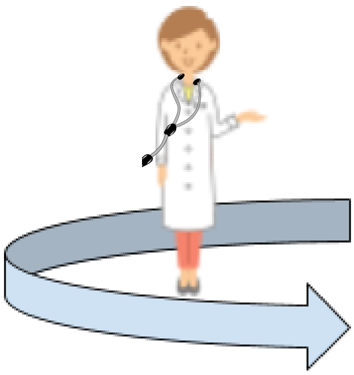
ENGR 122 Laboratory Instructions 2018



Lab 5: Spin Doctor Respun

5.1 Aim

This lab exercise is a follow-up to the Spin Doctor lab in ENGR 121. We will review a bit and then work through some harder problems. Our goals are:

1) Understanding rotation, translation, reflection, and the matrices that implement these operations.
2) Working with matrices
3) Developing familiarity and skill with trig identities.

5.2 Review and getting Python ready

The appendix at the end of the lab script has some notes about matrices in Python. Read through it for review. Include your Python work in your answers for the remainder of the lab. It is probably easiest to work in a text editor and paste into Python so you can reuse chunks of code.

You will need to import math and matrix functions:

Use from numpy import math, from numpy import matrix, from numpy.linalg import inv.

5.3 The Rotation Matrix

This is your basic rotation matrix. It rotates a point counterclockwise in the XY plane by angle ϑ:

$$R=\begin{pmatrix}\cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta\end{pmatrix}$$

Let's take this matrix out for a spin. We will start with the point (0,-1) and rotate it 270 degrees, and find out where it ends up.

Hints: Try this. Use a text editor. You can copy and paste this and then modify it for other parts of the lab. Remember to use ctrl-shift-v to paste in Python. Here is what you would do to create a matrix that rotates by pi. You can adapt that to your needs.

```
R = (math.cos(math.pi), -math.sin(math.pi), math.sin(math.pi), math.cos(math.pi))

R=matrix(R).reshape(2,2)

P=(0,-1)

P=matrix(P).reshape(2,1)

print R*P
```

Hint: Don’t forget to use radians rather than degrees throughout the lab.

CORE 1 (20 marks)
Verify two basic properties of rotations for at least one example in each case.

(1) A rotation by $\theta+2n\pi$ is the same as rotation by θ.
(2) Rotation by θ and then by $-\theta$ gets you back where you started.

<><><><><><><><><><><><><><><><><><><><><><><>

(1):

```
degrees = math.pi/180
theta = degrees*90

R = (math.cos(theta), -math.sin(theta), math.sin(theta), math.cos(theta))
R=matrix(R).reshape(2,2)
P = (0,-1)
P=matrix(P).reshape(2,1)
print R*P
```

<u>Output:</u>

```
[[ 1.000000e+00]
 [-6.123234e-17]]
```

```
degrees = math.pi/180
theta = degrees*90 + 2*math.pi

R = (math.cos(theta), -math.sin(theta), math.sin(theta), math.cos(theta))
R=matrix(R).reshape(2,2)
P = (0,-1)
P=matrix(P).reshape(2,1)
print R*P
```

<u>Output:</u>

```
[[ 1.000000e+00]
 [-3.061617e-16]]
```

Both theta and theta+2pi output the same result. This is because 2 pi radians is equal to 360 degrees or a full rotation, meaning that there is no difference in the result.

(2):

```
degrees = math.pi/180
theta = degrees*90

R = (math.cos(theta), -math.sin(theta), math.sin(theta), math.cos(theta))
Q = (math.cos(-theta), -math.sin(-theta), math.sin(-theta), math.cos(-theta))

R = matrix(R).reshape(2,2)
Q = matrix(Q).reshape(2,2)

P = (0,-1)

P = matrix(P).reshape(2,1)

Z = (R*P)

print(Z)
print(Q*Z)
```

<u>Outputs</u>

```
[[ 1.000000e+00]
 [-6.123234e-17]]

[[ 0.]
 [-1.]]
```

Returned to original point (0, -1)

<><><><><><><><><><><><><><><><><><><><><><><>

CORE 2 (20 marks)
Now let's work on a non-square rectangle. Your rectangle is (1,1) (4,1) (1,3) (4,3). Rotate it so that it is in the third quadrant. Hint: rotate each point.
<><><><><><><><><><><><><><><><><><><><><><><>
Rotate each point by 180 degrees:

```
degrees = math.pi/180
theta = degrees*180

P1 = (1,1)
P2 = (4,1)
P3 = (1,3)
P4 = (4,3)

R = (math.cos(theta), -math.sin(theta), math.sin(theta), math.cos(theta))

R = matrix(R).reshape(2,2)
P1 = matrix(P1).reshape(2,1)
P2 = matrix(P2).reshape(2,1)
P3 = matrix(P3).reshape(2,1)
P4 = matrix(P4).reshape(2,1)

print(R*P1)
[[ -1.]
 [-1.]]

print(R*P2)
[[ -4.]
 [-1.]]

print(R*P3)
[[ -1.]
 [-3.]]

print(R*P4)
[[ -4.]
 [-3.]]
```

<><><><><><><><><><><><><><><><><><><><><><><>

COMPLETION 1 (15 marks)

Now we will work with our rotation matrix algebraically. You can work on a sheet of paper and upload a photo of your work. Use the definition of the rotation matrix, matrix multiplication, and trig identities show that a rotation by an angle θ_1 and then θ_2 is the same as a rotation by $\theta_1+\theta_2$.

<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

Rotate By θ_1, then θ_2

$$\begin{bmatrix} \cos(\theta_1), & -\sin(\theta_1) \\ \sin(\theta_1), & \cos(\theta_1) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a\cos(\theta_1) - b\sin(\theta_1) \\ a\sin(\theta) + b\cos(\theta) \end{bmatrix}$$

$$\begin{bmatrix} \cos(\theta_2), & -\sin(\theta_2) \\ \sin(\theta_2), & \cos(\theta_2) \end{bmatrix} \begin{bmatrix} a\cos(\theta_1) - b\sin(\theta_1) \\ a\sin(\theta_1) + b\cos(\theta_1) \end{bmatrix}$$

$$\begin{bmatrix} \cos(\theta_2)\left(a\cos(\theta_1) - b\sin(\theta_1)\right) - \sin(\theta_2)\left(a\sin(\theta_1) + b\cos(\theta_1)\right) \\ \sin(\theta_2)\left(a\cos(\theta_1) - b\sin(\theta_1)\right) + \cos(\theta_2)\left(a\sin(\theta_1) + b\cos(\theta_1)\right) \end{bmatrix}$$

$$\begin{bmatrix} a\cos(\theta_2)\cos(\theta_1) - b\cos(\theta_2)\sin(\theta_1) - a\sin(\theta_2)\sin(\theta_1) - b\sin(\theta_2)\cos(\theta_1) \\ a\sin(\theta_2)\cos(\theta_1) - b\sin(\theta_2)\sin(\theta_1) + a\cos(\theta_2)\sin(\theta_2) + b\cos(\theta_2)\cos(\theta_1) \end{bmatrix}$$

$$\begin{bmatrix} a\left(\cos(\theta_2)\cos(\theta_1) - \sin(\theta_2)\sin(\theta_1)\right) - b\left(\cos(\theta_2)\sin(\theta_1) + \sin(\theta_2)\cos(\theta_1)\right) \\ a\left(\sin(\theta_2)\cos(\theta_1) + \cos(\theta_2)\sin(\theta_2)\right) + b\left(-\sin(\theta_2)\sin(\theta_1) + \cos(\theta_2)\cos(\theta_1)\right) \end{bmatrix}$$

$$\begin{bmatrix} a\cos(\theta_2+\theta_1) & -b\sin(\theta_2+\theta_1) \\ a\sin(\theta_2+\theta_1) & +b\cos(\theta_2+\theta_1) \end{bmatrix}$$

Now Rotate by $(\theta_2+\theta_1)$

$$\begin{bmatrix} \cos(\theta_1+\theta_2), & -\sin(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2), & \cos(\theta_1+\theta_2) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a\cos(\theta_1+\theta_2) - b\sin(\theta_1+\theta_2) \\ a\sin(\theta_1+\theta_2) + b\cos(\theta_1+\theta_2) \end{bmatrix}$$

<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

COMPLETION 2 (10 marks)

Find the inverse of $R(\vartheta)$ and compare the result to $R(-\vartheta)$. How is this related to our results in Completion 1? Note this was a superchallenge in the 121 lab.

<><><><><><><><><><><><><><><><><><><><><><><>

$$R(\theta) = \begin{bmatrix} \cos(\theta), & -\sin(\theta) \\ \sin(\theta), & \cos(\theta) \end{bmatrix} \qquad R(-\theta) = \begin{bmatrix} \cos(-\theta), & -\sin(-\theta) \\ \sin(-\theta), & \cos(-\theta) \end{bmatrix}$$

$$R(\theta)^{-1} = \frac{1}{\cos(\theta)\cos(\theta) + \sin(\theta)\sin(\theta)} \begin{bmatrix} \cos(\theta), & \sin(\theta) \\ -\sin(\theta), & \cos(\theta) \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta), & \sin(\theta) \\ -\sin(\theta), & \cos(\theta) \end{bmatrix}$$

Because $\cos(\theta) = \cos(-\theta)$, and $\sin(-\theta) = -\sin(\theta)$

we can see that $R(\theta)^{-1} = R(-\theta)$.

The inverse of the rotation matrix with a positive angle is the same as the rotation matrix of the negative angle.

<><><><><><><><><><><><><><><><><><><><><><><>

5.4 The Reflection Matrix

To reflect a point across a line through the origin and another point (x,y) we use

$$I=\frac{1}{(x^2+y^2)}\begin{pmatrix} x^2-y^2 & 2xy \\ 2xy & y^2-x^2 \end{pmatrix}$$

COMPLETION 3 (5 marks)

Show that for reflecting across the x axis this reduces to

$$I=\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

<><><><><><><><><><><><><><><><><><><><><><><>

$I = \frac{1}{x^2+y^2}\begin{bmatrix} x^2-y^2, & 2xy \\ 2xy, & y^2-x^2 \end{bmatrix}$ Reflection along x, take $x=1$ $y=0$

$I = \frac{1}{1}\begin{bmatrix} 1, & 0 \\ 0, & -1 \end{bmatrix} = \begin{bmatrix} 1, & 0 \\ 0, & -1 \end{bmatrix}$

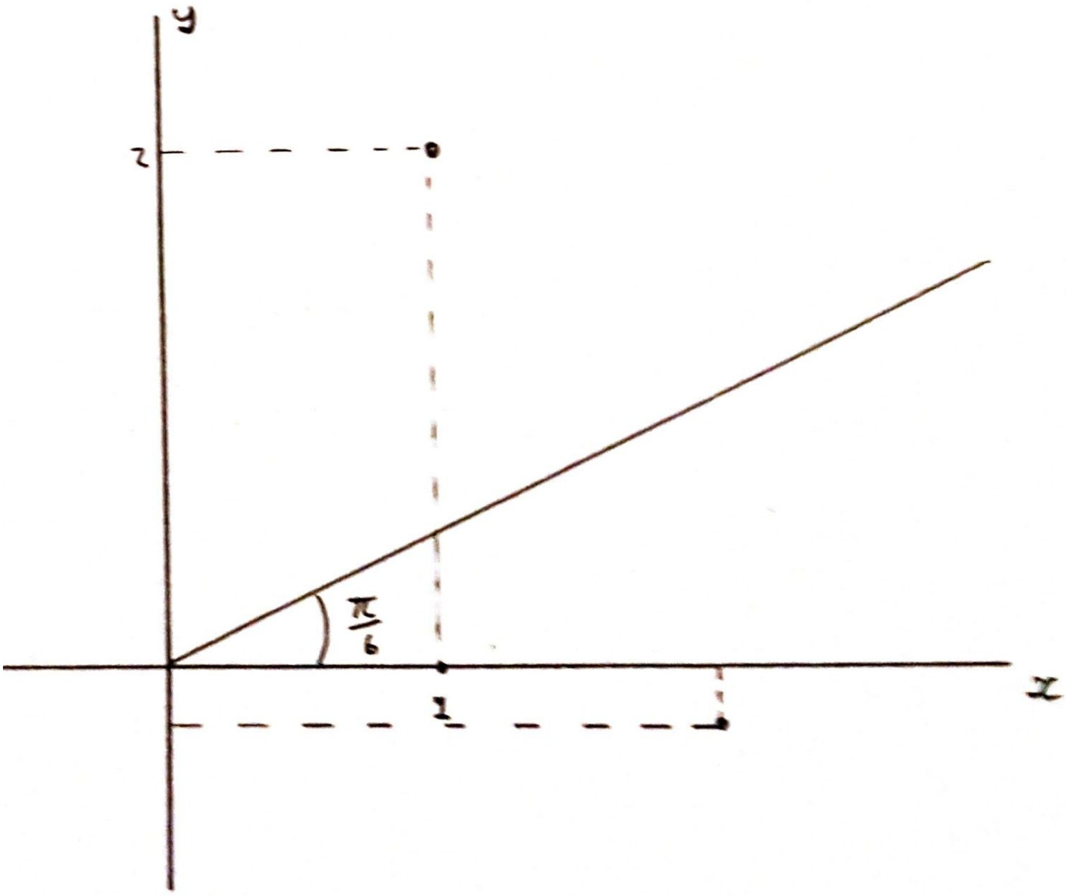
<><><><><><><><><><><><><><><><><><><><><><><>

COMPLETION 4 (10 marks)

Reflect the point (1,2) across a line at pi/6 radians above the x axis. Make a quick sketch of the point before and after reflection. Is the operation behaving as expected?

<><><><><><><><><><><><><><><><><><><><><><><>

```
reflect(math.pi/6, 1,2)

def reflect(theta, x1, y1):
    x = 1
    y = math.tan(theta)
    I = 1/((x**2)+(y**2))

    R = ((x**2)-(y**2), 2*x*y,2*x*y,(y**2)-(x**2))
    P = (x1, y1)

    R = matrix(R).reshape(2, 2)
    P = matrix(P).reshape(1, 2)

    R = I*R
    PR = P*R
    print PR

[[ 2.23205081 -0.1339746 ]]
```



Yes the function is acting as expected, the points (2.2, -0.13) are very close to what i approximate in the drawing above.

<><><><><><><><><><><><><><><><><><><><><><><>

5.4 Combining Operations

The stretch matrix below with stretch an object along the x axis by factor s.

$$S=\begin{pmatrix} s & 0 \\ 0 & 1 \end{pmatrix}$$

CHALLENGE 1 (10 marks)

Stretch your rectangle along the y direction by a factor of 2, rotate it clockwise by 45 degrees, and shift it to the right by 3. Include your python code and make rough sketches of the rectangle before and after the transformations. Hint: translation can be accomplished by adding a vector or by using a 3x3 matrix with homogeneous coordinates (CGRA students). Did you get what you expected?

<><><><><><><><><><><><><><><><><><><><><><><>

```
def rotate(theta, point):
    R = (math.cos(theta), -math.sin(theta), math.sin(theta), math.cos(theta))
    R = matrix(R).reshape(2, 2)
    P = point
    return P*R

def stretch(stretch_x, stretch_y, point):
    S = (stretch_x, 0, 0, stretch_y)
    P = point
    S = matrix(S).reshape(2, 2)
    return P*S

def translate(move_x, move_y, point):
    P = point
    T = (move_x, move_y)
    T = matrix(T).reshape(1, 2)
    return (P+T)

def challenge1(point, stretch_x, stretch_y, theta, move_x, move_y):
    P = point
    P = stretch(stretch_x, stretch_y, P)
    P = rotate(theta, P)
    P = translate(move_x, move_y, P)
    return P

P1 = (1,1)
P2 = (4,1)
P3 = (1,3)
P4 = (4,3)

P1 = matrix(P1).reshape(2,1)
P2 = matrix(P2).reshape(2,1)
P3 = matrix(P3).reshape(2,1)
P4 = matrix(P4).reshape(2,1)

print challenge1(P1, 1, 2, math.pi/4, 3, 0)
```

[[5.12, 0.71]]

```
print challenge1(P2, 1, 2, math.pi/4, 3, 0)
```

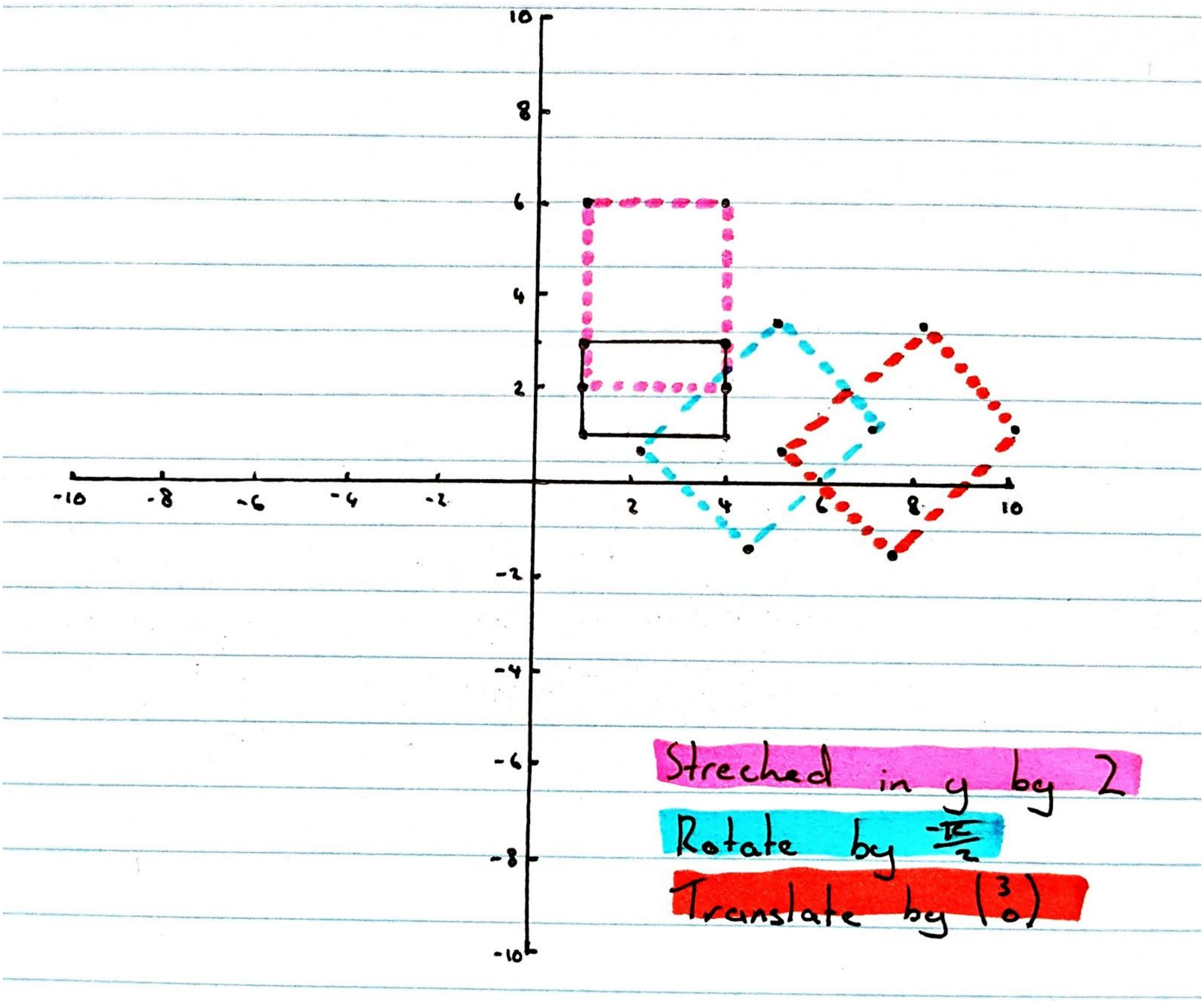
[[7.24, -1.41]]

```
print challenge1(P3, 1, 2, math.pi/4, 3, 0)
```

[[7.95, 3.54]]

```
print challenge1(P4, 1, 2, math.pi/4, 3, 0)
```

[[10.07, 1.41]]



Yes this was what was expected.

<><><><><><><><><><><><><><><><><><><><><><><>

CHALLENGE 2 (10 marks)
When you rotated your rectangle (CORE 2) it rotated around the origin. Now make it rotate around its own centre.

<><><><><><><><><><><><><><><><><><><><><><><>

```
def rotate(theta, point):
    R = (math.cos(theta), -math.sin(theta), math.sin(theta), math.cos(theta))
    R = matrix(R).reshape(2, 2)
    P = point
    return P*R

def challenge2(point, rotate_about, theta):
    P = point - rotate_about
    P = rotate(theta, P)
    return (P + rotate_about)

P1 = (1,1)
P2 = (4,1)
P3 = (1,3)
P4 = (4,3)
RP = (2.5, 2) #the point we are rotating about, here it is the centre of the rectangle

P1 = matrix(P1).reshape(1,2)
P2 = matrix(P2).reshape(1,2)
P3 = matrix(P3).reshape(1,2)
P4 = matrix(P4).reshape(1,2)
RP = matrix(RP).reshape(1,2)


print challenge2(P1, RP, math.pi/2)
[[1.5, 3.5]]
print challenge2(P2, RP, math.pi/2)
[[1.5, 0.5]]
print challenge2(P3, RP, math.pi/2)
[[3.5, 3.5]]
print challenge2(P4, RP, math.pi/2)
[[3.5, 0.5]]
```

This code treats the point you wish to rotate about as a vector acting on the point you are rotating about, by taking it away you are now just rotating about the point (0,0), from here we can now do the rotation and then re-apply the vector to get back to the new position.

<><><><><><><><><><><><><><><><><><><><><><><>

APPENDIX

Using Python to Work With Matrices

To launch Python just get a terminal and type python.

The first step is to import the maths libraries you will need:

```
from numpy import matrix
from numpy.linalg import inv
```

To set up a matrix with nine elements and then display it, for example, type

```
A = (1, 2, 2, 4, 3, 2, 3, 2, 1)

A = matrix(A).reshape(3, 3)

print A
```

You should see

$$\begin{pmatrix} 1 & 2 & 2 \\ 4 & 3 & 2 \\ 3 & 2 & 1 \end{pmatrix}$$

To create a column vector use, for example,

```
b = (1, 2, 5)

b=matrix(b).reshape(3, 1)
```

To find the inverse of a matrix use

InvA = inv(A)   this creates a matrix named InvA which is the inverse of A.

To multiply two matrices just use * as usual. For example, type

```
print InvA*A
```

you should see the identity matrix (with some slight rounding errors – keep that in mind!)