Из первой части (код представлен в приложении 2)

Получаем данны

LNAV Ephemeris (slot = 221472010) =

Crs = 8.812500e+00

Dn = 1.487138e-09 [deg/s]

M0 = -1.740945e+01 [deg]

Cuc = 4.526228e-07

e = 1.435236e-03

Cus = 9.626150e-06

sqrtA = 5.153648e+03

toe = 86400

Cic = -9.313226e-09

Omega0 = 1.321879e+02 [deg]

Cis = -1.490116e-08

i0 = 5.464168e+01 [deg]

Crc = 1.896875e+02

omega = 1.728546e+02 [deg]

OmegaDot= -4.551725e-07 [deg/s]

iDot = -2.879233e-08 [deg/s]

Tgd = -7.450581e-09

toc = 86400

af2 = 0.000000e+00

af1 = -4.547474e-12

af0 = -8.354103e-05

WN = 149

IODC = 114

URA = 0

Health = 0

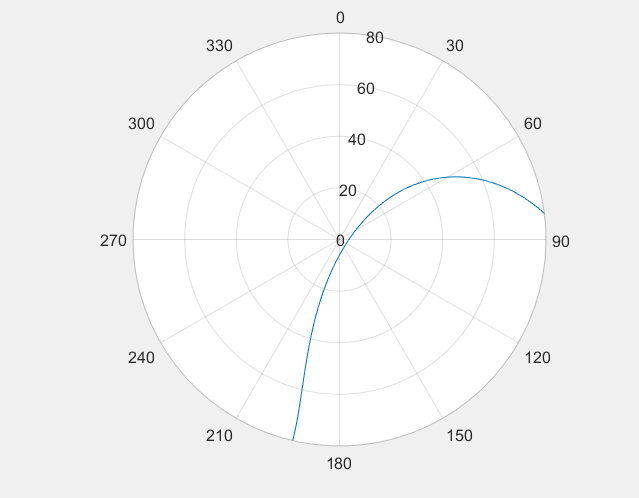
IODE2 = 114

IODE3 = 114

codeL2 = 1

L2P = 0

В результате получаем



Приложение1

Код для маткада

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]);

Приложение 1

Код первой части

#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 );  
}