ENGR 105 – Introduction to Scientific Computing

Assignment 10

Due by 11:59 pm on Weds. 3/30/2022 on Gradescope

Problem 1 (25 pts): Make an animal perform an unlikely stunt using real images and animation techniques in MATLAB. The only requirements are that you output the animation to a smooth video, that you employ at least one background image and at least one animal image, and that the animal moves at least somewhat interestingly. You can go over-the-top, or not. Bonus points are available for over-the-top algorithms and videos that make the TAs laugh.

Devise a title (not too long, 8 words or less recommended) for your work and name your video file accordingly.

Upload your code, all necessary image files, and the video you produced. Identify all m file names and explain how to launch your code in your README.txt. Let us know of any over-the-top code you produced (if applicable) in your README.txt.

Problem 2 (50 pts): Create script(s) and/or function(s) to track the center of mass of the blue ball shown in the video `Bouncing_ball.mp4`. Use (method 1) tracking via simultaneous envelopes of the red, green, and blue color components of each pixel in each video frame and (method 2) conversion of each frame to grayscale in order to differentiate the ball from the background. Average the row and column locations of pixels that belong to the ball to find the ball's center of mass.

Your script(s)/function(s) should produce a video of the ball's center of mass overlaid on each video frame with a trailing history of five center of mass points (current state inclusive). Your video output should take the form shown in `Bouncing_ball_COM_tracked.avi`, though you should omit the title text "`[Reference]`" from your video. As well, your output video should have the same framerate as the input video.

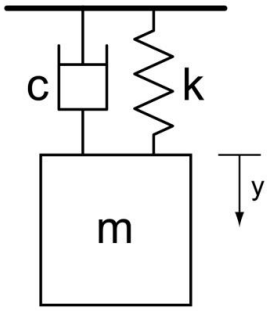
It is unlikely that your code will be able to track the ball in every frame of the video due to the simplicity of the tracking methods and motion/color blur. For example, the ball's center of mass j is missing in frames 10 and 28 of the reference video.

You are encouraged to crop the video after loading it into MATLAB, but before tracking the center of mass – otherwise you may track pixels outside of the photopaper backdrop.

Your code may not utilize any of the image tracking or center of mass functions provided by MATLAB/MathWorks. The central algorithm should be homebrew. You may use built-in MATLAB functions to convert the video frames from color to grayscale.

Upload your function(s)/script(s) and your output video. Identify the names of all files associated with this problem in your README.txt and communicate how one would launch your code.

Problem 3 (25 pts): Damped harmonic oscillators described by second order ordinary differential equations are often encountered in engineering problems. You might encounter such systems when studying diffusion, Newtonian particle physics/atomistics, and the dynamics of many mechanical systems (*e.g*. shock absorbers on cars or robots). Consider the example of a block of weight *m*, subject to gravity, and connected to a parallel spring and damper.



This system can be described by the following second order differential equation.

$$m\ddot{y} + c\dot{y} + ky = 0$$

where *c* is the damping coefficient (units: kg/s), *k* is the spring constant (units: kg/s^2 = N/m), and the derivatives (aka Newton's notation or dot notation) are with respect to time.

It is often useful to know the position and velocity of such a system as a function of time. Create a function that solves this differential equation using `ode45` – a differential equation solver built into MATLAB. Observe the position and velocity of this block subject to a set of initial conditions using the following function declaration.

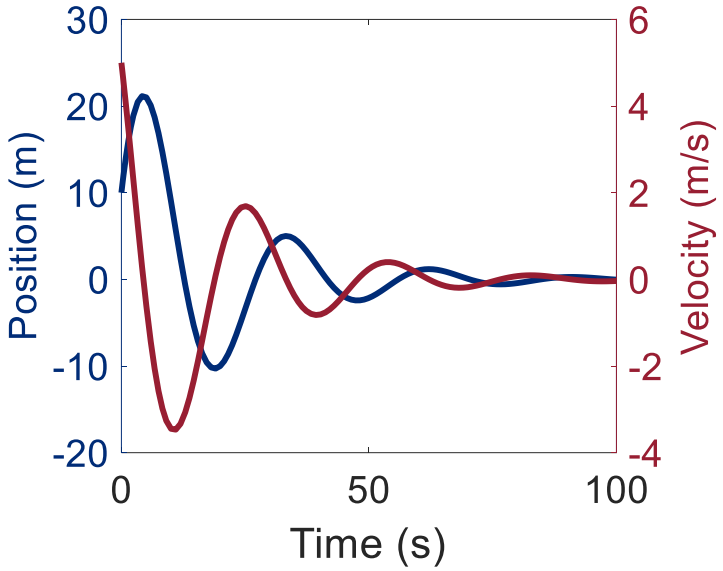
```
function msd_posVel(c,k,m,yi,vi,t_beg,t_end)
```

Input `yi` is the initial position of the system, input `vi` is the initial velocity of the system, input `t_beg` is the start time of the simulation (most often taken to be 0), and `t_end` is the end time of the simulation.

The function that stores the differential equations and is utilized by `ode45()` should have the following function declaration.

```
function dy = msd(t,y,c,k,m)
```

Your function `msd_posVel` should produce a plot with two y axes with the position of the block on the left axis and velocity of the block on the right axis. This plot should be pleasing and include labels on the x axis and both y axes. Include a plot for 0 ≤ `t` ≤ 100 s, `m` = 2 kg, `c` = 0.2 kg/s, `k` = 0.1 N/m, `yi` = 10 m, and `vi` = 5 m/s with your submission. See the following for an example of the plot.



Upload all associated m files and a jpg copy of your plot. Identify the names of all files associated with this problem in your README.txt.