ASEN 3128 LAB 1: Simulate EOM

- Assigned: Friday 22 Jan
- Due: Friday 5 Feb, before Lab Section starts

For all assignments students will work in groups of two to four students. Groups are determined by the instructors. A single assignment should be submitted for the group.

OBJECTIVES

- Introduce good programming habits.
- Review Newton's 2nd Law.
- Practice using Matlab to perform basic calculations and simulate dynamics.

BACKGROUND

Basic Program/Function Requirements

Good programming skills enable a student to create software that can be reused with minimal effort. It is tempting to write computer code, especially for a course or a short project, while assuming that you will remember what you did and how the code performs. This mentality limits your ability to share code with others and even to reuse your own software at a later time. A few basic procedures and additional information added up front can save significant time in the future. In this course we expect students to use a few simple coding practices on assignments.

In particular, the following practices should be followed at all times:

1. **File Headers:** All M-files developed for this course will have the same header information. The header should have the students names, the course name and number, the file name, and the date the file was created. For example:

```
% Imma A. Student
% ASEN 3128
% AirRelativeVelocityVectorToWindAngles.m
% Created: 1/22/2021
```

   For Simulink models you must add a text box in the corner of the file and include the same information.

2. **Function Descriptions and Comments:** Every software function will have text that describes the inputs to the function, the outputs, and a basic description of the approach. Matlab has the additional capability to display comments in the M-file immediately after the function declaration when the help function is used. For example:

```
function wind_angles = AirRelativeVelocityVectorToWindAngles(velocity_body)
%
% Inputs:       velocity_body = column vector of aircraft air-relative velocity in
%                                 body coordinates
%                             = [u,v,w]'
%
% Outputs:      wind_angles = [speed beta alpha]'
%               speed = aircraft airspeed
%               beta = side slip angle
%               alpha = angle of attack
%
% Methodology: Use definitions to calculate wind angles and speed from velocity vector
```

3. **Descriptive Variable and Function Names:** Variable and function names will clearly and concisely describe what they describe or the action they perform.

   For variable names it is common to use lowercase letters with the underscore to separate words. For example, use velocity_body to describe the aircraft velocity in body coordinates and never just use v (which could be confused with other variables like the speed).

   For function names it is common to use capital letters for each word instead of the underscore. For example, use a name something like: AirRelativeVelocityVectorToWindAngles.

4. **Functional decomposition:** Any procedure that needs to be performed multiple times should be encapsulated into a function. There are no hard and fast rules for creating and using functions. Functions can be nested into other functions to build complex software.

Simulating Dynamic Systems in Matlab

We will see shortly that the aircraft dynamic equations of motion are highly nonlinear and cannot be solved analytically. Instead, Matlab can be used to integrate the ordinary differential (vector) equation numerically. Matlab provides a variety of solvers for ordinary differential equations. One particular function is

[TOUT,YOUT] = ODE45(ODEFUN,TSPAN,Y0)

which takes as input the name of a function that returns the derivative of the state vector of the equation, the time span of the simulation, and the initial state vector. It returns the time vector and an array of the output vector. See Matlab Help for more information.

The ODEFUN has the form $\dot{y}$ = ODEFUN(t,y), returning the derivative $\dot{y}$ of the state y. For example, ODEFUN could be used to return the derivative $\dot{y} = 5y + 20\sin 4t$ since the equation is the form needed for ODEFUN. Note, the actual function passed into ODE45 can have any name, for example: [tout, yout] = ode45(@MyDerivFunction, [0 20], 5).

We assume students have used ODE45 (or similar function like ODE23) in the past and will not review it here.

PROBLEMS

1. Explore the use of ODE45 for simulating nonlinear dynamic system equations:

   a. Simulate the following set of equations with non-zero initial conditions and duration:

   $$\dot{x} = x + 2y + z$$
   $$\dot{y} = x - 5z$$
   $$\dot{z} = xy - y^2 + 3z^3$$

   Create one figure with three subplots, stacked vertically. Each subplot should contain the plot of one variable component (*x*, *y*, or *z*) versus time. Note, the ODEFUN function you create can return a vector so this problem should be solved as one single run of ODE45.

2. Construct a Matlab simulation of the translational dynamics of a ball moving through the air, where the forces on the body are not a function of the body attitude but include aerodynamic drag (which acts opposite the inertial velocity vector) and gravity (which acts downward). Use this to model a golf ball, with mass of 30 g, diameter 3.0 cm, and coefficient of drag of 0.6.

   a. Create a function

   ```
   function xdot = objectEOM(t,x,rho,Cd,A,m,g,wind)
   ```

   That gives the derivative xdot of the state vector x as a function of time, the state, and the rest of the parameters of the problem. The 6-dimension state vector should include the inertial velocity in inertial coordinates and the inertial position in inertial coordinates

   $$\mathbf{x} = [\mathbf{p}_{\boldsymbol{E}}^{\boldsymbol{E}};\ \mathbf{v}_{\boldsymbol{E}}^{\boldsymbol{E}}]^{\mathrm{T}}.$$

   b. Begin the simulation with an initial position at the origin of the inertial frame, with an initial velocity component of 20 m/s upward and 20 m/s East, and assume the wind is zero. Verify that the results make sense.
   c. How sensitive is the landing location (characterized by a vertical displacement relative to the origin of zero) to horizontal (North) wind, in m of deflection per m/s of wind? Provide plots to support your answer.
   d. If the initial velocity is constrained by a limited kinetic energy (i.e. due to human swing-strength limitations) equal to the case examined above, would longer distance be achieved by using a heavier or lighter golf ball? Provide plots to support your answer.

ASSIGNMENT

This assignment does not require a full lab report. Instead, submit copies of all files and plots generated for the problems. Be sure to add titles to all figures that include both the problem number and a description of the plot, e.g. 2.c Output x versus Time. The assignment will be evaluated based on i.) correct answers; ii.) proper commenting and documenting of code; and iii.) the quality of the figures submitted (e.g. labeling, axis, etc).

All lab assignments should include the Team Participation table and should be completed and acknowledged by all team members. Description of the Team Participation table is provided in a separate document.