Национальный исследовательский университет
«МЭИ»
Институт радиотехники и электроники им. В.А. Котельникова
Кафедра радиотехнических систем
Аппаратура потребителей СРНС

Курсовая работа

«Расчет траектории движения спутника GPS по данным с демодулятора его сигнала»

Группа: ЭР-15-17
ФИО студента: Цымбал Г.Р.
ФИО преподавателя: Корогодин И.В.

Оценка: _____
Дата: _____
Подпись: _____

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.

Конечная цель всего курсового проекта – получить библиотеку функций на Си++, позволяющую рассчитывать положение спутника GPS по данным с демодулятора его сигнала L1 C/A.

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

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.

Работа содержит 29 страниц, 12 рисунков, 4 источника, 2 приложения.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

GPS	Global Positioning System
КА	Космический аппарат
НАП	Навигационная аппаратура потребителя
ГНСС	Глобальная навигационная спутниковая система
НКА	Навигационный космический аппарат
ИКД	Интерфейсный контрольный документ
СК	Система координат

ВВЕДЕНИЕ

Космический сегмент, состоящий из навигационных спутников, представляет собой совокупность источников радионавигационных сигналов, передающих одновременно значительный объем служебной информации. Основные функции каждого спутника — формирование и излучение радиосигналов, необходимых для навигационных определений потребителей и контроля бортовых систем спутника.

Современная спутниковая навигация основывается на использовании принципа беззапросных дальномерных измерений между навигационными спутниками и потребителем. Это означает, что потребителю в составе навигационного сигнала передается информация о координатах спутников. Одновременно (синхронно) производятся измерения дальностей до навигационных спутников. Способ измерений дальностей основывается на вычислении временных задержек принимаемого сигнала от спутника по сравнению с сигналом, генерируемым НАП. Для функционирования ГНСС необходимы данные о параметрах вращения Земли, фундаментальные эфемериды Луны и планет, данные о гравитационном поле Земли, о моделях атмосферы, а также высокоточные данные об используемых системах координат и времени.

Каждый спутник принимает с наземных станций управления навигационную информацию, которая передается обратно пользователям в составе навигационного сообщения. Навигационное сообщение содержит разные типы информации, необходимые для того, чтобы определить местоположение пользователя и синхронизовать его шкалу времени с национальным эталоном. Выделяются основные типы информации навигационного сообщения:

- Эфемеридная информация, необходимая для вычисления координат спутника с достаточной точностью;

- Погрешность расхождения бортовой шкалы времени относительно системной шкалы времени для учета смещения времени КА при навигационных измерениях;
- Расхождение между шкалой времени навигационной системы и национальной шкалой времени, для решения задачи синхронизации потребителей;
- Признаки пригодности с информацией о состоянии спутника для оперативного исключения спутников с выявленными отказами из навигационного решения;
- Альманах с информацией об орбитах и состоянии всех КА в группировке для долгосрочного грубого прогноза движения спутников и планирования измерений;
- Параметры модели ионосферы, необходимые одночастотным приемникам для компенсации погрешностей навигационных измерений, связанных с задержкой распространения сигналов в ионосфере;
- Параметры вращения Земли для точного пересчета координат потребителя в разных системах координат.

GPS спутники передают два вида данных - альманах и эфимерис. Альманах содержит параметры орбит всех спутников. Каждый спутник передаёт альманах для всех спутников. Данные альманаха не отличаются большой точностью и действительны несколько месяцев. В свою очередь, данные эфимериса содержат очень точные корректировки параметров орбит и часов для каждого спутника, что требуется для точного определения координат.

Каждый GPS спутник передаёт только данные своего собственного эфимериса. Эти данные действительны только 30 минут. Спутники передают

свой эфимерис каждые 30 секунд. Если GPS был отключён более 30 минут, а потом включён, он начинает искать спутники, основываясь на известном ему альманахе. По нему GPS выбирает спутники для инициации поиска.

В первой главе производится выделение информации об эфемеридах отдельного КА из структуры кадра навигационного сообщения, а также осуществляется обработка полученной информации и запись в текстовый файл.

1 Обработка логов навигационного приемника

1.1 Задание к первому этапу

В неизвестной локации установлен навигационный приемник, принимающий сигналы GPS L1C/A и логирующий результаты этого приема в формате NVS BINR. Собранный на пятиминутном интервале файл приложен в архиве под именем BINR.bin. Файл содержит наблюдения псевдодальностей и прочих радионавигационных параметров, демодулированные и разобранные данные навигационного сообщения.

Данные демодулятора продублированы в текстовый файл in.txt. Каждая строка файла содержит данные одного сабфрейма одного навигационного сигнала в формате:

```
1 0 013 0R GpsL1CA  # 13         212130404  29 125  53
1000101110101010101010100010100101110000110010100101111110001010101010101010
```

Рисунок 1 – Структура одного сабфрейма одного НКА

где 13 - номер спутника, 212130404 - счетчик сабфреймов в сигнале, 53 - ID сабфрейма в навигационном сообщении, где в первых трех битах содержится номер сабфрейма в фрейме (5 в данном примере), а далее - номер фрейма в сообщении (6 в данном примере), 1000101110... символы с демодулятора в порядке возрастания времени слева направо.

Требуется:

1. Разработать программу, обрабатывающую файл in.txt и выводящую в файл out.txt таблицу эфемерид для спутника согласно варианту в заданном формате.

2. Обработать файл BINR.bin с помощью программы RTKNAVI из состава RTKLIB. Определить день и место проведения наблюдений, значения эфемерид для спутника согласно номеру варианта.

3. Сравнить полученные таблицы.

4. Оформить код программы и разместить на Github.
5. Оформить отчет по этапу и разместить на Github.
6. Завести Pull Request.

Программа должна компилироваться gcc, все входные данные брать из in.txt, весь вывод осуществлять в out.txt.

1.2 Разработка программы для обработки исходного файла и вывод в файл таблицы эфемерид

Исходные данные в файле in.txt имеют следующий вид:

Рисунок 2 – Исходные данные файла in.txt

Согласно варианту задания необходимо обработать данные для 16 спутника, а именно: эфемеридная информация, содержащаяся в первых трёх сабфреймах структуры навигационного сообщения. В приложении А представлен листинг кода программы, реализующий обработку файла in.txt на языке С.

Функция *file2subFrames* выделяет данные из первых трёх сабфреймов навигационного сообщения.

Функция *subFrames2Eph* выделяет из сабфреймов необходимую эфемеридную информацию о Кеплеровских элементах орбиты, шкале времени часов спутника.

Метод *twoCompl2int* вводит дополнение до двух, а именно позволяет получать данные из структуры сабфрейма с учётом знака.

Функция *printEmp* выводит на экран таблицу эфемерид для заданного КА.

Сохранение таблицы эфемерид производится с помощью функции *save* в файл out.txt.

Скомпилируем программу с помощью gcc и выведем на экран полученную таблицу эфемерид:

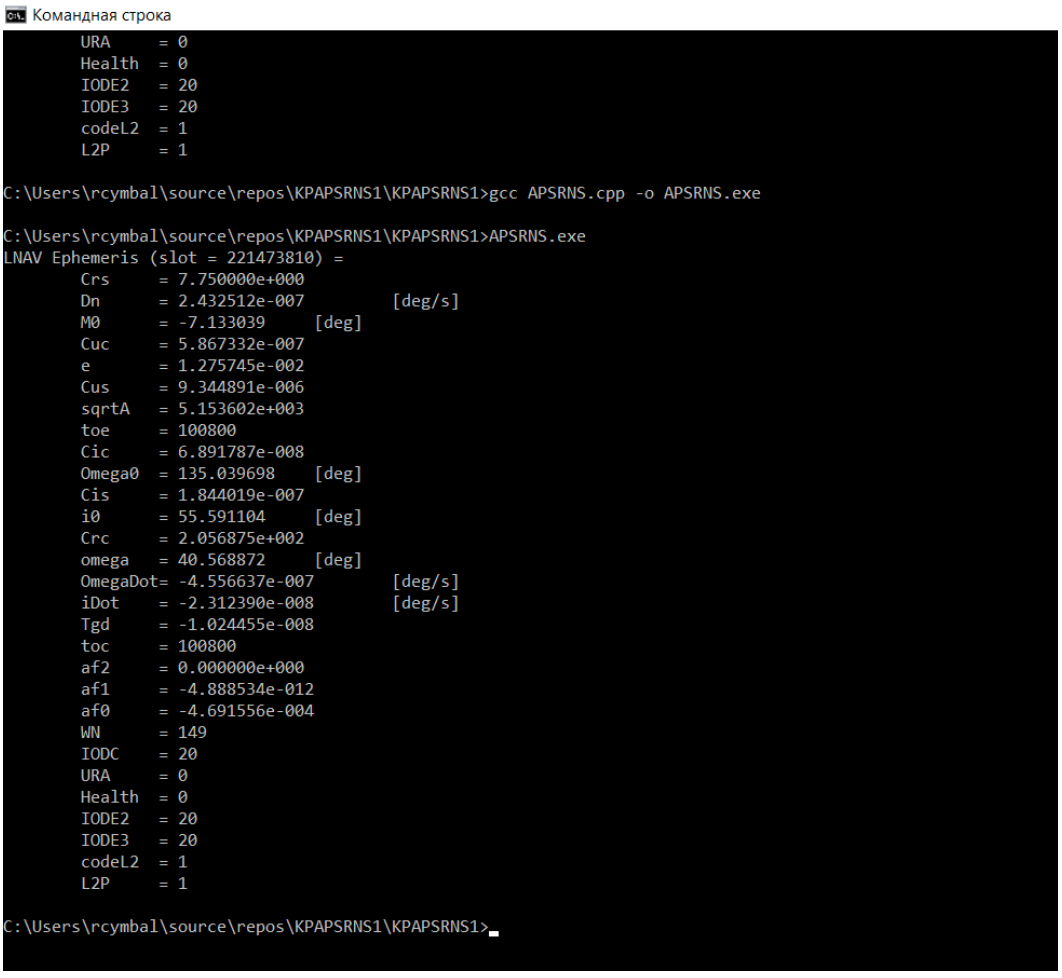

Рисунок 3 – Результат компиляции программы

Запишем в файл out.txt таблицу эфемерид:

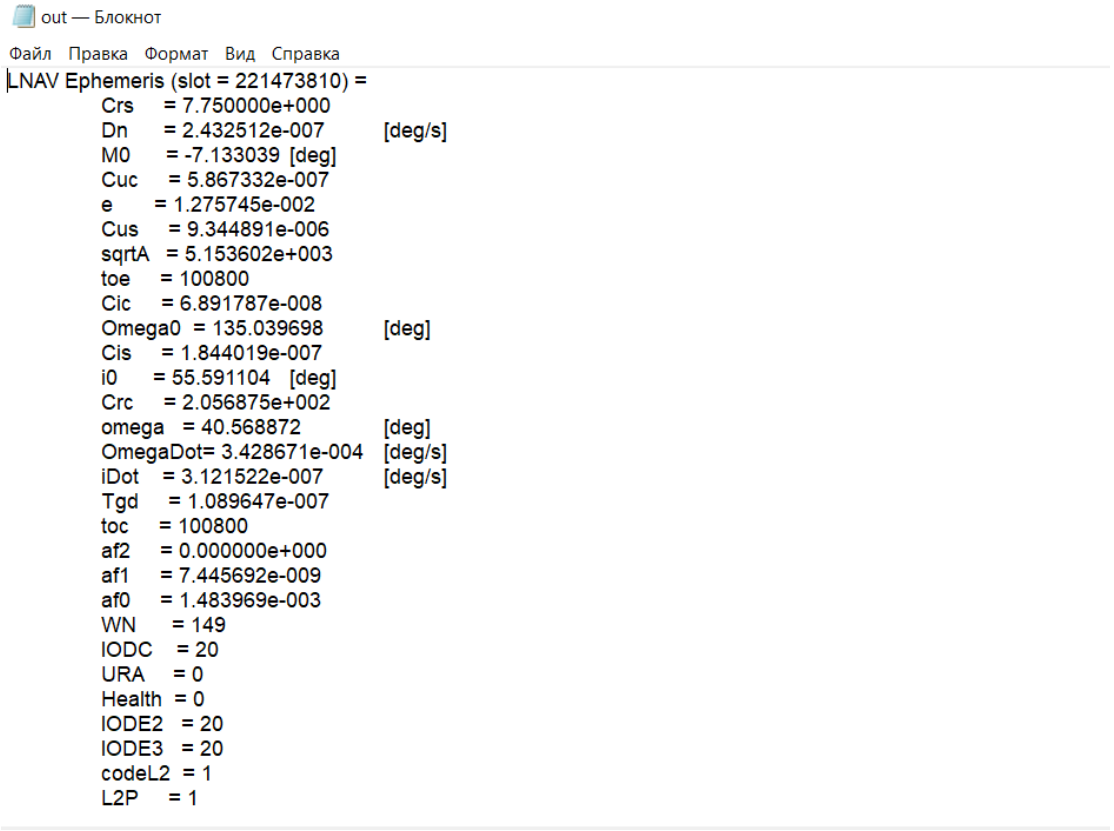

Рисунок 4 – Запись таблицы эфемерид в файл out.txt

1.3 Сравнение результатов разработанной программы и программы RTKNAVI

Эфемеридная информация также была представлена в файле BINR.bin.

После обработки файла BINR.bin с помощью программы RTKNAVI из состава RTKLIB получим таблицу эфемерид и сравним с рисунком 4 п.1.2:

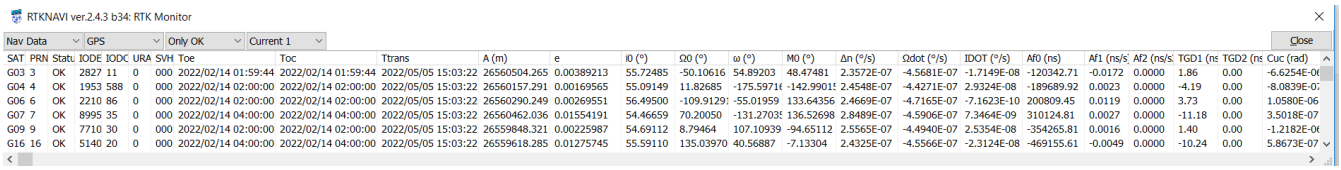

Рисунок 5 – Эфемериды, полученные из программы RTKNAVI

Комментарий к рисункам 4 – 5:

По полученным таблицам можно отметить, что эфемеридная информация была выделена корректно.

2 Моделирование траектории движения

2.1 Задание ко второму этапу

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

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 суточному интервалу на дне формирования наблюдений, определенном на предыдущем этапе. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал.

Вывести значения координат спутника в файл out.txt в системе ECEF WGS 84 в виде строк: Секунда_от_начала_дня X Y Z.

Используя оценку местоположения с предыдущего этапа, построить Sky Plot за указанный временной интервал и сравнить результат с Trimble GNSS Planning Online.

Требуется:

1. Реализовать в Matlab или Python (описание модели и её листинг)
2. Записать таблицу использованных эфемерид
3. Построить трехмерные графики положений спутника в ECEF и ECI (не забыть подписать оси, изобразить соответствующую Земле сферу в начале СК)
4. Построить расчётный и полученный в GNSS Planing Online SkyView

2.2 Разработка программы расчета положения спутника

Из [2] запишем таблицу значений эфемерид из п.1.2 в соответствии с моделью GPS и воспользуемся константами из ИКД GPS.

Таблица 1 – Значения эфемерид в соответствии с моделью GPS

C_{rs}	7.750000e+000
Δn	2.55652e-7
M_0	-7.133039
C_{uc}	5.867332e-007
e	1.275745e-002
C_{us}	9.344891e-006
$\sqrt{A}$	5.153602e+003
t_{oe}	100800
C_{ic}	6.891787e-008
Ω_0	135.039698
C_{is}	1.844019e-007
i_0	55.591104
C_{rc}	2.056875e+002
ω	40.568872
$\dot{\Omega}$	-4.556637e-007
IDOT	-2.312390e-008

Далее значения из табл.1 используются для расчета положения спутника GPS на заданный момент по шкале времени UTC на суточном интервале в системе координат ECEF WGS 84 и соответствующей ей инерциальной СК. Алгоритм расчёта координат КА взят из [2].

Для перевода в инерциальную СК расчёт проводится по (1):

$$\begin{cases} x' = x\cos(\theta) - y\sin(\theta) \\ y' = x\sin(\theta) + y\cos(\theta) \\ z' = z \end{cases} \quad (1)$$

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

Центр декартовой системы координат переносится в точку приёма (рисунок 6) с помощью метода *ecef2enu*. Далее координаты КА относительно точки приёма пересчитываются в полярную систему координат по (2) из алгоритма ИКД.

$$\begin{cases} r = \sqrt{x^2 + y^2 + z^2} \\ cos(\theta) = \dfrac{z}{\sqrt{x^2 + y^2 + z^2}} \\ tg(\varphi) = \dfrac{y}{x} \end{cases} \qquad (2)$$

Координаты приёмника в RTKNAVI:

N:	44° 09' 36.3261"
E:	39° 00' 13.0546"
He:	1.247 m

Рисунок 6 – Координаты приемника

Вывод значений координат производится в файл out.txt. В приложении Б представлен листинг кода, реализующий расчёт положения КА и вывод соответствующих графиков

2.3 Результаты моделирования

На рисунке 7 представлена траектория движения КА №16 на интервале суток в СК ECEF WGS 84 с отмеченным местоположением приёмника. На рисунке 8 изображена траектория движения КА в СК, а также точка, в которой находится приёмник. На рисунке 8 изображён Sky Plot с учётом угла места в 10 град.

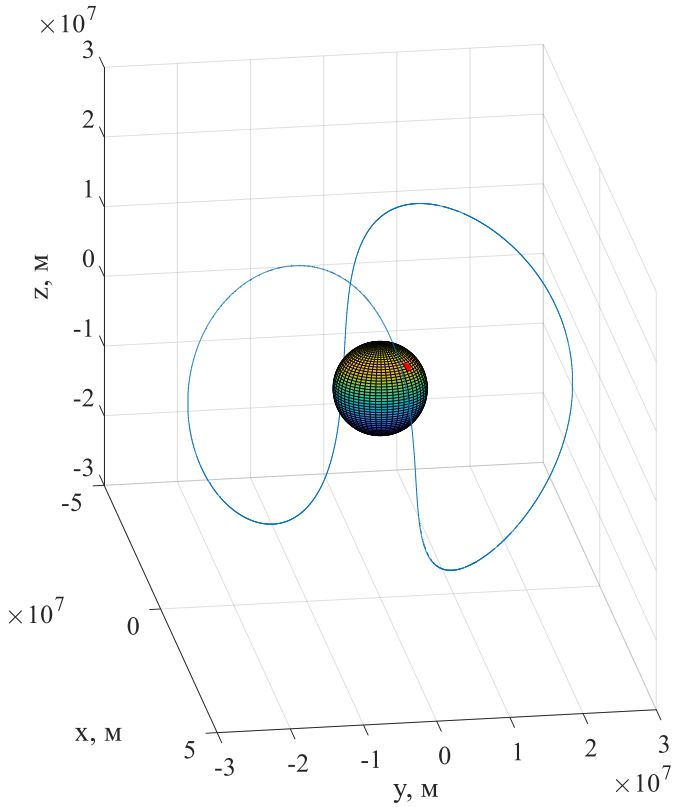

Рисунок 7 – Траектория движения КА в СК ECEF WGS84 и приёмник

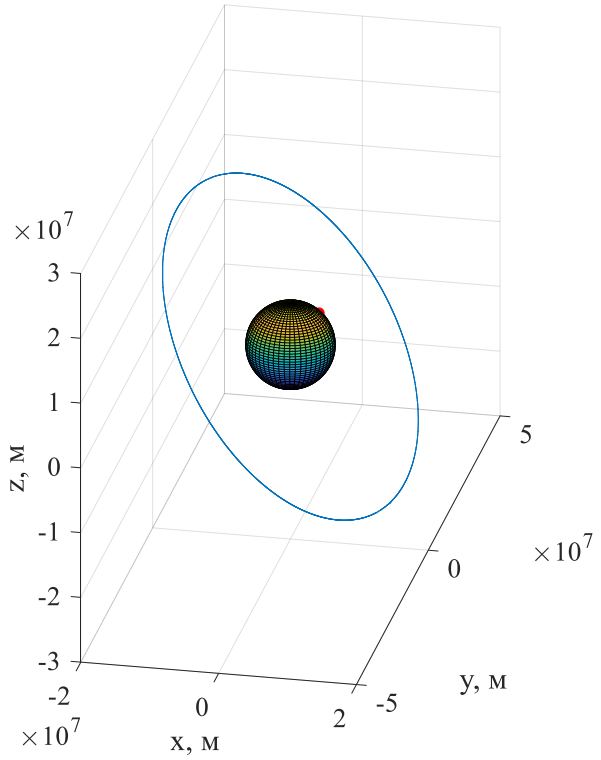

Рисунок 8 – Траектория движения КА в СК ECI и приёмник

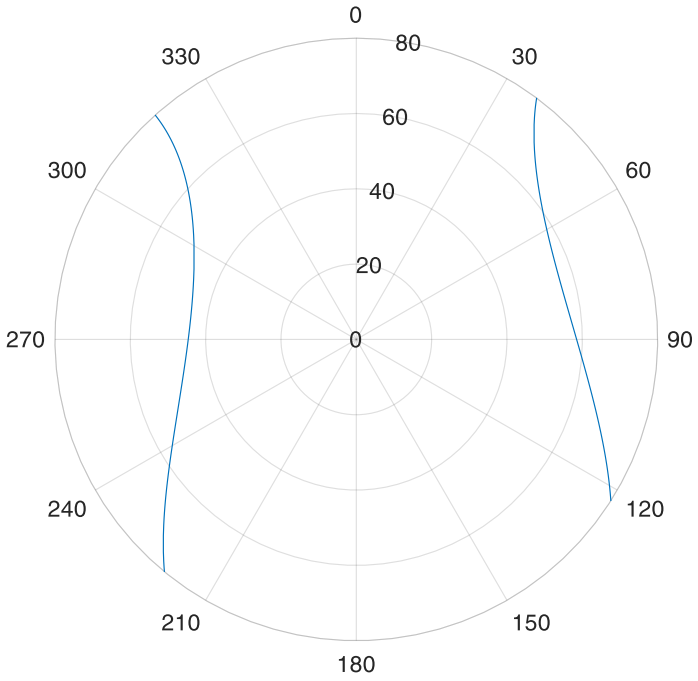

Рисунок 9 – Sky Plot

2.4 Сравнение результатов моделирования с Trimble GNSS Planning Online

Выставим необходимые параметры в Trimble GNSS Planning Online для построения SkyView на заданную дату и время:

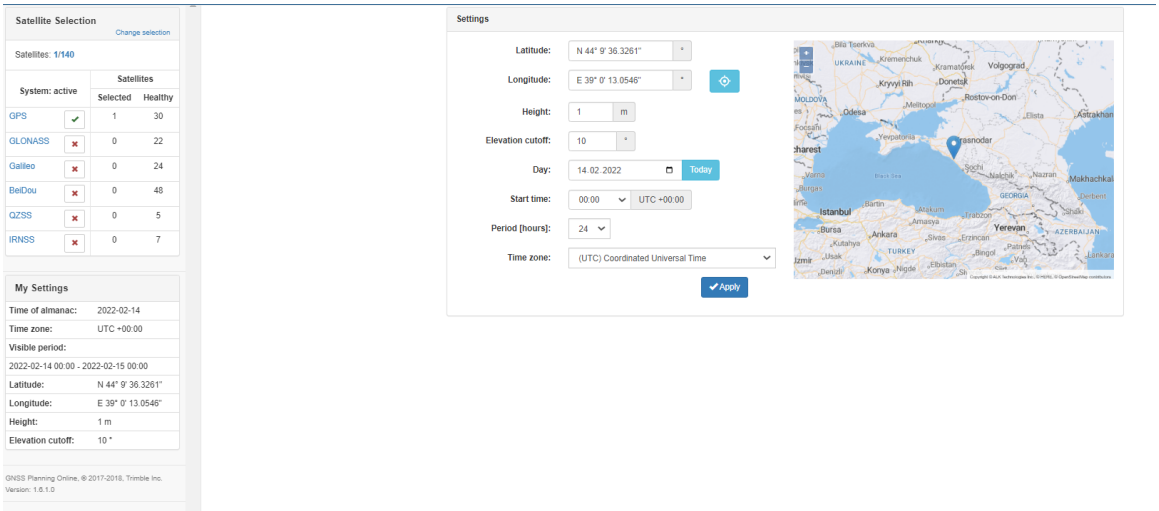

Рисунок 10 — Данные для построения SkyView

На рисунках 11,12 построен SkyView КА №16 системы GPS.

На интервале времени 14.02.2022 00:00 - 15.02.2022 00:00 из точки приёма КА виден 2 раза (утром и вечером).

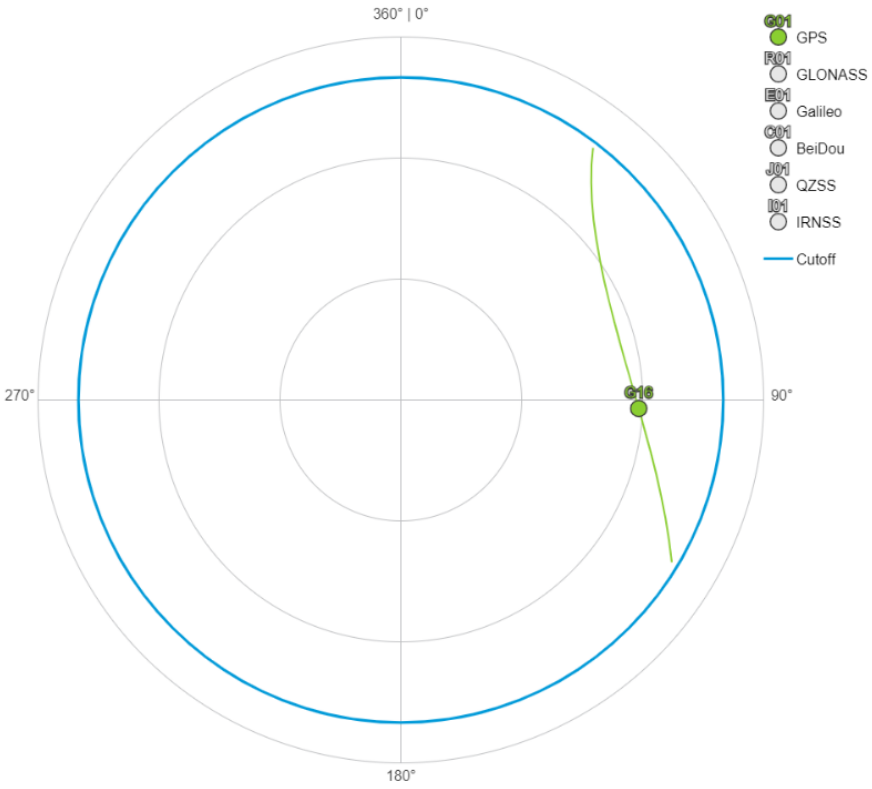

Рисунок 11 – Построение SkyView для первого пролёта КА утром

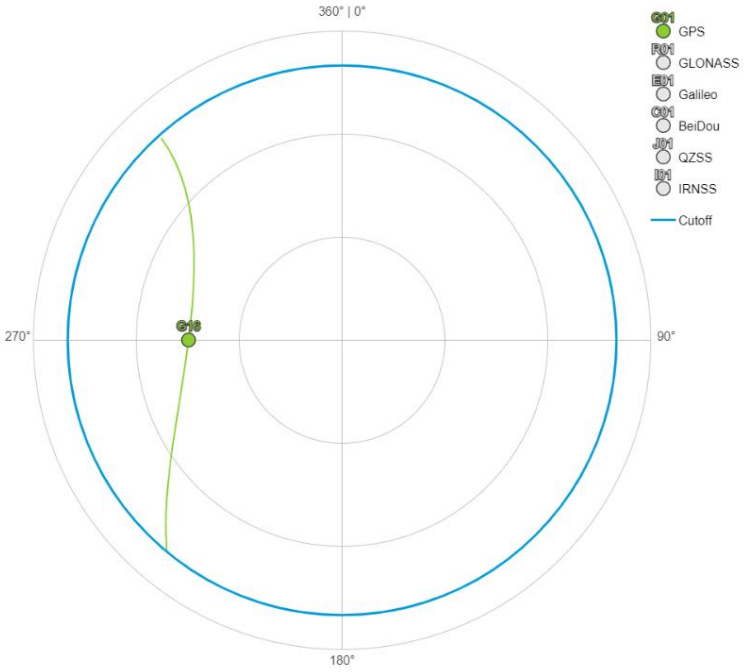

Рисунок 12 – Построение SkyView для второго пролёта КА вечером

Комментарий к рис. 9,11,12:

Рассчитанные положения КА в результате моделирования совпадают с Trimble GNSS Planning Online.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017. Структура и правила оформления отчета о научно-исследовательской работе.
2. Interface Control Contractor: SAIC (GPS SEI) 200 M. Pacific Coast Highway, Suite 1800 El Segundo, CA 90245.
3. https://www.glonass-iac.ru/ Прикладной потребительский центр ГЛОНАСС. Информационно-аналитический центр координатно-временного и навигационного обеспечения.
4. https://docs.microsoft.com/ Техническая документация Майкрософт.

ПРИЛОЖЕНИЕ А

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>  // loading libraries
#include <cmath>

#define sF1       pow(2,-5)
#define sF2       pow(2,-43)
#define sF3       pow(2,-31)
#define sF4       pow(2,-29)
#define sF5       pow(2,-33)
#define sF6       pow(2,-19)
#define sF7       pow(2,4)
#define sF8       pow(2,-55)    //  Scale factor
#define sc    180   //Semi circles


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);
void save(Ephemeris* ep, FILE* fod);

int main(void)
```

```
{
	uint8_t svNum = 16;
	FILE* fid = fopen("in.txt", "r");
	FILE* fod = fopen("out.txt", "w");
	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 ");
			}
			fclose(fid);
			if (fod) {
				save(ep, fod);
			}
			else
			{
				printf(" Cannot open out.txt ");
			}
			fclose(fod);
			free(ep);
		}
		else {
			printf(" Subframes not found\n ");
		}
	}
	else {
		printf(" Cannot open in.txt ");
	}


	return 0;
}
uint32_t str2uint(char *sf, int32_t start, int32_t stop) {
	uint32_t ans = 0;
	for (int i = start; i < stop; i++) {
		bool bit = (sf[i - 1] == '1');
		ans = ans | (bit << (stop - i - 1));

	}
	return ans;




}

uint32_t str2uint1(char *sf, int32_t start, int32_t stop, int32_t start1, int32_t stop1)
{
	uint32_t ans = 0;
	for (int i = start; i < stop; i++) {
		bool bit = (sf[i - 1] == '1');
		ans = (ans | bit) << 1;

	}
	for (int i = start1; i < stop1; i++) {
		bool bit = (sf[i - 1] == '1');
		ans = ans | bit;
		if (i < stop1 - 1) {
```

```
                        ans = ans << 1;
                }
        }
        return ans;
}
//uint32_t str2uint2(char *sf, int32_t start, int32_t stop, char *cf, int32_t start1,
int32_t stop1) {
//      uint32_t ans = 0;
//      for (int i = start; i < stop; i++) {
//              bool bit = (sf[i - 1] == '1');
//              ans = ans | (bit << (stop - i - 1));
//
//      }
//      for (int i = start1; i < stop1; i++) {
//              bool bit = (cf[i - 1] == '1');
//              ans = ans | (bit << (stop1 - i - 1));
//
//      }
//      return ans;
//}

int32_t twoCompl2int(uint32_t ans, int numBit) {        // twos-complement func
        int32_t m = 0xFFFFFFFF;
        if ((numBit < 32) && bool((1 << numBit - 1) & ans))
        {
                ans |= m << numBit;
                return -(~(ans - 1));
        }
        if (numBit == 32 && bool((1 << 31) & ans)) {
                return -(~(ans - 1));
        }
        return ans;
}

int32_t subFrames2Eph(Ephemeris* ep, SF1_3* subframes) {
        ep->slot = subframes->slot;
        ep->WN = str2uint(subframes->sf1, 61, 71);
        ep->Crs = twoCompl2int(str2uint(subframes->sf2, 69, 85), 16) * sF1;
        ep->Cuc = twoCompl2int(str2uint(subframes->sf2, 151, 167), 16)*sF4;
        ep->toe = str2uint(subframes->sf2, 271, 287)*sF7;
        ep->toc = twoCompl2int(str2uint(subframes->sf1, 219, 235), 16) * sF7;
        ep->IODC = str2uint1(subframes->sf1, 83, 85, 211, 219);
        ep->URA = str2uint(subframes->sf1, 73, 75);
        ep->Health = str2uint(subframes->sf1, 73, 79);
        ep->IODE2 = str2uint(subframes->sf2, 61, 69);
        ep->IODE3 = str2uint(subframes->sf3, 271, 279);
        //ep->IODE = twoCompl2int(str2uint2(subframes->sf2, 61, 69, subframes->sf3, 271,
279), 8);
        ep->codeL2 = bool(subframes->sf1[91]);
        ep->L2P = bool(subframes->sf1[91]);
        ep->e = str2uint1(subframes->sf2, 167, 175, 181, 205) * sF5;
        ep->af1 = twoCompl2int(str2uint(subframes->sf1, 249, 265), 16) * sF2;
        ep->af2 = twoCompl2int(str2uint(subframes->sf1, 241, 249), 8) * sF8;
        ep->af0 = twoCompl2int(str2uint(subframes->sf1, 271, 293), 22) * sF3;
        ep->Dn = twoCompl2int(str2uint(subframes->sf2, 91, 107), 16) * sF2 * sc;
        ep->M0 = twoCompl2int(str2uint1(subframes->sf2, 107, 115, 121, 145), 32) * sF3 *
sc;
        ep->Cus = twoCompl2int(str2uint(subframes->sf2, 211, 227), 16) * sF4;
        ep->sqrtA = str2uint1(subframes->sf2, 227, 235, 241, 265) * sF6;
        ep->Cic = twoCompl2int(str2uint(subframes->sf3, 61, 77), 16) * sF4;
        ep->Omega0 = twoCompl2int(str2uint1(subframes->sf3, 77, 85, 91, 115), 32) * sF3 *
sc;
        ep->Cis = twoCompl2int(str2uint(subframes->sf3, 121, 137), 16) * sF4;
```

```
	ep->i0 = twoCompl2int(str2uint1(subframes->sf3, 137, 145, 151, 175), 32) * sF3 *
sc;
	ep->Crc = twoCompl2int(str2uint(subframes->sf3, 181, 197), 16) * sF1;
	ep->omega = twoCompl2int(str2uint1(subframes->sf3, 197, 205, 211, 235), 32) * sF3
* sc;
	ep->OmegaDot = twoCompl2int(str2uint(subframes->sf3, 241, 265), 24) * sF2 * sc;
	ep->iDot = twoCompl2int(str2uint(subframes->sf3, 279, 293), 14) * sF2 * sc;
	ep->Tgd = twoCompl2int(str2uint(subframes->sf1, 197, 205), 8) * sF3;
	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      = %f \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  = %f \t[deg]        \n", ep->Omega0);
        printf("\tCis     = %e                \n", ep->Cis);
        printf("\ti0      = %f \t[deg]        \n", ep->i0);
        printf("\tCrc     = %e               \n", ep->Crc);
        printf("\tomega   = %f \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);
}

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

ПРИЛОЖЕНИЕ Б

```
clc
clear all
close all hidden
close all force

set(0,'defaultTextFontSize', 14)                          % Default Font Size
set(0,'defaultAxesFontSize', 14)                          % Default Font Size
set(0,'defaultAxesFontName','Times')                      % Default Font Type
set(0,'defaultTextFontName','Times')                      % Default Font Type
set(0,'defaultFigurePaperPositionMode','auto')            % Default Plot
position
set(0,'DefaultFigurePaperType','<custom>')                % Default Paper Type
set(0,'DefaultFigurePaperSize',[14.5 7.7])                % Default Paper Size
format longE

%% Константы
pi= 3.1415926535898; % Отношение длины окружности к ее диаметру
mu = 3.986004418*1e14; % Геоцентрическая гравитационная постоянная
omega_E = 7.2921151467*1e-5; % Средняя угловая скорость Земли
c = physconst('LightSpeed'); % Скорость света
Earth_radius = physconst('EarthRadius'); % Радиус Земли

%% Эфемериды в соответствии с моделью GPS
C_rs = 7.750000e+000; % Амплитуда поправочного члена синусоидальной гармоники
к радиусу орбиты
delta_n = deg2rad(2.55652e-7); % Среднее отклонение движения от вычисленного
значения
M_0 = deg2rad(-7.133039); % Средняя аномалия в контрольный момент времени
C_uc = 5.867332e-007; % Амплитуда поправочного члена косинусной гармоники к
аргументу широты
ecc = 1.275745e-002; % Эксцентриситет
C_us = 9.344891e-006; % Амплитуда поправочного члена синусоидальной гармоники
к аргументу широты
aSqRoot = 5.153602e+003; % Поправка к большой полуоси
t_oe = 100800; % Опорное время эфемерид
C_ic = 6.891787e-008  ; % Амплитуда поправочного члена косинусной гармоники к
углу наклона
Omega0 = deg2rad(135.039698); % Долгота восходящего узла
C_is = 1.844019e-007; % Амплитуда поправочного члена синусоидальной гармоники
к углу наклона
i0 = deg2rad(55.591104); % Наклонение в контрольный момент времени
относительно i0 = 56 град
C_rc = 2.056875e+002; % Амплитуда поправочного члена косинусной гармоники к
радиусу орбиты
omega = deg2rad(40.568872); % Аргумент перигея
OmegaDot = deg2rad(-4.556637e-007); % Скорость прямого восхождения
iDot = deg2rad(-2.312390e-008); % Скорость угла наклона


%% Расчёт Кеплеровских элементов орбиты
A = aSqRoot^2; % Большая полуось
n_0 = sqrt(mu/A^3); % Расчетное среднее движение

%% Алгоритм расчёта координат
leapSeconds = 18; % Секунды координации
daySecondsCount = 86400;

for ind = 1:daySecondsCount
t = daySecondsCount + leapSeconds + ind; % Количество секунд от начала
текущей недели
t_k = t - t_oe; % Время от опорной эпохи эфемерид
```

```
    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
E_0 = M_k;
E_k = 0;
count = 0;

while true
    E_k = E_0;
    E_0 = E_0 + (M_k - E_0 + ecc * sin(E_0))/(1 - ecc * cos(E_0));
    if abs(E_0-E_k) < 1e-8
        break
    end
    count = count + 1;
end

nu = atan2((sqrt(1 - ecc^2) * sin(E_k)/(1 - ecc * cos(E_k))), ((cos(E_k) -
ecc)/(1 - ecc * cos(E_k)))); % Истинная аномалия
Phi = nu + omega; % Аргумент широты
delta_u = C_us*sin(2*Phi) + C_uc*cos(2*Phi); % Аргумент поправки на широту
delta_r = C_rs*sin(2*Phi) + C_rc*cos(2*Phi); % Коррекция радиуса
delta_i = C_is*sin(2*Phi) + C_ic*cos(2*Phi); % Коррекция наклона
u_k = Phi + delta_u; % Скорректированный аргумент широты
r = A * (1 - ecc * cos(E_k)) + delta_r; % Скорректированный радиус
i_corr = i0 + delta_i + iDot * t_k; % Скорректированное наклонение

x_shtr = r * cos(u_k);
y_shtr = r * sin(u_k);

Omega_corr = Omega0 + (OmegaDot - omega_E) * t_k - omega_E * t_oe; %
Скорректированная широта восходящего узла

x(ind) = x_shtr * cos(Omega_corr) - y_shtr * cos(i_corr) * sin(Omega_corr); %
Координаты КА
y(ind) = x_shtr * sin(Omega_corr) + y_shtr * cos(i_corr) * cos(Omega_corr);
z(ind) = y_shtr * sin(i_corr);

% Инерциальная СК
Theta = omega_E * t_k;
x_ics(ind) = x(ind) * cos(Theta) - y(ind) * sin(Theta);
y_ics(ind) = x(ind) * sin(Theta) + y(ind) * cos(Theta);
z_ics(ind) = z(ind);

% Вычисление координат приёмника
lat = deg2rad(44.09363261); % Широта
lon = deg2rad(39.00130546); % Долгота
H = 1.247; % Высота

x_point = (Earth_radius + H)*cos(lat)*cos(lon);  % Расчёт координат приёмника
y_point = (Earth_radius + H)*cos(lat)*sin(lon);
z_point = (Earth_radius + H)*sin(lon);
```

```
[xRezult(ind), yRezult(ind), zRezult(ind)] =
ecef2enu(x(ind),y(ind),z(ind),lat,lon,H,wgs84Ellipsoid,'radians'); %
Отображение координат КА относительно передатчика

% Полярная СК
    if zRezult(ind) > 0
    rho(ind) = norm([xRezult(ind),yRezult(ind),zRezult(ind)]);
    theta(ind) = acos(zRezult(1,ind)/rho(ind));
        if xRezult(ind) > 0
            phi(ind) = -atan(yRezult(ind)/xRezult(ind))+pi/2;
        elseif (xRezult(ind)<0)&&(yRezult(ind)>0)
            phi(ind) = -atan(yRezult(ind)/xRezult(ind))+3*pi/2;
        elseif (xRezult(ind)<0)&&(yRezult(ind)<0)
            phi(ind) = -atan(yRezult(ind)/xRezult(ind))-pi/2;
        end
    else
        rho(ind) = nan;
        theta(ind) = nan;
        phi(ind) = nan;
    end
end

%% Вывод значений координат в файл out.txt
outTxt = fopen('out.txt', 'w');
formatSpec = '%1.0f %10.8f %10.8f %10.8f\n';
for indCount = 1:length(x)
    fprintf(outTxt, formatSpec, indCount, x(indCount), y(indCount),
z(indCount));
end
fclose(outTxt);
type out.txt

%% Расчёт для построения Земли
[x_sphere,y_sphere,z_sphere] = sphere(50);
x_Earth = Earth_radius*x_sphere;
y_Earth = Earth_radius*y_sphere;
z_Earth = Earth_radius*z_sphere;

%% Построение графиков
figure(1)
subplot(1,1,1)
surf(x_Earth,y_Earth,z_Earth)
hold on
plot3(x, y, z)
plot3(x_point,y_point ,z_point,'o','Color' , 'r', 'MarkerSize' , 5,
'MarkerFaceColor','r') % Построение точки
xlabel('x, м')
ylabel('y, м')
zlabel('z, м')
daspect([1,1,1])

figure(2)
subplot(1,1,1)
surf(x_Earth,y_Earth,z_Earth)
hold on
plot3(x_ics(1,:), y_ics(1,:), z_ics(1,:))
plot3(x_point,y_point ,z_point,'o','Color' , 'r', 'MarkerSize' , 5,
'MarkerFaceColor','r') % Построение точки
xlabel('x, м')
ylabel('y, м')
zlabel('z, м')
daspect([1,1,1])
```

```
figure (3)
axes = polaraxes;
polarplot(axes,phi,rad2deg(theta))
axes.ThetaDir = 'clockwise';
axes.ThetaZeroLocation = 'top';
rlim([0 80]) % Учёт угла места
```