**Национальный исследовательский университет**

**«МЭИ»**

**Институт радиотехники и электроники**

**Кафедра радиотехнических систем**

Курсовая работа

по дисциплине

Аппаратура потребителей спутниковых радионавигационных систем

ФИО студента: Борисов В.А.

Группа: ЭР-15-15

Вариант №:3

Дата:­ ­

Подпись:­ ­

ФИО преподавателя: Корогодин И.В.

Оценка: ­ ­

**Москва, 2020**

Содержание

[Введение 3](#_Toc41996039)

[Этап 1. Использование сторонних средств 3](#_Toc41996040)

[1.1 Описание процесса использования RTKLIB 3](#_Toc41996041)

[1.2 Получение графика угла места и SkyView с помощью Trimble GNSS Planning 6](#_Toc41996042)

[Этап 2. Моделирование 9](#_Toc41996043)

[2.1 Алгоритм расчёта положения спутника ГЛОНАСС 9](#_Toc41996044)

[2.2 Результаты моделирования в Matlab 11](#_Toc41996045)

[2.3 Построение SkyView в Matlab 12](#_Toc41996046)

[Этап 3. Реализация 14](#_Toc41996047)

[3.1. Реализация кода на С++ 14](#_Toc41996048)

[4 Заключение 15](#_Toc41996049)

[5 Литература 16](#_Toc41996050)

[ПРИЛОЖЕНИЕ 1 17](#_Toc41996051)

[ПРИЛОЖЕНИЕ 2 20](#_Toc41996052)

## **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.

# **Этап 1. Использование сторонних средств**

# **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.

Нужно обработать 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.

Скачиваем архив RTKLIB\_bin-master.zip по ссылке, указанной на SRNS, и запускаем rtklaunch.exe, затем открывается следующее окно (Рисунок 1):



Рисунок 1 – Окно выбора программ из пакета RTKLIB

Выбираем RTKNAVI, куда закидываем бинарный файл BINR.bin, записанный днем 10.02.2020. Программа RTKNAVI позволяет вывести таблицу текущих и предыдущих эфемерид (Рисунок 2).

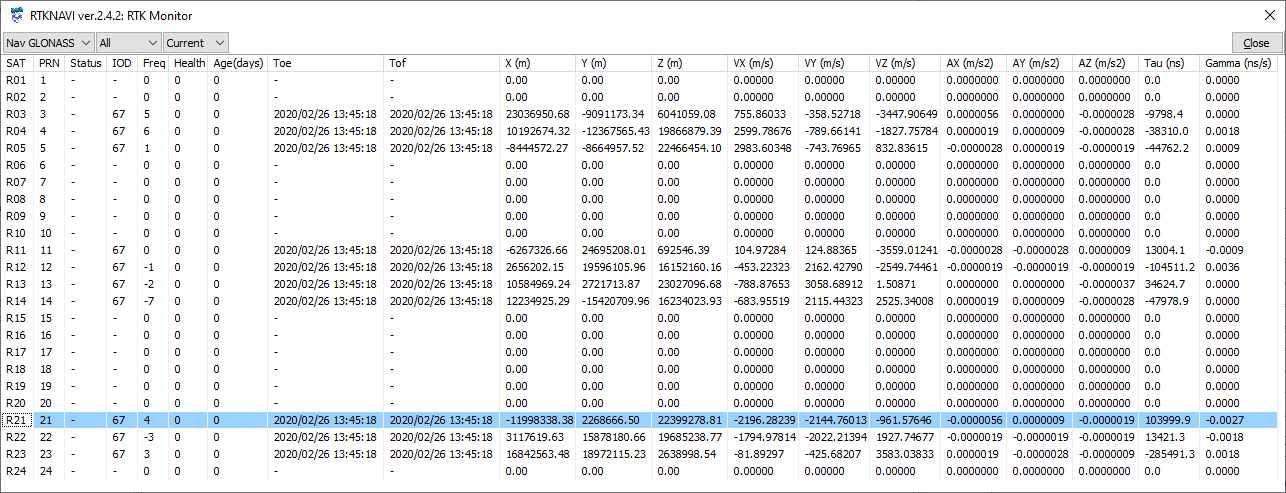


Рисунок 2 – Окно программы RTKNAVI ver.2.4.2

Последняя строка соответствует спутнику ГЛОНАСС №3, указанному в варианте.

Теперь конвертируем бинарный файл BINR.bin в текстовый формат NVS BINR. Для этого выбираем RTKCONV (Рисунок 3).

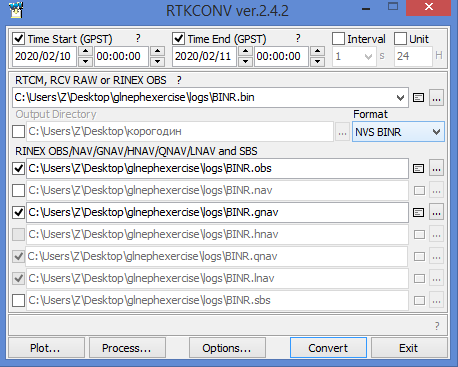


Рисунок 3 – Окно программы RTKCONV ver.2.4.2

После открытия в появившемся окне ставим галочки на Time Start (GPST), Time End (GPST), и прописываем время интервала наблюдений с 00:00 10.02.20 до 00:00 11.02.20. В меню «Options» (Рисунок 4) выбираем галочкой спутниковую систему ГЛОНАСС и указываем в поле «Excluded Satellite» следующее: R4, R5, R11, R12, R13, R14, R21, R22, R23, чтобы оставить только спутник №3. В первой строке RTKCONV указываем путь к файлу бинарного потока BINR.bin, выбираем формат NVS BINR, и ставим галочки для конвертации файлов в форматы .obs и .gnav.

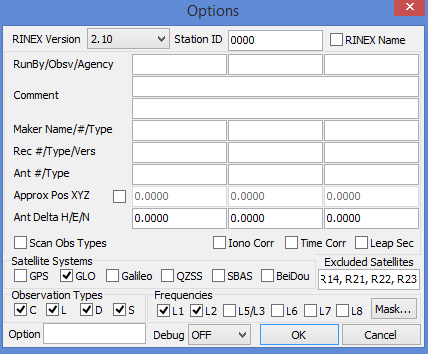


Рисунок 4 – Окно настроек программы RTKCONV ver.2.4.2

После нажатия на кнопку «Convert» получаем необходимые файлы. При нажатии на значок рядом со значком «троеточия» можем посмотреть содержимое файлов.

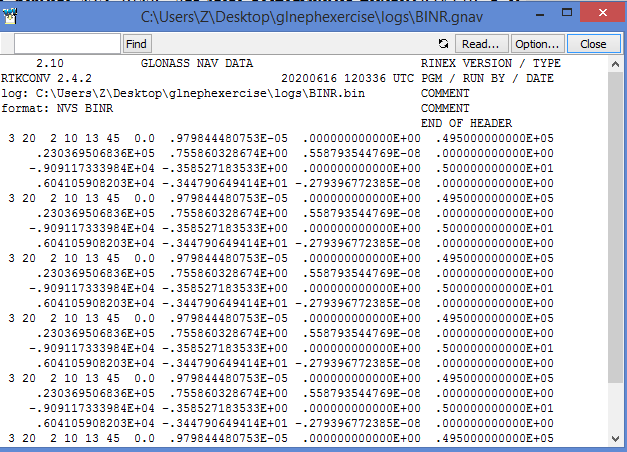


Рисунок 5 – Эфемериды спутника ГЛОНАСС №3 в .gnav-файле

Таким образом, получаем эфемериды собственного спутника в gnav-файле RINEX.

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

Нужно построить график угла места от времени и SkyView собственного спутника на заданный интервал времени. Для этого воспользуемся веб-сайтом Trimble GNSS Planning (<https://www.gnssplanning.com>). Во вкладке настроек (Settings) указываем координаты корпуса «Е» МЭИ и время проведения записи (Рисунок 6). Во вкладке библиотеки спутников (Satellite Library) отключаем отображение всех спутников, кроме заданного (Рисунок 7).

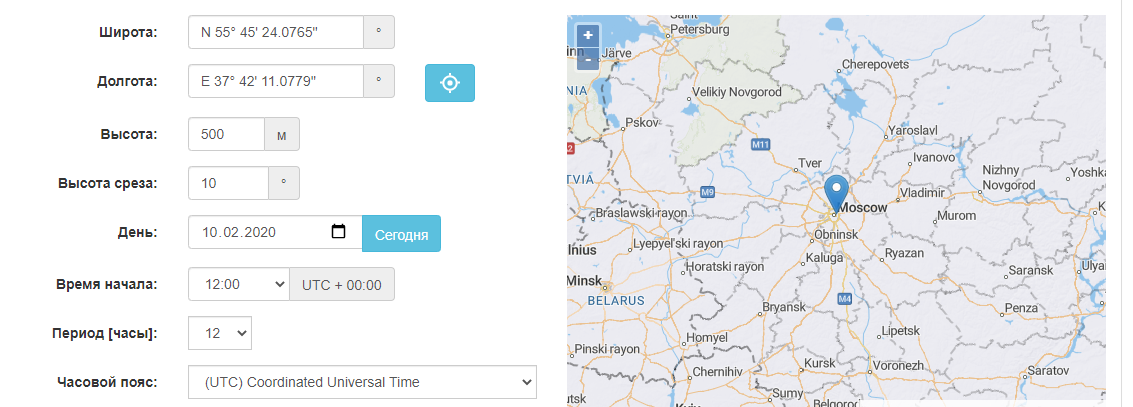


Рисунок 6 – Вкладка настроек (Settings) Trimble GNSS Planning

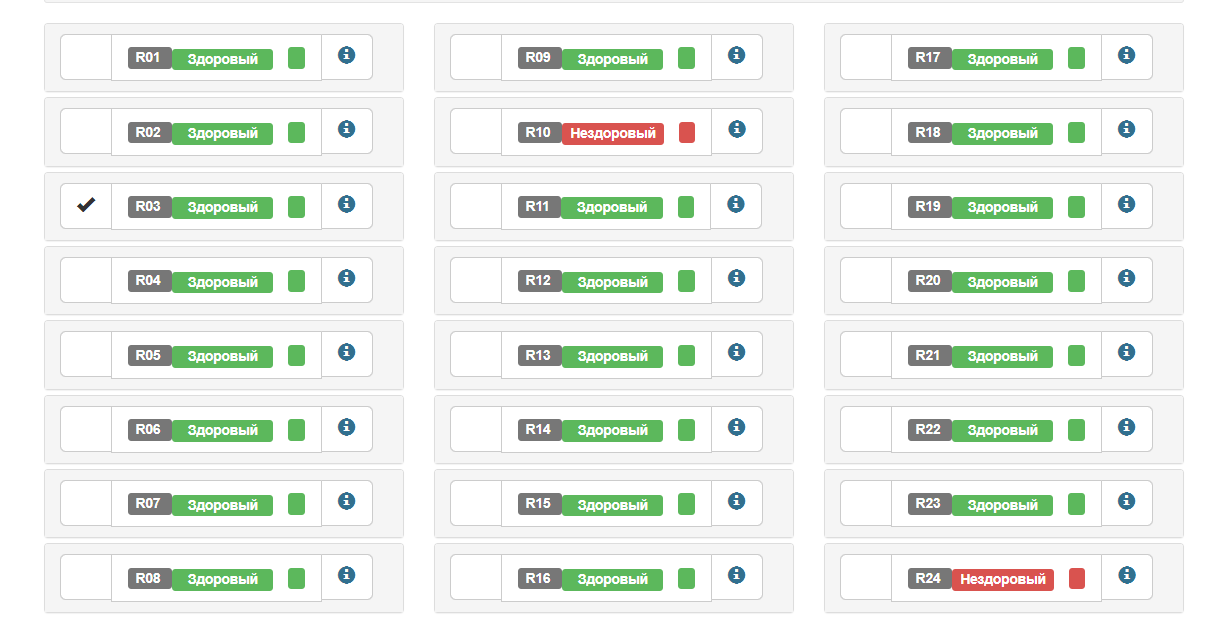


Рисунок 7 – Вкладка библиотека спутников (Satellite Library) Trimble GNSS Planning

Чтобы получить график угла места, нажимаем вкладку графики (Charts). По полученным данным, спутник был виден 2 раза (Рисунок 8). Первое появление с 12:00 до 13:50. Второе появление с 19:50 до 23:40. Время указано по UTC +00:00.

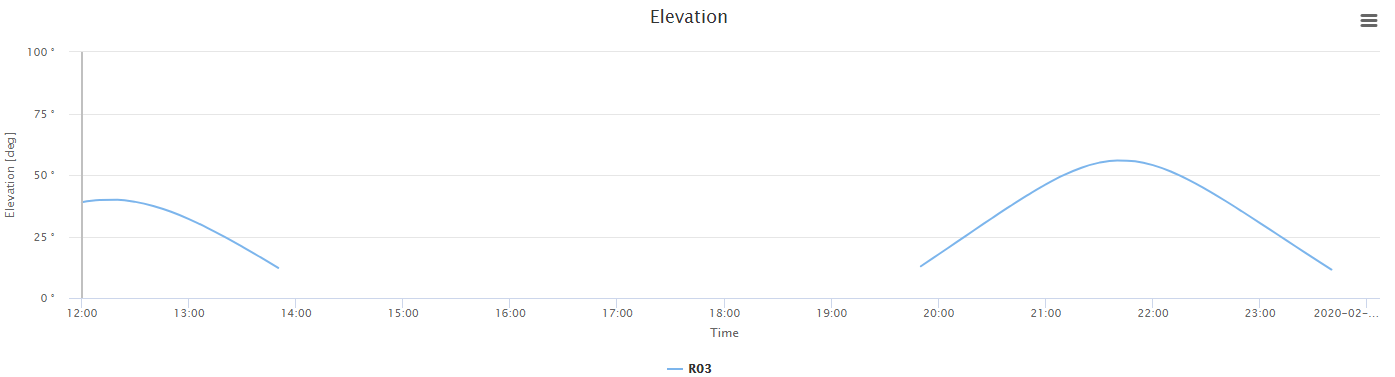


Рисунок 8 – График угла места спутника ГЛОНАСС №3

Во вкладке «Sky Plot» получаем карту небосвода (SkyView) (Рисунки 9-10).

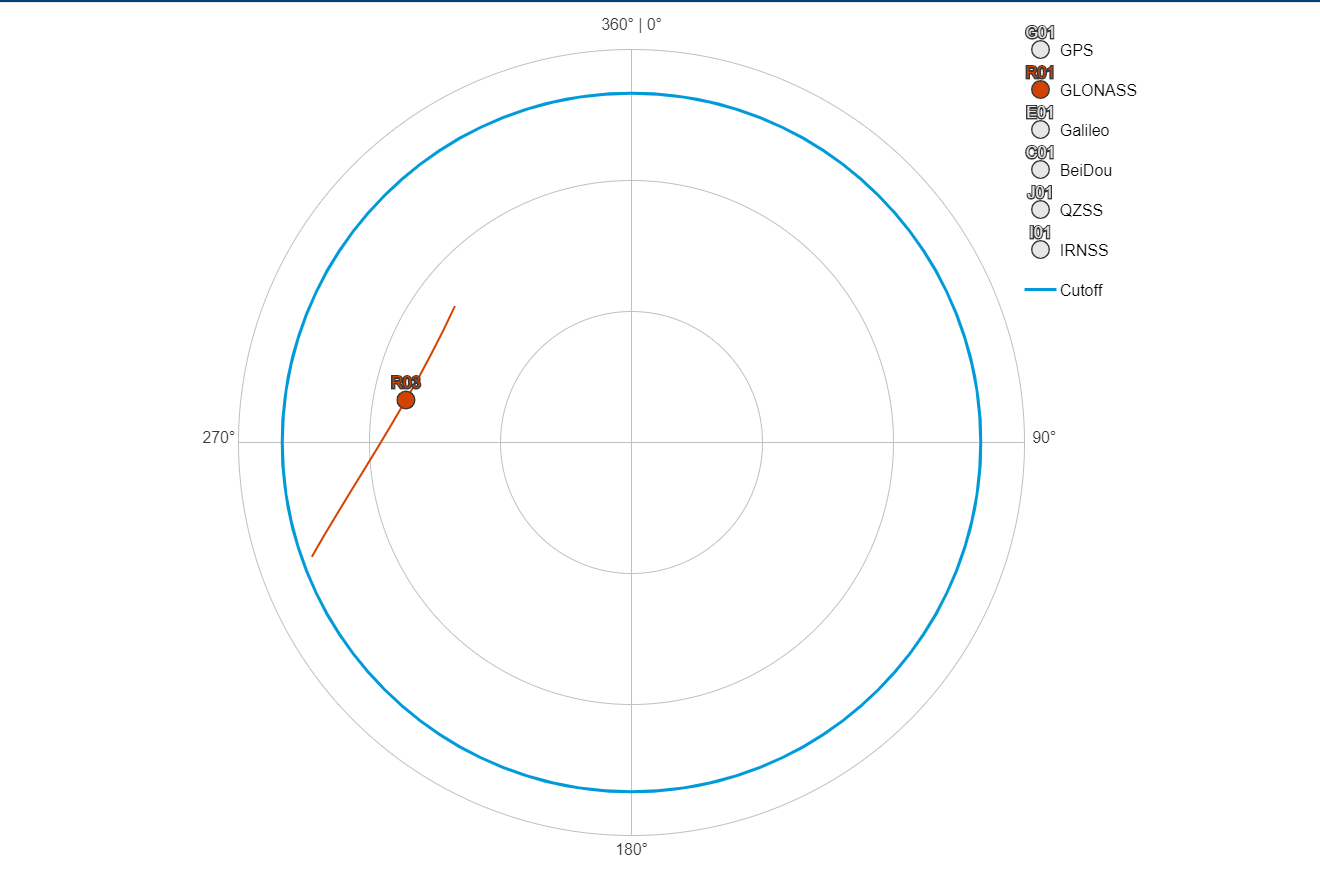


Рисунок 9 – SkyView спутника ГЛОНАСС №3

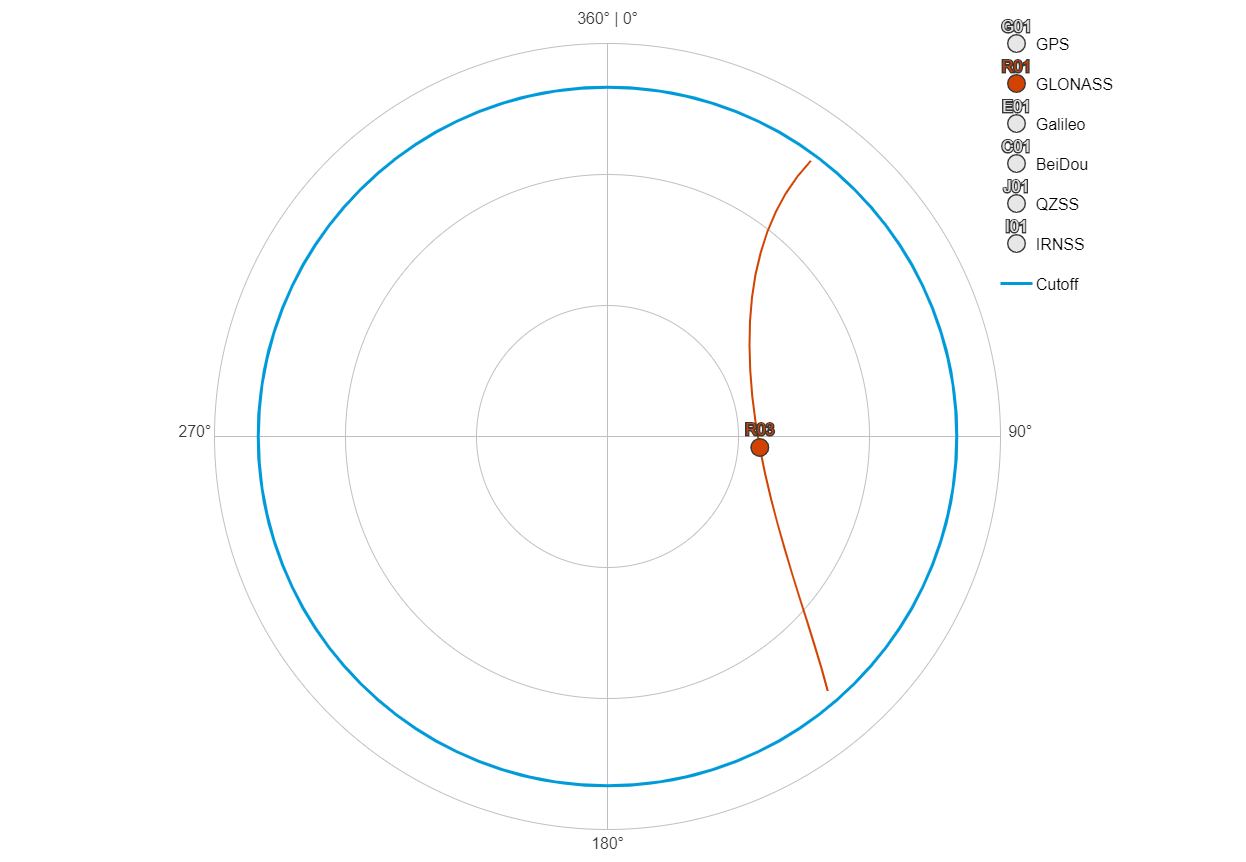


Рисунок 10 – SkyView спутника ГЛОНАСС №3

Траектория спутника на рис. 9 соответствует 1 появлению, а траектория на 10 рис. 2-ому.

# **Этап 2. Моделирование**

# **2.1 Алгоритм расчёта положения спутника ГЛОНАСС**

На предыдущем этапе 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. В разных ГНСС эти модели разные, а значит отличается и формат эфемерид, и алгоритмы расчета положения спутника.

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.

Для расчета положения спутника ГЛОНАСС по эфемеридным данным системы проводят численное интегрирование дифференциального уравнения.

Эфемериды, полученные на предыдущем этапе, будем брать из рисунка 2, так как они записаны в удобном варианте.

Необходимо построить трехмерные графики множества положений спутника №3 ГЛОНАСС. Графики в двух вариантах: в СК ECEF ПЗ-90.11 и соответствующей ей инерциальной СК. Положения должны соответствовать временному интервалу с 12:00 10.02.20 до 00:00 11.02.20. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал.

Для выполнения задания воспользуемся ИКД ГЛОНАСС, в котором описывается три алгоритма расчёта положения спутника на заданный момент времени ti шкалы МДВ по данным эфемерид: точный алгоритм (точный расчёт на 30-минутном интервале), упрощённый алгоритм (тот же расчёт на 30-минутном интервале, но более простой), долговременный алгоритм (точный расчёт на 4-часовом интервале).

Использовать будем точный алгоритм, рассмотрим его подробнее. Поскольку алгоритм использует шкалу МДВ, то и эфемериды нужно перевести в эту шкалу, добавив 3 часа (т. к. они передаются по шкале времени UTC).

Для расчёта используются в алгоритме следующие исходные данные:

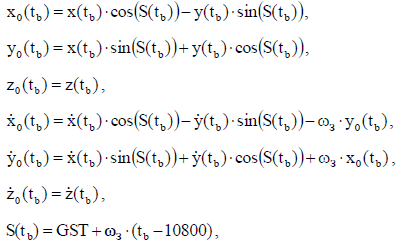
N4 – номер четырёхлетнего периода эфемерид, Nt – номер эфемеридных суток в четырёхлетнем периоде, tb – момент времени из оперативной информации ГЛОНАСС, координаты и составляющие вектора скорости центра масс НКА на момент времени tb.

Пересчет эфемерид потребителем с момента tb шкалы МДВ на заданный момент времени ti 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 J02, характеризующим полярное сжатие Земли, а также ускорениями от лунно-солнечных гравитационных возмущений. Эти уравнения движения определены в виде следующей системы:

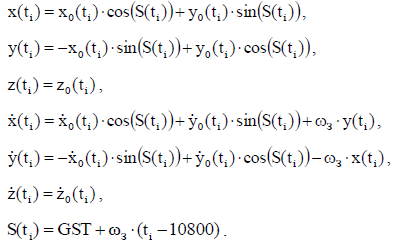


Дифференциальные уравнения движения интегрируются в прямоугольной инерциальной геоцентрической системе координат OX0Y0Z0, начало которой О совпадает с началом координат системы ПЗ-90, ось OX0 направлена в точку весеннего равноденствия, ось OZ0 направлена на Северный полюс мира и совпадает с осью OZ системы ПЗ-90, ось OY0 дополняет систему до правой. Интегрирование осуществляется численным методом, например, методом РунгеКутта 4-го порядка.

Начальными условиями для интегрирования системы являются координаты центра масс НКА , , и составляющие его вектора скорости , , в инерциальной геоцентрической системе координат OX0Y0Z0 на момент tb 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:

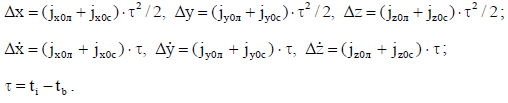


После интегрирования, полученные в инерциальной системе координат OX0Y0Z0 координаты центра масс , , и составляющие его вектора скорости , , могут быть пересчитаны в связанную с Землей систему ПЗ-90 Oxyz по формулам:



В данном алгоритме присутствуют примечания относительно ускорений от солнечно-лунных возмущений, которыми воспользуемся для упрощения расчёта:

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



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

# **2.2 Результаты моделирования в Matlab**

Алгоритм расчёта реализован в программе MATLAB R2017a, код приведён в приложении 1.

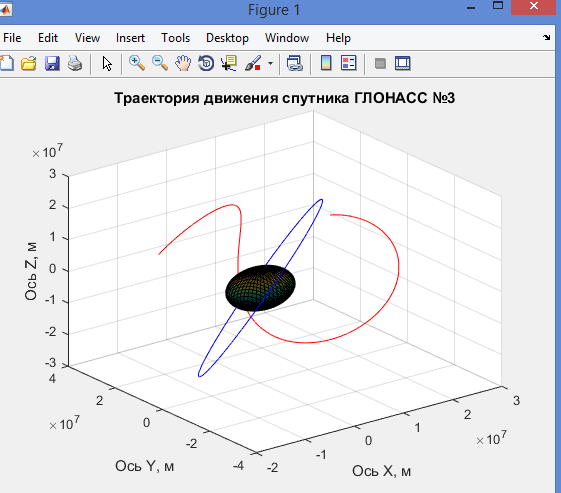


Рисунок 11 – Траектория движения спутника ГЛОНАСС №3 в системе координат ПЗ-90 (красная линия) и инерциальной системе координат (синяя линия)

# **2.3 Построение SkyView в Matlab**

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

Для этого необходимо декартовые координаты пересчитать в сферические, а затем по полученным углам построить графики в полярной системе координат (рисунок 12) и угла места в зависимости от времени (рисунок 13). Полученные результаты соответствуют информации, полученной на Trimble GNSS Planning c некоторыми погрешностями, связанными с рядом факторов.

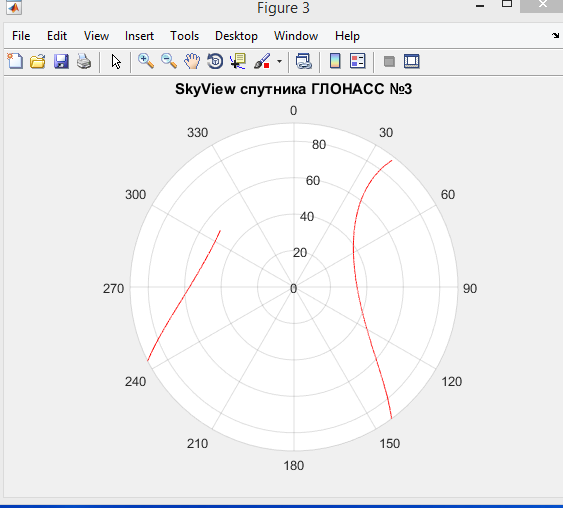


Рисунок 12 – SkyView спутника ГЛОНАСС №3

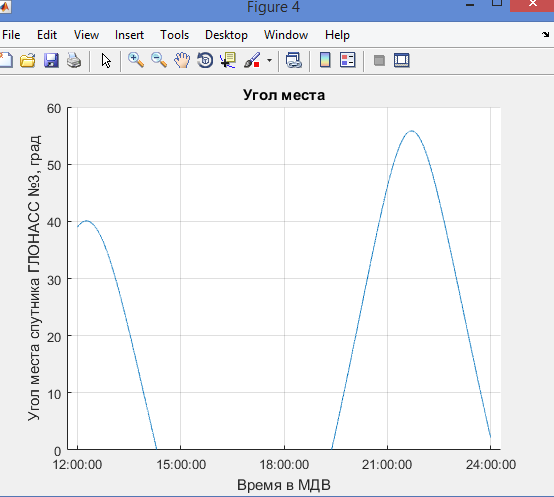


Рисунок 13 – График угла места спутника ГЛОНАСС №3

# **Этап 3. Реализация.**

**3.1. Реализация кода на С++**

Для реализации кода на С++ воспользуемся режимом командной строки, для чего создадим bat файл, текст которого представлен ниже

PATH C:\Qt\Tools\mingw810\_32\bin

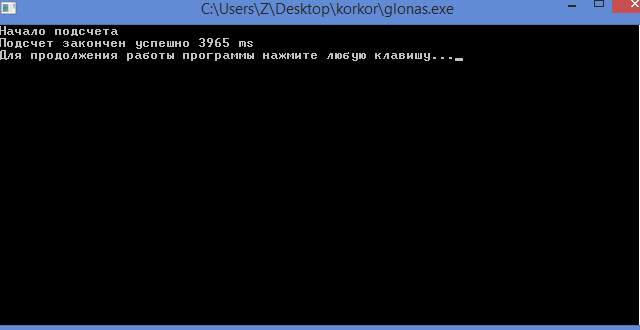
rem PATH D:\QT\Tools\mingw810\_64\bin

gcc main.cpp -std=c++11 -lstdc++ -o3 -o glonas.exe

pause

В качестве компилятора возьмем Mingw последней имеющейся версии, серьезной разницы по времени расчета с другими компилятора обнаружено не было, однако, использование более старых версий этого и других компиляторов серьезно(в 2 и более раза) тормозит расчет.

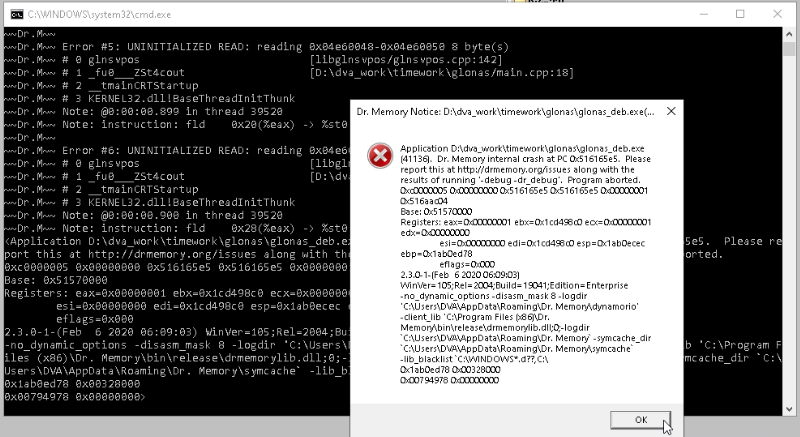
После запуска файла получаем результат, представленный ниже



Итоговое время подсчета программы около 3.9 секунд, в то время как со старыми версиями компиляторов время расчета доходило до 8 секунд.

Также код программы необходимо было проверить на утечку памяти. Так как windows не поддерживает большинство тестов, то выбор был невелик.

Использование скажем Dr Memory приводило к ошибке



Аналогичная ситуация происходила с другими программами и даже теми, которые являются кросплатформенными.

По итогу тесты памяти производились в самой программе

В начало файла, где находится main, необходимо добавить этот кусок кода:

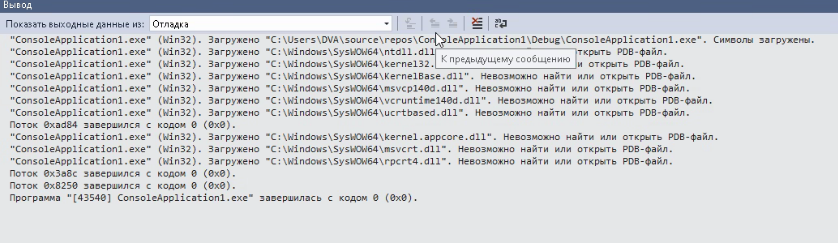
#define \_\_CRTDBG\_MAP\_ALLOC

#include <crtdbg.h>

#define DEBUG\_NEW new(\_NORMAL\_BLOCK, \_\_FILE\_\_, \_\_LINE\_\_)

#define new DEBUG\_NEW

Результаты представлены ниже, утечек памяти в коде не обнаружено



При внесении утечки памяти искусственно различными путями, такими как выделение лишних битов и удаление команд очищения памяти, программа сразу же указывала на утечку памяти.

# **4. Заключение.**

В ходе первого этапа 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.

В ходе второго этапа были получены следующие результаты:

* На языке Matlab была написана программа для расчета положения спутника ГЛОНАСС №3 ;
* Был построен SkyView спутника ГЛОНАСС №3;
* Был построен График угла места спутника ГЛОНАСС №3;
* Был построен график Траектория движения спутника ГЛОНАСС №3 в системе координат ПЗ-90.

В ходе третьего этапа были получены следующие результаты:

* На языке C++ была написана программа для расчета положения спутника ГЛОНАСС №3 ;
* Были проведены тесты получившейся программы на утечку памяти;
* Была искусственно внесена, а затем устранена утечка памяти в код;
* Был получен текстовый файл содержащий координаты спутника ГЛОНАСС №3.
* В код была внесена программа для вывода времени расчета, а также скорректировано это время путем использования компилятора определенной версии.

## **5 Литература**

1. ИКД ГЛОНАСС. Общее описание системы с кодовым разделением.
2. GLONASS Satellite Coordinates Computation – Navipedia.
3. Материалы лекций по курсам: «Математическое моделирование РТУ и С», и «Аппаратура потребителей спутниковых радионавигационных систем».
4. Интернет ресурс habr.com
5. Интернет ресурс github.com

**ПРИЛОЖЕНИЕ 1.**

Программа nom2.m

clear all, close all

tic

%внесем все константы

o\_z = 7.2921151467e-5;

Rz=6371000;

J02 = -1082.63\*10^-6;

ae = 6378136;

GM = 398600441.8e6;

N = 55.756687916667\*pi/180;

E=37.703077194444\*pi/180;

High = 500;

%известные данные спутника номер 3

%координаты

Xkoor = 23036950.68; % м

Ykoor = -9091173.34; % м

Zkoor = 6041059.08; % м

%скорости

VXkoor= 755.86033; % м/с

VYkoor = -358.52718; % м/с

VZkoor = -3447.90649; % м/с

%ускорения

AXkoor = 0.0000056; % м/с^2

AYkoor = 0.0000000; % м/с^2

AZkoor = -0.0000028; % м/с^2

%время

%номер текущего четырёхлетия

N4=(2020-1996)/4+1;

%номер текущих суток

Nt=365\*(2020-1996-4\*(N4-1))+31+10+1;

%время МДВ

time=13\*60\*60 + 45\*60 + 18 + 10800;

%расчёт текущей юлианской даты

JD0 = 1461\*(((2020-1996)/4)+1 - 1) + Nt + 2450008.5 - (Nt -3)/25;

%среднее звёздное время по Гринвичу

T\_d=(JD0-2451545)/36525;

ERA=2\*pi\*(0.7790572732640 + 1.00273781191135448\*(JD0-2451545));

GMST=ERA+0.0000000703270726+0.0223603658710194\*T\_d+...

+0.0000067465784654\*T\_d^2-0.0000000000021332\*T\_d^3+...

- 0.0000000001452308\*T\_d^4-0.0000000000001784\*T\_d^5;

%пересчет в инерциальную систему координат

S = GMST + o\_z\*(time - 3\*60\*60);

Xa=Xkoor\*cos(S)-Ykoor\*sin(S);

%координаты

Ya=Xkoor\*sin(S)+Ykoor\*cos(S);

Za=Zkoor;

%скорости

Vxa=VXkoor\*cos(S)-VYkoor\*sin(S)-o\_z\*Ya;

Vya=VXkoor\*sin(S)+VYkoor\*cos(S)+o\_z\*Xa;

Vza=VZkoor;

%ускорения

Ax=AXkoor\*cos(S)-AYkoor\*sin(S);

Ay=AXkoor\*sin(S)+AYkoor\*cos(S);

Az=AZkoor;

%расчет

r=sqrt(Xa^2 + Ya^2 + Za^2);

GMn = GM/r^2;

x1 = Xa/r;

y1 = Ya/r;

z1 = Za/r;

RO = ae/r;

%метод Рунге-Кутты

%задание времени и шага

TimeS = (12+3)\*60\*60;

TimeE = (24+3)\*60\*60;

del = 1;

pros = TimeS:del:TimeE;

F0 = [Xa Ya Za Vxa Vya Vza];

[t, F] = ode45('diffs', time:-del:pros(1), F0);

F1 = F(end:-1:2,:);

t1 = t(end:-1:2,:);

[t, F] = ode45('diffs', time:del:pros(end), F0);

F1 = [F1; F];

t1 = [t1;t];

% Учёт ускорений

tR1 = t1 - time;

AXR = AXkoor\*(tR1.^2)/2;

AYR = AYkoor\*(tR1.^2)/2;

AZR = AZkoor\*(tR1.^2)/2;

dVX = AXkoor\*tR1;

dVY = AYkoor\*tR1;

dVZ = AZkoor\*tR1;

d\_A = [AXR AYR AZR dVX dVY dVZ];

F1 = F1 + d\_A;

% Пересчёт координат в ПЗ-90

S = GMST + o\_z\*(t1 - 3\*60\*60);

PZ90(:,1) = F1(:,1).\*cos(S) + F1(:,2).\*sin(S);

PZ90(:,2) = -F1(:,1).\*sin(S) + F1(:,2).\*cos(S);

PZ90(:,3) = F1(:,3);

cord\_E = [N E High];

for i = 1:length(PZ90(:,1))

[Xkoor(i) Ykoor(i) Zkoor(i)] = ecef2enu(PZ90(i,1),PZ90(i,2),PZ90(i,3),N,E,High,wgs84Ellipsoid,'radians');

if Zkoor(i) > 0

r(i) = sqrt(Xkoor(i)^2 + Ykoor(i)^2 + Zkoor(i)^2);

teta(i) = acos(Zkoor(i)/r(i));

if Xkoor(i) > 0

phi(i) = -atan(Ykoor(i)/Xkoor(i))+pi/2;

elseif (Xkoor(i)<0)&&(Ykoor(i)>0)

phi(i) = -atan(Ykoor(i)/Xkoor(i))+3\*pi/2;

elseif (Xkoor(i)<0)&&(Ykoor(i)<0)

phi(i) = -atan(Ykoor(i)/Xkoor(i))-pi/2;

end

else teta(i) = NaN;

r(i) = NaN;

phi(i) = NaN;

end

end

%Построение графиков

figure(1)

[Xx,Yy,Zz]=sphere(30);

surf(Rz\*Xx,Rz\*Yy,Rz\*Zz)

hold on

grid on

plot3(F1(:,1), F1(:,2), F1(:,3), 'b')

xlabel('Ось Х, м')

ylabel('Ось Y, м')

zlabel('Ось Z, м')

surf(Rz\*Xx,Rz\*Yy,Rz\*Zz)

grid on

title({'Траектория движения спутника ГЛОНАСС №3'})

plot3(PZ90(:,1),PZ90(:,2),PZ90(:,3),'r')

xlabel('Ось Х, м')

ylabel('Ось Y, м')

zlabel('Ось Z, м')

hold off

figure (2)

pax = polaraxes;

polarplot(pax,phi,teta\*180/pi,'r')

pax.ThetaDir = 'clockwise';

pax.ThetaZeroLocation = 'top';

title('SkyView спутника ГЛОНАСС №3')

th = hours(t1./3600-3);

figure(3);

grid on

hold on

plot(th,(-teta\*180/pi+90),'DurationTickFormat','hh:mm:ss')

title('Угол места')

xlabel('Время в МДВ')

ylabel('Угол места спутника ГЛОНАСС №3, град')

Подпрограмма расчета дифура

function dF = diffs(t, F)

%константы

GM = 398600441.8e6;

J02 = -1082.63\*10^-6;

ae = 6378136;

%расчет

Xa=F(1);

Ya=F(2);

Za=F(3);

r=sqrt(Xa^2 + Ya^2 + Za^2);

GMn = GM/r^2;

x1 = Xa/r;

y1 = Ya/r;

z1 = Za/r;

RO = ae/r;

dF = F(:);

dF(1) = F(4);

dF(2) = F(5);

dF(3) = F(6);

dF(4) = -GMn\*x1 + 1.5\*J02\*GMn\*x1\*(RO^2)\*(1 - 5\*z1^2);

dF(5) = -GMn\*y1 + 1.5\*J02\*GMn\*y1\*(RO^2)\*(1 - 5\*z1^2);

dF(6) = -GMn\*z1 + 1.5\*J02\*GMn\*z1\*(RO^2)\*(3 - 5\*z1^2);

end

**ПРИЛОЖЕНИЕ 2.**

Программа main.cpp

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

#include <sys/timeb.h>

#include <time.h>

using namespace std;

// Исходные данные

#define CoordinateX 23036950.68

#define CoordinateY -9091173.34

#define CoordinateZ 6041059.08

#define VelocityX 755.86033

#define VelocityY -358.52718

#define VelocityZ -3447.90649

#define AccelerationX 0.0000056

#define AccelerationY 0.0000000

#define AccelerationZ -0.0000028

#define Time\_Year 2020

#define Time\_Month 2

#define Time\_Day 10

#define Time\_Hour 13

#define Time\_Minutes 45

#define Time\_Seconds 18

struct Ephemeris\_type {

uint16\_t N4; // Time in Gln

uint16\_t NT;

uint32\_t tb;

double X, Y, Z, // Coordinates

VX, VY,VZ, // Velocity

AX, AY, AZ; // Acceleration

};

struct **Y\_type** {

double X, Y, Z, VX, VY, VZ;

};

//#define \_USE\_MATH\_DEFINES

#include <math.h> // число pi

#ifndef M\_PI

#define M\_PI 3.14159265358979323846

#endif

using namespace std;

uint16\_t NT\_Calculate(uint8\_t N4, uint16\_t Tyear, uint16\_t year\_idx, uint16\_t Tmonth, uint16\_t Tday) {

uint16\_t NT, N42;

uint8\_t Mon[12] = {31, 30, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

if (Tmonth < 1 || Tmonth > 12) {

return 0;

}

N42 = (year\_idx\*31) + N4;

NT = Tyear - 1996 - 4 \* (N42 - 1);

NT \*= 365;

for (uint16\_t i = 1; i < Tmonth; i++) {

NT += Mon[i-1];

}

return (NT + Tday + 1);

}

double GMST\_Calculate(uint8\_t N4, uint16\_t NT) {

// Текущая Юлианская дата на 0 часов шкалы МДВ

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

// Время от эпохи 2000 г 1 января 12 ч (UTC(SU))

double Td = (JD0 - 2451545) / 36525.0;

// Угол поворота Земли [рад]

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

// Среднее звездное время по Гринвичу [рад]

double GMST = ERA;

GMST += 0.0000000703270726;

GMST += 0.0223603658710194 \* Td;

GMST += 0.0000067465784654 \* Td \* Td;

GMST -= 0.0000000000021332 \* Td \* Td \* Td;

GMST -= 0.0000000001452308 \* Td \* Td \* Td \* Td;

GMST -= 0.0000000000001784 \* Td \* Td \* Td \* Td \* Td;

return GMST;

}

void **CrdTrnsfToInertial**(struct Ephemeris\_type Ephemeris, double GMST, struct Ephemeris\_type &Ephemeris0) {

double OmegaE = 7.2921151467e-5,

ThetaGe = GMST + OmegaE \* (Ephemeris.tb - 3\*60\*60);

// Координаты:

Ephemeris0.X = Ephemeris.X \* cos(ThetaGe) - Ephemeris.Y \* sin(ThetaGe);

Ephemeris0.Y = Ephemeris.X \* sin(ThetaGe) + Ephemeris.Y \* cos(ThetaGe);

Ephemeris0.Z = Ephemeris.Z;

// Скорости:

Ephemeris0.VX = Ephemeris.VX \* cos(ThetaGe) - Ephemeris.VY \* sin(ThetaGe) - OmegaE \* Ephemeris0.Y;

Ephemeris0.VY = Ephemeris.VX \* sin(ThetaGe) + Ephemeris.VY \* cos(ThetaGe) + OmegaE \* Ephemeris0.X;

Ephemeris0.VZ = Ephemeris.VZ;

// Ускорения:

Ephemeris0.AX = Ephemeris.AX \* cos(ThetaGe) - Ephemeris.AY \* sin(ThetaGe);

Ephemeris0.AY = Ephemeris.AX \* sin(ThetaGe) + Ephemeris.AY \* cos(ThetaGe);

Ephemeris0.AZ = Ephemeris.AZ;

// Время

Ephemeris0.N4 = Ephemeris.N4;

Ephemeris0.NT = Ephemeris.NT;

Ephemeris0.tb = Ephemeris.tb;

}

void SaveData(struct Y\_type \*Y\_data, uint64\_t Size, char \*fname) {

// Сохранение данных (для матлаба)

FILE \*fs;

if ((fs = fopen(fname,"wb")) == NULL) {

printf("Error. File: %s, Line: %d\n", \_\_FILE\_\_, \_\_LINE\_\_);

} else {

for(uint32\_t i = 0; i <= Size; i++) {

fprintf(fs, "%.15e %.15e %.15e %.15e %.15e %.15e\n", Y\_data[i].X, Y\_data[i].Y, Y\_data[i].Z, Y\_data[i].VX, Y\_data[i].VY, Y\_data[i].VZ);

}

fclose(fs);

}

}

Y\_type\* **DiffsY**(struct Y\_type Y)

{

double J02 = 1082625.75e-9, // зональный гармонический коэффициент второй степени, характеризующий полярное сжатие Земли

GM = 398600441.8e6, // геоцентрическая константа гравитационного поля Земли с учетом атмосферы, [м3/c2]

a\_e = 6378136; // большая полуось общеземного эллипсоида, [м]

double crdX = Y.X, crdY = Y.Y, crdZ = Y.Z;

double r = sqrt(crdX \* crdX + crdY \* crdY + crdZ \* crdZ);

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

double Rho = a\_e / r;

double crdX0 = crdX / r, crdY0 = crdY / r, crdZ0 = crdZ / r;

struct Y\_type \*dY = new struct Y\_type;

// Дифуры

dY->X = Y.VX;

dY->Y = Y.VY;

dY->Z = Y.VZ;

dY->VX = - GM0 \* crdX0 - 1.5 \* J02 \* GM0 \* crdX0 \* Rho \* Rho \* (1.0 - 5.0 \* crdZ0 \* crdZ0);

dY->VY = - GM0 \* crdY0 - 1.5 \* J02 \* GM0 \* crdY0 \* Rho \* Rho \* (1.0 - 5.0 \* crdZ0 \* crdZ0);

dY->VZ = - GM0 \* crdZ0 - 1.5 \* J02 \* GM0 \* crdZ0 \* Rho \* Rho \* (3.0 - 5.0 \* crdZ0 \* crdZ0);

return dY;

}

int RK(uint32\_t N, double h, struct Y\_type\* Y) {

if (N == 0 || fabs(h)< 0.0000001 ) return 0;

struct Y\_type \*k1, \*k2, \*k3, \*k4, \*knextstep;

struct Y\_type Y2, Y3, Y4;

for (uint32\_t k = 1; k < N; k++) {

k1 = new struct Y\_type;

k2 = new struct Y\_type;

k3 = new struct Y\_type;

k4 = new struct Y\_type;

knextstep = new struct Y\_type;

k1 = DiffsY(Y[k-1]);

Y2.X = Y[k-1].X + h \* k1->X / 2.0;

Y2.Y = Y[k-1].Y + h \* k1->Y / 2.0;

Y2.Z = Y[k-1].Z + h \* k1->Z / 2.0;

Y2.VX = Y[k-1].VX + h \* k1->VX / 2.0;

Y2.VY = Y[k-1].VY + h \* k1->VY / 2.0;

Y2.VZ = Y[k-1].VZ + h \* k1->VZ / 2.0;

k2 = DiffsY(Y2);

Y3.X = Y[k-1].X + h \* k2->X / 2.0;

Y3.Y = Y[k-1].Y + h \* k2->Y / 2.0;

Y3.Z = Y[k-1].Z + h \* k2->Z / 2.0;

Y3.VX = Y[k-1].VX + h \* k2->VX / 2.0;

Y3.VY = Y[k-1].VY + h \* k2->VY / 2.0;

Y3.VZ = Y[k-1].VZ + h \* k2->VZ / 2.0;

k3 = DiffsY (Y3);

Y4.X = Y[k-1].X + h \* k3->X;

Y4.Y = Y[k-1].Y + h \* k3->Y;

Y4.Z = Y[k-1].Z + h \* k3->Z;

Y4.VX = Y[k-1].VX + h \* k3->VX;

Y4.VY = Y[k-1].VY + h \* k3->VY;

Y4.VZ = Y[k-1].VZ + h \* k3->VZ;

k4 = DiffsY (Y4);

knextstep->X = h / 6.0 \* ( k1->X + 2.0 \* k2->X + 2.0 \* k3->X + k4->X );

knextstep->Y = h / 6.0 \* ( k1->Y + 2.0 \* k2->Y + 2.0 \* k3->Y + k4->Y );

knextstep->Z = h / 6.0 \* ( k1->Z + 2.0 \* k2->Z + 2.0 \* k3->Z + k4->Z );

knextstep->VX = h / 6.0 \* ( k1->VX + 2.0 \* k2->VX + 2.0 \* k3->VX + k4->VX );

knextstep->VY = h / 6.0 \* ( k1->VY + 2.0 \* k2->VY + 2.0 \* k3->VY + k4->VY );

knextstep->VZ = h / 6.0 \* ( k1->VZ + 2.0 \* k2->VZ + 2.0 \* k3->VZ + k4->VZ );

Y[k].X = Y[k-1].X + knextstep->X;

Y[k].Y = Y[k-1].Y + knextstep->Y;

Y[k].Z = Y[k-1].Z + knextstep->Z;

Y[k].VX = Y[k-1].VX + knextstep->VX;

Y[k].VY = Y[k-1].VY + knextstep->VY;

Y[k].VZ = Y[k-1].VZ + knextstep->VZ;

delete k1;

delete k2;

delete k3;

delete k4;

delete knextstep;

}

return 0;

}

uint64\_t GlonassVPos(bool RK\_valid, double h) {

// Class Ephemeris

struct Ephemeris\_type Ephemeris;

// Coordinates

Ephemeris.X = CoordinateX;

Ephemeris.Y = CoordinateY;

Ephemeris.Z = CoordinateZ;

// Velocity

Ephemeris.VX = VelocityX;

Ephemeris.VY = VelocityY;

Ephemeris.VZ = VelocityZ;

// Acceleration

Ephemeris.AX = AccelerationX;

Ephemeris.AY = AccelerationY;

Ephemeris.AZ = AccelerationZ;

uint16\_t Time\_year = Time\_Year;

uint8\_t Time\_month = Time\_Month,

Time\_day = Time\_Day,

Time\_hour = Time\_Hour,

Time\_minutes = Time\_Minutes,

Time\_seconds = Time\_Seconds;

uint16\_t year\_idx = 0;

// Time in Gln

Ephemeris.N4 = ((Time\_year - 1996) / 4) + 1;

while (Ephemeris.N4 > 31) { // Учет 5-битности N4

Ephemeris.N4 -= 31;

year\_idx++;

}

Ephemeris.NT = NT\_Calculate( Ephemeris.N4, Time\_year, year\_idx, Time\_month, Time\_day );

Ephemeris.tb = Time\_seconds + Time\_minutes\*60 + Time\_hour\*60\*60 + 10800;

if (Ephemeris.tb >= 24\*60\*60) {

Ephemeris.tb -= 24\*60\*60;

Ephemeris.NT++;

if (Ephemeris.NT >= 1462) Ephemeris.N4++;

}

double GMST = GMST\_Calculate( Ephemeris.N4, Ephemeris.NT );

struct Ephemeris\_type Ephemeris0;

CrdTrnsfToInertial(Ephemeris, GMST, *Ephemeris0*);

uint32\_t tn = Ephemeris.tb; // Текущее время

uint32\_t Toe = (12+3)\*60\*60; // Начальное время

uint32\_t Tof = (24+3)\*60\*60; // Конечное время

uint64\_t N2inc = (Tof - tn) / h; // Количество отcчетов для времени большего текущего Ephemeris.tb

uint64\_t N2dec = (tn - Toe) / h; // Количество отcчетов для времени меньшего текущего Ephemeris.tb

uint64\_t N = N2inc + N2dec; // Общее число отсчетов

// Численное интегрирования для времени меньшего текущего Ephemeris.tb

struct Y\_type \*Ydec;

Ydec = new struct Y\_type[N2dec];

Ydec[0].X = Ephemeris0.X;

Ydec[0].Y = Ephemeris0.Y;

Ydec[0].Z = Ephemeris0.Z;

Ydec[0].VX = Ephemeris0.VX;

Ydec[0].VY = Ephemeris0.VY;

Ydec[0].VZ = Ephemeris0.VZ;

if (RK\_valid) RK( N2dec, -h, Ydec);

// Численное интегрирования для времени большего текущего Ephemeris.tb

struct Y\_type \*Yinc;

Yinc = new struct Y\_type[N2inc];

Yinc[0].X = Ephemeris0.X;

Yinc[0].Y = Ephemeris0.Y;

Yinc[0].Z = Ephemeris0.Z;

Yinc[0].VX = Ephemeris0.VX;

Yinc[0].VY = Ephemeris0.VY;

Yinc[0].VZ = Ephemeris0.VZ;

if (RK\_valid) RK( N2inc, h, Yinc);

// Формирование выходных данных

struct Y\_type \*Yout;

Yout = new struct Y\_type[N];

uint32\_t i;

for (i = 0; i < N2dec; i++) {

Yout[i] = Ydec[N2dec-i-1];

}

for (i = 0; i < N2inc; i++) {

Yout[i+N2dec] = Yinc[i];

}

// Учет ускорений

double tau = (double)Toe - (double)tn;

for (i = 0; i < N; i++) {

double tau2 =tau \* tau;

Yout[i].X += Ephemeris0.AX \* tau2 / 2.0;

Yout[i].Y += Ephemeris0.AY \* tau2 / 2.0;

Yout[i].Z += Ephemeris0.AZ \* tau2 / 2.0;

Yout[i].VX += Ephemeris0.AX \* tau;

Yout[i].VY += Ephemeris0.AY \* tau;

Yout[i].VZ += Ephemeris0.AZ \* tau;

tau += h;

}

SaveData(Yout, N, (char\*)"glonass\_data.txt");

// Очищение памяти

delete []Yout;

delete []Ydec;

delete []Yinc;

return N;

}

int **main**() {

/\* бла бла \*/

setlocale(LC\_ALL, "Russian");

cout << "Начало подсчета" << endl;

struct **timeb** tb;

::ftime(&tb);

uint64\_t st=1000ull\*(uint64\_t)tb.time+(uint64\_t)tb.millitm;

string msg;

uint64\_t rc=GlonassVPos(0, 0.1);

::ftime(&tb);

uint64\_t et=1000ull\*(uint64\_t)tb.time+(uint64\_t)tb.millitm;

cout << (rc>0?"Подсчет закончен успешно":"Подсчет закончен с ошибкой ") << " " << (et-st) << " ms" << endl;

cout<<"Для продолжения работы программы нажмите любую клавишу...";

cin.get();

// return rc;

}