Blase Londoño

Tania Morimoto

MAE 3 B01

4/17/2025

Clock Report

Executive Summary

The purpose of this project is to develop design strategy, manufacturing, analysis, and project management skills through the creation of a simple gravity-driven pendulum clock. The clock is manufactured in the mechanical engineering design studio, and introduces its students to studio machinery (drills, arbor presses, laser cutters) as well as elements of design (tapped and clearance holes, types of bolts and fits between materials). The students then analyze their clock's performance, comparing its actual physical properties and natural frequency to the predicted frequencies calculated through two distinct physical models for the pendulum: point mass and rigid body analysis. In this report, this pendulum was found to contain an 8.9 percent error deviation from the theoretical frequency of the point mass analysis, and a 2.8 percent error deviation from the theoretical frequency of the rigid body analysis. These error percentages are likely due to manufacturing errors, errors in the measurement and recording of state variables (mass, distance) in the setup of the physics analysis models, and rounding in the final analysis of the natural frequencies. Despite these errors, these results indicate that the pendulum was fabricated to an acceptable degree of accuracy, as its performance is similar to that of the theoretical design, and the error percentages remain within the maximum amount (specified as less than ten percent error).

Theoretical Analysis

Two distinct methods for calculating the period of a pendulum were employed in the timing analysis. The first method, *Calculating Natural Frequency Using Point Mass*, operates under the principle of simplifying the model by concentrating all the mass distributed throughout the pendulum into a single point located directly on the pendulum's center of mass. This method simplifies calculations, as the rotational inertia of the pendulum is disregarded. As a consequence of

the additional assumptions, however, point mass analysis also tends to be less accurate. Thus, another method can be employed to ensure that the predicted period of a pendulum is accurate. This method, called *Natural Frequency Using Rigid Body*, operates by calculating the moments of inertia of the pendulum and all eight weights about the pivot point, and using the summation of those inertias as the simplified representation of the pendulum's mass. Because this method does not neglect the distribution of mass around the pendulum, it is expected to be slightly more accurate. Below is a table comparing the components of each method.

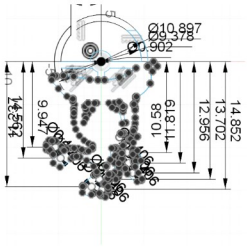
Table 1: Point Mass vs. Rigid Body Analysis

	Natural Frequency Using Point Mass	Natural Frequency Using Rigid Body
Assumptions	1. Friction neglected 2. Small range of motion 3. Pendulum swings freely 4. All mass concentrated in a single point	1. Friction neglected 2. Small range of motion 3. Pendulum swings freely
Variables	1. Mass of pendulum 2. Mass of each bolt 3. Number of bolts 4. Length to center of mass without bolts added 5. Length to center of mass of each bolt 6. Estimated center of mass of pendulum with bolts added	1. Mass of pendulum 2. Mass of each bolt 3. Number of bolts 4. Moment of inertia of the pendulum without bolts added 5. Moment of inertia at each bolt
Equations	$M_t = M_{calc} + n \cdot M_b$ $L_{com} = \frac{M_{calc} \cdot L_a + M_b \cdot (L_{b1} ... L_{bn})}{M_t}$ $\omega = \sqrt{\frac{g}{L_{com}}}$ $f = \frac{\omega}{2\pi}$ $T = \frac{1}{f}$	$I_i = mr_i^2$ $I = \int_m r^2 dm = \sum_i I_{b_i}$ $\omega = \sqrt{\frac{M_t \cdot g \cdot L_{com}}{I}}$ $f = \frac{\omega}{2\pi}$ $T = \frac{1}{f}$

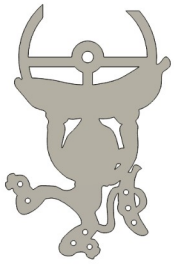
Table 2: Pendulum Properties

CAD of pendulum

Length calculation sketch



Top view



Isometric view

Front view

Right view

Pendulum physical properties

Cross-sectional area

Area	109.34 cm^2
Loop Length	156.169 cm

Physical properties

Mass	82.482 g
Volume	69.429 cm^3
Density	1.188 g / cm^3
Area	317.850 cm^2
World X,Y,Z	0.00 cm, 0.00 cm, 0.00 cm
Center of Mass	0.196 cm, -5.964 cm, 0.318 cm

Moment of inertia

Izz	5376.049

Predicted weight

$$w = m \cdot g = 0.083 \cdot 9.81 \approx 0.814kg$$

Experimental Results

To calculate the actual natural frequency and time for one complete revolution of the pendulum, three trials were hosted where the pendulum was allowed to “tick” (travel half a period, moving the gear along once) fourteen times. The total time for one revolution was recorded via stopwatch, and these times were averaged to find the actual natural pendulum frequency. This

value was compared to both methods for calculating natural frequency, that is, rigid body and point mass analysis. As expected, the method of rigid body analysis proved to be significantly more accurate than the method of point mass analysis, returning a percent error of ~2.8 percent, drastically less than the 8.9 percent error from point mass. This discrepancy in accuracy likely stems from the additional assumption (concentration of mass) inherent to the method of point mass analysis: while circumventing the moment of inertia calculations proves useful for quick, rough estimates, it is clear that rigid body analysis is still the more accurate method in the end.

Table 3: Comparison of Results

Method	Predicted time	Actual time	Percent error
Point mass analysis	8.264215579	9	8.903257841
Rigid body analysis	9.2585871	9	2.792943433

Of course, there is still significant error even in the rigid body analysis. There are several causes for error in our analysis. For example, all length and distance calculations were performed on the Autodesk Fusion CAD design, instead of with the physical model, which does not account for imperfections in the pendulum's size or scale. Secondly, the bolts were not weighed out individually (weighed only once), which does not account for imperfections in the weight of each individual mass. Further, there does exist friction between our bearing and pendulum shaft, which also affects the pendulum's final frequency. There are many more examples of possible errors in these analyses, these are just a few examples. Errors in measurement and calculation such as these can be prevented in future projects through more thorough definition of each object (e.g. weight), as well as utilizing less assumptions (such as negligent weight, constant oscillation) when setting up a physical model.

Discussion

To summarize the findings in the experimental results page, the rigid body analysis provided an error of approximately 2.8 percent, while the point mass analysis provided an error of approximately 8.9 percent. While the latter is close to the limit, both methods provide a tolerable level of error for this experiment.

Table 4: Composition of Error

	Pendulum weight (g)	Pendulum centroid (m)
Predicted	82.485	0.086
Actual	~84	0.011
Error	1.803%	28.1%

Notably, the percent error for the centroid is exceedingly high. This is due to a significant error in the process of documenting for this report and the fabrication of the pendulum– the bearing was press-fit into the pendulum and the weights were affixed before measuring the actual centroid, which significantly altered the center of mass from the predicted value. Despite this, the estimated natural frequencies for rigid body (and to a lesser extent, point mass) analysis remain within a tolerable level of accuracy.

Attempt At Optimization

While unsuccessful, an attempt at optimizing the locations of each of the eight weights was made during the initial stages of this project. To optimize the weights, a constraint function was derived from four constraints:

1. The weights must remain further than 4 inches from the pivot point of the pendulum.

2. The weights must not touch.
3. The weights must not intersect with the edges of the pendulum.
4. The weight array must preserve a period of 2 seconds (one tick/second).

To achieve a suitable result for this optimization, a MATLAB script was developed, which would "brute-force" thousands of possible combinations of weight layouts to determine an optimal solution. Below are important components of the MATLAB script.

Table 5: Pendulum Optimization Script

```
function optimize_pendulum()
  % variables
  target_period = 2.0;              % desired period
  gravity = 9.81;
  pendulum_mass = 0.08325;          % mass of pendulum
  pendulum_COM = [0.00193, -0.06026]; % center of mass of pendulum
  pendulum_moment = 0.00507;        % moment of inertia about pivot
  weight_mass = 0.04;               % mass of each weight
  number_of_weights = 8;            % number of weights
  % load outline
  % load poly_x region x-coordinates forall
  % load poly_y region y-coordinates forall
  load('pendulum_shape.mat', 'poly_x', 'poly_y');
  % initialize targets
  total_mass = pendulum_mass + (number_of_weights * weight_mass);
  target_y_COM = (pendulum_mass * pendulum_COM(2)) / total_mass;
  target_moment = total_mass * abs(target_y_COM) * gravity * target_period^2 / (4 * pi^2);
% target moment for desired period
  % build base case
  theta = linspace(0, 2*pi, number_of_weights);
  r = 0.05;
  initial_guess = reshape([r*cos(theta); -0.25 + r*sin(theta)], 1, []);

  lb = repmat([-0.5, -0.8], 1, number_of_weights); % lower bounds
  ub = repmat([ 0.5,  0.8], 1, number_of_weights); % upper bounds
  % define options
  options = optimoptions('fmincon', ...
    'Display','iter', ...
```

```
    'MaxFunctionEvaluations',2e5, ...
    'ConstraintTolerance',1e-6, ...
    'StepTolerance', 1e-10);
% define problem
% objective: objective function
% x0: initial point
% lb: lower bounds
% ub: upper bounds
% nonlcon: nonlinear constraints
% options: optimization options
problem = createOptimProblem('fmincon', ...
    'objective', @(w) objective(w, weight_mass, number_of_weights, pendulum_mass,
pendulum_COM, pendulum_moment, target_moment), ...
    'x0', initial_guess, ...
    'lb', lb, ...
    'ub', ub, ...
    'nonlcon', @(w) constraint(w, poly_x, poly_y), ...
    'options', options);

% start globalsearch call
gs = GlobalSearch('Display','iter',...
    'NumTrialPoints', 30, ...
    'NumStageOnePoints', 10, .....
    'MaxTime',600);
[best_weights, ~, exitflag, output, ~] = run(gs, problem);

% ensure feasibility
if exitflag <= 0
    error('Optimization exited early: %s', output.message);
end
% vectorize results
x = best_weights(1:2:end);
y = best_weights(2:2:end);

% print weight values
fprintf('\nBest weight positions (x, y):\n');
for i = 1:number_of_weights
     fprintf('Weight %d: (%.4f, %.4f) m\n', i, x(i), y(i));
end
% plot results
figure;
fill(poly_x(:), poly_y(:), [0.85 0.85 0.85]); hold on;
axis equal; grid on;
```

```
    title('Optimized Weight Positions Inside Pendulum');
    xlabel('x (m)'); ylabel('y (m)');

    xlim([min(poly_x)-0.02, max(poly_x)+0.02]);
    ylim([min(poly_y)-0.02, max(poly_y)+0.02]);

    % Plot hole and spacing buffers
    hole_radius = 0.002032;       % 0.08" in meters
    spacing_buffer = 0.00508;     % 0.2" in meters
    theta = linspace(0, 2*pi, 100);

    for i = 1:number_of_weights
        % throughhole (solid red)
        cx = x(i) + hole_radius * cos(theta);
        cy = y(i) + hole_radius * sin(theta);
        fill(cx, cy, 'r', 'EdgeColor', 'k', 'FaceAlpha', 0.6);

        % spacing buffer (dashed red)
        bx = x(i) + spacing_buffer * cos(theta);
        by = y(i) + spacing_buffer * sin(theta);
        plot(bx, by, 'r--');
    end
end
```

```
function err = objective(w, m_w, n, m_p, CoM_p, I_p, T_target)
    g = 9.81;
    % Extract x and y coordinates of weights
    x = w(1:2:end);
    y = w(2:2:end);
    % Total mass of pendulum system
    total_mass = m_p + n * m_w;
    % Center of mass
    x_cm = (m_p * CoM_p(1) + m_w * sum(x)) / total_mass;
    y_cm = (m_p * CoM_p(2) + m_w * sum(y)) / total_mass;
    % Moment of inertia about pivot
    I_weights = sum(m_w * (x.^2 + y.^2));
    I_total = I_p + I_weights;
    % Compute actual period from physics
    T_actual = 2 * pi * sqrt(I_total / (total_mass * g * abs(y_cm)));
    % Objective: minimize squared error in period
    period_error = (T_actual - T_target)^2;
```

```
    % Penalty for center of mass x offset (should be aligned vertically)
    cm_penalty = 10 * x_cm^2;
    % Penalty for spacing violations
    spacing_penalty = 0;
    min_spacing = 0.00508;  % 0.2" in meters
    for i = 1:n
        for j = i+1:n
            d_sq = (x(i) - x(j))^2 + (y(i) - y(j))^2;
            if d_sq < min_spacing^2
                spacing_penalty = spacing_penalty + 1e5 * (min_spacing^2 - d_sq)^2;
            end
        end
    end
    % Penalty for violating 4" radius
    radius_penalty = 0;
    min_radius = 0.1016;  % 4" in meters
    for i = 1:n
        r_sq = x(i)^2 + y(i)^2;
        if r_sq < min_radius^2
            radius_penalty = radius_penalty + 1e5 * (min_radius^2 - r_sq)^2;
        end
    end
    % Combine everything into a single error to minimize
    err = period_error + cm_penalty + spacing_penalty + radius_penalty;
end
```

```
function [c, ceq] = constraint(w, poly_x, poly_y)
    x = w(1:2:end); % x-coordinate vector
    y = w(2:2:end); % y-coordinate vector
    n = length(x);  % index
    % restrict iterations to within the polygon's outline
    inside = inpolygon(x(:), y(:), poly_x(:), poly_y(:));  % define polygon inner region
    bound_constraint = double(~inside(:));                % store all violations
    % keep iterations 4" away from origin (pivot)
    minimum_radius = 0.1016;                              % 4" in meters
    radius_constraint = minimum_radius^2 - (x.^2 + y.^2);  % store all centerpoint radii (from
origin)
    % keep centerpoint of weights 0.4" away from each other
    % ISSUE: constraint vector not preallocated, very slow :(
    minimum__centerpoint_distance = 0.00508;                                % 0.2" (radius)
    spacing_constraint = [];                              %initialize constraint vector
    for i = 1:n                                       % first-point iteration
```

```
        for j = i+1:n                                % second-point iteration
            distance_squared = (x(i) - x(j))^2 + (y(i) - y(j))^2;
            spacing_constraint(end+1) = minimum__centerpoint_distance^2 - distance_squared; % store all centerpoint distances
        end
    end
    % polygon-centerpoint edge buffer constraint
    edge_buffer = 0.00254;                    % define buffer as 0.16" + 0.1" (in meters)
    edge_constraint = zeros(n,1);             % define constraint vector
    for i = 1:n                               % iterate across index
        minimum_edge_distance = inf;          % cannot be 0 or would always return 0
        for j = 1:length(poly_x)              % iterate across x-values
            j2 = mod(j, length(poly_x)) + 1;
            point_1 = [poly_x(j), poly_y(j)];     % define comparison point 1
            point_2 = [poly_x(j2), poly_y(j2)];   % define comparison point 2
            edge_distance = point_to_segment_distance([x(i), y(i)], point_1, point_2); % calculate edge distance
            minimum_edge_distance = min(minimum_edge_distance, edge_distance);        % find closest edge
        end
        edge_constraint(i) = edge_buffer - minimum_edge_distance;  % store all closest edges
    end

    % vectorize all constraints
    c = [bound_constraint(:); radius_constraint(:); spacing_constraint(:); edge_constraint(:)];
    ceq = [];
end
```

```
function d = point_to_segment_distance(p, a, b)
    % computes the shortest distance from point p to line segment ab
    ab = b - a;                               % difference
    t = max(0, min(1, dot(p - a, ab) / dot(ab, ab)));  % proj dot product considerations
    proj = a + t * ab;                        % definition of projection
    d = norm(p - proj);                       % perpendicular decomposition of projection
end
```

Each MATLAB function is annotated for the benefit of the viewer, however, to summarize, the optimize_pendulum() function sets up all of the constants required for point mass analysis of

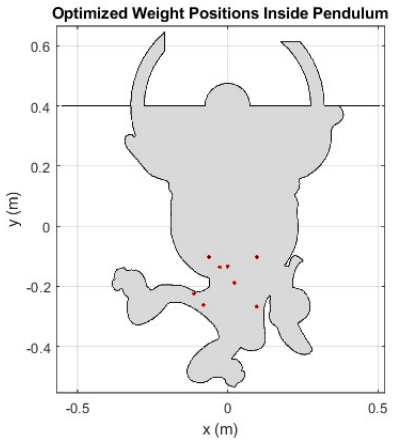
the pendulum (point mass for simplification of calculations, as they will be passed through thousands of iterations). The optimize_pendulum() function then builds target values for the pendulum's period, calls the optimization toolbox, and displays the results of the optimizer once the results are returned.

The aptly-named objective() function calculates each individual weight distribution's moments of inertia and error percentages. It also includes some hard-coded restrictions to the distribution of weights to ensure the penalties for essential constraints are not ignored. The constraint() function acts as a reference to the objective() function, defining each of the four constraints listed above and adjusting penalty values for each violation found in the optimizer distributions. Lastly, point_segment_to_distance() simply acts as a projection function in MATLAB to clear up space in the other, bulkier functions. Together, these scripts aim to find an ideal distribution for the eight weights such that they preserve a desired period. Unfortunately, due to time constraints for this project, this script was never fully completed, and the results displayed are not completely accurate. The optimization attempt and script are included in this report as an exploration of optimization in relation to the design process in engineering projects.

Below is a table categorizing the results of the script.

Table 6: Optimization Results

Optimizer results (graphed)	Optimizer output
	Best weight positions (x, y): Weight 1: (-0.0266, -0.1364) m Weight 2: (0.0966, -0.1028) m Weight 3: (-0.0621, -0.1025) m Weight 4: (-0.0807, -0.2616) m Weight 5: (-0.1123, -0.2249) m Weight 6: (0.0964, -0.2672) m



Weight 7: (0.0216, -0.1884) m
Weight 8: (-0.0003, -0.1336) m

Appendix

Pendulum Timing Analysis

Name: Blase Londoño
Section: B01

Variable Description	Variable Name	Values/Equations	Units	Comments
Acrylic Pendulum Specifications				
Area	A	109.341	cm^2	
Thickness	t	0.635	cm	
Volume	Vol	69.431535	cm^3	
Density	p	1.188	gm/cm^3	
Calculated Mass of Acrylic	M_Calc	82.48466358	gm	
Actual Mass of Acrylic	M_Act	84	gm	
Length to Center of Mass of Acrylic	La	7.173	cm	acrylic only
Percent Error in Acrylic Mass Calculation	M_Error	0.000180397192	%	
Calculate Total Mass of Pendulum				
Mass of One Bolt with Two Nuts	Mb	4	gm	
Number of Bolts with Two Nuts	Nb	8	bolts	

Total Mass of Pendulum with Nuts and Bolts	Mt	114.4846636	gm	calculated
Calculate Center of Mass of Pendulum with Bolts				
Length to Center of Mass of Bolt 1	L_bolt1	9.947	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 2	L_bolt2	11.237	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 3	L_bolt3	14.562	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 4	L_bolt4	13.702	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 5	L_bolt5	14.853	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 6	L_bolt6	10.58	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 7	L_bolt7	11.819	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 8	L_bolt8	12.956	cm	distance from pivot point to bolt
Length to Center of Mass **in Meters**	Lcom_meter	0.08649948918	**meters**	calculate with acrylic and bolts
Estimated Center of Mass of Pendulum with Nuts and Bolts	Lcom_est	0.11	meters	intermediate verification with acrylic and bolts
Percent Error in Pendulum Nuts and Bolts Lcom Estimate	Lcom_error	1.127168381	%	
Calculate Natural Frequency and Timing using Point Mass Assumption				
Gravitational Constant	g	9.8	m/s^2	
Natural Frequency in radians/sec	nat_freq_rad_sec	10.64403433	rad/s	
Natural Frequency in Hz	nat_freq_hz	1.694050678	1/s	
Period of Oscillation	period	0.5903011128	s	
Number of Teeth on Escapement Wheel	nteeth	14	teeth	
Calculated Time of One Revolution of Escapement Wheel	time_calc	8.264215579	s	
Measured Time of One Revolution of Escapement Wheel	time_meas	9	s	
Percent Error in Clock Timing	time_error	8.903257841	%	
Calculate Natural Frequency and Timing using Rigid Body Assumption				
Moment of Inertia of Pendulum	I_a	5376.049	g*cm^2	
Moment of Inertia of Bolt 1	I_bolt1	509.224356	g*cm^2	
Moment of Inertia of Bolt 2	I_bolt2	594.481924	g*cm^2	
Moment of Inertia of Bolt 3	I_bolt3	829.670416	g*cm^2	
Moment of Inertia of Bolt 4	I_bolt4	751.088836	g*c	

			m^2	
Moment of Inertia of Bolt 5	I_bolt5	882.446436	g*cm^2	
Moment of Inertia of Bolt 6	I_bolt6	484.264036	g*cm^2	
Moment of Inertia of Bolt 7	I_bolt7	599.2704	g*cm^2	
Moment of Inertia of Bolt 8	I_bolt8	724.794084	g*cm^2	
Total Moment of Inertia	I_total	0.001075128949	kg*m^2	
Natural Frequency in radians/sec	rb_nat_freq_rad_sec	9.500865882	rad/s	
Natural Frequency in Hz	rb_nat_freq_hz	1.512109769	hz	
Period of Oscillation	rb_period	0.66132765	s	
Calculated Time of One Revolution of Escapement Wheel	rb_time_calc	9.2585871	s	
Percent Error in Clock Timing	rb_time_error	2.792943433	%	

Pendulum Timing Analysis

Name: Blase Londoño				
Section: B01				
Variable Description	**Variable Name**	**Values/Equations**	**Units**	**Comments**
Acrylic Pendulum Specifications				
Area	A	109.341	cm^2	
Thickness	t	0.635	cm	
Volume	Vol	=MULTIPLY(A,t)	cm^3	
Density	p	1.188	gm/cm^3	
Calculated Mass of Acrylic	M_Calc	=MULTIPLY(Vol,p)	gm	
Actual Mass of Acrylic	M_Act	84	gm	
Length to Center of Mass of Acrylic	La	7.173	cm	acrylic only

Percent Error in Acrylic Mass Calculation	M_Error	=DIVIDE(SUM(M_Act,-M_Calc),MULTIPLY(M_Act,100))	%	
Calculate Total Mass of Pendulum				
Mass of One Bolt with Two Nuts	Mb	4	gm	
Number of Bolts with Two Nuts	Nb	8	bolts	
Total Mass of Pendulum with Nuts and Bolts	Mt	=SUM(M_Calc,MULTIPLY(Mb,Nb))	gm	calculated
Calculate Center of Mass of Pendulum with Bolts				
Length to Center of Mass of Bolt 1	L_bolt1	9.947	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 2	L_bolt2	11.237	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 3	L_bolt3	14.562	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 4	L_bolt4	13.702	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 5	L_bolt5	14.853	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 6	L_bolt6	10.58	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 7	L_bolt7	11.819	cm	distance from pivot point to bolt
Length to Center of Mass of Bolt 8	L_bolt8	12.956	cm	distance from pivot point to bolt
Length to Center of Mass **in Meters**	Lcom_meter	=MULTIPLY(DIVIDE(SUM(MULTIPLY(M_Calc,La),MULTIPLY(Mb,SUM(L_bolt1,L_bolt2,L_bolt3,L_bolt4,L_bolt5,L_bolt6,L_bolt7,L_bolt8))),Mt),0.01)	**meters**	calculate with acrylic and bolts
Estimated Center of Mass of Pendulum with Nuts and Bolts	Lcom_est	0.011	meters	intermediate verification with acrylic and bolts
Percent Error in Pendulum Nuts and Bolts Lcom Estimate	Lcom_error	=ABS(DIVIDE(SUM(Lcom_meter,Lcom_est),Lcom_meter))	%	
Calculate Natural Frequency and Timing using Point Mass Assumption				
Gravitational Constant	g	9.8	m/s^2	
Natural Frequency in radians/sec	nat_freq_rad_sec	=SQRT(DIVIDE(g,Lcom_meter))	rad/s	
Natural Frequency in Hz	nat_freq_hz	=DIVIDE(nat_freq_rad_sec,M	1/s	

		ULTIPLY(2,PI()))		
Period of Oscillation	period	=DIVIDE(1,nat_freq_hz)	s	
Number of Teeth on Escapement Wheel	nteeth	14	teeth	
Calculated Time of One Revolution of Escapement Wheel	time_calc	=MULTIPLY(nteeth,period)	s	
Measured Time of One Revolution of Escapement Wheel	time_meas	9	s	
Percent Error in Clock Timing	time_error	=ABS(time_meas-time_calc)/time_calc%	%	
Calculate Natural Frequency and Timing using Rigid Body Assumption				
Moment of Inertia of Pendulum	I_a	5376.049	g*cm^2	
Moment of Inertia of Bolt 1	I_bolt1	=MULTIPLY(Mb,11.283^2)	g*cm^2	
Moment of Inertia of Bolt 2	I_bolt2	=MULTIPLY(Mb,12.191^2)	g*cm^2	
Moment of Inertia of Bolt 3	I_bolt3	=MULTIPLY(Mb,14.402^2)	g*cm^2	
Moment of Inertia of Bolt 4	I_bolt4	=MULTIPLY(Mb,13.703^2)	g*cm^2	
Moment of Inertia of Bolt 5	I_bolt5	=MULTIPLY(Mb,14.853^2)	g*cm^2	
Moment of Inertia of Bolt 6	I_bolt6	=MULTIPLY(Mb,11.003^2)	g*cm^2	
Moment of Inertia of Bolt 7	I_bolt7	=MULTIPLY(Mb,12.24^2)	g*cm^2	
Moment of Inertia of Bolt 8	I_bolt8	=MULTIPLY(Mb,13.461^2)	g*cm^2	
Total Moment of Inertia	I_total	=MULTIPLY(SUM(I_a,I_bolt1,I_bolt2,I_bolt3,I_bolt4,I_bolt5,I_bolt6,I_bolt7,I_bolt8),0.0000001)	kg*m^2	
Natural Frequency in radians/sec	rb_nat_freq_rad_sec	=SQRT(DIVIDE(MULTIPLY(MULTIPLY(Mt,0.001),MULTIPLY(g,Lcom_meter)),I_total))	rad/s	

Natural Frequency in Hz	rb_nat_freq_hz	=DIVIDE(rb_nat_freq_rad_sec,2*PI())	hz	
Period of Oscillation	rb_period	=DIVIDE(1,rb_nat_freq_hz)	s	
Calculated Time of One Revolution of Escapement Wheel	rb_time_calc	=rb_period*nteeth	s	
Percent Error in Clock Timing	rb_time_error	=ABS(time_meas-rb_time_calc)/rb_time_calc%	%	