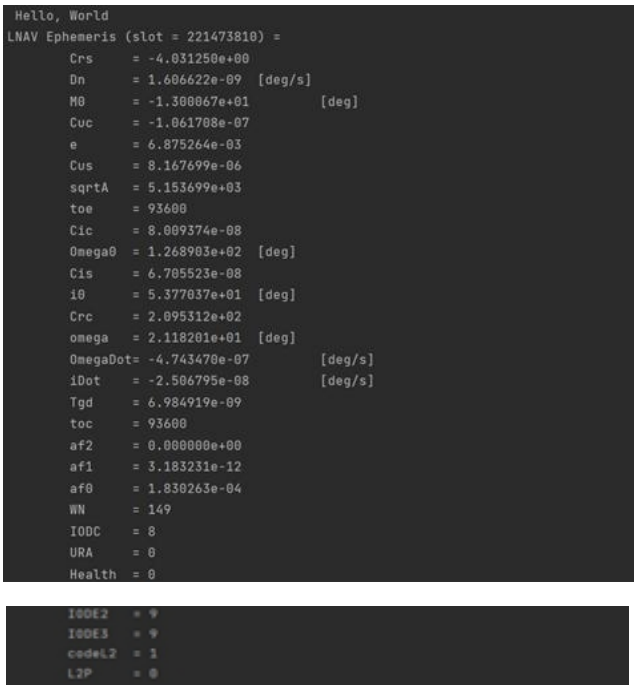

Рисунок 3 — Значения эфемерид на заданную дату

На прошлом этапе эфемериды получены для даты 14.02.2022 0:00:00, Далее на суточном интервале (93600 секунд) по алгоритму из ИКД рассчитываются координаты спутника в системе ECEF WGS 84. Перевод координат в инерциальную систему осуществляется по заданным формулам:

$$x' = x \cos(\theta) - y \sin(\theta)$$

$$y' = x \sin(\theta) + y \cos(\theta)$$

$$z' = z$$

where

$$\theta = \dot{\Omega}_e \, (t - t_0)$$

Далее с помощью встроенной функции ecef2enu центр декартовой системы координат перемещается в точку приема, местоположение которой определяется с помощью RTKNAVI (рисунок 5). Полученные координаты спутника относительно приемника с учетом их знака пересчитываются в полярную систему координат с помощью известных соотношений:

$$r = \sqrt{x^2 + y^2 + z^2}\,, \quad \cos\theta = \frac{z}{\sqrt{x^2 + y^2 + z^2}}, \quad \operatorname{tg}\varphi = \frac{y}{x}.$$

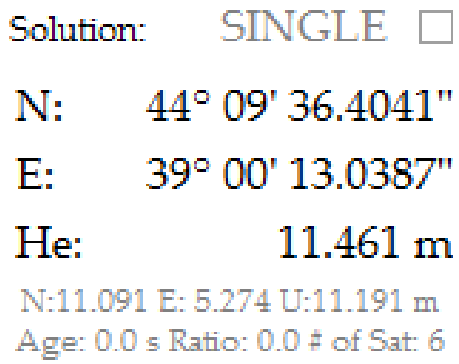

Рисунок 4 – Координаты приемника

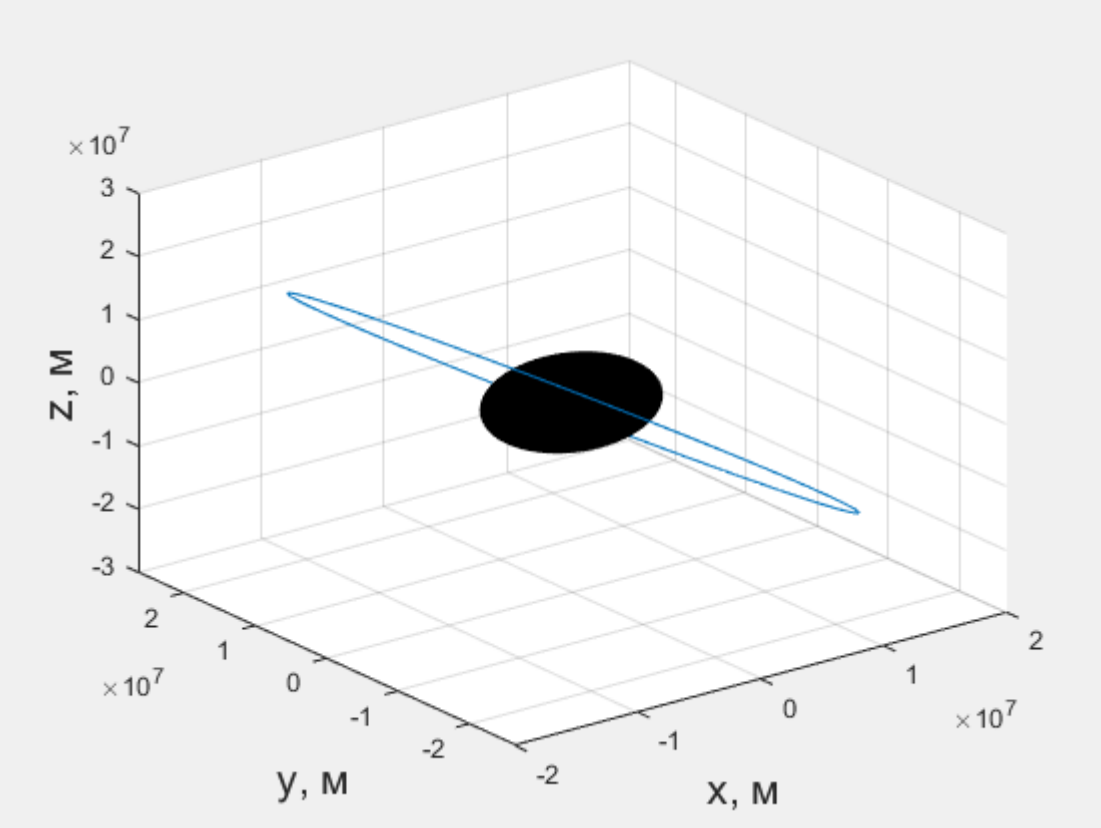

Рисунок 5 — Множество положений спутника GPS в системе ECI

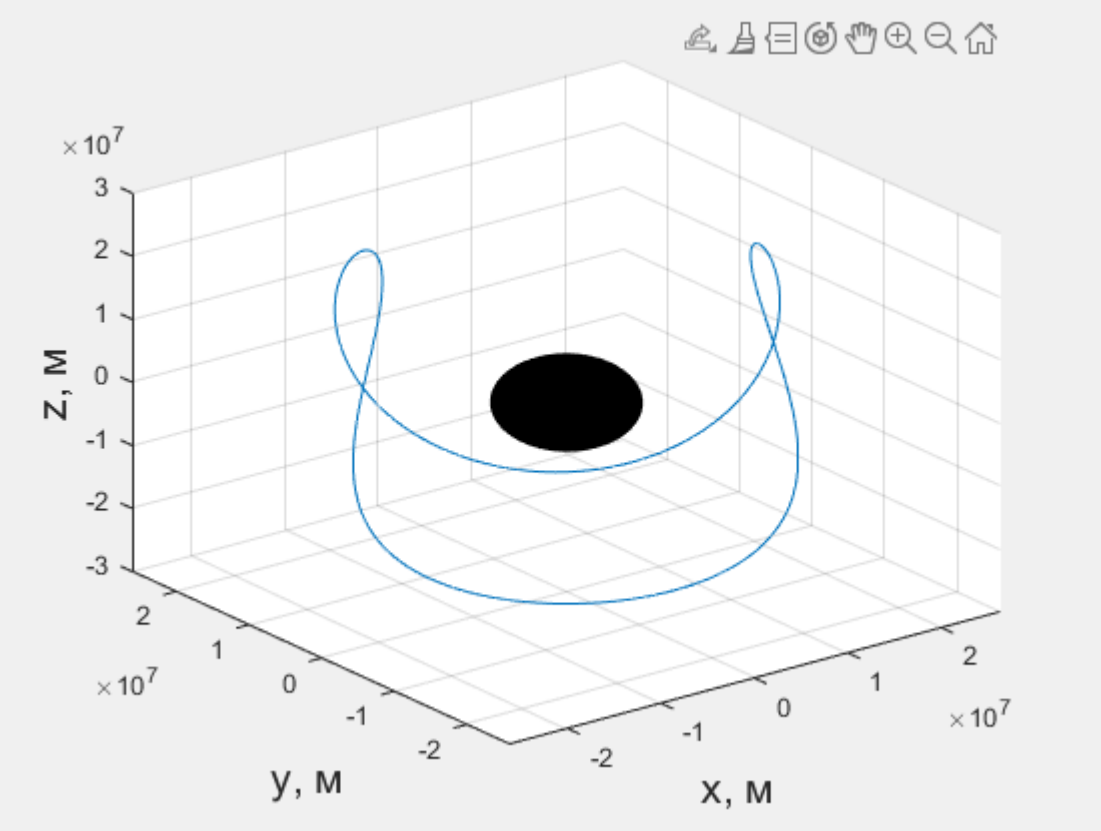

Рисунок 6 — Множество положений спутника GPS в системе ECEF WGS84

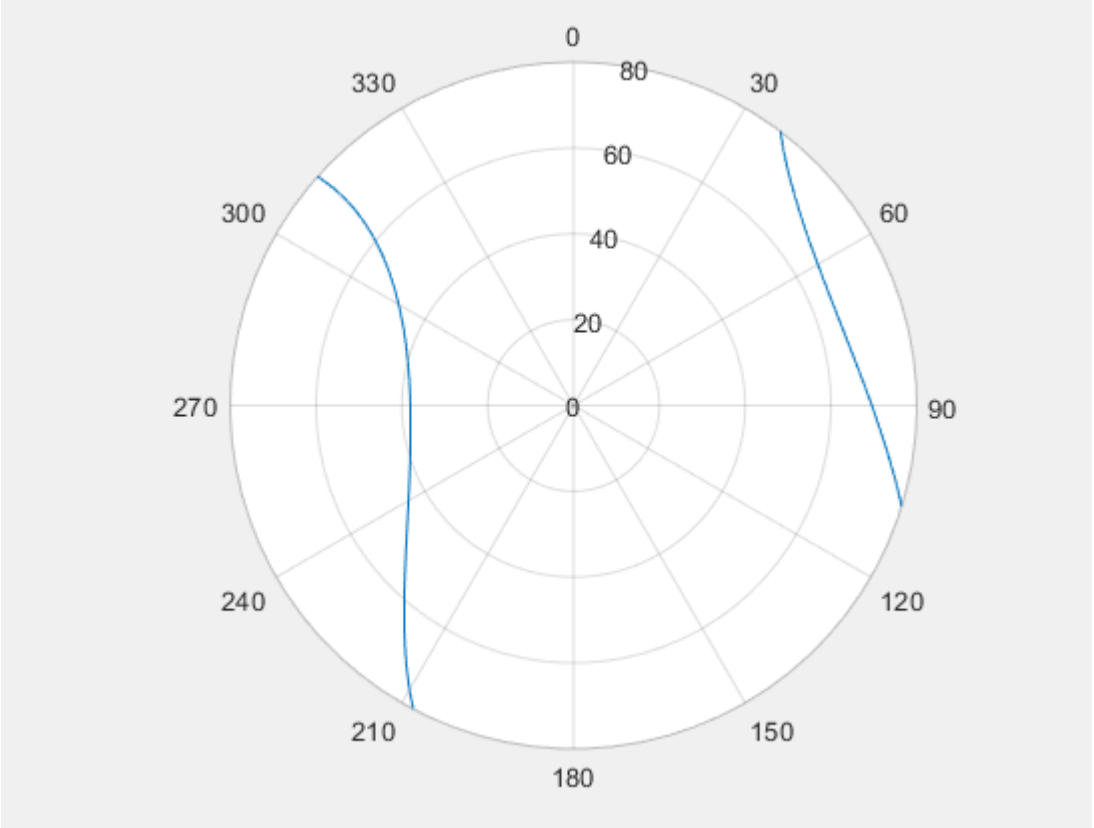

Рисунок 7 — Sky Plot

Для построения SkyView с помощью Trimble GNSS Planning Online необходимо выбрать заданный спутник, дату, рассматриваемый период, а также координаты точки приема. Полученные настройки приведены на рисунке 6.

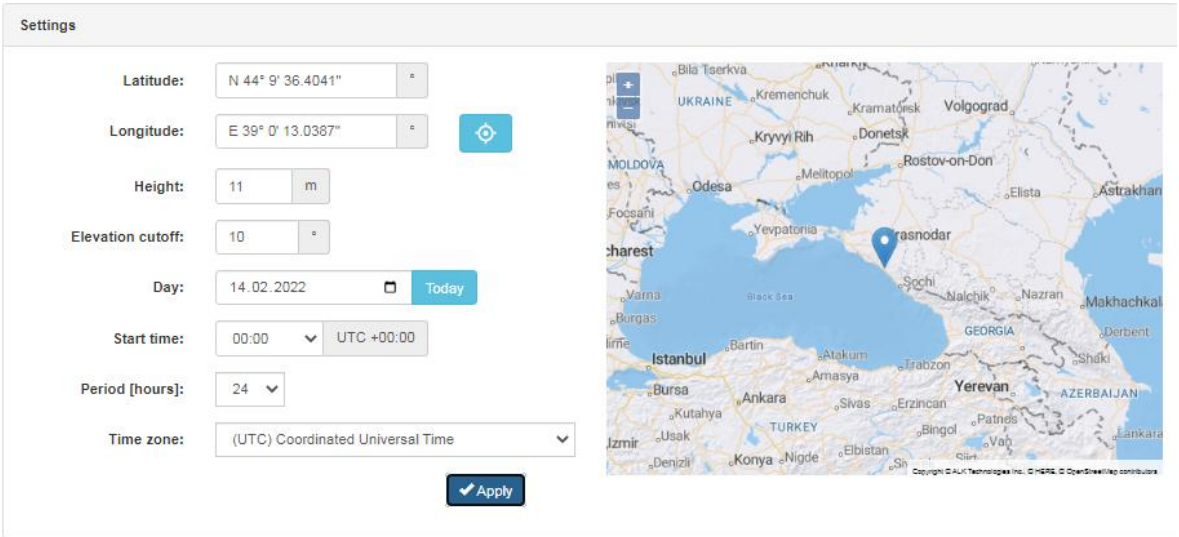

Рисунок 8 – Исходные данные для Trimble GNSS Planning Online

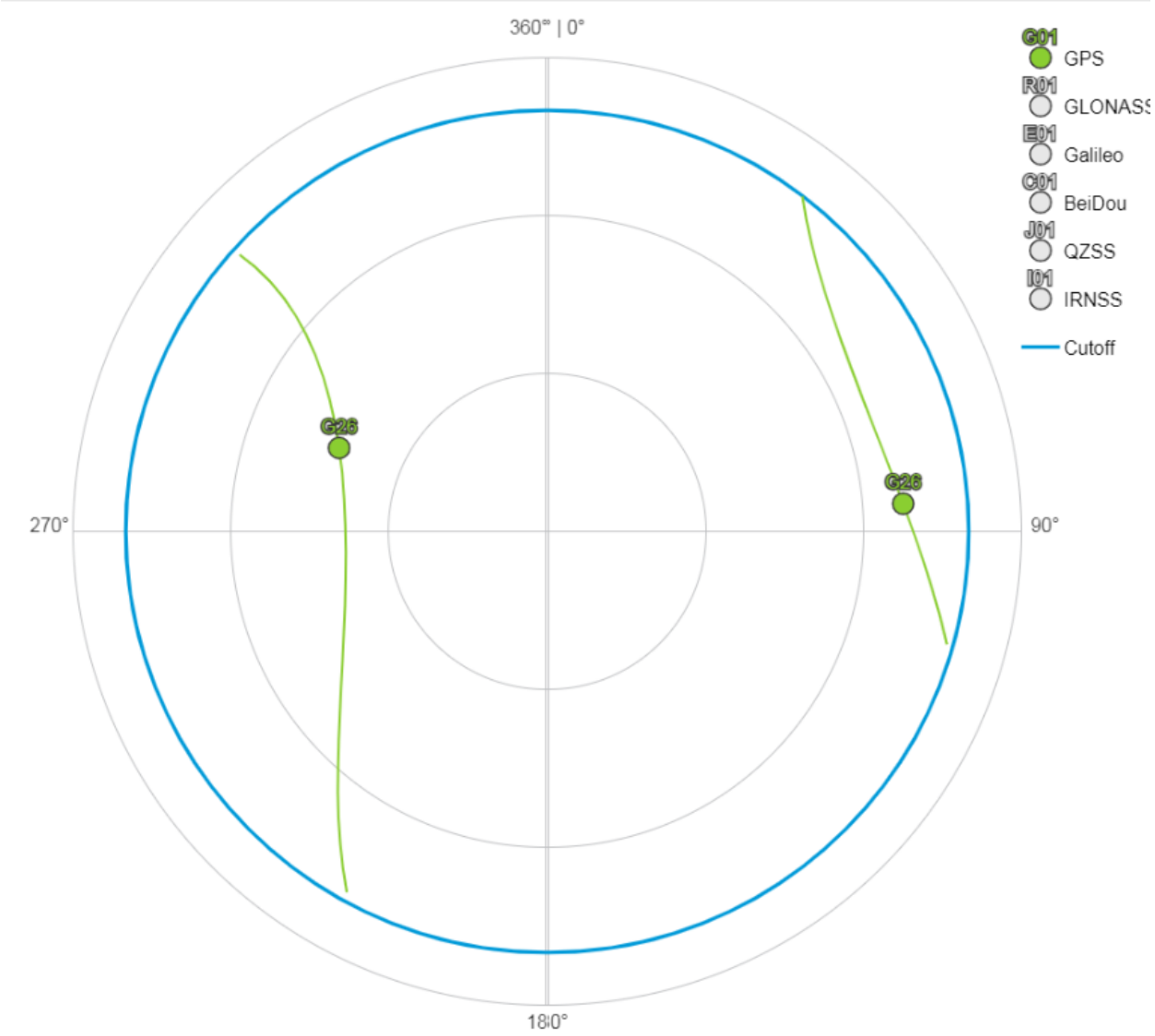

Рисунок 9 – SkyView из Trimble GNSS Planning Online

На данном этапе курсового проекта была разработана программа, выполняющая расчет положения спутника GPS, выводящая в файл out.txt значения координат спутника в заданном формате. При сравнении графиков на рисунках 5 и 7 видно сходство рассчитанного SkyView и полученногос помощью Trimble GNSS Planning Online.

ЭТАП 3. РЕАЛИЗАЦИЯ МОДУЛЯ РАСЧЕТА КООРДИНАТ

Требуется разработать на языке C/C++ функцию расчета положения спутника GPS на заданное время по шкале UTC, минимизировать время её исполнения и количество затрачиваемой оперативной памяти. Вызов функции не должен приводить к выбросу исключений или утечкам памяти при любом наборе входных данных.

Функции расчета положения спутника в Matlab относительно проста, т.к. доступны библиотеки линейной алгебры и решения уравнений. Но при разработке встраиваемого ПО приходится сохранять лицензионную частоту, минимизировать вычислительную нагрузку и затраты памяти. Требуется выполнить свою реализацию решения трансцендентного уравнения.

Программный модуль должен сопровождаться unit-тестами под check:

– Тесты функций решения уравнения Кеплера.
– Тест расчетного положения спутника в сравнении с Matlab с шагом 0.1 секунды.

Во время второго теста должно вычисляться и выводиться средняя длительность исполнения функции. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал.

Требуется провести проверку на утечки памяти.

Программа была реализована на языке программирования C/C++. Расчёт местоположения КА осуществлялся при помощи уравнения Кеплера. Алгоритм расчета местоположения спутника включает в себя уравнение Кеплера. Уравнение Кеплера решается в виде трансцендентного уравнения. Ниже приведен алгоритм решения уравнения Кеплера:

```
// Решение уравнения Кеплера
        double E = M;
        double E_old = M + 1;
        double epsilon = 1e-6;

        while (abs(E - E_old) > epsilon) {
            E_old = E;
```

```
            E = M + e * sin(E);
        }
```

Необходимо реализовать передачу и вывод чисел с высокой точностью. Ниже представлена максимальная разница в координатах и время выполнения разработанной программы.

```
Максимальная разница координат (м) = 5.85686e-07
Время выполнения расчета (с) = 8
```

Рисунок 10 — Результат работы программы

Средний объем потребляемой памяти составляет примерно 4.7 МБ.

Рисунок 11 — Анализ объема потребляемой памяти

Вывод:

На третьем этапе курсового проекта была написана программа на языке C++, что целесообразно при разработке встраиваемого ПО. Было определено, что реализация алгоритмов в Matlab и в C++ и не приводит к серьезному расхождению в результатах. Объем потребляемой памяти составляет примерно 4.7 МБ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017. Структура и правила оформления отчета о научно-исследовательской работе.
2. GPS Interface Specification IS-GPS-200, Revision M - May 2021 Interface Control Contractor: SAIC (GPS SEI) 200 M. Pacific Coast Highway, Suite 1800 El Segundo, CA 90245.
3. Trimble GNSS Planning Online. (Дата обращения: 02.06.2022г.).

ПРИЛОЖЕНИЕ К ЭТАПУ 1

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#define _USE_MATH_DEFINES
#include <cmath>
#include <iostream>


using namespace std;
struct Ephemeris {
   double   Crs;
   double   Dn;
   double   M0;
   double   Cuc;
   double   e;
   double   Cus;
   double   sqrtA;
   uint32_t toe;
   double   Cic;
   double   Omega0;
   double   Cis;
   double   i0;
   double   Crc;
   double   omega;
   double   OmegaDot;
   double   iDot;
   double   Tgd;
   uint32_t toc;
   double   af2;
   double   af1;
   double   af0;
   uint32_t WN;
   uint16_t IODC;
   uint8_t  URA;
   uint8_t  Health;
   uint16_t IODE2;
   uint16_t IODE3;
   bool     codeL2;
   bool     L2P;
   uint32_t slot;
};
const int32_t subFrameLength = 300;
struct SF1_3 {
   uint32_t slot;
   char sf1[subFrameLength+1];
   char sf2[subFrameLength+1];
   char sf3[subFrameLength+1];
};
void printEmp(Ephemeris* ep);
int32_t file2subFrames(SF1_3* sf, FILE* fid, uint8_t svNum);
int32_t subFrames2Eph(Ephemeris* ep, SF1_3* subframes);


int main(void)
{
   printf(" Hello, World \n");
   uint8 t svNum = 26;
   FILE* fid = fopen("in.txt", "r");
```

```
   if (fid != nullptr) {
      SF1_3 subframes;
      if (!file2subFrames(&subframes, fid, svNum)) {

         Ephemeris *ep = (Ephemeris*) calloc(1, sizeof(Ephemeris));
         if (!subFrames2Eph(ep, &subframes)) {
         printEmp(ep);

         } else {
            printf(" Cannot decode subframes\n ");
         }
         free(ep);
         fclose(fid);
      }
      else {
         printf(" Subframes not found\n ");
      }
   }
   else {
      printf(" Cannot open in.txt ");
   }


   return 0;
}
int64_t str2uint(char *sf, int32_t start, int32_t stop) {
   int64_t ans = 0;
   for(int i = start; i < stop; i++) {
      bool bit = (sf[i-1] == '1');
      ans = ans | (bit << (stop - i - 1));

   }
   return ans;




}
int64_t str3int(uint64_t ans, int count_bit) {

    int64_t Ians = 0;
    if (count_bit == 8) {
        if (bool((1<<7) & ans)){
            ans |= 0xFFFFFFFFFFFFFF00;
            Ians = ~(ans - 1);
            return -Ians;

        }
    }
    if (count_bit == 14) {
        if (bool((1<<13) & ans)) {
            ans |= 0xFFFFFFFFFFFFC000;
            Ians = ~(ans - 1);
            return -Ians;
        }
    }
    if (count_bit == 16) {
        if (bool((1 << 15) & ans)) {
            ans |= 0xFFFFFFFFFFFF0000;
            Ians = ~(ans - 1);
            return -Ians;
        }
    }
```

```
    if (count_bit == 22) {
        if (bool((1 << 21) & ans)) {
            ans |= 0xFFFFFFFFFC00000;
            Ians = ~(ans - 1);
            return -Ians;
        }
    }
    if (count_bit == 24) {
        if (bool((1 << 23) & ans)) {
            ans |= 0xFFFFFFFFF000000;
            Ians = ~(ans - 1);
            return -Ians;
        }
    }
    if (count_bit == 32) {
        if (bool((1 << 31) & ans)) {
            ans |= 0xFFFFFFFF00000000;
            Ians = ~(ans - 1);
            return -Ians;
        }
    }
    return ans;
}
int64_t str4uint(char *sf, int32_t start, int32_t stop, int32_t start2,
int32_t stop2){
    uint32_t ans = 0;
    for(int i = start; i < stop; i++) {
        ans = (ans | ((sf[i-1] == '1')? 1 : 0)) << 1;
    }
    for(int i = start2; i < stop2-1; i++) {

        ans = ans | ((sf[i-1] == '1')? 1 : 0);
        if (i < stop2-1) {
            ans = ans << 1;
        }

    }
    return ans;
}

int32_t subFrames2Eph(Ephemeris* ep, SF1_3* subframes) {
   ep->slot = subframes->slot;
   ep->WN = str2uint(subframes->sf1, 61, 71);
   ep->URA = str2uint(subframes->sf1, 73, 77);
   ep->toe = str2uint(subframes->sf2, 271, 287)*pow(2,4);
   ep->Health = str2uint(subframes->sf1, 73, 73+6);
   ep->IODE2 = str2uint(subframes->sf2, 61, 69);
   ep->IODE3 = str2uint(subframes->sf3, 271, 271+8);
   ep->codeL2 = str2uint(subframes->sf1, 71, 73);
    //ep->L2P = subframes->sf1[90];
   ep->L2P = str2uint(subframes->sf1, 90, 91);
    ep->Crc = str3int(str2uint(subframes->sf3,181,181+16),16)*pow(2, -5);
    ep->Dn = str3int(str2uint(subframes->sf2, 91, 91+16), 16)*pow(2, -43);
    ep->Cuc = str3int(str2uint(subframes->sf2,151,151+16),16)*pow(2, -29);
    ep->Cus = str3int(str2uint(subframes->sf2,211,211+16),16)*pow(2, -29);
    ep->e = str4uint(subframes->sf2,167, 167+8, 181, 181+24) * pow(2, -33);
    ep->sqrtA = str4uint(subframes->sf2,227, 227+8, 241, 241+24) * pow(2, -
19);
    ep->Cic = str3int(str2uint(subframes->sf3,61,61+16),16)*pow(2, -29);
    ep->Omega0 = str3int(str4uint(subframes->sf3,77, 77+8, 91,
91+24),32)*pow(2, -31)*180;
    ep->Cis = str3int(str2uint(subframes->sf3,121,121+16),16)*pow(2, -29);
    ep->i0 = str3int(str4uint(subframes->sf3,137, 137+8, 151,
```

```
151+24),32)*pow(2, -31)*180;
    ep->omega = str3int(str4uint(subframes->sf3,197, 197+8, 211,
211+24),32)*pow(2, -31)*180;
    ep->OmegaDot = str3int(str2uint(subframes->sf3,241,241+24),24)*pow(2, -
43)*180;
    ep->iDot = str3int(str2uint(subframes->sf3,279,279+14),14)*pow(2, -
43)*180;
    ep->Tgd = str3int(str2uint(subframes->sf1,197,197+8),8)*pow(2, -31);
    ep->toc = str3int(str2uint(subframes->sf1,219,219+16),16)*pow(2, 4);
    ep->af2 = str3int(str2uint(subframes->sf1,241,241+8),8)*pow(2, -55);
    ep->af1 = str3int(str2uint(subframes->sf1,249,249+16),16)*pow(2, -43);
    ep->af0 = str3int(str2uint(subframes->sf1,271,271+22),22)*pow(2, -31);
    ep->IODC = str4uint(subframes->sf1,83, 83+2, 211, 211+8);
    ep->Crs = str3int(str2uint(subframes->sf2,69,69+16),16)*pow(2, -5);
    ep->M0 = str3int(str4uint(subframes->sf2,107, 107+8, 121,
121+24),32)*pow(2, -31)*180;



    return 0;




}
int32_t file2subFrames(SF1_3* sf, FILE* fid, uint8_t svNum){
   int32_t sth1, sth2, sth3, sth4, sth5;
   char str_0R[8];
   char str_GPSL1CA[12];
   char str_reh[8];
   char str[1000];
   uint32_t svStr;
   uint32_t slot;
   int32_t subFrameNum;

   uint32_t slot_SF1 = 0;
   uint32_t slot_SF2 = 0;
   uint32_t slot_SF3 = 0;
   int32_t readres = 0;

   while(readres != EOF)
      {
      svStr = 0;
      readres = fscanf( fid, "%d %d %d %s %s %s %u\t %u %d %d %d %s", &sth1,
&sth2, &sth3, str_0R, str_GPSL1CA, str_reh, &svStr, &slot, &sth4, &sth5,
&subFrameNum, str);
      if (( svStr == svNum ) && (slot >= (604800/6))) {
         if ( subFrameNum == 1 ) {
            slot_SF1 = slot;
            strncpy(sf->sf1, str, sizeof(sf->sf1));
         }
         else if (subFrameNum == 2) {
            slot_SF2 = slot;
            strncpy(sf->sf2, str, sizeof(sf->sf2));
         }
         else if (subFrameNum == 3) {
            slot_SF3 = slot;
            strncpy(sf->sf3, str, sizeof(sf->sf3));
         }
         if ((slot_SF1 + 1 == slot_SF2) && (slot_SF2 + 1 == slot_SF3)) {
            sf->slot = slot_SF1;
            return 0;
```

```
          }
       }
    }
    return 1;
}
void printEmp(Ephemeris* ep)
{
    printf("LNAV Ephemeris (slot = %u) =  \n",  ep->slot      );
    printf("\tCrs      = %e                \n",  ep->Crs       );
    printf("\tDn       = %e \t[deg/s]      \n",  ep->Dn        );
    printf("\tM0       = %e \t[deg]        \n",  ep->M0        );
    printf("\tCuc      = %e                \n",  ep->Cuc       );
    printf("\te        = %e                \n",  ep->e         );
    printf("\tCus      = %e                \n",  ep->Cus       );
    printf("\tsqrtA    = %e                \n",  ep->sqrtA     );
    printf("\ttoe      = %u                \n",  ep->toe       );
    printf("\tCic      = %e                \n",  ep->Cic       );
    printf("\tOmega0   = %e \t[deg]        \n",  ep->Omega0    );
    printf("\tCis      = %e                \n",  ep->Cis       );
    printf("\ti0       = %e \t[deg]        \n",  ep->i0        );
    printf("\tCrc      = %e                \n",  ep->Crc       );
    printf("\tomega    = %e \t[deg]        \n",  ep->omega     );
    printf("\tOmegaDot= %e \t[deg/s]       \n",  ep->OmegaDot  );
    printf("\tiDot     = %e \t[deg/s]      \n",  ep->iDot      );
    printf("\tTgd      = %e                \n",  ep->Tgd       );
    printf("\ttoc      = %u                \n",  ep->toc       );
    printf("\taf2      = %e                \n",  ep->af2       );
    printf("\taf1      = %e                \n",  ep->af1       );
    printf("\taf0      = %e                \n",  ep->af0       );
    printf("\tWN       = %u                \n",  ep->WN        );
    printf("\tIODC     = %u                \n",  ep->IODC      );
    printf("\tURA      = %u                \n",  ep->URA       );
    printf("\tHealth   = %u                \n",  ep->Health    );
    printf("\tIODE2    = %u                \n",  ep->IODE2     );
    printf("\tIODE3    = %u                \n",  ep->IODE3     );
    printf("\tcodeL2   = %u                \n",  ep->codeL2    );
    printf("\tL2P      = %u                \n",  ep->L2P       );

}
```

ПРИЛОЖЕНИЕ К ЭТАПУ 2

```
clc
clear all
close all
% Эфемериды
C_rs = 2.095312e+02; % Амплитуда поправочного члена синусоидальной гармоники
к радиусу орбиты
delta_n = deg2rad(1.606622e-9); % Среднее отклонение движения от вычисленного
значения
M_0 = deg2rad(-1.300067e+01); % Средняя аномалия
C_uc = -1.061708e-07; % Амплитуда поправочного члена косинусной гармоники к
аргументу широты
eks = 6.875264e-03; % Эксцентриситет
C_us = 8.167699e-06; % Амплитуда поправочного члена синусоидальной гармоники
к аргументу широты
rootA = 5.153699e+03; % Корень из большой полуоси
t_oe = 93600; % Опорное время эфемерид
C_ic = 8.009374e-08; % Амплитуда поправочного члена косинусной гармоники к
углу наклона
Omega_0 = deg2rad(1.268903e+02); % Долгота узла
C_is = 6.705523e-08; % Амплитуда поправочного члена синусоидальной гармоники
к углу наклона
i_0 = deg2rad(5.377037e+01); % Наклонение
C_rc = 2.095312e+02; % Амплитуда поправочного члена косинусной гармоники к
радиусу орбиты
omega = deg2rad(2.118201e+01); % Аргумент перигея
Omega_dot = deg2rad(-4.743470e-07); % Скорость прямого восхождения
IDOT = deg2rad(-2.506795e-08); % Скорость угла наклона
% Константы из ИКД
pi = 3.1415326535898;
mu = 3.986005e+14; % Гравитационная постоянная
Omega_e_dot = 7.2921151467e-5; % Скорость вращения Земли
% Расчет значений элементов Кеплера
A = rootA^2; % Большая полуось
n_0 = sqrt(mu/A^3); % Расчетное среднее движение
for i = 1:86400 % Суточный интервал рассмотрения
t = 86400+18 + i; % Количество секунд от начала текущей недели при начале
рассмотрения в 00:00:01
% Время от опорной эпохи эфемерид
t_k = t - t_oe;
if t_k>302400
t_k = t_k - 604800;
elseif t_k<-302400
t_k = t_k + 604800;
end
n = n_0 + delta_n; % Скорректированное среднее движение
M_k = M_0 + n * t_k; % Средняя аномалия
% Эксцентрическая аномалия
E_0 = M_k;
E_k = 0;
k = 0;
while abs(E_0 - E_k)>1e-8
E_k = E_0;
E_0 = E_0 + (M_k - E_0 + eks * sin(E_0))/(1 - eks * cos(E_0));
k = k + 1;
end
nu_k = atan2((sqrt(1 - eks^2) * sin(E_k)/(1 - eks * cos(E_k))), ((cos(E_k) -
eks)/(1 - eks * cos(E_k)))); % Истинная аномалия
Phi_k = nu_k + omega; % Аргумент широты
delta_u_k = C_us*sin(2*Phi_k) + C_uc*cos(2*Phi_k); % Аргумент поправки на
широту
```

```
delta_r_k = C_rs*sin(2*Phi_k) + C_rc*cos(2*Phi_k); % Коррекция радиуса
delta_i_k = C_is*sin(2*Phi_k) + C_ic*cos(2*Phi_k); % Коррекция наклона
u_k = Phi_k + delta_u_k; % Скорректированный аргумент широты
r_k = A * (1 - eks * cos(E_k)) + delta_r_k; % Скорректированный радиус
i_k = i_0 + delta_i_k + IDOT * t_k; % Скорректированное наклонение
% Позиция в орбитальной плоскости
x_k_sh = r_k * cos(u_k);
y_k_sh = r_k * sin(u_k);
Omega_k = Omega_0 + (Omega_dot - Omega_e_dot) * t_k - Omega_e_dot * t_oe;
%Скорректированная широта восходящего узла
% Расчет координат спутника
x_k = x_k_sh * cos(Omega_k) - y_k_sh * cos(i_k) * sin(Omega_k);
y_k = x_k_sh * sin(Omega_k) + y_k_sh * cos(i_k) * cos(Omega_k);
z_k = y_k_sh * sin(i_k);
x_current(1,i) = x_k;
y_current(1,i) = y_k;
z_current(1,i) = z_k;
% Инерциальная система координат
theta = Omega_e_dot * t_k;
x_k_1 = x_k * cos(theta) - y_k * sin(theta);
y_k_1 = x_k * sin(theta) + y_k * cos(theta);
z_k_1 = z_k;
x_converted(1,i) = x_k_1;
y_converted(1,i) = y_k_1;
z_converted(1,i) = z_k_1;
% Данные из RTKNAVI (координаты приемника)
B = deg2rad(44.09363261); % Широта
L = deg2rad(39.00130546); % Долгота
H = 1.247; % Высота
[x(i), y(i), z(i)] =
ecef2enu(x_current(1,i),y_current(1,i),z_current(1,i),B,L,H,wgs84Ellipsoid,'r
adians'); % Отображение координат спутника относительно точки приема
% Полярная система координат
if z(i) > 0
rho(i) = sqrt(x(i)^2 + y(i)^2 + z(i)^2);
theta_1(i) = acos(z(1,i)/rho(i));
if x(i) > 0
phi(i) = -atan(y(i)/x(i))+pi/2;
elseif (x(i)<0)&&(y(i)>0)
phi(i) = -atan(y(i)/x(i))+3*pi/2;
elseif (x(i)<0)&&(y(i)<0)
phi(i) = -atan(y(i)/x(i))-pi/2;
end
else theta_1(i) = NaN;
 rho(i) = NaN;
 phi(i) = NaN;
 end
end
% Вывод значений координат
out = fopen('../out.txt', 'w+');
for i = 1:length(x_current)
 fprintf(out, '%6.0f %10.10f %10.10f %10.10f\n', i, x_current(1,i),
y_current(1,i), z_current(1,i));
end
fclose(out);
% Вывод графиков
R = 6378136; % Радиус Земли
[x_sphere, y_sphere, z_sphere] = sphere(100);
x_Earth=R*x_sphere;
y_Earth=R*y_sphere;
z_Earth=R*z_sphere;
figure (1)
subplot (1,1,1)
```

```
surf(x_Earth,y_Earth,z_Earth);
hold on;
plot3(x_current(1,:), y_current(1,:), z_current(1,:));
xlabel('x, м','FontSize',16);
ylabel('y, м','FontSize',16);
zlabel('z, м','FontSize',16);
figure (2)
subplot (1,1,1)
surf(x_Earth,y_Earth,z_Earth);
hold on;
plot3(x_converted(1,:), y_converted(1,:), z_converted(1,:));
xlabel('x, м','FontSize',16);
ylabel('y, м','FontSize',16);
zlabel('z, м','FontSize',16);
figure (3)
axes = polaraxes;
polarplot(axes,phi,rad2deg(theta_1))
axes.ThetaDir = 'clockwise';
axes.ThetaZeroLocation = 'top';
rlim([0 80]);
```

ПРИЛОЖЕНИЕ К ЭТАПУ 3

```
#include <iostream>
#include <fstream>
#include <cmath>


using namespace std;

int main()
{

    setlocale(LC_ALL, "rus");
    std::cout << "Начало расчета!\n";
    time t t start, t stop;
    time(&t_start);

    double SatNum = 1;
    double toe = 93600;
    double Crs = -4.031250e+00;
    double Dn = 1.606622e-9;
    double M0 = -1.300067e+01;
    double Cuc = -1.061708e-07;
    double e = 6.875264e-03;
    double Cus = 8.167699e-06;
    double sqrtA = 5.153699e+03;
    double Cic = 8.009374e-08;
    double Omega0 = 1.268903e+02;
    double Cis = 6.705523e-08;
    double i0 = 5.377037e+01;
    double Crc = 2.095312+02;
    double omega = 2.118201e+01;
    double OmegaDot = -4.743470e-07;
    double iDot = -2.506795e-09;
    double Tgd = 6.984919e-09;
    double toc = 93600;
    double af2 = 0.000000e+00;
    double af1 = 3.183231e-12;
    double af0 = 1.830263e-04;
    double URA = 0;
    double IODE = 9;
    double IODC = 8;
    double codeL2 = 1;
    double L2P = 0;
    double WN = 149;

    // Значения констант
    double mu = 3.986004418e14; // гравитационная постоянная
    double omega_e = 7.2921151467e-5;// скорость вращения

    // Временной промежуток
    int begin_time = 24 * 2  * 60 * 60;// время начала 0:00 по МСК 14 февраля
    int end_time = 24 * 3  * 60 * 60;// время окончания 0:00 по МСК 15
февраля

    //Длина временного промежутка
    int t_arr = end_time - begin_time;

    //Большая полуось
    double A = pow(sqrtA,2);

    // Среднее движение
    double n0 = sqrt(mu /pow(A,3));
```

```
    double n = n0 + Dn;

    // Координаты
    double** coord = new double*[3];
    for (int i = 0; i < 3; i++) {
        coord[i] = new double[t_arr];
    }

    // Координаты из матлаба
    double** coordMat = new double*[3];
    for (int i = 0; i < 3; i++) {
        coordMat[i] = new double[t_arr];
    }

    for (int k = 0; k < t_arr; k++) {
        // Время
        double t = begin_time + k - (int)toe;

        if (t > 302400)
            t = t - 604800;
        if (t < -302400)
            t = t + 604800;

        // Средняя аномалия
        double M = M0 + n * t;

        // Решение уравнения Кеплера
        double E = M;
        double E_old = M + 1;
        double epsilon = 1e-6;

        while (abs(E - E_old) > epsilon) {
            E_old = E;
            E = M + e * sin(E);
        }
        // Истинная аномалия
        double nu = atan2(sqrt(1 - e*e) * sin(E), cos(E) - e);

        // Коэффициент коррекции
        double cos_correction = cos(2 * (omega + nu));
        double sin_correction = sin(2 * (omega + nu));

        // Аргумент широты
        double u = omega + nu + Cuc * cos_correction + Cus * sin_correction;

        // Радиус
        double r = A * (1 - e * cos(E)) + Crc * cos_correction + Crs *
sin_correction;

        // Наклон
        double i = i0 + iDot * t + Cic * cos_correction + Cis *
sin_correction;

        // Долгота восходящего угла
        double lambda = Omega0 + (OmegaDot - omega_e) * t - omega_e * toe;

        // Положение на орбите
        double x = r * cos(u);
        double y = r * sin(u);

        // Координаты
        double Xk = x * cos(lambda) - y * cos(i) * sin(lambda);
        double Yk = x * sin(lambda) + y * cos(i) * cos(lambda);
```

```
        double Zk = y * sin(i);

        coord[0][k] = Xk;
        coord[1][k] = Yk;
        coord[2][k] = Zk;
    }

    // Чтение координат из матлаба
    ifstream file("data_matlab.txt");

    if (!file.is_open())
        cout << "Файл не может быть открыт!" << endl;
    else {
        for (int k = 0; k < t_arr; k++) {
            file >> coordMat[0][k] >> coordMat[1][k] >> coordMat[2][k];
        }
        file.close();
    }


    // Сравнение с матлабом
    double max_del = 0;
    for (int i = 0; i < 3; i++) {
        for (int k = 0; k < t_arr; k++) {
            if (abs(coord[0][k] - coordMat[0][k]) > max_del) {
                max_del = abs(coord[0][k] - coordMat[0][k]);
            }
        }
    }

// Очищение памяти
    delete[] *coord;
    delete[] coord;
    delete[] * coordMat;
    delete[] coordMat;

    time(&t_stop);

    double del_time = difftime(t_stop, t_start);

    cout << "Максимальная разница координат (м) = " << max_del << endl;
    cout << "Время выполнения расчета (с) = " << del_time << endl;
}
```