ASEN 2003 Lab 4:
Balanced and Unbalanced Wheel
Section 013
March 16, 2017

Joseph Grengs* , Kian Tanner† , Nicholas Renninger‡
University of Colorado - Boulder

The impact of mass and inertia distribution on the motion of a rigid body was explored in this lab. A balanced and an unbalanced wheel were rolled down a ramp. The angular position of the wheel was recorded, from which the angular velocity was numerically calculated. The experimental data was then compared to a model we constructed using energy methods. The first model did not account for the friction from the bearings, so an empirical constant was determined ($M = 0.984\ N * m$). The mean residual for models one, two, three, and four were approximately -1.04 $\frac{rad}{s}$, 0.09 $\frac{rad}{s}$, 0.2 $\frac{rad}{s}$, and 0.2 $\frac{rad}{s}$, respectively. This shows the necessity of the empirical constant. As the extra mass is rotating at the same rate as the main wheel, treating the extra mass as a three-dimensional object rather than as a point mass is not important to the accuracy of the model.

Nomenclature

β	=	Angle of ramp $[rad]$
θ	=	Angle of wheel $[rad]$
d_{em}	=	Distance from center of wheel to extra mass $[m]$
g	=	Gravitational acceleration $[m/s^2]$
I_w	=	Moment of inertia of wheel $[kg \cdot m^2]$
M	=	Empirical Constant to model frictional losses $[N \cdot m]$
m_w	=	Mass of wheel $[kg]$
m_{em}	=	Mass of extra mass $[kg]$
m_{ta}	=	Mass of trailing apparatus $[kg]$
R_w	=	Radius of wheel $[m]$
R_{em}	=	Radius of extra mass $[kg]$
$R_{gy,w}$	=	Radius of gyration of wheel $[m]$

*SID: 104429750
†SID: 105010299
‡SID: 105492876

Contents

I. Model

The equations for ω, as defined by each model, are the following derived Eqns. 1 - 4. Each of these models were determined using principles of conservation of energy and used the given geometry of the system as shown in Figure 2. A comparison between the models can be found in Figure 1, demonstrating the vast difference in the motion of the cylinder when the extra mass is added. The full derivation of each model can be found in Appendix B. The MATLAB code used to implement these models in the analysis of experimental data is included in Appendix C.

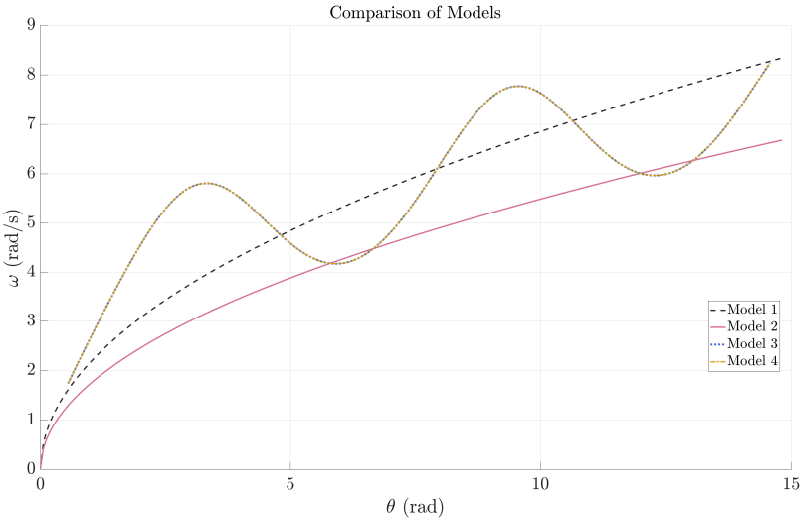


Figure 1: Comparison of Models 1-4. *Models 3 and 4 are indistinguishable, while models 1 and 2 present very different results*

Model 1, given by Eqn. 1, is an equation derived for the rotation of a cylinder, which has no additional mass added. A more detailed look at this model's performance can be found in Figure 4 and in Table 1. The model does not take into account the empirical constant M, which is a part of models 2-4 (Eqns. 2 - 4), so it does not take into account frictional losses.

$$\omega = \sqrt{\frac{2gR_w\theta sin(\beta)(m_w + m_{ta})}{R_w^2(m_w + m_{ta}) + I_w}} \tag{1}$$

Model 2, given by 2, is the same as model 1, with the empirical constant, M, added into the equation. A more detailed look at this model's performance can be found in Figure 4, and in Table 1.

$$\omega = \sqrt{\frac{2\theta[gR_w sin(\beta)(m_w + m_{ta}) - M]}{R_w^2(m_w + m_{ta}) + I_w}} \tag{2}$$

Model 3, given by Eqn. 3 is an equation derived for the rotation of the same cylinder as model 1. However, in this case, an extra rod was added on one side of the cylinder. Additionally, the extra mass is assumed to be dimensionless and will be treated as a point mass. A more detailed look at this model's performance can be found in Figure 5 and in Table 1.

$$\omega = \sqrt{\frac{2[gRw\theta sin(\beta)(m_w + m_{ta} + m_{em}) - M\theta - gd_{em}(cos(\theta + \beta) - cos(\beta))]}{R_w^2(m_w + m_{ta}) + I_w + m_{em}(R_w^2 + 2R_w d_{em} cos(\theta) + d_{em}^2)}} \tag{3}$$

Model 4, given by Eqn. 4 is the same as model 3, except model 4 accounts for the dimensions of the extra rod. Additionally, the extra rod is fixed within the cylinder and therefore rotates at the same rate as the cylinder. A more detailed look at this model's performance can be found in Figure 5 and in Table 1.

$$\omega = \sqrt{\frac{2[gRw\theta sin(\beta)(m_w + m_{ta} + m_{em}) - M\theta - gd_{em}(cos(\theta + \beta) - cos(\beta))]}{R_w^2(m_w + m_{ta}) + I_w + \frac{1}{2}m_{em}R_{em}^2 + m_{em}(R_w^2 + 2R_w d_{em} cos(\theta) + d_{em}^2)}} \tag{4}$$

II. Experiment

To understand how inertia plays a role in motion, we begin this experiment by creating a setup of a large cylinder, which is placed on an inclined ramp.[1] The cylinder has an attached encoder located about its axle. Also affixed to the cylinder is an extension to a small wheel, which is used to control the cylinder, as it rolls down the incline. The incline itself is covered by a sort of rubber-like material, so the cylinder can only roll down it without slipping.

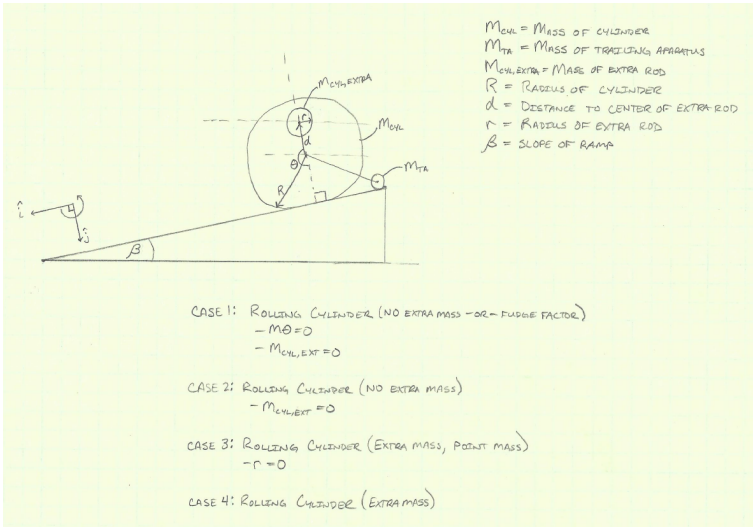


Figure 2: Diagram of the Experimental Setup.

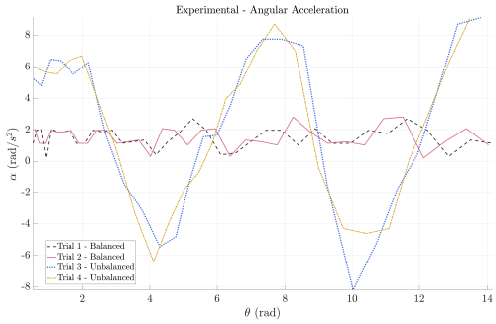
To start the experiment, the cylinder is placed at the top of the ramp and held in place by a team member, as well as a small stop being placed under the cylinder. Once the cylinder is aligned to be pointing straight down the ramp, the stop is quickly pulled out from under it, to allow it to roll, unimpeded, down the incline. While the cylinder is rolling down the incline, the encoder is measuring the angular displacement and angular velocity. Once the cylinder reaches the end of the incline, a team member catches the cylinder, to keep it from hitting the hard floor and breaking.

To fully understand moments of inertia, two different experiments were conducted. The first is a balanced wheel, in which the cylinder has no extra mass added asymmetrically. The second experiment has an extra mass added on one side of the wheel. The extra mass is affixed into the cylinder, so it does not rotate, with respect to the cylinder.

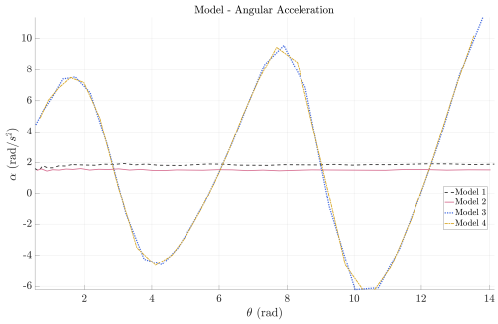
In observing the behavior of the cylinder as it rolls down the incline, it is noted that the two different cases exhibit varying motions. In the first experiment, the cylinder rolls smoothly down the incline, without any "hiccups" as shown in Figure 4. However, adding the additional mass in the second experiment, the cylinder appears to be off center, as an observer can see the cylinder accelerate down the incline when the extra mass is on the front half of it, and decelerates when the extra mass is on the back half of the cylinder. This can be seen by the sinusoidally varying ω, as seen in Figure 5.

III. Results and Analysis

With a few exceptions, the models matched the experimental data very well. When looking at the residuals of model one, (Fig. 4) it can be seen that the residuals grow larger as the wheel spins, indicating a bias in the system. This clearly shows the need for the empirical constant to account for the friction in the bearings. Using a Monte Carlo simulation, we determined the value of M best fitting model 2 to be $0.984N * m$. This empirical constant is important to the model. When looking at models three and four, (Fig. 5) we can see that that the models are very similar. In addition, in both trials 3 and 4, the only change in the residual statistics was that the magnitude of σ decreased by 0.001 (Table 1). This shows that modeling the extra mass as a rod, instead of a cylinder is not a necessary part of the model. The error in model 1 is mostly bias caused by the friction in the bearings. Model 2 accounts for this source of error, and so the error in model 2 is mostly noise. This can be seen by looking at the graph of the residuals for model 2 (Fig. 4); the residuals for trials one and two have different shapes. The error in models three and four, however, seems to be more systematic. They are not biased, as the means of their residuals are low, (Table 1) but the residuals of both trials three and four have roughly the same shape, (Fig. 5) meaning the error is systematic. This is likely caused by either an error in the model, or a misalignment of the starting position of the wheel. There still is some noise in the residuals, but it doesn't contribute to the error as much as other sources.



(a) Angular Acceleration Calculated for the Experimental Data. *The balanced wheel here is shown to have a fairly constant, always positive, angular acceleration, with its slight sinusoidal nature more likely due to measurement error than to its actual behavior. The unbalanced wheel experiences very sinusoidal angular acceleration, as the extra mass creates torques about the center of the larger cylinder which cause it to accelerate and decelerate.*



(b) Angular Acceleration Calculated for the Model Data. *The model data matches the the experimental angular acceleration, with the balanced wheel exhibiting largely constant, continuously positive angular acceleration, while the unbalanced wheel exhibited very sinusoidal behavior owing to the torque contributions from the extra mass.*

Figure 3: Angular Accelerations Calculated from Model and Experimental Data

Table 1: Residual Statistics

Model	**Trial**	$\boldsymbol{\sigma}$ $[\frac{rad}{s}]$	**Mean Residual** $[\frac{rad}{s}]$	$\frac{\boldsymbol{\sigma}}{\sqrt{\boldsymbol{N}}}$ $[\frac{rad}{s}]$	$\boldsymbol{N}$	**Outliers** $> 3\sigma$
1	1	0.4	-1.04	0.06	38	0
1	2	0.4	-0.953	0.07	36	0
2	1	0.09	-0.0624	0.01	38	0
2	2	0.06	0.0421	0.01	36	1
3	3	0.2	-0.11	0.03	28	0
3	4	0.2	-0.115	0.04	27	0
4	3	0.2	-0.109	0.03	28	0
4	4	0.2	-0.114	0.04	27	0

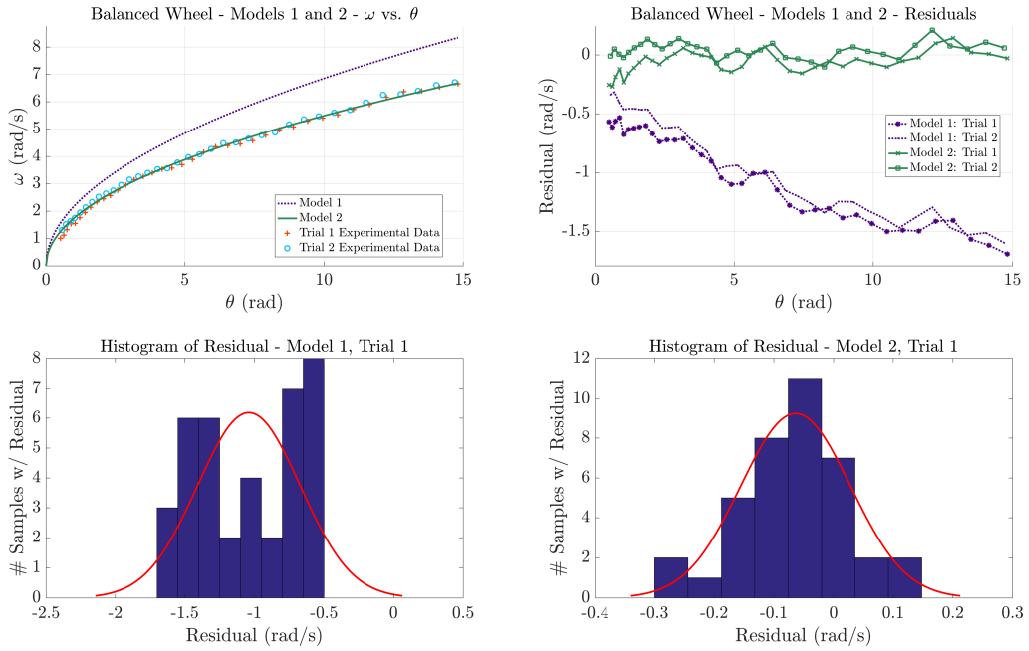


Figure 4: Experimental Data, Model Data (models 1 & 2), Residual Time Histories, and Residual Histograms for the Balanced Wheel Trials. *For the balanced wheel, models 1 & 2 were used (See Eqns. 1 & 2 or Appendix B) as shown in the top left subplot. Model 2 fit the experimental data much better, as can be seen in the residual time histories. Model 2 had residuals uniformly distributed about a 0 rad/s mean, while model 1 had a very large negative bias in the residuals, indicating Model 2's accuracy. Also, as seen in the histograms, model 2's residuals fit a Gaussian distribution about 0 while model 1 did not, again indicating model 2's better fit.*

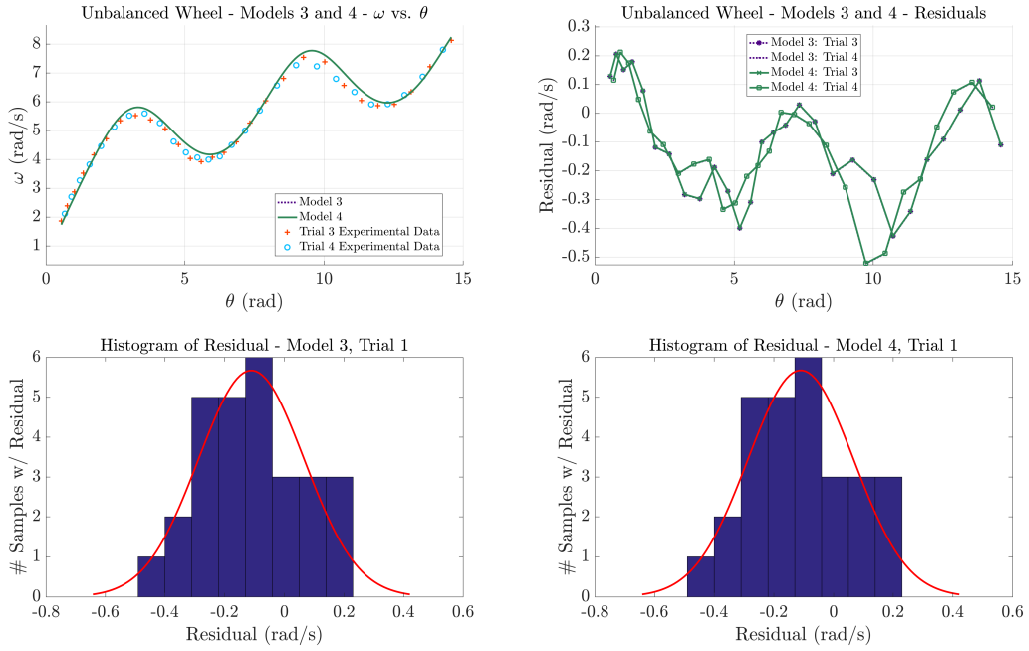


Figure 5: Experimental Data, Model Data (models 3 & 4), Residual Time Histories, and Residual Histograms for the Unbalanced Wheel Trials. *For the Unbalanced wheel, models 3 & 4 were used (See Eqns. 3 & 4 or Appendix B) as shown in the top left subplot. Models 3 & 4 were shown to be indistinguishable, and matched the experimental data very well, for the same reasons given in the caption of figure 5.*

IV. Conclusions and Recommendations

Comparing the data between model 3 and model 4, as seen in figure 1, it was determined that the difference between treating the extra rod as a point mass versus a three dimensional object, is negligible. This is due to the extra rod rotating at the same rate as the cylinder, which adds or subtracts negligible energy. Adding the extra dimension does produce a change in the model. However, it is too small for a macroscopic analysis, so it can be ignored. For other complex bodies (structures within structures) in which the inertial energy is small compared to the overall structure, the model of motion can be simplified by treating smaller structures as a point mass.

To complete an accurate analysis of the motion of a cylinder down an incline, a properly derived model of the motion must be completed and accurately compared to the data collected. In this case, the model was not fully derived, as an additional empirical constant was added, to account for friction, drag and any other force acting on the cylinder that was not accounted for. This empirical constant was determined by approximating a "best-fit" value using a Monte-Carlo simulation. Completing this process is not the most efficient, or the most accurate way of determining the missing factors from the model. In future experiments, some of these other factors should be included in the model, to more accurately predict the motion of the cylinder.

References

[1]Axelrad. ASEN 2003 Lab 4: Balanced and Unbalanced Wheel. CU, 2017. PDF.

Acknowledgments

Thank you to Professor Penina Axelrad for the lectures regarding the information that directly pertain to this experiment. Thanks to Bobby Hodgkinson for the instructions of the lab and explaining the methods for collecting the data. Thank you to the Teaching and Learning Assistants of Dynamics and Systems for assisting us with the derivations and explanations of the results of our experiment.

Appendix A

The primary contributions of each lab member are as follows:

Joseph Grengs: Experimentation, Coding, Derivations, Final Report.

Kian Tanner: Experimentation, Derivations, Final Report.

Nicholas Renninger: Coding, Derivations, Final Report.

Appendix B: Derivations

Case 1: No Extra Mass, No Added Friction Factor

$$PE_0 = KE_1 \tag{5}$$
$$PE_0 = gh_0(m_w + m_{ta}) \tag{6}$$
$$v = R_w\omega \tag{7}$$
$$KE_1 = \frac{1}{2}(m_w + m_{ta})v^2 + \frac{1}{2}I_w\omega^2 = \frac{1}{2}\omega^2[(m_w + m_{ta})R_w^2 + I_w] \tag{8}$$
$$gh_0(m_w + m_{ta}) = \frac{1}{2}\omega^2[(m_w + m_{ta})R_w^2 + I_w] \tag{9}$$
$$\omega^2 = \frac{2gh_0(m_w + m_{ta})}{R_w^2(m_w + m_{ta}) + I_w} \tag{10}$$
$$h_0 = R_w\theta sin(\beta) \tag{11}$$
$$\omega = \sqrt{\frac{2gR_w\theta sin(\beta)(m_w + m_{ta})}{R_w^2(m_w + m_{ta}) + I_w}} \tag{12}$$

Case 2: No Extra Mass, Added Friction Factor

$$PE_0 = KE_1 \tag{13}$$
$$PE_0 = gh_0(m_w + m_{ta}) - M\theta \tag{14}$$
$$KE_1 = \frac{1}{2}(m_w + m_{ta})v^2 + \frac{1}{2}I_w\omega^2 = \frac{1}{2}\omega^2[(m_w + m_{ta})R_w^2 + I_w] \tag{15}$$
$$gh_0(m_w + m_{ta}) - M\theta = \frac{1}{2}\omega^2[(m_w + m_{ta})R_w^2 + I_w] \tag{16}$$
$$\omega^2 = \frac{2[gh_0(m_w + m_{ta}) - M\theta]}{R_w^2(m_w + m_{ta}) + I_w} \tag{17}$$
$$\omega = \sqrt{\frac{2\theta[gR_w sin(\beta)(m_w + m_{ta}) - M]}{R_w^2(m_w + m_{ta}) + I_w}} \tag{18}$$

Case 3: Extra Point Mass, Added Friction Factor

$$PE_0 = KE_1 + PE_1 \tag{19}$$
$$PE_0 = gh_0(m_w + m_{ta} + m_{em}) - M\theta \tag{20}$$
$$PE_1 = gh_1 m_{em} \tag{21}$$

$$\begin{aligned} I_{em} &= m_{em}[(R_w + d_{em}cos(\theta))^2 + (d_{em}cos(\theta))^2] \\ &= m_{em}(R_w^2 + 2R_w d_{em}cos(\theta) + d_{em}^2cos(\theta)^2 + d_{em}^2sin(\theta)^2) \\ &= m_{em}(R_w^2 + 2R_w d_{em}cos(\theta) + d_{em}^2) \end{aligned} \tag{22}$$

$$\begin{aligned} KE_1 &= \frac{1}{2}(m_w + m_{ta})v^2 + \frac{1}{2}I_w\omega^2 + \frac{1}{2}I_{em}\omega^2 \\ &= \frac{1}{2}\omega^2[(m_w + m_{ta})R_w^2 + I_w + I_{em}] \\ &= \frac{1}{2}\omega^2[(m_w + m_{ta})R_w^2 + I_w + m_{em}(R_w^2 + 2R_w d_{em}cos(\theta) + d_{em}^2)] \end{aligned} \tag{23}$$

$$PE_0 - PE_1 = KE_1 \tag{24}$$

$$gh_0(m_w + m_{ta} + m_{em}) - M\theta - gh_1 m_{em} = \frac{1}{2}\omega^2[(m_w + m_{ta})R_w^2 + I_w + m_{em}(R_w^2 + 2R_w d_{em} cos(\theta) + d_{em}^2)] \tag{25}$$

$$\omega^2 = \frac{2[gh_0(m_w + m_{ta} + m_{em}) - M\theta - gh_1 m_{em}]}{R_w^2(m_w + m_{ta}) + I_w + m_{em}(R_w^2 + 2R_w d_{em} cos(\theta) + d_{em}^2)} \tag{26}$$

$$h_1 = d_{em}(cos(\theta + \beta) - cos(\beta)) \tag{27}$$

$$\omega = \sqrt{\frac{2[gRw\theta sin(\beta)(m_w + m_{ta} + m_{em}) - M\theta - gd_{em}(cos(\theta + \beta) - cos(\beta))]}{R_w^2(m_w + m_{ta}) + I_w + m_{em}(R_w^2 + 2R_w d_{em} cos(\theta) + d_{em}^2)}} \tag{28}$$

Case 4: Extra Mass, Added Friction Factor

Case 4 is the same as case 3, except that the added mass has rotational kinetic energy.

$$KE_1 = \frac{1}{2}\omega^2[(m_w + m_{ta})R_w^2 + I_w + m_{em}(R_w^2 + 2R_w d_{em} cos(\theta) + d_{em}^2) + \frac{1}{2}m_{em}R_{em}^2] \tag{29}$$

$$PE_0 - PE_1 = KE_1 \tag{30}$$

$$gh_0(m_w + m_{ta} + m_{em}) - M\theta - gh_1 m_{em} = \frac{1}{2}\omega^2[(m_w + m_{ta})R_w^2 + I_w + m_{em}(R_w^2 + 2R_w d_{em} cos(\theta) + d_{em}^2) + \frac{1}{2}m_{em}R_{em}^2] \tag{31}$$

$$\omega^2 = \frac{2[gh_0(m_w + m_{ta} + m_{em}) - M\theta - gh_1 m_{em}]}{R_w^2(m_w + m_{ta}) + I_w + m_{em}(R_w^2 + 2R_w d_{em} cos(\theta) + d_{em}^2) + \frac{1}{2}m_{em}R_{em}^2} \tag{32}$$

$$\omega = \sqrt{\frac{2[gh_0(m_w + m_{ta} + m_{em}) - M\theta - gh_1 m_{em}]}{R_w^2(m_w + m_{ta}) + I_w + m_{em}(R_w^2 + 2R_w d_{em} cos(\theta) + d_{em}^2) + \frac{1}{2}m_{em}R_{em}^2}} \tag{33}$$

Appendix C: MATLAB Code

Main.m

```
%% ASEN 2003: Dynamics & Systems – Spring 2017
% Project: Rolling Wheel Lab (#4)
% Project Members:  Joseph Grengs
%                   Kian Tanner
%                   Nicholas Renniger
% Project Due Date: Thursday, March 16, 2017 @ 4:00p
% MATLAB Code Created on: 03/02/2017
% MATLAB Code last updated on: 03/14/2017

clear
close all
clc

%% Project Scope:
% This project is designed to analyze a rolling wheel down a ramp under two
% different scenarios.  The first is a uniform cylinder and the second is a
% cylinder with an extra mass added to one side of the cylinder.  This code
% will analyze the data collected from a LABVIEW (.vi) and compare it to a
% analytical model.


%% Inputs/Outputs:
% Inputs: Data files collected from LABVIEW (vi) and measured constants
% Outputs: Measured and modeled angular velocity vs. time plots
%          Residual data plots
%          Statistical information from data

%% Run Program:

%%% Set constants %%%
Mc = 11.7;                          % Mass of cylinder (kg)
Mta = 0.7;                          % Mass of trailing apparatus (kg)
Mem = 3.4;                          % Mass of extra mass (kg)
Rc = 0.235;                         % Radius of cylinder (m)
k = 0.203;                          % Radius of gyration (m)
Ic = Mc * k^2;                      % MOI of cylinder (kg*m^2)
Beta = 5.5 * (pi/180);              % Angle of ramp (radians)
Rtem = 0.178;                       % Radius to extra mass (m)
Rem = 0.019;                        % Radius of extra mass (m)
g = 9.81;                           % Gravitational acceleration (m/s^2)
NUM_PLOT_PTS = 250;                 % number of pts to plot w/ model
shouldSaveFigures = false;           % decide wheter or not to save plots

constants = [Mc Mta Mem Rc k Ic Beta Rtem Rem g ...
             NUM_PLOT_PTS shouldSaveFigures];

load_and_Analyze_Data(constants);   % Import data and begin analysis
```

load_and_Analyze_Data.m

```
%% ASEN 2003: Dynamics & Systems - Spring 2017
% Project: Rolling Wheel Lab (#4)
% Project Members:  Joseph Grengs
%                   Kian Tanner
%                   Nicholas Renniger
%
%
% Function takes all of the constants defined in the main script and
% calculates the optimal M value with a monte carlo simulation, then
% calculates the expected omega values using each model, calculates
% statistics on the residual, then plots theh results.
%
% Project Due Date: Thursday, March 16, 2017 @ 4:00p
% MATLAB Code Created on: 03/02/2017
% MATLAB Code last updated on: 03/15/2017


function load_and_Analyze_Data(constants)

    unBal_Idx = 1;
    bal_Idx = 1;
    NUM_PLOT_PTS = constants(end - 1);
    shouldSaveFigures = constants(end);

    % Look for data files in directory
    listing = dir('*.txt');

    for i = 1:length(listing)
        %% Read in Files
        filename = listing(i);
        name = filename.name;

        data = load(name);
        search = find(data(:,2) > 0.5 & data(:,2) < 15);
        theta = data(search, 2);
        omega = data(search, 3);

        [~, val] = sort(theta);

        % determine if the data is from unbalanced or balanced test trials
        % and save it in the appropriate structure
        if char(name(1:2)) == ('un')

            theta_exp.unbalanced{unBal_Idx} = theta(val);

            % vector containing the omega used for the model calculation
            theta_mod.unbalanced{unBal_Idx} = linspace(0, ...
                                                max(theta), ...
                                                NUM_PLOT_PTS);

            omega_exp.unbalanced{unBal_Idx} = omega(val);
            unBal_Idx = unBal_Idx + 1;

        else

```

```
56              theta_exp.balanced{bal_Idx} = theta(val);
57
58              % vector containing the omega used for the model calculation
59              theta_mod.balanced{bal_Idx} = linspace(0, ...
60                                                     max(theta), ...
61                                                     NUM_PLOT_PTS);
62
63              omega_exp.balanced{bal_Idx} = omega(val);
64              bal_Idx = bal_Idx + 1;
65
66          end
67
68
69      end
70
71      %% Calculate M using the data from the balanced trial, using Model 2
72
73      tic
74      modelUsed = 2;
75      M_lims = [0, 1.5]; % set limits for monte carlo search for optimal M
76
77      % find optimal M for each trial run and avg. to find experimental M
78      for i = 1:length(omega_exp.balanced)
79          M_vec(i) = monteCarloCalcM(M_lims, omega_exp.balanced{i}, ...
80                                theta_exp.balanced{i}, constants, modelUsed);
81      end
82      toc
83
84      % avg the M found for each trial
85      M = mean(M_vec);
86
87      fprintf('Optimal Value of M: %0.3g N*m\n', M)
88
89
90      % Calculate the Model 1 and 2 Omega and their Residuals
91      for i = 1:length(theta_exp.balanced)
92
93          modelUsed = 1;
94          omega_mod_res_1{i} = calcModelOmega(theta_exp.balanced{i}, M, ...
95                                              constants, modelUsed);
96          omega_mod_1{i} = calcModelOmega(theta_mod.balanced{i}, M, ...
97                                              constants, modelUsed);
98
99          modelUsed = 2;
100         omega_mod_res_2{i} = calcModelOmega(theta_exp.balanced{i}, M, ...
101                                             constants, modelUsed);
102         omega_mod_2{i} = calcModelOmega(theta_mod.balanced{i}, M, ...
103                                             constants, modelUsed);
104
105         % Calculate Residuals for both models
106         residual_vec_model_1{i} = ( omega_exp.balanced{i} - ...
107                                     omega_mod_res_1{i} )';
108         residual_vec_model_2{i} = ( omega_exp.balanced{i} - ...
109                                     omega_mod_res_2{i} )';
110
```

```
111
112         % Calculate Statistics for Models 1 and 2
113         dataType = sprintf('Balanced Wheel Trial %d - Model 1 Residuals', i);
114         calcStatistics(residual_vec_model_1{i}, dataType);
115         dataType = sprintf('Balanced Wheel Trial %d - Model 2 Residuals', i);
116         calcStatistics(residual_vec_model_2{i}, dataType);
117
118         % Packaging Data for plotting
119         theta_exp_for_plot{i} = theta_exp.balanced{i};
120         omega_exp_for_plot{i} = omega_exp.balanced{i};
121         theta_mod_for_plot{i} = theta_mod.balanced{i};
122
123     end
124
125
126     %%% Plot Results for Balanced Wheel analysis, Model 1 and 2
127     testString = sprintf('Balanced Wheel - Models 1 and 2');
128     modelsTested = {'1', '2'};
129     outputData(theta_exp_for_plot, theta_mod_for_plot, ...
130                omega_exp_for_plot, ...
131                omega_mod_1, omega_mod_2, ...
132                residual_vec_model_1, residual_vec_model_2, ...
133                testString, modelsTested, shouldSaveFigures);
134     fprintf('\n\n\n')
135
136     %% Calculate the Model 3 and 4 Omega and their Residuals
137     for i = 1:length(theta_exp.unbalanced)
138
139         modelUsed = 3;
140         omega_mod_res_3{i} = calcModelOmega(theta_exp.unbalanced{i}, M, ...
141                                             constants, modelUsed);
142         omega_mod_3{i} = calcModelOmega(theta_mod.unbalanced{i}, M, ...
143                                             constants, modelUsed);
144
145         modelUsed = 4;
146         omega_mod_res_4{i} = calcModelOmega(theta_exp.unbalanced{i}, M, ...
147                                             constants, modelUsed);
148         omega_mod_4{i} = calcModelOmega(theta_mod.unbalanced{i}, M, ...
149                                             constants, modelUsed);
150
151         % Calculate Residuals for both models
152         residual_vec_model_3{i} = ( omega_exp.unbalanced{i} - ...
153                                     omega_mod_res_3{i} )';
154         residual_vec_model_4{i} = ( omega_exp.unbalanced{i} - ...
155                                     omega_mod_res_4{i} )';
156
157
158         % Calculate Statistics for Models 3 and 4
159         dataType = sprintf('Unbalanced Wheel Trial %d - Model 3 Residuals', i)
                ;
160         calcStatistics(residual_vec_model_3{i}, dataType);
161         dataType = sprintf('Unbalanced Wheel Trial %d - Model 4 Residuals', i)
                ;
162         calcStatistics(residual_vec_model_4{i}, dataType);
163
```

```
164         % Packaging Data for plotting
165         theta_exp_for_plot{i} = theta_exp.unbalanced{i};
166         omega_exp_for_plot{i} = omega_exp.unbalanced{i};
167         theta_mod_for_plot{i} = theta_mod.unbalanced{i};
168
169     end
170
171
172     %%% Plot Results for Balanced Wheel analysis, Model 3 and 4
173     testString = sprintf('Unbalanced Wheel - Models 3 and 4');
174     modelsTested = {'3', '4'};
175     outputData(theta_exp_for_plot, theta_mod_for_plot, ...
176                omega_exp_for_plot, ...
177                omega_mod_3, omega_mod_4, ...
178                residual_vec_model_3, residual_vec_model_4, ...
179                testString, modelsTested, shouldSaveFigures);
180
181
182
183     %% Plot all 4 Models Togther
184
185     xdata = {theta_mod.balanced{1}, theta_mod.balanced{1}, ...
186              theta_mod.unbalanced{1}, theta_mod.unbalanced{1}};
187     ydata = {omega_mod_1{1}, omega_mod_2{1}, omega_mod_3{1}, ...
188              omega_mod_4{1}};
189
190     plotAllModels(xdata, ydata, shouldSaveFigures)
191
192
193     %% Plot Angular Acceleration
194
195     % Plot Model
196     type_str = 'Model';
197     xdata = {theta_mod.balanced{1}, theta_mod.balanced{1}, ...
198              theta_mod.unbalanced{1}, theta_mod.unbalanced{1}};
199     ydata = {omega_mod_1{1}, omega_mod_2{1}, omega_mod_3{1}, ...
200              omega_mod_4{1}};
201     plotAllAngAccel(xdata, ydata, type_str, shouldSaveFigures)
202
203
204     % Plot Experimental
205     type_str = 'Experimental';
206     xdata =  {theta_exp.balanced{1}, theta_exp.balanced{2}, ...
207               theta_exp.unbalanced{1}, theta_exp.unbalanced{2}};
208     ydata = {omega_exp.balanced{1}, omega_exp.balanced{2}, ...
209              omega_exp.unbalanced{1}, omega_exp.unbalanced{2}};
210     plotAllAngAccel(xdata, ydata, type_str, shouldSaveFigures)
211
212
213 end
```

monteCarloCalcM.m

```
1 function newM = monteCarloCalcM(M_lims, ...
```

```
                                    omega_exp, ...
                                    theta_exp, ...
                                    constants, ...
                                    modelUsed)

%% ASEN 2003: Dynamics & Systems - Spring 2017
% Project: Rolling Wheel Lab (#4)
% Project Members:  Joseph Grengs
%                   Kian Tanner
%                   Nicholas Renniger
%
%
% Function takes an interval of M value over which to search, along
% with the experimental data, and runs a short Monte Carlo simulation
% to randomly generate M values over the specified M interval, and
% build up a vector of least-squares differences. These differences are
% computed by trying a random M over the interval, calculating the
% model omega with this M, and summing the squared difference between
% the model and experimental data for the random M. This is done
% thousands of times, and at the end, the minimum sum of squared
% difference is chosen as the optimal solution, and the M value
% corresponding to this minimum squared difference is returned as the
% emprical constant for this lab.
%
% Project Due Date: Thursday, March 16, 2017 @ 4:00p
% MATLAB Code Created on: 03/02/2017
% MATLAB Code last updated on: 03/15/2017

%% Setup
MIN_M = M_lims(1);
MAX_M = M_lims(2);
NUM_STEPS = 1e3;
new_M = MIN_M;

% initialize
[leastSquaresDiff, M_vec] = deal(zeros(1, NUM_STEPS));

%% Find minimizing M

% calculate least squares difference NUM_STEPS times using randomly
% generated M value.
for i = 1:NUM_STEPS

    curr_M = new_M;
    omega_model = calcModelOmega(theta_exp, curr_M, ...
                                 constants, modelUsed);

    leastSquaresDiff(i) = sum((omega_exp - omega_model).^2);
    M_vec(i) = curr_M;
    new_M = rand * MAX_M - MIN_M;

end

% find minimum least squares difference
[~, minIdx] = min(leastSquaresDiff);
```

```
57
58      % return M value that minimizes the difference
59      newM = M_vec(minIdx);
60
61
62  end
```

calcModelOmega.m

```
1   %% ASEN 2003: Dynamics & Systems - Spring 2017
2   % Project: Rolling Wheel Lab (#4)
3   % Project Members:  Joseph Grengs
4   %                   Kian Tanner
5   %                   Nicholas Renniger
6   %
7   % Function calculates the model omega based on the given values of theta,
8   % and the chosen model to use to calculate the new omega.
9   %
10  % Project Due Date: Thursday, March 16, 2017 @ 4:00p
11  % MATLAB Code Created on: 03/02/2017
12  % MATLAB Code last updated on: 03/14/2017
13
14  function [ omega ] = calcModelOmega( theta, M, constants, modelUsed )
15
16      %% Function purpose:
17
18
19      Mc = constants(1);              % Mass of cylinder (kg)
20      Mta = constants(2);             % Mass of trailing apparatus (kg)
21      Mem = constants(3);             % Mass of extra mass (kg)
22      Rc = constants(4);              % Radius of cylinder (m)
23      Ic = constants(6);              % MOI of cylinder (kg*m^2)
24      Beta = constants(7);            % Angle of ramp (radians)
25      Rtem = constants(8);            % Radius to extra mass (m)
26      Rem = constants(9);             % Radius of extra mass (m)
27      g = constants(10);              % Acceleration due to gravity (m/s^2)
28
29      switch modelUsed
30
31          % Case 1: Balanced Wheel Analysis (no fudge factor):
32          case 1
33
34              Num1 = 2 * (Mc + Mta) * g * Rc .* theta .* sin(Beta);
35              Den1 = (Mc + Mta) * Rc^2 + Ic;
36              omega(:, 1) = sqrt(Num1 ./ Den1);
37
38
39          % Case 2: Balanced Wheel Analysis (fudge factor added):
40          case 2
41
42              Num2 = 2 .* theta .* (-M + (Mc + Mta) .* g .* Rc .* sin(Beta));
43              Den2 = (Mc + Mta) * Rc^2 + Ic;
44              omega(:, 1) = sqrt(Num2 ./ Den2);
45
```

```
46
47          % Case 3: Unbalanced Wheel Analysis with extra cylinder modeled as
48          % point mass for MOI calcs.
49          case 3
50
51              Num1 = (Mc + Mta) * g * Rc .* theta .* sin(Beta) - ...
52              Mem .* g .* Rtem .* (cos(theta + Beta) - cos(Beta)) + ...
53              Mem .* g .* Rc .* theta.* sin(Beta) - M .* theta;
54
55              Den1 = 0.5 .* ((Mc + Mta) .* Rc^2 + Mem .*...
56                     (Rc^2 + 2 .* Rc .* Rtem .* cos(theta) + Rtem^2) + Ic);
57
58              omega(:, 1) = sqrt(Num1 ./ Den1);
59
60
61          % Case 4: Unbalanced Wheel Analysis with extra cylinder modeled as
62          % a cylinder for MOI calcs.
63          case 4
64
65              Num1 = (Mc + Mta) * g * Rc .* theta .* sin(Beta) - ...
66              Mem .* g .* Rtem .* (cos(theta + Beta) - cos(Beta)) + ...
67              Mem .* g .* Rc .* theta.* sin(Beta) - M .* theta;
68
69              Den1 = 0.5 .* ((Mc + Mta) .* Rc^2 + Mem .*...
70                     (Rc^2 + 2 .* Rc .* Rtem .* cos(theta) + Rtem^2) + ...
71                      + Ic + (1/4 * Mem * Rem^2) );
72
73              omega(:, 1) = sqrt(Num1 ./ Den1);
74      end
75
76  end
```

calcStatistics.m

```
1   function calcStatistics(residuals, dataType)
2
3       %%% calcStatistics(residuals, testType)
4       %%%
5       %%% Calculates the statistics of the calculated residuals, and prints
6       %%% the results in a table to the command window and to a file in the
7       %%% following directory:
8       %%%                       ../Data/Data Statistics/
9       %%%
10      %%% Inputs:
11      %%%         - residuals: the vector of differences between experimental
12      %%%                      results and the values predicted by the model.
13      %%%
14      %%%         - dataType: string indicating which test was being run.
15      %%%
16      %%% Outputs:
17      %%%          - writes statistics to command window and to an
18      %%%           appropriately named file.
19      %%%
20      %%% Author: Nicholas Renninger
```

```
%%% Date Created: 3/7/17
%%% Last Modified: 3/14/17


%% Calc Standard Deviation of the Residuals for Collar Velocity14
sigma = std(residuals);


%% Mean of the Residuals
meanResidual = mean(residuals);


%% Uncertainty of the Mean Residual
numResiduals = length(residuals);
sigmaMean = sigma / sqrt(numResiduals);


%% Number of Observations
numObservations = numResiduals;


%% Number of Residuals that are greater than 3*simga
num3SigmaResiduals = length(residuals(residuals > 3*sigma));


%% Print to File

% Creating File and Labeling what data is being Analyzed
testName = sprintf('%s - Statistics', dataType);
filename = sprintf('../Data/Data Statistics/%s.txt', testName);
fid = fopen(filename, 'w+');

%%%%%%%%%%%%%%% print to file %%%%%%%%%%%%%%%

numBars = 82;

% header
fprintf(fid, '%s:\r\n\r\n', testName);
fprintf(fid, ['|  sigma  |  mean residual  | sigma/sqrt(N) |', ...
              '    N     | num. residuals > 3*sigma  |\r\n|']);

% print horiz bars
for i = 1:numBars
    fprintf(fid, '-');
end

% rest of data
fprintf(fid,['|\r\n|%08.1g |    %09.3g    |    %08.1g    |   %04d   ', ...
         '|          %03d             |\r\n'], sigma,...
         meanResidual, sigmaMean, numObservations, num3SigmaResiduals);

fclose(fid);


%%%%%%%%%%%%%%% print to command window %%%%%%%%%%%%%%%
```

```
76      numBars = 82;
77
78      % header
79      fprintf('\n%s:\n', testName);
80      fprintf(['|  sigma  |  mean residual  | sigma/sqrt(N) |', ...
81               '    N     | num. residuals > 3*sigma  |\n|']);
82
83      % print horiz bars
84      for i = 1:numBars
85          fprintf('-');
86      end
87
88      % rest of data
89      fprintf(['|\n|%08.1g |    %09.3g    |    %08.1g    |   %04d    ', ...
90               '|           %03d             |\n'], sigma,...
91              meanResidual, sigmaMean, numObservations, num3SigmaResiduals);
92
93
94  end
```

outputData.m

```
1   function outputData(theta_exp, theta_mod, omega_exp, ...
2                       omega_model_a, omega_model_b, ...
3                       residuals_a, residuals_b, ...
4                       testString, modelsTested, shouldSaveFigures)
5
6
7       %% ASEN 2003: Dynamics & Systems - Spring 2017
8       % Project: Rolling Wheel Lab (#4)
9       % Project Members:  Joseph Grengs
10      %                   Kian Tanner
11      %                   Nicholas Renniger
12      %
13      %
14      % Function takes all of the calculated and measured values for theta
15      % and omega and plots the experimental and model data, along with a
16      % time history and histogram of the residuals.
17      %
18      % Project Due Date: Thursday, March 16, 2017 @ 4:00p
19      % MATLAB Code Created on: 03/02/2017
20      % MATLAB Code last updated on: 03/15/2017
21
22      %% Plot Setup
23
24      set(0, 'defaulttextinterpreter', 'latex');
25      titleString = sprintf('%s', testString);
26      saveLocation = '../Figures/';
27      saveTitle = cat(2, saveLocation, sprintf('%s', titleString));
28      LINEWIDTH = 2;
29      MARKERSIZE = 6;
30      FONTSIZE = 20;
31      SCALE_FACTOR = 1;
32      colorVecs = [0.294118 0 0.509804; % indigo
```

```
                0.180392 0.545098 0.341176; % sea green
                1 0.270588 0; % orange red
                0 0.74902 1; % deep sky blue
                0.858824 0.439216 0.576471; % forestgreen
                0.133333 0.545098 0.133333; % palevioletred
                0.803922 0.521569 0.247059; % peru
                1 0.498039 0.313725]; % coral

markers = {'+','o','*','.','x','s','d','^','v','>','<','p','h'};

hFig = figure('name', titleString);
scrz = get(groot, 'ScreenSize');
set(hFig, 'Position', scrz)


%% Subplot (1): Model and Experimental Omega vs. Theta
subplot(2, 2, 1)

xmin = 0;
xmax = 15;

y_info = omega_model_a{1};
ymin = min(y_info) * 0.2;
ymax = max(y_info) * 1.05;

hold on
grid on

% plot the experimental data
for i = 1:length(theta_exp)

    p_vec(i) = plot(theta_exp{i}, omega_exp{i}, markers{i}, ...
                'linewidth', LINEWIDTH, 'markersize', MARKERSIZE, ...
                'Color', colorVecs(i + 2, :));
    exp_data_leg_string{i} = sprintf('Trial %d Experimental Data', ...
                                     i);
end

% plot the model "a" data
p1 = plot(theta_mod{1}, omega_model_a{1}, ':', ...
     'linewidth', LINEWIDTH, 'Color', colorVecs(1, :));
p1_leg_string = sprintf('Model %s', modelsTested{1});

% plot the model "b" data
p2 = plot(theta_mod{1}, omega_model_b{1}, '-', ...
     'linewidth', LINEWIDTH, 'Color', colorVecs(2, :));
p2_leg_string = sprintf('Model %s', modelsTested{2});

plot_handles = [p1, p2, p_vec];
leg_string = [p1_leg_string, p2_leg_string, exp_data_leg_string];

xlim([xmin, xmax])
ylim([ymin, ymax])
xlabel('$\theta$ (rad)')
ylabel('$\omega$ (rad/s)')
```

```
    leg = legend(plot_handles, leg_string, ...
                 'location', 'best', 'interpreter', 'latex');
    set(leg, 'FontSize', round(FONTSIZE * 0.7))
    set(gca,'FontSize', FONTSIZE)
    title(sprintf('%s - $\\omega$ vs. $\\theta$', titleString), ...
          'fontsize', round(FONTSIZE * SCALE_FACTOR))
    set(gca, 'defaulttextinterpreter', 'latex')
    set(gca, 'TickLabelInterpreter', 'latex')



    %% Subplot (2): Residuals vs. Theta
    subplot(2, 2, 2)

    xmin = 0;
    xmax = 15;

    ymin = min(residuals_a{1}) - 0.1;
    ymax = max(residuals_b{1}) + 0.1;

    hold on
    grid on

    % plot the residual data against theta
    for i = 1:length(theta_exp)

        % model "a"
        resid_plot_a(i) = plot(theta_exp{i}, residuals_a{i}, ...
              [markers{i + 2}, ':'], ...
             'linewidth', LINEWIDTH, 'markersize', MARKERSIZE, ...
             'Color', colorVecs(1, :));
        res_a_leg_string{i} = sprintf('Model %s: Trial %d', ...
                                      modelsTested{1}, i);

        % model "b"
        resid_plot_b(i) = plot(theta_exp{i}, residuals_b{i}, ...
             [markers{i + length(theta_exp) + 2}, '-'], ...
             'linewidth', LINEWIDTH, 'markersize', MARKERSIZE, ...
             'Color', colorVecs(2, :));
        res_b_leg_string{i} = sprintf('Model %s: Trial %d', ...
                                      modelsTested{2}, i);

    end

    leg_string = cat(2, res_a_leg_string, res_b_leg_string);
    plot_handles = cat(2, resid_plot_a, resid_plot_b);

    xlim([xmin, xmax])
    ylim([ymin, ymax])
    xlabel('$\theta$ (rad)')
    ylabel('Residual (rad/s)')
    leg = legend(plot_handles, leg_string, ...
                 'location', 'best', 'interpreter', 'latex');
    set(gca, 'FontSize', FONTSIZE)
    title(sprintf('%s - Residuals', testString), 'fontsize', ...
```

```
            round(FONTSIZE * SCALE_FACTOR))
    set(leg, 'FontSize', round(FONTSIZE * 0.7))
    set(gca, 'defaulttextinterpreter', 'latex')
    set(gca, 'TickLabelInterpreter', 'latex')



    %% Subplot (3): Histogram of Model "a" Residuals

    subplot(2,2,3)

    histfit(residuals_a{1}, 20)
    title_string = sprintf('Histogram of Residual - Model %s, Trial 1', ...
                           modelsTested{1});
    xlabel('Residual (rad/s)')
    ylabel('\# Samples w/ Residual')
    set(gca,'FontSize', round(FONTSIZE * SCALE_FACTOR))
    title(title_string, 'fontsize', round(FONTSIZE * SCALE_FACTOR))
    set(gca, 'defaulttextinterpreter', 'latex')
    set(gca, 'TickLabelInterpreter', 'latex')



    %% Subplot (4): Histogram of Model "b" Residuals

    subplot(2,2,4)

    histfit(residuals_b{1}, 20)
    title_string = sprintf('Histogram of Residual - Model %s, Trial 1', ...
                           modelsTested{2});
    xlabel('Residual (rad/s)')
    ylabel('\# Samples w/ Residual')
    set(gca,'FontSize', round(FONTSIZE * SCALE_FACTOR))
    title(title_string, 'fontsize', round(FONTSIZE * SCALE_FACTOR))
    set(gca, 'defaulttextinterpreter', 'latex')
    set(gca, 'TickLabelInterpreter', 'latex')



    %% setup and save figure as .pdf
    if shouldSaveFigures
        savefig(saveTitle, 'pdf', '-r500');
    end

end
```

plotAllModels.m

```
function plotAllModels(xdata, ydata, shouldSaveFigures)

    %% ASEN 2003: Dynamics & Systems - Spring 2017
    % Project: Rolling Wheel Lab (#4)
    % Project Members:  Joseph Grengs
```

```
%                   Kian Tanner
%                   Nicholas Renniger
%
%
% Function takes all of the model omega data and plots it all together
% on one plot to allow for easy comparison between the four models.
%
% Project Due Date: Thursday, March 16, 2017 @ 4:00p
% MATLAB Code Created on: 03/02/2017
% MATLAB Code last updated on: 03/15/2017


%% Plot Setup
hFig = figure('name', 'Model Comparisons');
scrz = get(groot, 'ScreenSize');
set(hFig, 'Position', scrz)

xlabel_str = '$\theta$ (rad)';
ylabel_str = '$\omega$ (rad/s)';
title_str = 'Comparison of Models';
legend_str = {'Model 1', 'Model 2', 'Model 3', 'Model 4'};

colorVecs =     {[0.156863 0.156863 0.156863], ... % sgivery dark grey
                 [0.858824 0.439216 0.576471], ... % palevioletred
                 [0.254902 0.411765 0.882353], ... % royal blue
                 [0.854902 0.647059 0.12549]}; % golden rod

linetypes = {'--', '-', ':', '-.'};
LINEWIDTH = 2;
FONTSIZE = 26;
MARKERSIZE = 5;


xmin = 0;
xmax = 15;

ymin = 0;
ymax = 9;

hold on
grid on

%% plot each model
for i = 1:length(xdata)

    % make line thicker for model 3 to differentiate it from model 4
    if i == 3
        LINEWIDTH = LINEWIDTH + 1;
        p_vec(i) = plot(xdata{i}, ydata{i}, linetypes{i}, ...
                'linewidth', LINEWIDTH, 'markersize', MARKERSIZE, ...
                'Color', colorVecs{i});
        LINEWIDTH = LINEWIDTH - 1;
    else
        p_vec(i) = plot(xdata{i}, ydata{i}, linetypes{i}, ...
                'linewidth', LINEWIDTH, 'markersize', MARKERSIZE, ...
```

```
                    'Color', colorVecs{i});
        end
    end


    xlim([xmin, xmax])
    ylim([ymin, ymax])
    xlabel(xlabel_str)
    ylabel(ylabel_str)
    leg = legend(p_vec, legend_str, ...
                 'location', 'best', 'interpreter', 'latex');
    set(leg, 'FontSize', round(FONTSIZE * 0.8))
    set(gca,'FontSize', FONTSIZE)
    title(title_str, 'fontsize', round(FONTSIZE))
    set(gca, 'defaulttextinterpreter', 'latex')
    set(gca, 'TickLabelInterpreter', 'latex')

    %% setup and save figure as .pdf

    saveTitle = '../Figures/All Models';

    if shouldSaveFigures
        savefig(saveTitle, 'pdf', '-r500');
    end

end
```

plotAllAngAccel.m

```
function plotAllAngAccel(xdata, ydata, type_str, shouldSaveFigures)

    %% ASEN 2003: Dynamics & Systems - Spring 2017
    % Project: Rolling Wheel Lab (#4)
    % Project Members:  Joseph Grengs
    %                   Kian Tanner
    %                   Nicholas Renniger
    %
    %
    % Function takes all of the model omega data, calculates angular
    % acceleration, and plots it all together on one plot to allow for easy
    % comparison between the four models.
    %
    % Project Due Date: Thursday, March 16, 2017 @ 4:00p
    % MATLAB Code Created on: 03/02/2017
    % MATLAB Code last updated on: 03/15/2017


    %% Differentiate Omega to find Alpha

    for i = 1:length(ydata)

        angAccelData{i} = diff(ydata{i});
        xdata{i} = xdata{i}(1:end - 1);

```

```
26      end
27
28      ydata = angAccelData;
29
30      %% Plot Setup
31      hFig = figure('name', 'Model Comparisons');
32      scrz = get(groot, 'ScreenSize');
33      set(hFig, 'Position', scrz)
34
35      xlabel_str = '$\theta$ (rad)';
36      ylabel_str = '$\alpha$ (rad/$s^2$)';
37      title_str = sprintf('%s - Angular Acceleration', type_str);
38      if strfind(type_str, 'Model')
39          legend_str = {'Model 1', 'Model 2', 'Model 3', 'Model 4'};
40      else
41          legend_str = {'Trial 1 - Balanced', 'Trial 2 - Balanced',...
42                        'Trial 1 - Unbalanced', 'Trial 2 - Unbalanced'};
43      end
44
45
46      colorVecs =     {[0.156863 0.156863 0.156863], ... % sgivery dark grey
47                       [0.858824 0.439216 0.576471], ... % palevioletred
48                       [0.254902 0.411765 0.882353], ... % royal blue
49                       [0.854902 0.647059 0.12549]}; % golden rod
50
51
52      linetypes = {'--', '-', ':', '-.'};
53      LINEWIDTH = 2;
54      FONTSIZE = 26;
55      MARKERSIZE = 5;
56
57
58      xmin = min(xdata{1});
59      xmax = max(xdata{1});
60
61      ymin = min(ydata{3});
62      ymax = max(ydata{3});
63
64      hold on
65      grid on
66
67      %% Plot each model
68
69      for i = 1:length(xdata)
70
71          % make line thicker for model 3 to differentiate it from model 4
72          if i == 3
73              LINEWIDTH = LINEWIDTH + 1;
74              p_vec(i) = plot(xdata{i}, ydata{i}, linetypes{i}, ...
75                      'linewidth', LINEWIDTH, 'markersize', MARKERSIZE, ...
76                      'Color', colorVecs{i});
77              LINEWIDTH = LINEWIDTH - 1;
78          else
79              p_vec(i) = plot(xdata{i}, ydata{i}, linetypes{i}, ...
80                      'linewidth', LINEWIDTH, 'markersize', MARKERSIZE, ...
```

```
                    'Color', colorVecs{i});
        end
    end


    xlim([xmin, xmax])
    ylim([ymin, ymax])
    xlabel(xlabel_str)
    ylabel(ylabel_str)
    leg = legend(p_vec, legend_str, ...
                 'location', 'best', 'interpreter', 'latex');
    set(leg, 'FontSize', round(FONTSIZE * 0.8))
    set(gca,'FontSize', FONTSIZE)
    title(title_str, 'fontsize', round(FONTSIZE))
    set(gca, 'defaulttextinterpreter', 'latex')
    set(gca, 'TickLabelInterpreter', 'latex')

    %% setup and save figure as .pdf

    saveTitle = sprintf('../Figures/%s - Angular Acceleration', type_str);

    if shouldSaveFigures
        savefig(saveTitle, 'pdf', '-r500');
    end

end
```

savefig.m

```
function savefig(fname, varargin)

% Usage: savefig(filename, fighdl, options)
%
% Saves a pdf, eps, png, jpeg, and/or tiff of the contents of the fighandle's
    (or current) figure.
% It saves an eps of the figure and the uses Ghostscript to convert to the
    other formats.
% The result is a cropped, clean picture. There are options for using rgb or
    cmyk colours,
% or grayscale. You can also choose the resolution.
%
% The advantage of savefig is that there is very little empty space around the
     figure in the
% resulting files, you can export to more than one format at once, and
    Ghostscript generates
% trouble-free files.
%
% If you find any errors, please let me know! (peder at axensten dot se)
%
% filename: File name without suffix.
%
% fighdl:  (default: gcf) Integer handle to figure.
%
% options: (default: '-r300', '-lossless', '-rgb') You can define your own
```

```
21 %          defaults in a global variable savefig_defaults, if you want to, i.e.
22 %          savefig_defaults={'-r200','-gray'};.
23 % 'eps':   Output in Encapsulated Post Script (no preview yet).
24 % 'pdf':   Output in (Adobe) Portable Document Format.
25 % 'png':   Output in Portable Network Graphics.
26 % 'jpeg':  Output in Joint Photographic Experts Group format.
27 % 'tiff':  Output in Tagged Image File Format (no compression: huge files!).
28 % '-rgb':  Output in rgb colours.
29 % '-cmyk': Output in cmyk colours (not yet 'png' or 'jpeg' -- '-rgb' is used).
30 % '-gray': Output in grayscale (not yet 'eps' -- '-rgb' is used).
31 % '-fonts':  Include fonts in eps or pdf. Includes only the subset needed.
32 % '-lossless':  Use lossless compression, works on most formats. same as '-c0', below.
33 % '-c<float>':  Set compression for non-indexed bitmaps in PDFs -
34 %               0: lossless; 0.1: high quality; 0.5: medium; 1: high compression.
35 % '-r<integer>':  Set resolution.
36 % '-crop': Removes points and line segments outside the viewing area -- permanently.
37 %          Only use this on figures where many points and/or line segments are outside
38 %          the area zoomed in to. This option will result in smaller vector files (has no
39 %          effect on pixel files).
40 % '-dbg':  Displays gs command line(s).
41 %
42 % EXAMPLE:
43 % savefig('nicefig', 'pdf', 'jpeg', '-cmyk', '-c0.1', '-r250');
44 % Saves the current figure to nicefig.pdf and nicefig.png, both in cmyk and at 250 dpi,
45 %          with high quality lossy compression.
46 %
47 % REQUIREMENT: Ghostscript. Version 8.57 works, probably older versions too, but '-dEPSCrop'
48 %          must be supported. I think version 7.32 or newer is ok.
49 %
50 % HISTORY:
51 % Version 1.0, 2006-04-20.
52 % Version 1.1, 2006-04-27:
53 % - No 'epstopdf' stuff anymore! Using '-dEPSCrop' option in gs instead!
54 % Version 1.2, 2006-05-02:
55 % - Added a '-dbg' option (see options, above).
56 % - Now looks for a global variable 'savefig_defaults' (see options, above).
57 % - More detailed Ghostscript options (user will not really notice).
58 % - Warns when there is no device for a file-type/color-model combination.
59 % Version 1.3, 2006-06-06:
60 % - Added a check to see if there actually is a figure handle.
61 % - Now works in Matlab 6.5.1 (R13SP1) (maybe in 6.5 too).
62 % - Now compatible with Ghostscript 8.54, released 2006-06-01.
63 % Version 1.4, 2006-07-20:
64 % - Added an option '-soft' that enables anti-aliasing on pixel graphics (on by default).
65 % - Added an option '-hard' that don't do anti-aliasing on pixel graphics.
66 % Version 1.5, 2006-07-27:
```

```
67 % - Fixed a bug when calling with a figure handle argument.
68 % Version 1.6, 2006-07-28:
69 % - Added a crop option, see above.
70 % Version 1.7, 2007-03-31:
71 % - Fixed bug: calling print with invalid renderer value '-none'.
72 % - Removed GhostScript argument '-dUseCIEColor' as it sometimes discoloured
       things.
73 % Version 1.8, 2008-01-03:
74 % - Added MacIntel: 'MACI'.
75 % - Added 64bit PC (I think, can't test it myself).
76 % - Added option '-nointerpolate' (use it to prevent blurring of pixelated).
77 % - Removed '-hard' and '-soft'. Use '-nointerpolate' for '-hard', default for
       '-soft'.
78 % - Fixed the gs 8.57 warning on UseCIEColor (it's now set).
79 % - Added '-gray' for pdf, but gs 8.56 or newer is needed.
80 % - Added '-gray' and '-cmyk' for eps, but you a fairly recent gs might be
       needed.
81 % Version 1.9, 2008-07-27:
82 % - Added lossless compression, see option '-lossless', above. Works on most
       formats.
83 % - Added lossy compression, see options '-c<float>...', above. Works on 'pdf
       '.
84 %   Thanks to Olly Woodford for idea and implementation!
85 % - Removed option '-nointerpolate' -- now savefig never interpolates.
86 % - Fixed a few small bugs and removed some mlint comments.
87 % Version 2.0, 2008-11-07:
88 % - Added the possibility to include fonts into eps or pdf.
89 %
90 % TO DO: (Need Ghostscript support for these, so don't expect anything soon
       ...)
91 % - svg output.
92 % - '-cmyk' for 'jpeg' and 'png'.
93 % - Preview in 'eps'.
94 % - Embedded vector fonts, not bitmap, in 'eps'.
95 %
96 % Copyright (C) Peder Axensten (peder at axensten dot se), 2006.
97
98 % KEYWORDS:      eps, pdf, jpg, jpeg, png, tiff, eps2pdf, epstopdf, ghostscript
99 %
100 % INSPIRATION:  eps2pdf (5782), eps2xxx (6858)
101 %
102 % REQUIREMENTS: Works in Matlab 6.5.1 (R13SP1) (maybe in 6.5 too).
103
104 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
105
106         op_dbg=         false;
                                                          % Default value.
107
108         % Compression
109         compr=                  [' -dUseFlateCompression=true -dLZWEncodePages
              =true -dCompatibilityLevel=1.6' ...
110                                  ' -dAutoFilterColorImages=false -
                                       dAutoFilterGrayImages=false ' ...
111                                  ' -dColorImageFilter=%s -
                                       dGrayImageFilter=%s'];        %
```

```
                                                Compression.
112         lossless=               sprintf (compr, '/FlateEncode', '/FlateEncode'
                );
113         lossy=                  sprintf (compr, '/DCTEncode',   '/DCTEncode'
                );
114         lossy=                  [lossy ' -c ".setpdfwrite << /ColorImageDict
               << /QFactor %g ' ...
115                                         '/Blend 1 /HSample [%s] /VSample [%s]
                                              >> >> setdistillerparams" '];
116
117         % Create gs command.
118         cmdEnd=                 ' -sDEVICE=%s -sOutputFile="%s" ';
                                          % Essential.
119         epsCmd=                 '';
120         epsCmd= [epsCmd ' -dSubsetFonts=true -dNOPLATFONTS'];
                           % Future support?
121         epsCmd= [epsCmd ' -dColorConversionStrategy=/UseDeviceIndependentColor
                '];
122         epsCmd= [epsCmd ' -dProcessColorModel=/%s'];
                                  % Color conversion.
123         pdfCmd= [epsCmd ' -dAntiAliasColorImages=false' cmdEnd];
124         epsCmd= [epsCmd cmdEnd];
125
126         % Get file name.
127         if((nargin < 1) || isempty(fname) || ~ischar(fname))
                           % Check file name.
128                 error('No file name specified.');
129         end
130         [pathstr, namestr] = fileparts(fname);
131         if(isempty(pathstr)), fname= fullfile(cd, namestr);      end
132
133         % Get handle.
134         fighdl=         get(0, 'CurrentFigure'); % See gcf.
                                          % Get figure handle.
135         if((nargin >= 2) && (numel(varargin{1}) == 1) && isnumeric(varargin
               {1}))
136                 fighdl=         varargin{1};
137                 varargin=       {varargin{2:end}};
138         end
139         if(isempty(fighdl)), error('There is no figure to save!?');      end
140         set(fighdl, 'Units', 'centimeters')
                                                  % Set paper stuff.
141         sz=                     get(fighdl, 'Position');
142         sz(1:2)=        0;
143         set(fighdl, 'PaperUnits', 'centimeters', 'PaperSize', sz(3:4), '
               PaperPosition', sz);
144
145         % Set up the various devices.
146         % Those commented out are not yet supported by gs (nor by savefig).
147         % pdf-cmyk works due to the Matlab '-cmyk' export being carried over
               from eps to pdf.
148         device.eps.rgb=         sprintf(epsCmd, 'DeviceRGB',    'epswrite', [
               fname '.eps']);
149         device.jpeg.rgb=        sprintf(cmdEnd, 'jpeg',
                                      [fname '.jpeg']);
```

```
150 %       device.jpeg.cmyk=       sprintf(cmdEnd, 'jpegcmyk',
                [fname '.jpeg']);
151         device.jpeg.gray=       sprintf(cmdEnd, 'jpeggray',
                [fname '.jpeg']);
152         device.pdf.rgb=         sprintf(pdfCmd, 'DeviceRGB',    'pdfwrite', [
            fname '.pdf']);
153         device.pdf.cmyk=        sprintf(pdfCmd, 'DeviceCMYK',   'pdfwrite', [
            fname '.pdf']);
154         device.pdf.gray=        sprintf(pdfCmd, 'DeviceGray',   'pdfwrite', [
            fname '.pdf']);
155         device.png.rgb=         sprintf(cmdEnd, 'png16m',
                [fname '.png']);
156 %       device.png.cmyk=        sprintf(cmdEnd, 'png???',
                [fname '.png']);
157         device.png.gray=        sprintf(cmdEnd, 'pnggray',
                [fname '.png']);
158         device.tiff.rgb=        sprintf(cmdEnd, 'tiff24nc',
                [fname '.tiff']);
159         device.tiff.cmyk=       sprintf(cmdEnd, 'tiff32nc',
                [fname '.tiff']);
160         device.tiff.gray=       sprintf(cmdEnd, 'tiffgray',
                [fname '.tiff']);
161
162         % Get options.
163         global savefig_defaults;
                                                    % Add global
            defaults.
164         if( iscellstr(savefig_defaults)), varargin=     {savefig_defaults{:},
            varargin{:}};
165         elseif(ischar(savefig_defaults)), varargin=     {savefig_defaults,
            varargin{:}};
166         end
167         varargin=       {'-r300', '-lossless', '-rgb', varargin{:}};
                            % Add defaults.
168         res=            '';
169         types=          {};
170         fonts=          'false';
171         crop=           false;
172         for n= 1:length(varargin)
                                                    % Read options.
173                 if(ischar(varargin{n}))
174                         switch(lower(varargin{n}))
175                         case {'eps','jpeg','pdf','png','tiff'},         types{
                            end+1}=   lower(varargin{n});
176                         case '-rgb',                            color=  'rgb';
                                deps= {'-depsc2'};
177                         case '-cmyk',                           color=  'cmyk'
                            ; deps= {'-depsc2', '-cmyk'};
178                         case '-gray',                           color=  'gray'
                            ; deps= {'-deps2'};
179                         case '-fonts',                          fonts=
                                                'true';
180                         case '-lossless',                       comp=
                                                0;
181                         case '-crop',                           crop=
```

```
                                        true;
182                     case '-dbg',                                    op_dbg=
                                        true;
183                     otherwise
184                             if(regexp(varargin{n}, '^\-r[0-9]+$')),
                                    res=  varargin{n};
185                             elseif(regexp(varargin{n}, '^\-c[0-9.]+$')),
                                comp= str2double(varargin{n}(3:end));
186                             else    warning('pax:savefig:inputError', '
                                Unknown option in argument: ''%s''.',
                                varargin{n});
187                             end
188                     end
189             else
190                     warning('pax:savefig:inputError', 'Wrong type of
                            argument: ''%s''.', class(varargin{n}));
191             end
192     end
193     types=          unique(types);
194     if(isempty(types)), error('No output format given.');   end
195
196     if (comp == 0)                                                  % Lossless
         compression
197             gsCompr=                lossless;
198     elseif (comp <= 0.1)                                            % High quality
        lossy
199             gsCompr=                sprintf(lossy, comp, '1 1 1 1', '1 1 1
                    1');
200     else                                                            %
        Normal lossy
201             gsCompr=                sprintf(lossy, comp, '2 1 1 2', '2 1 1
                    2');
202     end
203
204     % Generate the gs command.
205     switch(computer)                                                % Get gs
        command.
206             case {'MAC','MACI'},                    gs= '/usr/local/bin/gs
                    ';
207             case {'PCWIN','PCWIN64'},               gs= 'gswin32c.exe';
208             otherwise,                                              gs= '
                    gs';
209     end
210     gs=             [gs             ' -q -dNOPAUSE -dBATCH -dEPSCrop'];
                                        % Essential.
211     gs=             [gs     ' -dPDFSETTINGS=/prepress -dEmbedAllFonts='
        fonts];     % Must be first?
212     gs=             [gs             ' -dUseFlateCompression=true'];
                                        % Useful stuff.
213     gs=             [gs             ' -dAutoRotatePages=/None'];
                                        % Probably good.
```

```
214         gs=             [gs             ' -dHaveTrueTypes'];
                                                    % Probably good.
215         gs=             [gs             ' ' res];
                                                                    %
               Add resolution to cmd.
216
217         if(crop && ismember(types, {'eps', 'pdf'}))
                                        % Crop the figure.
218                 fighdl= do_crop(fighdl);
219         end
220
221         % Output eps from Matlab.
222         renderer=       ['-' lower(get(fighdl, 'Renderer'))];
                                % Use same as in figure.
223         if(strcmpi(renderer, '-none')), renderer=       '-painters';    end
                        % We need a valid renderer.
224         print(fighdl, deps{:}, '-noui', renderer, res, [fname '-temp']);
                    % Output the eps.
225
226         % Convert to other formats.
227         for n= 1:length(types)
                                                    % Output them.
228                 if(isfield(device.(types{n}), color))
229                         cmd=            device.(types{n}).(color);
                                                            % Colour
                            model exists.
230                 else
231                         cmd=            device.(types{n}).rgb;
                                                            % Use
                            alternative.
232                         if(~strcmp(types{n}, 'eps'))    % It works anyways for
                             eps (VERY SHAKY!).
233                             warning('pax:savefig:deviceError', ...
234                                         'No device for %s using %s.
                                            Using rgb instead.', types
                                            {n}, color);
235                         end
236                 end
237                 cmp=    lossless;
238                 if (strcmp(types{n}, 'pdf')),   cmp= gsCompr;           end
                                        % Lossy compr only for pdf.
239                 if (strcmp(types{n}, 'eps')),   cmp= '';
                            end                         % eps can't use lossless.
240                 cmd=    sprintf('%s %s %s -f "%s-temp.eps"', gs, cmd, cmp,
                    fname);% Add up.
241                         status= system(cmd);
                                                                    %
                            Run Ghostscript.
242                         if (op_dbg || status),          display (cmd),
                                    end
243         end
244         delete([fname '-temp.eps']);
                                                    % Clean up.
245 end
246
```

```
247
248 function fig= do_crop(fig)
249 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
250 %       Remove line segments that are outside the view.
251 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
252
253         haxes=  findobj(fig, 'Type', 'axes', '-and', 'Tag', '');
254         for n=1:length(haxes)
255                 xl=             get(haxes(n), 'XLim');
256                 yl=             get(haxes(n), 'YLim');
257                 lines=  findobj(haxes(n), 'Type', 'line');
258                 for m=1:length(lines)
259                         x=                              get(lines(m), 'XData')
                            ;
260                         y=                              get(lines(m), 'YData')
                            ;
261
262                         inx=                    (xl(1) <= x) & (x <= xl(2));
                                % Within the x borders.
263                         iny=                    (yl(1) <= y) & (y <= yl(2));
                                % Within the y borders.
264                         keep=                   inx & iny;
                                                        % Within the box.
265
266                         if(~strcmp(get(lines(m), 'LineStyle'), 'none'))
267                                 crossx=         ((x(1:end-1) < xl(1)) & (xl(1)
                                    < x(2:end))) ...    % Crossing border x1.
268                                                 |       ((x(1:end-1) < xl(2))
                                                    & (xl(2) < x(2:end)))
                                                    ...     % Crossing border x2.
269                                                 |       ((x(1:end-1) > xl(1))
                                                    & (xl(1) > x(2:end)))
                                                    ...     % Crossing border x1.
270                                                 |       ((x(1:end-1) > xl(2))
                                                    & (xl(2) > x(2:end)));
                                                    % Crossing border x2.
271                                 crossy=         ((y(1:end-1) < yl(1)) & (yl(1)
                                    < y(2:end))) ...    % Crossing border y1.
272                                                 |       ((y(1:end-1) < yl(2))
                                                    & (yl(2) < y(2:end)))
                                                    ...     % Crossing border y2.
273                                                 |       ((y(1:end-1) > yl(1))
                                                    & (yl(1) > y(2:end)))
                                                    ...     % Crossing border y1.
274                                                 |       ((y(1:end-1) > yl(2))
                                                    & (yl(2) > y(2:end)));
                                                    % Crossing border y2.
275                                 crossp= [(      (crossx & iny(1:end-1) & iny
                                    (2:end)) ...    % Crossing a x border within
                                    y limits.
276                                                 |       (crossy & inx(1:end-1)
                                                     & inx(2:end)) ...   %
                                                    Crossing a y border within
                                                     x limits.
277                                                 |       crossx & crossy
```

```
                                                    ...

                                                                % Crossing a x and
                                                              a y border (corner).
278                                                    ),       false ...
279                                                   ];
280                                 crossp(2:end)=  crossp(2:end) | crossp(1:end
                                        -1);                  % Add line segment's
                                        secont end point.
281
282                                 keep=                     keep | crossp;
283                             end
284                             set(lines(m), 'XData', x(keep))
285                             set(lines(m), 'YData', y(keep))
286                     end
287             end
288 end
```