Листинг Программм
Программа этапа моделирование

```
clear; close all; clc;
 format long g

%const
ae=6378136;         %
we=7.2921151467e-5; %earth's rotation rate
pi=3.14159265359;

%data RTKNAVI

x0=10192674.32;
y0=-12367565.43;
z0=19866879.39;
vx=2599.78676;
vy=-789.66141;
vz=-1827.75784;
ax=0.0000019;
ay=0.000009;
az=-0.0000028;
Tau=-38310; %ns
Gamma=0.0018; %ns
%%
%time format
 %2020.02.10 13.45.18
 N4=7;
 Nt=41;
 hour=13;
 min=45;
 sec=0;
 h_st=12;   %start hour
 h_fin=24;  %end hour

TIME=time(N4,Nt,hour,min,sec, h_st,h_fin);
S=TIME(1);
    time_start=TIME(2);
    time_final=TIME(3);
T=TIME(4);
te=TIME(5);
GMST=TIME(6);


%%

%Position
```

```
xa=x0*cos(S)-y0*sin(S);
ya=xa*sin(S)+y0*cos(S);
za=z0;
%Velocity
vxa=vx*cos(S)-vy*sin(S)-we*ya;
vya=vx*sin(S)+vy*cos(S)+we*xa;
vza=vz;

Jsm_x=ax*cos(S)-ay*sin(S);
Jsm_y=ax*sin(S)+ay*cos(S);
Jsm_z=az;

%% load 3 c++
X_trynotcry = load('INERT_x.txt');
Y_trynotcry = load('INERT_y.txt');
Z_trynotcry = load('INERT_z.txt');
PZ_X_trynotcry = load('PZ_x.txt');
PZ_Y_trynotcry = load('PZ_y.txt');
PZ_Z_trynotcry = load('PZ_z.txt');
%%


coordinat=math_2(xa,ya,za,vxa,vya,vza,Jsm_x,Jsm_y,Jsm_z,time_start,time_final, te,T);

%bilding earth
[EAR_x,EAR_y,EAR_z] = sphere(20);
EAR_x=ae.*EAR_x;
EAR_y=ae.*EAR_y;
EAR_z=ae.*EAR_z;
%
figure (1)
surf(EAR_x,EAR_y,EAR_z)
hold on
grid on
plot3(X_trynotcry,Y_trynotcry,Z_trynotcry)
plot3(coordinat(:,1),coordinat(:,2),coordinat(:,3))
title('
trajectory in an inertial coordinate system')
xlabel('x,m')
ylabel('y,m')
zlabel('z,m')

coordinat(end,7)
% PZ 90.11
ti=coordinat(:,7);
```

```
S_pz=GMST+we*(ti-10800);
x_pz=coordinat(:,1).*cos(S_pz)+coordinat(:,2).*sin(S_pz);
y_pz=-coordinat(:,1).*sin(S_pz)+coordinat(:,2).*cos(S_pz);
z_pz=-coordinat(:,3);

vx_pz=coordinat(:,4).*cos(S_pz)+coordinat(:,5).*sin(S_pz)+we*coordinat(:,2);
vy_pz=-coordinat(:,4).*sin(S_pz)+coordinat(:,5).*cos(S_pz)+we*coordinat(:,1);
vz_pz=-coordinat(:,6);
figure (2)
surf(EAR_x,EAR_y,EAR_z)
hold on
grid on
plot3(PZ_X_trynotcry,PZ_Y_trynotcry,PZ_Z_trynotcry)
plot3(x_pz,y_pz,z_pz)
title('
trajectory in an PZ90 coordinate system')
xlabel('x,m')
ylabel('y,m')
zlabel('z,m')


%%
% SkyView

PZ90=[x_pz;y_pz;z_pz];
a=[1 -0.9696*10^-6 0;-0.9696*10^-6 1 0; 0 0 1];
b=[-1.10;-0.30;-0.90];

for i=1: length(x_pz)
    WGS84_x(i)=(1-0.12*10^-6)*(x_pz(i)*1+y_pz(i)*-0.9696*10^-6+z_pz(i)*0)+b(1);
     WGS84_y(i)=(1-0.12*10^-6)*(x_pz(i)*-0.9696*10^-6+y_pz(i)*1+z_pz(i)*0)+b(2);
      WGS84_z(i)=(1-0.12*10^-6)*(x_pz(i)*0+y_pz(i)*0+z_pz(i)*1)+b(3);

end
figure (3)
surf(EAR_x,EAR_y,EAR_z)
hold on
grid on
plot3(WGS84_x,WGS84_y,WGS84_z)
title('
trajectory in an WGS84 coordinate system ')
xlabel('x,m')
ylabel('y,m')
zlabel('z,m')

N_gr = 55;
```

```
N_min = 45;
N_sec = 23.6675;
E_gr = 37;
E_min = 42;
E_sec = 12.3895;
H = 150;%
N = N_gr*pi/180 + N_min/3437.747 + N_sec/206264.8; % широта в радионах
E = E_gr*pi/180 + E_min/3437.747 + E_sec/206264.8; % долгота в радионах
llh = [N E H];
%PRM_coor = llh2xyz(llh)';

coor=[N;E;H];
for i=1:length(WGS84_x)
   [x(i) y(i) z(i)] =
ecef2enu(WGS84_x(i),WGS84_y(i),WGS84_z(i),N,E,H,wgs84Ellipsoid,'radians');
   if z(i) > 0
    teta(i) = atan2(sqrt(x(i)^2 + y(i)^2),z(i));
    r(i) = sqrt(x(i)^2 + y(i)^2 + z(i)^2);
    phi(i) = atan2(y(i),x(i));
    else teta(i) = NaN;
    r(i) = NaN;
    phi(i) = NaN;
   end
end

figure(4);
polar(phi,(teta*180-pi)/pi,'r')
title('SkyVeiw ')


function [res] = math_2(xa,ya,za,vxa,vya,vza,Jsm_x,Jsm_y,Jsm_z,time_start,time_final,
te,T)
%надо рассмотреть два случая: когда наблюдения есть до начала старта, и
%когда они есть после старта. Во втором случае задача разбивается на 2
%части: поиск координат от старта до наблюдений и от наблюдений до финала.
dt=1;
if(te<=time_start&&te<time_final)
  t=time_start:dt:(time_final-1);
   result=nan(length(t),6);
   result(1,:)=[xa,ya,za,vxa,vya,vza];
   result=RungKUTT( t, result, T );


elseif(te>time_start&&te<time_final)
   t_bef=te:-dt:time_start; %из-за того что в этом времени матрица записывается как бы
наоборот, вращаем столбцы!!!!
```

```
   t_after=te:dt:(time_final-1);
   %before

      result_before=nan(length(t_bef),6);
      result_before(1,:)=[xa,ya,za,vxa,vya,vza];
      result_before=RungKUTT( t_bef, result_before, -dt );
      t_before=rot90(t_bef,2);
      %result_bef=RungKUTT( t_bef,result_bef, T );
      %вот тут вращаем, и время тоже! потому что иначе будет [te t-1 t-2
      %... t_start], и это нормально не скоеитть! а нам надо [t_start
      %t_start+1 ... te] для этого поворачиваем каждый столбец на 180
      %градусов!


            result_before = [rot90(result_before(:,1),2) rot90(result_before(:,2),2)
rot90(result_before(:,3),2) rot90(result_before(:,4),2) rot90(result_before(:,5),2)
rot90(result_before(:,6),2)];
   %after

      result_after=nan(length(t_after),6);
      result_after(1,:)=[xa,ya,za,vxa,vya,vza];
      result_after=RungKUTT( t_after, result_after, dt );
         %result_after(1:10,1)
      result_before(1:10,1)
   %соединяем
     result=[result_before;result_after];
      t=[t_before t_after];


end
%%поправка на небесные тела
tau=t-te;
tau=rot90(tau);


dx=Jsm_x*0.5*tau.^2;
dy=Jsm_y*0.5*tau.^2;
dz=Jsm_z*0.5*tau.^2;

dvx=Jsm_x*tau;
dvy=Jsm_y*tau;
dvz=Jsm_z*tau;


result(:,1)=result(:,1)+dx;
result(:,2)=result(:,2)+dy;
```

```
result(:,3)=-1*(result(:,3)+dz);     % чтоб земля на место встала

result(:,4)=result(:,4)+dvx;
result(:,5)=result(:,5)+dvy;
result(:,6)=result(:,6)+dvz;
res = [result rot90(t,3)];
end

function [ time ] = time( N4,Nt,hour,min,sec, h_st,h_fin )
pi=3.14159265359;
we=7.2921151467e-5; %earth's rotation rate
te=(hour+3)*60*60+min*60+sec;
time_start=(h_st+3)*60*60; %(+3 UTC)
time_final=(h_fin+3)*60*60;


JD0=1461*(N4-1)+Nt+2450082.5;%текущая юлианская дата на 0 часов шкалы МДВ
T=(JD0+(te -10800)/86400-2451545.0)/36525;
%расчеты времени всякие GMST и прочее (Приложение Л ИКД)
JDN=JD0+0.5;

Tdel=(JD0-2451545.0)/36525;

ERA=2*pi*(0.7790572732640 + 1.00273781191135448*(JD0 - 2451545.0));%угол
поворота Земли, рад
GMST=ERA+0.0000000703270726+0.0223603658710194*Tdel+0.0000067465784654*Tde
l^2-0.0000000000021332*Tdel^3-0.0000000001452308*Tdel^4-0.0000000000001784*Tdel^
5;   %истинное звездное время по Гринвичу (рад) (GST ИКД)
S=GMST+we*(te-10800);  %10800 из ИКД
[time]=[S,time_start,time_final,T,te,GMST,ERA,JD0] ;
end

function [RungKUTT ] = RungKUTT( t, result, dt)


for i=1:length(t)-1


    K0=F(result(i,:));


    K1=F((result(i,:)+0.5*dt.*K0));

    K2=F(result(i,:)+0.5*dt.*K1);

    K3=F(result(i,:)+dt.*K2);
```

```
   result(i+1,:)=result(i,:)+(dt/6)*(K0+2*K1+2*K2+K3);
%     (dt/6)*(K0+2*K1+2*K2+K3)
%     K0
%     2*K1
%     2*K2
%     K3

end
[RungKUTT]=result;
end

function [ F ] = F( inp)

J02=1082625.75e-9;  %зональный гармонический коэффициент второй степени
GM=398600441.8e6;    %– геоцентрическая константа гравитационного поля Земли
ae=6378136;         %большая (экваториальная) полуось общеземного эллипсоида

xa=inp(1);
ya=inp(2);
za=inp(3);
vxa=inp(4);
vya=inp(5);
vza=inp(6);

r=sqrt(xa^2+ya^2+za^2);
GMrat=GM/(r^2);
xarat=xa/r;
yarat=ya/r;
zarat=za/r;
ro=ae/r;


%вот это надо интегрировать Рунге Кутты 4 порядка
dxadt=vxa;
dyadt=vya;
dzadt=vza;
dvxadt=-GMrat*xarat-(3/2)*J02*GMrat*xarat*(ro^2)*(1-5*zarat^2);  %+Jxas+Jxam;
dvyadt=-GMrat*yarat-(3/2)*J02*GMrat*yarat*(ro^2)*(1-5*zarat^2);%+Jyas+Jyam;
dvzadt=-GMrat*zarat-(3/2)*J02*GMrat*zarat*(ro^2)*(3-5*zarat^2);%+Jzas+Jzam;
[F]=[dxadt,dyadt,dzadt,dvxadt,dvyadt,dvzadt];
end
```

Программа этапа Реализация

```
#include <iostream>
#include <vector>
```

```
#include <array>
#include <math.h>
#include <fstream>

const double pi=3.14159265359;
const double we=7.2921151467e-5; //earth's rotation rate

using namespace std;
//обявление всякого
struct coord
{
   double xa, ya, za,vxa,vya,vza;
};




void RungKUTT(coord res[], double t, double dt);


void math_2(coord result[],int delt, int time_start,int time_final,int te,
          double xa, double ya,double za,double vxa, double vya, double vza, double Jsm_x,
double Jsm_y,double Jsm_z);


int main()
{
double S,T,GMST,ERA,JD0, Tdel;
double N4,Nt,hour,minut,sec, h_st,h_fin ;
int time_start,time_final, te;
//Тут про время
coord *result;

N4=7;
 Nt=41;
 hour=13;
 minut=45;
 sec=0;
 h_st=12;  // %начало наблюдения, часов
 h_fin=24;  //%конец наблюдения, часов


te=(hour+3)*60*60+60*minut+sec;//+3UTC
time_start=(h_st+3)*60*60; //(+3 UTC)
time_final=(h_fin+3)*60*60;
JD0=1461*(N4-1)+Nt+2450082.5;//текущая юлианская дата на 0 часов шкалы МДВ
```

```
T=(JD0+(te -10800)/86400-2451545.0)/36525;
//расчеты времени всякие GMST и прочее (Приложение Л ИКД)
double TE=te;
Tdel=(JD0-2451545.0)/36525;
ERA=2*pi*(0.7790572732640 + 1.00273781191135448*(JD0 - 2451545.0));//%угол
поворота Земли, рад
GMST=ERA+0.0000000703270726+0.0223603658710194*Tdel+0.0000067465784654*Tde
l*Tdel-0.0000000000021332*Tdel*Tdel*Tdel-0.0000000001452308*Tdel*Tdel*Tdel*Tdel-0.0
000000000001784*Tdel*Tdel*Tdel*Tdel*Tdel;  // %истинное звездное время по Гринвичу
(рад) (GST ИКД)
S=GMST+we*(TE-10800);  //%10800 из ИКД


// Тут вводим данные эфемерид, пересчитываем все

 double x0, y0,z0,vx,vy,vz,ax,ay,az,Tau,Gamma;
x0=10192674.32;
y0=-12367565.43;
z0=19866879.39;
vx=2599.78676;
vy=-789.66141;
vz=-1827.75784;
ax=0.0000019;
ay=0.000009;
az=-0.0000028;
Tau=-38310; //%ns
Gamma=0.0018; //%ns

// Тут пересчитываем координаты в др.формат

double xa, ya, za, vxa,vya,vza,Jsm_x,Jsm_y,Jsm_z;
xa=x0*cos(S)-y0*sin(S);
ya=xa*sin(S)+y0*cos(S);
za=z0;
//%Velocity
vxa=vx*cos(S)-vy*sin(S)-we*ya;
vya=vx*sin(S)+vy*cos(S)+we*xa;
vza=vz;

Jsm_x=ax*cos(S)-ay*sin(S);
Jsm_y=ax*sin(S)+ay*cos(S);
Jsm_z=az;

int delt=43200;

result = new coord [delt];
```

```
math_2(result,delt,  time_start, time_final, te,
          xa,  ya, za, vxa, vya,  vza, Jsm_x, Jsm_y, Jsm_z);

ofstream INERT_x("INERT_x.txt");
for (int i = 0; i < delt; ++i)

{
INERT_x << result[i].xa <<'\n';
}


INERT_x.close();

ofstream INERT_y("INERT_y.txt");
for (int i = 0; i < delt; ++i)

{
INERT_y << result[i].ya <<'\n';
}


INERT_y.close();
ofstream INERT_z("INERT_z.txt");
for (int i = 0; i < delt; ++i)
{
INERT_z << result[i].za <<'\n';
}
INERT_z.close();

//П390
coord *result_pz;
result_pz = new coord [delt];
double ti=time_start;
double S_pz;
for(int i=0; i<delt;i++)
{
   S_pz=GMST+we*(ti-10800);


    result_pz[i].xa=result[i].xa*cos(S_pz)+result[i].ya*sin(S_pz);
    result_pz[i].ya=-result[i].xa*sin(S_pz)+result[i].ya*cos(S_pz);
    result_pz[i].za=-result[i].za;
    result_pz[i].vxa= result[i].vxa*cos(S_pz)+result[i].vya*sin(S_pz)+we*result[i].ya;
    result_pz[i].vya= -result[i].vxa*sin(S_pz)+result[i].vya*cos(S_pz)+we*result[i].xa;
    result_pz[i].vza=-result[i].vza;
```

```
    ti=ti+1;
}


ofstream PZ_x("PZ_x.txt");
for (int i = 0; i < delt; ++i)

{
PZ_x << result_pz[i].xa <<'\n';
}


PZ_x.close();

ofstream PZ_y("PZ_y.txt");
for (int i = 0; i < delt; ++i)

{
PZ_y << result_pz[i].ya <<'\n';
}


PZ_y.close();
ofstream PZ_z("PZ_z.txt");
for (int i = 0; i < delt; ++i)
{
PZ_z << result_pz[i].za <<'\n';
}
PZ_z.close();
    delete []result;
}


#include <math.h>
#include <iostream>
#include <stdio.h>


using namespace std;
//константы
const double J02=1082625.75e-9;
const double GM=398600441.8e6;
const double ae=6378136;

struct coord
```

```
{
   double xa, ya, za,vxa,vya,vza;
};

coord  F(struct coord cor, double xa ,double ya,double za,double vxa,double vya,double vza)
{


double r, GMrat, xarat, yarat, zarat, ro, Vxa, Vya, Vza ;
r=sqrt(pow(xa,2)+pow(ya,2)+pow(za,2));
GMrat=GM/(pow(r,2));
xarat=xa/r;
yarat=ya/r;
zarat=za/r;
ro=ae/r;

Vxa=-GMrat*xarat-(3/2)*J02*GMrat*xarat*(pow(ro,2))*(1-5*pow(zarat,2));//  %+Jxas+Jxam;
Vya=-GMrat*yarat-(3/2)*J02*GMrat*yarat*(pow(ro,2))*(1-5*pow(zarat,2));//%+Jyas+Jyam;
Vza=-GMrat*zarat-(3/2)*J02*GMrat*zarat*(pow(ro,2))*(3-5*pow(zarat,2));//%+Jzas+Jzam;
cor.xa=vxa;
cor.ya=vya;
cor.za=vza;
cor.vxa=Vxa;
cor.vya=Vya;
cor.vza=Vza;
return cor;
}


#include <math.h>
#include <iostream>
#include <iostream>
#include <vector>
#include <array>
#include <math.h>

struct coord
{
   double xa, ya, za,vxa,vya,vza;
};

coord  F(struct coord cor, double xa ,double ya,double za,double vxa,double vya,double
vza);

//t- длина всего безобразия. dt- шаг. tst- начальное время для интегрирования
void RungKUTT(coord res[], double t, double dt)
```

```
{

coord K0;
coord K1;
coord K2;
coord K3;
coord ink1, ink2, ink3, ink;
coord cyk;

ink=res[0];
for (int i=1; i<t; i++)
{
   cyk=ink;

   K0=F(K0,cyk.xa,cyk.ya,cyk.za,cyk.vxa,cyk.vya,cyk.vza );


ink1={cyk.xa+0.5*dt*K0.xa,cyk.ya+0.5*dt*K0.ya,cyk.za+0.5*dt*K0.za,cyk.vxa+0.5*dt*K0.vxa,
cyk.vya+0.5*dt*K0.vya,cyk.vza+0.5*dt*K0.vza};       //(result(i,:)+0.5*dt.*K0));
   K1=F(K1, ink1.xa,ink1.ya,ink1.za,ink1.vxa,ink1.vya,ink1.vza );


ink2={cyk.xa+0.5*dt*K1.xa,cyk.ya+0.5*dt*K1.ya,cyk.za+0.5*dt*K1.za,cyk.vxa+0.5*dt*K1.vxa,
cyk.vya+0.5*dt*K1.vya,cyk.vza+0.5*dt*K1.vza};
   K2=F(K2, ink2.xa,ink2.ya,ink2.za,ink2.vxa,ink2.vya,ink2.vza);


ink3={(cyk.xa+dt*K2.xa),(cyk.ya+dt*K2.ya),(cyk.za+dt*K2.za),(cyk.vxa+dt*K2.vxa),(cyk.vya+
dt*K2.vya),(cyk.vza+dt*K2.vza)};
   K3=F(K3, ink3.xa,ink3.ya,ink3.za,ink3.vxa,ink3.vya,ink3.vza);


ink={cyk.xa+(dt/6)*(K0.xa+2*K1.xa+2*K2.xa+K3.xa),cyk.ya+(dt/6)*(K0.ya+2*K1.ya+2*K2.ya+
K3.ya),cyk.za+(dt/6)*(K0.za+2*K1.za+2*K2.za+K3.za),cyk.vxa+(dt/6)*(K0.vxa+2*K1.vxa+2*K
2.vxa+K3.vxa),cyk.vya+(dt/6)*(K0.vya+2*K1.vya+2*K2.vya+K3.vya),cyk.vza+(dt/6)*(K0.vza+
2*K1.vza+2*K2.vza+K3.vza)};

res[i]=ink;


}

}
```