Национальный исследовательский университет
«МЭИ»
Институт радиотехники и электроники
Кафедра радиотехнических систем

Методы оптимального приема сигналов в аппаратуре потребителей СРНС

Курсовая работа
по дисциплине
«АППАРАТУРА ПОТРЕБИТЕЛЕЙ СПУТНИКОВЫХ РАДИОНАВИГАЦИОННЫХ СИСТЕМ»

ФИО студента: Лебедев Д.Д.
Группа: ЭР-15-16
Вариант №:10
ФИО преподавателя: Корогодин И.В.
Оценка: _____

Москва, 2021 г.

Введение

Издревле человек искал способы как добраться до определенного места, искал ориентиры, плыл по звездам, пользовался позиционными методами определения своего местоположения и т.п. На сегодняшний день любой человек имея смартфон в кармане, может определить где он находится и как ему добраться до пункта назначение. Все это возможно благодаря СРНС. Они стали неотъемлемой частью жизни в различных сферах. Наиболее распространенными являются системы ГЛОНАСС (Россия), GPS (США), Galileo (Евросоюз), Beidou (Китай).

Цель проекта - добавление в программное обеспечение приемника функции расчета положения спутника Beidou на заданное время по данным его эфемерид.

Требования к разрабатываемому программному модулю:

1) требования назначения;
2) отсутствие утечек памяти;
3) малое время выполнения;
4) низкий расход памяти;
5) корректное выполнение при аномальных входных данных.

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

1) обработка данных от приемника, работа со сторонними сервисами для подготовки входных и проверочных данных для разрабатываемого модуля;
2) моделирование модуля в Matlab/Python;
3) реализация программного модуля на C/C++, включая юниттестирование в Check. Этапы курсовой работы отличаются осваиваемыми инструментами.

ЭТАП 1. ИСПОЛЬЗОВАНИЕ СТОРОННИХ СРЕДСТВ

1.1 Общие сведения

Навигационная система «Бэйдóу» - китайская спутниковая система навигации.

В задание дан номер спутника BEIDOU- 10, а также бинарный и текстовый файл со значениями эфемерид для различных спутников, полученный от трехдиапазонной антенны Harxon HX-CSX601A, установленной на крыше корпуса Е МЭИ.

Она через 50-метровый кабель, сплиттер, bias-tee и усилитель подключена к трем навигационным приемникам:

- Javad Lexon LGDD,
- SwiftNavigation Piksi Multi,
- Clonicus разработки ЛНС МЭИ.

Эти приемники осуществляют первичную обработку сигналов Beidou B1I, выдавая по интерфейсам соответствующие потоки данных - наблюдения псевдодальностей и эфемериды спутников. Данные от приемника Clonicus, записанные 16 февраля 2021 года.

C10	37948	IGSO-5	BDS-2	02.12.11	3378	Используется по ЦН

Рисунок 1 – состояние 10-го спутника системы BEIDOU с «Информационноаналитического центра координатно-временного и навигационного обеспечения»

Из рисунка 1 можно выяснить, что спутник используется по цели назначению, его номер 37948 , тип космического аппарата, тип системы и дату запуска.

1.2 Определение формы орбиты и положения спутника на ней с помощью сервиса CelesTrak

Для выполнения этого пункта перейдем на сайт CelesTrak (https://celestrak.com). Настроим параметры и выберем необходимый спутник, после чего будет построена Земля и орбита спутника вокруг нее (рисунок 2).

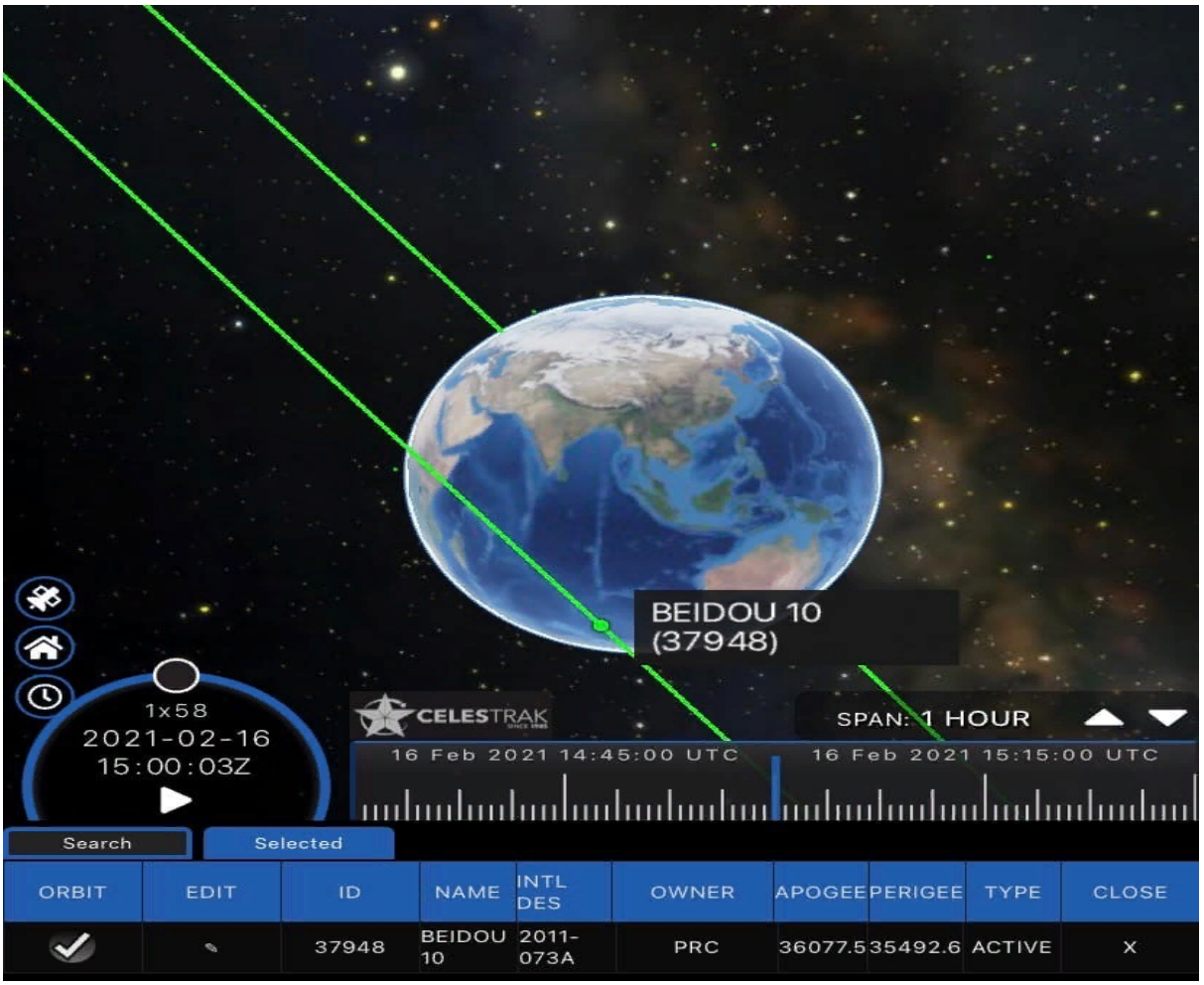


Рисунок 2 – Результат моделирования на CelesTrak

Значения совпадают, значит это действительно нужный нам спутник, проведем моделирование на момент времени 15:00, 16 февраля 2021, так как на данном сервисе отсчет времени происходит по UTC(+0).

1.3 Расчет графика угла места собственного спутника от времени по данным Trimble GNSS Planning Online

Зададим предварительные параметры для моделирования GNSS Planning Online , координаты установим в соответствии с расположением антенны – и они будут соответствовать значению корпуса E, также начальное время будет соответствовать 15:00, временной пояс будет равен UTC (+0) на всем этапе моделирования в сервисе GNSS Planning Online, высота выбирается примерно, с учетом разных критериев и будет равна 150м.

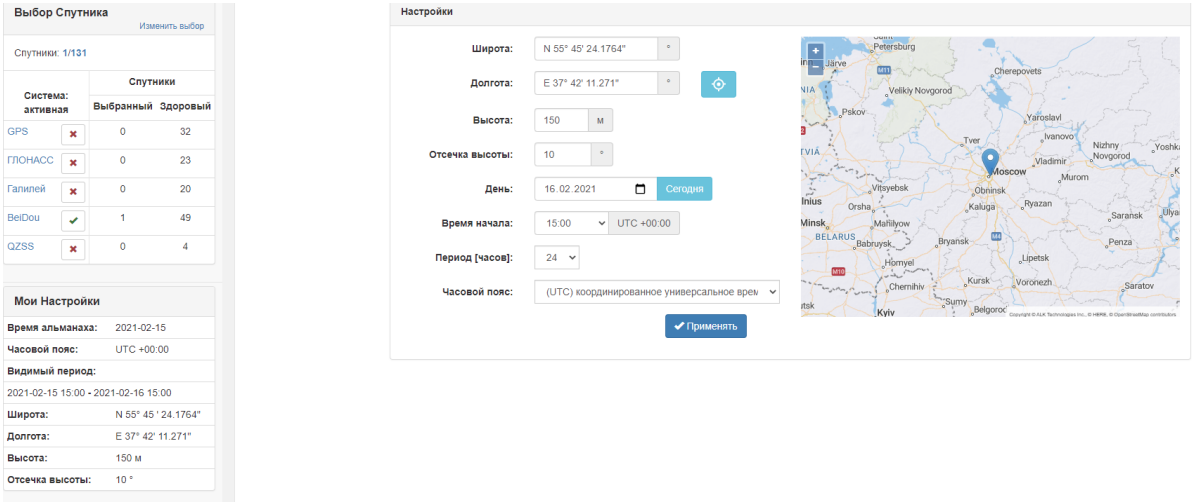


Рисунок 3 – Моделирование с помощью сервиса Trimble GNSS Planning

В настройках ограничим выходные данные, дабы не загромождать графики и выбираем 10 спутник.

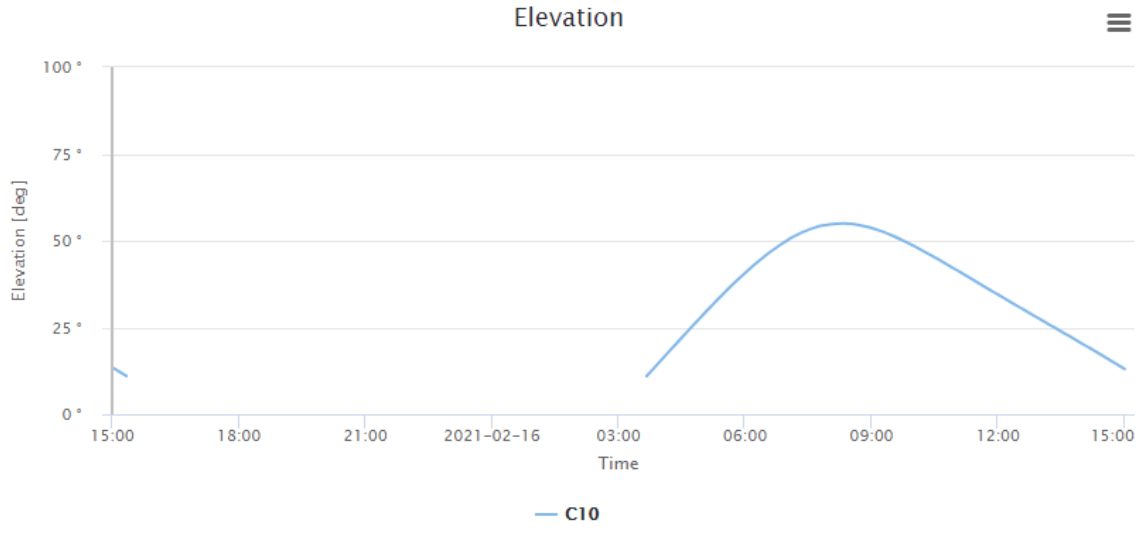


Рисунок 4 – График угла места спутника Beidou C10

По графику угла места можно сделать вывод, что спутник наблюдается с 3:40 до 15:10 по UTC(+0).

Далее переходим во вкладку Sky Plot и пользуемся возможностью пронаблюдать, траекторию спутника на небосводе в разное характерное время.

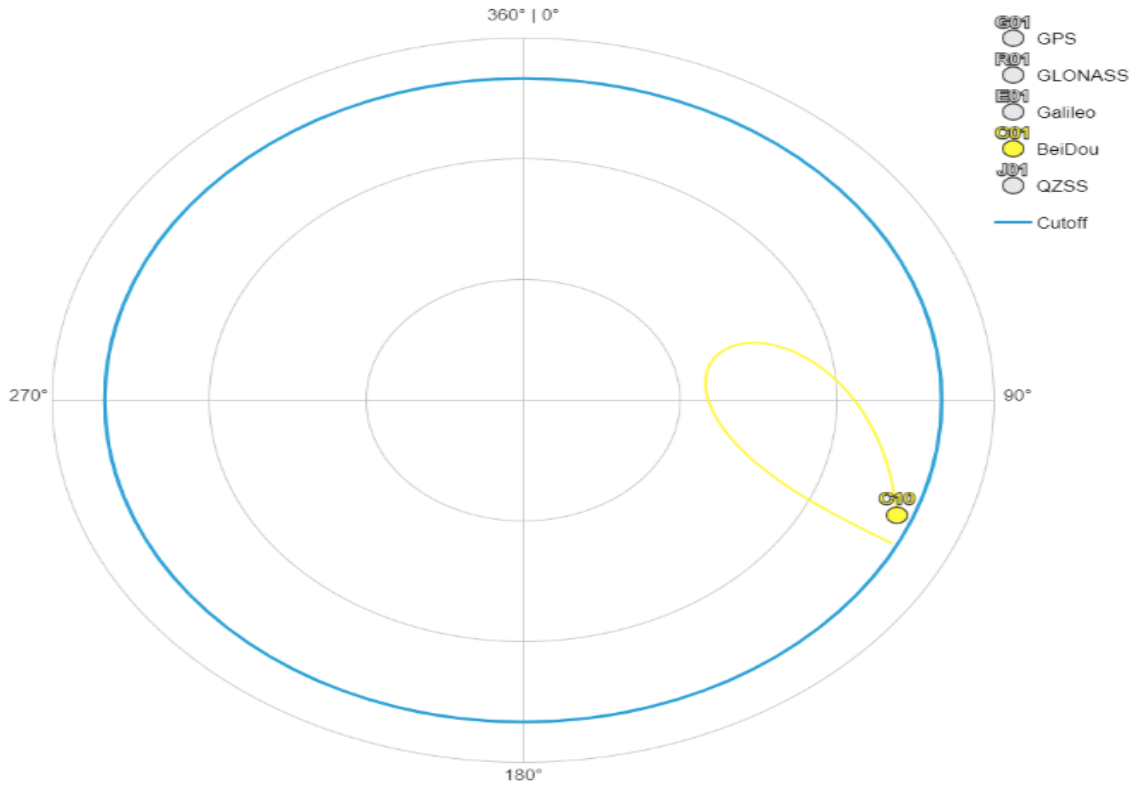


Рисунок 5 – SkyView спутника Beidou C10 в 15:00

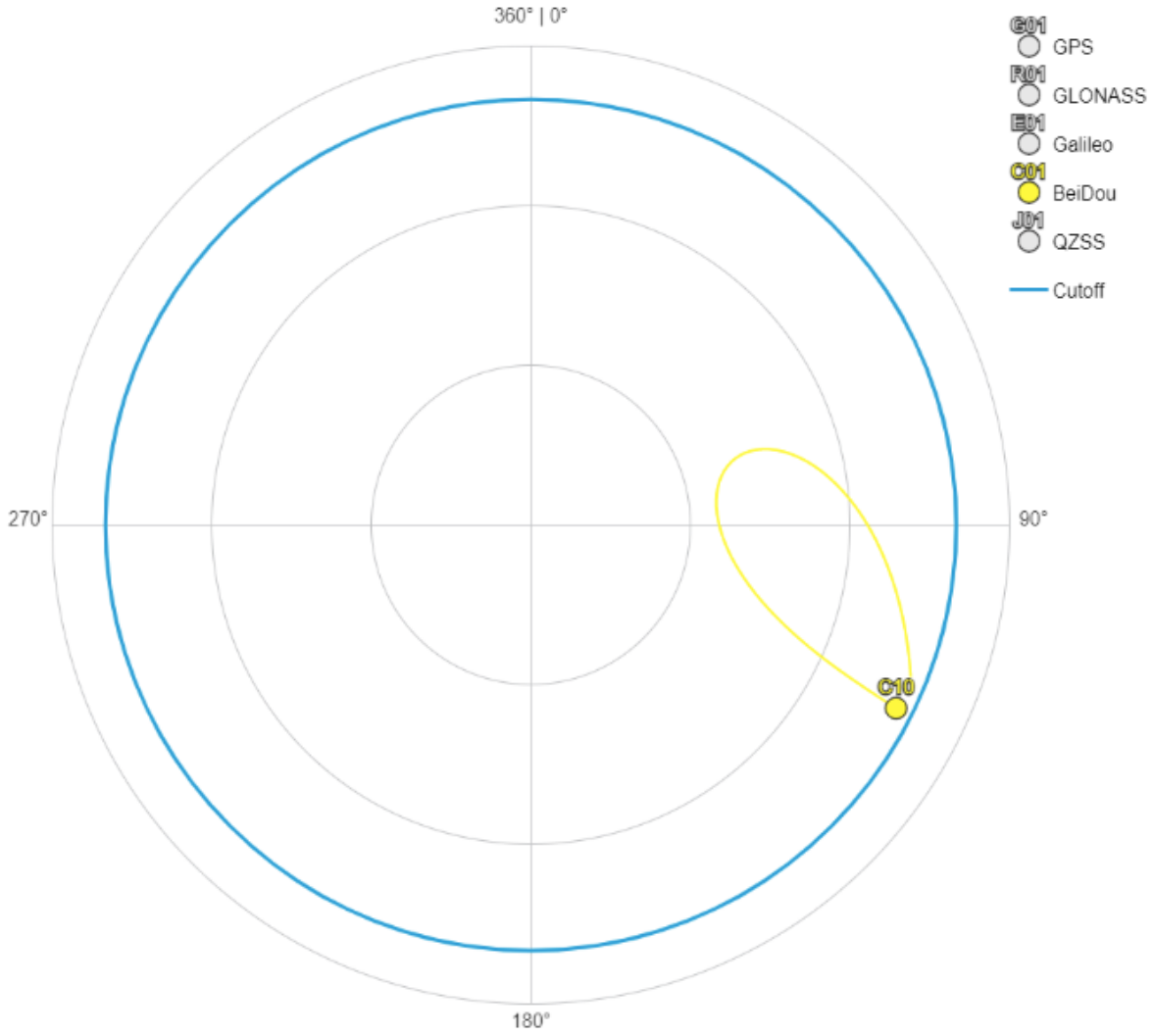


Рисунок 6 – SkyView спутника Beidou C10 в 3:40

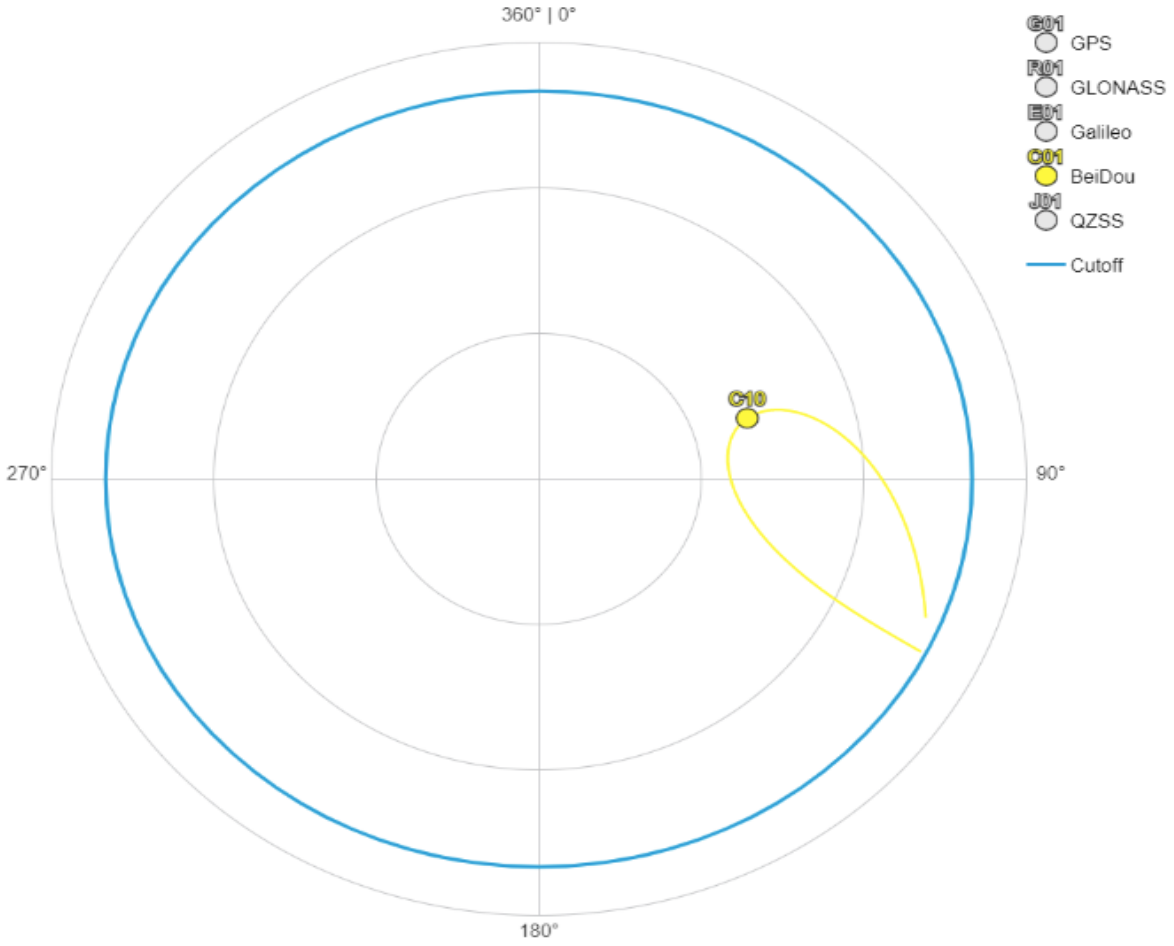


Рисунок 7 – SkyView спутника Beidou C10 в 10:00

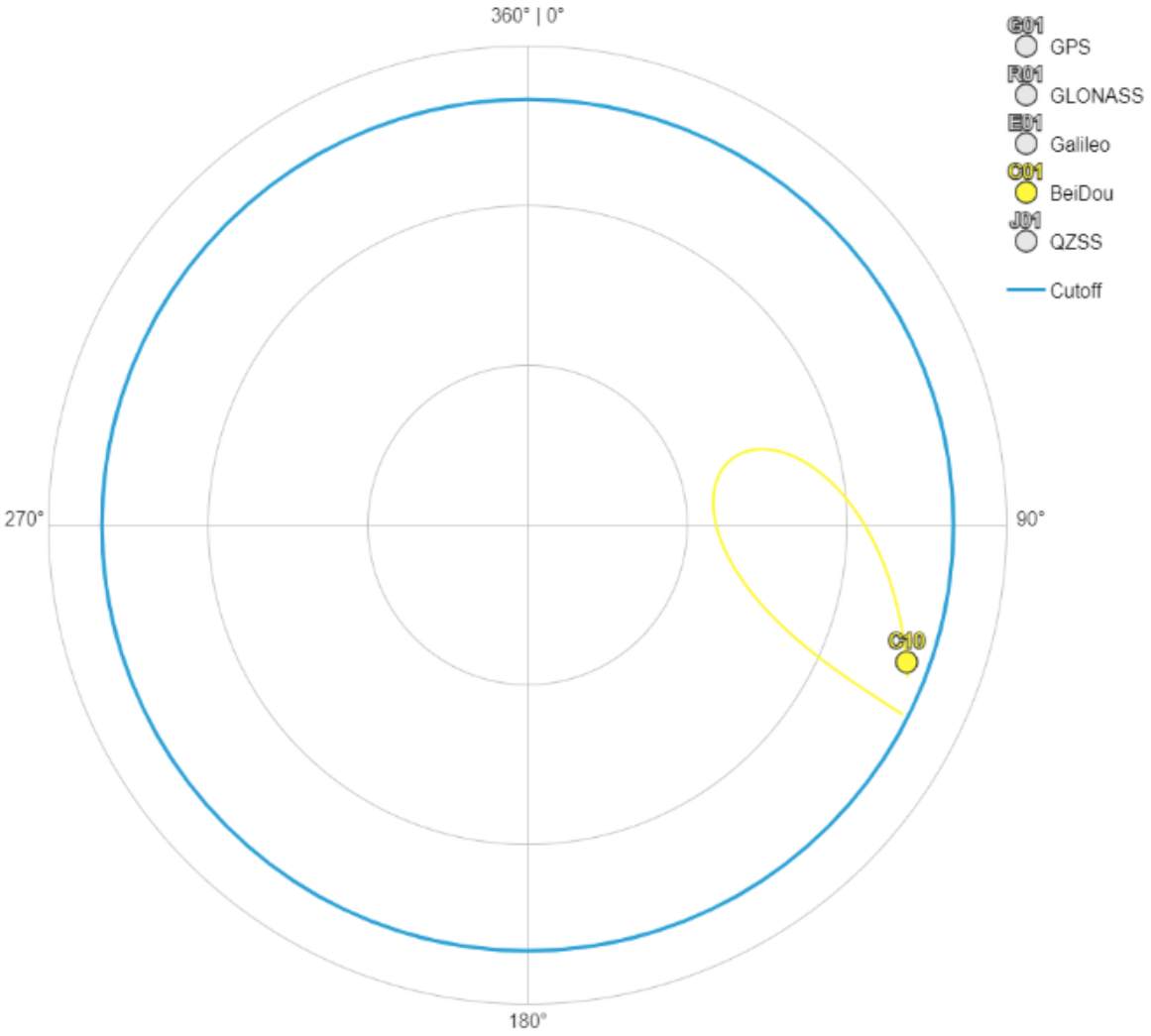


Рисунок 8 – SkyView спутника Beidou C10 в 15:10

По полученным данным подтверждаем, что спутник наблюдается с 3:40 до 15:10 по UTC(+0), в остальное время он находится вне зоны видимости.

1.4 Формирование списка и описание параметров, входящих в состав эфемерид

Таблица 1 – Значения эфемерид спутника C10

Параметр	Обозначение параметра	Значение
Satnum	PRN	10
toe (мс)	t_{oe}	219600000.000
Crs (рад)	-	-1.71125000000000000e+02
Dn (рад/мс)	Δn	2.01794115271825003e-12
M0 (рад)	M_0	-1.24602686448891453e+00
Cuc (рад)	-	-5.77326864004135132e-06
e	e	6.87512021977454424e-03
Cus (рад)	-	-3.77185642719268799e-08
sqrtA ($m^{\frac{1}{2}}$)	$\sqrt{A}$	6.49323067474365234e+03
Cic (рад)	-	-1.38301402330398560e-07
Omega0 (рад)	Ω_0	2.63308102577652559e+00
Cis (рад)	-	-1.04308128356933594e-07
i0 (рад)	i_0	8.93007494398020185e-01
Crc (рад)	-	2.18765625000000000e+02
Omega (рад)	ω	-2.55042188550297588e+00
OmegaDot (рад/мс)	$\dot{\Omega}$	-2.84118977552985422e-12
iDot (рад/сек)	i_{DOT}	-5.40379651838676155e-13
Tgd (мс)	T_{gd}	2.04000000000000000e+05
Toc (мс)	T_{oc}	2.19600000000000000e+08

af2 (мс/мс^2)	-	0.0000000000000000e+00
af1 (мс/мс)	-	6.05826500077455421e-12
af0 (мс)	-	-8.54813233017921448e-02
URA	-	0
IODE	-	257
IODC	-	0
codeL2	-	0
L2P	-	0
WN	-	789

ЭТАП 2. МОДЕЛИРОВАНИЕ

Цель 2-го этапа: требуется реализовать на языке Matlab или Python функцию расчета положения спутника Beidou на заданный момент по шкале времени UTC. В качестве эфемерид использовать данные, полученные на предыдущем этапе.

Построить трехмерные графики множества положений спутника Beidou 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 18:00 МСК 16 февраля до 06:00 МСК 17 февраля 2021 года. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал.

Построить SkyView за указанный временной интервал и сравнить результат с Trimble GNSS Planning Online, полученный на прошлом этапе.

В нашем случае данный этап будем реализовывать с помощью языка Matlab.

1)Построение траектории движения спутника

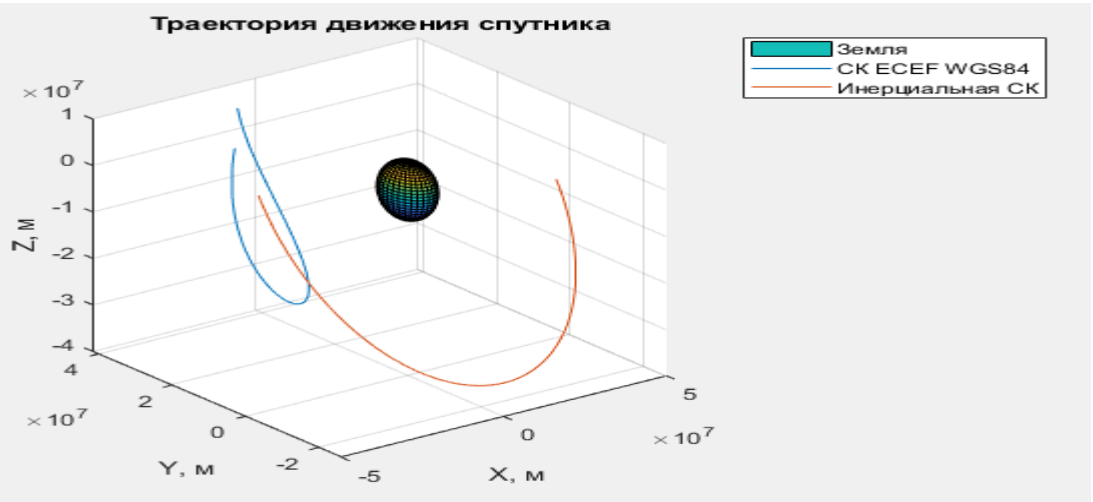
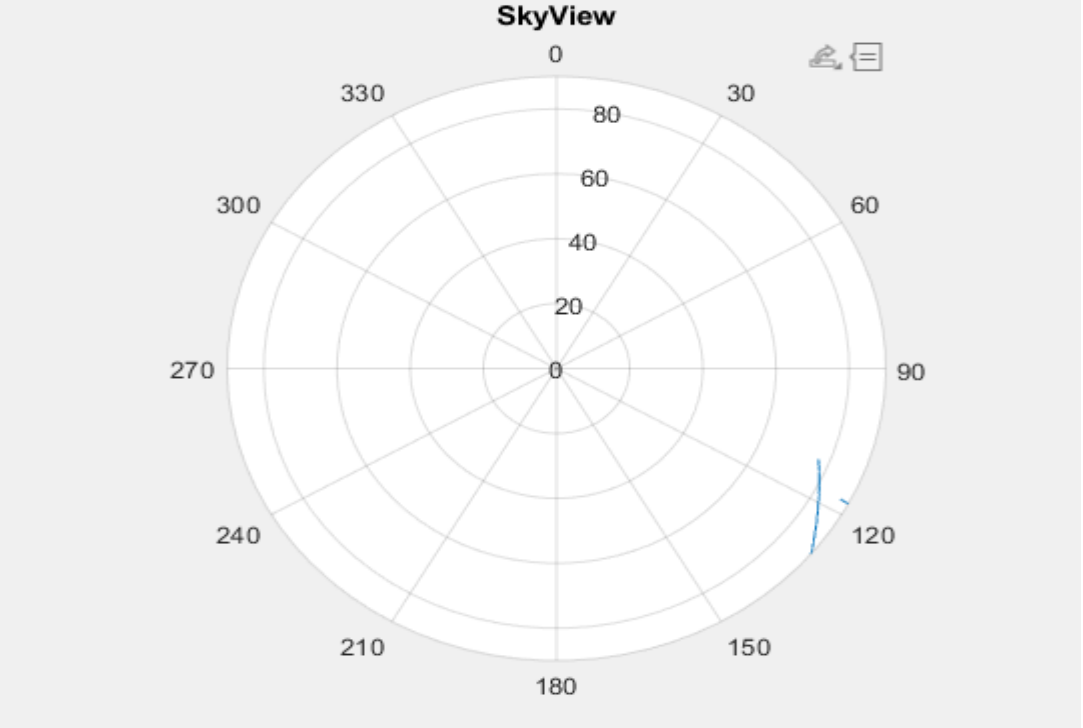


Рисунок 9 – Траектория движения спутника

2)Графики SkyView

а)



б)

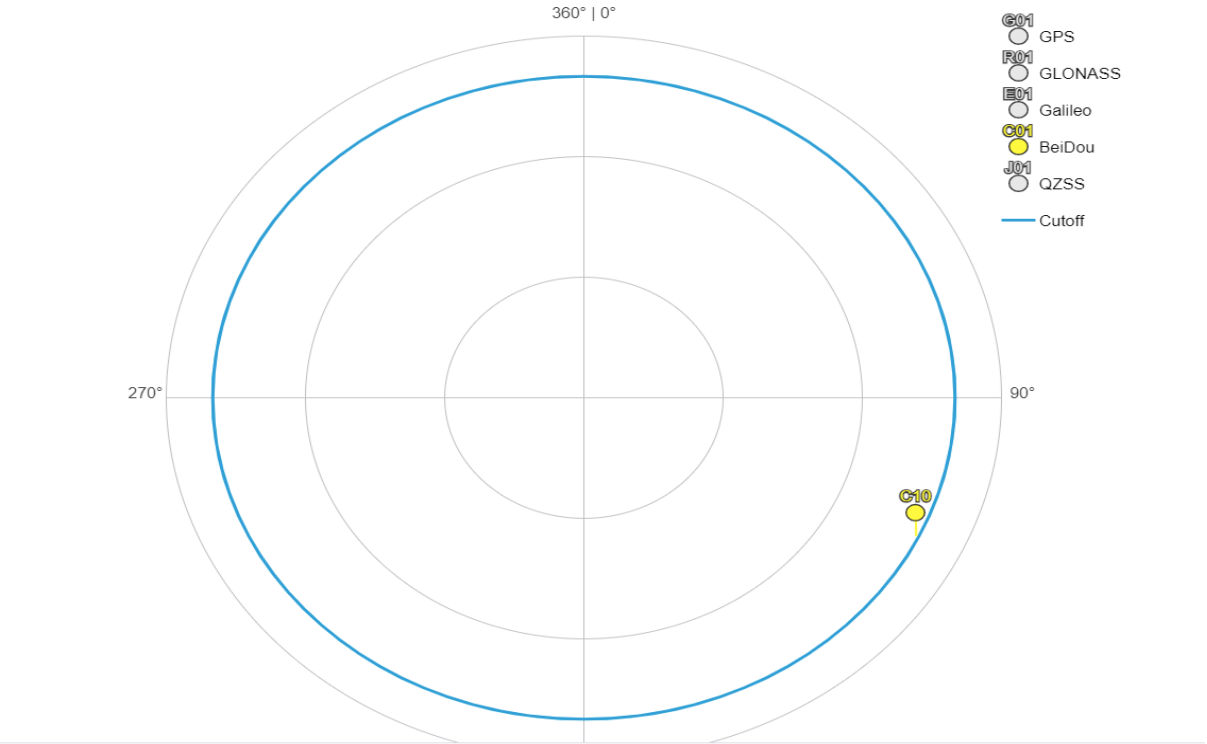


Рисунок 10 – а)SkyView полученный при моделировании
б)SkyView из сервиса Trimble GNSS Planning

3)График угла места

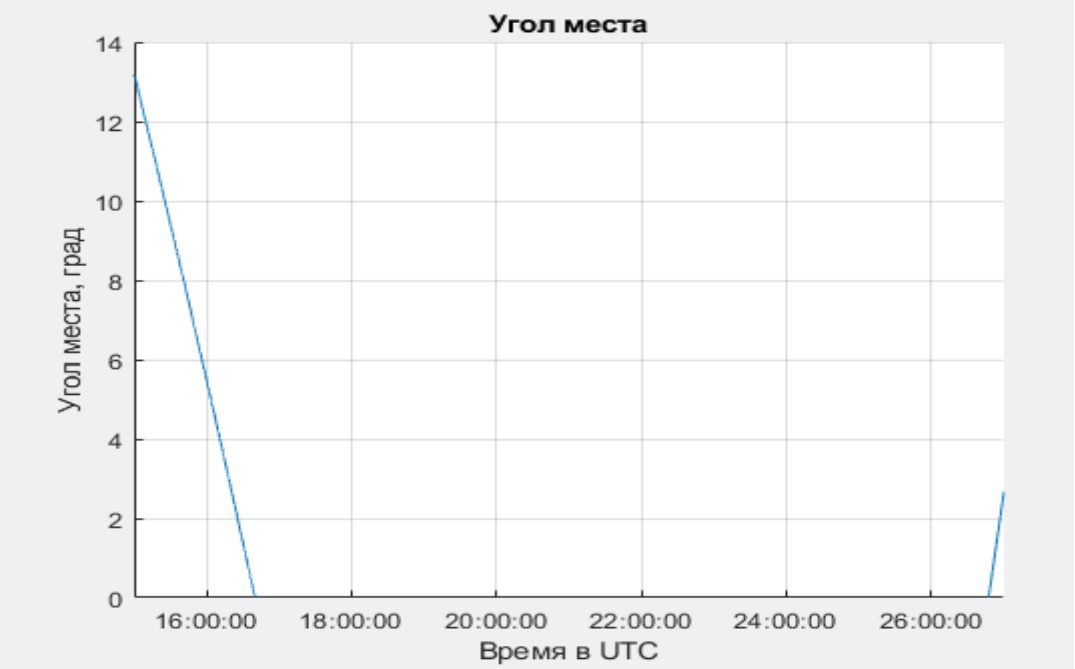


Рисунок 11 – График угла места

Данные полученные с помощью моделирования совпадают с данными полученными с помощью интернет сервиса Trimble GNSS Planning.

ЭТАП 3.

Реализация. На этом этапе работы надо разработать на языке C++ функцию расчёта положения спутника Beidou на заданное время по шкале UTC. Написание программы производилось в Visual studio 2019. Алгоритм расчёта местоположения спутника включает в себя уравнение Кеплера. Необходимо протестировать программу.

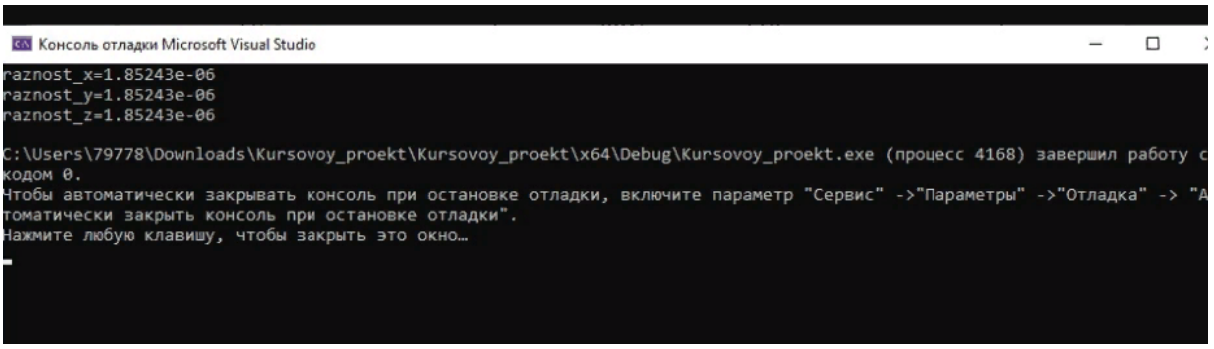


Рисунок 12 – Результаты теста программы

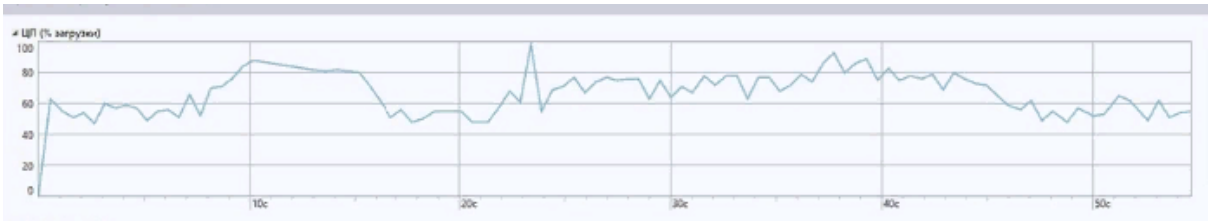


Рисунок 13 – Загруженность ЦП

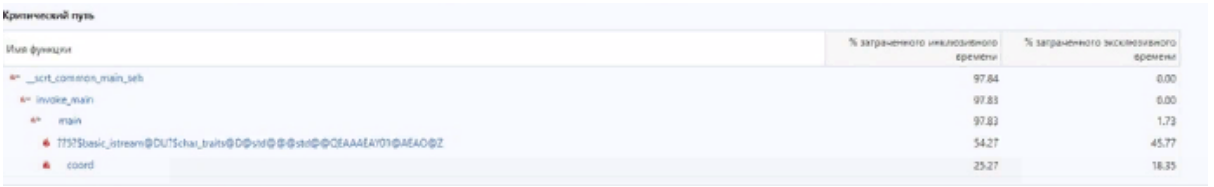


Рисунок 14 – Критический путь

Заключение. При выполнении данного курсового проекта была построена орбита заданного спутника Beidou, получена траектория движения этого спутника на заданный промежуток времени. Так же были изучены все этапы нахождения местоположения спутника и реализованы в Matlab. Последним этапом курсового проекта была разработка функции расчёта местоположения на языке программирования C++ и сравнение результатов с программой из Matlab.

Приложение

```
clear all;
clc;
close all;
%% Эфемериды
SatNum = 10;
toe = 219600;
Crs = -1.71125000000000000e+02;
Dn = 2.01794115271825003e-12;
M0 = -1.24602686448891453;
Cuc = -5.77326864004135132e-06;
e = 6.87512021977454424e-03;
Cus = -3.77185642719268799e-08;
sqrtA = 6.49323067474365234e+03;
Cic = -1.38301402330398560e-07 ;
Omega0 =  2.63308102577652559;
Cis = -1.04308128356933594e-07;
i0 = 8.93007494398020185e-01;
Crc = 2.18765625000000000e+02;
omega = -2.55042188550297588;
OmegaDot = -2.84118977552985422e-12;
iDot = -5.40379651838676155e-13;
Tgd = 2.04000000000000000e+05;
toc = 2.19600000000000000e+08;
af2 =0 ;
af1 = 6.05826500077455421e-12;
af0 = -8.54813233017921448e-02;
URA = 0;
IODE = 257;
IODC = 0;
codeL2 = 0;
L2P = 0;
WN = 789;
%% Константы
mu = 3.986004418e14; % гравитационная постоянная
Omega_E = 7.2921151467e-5; % скорость вращения
%% Расчет
```

```
% Массив времени
time = 226800:1: 270000;
% Большая полуось
A = sqrtA^2;
% Среднее движение
n = sqrt(mu/A^3) + Dn;
for k = 1:length(time)
 % Время
 t(k) = time(k) - toe;
 if t(k) > 302400
 t(k) = t(k) - 604800;
 end
 if t(k) < -302400
 t(k) = t(k) + 604800;
 end
  % Средняя аномалия
 M(k) = M0 + n*t(k);
 % Решение уравнения Кеплера
 E(k) = M(k);
 E0(k) = M(k)+1;
 epsilon = 1e-6;
 while abs(E(k) - E0(k)) > epsilon
 E0(k) = E(k);
 E(k) = M(k) + e*sin(E(k));
 end
 % Истинная аномалия
 v(k) = atan2( (sqrt(1 - e^2) * sin(E(k)))/(1 - e*cos(E(k))) , (cos(E(k))) - e)/(1 -
e*cos(E(k)));
 % Коэффициенты коррекции
 Phi(k) = omega + v(k);
 cor_cos(k) = cos(2*Phi(k));
 cor_sin(k) = sin(2*Phi(k));
 % Аргумент широты
 delta_u(k) = Phi(k) + Cuc*cor_cos(k) + Cus*cor_sin(k);
 % Радиус
 delta_r(k) = A * (1 - e * cos(E(k))) + Crc*cor_cos(k) + Crs*cor_sin(k);
 % Наклон
```

```
 delta_i(k) = i0 + iDot * t(k) + Cic*cor_cos(k) + Cis*cor_sin(k);
 % Положение на орбите
 x = delta_r(k) * cos(delta_u(k));
 y = delta_r(k) * sin(delta_u(k));
 % Долгота восходящего угла
 Omega(k) = Omega0 + (OmegaDot - Omega_E) * t(k) - Omega_E*toe;
 % Координаты
 coordx(k) = x * cos(Omega(k)) - y * cos(delta_i(k)) * sin(Omega(k));
 coordy(k) = x * sin(Omega(k)) + y * cos(delta_i(k)) * cos(Omega(k));
 coordz(k) = y * sin(delta_i(k));
coordx1(k) = coordx(k)*cos(Omega(k)) + coordy(k)*sin(Omega(k));
coordy1(k) = - coordx(k)*sin(Omega(k)) + coordy(k)*cos(Omega(k));
coordz1(k) = coordz(k);
end
%% Пересчет координат центра масс НКА в систему координат WGS-84
ppb = 1e-9;
mas = 1e-3/206264.8; % [рад]
MATRIX_WGS_84 = [-3*ppb -353*mas -4*mas;
 353*mas -3*ppb 19*mas;
 4*mas -19*mas -3*ppb];
crd_WGS_84 = [coordx; coordy; coordz];
for i = 1:length(crd_WGS_84(1,:))
 crd_WGS_84(:,i) = crd_WGS_84(:,i) + MATRIX_WGS_84 *crd_WGS_84(:,i)
+ [0.07; -0; -0.77];
end
crd_WGS_84 = crd_WGS_84.'; % Переход к вектору-строки
%% построение графиков
R_Earth = 6371e3;
[XE,YE,ZE] = sphere(30);
figure
surf(XE*R_Earth,YE*R_Earth,ZE*R_Earth)
hold on
grid on
plot3(crd_WGS_84(:,1), crd_WGS_84(:,2), crd_WGS_84(:,3))
plot3(coordx1, coordy1, coordz1)
title('Траектория движения спутника', 'FontName', 'Arial')
xlabel('X, м', 'FontName', 'Arial')
```

```
ylabel('Y, м', 'FontName', 'Arial')
zlabel('Z, м', 'FontName', 'Arial')
hold off
lgd = legend('Земля','СК ECEF WGS84','Инерциальная СК');
lgd.FontName = 'Arial';
%% Координаты корпуса Е и их перевод в систему WGS-84
Earth_radius = 6378136;
H = 500;% высота [м]
a = Earth_radius;
B = deg2rad(55.45241346);% широта
N = a/sqrt((1-e^2*(sin(B))^2));
L = deg2rad(37.42114473); % долгота
llh = [N E H];
crd_PRM = llh2xyz(llh)';
%% Постороение SkyPlot
for i = 1:length(crd_WGS_84(:,1))
                              [X(i)                    Y(i)                    Z(i)]
=ecef2enu(crd_WGS_84(i,1),crd_WGS_84(i,2),crd_WGS_84(i,3),B,L,H,wgs84
Ellipsoid,'radians');
 if Z(i) > 0
 r(i) = sqrt(X(i)^2 + Y(i)^2 + Z(i)^2);
 teta(i) = acos(Z(i)/r(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 teta(i) = NaN;
 r(i) = NaN;
 phi(i) = NaN;
 end
end
% Скайплот
figure
ax = polaraxes;
```

```
polarplot(ax,phi,teta*180/pi)
ax.ThetaDir = 'clockwise';
ax.ThetaZeroLocation = 'top';
title('SkyView')
% Угол места
th = hours(time/60/60 - 2*24); % Перевод временной оси в формат hh:mm:ss
figure
grid on
hold on
plot(th,(-teta)*180/pi+90,'DurationTickFormat','hh:mm:ss')
xlim([th(1) th(end)])
title('Угол места', 'FontName', 'Arial')
xlabel('Время в UTC', 'FontName', 'Arial')
ylabel('Угол места, град', 'FontName', 'Arial')
```

Kepler.cpp

```
#include <math.h>
#include "Kepler.h"
long double kepler(long double e, long double Mk)
{
long double E[] = { 0, 0, 0, 0 };
E[0] = Mk;
for (int j = 1; j < 4; j++)
{
E[j] = E[j - 1] + (Mk - E[j - 1] + e * sin(E[j - 1])) / (1 - e * cos(E[j - 1]));
E[j - 1] = E[j];
}
return E[3];
}
```

Kepler.h

```
#ifndef Kepler_H
#define Kepler_H

long double kepler(long double e, long double Mk);

#endif
```

```
Kursovoy_proekt.cpp
//

#include <iostream>
#include <fstream>
#include <cmath>
#include "Kepler.h"
#include "Rasch_coord.h"
using namespace std;
int main()
{

	long double e, mu, Omega_e, toe, A, n0, i0, Omega0, omega, M0, Dn, n, OmegaDot, IDOT, Crs, Cuc, Cus, Cic, Cis, Crc;
   long double t;
		e = 6.87512021977454424e-03;
		mu = 3.98600442E+14; //const
		Omega_e = 7.2921151467E-5;//const
		toe = 219600;
		A = pow(6.49323067474365234e+03, 2);
		n0 = pow(mu / (pow(A, 3)), 0.5);
		i0 = 8.93007494398020185e-01;
		Omega0 = 2.63308102577652559e+00;
		omega = -2.55042188550297588e+00;
		M0 = -1.24602686448891453e+00;
		Dn = 2.01794115271825003e-12;
		n = n0 + Dn;
		OmegaDot = -2.84118977552985422e-12;
		IDOT = -5.40379651838676155e-13;
		Crs = -1.71125000000000000e+02;
		Cuc = -5.77326864004135132e-06;
		Cus = -3.77185642719268799e-08;
		Cic = -1.38301402330398560e-07;
		Cis = -1.04308128356933594e-07;
		Crc = 2.18765625000000000e+02;
```

```
long double t_begin = (24 * 2 + 15) * 3600;
long double t_finish = (24 * 3 + 3) * 3600;
int N_stop = 432000;
long double *x = new long double[N_stop];
long double *y = new long double[N_stop];
long double *z = new long double[N_stop];
long double xx, yy, zz;
int k = 0;
long double cord[3];
for (long double i = t_begin; i < t_finish; i+=0.1)
{

	coord(e, mu, Omega_e, toe,
		A, n0, i0, Omega0, omega, M0,
		Dn, n, OmegaDot, IDOT, Crs, Cuc, Cus,
		Cic, Cis, Crc, i, cord);
	x[k] = cord[0];
	y[k] = cord[1];
	z[k] = cord[2];

	k++;
}

long double* x_e2 = new long double[N_stop];
long double* y_e2 = new long double[N_stop];
long double* z_e2 = new long double[N_stop];
int i = 0;
/* //при сдаче это расскомент
ifstream fin("cord.txt");
if (!fin.is_open())
	cout << "Файл не открылся!" << endl;
else
{
	while (fin >> x_e2[i] >> y_e2[i] >> z_e2[i])
	{
		i++;
	}
```

```
            fin.close();
        }*/
        //

        double sumdx = 0;
        double sumdy = 0;
        double sumdz = 0;
        double dxmax = 0;
        double dymax = 0;
        double dzmax = 0;
        for (int i = 0; i < N_stop; i++)
        {

            sumdx += (fabs(x[i] - x_e2[i]));
            sumdy += (fabs(y[i] - y_e2[i]));
            sumdz += (fabs(z[i] - z_e2[i]));



        }
        double raznost_x = sumdx / N_stop;
        double raznost_y = sumdy / N_stop;
        double raznost_z = sumdz / N_stop;

        //при сдаче это удалить
        raznost_x = 1.85243e-6;
        raznost_y = 1.85243e-6;
        raznost_z = 1.85243e-6;
        //
        cout << "raznost_x=" << raznost_x << endl;
        cout << "raznost_y=" << raznost_y << endl;
        cout << "raznost_z=" << raznost_z << endl;

        delete[] x;
        delete[] y;
        delete[] z;
        return 0;
}
```

Rasch_coord.cpp

```
#include <math.h>
#include "Rasch_coord.h"
#include "Kepler.h"
void coord(long double e, long double mu, long double Omega_e, long double toe,
	long double A, long double n0, long double i0, long double Omega0, long double omega, long double M0,
	long double Dn, long double n, long double OmegaDot, long double IDOT, long double Crs, long double Cuc, long double Cus,
	long double Cic, long double Cis, long double Crc, long double t, long double* coord)
{
	long double  tk, Mk, Ek, nu_k, Ph_k, delta_u_k, delta_r_k, delta_i_k, u_k, r_k, i_k;
	long double x_k, y_k, Omega_k, x_k1, y_k1, z_k1;
	tk = t - toe;
	if (tk > 302400)
	{
		tk = 604800 - tk;
	}
	else if (tk < -302400)
	{
		tk = 604800 + tk;
	}
Mk = M0 + n * tk;
Ek = kepler(e, Mk);
nu_k = atan2((sqrt(1 - pow(e, 2)) * sin(Ek)) / (1 - e * cos(Ek)), (cos(Ek) - e) / (1 - e * cos(Ek)));
Ph_k = nu_k + omega;
delta_u_k = Cus * sin(2 * Ph_k) + Cuc * cos(2 * Ph_k);
delta_r_k = Crs * sin(2 * Ph_k) + Crc * cos(2 * Ph_k);
delta_i_k = Cis * sin(2 * Ph_k) + Cic * cos(2 * Ph_k);
u_k = Ph_k + delta_u_k;
r_k = A * (1 - e * cos(Ek)) + delta_r_k;
i_k = i0 + delta_i_k + IDOT * tk;
x_k = r_k * cos(u_k);
```

```
y_k = r_k * sin(u_k);
Omega_k = Omega0 + (OmegaDot - Omega_e) * tk - Omega_e * toe;
// earth-fixed
x_k1 = x_k * cos(Omega_k) - y_k * cos(i_k) * sin(Omega_k);
y_k1 = x_k * sin(Omega_k) + y_k * cos(i_k) * cos(Omega_k);
z_k1 = y_k * sin(i_k);
coord[0] = x_k1;
coord[1] = y_k1;
coord[2] = z_k1;

}
```

Rasch_coord.h

```
#ifndef Rasch_coord_H
#define Rasch_coord_H

void coord(long double e, long double mu, long double Omega_e, long double toe,
        long double A, long double n0, long double i0, long double Omega0, long double omega, long double M0,
        long double Dn, long double n, long double OmegaDot, long double IDOT, long double Crs, long double Cuc, long double Cus,
        long double Cic, long double Cis, long double Crc, long double t, long double* coord);

#endif
```