НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«МЭИ»

ИНСТИТУТ РАДИОТЕХНИКИ И ЭЛЕКТРОНИКИ

КАФЕДРА РАДИОТЕХНИЧЕСКИХ СИСТЕМ

КУРСОВОЙ ПРОЕКТ

по дисциплине

Аппаратура потребителей спутниковых радионавигационных систем

«Разработка модуля расчет координат спутника ГЛОНАСС»

ФИО СТУДЕНТА: ЖЕРЕБИН В.Р.

ГРУППА: ЭР-15-15

ВАРИАНТ №:5

ДАТА:_____

ПОДПИСЬ:_____

ФИО ПРЕПОДАВАТЕЛЯ: КОРОГОДИН И.В.

ОЦЕНКА: _____

МОСКВА, 2020

Содержание

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 C++, 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 C/C++, 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):

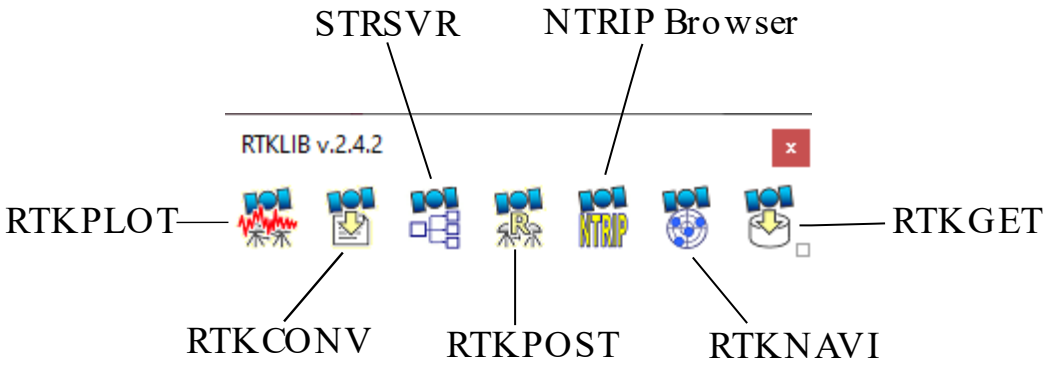


Рисунок 1 – Окно программы RTKLIB v.2.4.2

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

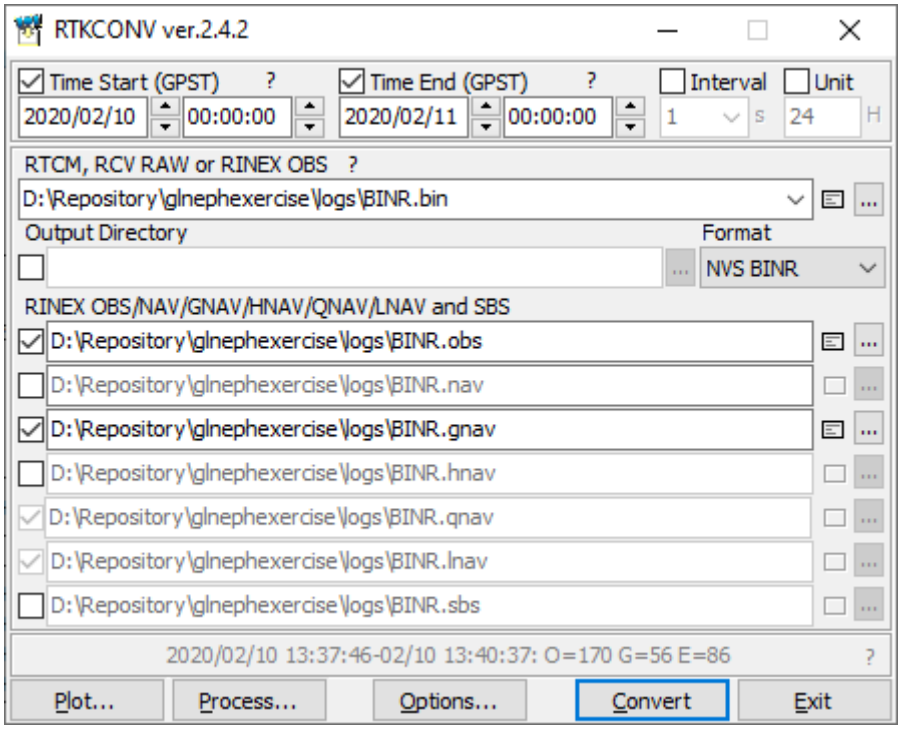


Рисунок 2 – Окно программы RTKCONV ver.2.4.2

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, R11, R12, 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.

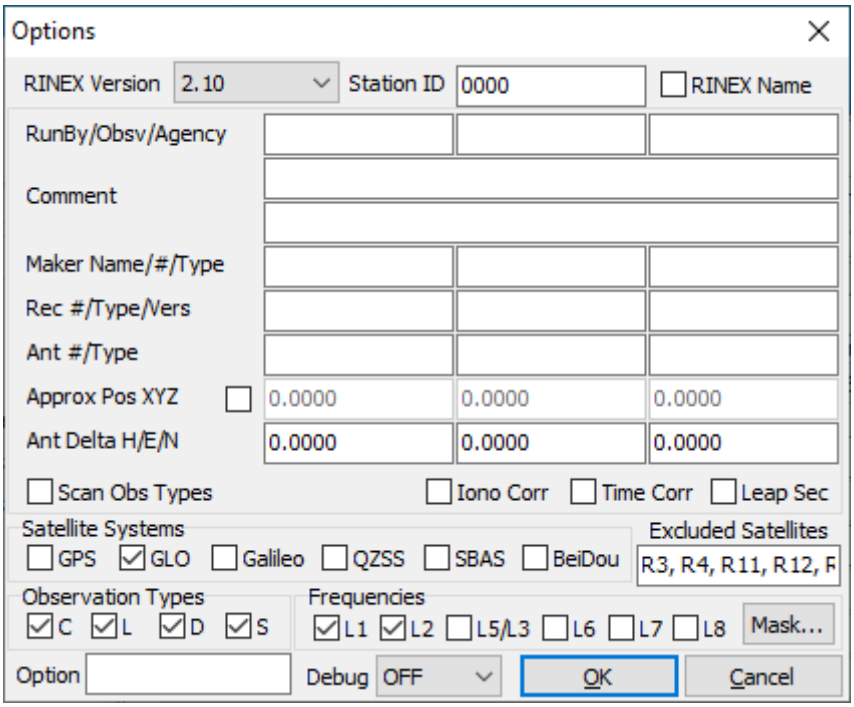
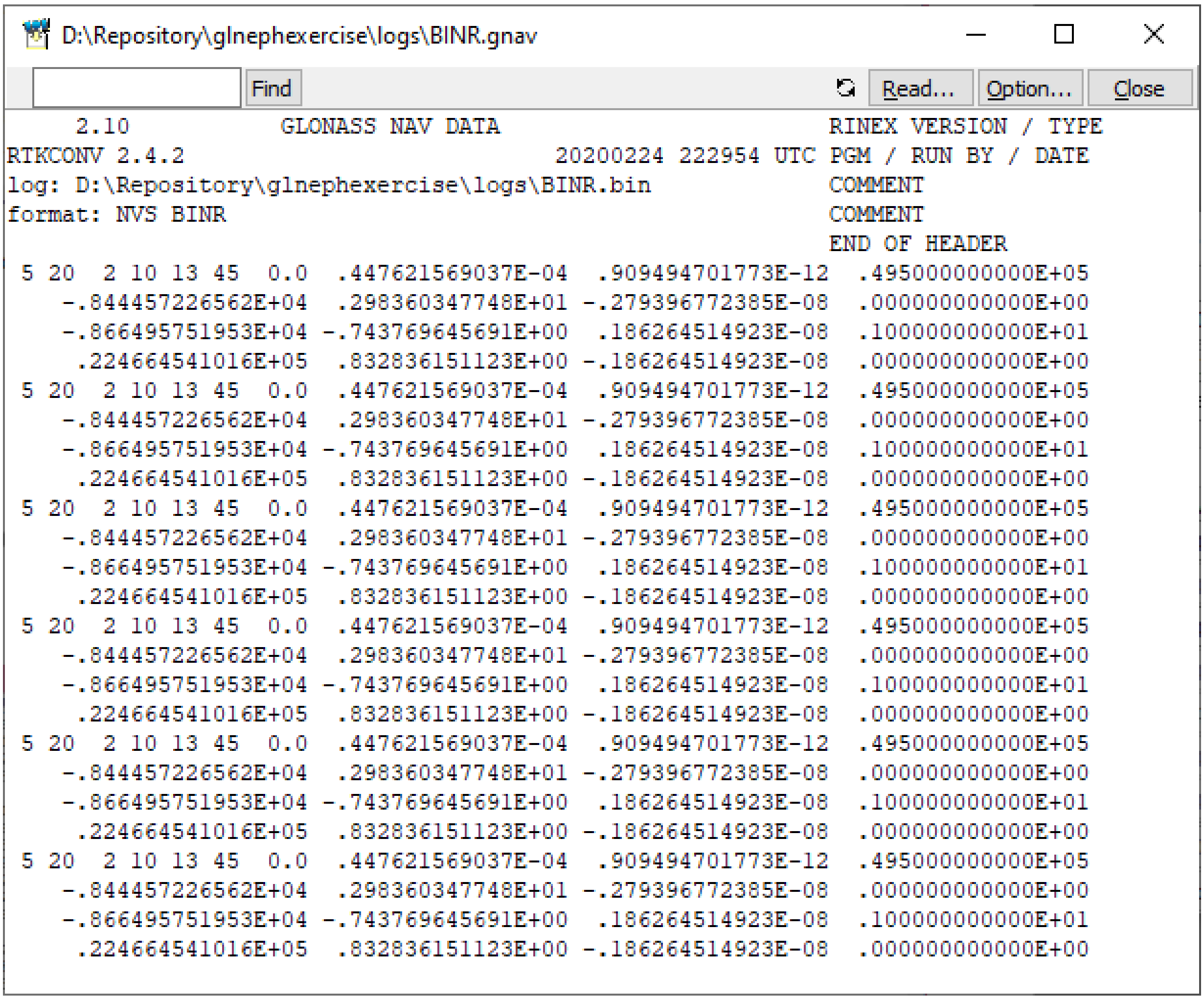


Рисунок 3 – Окно настроек программы RTKCONV ver.2.4.2

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).



```
D:\Repository\glnephexercise\logs\BINR.gnav
     2.10           GLONASS NAV DATA                        RINEX VERSION / TYPE
RTKCONV 2.4.2                           20200224 222954 UTC PGM / RUN BY / DATE
log: D:\Repository\glnephexercise\logs\BINR.bin             COMMENT
format: NVS BINR                                            COMMENT
                                                            END OF HEADER
 5 20  2 10 13 45  0.0  .447621569037E-04  .909494701773E-12  .495000000000E+05
   -.844457226562E+04  .298360347748E+01 -.279396772385E-08  .000000000000E+00
   -.866495751953E+04 -.743769645691E+00  .186264514923E-08  .100000000000E+01
    .224664541016E+05  .832836151123E+00 -.186264514923E-08  .000000000000E+00
 5 20  2 10 13 45  0.0  .447621569037E-04  .909494701773E-12  .495000000000E+05
   -.844457226562E+04  .298360347748E+01 -.279396772385E-08  .000000000000E+00
   -.866495751953E+04 -.743769645691E+00  .186264514923E-08  .100000000000E+01
    .224664541016E+05  .832836151123E+00 -.186264514923E-08  .000000000000E+00
 5 20  2 10 13 45  0.0  .447621569037E-04  .909494701773E-12  .495000000000E+05
   -.844457226562E+04  .298360347748E+01 -.279396772385E-08  .000000000000E+00
   -.866495751953E+04 -.743769645691E+00  .186264514923E-08  .100000000000E+01
    .224664541016E+05  .832836151123E+00 -.186264514923E-08  .000000000000E+00
 5 20  2 10 13 45  0.0  .447621569037E-04  .909494701773E-12  .495000000000E+05
   -.844457226562E+04  .298360347748E+01 -.279396772385E-08  .000000000000E+00
   -.866495751953E+04 -.743769645691E+00  .186264514923E-08  .100000000000E+01
    .224664541016E+05  .832836151123E+00 -.186264514923E-08  .000000000000E+00
 5 20  2 10 13 45  0.0  .447621569037E-04  .909494701773E-12  .495000000000E+05
   -.844457226562E+04  .298360347748E+01 -.279396772385E-08  .000000000000E+00
   -.866495751953E+04 -.743769645691E+00  .186264514923E-08  .100000000000E+01
    .224664541016E+05  .832836151123E+00 -.186264514923E-08  .000000000000E+00
 5 20  2 10 13 45  0.0  .447621569037E-04  .909494701773E-12  .495000000000E+05
   -.844457226562E+04  .298360347748E+01 -.279396772385E-08  .000000000000E+00
   -.866495751953E+04 -.743769645691E+00  .186264514923E-08  .100000000000E+01
    .224664541016E+05  .832836151123E+00 -.186264514923E-08  .000000000000E+00
```

Рисунок 4 – Эфемериды спутника ГЛОНАСС №5 в .gnav файле

После чего нажимаем «Process…», запускается программа RTKPOST (Рисунок 5) для решения навигационной задачи. Аналогичным образом выбираем Time Start (GPST), Time End (GPST), и ставим время интервала наблюдений с 00:00 10.02.20 до 00:00 11.02.20, указываем путь к файлам наблюдений форматов .obs и .gnav.

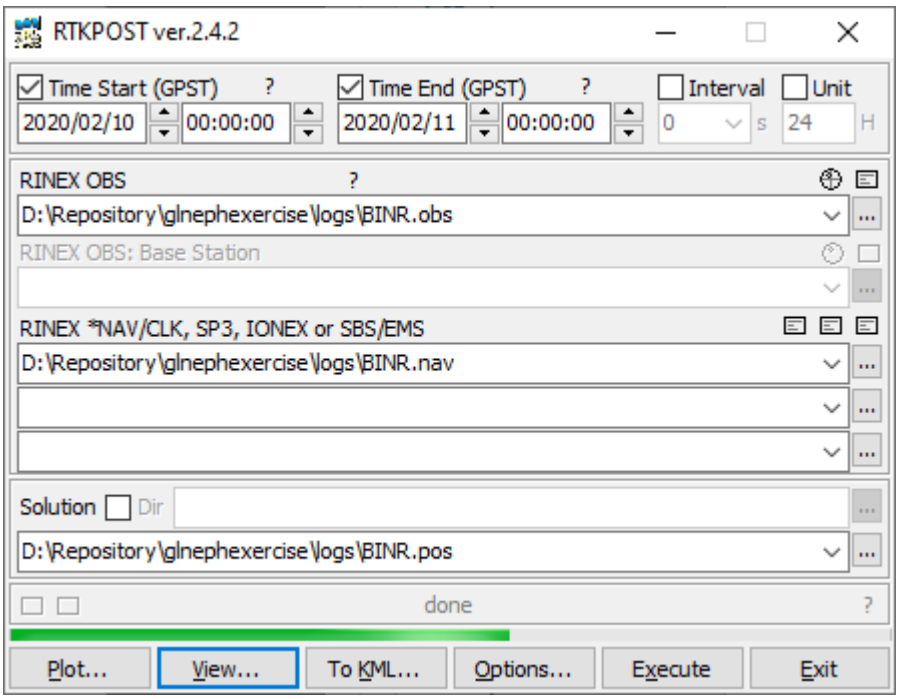


Рисунок 5 – Окно программы RTKPOST ver.2.4.2

После нажатия кнопки «Execute» программа производит вторичную обработку, результаты которой записываются в файл с расширением .pos. Нажатие кнопки «Plot..» открывает программу RTKPLOT, в которой можно увидеть графическое отображение некоторых значений, к примеру отношение сигнал/шум и угла места (Рисунок 6):

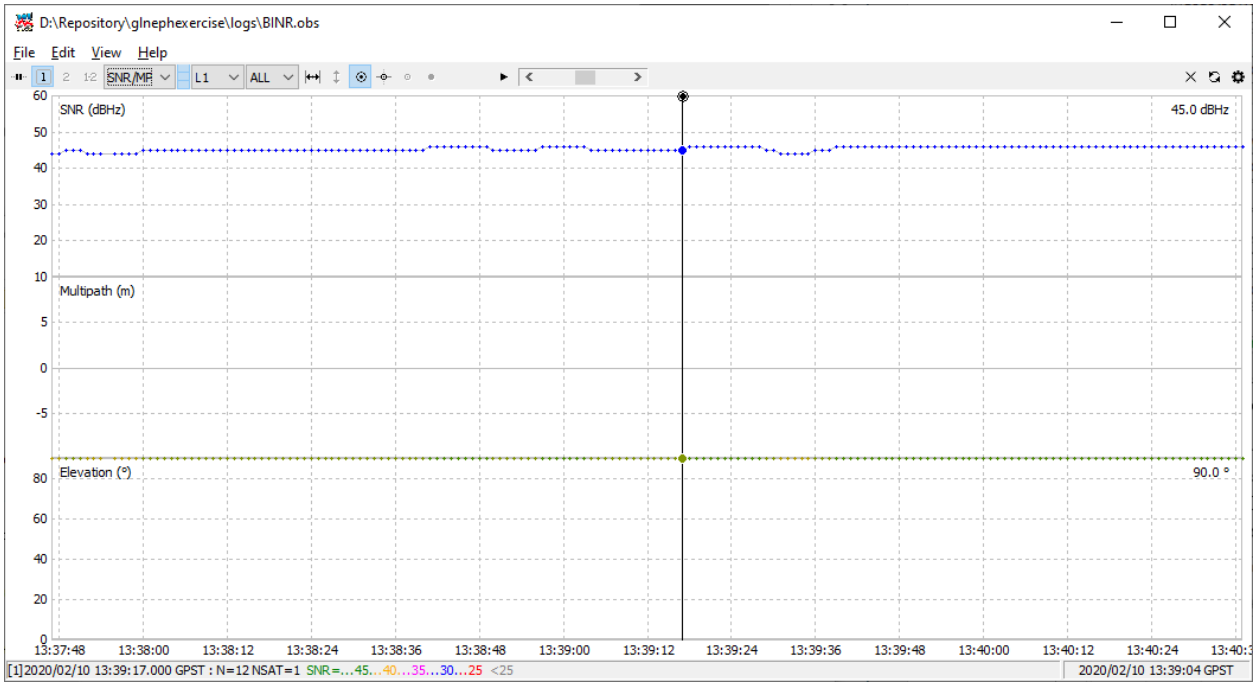
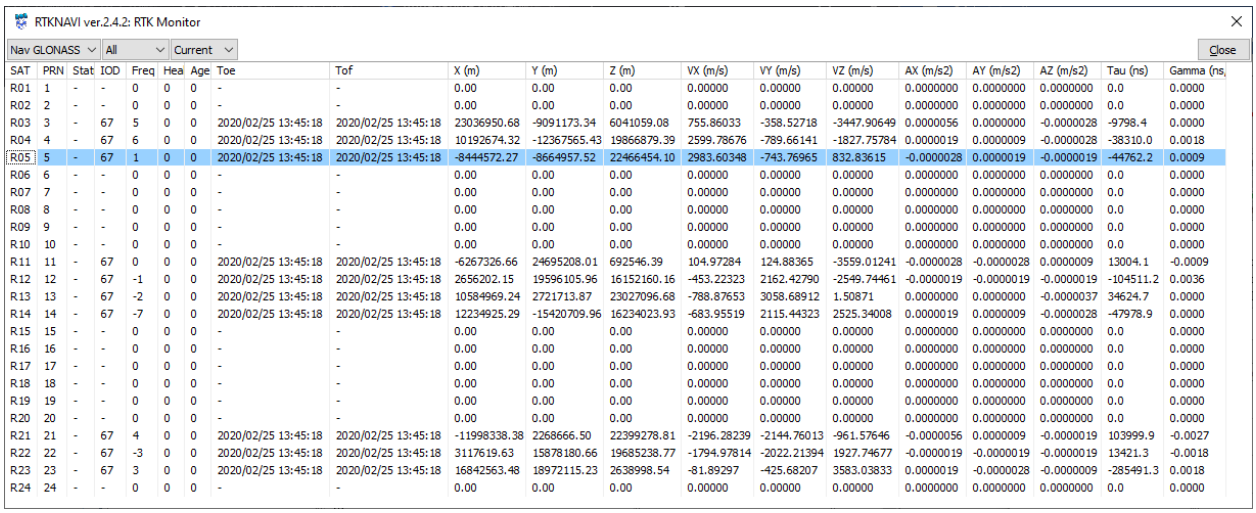


Рисунок 6 – Графики для спутника ГЛОНАСС №5

Теперь получим эфемериды спутника по данным RTKNAVI из состава RTKLIB. Программа RTKNAVI позволяет вывести таблицу текущих и предыдущих эфемерид (Рисунок 7).



SAT	PRN	Stat	IOD	Freq	Hea	Age	Toe	Tof	X (m)	Y (m)	Z (m)	VX (m/s)	VY (m/s)	VZ (m/s)	AX (m/s2)	AY (m/s2)	AZ (m/s2)	Tau (ns)	Gamma (ns
R01	1	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R02	2	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R03	3	-	67	5	0	0	2020/02/25 13:45:18	2020/02/25 13:45:18	23036950.68	-9091173.34	6041059.08	755.86033	-358.52718	-3447.90649	0.0000056	0.0000000	-0.0000028	-9798.4	0.0000
R04	4	-	67	6	0	0	2020/02/25 13:45:18	2020/02/25 13:45:18	10192674.32	-12367565.43	19866879.39	2599.78676	-789.66141	-1827.75784	0.0000019	0.0000009	-0.0000028	-38310.0	0.0018
R05	5	-	67	1	0	0	2020/02/25 13:45:18	2020/02/25 13:45:18	-8444572.27	-8664957.52	22466454.10	2983.60348	-743.76965	832.83615	-0.0000028	0.0000019	-0.0000019	-44762.2	0.0009
R06	6	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R07	7	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R08	8	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R09	9	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R10	10	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R11	11	-	67	0	0	0	2020/02/25 13:45:18	2020/02/25 13:45:18	-6267326.66	24695208.01	692546.39	104.97284	124.88365	-3559.01241	-0.0000028	-0.0000028	0.0000009	13004.1	-0.0009
R12	12	-	67	-1	0	0	2020/02/25 13:45:18	2020/02/25 13:45:18	2656202.15	19596105.96	16152160.16	-453.22323	2162.42790	-2549.74461	-0.0000019	-0.0000019	-0.0000019	-104511.2	0.0036
R13	13	-	67	-2	0	0	2020/02/25 13:45:18	2020/02/25 13:45:18	10584969.24	2721713.87	23027096.68	-788.87653	3058.68912	1.50871	0.0000000	0.0000000	-0.0000037	34624.7	0.0000
R14	14	-	67	-7	0	0	2020/02/25 13:45:18	2020/02/25 13:45:18	12234925.29	-15420709.96	16234023.93	-683.95519	2115.44323	2525.34008	0.0000019	0.0000009	-0.0000028	-47978.9	0.0000
R15	15	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R16	16	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R17	17	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R18	18	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R19	19	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R20	20	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000
R21	21	-	67	4	0	0	2020/02/25 13:45:18	2020/02/25 13:45:18	-11998338.38	2268666.50	22399278.81	-2196.28239	-2144.76013	-961.57646	-0.0000056	0.0000009	-0.0000019	103999.9	-0.0027
R22	22	-	67	-3	0	0	2020/02/25 13:45:18	2020/02/25 13:45:18	3117619.63	15878180.66	19685238.77	-1794.97814	-2022.21394	1927.74677	-0.0000019	-0.0000019	-0.0000019	13421.3	-0.0018
R23	23	-	67	3	0	0	2020/02/25 13:45:18	2020/02/25 13:45:18	16842563.48	18972115.23	2638998.54	-81.89297	-425.68207	3583.03833	0.0000019	-0.0000028	-0.0000009	-285491.3	0.0018
R24	24	-	-	0	0	0	-	-	0.00	0.00	0.00	0.00000	0.00000	0.00000	0.0000000	0.0000000	0.0000000	0.0	0.0000

Рисунок 7 – Окно программы RTKNAVI ver.2.4.2: RTK Monitor

Синий линей выделена строка, соответствующая спутнику ГЛОНАСС №5. Значения, полученные из этой таблицы, будут нужны в следующих этапах.

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

Необходимо построить график угла места от времени и SkyView собственного спутника на заданный интервал времени. Для этого воспользуемся интернет ресурсом Trimble GNSS Planning (https://www.gnssplanning.com). Во вкладке настроек (Settings) указываем координаты и время места преступления (Рисунок 8). Во вкладке библиотеки спутников (Satellite Library) отключаем отображение всех спутников, кроме заданного (Рисунок 9).

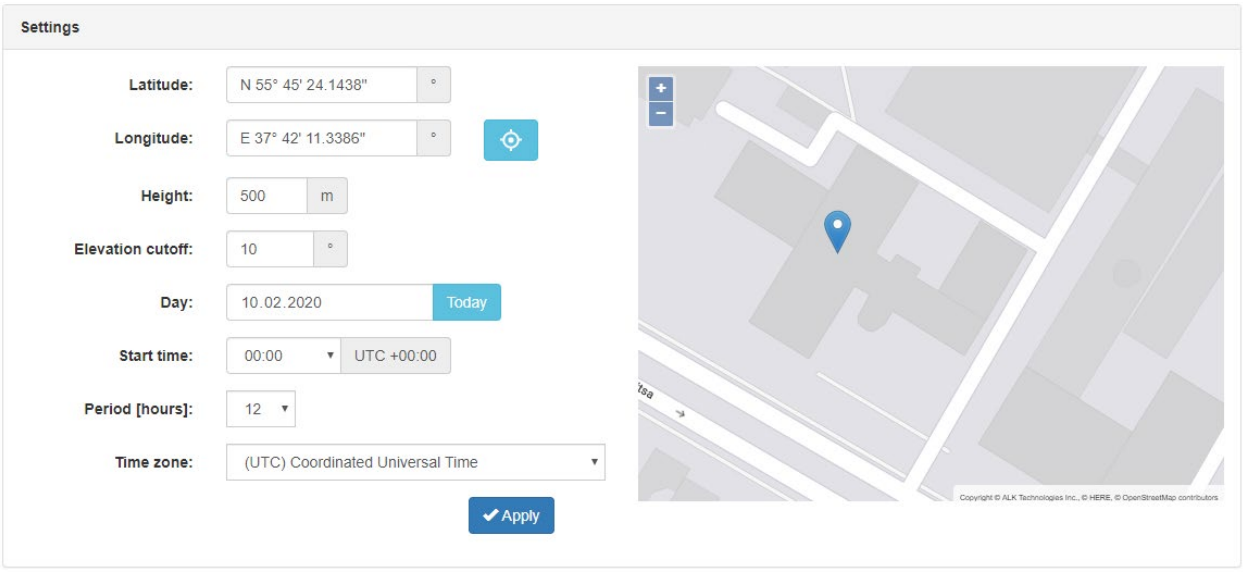


Рисунок 8 – Вкладка настроек (Settings) интернет ресурса Trimble GNSS Planning

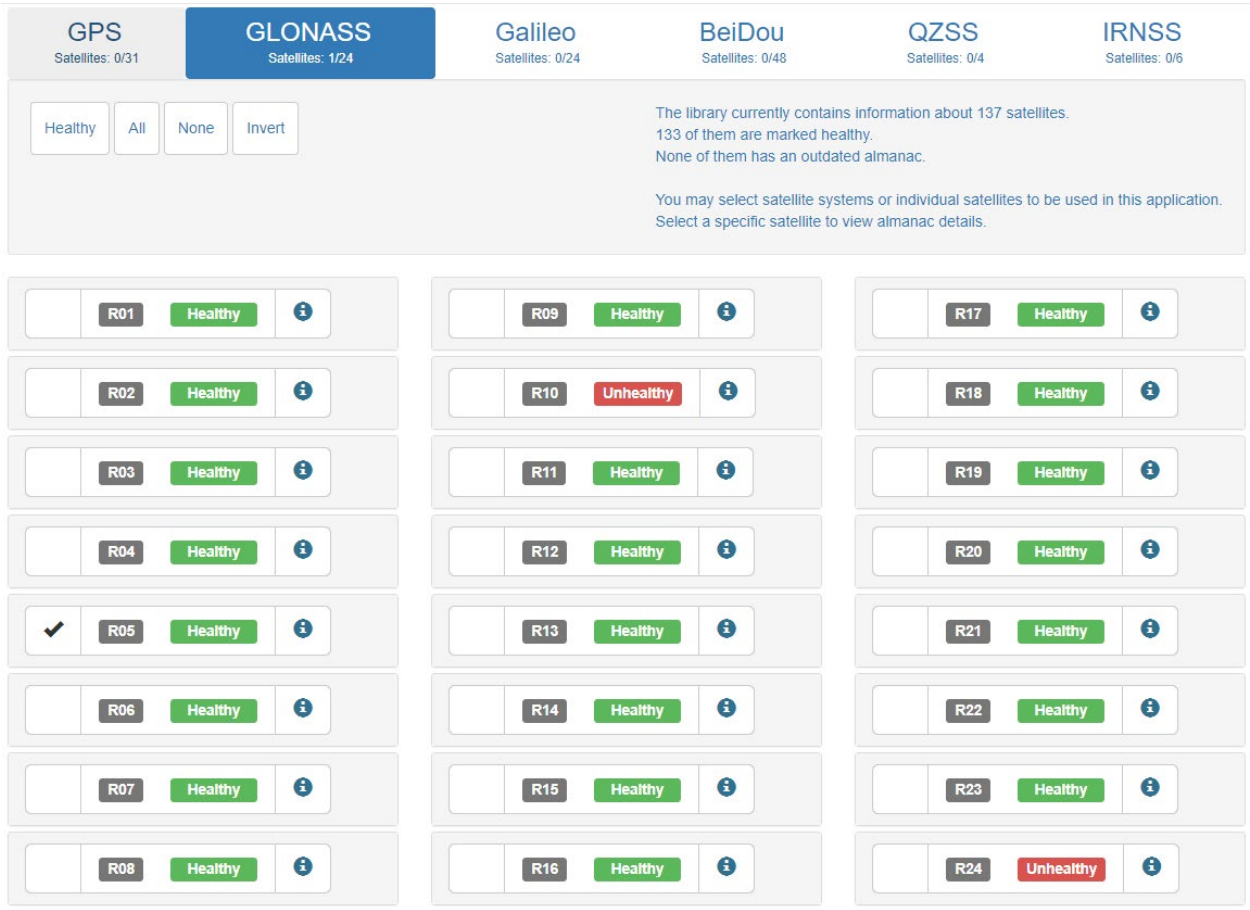


Рисунок 9 – Вкладка библиотека спутников (Satellite Library) интернет ресурса Trimble GNSS Planning

Для получения графика угла места, переходим во вкладку графики (Charts). По полученным данным, спутник был виден 2 раза (Рисунок 10). Первое появление с 13:40 до 15:30, второе с 22:20. Время указано по UTC +00:00.

Рисунок 10 – График угла места спутника ГЛОНАСС №5

Соответственно, перейдя во вкладку «Sky Plot», получаем карту небосвода (SkyView) (Рисунок 11). Траектория движения спутника, располагающаяся во второй четверти SkyView, соответствует первому появлению спутника, а в третей четверти, соответственно, второму.

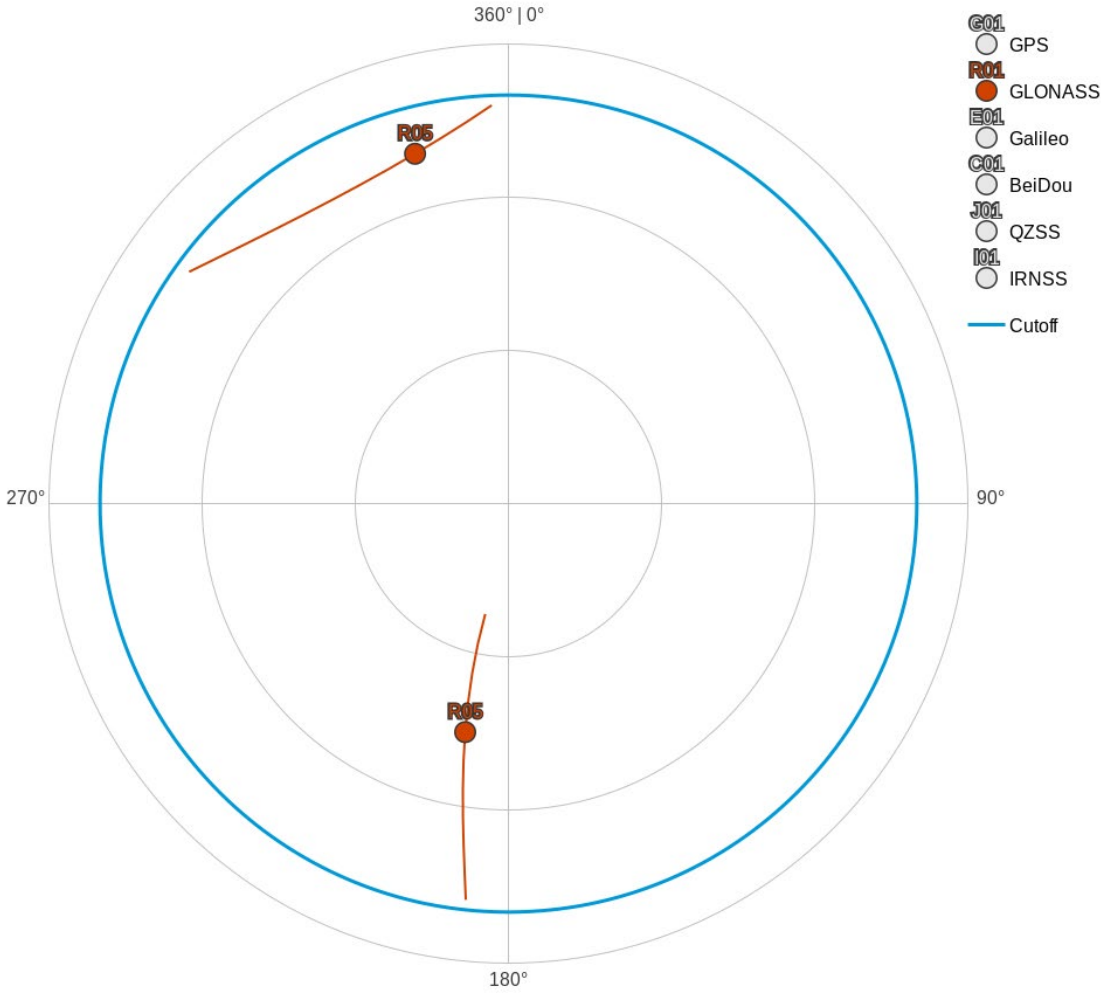


Рисунок 11 – SkyView спутника ГЛОНАСС №5

1.3 Заключение по результатам использования сторонних средств

В результате использования пакета RTKLIB и интернет-ресурса Trimble GNSS Planning Online, 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.

2 Моделирование

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 спутника ГЛОНАСС, полученные на предыдущем этапе, сведены в таблицу 1.

Таблица 1. Эфемериды спутника ГЛОНАСС №5

Параметр	Размерность	Значение
Toe/Tof	год/месяц/день час:минута:секунда	2020/02/10 13:45:18
$x(t_b)$	м	-8444572.27
$y(t_b)$		-8664957.52
$z(t_b)$		22466454.10
$\dot{x}(t_b)$	м/с	2983.60348
$\dot{y}(t_b)$		-743.76965
$\dot{z}(t_b)$		832.83615
$\ddot{x}(t_b)$	м/с^2	-0.0000028
$\ddot{y}(t_b)$		0.0000019
$\ddot{z}(t_b)$		-0.0000019

2.1 Алгоритм расчета положения спутника ГЛОНАСС

Необходимо построить трехмерные графики множества положений спутника №5 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.

В ИКД ГЛОНАСС приведены три алгоритма расчета положения спутника на заданным момент времени t_i шкалы МДВ по данным эфемерид:

- точный алгоритм (точный расчет на 30-минутном интервале);
- упрощенный алгоритм (более простой расчет на 30-минутном интервале);
- долговременный алгоритм (точный расчет на 4-часовом интервале).

Так как, допускается использовать одни и те же эфемериды на весь рассматриваемый интервал, то будем использовать точный алгоритм.

Эфемериды передаются в шкале времени UTC, а алгоритм использует шкалу МДВ, следовательно необходимо перевести время эфемерид в МДВ, добавив +3 часа.

Исходные данные, необходимые для пересчета эфемерид в соответствии с точным алгоритмом:

- N4 – номер эфемеридного четырехлетнего периода;
- NT – номер эфемеридных суток в эфемеридном четырехлетнем периоде;
- момент времени t_b из оперативной информации ГЛОНАСС;
- координаты и составляющие вектора скорости центра масс НКА на момент времени t_b из оперативной информации ГЛОНАСС;
- заданный момент времени t_i шкалы МДВ, на который необходимо пересчитать координаты и составляющие вектора скорости НКА.

Пересчет эфемерид потребителем с момента t_b шкалы МДВ на заданный момент времени t_i той же шкалы проводится методом численного интегрирования дифференциальных уравнений движения центра масс НКА. Эти уравнения движения определены в виде следующей системы:

$$\frac{dx_0}{dt} = Vx_0,$$
$$\frac{dy_0}{dt} = Vy_0,$$
$$\frac{dz_0}{dt} = Vz_0,$$
$$\frac{dVx_0}{dt} = -G\hat{M}\cdot\hat{x}_0 - \frac{3}{2}J_2^0 G\hat{M}\cdot\hat{x}_0\rho^2(1-5\hat{z}_0^{\,2}) + j_{x0c} + j_{x0л},$$
$$\frac{dVy_0}{dt} = -G\hat{M}\cdot\hat{y}_0 - \frac{3}{2}J_2^0 G\hat{M}\cdot\hat{y}_0\rho^2(1-5\hat{z}_0^{\,2}) + j_{y0c} + j_{y0л},$$
$$\frac{dVz_0}{dt} = -G\hat{M}\cdot\hat{z}_0 - \frac{3}{2}J_2^0 G\hat{M}\cdot\hat{z}_0\rho^2(3-5\hat{z}_0^{\,2}) + j_{z0c} + j_{z0л},$$

Начальными условиями для интегрирования системы являются координаты центра масс НКА $x_0(t_b)$, $y_0(t_b)$, $z_0(t_b)$ и составляющие его вектора скорости $\dot{x}_0(t_b)$, $\dot{y}_0(t_b)$, $\dot{z}_0(t_b)$ в инерциальной геоцентрической системе координат $OX_0Y_0Z_0$ на момент шкалы МДВ. Эти начальные условия вычисляются путем пересчета передаваемых в навигационном

сообщении координат x(t_b), $y(t_b)$, $z(t_b)$ и составляющих вектора скорости $\dot{x}(t_b)$, $\dot{y}(t_b)$, $\dot{z}(t_b)$ центра масс НКА в связанной с Землей системе координат ПЗ-90. Пересчет осуществляется по следующим формулам:

$$x_0(t_b) = x(t_b)\cdot\cos(S(t_b)) - y(t_b)\cdot\sin(S(t_b)),$$

$$y_0(t_b) = x(t_b)\cdot\sin(S(t_b)) + y(t_b)\cdot\cos(S(t_b)),$$

$$z_0(t_b) = z(t_b),$$

$$\dot{x}_0(t_b) = \dot{x}(t_b)\cdot\cos(S(t_b)) - \dot{y}(t_b)\cdot\sin(S(t_b)) - \omega_з\cdot y_0(t_b),$$

$$\dot{y}_0(t_b) = \dot{x}(t_b)\cdot\sin(S(t_b)) + \dot{y}(t_b)\cdot\cos(S(t_b)) + \omega_з\cdot x_0(t_b),$$

$$\dot{z}_0(t_b) = \dot{z}(t_b),$$

$$S(t_b) = GST + \omega_з\cdot(t_b - 10800),$$

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

После интегрирования, полученные в инерциальной системе координат $OX_0Y_0Z_0$ координаты центра масс $x_0(t_i)$, $y_0(t_i)$, $z_0(t_i)$ и составляющие его вектора скорости $\dot{x}(t_i)$, $\dot{y}(t_i)$, $\dot{z}(t_i)$ могут быть пересчитаны в связанную с Землей систему ПЗ-90 Oxyz по формулам:

$$x(t_i) = x_0(t_i)\cdot\cos(S(t_i)) + y_0(t_i)\cdot\sin(S(t_i)),$$

$$y(t_i) = -x_0(t_i)\cdot\sin(S(t_i)) + y_0(t_i)\cdot\cos(S(t_i)),$$

$$z(t_i) = z_0(t_i),$$

$$\dot{x}(t_i) = \dot{x}_0(t_i)\cdot\cos(S(t_i)) + \dot{y}_0(t_i)\cdot\sin(S(t_i)) + \omega_з\cdot y(t_i),$$

$$\dot{y}(t_i) = -\dot{x}_0(t_i)\cdot\sin(S(t_i)) + \dot{y}_0(t_i)\cdot\cos(S(t_i)) - \omega_з\cdot x(t_i),$$

$$\dot{z}(t_i) = \dot{z}_0(t_i),$$

$$S(t_i) = GST + \omega_з\cdot(t_i - 10800).$$

В данном расчете используются следующие примечания:

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

$$\Delta x = (j_{x0л} + j_{x0c})\cdot\tau^2/2,\ \Delta y = (j_{y0л} + j_{y0c})\cdot\tau^2/2,\ \Delta z = (j_{z0л} + j_{z0c})\cdot\tau^2/2;$$

$$\Delta\dot{x} = (j_{x0л} + j_{x0c})\cdot\tau,\ \Delta\dot{y} = (j_{y0л} + j_{y0c})\cdot\tau,\ \Delta\dot{z} = (j_{z0л} + j_{z0c})\cdot\tau;$$

$$\tau = t_i - t_b.$$

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

2.2 Результаты моделирования положения спутника ГЛОНАСС

Алгоритм реализован на языке MATLAB, листинг программы приведен в приложении.

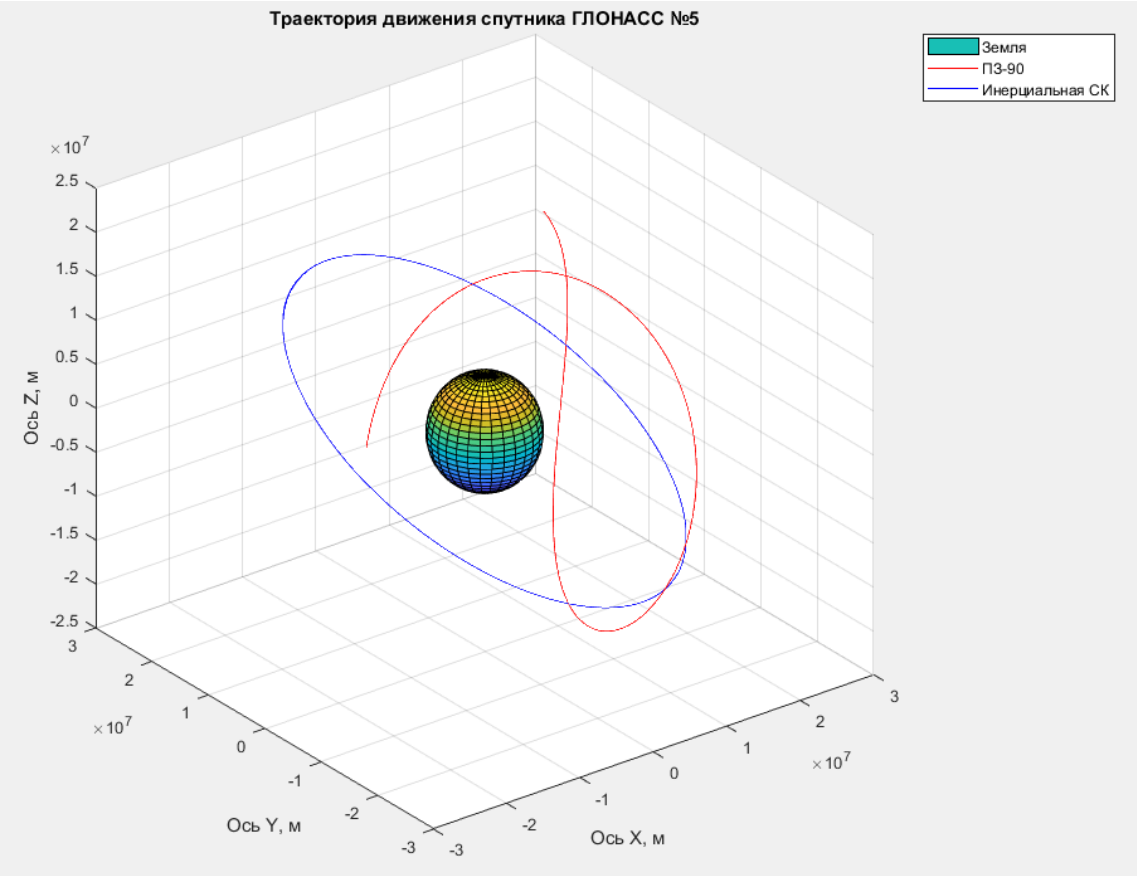


Рисунок 12 – Траектория движения спутника ГЛОНАСС №5 в системе координат ПЗ-90 (красная линия) и в инерциальной системе координат (синяя линия)

2.3 Построение SkyView

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

Для построения SkyView перейдем в локальную систему координат приемника WGS-84. Координаты приемника в WGS-84:

$X = 2846344{,}28450928\ м$

$Y = 2200154{,}79994168\ м$

$Z = 5249660{,}40233402\ м$

Следующим шагом, пересчитаем локальные декартовы координаты в сферические, тем самым получив азимут и угол места. По полученным углам построим графики в

полярной системе координат (рисунок 13) и график зависимости угла места от времени (рисунок 14).

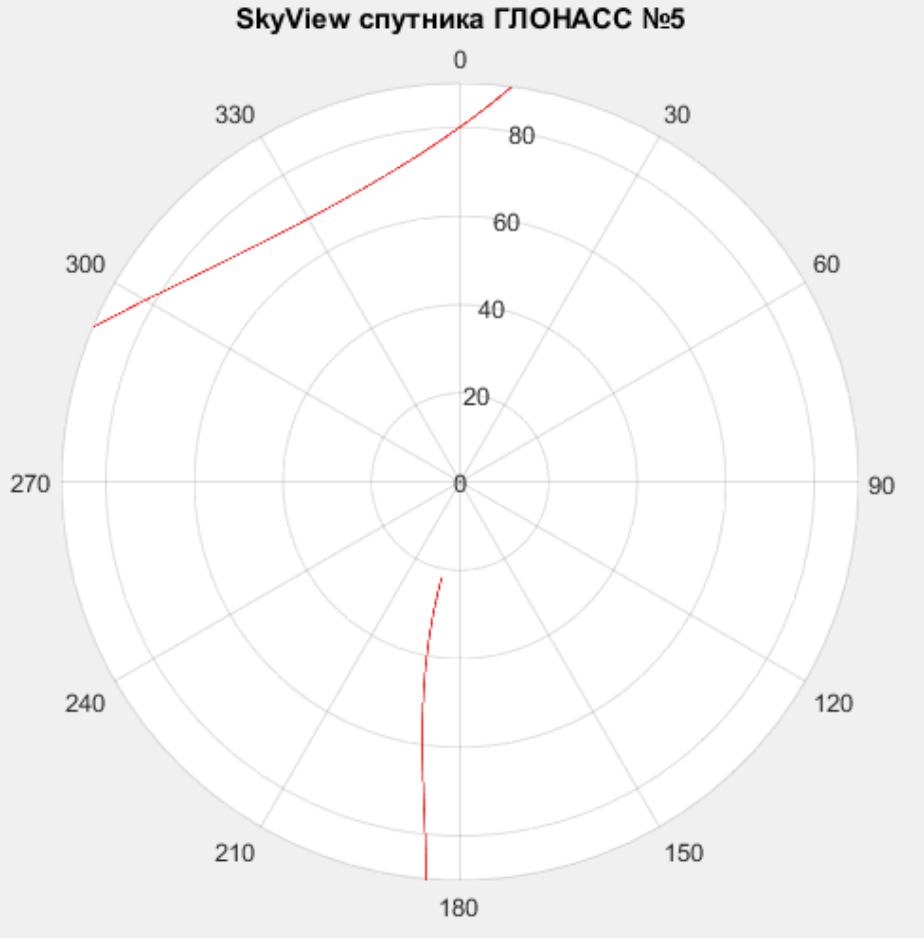


Рисунок 13 – SkyView спутника ГЛОНАСС №5

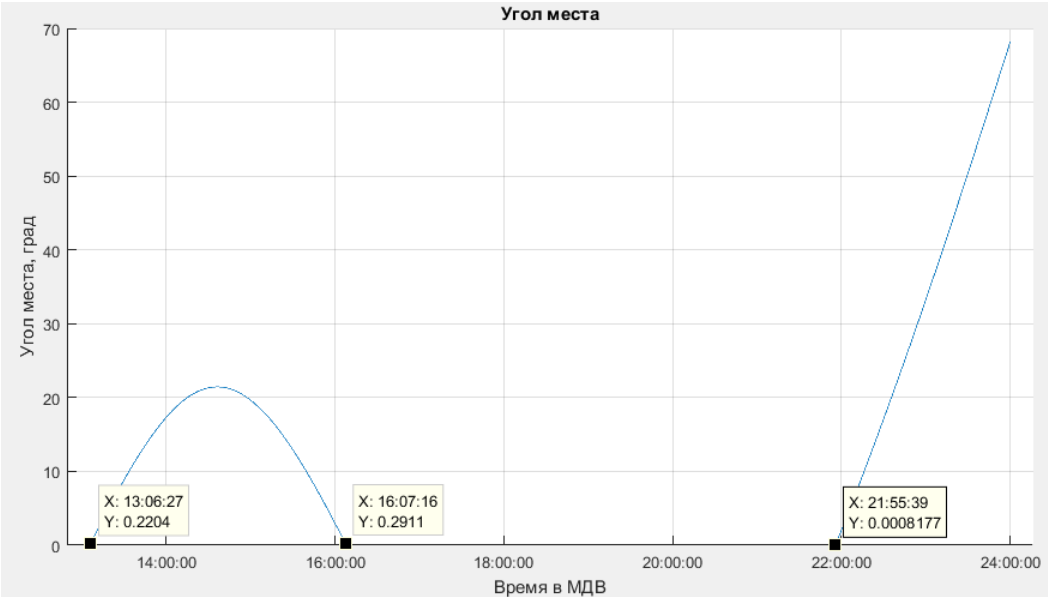


Рисунок 14 – График угла места спутника ГЛОНАСС №5

По SkyView и графику угла места видно, что спутник появлялся в зоне видимости приемника два раза, в первый раз с 13:06:27 по 16:07:16, и второй раз с 21:55:39 и до конца суток. Данные результаты совпадают данными сервиса Trimble GNSS Planning Online,

полученными на предыдущем этапе, с существенной погрешностью. Объясняется это тем, что выбранный алгоритм осуществляет точный расчет только на 30 интервале времени.

2.4 Заключение по результатам моделирования

На данном этапе была реализована на языке Matlab функция расчета положения спутника ГЛОНАСС №5 на временном интервале с 12:00 10.02.20 до 00:00 11.02.20 по шкале времени UTC. В качестве эфемерид использовались данные, полученные на предыдущем этапе. Использовались одни и те же эфемериды на весь рассматриваемый интервал.

В результате были получены графики траекторий движения спутника ГЛОНАСС №5 в системах координат: ПЗ-90 и инерциальной, и SkyView с графиком угла места для точки, в которой находился приемник.

Точный алгоритм дает существенную погрешность при расчете на интервале времени более чем ±15 минут. Таким образом, выбранный алгоритм можно применять при постоянном получении новый эфемерид. Для прогноза на большой интервал (превышающий 15 минут) времени лучше использовать долговременный алгоритм.

3 Реализация

Требуется разработать на языке C/C++ 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. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал.

Программный модуль должен сопровождаться unit-тестами:

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

Во время второго теста должно вычисляться и выводиться средняя длительность исполнения функции.

Требуется провести проверку на утечки памяти.

3.1 Особенности реализации

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

Для выполнения поставленных задач используются:

- Qt Creator – кроссплатформенный фреймворк для разработки программного обеспечения.
- MinGW – набор инструментов разработки программного обеспечения, включающий в себя компилятор и необходимые библиотеки.
- Boost Test – библиотека для C++, включающая в себя фреймворк для тестирования.
- Dr. Memory – инструмент, позволяющий выявлять утечки памяти.
- http://hilite.me/ – интернет-ресурс для стилизации кода.

Все указанные функции и тесты приведены в приложении.

3.2 Тестирование

На рисунке 14 представлен вывод unit-тестов Boost Test. Тестировались 3 функции: glnsvpos() – функция реализации метода Рунге-Кутты, add() – функция сложения, mult() – функция умножения. В функцию add() внесена ошибка, поэтому ожидается, что тест завершиться не успешно.

Тесты функции add() показали ошибку, а время выполнения составило около 1,7 мс. Тест функции mult завершился успешно за 18 мкс. Время тестирования функции glnsvpos() составило 3,62с для шага 0,1, и тест во всех случаях завершался успешно.

```
Running 3 test cases...
Entering test module "Test"
..\Boost_test\main.cpp(59): Entering test case "test_glnsvpos"
..\Boost_test\main.cpp(59): Leaving test case "test_glnsvpos"; testing time: 3618716us
..\Boost_test\main.cpp(129): Entering test case "test_add"
../Boost_test/main.cpp(135): error: in "test_add": check c == a+b has failed
..\Boost_test\main.cpp(129): Leaving test case "test_add"; testing time: 1740us
..\Boost_test\main.cpp(138): Entering test case "test_mult"
..\Boost_test\main.cpp(138): Leaving test case "test_mult"; testing time: 18us
Leaving test module "Test"; testing time: 3627850us

*** 1 failure is detected in the test module "Test"
```

Рисунок 15 – вывод unit-тестов

Для расчета положения спутника использовался тип данных с плавающей точкой double. Погрешность double с данными из matlab должна быть не более 10^{-14}. Для координат, размерность которых составляет 10^7 м, погрешность должна быть не более 10^{-7} м или 0,1 мкм. На рисунке 16 показан вывод максимальной разницы в координатах модели matlab и программы на C++.

```
Running 3 test cases...
Max delta X = 4.47035e-008
Max delta Y = 1.08033e-007
Max delta Z = -7.63685e-008
../Boost_test/main.cpp(150): error: in "test_add": check c == a+b has failed

*** 1 failure is detected in the test module "Test"
```

Рисунок 16 – вывод unit-тестов

При тестировании функции RK() – функции реализации метода Рунге-Кутты, в качестве входных данных выступали случайные начальные условия. Для 100 000 отсчетов, время выполнения составило 0,83 мс.

```
Entering test module "Test"
..\Boost_test\main.cpp(60): Entering test case "test_RK"
..\Boost_test\main.cpp(60): Leaving test case "test_RK"; testing time: 83872us
```

Рисунок 17 – вывод unit-тестов

3.3 Проверка памяти

Проверка памяти осуществлялась Dr. Memory с стандартными настройками.

Вывод Dr. Memory:

```
Dr. Memory version 2.3.0 build 1 built on Feb  6 2020 06:07:09
Windows version: WinVer=105;Rel=1903;Build=18362;Edition=Professional
Dr. Memory results for pid 10432: "libglnsvpos.exe"
Application cmdline: "D:\Repository\glnephexercise\build-libglnsvpos-
Desktop_Qt_5_14_2_MinGW_64_bit-Debug\debug\libglnsvpos.exe"
Recorded 118 suppression(s) from default C:\Program Files (x86)\Dr.
Memory\bin64\suppress-default.txt

Error #1: UNADDRESSABLE ACCESS beyond top of stack: reading
0x000000000063fb20-0x000000000063fb28 8 byte(s)
```

```
# 0 .text                                    [../../../../../src/gcc-
7.3.0/libgcc/config/i386/cygwin.S:152]
# 1 _pei386_runtime_relocator
[../libglnsvpos/src/rungekutta.cpp:107]
# 2 __tmainCRTStartup
# 3 .l_start
# 4 KERNEL32.dll!BaseThreadInitThunk
Note: @0:00:00.177 in thread 10684
Note: 0x000000000063fb20 refers to 632 byte(s) beyond the top of the stack
0x000000000063fd98
Note: instruction: or     $0x0000000000000000 (%rcx) -> (%rcx)

Error #2: UNADDRESSABLE ACCESS beyond top of stack: reading
0x000000000063f9d0-0x000000000063f9d8 8 byte(s)
# 0 .text                                  [../../../../../src/gcc-
7.3.0/libgcc/config/i386/cygwin.S:152]
# 1 __pformat_int.isra.0                   [../../../../../src/gcc-
7.3.0/libgcc/config/i386/cygwin.S:158]
# 2 __mingw_pformat                        [../../../../../src/gcc-
7.3.0/libgcc/config/i386/cygwin.S:158]
# 3 __mingw_vfprintf                       [../../../../../src/gcc-
7.3.0/libgcc/config/i386/cygwin.S:158]
# 4 printf                                 [C:/IDE/Qt/Tools/mingw730_64/x86_64-
w64-mingw32/include/stdio.h:349]
# 5 write_struct_Y                         [../libglnsvpos/src/func.cpp:81]
# 6 glnsvpos                               [../libglnsvpos/src/glnsvpos.cpp:127]
# 7 main                                   [../libglnsvpos/main.cpp:14]
Note: @0:00:02.638 in thread 10684
Note: 0x000000000063f9d0 refers to 24 byte(s) beyond the top of the stack
0x000000000063f9e8
Note: instruction: or     $0x0000000000000000 (%rcx) -> (%rcx)

Error #3: POSSIBLE LEAK 123 direct bytes 0x00000000030a01c0-
0x00000000030a023b + 0 indirect bytes
# 0 replace_malloc
[d:\drmemory_package\common\alloc_replace.c:2577]
# 1 msvcrt.dll!malloc_crt
# 2 msvcrt.dll!_setargv
# 3 msvcrt.dll!_getmainargs
# 4 pre_cpp_init
# 5 msvcrt.dll!initterm
# 6 __tmainCRTStartup
# 7 .l_start
# 8 KERNEL32.dll!BaseThreadInitThunk

===============================================================================
FINAL SUMMARY:

DUPLICATE ERROR COUNTS:
        Error #   1:      2
        Error #   2:      2

SUPPRESSIONS USED:

ERRORS FOUND:
      2 unique,     4 total unaddressable access(es)
      0 unique,     0 total uninitialized access(es)
      0 unique,     0 total invalid heap argument(s)
      0 unique,     0 total GDI usage error(s)
```

```
      0 unique,     0 total handle leak(s)
      0 unique,     0 total warning(s)
      0 unique,     0 total,      0 byte(s) of leak(s)
      1 unique,     1 total,    123 byte(s) of possible leak(s)
ERRORS IGNORED:
      4 potential error(s) (suspected false positives)
         (details: C:\Users\Zherebin\AppData\Roaming\Dr. Memory\DrMemory-
libglnsvpos.exe.10432.000\potential_errors.txt)
      8 unique,     8 total,    978 byte(s) of still-reachable allocation(s)
         (re-run with "-show_reachable" for details)
```

Dr. Memory обнаружила ошибки UNADDRESSABLE ACCESS в количестве 2 штук при вызове функции:

```
printf("Error. File: %s, Line: %d\n", __FILE__, __LINE__);
```

Возможно, это связанно с внутренними библиотеками компилятора, так как все указывает на них.

Так же, Dr. Memory обнаружила возможные утечки памяти на 123 байта в своих же библиотеках.

Утечек памяти в программе обнаружено не было.

3.4 Заключение по результатам реализации

На данном этапе была реализована на языке C/C++ функция расчета положения спутника ГЛОНАСС №5 на заданное время по шкале UTC. Функция сопровождается unit-тестами и проверкой на утечки памяти.

Погрешность вычисления координат спутника функции C/C++ и модели maltab не превышает 0,1 мкм, при использовании типа данных с плавающей точкой, двойной точности double.

Время выполнения расчета функции, при шаге 0,1 с, составляет 3,62с. Выполнение такого же расчета в maltab – более 5 минут. Таким образом функция минимизирует время расчета, относительно модели.

Для минимизации количества затрачиваемой оперативной памяти переменные, которые содержат только положительные значения, использовали беззнаковые (unsigned) типы данных с учетом их максимальной размерности и разрядности. Тип данных double занимает в памяти 8 байт. Для массива координат, скоростей и ускорений, при шаге расчета 0,1 с для временного интервала с 12:00 10.02.20 до 00:00 11.02.20 (432 000 отсчетов), необходимо 20 736 000 байт или 19,78 Мбайт памяти.

4 Заключение

В рамках данного проекта ознакомились с рядом инструментов и техник, используемых при разработке аппаратуры потребителей спутниковых радионавигационных систем. Научились использовать интерфейсные контрольные документы. Получен опыт извлечения эфемерид спутников из навигационного сообщения, моделирования алгоритма расчета положения спутника с использованием численного интегрирования системы дифференциальных уравнений, реализации алгоритма на языке C/C++, тестирования unit-тестами, проверки на утеки памяти.

С помощью интернет-ресурса https://github.com/ овладели навыком пользования системы контроля версий Git.

При реализации программного обеспечения дословно перенести модель в прошивку приемника, как правило, не получается. Определили погрешности вычисления между функцией на C/C++ и моделью. Библиотеки линейной алгебры для программ, как правило, недоступны, поэтому необходима реализация численного интегрирования базовыми функциями.

В результате выполнения проекта были получены библиотечные функции на C/C++, позволяющее рассчитывать положения спутника ГЛОНАСС по эфемеридам на заданный интервал времени.

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».

6 Приложения

6.1 Листинг кода этапа моделирования

Файл GLONASS_Satellite_Coordinates_Computation.m

```
clear all; close all; tic; clc;
format long;

%% Заданные параметры
% Эфемериды на заданную  эпоху:
% 2020/02/25 13:45:18
Time_year = 2020;
Time_month = 2;
Time_day = 25;
Time_hour = 13;
Time_minutes = 45;
Time_seconds = 18;

% Координаты на Te в системе ПЗ-90, [м]:
X = -8444572.27;
Y = -8664957.52;
Z = 22466454.10;

% Компоненты вектора скорости на Te в системе ПЗ-90, [м/с]:
VX = 2983.60348;
VY = -743.76965;
VZ =  832.83615;

% Ускорения лунно-солнечные на Te в системе ПЗ-90, [м/с2]:
AX = -0.0000028;
AY =  0.0000019;
AZ = -0.0000019;

% SV временное смещение, [нс]:
Tau = -44762.2;
% SV относительное смещение частоты, [нс/с]:
Gamma = 0.0009;

%% Расчет времени формата ГЛОНАСС
N4 = floor((Time_year-1996)/4) + 1; % Номер четырехлетнего интервала
NT = 365*(Time_year-1996-4*(N4-1)) + 31 + Time_day + 1; % Номер суток в
четырехлетнем интервале
tb = Time_seconds + Time_minutes*60 + Time_hour*60*60 + 10800; % Текущее
вермя в МДВ [с]

% Расчет среднего звездного времени по Гринвичу
GMST = GMST_calc( N4,NT );

%% Пересчет координат и оставляющих вектора скорости центра масс НКА в
связанной с Землей систему координат ПЗ-90
% средняя угловая скорость вращения Земли относительно точки весеннего
равноденствия, [рад/с]:
Omega_E = 7.2921151467e-5;

Theta_Ge = GMST + Omega_E * (tb - 3 * 60 * 60);

% Координаты:
X0 = X * cos(Theta_Ge) - Y * sin(Theta_Ge);
Y0 = X * sin(Theta_Ge) + Y * cos(Theta_Ge);
Z0 = Z;
```

```
% Скорости:
VX0 = VX * cos(Theta_Ge) - VY * sin(Theta_Ge) - Omega_E * Y0;
VY0 = VX * sin(Theta_Ge) + VY * cos(Theta_Ge) + Omega_E * X0;
VZ0 = VZ;

% Ускорения:
JX0ms = AX * cos(Theta_Ge) - AY * sin(Theta_Ge);
JY0ms = AX * sin(Theta_Ge) + AY * cos(Theta_Ge);
JZ0ms = AZ;

%% Интегрирование численным методом
Toe = (12+3)*60*60;
Tof = (24+3)*60*60;
h = 1;

ti = Toe:h:Tof;

F0 = [X0 Y0 Z0 VX0 VY0 VZ0]; % Начальные условия

% [t, F] = ode45('diffs', tb:-Ts:ti(1), F0); % Метод Рунге-Кутта 4-го порядка
[t, F] = RungeKutta4( tb,-h,ti(1),F0 );
Fout = F(end:-1:2,:);
tout = t(end:-1:2,:);
% [t, F] = ode45('diffs', tb:Ts:ti(end), F0); % Метод Рунге-Кутта 4-го
порядка
[t, F] = RungeKutta4( tb,h,ti(end),F0 );
Fout = [Fout; F];
tout = [tout; t];

th = hours(tout/60/60-3); % Перевод временной оси в формат hh:mm:ss

%% Учет ускорений
tau = tout - tb;
deltaX = JX0ms*(tau.^2)/2;
deltaY = JY0ms*(tau.^2)/2;
deltaZ = JZ0ms*(tau.^2)/2;

deltaVX = JX0ms*tau;
deltaVY = JY0ms*tau;
deltaVZ = JZ0ms*tau;

delta = [deltaX deltaY deltaZ deltaVX deltaVY deltaVZ];

Fout = Fout + delta;

%% Чтение и запись файликов
% Чтение из файла
FoutC = load('../source/data_out.txt');
% Fout = load('../source/Matlab_data_for_h01.txt');

% Запись в файл
data_out = fopen('../source/Matlab_test_data_out.txt', 'w+');     % открытие
файла на запись
if data_out == -1                       % проверка корректности открытия
    error('File is not opened');
end

F_out = [Fout(:,1), Fout(:,2), Fout(:,3)];
fprintf(data_out, '%.15f\n', F_out);   % запись в файл
fclose(data_out);                 % закрытие файла
```

```
%% Расчет разницы с Си
DeltaF1 = Fout - FoutC;
D_DeltaF1 = sqrt( DeltaF1(:,1).^2 + DeltaF1(:,2).^2 + DeltaF1(:,3).^2 );

figure
hold on
grid on
plot(th, D_DeltaF1, 'DurationTickFormat','hh:mm:ss')
%% Пересчет координат центра масс НКА в систему координат ПЗ-90
Theta_Ge = GMST + Omega_E * (tout - 3 * 60 * 60);

crd_PZ90(:,1) =  Fout(:,1).*cos(Theta_Ge) + Fout(:,2).*sin(Theta_Ge);
crd_PZ90(:,2) = -Fout(:,1).*sin(Theta_Ge) + Fout(:,2).*cos(Theta_Ge);
crd_PZ90(:,3) =  Fout(:,3);

%% Пересчет координат центра масс НКА в систему координат WGS-84
ppb = 1e-9;
mas = 1e-3/206264.8; % [рад]

MATRIX_WGS_84 = [-3*ppb -353*mas -4*mas;
                 353*mas -3*ppb 19*mas;
                 4*mas -19*mas -3*ppb];

crd_WGS_84 = crd_PZ90.'; % Переход к вектору-столбцу

for i = 1:length(crd_WGS_84(1,:))
    crd_WGS_84(:,i) =  crd_WGS_84(:,i) + MATRIX_WGS_84 * crd_WGS_84(:,i) +
[0.07; -0; -0.77];
end

crd_WGS_84 = crd_WGS_84.'; % Переход к вектору-строки

%% Географические координаты корпуса Е и их перевод в систему WGS-84
N_gr = 55;
N_min = 45;
N_sec = 24.1438;
N = N_gr*pi/180 + N_min/3437.747 + N_sec/206264.8; % широта [рад]

E_gr = 37;
E_min = 42;
E_sec = 11.3386;
E = E_gr*pi/180 + E_min/3437.747 + E_sec/206264.8; % долгота [рад]

H = 500; % высота [м]

llh = [N E H];
crd_PRM = llh2xyz(llh)';

 %% Постороение SkyPlot
for i = 1:length(crd_WGS_84(:,1))

    [X(i) Y(i) Z(i)] =
ecef2enu(crd_WGS_84(i,1),crd_WGS_84(i,2),crd_WGS_84(i,3),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));
        %teta(i) = atan2(sqrt(X(i)^2 + Y(i)^2),Z(i));
        %phi(i) = atan2(Y(i),X(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
R_Earth = 6371e3;
[Xz,Yz,Zz] = sphere(30);

% Инерциальная СК и ПЗ-90
figure
surf(Xz*R_Earth,Yz*R_Earth,Zz*R_Earth)
hold on
grid on
plot3(crd_PZ90(:,1), crd_PZ90(:,2), crd_PZ90(:,3), 'r')
plot3(Fout(:,1), Fout(:,2), Fout(:,3), 'b')
title('Траектория движения спутника ГЛОНАСС №5')
xlabel('Ось X, м')
ylabel('Ось Y, м')
zlabel('Ось Z, м')
hold off
legend('Земля','ПЗ-90', 'Инерциальная СК');

figure
surf(Xz*R_Earth,Yz*R_Earth,Zz*R_Earth)
hold on
grid on
plot3(Fout(:,1), Fout(:,2), Fout(:,3), 'b')
title('Траектория движения спутника ГЛОНАСС №5')
xlabel('Ось X, м')
ylabel('Ось Y, м')
zlabel('Ось Z, м')
hold off

% СК ПЗ-90
figure
surf(Xz*R_Earth,Yz*R_Earth,Zz*R_Earth)
hold on
grid on
plot3(crd_PZ90(:,1), crd_PZ90(:,2), crd_PZ90(:,3), 'b')
title({'Траектория движения КА №5 ГЛОНАСС,' ; 'в системе координат ПЗ-90'})
xlabel('Ось X, м')
ylabel('Ось Y, м')
zlabel('Ось Z, м')
hold off

% СК WGS-84
figure
surf(Xz*R_Earth,Yz*R_Earth,Zz*R_Earth)
grid on
hold on
plot3(crd_WGS_84(:,1),crd_WGS_84(:,2),crd_WGS_84(:,3), 'b')
title({'Траектория движения КА №5 ГЛОНАСС,' ; 'в системе координат WGS-84'})
xlabel('Ось X, м')
ylabel('Ось Y, м')
zlabel('Ось Z, м')
```

```
hold off

% Скайплот
figure
ax = polaraxes;
polarplot(ax,phi,teta*180/pi)
ax.ThetaDir = 'clockwise';
ax.ThetaZeroLocation = 'top';
title('SkyView спутника ГЛОНАСС №5')

% Угол места
figure
grid on
hold on
plot(th,(-teta)*180/pi+90,'DurationTickFormat','hh:mm:ss')
title('Угол места')
xlabel('Время в МДВ')
ylabel('Угол места, град')
```

Файл GMST_calc.m

```
function GMST = GMST_calc( N4,N_T )

% Текущая Юлианская дата на 0 часов шкалы МДВ
JD0 = 1461 * (N4 - 1) + N_T + 2450082.5 - (N_T - 3) / 25;
% Время от эпохи 2000 г 1 января 12 ч (UTC(SU))
T_delta = (JD0 - 2451545) / 36525;
% Угол поворота Земли [рад]
ERA = 2 * pi * ( 0.7790572732640 + 1.00273781191135448 * (JD0 - 2451545));
% Среднее звездное время по Гринвичу [рад]
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;

end
```

Файл RungeKutta4.m

```
function [t, Y] = RungeKutta4( tn, h, Tlim, Y0 )

t = tn:h:Tlim;
t = t.';

Y(1,:) = Y0;

for k = 2:length(t)

    K1 = diffs(tn, Y(k-1,:));

    Y2 = Y(k-1,:) + h*K1.'/2;
    K2 = diffs(tn + h/2, Y2);

    Y3 = Y(k-1,:) + h*K2.'/2;
    K3 = diffs(tn + h/2, Y3);

    Y4 = Y(k-1,:) + h*K3.';
    K4 = diffs(tn + h, Y4);

    Knextstep = h/6 * (K1 + 2*K2 + 2*K3 + K4);
    Y(k,:) = Y(k-1,:) + Knextstep.';
```

```
end

end
```

Файл diffs.m

```
function dF = diffs( t, F )
%% Расчет переменных
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/с2]
a_e = 6378136; % большая полуось общеземного эллипсоида, [м]

crdX = F(1);
crdY = F(2);
crdZ = F(3);

r = sqrt(crdX^2 + crdY^2 + crdZ^2);

GM0 = GM / (r^2);
Rho = a_e / r;
crdX0 = crdX / r;
crdY0 = crdY / r;
crdZ0 = crdZ / r;

%% Дифуры
dF = F(:);
dF(1) = F(4);
dF(2) = F(5);
dF(3) = F(6);

dF(4) = - GM0 * crdX0 - 3/2 * J02 * GM0 * crdX0 * Rho^2 * (1 - 5 * crdZ0^2);
dF(5) = - GM0 * crdY0 - 3/2 * J02 * GM0 * crdY0 * Rho^2 * (1 - 5 * crdZ0^2);
dF(6) = - GM0 * crdZ0 - 3/2 * J02 * GM0 * crdZ0 * Rho^2 * (3 - 5 * crdZ0^2);
end
```

6.2 Листинг кода этапа реализации

Файл main.cpp

```
#include <iostream>

#include <include/libglnsvpos/func.h>
#include "include/libglnsvpos/glnsvpos.h"
#include "include/libglnsvpos/rungekutta.h"
#include "include/libglnsvpos/structures.h"

using namespace std;

int main() {

    cout << "Calculation started" << endl;

    if(glnsvpos(0, 0.1)) { // (RK_valid h): RK_valid - allows calculation
RungeKutta; h - time step [s]
        cout << "Calculation finished successful" << endl;
    } else {
        cout << "Calculation finished failed " << endl;
    }
```

```
    cout << "Press any key to continue..." << endl;
}
```

Файл glnsvpos.h

```
#ifndef GLNSVPOS_H
#define GLNSVPOS_H

#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

uint64_t glnsvpos(bool RK_valid, double h);

#endif /* #ifndef GLNSVPOS_H */
```

Файл glnsvpos.cpp

```
#include <func.h>
#include <glnsvpos.h>
#include <rungekutta.h>
#include <structures.h>

using namespace std;

uint64_t glnsvpos(bool RK_valid, double h) {

    // Class Ephemeris
    struct Ephemeris_s Eph;

    // Coordinates
    Eph.X = -8444572.27;
    Eph.Y = -8664957.52;
    Eph.Z = 22466454.10;
    // Velocity
    Eph.VX = 2983.60348;
    Eph.VY = -743.76965;
    Eph.VZ =  832.83615;
    // Acceleration
    Eph.AX = -0.0000028;
    Eph.AY =  0.0000019;
    Eph.AZ = -0.0000019;

    uint16_t Time_year = 2020;
    uint8_t Time_month = 2;
    uint8_t Time_day = 25;
    uint8_t Time_hour = 13;
    uint8_t Time_minutes = 45;
    uint8_t Time_seconds = 18;

    uint16_t year_idx = 0;

    // Time in Gln
    Eph.N4 = ((Time_year - 1996) / 4) + 1;
    while (Eph.N4 > 31) { // Учет 5-битности N4
        Eph.N4 -= 31;
        year_idx++;
    }

    Eph.NT = NT_calc( Eph.N4, Time_year, year_idx, Time_month, Time_day );
```

```
    Eph.tb = Time_seconds + Time_minutes*60 + Time_hour*60*60 + 10800;
    if (Eph.tb >= 24*60*60) {
        Eph.tb -= 24*60*60;
        Eph.NT++;
        if (Eph.NT >= 1462) Eph.N4++;
    }

    double GMST = GMST_calc( Eph.N4, Eph.NT );

    struct Ephemeris_s Eph0 = CrdTrnsf2Inertial( Eph, GMST );

    uint32_t tn = Eph.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) / (double)h; // Количесвио отсчетов для
времени большего текущего Eph.tb
    uint64_t N2dec = (tn - Toe) / (double)h; // Количесвио отсчетов для
времени меньшего текущего Eph.tb
    uint64_t N = N2inc + N2dec; // Общее число отсчетов

    // Численное интегрирования для времени меньшего текущего Eph.tb
    struct Y_s *Ydec;
    Ydec = new struct Y_s[N2dec];
    Ydec[0].X = Eph0.X;
    Ydec[0].Y = Eph0.Y;
    Ydec[0].Z = Eph0.Z;
    Ydec[0].VX = Eph0.VX;
    Ydec[0].VY = Eph0.VY;
    Ydec[0].VZ = Eph0.VZ;

    if (RK_valid) RK( N2dec, -h, Ydec);

    // Численное интегрирования для времени большего текущего Eph.tb
    struct Y_s *Yinc;
    Yinc = new struct Y_s[N2inc];
    Yinc[0].X = Eph0.X;
    Yinc[0].Y = Eph0.Y;
    Yinc[0].Z = Eph0.Z;
    Yinc[0].VX = Eph0.VX;
    Yinc[0].VY = Eph0.VY;
    Yinc[0].VZ = Eph0.VZ;

    if (RK_valid) RK( N2inc, h, Yinc);

    // Формирование выходного массива
    struct Y_s *Yout;
    Yout = new struct Y_s[N];

    uint32_t i;

    // ----------------------------------------
    // M:  1   2   3   4   5    - adress
    //    x1  x2  x3  x4  x5    - value
    // C:  0   1   2   3   4    - adress
    //    x1  x2  x3  x4  x5    - value
    // if MATLAB array size N, then in C++ N-1
    // ----------------------------------------

    for (i = 0; i < N2dec; i++) {
        Yout[i] = Ydec[N2dec-i-1];
        //cout << " i = " << i << " N2dec-i = " << N2dec-i << endl;
    }
```

```
    for (i = 0; i < N2inc; i++) {
        Yout[i+N2dec] = Yinc[i];
        //cout << " i = " << i << " i+N2dec = " << i+N2dec << endl;
    }

    // Учет ускорений
    double tau = (double)Toe - (double)tn;
    for (i = 0; i < N; i++) {

        //cout << "tau = " << tau << endl;

        Yout[i].X += Eph0.AX * (tau * tau) / (double)2;
        Yout[i].Y += Eph0.AY * (tau * tau) / (double)2;
        Yout[i].Z += Eph0.AZ * (tau * tau) / (double)2;

        Yout[i].VX += Eph0.AX * tau;
        Yout[i].VY += Eph0.AY * tau;
        Yout[i].VZ += Eph0.AZ * tau;

        tau += (double)h;
    }

    write_struct_Y(Yout, N, "data_out.txt");

    // Очищение памяти
//    delete []Yout;
//    delete []Ydec;
//    delete []Yinc;

    return N;
}
```

Файл rungekutta.h

```
#ifndef RUNGEKUTTA_H
#define RUNGEKUTTA_H

#include <math.h>
#include <iostream>
#include <stdio.h>

#include "structures.h"
#include "func.h"

Y_s* diffs(double tn , struct Y_s Y);

int RK(uint32_t N, double h, struct Y_s* Y) ;

int mult(int a, int b);

#endif /* #ifndef RUNGEKUTTA_H */
```

Файл rungekutta.cpp

```
#include <rungekutta.h>

using namespace std;

Y_s* diffs(double tn , struct Y_s Y)
{
```

```
    double J02 = 1082625.75e-9; // зональный гармонический коэффициент второй
степени, характеризующий полярное сжатие Земли
    double GM = 398600441.8e6; // геоцентрическая константа гравитационного
поля Земли с учетом атмосферы, [м3/с2]
    double a_e = 6378136; // большая полуось общеземного эллипсоида, [м]

    double crdX = Y.X;
    double crdY = Y.Y;
    double 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;
    double crdY0 = crdY / r;
    double crdZ0 = crdZ / r;

    struct Y_s* dY;
    dY = new struct Y_s;
    // Дифуры
    dY->X = Y.VX;
    dY->Y = Y.VY;
    dY->Z = Y.VZ;

    dY->VX = - GM0 * crdX0 - (double)1.5 * J02 * GM0 * crdX0 * Rho * Rho * (1
- (double)5 * crdZ0 * crdZ0);
    dY->VY = - GM0 * crdY0 - (double)1.5 * J02 * GM0 * crdY0 * Rho * Rho * (1
- (double)5 * crdZ0 * crdZ0);
    dY->VZ = - GM0 * crdZ0 - (double)1.5 * J02 * GM0 * crdZ0 * Rho * Rho * (3
- (double)5 * crdZ0 * crdZ0);

    return dY;
}

int RK(uint32_t N, double h, struct Y_s* Y) {

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

    struct Y_s *k1, *k2, *k3, *k4, *knextstep;
    struct Y_s Y2, Y3, Y4;

    for (uint32_t k = 1; k < N; k++)
    {
        k1 = new struct Y_s;
        k2 = new struct Y_s;
        k3 = new struct Y_s;
        k4 = new struct Y_s;
        knextstep = new struct Y_s;

        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);

            knextstep->X = h / (double)6 * ( k1->X + 2 * k2->X + 2 * k3->X + k4-
>X );
            knextstep->Y = h / (double)6 * ( k1->Y + 2 * k2->Y + 2 * k3->Y + k4-
>Y );
            knextstep->Z = h / (double)6 * ( k1->Z + 2 * k2->Z + 2 * k3->Z + k4-
>Z );
            knextstep->VX = h / (double)6 * ( k1->VX + 2 * k2->VX + 2 * k3->VX +
k4->VX );
            knextstep->VY = h / (double)6 * ( k1->VY + 2 * k2->VY + 2 * k3->VY +
k4->VY );
            knextstep->VZ = h / (double)6 * ( k1->VZ + 2 * k2->VZ + 2 * 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;
}

int mult(int a, int b){
    return a*b;
}
```

Файл func.h

```
#ifndef FUNC_H
#define FUNC_H

#include <math.h>
#include <iostream>

#include "structures.h"

uint16_t NT_calc(uint8_t N4, uint16_t T_year, uint16_t year_idx, uint16_t
T_month, uint16_t T_day);

double GMST_calc(uint8_t N4, uint16_t NT);
```

```
Ephemeris_s CrdTrnsf2Inertial(struct Ephemeris_s Eph, double GMST);

void write_struct_Y(struct Y_s *Y_data, uint64_t Size, char *fname);

void read_struct_Y(struct Y_s *Y_data, uint64_t Size, char *fname);

int add(int a, int b);

#endif // FUNC_H
```

Файл func.cpp

```
#include <func.h>
#include <glnsvpos.h>
#include <rungekutta.h>
#include <structures.h>

using namespace std;

uint16_t NT_calc(uint8_t N4, uint16_t T_year, uint16_t year_idx, uint16_t
T_month, uint16_t T_day) {
    uint16_t NT;
    uint16_t N42;
    uint16_t Month[12] = {31, 30, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    if (T_month < 1 || T_month > 12)
        return 0;

    N42 = (year_idx*31) + N4;
    NT = T_year - 1996 - 4 * (N42 - 1);
    NT *= 365;

    for (uint16_t i = 1; i < T_month; i++)
        NT += Month[i-1];

    return NT + T_day + 1;
}

double GMST_calc(uint8_t N4, uint16_t NT) {
    // Текущая Юлианская дата на 0 часов шкалы МДВ
    double JD0 = 1461 * (N4 - 1) + NT + 2450082.5 - ((NT - 3) / (double)25);
    // Время от эпохи 2000 г 1 января 12 ч (UTC(SU))
    double T_delta = (JD0 - 2451545) / (double)36525;
    // 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 * T_delta;
    GMST += 0.0000067465784654 * T_delta * T_delta;
    GMST -= 0.0000000000021332 * T_delta * T_delta * T_delta;
    GMST -= 0.0000000001452308 * T_delta * T_delta * T_delta * T_delta;
    GMST -= 0.0000000000001784 * T_delta * T_delta * T_delta * T_delta *
T_delta;

    return GMST;
}

Ephemeris_s CrdTrnsf2Inertial(struct Ephemeris_s Eph, double GMST) {

    struct Ephemeris_s Eph0;
```

```
    double Omega_E = 7.2921151467e-5;

    double Theta_Ge = GMST + Omega_E * (Eph.tb - 3 * 60 * 60);

    // Координаты:
    Eph0.X = Eph.X * cos(Theta_Ge) - Eph.Y * sin(Theta_Ge);
    Eph0.Y = Eph.X * sin(Theta_Ge) + Eph.Y * cos(Theta_Ge);
    Eph0.Z = Eph.Z;

    // Скорости:
    Eph0.VX = Eph.VX * cos(Theta_Ge) - Eph.VY * sin(Theta_Ge) - Omega_E *
Eph0.Y;
    Eph0.VY = Eph.VX * sin(Theta_Ge) + Eph.VY * cos(Theta_Ge) + Omega_E *
Eph0.X;
    Eph0.VZ = Eph.VZ;

    // Ускорения:
    Eph0.AX = Eph.AX * cos(Theta_Ge) - Eph.AY * sin(Theta_Ge);
    Eph0.AY = Eph.AX * sin(Theta_Ge) + Eph.AY * cos(Theta_Ge);
    Eph0.AZ = Eph.AZ;

    // Время
    Eph0.N4 = Eph.N4;
    Eph0.NT = Eph.NT;
    Eph0.tb = Eph.tb;

    return Eph0;
}

void write_struct_Y(struct Y_s *Y_data, uint64_t Size, char *fname) {

    // Запись в файл (для матлаба)
    FILE *file;
    if ((file = fopen(fname,"wb")) == NULL) {
        printf("Error. File: %s, Line: %d\n", __FILE__, __LINE__);
    }
    else {
        for(uint32_t i = 0; i <= Size; i++) {
            fprintf(file,"%.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(file);
}

void read_struct_Y(struct Y_s *Y_data, uint64_t Size, char *fname) {

    // Чтение из файла (из матлаба)

    ifstream file("Matlab_data_for_h1.txt");
    if (file.is_open()) { //Если открытие файла прошло успешно

        string line; //Строчка текста

        uint32_t i;

        for (i = 0; i <= Size; i++) {
            getline(file, line);
            istringstream iss(line);
            iss >> Y_data[i].X;
        }
        for (i = 0; i <= Size; i++) {
            getline(file, line);
```

```
            istringstream iss(line);
            iss >> Y_data[i].Y;
        }
        for (i = 0; i <= Size; i++) {
            getline(file, line);
            istringstream iss(line);
            iss >> Y_data[i].Z;
        }
    }
    else printf("Error. File: %s, Line: %d\n", __FILE__, __LINE__);
}

int add(int a, int b) {
    return mult(a,b) + b;
}
```

Файл structures.h

```
#ifndef STRUCTURES_H
#define STRUCTURES_H

#include <iostream>

struct Ephemeris_s {
    // Time in Gln
    uint16_t   N4;
    uint16_t  NT;
    uint32_t  tb;
    // Coordinates
    double X, Y, Z;
    // Velocity
    double VX, VY,VZ;
    // Acceleration
    double AX, AY, AZ;
};

struct Y_s {
    double X, Y, Z, VX, VY, VZ;
};

#endif // STRUCTURES_H
```