ASEN2012 - Coding Challenge 6 (25 pts) - Due Wednesday 12/6 by 11:59 pm
Experimental and Computational Methods in Aerospace Engineering Sciences

---

Key Deliverables:

- 5 "Check your answer" questions in Canvas (**10 pts**)
  - **You only get 3 attempts** (correct answers carry over to next attempt) at the quiz.
  - **There is a 5 minute "cool-down" between attempts.** Use this time to debug and reevaluate your code. Once you compute a new answer, make sure to critically think about it before submitting again.
  - **No additional attempts will be provided for any reason.**
- 1 .zip file containing the following (naming convention lastname_firstname_CC5.zip) (**15pts**):
  - Functioning Matlab code (.m file(s) - if using multiple files, please name driver code main.m)
  - Published code (.pdf file)
  - Flowchart sketch (.pdf file)
- All numeric quiz answers must be **computed** in your code!

The Balloon Problem

We are going to model the **2D** flight of a high altitude balloon being launched from Boulder (elevation 1,624 m) with an initial velocity of 2 m/s in the z direction (vertically) **and will now include an 8 m/s crosswind**! The balloon will have 4 phases of flight:

1. Ascent prior to popping: the balloon is full of helium (Buoyancy force) and ascends (for the balloon $C_D \approx 0.5$) until 2,100 m when the balloon is popped.
2. Ballistic phase prior to parachute deployment: the balloon instantaneously when the altitude is reached. This removes not only the lifting force, but also reduces the mass of the system (since there will be no more helium). Since the balloon was the primary source of drag, and it is now gone, you can assume drag is negligible (**HINT**: only one force is nonzero during this phase).
3. Controlled descent after parachute deployment at 1950 m: On the return back to Earth, once the balloon drops below pop altitude again, a parachute is deployed ($C_D \approx 1.03$), which will add a new drag force (**HINT**: think about the free body diagram and ensure you're applying the drag in the correct direction).
4. The balloon reaches the initial launch elevation and should no longer be moving, with no forces on it for the remainder of the time (**HINT**: determine which values of the state vector should be zero).

The constants required for this problem are the following:

- $m_{payload} = 450$ kg (assume structure and balloon material mass are negligible)
- $r_{balloon} = 17$ m (assume same radius for parachute)
- $\rho_{air} = 1.225$ kg/m^3 (assumed to be constant)
- $\rho_{helium} = 0.1786$ kg/m^3 (assumed to be constant)

- $g = 9.81$ m/s^2 (assumed to be constant)

To solve this problem, you will use the code that you developed for Coding Challenge 5 and change the state vector to include information about the x position and horizontal velocity, v_x. It will be your responsibility to determining the equations required for computing the time derivative of the newly defined state vector. The forces are those associated with gravity, buoyancy, and drag, with equations corresponding to each force defined as follows:

$$F_{gravity} = m_{sys}g$$

$$F_{buoyancy} = \rho_{air}\cancel{V}g$$

$$F_{drag} = \frac{1}{2}C_D A_c \rho_{air} v^2$$

where ρ_{air} is the air density, $\cancel{V}$ the volume of the balloon, C_D is the coefficient of drag, A_c is the cross-sectional area of the balloon, and v is the balloon's **relative** velocity **magnitude**.

For this problem you are *strongly* encouraged to use the unit vector of velocity to determine the heading:

$$\hat{h} = \frac{v_{rel}}{|v_{rel}|}\text{ , where } v_{rel} = v_{balloon} - v_{wind}$$

This heading will be applied to the drag force, such that $\vec{F}_{drag} = -\hat{h}D$, where $D = \frac{1}{2}C_D A_c \rho_{air} v^2$ (v is the relative velocity magnitude).

Step 1: Edit the function for the Constants

In your function called `getConst` that creates a structure called `const` containing all constants given and calculated to solve this problem (masses, volumes, etc.), add in the horizontal crosswind of 8 m/s (in the positive x direction).

Step 2: Edit your function for `ode45` to integrate (eventually)

The inputs should be t and X (your state vector consisting of two positions and two velocities), and `const`. Use `if` and `elseif` statements to determine which stage you are in and calculate m, $F_{gravity}$, $F_{buoyancy}$, and F_{drag}, accordingly to calculate F_{net} and the resulting value of acceleration. You should also determine when you've fallen back to Boulder. You will also need to edit the initial conditions accordingly (you may assume the initial position and velocity in the x direction are 0 m and 0 m/s, respectively). You will also need to make edits to include the heading vector above into the drag equation.

Problem 3: Run `ode45` and plot

Run your Balloon Equation of Motion function! Set up your initial conditions (1×4 column vector), and use a time span ranging from 0 to 150 seconds. Plot the horizontal/vertical position and horizontal/vertical velocity as functions of time (4 total plots) - you may do one of the following:

- use `subplot` to create 2 figures with 2×1 plots
- use `yyaxis` to create 2 figures with 2 axes each

Use `plot3d` to create a three-dimensional plot showing the horizontal and vertical position versus time for the balloon flight.

Quiz Questions

You will be asked to determine the **position** of the balloon when it reaches back to the ground.

You will be asked to determine the **magnitude** of the displacement of the balloon at various times (use `interp1`).

Reflection Question

Use your plots and knowledge of how `ode45` works to critically analyze this problem. With a constant wind applied, was it **required** to use `ode45` to calculate the x displacement? To help answer this question, you may want to create an additional figure of the first 2 seconds of the flight. What do you see?
Please write out the answers to these questions in the comments of your Matlab script. This should be about 1 paragraph in length.

Coding Rubric

	Excellent (100%)	Above Average (80%)	Average (70%)	Below Average (50%)
Requirements and Delivery (2pts)	• Completed 90-100% of the requirements • Delivered on time, and in correct format.	• Completed 80-90% of the requirements • Delivered on time, and in correct format.	• Completed 70-80% of the requirements • Delivered on time, and in correct format.	• Completed <70% of the requirements • Delivered on time, but not in correct format.
Coding Standards (2pts)	• Includes full header* • Excellent use of variables (no global or unambiguous variable naming)	• Includes full header* • Good use of variables (1-2 global or ambiguous variable naming)	• Includes incomp. header* • Fine use of variables (3-5 global or ambiguous variable naming)	• No header • Poor use of variables (many global or ambiguous variable naming)
Documentation (2pts) Comment your code	• Clearly documented • Specific purpose noted for each function and/or section	• Well documented • Specific purpose noted for each function and/or section	• Some documentation • Purpose noted for each function and/or section	• Limited to no documentation
Runtime (1pt)	• Executes quickly, without errors	• Executes without errors, over 1 min runtime	• Executes with warnings/errors	• Does not execute
Efficiency (2pts)	• Easy to understand, and maintain	• Logical, without sacrificing readability and understanding	• Difficult to follow	• Difficult to follow, huge and appears patched together
Figure Quality (2pts)	• Easy to understand, labels and legend present	• Easy to understand, lacks labels and legend	• Difficult to understand, labels and legend present	• Difficult to understand, lacks labels and legend
Reflection Questions (4pts)	• Easy to understand, fully thought out	• Easy to understand, mostly thought out	• Hard to understand, effort made	• Hard to understand, lacking effort

* Header includes author name(s), assignment title, purpose, creation date, revisions (applicable to **group** projects only)