Copied below is a Blackboard communication I sent to students stemming from a suggestion by Bob Darling that I share this resource, following his visit to my PHY 420 class. He suggested that I should share with students the Python script I developed to illustrate the trajectory of a particle in a rotating reference frame.

This question was presented to the class with the following goal: At what angle should we throw a ball in a rotating reference frame if we want to play catch with ourselves? The Python script presents a set of scenarios and a solution to the catch game. A series of screenshots from the Python code (as was presented in class) are also shown below.

- -

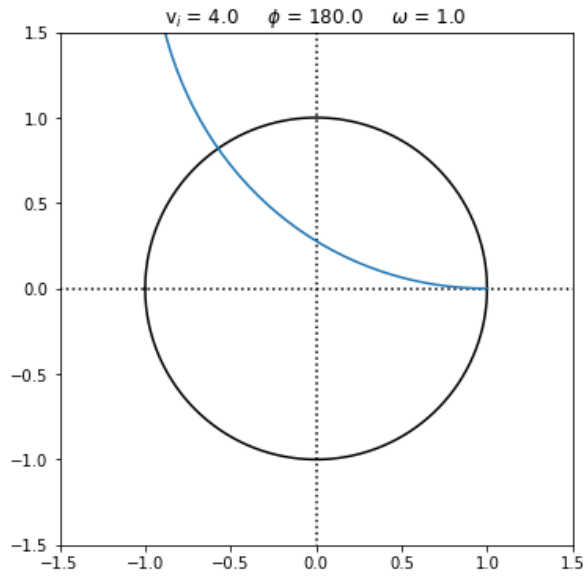
Rotating reference frame script
Posted on Tuesday, October 15, 2019 10:26:41 AM EDT

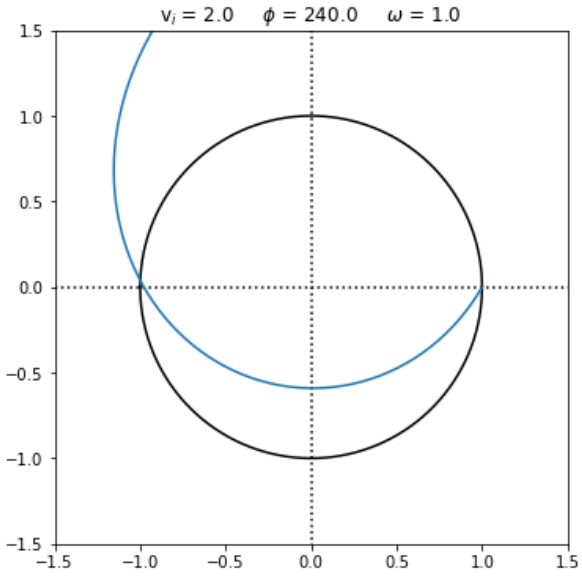
I have uploaded the python script that we used yesterday for the discussion of trajectories in a rotating reference frame. In contrast to how our textbook arrives at a solution, this script solves by the second method we discussed, that is, transformation of coordinates. If you are interested, you can go into the code and play around with the values to observe what different trajectories you can develop.

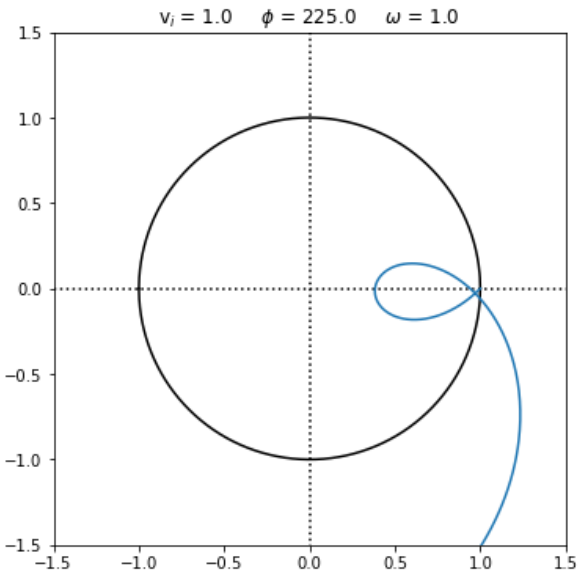
To run this you will need to have a local version of a Python compiler on your computer. There are many that you can use, though I recommend the Anaconda package (https://www.anaconda.com/distribution/) because it is rather seamless and has a nice interface. Try using the Spyder application (Anaconda has a number of different editing tools within it, you will see when you launch it) in particular.

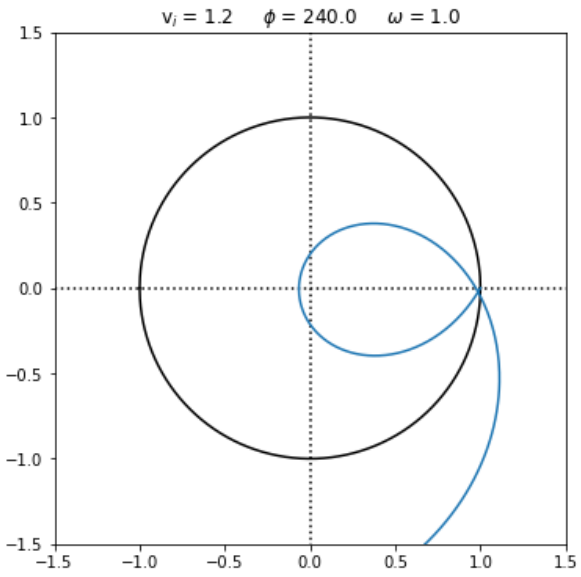
- -

Screenshots from calculations:









Copy of Python code uploaded to Blackboard

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Oct  8 09:39:11 2019

@author: eric.edlund
"""

import numpy as np
import matplotlib.pyplot as plt

case = 7

if case == 0:
    om = 1.0
    ri = 1.0
    vi = 1.0
    th = 0.0
    tr = th * np.pi / 180

if case == 1:
    om = 1.0
    ri = 1.0
    vi = 1.0
    th = 45.0
    tr = th * np.pi / 180

if case == 2:
    om = 1.0
    ri = 1.0
```

```
    vi = 1.0
    th = 90.0
    tr = th * np.pi / 180

if case == 3:
    om = 1.0
    ri = 1.0
    vi = 1.0
    th = 180.0
    tr = th * np.pi / 180

if case == 4:
    om = 1.0
    ri = 1.0
    vi = 4.0
    th = 180.0
    tr = th * np.pi / 180

if case == 5:
    om = 1.0
    ri = 1.0
    vi = 1.0
    th = 225.0
    tr = th * np.pi / 180

if case == 6:
    om = 1.0
    ri = 1.0
    vi = 1.2
    th = 240.0
    tr = th * np.pi / 180

if case == 7:
    om = 1.0
    ri = 1.0
    vi = 1.0
    th = 240.0
    tr = th * np.pi / 180

if case == 8:
    om = 1.0
    ri = 1.0
    vi = 2.0
    th = 240.0
    tr = th * np.pi / 180



if case == 11:
    om = 1.0
    ri = 0.0
    vi = 1.0
    th = 240.0
```

```
    tr = th * np.pi / 180


vx = vi * np.cos(tr)
vy = vi * np.sin(tr) + om * ri


x0 = ri
y0 = 0.0

t  = np.arange(0,5,0.01)
Nt = len(t)

rx = ri + vx * t
ry = vy * t

th_rot = om * t

xp = +rx * np.cos(th_rot) + ry * np.sin(th_rot)
yp = -rx * np.sin(th_rot) + ry * np.cos(th_rot)

rotx = ri * np.cos(th_rot)
roty = ri * np.sin(th_rot)



r = np.sqrt((rx - rotx)**2 + (ry - roty)**2)

r0 = 1.0
xcirc = r0*(-1 + 2*np.arange(1001)/1000)
ycirc = np.sqrt(r0**2 - xcirc**2)

limr = 1.5

plt.figure(figsize=(6,6))
plt.title('v$_i$ = {}     $\phi$ = {}     $\omega$ = {}'.format(vi,th,om))
plt.plot([-10,10],[0,0],color='k',linestyle=':')
plt.plot([0,0],[-10,10],color='k',linestyle=':')
plt.plot(xcirc,+ycirc,color='k')
plt.plot(xcirc,-ycirc,color='k')
plt.plot(xp,yp)
plt.xlim(-limr,+limr)
plt.ylim(-limr,+limr)
plt.gca().set_aspect('equal', adjustable='box')
plt.draw()
```