Building A Better Bomb Code

In this exercise you will use what you have learned so far about Fortran to write a program to solve a more "realistic" problem in aerodynamics.

Problem: The assignment is to write a program similar to your previous programs which computed the trajectory of a bomb released from an airplane (Exercises 08 and 09), but taking into into account the drag forces on the bomb as it falls.

Once we include the force of drag we must make some assumptions about the size and shape of the bomb. We will assume that the bomb is a small sack of flour (about one pound or so) which is dropped from a small airplane. (Pilots do this sometimes for entertainment to see who can get closest to a target on the ground.) A good first guess for the altitude and airspeed of this "Flour Bomb" is that it would be dropped from around 200 feet at about 75 knots.

Drag Force: The drag force exerted on an object moving through a fluid (such as air) is generally proportional to a quantity q, called the "dynamic pressure," which is the pressure that would be exerted on the front of the object by bringing a small parcel of the impinging fluid to rest. Since this is simply the conversion of the kinetic energy of the fluid to the potential energy of pressure exerted on the object, the dynamic pressure is given by

$$q = \tfrac{1}{2}\rho v^2 \,, \tag{1}$$

where ρ is the density of the fluid and v is the velocity of the fluid. The drag force on the object is also proportional to the cross sectional area S the object presents to the fluid. There are various ways to define what is meant by the "cross sectional area," but disregarding such distinctions for the moment this means that the drag force, F_D can be written as

$$F_D = C_D \, q \, S \tag{2}$$

where C_D is a dimensionless constant of proportionality known as the "coefficient of drag." Using Eq. (1) then yields

$$F_D = \tfrac{1}{2} C_D S \rho v^2 \,. \tag{3}$$

Thus the drag force on the object is proportional to the cross sectional area of the object, the density of the fluid, and the square of the velocity of (or through) the fluid. Drag force always acts to resist motion, which means that it acts to slow down the object as it moves through the fluid.

Acceleration: The effect of the drag force on the bag of flour as it falls can be computed by applying Newton's law that $\mathbf{F} = m\mathbf{a}$. The acceleration of the bag will be $\mathbf{a} = \mathbf{F}/m$, where $\mathbf{F}$ is the sum of all forces acting on the bag and m is the mass of the bag. It will be very useful to consider the forces (and accelerations) in the horizontal (x) and vertical (y) directions separately. Solving first for the acceleration of the bag in the horizontal direction gives:

$$a_x = -\frac{C_D S \rho v v_x}{2m} \tag{4}$$

where the speed $v \equiv \sqrt{v_x^2 + v_y^2}$. Note that everything on the right hand side is positive except for the minus sign and possibly v_x. Consequently, this acceleration component will always be in the opposite direction to the velocity component and will act to slow the object down. For example, if the velocity component v_x is negative (the bag is moving in the negative x-direction) then the a_x is positive, so that the drag force still acts to slow the object down. In the vertical direction, the force of gravity is negative because it always pulls the object downward, and combining gravity with the drag force gives

$$a_y = -g - \frac{C_D S \rho v v_y}{2m} \tag{5}$$

If the velocity component v_y is negative (which means that the object is falling downward) then the contribution due to the drag force is positive, against the direction of motion. Drag always acts against the direction of motion.

Computer Simulation: We will model the motion of the bag of flour as it falls by taking many small time-steps, each of size Δt, and at each new time we will compute the acceleration on the bag, the new velocity of the bomb, and the the new position of the bomb. We must be careful about the order in which we compute these quantities. We don't want to put a new value in a variable until we no longer need the old value. Let `X` and `Y` be the position of the bomb and `VX` and `VY` be it's velocity in the horizontal and vertical directions. Let `AX` and `AY` be the acceleration of the bomb in the horizontal and vertical directions, and let `BAGMASS` be the mass of the bomb. Then for each time step the acceleration in both directions is given by something like:

```
SPEED = (VX**2 + VY**2)**0.5
AX = -CDRAG*SURF*AIRDEN*SPEED*VX/(2.0*BAGMASS)
AY = -CDRAG*SURF*AIRDEN*SPEED*VY/(2.0*BAGMASS)
AY = AY - GRAVITY
```

Here `AIRDEN` is ρ, the density of the air, `CDRAG` is C_D, the coefficient of drag, `SURF` is S, the cross sectional area of the bag, and `GRAVITY` is g, the acceleration due to gravity (9.81 m/sec^2 or 32.17 ft/sec^2).

Given the accelerations above, the new values of the velocities in both directions are

```
VX = VX + AX*DT
VY = VY + AY*DT
```

Here `DT` is Δt, the size of our time steps. With the new velocities we can compute the new positions:

```
X = X + VX*DT
Y = Y + VY*DT
```

This series of computations is to be repeated until the bomb strikes the ground (`Y` becomes less than or equal to zero) or until the time in the air reaches 30 seconds. If possible, you should use the same variable names given here in your own program, to make it easier for me to read your code.

Terminal Velocity: As the bomb falls under the force of gravity it picks up speed, and therefore the drag force acting upon it increases. Eventually the drag force will match the force of gravity. When this equilibrium is reached the bomb will no longer accelerate. The velocity at which this happens – the velocity at which the drag force exactly matches the force of gravity – is called the "terminal velocity" of the bomb, because it is the final velocity attained by the falling object. (In practice, the forces rarely balance *exactly*.) In the computer simulation, the signal that the terminal velocity has been reached would be that the acceleration is at or near zero.

Coefficient of Drag: The exact value of the coefficient of drag of a bag of flour is unknown. The coefficient of an infinite flat plate should be 1.0, but the coefficient of drag of a perfect cube is $C_D = 1.05$; for a perfect sphere it is 0.47. Presumably the coefficient of drag for a flour bomb is between these last two values, but we cannot know this for sure. Because we do not know the exact value for the coefficient of drag we will compute the trajectory of the bomb for a number of different values of C_D.

If we could drop several flour bombs of known size and mass from an airplane at a known altitude and airspeed, then measuring where the bomb lands would let us estimate from our model what is the actual coefficient of drag of a bag of flour. Once C_D is known we can compute where subsequent flour bombs will land.

Inputs: As in previous exercises we must know the altitude from which the flour bomb is dropped and the airspeed of the aircraft when the bomb is released. You may again assume that the bomb is dropped from level flight. To compute the drag force we will also need to know the cross sectional area of the bag of flour and its mass. You will also need to know the density of air, ρ, but instead of reading in a value you should simply use the value $\rho = 1.29$ kg/m^3, the density of air under "standard" atmospheric conditions. Since the size of the bag is only several centimeters in all dimensions we would like to input the cross sectional area in square centimeters, but this should be converted to SI units (square meters). The mass of the bomb should be input in grams, and converted to kilograms. It is possible that the program would be used to compute the trajectories of several bombs of different size and weight, so an identifying number or label should also be read for each bomb.

Thus...

- Your program should first read a single line containing the name or label of the bomb. Use a character variable for this name (and don't forget quote marks when entering this name, unless you use "A" format).

- Your program should then read in a line of information describing the bomb. In particular, you should read in the cross sectional area of the bomb (in square centimeters) and the mass (in grams), in that order.

- Finally, your program should read the conditions under which the bomb was dropped. In particular, it should read the altitude (in feet) and the airspeed (in knots), in that order, on a single line.

Output: You should use 51 values for C_D, ranging from zero to 1.25. It is important to include zero because we already know the answer for $C_D = 0.0$ from the previous programs. It is also important to extend the values beyond what is expected, to be sure that we really get the true value within the range of values studied. We want to use many values, so that we can pin down the true value of C_D as accurately as possible, but at the same time we want the whole table to fit on one page. A typical printer page has 66 lines on it, so we want to use less than 66 different values for C_D. If we divide the range 0.0 to 1.25 into 50 intervals we get a regular and orderly spacing between values. If there are 50 *intervals*, then there should be 51 *values* of C_D.

The results of the computations which will be of interest are the time and position of impact, as well as the velocities at impact and the accelerations at impact (to determine whether the terminal velocity has been reached). All of this information should be displayed in a well-organized table, with one line for each different value of C_D. The value of C_D should be the first item on each line, followed by the time of flight, then position, velocity and acceleration information. You can either display the position, velocity and acceleration in the x direction and then the same for the y direction, or you can present the x and y positions, then velocities, then accelerations. Your table should have a width of no more than 80 columns, so that it can be viewed on a normal computer screen.

Your calculations, and the vertical and horizontal positions of the bomb as it falls should now all be in meters rather than feet.

Additional information about the bag (its size and mass) and the altitude and airspeed of the drop – should be displayed at the top of the table. Don't forget to include units for all numbers.

As in the previous exercise, your output should give some kind of indication when a simulation does not finish (the bomb did not yet reach the ground). To make your output easy to read you should put this indicator on the same line as the other information for a given value of C_D.

Optional Improvements: One optional improvment you may choose to make to your code is to have it continue to read additional bomb data and repeat the calculations for those additional data, as long as there are data available. In order to do this you will need to learn how to use the `END=` clause of the `READ` statement.

Note: If you are familiar with calculus then you may like to know that your program is finding the numerical solution to a differential equation ($\mathbf{F} = m\mathbf{a}$) by what is known as Euler's method. If you are not yet familiar with calculus then it may interest you to know that you are doing calculus anyway.

References:

1. Hoerner, Sighard F., "Fluid-Dynamic Drag" (Hoerner Fluid Dynamics, Albuquerque, N.M., 1965)

2. "Topic 1: Projectile Motion with Air Resistance,"
`http://wps.aw.com/wps/media/objects/877/898586/topics/topic01.pdf`