Please also note the general rules for this HW:

1) Your HW report must be self-contained. Ensure all steps and values are included in the report, not just in your code. We won't run your code to verify your results; everything needed should be in the report itself.
2) Similar to previous HWs: You will need to include your code with all the MATLAB formatting.
3) Not including your code will result in you losing points.

I. Attitude Determination

A. Extracting Body Frame unit vectors

Consider a spacecraft equipped with a star sensor. After exiting a safe-mode operation, the spacecraft needs to recompute its orientation with respect to the inertial frame. The star sensor acquires the following image (attached): **Starfield.png**. The onboard computer is able to identify 5 distinct bright stars in the image: labeled as **Star1, Star2, Star3, Star4, Star5** in the image **Starfield_w_Labels_sameSize.png**.

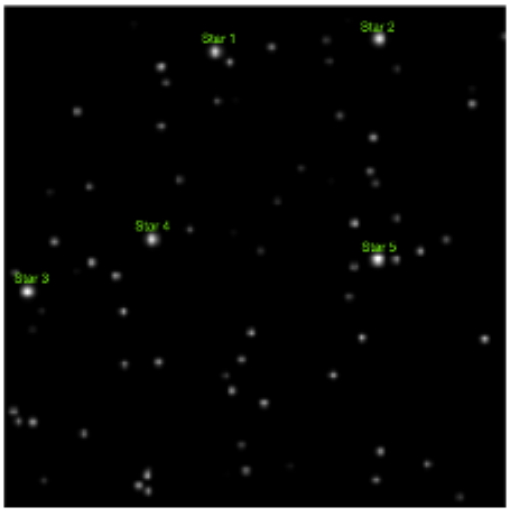

Fig. 1. Star field with Labels

This question will introduce you to some basics of image processing to extract the body-frame unit vectors locating the stars. Follow the steps below to obtain this information. Show the output of each of the steps in your report.

1) **Step 1:** Use the following psueudo code to view the StarField Image:

```
grayImg = imread('StarField.png');
imshow(grayImg)
```

   You can replace the filename to view the labeled file as well.

2) **Step 2:** There are multiple stars in this image. So, our next step will be to extract all of the stars in the image using a thresholding function:

```
binaryImg = imbinarize(grayImg, 'adaptive'); % Adaptive thresholding
imshow(binaryImg)
```

3) **Step 3:** We will now extract the centroid of each of the stars in the image. This will provide you the **x** and **y** pixels of the centroid of all stars in the image.

```
stats = regionprops(binaryImg, 'Centroid');
```

```
centroids = cat(1, stats.Centroid); % Extract x and y pixel coordinates
```
Let us denote these centroids as C

4) **Step 4:** Now, out of all the star centroids, we need to extract those that belong to the 5 stars of interest. In order to do so, open the Labeled star-field image in MATLAB, zoom in on the star labeled **Star1**, and use the *"data tips"* tool to select the center of the star of interest. You will do this for all five stars. Store your result in the form
```
Star1 = [X,Y];
Star2 = [_,_];
Star3 = [_,_];
Star4 = [_,_];
Star5 = [_,_];
```
The "X,Y" data you get from manual selection will not correspond to the exact centroid. So, find the true centroid of the star using a RSME type error between your selected X-Y coordinate and the centroids extracted in the previous step. The centroid of interest will satisfy

$$r_{\text{centroid}} = \min(\sqrt{(C_x - X_{\text{selected}})^2 + (C_y - Y_{\text{selected}})^2}$$

where, $r_{\text{centroid}} = [x_{\text{centroid}}, y_{\text{centroid}}]$
Store the five centroids in a variable (say)
```
Cents =[Cen_S1;
        Cen_S2;
        Cen_S3;
        Cen_S4;
        Cen_S5];
```

5) **Step 5:** You can verify if you have computed the correct centroids by re-projecting them onto the image.
```
imshow("StarField.png")
hold on
plot(centroids(:,1),centroids(:,2),'.r')
hold on
plot(Cents(:,1),Cents(:,2),'og')
```
The red dots are the location of all centroids and the green circles are for the five stars of interest. Do the green circles match the labeled stars? They should, else, revisit step 4.

6) **Step 6:** What you have extracted so far are the pixel coordinates. However, we need the unit vectors pointing to each of the stars.
For images, the pixel coordinate (0,0) is at the top left of the image. However, our physical coordinate system has the center at the origin. So, we must correct for this. Furthermore, the relation between the pixel coordinates and the physical coordinates is provided by the field-of-view of the camera which is related to the focal length of the camera.
The relation between the pixel coordinates and the physical coordinates is given as:

$$P = \left(r_{\text{centroid}} - \frac{W}{2}\right) \times \tan\left(\frac{FOV}{2}\right)$$

where P is the physical coordinate, W is the width of the image, and FOV is the field of view of the camera. Take the FOV as 4 degrees.

7) **Step 7:** Note that we still don't have a 3D unit vector. The physical coordinate is still 2D. This is because we are analyzing an image, so the z coordinate (or depth) information is lost. A simple trick is to append the z coordinate to be 1. This makes the 3D location of the star as

$$[P_x, P_y, 1]$$

You can now normalize this vector to be a unit vector denoting the body-frame location of the star of interest.

B. Inertial frame unit vectors

The inertial frame unit vectors for the five stars are given as follows. Store these in your code.

```
Star 1= [-0.921069884293268  -0.342599924017704   0.185082577005643]
Star 2= [-0.476980639452282  -0.711962047538100   0.515363476056510]
Star 3= [-0.496592767571065   0.816782636490539  -0.293703503424244]
Star 4= [-0.736236774114155   0.676644212846351   0.010393347079969]
Star 5= [ 0.304668730748568  -0.299446239108458   0.904161995655567]
```

C. Attitude Estimation using TRIAD

Now that you have extracted Body frame components: $\mathbf{b}_i$ and you have been given inertial frame components $\mathbf{r}_i$, you may choose any two measurements to implement the TRIAD method of attitude estimation.

1) Write down the DCM obtained from the TRIAD method.
2) Since we don't really know the ground truth for the measurements to check if we have found the correct DCM, a good way to check the error is to compute the projection error. For each of the 5 stars, compute the projection error as

$$\mathbf{e}_i = \mathbf{b}_i - C_{BN}\mathbf{r}_i$$

You can then stack all of the e_i in a long vector and take the norm of the resulting long vector. This is a measure of the accuracy of your TRIAD method.

D. Attitude Estimation using OLAE

Repeat the steps above by implementing the OLAE method.

E. Attitude Estimation using Davenport q-method

Repeat the steps above by implementing the Davenport's q- method.
Comment on which of the three methods provides the most accurate answer. Why do you say so?

II. Attitude Dynamics: Analytical

Let us derive the general rotational equations of motion for a rigid body free of any external torques. The attitude coordinates are chosen to be the (3-2-1) Euler angles, also known as the yaw, pitch, and roll angles (ψ, θ, ϕ). Assume the rigid body has a coordinate system $\mathcal{B}$ attached to it that is aligned with the principal inertia axes, and let $\mathcal{N}$ be an inertial reference frame. For the free motion of a rigid body, the angular momentum vector $\mathbf{H}$ will remain constant. Using a trick due to Jacobi, we can therefore always align our inertial space unit axes $\hat{\mathbf{n}}_i$ such that $\hat{\mathbf{n}}_3$ is aligned with $-\mathbf{H}$:

$$\mathbf{H} = {}^{\mathcal{N}}\mathbf{H} = -H\hat{\mathbf{n}}_3$$

1) Using the Euler angle and DCM relations from attitude kinematics, and the definition of angular momentum, derive the equations of motion for the roll, pitch, and yaw angles $(\dot{\psi}, \dot{\theta}, \dot{\phi}\ )$ as a function of the angular momentum H, the Euler angles (ψ, θ, ϕ), and the moment of inertia, I_1, I_2, I_3.

2) Once the general equations of motion are derived, study a particular case of these equations in which the rigid body is axisymmetric. Assume $I_2 = I_3$. What do the Euler angle rates simplify to? Note, $\dot{\psi}$ is typically referred to as the precession rate and $\dot{\phi}$ is referred to as the spin rate. Show that the precession and spin rates are a constant?

3) Let us further take the body spin rate about the axis of symmetry as $\Omega = \omega_1$. Write an expression for Ω in terms of the angular momentum, moment of inertia and Euler angles. Is this value a constant? Why?

4) Using the definition of Ω, rewrite the expressions for the rate of change of yaw and roll. Write an analytic expression for the evolution of the Euler angles for this axisymmetric rigid body.

III. Attitude Dynamics: Numerical

Modern mobile phones contain a three-axis microelectromechanical system (MEMS) gyroscope, capable of taking accurate measurements of the angular velocity along the three principal axes of the phone with a sampling rate of 100 Hz or better (as shown in class). If the phone is tossed in the air, then, neglecting air resistance, it is in free rotation (torque-free rotation) with respect to its center of mass, and the phone's gyroscope can be used to record the rotational dynamics.

Consider the motion of the phone whose body frame is assumed to be aligned with the principal moments of inertia, as shown in the figure below:

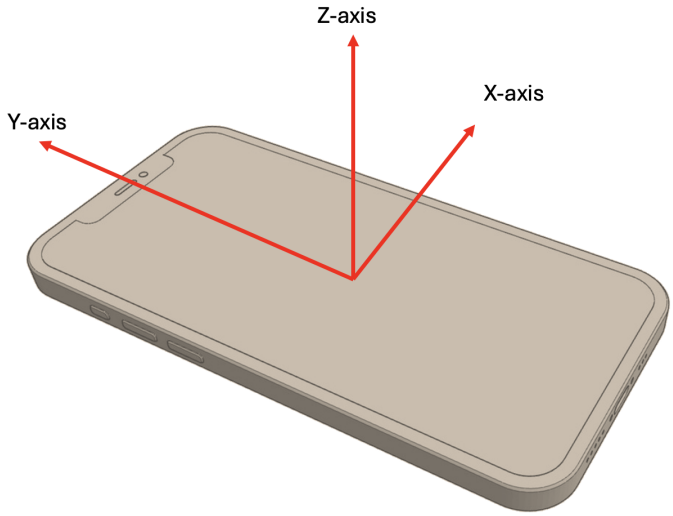

Fig. 2. Principal axis of the phone - Center of Mass is at origin

Your tasks are as follows. Provide sufficient information to clearly convey your thought process, including detailed explanations for each step. Substantiate all statements with appropriate mathematical formulas, as insufficient information will be penalized.

A. *Data handling and preliminaries*

1) The specifications of the phone is obtained from the apple website: The Apple iPhone has height of 5.78" (146.7 mm), width of 2.82" (71.5 mm), depth of .29" (7.4 mm), and weighs 6.66 oz (189 g). The phone can be approximated as a rectangular cuboid. Calculate the approximate principal moments of inertia assuming the rectangular cuboid shape.

2) For the standard phone shape h $>$ w $>$ d. Identify the minimum, maximum, and intermediate moment of inertia axes. Also write the expressions for the total Energy and the Angular momentum of the rigid body.

3) Now, consider three experiments in which the mobile phone is tossed in the air, and then caught again. In each case, the aim is to set the phone to spinning predominantly about one of the principal axes. A time history of the angular velocities is provided to you in a .mat file for each of the Experiments. Load the .mat file into MATLAB and plot the angular velocities as a function of time. Use 3 subplots to plot the 3 angular velocities one below the other so that their time stamps are aligned. For each of the experiments, identify the distinct time intervals when the phone is in the air by the smooth variation in the angular velocities. Denote them as dashed lines in your plot and label them. You may use the following Psudo-Code to load the data:

```
load(filename)
wx = AngularVelocity.X;
```

```
wy = AngularVelocity.Y;
wz = AngularVelocity.Z;
T  = AngularVelocity.Timestamp;
timeVec = datetime(T, 'InputFormat', 'dd-MMM-yyyy HH:mm:ss.SSS');
t = seconds(timeVec - timeVec(1));
```

B. Now the fun parts

1) By looking at the angular velocity data, associate each experiment with the rotation axis. Provide reasonings for your answer. For example, your answer should be of the form:
Experiment 1 represents a rotation about the "major / minor / intermediate" axis of inertia. This is because, ...

2) For each experiment, calculate the total energy, and the magnitude of the angular momentum vector. Plot these as a function of time. Do you find that for the duration of when the rigid body in in the air, these quantities are conserved? *Note that these are true data collected from a real sensor, so there is bound to be some noise.* How well do you think the torque-free assumption is valid? Why?

3) Identify the time interval for when the rigid-body is in the air. Now, using the value of the angular velocity at the initial point of that interval as an initial condition, propagate the Euler's equations of motion for a rigid body in torque-free environment for that time interval. Overlay the measured angular velocity (from the experiment) and that obtained through the propagation of the Equations of motion. How closely do they match? Discuss your results for each experiment separately. Specifically, discuss your observations for the Experiment corresponding to the rotation about the intermediate axis of inertia.

4) Recall the discussion of the inertia ellipsoid and the momentum sphere in the class. A slightly different way of writing the energy ellipsoid is as follows:

$$\frac{\omega_1^2}{\frac{2T}{I_1}} + \frac{\omega_2^2}{\frac{2T}{I_2}} + \frac{\omega_3^2}{\frac{2T}{I_3}} = 1$$

Use the following Pseudo code to plot this ellipsoid:

```
% Compute semi-axes of the ellipsoid from equation above
a =
b =
c =

% Create the ellipsoid
[x, y, z] = ellipsoid(0, 0, 0, a, b, c);

surf(x, y, z, 'FaceColor', [0.5, 0.5, 0.5], 'EdgeColor',...
'y', 'FaceAlpha', 0.5); % Uniform gray
```

Overlay a 3D plot of the angular velocity measured for each experiment. On the plot, draw the major, minor, and intermediary axis of inertia, i.e. b_1, b_2, b_3. Describe what you observe? Is this correct? Can you explain it? Does your 3D plot of the angular velocity match that on the energy ellipsoid axes? Can you identify the stable and unstable motions from this Poinsot construction?