Московский Энергетический Институт (Технический Университет)

Институт радиотехники и электроники

Кафедра радиотехнических систем

Курсовой проект по курсу “Аппаратура потребителей СРНС”

Разработка модуля расчета координат спутника GPS

Руководитель проекта:

Корогодин И. В.

Автор проекта:

Студент гр. ЭР-15-14 Цырульникова Л. А.

Оценка _____

Члены комиссии _____

_____

_____

“_____”_____ 2019 г.

Москва 2019

Оглавление

ЭТАП I. Использование сторонних средств

1. Описание процесса использования RTKLIB

В пакете RTKLIB для первой части курсового расчета использовали такие программы, как RTKLIB и RTKCONV.

Программа RTKNAVI позволяет вывести таблицу текущих и предыдущих эфемерид. После запуска программы загружаем бинарный файл с данными со спутника. Для этого в верхней панели, справа от GPST, нажимаем на кнопку I, после чего открывается окно Input Streams. Отмечаем галочкой Rover (приемник), выбираем Type–> File, Format –> NVS BINR, в строке Input File Paths задаем место нахождения бинарного файла (BINR_evening или BINR_morning). Нажимаем «ОК». В окне RTKNAVI ver.2.4.2 запускаем программу (нажимаем Start). Далее нажимаем на маленький квадратик над кнопкой Start. В верхней панели в выпадающем меню выбираем Nav GPS. По окончанию работы нажать ‘Stop’ и ‘Exit’.

Программа RTKCONV позволяет конвертировать бинарный файл в текстовый формат RINEX, в частности получить текстовый nav-файл с эфемеридами GPS. Запускаем программу. В строке “RTSM, RCV RAW or RINEX OBS ?” выбираем бинарный файл, который хотим конвертировать. В строке Output Directory выбираем место, куда нужно сохранить полученные, конвертированные файлы. В панели “RINEX OBS/NAV/GNAV/HNAV/QNAV/LNAV and SBS” указываем нужные нам форматы, которые хотим получить, в нашем случае это nav формат. По окончанию работы нажать “Exit”.

2. Эфемериды собственного спутника по данным RTKNAVI из состава RTKLIB (спутник GPS №15)

Утро:

SAT	PRN	Statu	IODE	IODC	Accu	Hea	Toe	Toc	Ttrans
G15	15	-	6168	24	0	00	2019/02/13 14:00:00	2019/02/13 14:00:00	-

A (m)	e	i0 (deg)	OMEGA0 (d	omega (deg	M0 (deg)	deltan (deg/s	OMEGAdot (d	IDOT (deg/s)
26559933.663	0.01121171	53.17113	51.38077	43.73662	111.08682	3.2150E-07	-4.8607E-07	-1.4325E-08

af0 (ns)	af1 (ns/s	af2 (ns/s	TGD (ns)	BGD5a(ns	BGD5b(ns
-328027.6	0.0017	0.0000	-10.7	0.0	0.0

Cuc(rad)	Cus(rad)	Crc(m)	Crs(m)	Cic(rad)	Cis(rad)	Code	Flag
-9.8161E-07	8.2161E-06	2.0747E+02	-1.7781E+01	-2.7195E-07	-1.1362E-07	1	0

Рисунок 1 - Эфемериды спутника (утро)

Вечер:

SAT	PRN	Statu	IODE	IODC	Accu	Hea	Toe	Toc	Ttrans
G15	15	-	6168	24	0	00	2019/02/13 14:00:00	2019/02/13 14:00:00	2019/03/28 08:13:18

A (m)	e	i0 (deg)	OMEGA0 (d	omega (deg	M0 (deg)	deltan (deg/s	OMEGAdot (d	IDOT (deg/s)
26559933.663	0.01121171	53.17113	51.38077	43.73662	111.08682	3.2150E-07	-4.8607E-07	-1.4325E-08

af0 (ns)	af1 (ns/s	af2 (ns/s	TGD (ns)	BGD5a(ns	BGD5b(ns
-328027.6	0.0017	0.0000	-10.7	0.0	0.0

Cuc(rad)	Cus(rad)	Crc(m)	Crs(m)	Cic(rad)	Cis(rad)	Code	Flag
-9.8161E-07	8.2161E-06	2.0747E+02	-1.7781E+01	-2.7195E-07	-1.1362E-07	1	0

Рисунок 2 - Эфемериды спутника (вечер)

Эфемериды в вечернее и утреннее время совпадают.

3. Эфемериды собственного спутника в nav-файле RINEX

BINR_evening

```
15 19  2 13 14  0  0.0 -.328027635813E-03  .170530256582E-11  .000000000000E+00
     .616800000000E+04 -.177812500000E+02  .561130525725E-08  .193883079254E+01
    -.981613993645E-06  .112117107492E-01  .821612775326E-05  .515363305473E+04
     .309600000000E+06 -.271946191788E-06  .896763633119E+00 -.113621354103E-06
     .928011329225E+00  .207468750000E+03  .763347995639E+00 -.848356766036E-08
    -.250010413937E-09  .100000000000E+01  .204000000000E+04  .000000000000E+00
     .240000000000E+01  .000000000000E+00 -.107102096081E-07  .240000000000E+02
     .302718000000E+06  .000000000000E+00
```

Рисунок 3 - Эфемериды в nav-файле

4. График угла места собственного спутника от времени по данным Trimble GNSS Planning Online на заданный интервал времени: с 12:00 13.02.19 до 00:00 14.02.19.

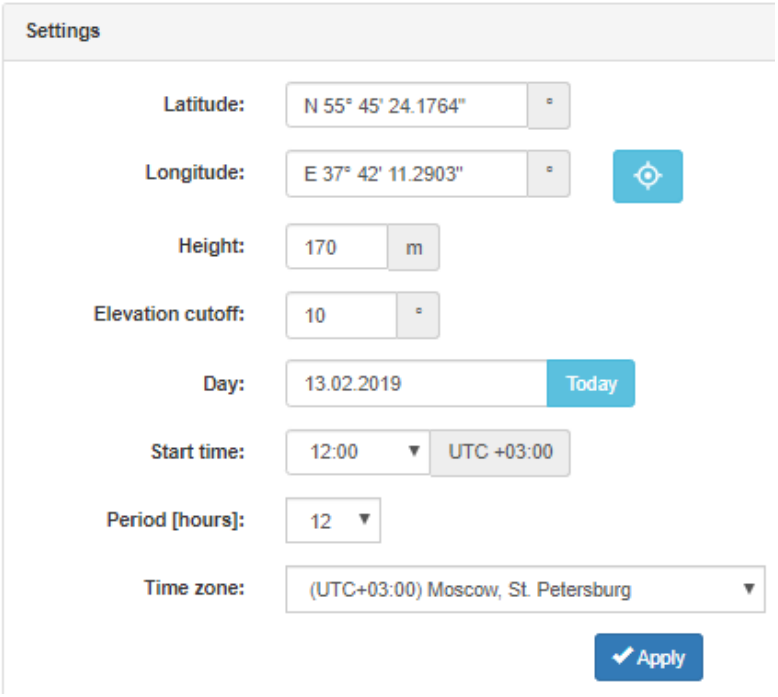

Рисунок 4 - Настройки на заданное время, место и продолжительность

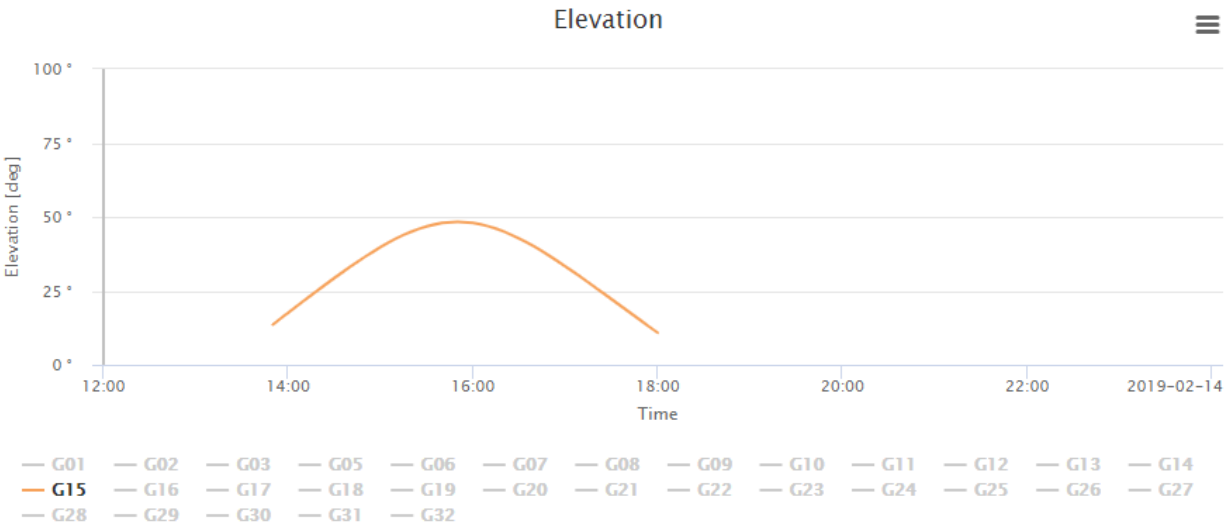

Рисунок 5 - График угла места

5. SkyView по данным Trimble GNSS Planning Online на заданный интервал времени: с 12:00 13.02.19 до 00:00 14.02.19.

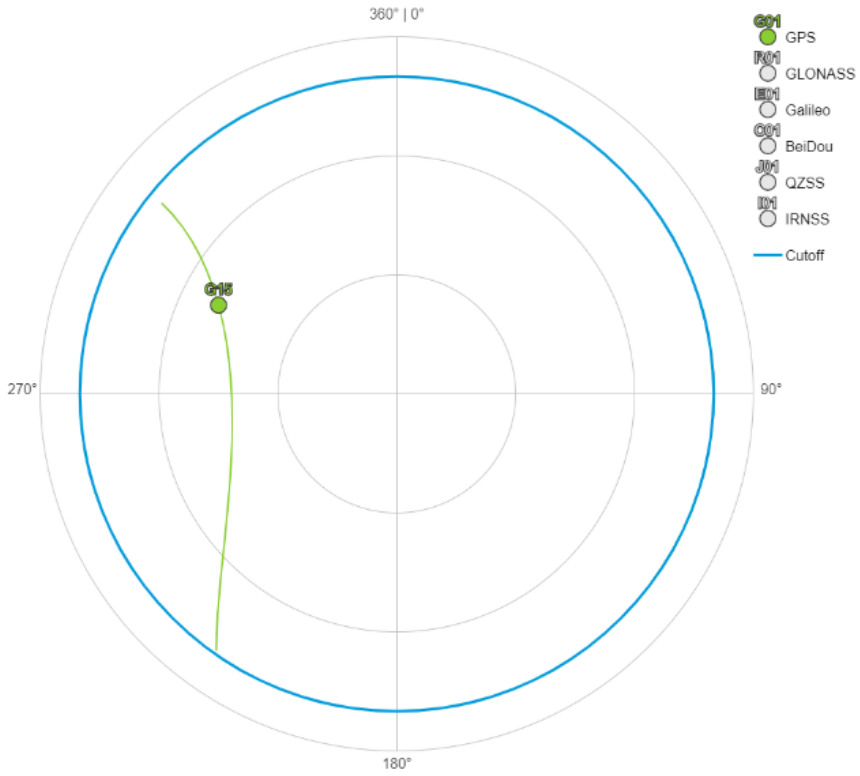

Рисунок 6 - SkyView

ЭТАП II. Моделирование

Постановка задачи:

Требуется реализовать на языке Matlab или Python функцию расчета положения спутника GPS на заданный момент по шкале GPST. 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 временному интервалу с 12:00 13.02.19 до 00:00 14.02.19. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал.

Построить SkyView за указанный временной интервал (напоминаю, антенна на крыше корпуса Е) и сравнить результат с Trimble GNSS Planning Online, полученный на прошлом этапе.

2. Результаты модулирования в Matlab

Для расчетов координат использовалось ИКД системы GALILEO, т.к. алгоритм идентичен алгоритму GPS, но при этом он более наглядный.

Parameter	Definition
M_0	Mean anomaly at reference time
Δn	Mean motion difference from computed value
e	Eccentricity
$A^{1/2}$	Square root of the semi-major axis
Ω_0	Longitude of ascending node of orbital plane at weekly epoch***
i_0	Inclination angle at reference time
ω	Argument of perigee
$\dot{\Omega}$	Rate of change of right ascension
$\dot{i}$	Rate of change of inclination angle
C_{uc}	Amplitude of the cosine harmonic correction term to the argument of latitude
C_{us}	Amplitude of the sine harmonic correction term to the argument of latitude
C_{rc}	Amplitude of the cosine harmonic correction term to the orbit radius
C_{rs}	Amplitude of the sine harmonic correction term to the orbit radius
C_{ic}	Amplitude of the cosine harmonic correction term to the angle of inclination
C_{is}	Amplitude of the sine harmonic correction term to the angle of inclination
t_{0e}	Ephemeris reference time

Рисунок 7 – Эфемериды, обозначения

Constant	Description
π = 3.1415926535898	Ratio of a circle's circumference to its diameter
μ = 3.986004418 × 10^{14} m^3/s^2	Geocentric gravitational constant
ω_E = 7.2921151467 × 10^{-5} rad/s	Mean angular velocity of the Earth
c = 299792458 m/s	Speed of light

Рисунок 8 - константы

Таким образом, эфемериды на заданный НКА (15):

```
mu = 3.986004418e+14;
we = 7.2921151467e-5;
c = 299792458;


A = 26559933.663;
e=0.01121171;
i0=deg2rad(53.17113);
W0=deg2rad(51.38077);
w=deg2rad(43.73662);
M0=deg2rad(111.08682);
deltan=deg2rad(3.2150e-7);
Wdot=deg2rad(-4.8607e-07);
idot=deg2rad(-1.4325e-8);

time_of_week=302418;
t=time_of_week;
```

Для расчета координат НКА воспользуемся алгоритмом:

Computation	Description
$A=(A^{1/2})^2$	Semi-major axis
$n_0=\sqrt{\frac{\mu}{A^3}}$	Computed mean motion (rad/s)
$t_k=t-t_{0e}$*	Time from ephemeris reference epoch
$n=n_0+\Delta n$	Corrected mean motion
$M=M_0+nt_k$	Mean anomaly
$M=E-e\sin(E)$	Kepler's Equation for Eccentric Anomaly E (may be solved by iteration)

Computation	Description
$\nu = \tan^{-1}\left\{\frac{\sin\nu}{\cos\nu}\right\}$ $= \tan^{-1}\left\{\frac{\sqrt{1-e^2}\sin E/(1-e\cos E)}{(\cos E - e)/(1-e\cos E)}\right\}$	True Anomaly
$\Phi = \nu + \omega$	Argument of Latitude
$\delta u = C_{us}\sin 2\Phi + C_{uc}\cos 2\Phi$	Argument of Latitude Correction
$\delta r = C_{rs}\sin 2\Phi + C_{rc}\cos 2\Phi$	Radius Correction
$\delta i = C_{is}\sin 2\Phi + C_{ic}\cos 2\Phi$	Inclination Correction
$u = \Phi + \delta u$	Corrected Argument of Latitude
$r = A(1\text{-}e\cos E) + \delta r$	Corrected Radius
$i = i_0 + \delta i + \dot{i} t_k$	Corrected Inclination
$x' = r\cos u$ $y' = r\sin u$	Position in orbital plane
$\Omega = \Omega_0 + (\dot{\Omega} - \omega_E)t_k - \omega_E t_{0e}$	Corrected longitude of ascending node
$x = x'\cos(\Omega) - y'\cos(i)\sin(\Omega)$ $y = x'\sin(\Omega) + y'\cos(i)\cos(\Omega)$ $z = y'\sin(i)$	GTRF coordinates of the SV antenna phase center position at time t

Рисунок 9 - Алгоритм расчета

Результат:

1.Трехмерные графики траектории движения спутника GPS(15) в инерциальной и неинерциальной системах координат:

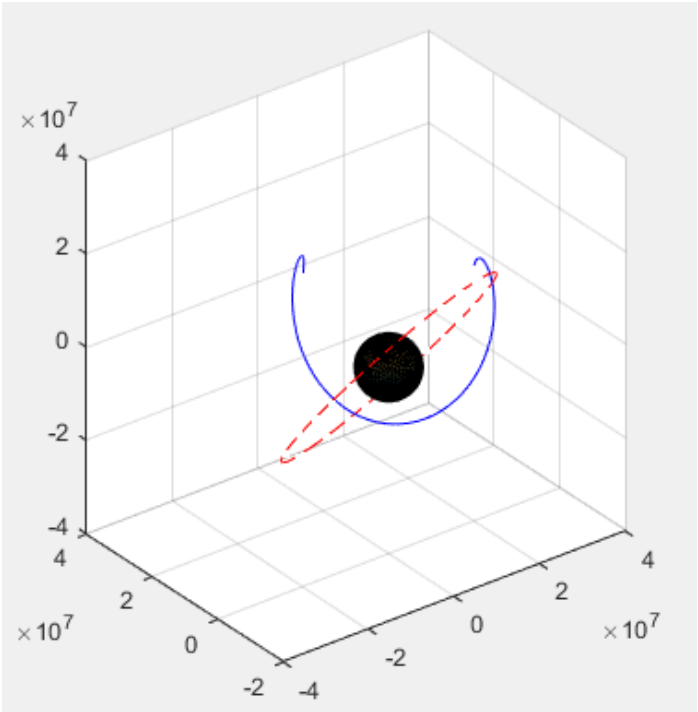

Рисунок 10 - Траектория НКА. Сплошная линия – траектория НКА в инерциальной СК, пунктирная – траектория НКА в неинерциальной СК

2.SkyView

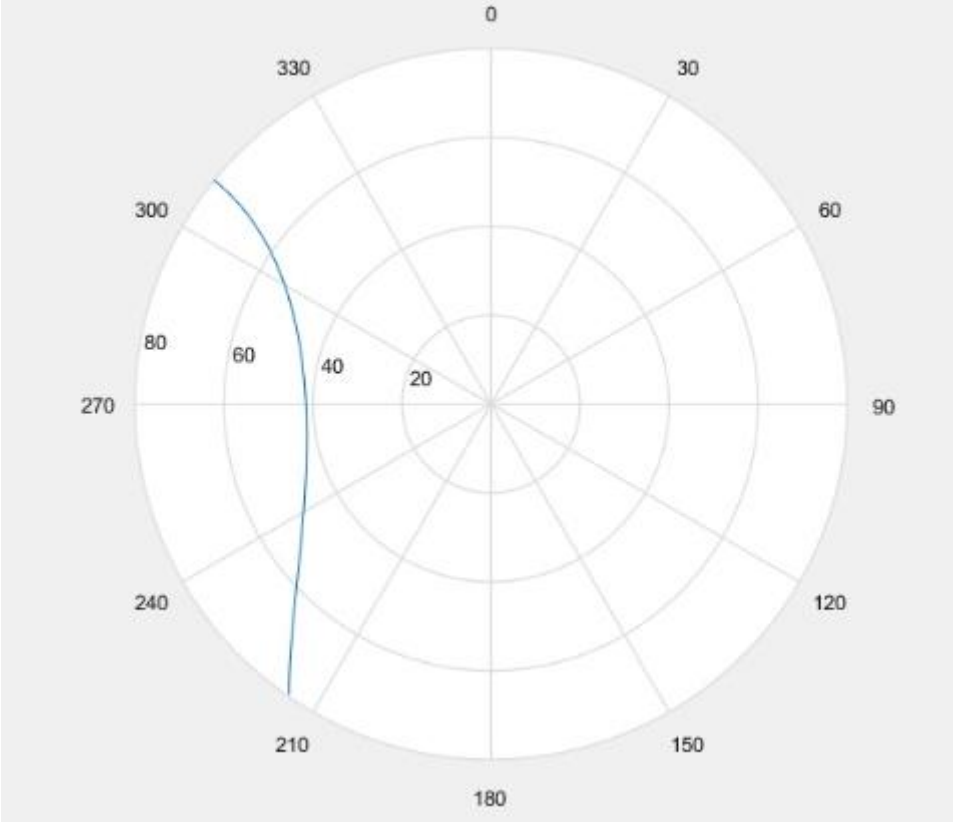

Рисунок 11 - SkyView

Вывод: SkyView рассчитанный в Matlab и построенный через специализированную программу Trimble GNSS Planning Online совпадают.

Листинг программы в Matlab:

```
clc;
clear all;
close all;
% константы
 mu = 3.986004418e+14;
 we = 7.2921151467e-5;
 c = 299792458;

% эфемериды
A = 26559933.663;
e=0.01121171;
i0=deg2rad(53.17113);
W0=deg2rad(51.38077);
w=deg2rad(43.73662);
M0=deg2rad(111.08682);
deltan=deg2rad(3.2150e-7);
Wdot=deg2rad(-4.8607e-07);
idot=deg2rad(-1.4325e-8);

time_of_week=302418;
t=time_of_week;

Cuc=-9.8161e-7;
Cus=8.2161e-6;
Crc=2.0747e+2;
Crs=-1.7781e+1;
Cic=-2.7195e-7;
Cis=-1.1362e-7;
t0e=309600+18;

% координаты приемника
latitude=55.756727964;
longitude=37.703259108;
h=189.4054;

  %расчет координат спутника
for k=1:432


 n0 = sqrt(mu/(A^3));
 tk=t-t0e;
 n=n0+deltan;
 M = M0+n*tk;
 E=0;
for l=1:100
 E=M+e*sin(E);
 end
```

```
v = atan2(sqrt(1-e^2)*sin(E),cos(E)-e);
F=v+w;
deltau=Cus*sin(2*F)+Cuc*cos(2*F);
deltar=Crs*sin(2*F)+Crc*cos(2*F);
deltai=Cis*sin(2*F)+Cic*cos(2*F);
u=F+deltau;
r=A*(1-e*cos(E))+deltar;
i=i0+deltai+idot*tk;
X=r*cos(u);
Y=r*sin(u);


ww = W0 - we*t0e+(Wdot-we)*tk;
WW = W0 - we*t0e;
% расчет координат в неинерциальной системе
x(k) = X*cos(ww)-Y*cos(i)*sin(ww);
y(k) = X*sin(ww)+Y*cos(i)*cos(ww);
z(k) = Y*sin(i);
% расчет координат в инерциальной системе
xeci(k) = X*cos(WW)-Y*cos(i)*sin(WW);
yeci(k) = X*sin(WW)+Y*cos(i)*cos(WW);
zeci(k) = Y*sin(i);
t=t+100;
end

% построение графиков
[X,Y,Z] = sphere(50);
surf(X*6400000,Y*6400000,Z*6400000)
hold on
plot3(x,y,z,'b')
plot3(xeci,yeci,zeci,'r')
daspect([1 1 1])
hold on

  % пересчет координат из глобальной неинерциальной СК к локальной (ENU)
for k = 1:432
    [x0(k) y0(k) z0(k)] = ecef2enu(x(k),y(k),z(k),latitude,longitude,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))+pi;
        elseif (x0(k)<0)&&(y0(k)<0)
        phi(k) = atan(y0(k)/x0(k))-pi;
    end
    else teta(k) = NaN;
         r(k) = NaN;
         phi(k) = NaN;
    end
        k
end
```

```
figure(2)
plot3(x0,y0,z0)
grid on
daspect([1 1 1])
hold on
plot3(0,0,0,'*')
xlabel('x')
ylabel('y')
% Построение SkyView
figure(3)
polar(phi,teta*180/pi,'-r')
```

ЭТАП III. Релазиация

Проект собирается и запускается, ошибок и исключений не выбрасывает. Лог-файлы при запуске проекта:

"Kursovoy3.exe" (Win32). Загружено "C:\Users\andre\Desktop\MVS tests\Kursovoy3\Debug\Kursovoy3.exe". Символы загружены.
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\ntdll.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\mscoree.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\kernel32.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\KernelBase.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Program Files\AVAST Software\Avast\x86\aswhook.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\vcruntime140d.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\ucrtbased.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\msvcp140d.dll".
Поток 0x2d70 завершился с кодом 0 (0x0).
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\advapi32.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\msvcrt.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\sechost.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\rpcrt4.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\sspicli.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\cryptbase.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\bcryptprimitives.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscoreei.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\shlwapi.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\combase.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\ucrtbase.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\gdi32.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\gdi32full.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\msvcp_win.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\user32.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\win32u.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\imm32.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\kernel.appcore.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\version.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\Microsoft.NET\Framework\v4.0.30319\clr.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\msvcr120_clr0400.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\psapi.dll".
"Kursovoy3.exe" (Win32). Выгружено "C:\Windows\SysWOW64\psapi.dll"
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\assembly\NativeImages_v4.0.30319_32\mscorlib\a5a47e8e5e2880adecca43eb928673f1\mscorlib.ni.dll".
"Kursovoy3.exe" (CLR v4.0.30319: DefaultDomain). Загружено "C:\Windows\Microsoft.Net\assembly\GAC_32\mscorlib\v4.0_4.0.0.0__b77a5c561934e089\mscorlib.dll". Загрузка символов пропущена. Модуль оптимизирован, включен параметр отладчика "Только мой код".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\ole32.dll".
"Kursovoy3.exe" (CLR v4.0.30319: DefaultDomain). Загружено "C:\Users\andre\Desktop\MVS tests\Kursovoy3\Debug\Kursovoy3.exe". Символы загружены.

"Kursovoy3.exe" (Win32). Загружено
"C:\Windows\Microsoft.NET\Framework\v4.0.30319\clrjit.dll".
"Kursovoy3.exe" (Win32). Загружено "C:\Windows\SysWOW64\oleaut32.dll".
"Kursovoy3.exe" (Win32). Загружено
"C:\Windows\Microsoft.NET\Framework\v4.0.30319\diasymreader.dll".
"Kursovoy3.exe" (Win32). Загружено
"C:\Windows\assembly\NativeImages_v4.0.30319_32\System\2fb412bd4cb1d503da1dc52104e9b1a4\System.ni.dll".
"Kursovoy3.exe" (CLR v4.0.30319: DefaultDomain). Загружено
"C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System\v4.0_4.0.0.0__b77a5c561934e089\System.dll". Загрузка символов пропущена. Модуль оптимизирован, включен параметр отладчика "Только мой код".
Поток 0x1384 завершился с кодом 0 (0x0).
Поток 0x197c завершился с кодом 0 (0x0).
Поток 0x9c8 завершился с кодом 0 (0x0).
Программа "[4656] Kursovoy3.exe" завершилась с кодом 0 (0x0).

На рисунке показано, что все тесты пройдены успешно

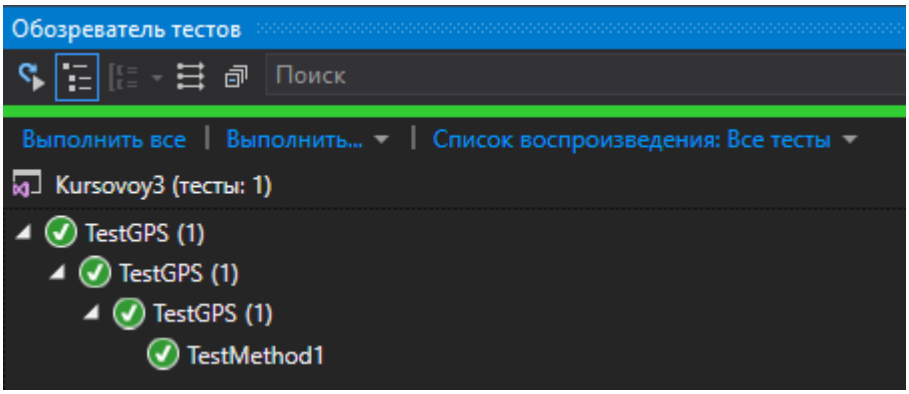

Рисунок 12 - Обозреватель тестов

Время выполнения программы в среде Matlab – 0.214852 сек.
Время выполнения программы в IDE Visual Studio – 0.032503 сек.

Проверка на утечки памяти осуществлялась в библиотеке Microsoft – библиотеки CRT. Необходимо подключить два заголовочных файла (stdlib.h, crtdbg.h) и вызвать метод _CrtSetDbgLeaks() после выполнения основной программы. Программа не выявила никаких утечек памяти (видно из консоль логов). Вероятно это связано с тем, что динамическая память не выделялась “вручную”, а вместо динамических массивов использовались вектора и ассоциативные массивы из STL, где эти операции скрыты “под капотом”.

Вывод:

В результате время выполнения программы в IDE Visual Studio примерно в 2 раза меньше, чем время выполнения в Matlab. Хотя в идеале разница должна быть более значительной (порядка 50 раз). Несоответствие вызвано недостаточной оптимизацией.

Листинг программы:

Файл testGPS.cpp:

```
#include "pch.h"
#include "CppUnitTest.h"
#include "time.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include "C:\\Users\\Lyuda\\Desktop\\MVS tests\Kursovoy3\Kursovoy3\Kursovoy3.cpp"
#include <stdlib.h>
#include <crtdbg.h>

#define _CRTDBG_MAP_ALLOC

using namespace Microsoft::VisualStudio::CppUnitTestFramework;
using std::ifstream;
using std::string;
using std::vector;

namespace TestGPS
{
	TEST_CLASS(TestGPS)
	{
	public:

		TEST_METHOD(TestMethod1)
		{
			// Массивы для записи результата
			float xCoordsOfSat[43200];
			float yCoordsOfSat[43200];
			float zCoordsOfSat[43200];
			// Структура с параметрами спутника
			params gpsParams;

			// Векторы для координат из матлаба
			vector<float> coordsXMatlab;
			vector<float> coordsYMatlab;
			vector<float> coordsZMatlab;

			//  Считываем координаты из файлов
			ifstream file1("C:\\Users\\Lyuda\\Desktop\\MVS tests\\x_coords.txt");
			ifstream file2("C:\\Users\\Lyuda\\Desktop\\MVS tests\\y_coords.txt");
			ifstream file3("C:\\Users\\Lyuda\\Desktop\\MVS tests\\z_coords.txt");

			if (file1 && file2 && file3) {

				vector<string> vecStr1, vecStr2, vecStr3;
				string str1, str2, str3;

				while (true) {
					str1.clear();
					str2.clear();
					str3.clear();

					getline(file1, str1);
					getline(file2, str2);
					getline(file3, str3);

					if (!str1.empty() && !str2.empty() && !str3.empty()) {
						vecStr1.push_back(str1);
						vecStr2.push_back(str2);
```

```
                                        vecStr3.push_back(str3);
                                } else {
                                        break;
                                }
                        }

                        for (int i = 0; i < vecStr1.size(); ++i) {
                                for (int j = 0; j < vecStr1[i].size(); ++j) {
                                        coordsXMatlab.push_back(vecStr1[i][j]);
                                }
                        }

                        for (int i = 0; i < vecStr2.size(); ++i) {
                                for (int j = 0; j < vecStr2[i].size(); ++j) {
                                        coordsYMatlab.push_back(vecStr2[i][j]);
                                }
                        }

                        for (int i = 0; i < vecStr3.size(); ++i) {
                                for (int j = 0; j < vecStr3[i].size(); ++j) {
                                        coordsZMatlab.push_back(vecStr3[i][j]);
                                }
                        }

                        file1.close();
                        file2.close();
                        file3.close();
                }

                // Вычисляем время начала программы
                unsigned int startTime = clock();

                positionOfSatellite(gpsParams, xCoordsOfSat, yCoordsOfSat,
zCoordsOfSat);

                for (int i = 0; i < 43199; i++) {
                        Assert::AreEqual(xCoordsOfSat[i], coordsXMatlab[i]);
                        Assert::AreEqual(yCoordsOfSat[i], coordsYMatlab[i]);
                        Assert::AreEqual(zCoordsOfSat[i], coordsZMatlab[i]);
                }

                // Анализ утечки памяти
                _CrtDumpMemoryLeaks();

                std::cout << (startTime / CLOCKS_PER_SEC);
        }
    };
}
```

Файл Kursovoy3.cpp:

```
#include "pch.h"
#include <iostream>
#include "time.h"

#define pi 3,1415926535

// Структура с входными параметрами
struct params {
    float A,
          M0,
          Toe,
          omegaZero,
          omega,
```

```
                omegaDot,
                omegaDotE,
                eccentricity,
                inclination,
            motionDiff,
            M,
            IDOT,
                Cus,
                Cuc,
                Crs,
                Crc,
                Cis,
            Cic;
};

// Функция, реализующая решение уравнения Кеплера
float keplersEquation(float anomaly, float eccentricity, float acc) {

       float previouslyEk = 0;
       float solution = 0;

       while (true) {
              solution = anomaly + eccentricity * sin(previouslyEk);

              if (abs(previouslyEk - solution) <= acc) {
                     break;
              }

              previouslyEk = solution;
       }

       return solution;
}

// Перевод из градусов в радианы
float degToRad(float degree) {
       return degree * (pi / 180);
}

// Вычисление положения спутника на заданный момент времени
void positionOfSatellite(struct params gpsParams, float (&coordsX)[43200], float
(&coordsY)[43200], float (&coordsZ)[43200]) {

       gpsParams.A            = 26559933.663;
       gpsParams.M0           = degToRad(111.08682);
       gpsParams.Toe          = 302418 + 18;
       gpsParams.omegaZero    = degToRad(51.38077);
       gpsParams.omega        = degToRad(43.73662);
       gpsParams.omegaDot     = degToRad(-4.8607e-07);
       gpsParams.omegaDotE     = 7.2921151467e-5;
       gpsParams.eccentricity = 0.01121171;
       gpsParams.inclination  = degToRad(53.17113);
       gpsParams.motionDiff   = degToRad(3.2150e-7);
       gpsParams.M            = 3.986004418e+14;
       gpsParams.IDOT         = degToRad(-1.4325e-8);
       gpsParams.Cus          = 8.2161e-6;
       gpsParams.Cuc          = -9.8161e-7;
       gpsParams.Crs          = -1.7781e+1;
       gpsParams.Crc          = 2.0747e+2;
       gpsParams.Cis          = -1.1362e-7;
       gpsParams.Cic          = -2.7195e-7;

       for (int i = 0; i < 43199; i++) {
```

```
            int momentOfTime = 302418 + i;

            int Tk = momentOfTime - gpsParams.Toe;
            // Определяем точность параметров
            float accuracy = pow(10, -7);

            if (Tk > 302400) {
                  Tk = Tk - 604800;
            }
            else if (Tk < -302400) {
                  Tk = Tk + 604800;
            }

            float n0 = pow(gpsParams.M / (pow(gpsParams.A, 3)), 0.5);
            float n = n0 + gpsParams.motionDiff;
            float anomaly = gpsParams.M0 + n * Tk;

            // Решаем уравнение Кеплера
            float Ek = keplersEquation(anomaly, gpsParams.eccentricity, accuracy);
            float Vk = atan2((pow((1 - gpsParams.eccentricity *
gpsParams.eccentricity), 0.5) * sin(Ek) / (1 - gpsParams.eccentricity * cos(Ek))),
((cos(Ek) - gpsParams.eccentricity) / (1 - gpsParams.eccentricity * cos(Ek))));

            float Fk = Vk + gpsParams.omega;

            float deltaUk = gpsParams.Cus * sin(2 * Fk) + gpsParams.Cuc * cos(2 * Fk);
            float deltaRk = gpsParams.Crs * sin(2 * Fk) + gpsParams.Crc * cos(2 * Fk);
            float deltaIk = gpsParams.Cis * sin(2 * Fk) + gpsParams.Cic * cos(2 * Fk);

            float Uk = Fk + deltaUk;
            float Rk = gpsParams.A * (1 - gpsParams.eccentricity * cos(Ek)) + deltaRk;
            float Ik = gpsParams.inclination + deltaIk + gpsParams.IDOT * Tk;
            float Wk = gpsParams.omegaZero + (gpsParams.omegaDot - gpsParams.omegaDotE)
* Tk - gpsParams.omegaDotE * gpsParams.Toe;

            float x = Rk * cos(Uk);
            float y = Rk * sin(Uk);
            // Координаты в системе ECEF
            float ecefX = x * cos(Wk) - y * cos(Ik) * sin(Wk);
            float ecefY = x * sin(Wk) + y * cos(Uk) * cos(Wk);
            float ecefZ = y * sin(Ik);

            // Координаты в системе ECI
            float theta = gpsParams.omegaDot * Tk;

            float eciX = ecefX * cos(theta) - ecefY * sin(theta);
            float eciY = ecefX * sin(theta) + ecefY * cos(theta);
            float eciZ = ecefZ;

            coordsX[i] = eciX;
            coordsY[i] = eciY;
            coordsZ[i] = eciZ;
      }

}

int main()
{


    return 0;
}
```