University of Kentucky, Physics 404G
Homework #1, Rev. A, due Wednesday, 2019-09-04

1. Save the Frog-Prince—A hypothetical prince, out wandering in forest, got turned into a frog of mass 0.454 kg. The object of this exercise is to safely return him from the swamp to his rightful place in the royal water fountain. We will perform our task with a cannon placed 10 m outside the castle wall. Since the prince is moderately afraid of heights, we will fire him through an opening in a window at an elevation of 5 m above swamp-level, with splash-down in the fountain, 4 m inside the wall, aligned horizontally with the window and cannon, at an elevation 1 m above the swamp. To avoid a 'pane'-full splat, please keep all tolerances within 10 cm. Ignore the effects of wind velocity and air resistance.

a) Let's practice in Excel with a 1 lb cube of butter before putting anyone's life in danger. Integrate the equations of motion step-by-step using the leap-frog technique [no pun intended]: for the differential equations $\dot{\boldsymbol{x}} = \boldsymbol{v}$ and $\dot{\boldsymbol{v}} = \boldsymbol{g}$, where $\boldsymbol{g} = [0, -9.81]$ m/s^2, starting from $\boldsymbol{x}_0 = [0, 0]$ m and $\boldsymbol{v}_0 = [5, 14]$ m/s, calculate the new position and velocity $(\boldsymbol{x}, \boldsymbol{v})$ after each time step of Δt=0.01 s, till at least $x = 15$ m, a little past the target. Tabulate each time step on a grid with columns t, x, y, v_x, v_y. Update each new row from values of the previous one, starting with initial conditions on the first row. Use the functions `match` and `index` to find the height y_1 at the wall ($x = 10$ m), and y_2 at the fountain ($x = 14$ m). Adjust the initial conditions so $[y_1, y_2] = [5, 1]$ m within 10 cm.

b) Repeat in Matlab using expressions like `v(end+1,:)=v(end,:)+...` in a `for` or `while` loop to step through the trajectory. Use the function `max` to calculate the maximum height of the frog, and `interp1` to calculate its position at the wall and fountain. Type `help interp1` for details. Plot the frog's trajectory using `plot(x,y)`, where `x` and `y` are column vectors of coordinates.

c) Use Matlab's `ode45` function to automatically integrate the trajectory. Create the file `shoot.m` with a function `yh = shoot(v0)`, where `v0=[vx;vy]` (input) and `yh=[y1;y2]` (output) are 2×1 column vectors. Tune the initial conditions $\boldsymbol{v}_0$ by hand to obtain a safe splashdown in the fountain with target heights `yt=[5;1]`.

d) Instead of solving the trajectory by guess work, we now implement the *Newton-Raphson method* [Numerical Recipes, §9.4] to iteratively refine the input coordinates $\boldsymbol{v}_0$ leading to the desired output $\boldsymbol{y}_t$. For each iteration, numerically calculate the *Jacobian matrix*, $\boldsymbol{J} = \begin{pmatrix} \partial y_1/\partial v_x & \partial y_1/\partial v_y \\ \partial y_2/\partial v_x & \partial y_2/\partial v_y \end{pmatrix}$, of partial derivatives of the function `shoot()`, such that $\boldsymbol{dy}_h = \boldsymbol{J}\,\boldsymbol{dv}_0$, to characterize the effect of adjusting the initial velocity by an arbitrary correction $\boldsymbol{dv}_0$. Use it to tune $\boldsymbol{v}_0$ by the associated correction `v0 = v0 + J\(yt-shoot(v0))`, where the operator `J\dy` is shorthand for left division via the inverse matrix `inv(J)*dy`. Iterate to solve for $\boldsymbol{v}_0$ such that $\boldsymbol{y}_h(\boldsymbol{v}_0) = \boldsymbol{y}_t$.

e) Perform the whole procedure automatically with Matlab's root-finding functionality using the command `fsolve ( @(v)shoot(v)-yt, v0 )`, which finds the vector `v0` such that `shoot(v0)==yt`. Can you solve the whole problem with one command?