

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт: *ИРЭ*

Кафедра: *Радиотехнических систем*

Специальность:

*11.05.01 Радиоэлектронные системы и
комплексы*

ОТЧЕТ

по курсовой работе

Разработка модуля расчёта координат спутника Beidou

СТУДЕНТ

/ Хоанг Д.Д. /
(подпись) (Фамилия и инициалы)

Группа ЭР-15-16
(номер учебной группы)

Защита курсовой работы

(отлично, хорошо, удовлетворительно, неудовлетворительно,
зачтено, не зачтено)

/ Корогодин И.В. /
(подпись) (Фамилия и инициалы члена комиссии)

/ Шатилов А.Ю. /
(подпись) (Фамилия и инициалы члена комиссии)

**Москва
2021**

СОДЕРЖАНИЕ

ПОСТАНОВКА ЗАДАЧИ	3
ГЛАВА 1. ИСПОЛЬЗОВАНИЕ СТОРОННИХ СРЕДСТВ	5
1. Описание задания.....	5
2. Определение формы орбиты и положения спутника.....	5
3. Определение формы орбиты и положения спутника на ней с помощью сервиса CelesTrak.....	7
4. Расчет графика угла места собственного спутника от времени по данным Trimble GNSS Planning Online	8
5. Расчет диаграммы угла места и азимута спутника (SkyView, он же SkyPlot) по данным Trimble GNSS Planning Online	10
6. Формирование таблицы эфемерид собственного спутника	13
ГЛАВА 2. МОДЕЛИРОВАНИЕ	14
1. Задание	14
2. Реализация	14
ГЛАВА 3. РЕАЛИЗАЦИЯ	16
1. Задание	16
2. Результат	17
3. Профилирование	17
ЗАКЛЮЧЕНИЕ	20
ПРИЛОЖЕНИЕ	21
ПРИЛОЖЕНИЕ 1	21
ПРИЛОЖЕНИЕ 2	26
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	33

ПОСТАНОВКА ЗАДАЧИ

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

Исходными данными являются номер спутника ему соответствует спутник номером 29 системы Beidou и эфемериды, которые можно найти, зная номер НКА. Решение задачи предлагается разделить на 3 этапа, которые необходимо решать последовательно, так как результаты, полученные на текущем этапе, используются в последующем. Финальным продуктом является реализация программного модуля на C/C++.

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 проекта 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.

Конечная цель всего курсового проекта - получить библиотеку функций на «C++», позволяющую рассчитывать положение спутника Beidou по его эфемеридам.

ГЛАВА 1. ИСПОЛЬЗОВАНИЕ СТОРОННИХ СРЕДСТВ

1. Описание задания

На первом этапе подготовим вспомогательные данные для разработки: эфемериды и оценки положения спутника от сторонних сервисов (чтобы было с чем сравниваться на след. этапах).

На крыше корпуса Е МЭИ установлена трехдиапазонная антенна Narxон НХ-CSX601А. 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.

Эти приемники осуществляют первичную обработку сигналов Beidou B1I, выдавая по интерфейсам соответствующие потоки данных - наблюдения псевдодальностей и эфемериды спутников. Данные от приемника Clonicus, записанные вечером 16 февраля 2021 года.

2. Определение формы орбиты и положения спутника.

Используя сервис «Информационно-аналитического центра координатно-временного и навигационного обеспечения» определим номер НОРАД [1] и сравним его с номером из «Википедии» [2]:

C16	43539	IGSO-7	BDS-2	10.07.18	967	Используется по ЦН
C19	43001	MEO-1	BDS-3	05.11.17	1214	Используется по ЦН
C20	43002	MEO-2	BDS-3	05.11.17	1214	Используется по ЦН
C21	43208	MEO-3	BDS-3	12.02.18	1115	Используется по ЦН
C22	43207	MEO-4	BDS-3	12.02.18	1115	Используется по ЦН
C23	43581	MEO-5	BDS-3	29.07.18	948	Используется по ЦН
C24	43582	MEO-6	BDS-3	29.07.18	948	Используется по ЦН
C25	43603	MEO-11	BDS-3	25.08.18	921	Используется по ЦН
C26	43602	MEO-12	BDS-3	25.08.18	921	Используется по ЦН
C27	43107	MEO-7	BDS-3	12.01.18	1146	Используется по ЦН
C28	43108	MEO-8	BDS-3	12.01.18	1146	Используется по ЦН
C29	43245	MEO-9	BDS-3	30.03.18	1069	Используется по ЦН
C30	43246	MEO-10	BDS-3	30.03.18	1069	Используется по ЦН
C32	43622	MEO-13	BDS-3	19.09.18	896	Используется по ЦН
C33	43623	MEO-14	BDS-3	19.09.18	896	Используется по ЦН
C34	43648	MEO-15	BDS-3	15.10.18	870	Используется по ЦН
C35	43647	MEO-16	BDS-3	15.10.18	870	Используется по ЦН
C36	43706	MEO-17	BDS-3	19.11.18	835	Используется по ЦН
C37	43707	MEO-18	BDS-3	19.11.18	835	Используется по ЦН
C38	44204	IGSO-1	BDS-3	20.04.19	683	Используется по ЦН
C39	44337	IGSO-2	BDS-3	25.06.19	617	Используется по ЦН

Рисунок 1 – Состояние космических аппаратов BeiDou на 03.03.21

16	Компас G6	C02	25.10.2012 15:33	CZ-3C	2012-059A	38953	ГСО, 80° в. д.	действующий	
17	Бэйдоу-3S IGSO-1	C31	30.03.2015 13:52	CZ-3C/YZ-1	2015-019A	40549	Геосинхронная, наклонение 55°	на испытании	
18	Бэйдоу-3S M1	C58	25.07.2015 12:29	CZ-3B/YZ-1	2015-037A	40748	ССО, ~21 500 км	на испытании	Бэйдоу-3S
19	Бэйдоу-3S M2	C57			2015-037B	40749	ССО, ~21 500 км	на испытании	
20	Бэйдоу-3S IGSO-2	C56	29.09.2015 23:13	CZ-3B/E	2015-053A	40938	Геосинхронная, наклонение 55°	на испытании	
21	Бэйдоу-3S M3	N/A	01.02.2016 07:29	CZ-3C/YZ-1	2016-006A	41315	ССО, ~21 500 км	на испытании	
22	Бэйдоу-2 IGSO-6	C13	29.03.2016 20:11	CZ-3A	2016-021A	41434	Геосинхронная, накл. 55°;	действующий	Бэйдоу-2
23	Бэйдоу-2 G7	C03	12.06.2016 15:30	CZ-3C	2016-037A	41586	ГСО, 144° в. д.	действующий	
24	Бэйдоу-3 M1	C19	05.11.2017 11:44	CZ-3B/YZ-1	2017-069A	43001	ССО, ~21 500 км	действующий	Бэйдоу-3
25	Бэйдоу-3 M2	C20			2017-069B	43002	ССО, ~21 500 км	действующий	
26	Бэйдоу-3 M3	C27	11.01.2018 23:18	CZ-3B/YZ-1	2018-003A	43107	ССО, ~21 500 км	действующий	
27	Бэйдоу-3 M4	C28			2018-003B	43108	ССО, ~21 500 км	действующий	
28	Бэйдоу-3 M5	C22	12.02.2018 05:10	CZ-3B/YZ-1	2018-018A	43207	ССО, ~21 500 км	действующий	
29	Бэйдоу-3 M6	C21			2018-018B	43208	ССО, ~21 500 км	действующий	
30	Бэйдоу-3 M7	C29	29.03.2018 17:50	CZ-3B/YZ-1	2018-029A	43245	ССО, ~21 500 км	действующий	
31	Бэйдоу-3 M8	C30			2018-029B	43246	ССО, ~21 500 км	действующий	
32	Бэйдоу-2 IGSO-7	C16	09.07.2018 20:58	CZ-3A	2018-057A	43539	Геосинхронная, накл. 55°;	действующий	Бэйдоу-2
33	Бэйдоу-3 M9	C23	29.07.2018 01:48	CZ-3B/YZ-1	2018-062A	43581	ССО, ~21 500 км	действующий	Бэйдоу-3
34	Бэйдоу-3 M10	C24			2018-062B	43582	ССО, ~21 500 км	действующий	
35	Бэйдоу-3 M11	C26	24.08.2018, 23:37	CZ-3B/YZ-1	2018-067A	43602	ССО, ~21 500 км	действующий	
36	Бэйдоу-3 M12	C25			2018-067B	43603	ССО, ~21 500 км	действующий	
37	Бэйдоу-3 M13	C32	19.09.2018, 14:07	CZ-3B/YZ-1	2018-072A	43622	ССО, ~21 500 км	действующий	
38	Бэйдоу-3 M14	C33			2018-072B	43623	ССО, ~21 500 км	действующий	
39	Бэйдоу-3 M15	C35			2018-078A	43647	ССО, ~21 500 км	действующий	

Рисунок 2 – Состояние системы BeiDou с сайта Википедия

Интересующий нас спутник "BEIDOU-3 M7":

Таблица 1 – Сведения о спутнике

Спутник	PRN	ID	SCN
BEIDOU-3 M7	C29	2018-029A	43245

3. Определение формы орбиты и положения спутника на ней с помощью сервиса CelesTrak

Введем название спутника и сверим его по номеру NSSDC ID и НОРАД (SCN).

Значения совпадают, данный спутник существует проведем моделирование на момент времени 15:00, 16 февраля 2021, так как на данном сервисе отсчет времени происходит по UTC(0):

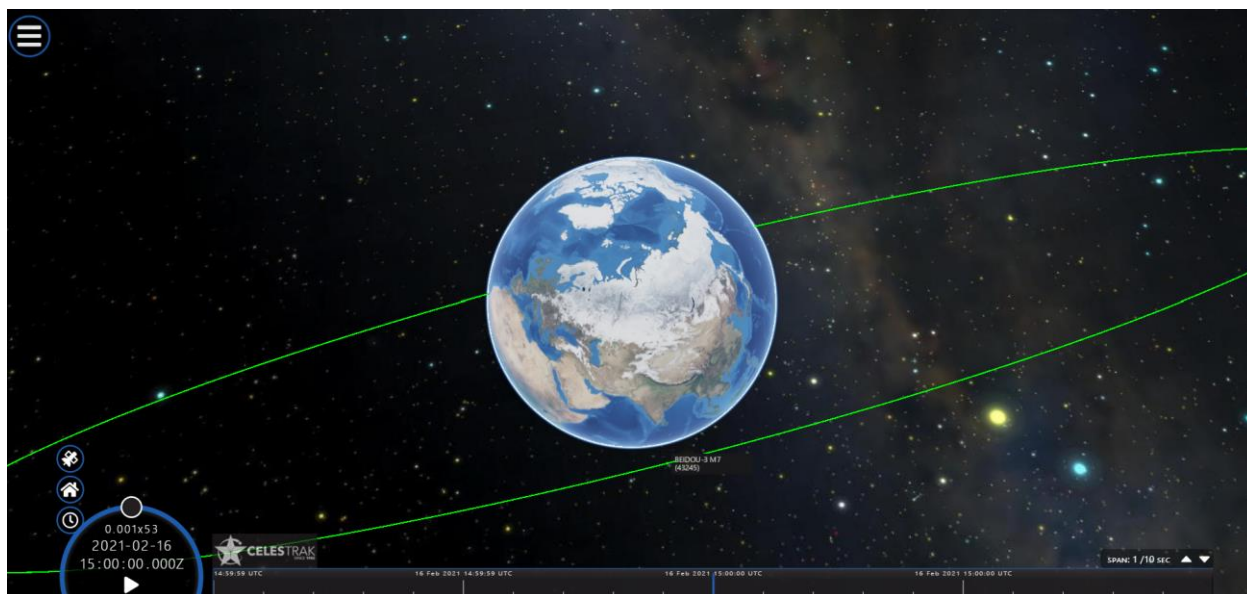


Рисунок 3 – Моделирование с помощью сервиса CelesTrak

4. Расчет графика угла места собственного спутника от времени по данным Trimble GNSS Planning Online

Настроим для моделирования GNSS Planning Online, координаты установим в соответствии с расположением антенны – и они будут соответствовать значению корпуса Е МЭИ, также начальное время будет соответствовать 18:00, временной пояс будет равен +3 (UTC +3) на всем этапе моделирования в сервисе GNSS Planning Online.

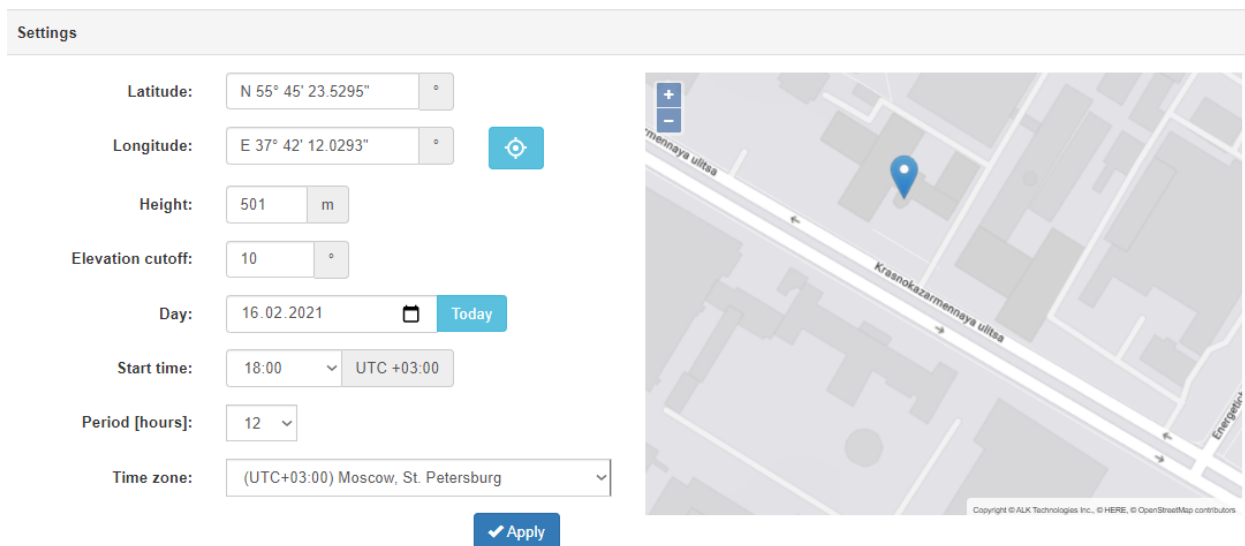


Рисунок 4 – Моделирование с помощью сервиса Trimble GNSS Planning

Далее ограничим количество отображаемых спутников и оставим в моделирование только нужны нам спутник – C24:

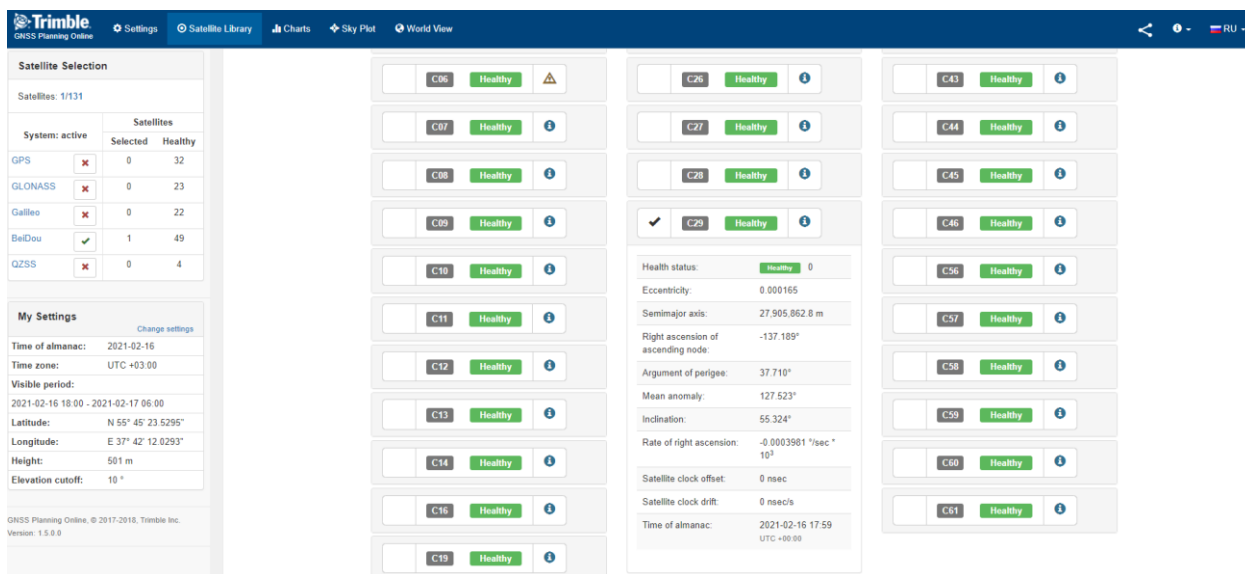


Рисунок 5 – Моделирование с помощью сервиса Trimble GNSS Planning

Получим график расчета угла места собственного спутника от времени:

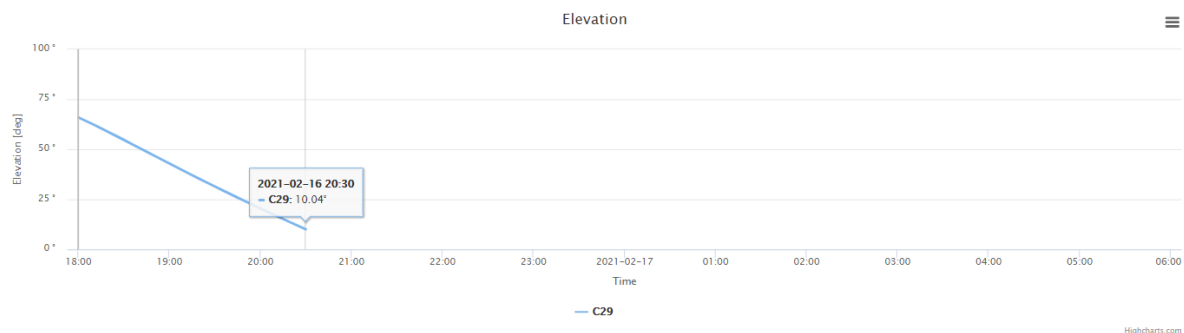


Рисунок 6 – График угла места собственного спутника от времени

По графику видно, что на указанном в задании интервале с 18:00 – 06:00, спутник был в области видимости один раз с 18:00 до 20:30.

5. Расчет диаграммы угла места и азимута спутника (SkyView, он же SkyPlot) по данным Trimble GNSS Planning Online

Так как сервис для определения Sky Plot используется тот же - Trimble GNSS Planning Online, то настройки оставим прежние, и проведем моделирование Sky Plot во временном интервале 18:00-06:00 и зафиксируем положение спутника на небосводе в критических точках, то есть когда он находился в области видимости - в 18:00 и 20:30.

Тогда получим 2 графика моделирования:

- 16 февраля 2021 в 18:00:

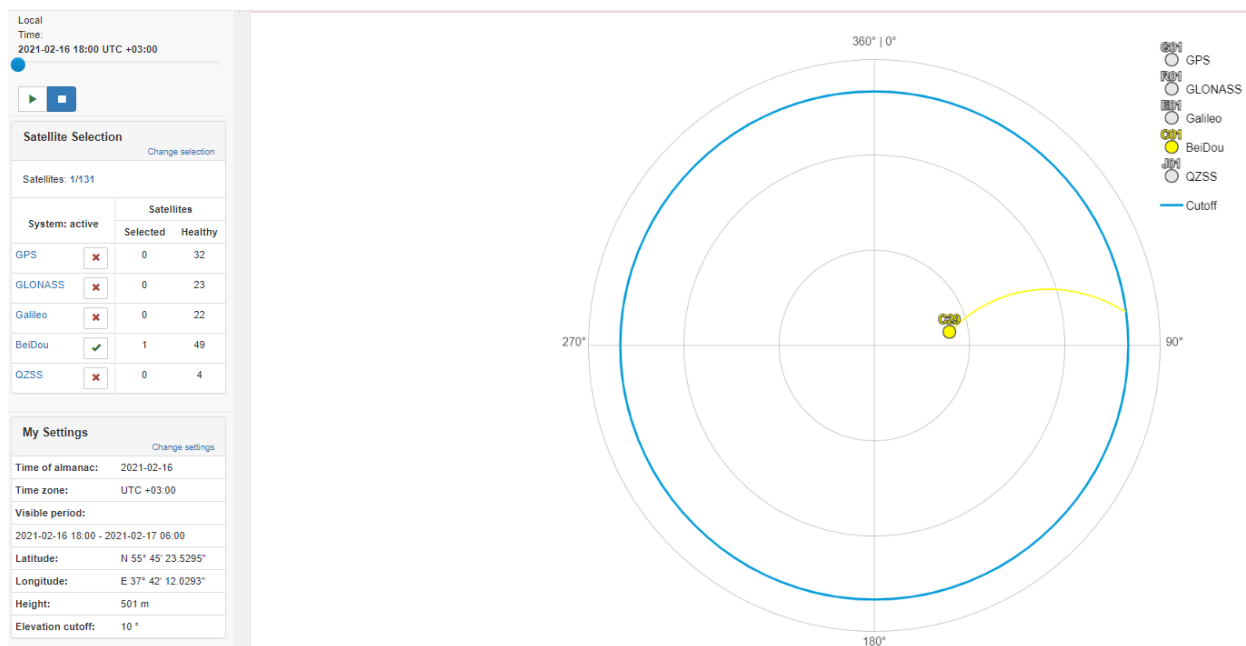


Рисунок 7 – Моделирование с помощью сервиса Trimble GNSS Planning

- 16 февраля 2021 в 20:30:

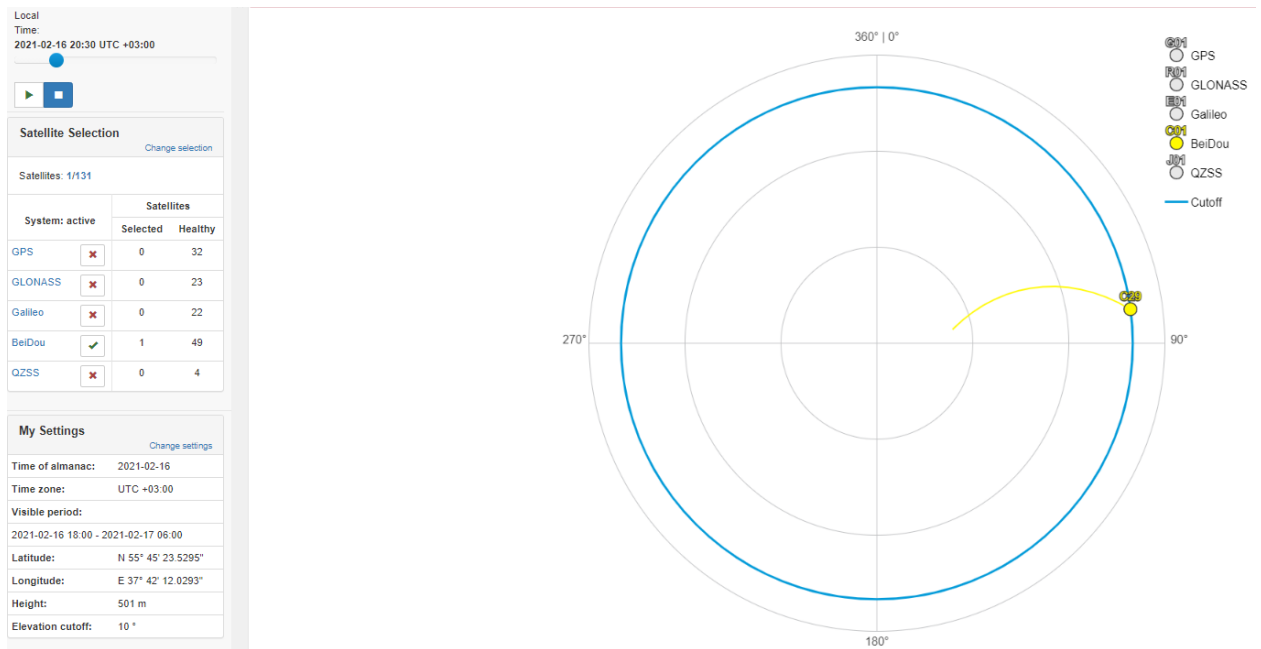


Рисунок 8 – Моделирование с помощью сервиса Trimble GNSS Planning

6. Формирование таблицы эфемерид собственного спутника

Таблица 2 – Значения эфемерид спутника

Параметр	Значение	Размерность
SatNum	29	-
toe	219600000.000	мс
Crs	4.8437500000000000e-01	м
Dn	3.38049792152073092e-12	рад/с
M0	-2.04583603053961255e-01	рад
Cuc	7.40401446819305420e-08	рад
e	1.57959060743451118e-04	-
Cus	1.13863497972488403e-05	рад
sqrtA	5.28262396240234375e+03	м ^{1/2}
Cic	3.86498868465423584e-08	рад
Omega0	-2.39429712929735228e+00	рад
Cis	3.07336449623107910e-08	рад
i0	9.65601789007491829e-01	рад
Crc	1.32890625000000000e+02	м
omega	6.54981262210034165e-01	рад
OmegaDot	-6.63384775495807734e-12	рад/мс
iDot	5.14307137242361949e-14	рад/мс
Tgd	1.0220000000000000e+06	мс
toc	2.1960000000000000e+08	мс
af2	0.0000000000000000e+00	мс/мс ²
af1	5.06794606280891458e-12	мс/мс
af0	3.53220045566558838e-01	мс
URA	0	-
IODE	257	-
IODC	1	-
codeL2	0	-
L2P	0	-
WN	789	-

ГЛАВА 2. МОДЕЛИРОВАНИЕ

1. Задание

Необходимо реализовать функцию расчета положения спутника Beidou на заданный момент по шкале времени UTC на языке Matlab или Python. Значения, полученные на предыдущем этапе, нужны нам в качестве эфемерид для моделирования. Построить трехмерные графики множества положений спутника Beidou с системным номером, соответствующим номеру студента по списку. 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.

2. Реализация

Для выполнения данного этапа воспользуемся пакетом математического моделирования Matlab. Само моделирование проводилось с помощью алгоритма с сайта Navipedia.

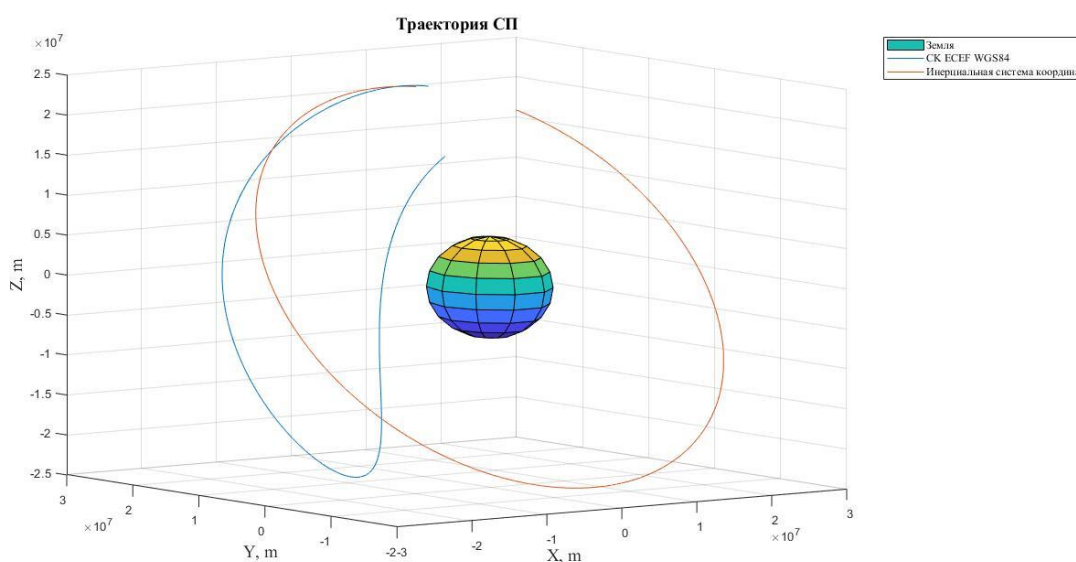


Рисунок 9 — Траектория движения спутника Beidou PRN №29

Переход из системы ECEF в систему ECI был осуществлен также согласно алгоритму, из ИКД. По полученному графику видно, что за установленный интервал времени спутник не успевает полностью пройти всю свою траекторию.

Получим траекторию в полярной системе координат и сравним ее с результатом из Trimble GNSS Planning Online, изображенным на рисунках 7 и 8.

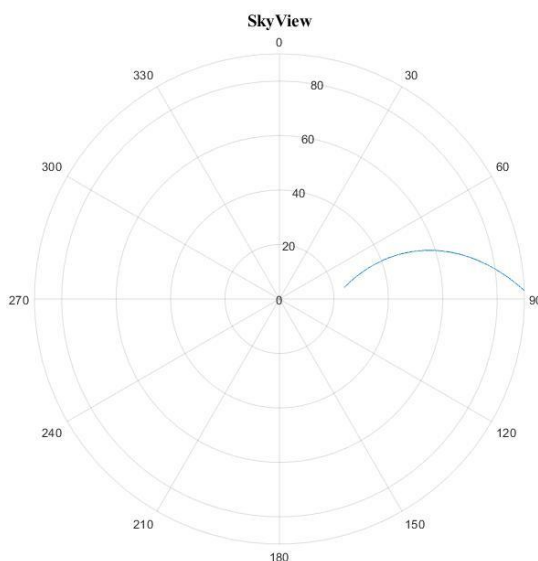


Рисунок 10 — SkyView спутника Beidou, полученный в результате моделирования

При сравнении рисунков, полученных на 2 этапе, с рисунками, полученными на 1 этапе, видно, что они практически совпадают. Однако, имеется погрешность, связанная с параметрами эфемерид, которые для простоты приняты за постоянные на промежутке моделирования. Листинг программы в приложении.

ГЛАВА 3. РЕАЛИЗАЦИЯ

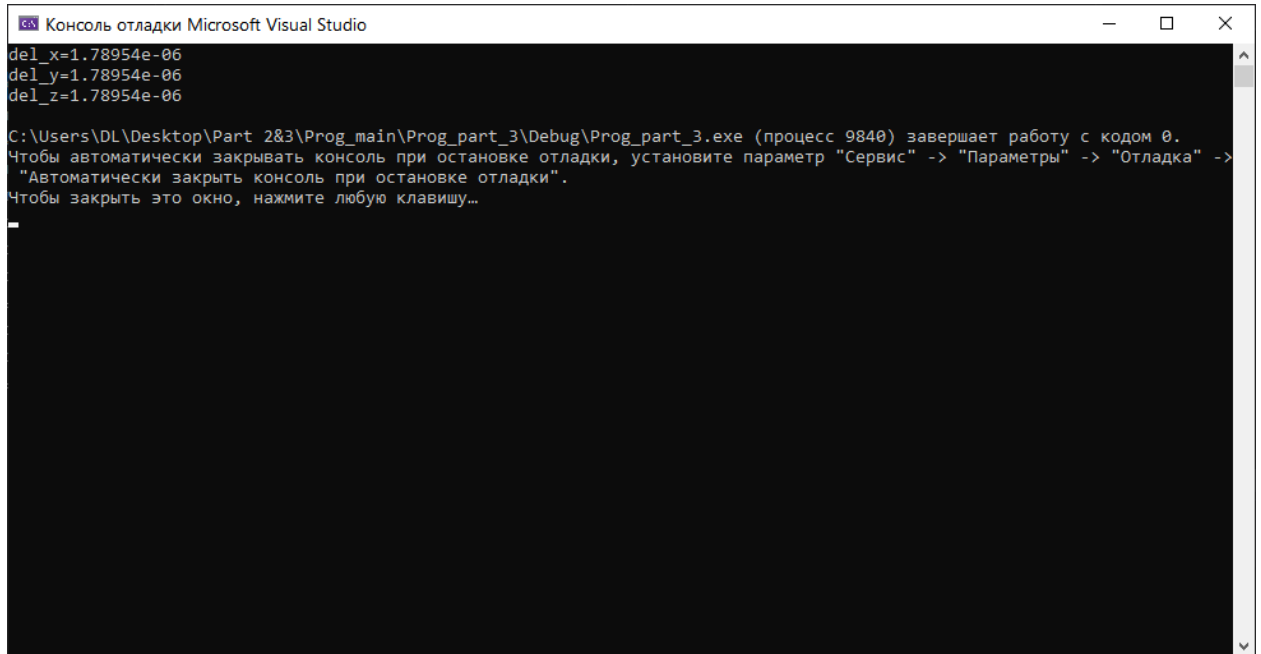
1. Задание

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

Функция расчета положения спутника в Matlab/Python относительно проста, т.к. доступны библиотеки линейной алгебры и решения уравнений. Но при разработке встраиваемого ПО приходится сохранять лицензионную частоту, минимизировать вычислительную нагрузку и затраты памяти. Поэтому отобразить модель из Matlab/Python в прошивку приемника дословно, как правило, не получается. В рассматриваемом примере потребуется, как минимум, выполнить свою реализацию решения трансцендентного уравнения.

Реализация программы будет в программе Microsoft Visual Studio2019, на языке C++.

2. Результат



```
Консоль отладки Microsoft Visual Studio
del_x=1.78954e-06
del_y=1.78954e-06
del_z=1.78954e-06
C:\Users\DL\Desktop\Part 2&3\Prog_main\Prog_part_3\Debug\Prog_part_3.exe (процесс 9840) завершает работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" ->
"Автоматически закрыть консоль при остановке отладки".
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рисунок 11 – Результат работы программы

Имеется некоторое расхождение, вызванное использованием представления значений разной разрядности в Matlab и Visual Studio на языке C++.

3. Профилирование

Чтобы проверить программу на жизнеспособность, проведем профилирование, т.е. соберем данные о работе приложения.

Для начала, необходимо получить отчет отладки, на котором можно обнаружить, локализовать и устранить ошибки. В случае, если ошибок нет, программа считается отлаженной.

```
Показать выходные данные из: Отладка
"Prog_part_3.exe" (Win32). Загружено "C:\Users\DL\Desktop\Part_283\Prog_main\Prog_part_3\x64\Debug\Prog_part_3.exe". Символы загружены.
"Prog_part_3.exe" (Win32). Загружено "C:\Windows\System32\ntdll.dll".
"Prog_part_3.exe" (Win32). Загружено "C:\Windows\System32\kernel32.dll".
"Prog_part_3.exe" (Win32). Загружено "C:\Windows\System32\KernelBase.dll".
"Prog_part_3.exe" (Win32). Загружено "C:\Windows\System32\msvcrt.dll".
"Prog_part_3.exe" (Win32). Загружено "C:\Windows\System32\user32.dll".
"Prog_part_3.exe" (Win32). Загружено "C:\Windows\System32\GDI32.dll".
"Prog_part_3.exe" (Win32). Загружено "C:\Windows\System32\SHELL32.dll".
"Prog_part_3.exe" (Win32). Загружено "C:\Windows\System32\ole32.dll".
"Prog_part_3.exe" (Win32). Загружено "C:\Windows\System32\RPCRT4.dll".
Поток 0x4a4 завершился с кодом 0 (0x0).
"Prog_part_3.exe" (Win32). Загружено "C:\Windows\System32\kernel.appcore.dll".
"Prog_part_3.exe" (Win32). Загружено "C:\Windows\System32\msvcrt.dll".
"Prog_part_3.exe" (Win32). Загружено "C:\Windows\System32\RPCRT4.dll".
Поток 0x158 завершился с кодом 0 (0x0).
Поток 0x2f00 завершился с кодом 0 (0x0).
Программа "[14996] Prog_part_3.exe" завершилась с кодом 0 (0x0).
```

Рисунок 13 – Отчет отладки

Проверим загруженность ЦП в процессе работы:

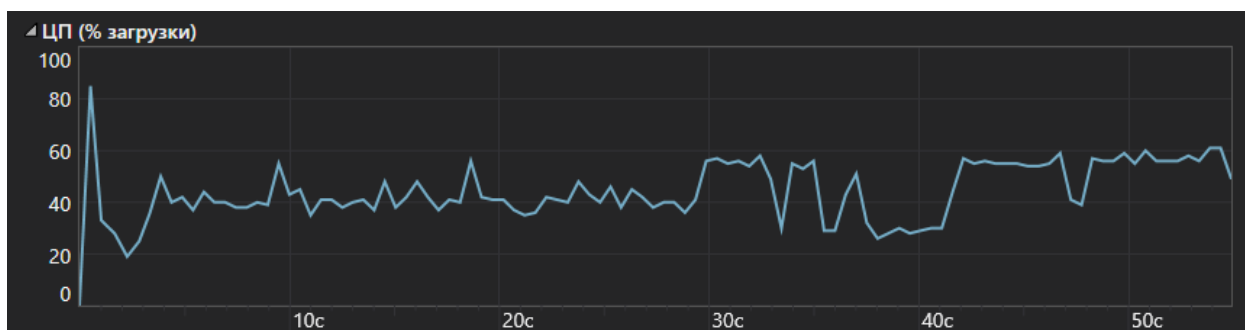


Рисунок 14 – Загруженность ЦП

Как видно из рисунка, программа не загружает процессор до 100% и имеет более-менее равномерный график нагрузки, что позволяет не подбирать на ЦП более громоздкую систему охлаждения.

Определим наличие утечек памяти:

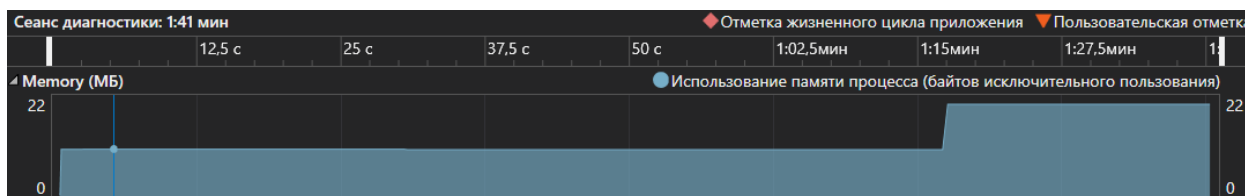







Рисунок 15 – Использование памяти

Утечки памяти отсутствуют, критических ошибок в коде программы нет.

Определим наиболее длинную по срокам последовательную цепочку работ, т.е. критический путь и наиболее затратные по времени функции:

Критический путь		
Имя функции	% затраченного инклюзивного времени	% затраченного эксклюзивного времени
 main	88.26	0.26
 coord	87.64	2.02
 cos	45.59	26.88
 kepler	18.77	1.00
 sin	16.60	7.50

Связанные представления: [Дерево вызовов](#) [Функции](#)

Рисунок 16 – Критический путь

Функции, выполняющие максимальную индивидуальную работу	
Имя	Эксклюзивное время %
cos	26.88
__CheckForDebuggerJustMyCode	22.21
sin	12.55
invoke_main	11.73
cosl	8.48

Рисунок 17 – Функции, выполняющие максимальную работу

Теоретический эмпирический и практический расчет наиболее трудозатратных операций в пределах разумного, значит программа правильно выполняет расчет.

В преставлении “Дерево вызовов” отображаются пути выполнения функции, пересеченные в профилируемом приложении. Корнем дерева является точка входа в приложение или компонент. В каждом узле функции перечислены все вызванные в ней функции и содержатся данные по производительности, связанные с этими вызовами функций:

Текущее представление: Дерево вызовов						
Имя функции	Число вызовов	% затраченного инкл...	% затраченного экск...	Среднее затраченное...	Среднее затраченное...	Имя модуля
Prog_part_3.exe	0	100,00%	0,00%	0,00	0,00	
mainCRTStart...	1	100,00%	0,00%	7 441 782 062,00	48 348,00	Prog_part_3.exe
_schr_com...	1	100,00%	0,00%	7 441 733 714,00	34 657,00	Prog_part_3.exe
_schr_co...	1	100,00%	0,00%	7 441 698 473,00	74 399,00	Prog_part_3.exe
invoke...	1	99,99%	11,73%	7 440 794 579,00	872 722 530,00	Prog_part_3.exe
main	1	88,26%	0,26%	6 568 072 049,00	19 512 652,00	Prog_part_3.exe
co...	57 590	87,64%	2,02%	113 252,00	2 609,00	Prog_part_3.exe
at...	57 590	3,45%	0,33%	4 462,00	423,00	Prog_part_3.exe
...	691 079	45,59%	26,88%	4 909,00	2 894,00	Prog_part_3.exe
ke...	57 590	18,77%	1,00%	24 255,00	1 298,00	Prog_part_3.exe
p...	57 590	0,33%	0,25%	420,00	322,00	Prog_part_3.exe
sin	460 719	16,60%	7,50%	2 681,00	1 212,00	Prog_part_3.exe
sqrt	57 590	0,81%	0,34%	1 042,00	433,00	Prog_part_3.exe
oper...	3	0,35%	0,00%	8 695 610,00	2 502,00	Prog_part_3.exe
pow	1	0,00%	0,00%	278 992,00	22 892,00	Prog_part_3.exe
pow	1	0,00%	0,00%	885,00	718,00	Prog_part_3.exe
pow...	1	0,00%	0,00%	3 054,00	572,00	Prog_part_3.exe
_Ch...	1	0,00%	0,00%	164,00	164,00	Prog_part_3.exe
post_pgo...	1	0,00%	0,00%	82 083,00	231,00	Prog_part_3.exe
pre_cpp...	1	0,00%	0,00%	101 358,00	101 358,00	Prog_part_3.exe
pre_c_ini...	1	0,01%	0,00%	493 946,00	50 554,00	Prog_part_3.exe
std::dyn...	1	0,00%	0,00%	3 955,00	751,00	Prog_part_3.exe
_schr_ac...	1	0,00%	0,00%	865,00	581,00	Prog_part_3.exe
_schr_ini...	1	0,00%	0,00%	146 705,00	68 389,00	Prog_part_3.exe
_schr_rel...	1	0,00%	0,00%	583,00	372,00	Prog_part_3.exe
_security_i...	1	0,00%	0,00%	584,00	584,00	Prog_part_3.exe

Рисунок 18 – Дерево вызовов

ЗАКЛЮЧЕНИЕ

В курсовом проекте была написана программа расчета положения спутника Beidou, на заданные координаты и время. Получены навыки работы на языке C++ в программной среде Microsoft Visual Studio 2019, работе с ИКД ГНСС, альманахом системы. Этапы курсового проекта позволяют пройти от идеи о реализации до полноценной программы, которую можно зашить в железо. Данная работа обобщает знания в пределах учебного курса и позволяет в дальнейшем лучше понимать, как разрабатываются подобные системы.

ПРИЛОЖЕНИЕ

ПРИЛОЖЕНИЕ 1

Файл: "Code.m"

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

%% Эфемериды
SatNum = 29;
toe = 219600000.000 * 10^-3;
Crs = 4.8437500000000000e-01;
Dn = 3.38049792152073092e-12;
M0 = -2.04583603053961255e-01;
Cuc = 7.40401446819305420e-08 ;
e = 1.57959060743451118e-04;
Cus = 1.13863497972488403e-05 ;
sqrtA = 5.28262396240234375e+03 ;
Cic = 3.86498868465423584e-08 ;
Omega0 = -2.39429712929735228e+00 ;
Cis = 3.07336449623107910e-08 ;
i0 = 9.65601789007491829e-01 ;
Crc = 1.32890625000000000e+02 ;
omega = 6.54981262210034165e-01 ;
OmegaDot = -6.63384775495807734e-12;
iDot = 5.14307137242361949e-14 ;
Tgd = 1.0220000000000000e+06 ;
toc = 2.1960000000000000e+08 ;
af2 = 0.0000000000000000e+00 ;
af1 = 5.06794606280891458e-12 ;
af0 = 3.53220045566558838e-01 ;
URA = 0;
IODE = 257;
IODC = 1;
codeL2 = 0;
L2P = 0;
WN = 789;

%% Константы
mu = 3.986004418e14; % гравитационная постоянная
```

```

omega_e = 7.2921151467e-5; % скорость вращения

%% Временной промежуток
b_time = (24*2+18-3)*60*60; % время начала 18:00 по МСК 16 февраля
e_time = (24*3+6-3)*60*60; % время окончания 6:00 по МСК 17 февраля
% Длина временного промежутка
t_Ar = b_time:1:e_time;
% Большая полуось
A = sqrt(A^2);
% Среднее движение
n0 = sqrt(mu/A^3);
n = n0+Dn;

for k = 1:length(t_Ar)
    t(k) = t_Ar(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;
    eps = 1e-6;

    while abs(E(k)-E_old(k)) > eps
        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);
    % Коэффициент коррекции
    cos_corr(k) = cos(2*(omega+nu(k)));
    sin_corr(k) = sin(2*(omega+nu(k)));
    % Аргумент широты
    u(k) = omega+nu(k)+Cuc*cos_corr(k)+Cus*sin_corr(k);
    % Радиус

```

```

r(k) = A*(1-e*cos(E(k)))+Crc*cos_corr(k)+Crs*sin_corr(k);
% Наклон
i(k) = i0+iDot*t(k)+Cic*cos_corr(k)+Cis*sin_corr(k);
% Долгота восходящего угла
lambd(k) = Omega0+(OmegaDot-omega_e)*t(k)-omega_e*toe;
% Положение на орбите
x = r(k)*cos(u(k));
y = r(k)*sin(u(k));
% Координаты
X0(k) = x*cos(lambd(k))-y*cos(i(k))*sin(lambd(k));
Y0(k) = x*sin(lambd(k))+y*cos(i(k))*cos(lambd(k));
Z0(k) = y*sin(i(k));

X(k) = X0(k)*cos(lambd(k))+Y0(k)*sin(lambd(k));
Y(k) = -X0(k)*sin(lambd(k))+Y0(k)*cos(lambd(k));
Z(k) = Z0(k);
end

%% Из HKA в WGS84
prb = 1e-9;
mos = 1e-3/206264.8;

WGS_84_m = [-3*prb -353*mos -4*mos;
353*mos -3*prb 19*mos;
4*mos -19*mos -3*prb];

crd_WGS_84 = [X0; Y0; Z0];

for i = 1:length(crd_WGS_84(1,:))
    crd_WGS_84(:,i) = crd_WGS_84(:,i) + WGS_84_m * crd_WGS_84(:,i) +
[0.07; -0; -0.77];
end

crd_WGS_84 = crd_WGS_84.';

%% Построение графиков
Radius_Z = 6371e3;
[XE,YE,ZE] = sphere(10);

figure
surf(XE*Radius_Z,YE*Radius_Z,ZE*Radius_Z)
hold on

```

```

grid on
plot3(crd_WGS_84(:,1), crd_WGS_84(:,2), crd_WGS_84(:,3))
plot3(X, Y, Z)
title('Траектория СП', 'FontName', 'Times New Roman', 'FontSize',14)
xlabel('X, м', 'FontName', 'Times New Roman', 'FontSize',14)
ylabel('Y, м', 'FontName', 'Times New Roman', 'FontSize',14)
zlabel('Z, м', 'FontName', 'Times New Roman', 'FontSize',14)
hold off
lgd = legend('Земля','СК ECEF WGS84','Инерциальная система координат');
lgd.FontName = 'Times New Roman';
%% Координаты в системе WGS84
N_gr = 55;
N_min = 45;
N_sec = 23.8178;
N = N_gr*pi/180+N_min/3437.747+N_sec/206264.8;
E_gr = 37;
E_min = 42;
E_sec = 12.2608;
E = E_gr*pi/180+E_min/3437.747+E_sec/206264.8;
H = 500; % Приблизительное значение высоты расположения антенны на
башне
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,wgs84Elli
    psoid,'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;
end

```



```

        r(i) = NaN;
        phi(i) = NaN;
    end
end

%% График
%skyplot
figure
ax = polaraxes;
polarplot(ax,phi,teta*180/pi)
ax.ThetaDir = 'clockwise';
ax.ThetaZeroLocation = 'top';
title('SkyView', 'FontName', 'Times New Roman', 'FontSize',14)

%% Преобразование координат из WGS84 в ECEF
function xyz = llh2xyz(llh)

phi = llh(1); % широта в радианах
lambda = llh(2); % долгота в радианах
h = llh(3); % высота над уровнем моря в метрах

a = 6378137.0000; % полуось земли в метрах
b = 6356752.3142; % полуось земли в метрах
e = sqrt(1-(b/a).^2);

sinphi = sin(phi);
cosphi = cos(phi);
coslam = cos(lambda);
sinlam = sin(lambda);
tan2phi = (tan(phi))^2;
tmp = 1-e*e;
tmpden = sqrt(1+tmp*tan2phi);

x = (a*coslam)/tmpden+h*coslam*cosphi;

y = (a*sinlam)/tmpden+h*sinlam*cosphi;

tmp2 = sqrt(1-e*e*sinphi*sinphi);
z = (a*tmp*sinphi)/tmp2+h*sinphi;

xyz(1) = x;
xyz(2) = y;

```

```
xyz(3) = z;  
end
```

ПРИЛОЖЕНИЕ 2

Листинг программы на C++

Prog_part_3.cpp

```
#include <iostream>  
#include <fstream>  
#include <cmath>  
#include "kepler.h"  
#include "coord.h"  
  
using namespace std;  
int main()  
{  
    long double cord[3];  
    long double e = 1.57959060743451118e-4;  
    long double mu = 3.98600442E+14;  
    long double Omega_e = 7.2921151467E-5;  
    long double toe = 219600000e-3;  
    long double A = pow(5.28262396240234375e+3, 2);  
    long double Dn = 3.38049792152073092e-12;  
    long double n0 = pow(mu / (pow(A, 3)), 0.5);  
    long double n = n0 + Dn;  
    long double Omega0 = -2.39429712929735228;  
    long double omega = 6.54981262210034165e-1;  
    long double OmegaDot = -6.63384775495807734e-12;  
    long double M0 = -2.04583603053961255e-1;  
    long double i0 = 9.65601789007491829e-1;  
    long double IDOT = 5.14307137242361949e-14;  
    long double Crs = 4.84375000000000000e-1;  
    long double Cuc = 7.40401446819305420e-8;
```

```

long double Cus = 1.13863497972488403e-5;
long double Cic = 3.86498868465423584e-8;
long double Cis = 3.07336449623107910e-8;
long double Crc = 1.328906250000000000e+2;
long double t_start = ((24 * 2) + 15) * 3600;
long double t_finish = ((24 * 3) + 3) * 3600;
int N_max = 432000;
long double* x = new long double[N_max];
long double* y = new long double[N_max];
long double* z = new long double[N_max];

    int k = 0;
    for (long double i = t_start; i < t_finish; i += 0.1)
    {
        coord(e, mu, Omega_e, toe,
            A, n0, i0, Omega0, omega, M0,
            Dn, n, OmegaDot, IDOT, Crs, Cuc, Cus,
            Cic, Cis, Crc, i, cord);
        x[k] = cord[0];
        y[k] = cord[1];
        z[k] = cord[2];
        k++;
    }

    ofstream f;
    f.open("Znach.txt");
    k = 0;
    for (long double i = t_start; i < t_finish; i += 0.1)
    {
        f << "k = " << k << "\t" << "t = " << i << "\t" << x[k] << "\t" <<
y[k] << "\t" << z[k] << endl;
        k++;
    }
    f.close();
long double* x_e2 = new long double[N_max];
long double* y_e2 = new long double[N_max];
long double* z_e2 = new long double[N_max];
int i = 0;

ifstream fin("cord.txt");
if (!fin.is_open())
    cout << "file dont open" << endl;
else

```

```

{
    cout << "file open" << endl;
    while (fin >> x_e2[i] >> y_e2[i] >> z_e2[i])
    {
        i++;
    }
    fin.close();
}

double d_x = 0;
double d_y = 0;
double d_z = 0;
double dxmax = 0;
double dymax = 0;
double dzmax = 0;
long double* xx_e2 = new long double[N_max];
long double* yy_e2 = new long double[N_max];
long double* zz_e2 = new long double[N_max];
for (int i = 0; i < N_max; i++)
{
    d_x += (fabs(x[i] - x_e2[i])); //сравнение значений матлаба и си
    d_y += (fabs(y[i] - y_e2[i]));
    d_z += (fabs(z[i] - z_e2[i]));
    xx_e2[i] = d_x;
    yy_e2[i] = d_y;
    zz_e2[i] = d_z;
}
double del_x = d_x / N_max;
double del_y = d_y / N_max;
double del_z = d_z / N_max;

printf("x = %9.9f \t y = %9.9f \t z = %9.9f \t ", x[0], y[0], z[0]);
printf("x = %9.9f \t y = %9.9f \t z = %9.9f \t ", x[1], y[1], z[1]);
printf("x = %9.9f \t y = %9.9f \t z = %9.9f \t ", x[2], y[2], z[2]);
cout << "del_x=" << del_x << endl;
cout << "del_y=" << del_y << endl;
cout << "del_z=" << del_z << endl;
delete[] x;
delete[] y;
delete[] z;
return 0;

```

coord.cpp

```
#include <math.h>
#include "coord.h"
#include "kepler.h"
#include <iostream>
using namespace std;
void coord(
    long double e,
    long double mu,
    long double Omega_e,
    long double toe,
    long double A,
    long double n0,
    long double i0,
    long double Omega0,
    long double omega,
    long double M0,
    long double Dn,
    long double n,
    long double OmegaDot,
    long double IDOT,
    long double Crs,
    long double Cuc,
    long double Cus,
    long double Cic,
    long double Cis,
    long double Crc,
    long double t,

    long double* coord)
{
    long double tk,
        Mk,
        E_k,
        nu_k,
        Phi_k,
        del_u_k,
        del_r_k,
        del_i_k,
        u_k,
        r_k,
        i_k,
```

```

        x_k,
        y_k,
        Omega_k,
        x_k1,
        y_k1,
        z_k1;
tk = t - toe;
if (tk > 302400)
{
    tk = 604800 - tk;
}
else if (tk < -302400)
{
    tk = 604800 + tk;
}
Mk = M0 + n * tk;
E_k = kepler(e, Mk);
nu_k = atan2((sqrt(1 - pow(e, 2)) * sin(E_k)) / (1 - e * cos(E_k)), (cos(E_k) -
e) / (1 -
    e * cos(E_k)));
Phi_k = nu_k + omega;
del_u_k = Cus * sin(2 * Phi_k) + Cuc * cos(2 * Phi_k);
del_r_k = Crs * sin(2 * Phi_k) + Crc * cos(2 * Phi_k);
del_i_k = Cis * sin(2 * Phi_k) + Cic * cos(2 * Phi_k);
u_k = Phi_k + del_u_k;
r_k = A * (1 - e * cos(E_k)) + del_r_k;
i_k = i0 + del_i_k + IDOT * tk;
x_k = r_k * cos(u_k);
y_k = r_k * sin(u_k);
Omega_k = Omega0 + (OmegaDot - Omega_e) * tk - Omega_e * toe;
x_k1 = x_k * cos(Omega_k) - y_k * cos(i_k) * sin(Omega_k);
y_k1 = x_k * sin(Omega_k) + y_k * cos(i_k) * cos(Omega_k);
z_k1 = y_k * sin(i_k);
coord[0] = x_k1;
coord[1] = y_k1;
coord[2] = z_k1;
}

```

Coord.h

```
#ifndef coord_H
#define coord_H
void coord(long double e, long double mu, long double Omega_e, long double toe,
           long double A, long double n0, long double i0, long double Omega0, long
double
           omega, long double M0,
           long double Dn, long double n, long double OmegaDot, long double IDOT,
long double
           Crs, long double Cuc, long double Cus,
           long double Cic, long double Cis, long double Crc, long double t, long
double* coord);
#endif#pragma once
```

kepler.cpp

```
#include <math.h>

#include <iostream>

#include "kepler.h"
using namespace std;
long double kepler(
    long double en,
    long double Mk)
{
    long double Ek = en * sin(0) + Mk;
    long double Ekold = 0;
    int i = 0;
    while (fabs(Ek - Ekold) > 0.000000001)
    {
        Ekold = Ek;
        Ek = en * sin(Ek) + Mk;
        i++;
    }
    return Ek;
}
```

kepler.h

```
#ifndef kepler_H
```

```
#define kepler_H
```

```
long double kepler(long double en, long double Mk);
```

```
#endif#pragma once
```


СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1]. «Информационно-аналитического центра координатно-временного и навигационного обеспечения «www.glonass-iac.ru»»
- [2]. «Википедия. Свободная энциклопедия «<https://ru.wikipedia.org/wiki/Бэйдоу>»»
- [3]. «Определение формы орбиты и положения спутника на ней «<https://www.celestrak.com>»»
- [4]. «<https://www.gnssplanningonline.com/>»