ASEN 2003 - Lab 2
Bouncing Ball Lab
2/16/17

Jeffrey Mariner Gonzalez, * Nicholas Renninger, † Joshua Elster, ‡
University of Colorado - Boulder

The premise of this lab was to estimate the coefficient of restitution, e, of a bouncing ball using a system of MATLAB functions and mathematical derivations. Using the provided MATLAB GUI, position and time information could be extracted from slow-motion videos of each trial. The resulting data was used to estimate e with three different methods: using bounce heights, using the time between bounces, and using both time to rest and initial release height. The value of e was most accurately measured using method 3, a method dependent on an infinite series representation of the system. The least accurate method was the first one. The third method was then improved on using modifications to the procedure to get more accurate data with new trials. The data resulting from the improved procedure was used for the final estimate of e for a ping-pong ball to be 0.920 ± 0.001, and for a golf ball to be 0.908 ± 0.001.

Nomenclature

δh_n = Uncertainty in the current bounce height
δh_{n-1} = Uncertainty in the previous bounce height
δt_n = Uncertainty in the current time between bounces
δt_{n-1} = Uncertainty in the previous time between bounces
δt_{stop} = Uncertainty in the time until the ball stops bouncing
e = Coefficient of Restitution
g = Gravitational acceleration
h_0 = Initial height
t_0 = Time from release to first bounce

*105219585
†105492876
‡104764281

Contents

I. Theory

When two bodies collide, the total momentum of the system remains constant, as momentum cannot be created or destroyed. However, during this collision energy can be removed from the system, which results in lower velocities after the collision. This energy is put into the work that deforms the objects involved. Through analysis it is possible to predict this behavior because the energy lost can be calculated if the coefficient of restitution, e, is known. This coefficient can be described due to the fact that a low amount of energy is carried over during the collision. However, e usually has to be experimentally determined. This lab experiment is designed to demonstrate how to find e for a bouncing ball by three different methods for the same data set, as well as the same calculations for a different type of ball. The first method compares the height the ball bounces to after each collision with the ground, and is solely dependent on the kinetic and potential energy of the ball throughout the trial. This is done using equation 1:

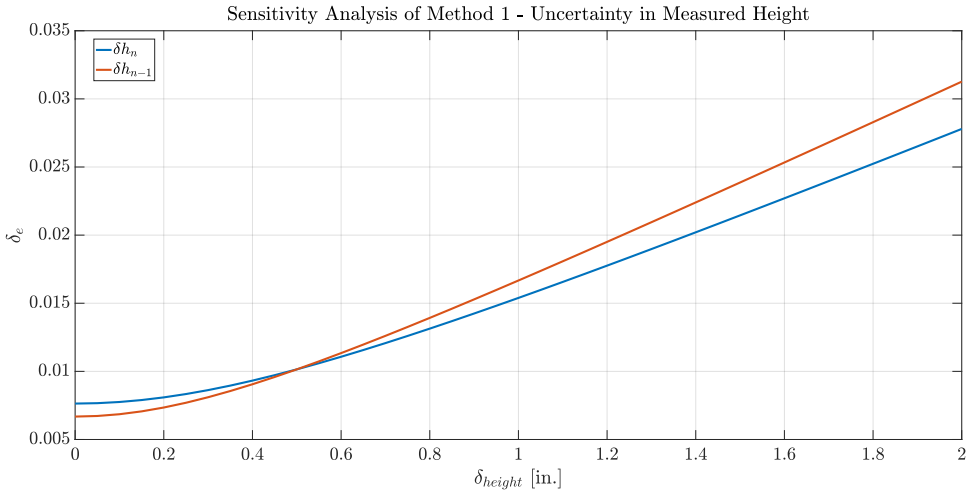


Figure 1: Sensitivity Analysis Graph for Method 1. *Here, δh_n and δh_{n-1}* both start at 0.5 inches in the simulation. The plot shows that the dominant error is δh_{n-1}, as when both δh_n & δh_{n-1} are increased the same amount over their base value of 0.5 in., δe increases more with increasing δh_{n-1}.

$$e = \left(\frac{h_0}{h_n}\right)^{\frac{1}{2n}} = \left(\frac{h_n}{h_{\text{n-1}}}\right)^{\frac{1}{2}} \tag{1}$$

h_o	Initial Height
n	Number of Bounces
h_n	nth Bounce

The second method compares the time between each bounce, using equation 2:

$$e = \left(\frac{t_n}{t_{\text{n-1}}}\right) \tag{2}$$

t_n	Current Time
$t_{\text{n-1}}$	Previous Time

The final method is to analyze e based off of the drop height and time it takes the ball to stop bouncing, using equation 3:

$$e = \frac{t_s - \sqrt{\frac{2h_o}{g}}}{t_s + \sqrt{\frac{2h_0}{g}}} \tag{3}$$

t_s	Time to ball stops bouncing
g	Gravity
h_0	Initial Release Height

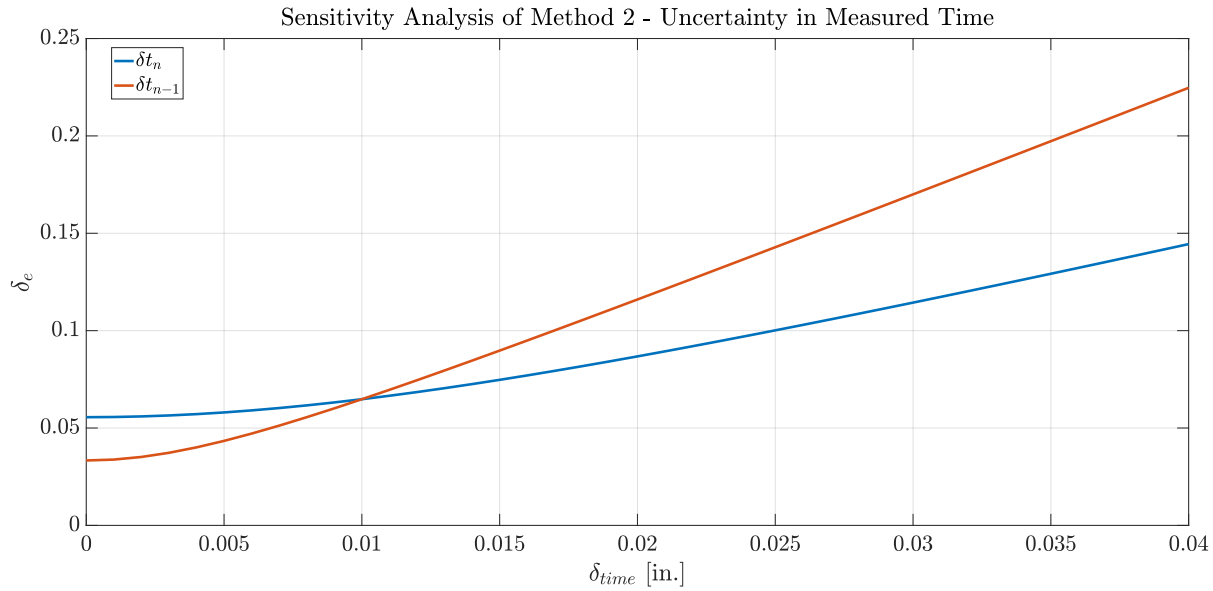


Figure 2: Sensitivity Analysis Graph for Method 2. *Here, δt_n and δt_{n-1}* both start at 0.01 seconds in the simulation. The plot shows that the dominant error is δt_{n-1}, as when both δt_n & δt_{n-1} are increased the same amount over their base value of 0.01 s, δe increases more with increasing δt_{n-1}.

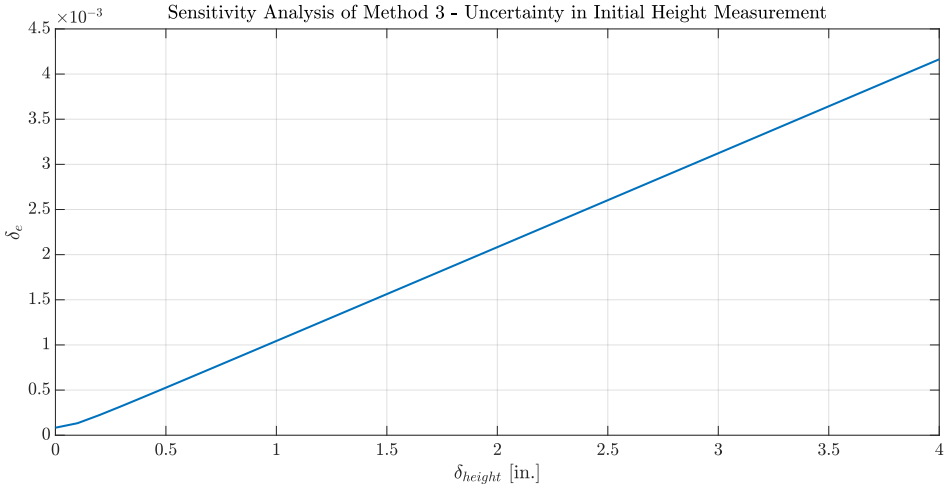


Figure 3: Sensitivity Analysis Graph for Method 3. *In this case, the sensitivity of δe to just δh_o is plotted.*

These three equations are each dependent on a different set of variables, all of which are dependent on the same initial conditions of the lab (including the momentum, energy and coefficient of restitution of the ball). These equations have been derived in Appendix B. These dependent variables also reveal how the the equations and their results are sensitive to different pieces of data that can be collected. Equation 1 is solely dependent on the difference in height between bounces of the ball. As a result, the initial release height does not have to be accurate because any two consecutive bounces can be used. This means that even when the ball cannot be accurately timed or the initial height is not accurately set this equation will be accurate as shown in Fig. 1. However, the second equation is completely dependent on the time between consecutive bounces, meaning that any error in timing the bounces will be significant when using this equation 2. A useful method to use this equation would be to time the sounds of each bounce. As it is a short sound that only occurs on the bounce this would give an accurate measure of the time between bounces and the height of release and each bounce could be ignored. Both equations 1 and 2 have the same issue, namely how to measure accurately over milliseconds, which is a fairly difficult task. Equation 3 presents a solution that is only dependent on the initial release height and the total time that the ball requires to stop bouncing, as seen in Figures 3 & 4. With the initial release height being accurately measured thanks to using a release mechanism on the same surface for each trial, the major source of error becomes determining when the ball stops bouncing. It can be difficult to tell when the ball stops bouncing because the bounces

get smaller and smaller until its energy cannot overcome gravity. It is still a single data point within a set of ten to twenty, meaning that equation 3 becomes the most accurate when this data collection is difficult as it does not depend on these bounces.

Within each equation, which variable is most influential can change depending on the magnitude of the error. This sensitivity can be seen in figures (numbers of figure for sensitivity). Notice that all three methods tend to follow a similar trend, where one variable overtakes the others in influence as the error is scaled. This behavior is understandable as all three equations for the methods follow the trend of two similar terms divided by each other. This means that one partial derivative in the general error propagation formula is a coefficient or function and the other is a function or coefficient over one. Because the former situation has its error increased it more influential, and the overall error can increase. However, when the coefficient or function is over one when the error of the variable causes it to become important, there smaller increases because dividing a number by a larger number results in a smaller number as the product.

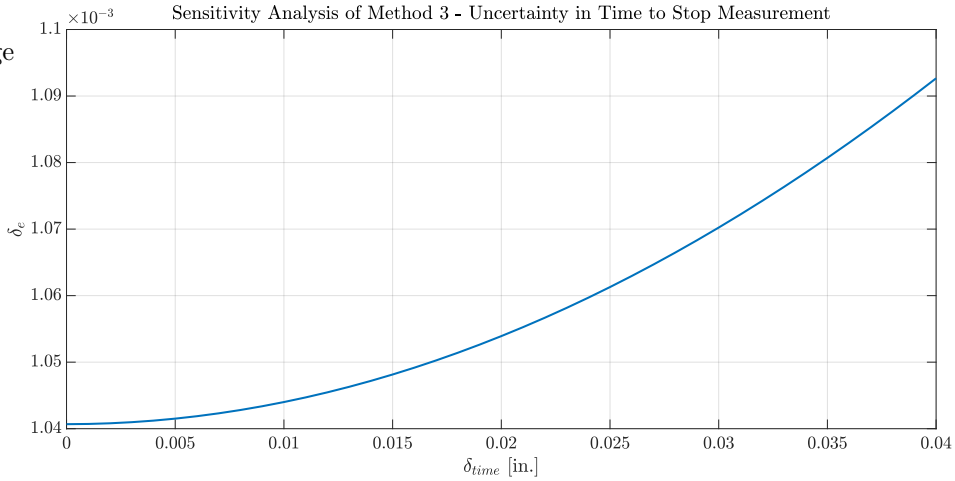


Figure 4: Sensitivity Graph for Method 3. *In this case, the sensitivity of δe to just δt_s is plotted.*

II. Procedure

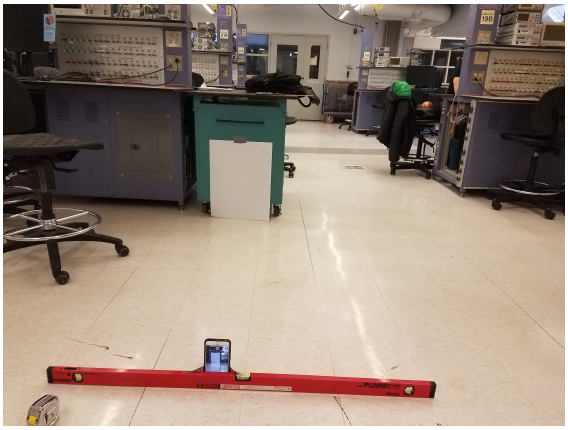
The procedure was used to experimentally determine the best method of solving for the coefficient of restitution and was designed to minimize the impact of mistakes or the effects of unaccounted-for variables. This was achieved by using the trials themselves as a constant, and only ten trials where carried out for the initial round of calculations. These ten trials where then used for all three methods of finding the coefficients of restitution.

Steps (First Set of Trials):

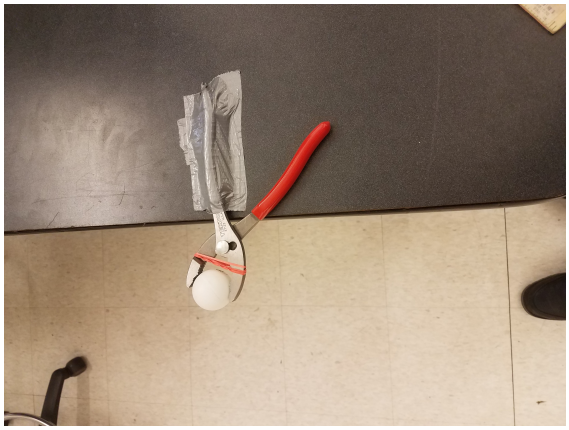
1. Take two pieces of 17" by 22" 0.25 grid paper. The two pieces were taped together and attached to a flat section of wall with limited airflow and the base on a level surface, measured with a level.
2. Mark the release height a half inch from the top.
3. Place a stand in front of the paper on the wall, at a height near the middle of the paper to reduce the effect of the angle between the lens and the ball distorting the height of the ball when viewed through the camera.
4. Have one person hold and run the camera, and another run a stop watch while the third person releases the ball.
5. At the beginning of each trial, the person with the stop watch holds the stop watch in frame of the camera, in the upper of corner of the view. While the cameraman rest the camera on the support, and the ball holder places the ball at the marked starting height. It may be necessary to move the ball forward to avoid it striking the wall or landing and bouncing on a uneven section of bouncing surface. The watch holder counts down from three to one.
6. On three, cameraman beings recording the trial.
7. On two, the timer starts.

8. On one, the ball is released from the marked initial height.
9. Continue timing and recording until the ball has stopped bouncing
10. Record the total time of the trial and save the recorded video.
11. Reset and repeat steps 1 through 5 until ten trials have been collected.

    During this procedure, sources of error include the fact that the initial height varied, the estimation of time to stop was made by human guess, and the floor was not fully even, leading to bounces at angles and some rolling.



(a) Modified Camera and Table Top Setup for use with Method 3. *By placing the camera on the ground, very accurate measurements of when the ball stopped bouncing could be made.*



(b) Modified Ball Release System. *Instead of dropping the ball by hand, a mechanism for consistent release height of the ball was developed to minimize the error in the measurement of the initial height of the ball*

Figure 5: The improved experimental setup

The experimental method was modified to improve one of the three methods. For this lab, the third method was selected for modification as it was reliant on only two data points to calculate the coefficient of restitution and lacks the errors found when collecting each data point with the provided video analysis software. Steps (Modified):

1. Place a rubber band around a pair of large pliers.
2. Affix the pliers to a level tabletop above a level section of hard surfaced floor. Only affix one of the handle to the surface of the table, making sure that the other can swing to open and close the pliers.
3. Tape a piece of 17" by 22" 0.25 inch grid paper to the surface running from the tabletop to the floor. Make sure it contacts the floor along its bottom edge.
4. Place a camera at ground level far enough back to capture both the release point and the bouncing of the ball.
5. Place the ball of choice into the mouth of the pliers, and close them around the ball using the rubber band.
6. Start the camera.
7. Release the ball by quickly opening the pliers.
8. Wait for the ball to stop bouncing and save the recording.
9. Reset and repeat from step 5 for ten trials total.

Although there are fewer sources of error in this modified procedure, the ball still tended to roll or otherwise not bounce directly up in frame.

III. Results

Figures 6 - 10 reveal the final results of the trials with error bars, as well as overall statistics for each method. These were the results used to decide which method was most worth improving, as well as how that method worked with a different type of ball. Method three was shown to have the least uncertainty in each of its measurements, and also possessed the smallest standard deviation across its set of trials. This Here the means of all of the trails provide coefficient of restitution for each method. Which are 0.877, .886, .904, .920 and .908 for methods 1 through 3, modified 3, and with the golf ball, respectively.

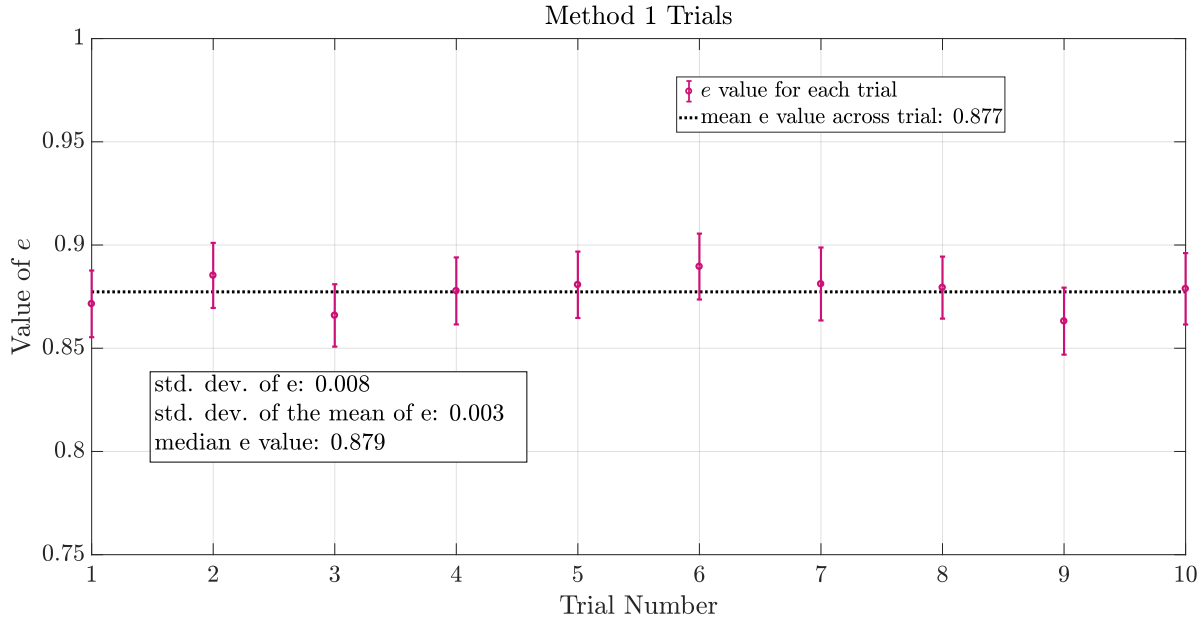


Figure 6: Results of Method 1 Trials

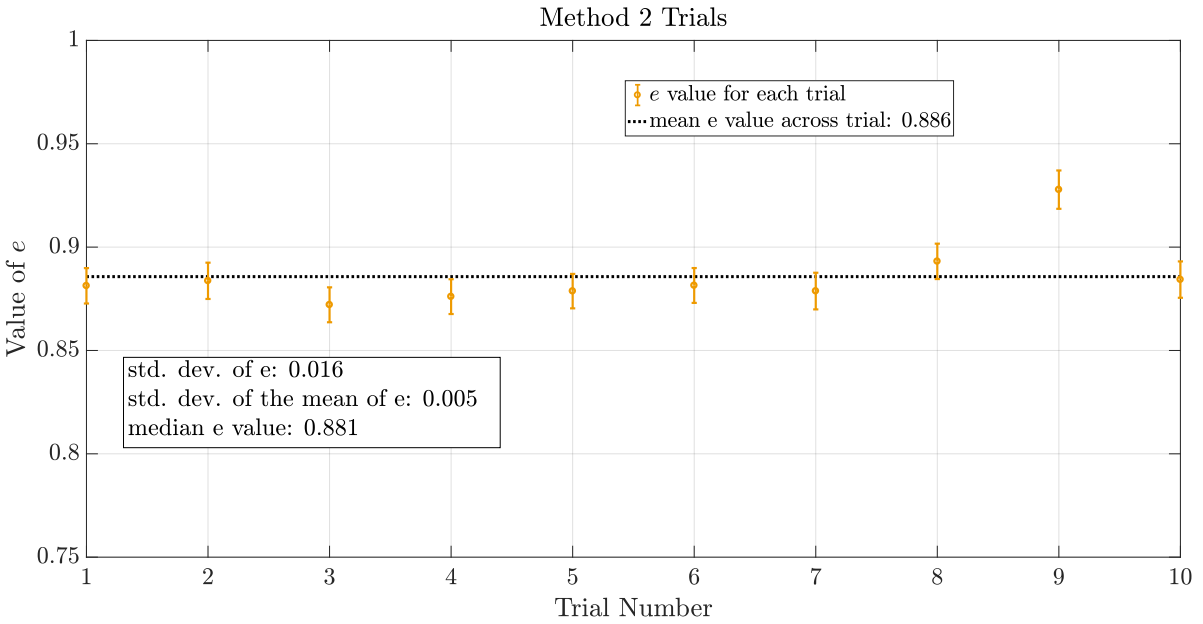


Figure 7: Results of Method 2 Trials

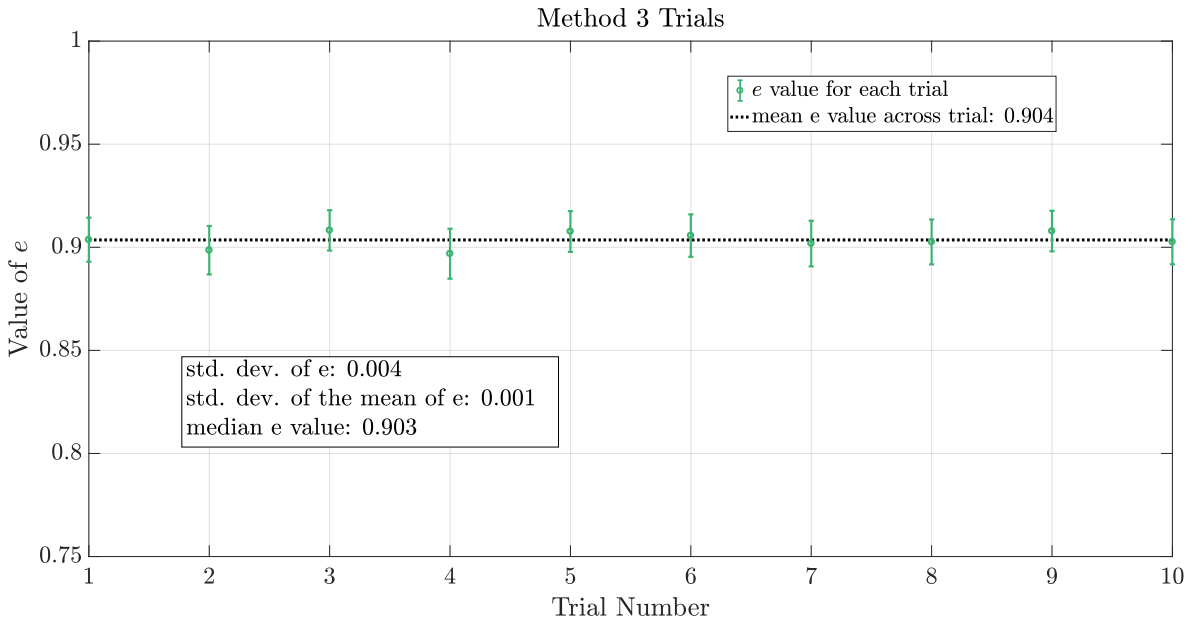


Figure 8: Results of Method 3 Trials

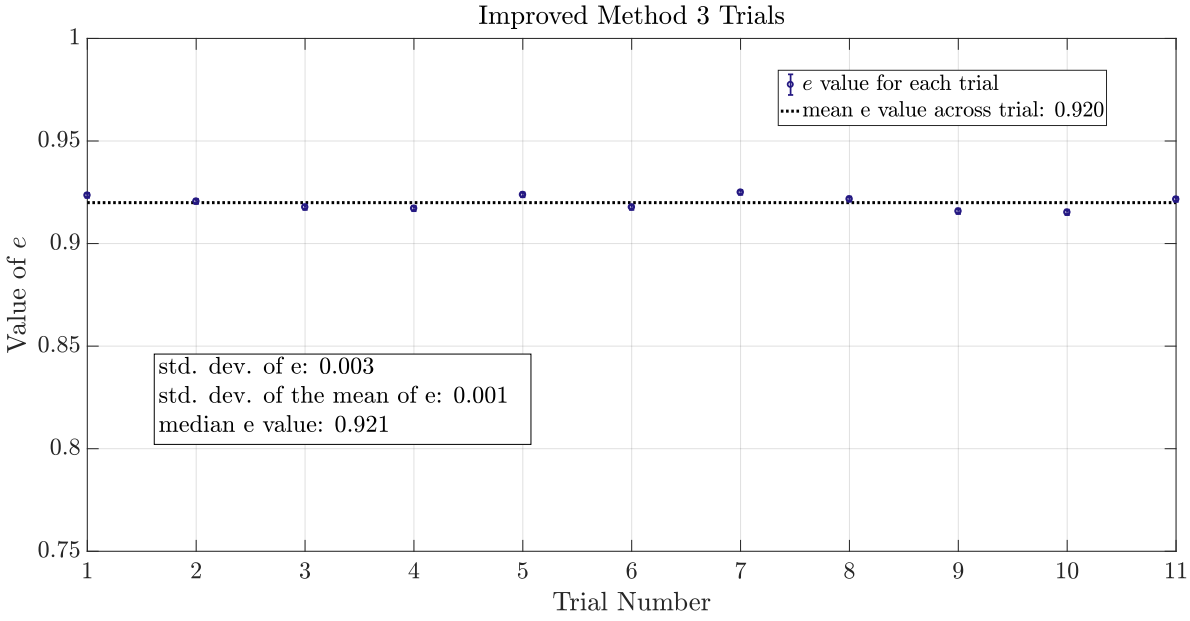


Figure 9: Modified Method 3 Procedure Results

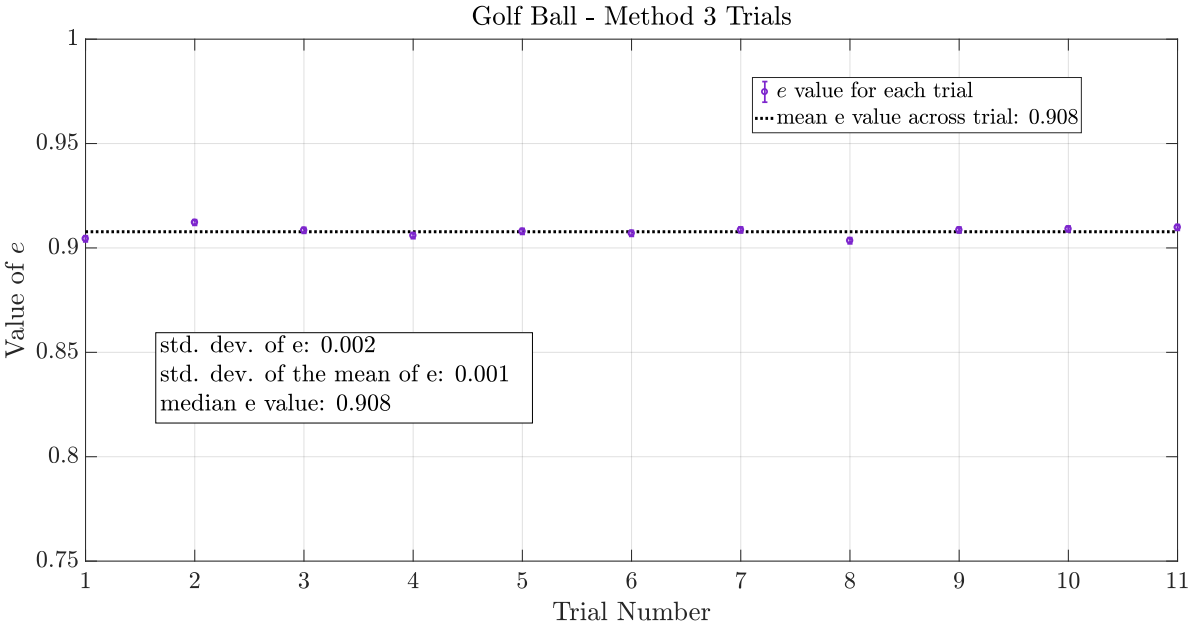


Figure 10: Results of Golf Ball with Modified Method 3 Procedure

IV. Performance Analysis

To determine the uncertainty in the measured value of e, the general error formula was used:

$$\delta_f = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 \delta_x^2 + \left(\frac{\partial f}{\partial y}\right)^2 \delta_y^2 + \left(\frac{\partial f}{\partial z}\right)^2 \delta_z^2 + ...} \tag{4}$$

where each δ was an estimate of the uncertainty in the measured quantity, based on our best judgment and our experiences during the trials. For the our measurements of the height of the ball, we used a δh equal to approximately the radius of the ball. For our measurements of the time, all estimates of our δt were about 2-10 frames (depending on which method for calculation of e was being used), with the frame rate of the slow motion video being 240fps.

By examining the error bars of Figures 6 - 10, the most accurate of the unmodified methods was method three, as its error bars were of visibly lower magnitudes than those of methods one and three. The first two methods both had very visible error bars, with method one having a far larger error then the second method. Method 1 on average had an uncertainty in each measurement of e of ± 0.016, while method two on average had an uncertainty in each measurement of e of ± 0.01, while method three on average had an uncertainty in each measurement of e of ± 0.009. The superiority of method three was also supported by the statistics of each set of data; the third method has a standard deviation of 0.004 which was lower then the first two trials, which had values of 0.08 and 0.016 respectively. This reveals that the third method had the least variance and can be considered to be more trustworthy. From our findings, it was decided that the third method would be the best to improve as it was already the most accurate and precise, owing to the fact that is was reliant on two data points for each trial, instead of many that where closely spaced. In addition, it lacked the error unaccounted for when the visual analysis GUI was used. This error being that the user of the program chose the center or bottom of the ball with some error of possibly a few pixels on the screen, as well as the error that was estimated in the time or height collection, that was in the data once for each bounce, causing it to add up over the entire data collection process. This also explains the large error in the first method, as it was completely reliant on finding the heights correctly, while it was difficult to find the exact point where is was a local maximum. With the second method, it was easier to find when the ball bounces up as it occurred at consistent a location on the interface. The modification made to method 3 decreased both the size of the error and the standard deviation of the results. This is because it worked off of the strengths of the final method, where the release height was far more consistent from the use of the release pliers, which removed the human error of the ball being released from a slightly different height each trial. Additionally, having the camera lower to the ground made the estimate of when the ball stopped bouncing much more precise. This decreased our uncertainty in the measurement of e for each trial, and decreased variance of e across the trials. In the improved method three trials, the uncertainty was decreased to ± 0.001 on average, and the standard deviation across the trials was computed to be 0.03. As a result, the final estimate for e for the ping-pong ball is made from the improved method three trial, where e was measured to be 0.920 with an uncertainty of ± 0.001. Using the same improved procedure with the second ball, a golf ball, e was determined to be 0.908 with an uncertainty of ± 0.001. These results make sense when the theory behind each method's equation is taken into account, because method three was simplest to obtain accurate data for. Method one was based on the conservation of energy, and energy, particularly kinetic energy, can be rather difficult to measure due to the need for accurate release height, speed and object mass. These variables all add their own error to the experiment. The second method suffers from a similar drawback, that it depended on the comparison of the velocities, and simplifies down to the time of each bounce. Both time and velocity are rather difficult to measure because they are dependent on so many variables in this situation, such as the release height, ball spin, air friction, gravity, and bounce angle. The third method was simply added together all of the times of the bounces, and did so in terms of the time to hit the ground initially and e. This simplified into an equation only dependent on the release height and time to stop, as gravity is constant. This is effectively the time to strike the ground for the first time plus a summation series of the total time of bounces after, in terms of the initial time to strike. This can be considered a infinite summation series, which is equal to an the non-infinite series (as shown in App. B),

$$\frac{-e}{e-1} \tag{5}$$

Which simplifies to the most basic equation for the three methods.

V. Conclusions and Recommendations

This lab allowed for us to better comprehend the idea of a coefficient of restitution and how to find it. Additionally, the derivations done during this lab showed us the various parameters that e is reliant on. During our trials, the best performing method was our modified method three trial, using the new procedure with the plier drop mechanism and consistent drop height. The worst performing method was method 1 due to its reliance on the height of each ball bounce, which proved to be the least reliable to accurately measure. If we were to give advice to someone trying to measure the coefficient of restitution of an object, we would say to try to keep your location and other data consistent. Additionally, using the sound of the bounces is incredibly beneficial to getting accurate data, as the ball has very small bounces towards the end of a trial, and it is easier to listen for it than to try and see it. Coding the GUI to accept .MOV files also is an incredible time saver, depending on what format your recording device uses.

References

[1]Nerem, Steven. ASEN 2003 Lab 2: Bouncin' Balls. CU, 2017. PDF.
[2]Bedford, A., and Fowler, W.,*Engineering Mechanics: Dynamcis*, Upper Saddle River, NJ: Pearson, 2008.

Acknowledgments

We would like to thank Professor Nerem for additional help understanding the lab, as well as the T.A.'s for explanations of calculations and derivations used.

Appendix A

The primary contributions of each lab member are as follows:

Jeffrey Mariner Gonzalez: Developed the sensitivity analysis code, "findFive.m" function, error propagation code, worked on report, derived equations

Nicholas: Developed main code for data input, error propagation, statistical analysis, and all plotting. Collaboration on report writing (theory, results, performance analysis) and editing.

Joshua: Wrote abstract, inputted derivations, conclusions and recommendations, edited report, used the GUI to make data tables for 8 of the videos in trial one, all 10 of the golf ball trials, and all 10 of the day two trials.

Appendix B: Derivations

Method 1:

$$mgh_0 = \frac{1}{2}mv_0^2 \tag{6}$$

$$mgh_n = \frac{1}{2}mv_n^2 \tag{7}$$

$$\sqrt{2gh_0} = v_0 \tag{8}$$

$$\sqrt{2gh_n} = v_n \tag{9}$$

$$\frac{v_n}{v_0} = \sqrt{\frac{h_n}{h_0}} \tag{10}$$

$$e = \left(\frac{h_n}{h_0}\right)^{\frac{1}{2n}} \tag{11}$$

Method 2:

$$mgh_0 = \frac{1}{2}mv_0^2 \tag{12}$$

$$mgh_0 = \frac{1}{2}mv_n^2 \tag{13}$$

$$\sqrt{2gh_0} = v_0 \tag{14}$$

$$\sqrt{2gh_n} = v_n \tag{15}$$

$$\frac{v_n}{v_0} = \sqrt{\frac{h_n}{h_0}} \tag{16}$$

$$t_n = \sqrt{\frac{2h_0}{g}} \tag{17}$$

$$\frac{t_n}{t_{n-1}} = \sqrt{\frac{h_0}{h_n}} \tag{18}$$

$$e = \frac{v_n}{v_{n-1}} \tag{19}$$

$$e = \frac{gt}{gt} \tag{20}$$

$$e = \frac{g\sqrt{\frac{2h_n}{g}}}{g\sqrt{\frac{2h_{n-1}}{g}}} \tag{21}$$

$$e = \frac{\sqrt{\frac{2h_n}{g}}}{\sqrt{\frac{2h_{n-1}}{g}}} \tag{22}$$

$$e = \frac{t_n}{t_{n-1}} \tag{23}$$

Method 3:

$$t_s = t_0 + \sum_{k=1}^{n} t_n \tag{24}$$

$$t_s = t_0 + \sum_{k=1}^{n} 2e^n t_0 \tag{25}$$

$$t_s = t_0 + 2t_0 \sum_{k=1}^{n} e^n \tag{26}$$

$$t_s = t_0 + 2t_0 \left(\frac{-e}{e-1} \right) \tag{27}$$

$$t_s - t_0 = \left(\frac{-2t_0 e}{e-1} \right) \tag{28}$$

$$1 - \frac{1}{e} = \frac{2t_0}{t_0 - t_s} \tag{29}$$

$$\frac{1}{e} = \frac{-t_s - t_0}{t_0 - t_s} \tag{30}$$

$$e = \frac{t_s - t_0}{t_s + t_0} \tag{31}$$

Appendix C: Additional Procedures

Appendix D: MATLAB Code

```
%%% ASEN 2003 Lab 2 Main
%%%
%%% Author: Nicholas Renninger
%%% Created: 2/2/2017
%%% Last Modified: 2/17/2017


%% Housekeeping

clear variables
close all
clc


%% Define Constants

% define how many video pixels make up one inch
PIXEL_TO_IN_SETUP_1 = 15.826;
PIXEL_TO_IN_SETUP_2 = 16.510;

% define the possible error in the determination of the ball's height in
% method 1 analysis
BALL_RAD_SETUP_1 = 35.048924 / 2; % [pixels]

% define the possible error in the measurement of the time between bounces
% as the time between frames given the frame rate of the camera - 240 fps.
TIME_ERROR = (2 / 240); % [s]

% Define the height at which the ball was dropped for the trials
H_INIT_DAY_1 = 43.75; % [in]
H_INIT_DAY_2 = 34; % [in]

% Define the uncertainty in the initial height of the ball for Day 1 Data
H_INIT_ERROR_DAY_1 = 4; % [in]
H_INIT_ERROR_DAY_2 = 1; % [in]


% Define the uncertainty in the time till the ball stops - initial time
% measurement uncertainty + final time measurement uncertainty
init_time_error = (2 / 240); % [s]
final_time_error_day_1 = (240 / 240); % [s]
final_time_error_day_2 = (5 / 240); % [s]

STOP_TIME_ERROR_DAY_1 = init_time_error + final_time_error_day_1; % [s]
STOP_TIME_ERROR_DAY_2 = init_time_error + final_time_error_day_2; % [s]


% plot constants
FONTSIZE = 28;
LINEWIDTH = 3.5;
set(0, 'defaulttextinterpreter', 'latex');
dim = [.2 .5 .3 .3];
mean_vec = ones(1, 100);
```

```
trials = linspace(0, 11, 100);


%% Read in all of the trials data

% read in data in trial 1 folder
trial_num = 1;
trial_1_data = read_data_files(PIXEL_TO_IN_SETUP_1, trial_num);

% read in data in trial 2 - golf ball folder
trial_num = 2;
trial_2_data = read_data_files(PIXEL_TO_IN_SETUP_1, trial_num);

% read in data in trial 3 - new method folder
trial_num = 3;
trial_3_data = read_data_files(PIXEL_TO_IN_SETUP_1, trial_num);


% define number of trials for Day 1 and Day 2
num_trials_day_1 = length(trial_1_data);
num_trials_day_2 = length(trial_2_data);

%% Calc. e & uncertainty for each method, improved method, and new ball

%%% For Day 1 Data
for i = 1:length(trial_1_data)

    h_vec = trial_1_data{i}.y;
    t_vec = trial_1_data{i}.time;


    %%% Day 1 data, method 1
    [e_curr, sigma_e_curr] = calc_e_method_1(h_vec, t_vec, ...
                                BALL_RAD_SETUP_1, BALL_RAD_SETUP_1);

    e_vec.one.values(i) = e_curr;
    e_vec.one.error(i) = sigma_e_curr;


    %%% Day 1 data, method 2
    [e_curr, sigma_e_curr] = calc_e_method_2(h_vec, t_vec, TIME_ERROR);

    e_vec.two.values(i) = e_curr;
    e_vec.two.error(i) = sigma_e_curr;


    %%% Day 1 data, method 3
    [e_curr, sigma_e_curr] = calc_e_method_3(t_vec, H_INIT_DAY_1, ...
                                            STOP_TIME_ERROR_DAY_1, ...
                                            H_INIT_ERROR_DAY_1);

    e_vec.three.values(i) = e_curr;
    e_vec.three.error(i) = sigma_e_curr;

```

```
end


%%% For Day 2 Data (improved method 3)
for i = 1:length(trial_2_data)

    h_vec = trial_2_data{i}.y;
    t_vec = trial_2_data{i}.time;

    %%% Day 2 data, method 3
    [e_curr, sigma_e_curr] = calc_e_method_3(t_vec, H_INIT_DAY_2, ...
                                             STOP_TIME_ERROR_DAY_2, ...
                                             H_INIT_ERROR_DAY_2);

    e_vec.four.values(i) = e_curr;
    e_vec.four.error(i) = sigma_e_curr;

end


%%% For Day 2 Data (golf ball - method 3)
for i = 1:length(trial_3_data)

    h_vec = trial_3_data{i}.y;
    t_vec = trial_3_data{i}.time;

    %%% Day 2 data, method 3
    [e_curr, sigma_e_curr] = calc_e_method_3(t_vec, H_INIT_DAY_2, ...
                                             STOP_TIME_ERROR_DAY_2, ...
                                             H_INIT_ERROR_DAY_2);

    e_vec.five.values(i) = e_curr;
    e_vec.five.error(i) = sigma_e_curr;

end


%% Calc. the stats of e vector for each method, improved method, & new ball

% find mean value of e for each data set
mean_e_1 = mean(e_vec.one.values);
mean_e_2 = mean(e_vec.two.values);
mean_e_3 = mean(e_vec.three.values);
mean_e_4 = mean(e_vec.four.values);
mean_e_5 = mean(e_vec.five.values);

% std. dev. for each data set
std_dev_e_1 = std(e_vec.one.values);
std_dev_e_2 = std(e_vec.two.values);
std_dev_e_3 = std(e_vec.three.values);
std_dev_e_4 = std(e_vec.four.values);
std_dev_e_5 = std(e_vec.five.values);

% find std. dev. of the mean for each data set
std_dev_mean_e_1 = std_dev_e_1 / sqrt(num_trials_day_1);
```

```
std_dev_mean_e_2 = std_dev_e_2 / sqrt(num_trials_day_1);
std_dev_mean_e_3 = std_dev_e_3 / sqrt(num_trials_day_1);
std_dev_mean_e_4 = std_dev_e_4 / sqrt(num_trials_day_2);
std_dev_mean_e_5 = std_dev_e_5 / sqrt(num_trials_day_2);


% find median value of e for each data set
median_e_1 = median(e_vec.one.values);
median_e_2 = median(e_vec.two.values);
median_e_3 = median(e_vec.three.values);
median_e_4 = median(e_vec.four.values);
median_e_5 = median(e_vec.five.values);

%% Do sensitivity analysis

sensitivityAnalysis


%% Plot Method 1

figure_title = sprintf('Method 1 Trials');

legend_string = {'$e$ value for each trial', ...
        sprintf('mean e value across trial: %0.3f', mean_e_1)};

textbox_str = {sprintf('std. dev. of e: %0.3f', std_dev_e_1), ...
        sprintf('std. dev. of the mean of e: %0.3f', std_dev_mean_e_1), ...
        sprintf('median e value: %0.3f', median_e_1)};


xlabel_string = sprintf('Trial Number');
ylabel_string = sprintf('Value of $e$');
y_limits = [0.75, 1];
x_limits = [1, num_trials_day_1];
LEGEND_LOCATION = 'best';
color = [205;16;118] ./ 255;


method_1_plot = figure('name', 'Method 1');
scrz = get(groot,'ScreenSize');
set(method_1_plot, 'Position', scrz)


p2 = plot(trials, mean_vec .* mean_e_1, ':k',...
          'LineWidth', LINEWIDTH);
hold on
p1 = errorbar(e_vec.one.values, e_vec.one.error, 'o',...
              'LineWidth', LINEWIDTH - 1, 'color', color);

annotation('textbox',dim,'String',textbox_str,'FitBoxToText','on', ...
           'interpreter', 'latex', 'fontsize', ...
           FONTSIZE, 'backgroundcolor', 'w');

grid on
set(gca,'FontSize', FONTSIZE)
```

```
set(gca, 'defaulttextinterpreter', 'latex')
set(gca, 'TickLabelInterpreter', 'latex')
xlim(x_limits)
ylim(y_limits)
title(figure_title)
xlabel(xlabel_string)
ylabel(ylabel_string)
legend([p1, p2], legend_string, 'location', LEGEND_LOCATION, ...
       'interpreter', 'latex')


%% Plot Method 2

figure_title = sprintf('Method 2 Trials');

legend_string = {'$e$ value for each trial', ...
        sprintf('mean e value across trial: %0.3f', mean_e_2)};

textbox_str = {sprintf('std. dev. of e: %0.3f', std_dev_e_2), ...
        sprintf('std. dev. of the mean of e: %0.3f', std_dev_mean_e_2), ...
        sprintf('median e value: %0.3f', median_e_2)};


xlabel_string = sprintf('Trial Number');
ylabel_string = sprintf('Value of $e$');
y_limits = [0.75, 1];
x_limits = [1, num_trials_day_1];
LEGEND_LOCATION = 'best';
color = [238;154;0] ./ 255;


method_2_plot = figure('name', 'Method 2');
scrz = get(groot,'ScreenSize');
set(method_2_plot, 'Position', scrz)


p2 = plot(trials, mean_vec .* mean_e_2, ':k',...
          'LineWidth', LINEWIDTH);
hold on
p1 = errorbar(e_vec.two.values, e_vec.two.error, 'o',...
              'LineWidth', LINEWIDTH - 1, 'color', color);

annotation('textbox',dim,'String',textbox_str,'FitBoxToText','on', ...
           'interpreter', 'latex', 'fontsize', ...
           FONTSIZE, 'backgroundcolor', 'w');

grid on
set(gca,'FontSize', FONTSIZE)
set(gca, 'defaulttextinterpreter', 'latex')
set(gca, 'TickLabelInterpreter', 'latex')
xlim(x_limits)
ylim(y_limits)
title(figure_title)
xlabel(xlabel_string)
ylabel(ylabel_string)
```

```
legend([p1, p2], legend_string, 'location', LEGEND_LOCATION, ...
       'interpreter', 'latex')


%% Plot Method 3

figure_title = sprintf('Method 3 Trials');

legend_string = {'$e$ value for each trial', ...
        sprintf('mean e value across trial: %0.3f', mean_e_3)};

textbox_str = {sprintf('std. dev. of e: %0.3f', std_dev_e_3), ...
        sprintf('std. dev. of the mean of e: %0.3f', std_dev_mean_e_3), ...
        sprintf('median e value: %0.3f', median_e_3)};


xlabel_string = sprintf('Trial Number');
ylabel_string = sprintf('Value of $e$');
y_limits = [0.75, 1];
x_limits = [1, num_trials_day_1];
LEGEND_LOCATION = 'best';
color = [60,179,113] ./ 255;


method_3_plot = figure('name', 'Method 3');
scrz = get(groot,'ScreenSize');
set(method_3_plot, 'Position', scrz)


p2 = plot(trials, mean_vec .* mean_e_3, ':k',...
          'LineWidth', LINEWIDTH);
hold on
p1 = errorbar(e_vec.three.values, e_vec.three.error, 'o',...
              'LineWidth', LINEWIDTH - 1, 'color', color);

annotation('textbox',dim,'String',textbox_str,'FitBoxToText','on', ...
           'interpreter', 'latex', 'fontsize', ...
           FONTSIZE, 'backgroundcolor', 'w');

grid on
set(gca,'FontSize', FONTSIZE)
set(gca, 'defaulttextinterpreter', 'latex')
set(gca, 'TickLabelInterpreter', 'latex')
xlim(x_limits)
ylim(y_limits)
title(figure_title)
xlabel(xlabel_string)
ylabel(ylabel_string)
legend([p1, p2], legend_string, 'location', LEGEND_LOCATION, ...
       'interpreter', 'latex')


%% Plot Improved Method 3

figure_title = sprintf('Improved Method 3 Trials');
```

```
329
330 legend_string = {'$e$ value for each trial', ...
331         sprintf('mean e value across trial: %0.3f', mean_e_4)};
332
333 textbox_str = {sprintf('std. dev. of e: %0.3f', std_dev_e_4), ...
334         sprintf('std. dev. of the mean of e: %0.3f', std_dev_mean_e_4), ...
335         sprintf('median e value: %0.3f', median_e_4)};
336
337
338 xlabel_string = sprintf('Trial Number');
339 ylabel_string = sprintf('Value of $e$');
340 y_limits = [0.75, 1];
341 x_limits = [1, num_trials_day_2];
342 LEGEND_LOCATION = 'best';
343 color = [36;24;130] ./ 255;
344
345
346 method_3_imp_plot = figure('name', 'Improved Method 3');
347 scrz = get(groot,'ScreenSize');
348 set(method_3_imp_plot, 'Position', scrz)
349
350
351 p2 = plot(trials, mean_vec .* mean_e_4, ':k',...
352           'LineWidth', LINEWIDTH);
353 hold on
354 p1 = errorbar(e_vec.four.values, e_vec.four.error, 'o',...
355               'LineWidth', LINEWIDTH - 1, 'color', color);
356
357 annotation('textbox',dim,'String',textbox_str,'FitBoxToText','on', ...
358            'interpreter', 'latex', 'fontsize', ...
359            FONTSIZE, 'backgroundcolor', 'w');
360
361 grid on
362 set(gca,'FontSize', FONTSIZE)
363 set(gca, 'defaulttextinterpreter', 'latex')
364 set(gca, 'TickLabelInterpreter', 'latex')
365 xlim(x_limits)
366 ylim(y_limits)
367 title(figure_title)
368 xlabel(xlabel_string)
369 ylabel(ylabel_string)
370 legend([p1, p2], legend_string, 'location', LEGEND_LOCATION, ...
371        'interpreter', 'latex')
372
373
374
375 %% Plot Golf Ball - Method 3
376
377 figure_title = sprintf('Golf Ball - Method 3 Trials');
378
379 legend_string = {'$e$ value for each trial', ...
380         sprintf('mean e value across trial: %0.3f', mean_e_5)};
381
382 textbox_str = {sprintf('std. dev. of e: %0.3f', std_dev_e_5), ...
383         sprintf('std. dev. of the mean of e: %0.3f', std_dev_mean_e_5), ...
```

```
        sprintf('median e value: %0.3f', median_e_5)};


xlabel_string = sprintf('Trial Number');
ylabel_string = sprintf('Value of $e$');
y_limits = [0.75, 1];
x_limits = [1, num_trials_day_2];
LEGEND_LOCATION = 'best';
color = [125;38;205] ./ 255;


golf_ball_plot = figure('name', 'Golf Ball - Method 3');
scrz = get(groot,'ScreenSize');
set(golf_ball_plot, 'Position', scrz)


p2 = plot(trials, mean_vec .* mean_e_5, ':k',...
          'LineWidth', LINEWIDTH);
hold on
p1 = errorbar(e_vec.five.values, e_vec.five.error, 'o',...
              'LineWidth', LINEWIDTH - 1, 'color', color);

annotation('textbox',dim,'String',textbox_str,'FitBoxToText','on', ...
           'interpreter', 'latex', 'fontsize', ...
           FONTSIZE, 'backgroundcolor', 'w');

grid on
set(gca,'FontSize', FONTSIZE)
set(gca, 'defaulttextinterpreter', 'latex')
set(gca, 'TickLabelInterpreter', 'latex')
xlim(x_limits)
ylim(y_limits)
title(figure_title)
xlabel(xlabel_string)
ylabel(ylabel_string)
legend([p1, p2], legend_string, 'location', LEGEND_LOCATION, ...
       'interpreter', 'latex')

%{
for i = 1:10

   figure
   plot(trial_1_data{i}.time, trial_1_data{i}.y)

end

for i = 1:11

   figure
   plot(trial_2_data{i}.time, trial_2_data{i}.y)

end

for i = 1:11

```

```
    figure
    plot(trial_3_data{i}.time, trial_3_data{i}.y)

end
%}
```

```
function trial_data = read_data_files(pixel_to_in, trial_num)

    %%% trial_data = read_data_files(pixel_to_in, trial_num)
    %%%
    %%% Requires that all of the data be organized into following structure
    %%% for the data to be read in:
    %%%
    %%%         '../Data/Trial_xxx/', where xxx = trial number
    %%%
    %%% save data must be in .mat format.
    %%%
    %%% Inputs:
    %%%       - pixel_to_in: defines how many pixels are in one inch
    %%%
    %%%       - tiral_num: this is the trial number of the folder of data
    %%%       to read in. To read in Data from the '../Data/Trial_1/'
    %%%       folder, use trial_num = 1.
    %%%
    %%% Outputs:
    %%%        - cell array containing data from each set of trials as a
    %%%        separate cell. In each cell are cells with the x, y, and
    %%%        time data in structs from each of the trails for that set.
    %%%
    %%% Author: Nicholas Renninger
    %%% Date Created: 2/2/2017
    %%% Last Modified: 2/14/2017
    %%%


    % Trial *trial_num*
    path_trial{1} = sprintf('../Data/Trial_%d/', trial_num);
    folder_path{1} = cat(2, path_trial{1}, '*.mat');
    dir_test{1} = dir(folder_path{1});

    trial_data{1} = cell(1, length(dir_test{1}));



    for j = 1:length(dir_test)

        for k = 1:length(dir_test{j})

            current_directory = dir_test{j};
            filename = cat(2, path_trial{j}, current_directory(k, 1).name);
            current_data = load(filename);

            temp.x = current_data.ball_xy_position_data.x_pixel;
            temp.y = current_data.ball_xy_position_data.y_pixel;
            temp.time = current_data.ball_xy_position_data.time_seconds;

```

```
            % convert from pixels to inches
            % temp.x = temp.x ./ pixel_to_in;
            % temp.y = temp.y ./ pixel_to_in;

            if length(temp.y) ~= length(temp.time)
                error('%s', filename);
            end

            if trial_num < 2
                for i = 1:length(temp.y)

                    if temp.y(i) == 0
                        temp.y(i) = [];
                    elseif i > 30
                        break
                    end

                end

                temp.y = temp.y - min(temp.y(1:8));
            end


            trial_data{k} = temp;

        end

    end



end
```

```
function [e, sigma_e] = calc_e_method_1(h_vec, t_vec, ...
                                        sigma_h_curr, sigma_h_prev)

    %%% [e, sigma_e] = calc_e_method_1(h_vec, t_vec, ...
    %%%                                sigma_h_curr, sigma_h_prev)
    %%%
    %%% Calculates e (coeff. of restitution) using the method 1 formula:
    %%%
    %%%              e = (h_(n) / h_(n-1)) ^ 1/2
    %%%
    %%% and its uncertainty based on the general error propagation formula
    %%% and weighted mean and weighted mean uncertainty theory.
    %%%
    %%% Inputs:
    %%%       - h_vec: a vector of the height of the ball as a function of
    %%%                time.
    %%%
    %%%       - t_vec: a vector containing the time at which each of the
    %%%                points in h_vec was recorded.
    %%%
    %%%       - sigma_h_curr: the uncertainty in the measurement of the
    %%%                       balls height.
```

```
    %%%
    %%%         - sigma_h_prev: the uncertainty in the measurement of the
    %%%                         balls height.
    %%%
    %%% Outputs:
    %%%          - e: value of the coefficient of restitution based on
    %%%          method
    %%%               1 formulation, averaged using the weighted mean across
    %%%               the entire trial.
    %%%
    %%%          - sigma_e: uncertainty of the value of e, with the
    %%%                     uncertainty computed during each pair of bounces
    %%%                     and then combined using the uncertainty of the
    %%%                     weighted mean theory.
    %%%
    %%% Author: Nicholas Renninger
    %%% Date Created: 2/12/17
    %%% Last Modified: 2/15/17

    %% Method 1

    % Define the method for calculating e for use in uncertainty analysis
    syms h_curr h_prev
    e_fcn = @(h_curr, h_prev) sqrt(h_curr / h_prev);

    % isolate good data, i.e. first 5 bounces during the trial
    [height, ~] = findFive(h_vec, t_vec);


    % calc e  and its uncertainty for each bounce up to ~5 bounces
    for i = 2:length(height)

        % compute e
        e_vec(i - 1) = sqrt( height(i) / height(i - 1) );

        % compute sigma_e
        sigma_e_vec(i - 1) = ErrorProp(e_fcn, [h_curr h_prev], ...
                                       [height(i) height(i-1)], ...
                                       [sigma_h_curr, sigma_h_prev]);

    end

    % get avg e val and uncertainty for the trial
    [e, sigma_e] = weightedMean_and_Variance(e_vec', sigma_e_vec');


end

function [e, sigma_e] = calc_e_method_2(h_vec, t_vec, sigma_time)

    %%% [e, sigma_e] = calc_e_method_2(h_vec, t_vec, sigma_time)
    %%%
    %%% Calculates e (coeff. of restitution) using the method 2 formula:
    %%%
    %%%              e = t_(n) / t_(n-1)
    %%%
```

```
%%% and its uncertainty based on the general error propagation formula
%%% and weighted mean and weighted mean uncertainty theory.
%%%
%%% Inputs:
%%%       - h_vec: a vector of the height of the ball as a function of
%%%                time.
%%%
%%%       - t_vec: a vector containing the time at which each of the
%%%                points in h_vec was recorded.
%%%
%%%       - sigma_time: the uncertainty in the measurement of the
%%%                     time between bounces.
%%%
%%%
%%% Outputs:
%%%        - e: value of the coefficient of restitution based on
%%%             method 2 formulation, averaged using the weighted mean
%%%             across the entire trial.
%%%
%%%        - sigma_e: uncertainty of the value of e, with the
%%%                   uncertainty computed during each pair of bounces
%%%                   and then combined using the uncertainty of the
%%%                   weighted mean theory.
%%%
%%% Author: Nicholas Renninger
%%% Date Created: 2/12/17
%%% Last Modified: 2/15/17

%% Method 2

% Define the method for calculating e for use in uncertainty analysis
syms t_curr t_prev
e_fcn = @(t_curr, t_prev) (t_curr / t_prev);

% isolate good data, i.e. first 5 bounces during the trial
[~, times] = findFive(h_vec, t_vec);


% calc e  and its uncertainty for each bounce up to ~5 bounces
for i = 2:length(times)

    % compute e
    e_vec(i - 1) =  times(i) / times(i - 1) ;

    % compute sigma_e
    sigma_e_vec(i - 1) = ErrorProp(e_fcn, [t_curr, t_prev], ...
                                   [times(i), times(i-1)], ...
                                   [sigma_time, sigma_time]);

end

% get avg e val and uncertainty for the trial
[e, sigma_e] = weightedMean_and_Variance(e_vec', sigma_e_vec');
```

```
64 end

1  function [e, sigma_e] = calc_e_method_3(t_vec, h_init, ...
2                                          sigma_t_stop, sigma_h_init)
3
4      %%% [e, sigma_e] = calc_e_method_3(t_vec, h_init, sigma_t_stop, ...
5      %%%                                 sigma_h_init)
6      %%%
7      %%%
8      %%%
9      %%% Calculates e (coeff. of restitution) using the method 3 formula:
10     %%%
11     %%%                                    / 2 * h_init \
12     %%%                                sqrt|--------------|
13     %%%                  t_(stop) -        \      g       /
14     %%%        e =      ------------------------------------
15     %%%                                    / 2 * h_init \
16     %%%                                sqrt|--------------|
17     %%%                  t_(stop) +        \      g       /
18     %%%
19     %%%
20     %%% and its uncertainty based on the general error propagation formula
21     %%% and weighted mean and weighted mean uncertainty theory.
22     %%%
23     %%% Inputs:
24     %%%       - t_vec: a vector containing the time at which each of the
25     %%%                points in h_vec was recorded.
26     %%%
27     %%%       - h_init: the initial height in inches that the ball was
28     %%%                 dropped from.
29     %%%
30     %%%       - sigma_t_stop: the uncertainty of the time until the ball
31     %%%                       stops bouncing.
32     %%%
33     %%%       - sigma_h_init: the uncertainty in the initial height that
34     %%%                       the ball was released at.
35     %%%
36     %%%
37     %%% Outputs:
38     %%%        - e: value of the coefficient of restitution based on
39     %%%             method 3 formulation, averaged using the weighted mean
40     %%%             across the entire trial.
41     %%%
42     %%%        - sigma_e: uncertainty of the value of e, with the
43     %%%                   uncertainty computed during each pair of bounces
44     %%%                   and then combined using the uncertainty of the
45     %%%                   weighted mean theory.
46     %%%
47     %%% Author: Nicholas Renninger
48     %%% Date Created: 2/12/17
49     %%% Last Modified: 2/15/17
50
51
52     %% Method 3
53
54     % define gravitational acceleration, at S.L.
```

```
    g = 386.4; % [in / s^2]

    % Define the method for calculating e for use in uncertainty analysis
    syms t_stop h_o
    e_fcn = @(t_stop, h_o) ( t_stop - sqrt((2*h_o) / g) ) / ...
                           ( t_stop + sqrt((2*h_o) / g) );


    % compute e
    t_start = t_vec(1);
    t_end = t_vec(end);
    time_stop = t_end - t_start;

    e = e_fcn(time_stop, h_init);

    % compute sigma_e
    sigma_e = ErrorProp(e_fcn, [t_stop, h_o], ...
                               [time_stop, h_init], ...
                               [sigma_t_stop, sigma_h_init]);

end
```

```
function totError = ErrorProp(func, dependents, vals, dependentError)

    %%% totError = ErrorProp(func, dependents, vals, dependentError)
    %%%
    %%% Uses a numeric formulation of the general error propogation formula
    %%% to calculate the total uncertainty of calculated value based on
    %%% the uncertainty in the measurements used to compute the value.
    %%%
    %%% Ex. function call:
    %%%
    %%%     syms ho hn
    %%%     e = @(ho,hn) ((hn/ho).^(1/2));
    %%%     sigma_ho = 0.1;
    %%%     sigma_hn = 0.2;
    %%%     sigma_e = ErrorProp(e,[hn ho], [10, 8], [sigma_ho, sigma_hn]);
    %%%
    %%% Inputs:
    %%%       - func: inline function that maps the measures values to the
    %%%               calculated value.
    %%%
    %%%       - dependents: symbolic arguments to func, i.e. the
    %%%                     measurement variables that might have error in
    %%%                     them.
    %%%
    %%%       - vals: values of the dependent variables when used to
    %%%               calculate the independen variable.
    %%%
    %%%       - dependentError: the uncertainty in the measured values of
    %%%                         the dependent variables.
    %%%
    %%% Author: Jeffrey Mariner Gonzalez - modified from a ...
    %%% Date Created: 2/5/17
    %%% Last Modified: 2/15/17
    %%%
```

```matlab

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    x = numel(dependents);
    temp = vpa(ones(1,x));

    % Compute Partials
    for i = 1:x
        temp(i) = diff(func,dependents(i),1);
    end

    % Sum in quadrature to compute general error
    totError = sqrt( ( sum( (subs(temp,dependents,vals).^2) .* ...
                            (dependentError.^2) ) ) );
    totError = double(totError);

end
```

```matlab
function [height, time] = findFive(y, tIN)

    %%% function [height, time] = findFive(y, tIN)
    %%%
    %%% Function takes a set of data and returns the data corresponding to
    %%% the first five bounces of the ball.
    %%%
    %%% Inputs:
    %%%       - y: vector containing the height of the ball as function of
    %%%            the time vector, tIN.
    %%%
    %%%       - tIN: vector containing the times corresponding to the
    %%%              measured values of height, y.
    %%%
    %%% Outputs:
    %%%        - height: vector containing only the peak height of
    %%%                  the first five bounces.
    %%%
    %%%        - time: vector containing the airborne time of each bounce
    %%%
    %%% Author: Jeffrey Mariner Gonzalez
    %%% Date Created: 2/10/17
    %%% Last Modified: 2/15/17


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % find the max height of first 5 bounces
    PKS = findpeaks(y);
    height = PKS(1:5);

    % find when it hits the ground
    updown = max(y) - y;
    [~, minIDX] = findpeaks(updown);

    % find time when it hits the ground
    temp = tIN(minIDX);
    temp = temp(1:6);
```

```
39
40      % find how long each bounce takes
41      for i = 1:5
42          time(i) = temp(i+1) - temp(i);
43      end
44
45  end
```

```
1   function [weightedMean, errorMean] = weightedMean_and_Variance(data, error)
2
3       %%% [weightedMean, errorMean] = weightedMean_and_Variance(data, error)
4       %%%
5       %%%
6       %%%
7       %%% Inputs:
8       %%%         - h_vec: a vector of the height of the ball as a function of
9       %%%                  time.
10      %%%
11      %%%         - t_vec: a vector containing the time at which each of the
12      %%%                  points in h_vec was recorded.
13      %%%
14      %%%         - sigma_h_curr: the uncertainty in the measurement of the
15      %%%                         balls height.
16      %%%
17      %%%         - sigma_h_prev: the uncertainty in the measurement of the
18      %%%                         balls height.
19      %%%
20      %%% Outputs:
21      %%%          - e: value of the coefficient of restitution based on
22      %%%          method
23      %%%               1 formulation, averaged using the weighted mean across
24      %%%               the entire trial.
25      %%%
26      %%%          - sigma_e: uncertainty of the value of e, with the
27      %%%                     uncertainty computed during each pair of bounces
28      %%%                     and then combined using the uncertainty of the
29      %%%                     weighted mean theory.
30      %%%
31      %%% Author: Nicholas Renninger
32      %%% Date Created: 2/12/17
33      %%% Last Modified: 2/15/17
34
35      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
36
37      % calculate weights based on statistical variance of data (error mat.)
38      weights = 1 ./ error.^2;
39
40      % check if any of the weights is zero and fix it
41      if ~isempty(find(weights == inf, 1))
42
43          [r, c] = find(weights == inf);
44
45          for i = 1:length(r)
46
47              % re make weight based on super small error for zero error
48              non_zero_errors = error(error(:, c(i)) ~= 0);
```

```
            new_error = min(non_zero_errors(:, 1)) * 1e-3;
            weights(r(i), c(i)) = 1 / new_error^2;

        end
    end



    %% calculate sums for the weighted mean of the matrix
    [numPoints, numVars] = size(data);

    [sumOfAvgPoints, sumOfWeights] = deal(zeros(1, numVars));

    for i = 1:numVars
        for j = 1:numPoints

            sumOfAvgPoints(i) = sumOfAvgPoints(i) + ...
                                weights(j, i) .* data(j, i);
            sumOfWeights(i) = sumOfWeights(i) + weights(j, i);
        end
    end

    weightedMean = sumOfAvgPoints ./ sumOfWeights;



    %% calculate the variance of the weighted mean

    errorMean = sqrt(1 ./ sumOfWeights);


end
```

```
function sensitivityAnalysis

    %%% sensitivityAnalysis
    %%%
    %%%     Will create plots showing the effect on the uncertainty of e by
    %%%     the variation of the uncertainty of each dependent variable
    %%%
    %%% Inputs:
    %%%         none
    %%%
    %%% Outputs:
    %%%          none
    %%%
    %%% Author: Jeffrey Mariner Gonzalez, Nicholas Renninger
    %%% Date Created: 2/12/17
    %%% Last Modified: 2/15/17

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % define gravitational acceleration, at S.L.
    g = 386.4; % [in / s^2]

    LINEWIDTH = 3;
```

```
FONTSIZE = 26;

% magnitude of uncertainties for each dependent variable
sigt = 0.01;
sigh = 0.5;
sigt2 = 0.01;
sigh2 = 1;

% defining fcns. for each method
syms h_curr h_prev t_curr t_prev ts ho
e1 = @(h_curr, h_prev) sqrt(h_curr / h_prev);
e2 = @(t_curr, t_prev) (t_curr / t_prev);
e3 = @(ts,ho) ((ts-sqrt(2*ho/g))/(ts+sqrt(2*ho/g)));
k = 0:.1:4;

% calculate how the uncertainty in e changes  with the uncertainty in
% each dependent variable
for i = 1:length(k)

    hcurre1(i) = ErrorProp(e1, [h_curr h_prev], ...
                                        [40 35], ...
                                        [sigh*k(i), sigh]);
    hpreve1(i) = ErrorProp(e1, [h_curr h_prev], ...
                                        [40 35], ...
                                        [sigh, sigh*k(i)]);
    tcurre2(i) = ErrorProp(e2, [t_curr t_prev], ...
                                        [.5 .3], ...
                                        [sigt*k(i), sigt]);
    tpreve2(i) = ErrorProp(e2, [t_curr t_prev], ...
                                        [.5 .3], ...
                                        [sigt, sigt*k(i)]);
    tse3(i) = ErrorProp(e3, [ts ho], ...
                                        [10 40], ...
                                        [sigt2*k(i), sigh2]);
    hoe3(i) = ErrorProp(e3, [ts ho], ...
                                        [10 40], ...
                                        [sigt2, sigh2*k(i)]);
end


%% method 1

figure_title = sprintf(['Sensitivity Analysis of Method 1', ...
                        ' - Uncertainty in Measured Height']);
legend_string = {'$\delta h_n$', '$\delta h_{n-1}$'};
xlabel_string = sprintf('$\\delta_{height}$ [in.]');
ylabel_string = sprintf('$\\delta_{e}$');
LEGEND_LOCATION = 'best';

sens_anal_1_plot = figure('name', 'Sensitivity Analysis - Method 1');
scrz = get(groot,'ScreenSize');
set(sens_anal_1_plot, 'Position', scrz)

%h current
plot(sigh*k, hcurre1, 'LineWidth', LINEWIDTH);
```

```
79
80      hold on;
81
82      % h prev
83      plot(sigh*k, hpreve1, 'LineWidth', LINEWIDTH);
84
85
86      grid on
87      set(gca,'FontSize', FONTSIZE)
88      set(gca, 'defaulttextinterpreter', 'latex')
89      set(gca, 'TickLabelInterpreter', 'latex')
90      title(figure_title)
91      xlabel(xlabel_string)
92      ylabel(ylabel_string)
93      legend(legend_string, 'location', LEGEND_LOCATION, ...
94             'interpreter', 'latex')
95
96      %%% end figure one %%%
97
98
99      %% method 2
100
101     figure_title = sprintf(['Sensitivity Analysis of Method 2', ...
102                             ' - Uncertainty in Measured Time']);
103     legend_string = {'$\delta t_n$', '$\delta t_{n-1}$'};
104     xlabel_string = sprintf('$\\delta_{time}$ [in.]');
105     ylabel_string = sprintf('$\\delta_{e}$');
106     LEGEND_LOCATION = 'best';
107
108     sens_anal_2_plot = figure('name', 'Sensitivity Analysis - Method 2');
109     scrz = get(groot,'ScreenSize');
110     set(sens_anal_2_plot, 'Position', scrz)
111
112
113     plot(sigt*k, tcurre2,'LineWidth', LINEWIDTH);
114
115     hold on;
116
117     plot(sigt*k, tpreve2, 'LineWidth', LINEWIDTH);
118
119     grid on
120     set(gca,'FontSize', FONTSIZE)
121     set(gca, 'defaulttextinterpreter', 'latex')
122     set(gca, 'TickLabelInterpreter', 'latex')
123     title(figure_title)
124     xlabel(xlabel_string)
125     ylabel(ylabel_string)
126     legend(legend_string, 'location', LEGEND_LOCATION, ...
127            'interpreter', 'latex')
128
129     %%% end figure two %%%
130
131     %% method 3 pt 1
132
133     figure_title = sprintf(['Sensitivity Analysis of Method 3', ...
```

```
                                ' - Uncertainty in Time to Stop Measurement']);
    xlabel_string = sprintf('$\\delta_{time}$ [in.]');
    ylabel_string = sprintf('$\\delta_{e}$');

    sens_anal_3_plot = figure('name', 'Sensitivity Analysis - Method 3');
    scrz = get(groot,'ScreenSize');
    set(sens_anal_3_plot, 'Position', scrz)

    plot(sigt2*k, tse3, 'LineWidth', LINEWIDTH);

    grid on
    set(gca,'FontSize', FONTSIZE)
    set(gca, 'defaulttextinterpreter', 'latex')
    set(gca, 'TickLabelInterpreter', 'latex')
    title(figure_title)
    xlabel(xlabel_string)
    ylabel(ylabel_string)

    %end figure 3

    %% method 3 pt 2

    figure_title = sprintf(['Sensitivity Analysis of Method 3', ...
                            ' - Uncertainty in Initial Height Measurement']);

    xlabel_string = sprintf('$\\delta_{height}$ [in.]');
    ylabel_string = sprintf('$\\delta_{e}$');


    sens_anal_4_plot = figure('name', 'Sensitivity Analysis - Method 3');
    scrz = get(groot,'ScreenSize');
    set(sens_anal_4_plot, 'Position', scrz)

    plot(sigh2*k, hoe3, 'LineWidth', LINEWIDTH);

    grid on
    set(gca,'FontSize', FONTSIZE)
    set(gca, 'defaulttextinterpreter', 'latex')
    set(gca, 'TickLabelInterpreter', 'latex')
    title(figure_title)
    xlabel(xlabel_string)
    ylabel(ylabel_string)


end
```