Приложение А

Ниже представлены результаты интегрирования траектории. Координаты цели приняты: x = 2492.7 м, z = 1888 м.

Таблица А.1 – Результаты интегрирования

t,s	x,m	y,m	z,m	Vx,m/s	Vy,m/s	Vz,m/s	V,m/s	ϑ,grad	ψ,grad	γ,grad	ωx,grad/s	ωy,grad/s	ωz,grad/s
0.00	0.00	9550.00	0.00	627.00	-1023.17	0.00	1200.00	-58.500	0.000	0.000	0.00000	0.00000	0.00000
1.00	651.43	8554.11	1.11	674.80	-967.52	0.25	1179.59	-52.742	1.396	2.377	-0.20227	0.09372	-0.05063
2.00	1347.80	7616.56	2.37	716.63	-906.93	-0.12	1155.89	-49.000	2.030	5.279	0.15162	0.41102	0.01595
3.00	2081.89	6741.04	3.91	750.12	-843.85	0.06	1129.05	-45.752	1.258	-3.651	0.10881	0.91292	0.09207
4.00	2844.86	5929.85	4.49	773.28	-778.61	8.24	1097.39	-42.592	-9.237	-1.520	0.01250	0.30846	0.01178
5.00	3626.19	5184.14	7.22	787.25	-713.65	-4.79	1062.58	-39.662	1.958	-3.097	-0.06098	-1.15719	0.00443
6.00	4417.00	4502.47	8.79	792.80	-650.54	1.57	1025.54	-36.354	6.008	7.401	-0.03309	0.71263	0.04627
7.00	5209.18	3882.03	9.64	790.43	-590.91	4.94	986.90	-33.737	-5.841	-5.879	0.15186	0.50571	0.06218
8.00	5995.86	3318.88	10.95	782.03	-535.97	-0.36	948.06	-31.639	-0.811	-0.127	-0.29400	-0.66947	0.03229
9.00	6771.42	2808.58	12.52	768.33	-485.34	1.41	908.78	-29.468	1.754	2.866	0.03087	0.13337	0.03329
10.00	7531.00	2346.73	13.63	750.13	-439.09	1.43	869.20	-27.545	-0.928	-0.559	0.13928	0.02148	0.03205
11.00	8270.42	1928.96	14.94	728.14	-397.15	1.28	829.41	-25.815	0.049	-0.241	-0.19539	0.00495	0.02850
12.00	8986.28	1551.02	16.12	703.14	-359.43	1.14	789.68	-24.286	-0.094	-0.209	0.02036	-0.00928	0.02492
13.00	9675.93	1208.76	17.28	675.85	-325.74	1.14	750.25	-22.953	-0.110	0.000	-0.00059	0.00137	0.02159
14.00	10337.40	898.28	18.39	646.84	-295.84	1.09	711.28	-21.809	-0.094	0.000	-0.00013	0.00031	0.01837
15.00	10969.20	615.94	19.45	616.52	-269.40	1.03	672.81	-20.845	-0.095	0.000	0.00002	-0.00006	0.01532
16.00	11570.10	358.46	20.45	585.07	-246.04	0.98	634.70	-20.051	-0.096	0.000	0.00001	-0.00003	0.01243
17.00	12138.90	122.99	21.41	552.33	-225.29	0.93	596.51	-19.417	-0.096	0.000	0.00000	-0.00001	0.00972
17.56	12442.50	0.00	21.92	533.24	-214.60	0.89	574.80	-19.129	-0.096	0.000	0.00001	-0.00003	0.00828

Продолжение таблицы А.1

t,s	α,grad	β,grad	δT, grad	δN, grad	δE, grad	r,m
0.00	0.00000	0.00000	15.00	-2.03	0.00	15725
1.00	2.33669	0.90829	15.00	-0.46	15.00	14608
2.00	2.57616	1.49555	15.00	2.69	15.00	13499
3.00	2.66821	0.67048	15.00	7.68	-15.00	12403
4.00	2.76681	-6.14319	15.00	-15.00	-15.00	11325
5.00	2.60202	1.05308	15.00	-15.00	-15.00	10271
6.00	2.54137	5.09823	15.00	9.96	15.00	9246
7.00	2.70262	-4.69068	15.00	14.52	-15.00	8253
8.00	2.78712	-0.69728	15.00	-6.27	-15.00	7295
9.00	2.74285	1.71125	15.00	0.87	15.00	6373
10.00	2.79379	-0.73407	15.00	0.37	-15.00	5489
11.00	2.79470	0.11970	15.00	0.12	-15.00	4642
12.00	2.78924	-0.01187	15.00	-0.09	-15.00	3834
13.00	2.77961	-0.01253	15.00	0.01	0.05	3065
14.00	2.76831	0.00218	15.00	0.00	-0.01	2335
15.00	2.75916	0.00070	15.00	0.00	0.00	1643
16.00	2.75784	0.00003	15.00	0.00	0.00	990
17.00	2.77287	-0.00008	15.00	0.00	0.00	375
17.56	2.79352	0.00007	15.00	0.00	0.00	50

На рисунках А.1-А.14 представлены графические зависимости полученной траектории.

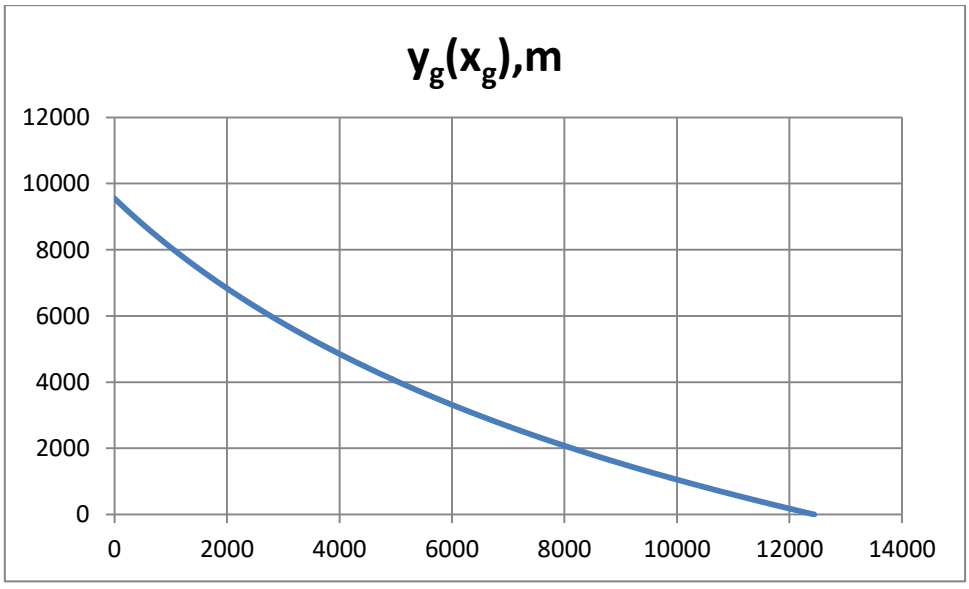

Рисунок А.1 – Траектория в проекции на вертикальную плоскость ОХУ

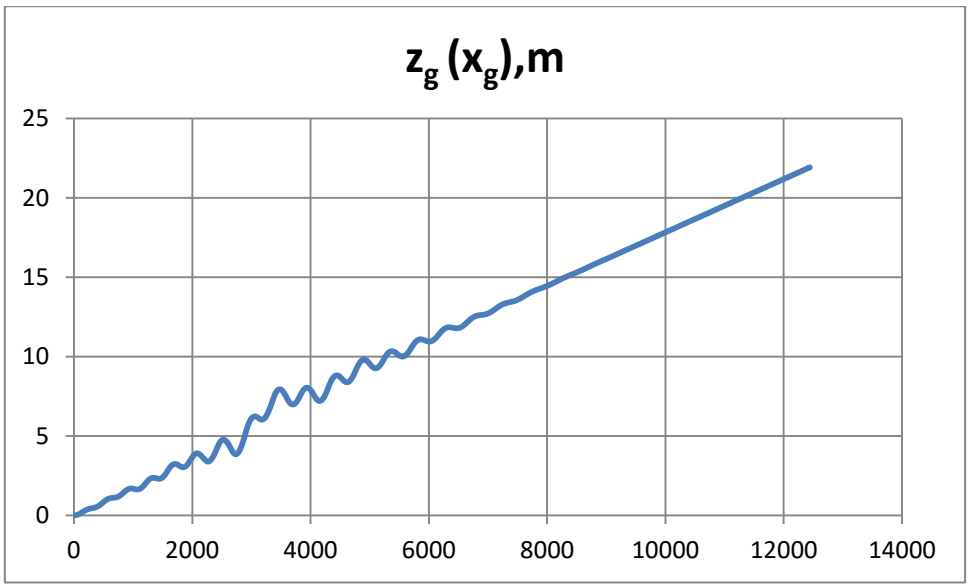

Рисунок А.2 – Траектория в проекции на горизонтальную плоскость OXZ

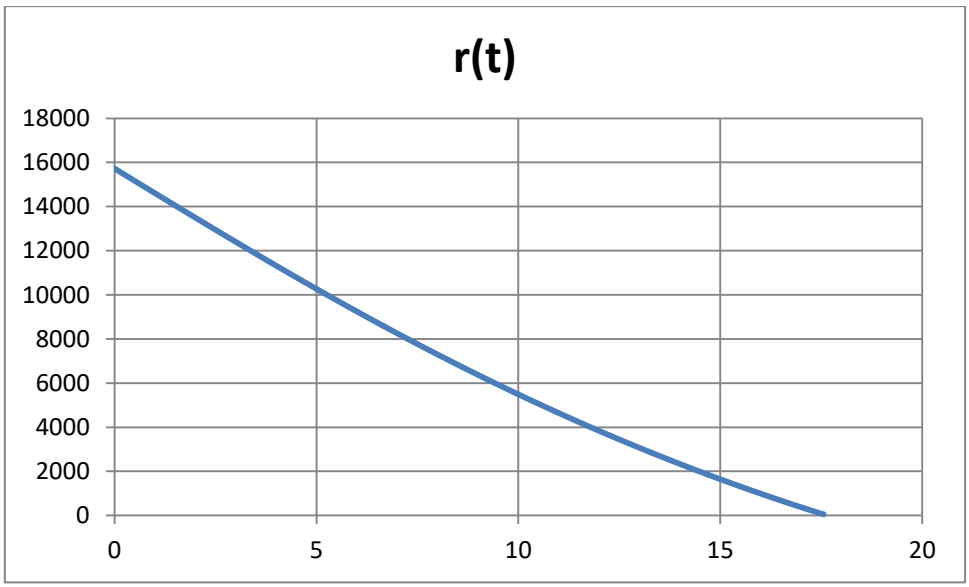

Рисунок А.3 – Зависимость радиус-вектора до цели от времени полета

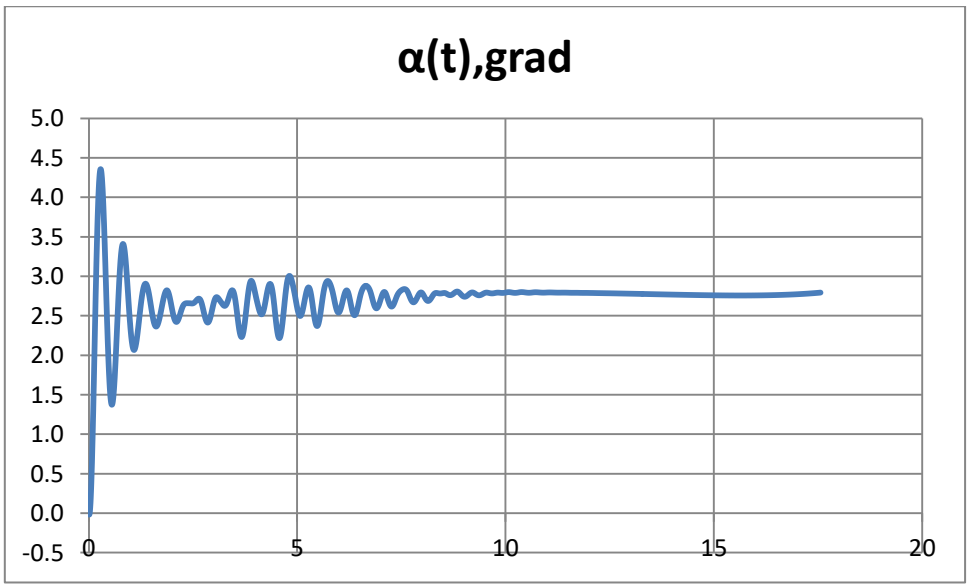

Рисунок А.4 – Зависимость угла атаки от времени полета

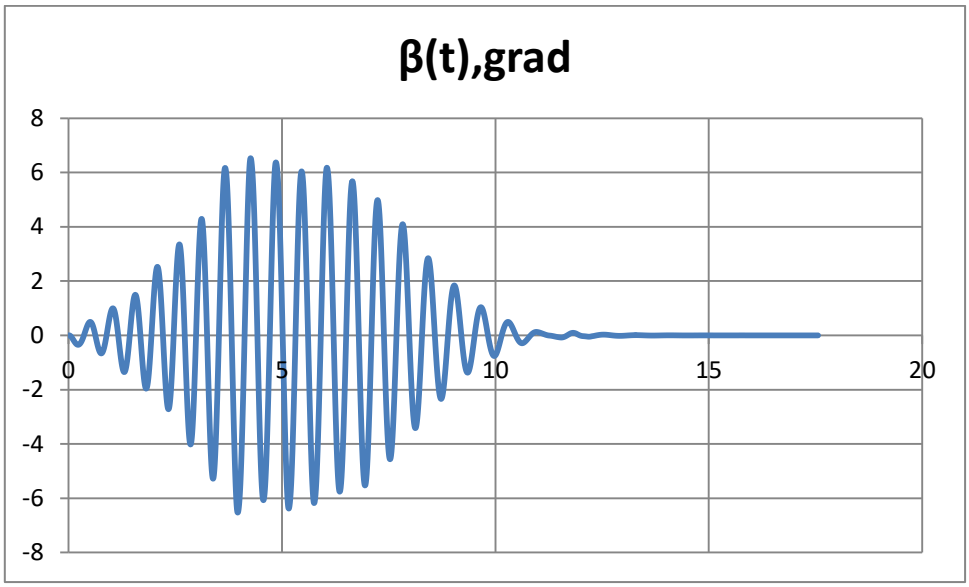

Рисунок А.5 – Зависимость угла скольжения от времени полета

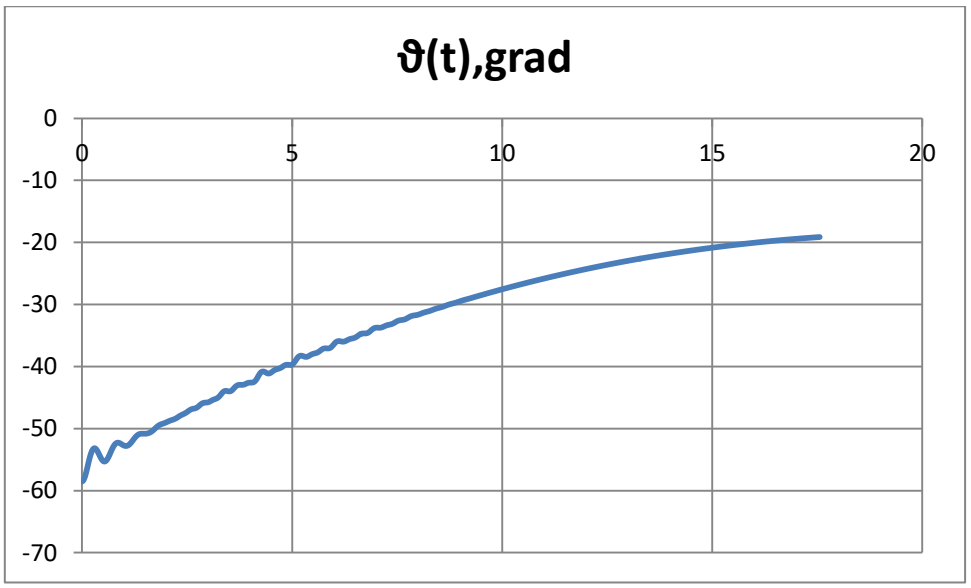

Рисунок А.6 – Зависимость угла тангажа от времени полета

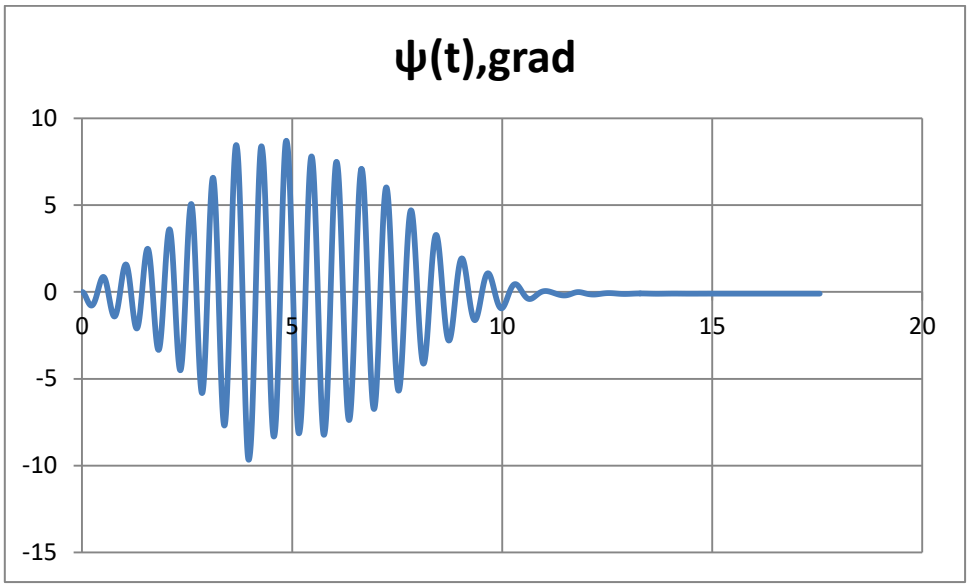

Рисунок А.7 – Зависимость угла рысканья от времени полета

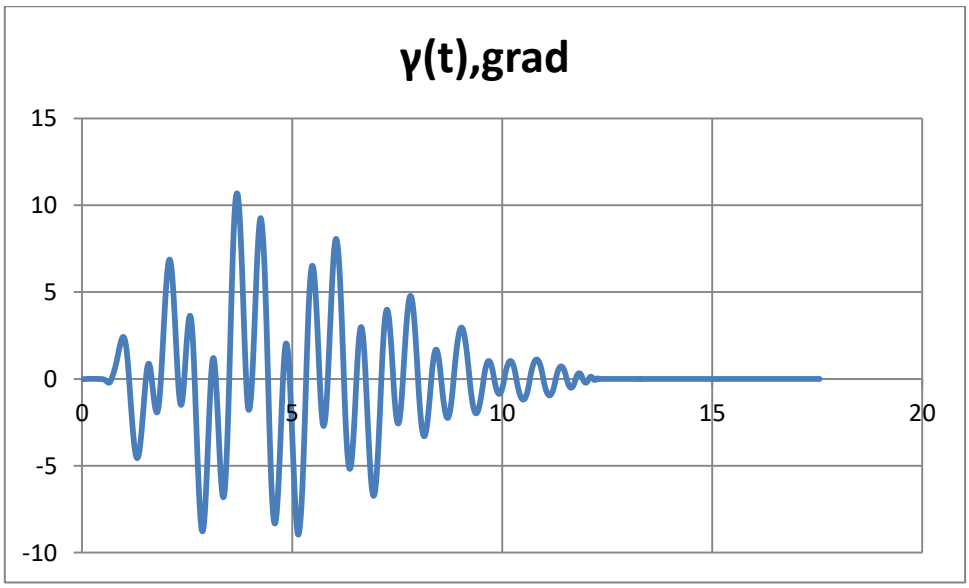

Рисунок А.8 – Зависимость угла крена от времени полета

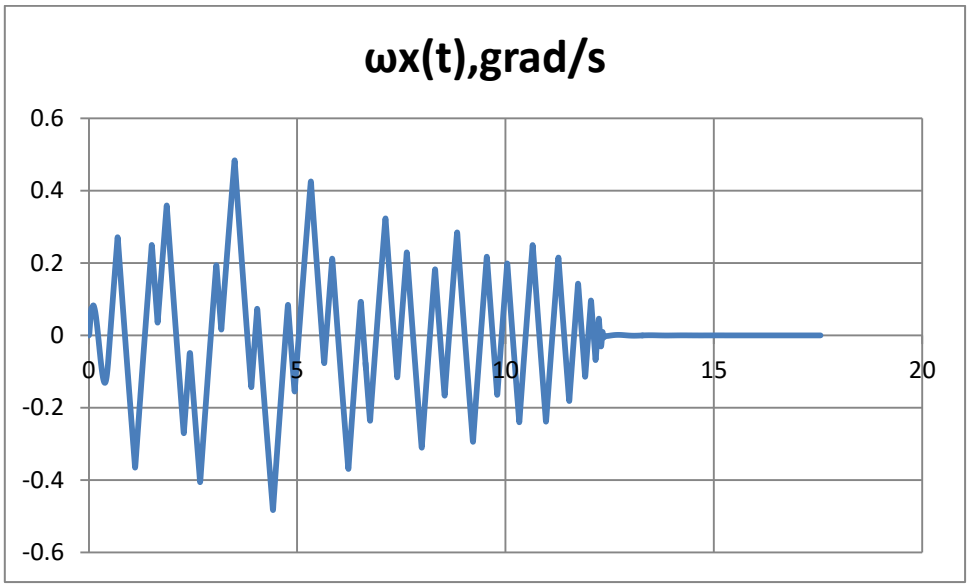

Рисунок А.9 – Зависимость угловой скорости по оси X от времени полета

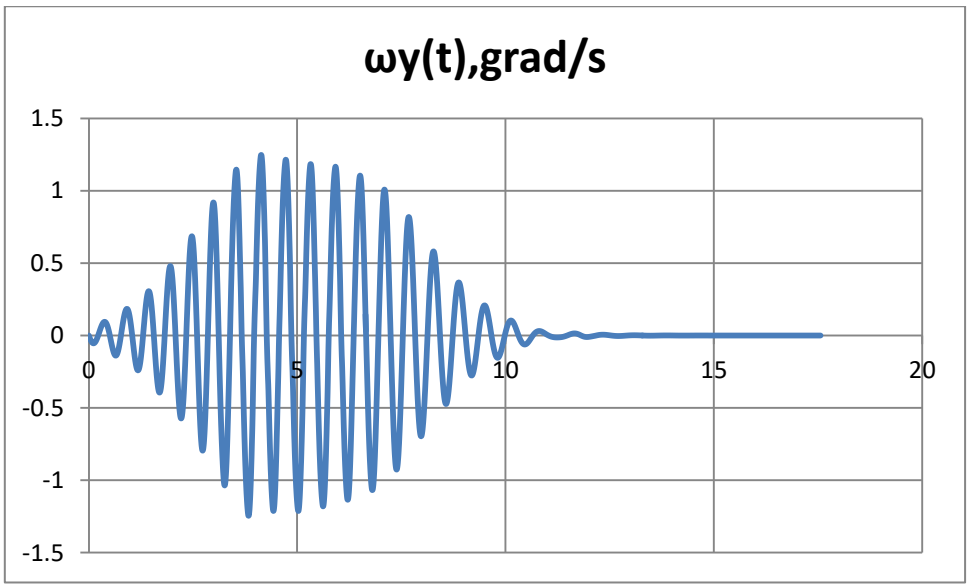

Рисунок А.10 – Зависимость угловой скорости по оси Y от времени полета

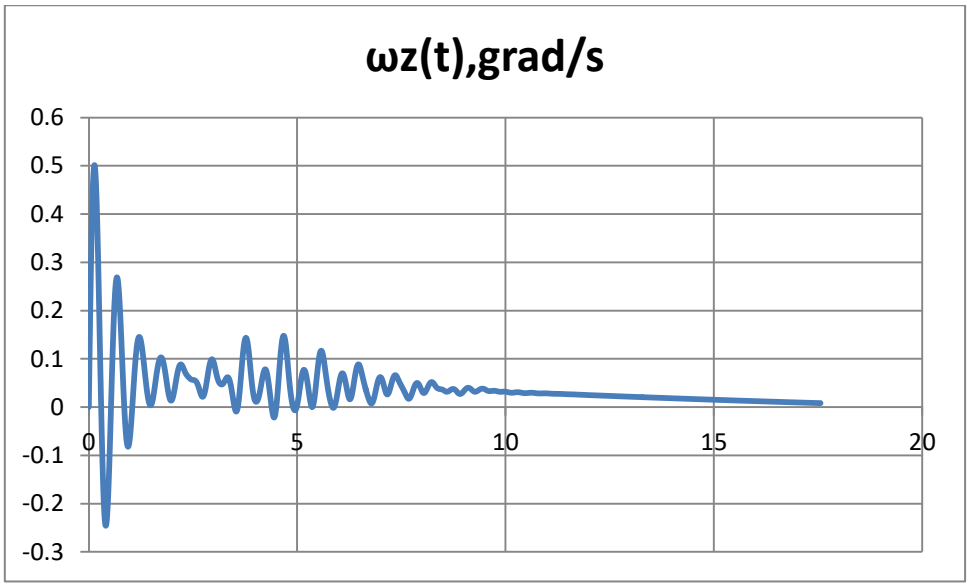

Рисунок А.11 – Зависимость угловой скорости по оси Z от времени полета

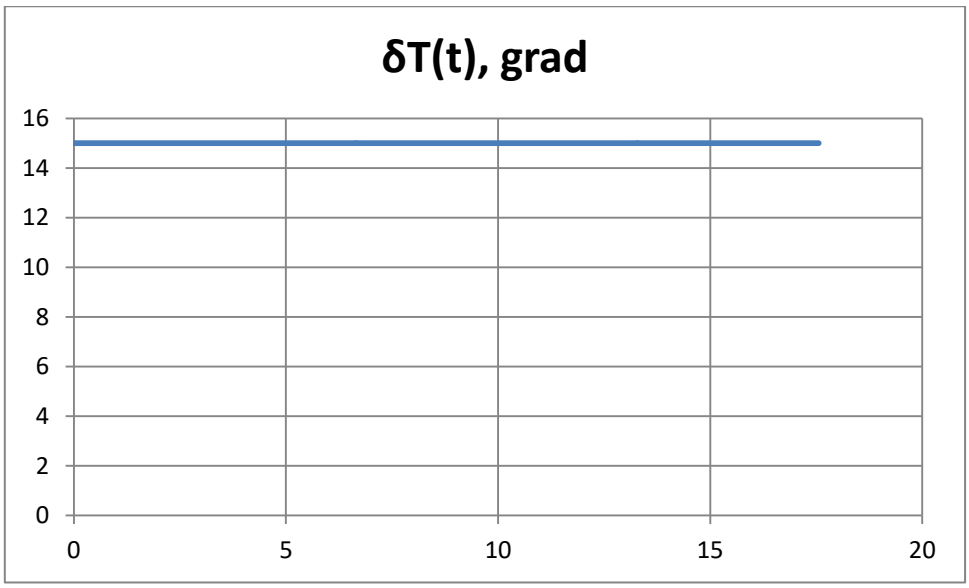

Рисунок А.12 – Зависимость угла отклонения рулей по каналу тангажа от времени полета

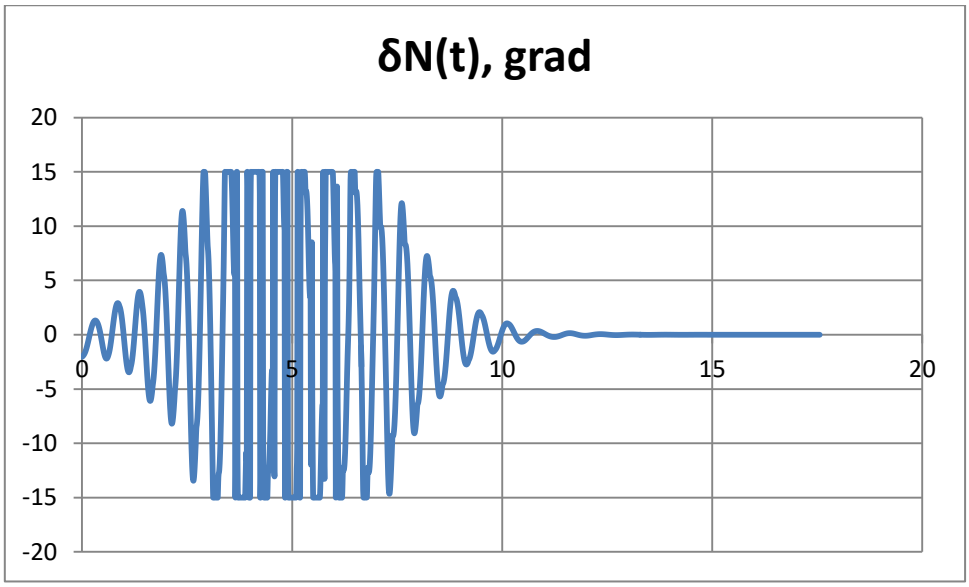

Рисунок А.13 – Зависимость угла отклонения рулей по каналу рысканья от времени полета

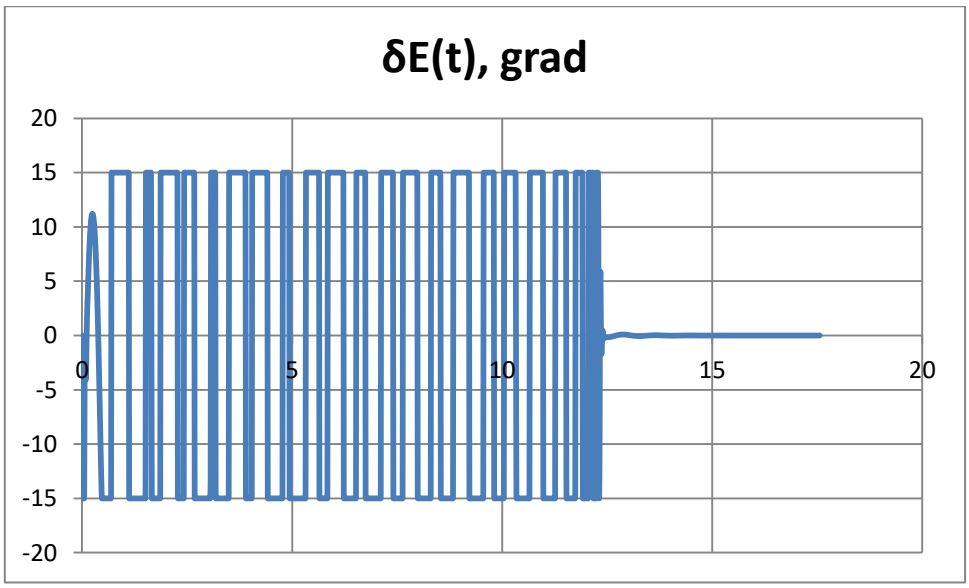

Рисунок А.14 – Зависимость угла отклонения рулей по каналу крена от времени полета

Приложение Б

В таблице Б.1 представлены результаты поиска зоны, в которой цель может быть поражена. Максимальный промах принят за 50 метров. На рисунке Б.1 представлен вид зоны.

Таблица Б.1 – Координаты полученной зоны и промах

$x_ц$, м	$z_ц$, м	r, м
2492.7	0	49.915
2492.7	1888	49.915
2992.7	2276	49.68
3492.7	2541	50.229
3992.7	2765	49.154
4492.7	2974	49.91
4992.7	3215	49.375
5492.7	3505	49.589
5992.7	3836	49.948
6492.7	4224	49.105
6992.7	4681	49.366
7492.7	5220	49.667
7992.7	5861	49.58
8492.7	6642	49.029
8992.7	6510	50.067
9492.7	6103	49.297
9992.7	5596	49.384
10493	4985	49.074
10993	4250	49.381
11493	3270	49.873
11993	1830	49.754
12493	21	50.208
12493	0	50.208

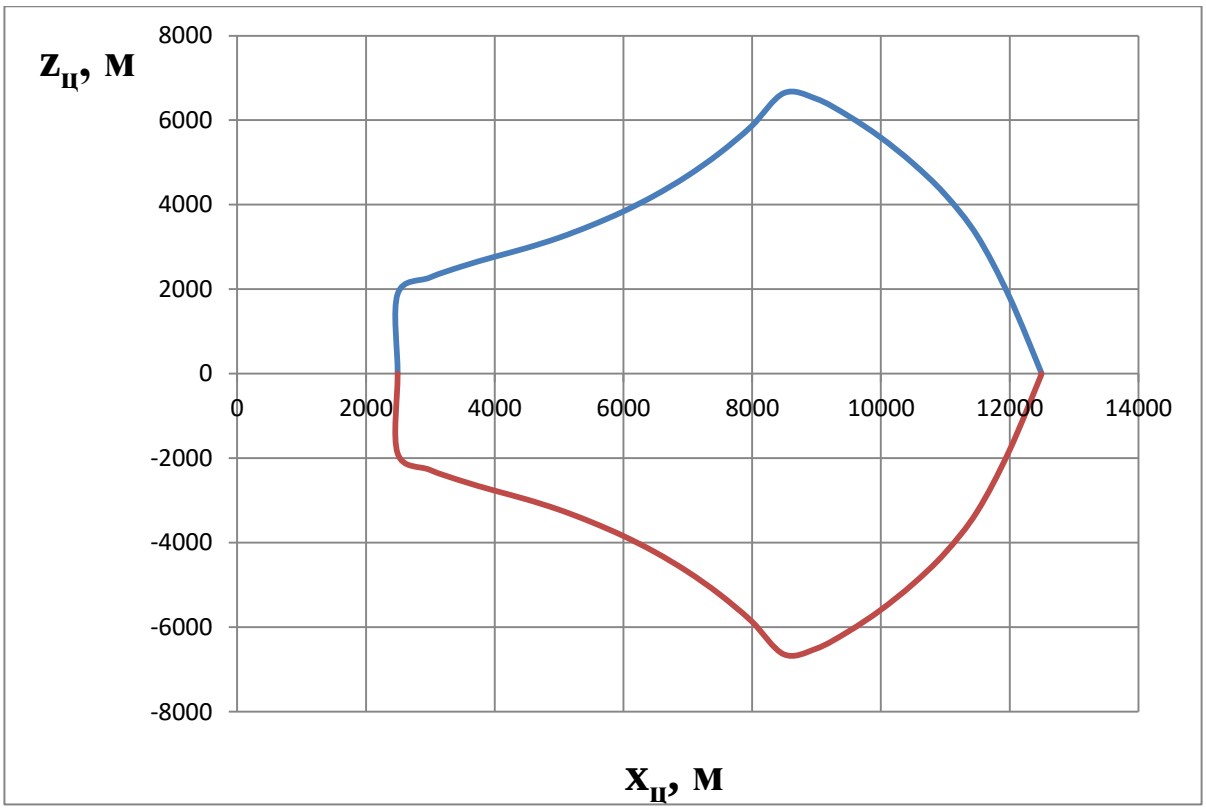

Рисунок Б.1 – Зона поражения

Приложение В

Для выполнения поставленной задачи было разработано программное обеспечение на языке программирования C++ в среде Microsoft Visual Studio 2017. Интегрирование полета осуществляется методом Рунге Кутты 4-го порядка. Листинг модуля для атмосферы опущен.

1. **Объявленные дефайны**:

```
#define dtdt  K[0][j]
#define dxgdt  K[1][j]
#define dygdt  K[2][j]
#define dzgdt  K[3][j]
#define dVxgdt K[4][j]
#define dVygdt K[5][j]
#define dVzgdt K[6][j]
#define domegaxdt K[7][j]
#define domegaydt K[8][j]
#define domegazdt K[9][j]
#define droRGdt K[10][j]
#define dlamdaRGdt K[11][j]
#define dmuRGdt K[12][j]
#define dnuRGdt K[13][j]

#define t  par[0]
#define xg  par[1]
#define yg  par[2]
#define zg  par[3]
#define Vxg par[4]
#define Vyg par[5]
#define Vzg par[6]
#define omegax par[7]
#define omegay par[8]
#define omegaz par[9]
#define roRG par[10]
#define lamdaRG par[11]
#define muRG par[12]
#define nuRG par[13]

#define t_RK  parbuf[0]
#define xg_RK  parbuf[1]
#define yg_RK  parbuf[2]
#define zg_RK  parbuf[3]
#define Vxg_RK parbuf[4]
#define Vyg_RK parbuf[5]
```

```
#define Vzg_RK parbuf[6]
#define omegax_RK parbuf[7]
#define omegay_RK parbuf[8]
#define omegaz_RK parbuf[9]
#define roRG_RK parbuf[10]
#define lamdaRG_RK parbuf[11]
#define muRG_RK parbuf[12]
#define nuRG_RK parbuf[13]
```

2. Вспомогательные функции:

```
//Аэродинамика
double Cx(double M) {
   return pow(73.211 / exp(M) - 47.483 / M + 16.878, -1);
}
double Cy_alpha(double M) {
   double A = 1.86*(11.554 / exp(M) - 2.5191E-3*M*M - 5.024 / M +
52.836E-3*M + 4.112);
   if (A < 0) return 1.86*1.039;
   else return pow(A, 0.5);
}
double Cy_delta(double M, double alpha) {
   alpha = abs(alpha);
   alpha *= 180 / M_PI;
   double p1 = pow(243.84E-3 / exp(-alpha) + 74.309E-3, -1),
         p2 = log10(1.9773*alpha*alpha - 25.587*alpha + 83.354),
         p3 = 18.985*alpha*alpha - 375.76*alpha + 1471,
         p4 = -51.164E-3*alpha*alpha + 805.52E-3*alpha + 1.8929;
   return 2*(-p1*1E-6*M*M + p2*1E-2*exp(M) - p3*1E-6*M - p4*1E-3);
}
double Cz_betta(double M, double betta) {
   return -Cy_alpha(M);
}
double Cz_delta(double M, double betta) {
   return -Cy_delta(M, betta);
}
double mx_omegax() {
   return -0.005*0.6786;
}
double mx_delta(double q) {
   return -1000 / (q);
}
double mz_omegaz(double M) {
   return 1.89*(146.79E-6*M*M - 158.98E-3 / M - 7.6396E-6*M -
68.195E-3);
}
double mz_alpha(double M) {
   return -766.79E-3 / exp(M) + 438.74E-3 / M + 5.8822E-3*M -
158.34E-3    ;
```

```
}
double mz_delta(double M, double alpha) {
   alpha = abs(alpha);
   alpha *= 180 / M_PI;
   double k1 = exp(-19.488E-3*alpha*alpha - 378.62E-3*alpha +
6.7518),
         k2 = exp(-21.234E-3*alpha*alpha - 635.84E-6*exp(alpha) -
98.296E-3*alpha + 2.5938);
   return 1.89*pow(k1*1E-9*M*M + k2*1E-6, 0.5);
}
double my_omegay(double M) {
   return mz_omegaz(M);
}
double my_betta(double M) {
   return mz_alpha(M);
}
double my_delta(double M, double betta) {
   return mz_delta(M, betta);
}

//Матрицы
void B_Matr(double fi, double hi, double(*B)[3][3]) {
   double b[3][3] = { 0 };
   b[0][0] = cos(fi)*cos(hi);
   b[0][1] = sin(fi);
   b[0][2] = -cos(fi)*sin(hi);

   b[1][0] = -sin(fi)*cos(hi);
   b[1][1] = cos(fi);
   b[1][2] = sin(fi)*sin(hi);

   b[2][0] = sin(hi);
   b[2][1] = 0;
   b[2][2] = cos(hi);
   for (int i = 0; i <= 2; i++) for (int j = 0; j <= 2; j++)
        (*B)[i][j] = b[i][j];
}
void MATR(double ro, double lamda, double mu, double nu,
double(*A)[3][3], double(*AT)[3][3]) {
   double a[3][3] = { 0 };
   // Матрица из нормальной земной в связную
   a[0][0] = ro*ro + lamda*lamda - mu*mu - nu*nu;
   a[0][1] = 2 * (-ro*nu + lamda*mu);
   a[0][2] = 2 * (ro*mu + lamda*nu);

   a[1][0] = 2 * (ro*nu + lamda*mu);
   a[1][1] = ro*ro - lamda*lamda + mu*mu - nu*nu;
   a[1][2] = 2 * (-ro*lamda + nu*mu);

   a[2][0] = 2 * (-ro*mu + lamda*nu);
   a[2][1] = 2 * (ro*lamda + nu*mu);
```

```
    a[2][2] = ro*ro - lamda*lamda - mu*mu + nu*nu;

    for (int i = 0; i <= 2; i++) for (int j = 0; j <= 2; j++)
         (*A)[i][j] = a[i][j];
    //Из связной в нормальную
    for (int i = 0; i <= 2; i++) for (int j = 0; j <= 2; j++)
         (*AT)[i][j] = a[j][i];
}

//Стабилизация и наведение
void Control_prog(double tang, double risk, double kren, double
dfidt, double dhidt, double *delta_tang, double *delta_risk, double
*delta_kren) {

    //Для наведения
    const double KSI_SST = 0.35;
    const double KSI_SSN = 0.35;
    const double KSI_SSE = 0.35;
    const double K_SST = 0.95;
    const double K_SSN = 0.95;
    const double T_SSE = 0.01;

    double a11 = -mz_omegaz(M) * q * Sm * L * L / (Iz * V);
    double a12 = -mz_alpha(M) * q * Sm * L / Iz;
    double a13 = -mz_delta(M, alpha) * q * Sm * L / Iz;
    double a42 = Cy_alpha(M) * q * Sm / (m * V);
    double a43 = Cy_delta(M, alpha) * q * Sm / (m * V);
    double vke = Cy_delta(M,alpha);

    double b11 = -my_omegay(M) * q * Sm * L * L / (Iy * V);
    double b12 = -my_betta(M) * q * Sm * L / Iy;
    double b13 = -my_delta(M, betta) * q * Sm * L / Iy;
    double b42 = 0;
    double b43 = -Cy_delta(M, betta) * q * Sm / (m * V);

    double c11 = mx_omegax() * q * Sm * L * L / (Ix * V);
    double c13 = mx_delta(q*Sm*L) * q*Sm*L / Ix;

    double K_T = (a12 * a43 - a13 * a42) / (a12 + a11 * a42);
    double T_1T = -(a13) / (a13 * a42 - a12 * a43);
    double T_T = 1 / sqrt(a12 + a11 * a42);
    double KSI_T = (a11 + a42) / (2 * sqrt(a12 + a11 * a42));

    double K_N = (b12 * b43 - b13 * b42) / (b12 + b11 * b42);
    double T_1N = -(b13) / (b13 * b42 - b12 * b43);
    double T_N = 1 / sqrt(b12 + b11 * b42);
    double KSI_N = (b11 + b42) / (2 * sqrt(b12 + b11 * b42));

    double K_E = c13 / c11;
    double T_E = 1 / c11;
```

```
   double K_fi = 9, K_hi = 10;
   double Kk_t = 6.0, Kk_n = 10;

   double sf = KSI_SST * KSI_SST*KSI_SST*KSI_SST*T_T*T_T;
   double gf = 2 * KSI_T*KSI_SST*KSI_SST*T_T*T_1T;
   double hf = T_1T * T_1T*KSI_SST*KSI_SST;
   double df = sqrt(sf - gf + hf);
   double K_T2 = -2 * T_T*(KSI_T*T_1T - KSI_SST * KSI_SST*T_T - df)
/ (K_T*T_1T*T_1T);
   double K_T1 = K_SST * (1 + K_T2 * K_T) / (K_T*T_1T*T_1T);
   double K_N2 = -2 * T_N * (KSI_N * T_1N - KSI_SSN * KSI_SSN*T_N -
sqrt(KSI_SSN * KSI_SSN * KSI_SSN * KSI_SSN * T_N * T_N - 2 *
KSI_N*KSI_SSN*KSI_SSN*T_N*T_1N + T_1N * T_1N*KSI_SSN*KSI_SSN)) /
(K_N*T_1N*T_1N);
   double K_N1 = K_SSN * (1 + K_N2 * K_N) / (K_N*T_1N*T_1N);

   double K_E1 = (2 * KSI_SSE*T_E - T_SSE) / (K_E*T_SSE);
   double K_E2 = T_E / (K_E*T_SSE*T_SSE);

   double K_T22 = K_T1 * K_fi * Kk_t;
   double K_N22 = K_N1 * K_hi * Kk_n;

   double deltaT = -K_T2 * (omegaz * cos(kren) + omegay * sin(kren))
+ K_T22 * dfidt;
   double deltaN = -K_N2 * (omegay * cos(kren) - omegaz * sin(kren))
/ cos(tang) + K_N22 * dhidt;
   double deltaE = -K_E2 * kren - K_E1 * (omegax - tan(tang)*(omegay
* cos(kren) - omegaz * sin(kren)));

   const double angleDelta = 15*M_PI/180;
   *delta_tang = (abs(deltaT) > angleDelta) ? angleDelta *(deltaT /
abs(deltaT)) : deltaT;
   *delta_risk = (abs(deltaN) > angleDelta) ? angleDelta *(deltaN /
abs(deltaN)) : deltaN;
   *delta_kren = (abs(deltaE) > angleDelta) ? angleDelta *(deltaE /
abs(deltaE)) : deltaE;
}
```

3. Интегрирование полета

```
const double g = 9.81;
double K[99][99], par[99], parbuf[99], par_drob[99],
A[3][3] = { 0 }, AT[3][3] = { 0 }, B[3][3] = { 0 };
double tang, risk, kren, m, V0, q, M, r, V, X, Y, Z, Mx, My, Mz,
Mstab, alpha, betta, alpha_full, vx, vy, vz, vr, vfi, vhi,
delta_tang, delta_risk, delta_kren, fi, hi, dfidt, dhidt, x_t, y_t,
z_t, Vx_t = 0, Vy_t = 0, Vz_t = 0,
Ix = 170,
Iy = 640,
```

```
Iz = 640,
dm = 0.95,
Sm = M_PI*dm*dm / 4.0,
L = 7,
dt = 0.01;

//Начальные условия
double N = 3,
tang = (-60 + 0.5*N)*M_PI / 180.0;
risk = 0;
kren = 0;
m = 1500 - 5 * N;
V0 = 1200;
dt = 0.01;
t = 0;
xg = 0;
yg = 10000 - 150 * N;
zg = 0;
Vxg = V0*cos(tang);
Vyg = V0*sin(tang);
Vzg = 0;
omegax = 0;
omegay = 0;
omegaz = 0;
roRG = cos(risk*0.5)*cos(tang*0.5)*cos(kren*0.5) -
sin(risk*0.5)*sin(kren*0.5)*sin(tang*0.5);
lamdaRG = sin(risk*0.5)*sin(tang*0.5)*cos(kren*0.5) +
cos(risk*0.5)*cos(tang*0.5)*sin(kren*0.5);
muRG = sin(risk*0.5)*cos(tang*0.5)*cos(kren*0.5) +
cos(risk*0.5)*sin(tang*0.5)*sin(kren*0.5);
nuRG = cos(risk*0.5)*sin(tang*0.5)*cos(kren*0.5) -
sin(risk*0.5)*cos(tang*0.5)*sin(kren*0.5);
ofstream txtfile("1.txt");

txtfile << "t,s" << ";" << "x,m" << ";" << "y,m" << ";" << "z,m" <<
";" << "Vx,m/s" << ";" << "Vy,m/s" << ";" << "Vz,m/s" << ";" <<
"V,m/s" << ";"
<< "tang,grad" << ";" << "risk,grad" << ";" << "kren,grad" << ";" <<
"omega_x,grad/s" << ";" << "omega_y,grad/s" << ";" <<
"omega_z,grad/s" << ";"
<< "alpha,grad" << ";" << "betta,grad" << ";" << "deltaT, drad" <<
";" << "deltaN, grag" << ";" << "deltaE, grad" << ";" << endl;
// Интегрирование
int step = 0;
for (;;) {
//cout << t << "  " << r << endl;
// Расчет промежуточных параметров на шаге
MATR(roRG, lamdaRG, muRG, nuRG, &A, &AT);
tang = asin(2 * (roRG*nuRG + lamdaRG*muRG));
risk = atan2(2 * (roRG*muRG - lamdaRG*nuRG), roRG*roRG +
lamdaRG*lamdaRG - nuRG*nuRG - muRG*muRG);
```

```
kren = atan2(2 * (roRG*lamdaRG - muRG*nuRG), roRG*roRG -
lamdaRG*lamdaRG - nuRG*nuRG + muRG*muRG);
r = pow((x_t - xg)*(x_t - xg) + (y_t - yg)*(y_t - yg) + (z_t -
zg)*(z_t - zg), 0.5),
V = pow(Vxg*Vxg + Vyg*Vyg + Vzg*Vzg, 0.5),
q = fro(yg)*V*V*0.5,
vx = Vxg*AT[0][0] + Vyg*AT[0][1] + Vzg*AT[0][2],
vy = Vxg*AT[1][0] + Vyg*AT[1][1] + Vzg*AT[1][2],
vz = Vxg*AT[2][0] + Vyg*AT[2][1] + Vzg*AT[2][2],
alpha = -atan2(vy, vx);
betta = asin(vz / V);
alpha_full = pow(alpha*alpha + betta*betta, 0.5);
fi = asin((y_t - yg) / r);
hi = -atan2(z_t - zg, x_t - xg);
B_Matr(fi, hi, &B);
vr = B[0][0] * (Vx_t - Vxg) + B[0][1] * (Vy_t - Vyg) + B[0][2] *
(Vz_t - Vzg);
vfi = B[1][0] * (Vx_t - Vxg) + B[1][1] * (Vy_t - Vyg) + B[1][2] *
(Vz_t - Vzg);
vhi = B[2][0] * (Vx_t - Vxg) + B[2][1] * (Vy_t - Vyg) + B[2][2] *
(Vz_t - Vzg);
dfidt = vfi / r;
dhidt = vhi / (r*cos(fi));
M = V / fa(yg);
Control_prog(tang, risk, kren, dfidt, dhidt, &delta_tang,
&delta_risk, &delta_kren);
X = -Cx(M)*q*Sm;
Y = (Cy_alpha(M)*alpha + Cy_delta(M, alpha)*delta_tang)*q*Sm;
Z = (Cz_betta(M, betta)*betta + Cz_delta(M, betta)*delta_risk)*q*Sm;
Mstab = 0.5 * dm * 1000 * delta_kren;
Mx = (mx_omegax()*omegax*L / V)*q*Sm*L + Mstab;
My = (my_omegay(M)*omegay*L / V + my_betta(M)*betta + my_delta(M,
betta)*delta_risk)*q*Sm*L;
Mz = (mz_omegaz(M)*omegaz*L / V + mz_alpha(M)*alpha + mz_delta(M,
alpha)*delta_tang)*q*Sm*L;

// Дробление шага
if (yg < 0) {
dt *= 0.5;
for (int i = 0; i <= 13; i++) par[i] = par_drob[i];
}
else {
for (int i = 0; i <= 13; i++) par_drob[i] = par[i];
//вывод в файл
int st = 1 / dt;
if (step % st == 0) txtfile
<< t << ";" << xg << ";" << yg << ";" << zg << ";" << Vxg << ";" <<
Vyg << ";" << Vzg << ";" << V << ";"
<< tang * 180 / M_PI << ";" << risk * 180 / M_PI << ";" << kren *
180 / M_PI << ";"
<< omegax << ";" << omegay << ";" << omegaz << ";"
```

```
<< alpha * 180 / M_PI << ";" << betta * 180 / M_PI << ";"
<< delta_tang * 180 / M_PI << ";" << delta_risk * 180 / M_PI << ";"
<< delta_kren * 180 / M_PI << ";" << r << ";" << endl;
}
if (abs(yg) < 0.0001) break;  //Условие выхода
step++;
//Цикл расчета коэффициентов
for (int j = 1; j <= 4; j++) {
for (int i = 0; i <= 13; i++)  parbuf[i] = par[i] + K[i][j - 1] *
a[j - 1]; // Считаем новые аргументы

// Нормировка параметров РГ
double norm_RG = pow(roRG_RK*roRG_RK + lamdaRG_RK*lamdaRG_RK +
nuRG_RK*nuRG_RK + muRG_RK*muRG_RK, 0.5);
roRG_RK /= norm_RG;
lamdaRG_RK /= norm_RG;
nuRG_RK /= norm_RG;
muRG_RK /= norm_RG;
MATR(roRG_RK, lamdaRG_RK, muRG_RK, nuRG_RK, &A, &AT);

// Расчет промежуточных параметров на шаге
MATR(roRG_RK, lamdaRG_RK, muRG_RK, nuRG_RK, &A, &AT);
tang = asin(2 * (roRG_RK*nuRG_RK + lamdaRG_RK*muRG_RK));
risk = atan2(2 * (roRG_RK*muRG_RK - lamdaRG_RK*nuRG_RK),
roRG_RK*roRG_RK + lamdaRG_RK*lamdaRG_RK - nuRG_RK*nuRG_RK -
muRG_RK*muRG_RK);
kren = atan2(2 * (roRG_RK*lamdaRG_RK - muRG_RK*nuRG_RK),
(roRG_RK*roRG_RK - lamdaRG_RK*lamdaRG_RK - nuRG_RK*nuRG_RK +
muRG_RK*muRG_RK));
r = pow((xg_RK - x_t)*(xg_RK - x_t) + (yg_RK - y_t)*(yg_RK - y_t) +
(zg_RK - z_t)*(zg_RK - z_t), 0.5),
V = pow(Vxg_RK*Vxg_RK + Vyg_RK*Vyg_RK + Vzg_RK*Vzg_RK, 0.5),
q = fro(yg_RK)*V*V*0.5,
vx = Vxg_RK*AT[0][0] + Vyg_RK*AT[0][1] + Vzg_RK*AT[0][2],
vy = Vxg_RK*AT[1][0] + Vyg_RK*AT[1][1] + Vzg_RK*AT[1][2],
vz = Vxg_RK*AT[2][0] + Vyg_RK*AT[2][1] + Vzg_RK*AT[2][2],
alpha = -atan2(vy, vx);
betta = asin(vz / V),
alpha_full = pow(alpha*alpha + betta*betta, 0.5),
fi = asin((y_t - yg_RK) / r),
hi = -atan2(z_t - zg_RK, x_t - xg_RK),
B_Matr(fi, hi, &B),
vr = B[0][0] * (Vx_t - Vxg_RK) + B[0][1] * (Vy_t - Vyg_RK) + B[0][2]
* (Vz_t - Vzg_RK);
vfi = B[1][0] * (Vx_t - Vxg_RK) + B[1][1] * (Vy_t - Vyg_RK) +
B[1][2] * (Vz_t - Vzg_RK);
vhi = B[2][0] * (Vx_t - Vxg_RK) + B[2][1] * (Vy_t - Vyg_RK) +
B[2][2] * (Vz_t - Vzg_RK);
dfidt = vfi / r,
dhidt = vhi / (r*cos(fi)),
M = V / fa(yg_RK),
```

```
Control_prog(tang, risk, kren, dfidt, dhidt, &delta_tang,
&delta_risk, &delta_kren);
X = -Cx(M)*q*Sm,
Y = (Cy_alpha(M)*alpha + Cy_delta(M, alpha)*delta_tang)*q*Sm,
Z = (Cz_betta(M, betta)*betta + Cz_delta(M, betta)*delta_risk)*q*Sm,
Mstab = 0.5*dm * 000 * delta_kren,
Mx = (mx_omegax()*omegax_RK*L / V +
mx_delta(q*Sm*L)*delta_kren)*q*Sm*L + Mstab,
My = (my_omegay(M)*omegay_RK*L / V + my_betta(M)*betta + my_delta(M,
betta)*delta_risk)*q*Sm*L,
Mz = (mz_omegaz(M)*omegaz_RK*L / V + mz_alpha(M)*alpha + mz_delta(M,
alpha)*delta_tang)*q*Sm*L;

//Производные (коэффициенты K[i][j])
dtdt = 1 * dt,
dxgdt = Vxg_RK*dt,
dygdt = Vyg_RK*dt,
dzgdt = Vzg_RK*dt,
dVxgdt = 1 / m*(X*A[0][0] + Y*A[0][1] + Z*A[0][2])*dt,
dVygdt = 1 / m*(X*A[1][0] + Y*A[1][1] + Z*A[1][2] - m*g)*dt,
dVzgdt = 1 / m*(X*A[2][0] + Y*A[2][1] + Z*A[2][2])*dt,
domegaxdt = (Mx / Ix - omegay_RK*omegaz_RK*(Iz - Iy) / Ix)*dt,
domegaydt = (My / Iy - omegax_RK*omegaz_RK*(Ix - Iz) / Iy)*dt,
domegazdt = (Mz / Iz - omegay_RK*omegax_RK*(Iy - Ix) / Iz)*dt,
droRGdt = -(omegax_RK*lamdaRG_RK + omegay_RK*muRG_RK +
omegaz_RK*nuRG_RK)*0.5*dt,
dlamdaRGdt = (omegax_RK*roRG_RK - omegay_RK*nuRG_RK +
omegaz_RK*muRG_RK)*0.5*dt,
dmuRGdt = (omegax_RK*nuRG_RK + omegay_RK*roRG_RK +
omegaz_RK*lamdaRG_RK)*0.5*dt,
dnuRGdt = (-omegax_RK*muRG_RK + omegay_RK*lamdaRG_RK +
omegaz_RK*roRG_RK)*0.5*dt;

}

// Приращения
for (int i = 0; i <= 13; i++) {
double b[5] = { 0.0, 1.0 / 6.0, 2.0 / 6.0, 2.0 / 6.0, 1.0 / 6.0 };
for (int j = 1; j <= 4; j++) par[i] += K[i][j] * b[j];
}
// Нормировка параметров РГ
double norm_RG = pow(roRG*roRG + lamdaRG*lamdaRG + nuRG*nuRG +
muRG*muRG, 0.5);
roRG /= norm_RG;
lamdaRG /= norm_RG;
nuRG /= norm_RG;
muRG /= norm_RG;
}  // Конец интегрирования траектории
// Вывод последнего шага
txtfile
```

```
<< t << ";" << xg << ";" << yg << ";" << zg << ";" << Vxg << ";" <<
Vyg << ";" << Vzg << ";" << V << ";"
<< tang * 180 / M_PI << ";" << risk * 180 / M_PI << ";" << kren *
180 / M_PI << ";"
<< omegax << ";" << omegay << ";" << omegaz << ";"
<< alpha * 180 / M_PI << ";" << betta * 180 / M_PI << ";"
<< delta_tang * 180 / M_PI << ";" << delta_risk * 180 / M_PI << ";"
<< delta_kren * 180 / M_PI << ";" << r << ";" << endl;
```

4. Расчет зоны поражения

```
double dz_t;
   for (x_t = 2492.7; ;x_t+=500) {
        if (x_t > 12611) break;
        dz_t = 1000;
        for (z_t = 1000; z_t <= 50000; z_t += dz_t) {
             y_t = 0;

%ИНТЕГРИРОВАНИЕ ТРАЕКТОРИИ, СМ. П. 3%

//Дробление шага
             if (r > 50) {
                  z_t -= dz_t;
                  dz_t /= 10;
             }

             //Выход из цикла поиска зоны
             if (abs(r-50) < 1) {
                  txtfile2 << x_t << ";" << z_t << ";" << r <<
endl;
                  break;
             }
             cout << "r = " << r << endl
                  << "x_t = " << x_t << endl
                  << "z_t = " << z_t << endl
                  << "dz_t = " << dz_t << endl;
             cout << endl;
        }
   }
```