Национальный исследовательский университет

«МЭИ»

Отчёт по курсовой работе

|  |  |
| --- | --- |
| Группа: | Эр-15-17\_\_\_\_\_ |
| ФИО студента: | Ливенцов Василий |
| ФИО преподавателя: | \_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
| Результат: | \_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Подпись: | \_\_\_\_\_\_\_\_\_\_\_\_\_ |

Москва

2022

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ](file:///C:\Users\Vasiliy\Downloads\Отчет_КП_п.1,2.docx#_Toc101728115) 2

[1 Обработка данных навигационного приемника 4](file:///C:\Users\Vasiliy\Downloads\Отчет_КП_п.1,2.docx#_Toc101728116)

[1.1 Задание 4](file:///C:\Users\Vasiliy\Downloads\Отчет_КП_п.1,2.docx#_Toc101728117)

[1.2 Разработка программы обработки 5](file:///C:\Users\Vasiliy\Downloads\Отчет_КП_п.1,2.docx#_Toc101728118)

[2 Моделирование траектории движения 6](file:///C:\Users\Vasiliy\Downloads\Отчет_КП_п.1,2.docx#_Toc101728121)

[2.1 Задание 6](file:///C:\Users\Vasiliy\Downloads\Отчет_КП_п.1,2.docx#_Toc101728122)

[2.2 Разработка программы расчета положения спутника 6](file:///C:\Users\Vasiliy\Downloads\Отчет_КП_п.1,2.docx#_Toc101728123)

[3 Реализация модуля расчета координат 9](file:///C:\Users\Vasiliy\Downloads\Отчет_КП_п.1,2.docx#_Toc101728121)

ВЫВОД 9

[ПРИЛОЖЕНИЕ 1 10](file:///C:\Users\Vasiliy\Downloads\Отчет_КП_п.1,2.docx#_Toc101728128)

[ПРИЛОЖЕНИЕ 2 16](file:///C:\Users\Vasiliy\Downloads\Отчет_КП_п.1,2.docx#_Toc101728129)

[ПРИЛОЖЕНИЕ 3 22](file:///C:\Users\Vasiliy\Downloads\Отчет_КП_п.1,2.docx#_Toc101728129)

# ВВЕДЕНИЕ

Спутниковая система навигации – комплексная электронно-техническая система, состоящая из совокупности наземного и космического оборудования, предназначенная для определения местоположения (географических координат и высоты), а также параметров движения (скорости и направления движения) для наземных, водных и воздушных объектоСовременная спутниковая навигация основывается на использовании принципа беззапросных дальномерных измерений между навигационными спутниками и потребителем. Это означает, что потребителю в составе навигационного сигнала передается информация о координатах спутников. Одновременно (синхронно) производятся измерения дальностей до навигационных спутников. Способ измерений дальностей основывается на вычислении временных задержек принимаемого сигнала от спутника по сравнению с сигналом, генерируемым НАП.

Принцип работы спутниковых систем навигации основан на измерении расстояния от антенны на объекте (координаты которого необходимо получить) до спутников, положение которых известно с большой точностью. Таблица положений всех спутников называется альманахом, которым должен располагать любой спутниковый приемник до начала измерений. Обычно приемник сохраняет альманах в памяти со времени последнего выключения и если он не устарел – мгновенно использует его. Каждый спутник передает в своём сигнале весь альманах. Таким образом, зная расстояния до нескольких спутников системы, с помощью обычных геометрических построений, на основе альманаха, можно вычислить положение объекта в пространстве.

Навигационные спутники передают два вида данных — альманах и эфимерис. Данные эфимериса содержат очень точные корректировки параметров орбит и часов для каждого спутника, что требуется для точного определения координат. Каждый навигационный спутник передает данные только своего собственного эфимериса. Первый этап курсового проекта нацелен на разработку модуля разбора навигационного сообщения до структуры эфемерид.

# Обработка данных навигационного приемника

# Задание

В неизвестной локации установлен навигационный приемник, принимающий сигналы GPS L1C/A и логирующий результаты этого приема в формате NVS BINR. Собранный на пятиминутном нтервале файл приложен в архиве под именем BINR.bin. Файл содержит наблюдения псевдодальностей и прочих радионавигационных параметров, демодулированные и разобранные данные навигационного сообщения.

Данные демодулятора продублированы в текстовый файл in.txt. Каждая строка файла содержит данные одного сабфрейма одного навигационного сигнала в формате:



Рисунок 1 – Структура одного сабфрейма одного НКА

где 13 - номер спутника, 212130404 - счетчик сабфреймов в сигнале, 53

- ID сабфрейма в навигационном сообщении, где в первых трех битах содержится номер сабфрейма в фрейме (5 в данном примере), а далее - номер фрейма в сообщении (6 в данном примере), 1000101110... символы с демодулятора в порядке возрастания времени слева направо.

Требуется:

Разработать программу, обрабатывающую файл in.txt и выводящую в файл out.txt таблицу эфемерид для спутника согласно варианту в заданном формате.

# Разработка программы для обработки

Согласно варианту задания необходимо обработать данные для 16 спутника, а именно: эфемеридная информация, содержащаяся в первых трёх сабфреймах структуры навигационного сообщения. В приложении 1 находится кода программы, реализующий обработку файла in.txt на языке C++.

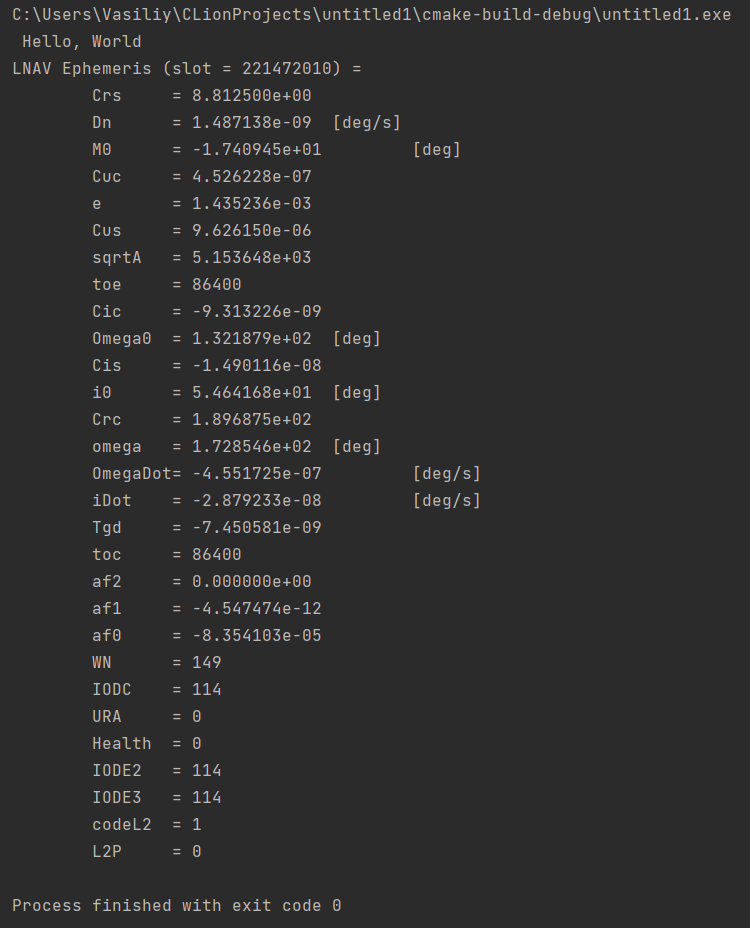


Рисунок 1 – Результат компиляции программы

По полученным таблицам можно отметить, что эфемеридная информация была выделена корректно.

# Моделирование траектории движения

# Задание ко второму этапу

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

Построить трехмерные графики множества положений спутника GPS с системным номером, соответствующим номеру студента по списку. Графики в двух вариантах: в СК ECEF WGS84 и соответствующей ей инерциальной СК. Положения должны соответствовать суточному интервалу на дне формирования наблюдений, определенном на предыдущем этапе. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал.

Вывести значения координат спутника в файл out.txt в системе ECEF WGS 84 в виде строк: Секунда\_от\_начала\_дня X Y Z.

# Результаты моделирования

На рисунке 7 представлена траектория движения КА №16 на интервале суток в СК ECEF WGS 84 с отмеченным местоположением приёмника. На рисунке 2 изображена траектория движения КА в СК, а также точка, в которой находится

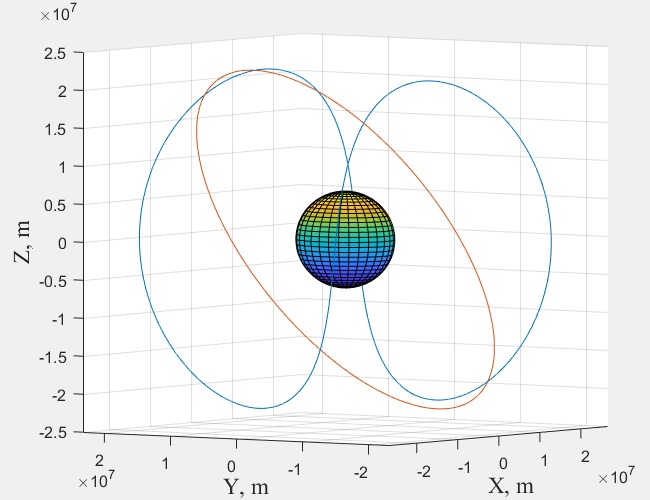


Рисунок 2-Множество положений спутника GPS в системе ECEF WGS84(Синий). Множество положений спутника GPS в системе ECI(красный)

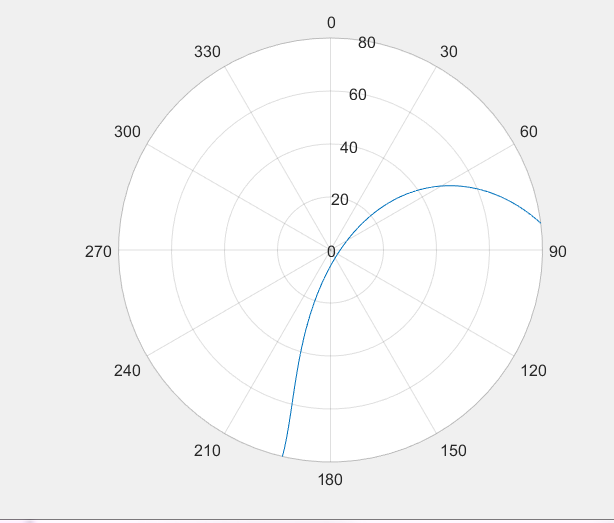
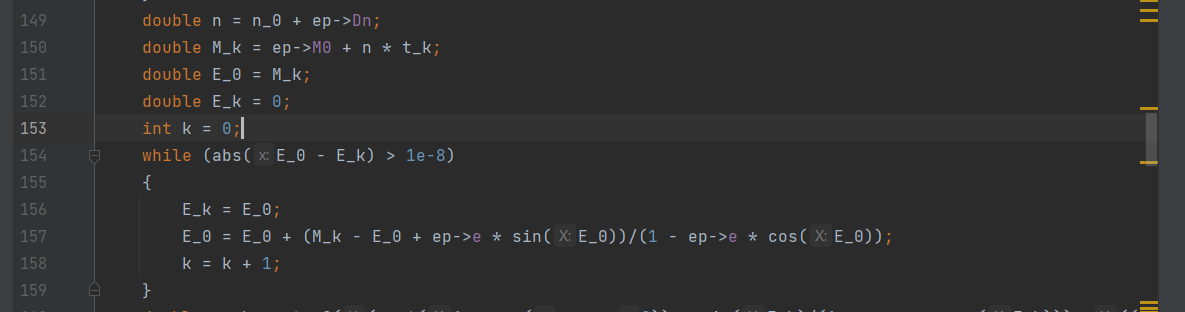


Рисунок 8 — Sky Plot

**3 Реализация модуля расчета координат**

Требуется сравнение расчётов в двух независимых программах, которыми были выполнены C++ и Matlab. В ходе было реализовано решение уравнения Кеплера, алгоритм расчёта которого представлен на рисунке 8.

Рисунок 9 – Алгоритм решения уравнения Кеплера

Для того, чтобы достичь высокой точности, необходимо реализовать передачу и вывод чисел с высокой точностью. Разница координат получалась :



Рисунок 10 – Результат работы программы

Проведена работа по анализу объёма потребляемой памяти и нахождения утечки памяти. Данные действия были проведены в программе Microsoft Visual Studio 2022.

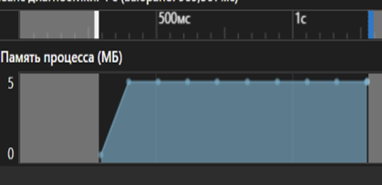


Рисунок 11 – Анализ объёма потребляемой памяти

**Вывод:**

На третьем этапе курсового проекта была написана программа на языке C++, что целесообразно при разработке встраиваемого ПО. Было показано, что реализация алгоритмов в C++и Matlab не приводит к серьёзному расхождению в результатах. Объем потребляемой памяти составил 4111,85 КБ, утечка памяти не была обнаружена.

Приложение1.

Код программы С++

#include <stdio.h>  
#include <stdint.h>  
#include <stdlib.h>  
#include <string.h>  
#define \_USE\_MATH\_DEFINES  
#include <cmath>  
#include <iostream>  
using namespace std;  
struct Ephemeris {  
 double Crs;  
 double Dn;  
 double M0;  
 double Cuc;  
 double e;  
 double Cus;  
 double sqrtA;  
 uint32\_t toe;  
 double Cic;  
 double Omega0;  
 double Cis;  
 double i0;  
 double Crc;  
 double omega;  
 double OmegaDot;  
 double iDot;  
 double Tgd;  
 uint32\_t toc;  
 double af2;  
 double af1;  
 double af0;  
 uint32\_t WN;  
 uint16\_t IODC;  
 uint8\_t URA;  
 uint8\_t Health;  
 uint16\_t IODE2;  
 uint16\_t IODE3;  
 bool codeL2;  
 bool L2P;  
 uint32\_t slot;  
};  
const int32\_t subFrameLength = 300;  
struct SF1\_3 {  
 uint32\_t slot;  
 char sf1[subFrameLength+1];  
 char sf2[subFrameLength+1];  
 char sf3[subFrameLength+1];  
};  
void printEmp(Ephemeris\* ep);  
int32\_t file2subFrames(SF1\_3\* sf, FILE\* fid, uint8\_t svNum);  
int32\_t subFrames2Eph(Ephemeris\* ep, SF1\_3\* subframes);  
  
  
int main(void)  
{  
 printf(" Hello, World \n");  
 uint8\_t svNum = 14;  
 FILE\* fid = fopen("in.txt", "r");  
 if (fid != nullptr) {  
 SF1\_3 subframes;  
 if (!file2subFrames(&subframes, fid, svNum)) {  
  
 Ephemeris \*ep = (Ephemeris\*) calloc(1, sizeof(Ephemeris));  
 if (!subFrames2Eph(ep, &subframes)) {  
 printEmp(ep);  
  
 } else {  
 printf(" Cannot decode subframes\n ");  
 }  
 free(ep);  
 fclose(fid);  
 }  
 else {  
 printf(" Subframes not found\n ");  
 }  
 }  
 else {  
 printf(" Cannot open in.txt ");  
 }  
 return 0;  
}  
int64\_t str2uint(char \*sf, int32\_t start, int32\_t stop) {  
 int64\_t ans = 0;  
 for(int i = start; i < stop; i++) {  
 bool bit = (sf[i-1] == '1');  
 ans = ans | (bit << (stop - i - 1));  
  
 }  
 return ans;  
}  
int64\_t str3int(uint64\_t ans, int count\_bit) {  
  
 int64\_t Ians = 0;  
 if (count\_bit == 8) {  
 if (bool((1<<7) & ans)){  
 ans |= 0xFFFFFFFFFFFFFF00;  
 Ians = ~(ans - 1);  
 return -Ians;  
  
 }  
 }  
 if (count\_bit == 14) {  
 if (bool((1<<13) & ans)) {  
 ans |= 0xFFFFFFFFFFFFC000;  
 Ians = ~(ans - 1);  
 return -Ians;  
 }  
 }  
 if (count\_bit == 16) {  
 if (bool((1 << 15) & ans)) {  
 ans |= 0xFFFFFFFFFFFF0000;  
 Ians = ~(ans - 1);  
 return -Ians;  
 }  
 }  
 if (count\_bit == 22) {  
 if (bool((1 << 21) & ans)) {  
 ans |= 0xFFFFFFFFFFC00000;  
 Ians = ~(ans - 1);  
 return -Ians;  
 }  
 }  
 if (count\_bit == 24) {  
 if (bool((1 << 23) & ans)) {  
 ans |= 0xFFFFFFFFFF000000;  
 Ians = ~(ans - 1);  
 return -Ians;  
 }  
 }  
 if (count\_bit == 32) {  
 if (bool((1 << 31) & ans)) {  
 ans |= 0xFFFFFFFF00000000;  
 Ians = ~(ans - 1);  
 return -Ians;  
 }  
 }  
 return ans;  
}  
int64\_t str4uint(char \*sf, int32\_t start, int32\_t stop, int32\_t start2, int32\_t stop2){  
 uint32\_t ans = 0;  
 for(int i = start; i < stop; i++) {  
 ans = (ans | ((sf[i-1] == '1')? 1 : 0)) << 1;  
 }  
 for(int i = start2; i < stop2-1; i++) {  
  
 ans = ans | ((sf[i-1] == '1')? 1 : 0);  
 if (i < stop2-1) {  
 ans = ans << 1;  
 }  
  
 }  
 return ans;  
}  
  
int32\_t subFrames2Eph(Ephemeris\* ep, SF1\_3\* subframes) {  
 ep->slot = subframes->slot;  
 ep->WN = str2uint(subframes->sf1, 61, 71);  
 ep->URA = str2uint(subframes->sf1, 73, 77);  
 ep->toe = str2uint(subframes->sf2, 271, 287)\*pow(2,4);  
 ep->Health = str2uint(subframes->sf1, 73, 73+6);  
 ep->IODE2 = str2uint(subframes->sf2, 61, 69);  
 ep->IODE3 = str2uint(subframes->sf3, 271, 271+8);  
 ep->codeL2 = str2uint(subframes->sf1, 71, 73);  
 //ep->L2P = subframes->sf1[90];  
 ep->L2P = str2uint(subframes->sf1, 90, 91);  
 ep->Crc = str3int(str2uint(subframes->sf3,181,181+16),16)\*pow(2, -5);  
 ep->Dn = str3int(str2uint(subframes->sf2, 91, 91+16), 16)\*pow(2, -43);  
 ep->Cuc = str3int(str2uint(subframes->sf2,151,151+16),16)\*pow(2, -29);  
 ep->Cus = str3int(str2uint(subframes->sf2,211,211+16),16)\*pow(2, -29);  
 ep->e = str4uint(subframes->sf2,167, 167+8, 181, 181+24) \* pow(2, -33);  
 ep->sqrtA = str4uint(subframes->sf2,227, 227+8, 241, 241+24) \* pow(2, -19);  
 ep->Cic = str3int(str2uint(subframes->sf3,61,61+16),16)\*pow(2, -29);  
 ep->Omega0 = str3int(str4uint(subframes->sf3,77, 77+8, 91, 91+24),32)\*pow(2, -31)\*180;  
 ep->Cis = str3int(str2uint(subframes->sf3,121,121+16),16)\*pow(2, -29);  
 ep->i0 = str3int(str4uint(subframes->sf3,137, 137+8, 151, 151+24),32)\*pow(2, -31)\*180;  
 ep->omega = str3int(str4uint(subframes->sf3,197, 197+8, 211, 211+24),32)\*pow(2, -31)\*180;  
 ep->OmegaDot = str3int(str2uint(subframes->sf3,241,241+24),24)\*pow(2, -43)\*180;  
 ep->iDot = str3int(str2uint(subframes->sf3,279,279+14),14)\*pow(2, -43)\*180;  
 ep->Tgd = str3int(str2uint(subframes->sf1,197,197+8),8)\*pow(2, -31);  
 ep->toc = str3int(str2uint(subframes->sf1,219,219+16),16)\*pow(2, 4);  
 ep->af2 = str3int(str2uint(subframes->sf1,241,241+8),8)\*pow(2, -55);  
 ep->af1 = str3int(str2uint(subframes->sf1,249,249+16),16)\*pow(2, -43);  
 ep->af0 = str3int(str2uint(subframes->sf1,271,271+22),22)\*pow(2, -31);  
 ep->IODC = str4uint(subframes->sf1,83, 83+2, 211, 211+8);  
 ep->Crs = str3int(str2uint(subframes->sf2,69,69+16),16)\*pow(2, -5);  
 ep->M0 = str3int(str4uint(subframes->sf2,107, 107+8, 121, 121+24),32)\*pow(2, -31)\*180;  
 return 0;  
}  
int32\_t file2subFrames(SF1\_3\* sf, FILE\* fid, uint8\_t svNum){  
 int32\_t sth1, sth2, sth3, sth4, sth5;  
 char str\_0R[8];  
 char str\_GPSL1CA[12];  
 char str\_reh[8];  
 char str[1000];  
 uint32\_t svStr;  
 uint32\_t slot;  
 int32\_t subFrameNum;  
  
 uint32\_t slot\_SF1 = 0;  
 uint32\_t slot\_SF2 = 0;  
 uint32\_t slot\_SF3 = 0;  
 int32\_t readres = 0;  
  
 while(readres != EOF)  
 {  
 svStr = 0;  
 readres = fscanf( fid, "%d %d %d %s %s %s %u\t %u %d %d %d %s", &sth1, &sth2, &sth3, str\_0R, str\_GPSL1CA, str\_reh, &svStr, &slot, &sth4, &sth5, &subFrameNum, str);  
 if (( svStr == svNum ) && (slot >= (604800/6))) {  
 if ( subFrameNum == 1 ) {  
 slot\_SF1 = slot;  
 strncpy(sf->sf1, str, sizeof(sf->sf1));  
 }  
 else if (subFrameNum == 2) {  
 slot\_SF2 = slot;  
 strncpy(sf->sf2, str, sizeof(sf->sf2));  
 }  
 else if (subFrameNum == 3) {  
 slot\_SF3 = slot;  
 strncpy(sf->sf3, str, sizeof(sf->sf3));  
 }  
 if ((slot\_SF1 + 1 == slot\_SF2) && (slot\_SF2 + 1 == slot\_SF3)) {  
 sf->slot = slot\_SF1;  
 return 0;  
 }  
 }  
 }  
 return 1;  
}  
void printEmp(Ephemeris\* ep)  
{  
 printf("LNAV Ephemeris (slot = %u) = \n", ep->slot );  
 printf("\tCrs = %e \n", ep->Crs );  
 printf("\tDn = %e \t[deg/s] \n", ep->Dn );  
 printf("\tM0 = %e \t[deg] \n", ep->M0 );  
 printf("\tCuc = %e \n", ep->Cuc );  
 printf("\te = %e \n", ep->e );  
 printf("\tCus = %e \n", ep->Cus );  
 printf("\tsqrtA = %e \n", ep->sqrtA );  
 printf("\ttoe = %u \n", ep->toe );  
 printf("\tCic = %e \n", ep->Cic );  
 printf("\tOmega0 = %e \t[deg] \n", ep->Omega0 );  
 printf("\tCis = %e \n", ep->Cis );  
 printf("\ti0 = %e \t[deg] \n", ep->i0 );  
 printf("\tCrc = %e \n", ep->Crc );  
 printf("\tomega = %e \t[deg] \n", ep->omega );  
 printf("\tOmegaDot= %e \t[deg/s] \n", ep->OmegaDot );  
 printf("\tiDot = %e \t[deg/s] \n", ep->iDot );  
 printf("\tTgd = %e \n", ep->Tgd );  
 printf("\ttoc = %u \n", ep->toc );  
 printf("\taf2 = %e \n", ep->af2 );  
 printf("\taf1 = %e \n", ep->af1 );  
 printf("\taf0 = %e \n", ep->af0 );  
 printf("\tWN = %u \n", ep->WN );  
 printf("\tIODC = %u \n", ep->IODC );  
 printf("\tURA = %u \n", ep->URA );  
 printf("\tHealth = %u \n", ep->Health );  
 printf("\tIODE2 = %u \n", ep->IODE2 );  
 printf("\tIODE3 = %u \n", ep->IODE3 );  
 printf("\tcodeL2 = %u \n", ep->codeL2 );  
 printf("\tL2P = %u \n", ep->L2P );  
}

Приложение 2

Код программы в Матлаб

close all;

clear all;

clc;

format long

% Координаты приемника

B = deg2rad(44.09363261);

L = deg2rad(39.00130546);

H = 1.247;

% Эфемериды

SatNum = 14;

toe = 86400;

Crs = 8.812500e+00;

Dn =1.487138e-09;

M0 = deg2rad(-1.740945e+01);

Cuc = 4.526228e-07;

e = 1.435236e-03;

Cus = 9.626150e-06;

sqrtA = 5.153648e+03;

Cic = -1.173466e-07;

Omega0 = deg2rad(1.321879e+02);

Cis = -9.313226e-09;

i0 = deg2rad(5.464168e+01);

Crc = 1.896875e+02;

omega = deg2rad(1.728546e+02);

OmegaDot = deg2rad(-4.551725e-07);

iDot = deg2rad(-2.879233e-08);

Tgd = -7.450581e-09;

toc = 86400;

af2 = 0;

af1 = -4.547474e-12;

af0 = -8.354103e-05;

URA = 0;

IODE = 23;

IODC = 23;

codeL2 = 1;

L2P = 0;

WN = 149;

% Значения констант

mu = 3.986004418e14; % гравитационная постоянная

omega\_e = 7.2921151467e-5; % скорость вращения

% Временной промежуток

begin\_time = 0; % время начала 0:00 по МСК 14 февраля

end\_time = 24\*60\*60; % время окончания 0:00 по МСК 15 февраля

% Длина временного промежутка

t\_arr = begin\_time:1:end\_time;

% Большая полуось

A = sqrtA^2;

% Среднее движение

n0 = sqrt(mu/A^3);

n = n0+Dn;

for k = 1:86400

% Время

t(k) = t\_arr(k)-toe;

if t(k) > 302400

t(k) = t(k)-604800;

elseif t(k) < -302400

t(k) = t(k)+604800;

end

% Средняя аномалия

M(k) = M0+n\*t(k);

% Решение уравнения Кеплера

E(k) = M(k);

E\_old(k) = M(k)+1;

epsilon = 1e-6;

while abs(E(k)-E\_old(k)) > epsilon

E\_old(k) = E(k);

E(k) = M(k)+e\*sin(E(k));

end

% Истинная аномалия

nu(k) = 2\*atan(sqrt((1+e)/(1-e))\*tan(E(k)/2));

% Коэффициент коррекции

cos\_correction(k) = cos(2\*(omega+nu(k)));

sin\_correction(k) = sin(2\*(omega+nu(k)));

% Аргумент широты

u(k) = omega+nu(k)+Cuc\*cos\_correction(k)+Cus\*sin\_correction(k);

% Радиус

r(k) = A\*(1-e\*cos(E(k)))+Crc\*cos\_correction(k)+Crs\*sin\_correction(k);

% Наклон

i(k) = i0+iDot\*t(k)+Cic\*cos\_correction(k)+Cis\*sin\_correction(k);

% Долгота восходящего угла

lambda(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(lambda(k))-y\*cos(i(k))\*sin(lambda(k));

Y0(k) = x\*sin(lambda(k))+y\*cos(i(k))\*cos(lambda(k));

Z0(k) = y\*sin(i(k));

X(k) = X0(k)\*cos(lambda(k))+Y0(k)\*sin(lambda(k));

Y(k) = -X0(k)\*sin(lambda(k))+Y0(k)\*cos(lambda(k));

Z(k) = Z0(k);

[x(k), y(k), z(k)] = ecef2enu(X0(k), Y0(k), Z0(k), B, L, H, wgs84Ellipsoid, 'radians');

if z(k) > 0

rho(k) = sqrt(x(k)^2 + y(k)^2 + z(k)^2);

theta1(k) = acos(z(k)/rho(k));

if x(k) > 0

phi(k) = -atan(y(k)/x(k)) + pi/2;

elseif (x(k) < 0)&&(y(k) > 0)

phi(k) = -atan(y(k)/x(k))+3\*pi/2;

elseif (x(k)<0)&&(y(k)<0)

phi(k) = -atan(y(k)/x(k))-pi/2;

end

else theta1(k) = NaN;

rho(k) = NaN;

phi(k) = NaN;

end

end

%График

figure (1)

axes = polaraxes;

polarplot(axes,phi,rad2deg(theta1))

axes.ThetaDir = 'clockwise';

axes.ThetaZeroLocation = 'top';

rlim([0 80]);

R\_Earth = 6371e3;

[XE,YE,ZE] = sphere(30);

figure(2)

surf(XE\*R\_Earth,YE\*R\_Earth,ZE\*R\_Earth)

hold on

grid on

plot3(X0, Y0, Z0)

plot3(X, Y, Z)

xlabel('X, m', 'FontName', 'Times New Roman', 'FontSize',14)

ylabel('Y, m', 'FontName', 'Times New Roman', 'FontSize',14)

zlabel('Z, m', 'FontName', 'Times New Roman', 'FontSize',14)

hold off

lgd.FontName = 'Times New Roman';

Приложение 3

Расширенная программа С++

#include <iostream>  
#include <fstream>  
#include <string>  
#include <stdlib.h>  
#include <cmath>  
#include <stdio.h>  
#include <windows.h>  
#include <ctime>  
#define Semi\_circles 180  
#define pi 3.1415326535898  
#define SF1 pow(2,-5)  
#define SF2 pow(2,-43)  
#define SF3 pow(2,-31)  
#define SF4 pow(2,-29)  
#define SF5 pow(2,-33)  
#define SF6 pow(2,-19)  
#define SF7 pow(2,4)  
#define SF8 pow(2,-55)  
using namespace std;  
struct Ephemeris  
{  
 float Crs;  
 float Dn;  
 float M0;  
 float Cuc;  
 float e;  
 float Cus;  
 float sqrtA;  
 uint32\_t toe;  
 float Cic;  
 double Omega0;  
 float Cis;  
 float i0;  
 float Crc;  
 float omega;  
 float OmegaDot;  
 float iDot;  
 int16\_t Tgd;  
 uint32\_t toc;  
 float af2;  
 float af1;  
 float af0;  
 uint16\_t WN;  
 uint16\_t IODC;  
 uint32\_t URA;  
 uint32\_t Health;  
 uint16\_t IODE2;  
 uint16\_t IODE3;  
 bool codeL2;  
 bool L2P;  
 uint32\_t slot;  
};  
struct sub  
{  
 string sf1;  
 string sf2;  
 string sf3;  
 uint32\_t slot;  
};  
struct Coordinates  
{  
 double x;  
 double y;  
 double z;  
};  
void sendStr(sub \*Subframes);  
void scale\_factor\_use(Ephemeris\* ep, sub \*data);  
int32\_t convert\_eph(string sf, int32\_t Begin, int End);  
void printEPH(Ephemeris\* ep);  
void saveEPH(Ephemeris\* ep);  
void CalcCoord(Ephemeris\* ep,uint32\_t t, Coordinates \*Position);  
int main(void)  
{  
 uint32\_t begin = clock();  
 sub data;  
 sendStr(&data);  
 Ephemeris \*ep = (Ephemeris\*) calloc(1, sizeof(Ephemeris));  
 scale\_factor\_use(ep,&data);  
 printEPH(ep);  
 saveEPH(ep);  
 uint32\_t start = 93584+1+18;  
 uint32\_t stop = 93602+86400+1;  
 Coordinates Position;  
 double\*\* coord = new double\*[3];  
 for (int i = 0; i < 3; i++)  
 {  
 coord[i] = new double[stop - start];  
 }  
 double\*\* coord\_from\_matlab = new double\*[3];  
 for (int i = 0; i < 3; i++)  
 {  
 coord\_from\_matlab[i] = new double[stop - start];  
 }  
 for (int t = start; t < stop; t++)  
 {  
 CalcCoord(ep, t, &Position);  
 coord[0][t-start] = Position.x;  
 coord[1][t-start] = Position.y;  
 coord[2][t-start] = Position.z;  
 }  
 ifstream file("output\_data.txt");  
 double sec;  
 if (!file.is\_open())  
 cout << "Can't open" << endl;  
 else {  
 for (int t = 0; t < stop - start; t++)  
 {  
 file >> sec >> coord\_from\_matlab[0][t] >> coord\_from\_matlab[1][t] >>  
 coord\_from\_matlab[2][t];  
 }  
 file.close();  
 }  
 double deviation = 0;  
 for (int i = 0; i < 3; i++)  
 {  
 for (int k = 0; k < stop - start; k++)  
 {  
 if (abs(coord[i][k] - coord\_from\_matlab[i][k]) > deviation)  
 {  
 deviation = abs(coord[i][k] - coord\_from\_matlab[i][k]);  
 }  
 }  
 }  
 delete[] \*coord;  
 delete[] coord;  
 delete[] \*coord\_from\_matlab;  
 delete[] coord\_from\_matlab;  
 uint32\_t end = clock();  
 uint32\_t calculation\_time = end - begin;  
 cout << "delta = " << deviation << "(m)" << endl;  
 cout << "program T = " << (double)calculation\_time/1000.0 << "(sec)"<<endl;  
 free(ep);  
}  
void CalcCoord(Ephemeris\* ep,uint32\_t t, Coordinates \*Position)  
{  
 double Omega\_e\_dot = 7.2921151467e-5;  
 double mu = 3.986005e14;  
 double A = pow(ep->sqrtA,2);  
 double n\_0 = sqrt(mu / pow(A, 3));  
 int32\_t t\_k = t - ep->toe;  
 if (t\_k > 302400)  
 {  
 t\_k = t\_k - 604800;  
 }  
 else if (t\_k < -302400)  
 {  
 t\_k = t\_k + 604800;  
 }  
 double n = n\_0 + ep->Dn;  
 double M\_k = ep->M0 + n \* t\_k;  
 double E\_0 = M\_k;  
 double E\_k = 0;  
 int k = 0;  
 while (abs(E\_0 - E\_k) > 1e-8)  
 {  
 E\_k = E\_0;  
 E\_0 = E\_0 + (M\_k - E\_0 + ep->e \* sin(E\_0))/(1 - ep->e \* cos(E\_0));  
 k = k + 1;  
 }  
 double nu\_k = atan2((sqrt(1 - pow(ep->e,2)) \* sin(E\_k)/(1 - ep->e \* cos(E\_k))), ((cos(E\_k) -  
 ep->e)/(1 - ep->e \* cos(E\_k))));  
 double Phi\_k = nu\_k + ep->omega;  
 double delta\_u\_k = ep->Cus \* sin(2 \* Phi\_k) + ep->Cuc \* cos(2 \* Phi\_k);  
 double delta\_r\_k = ep->Crs \* sin(2 \* Phi\_k) + ep->Crc \* cos(2 \* Phi\_k);  
 double delta\_i\_k = ep->Cis \* sin(2 \* Phi\_k) + ep->Cic \* cos(2 \* Phi\_k);  
 double u\_k = Phi\_k + delta\_u\_k;  
 double r\_k = pow(ep->sqrtA,2) \* (1 - ep->e \* cos(E\_k)) + delta\_r\_k;  
 double i\_k = ep->i0 + delta\_i\_k + ep->iDot \* t\_k ;  
 double x\_k\_sh = r\_k \* cos(u\_k);  
 double y\_k\_sh = r\_k \* sin(u\_k);  
 double Omega\_k = ep->Omega0 + (ep->OmegaDot - Omega\_e\_dot) \* t\_k - Omega\_e\_dot \*  
 ep->toe;  
 double x\_k = x\_k\_sh \* cos(Omega\_k) - y\_k\_sh \* cos(i\_k) \* sin(Omega\_k);  
 double y\_k = x\_k\_sh \* sin(Omega\_k) + y\_k\_sh \* cos(i\_k) \* cos(Omega\_k);  
 double z\_k = y\_k\_sh \* sin(i\_k);  
 Position->x = x\_k;  
 Position->y = y\_k;  
 Position->z = z\_k;  
}  
void sendStr(sub \*Subframes)  
{  
 string path = "in.txt";  
 ifstream fin;  
 fin.open(path);  
 if(fin.is\_open()) {  
 while (!fin.eof()) {  
 uint32\_t slot;  
 uint32\_t slot\_SF1 = 0;  
 uint32\_t slot\_SF2 = 0;  
 uint32\_t slot\_SF3 = 0;  
 uint32\_t subFrameNum;  
 int svStr;  
 int svNum = 3;  
 string strSF;  
 string useless;  
 fin >> useless >> useless >> useless >> useless >> useless >> useless  
 >> svStr >> slot >> useless >> useless >> subFrameNum >> strSF;  
 if (svStr == svNum and slot>= 604800/6){  
 if (subFrameNum == 1)  
 {  
 slot\_SF1 = slot;  
 Subframes->sf1 = strSF;  
 }  
 else if (subFrameNum == 2)  
 {  
 slot\_SF2 = slot;  
 Subframes->sf2 = strSF;  
 }  
 else if (subFrameNum == 3)  
 {  
 slot\_SF3 = slot;  
 Subframes->sf3 = strSF;  
 }  
 if (slot\_SF1 + 1 == slot\_SF2 and slot\_SF2 + 1 == slot\_SF3) {  
 Subframes->slot = slot\_SF1;  
 return;  
 }  
 }  
 }  
 }  
 else  
 {  
 cout << "Can't open" << endl;  
 }  
 fin.close();  
}  
int32\_t convert\_eph(string sf, int32\_t Begin, int End)  
{  
 int32\_t ans = 0;  
 for (int i = Begin; i < End; i++) {  
 bool bit = (sf[i - 1] == '1');  
 ans = ans | bit;  
 if (i < End-1){  
 ans = ans<<1;  
 }  
 }  
 return ans;  
}  
int32\_t compl2int(uint32\_t ans, int Lenght)  
{  
 int32\_t Result = 0;  
 int32\_t Mask = 0;  
 if (Lenght < 32){  
 if (bool((1<<Lenght-1) & ans)){  
 for (int i = 0; i < 32 - Lenght + 1; i++) {  
 Mask |= 0x80000000 >> i;  
 }  
 ans |= Mask;  
 Result = ~(ans-1);  
 return -Result;  
 }  
 }  
 if (Lenght == 32){  
 if (bool((1<<31) & ans)){  
 Result = ~(ans-1);  
 return -Result;  
 }  
 }  
 return ans;  
}  
uint32\_t splitconvert\_eph(string sf, uint16\_t Begin, int End, uint16\_t Contin,  
 int End\_of\_Contin)//когда параметр лежит в разных кадрах  
{  
 uint32\_t ans = 0;  
 for (int i = Begin; i < End; i++) {  
 bool bit = (sf[i - 1] == '1');  
 ans = ans | bit;  
 ans = ans<<1;  
 }  
 for (int i = Contin; i < End\_of\_Contin; i++) {  
 bool bit = (sf[i - 1] == '1');  
 ans = ans | bit;  
 if (i < End\_of\_Contin-1){  
 ans = ans<<1;  
 }  
 }  
 return ans;  
}  
void saveEPH(Ephemeris\* ep)  
{  
 ofstream fout;  
 string path = "out.txt";  
 fout.open(path);  
 if(fout.is\_open()) {  
 fout << endl << "LNAV Ephemeris (slot = " << ep->slot << ") =" << endl;  
 fout << "\t\t Crs\t\t\t= " << ep->Crs << endl;  
 fout << "\t\t Dn\t\t\t= " << ep->Dn << endl;  
 fout << "\t\t M0\t\t\t= " << ep->M0 << "\t\t[deg]" << endl;  
 fout << "\t\t Cuc\t\t\t= " << ep->Cuc << endl;  
 fout << "\t\t e\t\t\t= " << ep->e << endl;  
 fout << "\t\t Cus\t\t\t= " << ep->Cus << endl;  
 fout << "\t\t sqrtA\t\t= " << ep->sqrtA << endl;  
 fout << "\t\t toe\t\t\t= " << ep->toe << endl;  
 fout << "\t\t Cic\t\t\t= " << ep->Cic << endl;  
 fout << "\t\t Omega0\t\t= " << ep->Omega0 << "\t\t[deg]" << endl;  
 fout << "\t\t Cis\t\t\t= " << ep->Cis << endl;  
 fout << "\t\t i0\t\t\t= " << ep->i0 << "\t\t[deg]" << endl;  
 fout << "\t\t Crc\t\t\t= " << ep->Crc << endl;  
 fout << "\t\t omega\t\t= " << ep->omega << "\t\t[deg]" << endl;  
 fout << "\t\t omegaDot\t\t= " << ep->OmegaDot << "\t[deg/s]" << endl;  
 fout << "\t\t iDot\t\t\t= " << ep->iDot << "\t[deg/s]" << endl;  
 fout << "\t\t Tgd\t\t\t= " << ep->Tgd << "\t\t\t[sec]" << endl;  
 fout << "\t\t toc\t\t\t= " << ep->toc << endl;  
 fout << "\t\t af2\t\t\t= " << ep->af2 << endl;  
 fout << "\t\t af1\t\t\t= " << ep->af1 << endl;  
 fout << "\t\t af0\t\t\t= " << ep->af0 << endl;  
 fout << "\t\t WN\t\t\t= " << ep->WN << endl;  
 fout << "\t\t IODC\t\t\t= " << ep->IODC << endl;  
 fout << "\t\t URA\t\t\t= " << ep->URA << endl;  
 fout << "\t\t Health\t\t= " << ep->Health << endl;  
 fout << "\t\t IODE2\t\t= " << ep->IODE2 << endl;  
 fout << "\t\t IODE3\t\t= " << ep->IODE3 << endl;  
 fout << "\t\t codeL2\t\t= " << ep->codeL2 << endl;  
 fout << "\t\t L2P\t\t\t= " << ep->L2P << endl;  
 }  
 else  
 {  
 cout << "Cant open" << endl;  
 }  
 fout.close();  
}  
void printEPH(Ephemeris\* ep)  
{  
 cout << endl << "LNAV Ephemeris (slot = " << ep->slot << ") =" << endl;  
 cout << "\t\t Crs = " << ep->Crs << endl;  
 cout << "\t\t Dn = " << ep->Dn << endl;  
 cout << "\t\t M0 = " << ep->M0 << "\t\t[deg]" << endl;  
 cout << "\t\t Cuc = " << ep->Cuc << endl;  
 cout << "\t\t e = " << ep->e << endl;  
 cout << "\t\t Cus = " << ep->Cus << endl;  
 cout << "\t\t sqrtA = " << ep->sqrtA << endl;  
 cout << "\t\t toe = " << ep->toe << endl;  
 cout << "\t\t Cic = " << ep->Cic << endl;  
 cout << "\t\t Omega0 = " << ep->Omega0 << "\t\t[deg]" << endl;  
 cout << "\t\t Cis = " << ep->Cis << endl;  
 cout << "\t\t i0 = " << ep->i0 << "\t\t[deg]" << endl;  
 cout << "\t\t Crc = " << ep->Crc << endl;  
 cout << "\t\t omega = " << ep->omega << "\t\t[deg]" << endl;  
 cout << "\t\t omegaDot = " << ep->OmegaDot << "\t[deg/s]" << endl;  
 cout << "\t\t iDot = " << ep->iDot << "\t\t[deg/s]" << endl;  
 cout << "\t\t Tgd = " << ep->Tgd << "\t\t\t[sec]" << endl;  
 cout << "\t\t toc = " << ep->toc << endl;  
 cout << "\t\t af2 = " << ep->af2 << endl;  
 cout << "\t\t af1 = " << ep->af1 << endl;  
 cout << "\t\t af0 = " << ep->af0 << endl;  
 cout << "\t\t WN = " << ep->WN << endl;  
 cout << "\t\t IODC = " << ep->IODC << endl;  
 cout << "\t\t URA = " << ep->URA << endl;  
 cout << "\t\t Health = " << ep->Health << endl;  
 cout << "\t\t IODE2 = " << ep->IODE2 << endl;  
 cout << "\t\t IODE3 = " << ep->IODE3 << endl;  
 cout << "\t\t codeL2 = " << ep->codeL2 << endl;  
 cout << "\t\t L2P = " << ep->L2P << endl;  
}  
void scale\_factor\_use(Ephemeris\* ep, sub \*data)//учет scale factor-а  
{  
 double deg2rad = pi / Semi\_circles;  
 ep->slot = data->slot;  
 ep->Crs = compl2int(convert\_eph(data->sf2,69,85),16)\*SF1;  
 ep->Dn = compl2int(convert\_eph(data->sf2,91,107),16)\*SF2\*Semi\_circles\*deg2rad;  
 ep->M0 = compl2int(splitconvert\_eph(data->sf2,107, 115, 121,145),32)\*SF3\*Semi\_circles\*deg2rad;  
 ep->Cuc = compl2int(convert\_eph(data->sf2,151,167),16)\*SF4;  
 ep->e = splitconvert\_eph(data->sf2,167, 175, 181, 205)\*SF5;  
 ep->Cus = compl2int(convert\_eph(data->sf2,211,227),16)\*SF4;  
 ep->sqrtA = splitconvert\_eph(data->sf2,227, 235, 241, 265)\*SF6;  
 ep->toe = convert\_eph(data->sf2,271,287)\*SF7;  
 ep->Cic = compl2int(convert\_eph(data->sf3,61,77),16)\*SF4;  
 ep->Omega0 = compl2int(splitconvert\_eph(data->sf3,77, 85, 91,115),32)\*SF3\*Semi\_circles\*deg2rad;  
 ep->Cis = compl2int(convert\_eph(data->sf3,121,137),16)\*SF4;  
 ep->i0 = compl2int(splitconvert\_eph(data->sf3,137, 145, 151,175),32)\*SF3\*Semi\_circles\*deg2rad;  
 ep->Crc = compl2int(convert\_eph(data->sf3,181,197),16)\*SF1;  
 ep->omega = compl2int(splitconvert\_eph(data->sf3,197, 205,  
 211,235),32)\*SF3\*Semi\_circles\*deg2rad;  
 ep->OmegaDot = compl2int(convert\_eph(data->sf3,241,265),24)\*SF2\*Semi\_circles\*deg2rad;  
 ep->iDot = compl2int(convert\_eph(data->sf3,279,293),14)\*SF2\*Semi\_circles\*deg2rad;  
 ep->Tgd = compl2int(convert\_eph(data->sf1,197,205),8)\*SF3;  
 ep->toc = compl2int(convert\_eph(data->sf1,219,235),16)\*SF7;  
 ep->af2 = compl2int(convert\_eph(data->sf1,241,249),8)\*SF8;  
 ep->af1 = compl2int(convert\_eph(data->sf1,249,265),16)\*SF2;  
 ep->af0 = compl2int(convert\_eph(data->sf1,271,293),22)\*SF3;  
 ep->WN = convert\_eph(data->sf1,61,71);  
 ep->IODC = splitconvert\_eph(data->sf1,83, 85, 211, 219);  
 ep->URA = convert\_eph(data->sf1,73,75);  
 ep->Health = ep->IODE2 = convert\_eph(data->sf1,73,79);  
 ep->IODE2 = convert\_eph(data->sf2,61,69);  
 ep->IODE3 = convert\_eph(data->sf3,271,279);  
 ep->codeL2 = bool (data->sf1[91]);  
 ep->L2P = bool (data->sf1[90]);  
}