

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«МЭИ»**

ИНСТИТУТ РАДИОТЕХНИКИ И ЭЛЕКТРОНИКИ

КАФЕДРА РАДИОТЕХНИЧЕСКИХ СИСТЕМ

КУРСОВАЯ РАБОТА

по дисциплине

Аппаратура потребителей спутниковых радионавигационных систем

«Разработка модуля расчет координат спутника Beidou»

ФИО СТУДЕНТА: САЛИН Г.А.

ГРУППА: ЭР-15-16

ВАРИАНТ №:20

ДАТА: _____

ПОДПИСЬ: _____

ФИО ПРЕПОДАВАТЕЛЯ: КОРОГОДИН И.В.

ОЦЕНКА: _____

МОСКВА, 2021

Содержание

Введение	3
1 Использование сторонних средств	4
1.1 Описание этапа	4
1.2 CelesTrak	4
1.3 Trimble GNSS Planning Online	6
1.4 Эфемериды в сигнале Beidou B1I	8
1.5 Заключение по результатам использования сторонних средств	10
2 Моделирование	11
2.1 Описание этапа	11
2.2 Алгоритм расчета координат	12
2.3 Результаты расчета координат спутника	13
2.4 Расчет SkyView	14
2.4 Заключение по результатам моделирования	16
3 Реализация	17
3.1 Описание этапа	17
3.2 Программное окружение	17
3.3 Результаты реализации	18
3.3 Проверка на утечки памяти и профилирование	19
3.4 Заключение по результатам реализации	19
4 Заключение	20
5 Список использованных источников	20
6 Приложение	21
6.1 Листинг скрипта Matlab	21
6.2 Листинг программного модуля C/C++	24

Введение

Название проекта: Разработка модуля расчёта координат спутника Beidou.

Цель проекта - добавление в программное обеспечение приемника функции расчета положения спутника Beidou на заданное время по данным его эфемерид.

Требования к разрабатываемому программному модулю:

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

Для достижения цели выполняется ряд задач, соответствующих этапам и контрольным мероприятиям:

- обработка данных от приемника, работа со сторонними сервисами для подготовки входных и проверочных данных для разрабатываемого модуля;
- моделирование модуля в Matlab/Python;
- реализация программного модуля на C/C++, включая юнит-тестирование в Check.

Этапы курсовой работы отличаются осваиваемыми инструментами.

1 Использование сторонних средств

1.1 Описание этапа

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 Lexion LGDD,
- SwiftNavigation Piksi Multi,
- Clonicus разработки ЛНС МЭИ.

Эти приемники осуществляют первичную обработку сигналов Beidou B11, выдавая по интерфейсам соответствующие потоки данных – наблюдения псевдодальностей и эфемериды спутников. Данные от приемника Clonicus, записанные вечером 16 февраля 2021 года, доступны в рабочем репозитории (директория logs) в нескольких форматах.

Во-первых, это дам бинарного потока данных от приемника в формате NVS BINR.

Во-вторых, текстовый файл данных пакета 0xF7, полученный из данного дампа.

1.2 Celestrak

Определение формы орбиты и положения спутника на ней на начало рассматриваемого интервала времени (на 18:00 МСК 16 февраля 2021 года) по данным сервиса Celestrak.

Для определения имя и ID спутника воспользуемся таблицей из ru.wikipedia.org/wiki/Бэйдоу.

№ ↕	Спутник ↕	PRN ↕	Дата (UTC) ↕	Ракета ↕	NSSDC ID ↕	SCN ↕	Орбита ↕	Статус ↕
24	Бэйдоу-3 M1	C19	05.11.2017 11:44	CZ-3B/YZ-1	2017-069A ↗	43001 ↗	COO, ~21 500 км	действующий
25	Бэйдоу-3 M2	C20			2017-069B ↗	43002 ↗	COO, ~21 500 км	действующий

Рисунок 1 – Часть таблицы со списком спутников

Как видно из таблицы, спутник с номером 20 (PRN C20) имеет порядковый номер 25, имя спутника – Бэйдоу-3 M2 и ID 43002 (SCN в таблице).

Теперь зайдём на сайт celestrak.com. Выберем спутник Beidou-3 M2 с ID 43002, и установим заданное время. В сервисе используется время по UTC, которое отличается от времени по МСК на 3:00 часа, поэтому установим следующее время: 15:00:00 2021:02:16. Результаты приведены на рисунках 2 и 3.

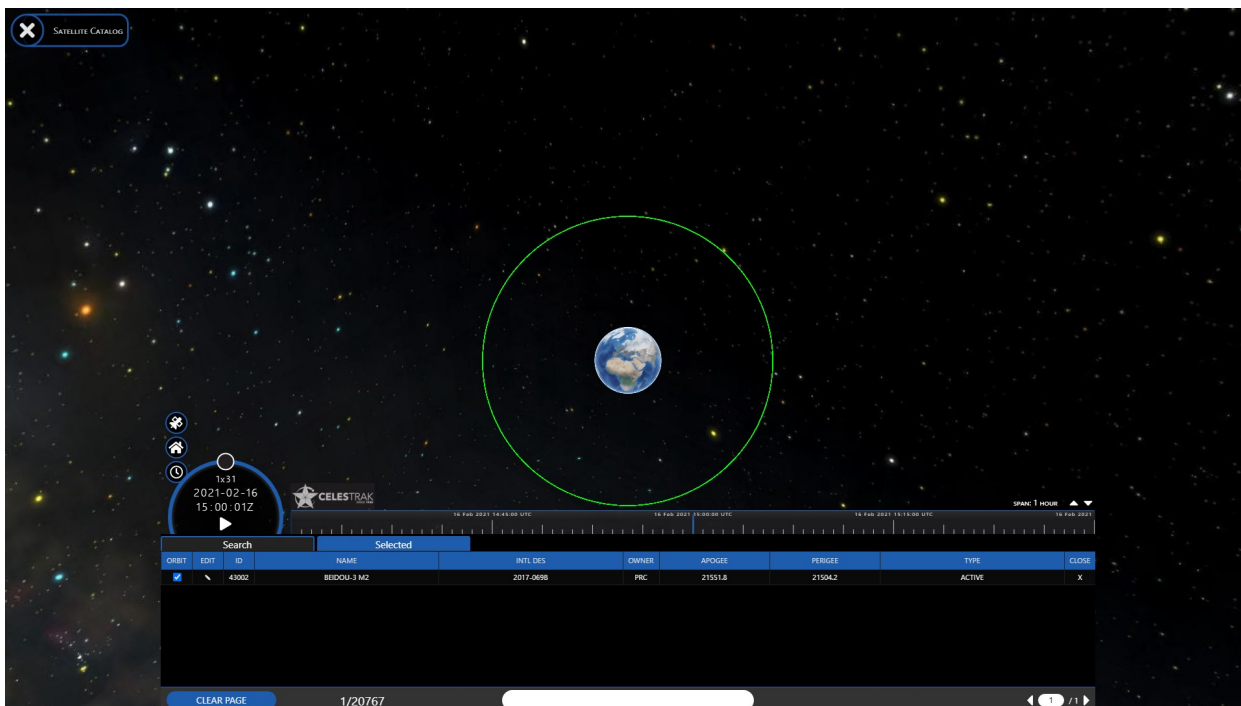


Рисунок 2 – Орбита спутника

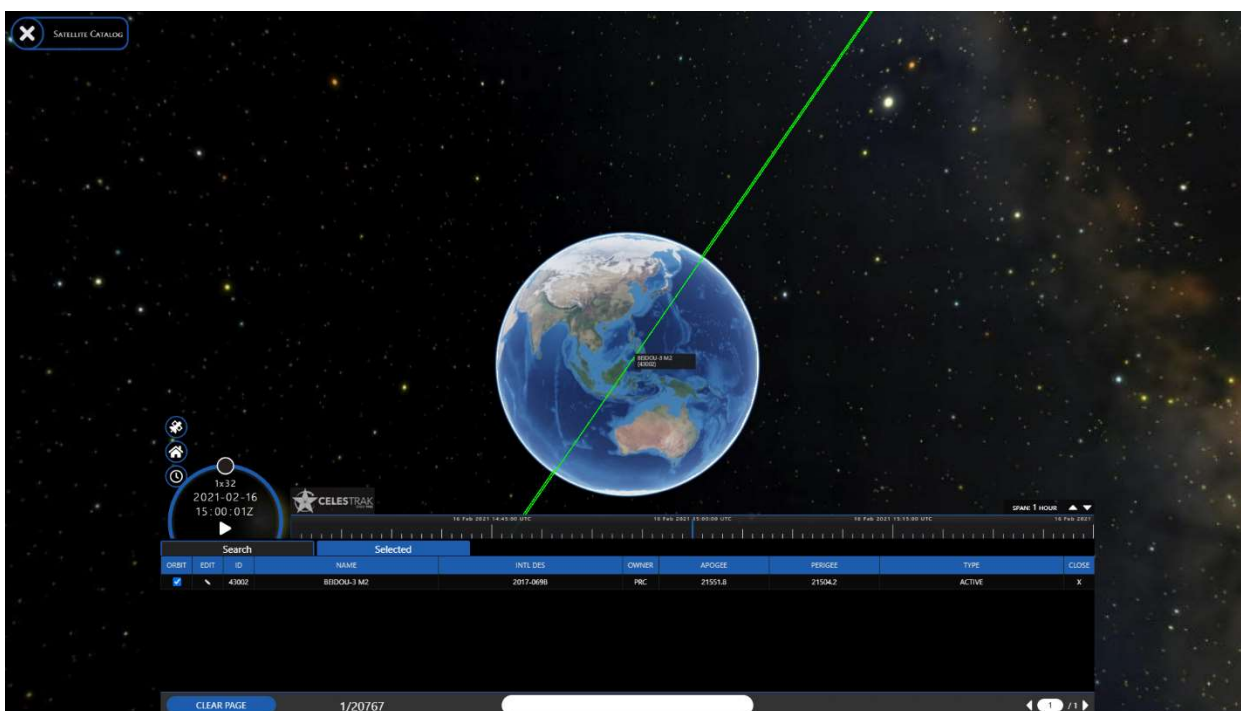


Рисунок 3 – Подспутниковая точка

По рисунку 2 видно, что орбита у спутника Beidou-3 M2 круговая.

По рисунку 3 видно, что подспутниковая точка на заданное время находилась в районе островов Филиппин.

1.3 Trimble GNSS Planning Online

Расчет графика угла места собственного спутника от времени по данным Trimble GNSS Planning Online на интервале времени с 18:00 МСК 16 февраля до 06:00 МСК 17 февраля 2021 года.

Зайдем на сайт gnssplanning.com. По заданному интервалу и расположению приемной антенны устанавливаем следующие настройки (рисунок 4).

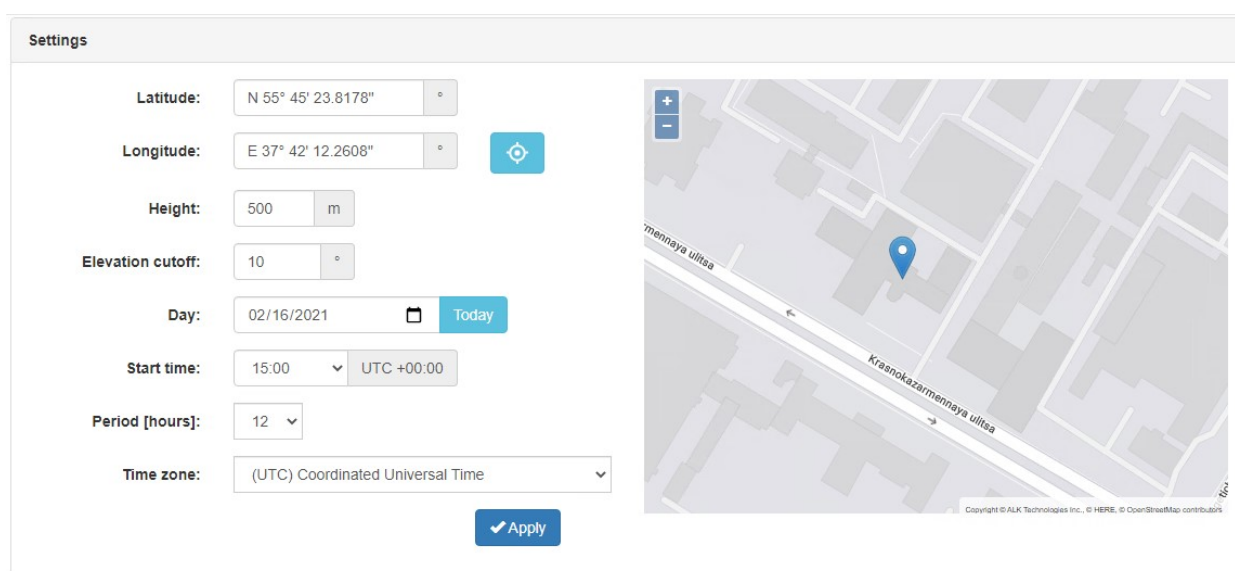


Рисунок 4 – Настройки сервиса Trimble GNSS Planning Online

Зная из предыдущего пункта имя спутника, выберем его и построим график угла места (рисунок 5), график SkyView (рисунок 6) и, для проверки, карту мира с траекторией движения спутника (рисунок 7).

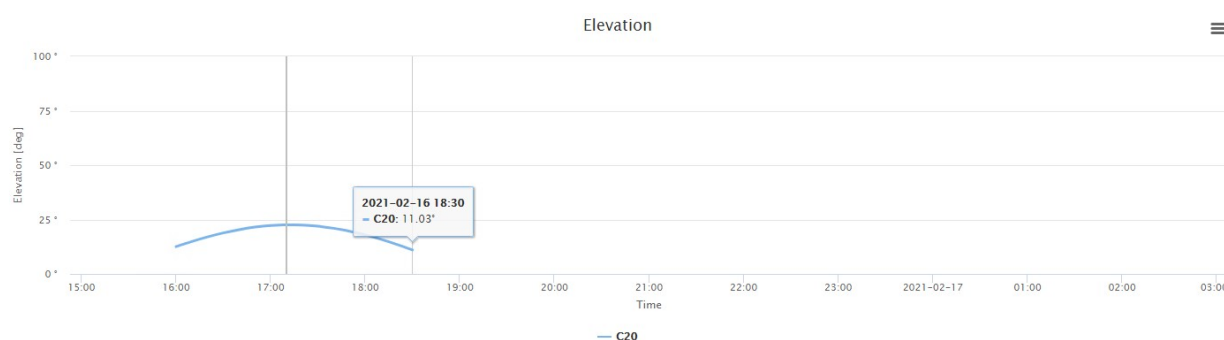


Рисунок 5 – График угла места

По рисунку 5 видно, что спутник попал в поле зрения приемной антенны в 16:00 по UTC (19:00 МСК) и пропал в 18:30 UTC (21:30 МСК). Угол места изменялся от 11.03 до 22.57 градусов.

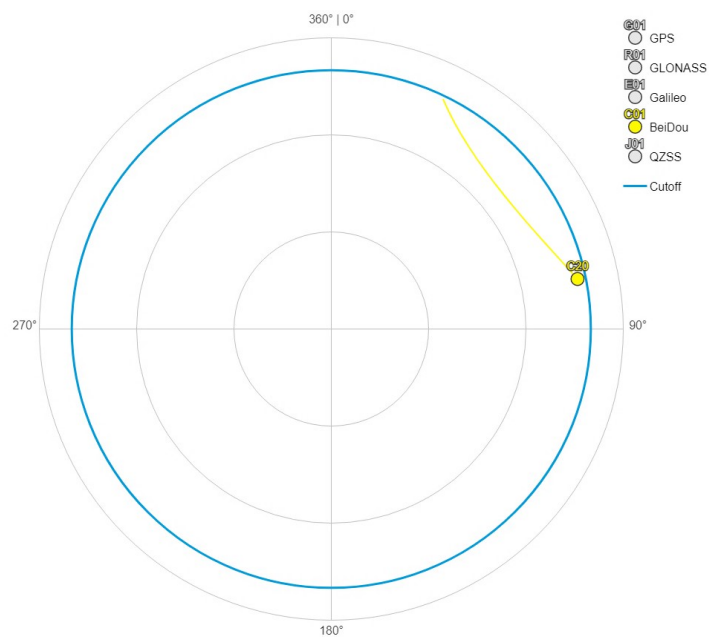


Рисунок 6 – График SkyView

График SkyView показывает, что спутник был виден антенной в азимуте от 0 до 90 градусов.

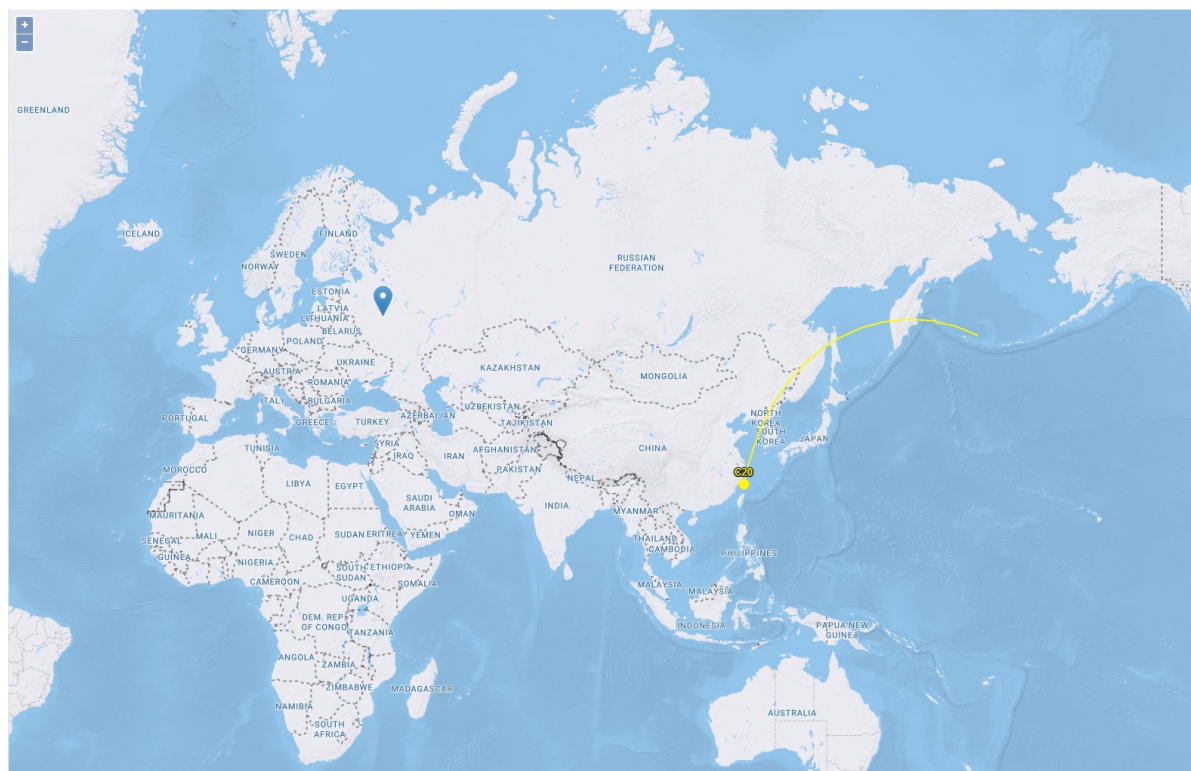


Рисунок 7 – Карта мира с траекторией движения спутника

В тот момент, когда спутник попал в поле зрения приемной антенны (19:00 МСК), его ПТ находилась чуть выше Филиппин, а когда исчез из поля зрения (21:30 МСК) – чуть восточнее полуострова Камчатка.

1.4 Эфемериды в сигнале Beidou B1I

Формирование списка и описание параметров, входящих в состав эфемерид в сигнале Beidou B1I.

Воспользуемся ИКД BeiDou: Navigation Satellite System Signal In Space, Interface Control Document, Open Service Signal B1I (Version 3.0), China Satellite Navigation Office, February 2019. Находим там таблицы 5-9 и 5-10, объединяем их и формируем таблицу с описанием параметров эфемерид.

Таблица 1 – Описание параметров, входящих в состав эфемерид Beidou.

Parameter	Definition	Units
t_{oe}	Ephemeris reference time	s
\sqrt{A}	Square root of semi-major axis	m ^{1/2}
e	Eccentricity	-
ω	Argument of perigee	π
Δn	Mean motion difference from computed value	π/s
M_0	Mean anomaly at reference time	π
Ω_0	Longitude of ascending node of orbital of plane computed according to reference time	π
$\dot{\Omega}$	Rate of right ascension	π/s
i_0	Inclination angle at reference time	π
$IDOT$	Rate of inclination angle	π/s
C_{uc}	Amplitude of cosine harmonic correction term to the argument of latitude	rad
C_{us}	Amplitude of sine harmonic correction term to the argument of latitude	rad
C_{rc}	Amplitude of cosine harmonic correction term to the orbit radius	m
C_{rs}	Amplitude of sine harmonic correction term to the orbit radius	m
C_{ic}	Amplitude of cosine harmonic correction term to the angle of inclination	rad

C_{is}	Amplitude of sine harmonic correction term to the angle of inclination	rad
----------	--	-----

Формирование таблицы эфемерид спутника с подписанными размерностями.

Для этого откроем файл binr_0x00F7_BdsB1I.txt из директории logs/, найдем эфемериды спутника, и сведем их в таблицу 2.

Таблица 2 – Эфемериды спутника Beidou C20.

Параметр	Обозначение переменной	Единица измерения	Значение
PRN	SatNum	-	20
t_{oe}	toe	мс	226800000.000
C_{rs}	Crs	м	-7.392187500000000000e+01
Δn	Dn	рад/с	3.97195124013371981e-12
M_0	m0	рад	8.71704768059675339e-01
C_{uc}	Cuc	рад	-3.64007428288459778e-06
e	e	-	6.97147799655795097e-04
C_{us}	Cus	рад	5.95534220337867737e-06
\sqrt{A}	sqrtA	м ^{1/2}	5.28262682533264160e+03
C_{ic}	Cic	рад	-7.49714672565460205e-08
Ω_0	Omega0	рад	-2.82333800290329728e-01
C_{is}	Cis	рад	-6.84522092342376709e-08
i_0	i0	рад	9.65664043486355039e-01
C_{rc}	Crc	м	2.445000000000000000e+02
ω	omega	рад	-7.73836711576575076e-01
$\dot{\Omega}$	OmegaDot	рад/мс	-7.00779190266137795e-12

$IDOT$	iDot	π/c	-1.97151069276238744e-13
T_{GD}	Tgd	mc	2.3000000000000000e+05
t_{oc}	toc	mc	2.2680000000000000e+08
a_{f2}	af2	mc/mc ²	0.0000000000000000e+00
a_{f1}	af1	mc/mc	-4.24460466774689849e-12
a_{f0}	af0	mc	-9.16242361068725586e-01
URA	URA	-	0
$IODE$	IODE	-	257
$IODC$	IODC	-	1
$codeL2$	codeL2	-	0
$L2P$	L2P	-	0
WN	WN	-	789

1.5 Заключение по результатам использования сторонних средств

В результате использования сторонних средств, такие как сервисы CelesTrak и Trimble GNSS Planning Online, а также данных от приемника Cloniscus, были получены следующие результаты:

- Определена форма орбиты и положение собственного спутника на ней на 18:00 МСК 16 февраля 2021 года.
- Получены график угла места и диаграмма угла места и азимута (SkyView) собственного спутника на интервале времени с 18:00 МСК 16 февраля до 06:00 МСК 17 февраля.
- Сформированы таблица со списком и описанием параметров, входящих в состав эфемерид спутника Beidou, и таблица эфемерид собственного спутника Beidou.

2 Моделирование

2.1 Описание этапа

На предыдущем этапе были получены эфемериды спутника. Эфемериды – параметры некоторой модели движения спутника. В разных ГНСС эти модели разные, а значит отличается и формат эфемерид, и алгоритмы расчета положения спутника.

Одна из самых простых и удобных моделей – в системе GPS. Beidou наследует эту модель.

Требуется реализовать на языке Matlab или Python функцию расчета положения спутника Beidou на заданный момент по шкале времени UTC. В качестве эфемерид использовать данные, полученные на предыдущим этапе.

Построить трехмерные графики множества положений спутника Beidou C20. Графики в двух вариантах: в СК ECEF WGS84 и соответствующей ей инерциальной СК. Положения должны соответствовать временному интервалу с 18:00 МСК 16 февраля до 06:00 МСК 17 февраля 2021 года. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал.

Построить SkyView за указанный временной интервал и сравнить результат с Trimble GNSS Planning Online, полученный на предыдущем этапе.

Таблица 3. Используемые эфемериды

Параметр	Обозначение переменной	Единица измерения	Значение
t_{oe}	toe	мс	226800000.000
C_{rs}	Crs	м	-7.392187500000000000e+01
Δn	Dn	рад/с	3.97195124013371981e-12
M_0	m0	рад	8.71704768059675339e-01
C_{uc}	Cuc	рад	-3.64007428288459778e-06
e	e	-	6.97147799655795097e-04
C_{us}	Cus	рад	5.95534220337867737e-06
\sqrt{A}	sqrtA	м ^{1/2}	5.28262682533264160e+03

C_{ic}	Cic	рад	-7.49714672565460205e-08
Ω_0	Omega0	рад	-2.82333800290329728e-01
C_{is}	Cis	рад	-6.84522092342376709e-08
i_0	i0	рад	9.65664043486355039e-01
C_{rc}	Crc	м	2.44500000000000000e+02
ω	omega	рад	-7.73836711576575076e-01
$\dot{\Omega}$	OmegaDot	рад/мс	-7.00779190266137795e-12
$IDOT$	iDot	π/c	-1.97151069276238744e-13

2.2 Алгоритм расчета координат

Как было сказано выше, Beidou наследует модель движения спутника GPS. Воспользуемся сайтом, https://gssc.esa.int/navipedia/index.php/GPS_and_Galileo_Satellite_Coordinates_Computation.

Алгоритм расчета следующий:

- Вычисление время t_k от эталонной эпохи эфемерид t_{oe} (t и t_{oe} выражаются в секундах от начала недели GPS):

$$t_k = t - t_{oe}$$

Если $t_k > 302\,400$ секунд, необходимо вычесть 604 800 секунд из t_k . Если $t_k < -302\,400$ секунд, необходимо добавить 604 800 секунд.

- Вычисление средней аномалии на t_k :

$$M_k = M_0 + \left(\frac{\sqrt{\mu}}{\sqrt{a^3}} + \Delta n \right) t_k$$

Где $a = (\sqrt{A})^2$ – длина большой полуоси;

$\mu = 3,986004118 \times 10^{14}$ – гравитационная постоянная.

- Решение (итеративное) уравнения Кеплера для нахождения аномалии эксцентриситета E_k :

$$M_k = E_k - e \sin E_k$$

$$E_k = M_k + e \sin E_k$$

Начальное условие: $E_k = M_k$;

Точность вычисления: $\varepsilon = 10^{-6}$.

- Вычисление истинной аномалии v_k :

$$v_k = \arctan\left(\frac{\sqrt{1-e^2} \sin E_k}{\cos E_k - e}\right)$$

- Вычисление аргумента широты u_k :

$$u_k = \omega + v_k + C_{uc} \cos 2(\omega + v_k) + C_{us} \sin 2(\omega + v_k)$$

- Вычисление радиальное расстояние r_k :

$$r_k = A(1 - e \cos E_k) + C_{rc} \cos 2(\omega + v_k) + C_{rs} \sin 2(\omega + v_k)$$

- Вычисление наклона орбитальной плоскости i_k :

$$i_k = i_0 + IDOT \cdot t_k + C_{ic} \cos 2(\omega + v_k) + C_{is} \sin 2(\omega + v_k)$$

- Вычисление долготы восходящего узла λ_k :

$$\lambda_k = \Omega_0 + (\dot{\Omega} - \omega_E)t_k - \omega_E t_{oe}$$

Где $\omega_E = 7,2921151467 \times 10^{-5}$ – скорость вращения Земли.

Далее, чтобы не пользоваться матрицами поворота, воспользуемся ИКД Beidou, ссылка: <http://www.beidou.gov.cn/xt/gfxz/201902/P020190227593621142475.pdf>

- Вычисление позиции спутника в орбитальной плоскости:

$$\begin{cases} x_k = r_k \cos u_k \\ y_k = r_k \sin u_k \end{cases}$$

- Вычисление координат спутника:

$$\begin{cases} X_k = x_k \cos \lambda_k - y_k \cos i_k \sin \lambda_k \\ Y_k = x_k \sin \lambda_k + y_k \cos i_k \cos \lambda_k \\ Z_k = y_k \sin i_k \end{cases}$$

2.3 Результаты расчета координат спутника

Для моделирования выбран язык Matlab. Скрипты программы находятся в директории simulation проекта. Листинг программы приведен в приложении.

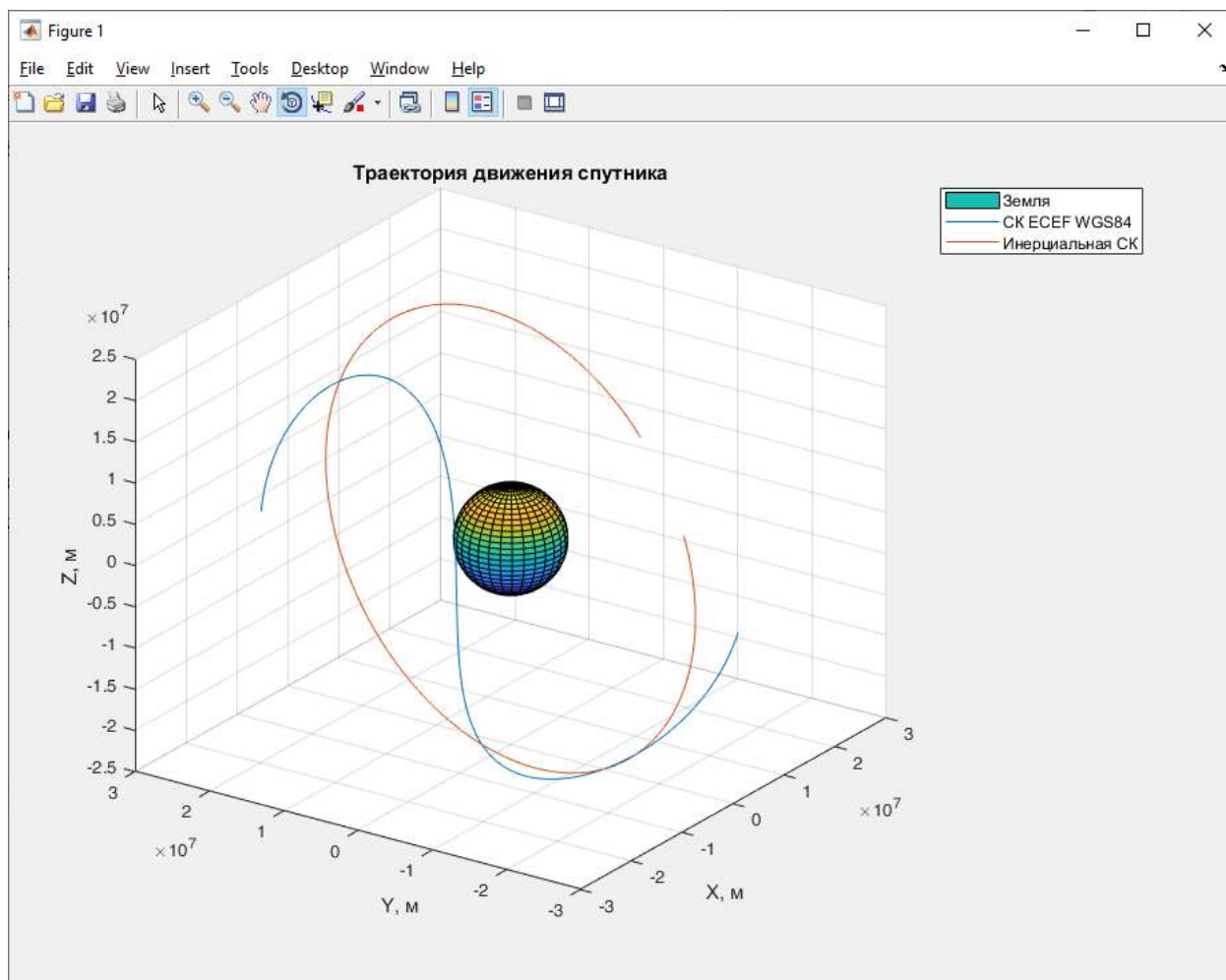


Рисунок 8 – Траектория движения спутника Beidou C20 в СК ECEF WGS84 и в инерциальной СК вокруг Земли.

2.4 Расчет SkyView

Для расчета SkyView необходимо перейти в локальную СК приемника ECEF WGS-84. Для этого воспользуемся сторонней функцией `llh2xyz()`, которая преобразует широту, долготу и высоту в декартовы координаты ECEF.

Расчетные координаты приемника:

$$X = 2846,341 \text{ км}$$

$$Y = 2200,173 \text{ км}$$

$$Z = 5249,655 \text{ км}$$

Далее пересчитаем локальные декартовы координаты в сферические, получив, тем самым азимут, угол места и расстояние. По этим данным построим графики SkyView и угла места и сравним их с данными Trimble GNSS Planning Online.

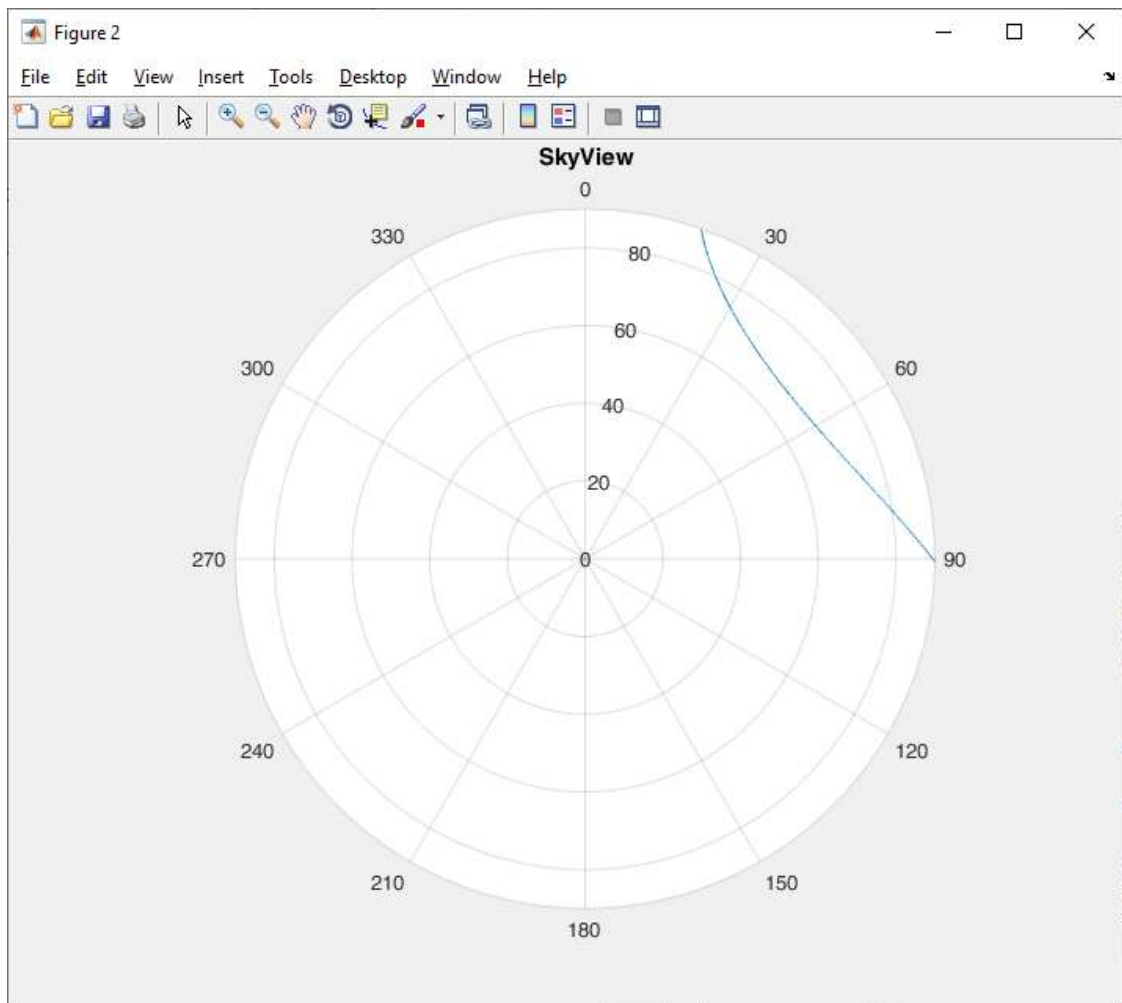


Рисунок 9 – График SkyView

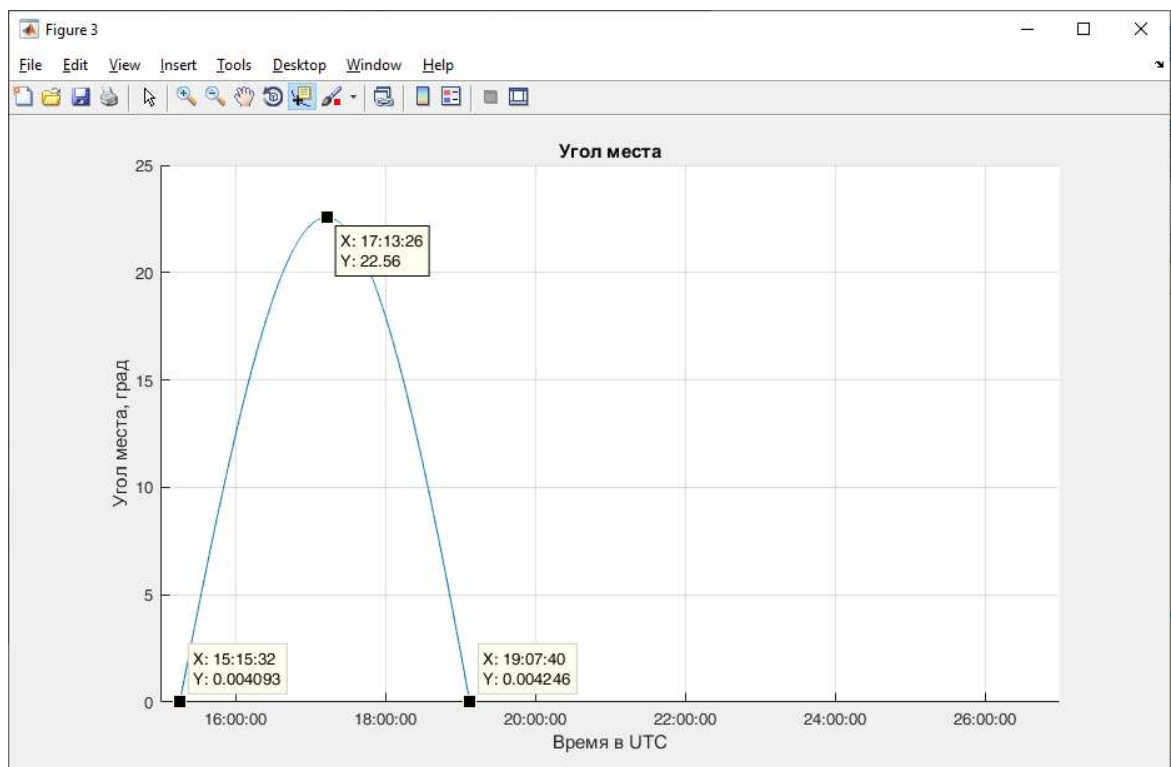


Рисунок 10 – График угла места

По моделированию SkyView и графика угла места видно, что спутник попал в зону видимости приемной антенны в 15:15:32 UTC и вышел из зоны – в 19:07:40 UTC. Данные результатов моделирования совпадают с данными Trimble GNSS Planning Online с погрешностью. Это объясняется тем, что использовались одни и те же эфемериды на весь интервал расчета, что приводит к такого рода погрешностям.

2.4 Заключение по результатам моделирования

На данном этапе была реализована на языке Matlab функция расчета положения спутника Beidou C20 на интервале времени с 18:00 МСК 16 февраля до 06:00 МСК 17 февраля. В качестве эфемерид использовались данные, полученные на предыдущем этапе.

Использовались одни и те же эфемериды на весь рассматриваемый интервал, что привело к погрешностям расчета. Для избежание погрешностей, необходимо обновлять эфемериды по мере их получения.

В результате моделирования были получены графики траекторий движения спутника Beidou C20 в системах координат ECEF WGS84 и соответствующей ей инерциальной СК. Графики SkyView и угла места от времени с учетом координат приемника.

3 Реализация

3.1 Описание этапа

Требуется разработать на языке C/C++ функцию расчета положения спутника Beidou на заданное время по шкале UTC, минимизировать время её исполнения и количество затрачиваемой оперативной памяти. Вызов функции не должен приводить к выбросу исключений или утечкам памяти при любом наборе входных данных.

Функции 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. Требуется выполнить свою реализацию решения трансцендентного уравнения.

Программный модуль должен сопровождаться unit-тестами:

- Тесты функций решения уравнения Кеплера.
- 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. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал.

Требуется провести проверку на утечки памяти с помощью утилиты Valgrind.

3.2 Программное окружение

Для удобства разработки используется операционная система Linux, установленная на виртуальную машину VirtualBox. Программный модуль реализован на языке C/C++ в среде разработки Qt Creator. Список используемых пакетов при разработке:

- gcc – компилятор языка C;
- g++ – компилятор языка C++;
- qt5-default – среда разработки Qt Creator;
- qtcreator – среда разработки Qt Creator;
- qt5-qmake – система сборки qmake;
- valgrind – программа для проверки на утечки памяти.

Листинг программы приведен в приложении.

3.3 Результаты реализации

Для сравнения координат, рассчитанных в Matlab и в программном модуле на C/C++, требуется, как минимум, получить массив координат из модели. Для этого добавим в скрипт матлаба, после расчета самих координат, следующие строки:

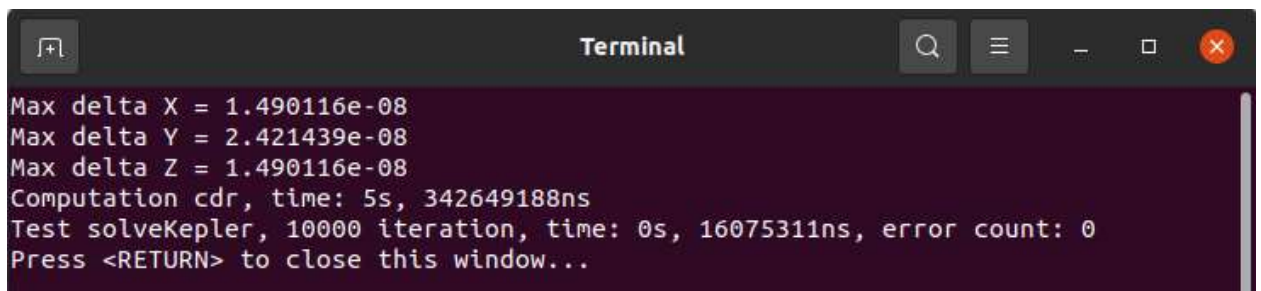
```
% Запись в файл
data_out = fopen(' ../data_matlab.txt', 'w+');      % открытие файла на запись
if data_out == -1                                   % проверка корректности открытия
    error('File is not opened');
end
for k = 1:length(X0)
    fprintf(data_out, '%22.17f %22.17f %22.17f\n', X0(k), Y0(k), Z0(k)); %
    запись в файл
end
fclose(data_out);                                   % закрытие файла
```

После запуска скрипта в директории проекта появится файл с именем data_matlab.txt, который содержит массив координат модели. Этот файл программный модуль должен считать и сравнить со значениями, полученными в результате своих внутренних вычислений.

При запуске программного модуля вызывается две функции: одна для расчета массива координат, записи его в файл и сравнения с моделью, при это высчитывается время выполнения bdssvpos(); вторая для тестирования уравнения Кеплера test_solveKepler(), входным параметром который служит количество итераций решения уравнения.

Вывод результатов исполнения программного модуля в консоль показан на рисунке 11. В результатах показаны:

- Максимальная разница в метрах для каждой координаты;
- Время исполнения первой функции;
- Результаты тестирования уравнения Кеплера:
 - Количество итераций;
 - Время выполнения;
 - Количество ошибок.



```
Terminal
Max delta X = 1.490116e-08
Max delta Y = 2.421439e-08
Max delta Z = 1.490116e-08
Computation cdr, time: 5s, 342649188ns
Test solveKepler, 10000 iteration, time: 0s, 16075311ns, error count: 0
Press <RETURN> to close this window...
```

Рисунок 11 – Вывод программы в консоль

3.3 Проверка на утечки памяти и профилирование

Для проведения процедуры проверки на утечки памяти и профилирования воспользуемся программой valgrind. Так модуль Memcheck позволяет выявить утечки памяти, а Callgrind – показать результаты профилирования. Модули сильно снижают производительность программы, поэтому снизим количество итераций тестирования уравнения Кеплера до одной. Результаты проверки модуля Memcheck на утечки памяти показан на рисунке 12. Как можно видеть – проблем не обнаружено. Результат профилирования модулем Callgrind показан на рисунке 13

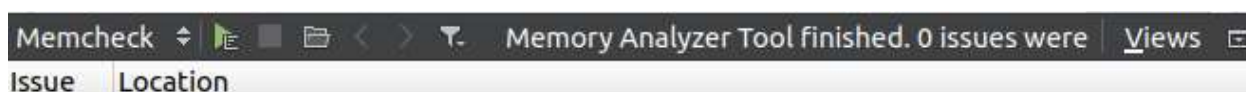
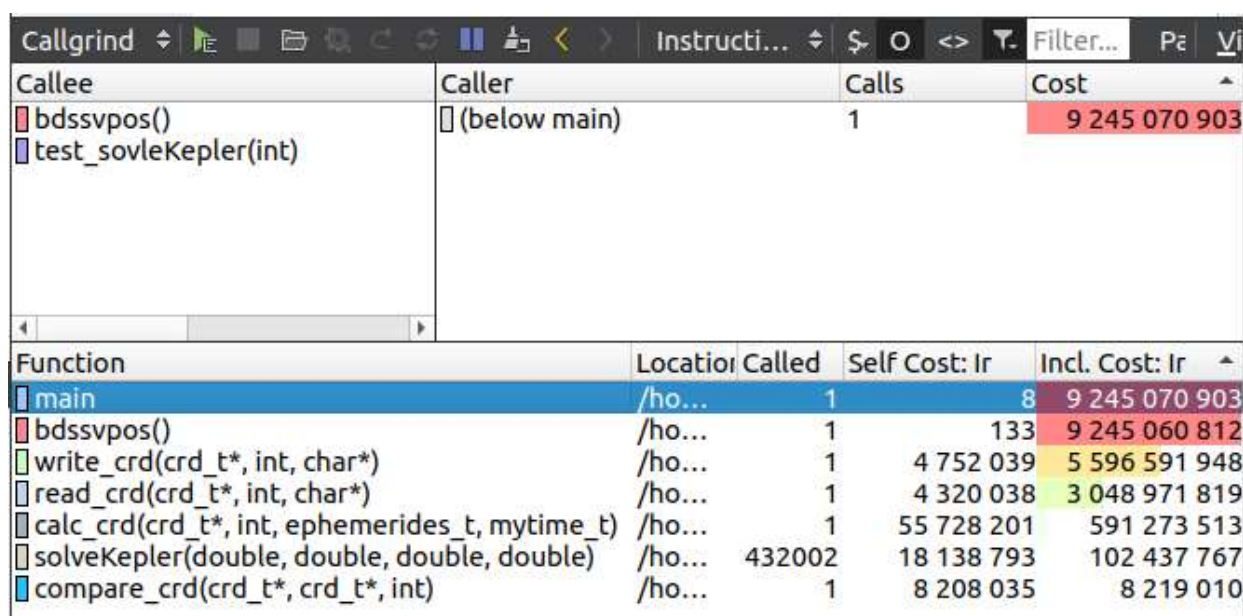


Рисунок 12 – Результат проверки модуля Memcheck

The image shows the Valgrind Callgrind interface. It displays a call graph with 'Callee' and 'Caller' columns. Below that is a table with columns: 'Function', 'Location', 'Called', 'Self Cost: Ir', and 'Incl. Cost: Ir'. The table lists several functions including 'main', 'bdssvpos()', 'write_crd()', 'read_crd()', 'calc_crd()', 'solveKepler()', and 'compare_crd()'.

Function	Location	Called	Self Cost: Ir	Incl. Cost: Ir
main	/ho...	1	8	9 245 070 903
bdssvpos()	/ho...	1	133	9 245 060 812
write_crd(crd_t*, int, char*)	/ho...	1	4 752 039	5 596 591 948
read_crd(crd_t*, int, char*)	/ho...	1	4 320 038	3 048 971 819
calc_crd(crd_t*, int, ephemerides_t, mytime_t)	/ho...	1	55 728 201	591 273 513
solveKepler(double, double, double, double)	/ho...	432002	18 138 793	102 437 767
compare_crd(crd_t*, crd_t*, int)	/ho...	1	8 208 035	8 219 010

Рисунок 13 – Результат профилирования модулем Callgrind

3.4 Заключение по результатам реализации

В данном этапе была реализована на языке C/C++ функция расчета положения спутника Veidou на заданное время по шкале UTC. Функция сопровождается тестами решения уравнения Кеплера, проверкой на утечки памяти и профилированием.

Погрешность вычисления координат функцией и моделью составляет порядка $2 \cdot 10^{-8}$ м, при использовании вещественного типа данных двойной точности double.

Время выполнение функции составляет 5,34 с. Если исключить из функции процедуры записи, чтения и сравнения координат, то время выполнения существенного увеличится.

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++, позволяющие рассчитывать положения спутников Beidou по эфемеридам на заданный интервал времени.

5 Список использованных источников

1. BeiDou Navigation Satellite System. System In Space. Interface Control Document.
2. ru.wikipedia.org/wiki/Бейдоу
3. celestrak.com
4. gnssplanningonline.com
5. gssc.esa.int/navipedia/index.php/GPS_and_Galileo_Satellite_Coordinates_Computation

6 Приложение

6.1 Листинг скрипта Matlab

Stage2.m

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

%% Эфемериды
SatNum = 20;
toe = 226800000.000 * 10^-3;
Crs = -7.392187500000000000e+01;
Dn = 3.97195124013371981e-12;
M0 = 8.71704768059675339e-01;
Cuc = -3.64007428288459778e-06;
e = 6.97147799655795097e-04;
Cus = 5.95534220337867737e-06;
sqrtA = 5.28262682533264160e+03;
Cic = -7.49714672565460205e-08;
Omega0 = -2.82333800290329728e-01;
Cis = -6.84522092342376709e-08;
i0 = 9.65664043486355039e-01;
Crc = 2.445000000000000000e+02;
omega = -7.73836711576575076e-01;
OmegaDot = -7.00779190266137795e-12;
iDot = -1.97151069276238744e-13;
Tgd = 2.300000000000000000e+05;
toc = 2.268000000000000000e+08;
af2 = 0.000000000000000000e+00;
af1 = -4.24460466774689849e-12;
af0 = -9.16242361068725586e-01;
URA = 0;
IODE = 257;
IODC = 1;
codeL2 = 0;
L2P = 0;
WN = 789;

%% Константы
mu = 3.986004418e14;      % гравитационная постоянная
OmegaE = 7.2921151467e-5; % скорость вращения

%% Расчет
t_start = (24*2 + 18 - 3)*60*60; % время старта 18:00 МСК 16 февраля
t_stop = (24*3 + 6 - 3)*60*60; % время окончания 6:00 МСК 17 февраля

% Массив времени
t_arr = t_start:0.1:t_stop;

% Большая полуось
A = sqrtA^2;

% Среднее движение
n0 = sqrt(mu/A^3);
n = n0 + Dn;

for k = 1:length(t_arr)
    % Время
    t(k) = t_arr(k) - toe;
```

```

if t(k) > 302400
    t(k) = t(k) - 604800;
end
if t(k) < -302400
    t(k) = t(k) + 604800;
end

% Средняя аномалия
M(k) = M0 + n*t(k);

% Решение уравнения Кеплера
E(k) = M(k);
E_old(k) = M(k)+1;
epsilon = 1e-6;

while abs(E(k) - E_old(k)) > epsilon
    E_old(k) = E(k);
    E(k) = M(k) + e*sin(E(k));
end

% Истинная аномалия
nu(k) = atan2( sqrt(1 - e^2) * sin(E(k)) , cos(E(k)) - e );

% Коэффициенты коррекции
cor_cos(k) = cos(2*(omega + nu(k)));
cor_sin(k) = sin(2*(omega + nu(k)));

% Аргумент широты
u(k) = omega + nu(k) + Cuc*cor_cos(k) + Cus*cor_sin(k);

% Радиус
r(k) = A * (1 - e * cos(E(k))) + Crc*cor_cos(k) + Crs*cor_sin(k);

% Наклон
i(k) = i0 + iDot * t(k) + Cic*cor_cos(k) + Cis*cor_sin(k);

% Долгота восходящего угла
lambda(k) = Omega0 + (OmegaDot - OmegaE) * t(k) - OmegaE*toe;

% Положение на орбите
x = r(k) * cos(u(k));
y = r(k) * sin(u(k));

% Координаты
X0(k) = x * cos(lambda(k)) - y * cos(i(k)) * sin(lambda(k));
Y0(k) = x * sin(lambda(k)) + y * cos(i(k)) * cos(lambda(k));
Z0(k) = y * sin(i(k));

% Пересчет в инерциальную СК
X(k) = X0(k) * cos(lambda(k)) + Y0(k) * sin(lambda(k));
Y(k) = -X0(k) * sin(lambda(k)) + Y0(k) * cos(lambda(k));
Z(k) = Z0(k);
end

% Запись в файл
data_out = fopen('../data_matlab.txt', 'w+'); % открытие файла на запись
if data_out == -1 % проверка корректности открытия
    error('File is not opened');
end
for k = 1:length(X0)

```

```

        fprintf(data_out, '%22.17f %22.17f %22.17f\n', X0(k), Y0(k), Z0(k)); %
запись в файл
end
fclose(data_out); % закрытие файла

%% 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 = [X0; Y0; Z0];

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

%% построение графиков
R_Earth = 6371e3;
[XE,YE,ZE] = sphere(30);

figure
surf(XE*R_Earth,YE*R_Earth,ZE*R_Earth)
hold on
grid on
plot3(crd_WGS_84(:,1), crd_WGS_84(:,2), crd_WGS_84(:,3))
plot3(X, Y, Z)
plot3(crdCpp(:,1), crdCpp(:,2), crdCpp(:,3))
title('Траектория движения спутника', 'FontName', 'Arial')
xlabel('X, м', 'FontName', 'Arial')
ylabel('Y, м', 'FontName', 'Arial')
zlabel('Z, м', 'FontName', 'Arial')
hold off
lgd = legend('Земля','СК ECEF WGS84','Инерциальная СК');
lgd.FontName = 'Arial';

%% Географические координаты корпуса Е и их перевод в систему WGS-84
% Lantitude
N_gr = 55;
N_min = 45;
N_sec = 23.8178;
N = N_gr*pi/180 + N_min/3437.747 + N_sec/206264.8; % широта [рад]

% Longitude
E_gr = 37;
E_min = 42;
E_sec = 12.2608;
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)';

```

```

%% Построение 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));
        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

% Скайплот
figure
ax = polaraxes;
polarplot(ax,phi,teta*180/pi)
ax.ThetaDir = 'clockwise';
ax.ThetaZeroLocation = 'top';
title('SkyView')

% Угол места
th = hours(t_arr/60/60 - 2*24); % Перевод временной оси в формат hh:mm:ss
figure
grid on
hold on
plot(th, (-teta)*180/pi+90, 'DurationTickFormat', 'hh:mm:ss')
xlim([th(1) th(end)])
title('Угол места', 'FontName', 'Arial')
xlabel('Время в UTC', 'FontName', 'Arial')
ylabel('Угол места, град', 'FontName', 'Arial')

```

6.2 Листинг программного модуля C/C++

main.cpp

```

#include "include/libbdssvpos/bdssvpos.h"
#include "include/libbdssvpos/kepler.h"

#include "test/tests.h"

int main()
{
    bdssvpos();

    test_sovleKepler(1);

    return 0;
}

```

bdssvpos.h

```

#ifndef BDSSVPOS_H
#define BDSSVPOS_H

```



```

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

#include "structures.h"

void bdssvpos();

void calc_crd(struct crd_t* crd, int N, struct ephemerides_t eph, struct
mytime_t t);

void write_crd(struct crd_t* crd, int N, char *fname);

void read_crd(struct crd_t* crd, int N, char *fname);

void compare_crd(struct crd_t* crd1, struct crd_t* crd2, int N);

int add(int a, int b);

#endif /* #ifndef BDSSVPOS_H */

```

bdssvpos.cpp

```

#include <include/libbdssvpos/bdssvpos.h>
#include <include/libbdssvpos/kepler.h>

void bdssvpos()
{
    // Структуры содержащие время начала, конца и разницу
    struct timespec time_start, time_stop, time_delta;

    // Получение времени начала
    timespec_get(&time_start, TIME_UTC);

    // Эфемериды
    struct ephemerides_t eph;
    eph.SatNum = 20;
    eph.toe = 226800000.000 * 1e-3;
    eph.Crs = -7.392187500000000000e+01;
    eph.Dn = 3.97195124013371981e-12;
    eph.M0 = 8.71704768059675339e-01;
    eph.Cuc = -3.64007428288459778e-06;
    eph.e = 6.97147799655795097e-04;
    eph.Cus = 5.95534220337867737e-06;
    eph.sqrtA = 5.28262682533264160e+03;
    eph.Cic = -7.49714672565460205e-08;
    eph.Omega0 = -2.82333800290329728e-01;
    eph.Cis = -6.84522092342376709e-08;
    eph.i0 = 9.65664043486355039e-01;
    eph.Crc = 2.44500000000000000e+02;
    eph.omega = -7.73836711576575076e-01;
    eph.OmegaDot = -7.00779190266137795e-12;
    eph.iDot = -1.97151069276238744e-13;
    eph.Tgd = 2.30000000000000000e+05;
    eph.toc = 2.26800000000000000e+05;
    eph.af2 = 0.00000000000000000e+00;
    eph.af1 = -4.24460466774689849e-12;
    eph.af0 = -9.16242361068725586e-01;
    eph.URA = 0;
    eph.IODE = 257;
    eph.IODC = 1;
}

```

```

eph.codeL2 = 0;
eph.L2P = 0;
eph.WN = 789;

// Формирование массива времени
struct mytime_t t;
// время старта 18:00 МСК 16 февраля
t.start = (24*2 + 18 - 3)*60*60;
// время окончания 6:00 МСК 17 февраля
t.stop = (24*3 + 6 - 3)*60*60;
// шаг
t.step = 0.1;

double t_len = (t.stop - t.start) / t.step;
int N = (int)t_len + 1;

struct crd_t* crdCpp = new struct crd_t[N];
struct crd_t* crdMatlab = new struct crd_t[N];

// Расчет координат
calc_crd(crdCpp, N, eph, t);

// Запись в файл
write_crd(crdCpp, N, "../data_cpp.txt");

// Чтение из файла
read_crd(crdMatlab, N, "../data_matlab.txt");

// Сравнение координат, рассчитанных с помощью C++ и матлаба
compare_crd(crdCpp, crdMatlab, N);

// Получение времени окончания
timespec_get(&time_stop, TIME_UTC);

// Расчет разницы времени
time_delta.tv_sec = time_stop.tv_sec - time_start.tv_sec;
time_delta.tv_nsec = time_stop.tv_nsec - time_start.tv_nsec;

printf("Computation cdr, time: %lds, %ldns\n", time_delta.tv_sec,
time_delta.tv_nsec);

delete [] crdCpp;
delete [] crdMatlab;
}

void calc_crd(struct crd_t* crd, int N, struct ephemerides_t eph, struct
mytime_t t)
{
    // Константы
    double mu = 3.986004418e14;
    double OmegaE = 7.2921151467e-5;

    // Большая полуось
    double A = pow(eph.sqrtA, 2);

    // Среднее движение
    double n0 = sqrt(mu/pow(A,3));
    double n = n0 + eph.Dn;

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

        double t_now = t.start + k * t.step;
        double t = t_now - eph.toe;

```

```

        if (t > 302400)
            t -= 604800;
        if (t < -302400)
            t += 604800;

        // Средняя аномалия
        double M = eph.M0 + n*t;
        double epsilon = 1e-6;

        // Решение уравнения Кеплера
        double E = solveKepler(M, M, eph.e, epsilon);

        // Истинная аномалия
        double nu = atan2( sqrt(1 - pow(eph.e,2)) * sin(E) , cos(E) -
eph.e );

        // Коэффициенты коррекции
        double cor_cos = cos(2*(eph.omega + nu));
        double cor_sin = sin(2*(eph.omega + nu));

        // Аргумент широты
        double u = eph.omega + nu + eph.Cuc*cor_cos + eph.Cus*cor_sin;

        // Радиус
        double r = A * (1 - eph.e * cos(E)) + eph.Crc*cor_cos +
eph.Crs*cor_sin;

        // Наклон
        double i = eph.i0 + eph.iDot * t + eph.Cic*cor_cos +
eph.Cis*cor_sin;

        // Долгота восходящего угла
        double lambda = eph.Omega0 + (eph.OmegaDot - OmegaE) * t -
OmegaE*eph.toe;

        // Положение на орбите
        double x = r * cos(u);
        double y = r * sin(u);

        // Координаты
        crd[k].X = x * cos(lambda) - y * cos(i) * sin(lambda);
        crd[k].Y = x * sin(lambda) + y * cos(i) * cos(lambda);
        crd[k].Z = y * sin(i);
    }
}

void write_crd(struct crd_t* crd, int N, char *fname)
{
    // Запись в файл (для матлаба)
    FILE *file;
    if ((file = fopen(fname, "wb")) == NULL) {
        printf("Cannot open file.\n");
        return;
    }

    for (int i = 0; i < N; i++) {
        fprintf(file, "%22.17e %22.17e %22.17e\n", crd[i].X, crd[i].Y,
crd[i].Z);
    }
    fclose(file);
}

void read_crd(struct crd_t* crd, int N, char *fname)
{

```

```

FILE *file;
if ((file = fopen(fname, "rb+")) == NULL) {
    printf("Cannot open file.\n");
    return;
}

for (int i = 0; i < N; i++) {
    fscanf(file, "%le %le %le\n", &crd[i].X, &crd[i].Y,
&crd[i].Z);
}
fclose(file);
}

void compare_crd(struct crd_t* crd1, struct crd_t* crd2, int N)
{
    struct crd_t crdMaxDelta = {0,0,0}, crdDelta;

    for (int i = 0; i < N; i++) {
        // Определение максимальной разницы по X
        crdDelta.X = fabs(crd1[i].X - crd2[i].X);
        if (crdDelta.X > crdMaxDelta.X)
            crdMaxDelta.X = crdDelta.X;

        // Определение максимальной разницы по Y
        crdDelta.Y = fabs(crd1[i].Y - crd2[i].Y);
        if (crdDelta.Y > crdMaxDelta.Y)
            crdMaxDelta.Y = crdDelta.Y;

        // Определение максимальной разницы по Z
        crdDelta.Z = fabs(crd1[i].Z - crd2[i].Z);
        if (crdDelta.Z > crdMaxDelta.Z)
            crdMaxDelta.Z = crdDelta.Z;
    }
    printf("Max delta X = %le\nMax delta Y = %le\nMax delta Z = %le\n",
crdMaxDelta.X, crdMaxDelta.Y, crdMaxDelta.Z);
}

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

```

kepler.h

```

#ifndef KEPLER_H
#define KEPLER_H

#include <math.h>

double solveKepler(double E, double M, double e, double epsilon);

int mult(int a, int b);

#endif /* #ifndef KEPLER_H */

```

kepler.cpp

```

#include <include/libbdssvpos/kepler.h>

double solveKepler(double E, double M, double e, double epsilon)
{
    double E_old = E + 1;

```

```

        while (fabs(E - E_old) > epsilon) {
            E_old = E;
            E = M + e * sin(E);
        }
        return E;
    }

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

```

tests.h

```

#ifndef TESTS_H
#define TESTS_H

#include "include/libbdssvpos/bdssvpos.h"
#include "include/libbdssvpos/kepler.h"

#include <time.h>
#include <stdio.h>

void test_solveKepler(int N);

#endif // TESTS_H

```

tests.cpp

```

#include "tests.h"

void test_solveKepler(int N)
{
    // Структуры содержащие время начала, конца и разницу
    struct timespec time_start, time_stop, time_delta;

    // Получение времени начала
    timespec_get(&time_start, TIME_UTC);
    int error_counter = 0;

    // Тестирование - N итераций решения ур. Кеплера
    for (int i = 0; i < N; i++) {
        double N_first = (double)(rand())/RAND_MAX;    // Случайное
        // число от 0 до 1
        double epsilon = 1e-6;
        double N_last = solveKepler(N_first, rand(), 0.5, epsilon);

        // Сравнение начального значения и рассчитанного
        if (fabs(N_last - N_first) < epsilon) {
            error_counter++;
        }
    }

    // Получение времени окончания
    timespec_get(&time_stop, TIME_UTC);

    // Расчет разницы времени
    time_delta.tv_sec = time_stop.tv_sec - time_start.tv_sec;
    time_delta.tv_nsec = time_stop.tv_nsec - time_start.tv_nsec;

    printf("Test solveKepler, %d iteration, time: %lds, %ldns, error
count: %d\n", N, time_delta.tv_sec, time_delta.tv_nsec, error_counter);
}

```