НИУ МЭИ

ИРЭ им. В.А. Котельникова

Кафедра радиотехнических систем

Курсовой проект по дисциплине

«Аппаратура потребителей спутниковый радионавигационных систем»

Выполнила:

Студентка

Климова А.П.

Группа:

ЭР-15-14

Проверил

Корогодин И.В.

Москва, 2019

**Этап 2. Моделирование**

Требуется реализовать на языке Matlab или Python функцию расчета положения спутника GPS на заданный момент по шкале GPST. В качестве источника эфемерид использовать предоставленный на предыдущем этапе файл с логами за 13.02.19

13 февраля в 18:00 спутник был виден лучше всего. Для этого времени .

Время по шкале GPS записывается в формате: WN:TOW.

Где WN – номер недели начиная с 6 января 1980 г, по модулю 1024 (10 бит);

TOW –количество секунд от начала текущей недели.

WN = 2040.

Тогда TOW будет равнятся TOW=2\*24\*3600+18\*3600=172800+64800=237600

Время по шкале GPS: 2040: 237600.

Файл эфемерид на заданный момент времени для спутника №24 выглядит следующим образом:

2.10 N: GPS NAV DATA RINEX VERSION / TYPE

CONVBIN 2.4.3 20190220 192733 UTC PGM / RUN BY / DATE

log: C:\Users\Anna\Documents\Linux\LOGZZZ\BINR\_evening.bin COMMENT

format: NVS BINR COMMENT

END OF HEADER

24 19 2 13 14 0 0.0 -.621117651463E-04 -.568434188608E-12 .000000000000E+00

.131070000000E+05 .620625000000E+02 .499342190860E-08 .951895577860E+00

.344403088093E-05 .816986314021E-02 .973232090473E-05 .515373414612E+04

.309600000000E+06 -.596046447754E-07 .198684878318E+01 .104308128357E-06

.939133299974E+00 .178218750000E+03 .594741691139E+00 -.834249035535E-08

-.275011455331E-09 .100000000000E+01 .204000000000E+04 .000000000000E+00

.240000000000E+01 .000000000000E+00 .279396772385E-08 .510000000000E+02

.302988000000E+06 .000000000000E+00

В соответствии с форматом RINEX файлов запишем в таблицу необходимые для расчеты положений КА параметры орбиты (табл.1).

|  |  |
| --- | --- |
|  | 0. 309600000000E+06 |
|  | 0. 515373414612E+04 |
| e, эксцентриситет | 0. 816986314021E-02 |
|  | 0. 951895577860E+00 |
|  | 0. 594741691139E+00 |
|  | 0. 939133299974E+00 |
|  | 0. 198684878318E+01 |
|  | 0. 499342190860E-08 |
| IDOT, скорость изменения наклонения | -0. 275011455331E-09 |
|  | -0. 834249035535E-08 |
|  | 0. 344403088093E-05 |
|  | 0. 973232090473E-05 |
|  | 0. 178218750000E+03 |
|  | 0. 620625000000E+02 |
|  | -0. 596046447754E-07 |
|  | 0. 104308128357E-06 |

Табл.1 Таблица использованных эфемерид.

По данным табл.1 в соответствии с алгоритмом расчета координат в системе GPS (см. приложение) были получены координаты КА №24 на заданный интервал времени (18:00:00 – 00:00:00). Полученные траектории для двух случаев (Первый – в неинерциальной СК, с учетом вращения земли; второй – в инерциальной СК, где ось координат зафиксирована) изображены на рис. 1:

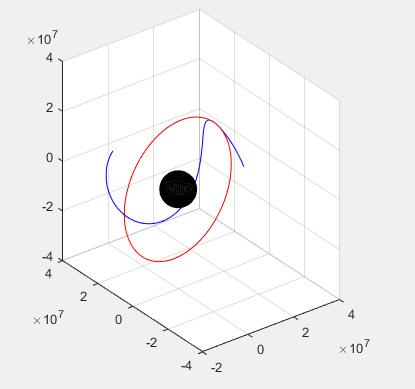


Рис. 1. Траектория движения спуника, голубая линия – для инерциальной СК, красная – для неинерциальной СК

На рис 2. изображен SkyView на заданный момент времени, полученный с помощью GNSS Planing Online SkyView, а на рис. 3 изрбражен SkyView, построенный в MATLAB:

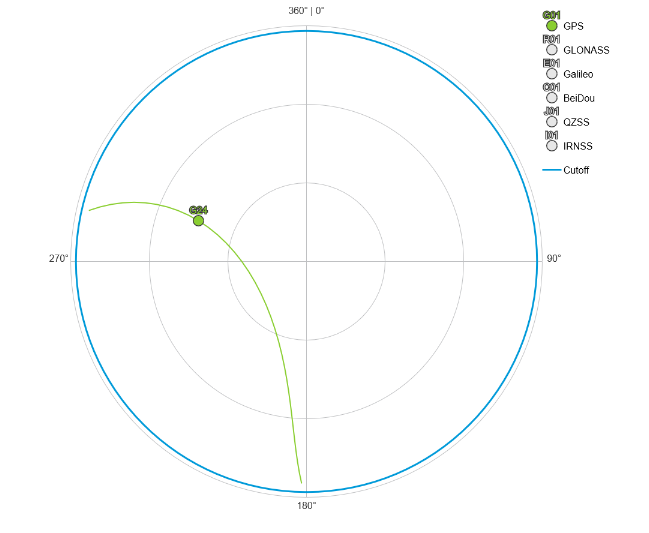


Рис. 2. SkyView на заданный момент времени

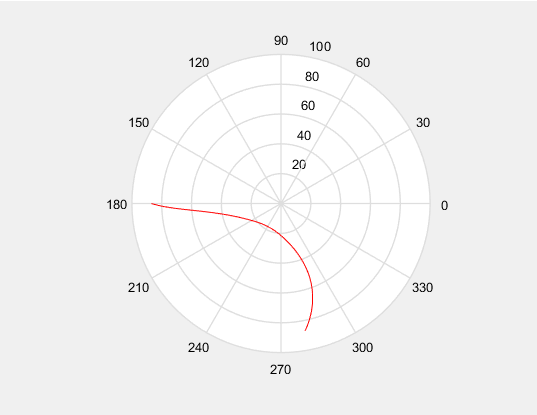


Рис. 3. Траектория заданного спутника на карте

К счастью, по рис. 2 и рис.3 можно утверждать, что заданный спутник в заданный интервал времени был виден и его сигнал принимался антенной, расположенной на крыше корпуса Е МЭИ. Видно, что полученный SkyView совпадает с построенным в MATLAB по эфемеридам НКА 24.

**Приложение:**

Ниже приведен код программы в Matlab, производящий расчет траектории КА.

clear all;

close all;

tic;

% считывание эфемерид из файла BINR\_evening.nav на НКА с заданным номером

delimiterIn = ' ';

headerlinesIn = 5;

filename = 'C:\Users\Anna\Documents\Linux\LOGZZZ\BINR\_evening.nav';

A = importdata(filename,delimiterIn,headerlinesIn);

ephemerides = A.data(:,1:6);

PRN = 24;

k = 0;

numbers = zeros(size(ephemerides,1)/8);

time = zeros(size(ephemerides,1)/8);

for i = 1:8:size(ephemerides,1)

if ephemerides(i,1) == PRN

k = k + 1;

numbers(k) = i;

time(k) = ephemerides(i,5);

end

end

nomer = 1;

% запись необходимых эфемерид в матрицу

R = ephemerides((numbers(nomer)+1):(numbers(nomer)+5),:);

% запись эфемерид в текстовый файл

f = fopen('C:\Users\Anna\Documents\Linux\LOGZZZ\ephemeridy.txt','w');

fprintf(f,'%f %f %f %f \n',R);

fclose(f);

% задание эфемерид для расчета

toe = R(3,1);

sqrt\_a = R(2,4);

e = R(2,2);

M0 = R(1,4);

omega = R(4,3);

i0 = R(4,1);

OMEGA0 = R(3,3);

delta\_n = R(1,3);

IDOT = R(5,1);

OMEGADOT = R(4,4);

cuc = R(2,1);

cus = R(2,3);

crc = R(4,2);

crs = R(1,2);

cic = R(3,2);

cis = R(3,4);

% константы

mu=3.986004418E14;

OMEGADOTe = 7.2921151467E-5;

% АЛГОРИТМ РАСЧЕТА ПОЛОЖЕНИЯ СПУТНИКА В GPS

A = (sqrt\_a)^2;

t = ((84-3)\*3600):0.1:((96-3)\*3600); % тут за 12 часов надо

N = size(t,2);

for k = 1:N

tk = t(k) - toe;

if tk > 302400

tk = tk - 604800;

end

if tk < -302400

tk = tk + 604800;

end

Mk = M0 + (sqrt(mu/A^3) + delta\_n) \* tk;

% решение уравнения Кеплера

m=1;

w(1)=0;

while 1

w(m+1)=Mk+e\*sin(w(m));

if abs(w(m+1)-w(m))<10^(-5)

break

end

m=m+1;

end

Ek=w(m+1);

vk = atan2((sqrt(1-e^2)\*sin(Ek)),cos(Ek)-e);

uk = omega + vk + cuc\*cos(2\*(omega+vk))+cus\*sin(2\*(omega+vk));

rk = A\*(1-e\*cos(Ek))+crc\*cos(2\*(omega+vk))+crs\*sin(2\*(omega+vk));

ik = i0+IDOT\*tk+cic\*cos(2\*(omega+vk))+cis\*sin(2\*(omega+vk));

x\_k = rk\*cos(uk);

y\_k = rk\*sin(uk);

lk = OMEGA0 - OMEGADOTe\*toe+(OMEGADOT-OMEGADOTe)\*tk;

lk1 = OMEGA0 - OMEGADOTe\*toe;

% расчет координат в неинерциальной системе

x(k) = x\_k\*cos(lk)-y\_k\*cos(ik)\*sin(lk);

y(k) = x\_k\*sin(lk)+y\_k\*cos(ik)\*cos(lk);

z(k) = y\_k\*sin(ik);

% расчет координат в инерциальной системе

x1(k) = x\_k\*cos(lk1)-y\_k\*cos(ik)\*sin(lk1);

y1(k) = x\_k\*sin(lk1)+y\_k\*cos(ik)\*cos(lk1);

z1(k) = y\_k\*sin(ik);

end

% построение графиков

[X,Y,Z] = sphere(50);

surf(X\*6400000,Y\*6400000,Z\*6400000)

hold on

plot3(x,y,z,'b')

plot3(x1,y1,z1,'r')

grid on

% colormap(gray)

daspect([1 1 1])

% сохранение траектории в текстовый файл

F = [x',y',z'];

f = fopen('C:\Users\Anna\Documents\Linux\LOGZZZ\testmatlab.txt','w');

fprintf(f,'%f %f %f\n',F');

fclose(f);

hold on

% координаты Москвы, расчитанные в первом этапе

position = [2835903.37, 2192470.92, 5265921.01];

plot3(position(1),position(2),position(3),'\*');

lat = atan(position(3)/sqrt(position(1)^2+position(2)^2));

lon = acos(position(1)/sqrt(position(1)^2+position(2)^2));

h = 180.59;

% пересчет координат из глобальной неинерциальной СК к локальной (ENU)

for k = 1:N

[x0(k) y0(k) z0(k)] = ecef2enu(x(k),y(k),z(k),lat,lon,h,wgs84Ellipsoid,'radians');

if z0(k)>0

teta(k) = atan(sqrt(x0(k)^2+y0(k)^2)/z0(k));

r(k) = sqrt(x0(k)^2+y0(k)^2+z0(k)^2);

if x0(k) > 0

phi(k) = atan(y0(k)/x0(k));

elseif (x0(k)<0)&&(y0(k)>0)

phi(k) = -atan(y0(k)/x0(k))+3\*pi/2;

elseif (x0(k)<0)&&(y0(k)<0)

phi(k) = -atan(y0(k)/x0(k))-pi/2;

end

else teta(k) = NaN;

r(k) = NaN;

phi(k) = NaN;

end

k;

end

% Построение SkyView

figure;

polar(phi,teta\*180/pi-pi,'r')

toc;

**Этап 3. Реализация**

Требуется разработать на языке С/С++ функцию расчета положения спутника GPS на заданное время по шкале GPST, минимизируя время её исполнения и количество затрачиваемой оперативной памяти. Вызов функции не должен приводить к выбросу исключений или утечкам памяти при любом наборе входных данных.

**Реализация на языке С++**

Реализуем написанный на предыдущем этапе код на языке С++. Будем создавать библиотеку для расчета траектории НКА сразу в каркасе программного модуля на базе системы QMake (QT).

Наш код будет иметь две функции:

1. Функция решения уравнения Кеплера, входными параметрами для которой являются: cредняя аномалия, эксцентриситет, точность (для итерационного метода).
2. Функция расчета положения спутника. Входные параметры: эфемериды, время в секундах от начала времени эфемерид, на которое необходимо расчитать координаты НКА. Выходным параметром является три массива координат X,Y,Z.

Функция решения уравнения Кеплера вызывается в ходе расчета координат НКА.

Итого, были получены следующие коды

Kepler.h

#ifndef KEPLER\_H

#define KEPLER\_H

double **kepler**(double Mk, double e, double eps);

#endif /\* #ifndef KEPLER\_H \*/

Gpssvpos.h

#ifndef GPSSVPOS\_H

#define GPSSVPOS\_H

typedef struct {

double X;

double Y;

double Z;

} Vec3D;

typedef struct {

double toe;

double sqrt\_a;

double e;

double M0;

double omega;

double i0;

double OMEGA0;

double delta\_n;

double IDOT;

double OMEGADOT;

double cuc;

double cus;

double crc;

double crs;

double cic;

double cis;

double mu;

double OMEGADOTe;

} Eph;

int **getSVPos**(double t, Eph eph, Vec3D\* pos);

#endif /\* #ifndef KEPLER\_H \*/

Kepler.cpp

#include <math.h>

#include <kepler.h>

double **kepler**(double Mk,double e, double eps)

{

int m = 0;

double w[10];

w[0] = 0;

do

{

w[m+1] = Mk + e\*sin(w[m]);

m++;

} while (fabs(w[m]-w[m-1])>eps);

return w[m];

}

Gpssvpos.cpp

#include <gpssvpos.h>

#include <kepler.h>

#include <math.h>

int **getSVPos**(double t, Eph eph, Vec3D\* pos) {

double Mk = eph.M0 + (sqrt(eph.mu/pow(eph.sqrt\_a,6)) + eph.delta\_n)\*t;

double eps = pow(10,-5);

double Ek = kepler(Mk,eph.e,eps);

double vk = atan2((sqrt(1 - pow(eph.e,2))\*sin(Ek)),cos(Ek) - eph.e);

double uk = eph.omega + vk + eph.cuc\*cos(2\*(eph.omega + vk)) + eph.cus\*sin(2\*(eph.omega + vk));

double rk = pow(eph.sqrt\_a,2)\*(1-eph.e\*cos(Ek))+eph.crc\*cos(2\*(eph.omega+vk))+eph.crs\*sin(2\*(eph.omega+vk));

double ik = eph.i0+eph.IDOT\*t+eph.cic\*cos(2\*(eph.omega+vk))+eph.cis\*sin(2\*(eph.omega+vk));

double x\_k = rk\*cos(uk);

double y\_k = rk\*sin(uk);

double lk = eph.OMEGA0 - eph.OMEGADOTe\*eph.toe+(eph.OMEGADOT-eph.OMEGADOTe)\*t;

pos->X = x\_k\*cos(lk)-y\_k\*cos(ik)\*sin(lk);

pos->Y = x\_k\*sin(lk)+y\_k\*cos(ik)\*cos(lk);

pos->Z = y\_k\*sin(ik);

return 0;

}

На основании составленных кодов был получен головной код main.cpp, представленный ниже

Main.cpp

#include <iostream>

#include <fstream>

#include <gpssvpos.h>

#include <kepler.h>

#include <ctime>

using namespace std;

int **main**()

{

Eph eph;

eph.toe = 309600;

eph.sqrt\_a = 5.15373414612e+03;

eph.e = 0.00816986314021;

eph.M0 = 0.951895577860;

eph.omega = 0.594741691139;

eph.i0 = 0.939133299974;

eph.OMEGA0 = 1.98684878318;

eph.delta\_n = 4.99342190860e-09;

eph.IDOT = -2.75011455331e-10;

eph.OMEGADOT = -8.34249035535e-09;

eph.cuc = 3.44403088093e-06;

eph.cus = 9.73232090473e-06;

eph.crc = 1.78218750000e+02;

eph.crs = 62.0625000000;

eph.cic = -5.96046447754e-08;

eph.cis = 1.04308128357e-07;

eph.mu = 3.986004418E14;

eph.OMEGADOTe = 7.2921151467E-5;

time\_t start, end;

double t = (84-3)\*3600 - eph.toe;

double delt = 0.1;

Vec3D pos;

double \*koord\_matlab = new double[3];

double max\_del = 0;

int i\_max = 0;

std::ofstream out;

out.open("C:\\rez\_cpp.txt");

time(&start);

std::ifstream in("C:\\testmatlab.txt");

if (!in)

{

std::cout << "File not open!" << std::endl;

} else {

std::cout << "File open!" << std::endl;

}

for (int i = 0; i < (12\*3600/delt); i++)

{

getSVPos(t, eph, &pos);

t += delt;

std::string koord\_str1 = std::to\_string(pos.X);

std::string koord\_str2 = std::to\_string(pos.Y);

std::string koord\_str3 = std::to\_string(pos.Z);

out << koord\_str1 << " " << koord\_str2 << " " << koord\_str3 << std::endl;

in >> koord\_matlab[0] >> koord\_matlab[1] >> koord\_matlab[2];

if (abs(pos.X-koord\_matlab[0]) > max\_del)

{

max\_del = abs(pos.X-koord\_matlab[0]);

i\_max = i;

}

if (abs(pos.Y-koord\_matlab[1]) > max\_del)

{

max\_del = abs(pos.Y-koord\_matlab[1]);

i\_max = i;

}

if (abs(pos.Z-koord\_matlab[2]) > max\_del)

{

max\_del = abs(pos.Z-koord\_matlab[2]);

i\_max = i;

}

}

time(&end);

in.close();

delete[] koord\_matlab;

koord\_matlab = nullptr;

double seconds = difftime(end, start);

std::string seconds1 = std::to\_string(seconds\*1000000/(12\*3600/delt));

cout << "Srednee vremia raschota, mcs: " << seconds1 << std::endl;

std::string max\_del1 = std::to\_string(max\_del);

cout << "Maximalnaia raznost koordinat: " << max\_del1 << std::endl;

std::string imax = std::to\_string(i\_max);

cout << "Nomer otcheta s max raznostiy: " << imax << std::endl;

out.close();

in.close();

}

Таким образом, данная программа считает максимальную разницу между рассчетами, производимыми на языке С++, и полученными в среде MatLab результатами. Результат выводится в текстовое окно, представленное на рисунке 4.

Видно, что время рассчета и сравнения полученных результатов составило 16,2 мс, а точность совпадения результатов оставила 10^-4 степени.

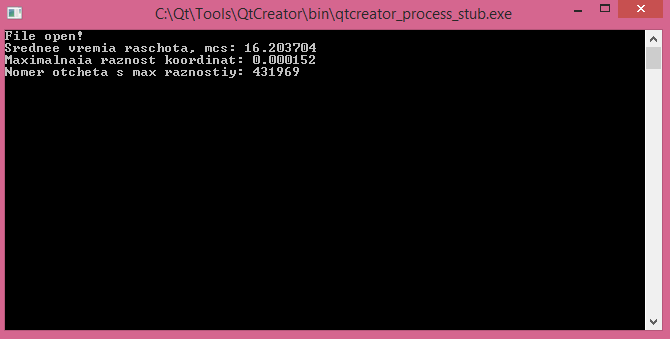


Рис. 4 Результат сравнения решения в QT (язык С++) и MatLab

**Контроль утечек**

Для контроля и устранения утечек памяти в среде QT используется утилита Heob. Скачав и установив ее с официального сайта, а также подключив в окне Анализ (рисунок 5) было получено, что данный код утечек не дает (окно отладчика после запуска анализа утечек не выдало сообщений о проблеме).

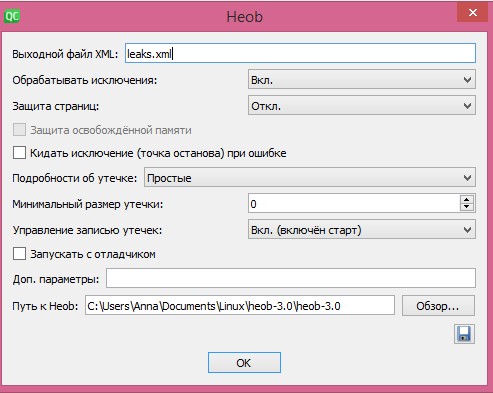


Рис. 5 Подключение контроля утечек Heob

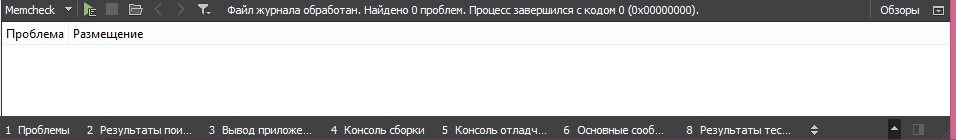


Рис. 6 Отсутствие ошибок после использования функции поиска утечек Heob

Чтобы проверить, правда ли программа анализа утечек работает корректно в исходном коде удалили строчки:

delete[] koord\_matlab;

koord\_matlab = nullptr;

В данных строчке производится удаление динамического массива и обнуление указателя на массив. На рисунке 7 представлено окно консоли отладчика, которое после анализа на утечки заполнилось ошибками.

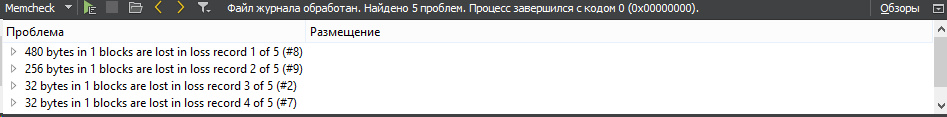


Рис. 7 Проверка работы функции поиска утечек Heob

**Вывод**

В ходе выполнения курсового проекта по дисциплине «Аппаратура потребителей спутниковых радионавигационных систем» были получены практические знания об обработке сигналов СРНС на основании измерений навигационных приемников. Была изучена структура принимаемого от спутника альманаха, а также методы простейшего моделирования его движения с помощью онлайн-сервиса Trimble GNSS Planning Online. После, при использовании программы RTKLib были получены эфемериды собственного спутника по данным RTKNAVI из состава RTKLIB. Во втором этапе курсового проекта было произведено моделирование траектории движения спутника по полученным эфемеридам из файла формата .nav. На этапе выполнения третьего и завершающего этапа курсового проекта была произведена вторичная обработка движения спутника с помощью кода, написанного на языке программирования С++ при помощи программы QMake. Полученный результат был очень близок к результату, полученному во втором этапе, а также был произведен анализ утечек, давший отрицательный результат.