Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

**«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт Инженерной Физики и Радиоэлектроники

Кафедра Радиоэлектроники

УТВЕРЖДАЮ

Заведующий кафедрой

\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись инициалы, фамилия

«\_\_\_\_\_» \_\_\_\_\_\_\_\_ 20 \_\_\_ г.

**БАКАЛАВРСКАЯ РАБОТА**

11.03.01 - Радиотехника

Исследование алгоритма вычисления ионосферной поправки для системы Galileo.

Руководитель \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. М. Валиханов

подпись, дата должность, ученая степень

Выпускник \_\_\_\_\_\_\_\_\_ Н. Г. Сюлин

подпись, дата

Красноярск 2022

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc105956785)

[1 Проблема ионосферной задержки 4](#_Toc105956786)

[2 NeQuick 6](#_Toc105956787)

[2.1 Что такое NeQuick, область применения 6](#_Toc105956788)

[2.2 Описание алгоритма NeQuick 7](#_Toc105956789)

[2.3 Установка и использование NeQuick 10](#_Toc105956790)

[2.4 РРРН 15](#_Toc105956791)

[2.5 Функция XYZ2BLH 16](#_Toc105956792)

[2.6 Угол места 18](#_Toc105956793)

[3 Что сделано и что сделать чтобы получить результат 18](#_Toc105956794)

[3.1 Пример работы алгоритма 18](#_Toc105956795)

[3.2 Результаты и анализ 21](#_Toc105956796)

[ЗАКЛЮЧЕНИЕ 22](#_Toc105956797)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc105956798)

[ПРИЛОЖЕНИЕ А 24](#_Toc105956799)

[Программный код, созданный и использованный в процессе работы 24](#_Toc105956800)

[А 1 Значимые функции для загрузки данных с помощью протокола FTP 24](#_Toc105956801)

[Продолжение приложения А 25](#_Toc105956802)

[А 2 Значимые функции для обработки входных данных 25](#_Toc105956803)

[Продолжение приложения А 26](#_Toc105956804)

[Продолжение приложения А 27](#_Toc105956805)

[А 3 Служебные функции 27](#_Toc105956806)

[Продолжение приложения А 28](#_Toc105956807)

[Окончание приложения А 29](#_Toc105956808)

[СПИСОК СОКРАЩЕНИЙ? 30](#_Toc105956809)

## ВВЕДЕНИЕ

На пути к приемникам, расположенным на поверхности Земли или вблизи нее, сигналы спутников ГНСС проходят через ионосферу. Свободные электроны, находящиеся в этой оболочке атмосферы, влияют на распространение сигналов, изменяя их скорость и направление движения. Это приводит к задержке в прибытии модулированных составляющих сигналов (из которых впоследствии получают измерения псевдодальности) и в несущих фазах волн сигналов (влияющих на измерения несущей фазы). Ионосфера - диспергирующая среда для радиосигналов, поэтому при одновременном проведении измерений на двух частотах, передаваемых спутником, большая часть влияния ионосферы может быть исключена. Однако одночастотные устройства, такие как большинство навигационных транспортных средств и портативных приемников, не могут позволить себе роскошь. Эти устройства должны полагаться на модели коррекции в одночастотном режиме. Коэффициенты для такой модели включены в навигационные сообщения, передаваемые со всех спутников *GPS*. Известный как алгоритм коррекции ионосферы или алгоритм Клобухар, он удаляет по меньшей мере 50% влияния ионосферы в сигналах.

Распространение радиоволн сигналов ГНСС зависит от атмосферы Земли и характеристик среды, окружающей приемник. Системы ГНСС основаны на трансляции радиоволн в микроволновой области (в основном все работают в так называемом *L*-диапазоне, хотя некоторые новые системы, такие как Индийская региональная навигационная спутниковая система, как ожидается, будет вести вещание в *S*-диапазоне). Эти электромагнитные сигналы могут страдать от ряда помех. При рассмотрении этих эффектов, мы можем разделить атмосферу Земли на две части: электрически нейтральную атмосферу (в первую очередь самую низкую часть - тропосферу), где основной эффект групповой задержки навигационных сигналов происходит из-за частиц водяного пара и газовых компонентов сухого воздуха, который для микроволнового излучения является недисперсным (зависит от частоты); и ионосферу, ионизированную часть атмосферы. Локальное окружение может повлиять на навигационный сигнал различным образом, приводя к созданию препятствий для свободного прохождения сигнала или полной его блокировке (например, растительностью или такими препятствиями, как здания). Сигнал также может рассеяться и стать многолучевым из-за отражения и дифракции на окружающие предметы. В этой дипломной работе будет рассмотрено влияние ионосферы на сигналы ГНСС и решение данной проблемы, предложенное командой *Galileo*.[1]

## Проблема ионосферной задержки

### **Фазовая и групповая скорость**

Фазовая скорость распространения электромагнитной волны с частотой *f* и длиной волны *λ* равна

(1)

С этой скоростью распространяются фазы несущих волн *GPS* и ГЛОНАСС. Для групповых волн частоты существенно другие, и скорость распространения для группы определяется как

(2)

Соотношение между групповой и фазовой скоростью можно установить через дифференцирование (1):

(3)

откуда

(4)

после подстановки получаем:

(5)

Скорость распространения волны в среде зависит от показателя  
преломления среды

, (6)

где *c* – скорость света в вакууме. Применяя выражение 6 к фазовой и групповой скорости, находим

*,*  (7)

где *nφ* и *ng* – соответствующие показатели преломления. Можно показать, дифференцируя *vφ*, через связь *λ* и подстановку в (5), что

(8)

Для тропосферы фазовый и групповой показатели преломления не зависят от частоты, поэтому *nφ = ng*.[2]

### **1.2 Ионосферная задержка**

Ионосфера, простирающаяся от высоты около 50 км до примерно 1000 км над Землей, является областью ионизованных газов (свободных электронов и ионов). Ионизация вызывается солнечной радиацией, и состояние ионосферы определяется преимущественно интенсивностью солнечной активности. Ионосфера состоит из слоев (называемых слоями *D, E, F*1 и *F*2) на различных высотах, каждый со своими скоростями образования и потери свободных электронов. Пик электронной плотности (количество электронов в кубическом метре) приходится на диапазон высот в 250-400 км (слой *F*2).

Физические характеристики ионосферы изменяются ото дня к ночи в широких пределах. Когда Солнце восходит, его ультрафиолетовое излучение начинает разлагать молекулы газа (в основном *H*2 и *He* на больших высотах, а на меньших высотах – *O*2 и *N*2) на ионы и свободные электроны. Пик электронной плотности наступает около 2 часов после местного полдня, а затем плотность начинает уменьшаться. Ночью ионизация не происходит, и ионы и электроны находят друг друга и рекомбинируют, уменьшая количество свободных электронов. Наблюдаются значительные изменения в зависимости от времени года и фазы 11-летнего цикла солнечной активности. (Текущий солнечный цикл начался в 1995 г. и имел пик в 2000 г.) Может также проявляться значительная изменяемость ото дня ко дню, в зависимости от солнечной активности и геомагнитных нарушений. Есть также непредсказуемые краткосрочные влияния и локальные аномалии (подвижные ионосферные нарушения).

Скорость распространения радиосигналов в ионосфере зависит от числа свободных электронов на их пути, определяемых величиной полной электронной концентрации *TEC* (*Total Electron Content*). Это число электронов, содержащихся в столбе сечением в 1 м2, простирающемся от приемника до спутника:

(9)

где *ne*(*s*) - переменная электронная плотность вдоль пути сигнала, а интегрирование производится вдоль пути сигнала от спутника *S* к приемнику *R*. Длина пути через ионосферу самая короткая в направлении зенита, и поэтому *TEC* имеет наименьшее значение в вертикальном направлении (*TECV*). Величина *TEC* измеряется в единицах *TECU* (*TEC Units*), определяемых как 1016 электронов/м2. Обычно *TECV* изменяется между 1 и 150 *TECU*. В данном месте и в данное время *TECV* может изменяться на 20-25% от его среднемесячного значения. Современные модели ионосферы не обеспечивают адекватное представление изменений в *TEC* между сутками.

Ионосфера обычно имеет спокойное поведение в умеренных широтах, но может флуктуировать вблизи экватора и магнитных полюсов. Район с наивысшей ионосферной задержкой лежит в пределах ±20° от магнитного экватора. Солнечные вспышки и последующие магнитные бури могут создавать обширные и быстрые флуктуации в фазе несущей (называемые сцинтилляциями) и в амплитуде (называемые затуханиями) сигналов СРНС. Это явление, хотя и кратковременное, и нечастое в средних широтах, может создавать трудности в непрерывном отслеживании сигналов в полярных и экваториальных районах.

Ионизованный газ представляет собой диспергирующую среду для радиоволн. Фазовый показатель преломления *nφ* для радиоволны с частотой *f:*

(10)

где *ne* – плотность электронов. Показатель преломления немного меньше 1(≈0.99998 для центральной частоты *L*1), и фазовая скорость *GPS* в ионосфере превышает скорость света в вакууме на достаточно большую для точного позиционирования величину. Из выражения для показателя преломления можно вычислить фазовую задержку *Iτ* (в секундах), которую испытывает сигнал при его распространении через ионосферу:

(11)

Фазовая задержка отрицательная, то есть наблюдается опережение по фазе. Фазовое опережение *IΦ* (в метрах) прямо пропорционально количеству электронов на пути сигнала, то есть

(12)

Из (8) можно определить групповой показатель преломления как

(13)

Тогда ионосферная задержка (в метрах) для псевдодальности, измеренной по кодам, будет равна

(14)

Величина фазового опережения часто выражается в циклах, то есть соответствующем числе длин волн, содержащихся в *IФ*. Поскольку *λ=с/f*, то

(15)

Формулы (12) и (14) дают значение ионосферной задержки в метрах, при условии, что *f* дается в герцах, а *TEC* – в электронах/м2. В последующем будем обозначать групповую ионосферную задержку просто как *I*, а фазовую задержку как – *I*, при этом возможно указание на ее размерность (в метрах *IФ* , *IР*, в циклах *Iφ* или секундах *Iτ*).

Таблица 1 – Ионосферная поправка в дальность (в метрах)

|  |  |  |
| --- | --- | --- |
| Частоты | *TEC* = 1016 [эл/м2] | *TEC* = 1018 [эл/м2] |
| 100 МГц | 40000 | 4000 |
| 400 МГц | 2500 | 250 |
| FL1 | 0,26 | 26 |
| FL2 | 0,16 | 16 |
| 2 ГГц | 0,1 | 10 |
| 10 ГГц | 0,004 | 0,4 |

В таблице 1 приводятся ионосферные задержки для нескольких частот и для двух значений величины *TEC*. Видно, что на диапазон частот *L*1 и *L*2 ионосфера оказывает большое влияние. Обычно величина *TEC* находится в пределах от 1016 до 1018.[2]

## 2 NeQuick

### **2.1 Что такое NeQuick, область применения**

*NeQuick* это трехмерная времязависимая модель плотности электронов в ионосфере. Она основана на эмпирическом климатическом представлении ионосферы, которое предугадывает среднюю месячную плотность электронов, опираясь на значения, связанные с солнечной активностью: 12 – месячный сглаженный номер солнечных пятен, месяц, ширина, долгота, высота и время (*UT*). Модель *NeQuick* была адаптирована для одночастотных приемников Galileo для коррекций в реальном времени для того, чтобы получать прогнозы в реальном времени на основе одного входного параметра – эффективного уровня ионизации *Az*, который определяется тремя коэффициентами, передаваемыми в навигационном сообщении.

*Galileo* это европейская глобальная навигационная спутниковая система, предоставляющая высокоточные глобальные навигационные данные для гражданского использования. *Galileo*, как и современная *GNSS* в целом, основана на передаче электромагнитных сигналов в полосе частот *L*. Эти спутниковые сигналы страдают от ряда нарушений при распространении через земную атмосферу, включая ионосферную задержку. [3]

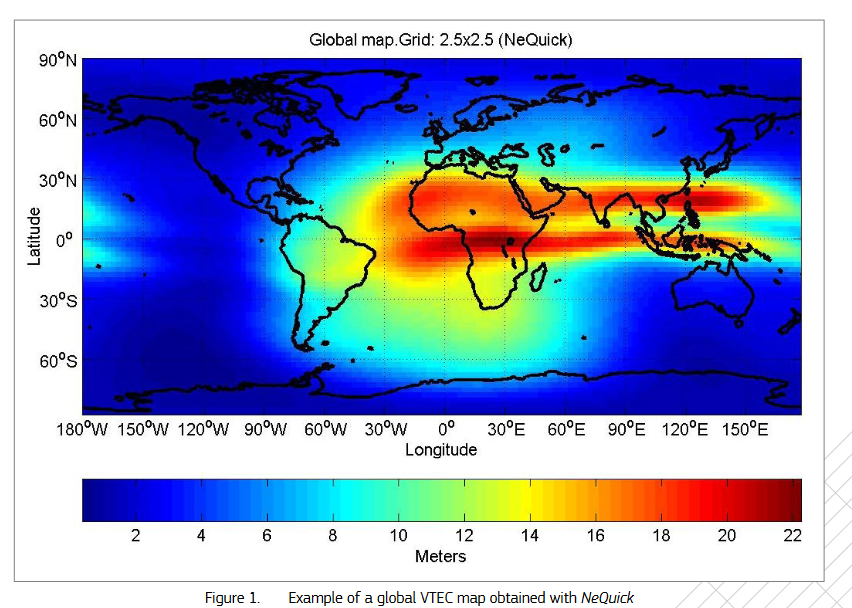


Рисунок 1 – Пример глобальной карты *VTEC*, полученной с помощью *NeQuick*

В номинальных условиях приемник *Galileo* декодирует навигационное сообщение *Galileo* от *N* спутников *Galileo* одновременно (*N* > 1), которые получат в данный отрезок времени одни и те же широковещательные ионосферные коэффициенты для каждого спутника. однако, когда ионосферные параметры обновляются, учитывая, что *up-link* сообщения спутники не всегда синхронизированы из-за видимости спутников до станций восходящей линии связи (*uls*), может случиться так, что некоторые спутники транслируют недавно обновленные коэффициенты, в то время как другие продолжают транслировать предыдущую партию коэффициентов. В такой ситуации любой приемник, расшифровывающий навигационное сообщение в любой данный момент, не имеет средств чтобы определить, какие коэффициенты новее, а какие старше. Кроме того, нет возможности определить, какие коэффициенты применять для коррекции псевдодальностей разных спутников. Тем не менее поправка все равно будет достигаться в статистическом смысле, независимо от принятого решения.

### **2.2 Описание алгоритма NeQuick**

Для реализации ионосферного алгоритма для приемников *Galileo* на одной частоте следующие должны быть выполнены:

Для каждого соединения спутник – приемник:

Получить оценку местонахождения приемника (, , *h*)i, спутника (, , *h*)j, и время (время суток и месяц)

Получить *MODIP*U приемника, используя , .

Получить эффективный уровень ионизации *Az*U, используя (УРАВНЕНИЕ 2 В КНИГЕ) с *MODIP*U и коэффициенты передачи (ai0, ai1, ai2)

Вызвать *NeQuick G STEC* для пути от (*x, y, z*)i до (*x, y, z*)j.

Для каждой точки в пути вызвать *NeQuick* для получения плотности электронов с *Az*U, времени и месяца

End

Интегрировать *STEC* для всех точек пути

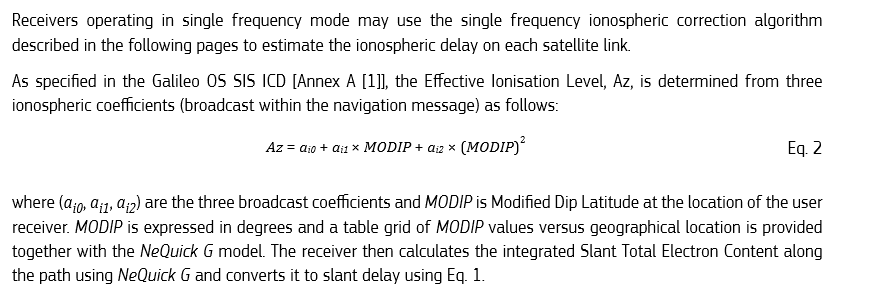
Получить коррекцию преобразуя *STEC* в кодовую (?) задержку используя (УРАВНЕНИЕ 1 В КНИГЕ) для соответствующей частоты

Наложить коррекцию для выбранного соединения

End

Приемники, работающие в режиме одной частоты, могут использовать алгоритм коррекции, описанный ниже для прогнозирования ионосферной задержки для каждого подключения к спутнику.

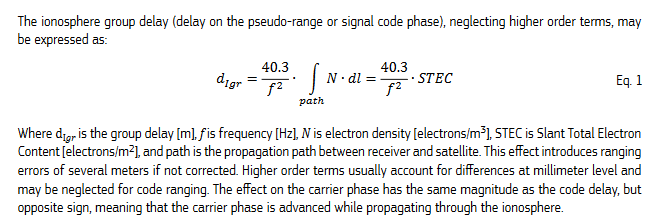
По определению в *Galileo OS SIS ICD*: эффективный уровень ионизации *Az* можно получить с помощью трех ионосферных коэффициентов (передаются в навигационных сообщениях):



где *ai0, ai1, ai2* – упомянутые коэффициенты, *MODIP* – *Modified Dip Latitude* в расположении приемника. *MODIP* измеряется в градусах, таблица соответствия сетки значений *MODIP* с географическими местоположением поставляется вместе с моделью *NeQuick G*.

Приемник затем считает *STEC* на протяжении всего пути и переводит их в наклонную задержку используя УРАВНЕНИЕ 1 В КНИГЕ.

Групповая ионосферная задержка (задержка диапазона или фазы кода сигнала), пренебрегая условия более высокого порядка, может быть получена следующим образом:



где *d1gr* – это групповая задержка (м), *f* – частота (Гц), *N* – плотность электронов (электрон / м3), *path* – путь от приемника до спутника. Этот эффект создает диапазонные ошибки в несколько метров. Условия высшего порядка обычно составляют миллиметры и ими можно пренебречь. Эффект несущей фазы имеет такую же магнитуду, как и кодовая задержка, но с противоположным знаком. Что означает несущая фаза увеличивается по мере распространения в ионосфере.

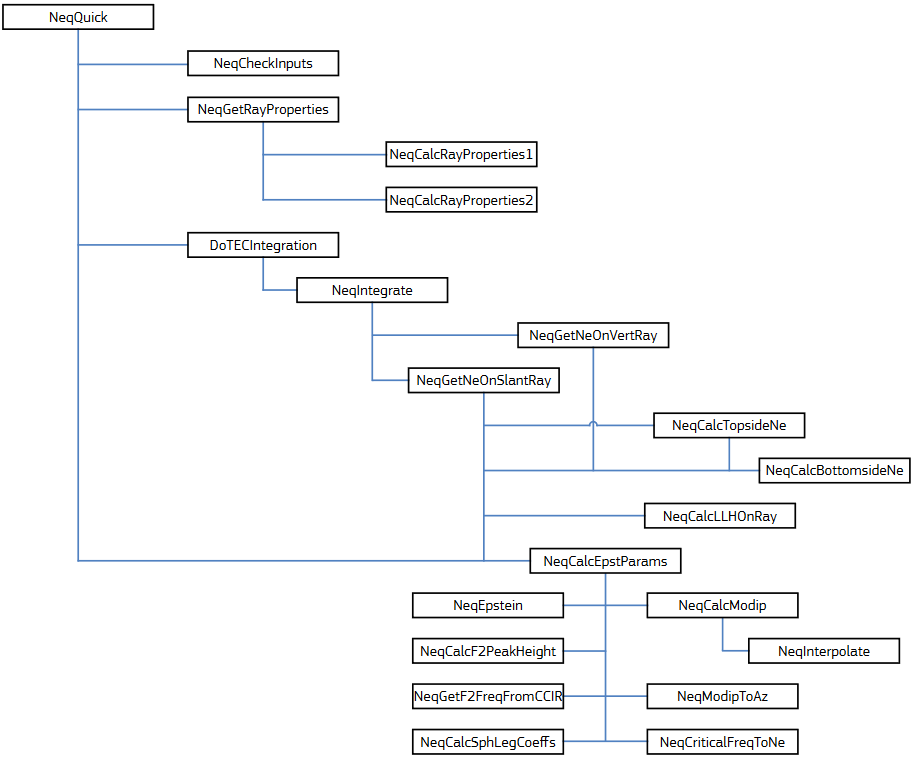


Рисунок N – Иерархическая схема *NeQuick*

На рисунке N представлена иерархическая структурная схема *NeQuick*. Краткое описание основных функций представлено в таблице N

Таблица 2 – Назначение каждой функции – состовляющей *NeQuick*

|  |  |
| --- | --- |
| Название функции | Назначение |
| *NeQuick* | Основная функция, возвращает *TEC* между 2мя точками |
| *NeqCheckInputs* | Проверяет введенные значения |
| *NeqGetRayProperties* | Вычисляет параметры луча и проверяет последний на правильность |
| *DoTECIntegration* |  |
| *NeqCalcEpstParams* | Вычисляет параметры ионосферы |
| *NeqGetF2FreqFromCCIR* | Возвращает *foF*2, вычисленный из *CCIR* файлов |
| *NeqCalcF2PeakHeight* | Вычисляет пик слоя *F*2 |
| *NeqCalcBottomSideNe* | Вычисляет количество электронов на определенной высоте в нижней части ионосферы под пиком *F*2 |

[3]

### **2.3 Установка и использование NeQuick**

*NeQuick* является ПО с открытым исходным кодом, скачать его можно на сайте Европейского Сервисного Центра ГНСС (<https://www.gsc-europa.eu/support-to-developers/nequick-g-source-code>). Реализация алгоритма на ЯП *С* стандарта 2011 года разделена на несколько частей:

Библиотека *NeQuick G JRC*

Тестовая программа

Несколько директорий с говорящими названиями: расширения, связанные с .*c* и .*h* файлами, текстовые файлы, файлы *MODIP* и *CCIR* соответствующие изначальному распространению *NeQuick*, *make* файлы, конфигурационные файлы (.*cfg*) и *perl* файлы (.*pl*).

Сборка осуществляется с помощью *cmake* и конфигурации в *Makefile*, можно выбрать один из нескольких вариантов конечной сборки, для работы в *Windows* лучше всего подойдет версия для *Visual Studio* 2015, конечный проект будет совместим и с более поздними версиями *Visual Studio*. После сборки будет создан .*exe* файл, который можно запустить из командной строки. Это непосредственно запуск *NeQuick*.

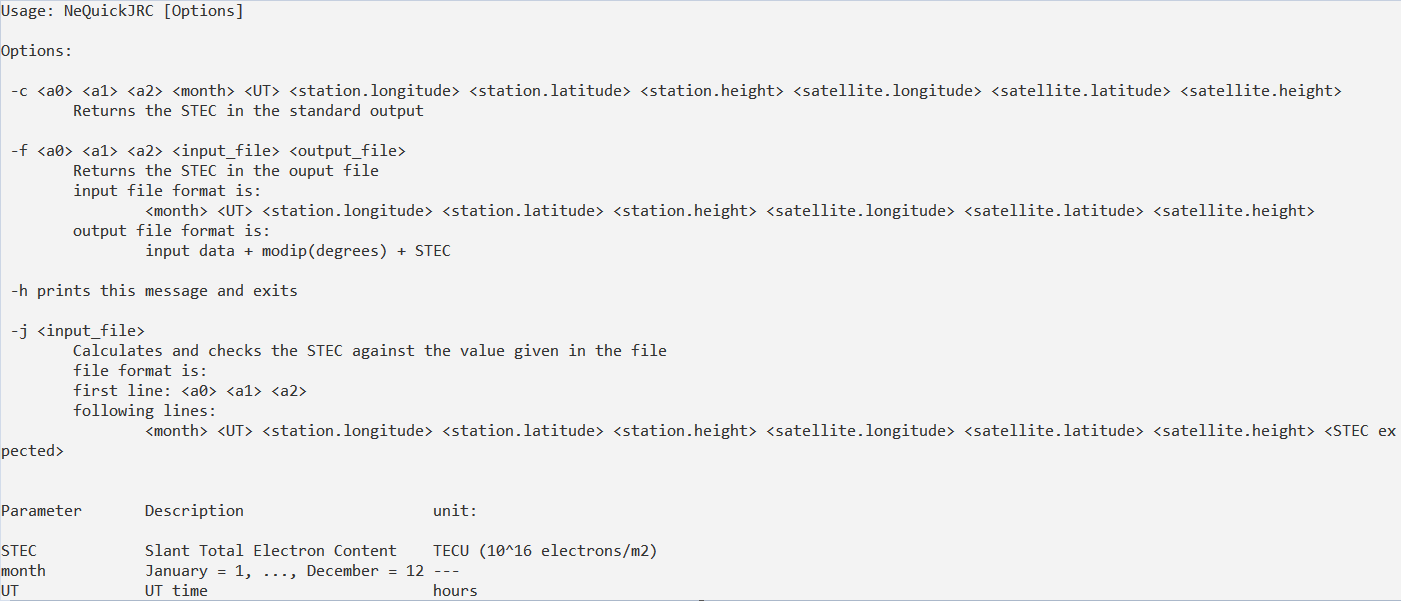


Рисунок N – Инструкции по запуску *NeQuick*

Как видно из рисунка выше, алгоритм может работать в 2х режимах. Первый вызывается параметром -c и на входе требует следующие данные:

Таблица N – Входные параметры

|  |  |  |
| --- | --- | --- |
| Входной параметр | Описание | Единица измерения |
| АЛЬФАi0 | Эффективный уровень ионизации 1 порядка | Sfu |
| АЛЬФАi1 | Эффективный уровень ионизации 2 порядка | Sfu / град |
| АЛЬФАi2 | Эффективный уровень ионизации 3 порядка | *Sfu* / град2 |
|  | Геоцентрическая широта приемника | град |
|  | Геоцентрическая долгота приемника | град |
| h1 | Геоцентрическая высота приемника | метры |
|  | Геоцентрическая широта спутника | град |
|  | Геоцентрическая долгота спутника | град |
| h2 | Геоцентрическая высота спутника | метры |
| *UT* | *UT* время | часы |
| месяц | Целое значение от 1 до 12 | без единиц измерения |

*Sfu* - единица солнечного потока, , местонахождение приемника и спутника должно быть в формате эллипсоидных координат *WGS*-84.

Второй режим, вызывающийся параметром - *f* принимает тот же набор параметров, но разница в количестве – если первый вариант работает по одному набору, то второй читает файл, где каждая строка соответствует 1 набору, поэтому в качестве параметров в консоли необходимо указать коэффициенты солнечной активности, входной файл и название выходного файла, который *NeQuick* создаст в текущей директории.

Коэффициенты солнечной активности, координаты станции наблюдения (которая и сняла эти коэффициенты) и время находятся в файлах *rinex*.

*RINEX* — это формат обмена данными для файлов исходных данных спутниковых навигационных приёмников. Он позволяет пользователям производить пост-обработку полученных данных для выполнения более точных вычислений — обычно с помощью других данных, неизвестных приемнику, например за счёт применения более точной модели атмосферных параметров в момент измерений. Выглядит такой формат следующим образом:

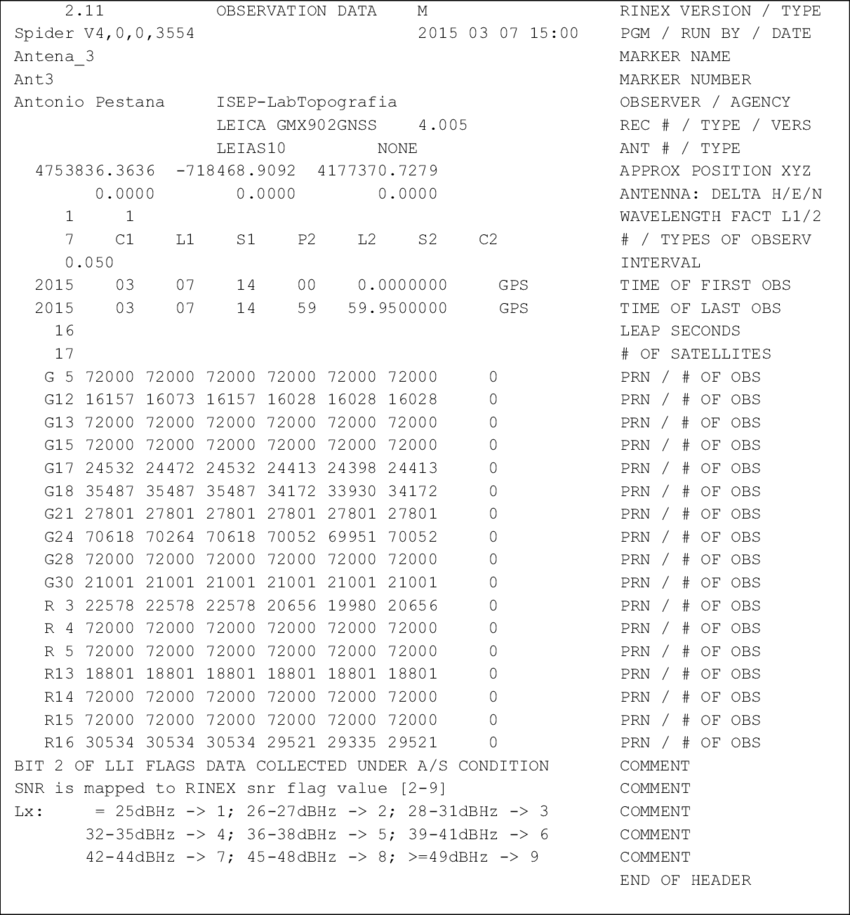


Рисунок N – пример заголовка *rinex* файла

Координаты спутника же передаются в другом формате – *sp*3.

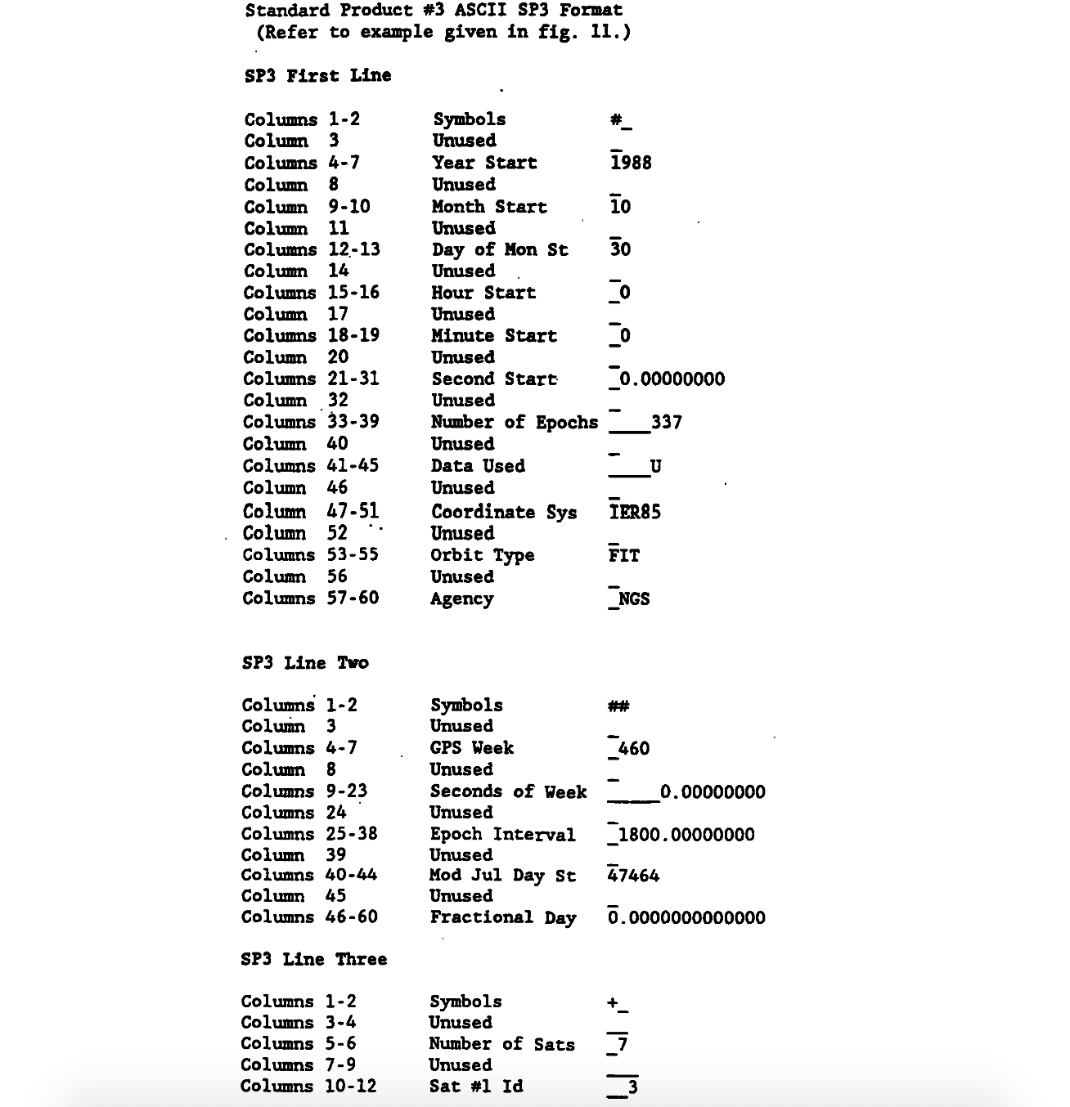


Рисунок N – Пример описания формата *sp*3

Параметр - *h* выдает сообщение на рисунке N. Последний параметр -*j* нужен для проверки получаемых *STEC* единиц – в нем указывается упомянутый выше набор данных и предполагаемые значения *STEC*.

Данные из *sp*3 и *rinex* необходимо обработать, т.к. *NeQuick* требует определенный формат данных и, как уже было упомянуто, систему координат *WGS*-84. Координаты в источниках же являются обычными *X, Y, Z*. Для этого требуется сначала обработать координаты спутника в программе *PPPH* (см. пункт 2.4), затем перевести их и положение станции наблюдения в широту, долготу и высоту (пункт 2.5). Также требуется соблюсти условия угла места (пункт 2.6). Общий алгоритм действий для получения *STEC* из исходных данных представлен на рисунке N.



Рисунок N – Алгоритм действий для получения STEC из входных данных

Результатом работы алгоритма являются значения *STEC* (*Slant Total Electron Content* – полное наклонное содержание электронов), измеряемые в *TECU* (*total electron content unit*), которые могут быть переведены в ионосферную задержку используя УРАВНЕНИЕ 1 В КНИГЕ.

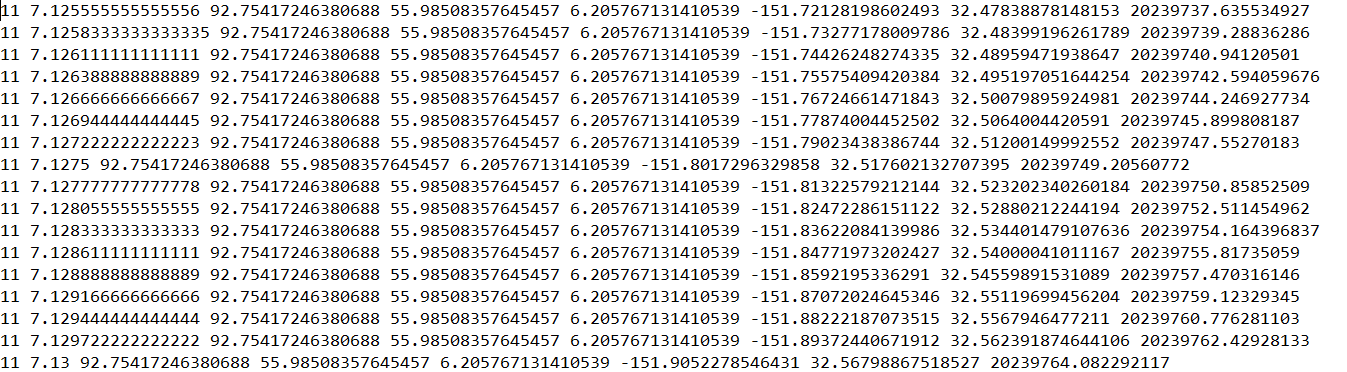


Рисунок N – Пример входного файла для *NeQuick*

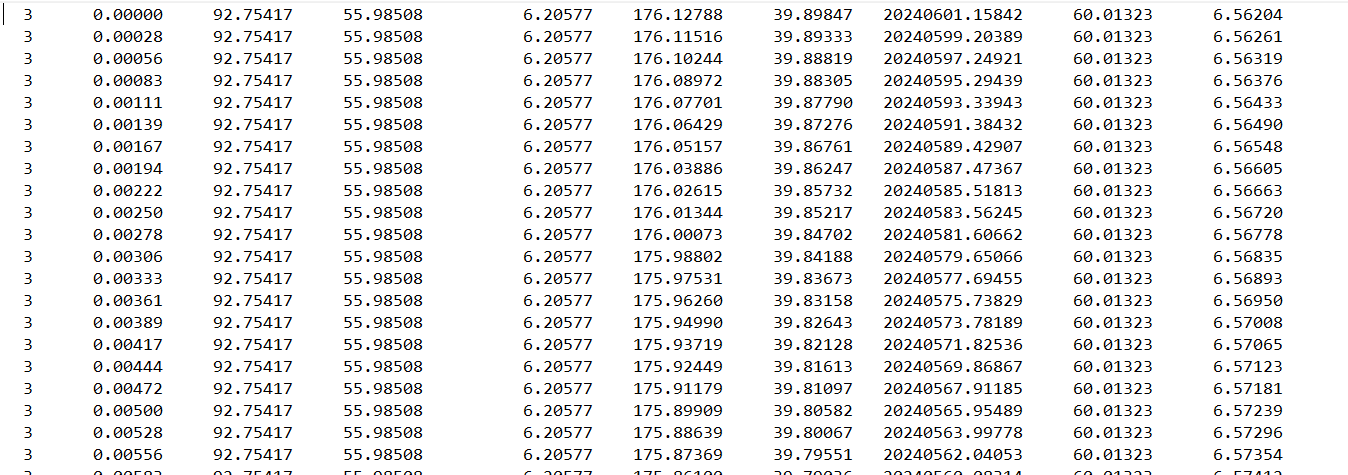


Рисунок N – Пример выходного файла *NeQuick*

Следует отметить, что *PPPH* требуется не только *sp*3 файлы, но и данные наблюдений (.*o*), данные о передающих антеннах (.*atx*)

### **2.4 РРРН**

PPPH – ПО для «точного точечного позиционирования» (Precise Point Positioning). Программа написана на Matlab и имеет точность до миллиметров, создана для анализа координат спутников различных навигационных систем включая GPS, GLONAS, BeiDou и Galileo. Есть графический пользовательский интерфейс, показан на рисунке N.

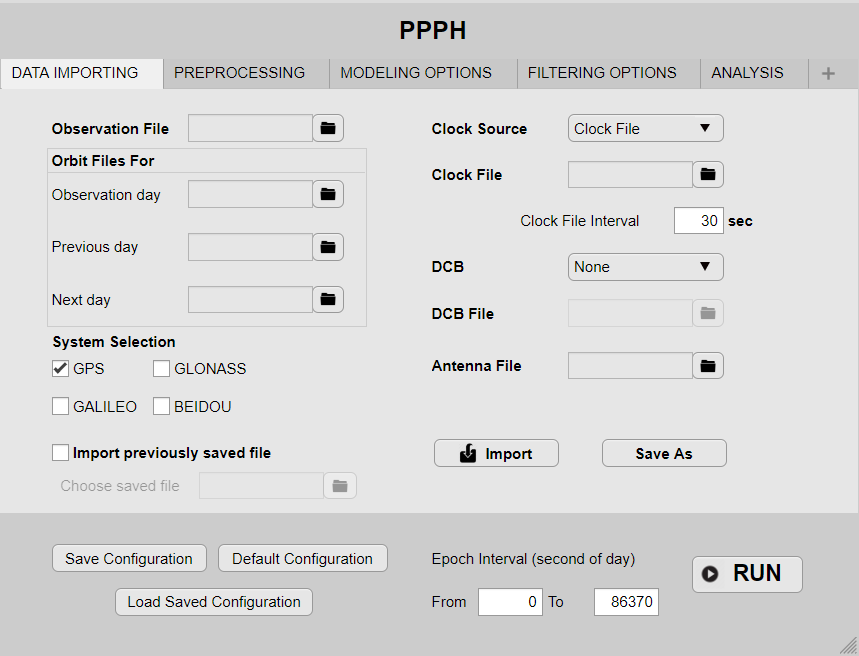


Рисунок N – Интерфейс программы PPPH

Для начала работы необходимо указать путь файла наблюдений, 3 sp3 файла в день наблюдения, день до и после, atx файл, содержащий информацию о передающей антенне. Clock file указывать необязательно, т.к. необходимая информация о времени обычно находится в sp3. После нажатия Import программа проверяет введенные файлы и проводит предобработку – убирает очевидные ошибки, скольжения циклов и т.д. Далее происходит коррекция ошибок временной шкалы и орбиты спутника, тропосферной задержки и, среди всего прочего, учитывается поворот Земли. Последний пункт важен для этой работы.

### **2.5 Функция XYZ2BLH**

Координаты в sp3 и rinex файлах имеют вид xyz координат, PPPH выводит положения спутника в этом же виде, однако NeQuick требуется WGS-84 формат: долгота, широта и высота (BLH). В данной работе рассматривалось 2 варианта алгоритма для перевода координат – Российский ГОСТ и рекомендованный разработчиками Galileo. Был выбран последний, т.к. первый вариант при конвертации определенных значений уходил в вечный цикл, что может привести к некорректным результатам.

Реализация использованного алгоритма перевода координат на Python:

def xyz\_to\_blh(x: float, y: float, z: float) -> tuple[float, float, float]:

a = 6378137.0 # in meters

b = 6356752.314245 # in meters

f = (a - b) / a

f\_inv = 1.0 / f

e\_sq = f \* (2 - f)

eps = e\_sq / (1.0 - e\_sq)

p = sqrt(x \* x + y \* y)

q = atan2((z \* a), (p \* b))

sin\_q = sin(q)

cos\_q = cos(q)

sin\_q\_3 = sin\_q \* sin\_q \* sin\_q

cos\_q\_3 = cos\_q \* cos\_q \* cos\_q

phi = atan2((z + eps \* b \* sin\_q\_3), (p - e\_sq \* a \* cos\_q\_3))

lam = atan2(y, x)

v = a / sqrt(1.0 - e\_sq \* sin(phi) \* sin(phi))

h = (p / cos(phi)) - v

lat = degrees(phi)

lon = degrees(lam)

return lat, lon, h

Алгоритм перевода координат по ГОСТу:

def xyz2blh\_gost(x: float, y: float, z: float) -> tuple[float, float, float]:

a, e = 6378137.0, 0.0167 # большая полуось и эксцентриситет

D = (x \*\* 2 + y \*\* 2) \*\* 0.5

if D == 0:

B = pi \* z / (2 \* abs(z))

L = 0

H = z \* sin(B) - a \* (1 - e \*\* 2 \* sin(B) \*\* 2) \*\* 0.5

else:

la = abs(asin(y / D))

if y < 0 and x > 0: L = 2 \* pi - la

elif y < 0 and x < 0: L = pi + la

elif y > 0 and x < 0: L = pi - la

elif y > 0 and x > 0: L = la

elif y == 0 and x > 0: L = 0

elif y == 0 and x < 0: L = pi

if z == 0:

B = 0

H = D - a

else:

r = sqrt(x \*\* 2 + y \*\* 2 + z \*\* 2)

c = asin(z / r)

p = e \*\* 2 \* a / (2 \* r)

s1 = 0

counter = 0

while counter < 100:

b = c + s1

s2 = asin(p \* sin(2 \* b) / sqrt(1 - e \*\* 2 \* sin(b) \*\* 2 \* b))

d = abs(s2 - s1)

if d < 1e-4:

B = b

H = D \* cos(B) + z \* sin(B) - a \* sqrt(1 - e \*\* 2 \* sin(B) \*\* 2)

break

else:

s1 = s2

counter += 1

else:

return None, None, None

return degrees(B), degrees(L), H

### **2.6 Угол места**

Угол места – это угловая высота объекта над истинным горизонтом. Если значение угла места положительное – значит наблюдаемый объект находится над наблюдателем, в противном случае наблюдатель смотрит в Землю. В этой работе важно, чтобы угол места от наблюдательной станции до спутника был положительный. Реализация на Python:

def calc\_um(sat\_cords: list[float, float, float], station\_cords[float, float, float]) -> float:

rang = sqrt(

(sat\_cords[0]-station\_cords[0])\*(sat\_cords[0]-station\_cords[0]) + \

(sat\_cords[1] - station\_cords[1])\*(sat\_cords[1] - station\_cords[1]) + \

(sat\_cords[2] - station\_cords[2])\*(sat\_cords[2] - station\_cords[2])

)

kx = (sat\_cords[0] - station\_cords[0]) / rang

ky = (sat\_cords[1] - station\_cords[1]) / rang

kz = (sat\_cords[2] - station\_cords[2]) / rang

um = asin(

(kx \* station\_cords[0] + ky \*station\_cords[1]+kz \* station\_cords[2])/\

sqrt(

station\_cords[0] \* station\_cords[0] + \

station\_cords[1] \* station\_cords[1] + \

station\_cords[2] \* station\_cords[2]

)) \* 180.0 / pi

return um

Реализация на Matlab:

function um = calcUm( crdSat, xyzPos )

Rang = sqrt( (crdSat(1) - xyzPos(1))\*(crdSat(1) - xyzPos(1)) + (crdSat(2) - xyzPos(2))\*(crdSat(2) - xyzPos(2)) + (crdSat(3) - xyzPos(3))\*(crdSat(3) - xyzPos(3)) );

kx = (crdSat(1) - xyzPos(1))/Rang;

ky = (crdSat(2) - xyzPos(2))/Rang;

kz = (crdSat(3) - xyzPos(3))/Rang;

um = asin( (kx\*xyzPos(1) + ky\*xyzPos(2) + kz\*xyzPos(3)) / sqrt( xyzPos(1)\*xyzPos(1) + xyzPos(2)\*xyzPos(2) + xyzPos(3)\*xyzPos(3)))\*180.0/pi;

end

## 3 Что сделано и что сделать чтобы получить результат

### **3.1 Пример работы алгоритма**

Для проверки и демонстрации работы рациональнее будет запустить модель в режиме чтения файла, чтобы получить сразу достаточное количество измерений для построения графика т.к. по одиночному значению трудно судить о физичности результатов. Исходя из информации, ранее описанной в данной работе, следует ожидать, что максимальное число концентрации электронов в ионосфере будет в полдень т.к. солнце в это время имеет наибольшую видимость и большие значения солнечной радиации, вероятно, будут присутствовать именно в это время. Аналогично, ночью, когда солнечная активность практически на нуле, следует ожидать минимальные значения концентрации электронов. Ожидаемый график представлен на рисунке N.

Рисунок N – Ожидаемый график

Данные о спутниках и наблюдениях можно взять из архива NASA, файлы загружались по протоколу FTP, т.к. требовалось обработать большое количество файлов и была нецелесообразно загружать файлы вручную, а такой метод передачи файлов автоматизировать программно легче, чем HTTP, не смотря на устарелость протокола FTP. Для проверки были взяты следующие файлы:

- CHPI00BRA\_R\_20200010000\_01D\_EN.rnx: необходим для получения коэффициентов солнечной активности, даты и положения станции

- igs20864.sp3: данные о положении спутника в день наблюдения

- igs20863.sp3: данные о положении спутника за день наблюдения

- igs20865.sp3: данные о положении спутника в следующий день после наблюдения

- igs14\_1935.atx: информация о передающей антенне

- ISTA00TUR\_R\_20171910000\_01D\_30S\_MO.o: информация о наблю-дениях, не требуется для целей этой работы, однако PPPH требует его наличия

График полученных значений:

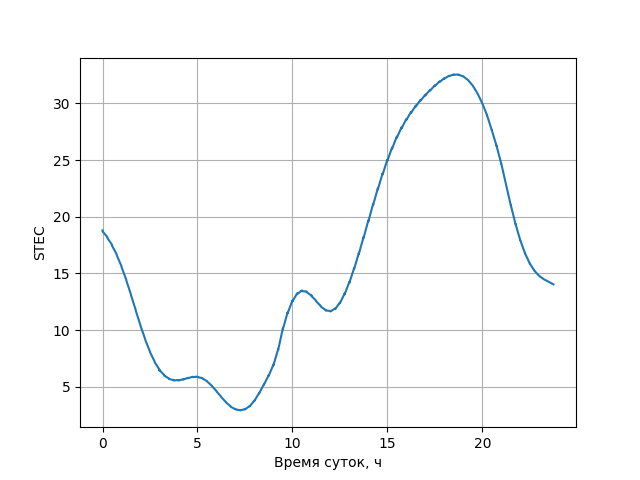


Рисунок N – График зависимости STEC от времени суток

На данном рисунке видно, что максимум находится в точке, соответствующей 19 часам. Это объясняется тем, что станция наблюдения находится в Бразилии:

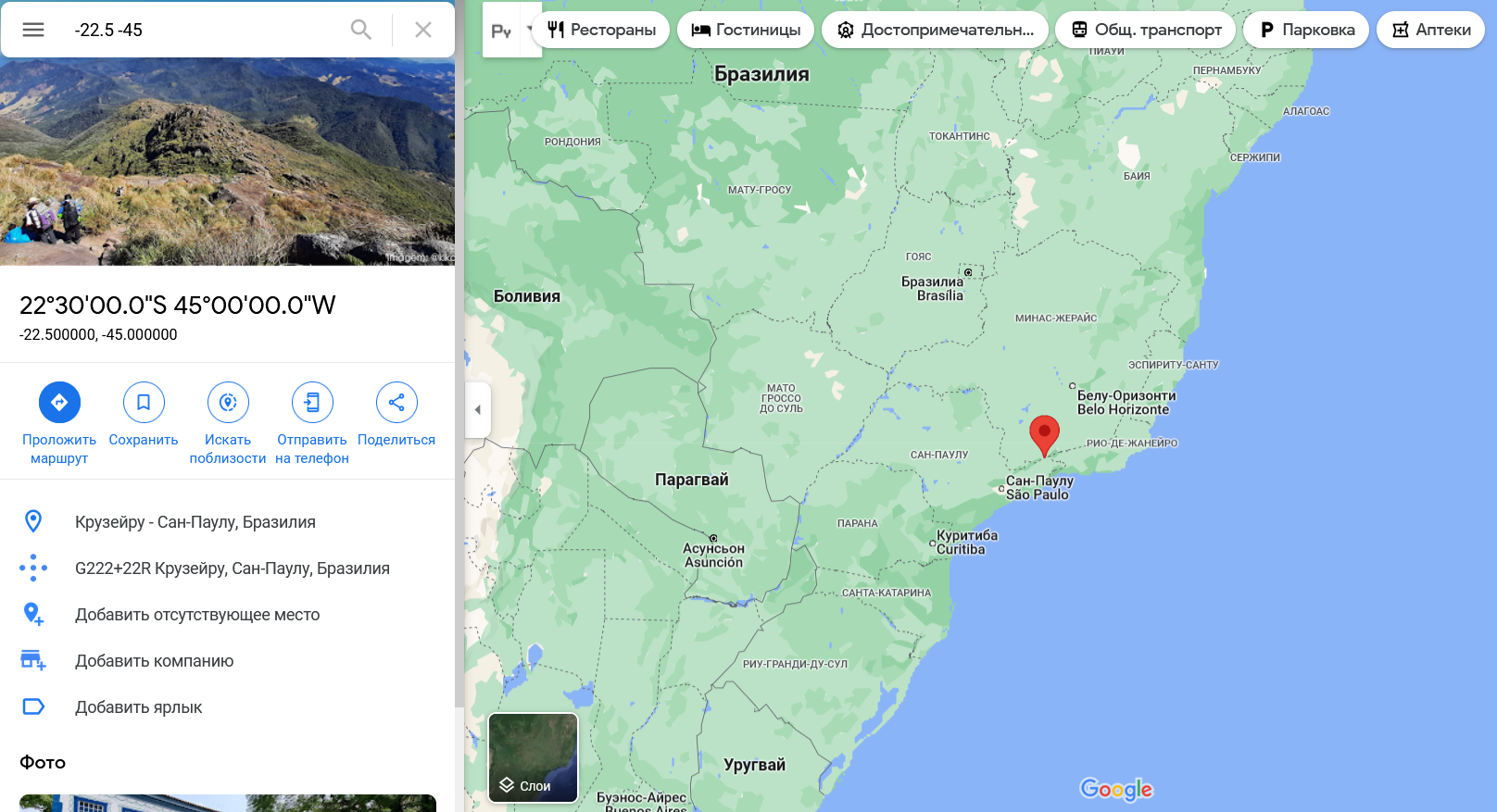


Рисунок N – Местоположение станции, координаты взяты из rinex

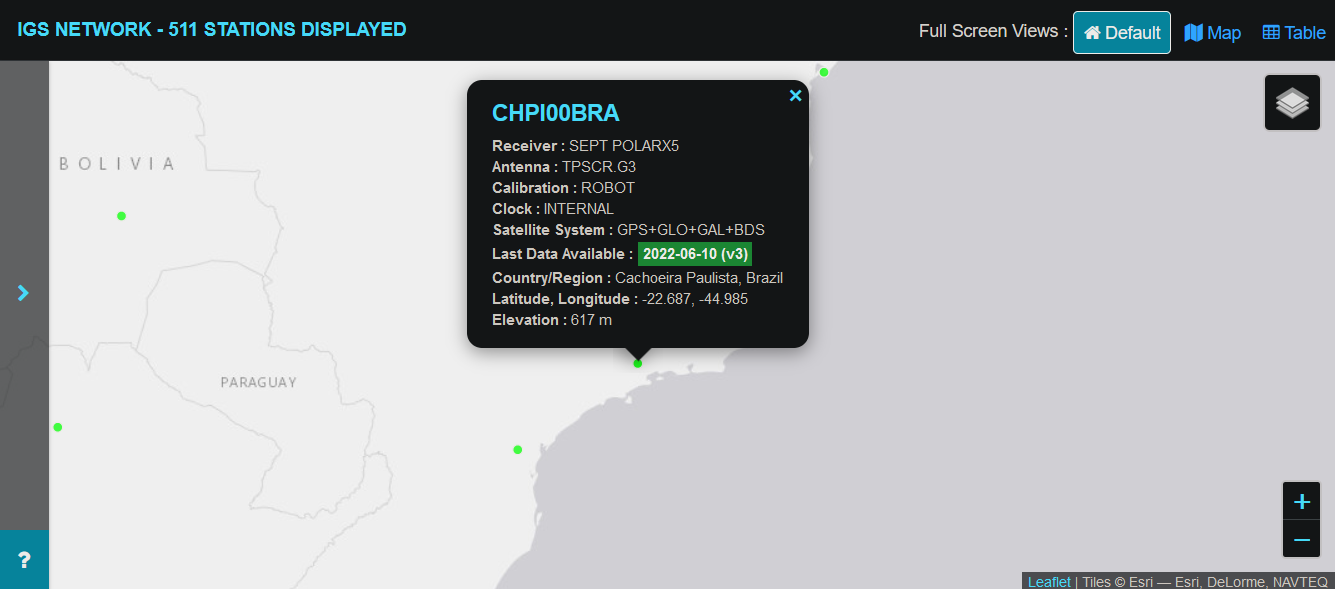


Рисунок N -Изображение с сайта igs.org

Как видно из рисунков N и N, в этом месте действительно есть станция наблюдения, разница с Гринвичем (NeQuick использует UTC, в котором часовой пояс является нулевым меридианом) как раз составляет -7 часов, после корректировки видно, что максимум и вправду находится в полудне:

Рисунок N – Скорректированный график STEC

Часовой пояс – это не единственная проблема, с которой можно столкнуться при получении результатов, на рисунке N представлен результат с другим набором данных:

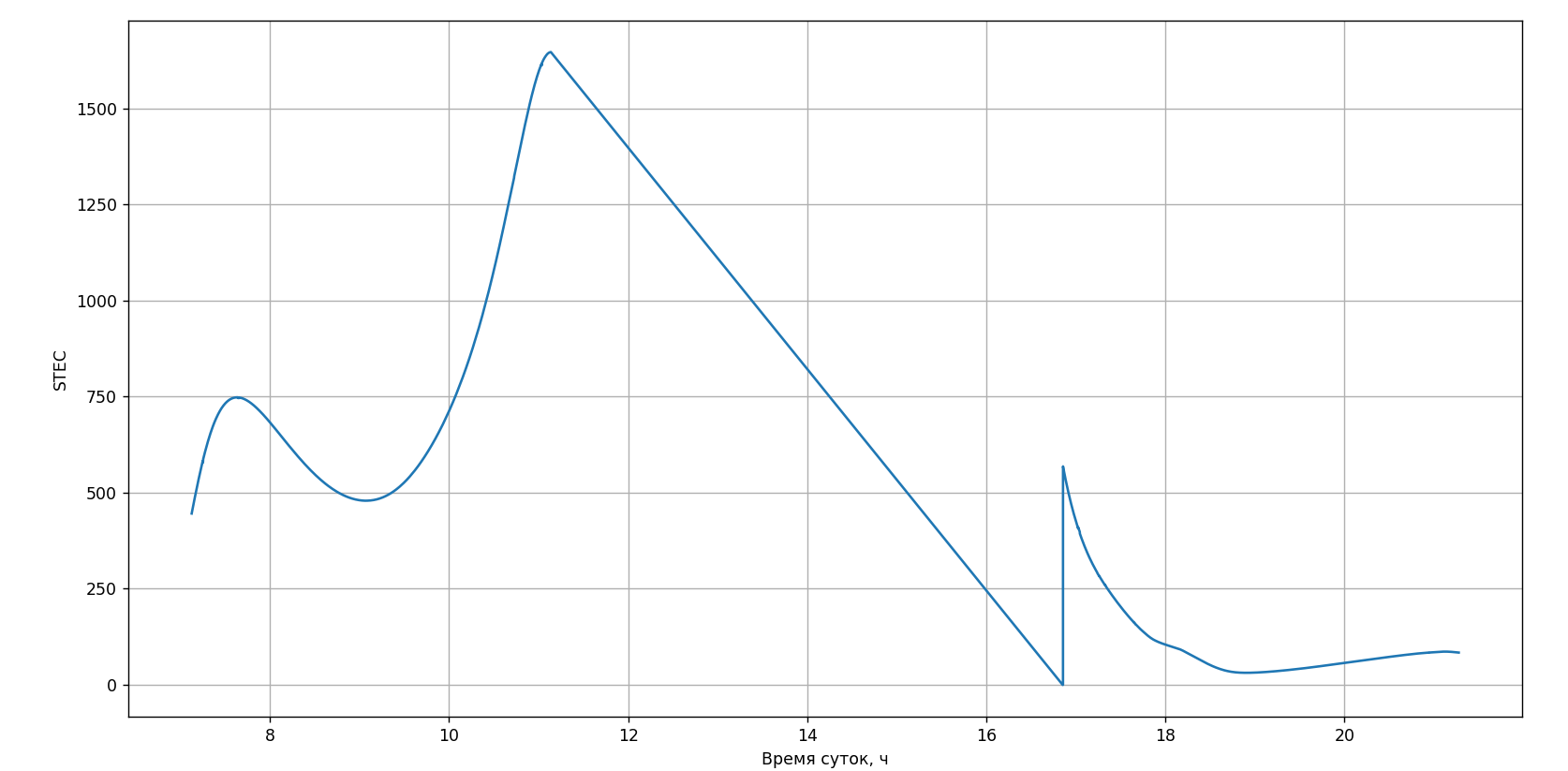


Рисунок N – График STEC от времени суток

Здесь максимумы и минимумы находятся в положенных местах, однако на 17 часах имеется скачек, который нельзя назвать естественным явлением. Дело в том, что в этот момент поменялись коэффициенты солнечной активности, что повлияло на форму графика. Стоит отметить, что на графике представлено более 30 тысяч значений и практически прямая линия, наблюдаемая от полудня до 17 часов это не результат низкой точности графика, в этом диапазоне NeQuick вычислил тысячи точек.

## 3.2 Результаты и анализ

## ЗАКЛЮЧЕНИЕ

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 <http://www.gisa.ru/104204.html>

2 К.М. Антонович ИСПОЛЬЗОВАНИЕ СПУТНИКОВЫХ РАДИОНАВИГАЦИОННЫХ СИСТЕМ В ГЕОДЕЗИИ Том 1 стр 232

3 Ionospheric correction Algorithm for Galileo single frequency users – European GNSS (Galileo) Open Service, SECTION 2:single frequency ionospheric correction algorithm.

## ПРИЛОЖЕНИЕ А Программный код, созданный и использованный в процессе работы

## А 1 Значимые функции для загрузки данных с помощью протокола FTP

def get\_rinex\_files():

path\_prefix, ftp\_path\_prefix = 'ftp\_data/', '/pub/'

ftps = ftp\_login()

for DIR in RINEX\_FILES\_DIRS:

get\_files\_from\_ftp\_dir(DIR, path\_prefix, ftp\_path\_prefix, '.rnx', '.gz', ftps)

ftps.quit()

def check\_rinex\_file(filename: str) -> None:

file\_no\_path = filename[54:]

i8e\_data = read\_file(filename, 3, 0, 3)

cords\_data = read\_file(filename, 9, 60, 69)

if i8e\_data == 'GAL' and cords\_data == 'COMMENT':

print(f'{file\_no\_path} is valid')

json\_data = {file\_no\_path: {'i8e': i8e\_data, 'cords': cords\_data}}

with open('rinex\_data.json', 'a') as f:

json.dump(json\_data, f)

return

print(f'GAL-{i8e\_data}|COMMENT-{cords\_data}')

def get\_sp3\_files():

path\_prefix, ftp\_path\_prefix = 'ftp\_data/', '/pub/'

ftps = ftp\_login()

for DIR in SP3\_FILES\_DIRS:

get\_files\_from\_ftp\_dir(DIR, path\_prefix, ftp\_path\_prefix, '.sp3', '.Z', ftps)

ftps.quit()

def get\_obs\_files():

path\_prefix, ftp\_path\_prefix = 'ftp\_data/', '/pub/'

ftps = ftp\_login()

for DIR in OBS\_FILES\_DIRS:

get\_files\_from\_ftp\_dir(DIR, path\_prefix, ftp\_path\_prefix, '.20o', '.Z', ftps)

ftps.quit()

def get\_sp3\_datetime():

for file in listdir('python\_sp3\_files/'):

datetime = read\_file('python\_sp3\_files/' + file, 22, 3, 31)

json\_data = {file: datetime}

with open('sp3\_data.json', 'a') as f:

json.dump(json\_data, f)

print(f'{file} has been parsed')

### **Продолжение приложения А**

### **А 2 Значимые функции для обработки входных данных**

def main():

print('reading rnx')

# Get ionospheric coefficients

gal = read\_file(RINEX, 2, 7, 41).replace('D', 'e').split(' ')

a0, a1, a2 = [float(x) for x in gal if x]

# Get station coordinates

station\_cords = read\_file(RINEX, 8, 2, 42).split(' ')

station\_cords = [float(x) for x in station\_cords if x]

x, y, z = xyz\_to\_blh(\*station\_cords)

# Get date

date = read\_file(RINEX, 1, 40, 48)

year, month, day = map(int, [date[:4], date[4:6], date[6:]])

print('writing stdin.txt\n reading sp3')

with open('сборка/stdin.txt', 'w') as input\_data:

with open(SP3, 'r') as sp3:

with open('major\_data.txt', 'r') as psat:

print('checking dates')

for line in sp3.readlines():

match line[0]:

# Get UT

case '\*':

ut = line[14:32].split(' ')

h, m, s = [float(x) for x in ut if x]

ut = h + m / 60 + s / 3600

date = line[3:14].split(' ')

s\_year, s\_month, s\_day = [int(x) for x in date if x]

continue

# Get position

case 'P':

print('checking um')

sat\_cords = psat.readline().split(' ')

sat\_cords = [float(x) for x in sat\_cords if x]

b, l, h = xyz\_to\_blh(\*sat\_cords)

b, l, h = x, y, z + 2e7

rang = sqrt(

(sat\_cords[0] - station\_cords[0]) \* (sat\_cords[0] - station\_cords[0]) + \

(sat\_cords[1] - station\_cords[1]) \* (sat\_cords[1] - station\_cords[1]) + \

(sat\_cords[2] - station\_cords[2]) \* (sat\_cords[2] - station\_cords[2])

)

kx = (sat\_cords[0] - station\_cords[0]) / rang

ky = (sat\_cords[1] - station\_cords[1]) / rang

kz = (sat\_cords[2] - station\_cords[2]) / rang

um = asin(

(kx \* station\_cords[0] + ky \* station\_cords[1] + kz \* station\_cords[2]) / \

sqrt(

station\_cords[0] \* station\_cords[0] + \

station\_cords[1] \* station\_cords[1] + \

station\_cords[2] \* station\_cords[2]

)

) \* 180.0 / pi

if um <= 0:

print(f'skipping - um is negative ({um})')

continue

### **Продолжение приложения А**

case \_:

continue

input\_data.write(f'{month} {ut} {y} {x} {z} {l} {b} {h}\n')

print('running nequick')

command = f'cd сборка; ./NeQuickG\_JRC\_CCIR\_MODIP\_constants\_UT\_D.exe -f {a0} {a1} {a2} stdin.txt stdout.txt'

print(system(command))

print('finished')

def read\_file(filename: str, line: int, start: int | None = 0, end: int | None = -1) -> str:

result = ''

with open(filename, 'r') as f:

for l in range(line):

try:

f.readline()

except UnicodeDecodeError:

print(f'decode error in file {filename}')

return

if l + 1 == line:

result = f.readline()

return result[start:end]

def find\_rinex():

rinex\_dir = 'ftp\_data/gnss/data/daily/'

for year in listdir(rinex\_dir):

dir\_with\_year = rinex\_dir + year + '/'

for d in listdir(dir\_with\_year):

folder = year[-2:] + 'l'

dir\_with\_d = dir\_with\_year + d + '/' + folder + '/'

for file in listdir(dir\_with\_d):

path = dir\_with\_d + file

is\_gal = read\_file(path, 2, 0, 3)

cords\_comment = read\_file(path, 8, 60, 67)

if is\_gal == 'GAL' and cords\_comment == 'COMMENT':

gal = read\_file(path, 2, 7, 41)

cords = read\_file(path, 8, 2, 42)

cords = cords.split(' ')

cords = [x for x in cords if x]

date = read\_file(path, 1, 41, 48)

year, month, day = date[:4], date[4:6], date[6:]

if year == '2017' and month == '7' and day == '10':

print(path)

return

print(f'{file} does not match')

else:

print(f'file {file} is wrong: gal: {gal} cords comment: {cords\_comment}')

def get\_data\_from\_major():

with open('NeQk/almanach/GLO\_CRD\_ALM\_04\_step001sec.csv', 'r') as f:

with open('major\_data.txt', 'w') as m:

for line in f.readlines():

data = line.split(' ')

m.write(f'{data[1]} {data[2]} {data[3]}\n')

### **Продолжение приложения А**

print('finished extracting majors data')

if \_\_name\_\_ == '\_\_main\_\_':

start = datetime.now()

get\_data\_from\_major()

main()

end = datetime.now()

time = end - start

print(f'execution time {time.microseconds / 1000}ms')

### **А 3 Служебные функции**

def unpack\_z\_file(filename: str) -> str:

if not filename.endswith('.Z'):

print('wrong file format, not Z')

return

new\_name = filename[:-2]

if path.exists(new\_name):

print(f'{new\_name} already exists')

remove(filename)

return new\_name

uncompressed\_data = unlzw3.unlzw(Path(filename)).decode()

with open(new\_name, 'w') as f:

f.write(uncompressed\_data)

remove(filename)

print(f'{filename} unpacked')

return new\_name

def unpack\_gz\_file(filename: str) -> str:

if not filename.endswith('.gz'):

print('wrong file format, not gz')

return

new\_name = filename[:-3]

if path.exists(new\_name):

print(f'{new\_name} already exists')

remove(filename)

return new\_name

with gzip.open(filename, 'rb') as f\_in:

with open(new\_name, 'wb') as f\_out:

shutil.copyfileobj(f\_in, f\_out)

remove(filename)

print(f'{filename} unpacked')

return new\_name

def download\_file\_ftp(dir: str, filename: str,

target\_dir: str | None = None,

ftps: FTP\_TLS | None = None) -> None:

if target\_dir[-1] != '/':

target\_dir += '/'

if filename in listdir(getcwd() + '/' + target\_dir):

print(f'{filename} already present')

return

if not ftps:

ftps = FTP\_TLS(host=HOST)

ftps.login(user=USER, passwd=PASSWORD)

ftps.prot\_p()

### **Продолжение приложения А**

ftps.cwd(dir)

ftps.retrbinary("RETR " + filename, open(target\_dir + filename, 'wb').write)

print(f'{filename} downloaded')

def ftp\_login() -> FTP\_TLS:

ftps = FTP\_TLS(host=HOST)

ftps.login(user=USER, passwd=PASSWORD)

ftps.prot\_p()

print('login successful')

return ftps

def ftp\_cwd(dir: str, ftps: FTP\_TLS | None = None) -> FTP\_TLS:

if not ftps:

ftps = ftp\_login()

ftps.prot\_p()

ftps.cwd(dir)

return ftps

def get\_directory\_ftp(dir: str, ftps: FTP\_TLS | None = None) -> list[str, ]:

if not ftps:

ftps = ftp\_cwd(dir)

ftps.prot\_p()

ftps = ftp\_cwd(dir, ftps)

return ftps.nlst()

def get\_dir\_with\_data(dir: str, ftps: FTP\_TLS | None) -> list[str, ]:

if not ftps:

ftps = ftp\_cwd(dir)

ftps.prot\_p()

ftps = ftp\_cwd(dir, ftps)

return ftps.retrlines('LIST')

def filter\_by\_format(files: list[str, ], format: str):

return [file for file in files if file.endswith(format)]

def coordinates\_parser(string: str) -> list[float, float, float]:

res = [x for x in string.split(' ') if x != '']

res = [x.replace('D', 'E') for x in res]

cords = []

for x in res:

try:

cords.append(float(x))

except ValueError:

continue

if len(cords) == 3:

print('Cords parsed successfully')

return cords

print(f'error parse cords {cords}')

return

def ionosphere\_parser(string: str) -> list[float, float, float]:

res = [x for x in string.split(' ') if x != '']

res = [x.replace('D', 'E') for x in res]

### **Окончание приложения А**

try:

if res[0] != 'GAL':

print('wrong data')

return

except IndexError:

print('error parse i8e')

return

a\_coefs = []

for x in res[1:]:

try:

a\_coefs.append(float(x))

except ValueError:

continue

if len(a\_coefs) == 4:

print('Ionosphere coefs parsed successfully')

return a\_coefs[:-1]

print(f'error parse coefs {a\_coefs}')

return

def get\_files\_from\_ftp\_dir(dir: str,

target\_root: str,

ftp\_root: str,

format: str,

archive: str | None = None,

ftps: FTP\_TLS | None = None) -> None:

if not ftps:

ftps = ftp\_login()

current\_dir = ftp\_root + dir

host\_dir = target\_root + dir

ftps = ftp\_cwd(current\_dir, ftps)

try:

makedirs(host\_dir)

except FileExistsError:

print(f'{dir} already exists')

print(f'getting all {format} files from {dir}')

files = get\_directory\_ftp(current\_dir, ftps)

files = [f for f in files if f.endswith(format + archive) and not path.exists(host\_dir + f[:-len(archive)])]

for file in files:

download\_file\_ftp(current\_dir, file, host\_dir)

filename = host\_dir + file

if archive == '.gz':

filename = unpack\_gz\_file(filename)

elif archive == '.Z':

filename = unpack\_z\_file(filename)

else:

print('File is not archived of unknown archive format')

print(f'{dir} downloaded successfully')

## СПИСОК СОКРАЩЕНИЙ?