**Национальный исследовательский университет**

**«МЭИ»**

**Институт радиотехники и электроники**

**Кафедра радиотехнических систем**

Курсовой проект

по дисциплине

Аппаратура потребителей спутниковых радионавигационных систем

ФИО студента: Калугин К.С.

Группа: ЭР-15-15

Вариант №:12

Дата:­ ­

Подпись:­ ­

ФИО преподавателя: Корогодин И.В.

Оценка: ­ ­

**Москва, 2020**

Содержание

[Введение 3](#_Toc43687347)

[1 Использование сторонних средств 3](#_Toc43687348)

[1.1 Описание процесса использования RTKLIB 3](#_Toc43687349)

[1.2 Получение графика угла места и SkyView с помощью Trimble GNSS Planning 7](#_Toc43687350)

[2 Моделирование 9](#_Toc43687351)

[2.1 Алгоритм расчета положения спутника ГЛОНАСС 10](#_Toc43687352)

[2.2 Результаты моделирования положения спутника ГЛОНАСС 12](#_Toc43687353)

[2.3 Построение SkyView 13](#_Toc43687354)

[3 Реализация 14](#_Toc43687355)

[3.1 Особенности реализации 15](#_Toc43687356)

[3.2 Результаты реализации 15](#_Toc43687357)

[Заключение 16](#_Toc43687358)

[Приложение 17](#_Toc43687359)

**Введение**

Название проекта: Разработка модуля расчёта координат спутника ГЛОНАСС.

Техническая цель - добавление в программное обеспечение приемника функции расчета положения спутника ГЛОНАСС на заданное время по данным его эфемерид.

Конечная цель проекта - получить библиотечные функции на С++, позволяющие рассчитывать положение спутника ГЛОНАСС по эфемеридам.

Для достижения цели выполняется ряд задач:

* обработка данных от приемника ГНСС в RTKLIB для проверки входных данных и формирования проверочных значений;
* обработка данных и моделирование в Matlab/Python для эскизного проектирования модуля;
* реализация программного модуля на С/С++, включая юнит-тестирование в Check.

Требования:

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

Курсовой проект разбит на три этапа, отличающиеся осваиваемыми инструментами.

**1 Использование сторонних средств**

**1.1 Описание процесса использования RTKLIB**

На крыше корпуса Е МЭИ установлена трехдиапазонная антенна Harxon HX-CSX601A. 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.

Приемники осуществляют первичную обработку сигналов, выдавая по интерфейсам соответствующие потоки данных - наблюдения псевдодальностей и эфемериды спутников.

Необходимо обрабатывать данные от приемника Clonicus, представленные в бинарном виде в формате NVS BINR. Для этого воспользуемся пакетом RTKLIB, в состав которого входит парсер формата NVS BINR и удобные средства отображения данных.

При запуске программы RTKLIB получаем следующее окно (Рисунок 1):



Рисунок 1 – Окно программы RTKLIB v.2.4.2

В окне программы RTKLIB выбираем RTKCONV (Рисунок 2), чтобы конвертировать бинарный файл BINR.bin в текстовый формат NVS BINR.

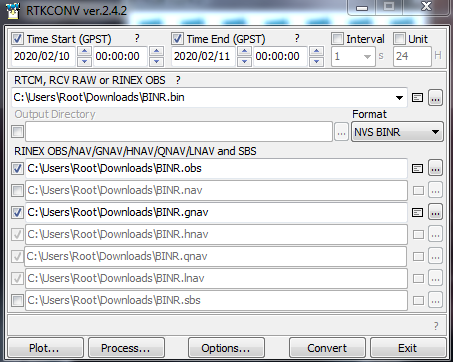


Рисунок 2 – Окно программы RTKCONV ver.2.4.2

В открывшемся окне выбираем Time Start (GPST), Time End (GPST), и ставим время интервала наблюдений с 00:00 10.02.20 до 00:00 11.02.20. В меню «Options» (Рисунок 3) выбираем спутниковую систему ГЛОНАСС и указываем в поле «Excluded Satellite» следующее: R3, R4, R5, R11, R13, R14, R21, R22, R23, тем самым исключая данные спутники из обработки. В первой строке RTKCONV указываем путь на файл бинарного потока .bin, указываем формат NVS BINR, и ставим галочки для конвертации файлов в форматы .obs и .gnav.

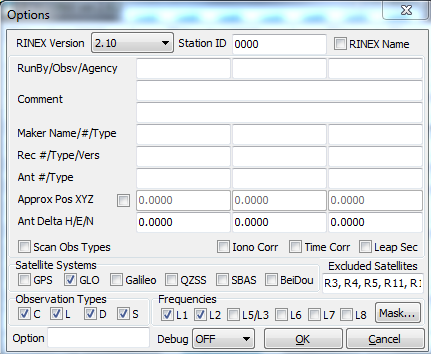


Рисунок 3 – Окно настроек программы RTKCONV ver.2.4.2

Затем нажимаем на кнопку «Convert» и получаем необходимые файлы. Для того, чтобы посмотреть содержимое открываем файл с расширением «gnav» и получаем эфемериды собственного спутника в gnav-файле RINEX (Рисунок 4).

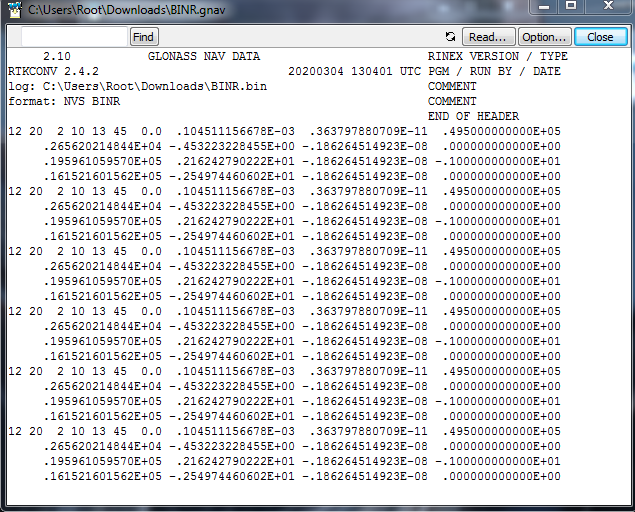


Рисунок 4 – Эфемериды спутника ГЛОНАСС №12 в .gnav файле

Для того, чтобы получить эфемериды собственного спутника по данным RTKNAVI из состава RTKLIB, в окне программы RTKLIB выбираем программу RTKNAVI, которая позволяет вывести таблицу текущих и предыдущих эфемерид (Рисунок 5).

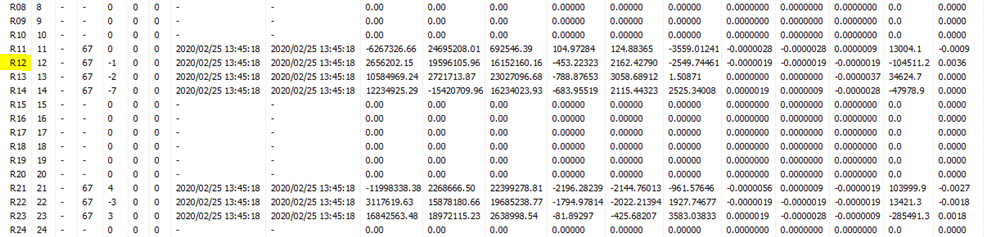


Рисунок 5 – Окно программы RTKNAVI ver.2.4.2

**1.2 Получение графика угла места и SkyView с помощью Trimble GNSS Planning**

Для построения графика угла места собственного спутника от времени по данным Trimble GNSS Planning Online на заданный интервал времени, во вкладке настроек (Settings) указываем координаты корпуса Е МЭИ и время, когда производились замеры (Рисунок 6). Во вкладке библиотеки спутников (Satellite Library) отключаем отображение всех спутников, кроме заданного (Рисунок 7).

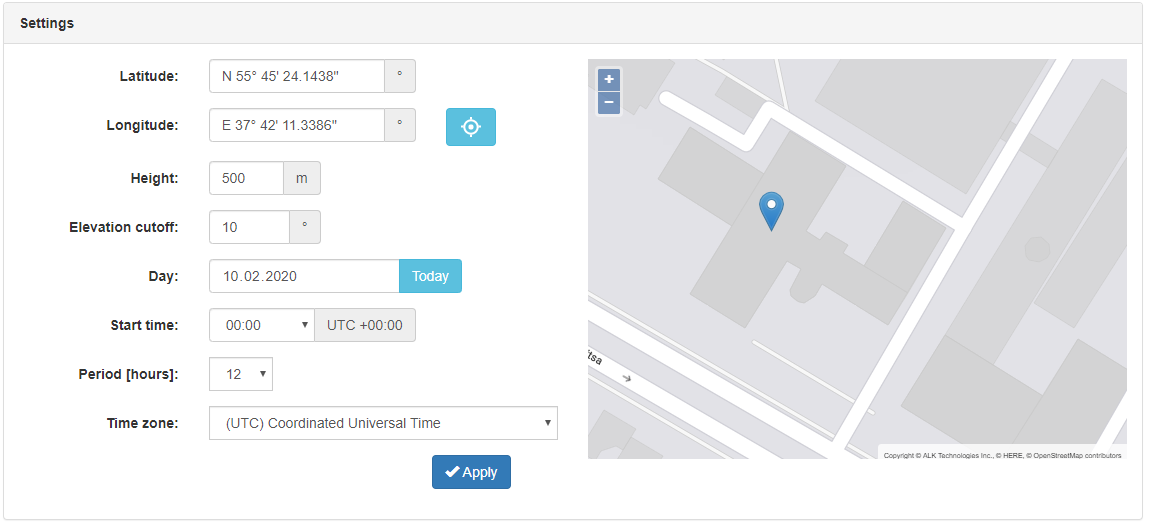


Рисунок 6 – Вкладка настроек (Settings) Trimble GNSS Planning

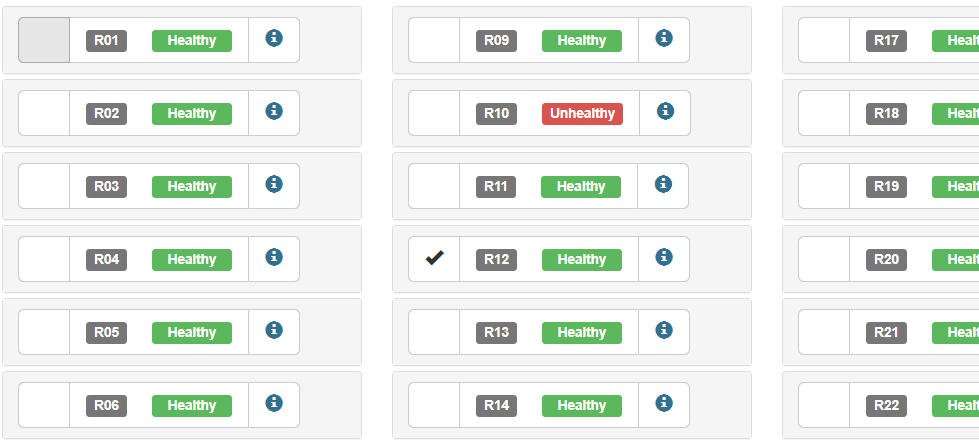


Рисунок 7 – Вкладка библиотеки спутников (Satellite Library) Trimble GNSS Planning

Далее переходим во вкладку «Charts» и получаем график угла места спутника (Рисунок 8). Полученные данные говорят нам о том, что спутник было видно 2 раза. Первое появление с 00:10 до 01:40, второе с 9:40 до 12:00.

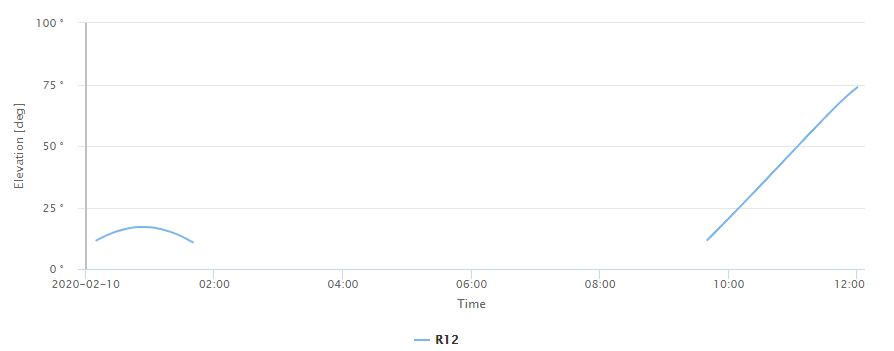


Рисунок 8 – График угла места спутника ГЛОНАСС №12

Перейдя во вкладку «Sky Plot», получаем карту небосвода (SkyView). Траектория движения спутника (Рисунок 9) соответствует его первому появлению. Траектория движения спутника (Рисунок 10) соответствует его второму появлению.

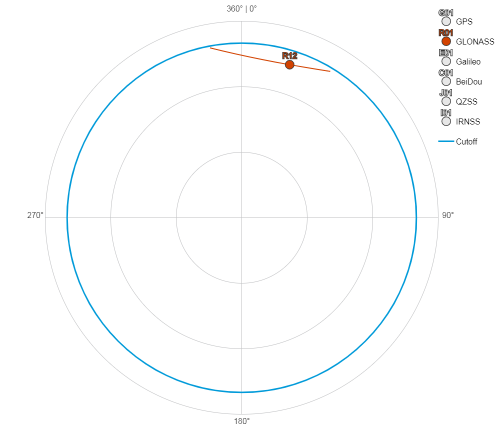


Рисунок 9 – SkyView спутника ГЛОНАСС №12

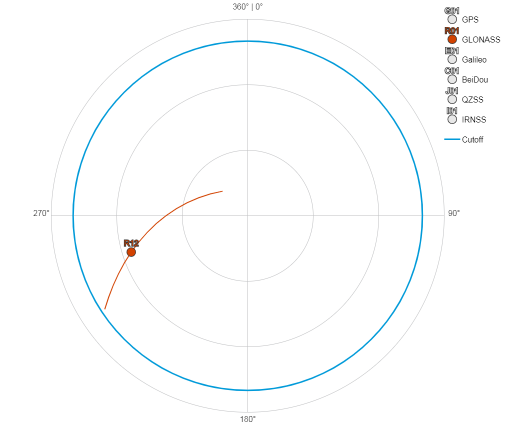


Рисунок 10 – SkyView спутника ГЛОНАСС №12

# 2 Моделирование

На предыдущем этапе было получено решение навигационной задачи с помощью программы вторичной обработки измерений – RTKLIB. В процессе работы она рассчитывает положение спутников на соответствующий момент сигнального времени. При этом используются эфемериды – параметры некоторой модели движения спутника. Для расчета положения спутника ГЛОНАСС по эфемеридным данным системы проводят численное интегрирование дифференциального уравнения.

Требуется реализовать на языке Matlab или Python функцию расчета положения спутника ГЛОНАСС на заданный момент по шкале времени UTC. В качестве эфемерид использовать данные, полученные на предыдущем этапе (Рисунок 11).

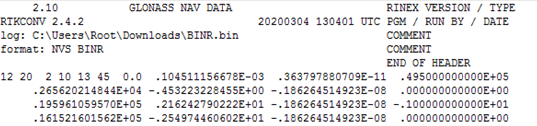
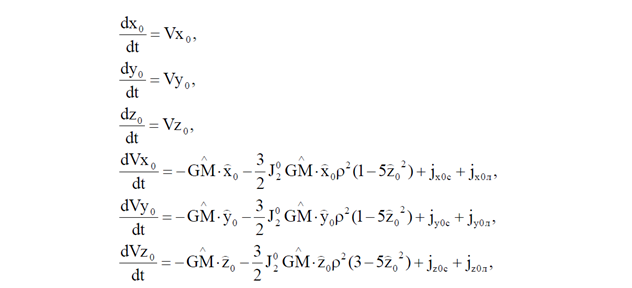


Рисунок 11 – Эфемериды, полученные на предыдущем этапе

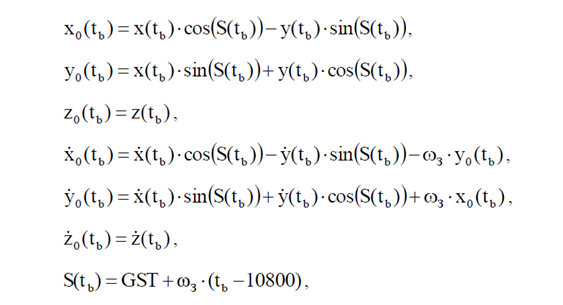
# 2.1 Алгоритм расчета положения спутника ГЛОНАСС

В ИКД ГЛОНАСС приведены алгоритмы пересчета координат и составляющих вектора скорости центра масс НКА на заданный момент времени по шкале МДВ, однако, эфемериды передаются в шкале времени UTC, поэтому необходимо добавить 3 часа, чтобы перевести время эфемерид в шкалу времени МДВ.

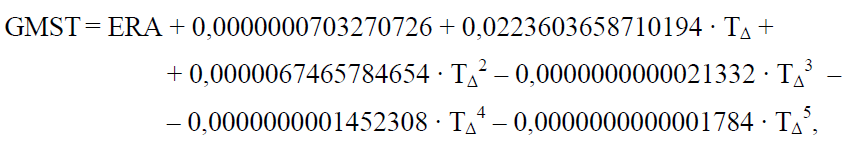
Пересчет эфемерид потребителем с момента шкалы МДВ на заданный момент времени той же шкалы проводится методом численного интегрирования дифференциальных уравнений движения центра масс НКА. В правых частях этих уравнений учитываются ускорения, определяемые геоцентрической константой гравитационного поля Земли с учетом атмосферы GM, зональным гармоническим коэффициентом второй степени , характеризующим полярное сжатие Земли, а также ускорениями от лунно-солнечных гравитационных возмущений. Эти уравнения движения определены в виде следующей системы:



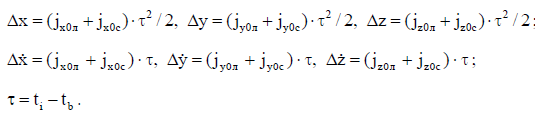
Начальными условиями для интегрирования системы являются координаты центра масс НКА , , и составляющие его вектора скорости , , в инерциальной геоцентрической системе координат на момент шкалы МДВ. Эти начальные условия вычисляются путем пересчета передаваемых в навигационном сообщении координат , , и составляющих вектора скорости , , центра масс НКА в связанной с Землей системе координат ПЗ-90. Пересчет осуществляется по следующим формулам:



Вместо истинного звездного времени по Гринвичу GST, в формулах допускается использовать среднее звездное время по Гринвичу GMST вычисляемое по формуле:

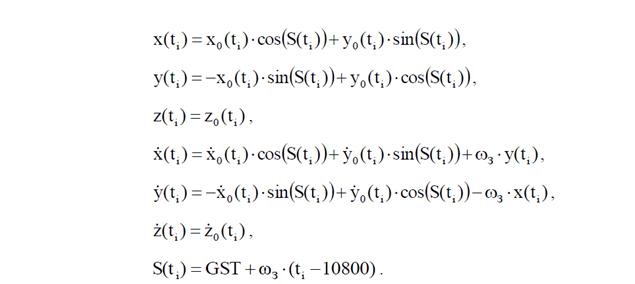


Ускорения солнечно-лунных гравитационных возмущений могут быть исключены с последующим добавлением к результатам интегрирования поправок:



Интегрирование осуществляется численным методом, например, методом Рунге- Кутта 4-го порядка.

После интегрирования, полученные в инерциальной системе координат координаты центра масс и составляющие его вектора скорости, могут быть пересчитаны в связанную с Землей систему ПЗ-90 по формулам:



# 2.2 Результаты моделирования положения спутника ГЛОНАСС

В программе Matlab реализована функция расчета положения спутника ГЛОНАСС на заданный момент времени. Построены трехмерные графики множества положений спутника ГЛОНАСС №12.



Рис 12 – Траектория движения спутника ГЛОНАСС №12 в системе координат ПЗ-90 (синяя линия) и в инерциальной системе координат (зеленая линия)

# 2.3 Построение SkyView

Для построения SkyView переводим координаты местоположения приемника в радианы. Для этого нам необходимо знать долготу и широту точки, в которой находится приемник, то есть корпуса Е.

Долгота - 55 градусов 45 минут 24 секунды

Широта - 37 градусов 42 минуты 11 секунд



Рисунок 13 – SkyView спутника ГЛОНАСС №12

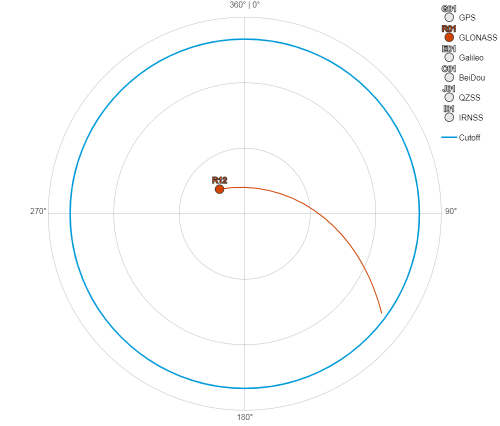


Рисунок 14 – SkyView, полученный в GNSS Planning Online

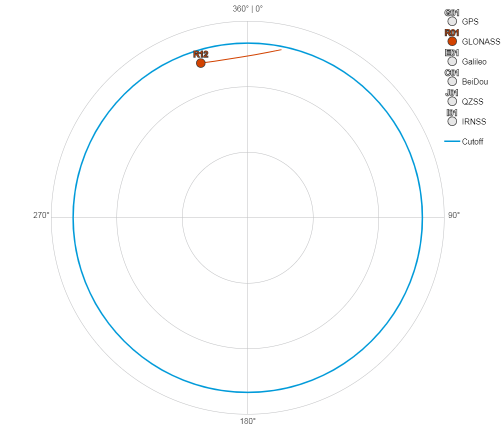


Рисунок 15 – SkyView, полученный в GNSS Planning Online

По SkyView видно, что спутник появлялся в зоне видимости приемника два раза. Данные результаты совпадают с данными, полученными в Trimble GNSS Planning Online, но имеют все же некоторое отличие. Это связано с тем, что мы пользуемся точным алгоритмом на 30-минутном интервале, для получения более точных графиков необходимо использовать долговременный алгоритм.

# 3 Реализация

Требуется разработать на языке С/С++ функцию расчета положения спутника ГЛОНАСС на заданное время по шкале UTC, минимизируя время её исполнения и количество затрачиваемой оперативной памяти. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал (как на предыдущем этапе).

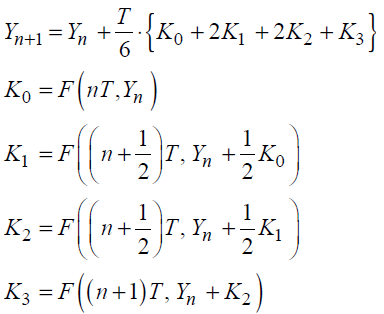
Программный модуль должен сопровождаться unit-тестами под check:

* Тесты функции реализации метода Рунге-Кутты
* Тест расчетного положения спутника в сравнении с Matlab/Python с шагом 0.1 секунды.

# 3.1 Особенности реализации

Функция расчета положения спутника в Matlab/Python 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. Поэтому отобразить модель из Matlab/Python в прошивку приемника дословно, как правило, не получается. В рассматриваемом примере потребуется, как минимум, выполнить свою реализацию решения дифура методом Рунге-Кутты.

Метод Рунге-Кутта 4-го порядка:



# 3.2 Результаты реализации

На данном этапе была реализована на языке С++ функция расчета положения спутника ГЛОНАСС №12 на заданное время по шкале UTC. Код программы приведен в приложении. Время выполнения расчета функции составило приближенно 287 мс. Для расчета в Matlab требуется больше времени, т.е. относительно модели, реализуемая функция минимизирует время расчета.

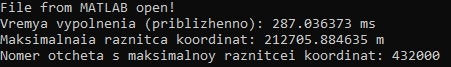


Рис 16 – Полученные результаты

В файле matlab\_results.txt содержатся координаты спутника на заданном отрезке времени, которые в дальнейшем используются для сравнения со значениями, полученными в результате расчета на языке C++. Функция выводит максимальное значение рассогласования координат (Рисунок 16).

Такая разница результатов может быть вызвана разной точностью расчетов в Matlab и C++. В Matlab мы пользуемся встроенной функцией, которая реализует метод Рунге-Кутта 4 и 5 порядков, что дает очень хорошие результаты. Возможно, для получения более схожих результатов, следует реализовать метод Рунге-Кутта в Matlab самостоятельно.

# Заключение

Нами были получены следующие результаты:

* Эфемериды собственного спутника по данным RTKNAVI из состава RTKLIB;
* Эфемериды собственного спутника в gnav-файле RINEX;
* График угла места от времени и SkyView собственного спутника по данным Trimble GNSS Planning Online на заданный интервал времени;
* Обработаны данные от приемника ГНСС в RTKLIB для проверки входных данных и формирования проверочных значений.
* Реализация на языке Matlab функции расчета положения спутника ГЛОНАСС на заданный момент по шкале времени UTC;
* Трехмерные графики множества положений спутника ГЛОНАСС с системным номером 12 в двух вариантах: в СК ECEF ПЗ-90.11 и соответствующей ей инерциальной СК;
* График SkyView, соответствующий временному интервалу  с 12:00 10.02.20 до 00:00 11.02.20.
* Реализация на языке С++ функции расчета положения спутника ГЛОНАСС на заданное время по шкале UTC;
* Погрешности вычислений между функцией на С++ и моделью в Matlab.

# Приложение

**Файл main.cpp**

#include <iostream>

#include <fstream>

#include <include\libglnsvpos\glnsvpos.h>

#include <include\libglnsvpos\rungekutta.h>

#include <ctime>

using namespace std;

int main()

{

time\_t start, end;

double del\_t = 1E-01;

double max\_del = 0;

int i\_max = 0;

int n = (int) 12\*3600/del\_t +1;

double \*\*koord\_raschet = new double \* [n];

for (int i = 0; i < n; i++)

{

koord\_raschet[i] = new double [6];

}

double \*koord\_file = new double[3];

ofstream out;

out.open("D:\\cpp\_results.txt");

time(&start);

ifstream in("D:\\matlab\_results.txt");

if (!in)

{

cout << "ERORR: File from MATLAB not open!" << endl;

} else {

cout << "File from MATLAB open!" << endl;

}

koordinate\_GLONASS(koord\_raschet);

for (int i = 0; i < n; i++)

{

in >> koord\_file[0] >> koord\_file[1] >> koord\_file[2];

string koord\_str1 = to\_string(koord\_raschet[i][0]);

string koord\_str2 = to\_string(koord\_raschet[i][1]);

string koord\_str3 = to\_string(koord\_raschet[i][2]);

out << koord\_str1 << "\t" << koord\_str2 << "\t" << koord\_str3 << endl;

for (int j = 0; j < 3; j++)

{

if (abs(koord\_raschet[i][j] - koord\_file[j]) > max\_del)

{

max\_del = abs(koord\_raschet[i][j] - koord\_file[j]);

i\_max = i;

}

}

delete [] koord\_raschet[i];

koord\_raschet[i] = nullptr;

}

time(&end);

in.close();

out.close();

delete[] koord\_raschet;

koord\_raschet = nullptr;

delete[] koord\_file;

koord\_file = nullptr;

double seconds = difftime(end, start);

string seconds1 = to\_string(seconds\*1000000/n);

cout << "Vremya vypolnenia (priblizhenno): " << seconds1 << " ms" << endl;

string max\_del1 = to\_string(max\_del);

cout << "Maksimalnaia raznitca koordinat: " << max\_del1 << " m" << endl;

string imax = to\_string(i\_max);

cout << "Nomer otcheta s maksimalnoy raznitcei koordinat: " << imax << endl;

}

**Файл diffs.h**

#ifndef DIFFS\_H

#define DIFFS\_H

void dif(double \*koord, double \*diffs);

#endif // DIFFS\_H

**Файл diffs.cpp**

#include <cmath>

#include <include\libglnsvpos\diffs.h>

void dif(double \*koord, double \*diffs)

{

double J\_2 = 1082625.75e-9; //зональный гармонический коэффициент второй степени, характеризующий полярное сжатие Земли

double GM = 398600.4418e9; //геоцентрическая константа гравитационного поля Земли с учетом атмосферы

double a\_e = 6378136; //большая полуось общеземного эллипсоида

double r = sqrt(koord[0]\*koord[0] + koord[1]\*koord[1] + koord[2]\*koord[2]);

diffs[0] = koord[3];

diffs[1] = koord[4];

diffs[2] = koord[5];

double GM0 = GM/(r\*r);

double x0 = koord[0]/r;

double y0 = koord[1]/r;

double z0 = koord[2]/r;

double R = a\_e/r;

diffs[3] = -GM0\*x0 - 1.5\*J\_2\*GM0\*x0\*R\*R\*(1-5\*z0\*z0);

diffs[4] = -GM0\*y0 - 1.5\*J\_2\*GM0\*y0\*R\*R\*(1-5\*z0\*z0);

diffs[5] = -GM0\*z0 - 1.5\*J\_2\*GM0\*z0\*R\*R\*(3-5\*z0\*z0);

}

**Файл glnsvpos.h**

#ifndef GLNSVPOS\_H

#define GLNSVPOS\_H

void koordinate\_GLONASS(double \*\*koord\_n);

#endif /\* #ifndef GLNSVPOS\_H \*/

**Файл glnsvpos.cpp**

#include <include\libglnsvpos\glnsvpos.h>

#include <include\libglnsvpos\rungekutta.h>

#include <iostream>

#include <cmath>

#include <ostream>

using namespace std;

void koordinate\_GLONASS(double \*\*koord\_n)

{

double \*koord\_0 = new double [6];

// начальные условия

koord\_0[0] = 2656202.15;

koord\_0[1] = 19596105.96;

koord\_0[2] = 16152160.15;

koord\_0[3] = -453.223228;

koord\_0[4] = 2162.4279;

koord\_0[5] = -2549.744606;

double Ax = -0.0000019;

double Ay = -0.0000019;

double Az = -0.0000019;

double w\_e = 7.2921151467e-5; //средняя угловая скорость вращения Земли

//Расчет времени формата ГЛОНАСС

//Дата: 10.02.20 Время: 13:45:18

double h = 0.1;

double N4 = (2020-1996)/4.0 + 1; //Номер четырехлетнего периода

double Nt = 31 + 25 + 1; //Текущие сутки от начала года

double tb = 18 + 45\*60 + 13\*60\*60 + 10800; //Момент времени по шкале МДВ

double t\_start = 12; //Время начала прогноза

double t\_end = 24; //Время окончания

double T\_start = (t\_start + 3) \* 60 \* 60; //по шкале МДВ

double T\_end = (t\_end + 3) \* 60 \* 60; //по шкале МДВ

double JD0 = 1461 \* (N4 - 1) + Nt + 2450082.5 - (Nt - 3)/25.0;

double del\_T = (JD0 - 2451545)/36525.0;

double ERA = 2\*M\_PI \* (0.7790572732640 + 1.00273781191135448 \* (JD0 - 2451545));

double GMST = ERA + 0.0000000703270726 + 0.0223603658710194 \* del\_T + 0.0000067465784654 \* pow(del\_T,2) - 0.0000000000021332 \* pow(del\_T,3) - 0.0000000001452308 \* pow(del\_T,4) - 0.0000000000001784 \* pow(del\_T,5);

int num = (int) abs((T\_end - T\_start)/h)+1;

double \*ti = new double [num];

for (int i = 0; i < num; i++)

{

ti[i] = T\_start + i\*h;

}

int num\_eph = (int) (tb - T\_start)/h;

double S = GMST + w\_e \* (tb - 10800);

double cos\_S = cos(S);

double sin\_S = sin(S);

// пересчет координат из ПЗ-90 в ECI

koord\_n[num\_eph][0] = koord\_0[0]\*cos\_S - koord\_0[1]\*sin\_S;

koord\_n[num\_eph][1] = koord\_0[0]\*sin\_S + koord\_0[1]\*cos\_S;

koord\_n[num\_eph][2] = koord\_0[2];

koord\_n[num\_eph][3] = koord\_0[3]\*cos\_S - koord\_0[4]\*sin\_S - w\_e\*koord\_n[num\_eph][1];

koord\_n[num\_eph][4] = koord\_0[3]\*sin\_S + koord\_0[4]\*cos\_S + w\_e\*koord\_n[num\_eph][0];

koord\_n[num\_eph][5] = koord\_0[5];

double Ax0 = Ax \* cos\_S - Ay \* sin\_S;

double Ay0 = Ax \* sin\_S + Ay \* cos\_S;

double Az0 = Az;

for (int i = num\_eph; i > 0; i--)

{

RungeKutta(ti[i], ti[i-1], koord\_n[i], koord\_n[i-1]);

}

for (int i = num\_eph; i < num-1 ; i++)

{

RungeKutta(ti[i], ti[i+1], koord\_n[i], koord\_n[i+1]);

}

for (int i = 0; i < num; i++)

{

double tau = ti[i] - tb;

double del\_X = Ax0 \* tau \* tau/2.0;

double del\_Y = Ay0 \* tau \* tau/2.0;

double del\_Z = Az0 \* tau \* tau/2.0;

double del\_Vx = Ax0 \* tau;

double del\_Vy = Ay0 \* tau;

double del\_Vz = Az0 \* tau;

//Добавляем поправки к результатам интегрирования

koord\_n[i][0] += del\_X;

koord\_n[i][1] += del\_Y;

koord\_n[i][2] += del\_Z;

koord\_n[i][3] += del\_Vx;

koord\_n[i][4] += del\_Vy;

koord\_n[i][5] += del\_Vz;

}

delete[] koord\_0;

koord\_0 = nullptr;

delete[] ti;

ti = nullptr;

**Файл rungekutta.h**

#ifndef RUNGEKUTTA\_H

#define RUNGEKUTTA\_H

void RungeKutta(double t0, double tn, double \*koord0, double \*koord);

#endif /\* #ifndef RUNGEKUTTA\_H \*/

**Файл rungekutta.cpp**

#include <include\libglnsvpos\rungekutta.h>

#include <cmath>

#include <include\libglnsvpos\diffs.h>

#include <iostream>

#include <fstream>

using namespace std;

void RungeKutta(double t0, double tn, double \*koord0, double \*koord){

double del\_t = tn-t0;

double \*proizv = new double [6];

double \*K\_1 = new double [6];

double \*K\_2 = new double [6];

double \*K\_3 = new double [6];

double \*K\_4 = new double [6];

double \*vremen = new double [6];

// Rschet K1

dif(koord0, proizv);

for (int j = 0; j<=5; j++){

K\_1[j] = del\_t\*proizv[j];

vremen[j] = koord0[j] + K\_1[j]\*0.5;

}

// Rschet K2

dif(vremen, proizv);

for (int j = 0; j<=5; j++){

K\_2[j] = del\_t\*proizv[j];

vremen[j] = koord0[j] + K\_2[j]\*0.5;

}

// Rschet K3

dif(vremen, proizv);

for (int j = 0; j<=5; j++){

K\_3[j] = del\_t\*proizv[j];

vremen[j] = koord0[j] + K\_3[j];

}

// Rschet K4

dif(vremen, proizv);

for (int j = 0; j<=5; j++){

K\_4[j] = del\_t\*proizv[j];

koord[j] = koord0[j] + (K\_1[j] + 2.0\*K\_2[j] + 2.0\*K\_3[j] + K\_4[j])/6.0;

}

delete[] K\_1;

K\_1 = nullptr;

delete[] K\_2;

K\_2 = nullptr;

delete[] K\_3;

K\_3 = nullptr;

delete[] K\_4;

K\_4 = nullptr;

delete[] proizv;

proizv = nullptr;

delete[] vremen;

vremen = nullptr;

}