УТВЕРЖДЁН
А.В.00000 - 00 00 00 0 – 0 32 01

Синтез алгоритмов наведения и стабилизации центра масс возвращаемого аппарата

Текст программы

А.В.00000 - 00 00 00 0 – 0 32 01

Листов 14

Подп. и дата	
ИнвNдубл	
Взаим инв. N	
Подп. и дата	
Инв. Nподп.	

2020

Литера

СОДЕРЖАНИЕ

1. Общие сведения 3

1.1. Среда разработки 3

1.2. Среда выполнения 3

2. Текст программы 4

2.1. Основные файлы программы 4

2.1.1. Файл SpaceCraft.m 4

2.1.2. Файл getCoefficient.m 13

2.1.3. Файл getPathSpeedBoost.m 13

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Среда разработки

Программа разработана в среде MatLab R2019b.

1.2. Среда выполнения

Программа предназначена для запуска в ОС Windows 10

2. ТЕКСТ ПРОГРАММЫ

2.1. Основные файлы программы

2.1.1. Файл SpaceCraft.m

```
clc;
clear variables;
close all;

%% Начальные условия
T = 26;
g = -9.8;
dt = 0.01;
t = 0:dt:T;
N = numel(t);
C = [0 1];
D = 0;
B = [0; 1 ];
l1=-8;
l2=-15;

A=[
    0 1;
    1 1
];

% sys = ss(A, B, C, D)
% sys2 = mksys(A, B, C, D);

% Массивы СВСК
FX1 = zeros(1,N); % суммарная сила, действующая относительно X1 [H]
FY1 = zeros(1,N); % суммарная сила, действующая относительно Y1 [H]
FZ1 = zeros(1,N); % суммарная сила, действующая относительно Z1 [H]
FuprY1 = zeros(1,N); % управляющая сила относительно Y1 [H]
FuprZ1 = zeros(1,N); % управляющая сила относительно Z1 [H]
Xc1 = zeros(1,N); % аэродинам. сила относительно X1 [H]
Yc1 = zeros(1,N); % аэродинам. сила относительно Y1 [H]
Zc1 = zeros(1,N); % аэродинам. сила относительно Z1 [H]
Teta = zeros(1,N); % угол [рад]
Fi = zeros(1,N);
Psi = zeros(1,N);
OmegaX1 = zeros(1,N); % угловая скорость относительно X1 [рад/с]
OmegaY1 = zeros(1,N); % угловая скорость относительно Y1 [рад/с]
OmegaZ1 = zeros(1,N); % угловая скорость относительно Z1 [рад/с]
MX1 = zeros(1,N); % суммарный момент относительно X1 [Н*м]
MY1 = zeros(1,N); % суммарный момент относительно Y1 [Н*м]
MZ1 = zeros(1,N); % суммарный момент относительно Z1 [Н*м]
MuprX1 = zeros(1,N); % управляющий момент относительно X1 [Н*м]
MuprY1 = zeros(1,N); % управляющий момент относительно Y1 [Н*м]
MuprZ1 = zeros(1,N); % управляющий момент относительно Z1 [Н*м]
MaY1 = zeros(1,N); % аэродинам. момент относительно Y1 [Н*м]
MaZ1 = zeros(1,N); % аэродинам. момент относительно Z1 [Н*м]
McmY1 = zeros(1,N); % момент смещения ЦМ относительно Y1 [Н*м]
McmZ1 = zeros(1,N); % момент смещения ЦМ относительно Z1 [Н*м]
```

```
Rsum = zeros(1,N); % суммарная тяга ПТДУ [Н]
R_TP = zeros(1,N); % требуемая тяга [Н]
dTeta = zeros(1,N);
dPsi = zeros(1,N);
dFi = zeros(1,N);
PsiTP = zeros(1,N);
TetaTP = zeros(1,N);
X1TP = zeros(1,N);
Y1TP = zeros(1,N);
Z1TP = zeros(1,N);
VX1TP = zeros(1,N);
VY1TP = zeros(1,N);
VZ1TP = zeros(1,N);
WX1TP = zeros(1,N);
WY1TP = zeros(1,N);
WZ1TP = zeros(1,N);
KsiTP = zeros(1,N);
VksiTP = zeros(1,N);
WksiTP = zeros(1,N);
EtaTP = zeros(1,N);
VetaTP = zeros(1,N);
WetaTP = zeros(1,N);
DzetaTP = zeros(1,N);
VdzetaTP = zeros(1,N);
WdzetaTP = zeros(1,N);
%-------НУ и коэффициенты-----------------------------------------------------
ro = 1.25; % плотность атмосферы [кг*сек^2/м^4]
m0 = 7770; % начальная масса аппарата [кг]
Sm = 15.2; % площадь миделя аппарата [м^2]
Yt = 0.2; % смещение ЦМ относительно оси X1 [м]
Zt = 0; % смещение ЦМ относительно оси Z1 [м]
L = 3.9; % длина аппарата [м]
Ixx = 19000; % момент инерции вокруг оси X [кг*м^2]
Iyy = 17000; % момент инерции вокруг оси Y [кг*м^2]
Izz = 17000; % момент инерции вокруг оси Z [кг*м^2]
g = 9.81; % ускорение свободного падения [м/с^2]
R_TP_min = 6500*9.81; % минимальная требуемая тяга [Н]
R_TP_max = 24000*9.81; % максимальная требуемая тяга [Н]
% параметры сопел
alphaC = deg2rad(5.73); % угол наклона сопел к продольной оси X [рад]
lc = 0.4; % плечо сопел [м]
Loy = 2;
Lx = Loy*cot(alphaC);
Xt = 1.35;
lx = Lx - Xt;
Lc = lx*sin(alphaC);
deltasummax = deg2rad(480);
Kteta = 0.16*57.3;
Kpsi = 0.16*57.3;
Kfi = 5*57.3;
Kw = 0;
Kwt = 0;
tauP = 0.4;
tauB = 0.4;
% углы
Fi(1,1) = 0; % [рад]
Psi(1,1) = 0; % [рад]
Teta(1,1) = -pi/2; % [рад]
Fik = 0; % [рад]
Psik = 0; % [рад]
```

```
Tetak = -pi/2; % [рад]
PsiTP(1,1) = 0;
TetaTP(1,1) = Tetak;
% Расчет НУ в СВСК
a11 = cos(Teta(1,1))*cos(Psi(1,1));
a12 = cos(Teta(1,1))*sin(Psi(1,1))*sin(Fi(1,1)) - sin(Teta(1,1))*cos(Fi(1,1));
a13 = cos(Teta(1,1))*sin(Psi(1,1))*cos(Fi(1,1)) + sin(Teta(1,1))*sin(Fi(1,1));
a21 = sin(Teta(1,1))*cos(Psi(1,1));
a22 = sin(Teta(1,1))*sin(Psi(1,1))*sin(Fi(1,1)) + cos(Teta(1,1))*cos(Fi(1,1));
a23 = sin(Teta(1,1))*sin(Psi(1,1))*cos(Fi(1,1)) - cos(Teta(1,1))*sin(Fi(1,1));
a31 = -sin(Psi(1,1));
a32 = cos(Psi(1,1))*sin(Fi(1,1));
a33 = cos(Psi(1,1))*cos(Fi(1,1));

%% Задание полюсов
% p = [-8; -15];
% pe = 2*[-8; -7];
% K = place(A, B, p);
% L = acker(A, B, pe)';

%% Формирование наблюдателя с выбранными полюсами
% newsys = append(sys, K)
% rsys = reg(sys, K, L);
% sys1 = parallel(sys, rsys, [1], [1], [2]);
% sys2 = connect(sys1, );

%% Проверка управляемости и наблюдаемости
% Co = ctrb(A, B);
% Unco = length(A) - rank(Co);
% Do = obsv(sys);
% Undo = length(A) - rank(Do);

%%  Расчёт траектории по эта, скорости и ускорения с учётом алгоритмов
описывающих функций

%   Начальные условия по координате Кси
xiPathInit = 100;
xiSpeedInit = 0;
xiBoostInit = 0;

%   Конечные условия по координате Кси
xiPathFin = 0;
xiSpeedFin = 0;
xiBoostFin = 0;

%   Получение коэффициентов по координате Кси
[coefXi0, coefXi1, coefXi2, coefXi3, coefXi4, coefXi5] =
getCoefficient(xiPathInit, xiSpeedInit, xiBoostInit, xiPathFin, xiSpeedFin,
xiBoostFin, T);

%   pathXi - траектория пути по Кси
%   speedXi - скорость по Кси
%   boostXi - ускорения по Кси
[pathXi, speedXi, boostXi] = getPathSpeedBoost(coefXi0, coefXi1, coefXi2,
coefXi3, coefXi4, coefXi5, t, N);

%   Начальные условия по координате Эта
etaPathInit = 800;
```

```
etaSpeedInit = -70;
etaBoostInit = etaSpeedInit ^ 2 / (2 * etaPathInit);

%   Конечные условия по координате Эта
etaPathFin = 20;
etaSpeedFin = -2;
etaBoostFin = 0;

%   Получение коэффициентов по координате Эта
[coefEta0, coefEta1, coefEta2, coefEta3, coefEta4, coefEta5] =
getCoefficient(etaPathInit, etaSpeedInit, etaBoostInit, etaPathFin, etaSpeedFin,
etaBoostFin, T);

%   pathEta - траектория пути по Эта
%   speedEta - скорость по Эта
%   boostEta - ускорения по эта
[pathEta, speedEta, boostEta] = getPathSpeedBoost(coefEta0, coefEta1, coefEta2,
coefEta3, coefEta4, coefEta5, t, N);



%   Начальные условия по координат Дзета
zetaPathInit = -100;
zetaSpeedInit = 0;
zetaBoostInit = 0;

%   Конечные условия по координате Дзета
zetaPathFin = 0;
zetaSpeedFin = 0;
zetaBoostFin = 0;

%   Получение коэффициентов по координате Дзета
[coefZeta0, coefZeta1, coefZeta2, coefZeta3, coefZeta4, coefZeta5] =
getCoefficient(zetaPathInit, zetaSpeedInit, zetaBoostInit, zetaPathFin,
zetaSpeedFin, zetaBoostFin, T);

%   pathZeta - траектория пути по Кси
%   speedZeta - скорость по Кси
%   boostZeta - ускорения по Кси
[pathZeta, speedZeta, boostZeta] = getPathSpeedBoost(coefZeta0, coefZeta1,
coefZeta2, coefZeta3, coefZeta4, coefZeta5, t, N);

%%  Моделирование системы по желаемой траектории
% f0 = [y01 y0]
% [f, t, h] = lsim(sys, y, t, f0);
% [freg, t, hreg ] = lsim(rsys, y, t, f0);

for i=1:N
     X1TP(1,i) = a11*pathXi(1,i) + a21*pathEta(1,i) + a31*pathZeta(1,i);
     VX1TP(1,i) = a11*speedXi(1,i) + a21*speedEta(1,i) + a31*speedZeta(1,i);
     WX1TP(1,i) = a11*boostXi(1,i) + a21*boostEta(1,i) + a31*boostZeta(1,i);

     Y1TP(1,i) = a12*pathXi(1,i) + a22*pathEta(1,i) + a32*pathZeta(1,i);
     VY1TP(1,i) = a12*speedXi(1,i) + a22*speedEta(1,i) + a32*speedZeta(1,i);
     WY1TP(1,i) = a12*boostXi(1,i) + a22*boostEta(1,i) + a32*boostZeta(1,i);

     Z1TP(1,i) = a13*pathXi(1,i) + a23*pathEta(1,i) + a33*pathZeta(1,i);
```

```
     VZ1TP(1,i) = a13*speedXi(1,i) + a23*speedEta(1,i) + a33*speedZeta(1,i);
     WZ1TP(1,i) = a13*boostXi(1,i) + a23*boostEta(1,i) + a33*boostZeta(1,i);
end

for i=1:N-1
     % Пересчет матрицы направляющих косинусов
     a11 = cos(Teta(1,i))*cos(Psi(1,i));
     a12 = cos(Teta(1,i))*sin(Psi(1,i))*sin(Fi(1,i)) -
sin(Teta(1,i))*cos(Fi(1,i));
     a13 = cos(Teta(1,i))*sin(Psi(1,i))*cos(Fi(1,i)) +
sin(Teta(1,i))*sin(Fi(1,i));
     a21 = sin(Teta(1,i))*cos(Psi(1,i));
     a22 = sin(Teta(1,i))*sin(Psi(1,i))*sin(Fi(1,i)) +
cos(Teta(1,i))*cos(Fi(1,i));
     a23 = sin(Teta(1,i))*sin(Psi(1,i))*cos(Fi(1,i)) -
cos(Teta(1,i))*sin(Fi(1,i));
     a31 = -sin(Psi(1,i));
     a32 = cos(Psi(1,i))*sin(Fi(1,i));
     a33 = cos(Psi(1,i))*cos(Fi(1,i));

    % Требуемая ориентация вектора тяги eR
     Wksi_TP = boostXi(1,i);
     Weta_TP = boostEta(1,i) + g;
     Wdzeta_TP = boostZeta(1,i);
     W_TP = (Wksi_TP^2 + Weta_TP^2 + Wdzeta_TP^2)^0.5;

     eRksi = - Wksi_TP / W_TP;
     eReta = - Weta_TP / W_TP;
     eRdzeta = - Wdzeta_TP / W_TP;

     % eR - требуемое направление оси X1 (против направления тяги)
     PsiTP(1,i) = - asin(eRdzeta);

     if eRksi >= 0 && eReta < 0
        TetaTP(1,i) = - acos(eRksi);
     end
     if eRksi < 0 && eReta < 0
        TetaTP(1,i) = - acos(eRksi);
     end

     if TetaTP(1,i) < -deg2rad(110)
        TetaTP(1,i) = -deg2rad(110);
     end
     if TetaTP(1,i) > -deg2rad(70)
        TetaTP(1,i) = -deg2rad(70);
     end
     if PsiTP(1,i) < -deg2rad(20)
        PsiTP(1,i) = -deg2rad(20);
     end
     if PsiTP(1,i) > deg2rad(20)
        PsiTP(1,i) = deg2rad(20);
     end
     dPsi(1,i) = Psi(1,i) - PsiTP(1,i);
     dTeta(1,i) = Teta(1,i) - TetaTP(1,i);
     dFi(1,i) = Fi(1,i);
     deltaP = Kpsi*dPsi(1,i) + Kpsi*tauP*OmegaY1(1,i);
     deltaT = Kteta*dTeta(1,i) + Kteta*tauP*OmegaZ1(1,i);
     deltaB = Kfi*dFi(1,i) + Kfi*tauB*OmegaX1(1,i);
```

```
% требуемая тяга
R_TP_X1 = m0 * W_TP;
R_TP(1,i) = R_TP_X1 / 0.7;
if R_TP(1,i) > R_TP_max
   R_TP(1,i) = R_TP_max;
end
if R_TP(1,i) < R_TP_min
   R_TP(1,i) = R_TP_min;
end

deltaR = 0;
if (R_TP(1,i) > 63765 && R_TP(1,i) < 147150)
deltaR = deg2rad(36);
end
if (R_TP(1,i) > 147150 && R_TP(1,i) < 235440)
deltaR = deg2rad(30);
end
if R_TP(1,i) == 235440
deltaR = deg2rad(24);
end

delta1 = deltaR + deltaT + deltaB;
delta2 = deltaR + deltaT - deltaB;
delta3 = deltaR - deltaP + deltaB;
delta4 = deltaR - deltaP - deltaB;
delta5 = deltaR - deltaT + deltaB;
delta6 = deltaR - deltaT - deltaB;
delta7 = deltaR + deltaP + deltaB;
delta8 = deltaR + deltaP - deltaB;
R1 = R_TP(1,i)*delta1/deltasummax;
R2 = R_TP(1,i)*delta2/deltasummax;
R3 = R_TP(1,i)*delta3/deltasummax;
R4 = R_TP(1,i)*delta4/deltasummax;
R5 = R_TP(1,i)*delta5/deltasummax;
R6 = R_TP(1,i)*delta6/deltasummax;
R7 = R_TP(1,i)*delta7/deltasummax;
R8 = R_TP(1,i)*delta8/deltasummax;
if ((abs(VZ1TP(1,i))<=0.1) && (VY1TP(1,i)<=0))
   Gamma = 0;
end
if ((abs(VZ1TP(1,i))<=0.1) && (VY1TP(1,i)>0))
   Gamma = deg2rad(180);
end
if VZ1TP(1,i)>0.1
   Gamma = atan(VY1TP(1,i)/VZ1TP(1,i)) + deg2rad(90);
end
if VZ1TP(1,i)<-0.1
   Gamma = atan(VY1TP(1,i)/VZ1TP(1,i)) + deg2rad(270);
end

if i>2
    al = atan(-VY1TP(1,i)/VX1TP(1,i)); % угол атаки
    bet = atan(VZ1TP(1,i)/VX1TP(1,i)); % угол скольжения
else
    al = 0;
    bet = 0;
end
```

```
V = (VX1TP(1,i)^2 + VY1TP(1,i)^2 + VZ1TP(1,i)^2)^0.5;
q = ro*V^2/2;

if al >= 0 && al < 10
    Cx = 0.812;
    Cy = 0.024;
    mz = 0.03;
end
if al >= 10 && al < 20
    Cx = 0.839;
    Cy = 0.026;
    mz = 0.0;
end
if al >= 20 && al < 30
    Cx = 0.776;
    Cy = -0.024;
    mz = -0.029;
end
if al >= 30 && al < 40
    Cx = 0.69;
    Cy = 0.016;
    mz = -0.05;
end
if al >= 40 && al < 50
    Cx = 0.571;
    Cy = 0.09;
    mz = -0.06;
end
if al >= 50 && al < 60
    Cx = 0.48;
    Cy = 0.25;
    mz = -0.04;
end
if al >= 60 && al < 70
    Cx = 0.38;
    Cy = 0.38;
    mz = -0.02;
end
if al >= 70 && al < 80
    Cx = 0.28;
    Cy = 0.37;
    mz = -0.013;
end
if al >= 80 && al < 90
    Cx = 0.08;
    Cy = 0.34;
    mz = -0.009;
end
if al >= 90
   mz = 0;
end

MaY1(1,i) = mz*q*Sm*L*sign(bet);
MaZ1(1,i) = mz*q*Sm*L*sign(al);
McmY1(1,i) = R_TP(1,i)*Zt;
McmZ1(1,i) = -R_TP(1,i)*Yt;
MuprX1(1,i) = (R2 + R4 + R6 + R8 - R1 - R3 - R5 - R7)*sin(alphaC)*lc;
MuprY1(1,i) = (R3 + R4 - R7 - R8)*Lc;
MuprZ1(1,i) = (R5 + R6 - R1 - R2)*Lc;
```

```
     FuprY1(1, i) = -1 * (R5 + R6 - R1 - R2)*sin(alphaC);
     FuprZ1(1, i) = (R3 + R4 - R7 - R8)*sin(alphaC);
     MX1(1,i) = MuprX1(1,i);
     MY1(1,i) = MuprY1(1,i) + 0*MaY1(1,i) + 0*McmY1(1,i);
     MZ1(1,i) = MuprZ1(1,i) + 0*MaZ1(1,i) + 0*McmZ1(1,i);
     Teta(1,i+1) = Teta(1,i) +
dt*(1/cos(Psi(1,i))*(OmegaY1(1,i)*sin(Fi(1,i))+OmegaZ1(1,i)*cos(Fi(1,i))));
     Psi(1,i+1) = Psi(1,i) + dt*(OmegaY1(1,i)*cos(Fi(1,i))-
OmegaZ1(1,i)*sin(Fi(1,i)));
     Fi(1,i+1) = Fi(1,i) +
dt*(OmegaX1(1,i)+tan(Psi(1,i))*(OmegaY1(1,i)*sin(Fi(1,i))+OmegaZ1(1,i)*cos(Fi(1,
i))));
     OmegaX1(1,i+1) = OmegaX1(1,i) + dt*MX1(1,i)/Ixx;
     OmegaY1(1,i+1) = OmegaY1(1,i) + dt*MY1(1,i)/Iyy;
     OmegaZ1(1,i+1) = OmegaZ1(1,i) + dt*MZ1(1,i)/Izz;
end

[coefEta0, coefEta1, coefEta2, coefEta3, coefEta4, coefEta5] =
getCoefficient(etaPathInit-5, etaSpeedInit+1, etaBoostInit, etaPathFin,
etaSpeedFin, etaBoostFin, T);

%   pathEta - траектория пути по Эта
%   speedEta - скорость по Эта
%   boostEta - ускорения по эта
[pathEtaBias, speedEtaBias, boostEtaBias] = getPathSpeedBoost(coefEta0,
coefEta1, coefEta2, coefEta3, coefEta4, coefEta5, t, N);
p = [-0.01; -0.5];
pe = 2*[-8; -7];
K = place(A, B, p);
u = zeros(1, N);
Kr = p(1, 1) * p(2, 1) *0.001;
% K(1, 1) = 0.5;
% K(1, 2) = 2.5;
for i = 1:N-1
        u(1, i) = - K(1, 1) * (pathEtaBias(1, i) - pathEta(1, i)) - K(1, 2) *
(speedEtaBias(1, i) - speedEta(1, i)) +Kr*pathEta(1, i);
        speedEtaBias(1, i+1) = speedEtaBias(1, i) + dt*u(1, i);
        pathEtaBias(1, i+1) = pathEtaBias(1, i) + dt*speedEtaBias(1, i);

end

% Графики траекторий
%   Траектория по Кси
figure('Name', 'Траектория движения по координате Кси, м');
subplot(3, 3, 1);
plot(t, pathXi);
title('Траектория движения по координате Кси, м');
xlabel('Время, с');

%   Траектория по Эта
subplot(3, 3, 2);
plot(t, pathEta);
title('Траектория движения по координате Эта, м')
xlabel('Время, с');

%   Траектория по Дзета
subplot(3, 3, 3);
plot(t, pathZeta);
title('Траектория движения по координате Дзета, м')
```

```
xlabel('Время, с');

%   Трехмерный график
% subplot(3, 3, 4);
% plot3(pathZeta, pathXi, pathEta);

%%  Графики изменения скорости
%   Скорость по Кси
subplot(3, 3, 4);
plot(t, speedXi);
title('График изменения скорости по Кси');
xlabel('Время, с');

%   Скорость по Эта
subplot(3, 3, 5);
plot(t, speedEta);
title('График изменения скорости по Эта');
xlabel('Время, с');

%   Скорость по Дзета
subplot(3, 3, 6);
plot(t, speedZeta);
title('График изменения скорости по Дзета');
xlabel('Время, с');

%%  Графики изменения ускорения
%   Ускорение по Кси
subplot(3, 3, 7);
plot(t, boostXi);
title('График изменения ускорения по Кси');
xlabel('Время, с');

%   Ускорение по Эта
subplot(3, 3, 8);
plot(t, boostEta);
title('График изменения ускорения по Эта');
xlabel('Время, с');

%  Ускорение по Дзета
subplot(3, 3, 9);
plot(t, boostZeta);
title('График изменения ускорения по Дзета');
xlabel('Время, с');
%%
% Тректория с учетом модального управления
figure('Name', 'Тректория с учетом модального управления')
plot(t, pathEta);
ylabel('Высота, м')
xlabel('Время, с');
hold on;
plot(t, pathEtaBias)
legend('Требуемая траектория', 'Траектория с учетом модального управления')
ylabel('Высота, м')
xlabel('Время, с');

% Графики управлений
figure('Name', 'управлений')
plot(t, u);
```

```
% График ошибки
figure('Name', 'Отклонение от требуемой траектории')
plot(t, (pathEtaBias - pathEta))
ylabel('м')
xlabel('Время, с');

% Пространственная траектория
figure('Name', 'Пространственная траектория')
plot3(pathXi, pathZeta, pathEtaBias);
grid on;
hold all;
xlabel('\xi');
ylabel('\zeta');
zlabel('\eta');
```

2.1.2. Файл getCoefficient.m

```
function [coef0, coef1, coef2, coef3, coef4, coef5] = getCoefficient(pathInit,
speedInit, boostInit, pathFin, speedFin, boostFin, T)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
    coef0 = pathInit;
    coef1 = speedInit;
    coef2 = boostInit/2;
    coef3 = boostFin /(2 * T) - (6 * speedInit) / T ^ 2 - (3 * boostInit)/(2*T)
- (4*speedFin)/T^2 - (10*pathInit)/T^3 + (10*pathFin)/T^3;
    coef4 = (15 * pathInit)/T^4 + (8 * speedInit) / T ^ 3 + (3 *
boostInit)/(2*T^2) + (7*speedFin)/T^3 - boostFin/T^2 - (15*pathFin)/T ^ 4;
    coef5 = boostFin / (2 * T ^ 3) - (3 * speedInit) / T ^ 4 - boostInit / (2 *
T ^ 3) - (3 * speedFin) / T ^ 4 - (6 * pathInit) / T ^ 5 + (6 * pathFin) / T ^
5;
end
```

2.1.3. Файл getPathSpeedBoost.m

```
function [path, speed, boost] = getPathSpeedBoost(a0, a1, a2, a3, a4, a5, t, N)
%Функция getPathSpeedBoost принимает в качестве аргументов коэффициенты
%a0-a5, t - вектор времени, N - количество точек
    path = zeros(1, N);
    speed = zeros(1, N);
    boost = zeros(1, N);
    for i = 1:N
        path(1, i) = a0 + a1*t(1, i) + a2*t(1, i)^2 + a3*t(1, i)^3 + a4*t(1,
i)^4 + a5*t(1, i)^5;
        speed(1, i) = a1 + 2*a2*t(1, i) + 3*a3*t(1, i)^2 + 4*a4*t(1, i)^3 +
5*a5*t(1, i)^4;
        boost(1, i) = 2*a2 + 6*a3*t(1, i) + 12*a4*t(1, i)^2 + 20*a5*t(1, i)^3;
    end
end
```

s									
Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Входящий № сопроводительного докум. и дата	Подп.	Дата
	измененных	замененных	новых	аннулированных					