НИУ «МЭИ»

Институт Радиотехники и электроники

Им. В.А. Котельникова

Аппаратура потребителей спутниковых радионавигационных систем

Курсовой проект

«Расчет траектории движения спутника GPS по данным с демодулятора его сигнала»

Часть 2 и 3

Студент: Коробков А.Ю.

Группа: ЭР-15-17

Преподаватель: Корогодин И.В.

Москва

2022

Оглавление

Цель работы

Изучение особенностей сигналов спутников GPS для определения положения спутника по данным с демодулятора его сигнала L1 C/A. На втором этапе происходит моделирование положения спутника и траектории (орбиты) его движения. На третьем этапе требуется разработать на языке C/C++ функцию расчета положения спутника GPS на заданное время по шкале UTC, минимизировать время её исполнения и количество затрачиваемой оперативной памяти.

Этап 2. Моделирование в среде Matlab

Исходные данные

Значения эфемерид, полученные в исходном сигнале и обработанные в первом пункте. Уточнение и проверка данных выполнено в RTKNAVI.

Решение

Исходные данные (эфемериды) для построения орбит получены в ходе выполнения моделирования в первой части КП и приведены на рисунке 1.

```
LNAV Ephemeris (slot = 221472010) =
        Crs     = 3,850000e+001
        Dn      = 1,511921e-009         [deg/s]
        M0      = -1,101883e+002        [deg]
        Cuc     = 2,538785e-006
        e       = 2,460024e-002
        Cus     = 1,490116e-006
        sqrtA   = 5,153582e+003
        toe     = 93584
        Cic     = -2,924353e-007
        Omega0  = -1,146463e+002        [deg]
        Cis     = 4,917383e-007
        i0      = 5,496088e+001         [deg]
        Crc     = 3,547188e+002
        omega   = -5,765251e+001        [deg]
        OmegaDot= -4,759227e-007        [deg/s]
        iDot    = 5,832135e-009         [deg/s]
        Tgd     = -1,024455e-008
        toc     = 93584
        af2     = 0,000000e+000
        af1     = 6,821210e-013
        af0     = 1,579938e-004
        WN      = 149
        IODC    = 2
        URA     = 0
        Health  = 0
        IODE2   = 2
        IODE3   = 2
        codeL2  = 1
        L2P     = 0
```

Рисунок 1 – Исходные данные

Для построения орбит используется отрытый ИКД (расположение: https://www.gps.gov/technical/icwg/IS-GPS-200M.pdf).

Дату и время начала моделирования можно определить по значению Toe или по RTKNAVI. Данный сигнал был получен 14.02.22 в 1:59:44 (примем как 2:00).

Согласно заданию, после моделирования необходимо вывести графическое отображение орбит. Оно представлено на рисунке 2.

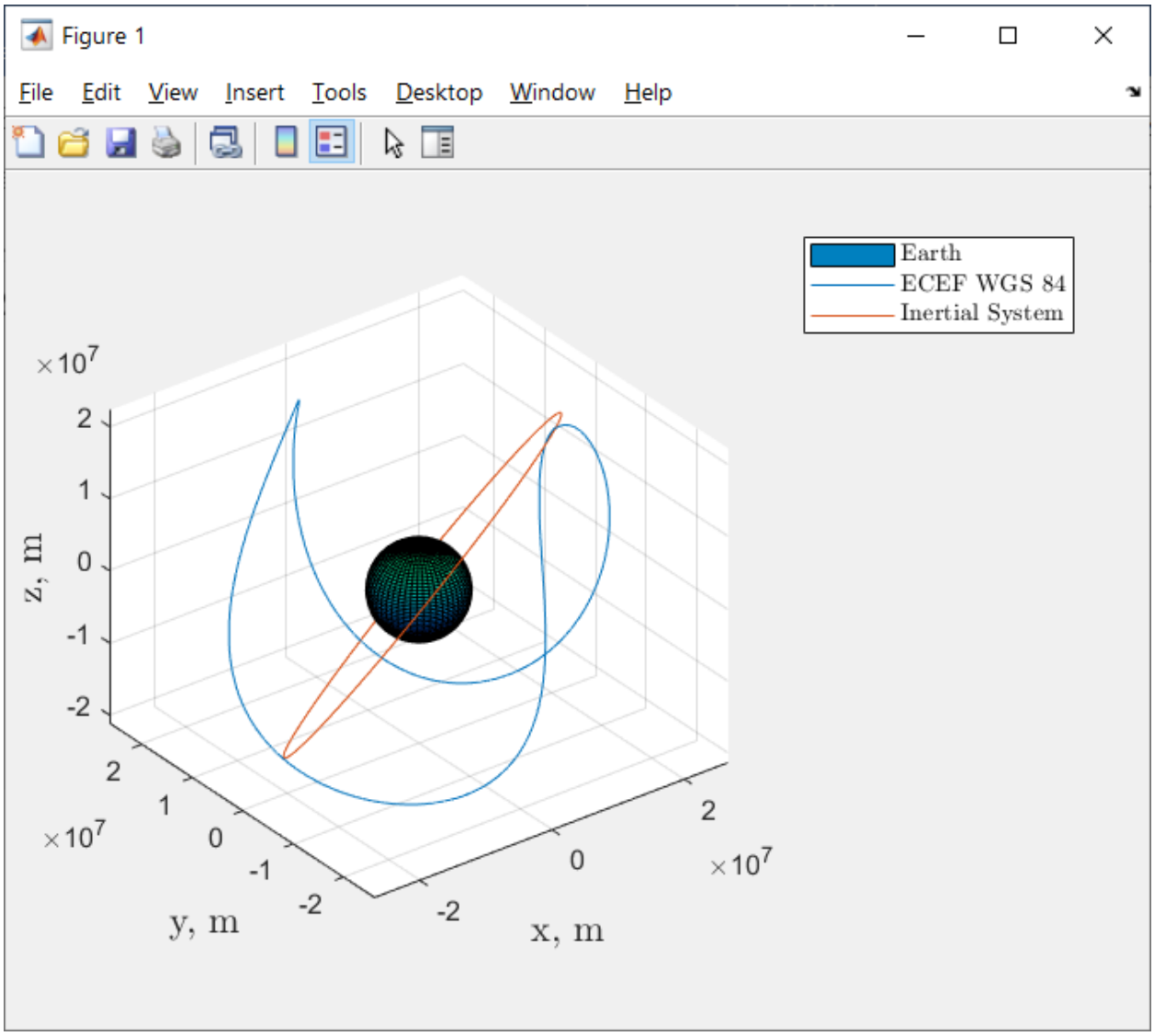

Рисунок 2 – Орбиты спутника GPS

Для проверки качества моделирования, построим график видимости искомого спутника (приведен на рисунке 3). Также воспользуемся внешней программой, выполняющей аналогичные функции и получим график для сравнения (приведен на рисунке 4).

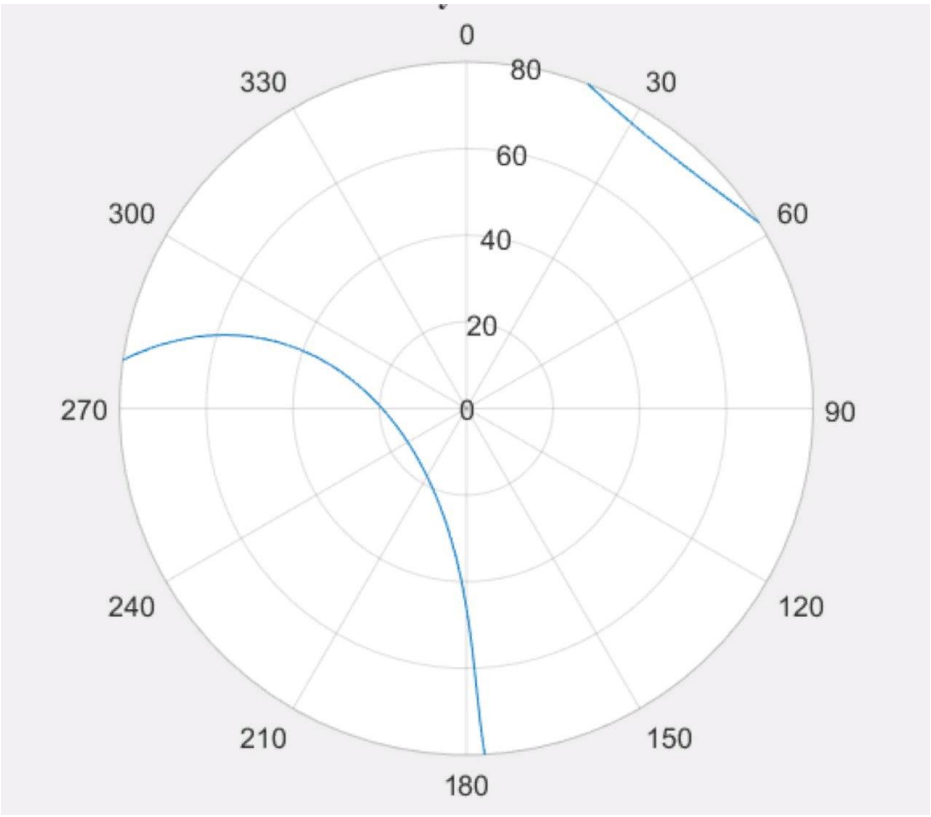

Рисунок 3 – График видимости НКА 21

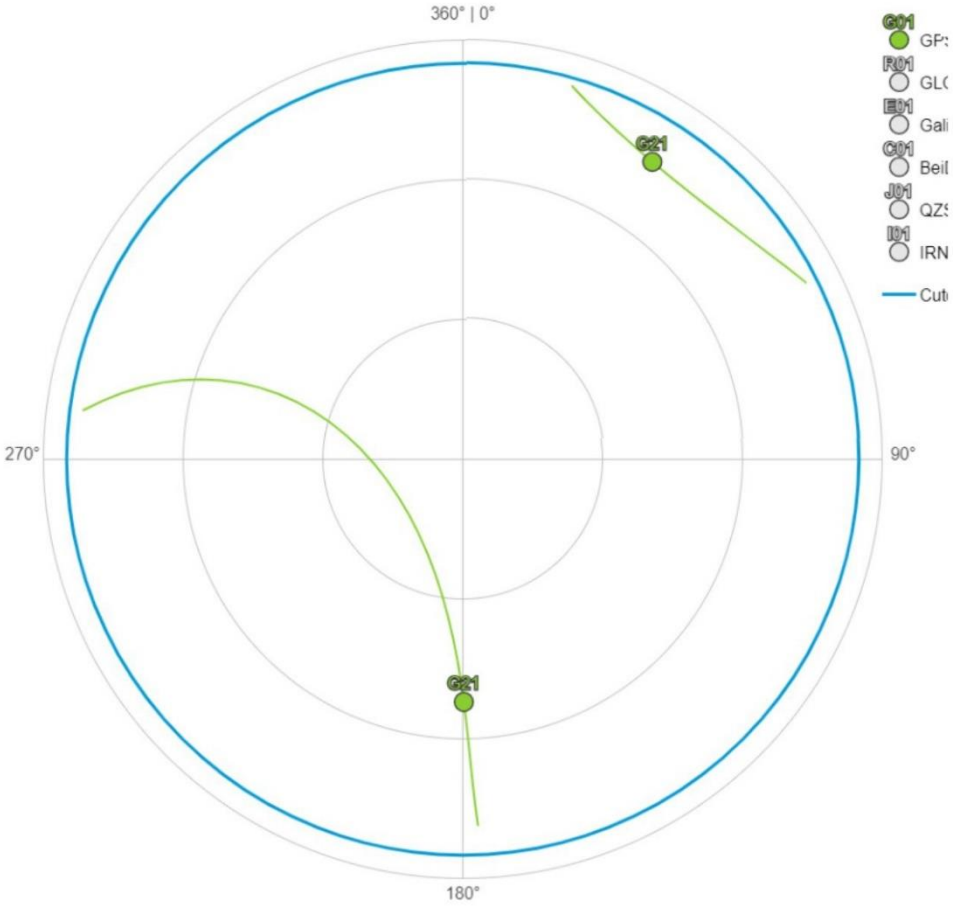

Рисунок 4 – График видимости спутника, из программы SkyView

Вывод

После выполнения второго этапа КР были рассчитаны и построены трехмерные графики орбит спутника GPS в различных СК. Так же были построены графики видимости заданного спутника. Из приведенных рисунков 3 и 4 видно, что расчет был проведен правильно, траектории движения КА, полученные разными способами, практически идентичны.

Этап 3. Моделирование в C++

Исходные данные

Значения эфемерид, полученные в исходном сигнале и обработанные в первом пункте. Вызов функции не должен приводить к выбросу исключений или утечкам памяти при любом наборе входных данных. Во время тестов должна вычисляться и выводиться средняя длительность исполнения функции. Требуется провести проверку на утечки памяти с помощью утилиты valgrind и/или memcheck.

Решение

Функция реализуется на языке программирования C++ с помощью компилятора Code::Blocks. Функция расчета расположим в том же файле, где расположен расчет главной функции программы первого пункта. Функция, помимо вычислений координат, так же считывает координаты из файла Malab, сравнивает их с рассчитанными, находит максимальную разницу, а также вычисляет общее время выполнения. Код реализации приведен в Приложении В.

Для точного расчета, в ходе выполнения моделирования, был создан алгоритм решения уравнения Кеплера, который приведен как формула (1).

$$\begin{aligned} &\text{Пока } \mathrm{E}_{0,i-1} - E_{0,i} > 1e-8 \\ &\quad E_{0,i-1} = E_{0,i} \\ &\quad E_{0,i} = M + e \cdot \sin\left(E_{0,i}\right) \\ &\text{Конец} \end{aligned} \tag{1}$$

После выполнения расчета, получим максимальную разницу координат и время выполнения моделирования (приведены на рисунке 5).

```
delta = 5.41926e-006(m)
program T = 1.501(sec)
```

Рисунок 5 – Результат работы третьей части программы

Далее проанализируем работу программы. Для этого воспользуемся встроенными инструментами отладки Microsoft Visual Studio. Воспользовавшись инструментами вкладки «производительность», получим потребляемую память программы и обнаружим возможность утечки памяти. Результат работы инструмента «Использование памяти» показан на рисунке 6.

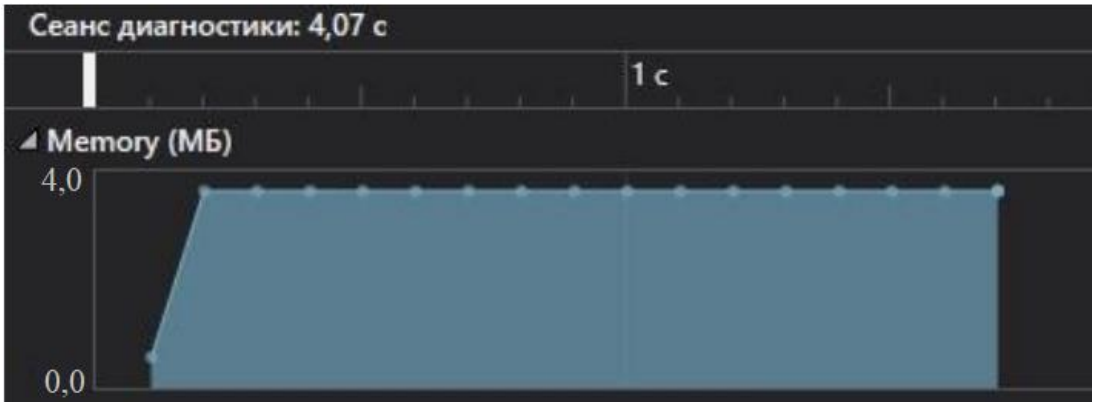

Рисунок 6 – Результаты работы инструмента «Использование памяти»

Вывод

В ходе выполнения третьего этапа была реализована на языке C/C++ функция расчета положения спутника GPS (21) на заданное время по шкале UTC. Функция сопровождается тестами решения уравнения Кеплера, и проверкой на утечки памяти.

Погрешность вычисления координат функцией и моделью составляет порядка 5.42e-6.

Приложение А

```
close all
clear all
clc


%% Эфемериды
SatNum = 21;
toe = 93584;
Crs = 3.850000e+001;
Dn = 2.7215e-07;
M0 = -1.101883e+002;
Cuc = 2.538785e-006;
e = 2.460024e-002;
Cus = 1.490116e-006;
sqrtA = 5.153582e+003;
Cic = -2.924353e-007;
Omega0 = -1.146463e+002;
Cis = 4.917383e-007;
i0 = 5.496088e+001;
Crc = 3.547188e+002;
omega = -5.765251e+001;
OmegaDot = -4.759227e-007;
iDot = 5.832135e-009;
Tgd = -1.024455e-008;
toc = 93584;
af2 = 0.000000e+000;
af1 = 6.821210e-013;
af0 = 1.579938e-004;
URA = 0;
IODE = 2;
IODC = 2;
codeL2 =1;
L2P = 0;
WN = 149;

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

%% Временной промежуток
startt = (24*0+2)*60*60; % 14.02.22  02:00
stopt = (24*1+2)*60*60; % 15.02.22

t_long = startt:1:stopt; % Временной промежуток (с шагом 1 с)

%% Параметры орбит
A = sqrtA^2;

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

```
for k = 1:length(t_long)

    t(k) = t_long(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
    Ee = M(k);
    E_old(k) = M(k)+1;

    for sch = 2:6 % В ИКД сказано что нужно минимум три
итерации, взял с запасом. Также можно с помощью while.
        Ee(sch) = M(k)+e*sin(Ee(sch-1));
    end
    E(k) = Ee(sch);
    sch = 0;
    % Истинная аномалия
    nu(k) = 2*atan(sqrt((1+e)/(1-e))*tan(E(k)/2));

    % Коррекция (не полная) возмущения второй гармоники
    c_corr(k) = cos(2*(omega+nu(k)));
    s_corr(k) = sin(2*(omega+nu(k)));

    % Аргумент широты
    u(k) = omega+nu(k)+Cuc*c_corr(k)+Cus*s_corr(k);

    r(k) = A*(1-e*cos(E(k)))+Crc*c_corr(k)+Crs*s_corr(k);
    % Наклонение
    i(k) = i0+iDot*t(k)+Cic*c_corr(k)+Cis*s_corr(k);

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

    % Орбитальные координаты
    xs = r(k)*cos(u(k));
    ys = r(k)*sin(u(k));

    % Координаты
    x(k) = xs*cos(lat(k)) - ys*cos(i(k))*sin(lat(k));
    y(k) = xs*sin(lat(k)) + ys*cos(i(k))*cos(lat(k));
    z(k) = ys*sin(i(k));%координаты фазового центра антенны во
время t



    %% ECI
```

```
    theta = omega_e*t(k);
    x1(k) = x(k)*cos(theta)-y(k)*sin(theta);
    y1(k) = x(k)*sin(theta)+y(k)*cos(theta);
    z1(k) = z(k);
end

%% График
Rz = 6371*10^3;
[x_sphere,y_sphere,z_sphere]=sphere(50);
x_Earh=Rz*x_sphere;
y_Earh=Rz*y_sphere;
z_Earh=Rz*z_sphere;

figure
surf(x_Earh,y_Earh,z_Earh); %Построение модели
colormap winter;
hold on
grid on
plot3(x, y, z)
plot3(x1, y1, z1)
xlabel('x, m', 'FontSize',14, 'Interpreter','latex')
ylabel('y, m', 'FontSize',14, 'Interpreter','latex')
zlabel('z, m', 'FontSize',14, 'Interpreter','latex')
hold off
legend('Earth','ECEF WGS 84','Inertial
System','Interpreter','latex');
axis equal;

filename = "output_data.txt";
out = fopen(filename, 'w+');
for i = 1:86400
    fprintf(out, '%6.0f %10.8f %10.8f %10.8f \n ', i,
x_coord(1,i), y_coord(1,i),z_coord(1,i)); % запись в файл
end
fclose(out);
```

Приложение В

```
#include <iostream>

#include <fstream>

#include <string>

#include <stdlib.h>

#include <cmath>

#include <stdio.h>

#include <windows.h>

#include <ctime> //библиотека для отобр длит вычисл
```

```
#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 // структ рабочего фрейма
   {
    string sf1;
    string sf2;
    string sf3;
    uint32_t slot;
   };
   struct Coordinates
   {
    double x;
    double y;
    double z;
   };
   void sendStr(sub *Subframes); //выделение эфемерид
   void scale_factor_use(Ephemeris* ep, sub *data); //преобраз эфемерид в необх типы данных
   int32_t convert_eph(string sf, int32_t Begin, int End);//преобраз эфемерид в необх типы данных
   void printEPH(Ephemeris* ep);//вывод в окно знач эфемерид
   void saveEPH(Ephemeris* ep);//сохр знач эфемерид
```

```
void CalcCoord(Ephemeris* ep,uint32_t t, Coordinates *Position);//расчет коорд по ИКД
int main(void)
{
 uint32_t begin = clock();//ставим начало отсчета времени вычислений
 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;//очистка
    uint32_t end = clock();//конечного времени расчета
    uint32_t calculation_time = end - begin;//расчет времени исполнения
    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-a
{
 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]);
   }
```