

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«МЭИ»**

ИНСТИТУТ РАДИОТЕХНИКИ И ЭЛЕКТРОНИКИ

КАФЕДРА РАДИОТЕХНИЧЕСКИХ СИСТЕМ

КУРСОВАЯ РАБОТА

по дисциплине

**«АППАРАТУРА ПОТРЕБИТЕЛЕЙ СПУТНИКОВЫХ РАДИОНАВИГАЦИОННЫХ
СИСТЕМ»**

ФИО СТУДЕНТА: ТАСКАНОВ В.Е.

Группа: ЭР-15-16

ВАРИАНТ №: 13

Дата: _____

Подпись: _____

ФИО ПРЕПОДАВАТЕЛЯ: КОРОГОДИН И.В.

Оценка: _____

МОСКВА, 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
ГЛАВА 1. ИСПОЛЬЗОВАНИЕ СТОРОННИХ СРЕДСТВ	5
1.1 Описание задания.....	5
1.2 Определение формы орбиты и положения спутника на ней с помощью сервиса CelesTrak.....	7
1.3 Расчет графика угла места собственного спутника от времени по данным Trimble GNSS Planning Online	8
1.4 Расчет диаграммы угла места и азимута спутника (SkyView, он же SkyPlot) по данным Trimble GNSS Planning Online	10
1.5 Формирование списка и описание параметров, входящих в состав эфемерид	13
1.6 Формирование таблицы эфемерид собственного спутника	14
ГЛАВА 2. МОДЕЛИРОВАНИЕ	16
2.1 Алгоритм расчета координат	16
2.2 Построим траектории движения спутника	18
ГЛАВА 3. РЕАЛИЗАЦИЯ	21
3.1 Сравнения расчетного положения спутника с «Matlab».....	21
3.2 Тест функции решения уравнения Кеплера	22
3.3 Профилирование программы	23
3.4 Проверка утечки памяти.....	24
3.5 Необходимые файлы для сборки проекта	24
ЗАКЛЮЧЕНИЕ.....	25
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	26

ПРИЛОЖЕНИЕ	27
Приложение 1. Моделирование в Matlab	27
Приложение 2. Моделирование в C++.....	31

ВВЕДЕНИЕ

Спутниковые радионавигационные системы (СРНС) являются самыми точными системами по определению координат потребителя. Они стали важной частью в различных сферах нашей жизни. Наиболее распространенными являются системы ГЛОНАСС (Россия), GPS (США), Galileo (Евросоюз), Beidou (Китай).

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 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 «С++», позволяющую рассчитывать положение спутника Beidou по его эфемеридам.

ГЛАВА 1. ИСПОЛЬЗОВАНИЕ СТОРОННИХ СРЕДСТВ

1.1 Описание задания

В задание дан номер спутника BEIDOU, в моем варианте – C24, а также бинарный и текстовый файл со значениями эфемерид для различных спутников, полученный от трехдиапазонной антенны Harxon HX-CSX601A, установленной на крыше корпуса Е МЭИ. Она через 50-метровый кабель, сплиттер, bias-tee и усилитель подключена к трем навигационным приемникам:

- Javad Lexon LGDD,
- SwiftNavigation Piksi Multi,
- Clonicus разработки ЛИС 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.

Определим с помощью «Информационно-аналитического центра координатно-временного и навигационного обеспечения» [1] номер НОРАД¹ и сравним его с номером из «Википедии» [2]:

¹ НОРАД(SCN) - номер по спутниковому каталогу представляет собой уникальный пятизначный идентификационный номер искусственных спутников Земли.

PRN	НОРАД	Тип КА	Тип системы	Дата запуска	Факт. сущ. (дней)	Примечание
C01	44231	GEO-8	BDS-2	17.05.19	653	Используется по ЦН
C02	38953	GEO-6	BDS-2	25.10.12	3048	Используется по ЦН
C03	41586	GEO-7	BDS-2	12.06.16	1722	Используется по ЦН
C04	37210	GEO-4	BDS-2	01.11.10	3772	Используется по ЦН
C05	38091	GEO-5	BDS-2	25.02.12	3291	Используется по ЦН
C06	36828	IGSO-1	BDS-2	01.08.10	3864	Используется по ЦН
C07	37256	IGSO-2	BDS-2	18.12.10	3725	Используется по ЦН
C08	37384	IGSO-3	BDS-2	10.04.11	3612	Используется по ЦН
C09	37763	IGSO-4	BDS-2	27.07.11	3504	Используется по ЦН
C10	37948	IGSO-5	BDS-2	02.12.11	3376	Используется по ЦН
C11	38250	MEO-3	BDS-2	30.04.12	3226	Используется по ЦН
C12	38251	MEO-4	BDS-2	30.04.12	3226	Используется по ЦН
C13	41434	IGSO-6	BDS-2	30.03.16	1796	Используется по ЦН
C14	38775	MEO-6	BDS-2	19.09.12	3084	Используется по ЦН
C16	43539	IGSO-7	BDS-2	10.07.18	964	Используется по ЦН
C19	43001	MEO-1	BDS-3	05.11.17	1211	Используется по ЦН
C20	43002	MEO-2	BDS-3	05.11.17	1211	Используется по ЦН
C21	43208	MEO-3	BDS-3	12.02.18	1112	Используется по ЦН
C22	43207	MEO-4	BDS-3	12.02.18	1112	Используется по ЦН
C23	43581	MEO-5	BDS-3	29.07.18	945	Используется по ЦН
C24	43582	MEO-6	BDS-3	29.07.18	945	Используется по ЦН

Рисунок 1 – Состав и состояние системы BEIDOU с «Информационно-аналитического центра координатно-временного и навигационного обеспечения»

№	Спутник	PRN	Дата (UTC)	Ракета	NSSDC ID	SCN	Орбита	Статус	Система
33	Бэйдоу-3 M9	C23	29.07.2018 01:48	CZ-3B/YZ-1	2018-062A	43581	COO, ~21 500 км	действующий	Бэйдоу-3
34	Бэйдоу-3 M10	C24			2018-062B	43582	COO, ~21 500 км	действующий	
35	Бэйдоу-3 M11	C26	24.08.2018, 23:37	CZ-3B/YZ-1	2018-067A	43602	COO, ~21 500 км	действующий	
36	Бэйдоу-3 M12	C25			2018-067B	43603	COO, ~21 500 км	действующий	
37	Бэйдоу-3 M13	C32	19.09.2018, 14:07	CZ-3B/YZ-1	2018-072A	43622	COO, ~21 500 км	действующий	
38	Бэйдоу-3 M14	C33			2018-072B	43623	COO, ~21 500 км	действующий	
39	Бэйдоу-3 M15	C35	15.10.2018, 04:23	CZ-3B/YZ-1	2018-078A	43647	COO, ~21 500 км	действующий	
40	Бэйдоу-3 M16	C34			2018-078B	43648	COO, ~21 500 км	действующий	
41	Бэйдоу-3 G1Q	C59	01.11.2018, 15:57	CZ-3B/E	2018-085A	43683	ГСО, 144.5° в. д.	действующий	
42	Бэйдоу-3 M17	C36	18.11.2018, 17:49	CZ-3B/YZ-1	2018-093A	43706	COO, ~21 500 км	действующий	
43	Бэйдоу-3 M18	C37			2018-093B	43707	COO, ~21 500 км	действующий	
44	Бэйдоу-3 IGSO-1	C38	20.04.2019, 14:41	CZ-3B/G2	2019-023A	44204	Геосинхронная, накл. 55°;	действующий	

Рисунок 2 – Состав и состояние системы BEIDOU с сайта Википедия

Из рисунков 1-2 видно, что номер спутника совпадает и равен 43582, название спутника - «BEIDOU-3 M10»

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

Зайдем на официальный сайт CelesTrak [3] и настроим данный сервис для определения формы орбита и положения 24-го спутника на ней

Введем наше название спутника и сверим его по номеру NSSDC ID² и НОРАД (SCN).

Значения совпадают, значит это действительно нужный нам спутник, проведем моделирование на момент времени 15:00, 16 февраля 2021, так как на данном сервисе отсчет времени происходит по UTC(0):

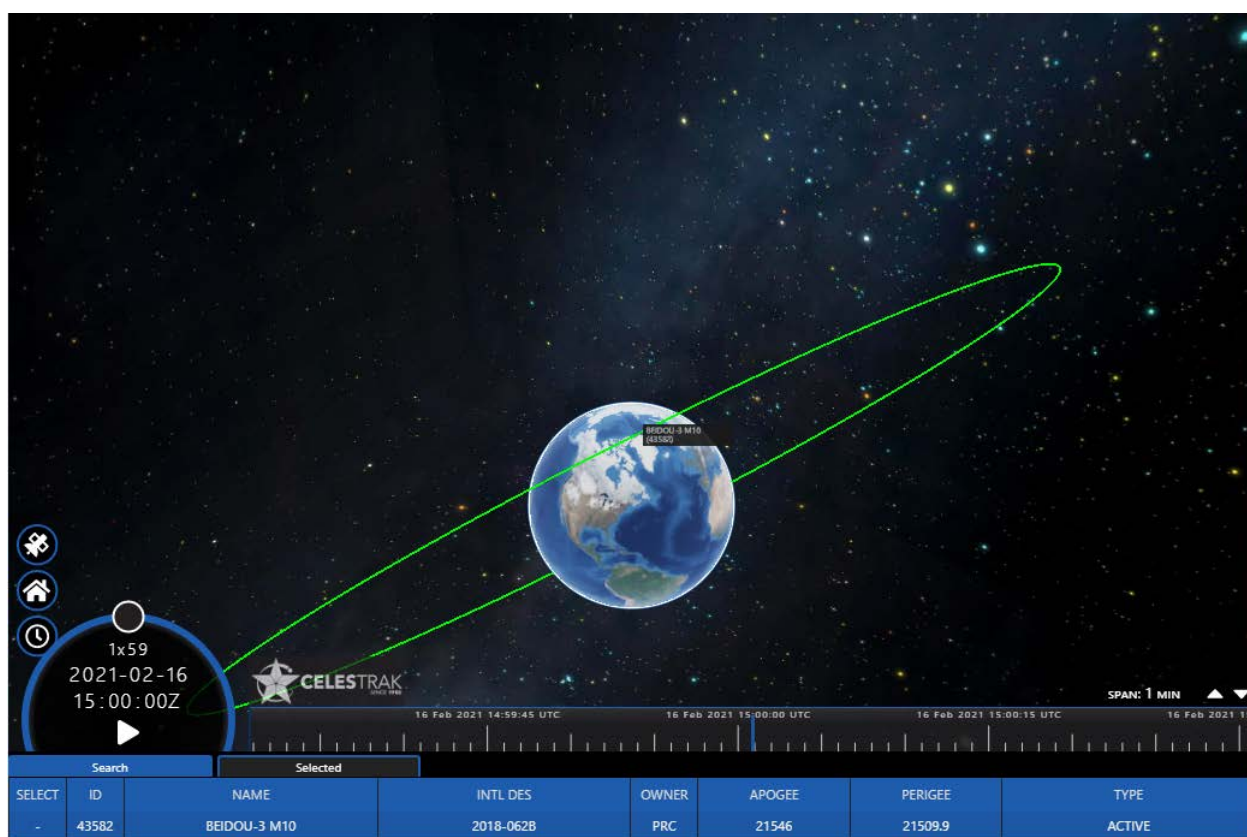


Рисунок 3 – Моделирование с помощью сервиса CelesTrak

² NSSDC ID - номер полёта представляет собой каталожный номер каждого летающего космического объекта, находящегося на орбите и зарегистрированного в COSPAR (Комитет по космическим исследованиям)

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

Настроим для моделирования GNSS Planning Online [4], 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, высота выбирается из суммы высоты над уровнем моря (146 м) и примерной высотой здания (25 м) и округляется до сотен.

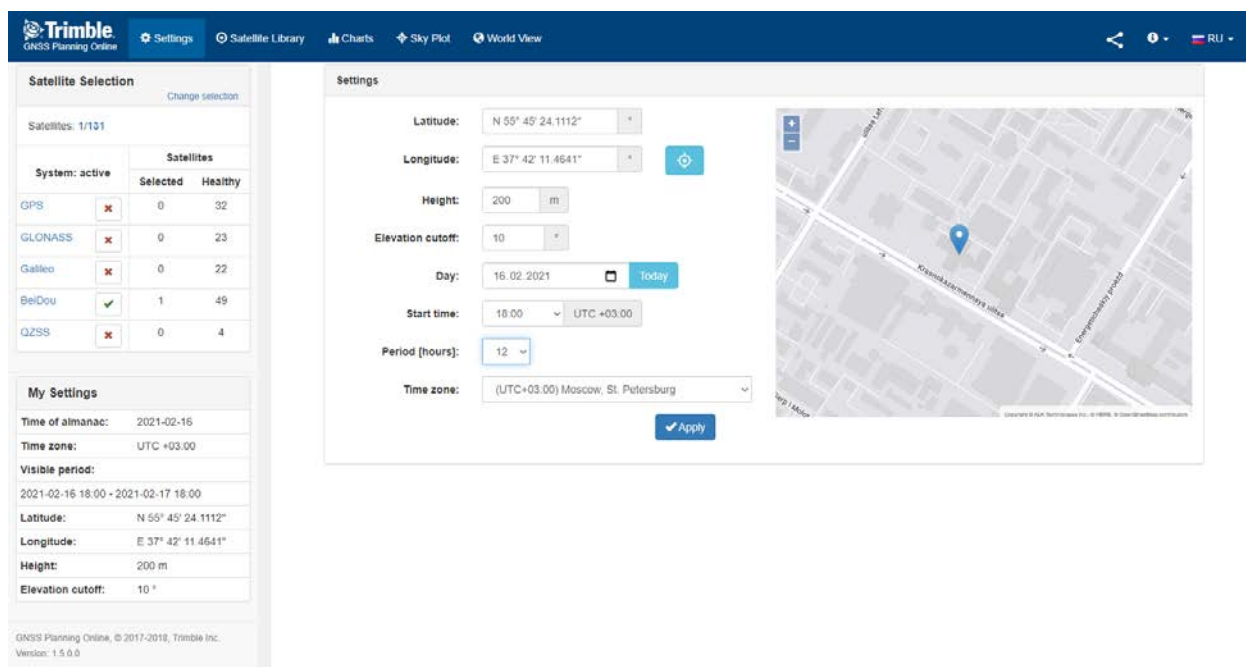


Рисунок 4 – Моделирование с помощью сервиса Trimble GNSS Planning

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

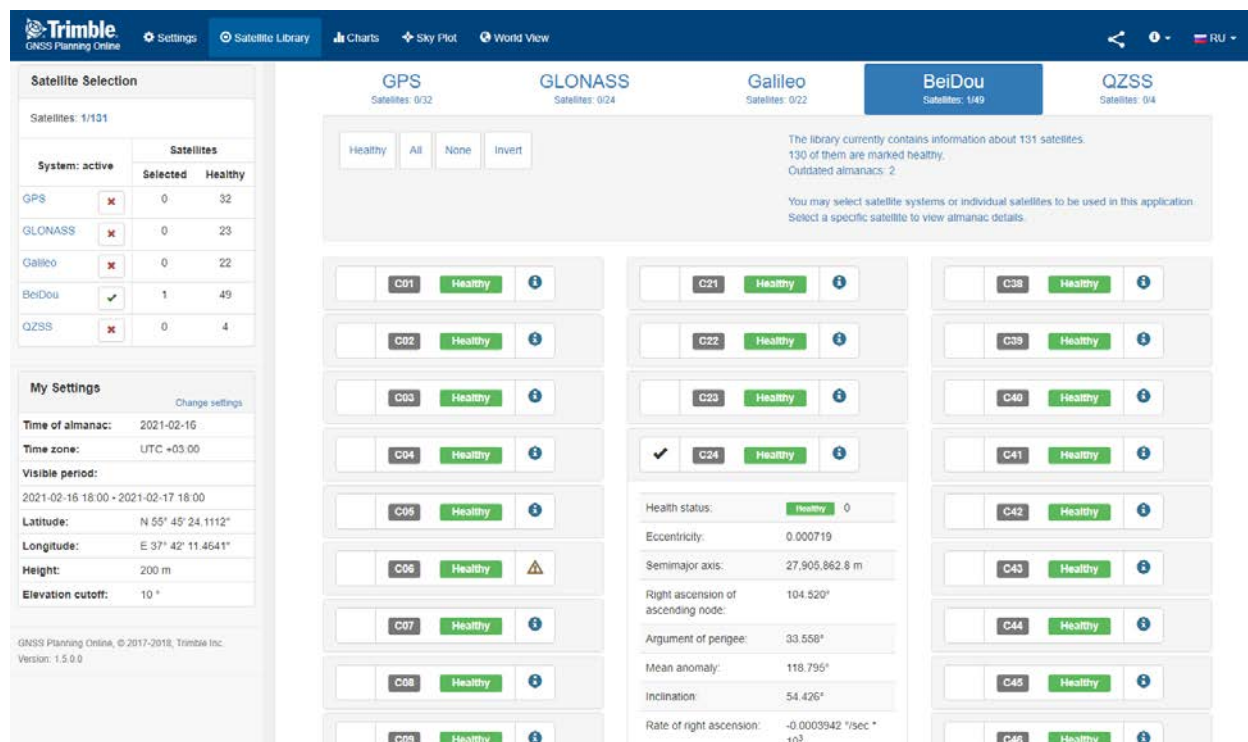


Рисунок 5 – Моделирование с помощью сервиса Trimble GNSS Planning

Получим график расчета угла места собственного спутника от времени

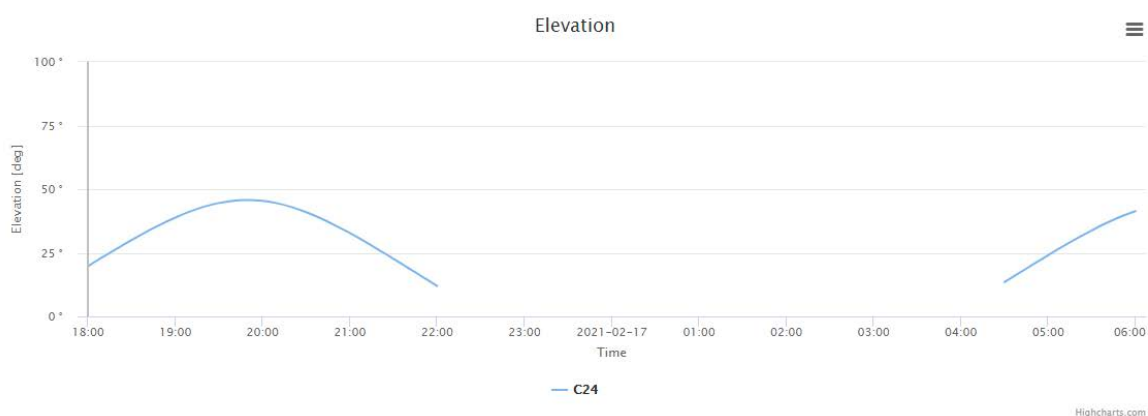


Рисунок 6 – График угла места собственного спутника от времени

По графику видно, что на указанном в задание интервале с 18:00 – 06:00, спутник был в области видимости 2 раза - с 18:00 до 22:00 и с 4:30 до 6:00.

1.4 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, 22:00, 4:30 и 6:00.

Тогда получим 4-е графика моделирования:

- 16 февраля 2021 в 18:00

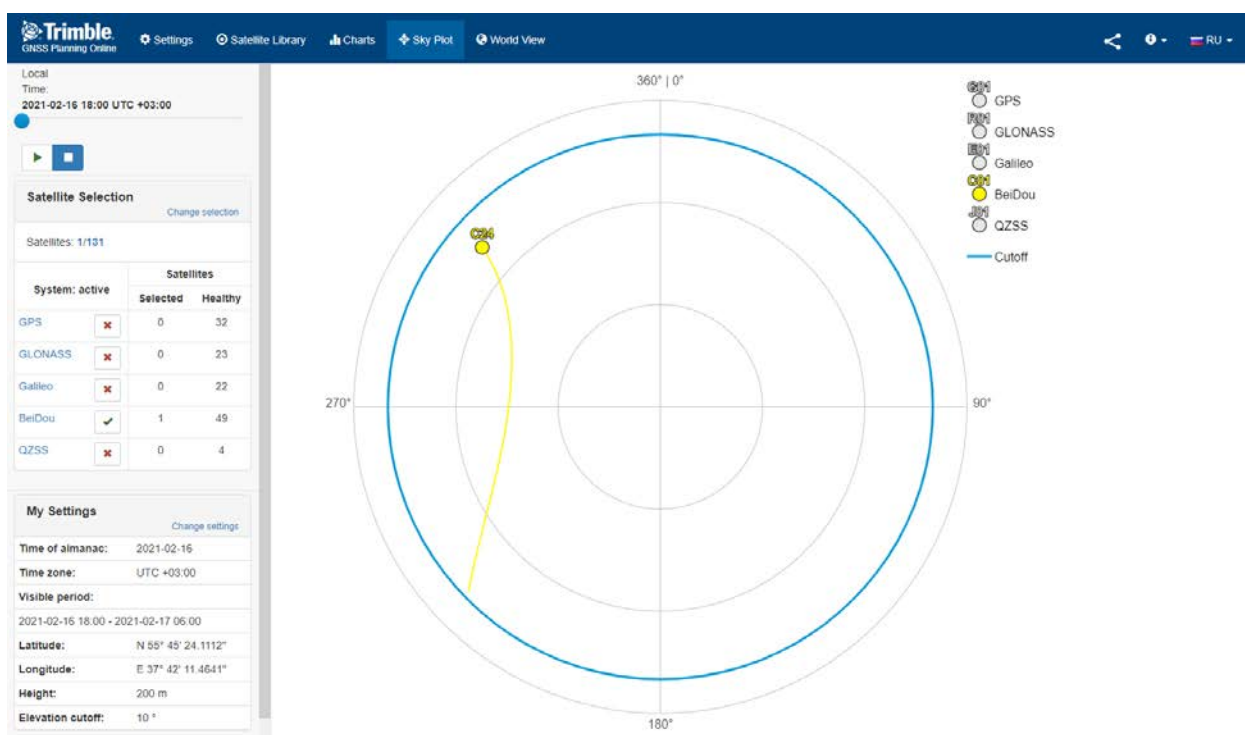


Рисунок 7 – Моделирование с помощью сервиса Trimble GNSS Planning

- 16 февраля 2021 в 22:00

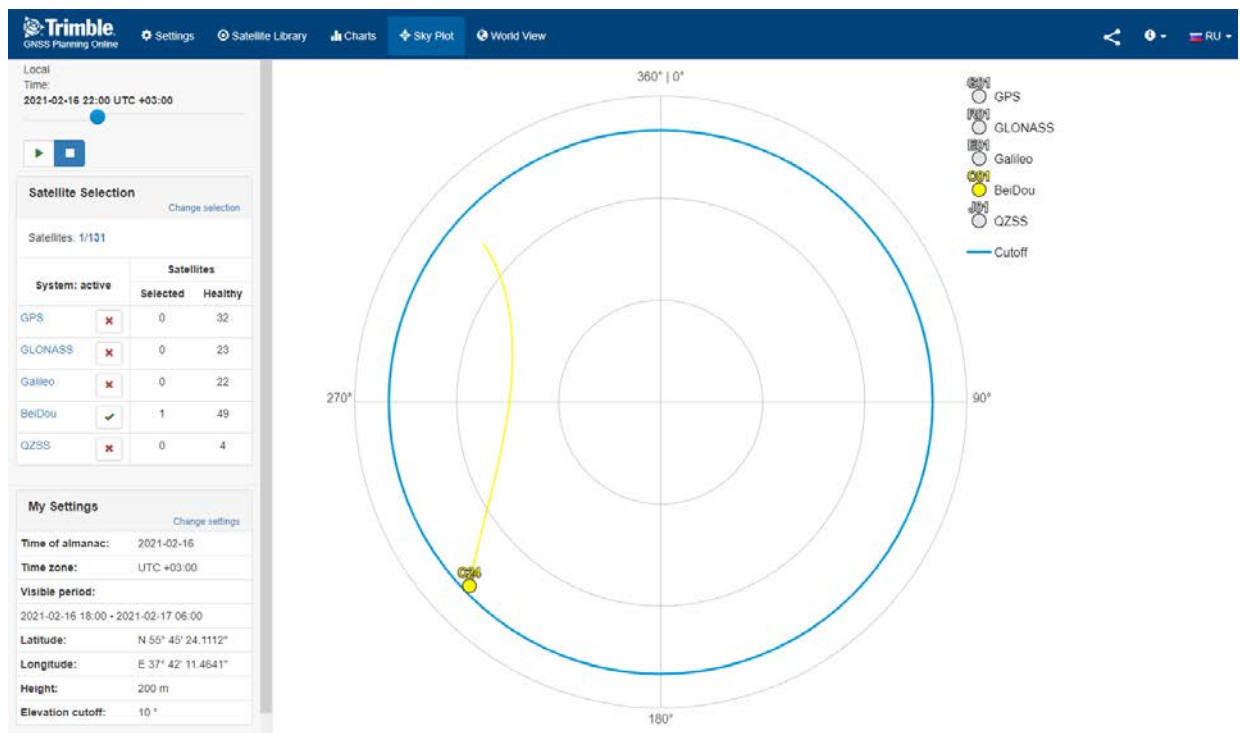


Рисунок 8 – Моделирование с помощью сервиса Trimble GNSS Planning

- 17 февраля 2021 в 4:30

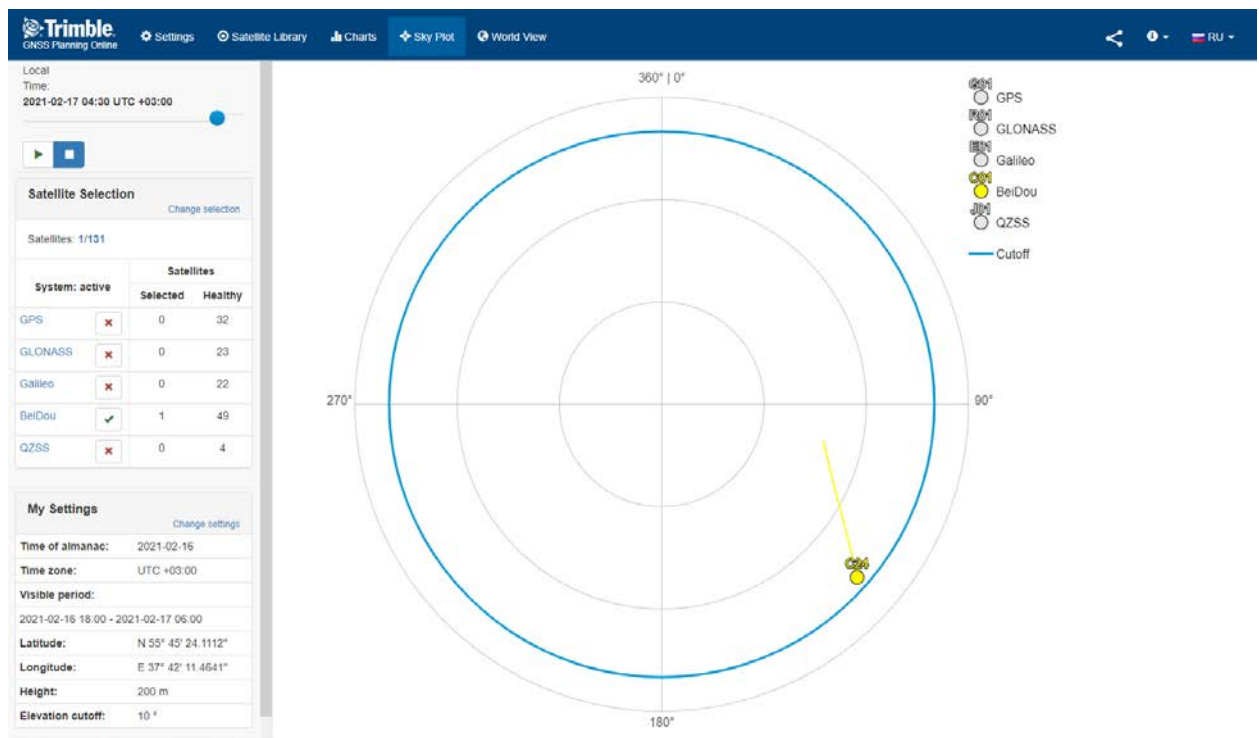


Рисунок 9 – Моделирование с помощью сервиса Trimble GNSS Planning

- 17 февраля 2021 в 6:00

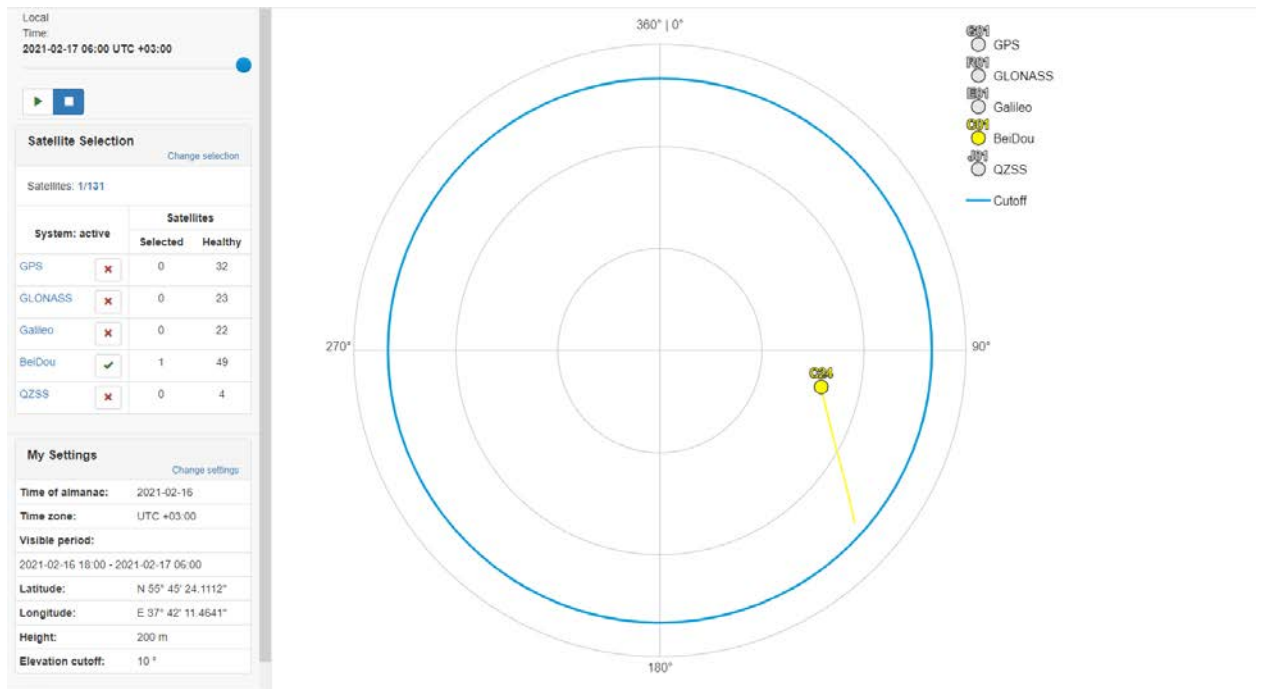


Рисунок 10 – Моделирование с помощью сервиса Trimble GNSS Planning

Для удобства наложим друг на друга полученные 4 графика - рисунок 7-10 и получим карту небосвода

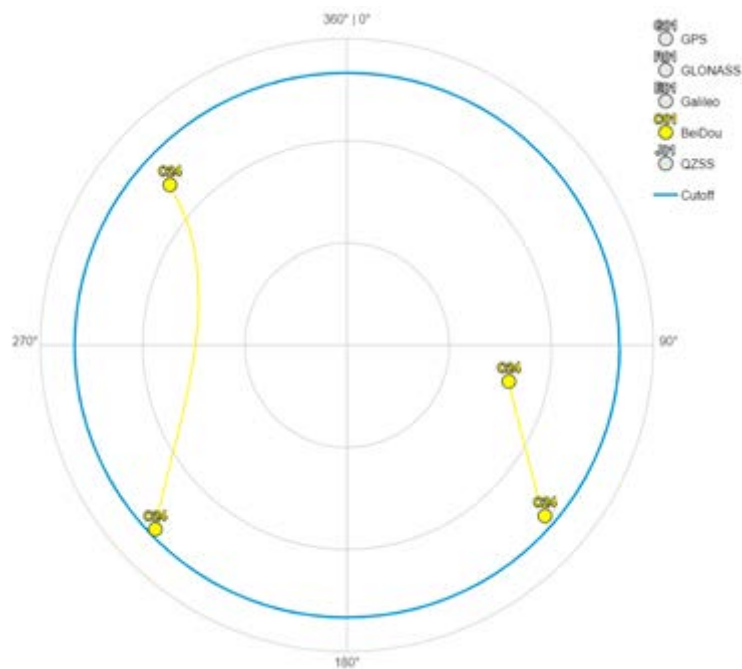


Рисунок 11 – Карта небосвода

1.5 Формирование списка и описание параметров, входящих в состав эфемерид

Таблица 1 – Описание параметров, входящих в состав эфемерид

Параметры	Определение
t_{oe}	Отсчет времени эфемерид
\sqrt{A}	Квадратный корень из большой полуоси орбиты
e	Эксцентриситет
ω	Аргумент перигея
Δn	Среднее отклонение движения от расчетного значения
M_0	Средняя аномалия в исходное время
Ω_0	Долгота восходящего узла орбитальной плоскости, вычисленная по опорному времени
$\dot{\Omega}$	Скорость прямого восхождения
i_0	Угол наклона в исходное время
$IDOT$	Скорость угла наклона
C_{uc}	Амплитуда косинусной поправки к аргументу широты
C_{us}	Амплитуда синусной поправки к аргументу широты
C_{rc}	Амплитуда косинусной поправки к радиусу орбиты
C_{rs}	Амплитуда синусной поправки к радиусу орбиты
C_{ic}	Амплитуда косинусной поправки к углу наклона
C_{is}	Амплитуда синусной поправки к углу наклона

1.6 Формирование таблицы эфемерид собственного спутника

Данные спутника берутся из текстового файла, полученного из дампа бинарного потока данных от приемника в формате NVS BINR.

Таблица 2 – Значения эфемерид спутника C24

Параметры	Значение	Размерность
SatNum	24	-
toe, t_{oe}	223200000.000	мс
Crs, C_{rs}	6.560937500000000000e+01	м
Dn, Δn	4.35125286496473862e-12	рад/мс
M0, M_0	1.40953592060026917e-01	рад
Cuc, C_{uc}	3.36393713951110840e-06	рад
e	7.20643904060125351e-04	-
Cus, C_{us}	5.78071922063827515e-06	рад
sqrtA, \sqrt{A}	5.28261434555053711e+03	м ^{1/2}
Cic, C_{ic}	2.32830643653869629e-09	рад
Omega0, Ω_0	1.82433512285772315e+00	рад
Cis, C_{is}	-6.56582415103912354e-08	рад
i0, i_0	9.49918991207442720e-01	рад
Crc, C_{rs}	2.343906250000000000e+02	м
omega, ω	5.72152208390331540e-01	рад
OmegaDot, $\dot{\Omega}$	-7.16708425211283236e-12	рад/мс
iDot, \dot{I}	1.51077721564943834e-13	рад/мс
Tgd, T_{GD}	7.000000000000000000e+04	мс
toc, t_{oc}	2.232000000000000000e+08	мс
af2, a_{f2}	0.000000000000000000e+00	мс/мс ²

af1, a_{f1}	-8.33733082572507556e-12	MC/MC
af0, a_{f0}	-7.61786103248596191e-01	MC
URA	0	-
IODE	257	-
IODC	1	-
codeL2	0	-
L2P	0	-
WN	789	-

ГЛАВА 2. МОДЕЛИРОВАНИЕ

Моделирование проводится в программе Matlab. Код программы приведен в приложение 1.

2.1 Алгоритм расчета координат

2.1.1 Определим время, отсчитываемое от опорной эпохи эфемерид:

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

Если $t_k > 302400$ сек, вычитаем 604800 сек из t_k . Если $t_k < -302400$ сек, добавим 604800 сек.

2.1.2 Определим среднее движение:

$$n_0 = \sqrt{\frac{\mu}{A_0^3}}$$

2.1.3 Определим скорректированное среднее движение:

$$n_A = n_0 + \Delta n$$

2.1.4 Определим среднюю аномалию:

$$M_k = M_0 + n_A \cdot t_k$$

2.1.5 Решим уравнение Кеплера минимум 3-мя итерациями и определим E_k :

$$M_k = E_k - e_n \cdot \sin(E_k) \Rightarrow E_k = M_k + e_n \cdot \sin(E_k)$$

2.1.6 Определим истинную аномалию:

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

2.1.7 Вычислим аргумент широты u_k из аргумента перигея ω , истинной аномалии v_k и поправок c_{uc} и c_{us} :

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

2.1.8 Вычислим радиальное расстояние r_k с учетом поправок c_{rc} и c_{rs} :

$$r_k = a(1 - e_n \cos(E_k)) + c_{rc} \cos 2(\omega + v_k) + c_{rs} \sin 2(\omega + v_k)$$

2.1.9 Вычислим наклон i_k орбитальной плоскости по наклону i_0 в эталонное время t_{oe} и поправки c_{ic} и c_{is} :

$$r_k = i_0 + \dot{i} \cdot t_k + c_{ic} \cos 2(\omega + v_k) + c_{is} \sin 2(\omega + v_k)$$

2.1.10 Вычислим долготу восходящего узла λ_k (относительно Гринвича):

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

2.1.11 Вычислите координаты, применив три поворота (вокруг u_k , i_k и λ_k)

$$\begin{bmatrix} X_k \\ Y_k \\ Z_k \end{bmatrix} = \mathbf{R}_3(-\lambda_k) \mathbf{R}_1(-i_k) \mathbf{R}_3(-u_k) \begin{bmatrix} r_k \\ 0 \\ 0 \end{bmatrix}$$

где \mathbf{R}_1 и \mathbf{R}_3 - матрицы вращения, определенные в преобразовании между наземными кадрами.

2.2 Построим траектории движения спутника

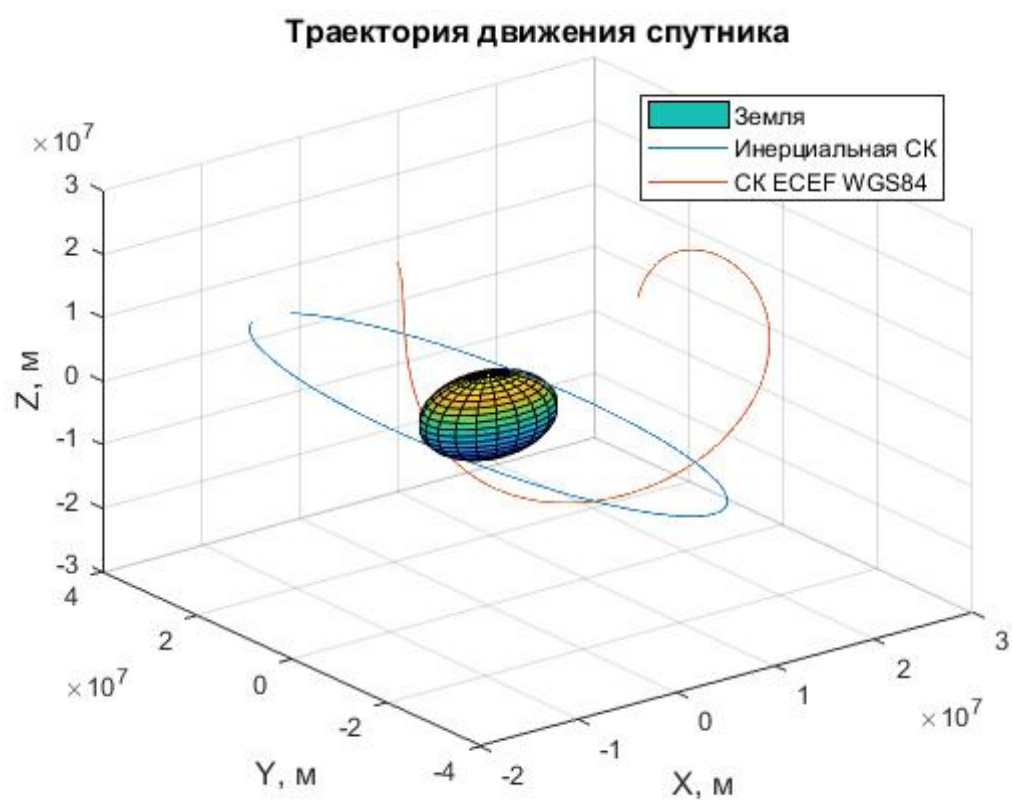


Рисунок 12 – Траектории движения спутника

Расчет графиков SkyView

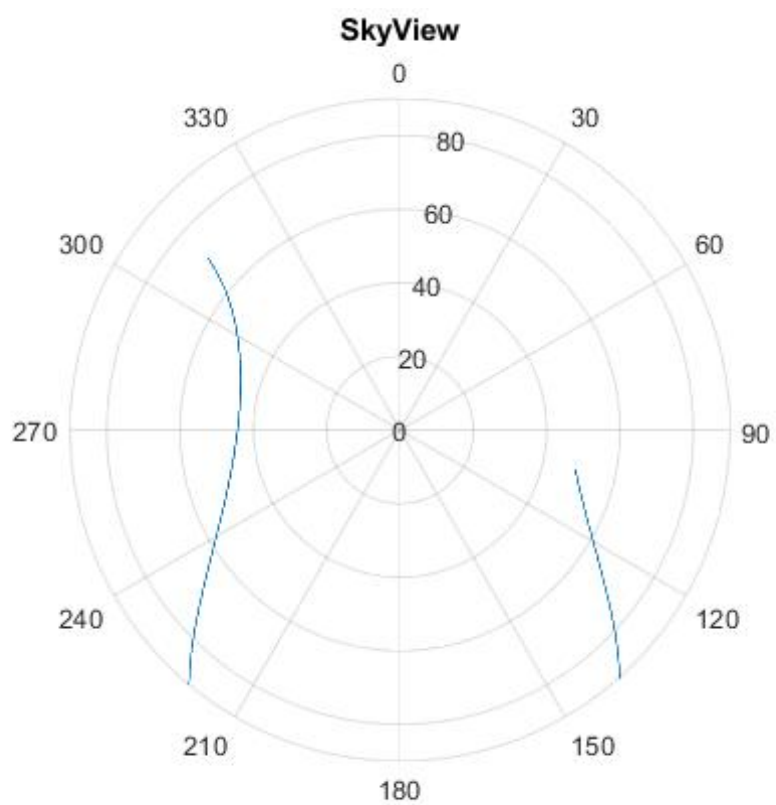


Рисунок 13 – SkyView

График угла места

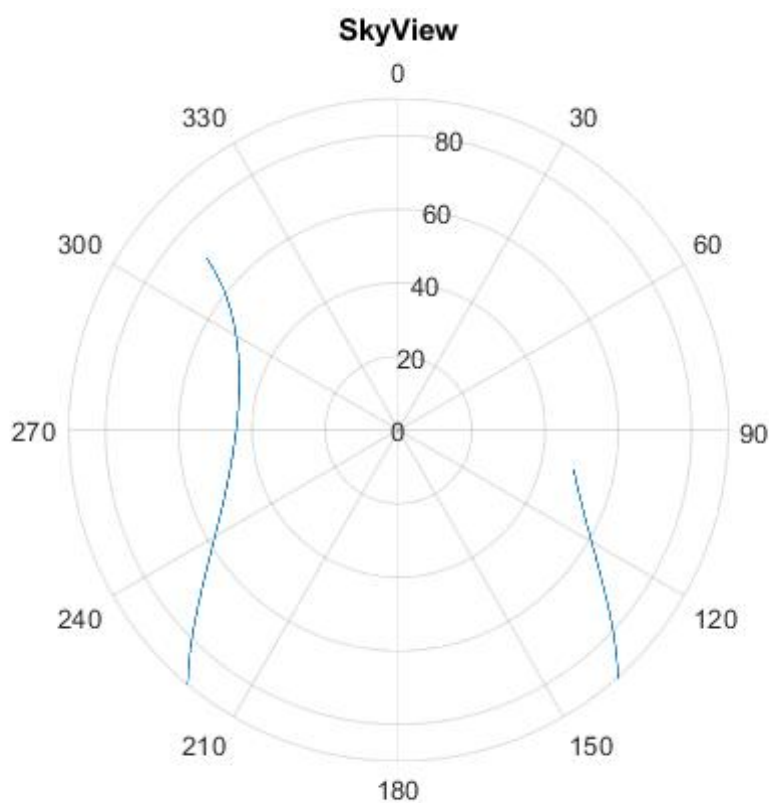


Рисунок 14 – График угла места

По рисункам 13-14 можно убедиться, что графики совпали с рассчитанными ранее на сайте «Trimble GNSS Planning Online», а значит моделирование выполнено верно.

ГЛАВА 3. РЕАЛИЗАЦИЯ

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.

Для реализации библиотеки выберем среду – CodeBlocks, где в качестве компилятора используется MinGW. Тест функции решения уравнения Кеплера будем проводить с помощью библиотеки UnitTest++[6].

Для упрощения работы с матрицами в «C++» воспользуемся библиотекой Armadillo.

3.1 Сравнения расчетного положения спутника с «Matlab»

Для этого модернизируем программу в Matlab и добавим функцию сохранения координат в файл для дальнейшей передачи его в C++, где добавим функцию сравнения расчетных координат.

Получим максимальную погрешность с данными из Matlab равную $2.72249 \cdot 10^{-6}$ метра, а средняя равна $1.06437 \cdot 10^{-6}$ метра (Рисунок 15). Эта погрешность обусловлена типом данных в Matlab, так как используется 16 цифр точности, а в C++ для верного расчета времени использую long double, где используется 15, 18 или 33 цифры (в зависимости от того, сколько байт занимает тип данных на компьютере), а также использования библиотеки Armadillo, которая при расчете в матрице сохраняет 6 значимых цифр.

3.2 Тест функции решения уравнения Кеплера

Для проведения теста необходимо рассчитать правильные значения, относительно которых будет сравниваться тестовое, для этого решим уравнение Кеплера при значениях $e = 7.20643904060125351e-04$ и $M_0 = 0.628525$, тогда получим значения $E_k = 0.62895$.

Запишем в сравнение теста правильный результат

```
Running target post-build steps
.\LP2.exe
srdx=1.06437e-006
srdy=1.06437e-006
srdz=1.06437e-006
dxmax=2.72249e-006
dymax=2.72249e-006
dzmax=2.72249e-006
Success: 1 tests passed.
Test time: 0.00 seconds.
Process terminated with status 0 (0 minute(s), 5 second(s))
0 error(s), 0 warning(s) (0 minute(s), 5 second(s))
```

Рисунок 15 – Тест функции решения уравнения Кеплера

Запишем в сравнение теста ошибочный результат

```
Running target post-build steps
.\LP2.exe
D:\my\test\testaadd\Test.cpp:13:1: error: Failure in OurFirstTest: Expected 0.62895 but was 0.628949
srdx=1.06437e-006
srdy=1.06437e-006
srdz=1.06437e-006
dxmax=2.72249e-006
dymax=2.72249e-006
dzmax=2.72249e-006
FAILURE: 1 out of 1 tests failed (1 failures).
Test time: 0.00 seconds.
Process terminated with status 1 (0 minute(s), 5 second(s))
1 error(s), 0 warning(s) (0 minute(s), 5 second(s))
```

Рисунок 16 – Тест функции решения уравнения Кеплера

Тест функция показала на ошибку, следовательно, работает правильно.

3.3 Профилирование программы

С помощью встроенной утилиты – Gprof, проведем профилирование программы, чтобы узнать на какую функцию затрачивается больше времени

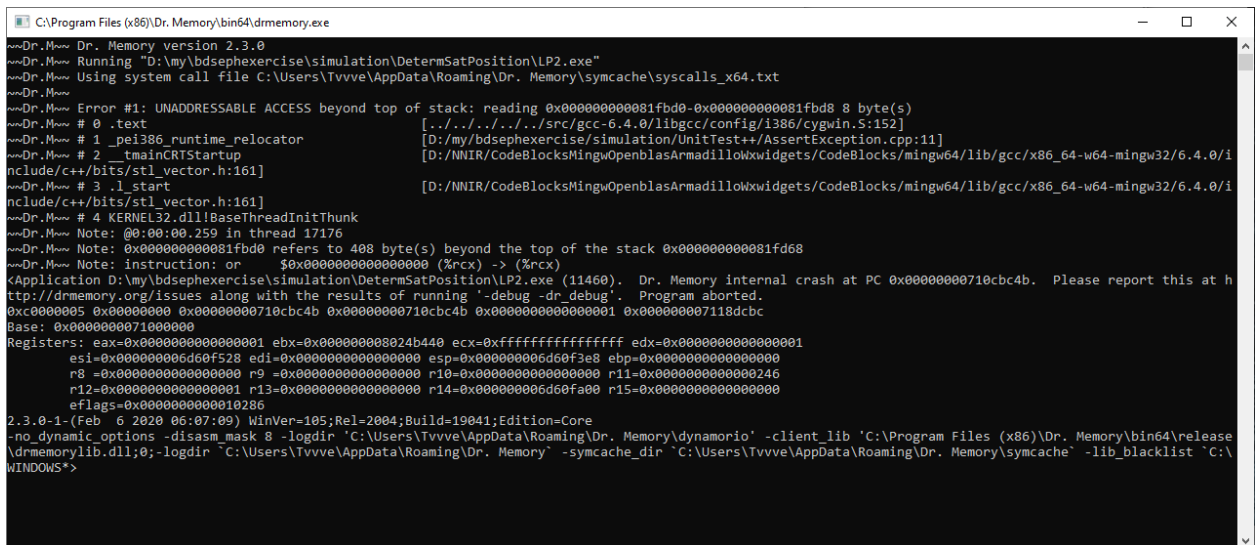
[illegible]

Рисунок 17 – Профилирование программы утилитой Gprof

Из полученных результатов видно, что наибольшую часть времени программа затрачивает на функцию расчета положения спутника, это минимум 29.7% процентов времени и связано это в большей части из-за использования библиотеки Armadillo, которая упрощает математические расчеты.

3.4 Проверка утечки памяти

Проверку на утечку памяти будем проводить с помощью утилиты Dr.Memory [7]



```
C:\Program Files (x86)\Dr. Memory\bin64\drmemory.exe
~Dr.Mem~ Dr. Memory version 2.3.0
~Dr.Mem~ Running "D:\my\bdsephexercise\simulation\DetermSatPosition\LP2.exe"
~Dr.Mem~ Using system call file C:\Users\Tvvve\AppData\Roaming\Dr. Memory\symcache\syscalls_x64.txt
~Dr.Mem~
~Dr.Mem~ Error #1: UNADDRESSABLE ACCESS beyond top of stack: reading 0x000000000081fbd0-0x000000000081fbd8 8 byte(s)
~Dr.Mem~ # 0 .text [D:\my\bdsephexercise\simulation\UnitTest++\AssertException.cpp:11]
~Dr.Mem~ # 1 _pei386_runtime_relocator [D:\my\bdsephexercise\simulation\UnitTest++\AssertException.cpp:11]
~Dr.Mem~ # 2 __tmainCRTStartup [D:\my\bdsephexercise\simulation\UnitTest++\AssertException.cpp:11]
~Dr.Mem~ # 3 .!_start [D:\my\bdsephexercise\simulation\UnitTest++\AssertException.cpp:11]
~Dr.Mem~ # 4 KERNEL32.dll!BaseThreadInitThunk [D:\my\bdsephexercise\simulation\UnitTest++\AssertException.cpp:11]
~Dr.Mem~ Note: @0:00:00.259 in thread 17176 [D:\my\bdsephexercise\simulation\UnitTest++\AssertException.cpp:11]
~Dr.Mem~ Note: 0x000000000081fbd0 refers to 400 byte(s) beyond the top of the stack 0x000000000081fd68 [D:\my\bdsephexercise\simulation\UnitTest++\AssertException.cpp:11]
~Dr.Mem~ Note: Instructions or $0x0000000000000000 (%rcx) -> (%rcx) [D:\my\bdsephexercise\simulation\UnitTest++\AssertException.cpp:11]
~Dr.Mem~ Note: Application D:\my\bdsephexercise\simulation\DetermSatPosition\LP2.exe (11460). Dr. Memory internal crash at PC 0x00000000710cbc4b. Please report this at h
http://drmemory.org/issues along with the results of running '-debug -dr debug'. Program aborted.
0xc0000005 0x00000000 0x00000000710cbc4b 0x00000000710cbc4b 0x0000000000000001 0x000000007118dcbc
Base: 0x0000000071000000
Registers: eax=0x0000000000000001 ebx=0x0000000000000000 ecx=0x0000000000000000 edx=0x0000000000000000
esi=0x0000000000000000 edi=0x0000000000000000 esp=0x0000000000000000 ebp=0x0000000000000000
r8 =0x0000000000000000 r9 =0x0000000000000000 r10=0x0000000000000000 r11=0x0000000000000000
r12=0x0000000000000000 r13=0x0000000000000000 r14=0x0000000000000000 r15=0x0000000000000000
eflags=0x0000000000000000
2.3.0-1-(Feb 6 2020 06:07:09) WinVer=105;Rel=2004;Build=19041;Edition=Core
-no_dynamic_options -disasm_mask 8 -logdir 'C:\Users\Tvvve\AppData\Roaming\Dr. Memory\dynamorio' -client_lib 'C:\Program Files (x86)\Dr. Memory\bin64\release
\drmemorylib.dll;0; -logdir 'C:\Users\Tvvve\AppData\Roaming\Dr. Memory' -symcache_dir 'C:\Users\Tvvve\AppData\Roaming\Dr. Memory\symcache' -lib_blacklist 'C:\
WINDOWS\>
```

Рисунок 18 – Проверка утечки памяти

Имеется одна ошибка, связанная с библиотекой UnitTest++. По самой программе и библиотеке для расчета координат спутника Veidou ошибок не наблюдается.

3.5 Необходимые файлы для сборки проекта

К отчету прикреплены 8 приложений, в которых содержатся основные файлы кода программы, необходимые для сборки проекта.

Файл “main.cpp” –содержит в себе основной алгоритм программы и работы с функциями, описанных в последующих 7 файлах.

В файле “ephemerids.cpp”- реализована функция расчета положения спутника в определенный момент.

Файл “ephemerids.h” – заголовочный файл, в котором объявляются применяемые классы и методы для функции “ephemerids”.

В файле “Kerpler.cpp” – реализована функция расчета уравнения Кеплера.

Файл “Kepler.h” - заголовочный файл, в котором объявляются применяемые классы и методы для функции “ Kepler”.

В файле “func.cpp” – описаны функции для симуляции функции “ephemerids” – “simul”, функция для обработки файла – “parser”, который содержит сохраненные координаты, рассчитанные в программе Matlab, функция для расчета погрешности при сравнении массивов координат – “comparr”.

Файл “func.h” - заголовочный файл, в котором объявляются применяемые классы и методы для функции “ simul ”, “parser”, “comparr”.

В файле “Test.cpp” – описана тест функция.

ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы написал на языке C++ функцию расчета положения спутника Beidou на заданное время по шкале UTC, ознакомился при проведении тестов с библиотекой UnitTest++, построил модель положения спутника в координатах ECEF и ECI, построил графики аналогичные с ресурсом GNSS Planing Online SkyView, научился использовать интерфейсные контрольные документы, а также использовать таблицы эфемерид для дальнейших расчетов. Определил погрешность вычисления между функции написанной на языке C++ и Matlab. Выясни с помощью встроенной утилиты Gprof, где наибольшую часть времени затрачивается выполнения программы, а также с помощью утилиты Dr.Memory провел тест на утечку памяти, которая выявила одну ошибку связанную с библиотекой UnitTest++.

В результате выполнения курсовой работы получил библиотечную функцию, написанную на языке C++, позволяющую рассчитать положение спутника Beidou на заданное время по шкале UTC.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1]. Электронный ресурс: «Информационно-аналитического центра координатно-временного и навигационного обеспечения «www.glonass-iac.ru»»
- [2]. Электронный ресурс: «Википедия. Свободная энциклопедия «<https://ru.wikipedia.org/wiki/Бэйдоу>»»
- [3]. Электронный ресурс: «<https://www.celestrak.com>»
- [4]. Электронный ресурс: «<https://www.gnssplanningonline.com/>»
- [5]. Электронный ресурс:
«https://gssc.esa.int/navipedia/index.php/GPS_and_Galileo_Satellite_Coordinates_Computation»
- [6]. Электронный ресурс: «Библиотека UnitTest++»
«<https://unittest-cpp.github.io>»
- [7]. Электронный ресурс: «Утилита Dr.Memory»
«<https://drmemory.org>»

ПРИЛОЖЕНИЕ

Приложение 1. Моделирование в Matlab

Calc.m

```
clear all;
clc;
close all;
a=(5.28261434555053711e+03)^2;
toe = 223200000.000 * 10^-3; %ms
M0 =1.40953592060026917e-01 ;
dn = (4.35125286496473862e-12)/10^-3; %rad/ms
w= 5.72152208390331540e-01;
Cuc =5.78071922063827515e-06;
Crc =2.34390625000000000e+02;
Crs = 6.56093750000000000e+01;
Cic =2.32830643653869629e-09;
Cis =-6.56582415103912354e-08;
Cus =5.78071922063827515e-06;
i0 = 9.49918991207442720e-01;
IDOT =1.51077721564943834e-13;
Omega0 =1.82433512285772315e+00;
OmegaDot = -7.16708425211283236e-12;
e= 7.20643904060125351e-04;
n = 3.986004418e+14;
OMEGA_e = 7.2921151467*10^-5; %%
omegaE=OMEGA_e;%%
tstart = (24*2 + 18 - 3)*60*60;
tstop = (24*3 + 6 - 3)*60*60;
t_arr = tstart:1:tstop;
%
dt = 12*60*60;
for i = 1:1:length(t_arr)
coord(:, i) = CoordGPS(t_arr(i),toe, M0,e, n, a, dn,w, Cuc, Cus, Crc,
Crs,Cic, Cis,i0, IDOT, Omega0,OmegaDot, omegaE );
coord_eci_x(i)= coord(i).ECI(1);
coord_eci_y(i)=coord(i).ECI(2);
coord_eci_z(i) = coord(i).ECI(3);
coord_ecef_x(i) = coord(i).ECEF(1);
coord_ecef_y(i) = coord(i).ECEF(2);
coord_ecef_z(i) = coord(i).ECEF(3);
coord_eci(1,i) = coord(i).ECI(1);
coord_eci(2,i) = coord(i).ECI(2);
```

```

coord_eci(3,i) = coord(i).ECI(3);
coord_ecef(1,i) = coord(i).ECEF(1);
coord_ecef(2,i) = coord(i).ECEF(2);
coord_ecef(3,i) = coord(i).ECEF(3);
end
%
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 = [coord_ecef_x; coord_ecef_y; coord_ecef_z];
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.'
figure (1)
E = wgs84Ellipsoid;
ellipsoid(0,0,0,E.SemimajorAxis, E.SemimajorAxis,
E.SemiminorAxis);
hold on;
plot3(coord_eci_x, coord_eci_y, coord_eci_z);
hold on;
plot3(crd_WGS_84(:,1), crd_WGS_84(:,2), crd_WGS_84(:,3))
title('Траектория движения спутника', 'FontSize',12)
xlabel('X, м', 'FontSize',12)
ylabel('Y, м', 'FontSize',12)
zlabel('Z, м', 'FontSize',12)
lgd = legend('Земля','Инерциальная СК','СК ECEF WGS84');
% Географические координаты корпуса E 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 = 200; % 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
% Skyplot
figure (2)
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+3); % Перевод временной оси в
формат hh:mm:ss
figure (3)
grid on
hold on
plot(th,(-teta)*180/pi+90,'DurationTickFormat','hh:mm:ss')
xlim([th(1) th(end)])
title('Угол места', 'FontSize',12)
xlabel('Время в UTC', 'FontSize',12)
ylabel('Угол места, град', 'FontSize',12)

```

```

function coord = CoordGPS(t,toe, M0,e, n, a, dn,w, Cuc, Cus, Crc,
Crs,Cic, Cis,i0, IDOT, Omega0,OmegaDot, omegaE )
tk = t-toe;
if tk>302400
    tk = tk-604800;
else if tk < -302400
    tk = tk + 604800;
end
end
Mk = M0+( (sqrt(n/(a^3))) + dn)*tk;
Ekold =0;
Ek = e*sin(0)+Mk;
while (abs(Ek-Ekold)> 10^-9)
    Ekold = Ek;
    Ek = Mk+e*sin(Ek);
end
vk = atan2( (sqrt(1-e^2))*sin(Ek) , cos(Ek)-e );
uk = w+ vk + Cuc*cos(2*(w+vk))+Cus*sin(2*(w+vk));
rk = (a*(1-e*cos(Ek)))+Crc*cos(2*(w+vk))+Crs*sin(2*(w+vk));
ik = i0 + IDOT*tk+ Cic*cos(2*(w+vk))+Cis*sin(2*(w+vk));
lymbdak_ECEF = Omega0 + (OmegaDot - omegaE)*tk-
omegaE*toe;
%%%
lymbdak_ECI = Omega0 + OmegaDot*tk ;
%
R1 = [1 0 0;
    0 cos(-ik) sin(-ik);
    0 -sin(-ik) cos(-ik)];
R3l_ECEF = [cos(-lymbdak_ECEF) sin(-lymbdak_ECEF) 0;
    -sin(-lymbdak_ECEF) cos(-lymbdak_ECEF) 0;
    0 0 1];

R3uk = [cos(-uk) sin(-uk) 0;
    -sin(-uk) cos(-uk) 0;
    0 0 1];

coord.ECEF = R3l_ECEF*R1*R3uk* [rk;0;0];
%
R3l_ECEF = [cos(-lymbdak_ECI) sin(-lymbdak_ECI) 0;
    -sin(-lymbdak_ECI) cos(-lymbdak_ECI) 0;
    0 0 1];

```

```
coord.ECI = R3l_ECEF*R1*R3uk* [rk;0;0];  
end
```

Приложение 2. Моделирование в C++

Main.cpp

```
#include "UnitTest++.h"  
#include "funct.h"  
#include "ephemerids.h"  
#include <iostream>  
#include <math.h>  
  
using namespace std;  
const int nn = 432001; //задаем n для массивов  
int main()  
{  
  
double* X = new double[nn];  
double* Y = new double[nn];  
double* Z = new double[nn];  
double* Xmatlab = new double[nn];  
double* Ymatlab = new double[nn];  
double* Zmatlab = new double[nn];  
  
parser(Xmatlab,Ymatlab,Zmatlab);  
simul(X,Y,Z);  
double srdx=0;  
double srdy=0;  
double srdz=0;  
double dxmax = 0;  
double dymax = 0;
```

```

double dzmax = 0;
comparr(X,Y,Z,          Xmatlab,Ymatlab,Zmatlab,nn-10,
&srdx,&srdy,&srdz, &dxmax,&dymax,&dzmax );
delete[] X;
delete[] Y;
delete[] Z;
delete[] Xmatlab;
delete[] Ymatlab;
delete[] Zmatlab;

cout << "srdx="<<srdx<<endl;
cout << "srdy="<<srdy<<endl;
cout << "srdz="<<srdz<<endl;
cout << "dxmax="<<dxmax<<endl;
cout << "dymax="<<dymax<<endl;
cout << "dzmax="<<dzmax<<endl;

return UnitTest::RunAllTests();
}

```

ephemerids.cpp

```

#include "ephemerids.h"
#include "Kepler.h"
#include <armadillo>

using namespace arma;

Coordinates CoordGPS(double t,

```



```

double toe,
double M0,
double e,
double n,
double a,
double dn,
double w,
double Cuc,
double Cus,
double Crc,
double Crs,
double Cic,
double Cis,
double i0,
double IDOT,
double Omega0,
double OmegaDot,
double omegaE )

```

```

{
Coordinates CoordinatesGps;
double tk = t-toe;
if (tk > 302400)
{
tk = 604800-tk;
}
else if (tk<-302400)
{
tk = 604800+tk;
}
}

```

```

}
double Mk = M0+( (sqrt(n/(pow(a,3)))) + dn)*tk;
double Ek = kepler(e,Mk);
double vk = atan2(sqrt(1-(pow(e,2)))*sin(Ek), (cos(Ek)-e) );
double      uk      =      w+      vk      +
Cuc*cos(2*(w+vk))+Cus*sin(2*(w+vk));
double      rk      =      (a*(1-
e*cos(Ek)))+Crc*cos(2*(w+vk))+Crs*sin(2*(w+vk));
double      ik      =      i0      +      IDOT*tk+
Cic*cos(2*(w+vk))+Cis*sin(2*(w+vk));
double  lymbdak_ECEF  =  Omega0  +  (OmegaDot  -
omegaE)*tk-omegaE*toe;
dmat R1(3,3);
R1(0,0) = 1;
R1(0,1) = 0;
R1(0,2) = 0;
R1(1,0) = 0;
R1(1,1) = cos(-ik);
R1(1,2) = sin(-ik);
R1(2,0) = 0;
R1(2,1)=-sin(-ik);
R1(2,2) = cos(-ik);
dmat R3l_ECEF(3,3);
R3l_ECEF(0,0) = cos(-lymbdak_ECEF);
R3l_ECEF(0,1) = sin(-lymbdak_ECEF);
R3l_ECEF(0,2) = 0;
R3l_ECEF(1,0) = -sin(-lymbdak_ECEF);
R3l_ECEF(1,1) = cos(-lymbdak_ECEF);
R3l_ECEF(1,2) = 0;

```

```

R3l_ECEF(2,0) = 0;
R3l_ECEF(2,1) = 0;
R3l_ECEF(2,2) = 1;
dmat R3uk(3,3) ;
R3uk(0,0) = cos(-uk);
R3uk(0,1) = sin(-uk);
R3uk(0,2) = 0;
R3uk(1,0) = -sin(-uk);
R3uk(1,1) = cos(-uk);
R3uk(1,2) = 0;
R3uk(2,0) = 0;
R3uk(2,1)=0;
R3uk(2,2) = 1;
dmat rkk(3,1);
rkk(0,0) = rk;
rkk(1,0) = 0;
rkk(2,0) = 0;
dmat coord = R3l_ECEF*R1*R3uk*rkk;
CoordinatesGps.X = coord(0,0);
CoordinatesGps.Y = coord(1,0);
CoordinatesGps.Z = coord(2,0);
return CoordinatesGps;
}

```

ephemerids.h

```

#ifndef EPHEMERIDS_H
#define EPHEMERIDS_H

```

```

typedef struct

```

```

{
    double X;
    double Y;
    double Z;

} Coordinates;
Coordinates CoordGPS(double t,
    double toe,
    double M0,
    double e,
    double n,
    double a,
    double dn,
    double w,
    double Cuc,
    double Cus,
    double Crc,
    double Crs,
    double Cic,
    double Cis,
    double i0,
    double IDOT,
    double Omega0,
    double OmegaDot,
    double omegaE );

#endif

#include "Kepler.h"

```

Kepler.cpp

```

#include <math.h>

double kepler(double en, double Mk)
{
    double Ek = en*sin(0)+Mk;
    double Ekold = 0;
    while (fabs(Ek- Ekold)>0.000000001 )
    { Ekold = Ek;
      Ek = en*sin(Ek)+Mk;
    }

    return Ek;
}

```

Kepler.h

```

#ifndef KEPLER_H
#define KEPLER_H

double kepler( double en, double Mk);

#endif

```

funct.cpp

```

#include <armadillo>
#include "funct.h"
#include "ephemerids.h"
#include <iostream>
#include <math.h>

using namespace std;

```

```

void simul(double *X,double *Y,double *Z)
{

    long double tstart = (24*2 + 18 - 3)*60*60;
    long double tstop = (24*3 + 6 - 3)*60*60;
    double a=pow(5.28261434555053711e+03,2);
    double toe = 223200000.000e-3; //ms
    double M0 =1.40953592060026917e-01 ;
    double dn = (4.35125286496473862e-12)/(pow(10,-3));
    //rad/ms

    double w= 5.72152208390331540e-01;
    double Cuc =5.78071922063827515e-06;
    double Crc =2.343906250000000000e+02;
    double Crs = 6.560937500000000000e+01;
    double Cic =2.32830643653869629e-09;
    double Cis =-6.56582415103912354e-08;
    double Cus =5.78071922063827515e-06;
    double i0 = 9.49918991207442720e-01;
    double IDOT =1.51077721564943834e-13;
    double Omega0 =1.82433512285772315e+00;
    double OmegaDot = -7.16708425211283236e-12;
    double e= 7.20643904060125351e-04;
    double n = 3.986004418e+14;
    double OMEGA_e = 7.2921151467*pow(10,-5);
    double omegaE=OMEGA_e;
    ofstream f;
    f.open("testCoord.txt");
    f<<"tstart"<<tstart<<endl;

```

```

int k=0;
for ( long double i = tstart; i<=tstop; i+=0.1)
{
// printf("i%12.12lf\n ",i);
Coordinates coord = CoordGPS( i,
                                toe,
                                M0,
                                e,
                                n,
                                a,
                                dn,
                                w,
                                Cuc,
                                Cus,
                                Crc,
                                Crs,
                                Cic,
                                Cis,
                                i0,
                                IDOT,
                                Omega0,
                                OmegaDot,
                                omegaE );

X[k] = coord.X;
Y[k] = coord.Y;
Z[k] = coord.Z;
k++;

```

```

f<<"tstart+i"<<i<<"\t" <<"X="<< coord.X<< "\t";
f<< "Y="<< coord.Y<<"\t";
f<< "Z="<< coord.Z<<endl;

}
f.close();

}

void parser(double *Xmatlab,double *Ymatlab,double
*Zmatlab )
{
ifstream fin("cord.txt");
for (int i=0; i<=432000; i++)
{
fin >> Xmatlab[i];
fin >> Ymatlab[i];
fin >> Zmatlab[i];
}
}

void comparr(double* X, double* Y, double* Z,double* X2,
double* Y2, double* Z2, double nnn, double *srdx, double
*srdy, double *srdz,
double *dxmax, double *dymax,double *dzmax)
{
// cout<< "comparr"<<endl;
double sumdx = 0;
double sumdy = 0;
double sumdz = 0;
int n = nnn;

```



```

double* dx= new double[n];
double* dy= new double[n];
double* dz= new double[n];
//int* wcompar;
// wcompar = new int[nn];

for (int i = 0; i < nnn; i++)
{
    dx[i] = (fabs(X[i]-X2[i]));
    dy[i] = (fabs(X[i]-X2[i]));
    dz[i] = (fabs(X[i]-X2[i]));
    sumdx +=dx[i];
    sumdy +=dy[i];
    sumdz +=dz[i];
    //найдем макс значение
    if (dx[i] > dx[i-1])
    {
        *dxmax = dx[i];
    }
    if (dy[i] > dy[i-1])
    {
        *dymax = dy[i];
    }
    if (dz[i] > dz[i-1])
    {
        *dzmax = dz[i];
    }
}
*srdx = sumdx/nnn;

```

```

        *srdy = sumdy/nnn;
        *srdz = sumdz/nnn;
        delete[] dx;
        delete[] dy;
        delete[] dz;
    }

```

funct.h

```

#ifndef FUNCT_H
#define FUNCT_H
void simul(double *X,double *Y,double *Z);
void comparr(double* X, double* Y, double* Z,double* X2,
double* Y2, double* Z2, double nnn, double *srdx, double
*srdy, double *srdz,
            double *dxmax, double *dymax,double *dzmax);
void parser(double *Xmatlab,double *Ymatlab,double
*Zmatlab );
#endif

```

Test.cpp

```

#include "UnitTest++.h"
#include "Kepler.h"

namespace
{

TEST(OurFirstTest)
{

```

```
double    Result    =    kepler(7.20643904060125351e-
04,0.628525);
// printf("%16.16f",Result);
//0.6289489513181900
CHECK_EQUAL(0.6289489513181900, Result);
}
}
```