Practical Exercise
PI 1.1 .- Initiation to Matlab and linear algebra
Related to lecture **Rigid Body Transformations**

Description: Much of our work in this class will be done using Matlab. The goal of this exercise is to get you familiar with Matlab (or just refresh if you already are comfortable with it). Here we will work on the basics of inputting matrices, manipulating them, and doing some plotting.

Workload: 2 hours
Programming platform: Matlab
THIS IS AN OPTIONAL EXERCISE AND YOU SHOULD TRY IT ON YOUR OWN OUT OF THE LAB SESSIONS. IT IS NOT EVALUATED. YOU DON'T NEED TO SUBMIT ANYTHING.

Part 1. Some matrix commands
In Matlab matrices can be entered manually, or by using some pre-defined Matlab functions. Here are some examples. If you have not used Matlab very much, you will want to work through these examples of the functions.

```
A=[1, 2, 3; 1, 4, 7; 2, 6, 1; 4, 5, 0]
% This creates a 4 x 3 matrix; each row is separated by a semi-colon; entries within a row are separated by a comma (like vectors)

B=zeros(4)                          % This creates a 4 x 4 matrix of all zeroes
C=ones(3,4)                         % This creates a 3 x 4 matrix of all ones
D=eye(5)                            % This creates a 5 x 5 identity matrix
E=diag([1,2,3])                     % 3 x 3 diagonal matrix with 1,2,3 on diagonal
F=5*C                               % scalar multiplication
G=[3, 4, 5; 1, 6, 3; 2, 2, 9; -1, 0, 17]
H=G+A                               % matrix addition
J=G-A                               % matrix subtraction
K=G.*A                              % multiplies the i-jth entry of each matrix
L=A'                                % transpose of A
M=G*L                               % matrix multiplication
N=L*G                               % matrix multiplication
O=[1, 2, 3; -1, 6, 0; 6, 2, 14]
P=inv(O)                            % inverse of O
A=det(P)                            % determinant of P
```

Part 2. Vectors and matrices
Show using index notation that **a** × **b** is perpendicular to both **a** and **b**.

[Hint]: $(a \times b)_i = \varepsilon_{ijk} a_j b_k$, where ε_{ijk}, the alternating tensor, is equal to 1 when i, j, k are in right-hand coordinate system, right-hand rule order (i.e., (1,2,3), (2,3,1), (3,1,2)); is equal to −1 when i, j, k are in left-hand coordinate system, right-hand rule order; (i.e., (2,1,3), (3,2,1), (1,3,2)); and is equal to 0 otherwise (i.e., if any two indices are the same).

Show that $|a \times b| = |a||b| \sin \phi$, where ϕ is the angle between a and b.

Hint: use the formulation from part (a).

Show that $a \cdot (b + c) = a \cdot b + a \cdot c$
Show that $a \times (b + c) = a \times b + a \times c$

2.2 Show that the transformation matrix, A for a rotation of a vector **a** in 3-space to a new vector **b** = A· **a**, satisfies $A^T A = I$ and is thus orthogonal.

Show that vector length is preserved under rotation.
Show the moved axis are orthonormal (norm=1 and the angle between them is 90º)

Part 3. Plotting Example
First, let's set up a matrix that will plot an object. Let the object be represented by a 2x9 matrix of 9 (x, y) points.

```
object = [0 1 3 3 2 1 1 1 3; 0 0 0 1 2 1 0 1 1]
```

To see what the object is, set up the axes for a plot:

```
hold on                          %This keeps the axis properties for subsequent plots
axis([-pi, pi, -pi, pi])         % This sets the axes to -_ to _.
                                 % Note pi is pre-defined in Matlab as 3.14159265358979.
axis('equal')                    % This requires that x and y axis scaling is equal
```

Then plot the matrix:

```
plot(object(1,:), object(2,:));  % plots all x (row 1) and y (row2) points
```

Part 4. Rotation Example
Let's rotate the object clockwise around the origin by multiples of 60^o, using a rotation matrix A to move the points in the object. Define A as

```
A = [cos(pi/3) -sin(pi/3); sin(pi/3) cos(pi/3)]
```

For the 60_ rotation of the object, multiply A by the object matrix:

```
object1 = A*object;
object2 = A*A*object;
object3 = A*A*A*object;
object4 = A*A*A*A*object;
```
Now add these to your plot.
```
plot(object1(1,:), object1(2,:));
plot(object2(1,:), object2(2,:));
plot(object3(1,:), object3(2,:));
plot(object4(1,:), object4(2,:));
```

You could also put all of these commands in an M-file, which is a text file containing Matlab statements. Running this file within Matlab will cause all the statements in the file to be executed.

Part 5. Eigenvalues and Eigenvectors
We can calculate the eigenvalues and eigenvectors of the rotation matrix A within Matlab using the built-in functions. Remember that the rotation matrix A can be decomposed into its eigenvalues and eigenvectors by the following:

$$A = X\lambda X{-1}$$

where X is the matrix of eigenvectors and λ is a diagonal matrix containing the eigenvalues. This is a very useful decomposition, especially to speed up mathematical operations such as repetitive multiplication.

$$A3 = AAA$$
$$= (X\ \lambda X{-1})\ (X\ \lambda X{-1})\ (X\ \lambda X{-1})$$
$$= X\lambda\ (X{-1}X)\ \lambda\ (X{-1}X)\ \lambda X{-1}$$
$$= X\lambda\lambda\lambda X{-1}$$

$= X\lambda^3 X{-}1$

But also to compute the closest rotation matrix from an arbitrary square matrix, and to solve linear systems.

Given A:

A = [cos(pi/3) -sin(pi/3); sin(pi/3) cos(pi/3)]

To find the eigenvalues of A, use the eig function in Matlab

vals = eig(A);

This produces a vector vals with the 2 eigenvalues of A.
-vals = [0.5 + 0.8660i; 0.5 - 0.8660i]

To find the eigenvectors of A, use the eig function with A
[*X, L*] = eig(A);

produces eigenvectors in the columns of X
X = [0.7071 0.7071
-0.7071i 0.7071i]

with the corresponding eigenvalues on the diagonal of the matrix L. Check to see that you can get back to the original matrix A using X * L * inv(X).

Part 6. Scripting
Now write a script function that computes the closest rotation matrix from an arbitrary square matrix;

```
function [R2] = closestrotation(R1)
% Compute the closest rotation matrix

% step 1: check if R1 is square. Return an error message otherwise.

% step 2: compute the Singular Value Decomposition

% step 3: Set the eigen values to the unity. Preserve the sign.
R2 =
```

Now check the function you have programmed and show that R2 is a rotation matrix.