Programming Lab 5 – Blindfolded Archer
PHYS 2511 – Prof. Matthew Newby – Spring 2020

Goal:	Simulate 2D ballistic motion with air resistance, and use this to "blindly" hit a target.
Requirements:	Create a program that solves for the motion of an object in constant gravity, with air resistance, in two dimensions. The object must hit a target on the ground, but each "try" is informed only by the amount by which the target was missed.
Inputs:	The following will need to included at the start of the program, or asked for from the user: • Gravitational constant • Initial velocity magnitude • Initial velocity angle • Time step • The drag coefficient, *b*
Outputs:	• The amount by which each shot misses the target • The number of shots needed to hit the target • The initial parameters of the "winning" shot
Optional:	• Add wind • Add obstacles • Create a plot or plots of the shot attempts

Background:

The force due to gravity is $\vec{F} = -mg\ \hat{y}$, while the force due to *Stokes' drag* is $\vec{F} = -b\vec{v}$, where *g* is gravitational acceleration, *b* is a constant that depends on the geometry of the object, and *v* is the instantaneous velocity of the object.

The average velocity and average acceleration are given by:

$$\bar{a} = \frac{\Delta\vec{v}}{\Delta t} \quad (1) \qquad \bar{v} = \frac{\Delta\vec{x}}{\Delta t} \quad (2)$$

Rearranging:

$$\Delta v = a\Delta t \quad (3) \qquad \Delta x = v\Delta t \quad (4)$$

Given initial values, and choosing a value for Δt, we can calculate *v*, Δt after 0, from that *x* at Δt.

We can keep going, using the output from the previous step (*i*), to get the velocity and position at a new step (*i+1*), like so:

$$v_{i+1} = v_i + a\Delta t \quad (5) \qquad x_{i+1} = x_i + v\Delta t \quad (6)$$

We can calculate as many steps as we need, depending on our ending condition.
Note: If you want to plot the trajectory after the simulation finishes, you should keep all of the position, velocity, and acceleration data.

Also note: This iterative method is more accurate if smaller steps sizes are used, but there is a limit to the time we have to run the program. Solving problems like this with high precision in short amounts of time requires more advanced techniques (which will be covered in the 2nd course in this series).

Hint: Have your program solve for motion in the case of zero air resistance ($b=0$) first, so that you can check it against the known, analytic solution. Once it matches to within 1%, add air resistance into the problem.

An example table of the first few steps in a single trajectory:

Step #	Time	x	y	v_x	v_y	a_x	a_y
0	t_0	x_0	y_0	$v_{x,0}$	$v_{y,0}$	$a_{x,0} = 0$	$a_{y,0} = -g$
1	t_1	$x_0 + v_{0,x}\Delta t$	$y_0 + v_{0,y}\Delta t$	$v_{x,0} + a_{x,0}\Delta t$	$v_{y,0} + a_{y,0}\Delta t$	$-b\, v_{x,0}$	$-g - b\, v_{y,0}$
2	t_2	$x_1 + v_{1,x}\Delta t$	$y_1 + v_{1,y}\Delta t$	$v_{x,1} + a_{x,1}\Delta t$	$v_{y,1} + a_{y,1}\Delta t$	$-b\, v_{x,1}$	$-g - b\, v_{y,1}$

Note that g and b are constants, and the "0" subscript values are initial conditions chosen by the user. We will assume the only acceleration present in the first step is the downward force of gravity. The different between successive t values is always some fixed time step, Δt, chosen prior to the simulation.

The Archer

The archer takes their first shot by choosing some set of initial conditions; the computer then calculates the trajectory and provides the amount by which the shot missed (absolute value!). The archer must then choose a new set of initial conditions, and continue to take shots until the target is hit (within 0.5m of the target center).

You may choose whether to automate the archer's choices, or to have the user aim each new shot.