Journal of Nonlinear Dynamics

MATLAB Code

for

High Performance Computing of the Nonlinear Dynamics of a Basketball

by

Larry M. Silverberg and Chau M. Tran

Mechanical and Aerospace Engineering, North Carolina State University

Raleigh, NC 27695-7910, USA, lmsilver@ncsu.edu

```
clc, clear, close all
% enter the parameters
RB = 29.75/12/2/pi;        % men ball radius
RH = (9 + 5/16)/12;        % hoop radius, assume it is a middle radius
A0 = 15/12;                % distance between hoop center and backboard
yBB = 36/12;               % right coordinate of backboard
zBB1 = -6/12;              % bottom coordinate of backboard
zBB2 = 36/12;              % top coordinate of backboard
yBR = 3/12;                % right coordinate of bridge
RC = 2.5/8/12;             % cross-section radius of ring
e = 0.7640;                % men coefficient of restitution
g = 32.2;                  % acceleration due to gravity
H = 0.2*RB;                % distance to determine the shot is successful
% general initialization
R = RB+RC;                 % ball + ring radius
n3 = zeros(3,1);           % normal unit vector
n1 = zeros(3,1);           % tangential unit vector in direction of max(n3)
n2 = zeros(3,1);           % tangential unit vector = n3 x n1
casehit = 0;               % 1 = success plane, 2 = backboard, 3 = ring
                           % 4 = bridge
CASE = [];                 % vector of casehits
COND_HIT = 0;              % default success condition (miss)
ncol = 40;                 % maximum number of collisions
tmax = 2.0;                % maximum time in the air
dtmax = 0.01;              % maximum step size
nstep = tmax/dtmax;
% initial conditions
x1 = 8.00; vx1 = -9.5; y1 = 0; vy1 = 0; z1 = -2; vz1 = 16.00;
omeg = 0; omegx1 = 0; omegy1 = omeg; omegz1 = 0;
ts = 0;
xs = x1; vxs = vx1; ys = y1; vys = vy1; zs = z1; vzs = vz1;
omegxs = omegx1; omegys = omegy1; omegzs = omegz1;
% calculate collision times of candidate events
for k = 1:ncol
    t_k = linspace(0.00001,tmax,nstep); % generate vector t from 0 to tmax.
% end-result plane contact (casehit = 1)
```

```
plane = @(t_k) z1 + vz1*t_k - g*t_k.^2/2 + H;
f1 = plane(t_k);
for i1=1:size(t_k,2)-1
    ratio_1 = f1(i1+1)/f1(i1);
    vz = vz1 - g*t_k(i1+1);
    if ratio_1<0 && vz<0
        int_1 = [t_k(i1),t_k(i1+1)];
        t1 = fzero(plane, int_1);
        break
    else
        t1 = t_k(i1+1);
    end
end
% backboard contact (casehit = 2)
board = @(t_k) x1 + vx1*t_k + A0 - RB;
f2 = board(t_k);
for i2=1:size(t_k,2)-1
    ratio_2 = f2(i2+1)/f2(i2);
    y = y1 + vy1*t_k(i2+1);
    z = z1 + vz1*t_k(i2+1) - g*t_k(i2+1)^2/2;
    if ratio_2<0 && abs(y)<=yBB && z<=zBB2 && z>=zBB1
        int_2 = [t_k(i2),t_k(i2+1)];
        t2 = fzero(board,int_2);
        break
    else
        t2 = t_k(i2+1);
    end
end
% ring contact (casehit = 3)
ring = @(t_k) (x1 + vx1*t_k).^2 + (y1 + vy1*t_k).^2 + ...
            (z1 + vz1*t_k - g*t_k.^2/2).^2 + RH^2 - R^2 - ...
            2*RH*sqrt((x1 + vx1*t_k).^2 + (y1 + vy1*t_k).^2);
f3 = ring(t_k);
for i3=1:size(t_k,2)-1
    ratio_3 = f3(i3+1)/f3(i3);
    if ratio_3<0
        int_3 = [t_k(i3),t_k(i3+1)];
        t3 = fzero(ring,int_3);
        break
    else
        t3 = t_k(i3+1);
    end
end
% bridge contact (casehit = 4)
bridge = @(t_k) z1 + vz1*t_k - g*t_k.^2/2 - R;
f4 = bridge(t_k);
for i4=1:size(t_k,2)-1
    ratio_4 = f4(i4+1)/f4(i4);
    x = x1 + vx1*t_k(i4+1);
    y = y1 + vy1*t_k(i4+1);
    if ratio_4<0 && abs(y)<=yBR && x<=0 && x>=-A0 ...
                                        && (x^2+y^2)>=(RH+RC)^2
        int_4 = [t_k(i4),t_k(i4+1)];
        t4 = fzero(bridge,int_4);
        break
    else
        t4 = t_k(i4+1);
```

```
    end
end
% determine true contact time and the case where the ball hits
[t,casehit] = min([t1,t2,t3,t4]);
if casehit == 1, where = ' through '; end
if casehit == 2, where = ' board '; end
if casehit == 3, where = ' rim '; end
if casehit == 4, where = ' bridge '; end
CASE = [CASE, where];
% find final conditions
x2 = x1 + vx1*t; y2 = y1 + vy1*t; z2 = z1 + vz1*t-g*t^2/2;
vx2 = vx1; vy2 = vy1; vz2 = vz1-g*t;
omegx2 = omegx1; omegy2 = omegy1; omegz2 = omegz1;
out(k,:) = [k, casehit, x2,vx2,y2, vy2, z2,vz2,t];
% the success of a shot
if casehit == 1 && sqrt((x2^2+y2^2)) < RH && vz2 < 0
    COND_HIT = 1 % Record the success
end
% update velocity after contact with the end-result plane
if casehit == 1 && vz2 < 0
    x1 = x2; y1 = y2; z1 = z2;
    ts = [ts,t]; xs = [xs, x1]; ys = [ys, y1]; zs = [zs, z1];
    vxs = [vxs, vx1]; vys = [vys, vy1]; vzs = [vzs, vz1];
    omegxs = [omegxs, omegx1];
    omegys = [omegys, omegy1];omegzs = [omegzs, omegz1];
    break % break out of collision loop
end
% update velocity after contact with the other planes
% 1. construct surface normal vectors
if casehit == 3 % ring normal
    xc = x2/sqrt(x2^2+y2^2)*RH; yc = y2/sqrt(x2^2+y2^2)*RH;
    mag = sqrt((x2-xc)^2+(y2-yc)^2+z2^2);
    n3(1) = (x2-xc)/mag; n3(2) = (y2-yc)/mag; n3(3) = z2/mag;
end
if casehit == 2  % backboard normal
    n3(1) = 1; n3(2) = 0; n3(3) = 0;
end
if casehit == 4 % bridge normal
    n3(1) = 0; n3(2) = 0; n3(3) = 1;
end
% 2. construct tangent vectors
[temp,index] = min(abs(n3)); n1(index) = 1;
n1 = n1 - dot(n1,n3)*n3; n1 = n1/sqrt(dot(n1,n1)); n2 = cross(n3,n1);
ROT = [n1,n2,n3];
% 3. rotate initial conditions
v123 = ROT'*[vx2,vy2,vz2]'; omeg123 = ROT'*[omegx2,omegy2,omegz2]';
% 4. determine velocity components after contact
b = [0,0,-e*v123(3),-v123(2)+2*RB/3*omeg123(1),...
v123(1)+2*RB/3*omeg123(2),omeg123(3)]';
AMATRIX = [1,0,0,0,-RB,0;0,1,0,RB,0,0;0,0,1,0,0,0;...
       0,-1,0,2*RB/3,0,0;1,0,0,0,2*RB/3,0;0,0,0,0,0,1];
vo123 = AMATRIX\b;
v123 = vo123(1:3); omeg123 = vo123(4:6);
vxyz = ROT*v123; omegxyz = ROT*omeg123;
x1 = x2; y1 = y2; z1 = z2;
vx1 = vxyz(1); vy1 = vxyz(2); vz1 = vxyz(3);
omegx1 = omegxyz(1); omegy1 = omegxyz(2); omegz1 = omegxyz(3);
```

```
    ts = [ts,t]; xs = [xs, x1]; ys = [ys, y1]; zs = [zs, z1];
    vxs = [vxs, vx1]; vys = [vys, vy1]; vzs = [vzs, vz1];
    omegxs = [omegxs, omegx1]; omegys = [omegys, omegy1];
    omegzs = [omegzs, omegz1];
end
exactballpath(xs,ys,zs,vxs,vys,vzs,ts) % plot trajectory
CASE
function exactballpath(xs,ys,zs,vxs,vys,vzs,ts) % plotting function
RB = 29.75/12/2/pi; RH = (9 + 5/16)/12; a = 15/12; zbb1 = -0.5; zbb2 = 2.5;
yB = -3; g = 32.2;
phi = 47;                  % azimuth angle of the view (degrees)
theta = 24;                % elevation angle of the view (degrees)
figure, hold, axis equal, view([phi,theta]), axis off
% backboard
lnw = 1;
plot3([-a,-a],[-yB,yB], [zbb1,zbb1],'k','linewidth', lnw)
plot3([-a,-a],[-yB,yB], [zbb2,zbb2],'k','linewidth', lnw)
plot3([-a,-a],[-yB,-yB],[zbb1,zbb2],'k','linewidth', lnw)
plot3([-a,-a],[yB,yB],  [zbb1,zbb2],'k','linewidth', lnw)
% backboard rectangle
plot3([-a,-a],[-1,1], [0,0],'k','linewidth', lnw)
plot3([-a,-a],[-1,1], [1.5,1.5],'k','linewidth', lnw)
plot3([-a,-a],[-1,-1],[0,1.5],'k','linewidth', lnw)
plot3([-a,-a],[1,1],  [0,1.5],'k','linewidth', lnw)
% rim
xr = zeros(361); yr = zeros(361); zr = zeros(361);
    for i = 1:361; xr(i) = RH*cosd(i); yr(i) = RH*sind(i); zr(i) = 0; end
plot3(xr,yr,zr,'r','linewidth', lnw)
% bridge
plot3([-a,-sqrt(RH^2-0.25^2)],[0.25,0.25],  [0,0],'r','linewidth', lnw)
plot3([-a,-sqrt(RH^2-0.25^2)],[-0.25,-0.25],[0,0],'r','linewidth', lnw)
plot3([-a,-a],[-0.25,0.25],[0,0],'r','linewidth', lnw)
% plot trajectory
n = size(ts,2);
x = zeros(101); y = zeros(101); z = zeros(101);
    for i = 1:n-1
        xi = xs(i); yi = ys(i); zi = zs(i);
        vxi = vxs(i); vyi = vys(i); vzi = vzs(i); tf = ts(i+1);
        dt = tf/100;
        for j = 1:101
            t = (j-1)*dt;
            x(j) = xi + vxi*t; y(j) = yi + vyi*t;
            z(j) = zi + vzi*t-g*t^2/2;
        end
        plot3(x,y,z,'b','linewidth', 1.5)
    end
% plot ball at beginning of the event
    for ii = 1:size(ts,2)
        [X,Y,Z]=sphere;
        X2=X*RB;Y2=Y*RB;Z2=Z*RB;
        surf(X2+xs(ii),Y2+ys(ii),Z2+zs(ii), ...
            'FaceColor',[.8500 .3250 .0980],'FaceAlpha',0.4, ...
            'FaceLighting','gouraud','EdgeColor','none')
        camlight
    end
return
end
```