Homework 3: Flying Box Version 2
(Updated to Improve Clarity)

MEAM 520, University of Pennsylvania
Katherine J. Kuchenbecker, Ph.D.

September 17, 2013

This assignment is due on **Friday, September 20, by midnight (11:59:59 p.m.)** Your code should be submitted via email according to the instructions at the end of this document. Late submissions will be accepted until Sunday, September 22, by midnight (11:59:59 p.m.), but they will be penalized by 10% for each partial or full day late, up to 20%. After the late deadline, no further assignments may be submitted.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you write down should be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. If you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

You may do this assignment either individually or with a partner, according to your personal preference. Read the assignment to decide which option is right for you. If you do this homework with a partner, you may work with anyone you choose, even someone with substantial MATLAB experience. If you are looking for a partner, consider using the "Search for Teammates!" tool on Piazza.

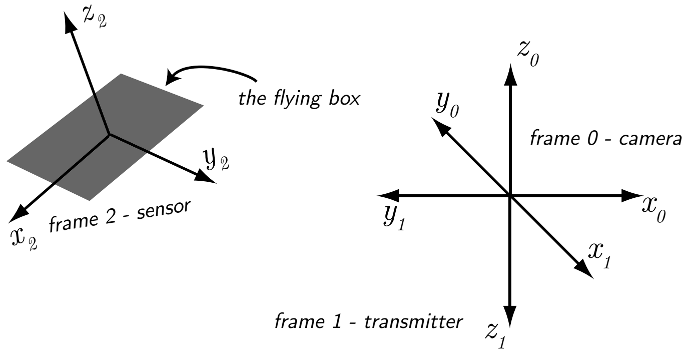
If you are in a pair, you should work closely with your partner throughout this assignment, following the paradigm of pair programming. You will turn in one MATLAB script for which you are both jointly responsible, and you will both receive the same grade. Please follow these pair programming guidelines, which were adapted from "All I really need to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000:

- Start with a good attitude, setting aside any skepticism and expecting to jell with your partner.
- Don't start writing code alone. Arrange a meeting with your partner as soon as you can.
- Use just one computer, and sit side by side; a desktop computer with a large monitor is better for this than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (using the mouse and keyboard or recording design ideas) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every thirty minutes, *even if one partner is much more experienced than the other.* You may want to set a timer to help you remember to switch.
- If you notice a bug in the code your partner is typing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Recognize that pair programming usually takes more effort than programming alone, but it produces better code, deeper learning, and a more positive experience for the participants.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.

System Description

Professor Kuchenbecker's research lab owns a 3D Guidance TrakSTAR system made by Ascension Technology Corporation. It includes a magnetic transmitter that generates a strong magnetic field in a spherical region that has a radius of about 1 meter. The other main component of this motion tracking system is a small magnetic sensor that connects back to the base station via a wire. At each instant in time, the TrakSTAR's hardware and software analyze the magnetic field detected by the sensor and use this information to calculate the position and orientation of the sensor with respect to the transmitter. The TrakSTAR is connected to a personal computer, and its software enables the user to record the movement of the sensor over time. The readings are saved to a file that can be analyzed at another time.

Professor Kuchenbecker used tape to attach the magnetic sensor to the bottom of a rectangular foam box that is about 8" × 6". The long axis of the sensor was aligned with the long axis of the box, and the sensor was centered on the box face. For this assignment, you can treat the sensor as though it was flat against the box, neglecting any small misalignments. The readings from the sensor reveal the position and orientation of a coordinate frame that we attach rigidly to the box. Its origin is at the center of the sensor. Its x-axis points straight out through the front of the box, its y-axis points to the box's left, and its z-axis is perpendicular to the box's main face. As illustrated in the diagram below, we name this frame 2.



We also place two frames at the center of the transmitter. They have the same origin but different orientations. The transmitter's frame is frame 1. It has x positive straight out of the page, y positive to the left, and z positive down. This frame orientation is defined by the TrakSTAR manufacturer, but it's unconventional because z points down. To make the motion easier to visualize and easier to graph in MATLAB, we define frame 0 to align with the camera's view, with x positive to the right, y positive straight back, and z positive up. Frames 0 and 1 are stationary, while frame 2 moves around and reorients with the motion of the box.

When the TrakSTAR records data, it stores six numerical values at each point in time. The first three are `x`, `y`, and `z`, the position of the origin of frame 2 (sensor) expressed in frame 1 (transmitter); this vector could be called $\mathbf{v}_2^1$. These positions are measured in inches. The other three values recorded by the TrakSTAR are `a`, `e`, and `r`, three angles expressed in degrees. The value `a` is the azimuth angle in degrees, denoting a rotation about the z axis. Similarly, `e` is the elevation angle in degrees, a rotation around the y axis, and `r` is the roll angle in degrees, a rotation around the x axis. Unfortunately, the TrakSTAR documentation is ambiguous about whether these three rotations should be made about the sensor's axes or the transmitter's axes, and it isn't clear about the order in which they should be performed.

Professor Kuchenbecker used the system described above to record a particular movement of the foam box. In addition to saving `x`, `y`, `z`, `a`, `e`, and `r` in a computer file, she recorded a video of the motion. You can view this movie here: `http://www.youtube.com/watch?v=FTC83piKiX0`

Task Description

Your task is to update a provided MATLAB script so that it correctly animates the recorded motion of the flying block. The animated box should move in the same way as the box in the video.

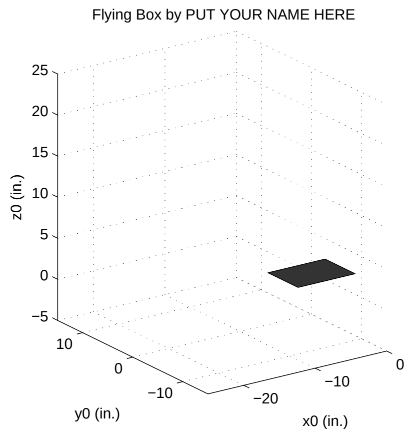
Download the following items from the Course Page / Resources / Homework section of Piazza:

- `flying_box_starter.m` – This MATLAB script loads the data into the workspace, defines the animation parameters, sets up a `for` loop to process the data, and graphs a flat gray rectangle that represents the box.
- `flying_box.mat` – This MATLAB data file contains the time history of the position and orientation of the flying box recorded during the movie.
- `flying_box.mov` – the movie, in case you need to download it for offline viewing.

Rename the starter file to `flying_box_yourpennkey.m` or `flying_box_pennkey1_pennkey2.m` so that everyone will have a unique file name. Your PennKey is the first part of your Penn email address.

Set the value of the `studentNames` variable at the top of the file to include your full name or the full names of both teammates.

Run the starter code and watch what happens. As provided, the first step of the animation should look like the plot at the right, and then the back corner of the box should move straight up so you can see movement.



Update the code between the two lines of stars to calculate the coordinates of the four corners of the box in the frame of the camera (frame 0). You should start with the provided coordinates of these corners in the sensor's frame (`pa2`, `pb2`, `pc2`, and `pd2`), and you should store your four final answers as `pa0`, `pb0`, `pc0`, and `pd0`. Achieving the correct movement will require you to calculate and apply the transformation T_2^0. Do not modify the angles themselves (by adding constants or changing their signs). Instead, change only the order in which you apply the rotations. You should write out the calculations before starting to program.

Your calculations may not use any built-in or downloaded functions dealing with rotation matrices, homogeneous transformations, Euler angles, roll/pitch/yaw angles, or related topics. Instead, you must type all your calculations yourself, using only low-level functions such as `sind`*,* `cosd`*, and vector/matrix math.*

Submitting Your Code

Follow these instructions to submit your code:

1. Start an email to `meam520@seas.upenn.edu`
2. Make the subject *Homework 3: Your Name* or *Homework 3: Your Name and Your Teammate's Name*, replacing *Your Name* and *Your Teammate's Name* with the appropriate full names.
3. Attach your correctly named MATLAB script to the email. Please do not put it in a zip file or include any other attachments.
4. Optionally include any comments you have about this assignment and the experience of pair programming if you worked with a teammate.
5. Send the email.