Varied Ramps and Rigid Wheels

Evan Wang (email adress)
Lucas Rodrigues (lr2894@nyu.edu)

November 2021

1 Introduction

We examine the performance of multiple ramps against a projectile rigid body wheel by rolling the wheel off the top of a flat platform and examining the distance it was launched. To standardize the problem, we include a similarly sized incline corresponding to each function and a standardized runway leading up to the ramp. We cut off the ramp at optimized points corresponding to its best-performing launch with an optimization algorithm.

We addressed the problem through a classic rigid body implementation - locking all wheel points together - and with each portion of the ramp exerting force corresponding to its function. The simulation is done with a simple Forward Euler numerical method to approximate the position of the rigid body in each time step. We run multiple of these simulations for each ramp to find their optimal cutoff points and finally compare their performances in launching the wheel the farthest. Both of these implementations are based on Dr. Charles Peskin's lectures on MODELING AND SIMULATION IN SCIENCE, ENGINEERING, AND ECONOMICS, as well as Ondrej Maxian's recitations on the topic.

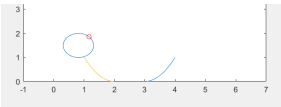

Figure 1: a ramp based on a parabola

As you can see in figure 1, the ramp decreases and grows near 0, modeling the behavior of a parabola. It then becomes flat and transitions into a rising parabola.

We made a ramp based on a semi-circle to accompany the parabolic ramp in figure 1, as both are curved ramps without any non-differentiable points, thus encouraging "rolling" .

All three ramps and the wheel undergo the same conditions, as each ramp is only 1 unit tall and only descends to the x-axis to provide the wheel, in each case, with the same amount of potential energy. The wheel is at the mouth of the ramp and given the same amount of velocity in each case.

To measure the performance of these ramps, a rigid and 3-dimensional infinitesimally thin wheel was used as a launching projectile. We use a numerical method to estimate each ramp's optimum cutoff. This cutoff is determined.

First, we fixed the wheel as a rigid body and used Forward Euler's numerical approximation method to simulate the performance of each ramp without a cutoff.

2 Equations

Our modeling involves a rigid circle governed by the classic rigid body model. We generate the circle by choosing various angle thetas and setting coordinate

$$(x_i, y_i) = (r * \cos(\theta_i), r * \sin(\theta_i))$$

We then find the center of mass to be the mean of all points in the rigid body, setting

$$x_c m = (\frac{\sum_i^k x_i}{k}, \frac{\sum_i^k y_i}{k})$$

The total force is:

$$M\frac{d\underline{U}_{cm}}{dt} = \underline{F} = \sum_{k=1}^{n} \underline{F}_k$$

The velocity of the center of mass is:

$$\frac{d\underline{X}_{cm}}{dt} = \underline{U}_{cm}$$

Angular momentum:

$$\underline{L} = \sum_{k=1}^{n} M_k(\underline{X}_k - \underline{X}_c m) \times (\underline{U}_k - \underline{U}_{cm})$$

Which has derivative:

$$\frac{d\underline{L}}{dt} = \underline{\tau} = \sum_{k=1}^{n} (\underline{X}_k - \underline{X}_{cm}) \times \underline{F}_k$$

All together, these equations govern the motion of the system due to its rigid nature. We can determine the velocities of all the points by the velocity of the center of mass $\underline{U}_{cm}(t)$ and by the angular velocity $\underline{\Omega}(t)$ of the entire body about its center of mass. The individual point velocities can be given by:

$$\underline{U}_k(t) = \underline{U}_{cm}(t) + \underline{\Omega}(t) \times (\underline{X}_k(t) - \underline{X}_{cm}(t))$$

Which finally gives the implemented formula:

$$\underline{L}(t) = \sum_{k=1}^{n} M_k(\underline{X}_k(t) - \underline{X}_{cm}(t)) \times (\Omega(t) \times (\underline{X}_k(t) - \underline{X}_{cm}(t))$$

We can rewrite $\underline{L}(t)$ as

$$\underline{L}(t) = I(t)\underline{\Omega}(t)$$

With component $I(t)$ being the momentum of inertia tensor

$$I(t) = \sum_{k=1}^{n} M_k(||\underline{\tilde{X}}_k(t)||^2 E - \underline{\tilde{X}}_k(t)(\underline{\tilde{X}}_k(t))^T)$$

Where $\underline{\tilde{X}}_k(t) = \underline{X}_k(t) - \underline{X}_{cm}(t)$

The rotation equation implemented was:

$$R(\underline{\Omega}, \Delta t)\underline{X} = P(\underline{\Omega}) + \cos(||\underline{\Omega}||\Delta t)(E - P(\underline{\Omega})) + \sin(||\underline{\Omega}||\Delta t)(\frac{\underline{\Omega}}{||\underline{\Omega}||}\times)$$

3 Numerical Method

We obtain the new $\underline{\tilde{X}}$ by rotating with

$$\underline{\tilde{X}}_k(t + \Delta t) = R(\underline{\Omega})(t), \Delta t)\underline{\tilde{X}}_k(t)$$

We can find the new velocity and angular momentum by

$$M\frac{\underline{U}_{cm}(t + \Delta t) - \underline{U}_{cm}(t)}{\Delta t} = \sum_{k=1}^{n} \underline{F}_k(t + \Delta t)$$

$$\frac{\underline{L}(t + \delta t) - \underline{L}(t)}{\Delta t} = \sum_{k=1}^{n} \underline{\tilde{X}}_k(t + \Delta t) \times \underline{F}_k(t + \Delta t)$$

And thus the new center of mass can be updated by

$$\underline{X}_{cm}(t + \Delta t) = \underline{X}_{cm}(t) + (\Delta t)\underline{U}_{cm}(t)$$

4 Results and Discussion

We found that the Parabolic Ramp had an optimal cut-off point at 3.300 units, as seen in See 2.. This led to a launch distance of 4.0263 units in length.

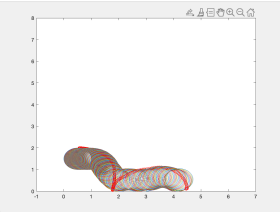

Figure 2: a ramp based on a parabola

The Semi-circle Ramp, on the other hand, had an optimal cut-off point at 3.700 units, leading to a launch distance of 4.9870 units and falling short of its counterpart. See 3.

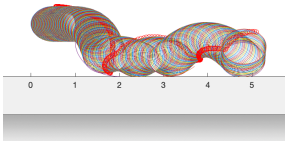

Figure 3: a ramp based on a semi-circle

After optimizing the cutoff value for the semi-circle-based ramp, we obtained the following trajectory following the points on the rigid wheel. As you can see, the cut off-ramp allows for the wheel to travel as further as possible, reaching the resulting value of 4.987

5 Summary and Conclusions

We adopted much of Ondrej's code from his third Recitation with us, in which we simulated a rolling wheel through applications of rigid bodies and the Rodrigues' Rotation Formula.

While we re-created many of the initial conditions used in his third recitation program and re-used one of his "GroundForce" functions, we modified the latter to accommodate better our goal of finding the optimal version of each ramp. By expanding on the concept of a flat road introduced in both Professor Peskin's lectures and Ondrej's recitations, we adopted a new function focused on a "piece-wise" function representing the ramp. Our numerical optimization method was centered around the gradual extension of the ramp from its' middling flat-section. We kept cutting off the ramp and yielding varying cut-off sections until the wheel could no longer leave the ramp. We ran into some problems with certain initial conditions making it hard to form a coherent project, which we overcame by restricting the optimization algorithm only to explore values that made sense in the context of the function. We conclude that the Semi circle modeled ramp was better optimized for projectile motion. Although the wheel landed on x=4.987, a larger value than the parabola ramp's 4.0263, its cut-off point was at x=3.7, for a total travel distance of 1.287. Meanwhile, the parabola ramp had a travel distance of merely around 0.7.

6 References

Charles Peskin's lectures: https://www.math.nyu.edu/ peskin/modsim$_l ecture_n otes/index.html$

Specifically, we extracted the main formulas governing the system of rigid bodies from them, as well as proper derivations and the numerical method idea all from the Rigid Bodies lecture.

Ondrej Maxian's recitation notes: https://cims.nyu.edu/ om759/ModelingClass/index.html

In particular we modified Recitation 3 material that already implemented a simple rigid body wheel moving, and combined with previous ideas mentioned in Recitation 2 where we simulate a point bouncing on the ground.

7 Computer Code

7.1 Semi Circle Ramp Optimization

```
% Simulating a wheel rolling on the ground
M = 5;
N = 100;
r=0.5;
g = 9.8;
S = 10000;
mu = 0.5;
Tf = 10;
thet = (0:N-1)'*2*pi/N;

%declare variables that will hold optimal values
optCutoff=0;
optVal=0;
cutoff=2;
target = zeros(1,100);
%optimize
for(in=2:50)
    %build wheel with rotational forces
    Omega = -[0 0 1];
    x = [r*sin(thet) r*cos(thet) zeros(N,1)]+[0 r+1 0];
    x_cm = mean(x);
    x_tilde = x-x_cm;
    u_cm = [0.5 0 0];
    I = zeros(3);

    for k=1:N
        I = I+M/N*(norm(x_tilde(k,:))^2*eye(3)-x_tilde(k,:)'*x_tilde(k,:));
    end
    % I does not change for a circle
    L = I*Omega';
    dt = 1e-2;
    nSteps=Tf/dt;
    xcms = zeros(nSteps+1,3);
    xcms(1,:)=x_cm;
    Omegas = zeros(nSteps,1);

    stop=0;



    for iStep=1:nSteps
        if (stop==0)
```

```
            % Numerical method
            Omega = (I \ L)';
            Omegas(iStep)=Omega(3);
            x = x_tilde+x_cm;
            u = cross(Omega.*ones(N,3),x_tilde)+u_cm;
            F = zeros(N,3);
            for iPt=1:N
                [stop, target(in),F(iPt,:)] = GroundForce3D(x(iPt,:),u(iPt,:),mu,S,M/N*g,cut
            end
            TotalF = sum(F);
            % Rotate the Xks
            for iPt=1:N
                x_tilde(iPt,:)=rotate(x_tilde(iPt,:),Omega*dt);
            end
            % Update center of mass
            u_cm = dt/M*TotalF+u_cm;
            L = L+dt*sum(cross(x_tilde,F))';
            x_cm = x_cm+dt*u_cm;
            xcms(iStep+1,:)=x_cm;

        end



    end
    %update if better values
    if(target(in)>optVal)
        optCutoff=cutoff;
       optVal=target(in);
    end
    cutoff=cutoff+0.1;

end


function [stop,target, force] = GroundForce3D(x,u,mu,S,mg,cutoff)
    [h,gradH] = Hvals3dSEMI(x);
    stop=0;
    force = -mg*[0 1 0];
    target=0;
    if (h <= 0 && x(1)<cutoff)
        % Add the ground forces
        n = gradH/norm(gradH);
        Utan = u - dot(u,n)*n;
```

```
        if (norm(Utan) > 1e-10)
            UtanHat = Utan/norm(Utan);
        else
            UtanHat = [0 0 0];
        end
        force = force + 0.5*S*(-h/norm(gradH)*(n-mu*UtanHat));
    end
   if(x(1)>cutoff && x(2)<1e-4)
     force=0;
     stop=1;
     target=x(1);
   end
end
%rotate function
function rotated_x = rotate(x, Om)
    nOm = norm(Om);
    Omhat = Om/nOm;
    Px = Omhat*dot(Omhat,x);
    rotated_x = Px+cos(nOm)*(x-Px)+sin(nOm)*cross(Omhat,x);
    if(nOm < 1e-10)
        rotated_x = x;
    end
end

function [h,gradRamp] = Hvals3dSEMI(pos) % specifies the ramp
    func1 = -sqrt(1-(pos(1)-2)^2)+1;
    func2 = 0;
    func3 = -sqrt(1-(pos(1)-3)^2)+1;
    func4 = 1;
if(pos(1)>=4||2 <= pos(1) && pos(1)<= 3)
    h = pos(2)-func2;
    gradRamp = [0 1 0];
elseif(pos(1)<=1)
    h = pos(2)-func4;
    gradRamp = [0 1 0];
elseif(pos(1)<2&& 1 < pos(1))
    deriv = (pos(1)-2)*(1-(pos(1)-2)^2)^(-1/2);
    h = pos(2)-func1;
    gradRamp = [-deriv 1 0];
elseif(pos(1)>3&&pos(1)<4)
    deriv = (pos(1)-3)*(1-(pos(1)-3)^2)^(-1/2);;
    h = pos(2)-func3;
    gradRamp = [-deriv 1 0];
end
end
```

```
\subsection{Semi Circle Optimized Trajectory}
\beign{minted}{matlab}
% Simulating a wheel rolling on the ground
% Calculate moment of inertia tensor
M = 5;
N = 100;
r= 0.5;
g = 9.8;
S = 9999;
mu = 0.5;
Tf = 10;
thet = (0:N-1)'*2*pi/N;
%build circle and rotational forces
Omega = -[0 0 1];
pos = [r*sin(thet) r*cos(thet) zeros(N,1)]+[0.5 1+r 0];
x_cm = mean(pos);
x_tilde = pos-x_cm;
u_cm = [0.5 0 0];
I = zeros(3);
for k=1:N
    I = I+M/N*(norm(x_tilde(k,:))^2*eye(3)-x_tilde(k,:)'*x_tilde(k,:));
end
% I does not change for a circle
L = I*Omega';
dt = 1e-2;
nSteps= Tf/dt;
xcms = zeros(nSteps+1,3);
xcms(1,:)=x_cm;
Omegas = zeros(nSteps,1);

for iStep=1:nSteps
% Numerical method
Omega = (I \ L)';
Omegas(iStep)= Omega(3);
pos = x_tilde+x_cm;
u = cross(Omega.*ones(N,3),x_tilde)+u_cm;
F = zeros(N,3);
for iPt=1:N
    F(iPt,:) = GroundForce3D(pos(iPt,:),u(iPt,:),mu,S,M*g/N);
end
TotalF = sum(F);

% Rotate the Xks
for iPt=1:N
    x_tilde(iPt,:)=rotate(x_tilde(iPt,:),Omega*dt);
end
```

```
% Update center of mass
u_cm = dt/M*TotalF+u_cm;
L = L+dt*sum(cross(x_tilde,F))';
x_cm = x_cm+dt*u_cm;
xcms(iStep+1,:)=x_cm;
plot([pos(:,1);pos(1,1)],[pos(:,2);pos(1,2)])
hold on
plot(pos(3,1),pos(3,2),'ro')
%plot ramp
x1=1:0.01:2;
y1=-sqrt(1-(x1-2).*(x1-2))+1;
x2=2:0.1:3;
y2=0;
x3=3:0.01:3.7;
y3=-sqrt(1-(x3-3).*(x3-3))+1;
x4=0:0.1:1;
y4=1;
plot(x1,y1)
plot(x2,y2)
plot(x3,y3)
plot(x4,y4)

axis ([-1 7 0 8])
drawnow
hold off
end

function force = GroundForce3D(x,u,mu,S,mg)
    [hOfRamp,gradH] = Hvals3dSEMI(x);
    force = -mg*[0 1 0];
    if (hOfRamp <= 0)
        % Add the ground forces
        n = gradH/norm(gradH);
        disp = -(hOfRamp)/norm(gradH);
        Utan = u - dot(u,n)*n;
        if (norm(Utan) > 1e-10)
            UtanHat = Utan/norm(Utan);
        else
            UtanHat = [0 0 0];
        end
        force = force + S*(disp*(n-mu*UtanHat));
    end
end

function [h,gradRamp] = Hvals3dSEMI(pos) % specifies the ramp
    func1 = -sqrt(1-(pos(1)-2)^2)+1;
```

```
    func2 = 0;
    func3 = -sqrt(1-(pos(1)-3)^2)+1;
    func4 = 1;
if(pos(1)>=3.7||2 <= pos(1) && pos(1)<= 3)
    %uses optimal cutoff point of 3.7

    h = pos(2)-func2;
    gradRamp = [0 1 0];
elseif(pos(1)<=1)
    h = pos(2)-func4;
    gradRamp = [0 1 0];
elseif(pos(1)<2&& 1 < pos(1))
    deriv = (pos(1)-2)*(1-(pos(1)-2)^2)^(-1/2);
    h = pos(2)-func1;
    gradRamp = [-deriv 1 0];
elseif(pos(1)>3&&pos(1)<3.7)
    deriv = (pos(1)-3)*(1-(pos(1)-3)^2)^(-1/2);;
    h = pos(2)-func3;
    gradRamp = [-deriv 1 0];
end
end

function rotated_x = rotate(x, Om)
    nOm = norm(Om);
    Omhat = Om/nOm;
    Px = Omhat*dot(Omhat,x);
    rotated_x = Px+cos(nOm)*(x-Px)+sin(nOm)*cross(Omhat,x);
    if(nOm < 1e-10)
        rotated_x = x;
    end
end
```

7.2 Parabola Ramp Optimization

```
% Simulating a wheel rolling on the ground
M = 5;
N = 100;
r=0.5;
g = 9.8;
S = 10000;
mu = 0.5;
Tf = 10;
thet = (0:N-1)'*2*pi/N;

%declare variables that will hold optimal values
optCutoff=0;
```

```
optVal=0;
cutoff=2;
target = zeros(1,100);
%optimize
for(in=2:50)
    %build wheel with rotational forces
    Omega = -[0 0 1];
    x = [r*sin(thet) r*cos(thet) zeros(N,1)]+[0 r+1 0];
    x_cm = mean(x);
    x_tilde = x-x_cm;
    u_cm = [0.5 0 0];
    I = zeros(3);

    for k=1:N
        I = I+M/N*(norm(x_tilde(k,:))^2*eye(3)-x_tilde(k,:)'*x_tilde(k,:));
    end
    % I does not change for a circle
    L = I*Omega';
    dt = 1e-2;
    nSteps=Tf/dt;
    xcms = zeros(nSteps+1,3);
    xcms(1,:)=x_cm;
    Omegas = zeros(nSteps,1);

    stop=0;



    for iStep=1:nSteps
        if (stop==0)
            % Numerical method
            Omega = (I \ L)';
            Omegas(iStep)=Omega(3);
            x = x_tilde+x_cm;
            u = cross(Omega.*ones(N,3),x_tilde)+u_cm;
            F = zeros(N,3);
            for iPt=1:N
                [stop, target(in),F(iPt,:)] = GroundForce3D(x(iPt,:),u(iPt,:),mu,S,M/N*g,cut
            end
            TotalF = sum(F);
            % Rotate the Xks
            for iPt=1:N
                x_tilde(iPt,:)=rotate(x_tilde(iPt,:),Omega*dt);
            end
            % Update center of mass
            u_cm = dt/M*TotalF+u_cm;
```

```
            L = L+dt*sum(cross(x_tilde,F))';
            x_cm = x_cm+dt*u_cm;
            xcms(iStep+1,:)=x_cm;

        end




    end
    %update if better values
    if(target(in)>optVal)
        optCutoff=cutoff;
       optVal=target(in);
    end
    cutoff=cutoff+0.1;

end


function [stop,target, force] = GroundForce3D(x,u,mu,S,mg,cutoff)
    [h,gradH] = Hvals3dPARA(x);
    stop=0;
    force = -mg*[0 1 0];
    target=0;
    if (h <= 0 && x(1)<cutoff)
        % Add the ground forces
        n = gradH/norm(gradH);
        Utan = u - dot(u,n)*n;

        if (norm(Utan) > 1e-10)
            UtanHat = Utan/norm(Utan);
        else
            UtanHat = [0 0 0];
        end
        force = force + 0.5*S*(-h/norm(gradH)*(n-mu*UtanHat));
    end
   if(x(1)>cutoff && x(2)<0.7)
     force=0;
     stop=1;
     target=x(1);
   end
end
%rotate function
function rotated_x = rotate(x, Om)
    nOm = norm(Om);
```

```
    Omhat = Om/nOm;
    Px = Omhat*dot(Omhat,x);
    rotated_x = Px+cos(nOm)*(x-Px)+sin(nOm)*cross(Omhat,x);
    if(nOm < 1e-10)
        rotated_x = x;
    end
end

function [h,gradRamp] = Hvals3dPARA(pos) % specifies the ramp
    func1 = (pos(1)-2)^2;
    func2 = 0;
    func3 = (pos(1)-3)^2;
    func4 = 1;
if(pos(1)<0||pos(1)>=4||2 <= pos(1) && pos(1)<= 3)
    h = pos(2)-func2;
    gradRamp = [0 1 0];
elseif(pos(1)<=1&&0<=pos(1))
    h = pos(2)-func4;
    gradRamp = [0 1 0];
elseif(pos(1)<2&& 1 < pos(1))
    deriv = 2*(pos(1)-2);
    h = pos(2)-func1;
    gradRamp = [-deriv 1 0];
elseif(pos(1)>3&&pos(1)<4)
    deriv = 2*(pos(1)-3);
    h = pos(2)-func3;
    gradRamp = [-deriv 1 0];
end
end
```

8 Parabola Optimized Trajectory

```
% Simulating a wheel rolling on the ground
% Calculate moment of inertia tensor
M = 5;
N = 100;
r= 0.5;
g = 9.8;
S = 9999;
mu = 0.5;
Tf = 10;
thet = (0:N-1)'*2*pi/N;
Omega = -[0 0 1];
pos = [r*sin(thet) r*cos(thet) zeros(N,1)]+[0.5 1+r 0];
x_cm = mean(pos);
```

```
x_tilde = pos-x_cm;
u_cm = [0.5 0 0];
I = zeros(3);
for k=1:N
    I = I+M/N*(norm(x_tilde(k,:))^2*eye(3)-x_tilde(k,:)'*x_tilde(k,:));
end
% I does not change for a circle
L = I*Omega';
dt = 1e-2;
nSteps= Tf/dt;
xcms = zeros(nSteps+1,3);
xcms(1,:)=x_cm;
Omegas = zeros(nSteps,1);

for iStep=1:nSteps
%numerical method implementation
Omega = (I \ L)';
Omegas(iStep)= Omega(3);
pos = x_tilde+x_cm;
u = cross(Omega.*ones(N,3),x_tilde)+u_cm;
F = zeros(N,3);
for iPt=1:N
    F(iPt,:) = GroundForce3D(pos(iPt,:),u(iPt,:),mu,S,M*g/N);
end
TotalF = sum(F);

% Rotate the Xks
for iPt=1:N
    x_tilde(iPt,:)=rotate(x_tilde(iPt,:),Omega*dt);
end
% Update center of mass
u_cm = dt/M*TotalF+u_cm;
L = L+dt*sum(cross(x_tilde,F))';
x_cm = x_cm+dt*u_cm;
xcms(iStep+1,:)=x_cm;
plot([pos(:,1);pos(1,1)],[pos(:,2);pos(1,2)])
hold on
plot(pos(3,1),pos(3,2),'ro')
%plot ramp function
x1=1:0.1:2;
y1=(x1-2).^2;
x2=2:0.1:3;
y2=0;
x3=3:0.1:3.5;
y3=(x3-3).^2;
x4=0:0.1:1;
```

```
y4=1;
plot(x1,y1)
plot(x2,y2)
plot(x3,y3)
plot(x4,y4)

%plot(xlim,[0 0],'-k')
axis ([-1 7 0 8])
drawnow

end

function force = GroundForce3D(x,u,mu,S,mg)
    [hOfRamp,gradH] = Hvals3dPARA(x);
    force = -mg*[0 1 0];
    if (hOfRamp <= 0)
        % Add the ground forces
        n = gradH/norm(gradH);
        disp = -(hOfRamp)/norm(gradH);
        Utan = u - dot(u,n)*n;
        if (norm(Utan) > 1e-10)
            UtanHat = Utan/norm(Utan);
        else
            UtanHat = [0 0 0];
        end
        force = force + S*(disp*(n-mu*UtanHat));
    end
end

function [h,gradRamp] = Hvals3dPARA(pos) % specifies the ramp
    func1 = (pos(1)-2)^2;
    func2 = 0;
    func3 = (pos(1)-3)^2;
    func4 = 1;
if(pos(1)<0||pos(1)>=3.3||2 <= pos(1) && pos(1)<= 3) %use optimal cut off
    h = pos(2)-func2;
    gradRamp = [0 1 0];
elseif(pos(1)<=1&&0<=pos(1))
    h = pos(2)-func4;
    gradRamp = [0 1 0];
elseif(pos(1)<2&& 1 < pos(1))
    deriv = 2*(pos(1)-2);
    h = pos(2)-func1;
    gradRamp = [-deriv 1 0];
elseif(pos(1)>3&&pos(1)<3.3) %use optimal cut off
    deriv = 2*(pos(1)-3);
```

```
    h = pos(2)-func3;
    gradRamp = [-deriv 1 0];
end
end
```

```
function rotated_x = rotate(x, Om)
    nOm = norm(Om);
    Omhat = Om/nOm;
    Px = Omhat*dot(Omhat,x);
    rotated_x = Px+cos(nOm)*(x-Px)+sin(nOm)*cross(Omhat,x);
    if(nOm < 1e-10)
        rotated_x = x;
    end
end
```