Table of Contents


```
function [t,boomState,flightParams] = boomerangTrajectory2(S,B)

clear global
```

Controllable Paramaters

```
%Throwing in x-direction
V0 = S.V0; %initial speed magnitude(m/s)
Z0 = S.Z0; %(m)
w0 = S.w0; %(rad/s)

thetaHor0 = S.thetaHor0; %(rad)
thetaLay0 = S.thetaLay0; %(rad)
alpha0 = S.alpha0; %(rad)

dt = S.dt; %(s)
tmax = S.tmax; %(s)
```

Boomerang Parameters

```
m = B.m; %mass (kg)
R = B.R; %length of blade (m)
Roffset = B.Roffset; %Distance from center of lift to axis of symettry
 of blade(m)
C = B.C; %chord of blade (m)
I = B.I; %Inertia Tensor(kgm^2)
A = B.A; %area of boomn (m^2)
```

Environmental Parameters

```
global rho; %density of air (kg/m^3)
rho = 1.225;
global g; %gravitational acceleration (m/s^2)
g = 9.81;
global Vwind;
Vwind = -S.Vw*[cosd(S.WindAng);sind(S.WindAng);0];
```

Inertial Axes

```
global i1 i2 i3;
i1 = [1;0;0];
i2 = [0;1;0];
i3 = [0;0;1];
```

Loop Paramaters

Iterator

```
n=1;
% Time:
t(1) = 0;
```

Initial Conditions

Boomerang State:

```
theta0 = asin(-
sin(thetaHor0)*sin(alpha0)+cos(thetaHor0)*sin(thetaLay0)*cos(alpha0)); %Precession
 angle(rad)
phi0 = acos(cos(thetaLay0)*cos(alpha0)/cos(theta0)); %Layover
 angle(rad)
xi0 =
 asin((sin(phi0)*sin(theta0)*cos(thetaHor0)+cos(theta0)*sin(thetaHor0))/
cos(alpha0)); %Alignment angle(rad)

boomState(:,1) = [0;... % X - Position(m)
                  0;... % Y - Position(m)
                  Z0;... % Z - Height(m)
                  V0*cos(thetaHor0);... % Vx - Velocity in X-dir(m/s)
                  0;... % Vy - Velocity in Y-dir(m/s)
                  V0*sin(thetaHor0);... % Vz - Velocity in Z-dir(m/s)
                  phi0; % phi - euler-angle(rad)
                  theta0;... % theta - euler-angle(rad)
                  xi0;... % xi - euler-angle(rad)
                  0;... % w1 - Rotation Rate about boom 1-axis(rad/s)
                  0;... % w2 - Rotation Rate about boom 2-axis(rad/s)
                  w0]; % w3 - Rotation Rate about boom 3-axis(rad/s)
```

Initial Forces, Moments, AOA, and Boomerang Non-Dimensional Coefficient:

```
[F,M,L,D,Mx,My,Mz,alpha,Xi] = FandM(boomState,B);
```

Parameters to Return:

```
global flightParams
```

```
flightParams.F(:,1) = F;
flightParams.M(:,1) = M;
flightParams.L(:,1) = L;
flightParams.D(:,1) = D;
flightParams.Mx(1) = Mx;
flightParams.My(1) = My;
flightParams.Mz(1) = Mz;
flightParams.alpha(1) = alpha;
flightParams.Xi(1) = Xi;
```

Loop

```
while t(n)<tmax && boomState(3,n)>=0
    % Iterator
    n = n+1;
    % Time
    t(n) = t(n-1)+dt;
    % Update State:
    [boomState(:,n)] = RK4(@boomN,boomState(:,n-1),@FandM,B,dt);
end

end

function [F,M,L,D,Mx,My,Mz,alpha,Xi] = FandM(boomState,B)

%Initialize Global Variables:
global rho g i1 i2 i3 Vwind;
% Velocities:
V = sqrt(sum(boomState(4:6).^2)); %Speed of Boomerang
Vu = boomState(4:6)/V; % Velocity Unit Vector
Vw = sqrt(sum((boomState(4:6)-Vwind).^2));
Vwu = (boomState(4:6)-Vwind)/Vw;
% Non-dimensional rotational velocity
Xi = boomState(12)*B.R/Vw;
% Body axes
b1 = Tbi(boomState(7),boomState(8),boomState(9))*i1; %b1 expressed in
 inertial frame coordinates
b2 = Tbi(boomState(7),boomState(8),boomState(9))*i2; %b2 expressed in
 inertial frame coordinates
b3 = Tbi(boomState(7),boomState(8),boomState(9))*i3; %b3 expressed in
 inertial frame coordinates
% Angle of Attack
alpha = asin(dot(-b3,Vwu));
% Moments and Forces
[CL, CD, CMX, CMY, CMZ] = boomerangCoefs(Xi,alpha,B);
%CD = 0; CMZ = 0;
%CD = 0; CMZ = 0; CL = 0; CMX = 0; CMY = 0;
L = .5*rho*Vw^2*B.A*CL;
D = 2*.5*rho*Vw^2*B.A*CD;
Mx = .5*rho*Vw^2*B.A*B.R*CMX;
My = .5*rho*Vw^2*B.A*B.Roffset*CMY;
Mz = 2*.5*rho*Vw^2*B.A*B.R*CMZ;
% Inertial Forces
```

```
LI = L*cross(cross(Vwu,b3),Vu)/norm(cross(cross(Vwu,b3),Vu));
DI = D*-Vwu;
GI = i3*B.m*-g;
F = LI+DI+GI;
% Apply Moments to rotational accelerations
% M1 = -Mx*dot(b1,(cross(b3,cross(Vwu,b3)))/
norm(cross(b3,cross(Vwu,b3))))+My*dot(b1,cross(Vwu,b3)/
norm(cross(Vwu,b3)));
% M2 = -Mx*dot(b2,cross(b3,cross(Vwu,b3))/
norm(cross(b3,cross(Vwu,b3))))+My*dot(b2,cross(Vwu,b3)/
norm(cross(Vwu,b3)));
% M3 = -Mz;
% M = [M1;M2;M3];
M = Tib(boomState(7),boomState(8),boomState(9))*(-
Mx*cross(b3,cross(Vwu,b3))/
norm(cross(b3,cross(Vwu,b3)))+My*cross(Vwu,b3)/norm(cross(Vwu,b3))-
Mz*b3);
%M = (-Mx*cross(i3,cross(Vwu,i3))/
norm(cross(i3,cross(Vwu,i3)))+My*cross(Vwu,i3)/norm(cross(Vwu,i3))-
Mz*i3)
end

function [N,F,M,L,D,Mx,My,Mz,alpha,Xi] = boomN(boomState,FandM,B)

[F,M,L,D,Mx,My,Mz,alpha,Xi] = FandM(boomState,B);

N = [boomState(4);...
    boomState(5);...
    boomState(6);...
    F/B.m;...%
+cross(boomState(4:6),Tib(boomState(7),boomState(8),boomState(9))*boomState(10:12)

 Twe(boomState(7),boomState(8),boomState(9))*[boomState(10);boomState(11);boomStat
    (M(1)+(B.I(2,2)-B.I(3,3))*boomState(11)*boomState(12))/
B.I(1,1);...
    (M(2)+(B.I(3,3)-B.I(1,1))*boomState(10)*boomState(12))/
B.I(2,2);...
    (M(3)+(B.I(1,1)-B.I(2,2))*boomState(10)*boomState(11))/B.I(3,3)];

end

function [nextState] = RK4(computeN,currentState,FandM,B,dt)

[f1,~,~,~,~,~,~,~,~,~] = computeN(currentState,FandM,B);
[f2,~,~,~,~,~,~,~,~,~] = computeN(currentState+dt*f1/2,FandM,B);
[f3,~,~,~,~,~,~,~,~,~] = computeN(currentState+dt*f2/2,FandM,B);
[f4,F,M,L,D,Mx,My,Mz,alpha,Xi] = computeN(currentState+dt*f3,FandM,B);

nextState = currentState+dt*(f1/6+(f2+f3)/3+f4/6);

global flightParams
flightParams.F(:,end+1) = F;
flightParams.M(:,end+1) = M;
flightParams.L(:,end+1) = L;
```

```
flightParams.D(:,end+1) = D;
flightParams.Mx(end+1) = Mx;
flightParams.My(end+1) = My;
flightParams.Mz(end+1) = Mz;
flightParams.alpha(end+1) = alpha;
flightParams.Xi(end+1) = Xi;

end

function T = Tbi(phi,theta,xi)

T = [cos(phi)*cos(xi)+sin(phi)*sin(theta)*sin(xi), cos(phi)*sin(xi)-
sin(phi)*sin(theta)*cos(xi), -sin(phi)*cos(theta);
sin(phi)*cos(xi)-cos(phi)*sin(theta)*sin(xi),
 sin(phi)*sin(xi)+cos(phi)*sin(theta)*cos(xi), cos(theta)*cos(phi);
    cos(theta)*sin(xi), -cos(theta)*cos(xi), sin(theta)];

end

function T = Tib(phi,theta,xi)

T = Tbi(phi,theta,xi)';

end

function T = Twe(phi,theta,xi)

T = [sin(xi)/cos(theta),-cos(xi)/cos(theta),0;cos(xi),sin(xi),0;-
tan(theta)*sin(xi),tan(theta)*cos(xi),-1];

end
```

Published with MATLAB® R2020b