**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.

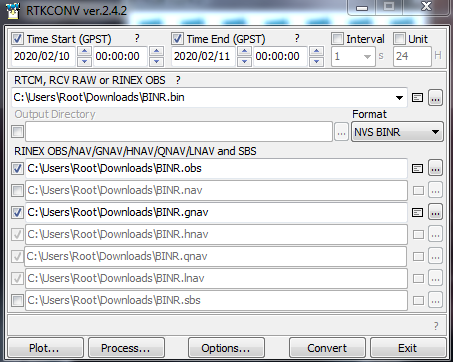


Рисунок 2 – Окно программы RTKCONV ver.2.4.2

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.

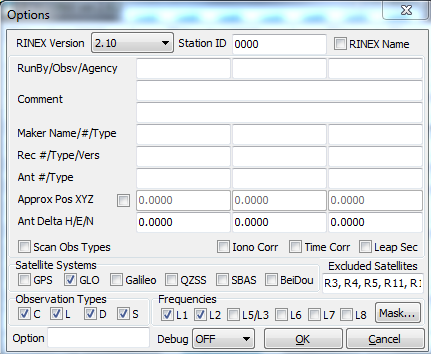


Рисунок 3 – Окно настроек программы RTKCONV ver.2.4.2

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).

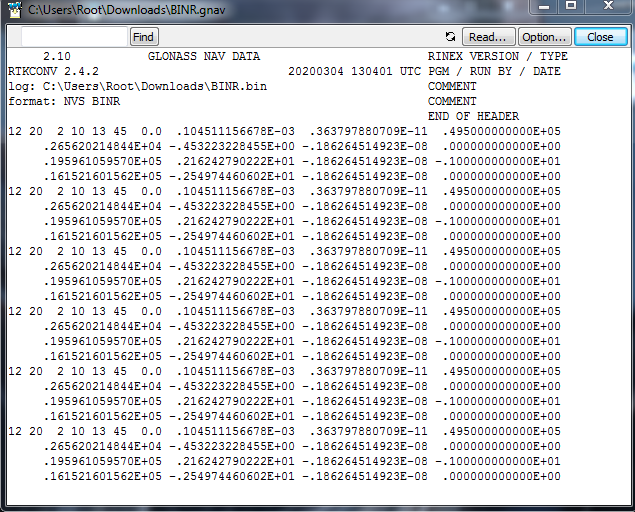


Рисунок 4 – Эфемериды спутника ГЛОНАСС №12 в .gnav файле

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).

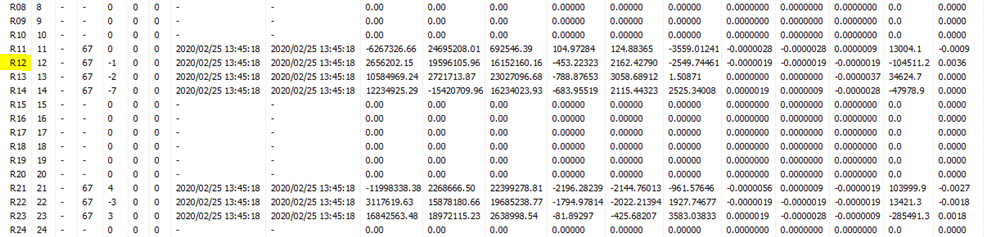


Рисунок 5 – Окно программы RTKNAVI ver.2.4.2

**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).

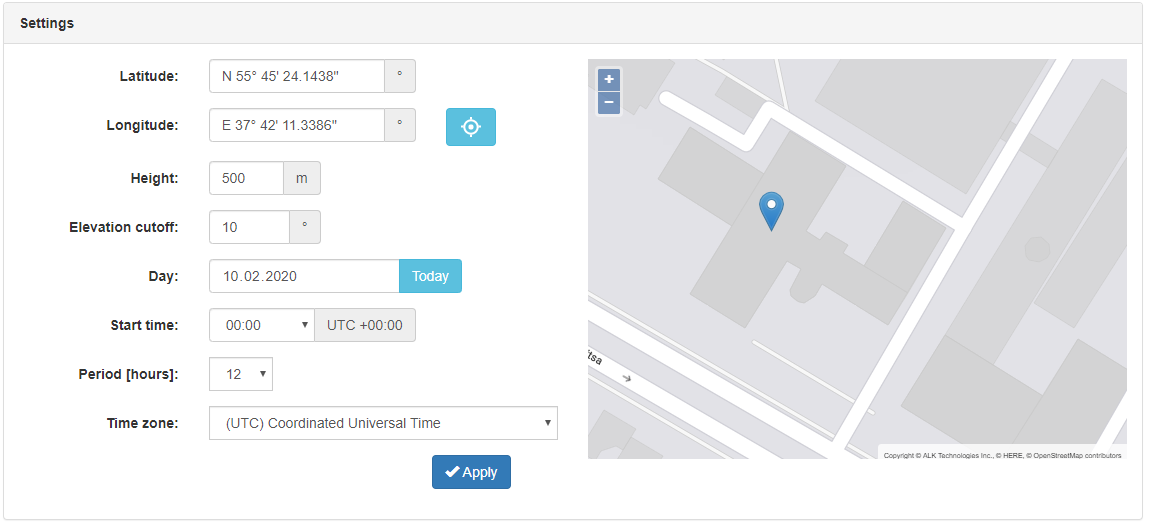


Рисунок 6 – Вкладка настроек (Settings) Trimble GNSS Planning

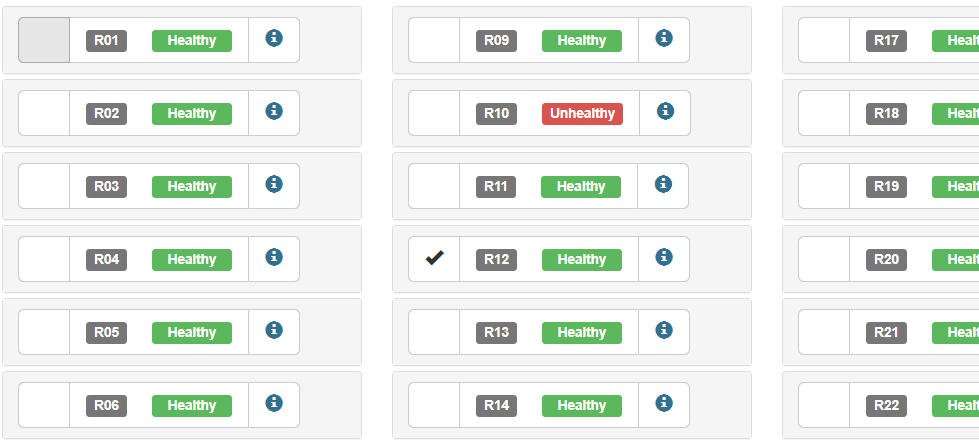


Рисунок 7 – Вкладка библиотеки спутников (Satellite Library) Trimble GNSS Planning

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.

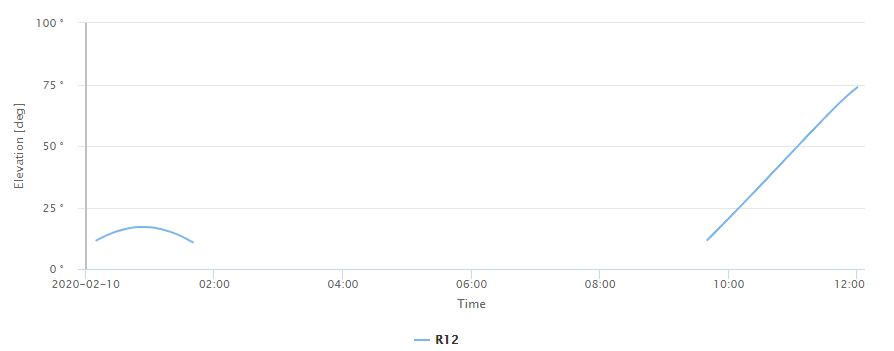


Рисунок 8 – График угла места спутника ГЛОНАСС №12

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.

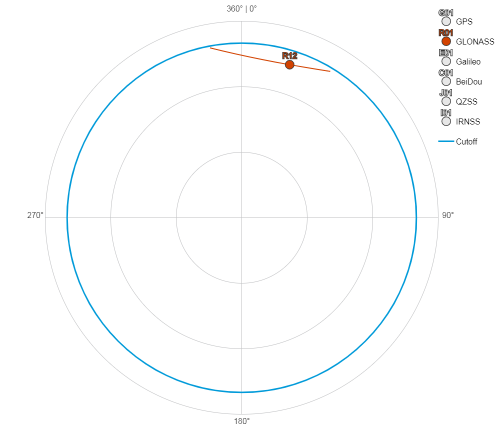


Рисунок 9 – SkyView спутника ГЛОНАСС №12

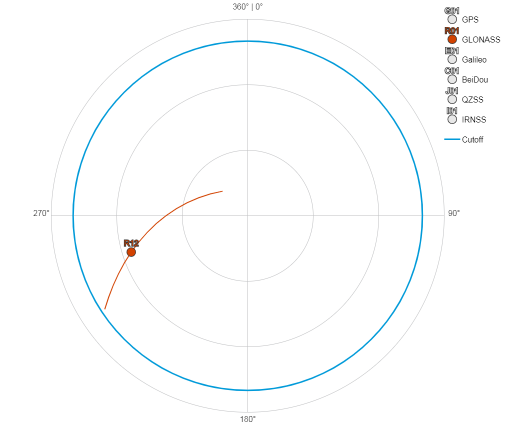


Рисунок 10 – SkyView спутника ГЛОНАСС №12

# 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).

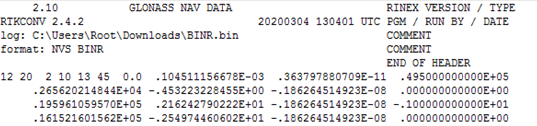
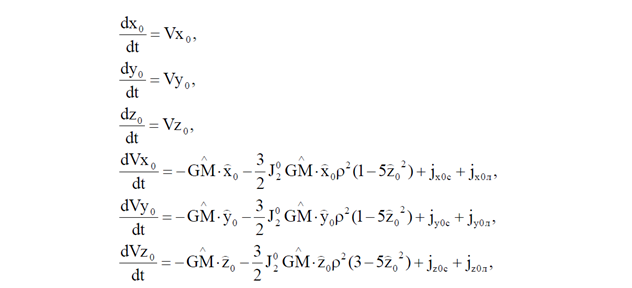


Рисунок 11 – Эфемериды, полученные на предыдущем этапе

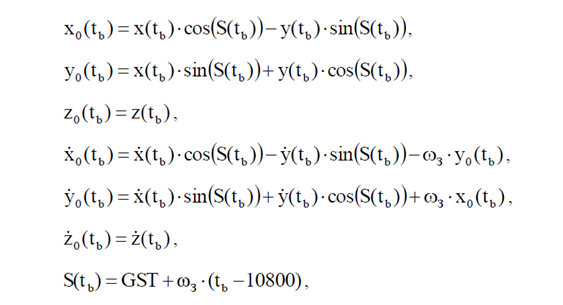
# 2.1 Алгоритм расчета положения спутника ГЛОНАСС

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.

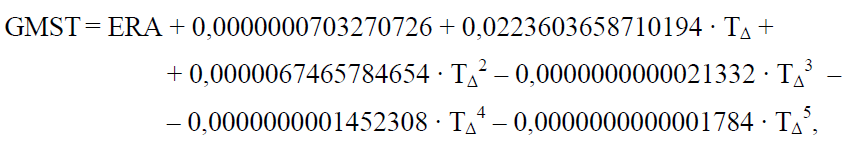
Пересчет эфемерид потребителем с момента шкалы МДВ на заданный момент времени той же шкалы проводится методом численного интегрирования дифференциальных уравнений движения центра масс НКА. В правых частях этих уравнений учитываются ускорения, определяемые геоцентрической константой гравитационного поля 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:



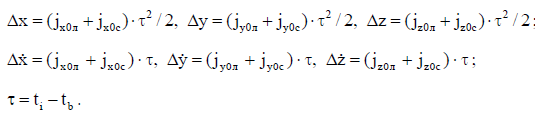
Начальными условиями для интегрирования системы являются координаты центра масс НКА , , и составляющие его вектора скорости , , в инерциальной геоцентрической системе координат на момент шкалы МДВ. Эти начальные условия вычисляются путем пересчета передаваемых в навигационном сообщении координат , , 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:

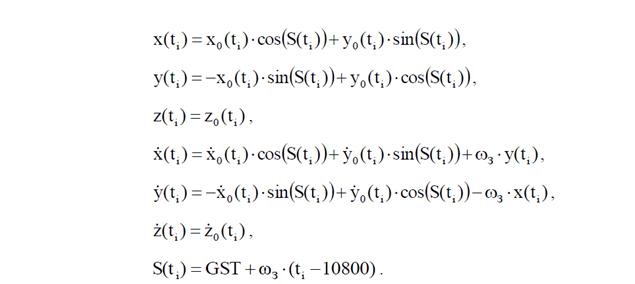


Ускорения солнечно-лунных гравитационных возмущений могут быть исключены с последующим добавлением к результатам интегрирования поправок:



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

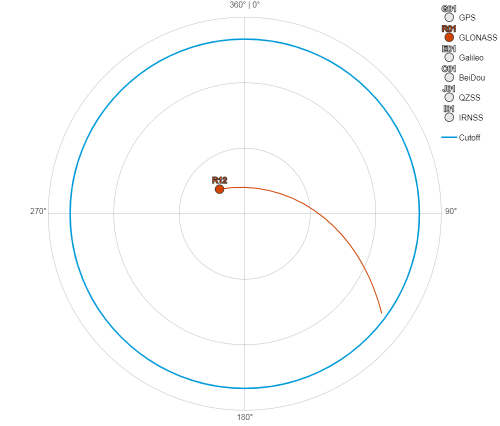


Рисунок 14 – SkyView, полученный в GNSS Planning Online

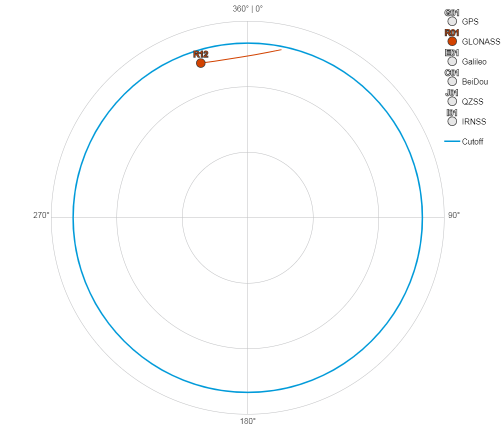


Рисунок 15 – SkyView, полученный в GNSS Planning Online

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).

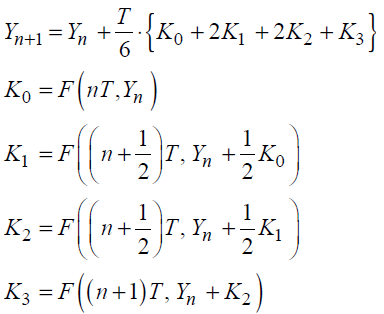
Программный модуль должен сопровождаться unit-тестами под check:

* 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. В файле matlab\_results\_RK.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. Результаты разницы координат и времени исполнения представлены на рисунке 16.

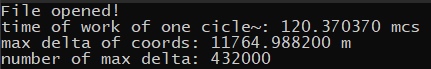


Рис 16 – Полученные результаты

Расхождение результатов может быть вызвано 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. Если исключить данную функцию из программы и реализовать метод Рунге-Кутта самостоятельно, то самое большое расхождение в координатах составит примерно 11 км в точке, максимально удаленной от времени эфемерид. На языке С++ программа работает быстрее, для расчета в Matlab требуется больше времени, т.е. относительно модели, реализуемая функция минимизирует время расчета.

Также на данном этапе была проведена проверка на утечки памяти. Результаты представлены на рисунке 17.

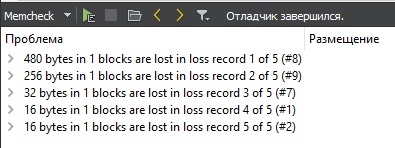


Рис 17 – Проверка на утечки памяти

# 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 <tuple>

#include <fstream>

#include <ctime>

#include <cmath>

#include <rk.h>

#include <integ.h>

using namespace std;

int main()

{

time\_t start, end;

//Исходные данные

//Заданные эфемериды:

//Координаты

double X = 2656202.15;

double Y = 19596105.96;

double Z = 16152160.15;

//Компоненты вектора скорости

double Vx = -453.223228;

double Vy = 2162.4279;

double Vz = -2549.744606;

//Лунно-солнечные ускорения

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

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

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

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

int T\_start = (12 + 3) \* 60 \* 60; //по шкале МДВ

int T\_end = (24 + 3) \* 60 \* 60; //по шкале МДВ

//Расчет среднего звездного времени по Гринвичу

double JD0 = 1461 \* (N4 - 1) + Nt + 2450082.5 - (Nt - 3.0)/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);

//Пересчет в инерциальную геоцентрическую систему координат

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

double sin\_S = sin(S);

double cos\_S = cos(S);

double X0 = X \* cos\_S - Y \* sin\_S;

double Y0 = X \* sin\_S + Y \* cos\_S;

double Z0 = Z;

double Vx0 = Vx \* cos\_S - Vy \* sin\_S - w\_e \* Y0;

double Vy0 = Vx \* sin\_S + Vy \* cos\_S + w\_e \* X0;

double Vz0 = Vz;

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

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

double Az0 = Az;

//Интегрирование численным методом

double h = 0.1; //задаем шаг

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

int num\_eph = (int) (tb - T\_start)/h+1;

double \*ti = new double [num+1];

double \*\*ff = new double \*[num+1];

for (int i = 0; i < num+1; i++){

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

ff[i] = new double [6];

}

//Вектор начальных состояний системы

double \*RK = new double [6] {X0, Y0, Z0, Vx0, Vy0, Vz0};

for (int i = 0; i < 6; i++){

ff[num\_eph][i] = RK[i];

}

time(&start);

for (int i = num\_eph-1; i >= 0; i--){

auto yn = RK4(ti[i], ti[i+1], ff[i+1]);

ff[i][0] = get<0>(yn);

ff[i][1] = get<1>(yn);

ff[i][2] = get<2>(yn);

ff[i][3] = get<3>(yn);

ff[i][4] = get<4>(yn);

ff[i][5] = get<5>(yn);

}

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

auto yn = RK4(ti[i], ti[i-1], ff[i-1]);

ff[i][0] = get<0>(yn);

ff[i][1] = get<1>(yn);

ff[i][2] = get<2>(yn);

ff[i][3] = get<3>(yn);

ff[i][4] = get<4>(yn);

ff[i][5] = get<5>(yn);

}

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

double tau = ti[i] - tb;

ff[i][0] += Ax0 \* tau\*tau\*0.5;

ff[i][1] += Ay0 \* tau\*tau\*0.5;

ff[i][2] += Az0 \* tau\*tau\*0.5;

ff[i][3] += Ax0 \* tau;

ff[i][4] += Ay0 \* tau;

ff[i][5] += Az0 \* tau;

}

time(&end);

ofstream out;

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

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

if (!in)

{

cout << "File not open, check file!" << endl;

} else {

cout << "File opened!" << endl;

}

double \* ff\_matlab = new double [3];

double max\_del = 0;

int i\_max = 0;

for (int i = 0; i < num+1; i++)

{

in >> ff\_matlab[0] >> ff\_matlab[1] >> ff\_matlab[2];

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

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

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

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

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

{

if (abs(ff[i][j] - ff\_matlab[j]) > max\_del)

{

max\_del = abs(ff[i][j] - ff\_matlab[j]);

i\_max = i;

}

}

}

time(&end);

in.close();

out.close();

delete [] ti;

ti = nullptr;

delete [] RK;

RK = nullptr;

for (int i = 0; i < num+1; i++)

{

delete [] ff[i];

ff[i] = nullptr;

}

delete [] ff;

ff = nullptr;

delete [] ff\_matlab;

ff\_matlab = nullptr;

double time\_RK = difftime(end, start);

string time\_RK1 = to\_string(time\_RK\*1000000/num);

cout << "time of work of one cicle~: " << time\_RK1 << " mcs" << endl;

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

cout << "max delta of coords: " << max\_del1 << " m" << endl;

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

cout << "number of max delta: " << imax << endl;

return 0;

}

**Файл integ.h**

#include<tuple>

using namespace std;

#ifndef INTEG\_H

#define INTEG\_H

tuple<double,double,double,double,double,double> integ(double \*f);

//void integ(double \*f, double \*dif);

#endif // INTEG\_H

**Файл integ.cpp**

#include<cmath>

#include<tuple>

#include<integ.h>

using namespace std;

//void integ(double \*f, double \*dif)

tuple<double,double,double,double,double,double> integ(double \*f)

{

double dX = f[3];

double dY = f[4];

double dZ = f[5];

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 = 6378136.0; //большая полуось общеземного эллипсоида

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

double xx = f[0]/r;

double yy = f[1]/r;

double zz = f[2]/r;

double Ro = a\_e/r;

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

double dVx = -GM0\*xx - (3.0/2)\*J\_2\*GM0\*xx\*Ro\*Ro\*(1-5\*zz\*zz);

double dVy = -GM0\*yy - (3.0/2)\*J\_2\*GM0\*yy\*Ro\*Ro\*(1-5\*zz\*zz);

double dVz = -GM0\*zz - (3.0/2)\*J\_2\*GM0\*zz\*Ro\*Ro\*(3-5\*zz\*zz);

return make\_tuple(dX,dY,dZ,dVx,dVy,dVz);

}

**Файл rk.h**

#include<tuple>

using namespace std;

#ifndef RK\_H

#define RK\_H

tuple<double,double,double,double,double,double> RK4(double tn, double t0, double \*y0);

//void RK4(double tn, double t0, double \*y0, double \*yn);

#endif // RK\_H

**Файл rk.cpp**

#include<integ.h>

#include<rk.h>

#include<tuple>

using namespace std;

//void RK4(double tn, double t0, double \*y0, double \*yn){

tuple<double,double,double,double,double,double> RK4(double tn, double t0, double \*y0){

double del\_t = tn - t0;

double \*K1 = new double[6];

double \*K2 = new double[6];

double \*K3 = new double[6];

double \*K4 = new double[6];

double \*fy1 = new double[6];

double \*fy2 = new double[6];

double \*fy3 = new double[6];

// Rschet K1

auto diff1 = integ(y0);

K1[0] = del\_t\*get<0>(diff1);

K1[1] = del\_t\*get<1>(diff1);

K1[2] = del\_t\*get<2>(diff1);

K1[3] = del\_t\*get<3>(diff1);

K1[4] = del\_t\*get<4>(diff1);

K1[5] = del\_t\*get<5>(diff1);

// integ(y0, K1);

for (int i = 0; i < 6; i++){

fy1[i] = y0[i] + K1[i]/2.0;

}

// Rschet K2

auto diff2 = integ(fy1);

K2[0] = del\_t\*get<0>(diff2);

K2[1] = del\_t\*get<1>(diff2);

K2[2] = del\_t\*get<2>(diff2);

K2[3] = del\_t\*get<3>(diff2);

K2[4] = del\_t\*get<4>(diff2);

K2[5] = del\_t\*get<5>(diff2);

// integ(fy, K2);

for (int i = 0; i < 6; i++){

fy2[i] = y0[i] + K2[i]/2.0;

}

// Rschet K3

auto diff3 = integ(fy2);

K3[0] = del\_t\*get<0>(diff3);

K3[1] = del\_t\*get<1>(diff3);

K3[2] = del\_t\*get<2>(diff3);

K3[3] = del\_t\*get<3>(diff3);

K3[4] = del\_t\*get<4>(diff3);

K3[5] = del\_t\*get<5>(diff3);

// integ(fy, K3);

for (int i = 0; i < 6; i++){

fy3[i] = y0[i] + K3[i];

}

// Rschet K4

auto diff4 = integ(fy3);

K4[0] = del\_t\*get<0>(diff4);

K4[1] = del\_t\*get<1>(diff4);

K4[2] = del\_t\*get<2>(diff4);

K4[3] = del\_t\*get<3>(diff4);

K4[4] = del\_t\*get<4>(diff4);

K4[5] = del\_t\*get<5>(diff4);

// integ(fy, K4);

double X = y0[0] + 1.0/6\*(K1[0] + 2.0\*K2[0] + 2.0\*K3[0] + K4[0]);

double Y = y0[1] + 1.0/6\*(K1[1] + 2.0\*K2[1] + 2.0\*K3[1] + K4[1]);

double Z = y0[2] + 1.0/6\*(K1[2] + 2.0\*K2[2] + 2.0\*K3[2] + K4[2]);

double Vx = y0[3] + 1.0/6\*(K1[3] + 2.0\*K2[3] + 2.0\*K3[3] + K4[3]);

double Vy = y0[4] + 1.0/6\*(K1[4] + 2.0\*K2[4] + 2.0\*K3[4] + K4[4]);

double Vz = y0[5] + 1.0/6\*(K1[5] + 2.0\*K2[5] + 2.0\*K3[5] + K4[5]);

delete[] K1;

K1 = nullptr;

delete[] K2;

K2 = nullptr;

delete[] K3;

K3 = nullptr;

delete[] K4;

K4 = nullptr;

delete[] fy1;

fy1 = nullptr;

delete[] fy2;

fy2 = nullptr;

delete[] fy3;

fy3 = nullptr;

return make\_tuple(X,Y,Z,Vx,Vy,Vz);

}