Week 4 Coriolis Effect

Written by: Caiden Bonney

Created: 2/5/25

Modified: 2/5/25

Background:

This lab studies the motion of rotating bodies that are translating relative to one another by modeling the position of a ball thrown from a merry-go-round through different reference frames through varying methods. Method 1 is by modeling the position of the ball in the fixed frame and converting the coordinates to the rotational frame. Method 2 computes the position of the ball by integrating the velocity of the ball in the rotational frame's coordinates. This lab also introduced the concepts of functions in matlab.

Required Files:

- ME_326_Coriolis_Effect_Report.mlx

Table of Contents

Problem Statement

This is a visualization of translational motion on a rotating body, which will induce the Coriolis effect. We will study the motion of a ball, thrown by a person (A), which will end up being caught by person (B),

The following representations are made for the simulation:

- Person (A) is represented by a blue star (*).
- Person (B) is represented by a black cross (x).
- The ball is represented by a red circle (o).
- The fixed coordinate system $XY(Z)$ is represented by black axes.
- The rotating coordinate system $xy(z)$ is represented by blue axes.

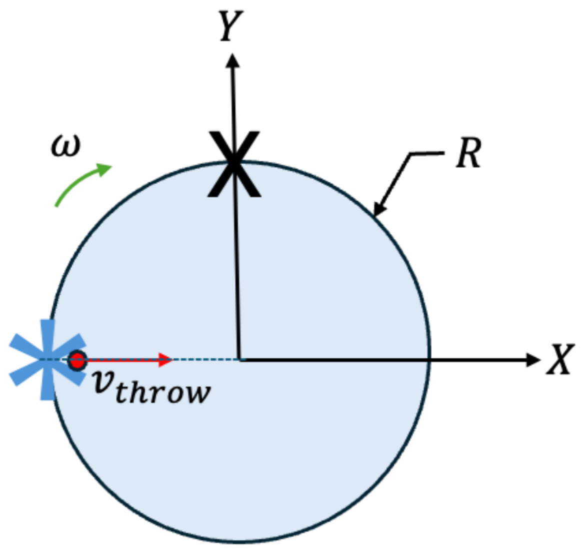


Figure 5 Merry-go-round. Ball (red circle) is being thrown by person A (blue asterisk). Due to the rotation of the merry-go-round in the clockwise direction, the ball will be caught by person B (black cross).

Figure depicting the merry-go-round setup for Part 2A, 2B Method 1 and 2B Method 2.

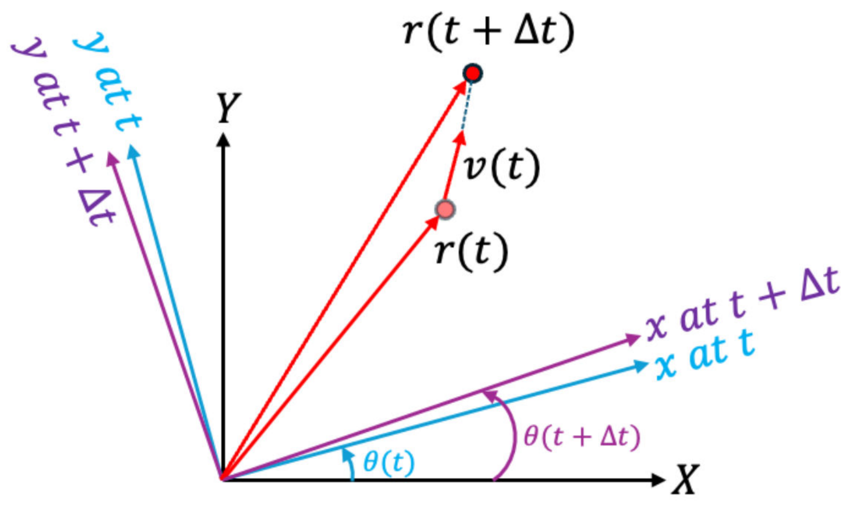


Figure 5 visualization of the rotating frame over time. Note that rotation here is counterclockwise as opposed to the clockwise rotation you have in your assignment.

Figure depicting Method 2 in which the position of the next time step is based on the previous time step in the rotated coordinate system.

Hand Calculations

Part 1: Computing the position of an object in two different frames

Consider point P shown in the figure below. We know the position of the point as measured in the fixed frame ${}^{XYZ}\vec{r}_P = 3\hat{I} + 4\hat{J}$. Find the position of the same point as measured in the rotated frame ${}^{xyz}\vec{r}_P$. Use the two following approaches:

a. Trigonometry (sines and cosines and projections).

b. Apply the rotational matrix equation highlighted above.

Compare your answers. Hint: you should get the same results for parts a and b.

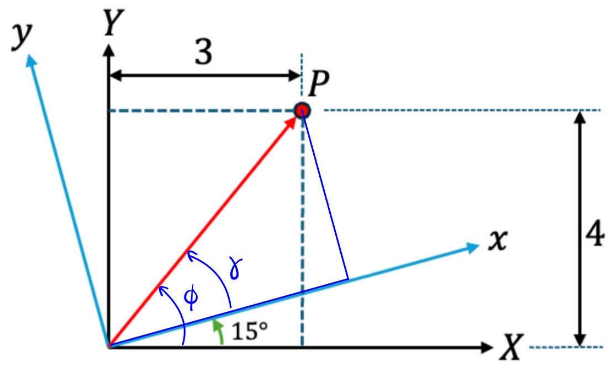


Figure 4 Part 1: figure showing the position of point P.

a)

$$\tan\phi = \frac{4}{3}$$
$$\phi = \tan^{-1}\left(\frac{4}{3}\right)$$
$$\phi = 53.13^\circ$$

$$\gamma = \phi - \theta$$
$$= 53.13^\circ - 15^\circ$$
$$= 38.13^\circ$$

$${}^{xyz}\vec{r}_p = \|{}^{XYZ}\vec{r}_p\|\left(\cos\gamma\,\hat{\imath} + \sin\gamma\,\hat{\jmath}\right)$$
$$= 5\left(\cos\gamma\,\hat{\imath} + \sin\gamma\,\hat{\jmath}\right)$$
$$= 5\left(0.7866\,\hat{\imath} + 0.6174\,\hat{\jmath}\right)$$
$${}^{xyz}\vec{r}_p = 3.933\,\hat{\imath} + 3.087\,\hat{\jmath}$$ ✓

$${}^{XYZ}\vec{r}_p = 3\hat{I} + 4\hat{J}$$
$$\|{}^{XYZ}\vec{r}_p\| = \sqrt{3^2 + 4^2}$$
$$= \sqrt{9 + 16}$$
$$= \sqrt{25}$$
$$= 5$$

b)

$${}^{xyz}\vec{r}_p = {}^{xyz}R_{XYZ} \cdot {}^{XYZ}\vec{r}_p$$

$${}^{xyz}R_{XYZ} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad {}^{XYZ}\vec{r}_p = \begin{bmatrix} 3 \\ 4 \\ 0 \end{bmatrix}$$

$${}^{xyz}\vec{r}_p = \begin{bmatrix} \cos 15^\circ & \sin 15^\circ & 0 \\ -\sin 15^\circ & \cos 15^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 4 \\ 0 \end{bmatrix} = \begin{bmatrix} 3.933 \\ 3.087 \\ 0 \end{bmatrix}$$

$${}^{xyz}\vec{r}_p = 3.933\,\hat{\imath} + 3.087\,\hat{\jmath}$$ ✓

The position of an object can be represented in both the fixed coordinate system XYZ and the rotational coordinate system xyz. The conversion between these coordinate systems can be done through both trigonometry in part a of Part 1 or through a rotational matrix such as done in part b of Part 1.

Part 2B Method 2 Hand Calcs

General Equation

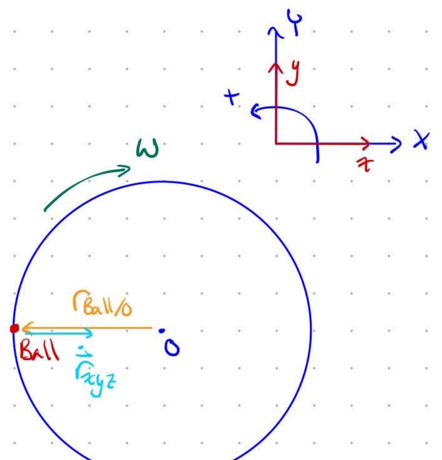
$$\vec{V}_B = \vec{V}_A + \vec{\Omega} \times \vec{r}_{B/A} + \dot{\vec{r}}_{xyz}$$

Ball is the location of the ball

O is the center of the merry-go-round

$\vec{\omega} = -0.739\hat{k}$ rad/s

$R = 1$ m (radius of merry-go-round)



$V_{throw} = 1.673$ m/s

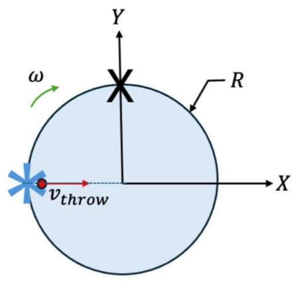


Figure 5 Merry-go-round. Ball (red circle) is being thrown by person A (blue asterisk). Due to the rotation of the merry-go-round in the clockwise direction, the ball will be caught by person B (black cross).

$$\vec{V}_{Ball} = \cancel{\vec{V}_O}^{0} + \vec{\omega} \times \vec{r}_{Ball/O} + \dot{\vec{r}}_{xyz}$$

$$\vec{V}_{Ball} = -0.739\hat{k} \times -\hat{I} + 1.639\hat{I}$$
$$= 0.739\hat{J} + 1.639\hat{I}$$
$$= 1.639\hat{I} + 0.739\hat{J} \text{ m/s}$$

$$\vec{r}_{Ball/O} = -R\hat{\imath}$$
$$= -\hat{\imath} \text{ m}$$

$$\dot{\vec{r}}_{xyz} = V_{throw}\hat{\imath}$$
$$= 1.639\hat{\imath} \text{ m/s}$$

The velocity of the ball does not change in the XYZ coordinates as there are no forces acting upon the ball after it is thrown.

Therefore to represent the velocity in xyz rotate $\vec{V}_{Ball}$ by the amount the rotating coordinate system rotated since time =0, since at time =0 the XYZ coordinate system matched with xyz.

Therefore by accounting for the rotation of the rotating coordinate system a velocity that is constant in the fixed frame can be modeled.

After obtaining the velocity in the rotational frame the position can be calculated through integration such as Euler's forward integration:

$$\vec{r}_{ball}(t+\Delta t) = \vec{r}_{ball_{xyz}}(t) + \vec{V}_{ball_{xyz}}(t) \cdot \Delta t$$

Setup

```
close all
clear
clc

w = -0.739; %angular velocity of the merry-go-round [rad/s]
t_final = 1; %time the simulation ends [s]
R = 1; %radius of the merry-go-round [m]
v_throw = 1.673; %throw speed relative to the merry-go-round [m/s]
```

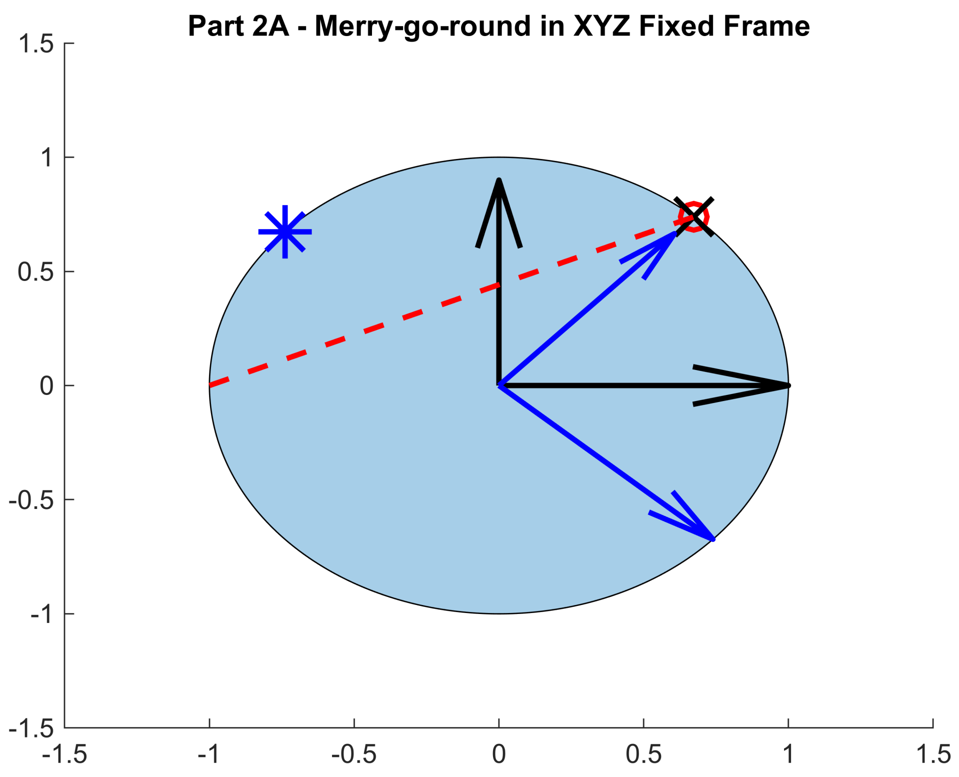
Part 2A Analysis in the Fixed Frame

Studying the motion of the ball as seen by a stationary observer (bird-eye view)

After calculating the inital velocity of the ball and setting the intial position the position for the next time step is then calculated through an Euler forward integration in vector form in the XYZ Fixed Frame coordinates. The ball's position in the current time step is calculated from the previous time step however the position of points A and B are not so the current position of A and B is calculated by rotating their positions based on the current theta of the merry-go-round. The current theta is calculated based on the amount of time that has elapsed since the start and the rate of rotation, w. Finally the coordinate system vectors are drawn. The plot is in the XYZ Fixed Frame therefore the fixed coordinate system does not change with time. The rotational coordinate system xyz rotates in the same way that positions A and B since the rotational coordinate system is attached to the merry-go-round.

```
v_Ball = [v_throw;0;0] + cross([0;0;w],[-R;0;0]); % initial velocity the ball is
thrown at in XYZ (Fixed Frame)

dt = 0.010; %integration time step
t = 0:dt:t_final; %time array

X(1) = -R; %initial position of the ball in the x direction
Y(1) = 0; %initial position of the ball in the y direction
position{1} = [X(1);Y(1);0];

Disk = nsidedpoly(1000, 'Center', [0 0], 'Radius', R); %Ask MATLAB create a
Polyshape
%this can be plotted to draw a disk (the merry-go-round).

%How to animate on MATLAB:

%Simply plot within a loop

%Use command 'pause(time)' to pause for 'time' seconds

%If you want to draw multiple things at once 'hold on' so that MATLAB keeps
%everything in the plot
```

```
%If you want to draw something and erase everything else 'hold off' then
%plot whatever you want

for i = 1:length(t)
    figure (1)
    plot(Disk)
    title("Part 2A - Merry-go-round in XYZ Fixed Frame")
    hold on

    %the instance t(i). Make sure this quantity is a 3x1 vector.

    %recommendation: for simplicity, apply Euler forward integration in
    %vector form like shown below:

    position{i+1} = position{i} + dt*v_Ball; %notice that ignoring air resistance
and gravity
    %means that z is constant and we dont care about it. (height of ball)

    X(i+1) = position{i+1}(1,1);
    Y(i+1) = position{i+1}(2,1);

    %Indicate the location of each person (person A and person B) as
    %measured by the fixed frame at this time step:

    theta = w*t(i);

    % the position vectors are in xyz coords. RotationMatrix is the
    % rotational matrix to convert from XYZ to xyz. Therefore to convert
    % from xyz the use the inverse of the rotational matrix from
    % RotationMatrix.
    B_position{i} = RotationMatrix(theta)^-1*[0;R;0];
    A_position{i} = RotationMatrix(theta)^-1*[-R;0;0];

    YB = B_position{i}(2,1);
    XB = B_position{i}(1,1);
    XA = A_position{i}(1,1);
    YA = A_position{i}(2,1);
    %%

    plot(X(i),Y(i),'ro',MarkerSize=10,LineWidth=2) %plot where the ball is at this
moment
    plot(XB,YB,'kx',MarkerSize=20,LineWidth=2) %plot where person B is at this
moment
    plot(XA,YA,'b*',MarkerSize=20,LineWidth=2) %plot where person A is at this
moment

    DrawCoordinateSystem(R,0,'k'); %make a plot of the X and Y axes at this time.
The plot must represent
    %two vectors (including vector arrows) with black color and line width
    %of 2. Use the DrawCoordinateSystem function
    DrawCoordinateSystem(R,theta,'b'); %make a plot of the x and y axes at this
time. The plot must represent
    %two vectors (including vector arrows) with blue color and line width
    %of 2. Use the DrawCoordinateSystem function

    % plots the trajectory of the ball during movement
    % plot(X,Y,'r--',LineWidth=2)

    pause(0.01) %update plot every 10 ms
    hold off %I'm holding off here so that, once the next loop starts,
    %I erase everything from this loop from my plot.
end

%Plot the total trajectory of the ball below on the same figure once the
%loop concludes:

hold on
plot(X,Y,'r--',LineWidth=2)
hold off
```



Graph depicts the location of the ball, person A, and person B in the fixed frame coordinate XYZ. The location of the axes for the fixed frame XYZ and the rotational frame xyz are also shown. After the ball is caught by person B the trajectory the ball took is plotted over the merry-go-round.

Part 2B Analysis in the Rotating Frame - Method 1

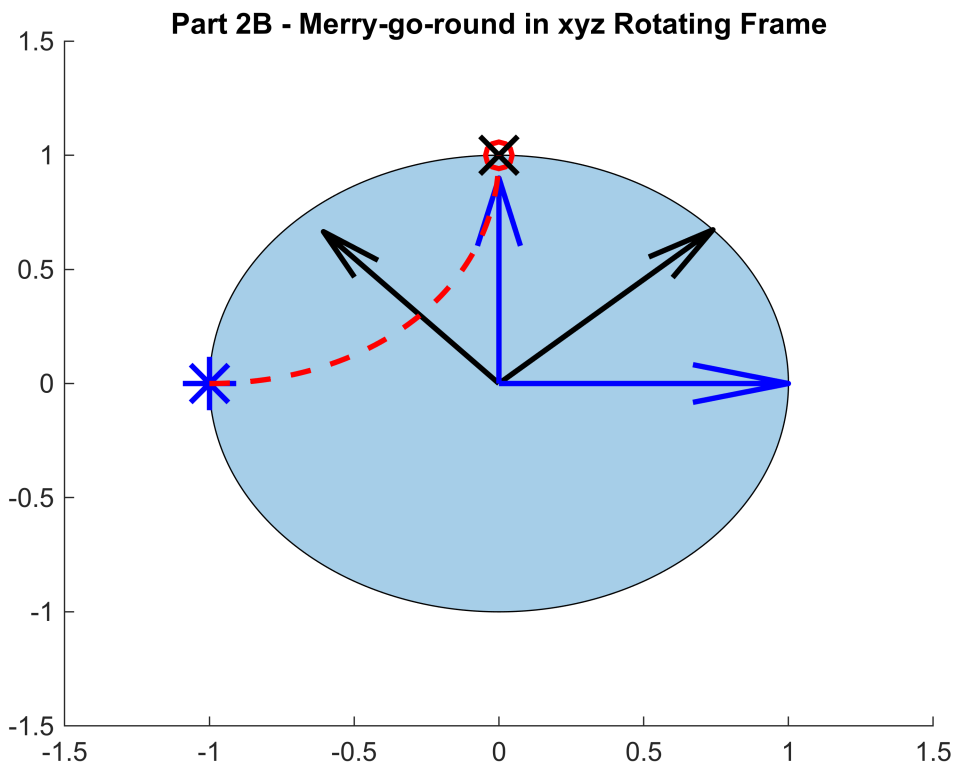
Here, we study the motion of the ball from the perspective of an observer rotating with the the merry-go-round.

For the first study, we simply find the location of the ball as a function of time by utilizing our results for X(t) and Y(t) from the part A and multiplying these results by a rotation matrix representing the angle of rotation of the merry-go-round frame xyz relative to the fixed frame XYZ at each time step. (no integration is needed in this part).

In this section we are calculating the rotational frame coordinates of the ball, person A, and person B given their coordinates in the fixed frame coordinate system XYZ. This is accomplished through rotating their location vectors with the rotational matrix from XYZ fixed frame coordinates to xyz rotational coordinates at each time step. the plot is depected in the rotational coordinates xyz therefore similar to Part 2A but for the rotational axes the rotational frame xyz axes do not move in this frame, however, the fixed frame coordinates XYZ rotate counter clockwise by the current theta.

```
for i=1:length(t)
    figure(2)
    theta = w*t(i); %find an expression for theta(t)

    position_rotating{i} = RotationMatrix(theta)*position{i}; %find the position of
the ball (as a vector) at this moment
    %use what you have learned about rotation matrices to your advantage.
    %Use the RotationMatrix function

    x_rotating(i) = position_rotating{i}(1,1); %position in the x direction at this
moment
    y_rotating(i) = position_rotating{i}(2,1); %position in the y direction at this
moment

    %Indicate the locations of person A and B as measured by the rotating
    %frame

    A_position_rotating{i} = RotationMatrix(theta)*A_position{i};
    B_position_rotating{i} = RotationMatrix(theta)*B_position{i};

    xA_rotating = A_position_rotating{i}(1,1);
    yA_rotating = A_position_rotating{i}(2,1);
    xB_rotating = B_position_rotating{i}(1,1);
    yB_rotating = B_position_rotating{i}(2,1);

    plot(Disk)
    title("Part 2B - Merry-go-round in xyz Rotating Frame")
    hold on
    plot(x_rotating(i),y_rotating(i),'ro',MarkerSize=10,LineWidth=2)
    plot(xA_rotating,yA_rotating,'b*',MarkerSize=20,LineWidth=2)
    plot(xB_rotating,yB_rotating,'kx',MarkerSize=20,LineWidth=2)

    DrawCoordinateSystem(R,-theta,'k');%make a plot of the X and Y axes at this
time. The plot must represent
```

```
    %two vectors (including vector arrows) with black color and line width
    %of 2

    DrawCoordinateSystem(R,0,'b'); %make a plot of the x and y axes at this time.
The plot must represent
    %two vectors (including vector arrows) with blue color and line width
    %of 2

    % plots the trajectory of the ball during movement
    % plot(x_rotating,y_rotating,'r--',LineWidth=2)

    pause(0.01) % update plot every 10 ms
    hold off
end

%Plot the total trajectory of the ball below on the same figure once the
%loop concludes:
hold on
plot(x_rotating,y_rotating,'r--',LineWidth=2)
hold off
```



Graph depicts the location of the ball, person A, and person B in the rotational frame coordinate xyz. The location of the axes for the fixed frame XYZ and the rotational frame xyz are also shown. After the ball is caught by person B the trajectory the ball took is plotted over the merry-go-round.

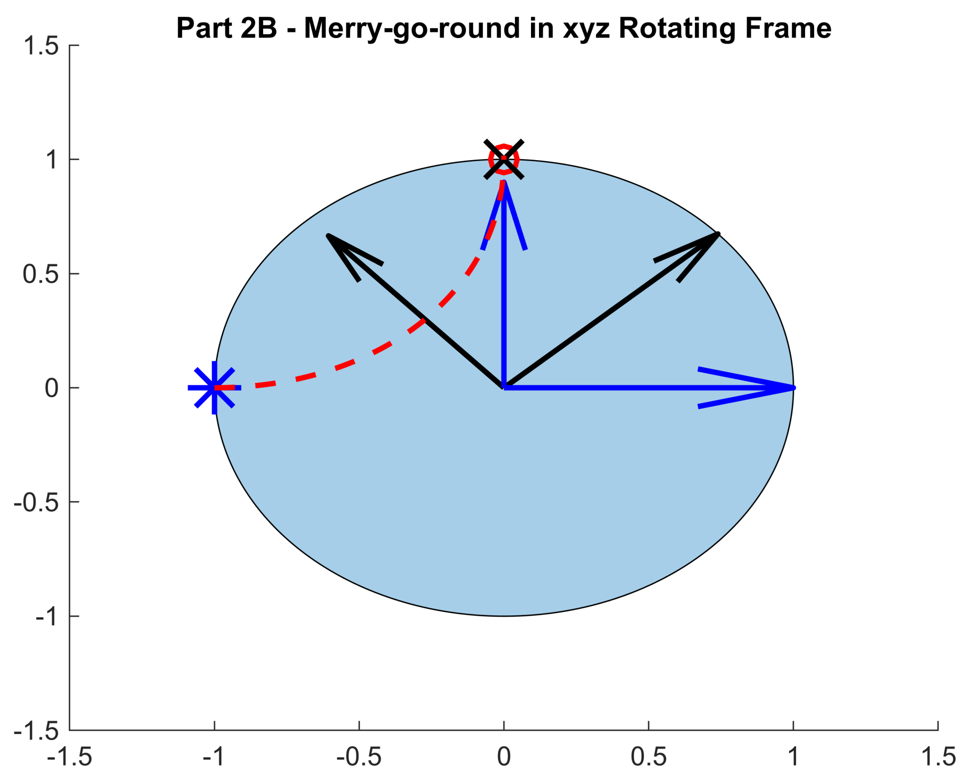
Part 2B Analysis in the Rotating Frame - Method 2

Here, we study the motion of the ball from the perspective of an observer rotating with the the merry-go-round.

For this study, we will do the entire analysis of motion i.e, going from velocity to position of the ball in the rotating frame. Thus, we do the integration much like we have done in part A, but in the rotating frame instead here. The analysis is more complicated here than B. Part 1, but it yields the same reults.

Given the inital position of the ball, the inital velocity the ball was thrown at in xyz rotational coordinates, and the intial positions of person A and person B, the location of the ball must be calculated through numerical integration at each time step. The velocity of the ball is not affected by the rotation of the merry-go-round after it has already been thrown therefore the velocity vector must change with repsect to the rotational xyz coordinates since the coordinates themselves are rotating. Therefore the rotation of the coordinate system at each time step is counteracted by rotating the previous velocity vector by the amount the roational frame coordinates rotated in that time step. After finding the velocity vector for the next time step the position of the ball in this next time step can be calculated through Euler's forward integration. Just as in Part 2B Method 1 the axes of the rotational frame xyz do not move in their own frame of reference, however, the fixed frame coordinates XYZ rotate counter clockwise.

```
x(1) = -R; %position of the ball as measured in the rotating frame initially in x
y(1) = 0; %position of the ball as measured in the rotating frame initially in y
r{1} = [x(1);y(1);0];
v_Ball_xyz{1} = [v_throw;0;0] + cross([0;0;w],r{1}); % velocity of ball when its
first thrown
A_position_rotating_m2 = [-R;0;0]; % starting position of A
B_position_rotating_m2 = [0;R;0]; % starting position of B
dtheta = w*dt; % the amount the axes shift per time step dt

for i = 1:length(t)
    figure(3)
    theta = w*t(i);

    % the ball does not have any forces acting upon it after it is thrown
    % since air resistance and gravity are neglected and therefore has no
    % change in acceleration after it is thrown. Visually shown in Part 2A

    % since there is no acceleration on the ball the velocity is constant.
    % (visually shown in Part 2A). Therefore to represent the velocity vector
    % in xyz coordinates based on its previous time step in xyz coordinates
    % the amount of rotation the xyz coordinates moved in that time step
    % must be counteracted
    v_Ball_xyz{i+1} = RotationMatrix(dtheta)*v_Ball_xyz{i};

    % Now that the velocity vector at the new time step (i+1) is known in
    % the xyz coordinate system its affect on the position can be
    % calculated based on the velocity integration in the rotating frame.
    r{i+1} = RotationMatrix(dtheta)*r{i} + v_Ball_xyz{i+1}*dt;
    x(i+1) = r{i+1}(1,1);
    y(i+1) = r{i+1}(2,1);
    plot(Disk)
    title("Part 2B - Merry-go-round in xyz Rotating Frame")
    hold on
    plot(r{i}(1,1),r{i}(2,1),'ro',MarkerSize=10,LineWidth=2)

    % A and B are always at the same position relative to the rotating axis

plot(A_position_rotating_m2(1,1),A_position_rotating_m2(2,1),'b*',MarkerSize=20,Line
Width=2)

plot(B_position_rotating_m2(1,1),B_position_rotating_m2(2,1),'kx',MarkerSize=20,Line
Width=2)

    DrawCoordinateSystem(R,-theta,'k');%make a plot of the X and Y axes at this
time. The plot must represent
    %two vectors (including vector arrows) with black color and line width
    %of 2

    DrawCoordinateSystem(R,0,'b'); %make a plot of the x and y axes at this time.
The plot must represent
    %two vectors (including vector arrows) with blue color and line width
    %of 2

    % plots the trajectory of the ball during movement
    % plot(x,y,'r--',LineWidth=2)

    pause(0.01) % update plot every 10 ms
    hold off
end

hold on
plot(x,y,'r--',LineWidth=2)
hold off
```



Graph depicts the location of the ball, person A, and person B in the rotational frame coordinate xyz. The location of the axes for the fixed frame XYZ and the rotational frame xyz are also shown. After the ball is caught by person B the trajectory the ball took is plotted over the merry-go-round. The locations of these objects are calculated through integration in the rotating frame instead of the fixed frame as it was in Parts 2A and 2B Method 1.

Functions Used

```
function Rot = RotationMatrix(theta)
    Rot = [cos(theta) sin(theta) 0; -sin(theta) cos(theta) 0; 0 0 1];
    % resulting matrix for converting XYZ to xyz so must inverse if going from xyz
to XYZ
end

function Result = DrawCoordinateSystem(Length, Angle, Color)
%x axis
    quiver(0, 0, Length*cos(Angle), Length*sin(Angle), 0, 'LineWidth', 2,
'MaxHeadSize', 1, 'Color', Color);
%y axis
    quiver(0, 0, Length*cos(Angle+pi()/2), Length*sin(Angle+pi()/2), 'LineWidth',
2, 'MaxHeadSize', 1, 'Color', Color);
end
```