Milk Frother:

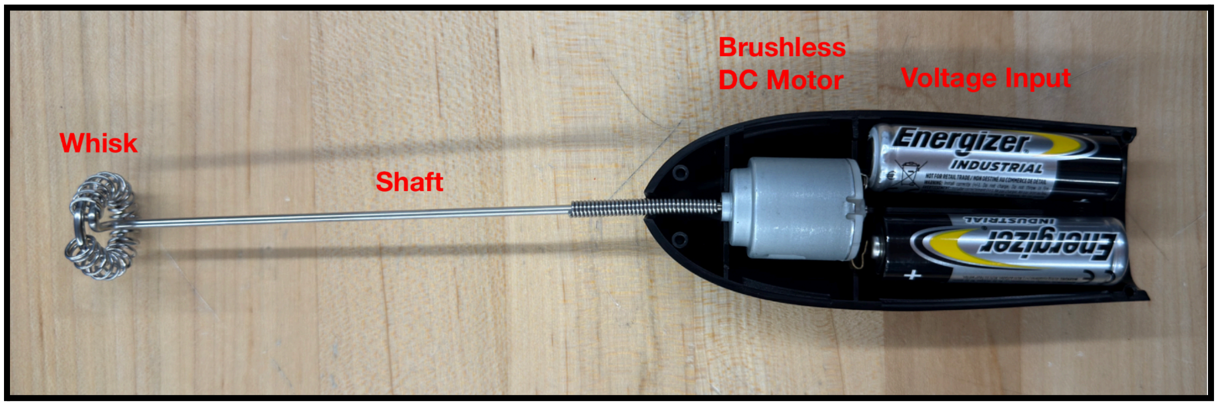


Figure 1: milk frother dissected

The whisk is driven by a 3 coil brushless DC motor powered by batteries. The motor is connected to a shaft via a spring. The spring acts as a flexible coupler between the motor and the shaft so the two do not have to be perfectly aligned. When powered, the whisk will start turning counterclockwise (when looking down from the handle). Then, it seems to operate at a constant steady steady rotation rate. When powered off, the whisk starts slowing and eventually comes to an abrupt stop.

Open Loop Behavior:

To observe the open loop behavior, we turned on the milk frother and waited for it to reach steady state. Then, we used a high speed camera to record the behavior. 1275 frames were recorded at 900 frames per second. The video was played back at approximately 30 frames per second. We manually counted 40 rotations for 6 seconds. This means that our milk frother's steady state speed was around 12000 rotations per minute or 1240 rad/s. This is similar to the reported RPM of milk frothers on the internet.

Then, we recorded the step response of the milk frother. To do this, we started from a steady state and turned off the frother. Every 60 frames (2 seconds of the playback video), the angular velocity was approximated by counting the number of frames for one rotation.

Ramp Down Data Collection:

When finding the angular velocity at different times in the video, we looked at the high speed video frame by frame. Within the ramp down video of the milk frother, the video took 900 frames per second(fps), this resulted in a 44 second video from about 1.4 seconds worth of live content. From there, we measured how many frames it took for one full rotation(fpr) every two seconds of the video. This allows us to find the rpm at each second of the video, and then the rpm at each second in real time.
Finding rpm per each second of the video:

$$rpm_{video\ second} = \frac{60 * fpr}{fps}$$

Converting the times in the video to time in real life:

$$t_{real} = \frac{t_{video} * \frac{Frames\ in\ Video}{fps}}{Video\ Length}$$

Ramp Down Data:

Following the steps outlined above, we collected and graphed the following data:

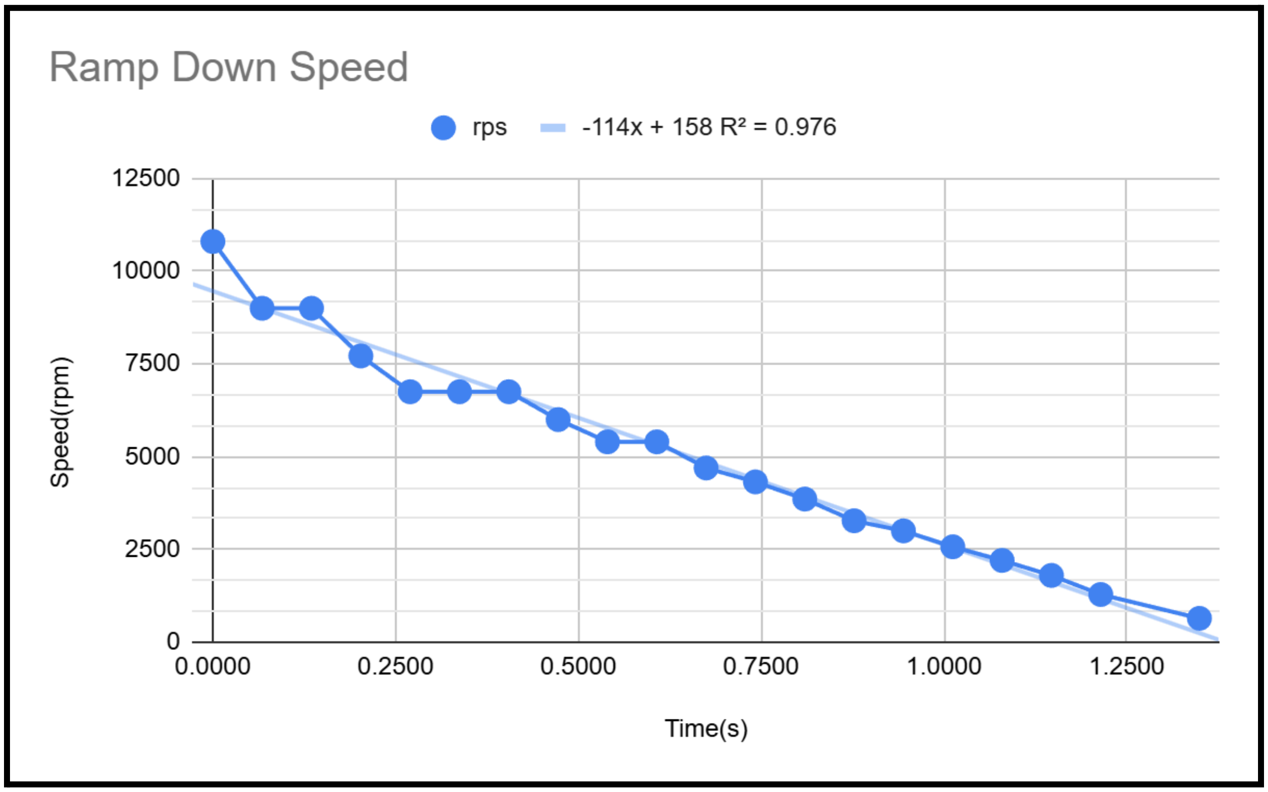


Figure 2: rotation rate measured every 2 seconds for the on to off step response of the frother

System Dynamics:

To accurately describe the system dynamics of the milk frother as it ramps down, we need to determine which physical damping model is best and how this helps us extract information about how the motor and whisk loses power. The rotational speed can decay due to various reasons such as friction in the bearings, leading to viscous damping modeled by:

$$\omega(t) = \omega_0 e^{-t/\tau}$$

$\omega(t)$: angular velocity as a function of time (rad/s)
ω_0: angular velocity at switch-off (initial condition) (rad/s)
t: time since switch-off (s)

τ: time constant for exponential decay, τ=J/b (s).
b: linear (viscous) damping coefficient (N·m·s/rad).

If instead the decay is dominated by aerodynamic drag, it is governed by a quadratic torque model, algebraically represented by:

$$\frac{1}{\omega(t)} = \frac{1}{\omega_0} + \frac{c}{J}t$$

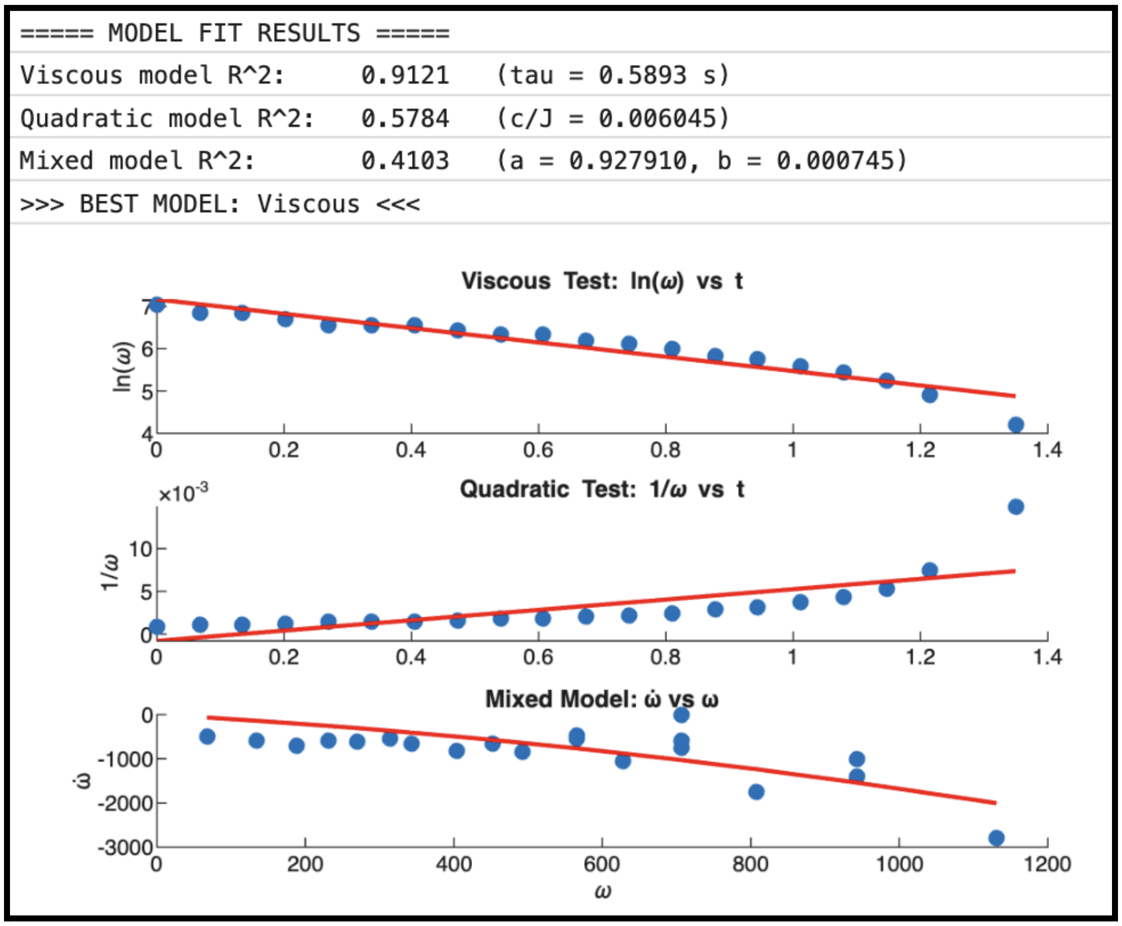
J: moment of inertia of rotor + whisk (kg·m²)
c: quadratic (aerodynamic) drag coefficient (N·m·s²/rad²), with drag torque $\tau_{drag}=-c\omega^2$.

Lastly, if the decay is a mix of both aerodynamic drag and viscous damping, we get a mixed linear model described by:

$$J\frac{d\omega}{dt} + b\omega + c\omega^2 = 0$$

A MatLab script can then be written, comparing the models and their accuracy (via R^2) to identify which physical model fits the milk frother coast down the best. This allows us to estimate certain system parameters.

MatLab output (code in Appendix)



```
===== MODEL FIT RESULTS =====
Viscous model R^2:      0.9121    (tau = 0.5893 s)
Quadratic model R^2:    0.5784    (c/J = 0.006045)
Mixed model R^2:        0.4103    (a = 0.927910, b = 0.000745)
>>> BEST MODEL: Viscous <<<
```

The R^2 value is the highest for the viscous model, implying that the ramp down systems dynamics is dominated by viscous damping, likely from the friction in the motor. While the code calculates constants from the slope in each graph. Since we can now determine the system to be dominated by viscous damping, we don't look at the c/J, a, and b values, as the tau is the most accurate.

Since we are only now looking at the viscous model, the time constant was calculated to be {}s. We also manually timed the time it took for frother to fully stop, averaging out to 1.846s.

$\Delta ln(\omega)$ = 7.16902 - 4.87911 = 2.28991

$\omega_0 -$ 0.632 y_{ss}= 7.16902 - 0.632 (2.28991) = 5.722

From the MatLab plot, this corresponds to a τ of 0.8528. With a τ of 0.5893 calculated by τ = -1/slope in the code, this leads to a 44.71% error, which while large, is acceptable when putting the scale of the values into reference. Since this is a first order system, the difference between τ values only scales the time axis, stretching the graph, it does not distort the overall shape of the model.

We can also then calculate J/b:

$$\tau = J/b = 0.8528$$

State Space Model:

Our system is a first-order ODE, therefore to find the space state model of this system, we take the state so that it is the output. Therefore, $x = \omega$, and $\dot{x} = \dot{\omega}$.

The resulting function becomes:

$$J\dot{x} + bx = Ku(t)$$

Solving for the state equations:

$$\dot{x} = \left[-\frac{b}{J}\right]x + \left[\frac{k}{J}\right]u(t)$$

The output is the same as the state, therefore the equation for the output is:

$$y = x = \omega$$

The state-space matrices A, B, C, D become:

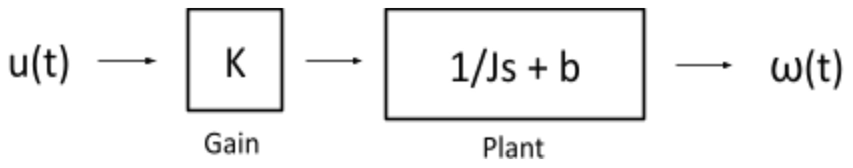
$$A = \left[-\frac{b}{J}\right]x, \quad B = \left[\frac{K}{J}\right]u(t), \quad C = [1]x, \quad D = [0]$$

Block Diagram:

The transfer function of the ODE is:

$$G(s) = \frac{\Omega(s)}{U(s)} = \frac{K}{Js+b}$$

Therefore, the block diagram for this system becomes:



- **u(t)** represents the electrical input, specifically the motor's applied voltage, provided by the batteries.
- **K** represents the actuator gain, characterizing the efficiency in which the motor converts the input voltage into torque.
- **1/Js+b** represents the plant, converting torque into angular velocity. The term Js captures the rotor's rotational inertia, and b represents viscous damping from fluid.
- **ω(t)** represents the system output in angular speed.

Changing Disturbance:

To use the milk frother, it needs to spin in milk, not air. We were unable to document this behavior, but we can discuss what would happen to the system if the disturbance is different. A different fluid would change the drag force on the whisk, which means there would be an extra drag factor of $b\omega$. This means the governing equation from above would still hold, except the time constant and damping value would be different depending on the fluid.

MatLab Code:

```
% --- RAW DATA ---
t = [ ...
0.0000
0.0675
0.1349
0.2024
0.2699
0.3374
0.4048
0.4723
0.5398
0.6073
0.6747
0.7422
0.8097
0.8772
0.9446
1.0121
1.0796
1.1471
1.2145
1.3495
];   % time in seconds
rpm = [ ...
10800
9000
9000
7714.285714
6750
6750
6750
6000
5400
5400
4695.652174
4320
3857.142857
3272.727273
3000
2571.428571
2204.081633
1800
1285.714286
642.8571429
];    % measured rpm values

% Convert rpm → rad/s
omega = rpm * 2*pi/60;

%% ================================
%  1. VISCOUS DAMPING TEST
%     ω(t) = ω0 * exp(-t/τ)
% =================================
ln_omega = log(omega);
% Linear regression: ln(ω) = a + b t
p_visc = polyfit(t, ln_omega, 1);
ln_fit = polyval(p_visc, t);
% Extract time constant τ
tau = -1/p_visc(1);
% Compute R²
SS_res_visc = sum((ln_omega - ln_fit).^2);
SS_tot_visc = sum((ln_omega - mean(ln_omega)).^2);
R2_visc = 1 - SS_res_visc/SS_tot_visc;

%% ================================
%  2. QUADRATIC DRAG TEST
%     1/ω(t) = 1/ω0 + (c/J)t
% =================================
inv_omega = 1 ./ ln_omega;
% Linear regression
p_quad = polyfit(t, inv_omega, 1);
inv_fit = polyval(p_quad, t);
% Extract slope = c/J
c_over_J = p_quad(1);
% Compute R²
SS_res_quad = sum((inv_omega - inv_fit).^2);
SS_tot_quad = sum((inv_omega - mean(inv_omega)).^2);
R2_quad = 1 - SS_res_quad/SS_tot_quad;

%% ================================
%  3. MIXED MODEL (NONLINEAR FIT)
%     dω/dt = -a ω - b ω²
% =================================
% Numerically estimate dω/dt
domega = gradient(omega, t);
% ODE model: domega/dt = -a*omega - b*omega.^2
model_fun = @(p, w)  -(p(1)*w + p(2)*w.^2);
% Nonlinear least squares fit
p0 = [0.1, 0.001];   % initial guess [a, b]
```

```
p_mixed = lsqcurvefit(model_fun, p0, omega, domega);
a = p_mixed(1);
b = p_mixed(2);
% Mixed model prediction
domega_fit = model_fun(p_mixed, omega);
% Compute R²
SS_res_mixed = sum((domega - domega_fit).^2);
SS_tot_mixed = sum((domega - mean(domega)).^2);
R2_mixed = 1 - SS_res_mixed/SS_tot_mixed;

%% ================================
%  REPORT RESULTS
% =================================
fprintf('\n===== MODEL FIT RESULTS =====\n');
fprintf('Viscous model R^2:     %.4f   (tau = %.4f s)\n', R2_visc, tau);
fprintf('Quadratic model R^2:   %.4f   (c/J = %.6f)\n', R2_quad, c_over_J);
fprintf('Mixed model R^2:       %.4f   (a = %.6f, b = %.6f)\n', R2_mixed, a,
b);
% Determine best model
[~, idx] = max([R2_visc, R2_quad, R2_mixed]);
models = ["Viscous", "Quadratic", "Mixed"];
fprintf('\n>>> BEST MODEL: %s <<<\n\n', models(idx));
y_target = 5.722;
x_target = (y_target - p_visc(2)) / p_visc(1);

%% ================================
%  PLOTS
% =================================
figure;
subplot(3,1,1)
scatter(t, ln_omega, 'filled'); hold on
plot(t, ln_fit, 'r', 'LineWidth', 1.5)
title('Viscous Test: ln(\omega) vs t')
ylabel('ln(\omega)')
subplot(3,1,2)
scatter(t, inv_omega, 'filled'); hold on
plot(t, inv_fit, 'r', 'LineWidth', 1.5)
title('Quadratic Test: 1/\omega vs t')
ylabel('1/\omega')
subplot(3,1,3)
scatter(omega, domega, 'filled'); hold on
plot(omega, domega_fit, 'r', 'LineWidth', 1.5)
title('Mixed Model: ω̇ vs ω')
xlabel('\omega')
ylabel('ω̇')
```