МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

(МАИ)

Кафедра «Системный анализ и управление»

ОТЧЕТ

По лабораторной работе

По курсу «Динамическое проектирование систем управления ЛА»

Задание 2. Исследование работы системы ориентации и стабилизации КА на этапе переориентации с использованием двигателей-маховиков

Выполнила:

Богатова А.А.

Группа М6О-407Б-18

Работу принял:

Розин П.Е.

Москва 2021

Содержание

[Исходные данные. 3](#_Toc89265866)

[Цели работы. 3](#_Toc89265867)

[Постановка задачи. 3](#_Toc89265868)

[Методика решения. 4](#_Toc89265869)

[Алгоритм решения. 8](#_Toc89265870)

[Полученные результаты. 10](#_Toc89265871)

[Выводы. 16](#_Toc89265872)

[Листинг. 17](#_Toc89265873)

# **Исходные данные.**

Задание 2. Вариант 1:

рад/с

­

Орбита №2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант | эпоха, ДМВ |  |  |  |
| 1 | 12.03.2019  04:00:00.000 | 6237.377369 | 2319.470899 | -1398.276906 |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| -0.528889 | 4.936036 | 5.828682 |

# **Цели работы.**

Научится моделировать управляемое вращательное движение КА, оснащённого ДМ, и закрепить понимания работы системы управления угловым движением КА с ДМ.

# **Постановка задачи.**

Рассмотрим задачу переориентации с последующей стабилизацией КА в инерциальном пространстве при использовании в качестве управляющих органов ДМ и в качестве измерительного прибора – датчика угловых скоростей.

Будем полагать, что датчик угловых скоростей перед началом работы прошел калибровку систематической составляющей дрейфа нулевого сигнала, масштабных коэффициентов и не ортогональности осей. Т.е. основной составляющей ошибок ориентации и стабилизации является шумовая составляющая измерений. Будем полагать, что шум измерений является некоррелированным белым шумом, который моделируется как нормальный закон распределения случайной величины с заданной дисперсией и нулевым математическим ожиданием.

# **Методика решения.**

Векторное дифференциальное уравнение динамики Эйлера для КА как твердого тела в форме Коши выглядит в виде (1).

(1)

Управление вращательным движением КА будет осуществляться за счет изменения вектора кинетического момента системы ДМ , которое будет возникать за счет подачи управляющего напряжения каждого из приборов. Итоговая динамическая система вращающегося КА, оснащенного ДМ, для случая идеального ДМ (отработка сигнала управления без запаздывания, перерегулирования и т.п.) будет выглядеть как соотношения (2).

(2)

где − вектор угловой скорости вращения КА вокруг центра масс; 𝐽 – тензор инерции КА; − вектор суммарного момента сил, действующих на КА; –вектор угловых скоростей вращения ДМ; диагональная матрица, содержащая значения величин моментов инерции роторов ДМ, установленных по соответствующей оси ССК КА. Соответственно, для управления вращательным движением КА требуется формировать на каждый из ДМ вектор управляющих сигналов , прямо пропорциональный .

Уравнения (1) и (2) описывают динамику вращательного движения КА. То есть их решение позволяет определить вектор угловой̆ скорости относительно инерциального пространства. В тоже время, угловое положение ССК КА относительно известного вектора угловой̆ скорости, описывается нелинейными соотношениями. Наиболее широкое распространение получили кинематические уравнения в кватернионной форме (уравнения Пуассона) или представленные в виде матрицы направляющих косинусов. В данном варианте рассматриваются кинематические соотношения в виде уравнений Пуассона (3).

(3)

где Λ – кватернион ориентации КА в пространстве, ⊗ - оператор кватернионного перемножения (см. приложение Г).

Геометрический̆ смысл кватерниона можно определить так:

Где 𝜑 – угол поворота вокруг орта 𝑒̅ =. То есть кватернион характеризует орт оси пространственного поворота и угол, на который нужно вокруг этой оси повернуть КА, чтобы оси его связанной системы координат заняли определенное положение относительно осей инерциальной системы координат.

При интегрировании уравнения (3) после каждого шага интегрирования осуществлять нормировку получаемого кватерниона, то есть обеспечивать выполнение равенства:

Комбинация уравнений кинематики и динамики полностью описывают вращательное движение КА как твердого тела.

Рассмотрим модель орбитального движения спутника, которая получается в результате интегрирования дифференциальных уравнений движения с учетом следующих внешних возмущений:

* несферичности гравитационного поля Земли (учитывая влияние только второй зональной гармоники),
* гравитационного воздействия Солнца.

В геоцентрической инерциальной̆ системе координат J2000 уравнении

возмущенного орбитального движения можно записать в виде (4).

(4)

радиус-вектор КА в геоцентрической инерциальной системе координат J2000; = 398600.4415 − гравитационный параметр Земли; − ускорение за счёт влияния несферичности гравитационного поля; − ускорение, приобретаемое за счёт гравитационного влияния Солнца.

Ускорение за счёт влияния несферичности гравитационного поля можно представить в виде (6).

(5)

где - матрица перехода от J2000 в гринвичскую СК, которая определяется из соотношения (7); - возмущающее ускорение в гринвичской̆ вращающейся системе координат определяется из соотношения (8).

(6)

(7)

Гравитационное ускорение, обусловленное влиянием Солнца, будет определяться из соотношения (9).

(8)

где: радиус-вектор Солнца, радиус вектор КА, расстояние между КА и Солнцем [км]. - гравитационный параметр Солнца= 132712517951.

Выражение для гравитационного момента может быть представлено в следующем виде:

(9)

Где,

𝜂 – единичный̆ вектор от центра масс Земли до центра масс КА, в проекции на оси ССК КА

𝐽 – тензор инерции КА.

𝜇 – гравитационный̆ параметр Земли, 𝜇 = 3,986 × 1014 [м3/с2].

𝑅 – расстояние от центра масс Земли до центра масс КА (м).

Для вычисления проекции вектора 𝜂 на оси ССК КА достаточно выполнить следующие преобразования:

, где

Λ𝑡 – текущий кватернион ориентации КА в момент времени t.

– нормированный̆ вектор из центра Земли в центре масс КА в инерциальной системе координат в момент времени t.

# **Алгоритм решения.**

Для решения задачи требуется реализовать программное обеспечение позволяющее моделировать следующую последовательность действий:

1. Осуществляется интегрирование уравнений (2) и (3) методом Ругне- Кутта 4-го порядка с заданным шагом h с учетом возмущений. Решаются уравнения (4).
2. Обозначим −h̅̇ из уравнения (8) как . Тогда линейный, асимптотически устойчивый закон управления ДМ можно представить в виде:

Где Λ1:3 = [Λ1 Λ2 Λ3]𝑇 – вектор из элементов кватерниона ориентации.

Следует также ввести две нелинейности типа «насыщение» для и для величины следующего вида.

,

Первая нелинейность характеризует существование предельного ускорения ДМ. Вторая – существование предельной скорости вращения ДМ.

Для анализа результатов, каждые 100 мс счета программы требуется выводить в файл следующие величины: Λ, 𝜔̅, Ω̅, 𝑟̅𝑗2000. Моделирование необходимо провести с учетом влияния гравитационного возмещающего момента (уравнение (9)).

**Полученные результаты.**

Результаты при и *Рисунок 1.Рисунок 2.Рисунок 3*

Результаты при и *Рисунок 4*

*Рисунок 5*

*Рисунок 6*

Результаты при и *Рисунок 7*

*Рисунок 8*

*Рисунок 9*

Результаты при и *Рисунок 10*

*Рисунок 11*

*Рисунок 12*

Результаты при и *Рисунок 13*

*Рисунок 14*

*Рисунок 15*

Результаты при и *Рисунок 16*

*Рисунок 17*

*Рисунок 18*

*Рисунок 19*

# 

# **Выводы.**

В результате выполнения лабораторной работы разработан алгоритм решения задачи приведения ориентации летательного аппарата, от текущего направления к заданному с использованием в качестве управляющих органов реактивных двигателей стабилизации. Был изучен процесс моделирования управляемого вращательного движения КА, оснащённого ДМ, закреплено понимания работы системы управления угловым движением КА с ДМ. Во время решения задачи было учтено влияние Солнца и не сферичности гравитационного поля Земли.

Из полученный результатов можно сделать следующие выводы:

* Увеличение и уменьшение приводит к уменьшению времени переходного процесса, но переходный процесс ухудшается.
* Уменьшение и увеличение приводит к увеличению времени переходного процесса.
* При сильном изменении коэффициентов усиления система теряет устойчивость и сопровождается колебаниями с недопустимой амплитудой.

# **Листинг.**

#include <iostream>

#include <math.h>

#include <fstream>

**using** **namespace** std;

**double** h = 0.01; //шаг интегрирования

**const** **int** n = 16; //количество уравнений в системе

**double** x[n], dx[n];

**double** tk = 500, t = 0; //начальное и конечное время

**double** mu = 398600.4415; //гравитационный параметр Земли

**double** agrav[3] = { 0, 0 ,0 }, asun[3] = { 0,0,0 };//гравитационное возущение и солнечное возмущение

**const** **double** pi = 3.141592653589793;//число pi

**double** J[9] = { 1, 0, 0, // тензор инерции

0, 2, 0,

0, 0, 1 };

**double** Jinv[9] = { 1, 0, 0, //

0, 0.5, 0,

0, 0, 1 };

**double** Msum[3];//сумарный момент

**double** Mu[3] = { 0 };//управляющий момент

**double** Idm = 0.00011126;

**int** i = 0;

**void** matr3Xvect3(**double** a[9], **double** v[3], **double** c[3]) //перемножение матрицы 3 на 3 на вектор 3 на 1

{

c[0] = a[0] \* v[0] + a[1] \* v[1] + a[2] \* v[2];

c[1] = a[3] \* v[0] + a[4] \* v[1] + a[5] \* v[2];

c[2] = a[6] \* v[0] + a[7] \* v[1] + a[8] \* v[2];

}

**void** vectrn(**double** a[3], **double** b[3], **double** c[3]) //Векторное произведение

{

c[0] = a[1] \* b[2] - a[2] \* b[1];

c[1] = a[2] \* b[0] - a[0] \* b[2];

c[2] = a[0] \* b[1] - a[1] \* b[0];

}

**void** quatmul(**double** b[4], **double** a[4], **double** c[4]) //Перемножение кватраниона v на a

{

c[0] = a[0] \* b[0] - a[1] \* b[1] - a[2] \* b[2] - a[3] \* b[3];

c[1] = a[0] \* b[1] + a[1] \* b[0] + a[3] \* b[2] - a[2] \* b[3];

c[2] = a[0] \* b[2] + a[2] \* b[0] + a[1] \* b[3] - a[3] \* b[1];

c[3] = a[0] \* b[3] + a[3] \* b[0] + a[2] \* b[1] - a[1] \* b[2];

}

**int** sign(**double** a) // Математическая функция sign

{

**if** (a > 0)

{

**return** 1;

}

**if** (a < 0)

{

**return** -1;

}

**return** 0;

};

**double** JData(**double** t) {//вычисление юлианской даты

**double** year = 2019, month = 3, day = 12, hours = 4, min = 0, sec = 0, msec = 0;

**double** a, y, m, JD;

a = (14 - month) / 12;

y = year + 4800 - a;

m = month + 12 \* a - 3;

JD = day + (153 \* m + 2) / 5 + 365 \* y + y / 4 - 32083 + (hours - 12) / 24 + (min + floor((sec + t) / 60)) / 1440 + fmod(sec + t, 60) / 86400;

**return** JD;

}

**void** q2m(**double**\* q, **double**\* m)// Кватернион в матрицу

{

**double** q01 = q[0] \* q[1], q02 = q[0] \* q[2], q03 = q[0] \* q[3], q12 = q[1] \* q[2],

q13 = q[1] \* q[3], q23 = q[2] \* q[3], q1s = q[1] \* q[1], q2s = q[2] \* q[2],

q3s = q[3] \* q[3], a = q[0] \* q[0] + q1s + q2s + q3s;

m[0] = 1. - 2. \* (q2s + q3s);

m[3] = 2. \* (q12 + q03);

m[6] = 2. \* (q13 - q02);

m[1] = 2. \* (q12 - q03);

m[4] = 1. - 2. \* (q1s + q3s);

m[7] = 2. \* (q23 + q01);

m[2] = 2. \* (q13 + q02);

m[5] = 2. \* (q23 - q01);

m[8] = 1. - 2. \* (q1s + q2s);

}

**void** AGR(**double** t, **double** x[n], **double** Agrav[3]) { //ускорение за счет влияния несферичности гравитационного поля

**double** muz = 3.986004418 \* 1e+14, Rz = 6.378137 \* 1e+6, J2 = 1.08262668355 \* 1e-3; //константы

**double** rj2000[3] = { x[0] \* 1000, x[1] \* 1000,x[2] \* 1000 }, r[3]; //вектор в ск j2000

**double** aj2[3];//матрица aj2

**double** Mj2gr[9] = { 0 }; // матрица перехода из ск

**double** A = -19089.451590, B = 8640184.812866, C = 0.093104, D = -6.2 \* 1e-6, JD0 = 2451545, JDD = 26525; //константы

**double** DS2R = 7.272205216643039903848712 \* 1e-5;

**double** time = (JData(t) - JD0) / JDD;

**double** f = 86400 \* fmod(JData(t), 1.0);

**double** alfa = DS2R \* ((A + (B + (C + D \* time) \* time) \* time) + f);

alfa = fmod(alfa, 2 \* pi);

**if** (alfa < 0) { alfa = alfa + 2 \* pi; }

Mj2gr[0] = cos(alfa); Mj2gr[1] = sin(alfa);

Mj2gr[3] = -sin(alfa); Mj2gr[4] = cos(alfa);

Mj2gr[8] = 1; //посчитали Mj2

matr3Xvect3(Mj2gr, rj2000, r);//считаем вектор r

**double** absr = sqrt(pow(r[0], 2) + pow(r[1], 2) + pow(r[2], 2));

aj2[0] = -1.5 \* J2 \* (muz / pow(absr, 2)) \* (Rz / absr) \* (1 - 5 \* pow(r[2] / absr, 2)) \* (r[0] / absr);

aj2[1] = -1.5 \* J2 \* (muz / pow(absr, 2)) \* (Rz / absr) \* (1 - 5 \* pow(r[2] / absr, 2)) \* (r[1] / absr);

aj2[2] = -1.5 \* J2 \* (muz / pow(absr, 2)) \* (Rz / absr) \* (3 - 5 \* pow(r[2] / absr, 2)) \* (r[2] / absr);//посчитали aj2

**for** (i = 0; i < 3; i++) { aj2[i] = aj2[i]/1000; }

**double** Mj2grT[9] = { Mj2gr[0],Mj2gr[3],Mj2gr[6],Mj2gr[1],Mj2gr[4],Mj2gr[7],Mj2gr[2],Mj2gr[5],Mj2gr[8] }; //транспонируюм матрицу Mj2k

matr3Xvect3(Mj2grT, aj2, Agrav);//получили agrav

}

**void** ASUN(**double** t, **double** x[n], **double** Asun[3]) {

**double** rj2000[3] = { x[0], x[1],x[2] }, rs[3]; //вектор rj2000 и вектор на солнце

**double** mus = 132712517951, eps = 23.43929111 \* pi / 180, omOm = 282.94 \* pi / 180; //константы

**double** T = (JData(t) - 2451545.5) / 36525.0, M = (357.5226 + 35999.049 \* T) \* pi / 180; //улучшенное Юлианское время

**double** lambda = omOm + M + 6892 \* pi \* sin(M) / (648000) + 72 \* pi \* sin(2 \* M) / (648000); //ламбда с точкой в кружочку

**double** absR = (149.619 - 2.499 \* cos(M) - 0.021 \* cos(2 \* M)) \* pow(10, 6); //модуль вектора с точкой в кружочке

rs[0] = absR \* cos(lambda);

rs[1] = absR \* sin(lambda) \* cos(eps);

rs[2] = absR \* sin(lambda) \* sin(eps); //считаем вектор на солнце

**for** (i = 0; i < 3; i++) {

Asun[i] = mus \* ((rs[i] - rj2000[i]) / (pow(sqrt(pow(rs[0] - rj2000[0], 2) + pow(rs[1] - rj2000[1], 2) + pow(rs[2] - rj2000[2], 2)), 3) - rs[i] / pow(sqrt(rs[0] \* rs[0] + rs[1] \* rs[1] + rs[2] \* rs[2]), 3)));

}//гравитационное ускорение, за счет влияния солнца

}

**void** RP(**double** t, **double** x[n], **double** dx[n]) { //Функция правых частей

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**double** Ommax = 628, Lmax = 0.01, L[3]; **double** q = sign(x[6]);

**double** kp = 0.1, kd = 2;

L[0] = -(kp)\*q \* x[7] + -kd \* x[10];

L[1] = -(kp)\*q \* x[8] + -kd \* x[11];

L[2] = -(kp)\*q \* x[9] + -kd \* x[12];

**for** (i = 0; i < 3; ++i)

{

**if** (fabs(L[i]) >= Lmax) { L[i] = Lmax \* sign(L[i]); }

**else** **if** (fabs(x[i + 13]) >= Ommax) { L[i] = 0; }

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// M = ( 3\*mu / 2\* R^(3) ) \* (nu x J\*nu); nu = Q\_ x rj2000 x Q

**double** Mvozm[3];

**double** Q[4] = { x[6],x[7], x[8], x[9] }; **double** Q\_[4] = { x[6],-x[7], -x[8],-x[9] };

**double** rj2000[4] = { 0, x[0] \* 1000, x[1] \* 1000, x[2] \* 1000 };

**double** nu[4], nuk[3], Jnu[3], nJn[3];

**double** mu\_M = mu \* 1000;

**double** r0j2000[4] = { 0 };

**double** m\_rj2000 = fabs(sqrt(rj2000[0] \* rj2000[0] + rj2000[1] \* rj2000[1] + rj2000[2] \* rj2000[2]));

**for** (i = 1; i < 4; i++) r0j2000[i] = rj2000[i] / m\_rj2000; // нормированный вектор r0j2000

quatmul(Q\_, r0j2000, nu);

quatmul(Q, nu, nu);//Получили кватарнион ню

matr3Xvect3(J, &nu[1], Jnu);

vectrn(&nu[1], Jnu, nJn);

**for** (i = 0; i < 3; i++) Mvozm[i] = (3 \* mu\_M / (2 \* pow(m\_rj2000, 3))) \* nJn[i];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// L` = 1/2 \* L \* Lw

**double** om[4] = { 0,x[10],x[11],x[12] }, Rez[4];

quatmul(Q, om, Rez); //умножение кватранионов

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// w` = Jinv\*(Msum - w x (Jw\_Idm\*Om)-L)

**double** temp1[3], temp2[3], temp3[3], omega[3] = { x[10],x[11],x[12] }, OMB[3] = { x[13],x[14],x[15] };

matr3Xvect3(J, omega, temp1); //J\*w

**for** (i = 0; i < 3; ++i)

{

temp2[i] = Idm \* OMB[i];//Idm\*Om

temp3[i] = temp1[i] + temp2[i];//J\*w+Idm\*Om

}

vectrn(omega, temp3, temp1); //w\*(J\*w+Idm\*Om)

**for** (i = 0; i < 3; ++i) temp2[i] = Mvozm[i] - temp1[i] + L[i]; //M-w\*(J\*w+Idm\*Om)-L

matr3Xvect3(Jinv, temp2, temp1);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

AGR(t, x, agrav);

ASUN(t, x, asun);

dx[3] = -mu \* x[0] / pow(sqrt(x[0] \* x[0] + x[1] \* x[1] + x[2] \* x[2]), 3) + agrav[0] + asun[0];//уравнения(12)

dx[4] = -mu \* x[1] / pow(sqrt(x[0] \* x[0] + x[1] \* x[1] + x[2] \* x[2]), 3) + agrav[1] + asun[1];

dx[5] = -mu \* x[2] / pow(sqrt(x[0] \* x[0] + x[1] \* x[1] + x[2] \* x[2]), 3) + agrav[2] + asun[2];

dx[0] = x[3]; //dVx

dx[1] = x[4]; //dVy

dx[2] = x[5]; //dVz

dx[6] = 0.5 \* Rez[0]; //d lambda 0

dx[7] = 0.5 \* Rez[1]; //d lambda x

dx[8] = 0.5 \* Rez[2]; //d lambda y

dx[9] = 0.5 \* Rez[3]; //d lambda z

dx[10] = temp1[0]; //d om x

dx[11] = temp1[1]; //d om y

dx[12] = temp1[2]; //d om z

dx[13] = pow(Idm, -1) \* L[0]; //d Om x

dx[14] = pow(Idm, -1) \* L[1]; //d Om y

dx[15] = pow(Idm, -1) \* L[2]; //d Om z

}

**void** RK4(**void** f(**double**, **double**\*, **double**\*),

**double** t, **double**\* y, **double**\* yk, **int** n, **double** h)

{

**double**\* y1, \* y2, \* y3, \* y4, \* yy, h2 = h / 2, t1 = t + h2;

y1 = **new** **double**[n]; y2 = **new** **double**[n];

y3 = **new** **double**[n]; y4 = **new** **double**[n];

yy = **new** **double**[n];

f(t, y, y1); **for** (i = 0; i < n; i++)yy[i] = y[i] + h2 \* y1[i];

f(t1, yy, y2); **for** (i = 0; i < n; i++)yy[i] = y[i] + h2 \* y2[i];

f(t1, yy, y3); **for** (i = 0; i < n; i++)yy[i] = y[i] + h \* y3[i];

f(t + h, yy, y4);

**for** (i = 0; i < n; i++)

yk[i] = y[i] + h \* (y1[i] + y4[i] + 2 \* (y2[i] + y3[i])) / 6.;

**delete**[] yy;

**delete**[] y1;

**delete**[] y2;

**delete**[] y3;

**delete**[] y4;

}

**int** main() {

x[0] = 6237.377369; x[1] = 2319.470899; x[2] = -1398.276906; //x, y, z;

x[3] = -0.528889; x[4] = 4.936036; x[5] = 5.828682; //Vx, Vy, Vz;

x[6] = 0.632866; x[7] = -0.1826981; x[8] = 0.3653963; x[9] = -0.657713; //Q0, Q1, Q2, Q3; Кватранион

x[10] = pow(10, -5); x[11] = pow(10, -6); x[12] = pow(10, -5); //om; угловое вращение

x[13] = 0; x[14] = 0; x[15] = 0; //Om

ofstream fout;

fout.open("res.txt");

**while** (t <= tk)

{

RK4(RP, t, x, dx, n, h); //Рунге-КУтт

**for** (i = 0; i < n; i++) x[i] = dx[i];

//cout << t << ' ' << x[0] << ' ' << ' ' << x[1] << ' ' << x[2] << ' ' << x[3] << ' ' << x[4] << ' ' << x[5] << ' ' << x[6] << ' ' << x[7] << ' ' << x[8] << ' ' << x[9] << ' ' << x[10] << ' ' << x[11] << ' ' << x[12] << ' ' << x[13] << ' ' << x[14] << ' ' << x[15] << ' ' << endl;

fout << t << ' ' << x[0] << ' ' << ' ' << x[1] << ' ' << x[2] << ' ' << x[3] << ' ' << x[4] << ' ' << x[5] << ' ' << x[6] << ' ' << x[7] << ' ' << x[8] << ' ' << x[9] << ' ' << x[10] << ' ' << x[11] << ' ' << x[12] << ' ' << x[13] << ' ' << x[14] << ' ' << x[15] << ' ' << endl;

//

t += h;

}

fout.close();

}