**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 3](#_Toc41685883)

[1 Использование сторонних средств 3](#_Toc41685884)

[1.1 Описание процесса использования RTKLIB 3](#_Toc41685885)

[1.2 Получение графика угла места и SkyView с помощью Trimble GNSS Planning 8](#_Toc41685886)

[2 Моделирование 11](#_Toc41685887)

[2.1 Алгоритм расчета положения спутника ГЛОНАСС 11](#_Toc41685888)

[2.2 Результаты моделирования положения спутника ГЛОНАСС 14](#_Toc41685889)

[2.3 Построение SkyView 14](#_Toc41685890)

[2.4 Заключение по результатам моделирования 16](#_Toc41685891)

[3 Реализация 17](#_Toc41685892)

[3.1 Особенности реализации 17](#_Toc41685893)

[3.2 Тестирование 17](#_Toc41685894)

[3.3 Проверка памяти 18](#_Toc41685895)

[3.4 Заключение по результатам реализации 20](#_Toc41685896)

[4 Заключение 21](#_Toc41685897)

[5 Литература 21](#_Toc41685898)

[6 Приложения 22](#_Toc41685899)

[6.1 Листинг кода этапа моделирования 22](#_Toc41685900)

[6.2 Листинг кода этапа реализации 27](#_Toc41685901)

**Введение**

Название проекта: Разработка модуля расчёта координат спутника ГЛОНАСС.

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

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

Для достижения цели выполняется ряд задач:

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

Требования:

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

Курсовой проект разбит на три этапа, отличающиеся осваиваемыми инструментами.

**1 Использование сторонних средств**

**1.1 Описание процесса использования RTKLIB**

На крыше корпуса Е МЭИ установлена трехдиапазонная антенна Harxon HX-CSX601A. Она через 50-метровый кабель, сплиттер, bias-tee и усилитель подключена к трем навигационным приемникам:

* Javad Lexon LGDD,
* SwiftNavigation Piksi Multi,
* Clonicus разработки ЛНС МЭИ.

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

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

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



Рисунок 1 – Окно программы RTKLIB v.2.4.2

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

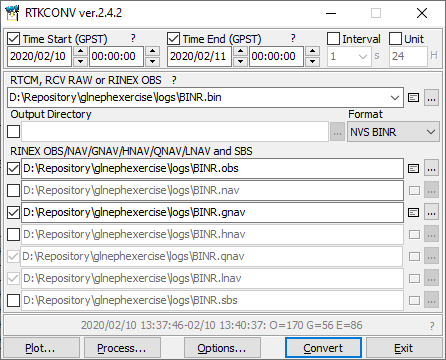


Рисунок 2 – Окно программы RTKCONV ver.2.4.2

В открывшемся окне выбираем Time Start (GPST), Time End (GPST), и ставим время интервала наблюдений с 00:00 10.02.20 до 00:00 11.02.20. В меню «Options» (Рисунок 3) выбираем спутниковую систему ГЛОНАСС и указываем в поле «Excluded Satellite» следующее: R3, R4, 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.

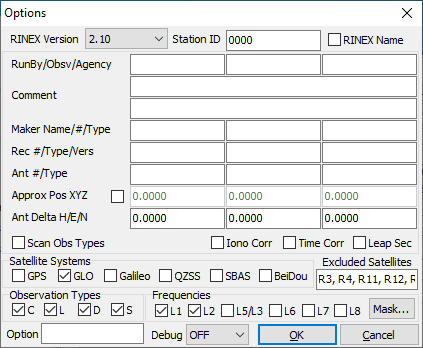


Рисунок 3 – Окно настроек программы RTKCONV ver.2.4.2

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).

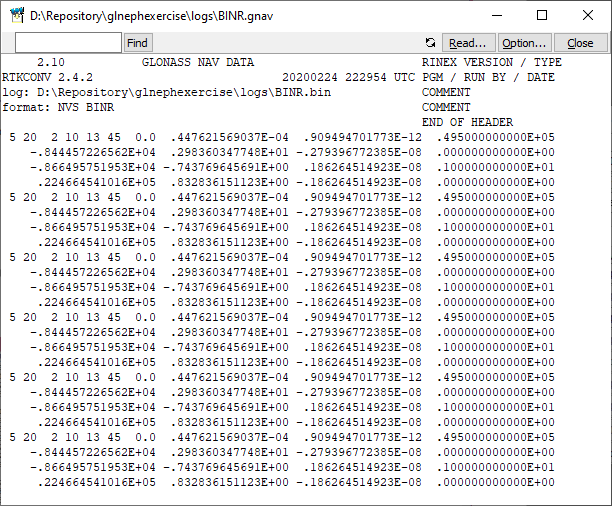


Рисунок 4 – Эфемериды спутника ГЛОНАСС №5 в .gnav файле

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.

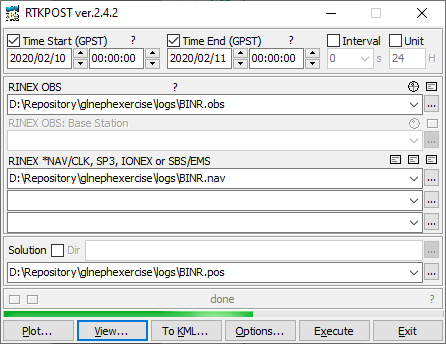


Рисунок 5 – Окно программы RTKPOST ver.2.4.2

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):

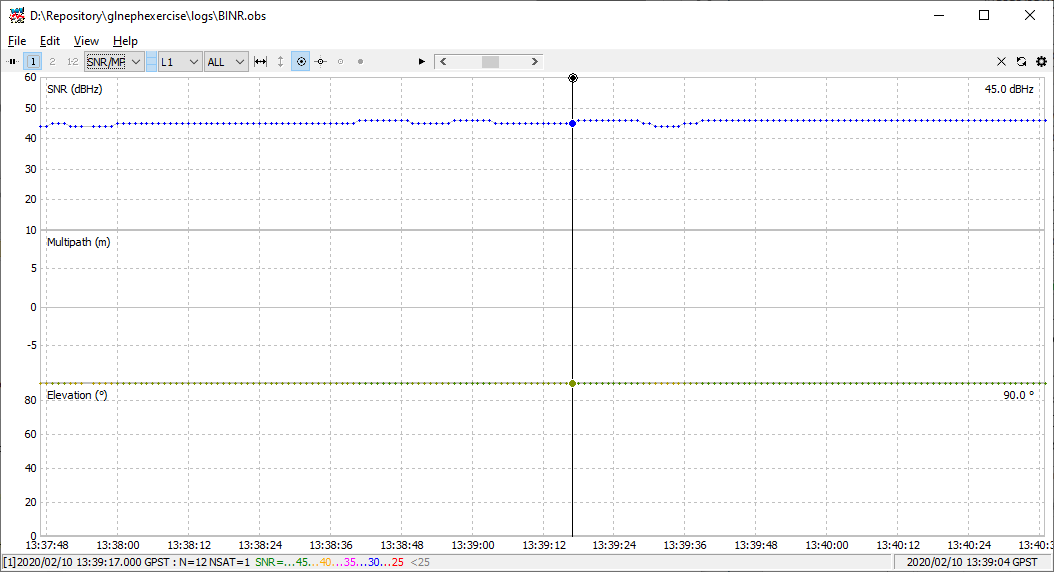


Рисунок 6 – Графики для спутника ГЛОНАСС №5

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

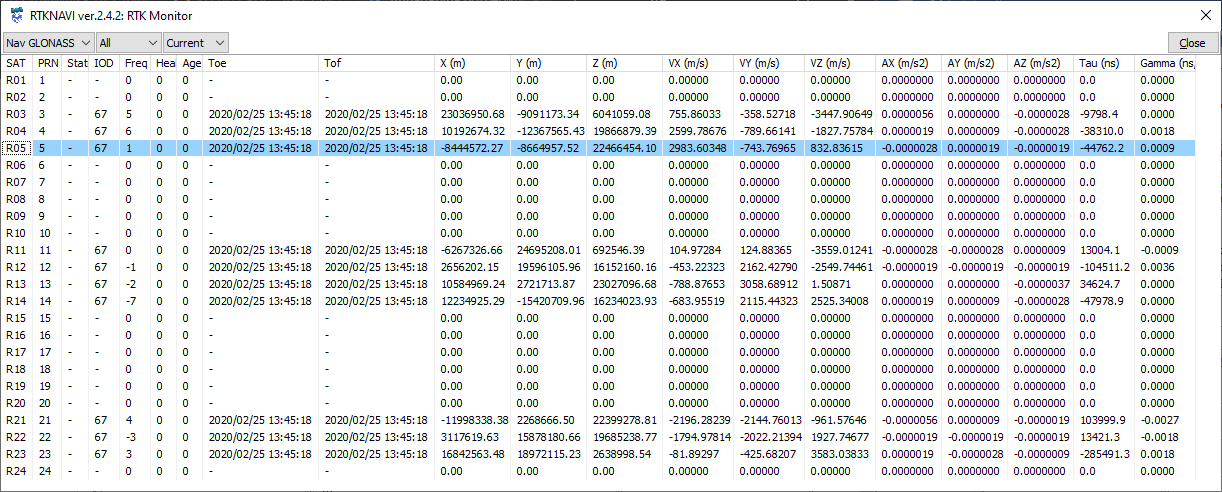


Рисунок 7 – Окно программы RTKNAVI ver.2.4.2: RTK Monitor

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).

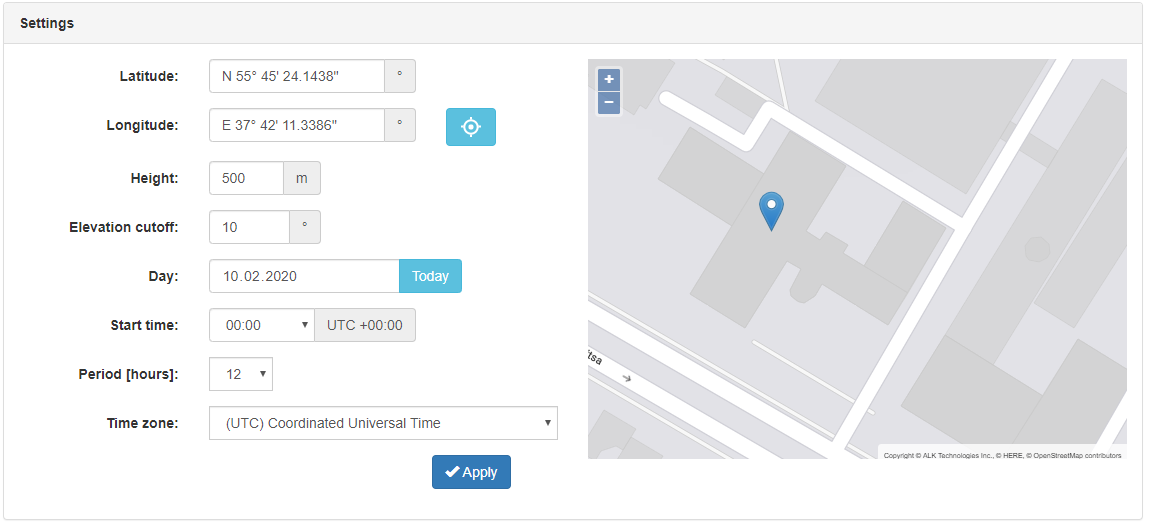


Рисунок 8 – Вкладка настроек (Settings) интернет ресурса Trimble GNSS Planning

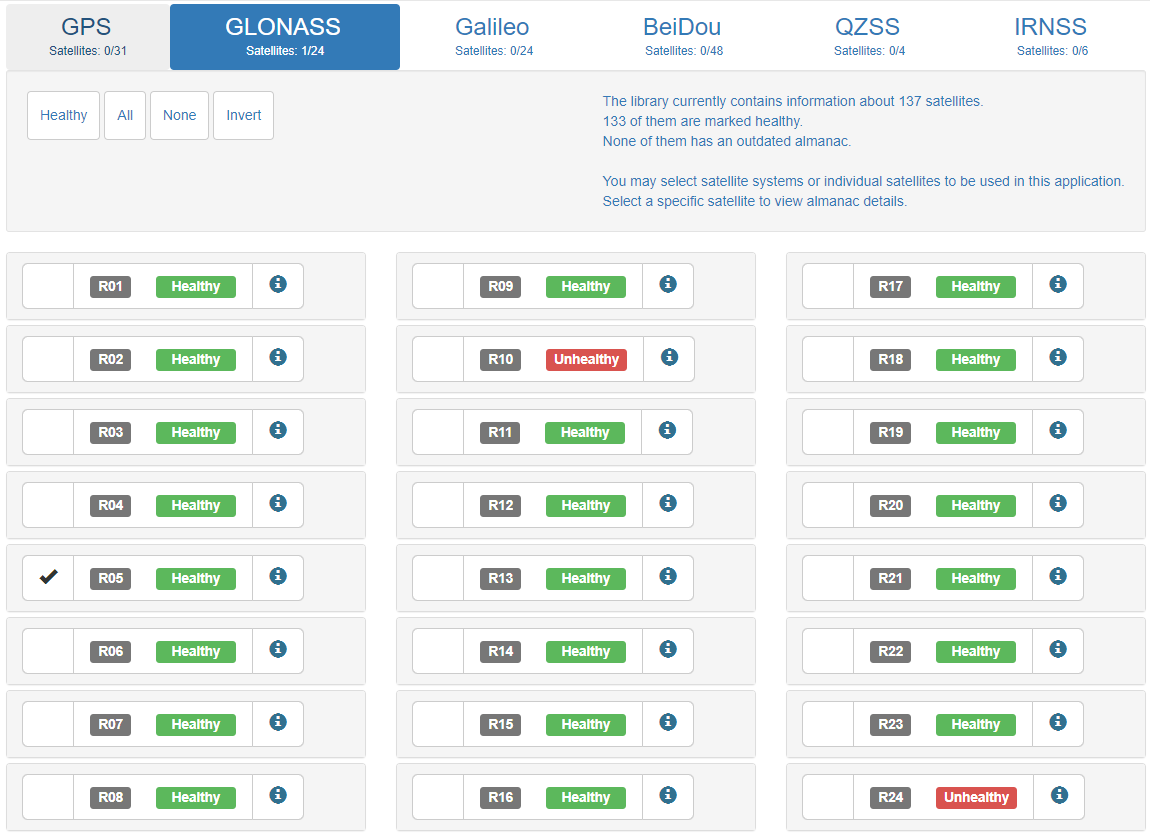


Рисунок 9 – Вкладка библиотека спутников (Satellite Library) интернет ресурса Trimble GNSS Planning

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.

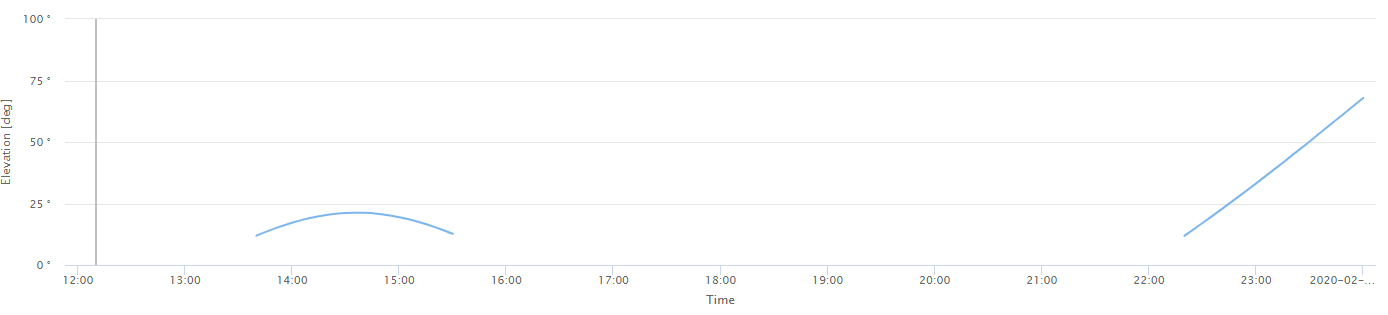


Рисунок 10 – График угла места спутника ГЛОНАСС №5

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.

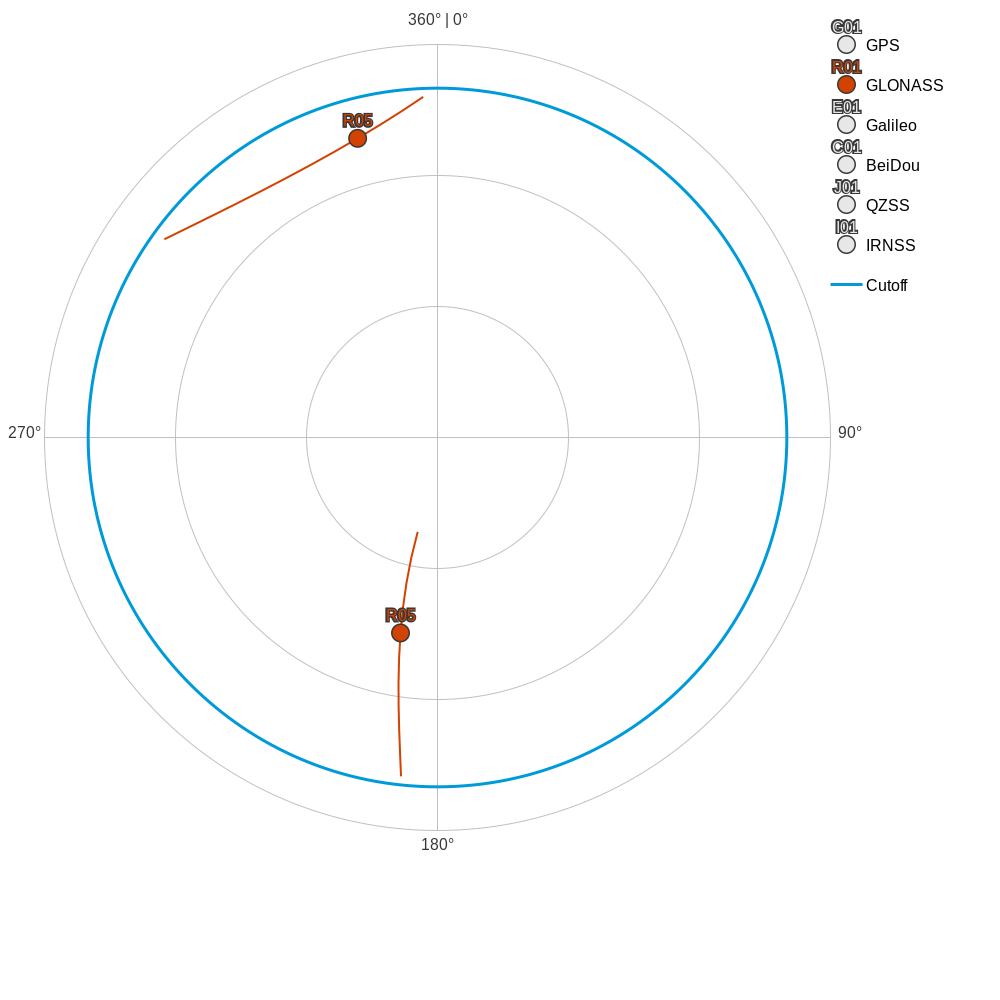


Рисунок 11 – SkyView спутника ГЛОНАСС №5

**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 |
|  | м | -8444572.27 |
|  | -8664957.52 |
|  | 22466454.10 |
|  | м/с | 2983.60348 |
|  | -743.76965 |
|  | 832.83615 |
|  | м/с2 | -0.0000028 |
|  | 0.0000019 |
|  | -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 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.

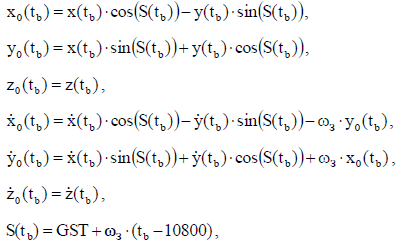
Исходные данные, необходимые для пересчета эфемерид в соответствии с точным алгоритмом:

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

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

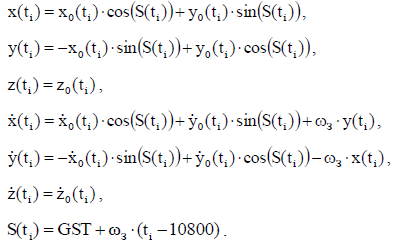


Начальными условиями для интегрирования системы являются координаты центра масс НКА , , и составляющие его вектора скорости , , в инерциальной геоцентрической системе координат OX0Y0Z0 на момент шкалы МДВ. Эти начальные условия вычисляются путем пересчета передаваемых в навигационном сообщении координат , , и составляющих вектора скорости , , центра масс НКА в связанной с Землей системе координат ПЗ-90. Пересчет осуществляется по следующим формулам:



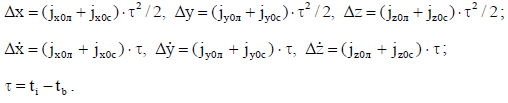
Интегрирование осуществляется численным методом, например, методом Рунге- Кутта 4-го порядка.

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 расчете используются следующие примечания:

* 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 положения спутника ГЛОНАСС**

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

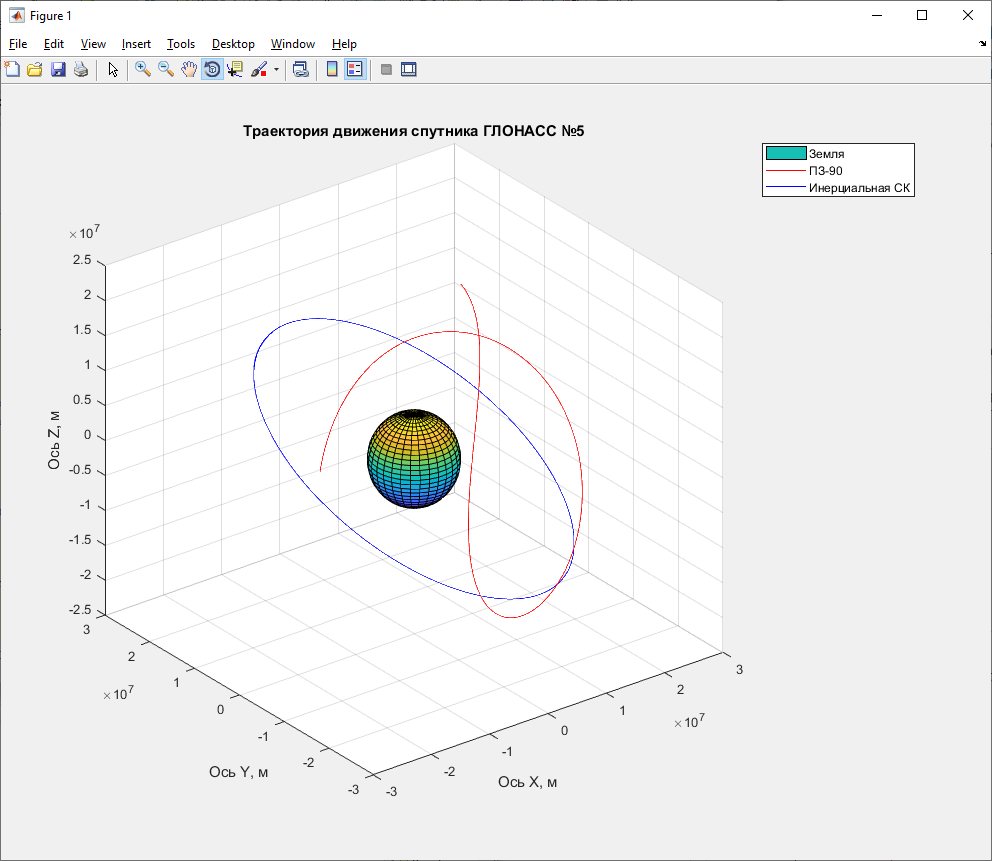


Рисунок 12 – Траектория движения спутника ГЛОНАСС №5 в системе координат ПЗ-90 (красная линия) и в инерциальной системе координат (синяя линия)

**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:

*м*

*м*

*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).

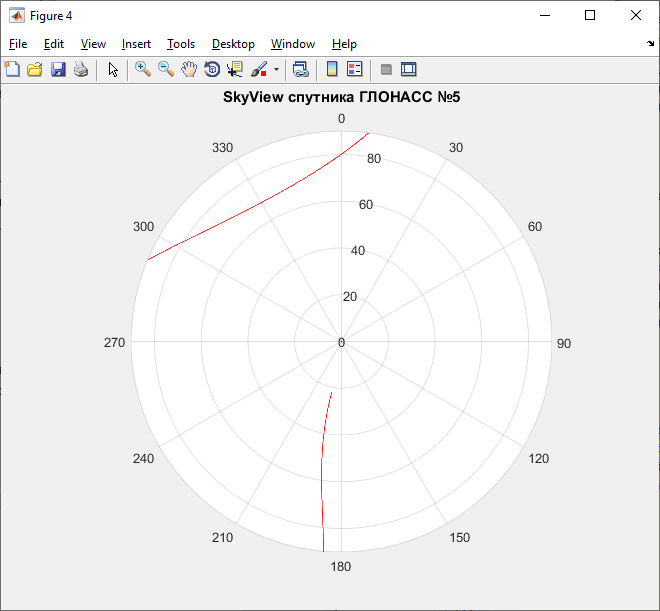


Рисунок 13 – SkyView спутника ГЛОНАСС №5

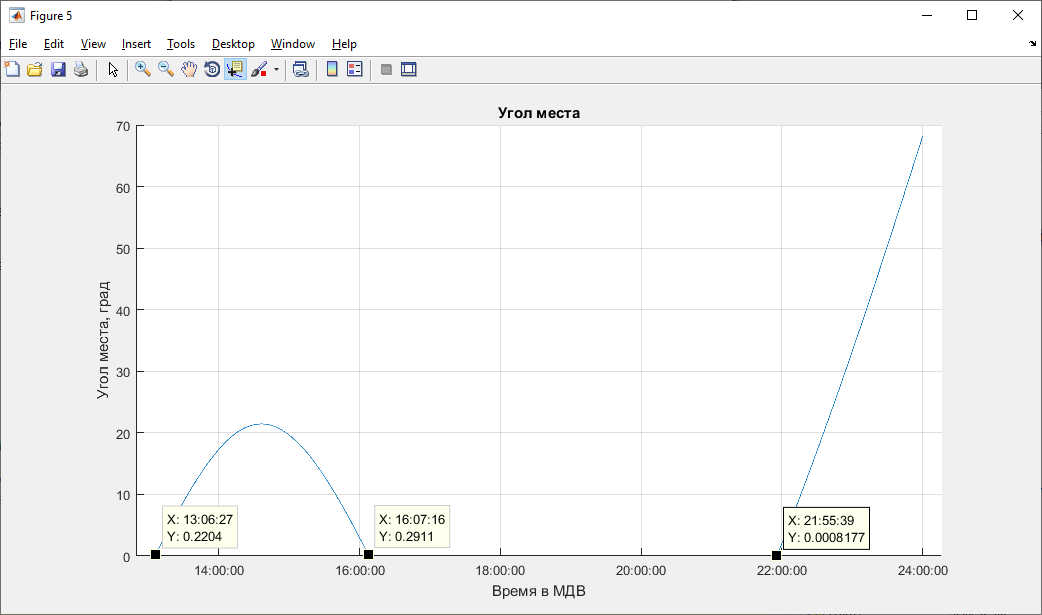


Рисунок 14 – График угла места спутника ГЛОНАСС №5

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 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. 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.

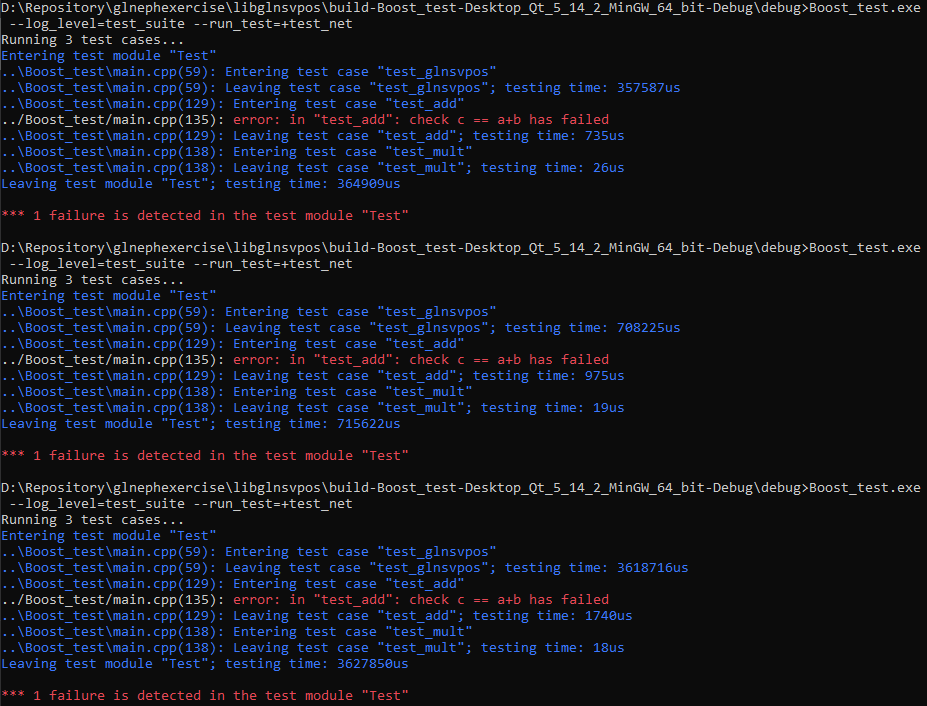


Рисунок 15 – вывод unit-тестов

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

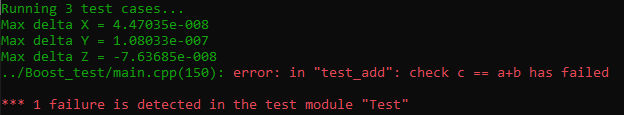


Рисунок 16 – вывод unit-тестов

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.



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 С/С++ 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 С/С++ 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 С/С++, 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 С/С++ 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 С/С++, 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 Х, 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 Х, 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 Х, 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 Х, 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/c2]

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 отcчетов для времени большего текущего Eph.tb

uint64\_t N2dec = (tn - Toe) / (double)h; // Количесвио отcчетов 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/c2]

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