**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. Перенос из Matlab в 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).

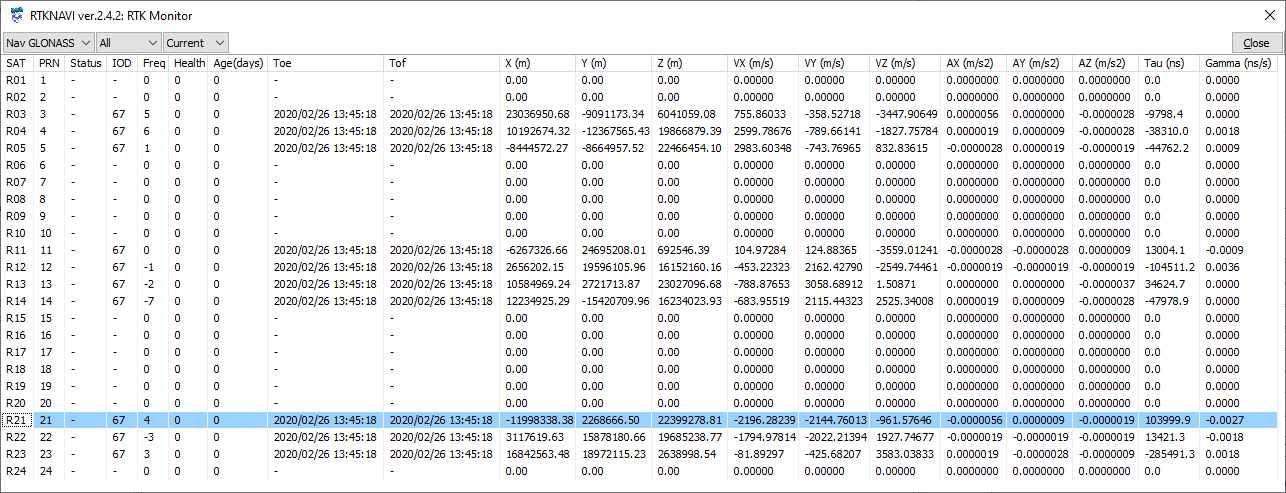


Рисунок 2 – Окно программы RTKNAVI ver.2.4.2

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).

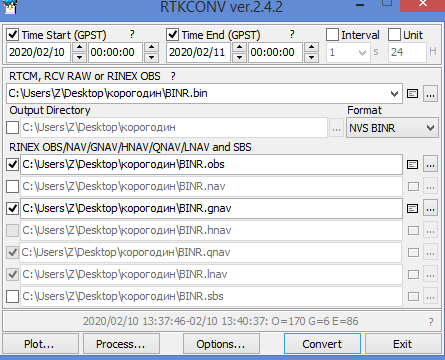


Рисунок 3 – Окно программы RTKCONV ver.2.4.2

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.

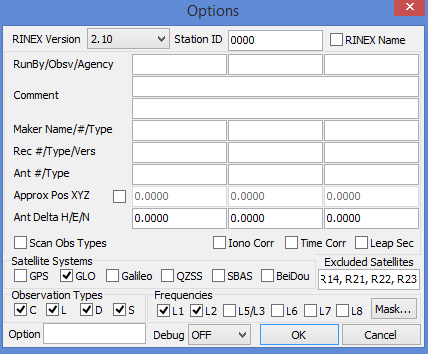


Рисунок 4 – Окно настроек программы RTKCONV ver.2.4.2

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

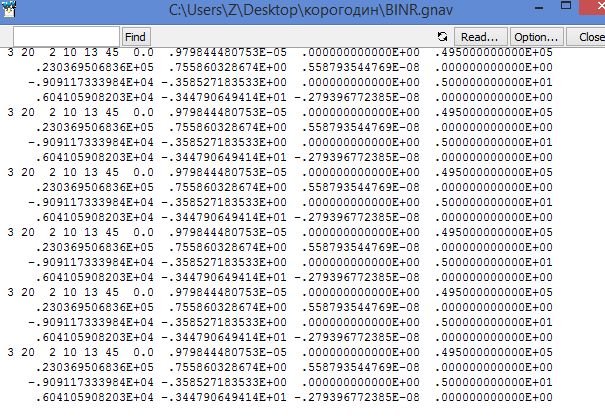


Рисунок 5 – Эфемериды спутника ГЛОНАСС №3 в .gnav-файле

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).

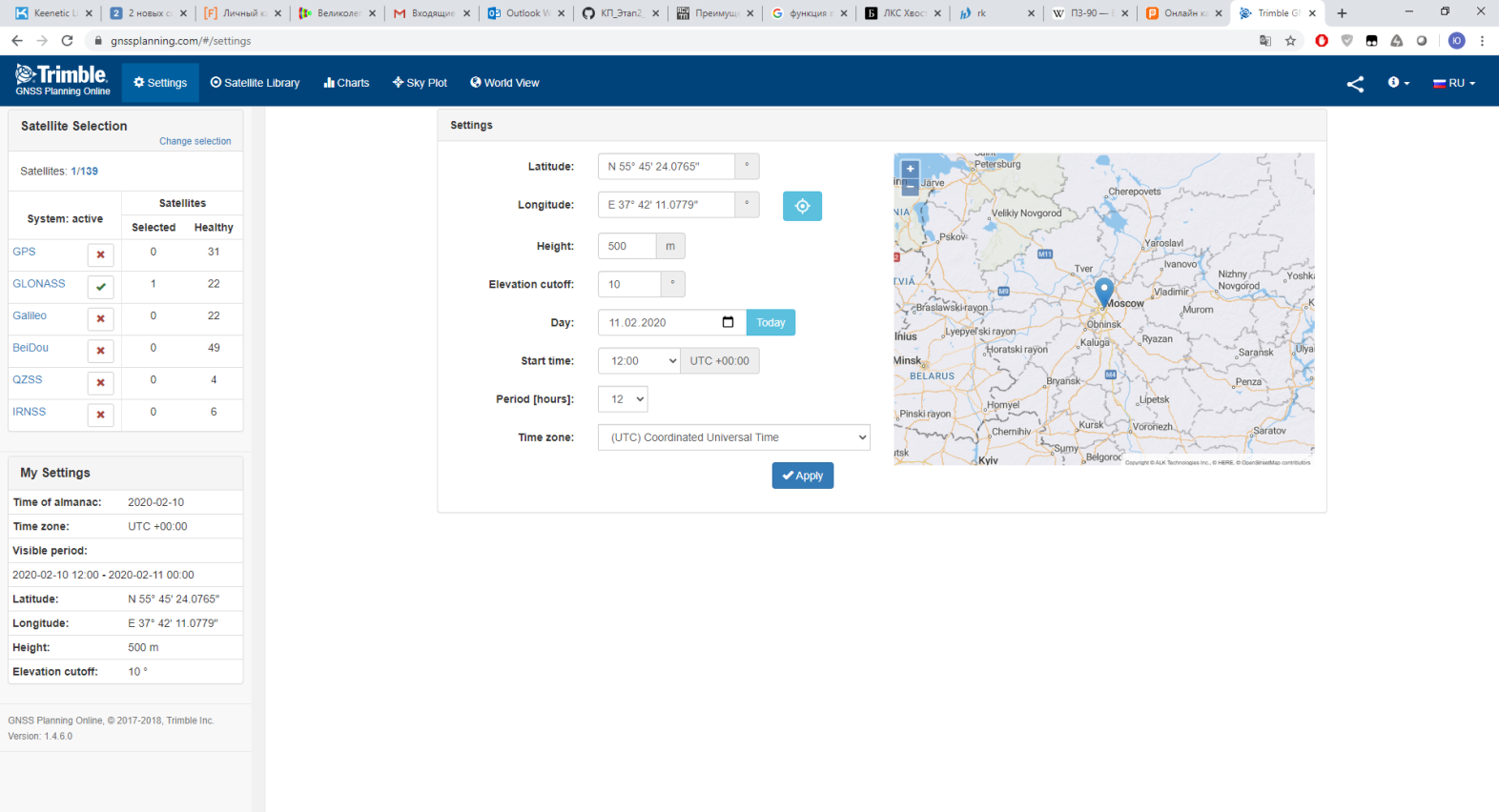


Рисунок 6 – Вкладка настроек (Settings) Trimble GNSS Planning

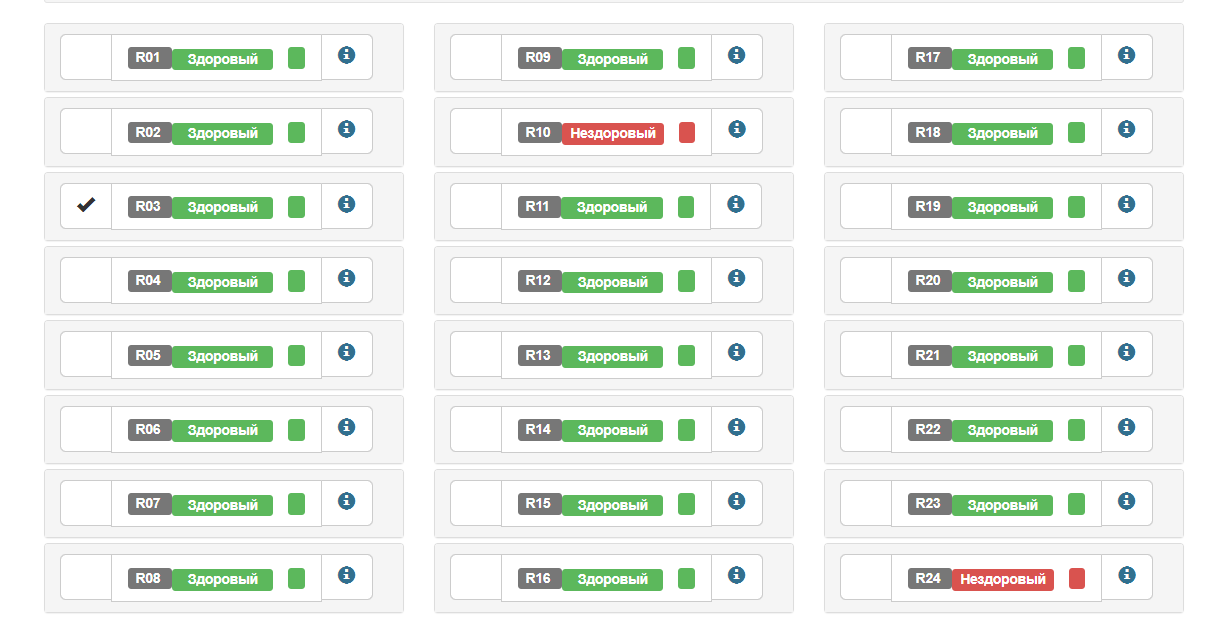


Рисунок 7 – Вкладка библиотека спутников (Satellite Library) Trimble GNSS Planning

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 12:40. Второе появление с 18:40 до 22:00. Время указано по UTC +00:00.

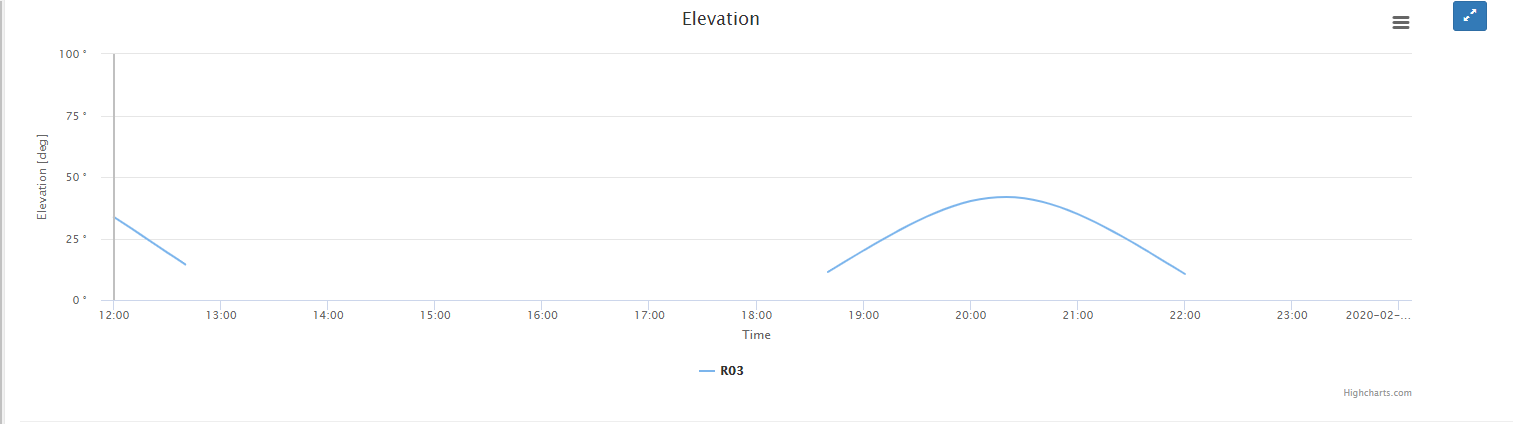


Рисунок 8 – График угла места спутника ГЛОНАСС №3

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

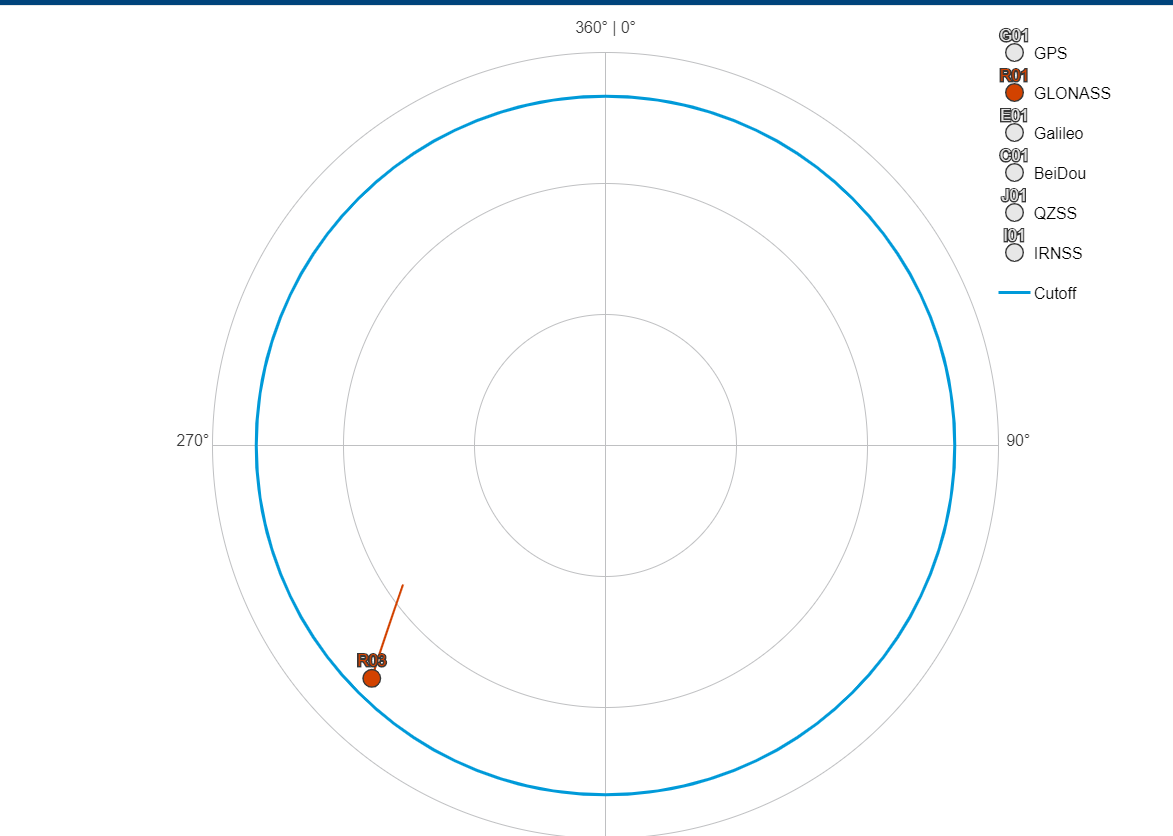


Рисунок 9 – SkyView спутника ГЛОНАСС №3

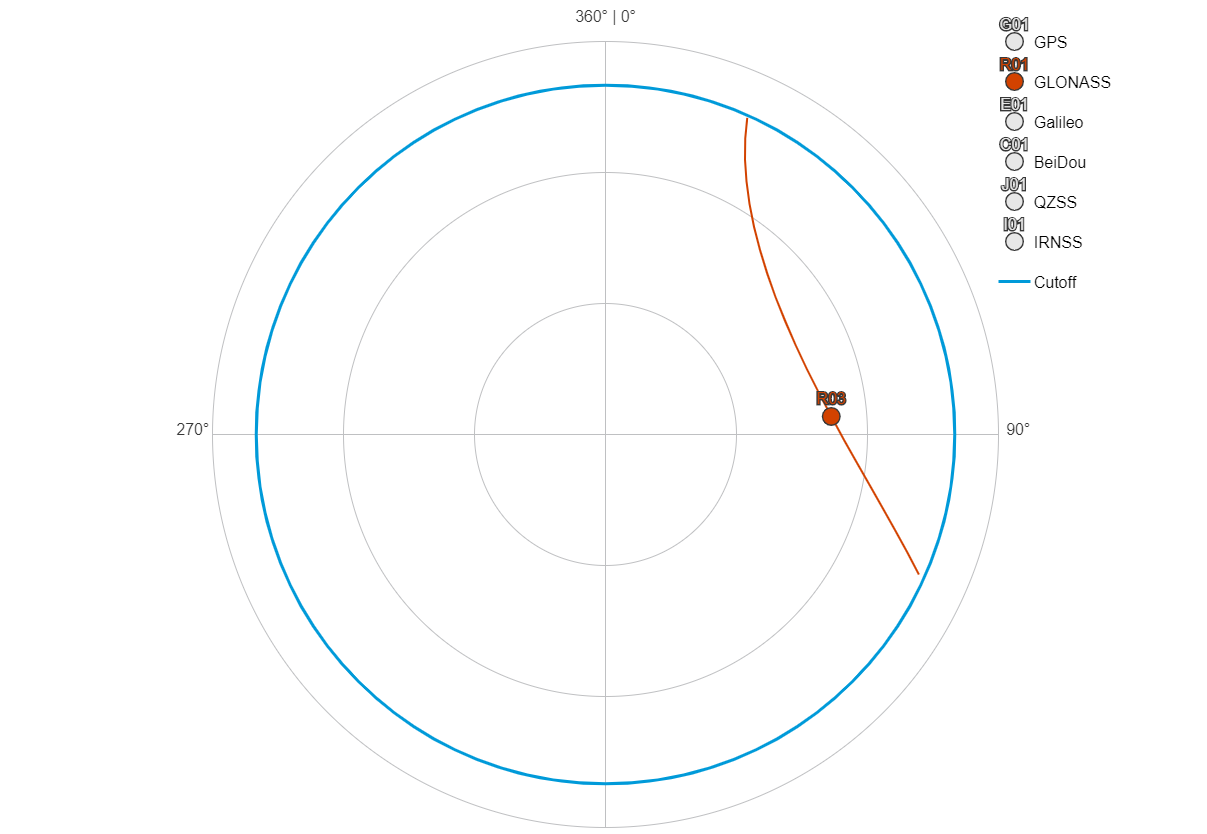


Рисунок 10 – SkyView спутника ГЛОНАСС №3

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:

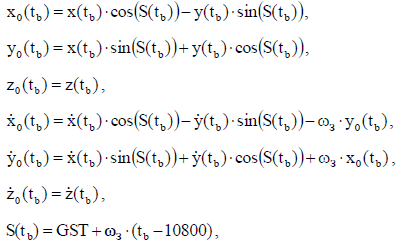
N4 – номер четырёхлетнего периода эфемерид, Nt – номер эфемеридных суток в четырёхлетнем периоде, tb – момент времени из оперативной информации ГЛОНАСС, координаты и составляющие вектора скорости центра масс НКА на момент времени tb.

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:

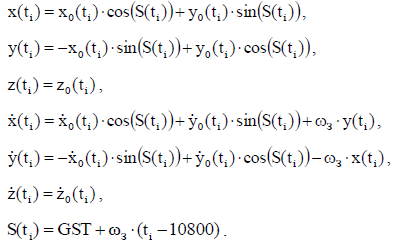


Дифференциальные уравнения движения интегрируются в прямоугольной инерциальной геоцентрической системе координат OX0Y0Z0, начало которой О совпадает с началом координат системы ПЗ-90, ось OX0 направлена в точку весеннего равноденствия, ось OZ0 направлена на Северный полюс мира и совпадает с осью OZ системы 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:

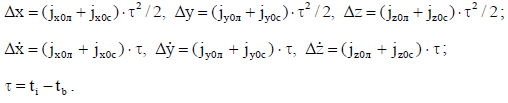


После интегрирования, полученные в инерциальной системе координат OX0Y0Z0 координаты центра масс , , и составляющие его вектора скорости , , могут быть пересчитаны в связанную с Землей систему ПЗ-90 Oxyz по формулам:



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.

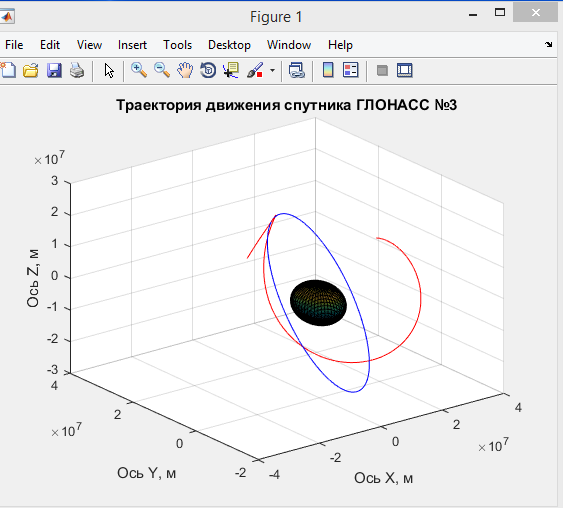


Рисунок 11 – Траектория движения спутника ГЛОНАСС №3 в системе координат ПЗ-90 (красная линия) и инерциальной системе координат (синяя линия)

# **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.

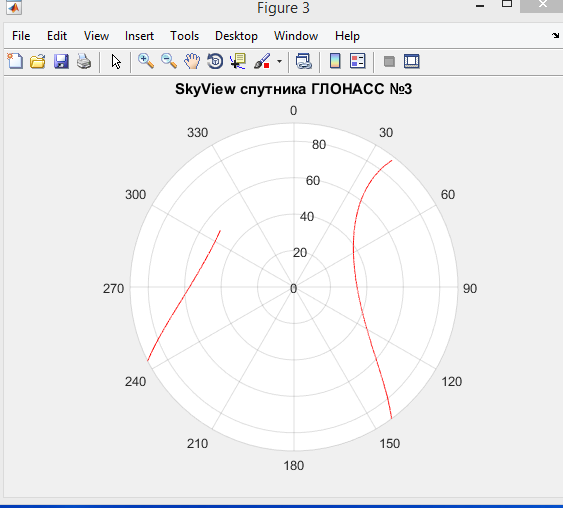


Рисунок 12 – SkyView спутника ГЛОНАСС №3

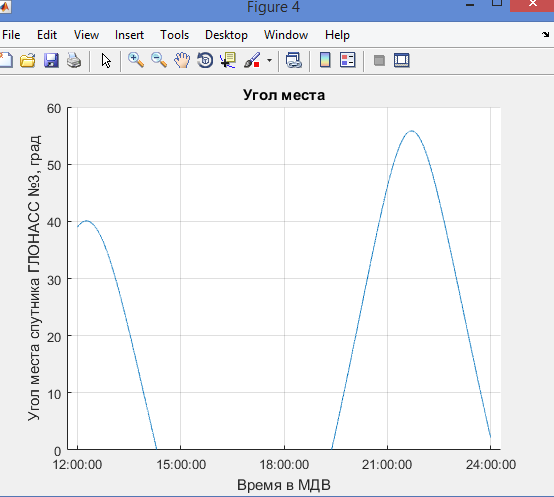


Рисунок 13 – График угла места спутника ГЛОНАСС №3

# **Этап 3. Реализация**

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. Вызов функции не должен приводить к выбросу исключений или утечкам памяти при любом наборе входных данных.

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 дифференциального уравнения методом Рунге-Кутты.

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.

Во время второго теста должно вычисляться и выводиться средняя длительность исполнения функции. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал (как на предыдущем этапе).

Требуется провести проверку на утечки памяти с помощью утилиты valgrind.

Для расчёта используется программа Visual Studio.

# **3.1. Перенос из Matlab в С++**

Для начала распишем программу расчёта положения спутника ГЛОНАСС в Visual Studio, используя тот же алгоритм. Поскольку здесь отсутствуют готовые функции решения дифференциального уравнения методом Рунге-Кутты, необходимо выполнить данный метод самостоятельно. Метод представлен на рисунке 14.

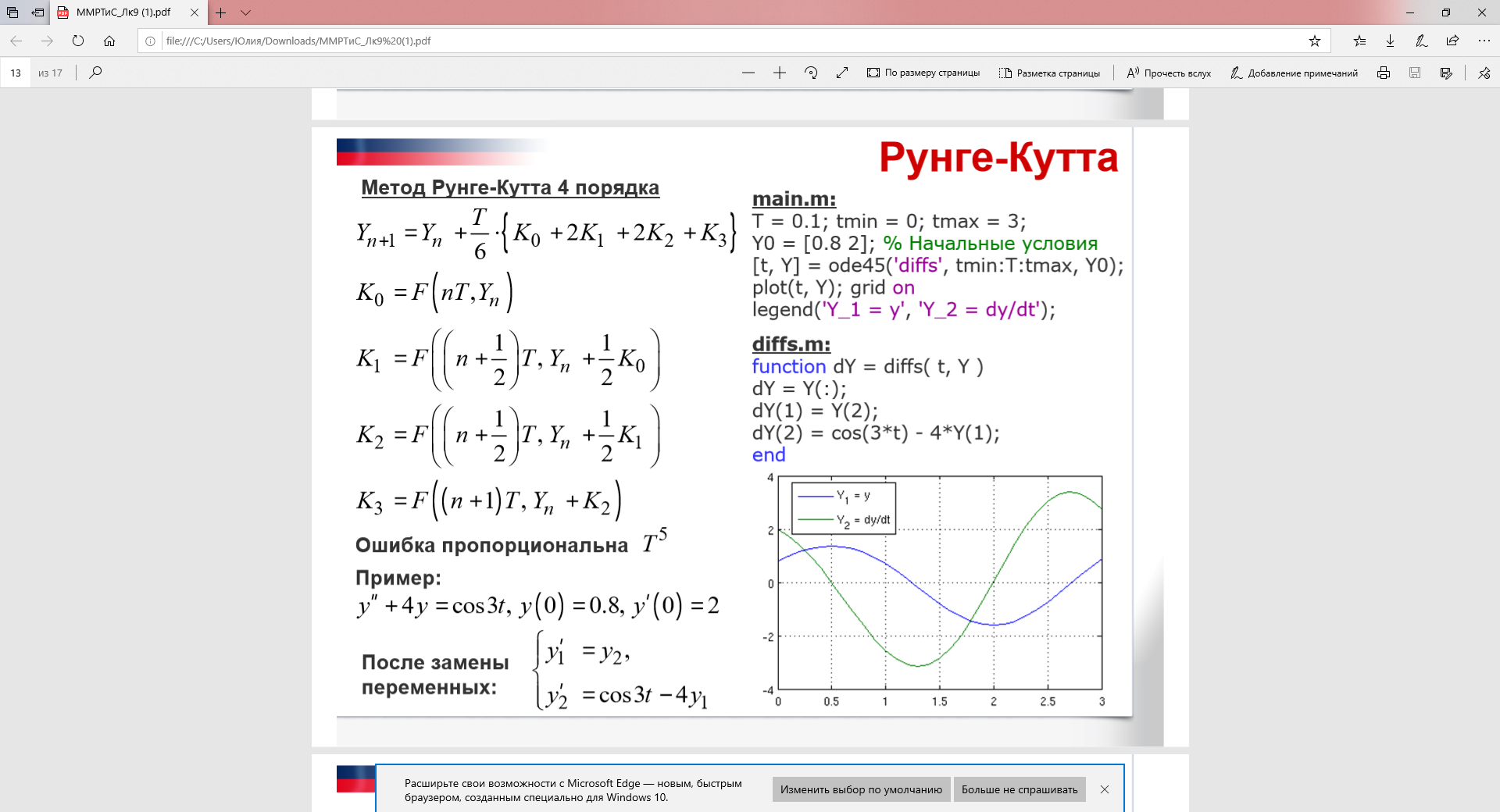


Рисунок 14 – Метод Рунге-Кутта 4-ого порядка

Код программы приведён в приложении 2.

# **4 Заключение**

В ходе выполнения первого этапа были получены следующие результаты:

* Обработаны данные от приёмника ГНСС с помощью RTKNAVI из пакета RTKLIB и преобразованы в таблицу эфемерид спутников ГЛОНАСС, в том числе и собственного спутника (№3), данные которого необходимы для следующих этапов в качестве проверочных значений;
* Эфемериды собственного спутника в конвертированном gnav-файле RINEX с помощью RTKCONV из пакета RTKLIB;
* График угла места собственного спутника на заданный интервал времени, а также установлено количество появлений спутника в заданный промежуток времени;
* SkyView по данным Trimble GNSS Planning Online на заданный интервал времени.

В ходе выполнения второго этапа были получены следующие результаты:

* Реализован алгоритм расчёта положения спутника ГЛОНАСС №3 на интервале времени с 12:00 10.02.2020 до 00:00 11.02.2020 в программе MATLAB.
* В качестве исходных данных использовались эфемериды, полученные на предыдущем этапе. На всём интервале использовались одни и те же эфемериды.
* Получены графики для траектории движения спутника ГЛОНАСС №3, а также график угла места и SkyView.

В ходе выполнения второго этапа были получены следующие результаты:

* Написана программа для реализации алгоритма расчёта положения спутника ГЛОНАСС №3.

## **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».

# **ПРИЛОЖЕНИЕ 1**

Листинг программы для этапа 2

Расчёт с учётом функции дифференциальных уравнений (основная программа):

clc

clear

%% Точный алгоритм пересчёта координат и составляющих вектора скорости центра масс НКА на заданный момент времени шкалы МВД

%% Эфемериды НКА 3 системы ГЛОНАСС в системе ПЗ-90.11

%Расчёт времени формата ГЛОНАСС

N4=floor((2020-1996)/4)+1;%Номер текущего четырёхлетия

Nt=365\*(2020-1996-4\*(N4-1))+31+25+1;%Номер текущих суток

tb=13\*60\*60 + 45\*60 + 18 + 10800;%момент по шкале МДВ, к которому привязаны эфемериды ГЛОНАСС, в сек

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

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

Ts = 1;

ti = Toe:Ts:Tof;

X = 23036950.68; %% м

Y = -9091173.34; %% м

Z = 6041059.08; %% м

VX = 755.86033; %% м/с

VY = -358.52718; %% м/с

VZ = -3447.90649; %% м/с

AX = 0.0000056; %% м/с^2

AY = 0.0000000; %% м/с^2

AZ = -0.0000028; %% м/с^2

%% Константы

omega\_z = 7.2921151467e-5;

%% Расчёт текущей юлианской даты

JD0 = 1461\*(N4 - 1) + Nt + 2450008.5 - (Nt -3)/25;

%% Среднее звёздное время по Гринвичу

%% Время от эпохи 2000 года 1 января до текущей эпохи

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

%% Угол поворота Земли, рад

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

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

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

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

%% Перевод в п/у инерциальную геоцентрическую систему

S = GMST + omega\_z\*(tb - 3\*60\*60);

Xate=X\*cos(S)-Y\*sin(S);

Yate=X\*sin(S)+Y\*cos(S);

Zate=Z;

Vxate=VX\*cos(S)-VY\*sin(S)-omega\_z\*Yate;

Vyate=VX\*sin(S)+VY\*cos(S)+omega\_z\*Xate;

Vzate=VZ;

Axte=AX\*cos(S)-AY\*sin(S);

Ayte=AX\*sin(S)+AY\*cos(S);

Azte=AZ;

%% Метод Рунге-Кутты

%% Начальные условия

res0 = [Xate Yate Zate Vxate Vyate Vzate];

[t, res] = ode45('diffs', tb:-Ts:ti(1), res0);

res1 = res(end:-1:2,:);

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

[t, res] = ode45('diffs', tb:Ts:ti(end), res0);

res1 = [res1; res];

t1 = [t1;t];

%% Учёт ускорений

tau1 = t1 - tb;

AXTE = AX\*(tau1.^2)/2;

AYTE = AY\*(tau1.^2)/2;

AZTE = AZ\*(tau1.^2)/2;

delta\_VX = AX\*tau1;

delta\_VY = AY\*tau1;

delta\_VZ = AZ\*tau1;

delta\_A = [AXTE AYTE AZTE delta\_VX delta\_VY delta\_VZ];

res1 = res1 + delta\_A;

%% Пересчёт координат в ПЗ-90

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

pz90(:,1) = res1(:,1).\*cos(S) + res1(:,2).\*sin(S);

pz90(:,2) = -res1(:,1).\*sin(S) + res1(:,2).\*cos(S);

pz90(:,3) = res1(:,3);

%% Координаты корпуса Е

%широта 55° 45' 24.0765",переводя получим 55.756687916667

N = 55.756687916667\*pi/180;% широта [рад]

%долгота 37° 42' 11.0779" переводя в десятичные доли градуса получаем 37.703077194444

E=37.703077194444\*pi/180;% долгота [рад]

H = 500; % высота [м]

cord\_E = [N E H];

%Skyplot

for i = 1:length(pz90(:,1))

[X(i) Y(i) Z(i)] = ecef2enu(pz90(i,1),pz90(i,2),pz90(i,3),N,E,H,wgs84Ellipsoid,'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

%Построение графиков

figure(1)

[Xx,Yy,Zz]=sphere(50);

Rz=6371000;%радиус Земли

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

hold on

grid on

plot3(res1(:,1), res1(:,2), res1(:,3), 'b')

xlabel('Ось Х, м')

ylabel('Ось Y, м')

zlabel('Ось Z, м')

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

grid on

plot3(pz90(:,1),pz90(:,2),pz90(:,3),'r')

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

xlabel('Ось Х, м')

ylabel('Ось Y, м')

zlabel('Ось Z, м')

hold off

%SkyPlot

figure (3)

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(4);

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')

Функция дифференциальных уравнений (подпрограмма, на которую ссылается основная программа):

%% Система дифф уравнений

function dres = diffs(t, res)

J02 = -1082.63\*10^-6;

ae = 6378136;

GM = 398600441.8e6;

Xate=res(1);

Yate=res(2);

Zate=res(3);

r=sqrt(Xate^2 + Yate^2 + Zate^2);

GM1 = GM/r^2;

x01 = Xate/r;

y01 = Yate/r;

z01 = Zate/r;

p = ae/r;

dres = res(:);

dres(1) = res(4);

dres(2) = res(5);

dres(3) = res(6);

dres(4) = -GM1\*x01 + 1.5\*J02\*GM1\*x01\*(p^2)\*(1 - 5\*z01^2);

dres(5) = -GM1\*y01 + 1.5\*J02\*GM1\*y01\*(p^2)\*(1 - 5\*z01^2);

dres(6) = -GM1\*z01 + 1.5\*J02\*GM1\*z01\*(p^2)\*(3 - 5\*z01^2);

end

# **ПРИЛОЖЕНИЕ 2**

Листинг программы для 3-его этапа

Расчёт по алгоритму, реализованному в Matlab

#include <iostream>

#include <cmath>

using namespace std;

int main()

{// Координаты

X = 23036950.68;

Y = -9091173.34;

Z = 6041059.08;

// Скорости

VX = 755.86033;

VY = -358.52718;

VZ = -3447.90649;

// Ускорения

AX = 0.0000056;

AY = 0.0000000;

AZ = -0.0000028;

// Константы

omega\_z = 7.2921151467e-5;

J02 = -1082.63 \* 10 ^ -6;

GM = 398600441.8e6;

// Номер текущего четырёхлетия

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

// Номер текущих суток

Nt = 365 \* (2020 - 1996 - 4 \* (N4 - 1)) + 31 + 25 + 1;

// Расчёт текущей юлианской даты

JD0 = 1461 \* (N4 - 1) + Nt + 2450008.5 - (Nt - 3) / 25;

// Момент по шкале МДВ, к которому привязаны эфемериды ГЛОНАСС

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

// Начальное время

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

// Конечное время

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

// Среднее звёздное время по Гринвичу

// Время от эпохи 2000 года 1 января до текущей эпохи

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

// Угол поворота Земли, рад

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

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

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

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

// Перевод в п/у инерциальную геоцентрическую систему

S = GMST + omega\_z \* (tb - 3 \* 60 \* 60);

Xate = X \* cos(S) - Y \* sin(S);

Yate = X \* sin(S) + Y \* cos(S);

Zate = Z;

Vxate = VX \* cos(S) - VY \* sin(S) - omega\_z \* Yate;

Vyate = VX \* sin(S) + VY \* cos(S) + omega\_z \* Xate;

Vzate = VZ;

Axte = AX \* cos(S) - AY \* sin(S);

Ayte = AX \* sin(S) + AY \* cos(S);

Azte = AZ;

// Метод Рунге-Кутты

// Начальные условия

Y0 = [Xate Yate Zate Vxate Vyate Vzate];

// Ситема диффуров

int diffs(double tb)

{

double Xate1 = Y.X;// по х

double Yate1 = Y.Y;// по у

double Zate1 = Y.Z;// по z

double r = sqrt(Xate1 ^ 2 + Yate1 ^ 2 + Zate1 ^ 2);

double GM1 = GM / r ^ 2;

double x01 = Xate / r;

double y01 = Yate / r;

double z01 = Zate / r;

double p = ae / r;

dY.X = Y.VX;

dY.Y = Y.VY;

dY.Z = Y.VZ;

dY.VX = -GM1 \* x01 - (double)1.5 \* J02 \* GM1 \* x01 \* p ^ 2 \* (1 - (double)5 \* z01 ^ 2);

dY.VY = -GM1 \* y01 - (double)1.5 \* J02 \* GM1 \* y01 \* p ^ 2 \* (1 - (double)5 \* z01 ^ 2);

dY.VZ = -GM1 \* z01 - (double)1.5 \* J02 \* GM1 \* z01 \* p ^ 2 \* (3 - (double)5 \* z01 ^ 2);

return dY;

}

int Runge\_Kutta(N, double h)

{

if (N == 0 || h == 0) return 0;

for (K = 1; K < N; K++)

K1 = diffs(0, Y[K - 1])

Y2.X = Y[K - 1].X + h \* K1.X / (double)2;

Y2.Y = Y[K - 1].Y + h \* K1.Y / (double)2;

Y2.Z = Y[K - 1].Z + h \* K1.Z / (double)2;

Y2.VX = Y[K - 1].VX + h \* K1.VX / (double)2;

Y2.VY = Y[K - 1].VY + h \* K1.VY / (double)2;

Y2.VZ = Y[K - 1].VZ + h \* K1.VZ / (double)2;

K2 = diffs(0 + h / 2, Y2);

Y3.X = Y[K - 1].X + h \* K2.X / (double)2;

Y3.Y = Y[K - 1].Y + h \* K2.Y / (double)2;

Y3.Z = Y[K - 1].Z + h \* K2.Z / (double)2;

Y3.VX = Y[K - 1].VX + h \* K2.VX / (double)2;

Y3.VY = Y[K - 1].VY + h \* K2.VY / (double)2;

Y3.VZ = Y[K - 1].VZ + h \* K2.VZ / (double)2;

K3 = diffs(0 + h / 2, 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 = diffs(0 + h, Y4);

Y[K].X = Y[K - 1].X + h / (double)6 \* (K1.X + 2 \* K2.X + 2 \* K3.X + K4.X);

Y[K].Y = Y[K - 1].X + h / (double)6 \* (K1.Y + 2 \* K2.Y + 2 \* K3.Y + K4.Y);

Y[K].Z = Y[K - 1].X + h / (double)6 \* (K1.Z + 2 \* K2.Z + 2 \* K3.Z + K4.Z);

Y[K].VX = Y[K - 1].X + h / (double)6 \* (K1.VX + 2 \* K2.VX + 2 \* K3.VX + K4.VX);

Y[K].VY = Y[K - 1].X + h / (double)6 \* (K1.VY + 2 \* K2.VY + 2 \* K3.VY + K4.VY);

Y[K].VZ = Y[K - 1].X + h / (double)6 \* (K1.VZ + 2 \* K2.VZ + 2 \* K3.VZ + K4.VZ);

cout << endl;

for (int K = 0; K <= N; K++) {

cout << "Y[" << K << "]=" << Y[K] << " ";

}

return 0;

}// Выходной массив

{

int Y[K];

int Y\_out[K];

cin >> K;

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

{

cin >> Y[i];

}

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

{

if (Y[k] != K)

{

Y\_out[i] = Y[i];

}

}

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

{

cout << Y\_out[i];

}

}

}

// Учёт ускорений

}

double tau1 = (double)Toe - (double)tb;

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

AXTE = AX \* (tau1 ^ 2) /(double)2;

AYTE = AY \* (tau1 ^ 2) /(double)2;

AZTE = AZ \* (tau1 ^ 2) /(double)2;

delta\_VX = AX \* tau1;

delta\_VY = AY \* tau1;

delta\_VZ = AZ \* tau1;

delta\_Y = [AXTE AYTE AZTE delta\_VX delta\_VY delta\_VZ];

Y\_out1[i] = Y\_out[i] + delta\_Y;

}

}