Mountain Bike Rear Suspension Damping Analysis

Authors:

Dean Rogers
Department of Mechanical Engineering,
University of North Dakota Grand Forks,
North Dakota 58202
Email: dean.rogers@und.edu

Prepared in partial fulfillment of the requirements of ME426

Instructor:

Assistant Professor Cai Xia Yang
Department of Mechanical Engineering,
University of North Dakota Grand Forks,
North Dakota 58202
Email caixia.yang@und.edu

June 28th, 2020

Abstract

A mountain bike's suspension setup is one of the most crucial parts of being able to ride efficiently, confidently, and safely. One of the components of mountain bike suspension that can be somewhat daunting to a beginner is the damping of the suspension. In this report, an analysis is performed on the damping rate of the rear suspension of a mountain bike in two common scenarios; small repeated bumps, and single large drop-offs. Since both require different settings for optimal performance, a damping coefficient was chosen to satisfy both as best as possible. The analysis showed that a higher damping was better for small bumps while a lower damping provided a better response to a drop-off, with the best value being 3,000 N-m/s. This value provided sufficient absorption of the drop-off impact with little 'kickback', and also absorbed the small bumps well with minimal 'kickback' between the bumps.

Introduction

In the world of mountain biking, especially when it comes to racing, suspension is integral in maintaining traction and smoothing out the trail, allowing the rider to go faster and push the bike further. For front suspension, there is largely only one type which consists of basic spring and damper systems, while the rear suspension uses a mechanical linkage which varies greatly between brands and bikes, each with their own advantages and disadvantages. While there are many linkages that can be used for the rear suspension, and much design analysis is done on the different linkages, the actual design of the shock (or spring and damper system) for the rear suspension is not often covered quantitatively, but rather making recommendations on 'feel'. While using 'feel' may work fairly well for experienced riders, many beginner to intermediate riders don't have enough riding consistency to feel the subtle differences between options, so performing a quantitative analysis to provide a recommended suspension setup would be beneficial.

Mountain bike suspension setup can be a daunting task for a beginner, especially the rear suspension. Since there are a multitude of different combinations of bikes, linkages, riders, riding conditions, and shocks, this analysis will be performed based on my own situation but could be adapted to fit other situations. The desired result of this analysis is that a damping coefficient is determined such that small repeated bumps will be absorbed by the suspension at moderate speed,

while larger drop-offs and jumps will absorb most of the impact without causing the bike to 'bounce' back.

Methods

Since there is a mechanical linkage that connects the rear wheel and the rear shock, the effective spring rate and damping rate at the wheel is determined by the spring rate and damping rate of the shock multiplied by the leverage ratio of the linkage. The leverage ratio was determined using a geometric model of the linkage and deriving the equations to relate the input from the wheel to the output at the shock. It is assumed that the frame of the bike will stay level with the ground at all times (essentially both wheels will travel the same amount at the same time) so the input at the wheel will only be in the vertical 'y' direction, while the shock displacement will be the overall change in length of the shock.

The bike being used in the analysis is the 2018 Giant Reign SX size large as shown in figure 1. The model of the rear suspension linkage is shown in figure 2 with no wheel displacement, or 'travel'.

Figure 1: 2018 Giant Reign SX - Size L

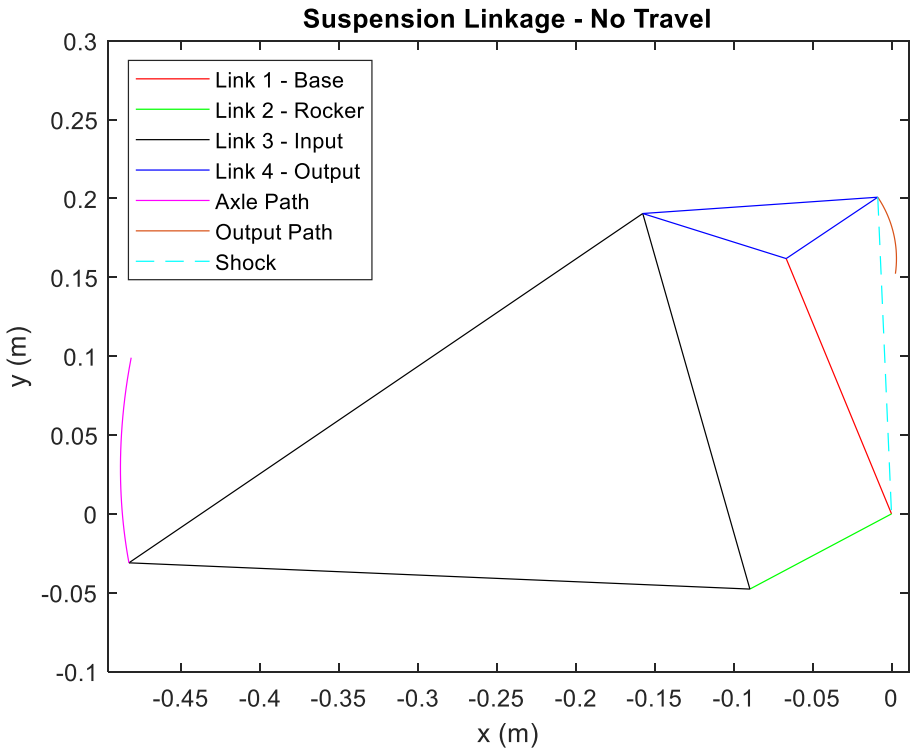

Figure 2: Suspension Linkage - No Travel

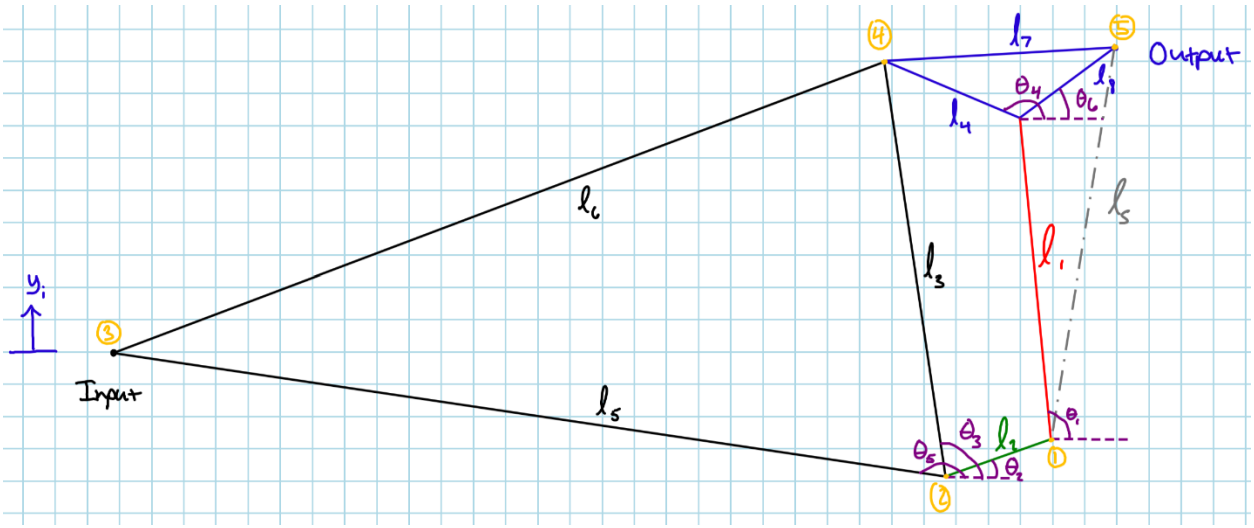

Figure 3: Linkage with Labeled Dimensions and Angles

Starting from the output, we obtain

$$\Delta l_s = l_s - \sqrt{(x_5 - x_6)^2 + (y_5 - y_6)^2} \tag{1}$$

Where

$$x_5 = l_1 \cos(\theta_1) + l_8 \cos(\theta_6) \tag{2a}$$

$$y_5 = l_1 \sin(\theta_1) + l_8 \sin(\theta_6) \tag{2b}$$

θ_1 is fixed, and using the cosine law

$$\theta_6 = \theta_4 - \arccos\left(\frac{l_8^2 + l_4^2 - l_7^2}{2 l_8 l_4}\right) \tag{3}$$

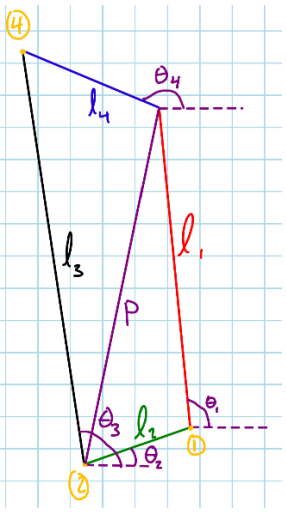

Figure 4: Close-up of four-bar linkage

By specifying a diagonal dimension, p, across the rectangle of the four-bar linkage and using the cosine law,

$$p = \sqrt{l_1^2 + l_2^2 + 2 l_1 l_2 \cos(\theta_2 - \theta_1)} \tag{4}$$

$$\theta_4 = \theta_3 + \arccos\left(\frac{l_3^2 + l_4^2 - p^2}{2 l_3 l_4}\right) \tag{5}$$

where

$$\theta_3 = \theta_2 + \phi_2 + \phi_3 \tag{6}$$

Using the cosine law,

$$\phi_2 = \arccos\left(\frac{p^2 + l_2^2 - l_1^2}{2pl_2}\right) \tag{7}$$

$$\phi_3 = \arccos\left(\frac{p^2 + l_3^2 - l_4^2}{2pl_3}\right) \tag{8}$$

The input can be defined by

$$y_i = l_5 \cos(\theta_5) - l_2 \cos(\theta_2) \tag{9}$$

Where

$$\theta_5 = \theta_3 + \arccos\left(\frac{l_3^2 + l_5^2 - l_6^2}{2l_3 l_5}\right) \tag{10}$$

The output Δl_s cannot be solved explicitly for the input y_i, however we can consider the angle θ_2 as the 'input' and use separate equations to compare Δl_s to y_i while varying the value of θ_2. Since the equation would be so long, it is not practical to combine equations 1 – 9 and the results are instead calculated using the separate equations in the correct order and are combined in the process.

This process was done in Matlab so that the equations could be easily solved over a range of values of θ_2 that correspond with the actual values that the bike is capable of, and the values of the Δl_s and y_i were recorded and compared to show the leverage ratio. In order to find the value of the leverage ratio over the movement of the linkage, a zero-intercept linear fit line was used, where the slope of that line is the leverage ratio.

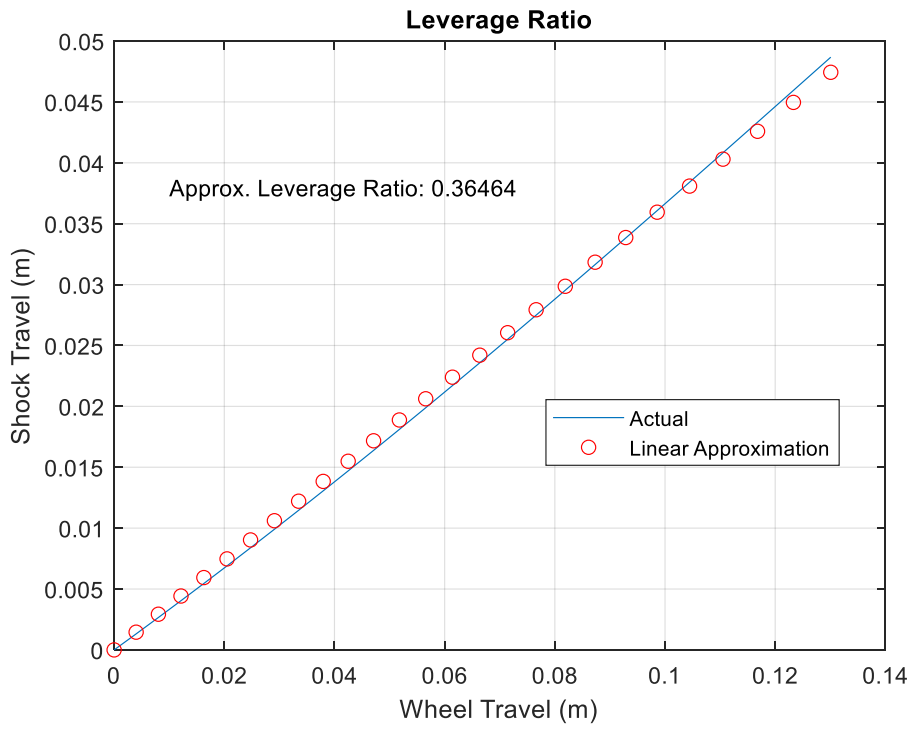

Figure 5: Leverage Ratio Estimation

Next, the base excitation function for the small repeated bumps was determined using Fourier analysis. The desired bump is a 7cm bump that occurs every second when riding at 33 m/s. Equation 11 shows uses the bump height, h_b, wheel radius, r, and the speed, v, to determine the amount of time the wheel will be in contact with the bump, t_b.

$$t_b = \frac{2\sqrt{r^2 - (r - h_b)^2}}{s} \tag{11}$$

This is then used to determine the shape of the bump in equation 12, where $y(t)$ is the axle displacement in the y direction, and y_{sag} is the displacement in the y direction due to the weight of the rider, commonly referred to as 'sag'. Since the sag in my circumstance is 30% of the shock travel, the values of bump time and sag displacement were calculated to be 0.013s and 0.143m respectively. The remainder of the one second before the next bump is a constant value of the sag displacement, so

$$Y(t) = \begin{vmatrix} y_{sag} + h_b \sin\left(\frac{\pi t}{t_b}\right) & [n\tau, n\tau + t_b] \\ y_{sag} & (n\tau + t_b, (n+1)\tau) \end{vmatrix} \tag{12}$$

Where $n = 0,1,2 \ldots$

One cycle of this function is shown in figure 6.

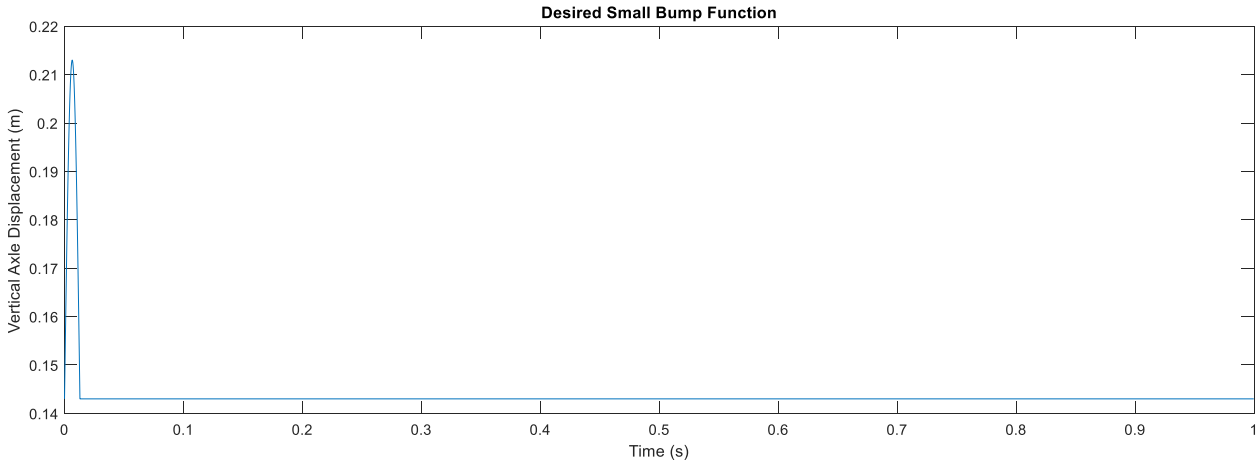

Figure 6: Single Cycle - Desired Small Bump Function

A function was created in Matlab that used the Fourier analysis equations 13 – 16 to model the discontinuous function $Y(t)$ as a continuous function $Y_F(t)$ made up of a sum of sine and cosine functions.

$$Y_F(t) = \frac{a_0}{2} + \sum_{j=1}^{\infty} a_j \cos(j\omega t) + \sum_{j=1}^{\infty} b_j \sin(j\omega t) \tag{13}$$

$$a_0 = \frac{2}{N}\sum_{i=1}^{N} Y_i \tag{14}$$

$$a_j = \frac{2}{N}\sum_{i=1}^{\infty} Y_i \cos\left(\frac{2j\pi t_i}{\tau}\right) \tag{15}$$

$$b_j = \frac{2}{N}\sum_{i=1}^{\infty} Y_i \sin\left(\frac{2j\pi t_i}{\tau}\right) \tag{16}$$

Where j is an arbitrary integer, N is the number of equal-time sections that the desired function, $Y(t)$ is broken into, and y_i is the value of the $Y(t)$ function at time t_i where $t_i N = \tau$.

In the analysis, the value j was chosen to be 10 so that the function $Y_F(t)$ was relatively accurate without being too computationally intense.

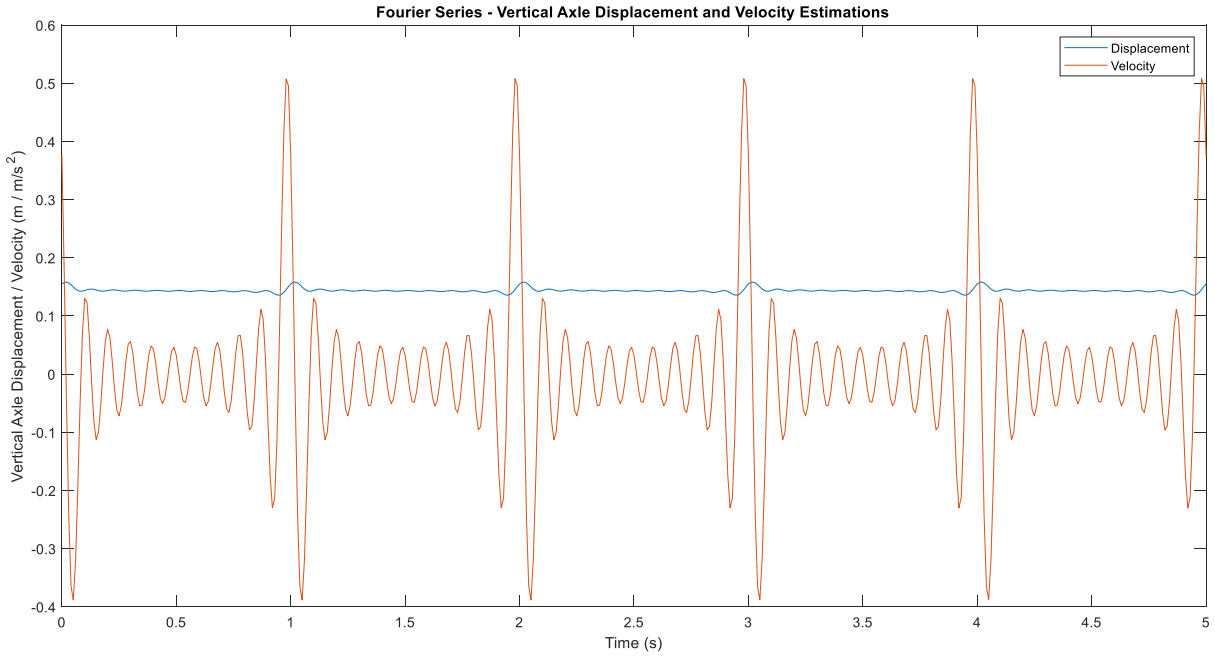

Figure 7: Small Bump Function - Fourier Analysis

The initial conditions for the small bump scenario were set assuming the bike was rolling on flat ground, such that $y(0) = y_{sag}$ and $\dot{y}(0) = 0$.

For the drop-off scenario, the base excitation function is just the constant sag value, so

$$Y(t) = y_{sag} \tag{17}$$

To take account of the drop-off, the $\dot{y}(0)$ initial condition was set as the vertical velocity of the rider and bike falling off a drop with a height of one meter, while the initial condition $y(0)$ was set to zero since the suspension would fully extend while in the air.

$$\dot{y}(0) = \sqrt{v_i^2 + 2gh} \tag{18}$$

The equation of motion for the rear suspension is shown in equation 19, which uses the base excitation of a damped system model where m is the mass, c is the damping coefficient, k is the spring constant, y is the equation of the axle displacement, and Y is the equation of base.

$$m\ddot{y} + c\dot{y} + ky = c\dot{Y} + kY \tag{19}$$

Where the effective spring and damping coefficients are calculated using the spring and damping coefficients of the shock, k_s and c_s respectively, and the leverage ratio, R.

$$c = c_s R \tag{20}$$

$$k = k_s R \tag{21}$$

For Matlab to solve higher order ODE's, they must be broken down into a system of first-degree ODE's

$$y_1 = \dot{y} \tag{22}$$

$$y_2 = \dot{y_1} = \ddot{y} = \frac{1}{m}\left(-c\dot{y} - ky + c\dot{Y} + kY\right) \tag{23}$$

To find an acceptable shock damping coefficient, different values of c_s were tried and compared to see the advantages and disadvantages of each value.

Results

After some trial and error in choosing the damping coefficient, four values were chosen that highlight how the system response changes with the damping coefficient; 200 N-m/s, 3000 N-m/s, 20,000 N-m/s, and 500,000 N/m-s. Starting with the small bump response, it can be seen that at 200 N-m/s the suspension absorbs the most compared to the other three values, especially on the initial bump. The severe downside of the low damping coefficient is the oscillation between bumps, which can cause the wheel to 'bounce' or 'skip' along the ground which significantly reduces traction. Additionally, this 'skipping' interferes with successive bumps if timed wrong, reducing the amount of the bump that is absorbed, as can be seen at seconds 1 and 3. On the other extreme, at 500,000 N-m/s all of the bumps are not properly absorbed and the wheel sits too high

in its travel (a low displacement) which indicates that the bump would have been felt significantly and the rider's balance may be disturbed with the back end of the bike being too high. The responses of 3,000 N-m/s and 20,000 N-m/s are relatively similar, except that the 20,000 N-m/s value absorbs more of the bump and has only small oscillations between bumps, while the 3,000 N-m/s still has one moderate oscillation between the bumps that could cause a bit of 'skipping'.

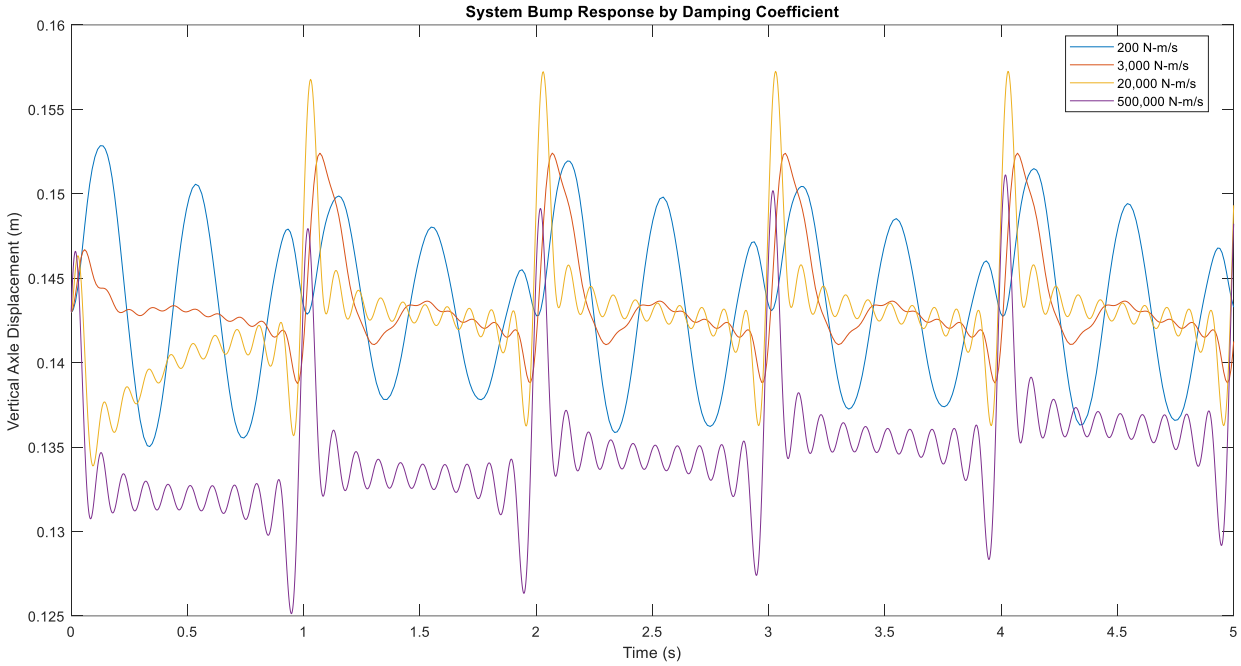

Figure 8: Small Bump Response

With the drop-off response, it is clear that at 200 N-m/s, the suspension would oscillate far too much and may even 'kick' the rider off the bike by forcing the wheel down too fast after the initial compression. At both 20,000 N-m/s and 500,000 N-m/s, the force from the landing is not absorbed at all, the wheel just slowly returns to the sag displacement, which means the rider would have to absorb the entirety of the impact, which could be dangerous. At 3000 N-m/s, the bump is absorbed and there is a slight amount of 'kick', but it quickly reaches the steady-state sag displacement.

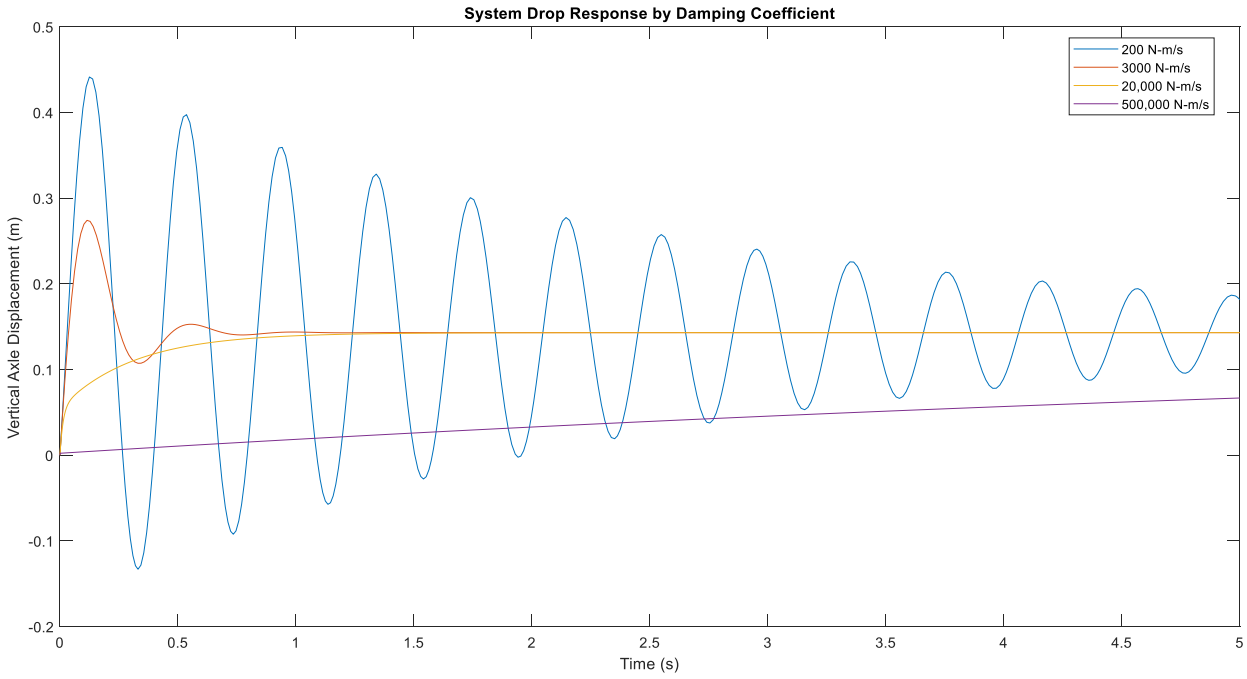

Figure 9: Drop-Off Response

Discussion

As can be expected, there is not one 'perfect' solution that satisfies both scenarios, small bumps, and large drop-offs. The small bumps require higher damping that allows the wheel to absorb the bump and return to the sag displacement at a speed that doesn't 'kick' the rider, while a large drop needs lower damping to absorb a sudden, large impact. While a damping coefficient of 20,000 N-m/s seems to be the best for small bumps, it does not work for the drop-off. The best option for the drop-off is 3,000 N-m/s which is still good for small bumps, so it seems that is the best option to satisfy both requirements.

Conclusion

While this is a good theoretical analysis, there are many more factors that affect the performance of the suspension in reality, one of the largest being the ability of the rider to manipulate the bike and change their position on the bike to ride more efficiently and maintain more traction. Since each rider is different, it isn't possible to include this in an analysis like this,

however, it would be very interesting to use physical data collection to analyze the displacement of each of the wheels as well as the frame of the bike using accelerometers and linear encoders to optimize the suspension setup.

Another factor to take into account is the use of different damping coefficients for rebound and compression, as well as for high- and low-speed compression, on more advanced shocks. These features allow for finer tuning of the response to all for more bump absorption with less ‘kickback’.

Appendix

```
clc
clear all
close all

% Set up linkage
l = [0.1751
    0.1016
    0.24765
    0.09525
    0.3937
    0.3937
    0.149225
    0.06985];                       % Linkage lengths (m)
theta1 = 112.4;                     % degrees
a_lims = [28,2];                    % theta 2 limits (degrees)
theta2 = a_lims(1):-1:a_lims(2);    % theta 2 range (degrees)
shock = [0 0];                      % Shock base coordinates (m)

link1 = zeros(((a_lims(1)-a_lims(2))/1)+1,4);
link2 = zeros(((a_lims(1)-a_lims(2))/1)+1,4);
link3 = zeros(((a_lims(1)-a_lims(2))/1)+1,6);
link4 = zeros(((a_lims(1)-a_lims(2))/1)+1,6);

link1(:,3) = [l(1)*cosd(theta1)];
link1(:,4) = [l(1)*sind(theta1)];

link2(:,3:4) = [l(2)*cosd(theta2+180); l(2)*sind(theta2+180)]';

p = sqrt(l(1)^2 + l(2)^2 - 2*l(1)*l(2)*cosd(180+theta2-theta1));
phi2 = acosd((p.^2 + l(2)^2 - l(1)^2)./(2*p*l(2)));
phi3 = acosd((p.^2 + l(3)^2 - l(4)^2)./(2*p*l(3)));
theta3 = theta2 + phi2 + phi3;
theta5 = theta3 + acosd((l(3)^2 + l(5)^2 - l(6)^2)/(2*l(3)*l(5)));

link3(:,1:2) = [l(2)*cosd(theta2+180); l(2)*sind(theta2+180)]';
link3(:,3:4) = [l(2)*cosd(theta2+180)+l(3)*cosd(theta3); l(2)*sind(theta2+180)+l(3)*sind↙
(theta3)]';
link3(:,5:6) = [l(2)*cosd(theta2+180)+l(5)*cosd(theta5); l(2)*sind(theta2+180)+l(5)*sind↙
(theta5)]';


theta4 = theta3 + acosd((l(3)^2 + l(4)^2 - p.^2)/(2*l(3)*l(4)));
theta6 = theta4 - acosd((l(8)^2 + l(4)^2 - l(7)^2)/(2*l(8)*l(4)));

link4(:,1) = [l(1)*cosd(theta1)];
link4(:,2) = [l(1)*sind(theta1)];
link4(:,3:4) = [l(2)*cosd(theta2+180)+l(3)*cosd(theta3); l(2)*sind(theta2+180)+l(3)*sind↙
(theta3)]';
link4(:,5:6) = [l(1)*cosd(theta1)+l(8)*cosd(theta6); l(1)*sind(theta1)+l(8)*sind↙
(theta6)]';
```

```
% Animate the linkage movement
figure(1)
for i = 1:((a_lims(1)-a_lims(2))/1)+1
    p1 = plot([link1(i,1) link1(i,3)],[link1(i,2) link1(i,4)],"r-");
    hold on
    p2 = plot([link2(i,1) link2(i,3)],[link2(i,2) link2(i,4)],"g-");

    p3 = plot([link3(i,1) link3(i,3)],[link3(i,2) link3(i,4)],"k-");
    p4 = plot([link3(i,1) link3(i,5)],[link3(i,2) link3(i,6)],"k-");
    p5 = plot([link3(i,5) link3(i,3)],[link3(i,6) link3(i,4)],"k-");

    p6 = plot([link4(i,1) link4(i,3)],[link4(i,2) link4(i,4)],"b-");
    p7 = plot([link4(i,1) link4(i,5)],[link4(i,2) link4(i,6)],"b-");
    p8 = plot([link4(i,5) link4(i,3)],[link4(i,6) link4(i,4)],"b-");

    p9 = plot(link3(:,5),link3(:,6),"m-");
    p10 = plot(link4(:,5),link4(:,6),"Color",[0.8500, 0.3250, 0.0980]);

    p11 = plot([link4(i,5) shock(1)],[link4(i,6) shock(2)],'--c');

    xlim([-0.06 0.15])
    ylim([-0.1 0.3])
    axis equal
    xlabel('x (m)')
    ylabel('y (m)')
    hold off

    pause(0.00001);
end
legend([p1 p2 p3 p6 p9 p10 p11],'Link 1 - Base','Link 2 - Rocker','Link 3 - Input',...
     'Link 4 - Output','Axle Path','Output Path','Shock','Location','northwest')

%%

% Determine the leverage ratio
yi = link3(2:length(link3),6) - link3(1:length(link3)-1,6);

w_trav_y = link3(2:length(link3),6) - link3(1:1,6);
w_trav = w_trav_y;

x1o = link4(:,5);
y1o = link4(:,6);
delta_o = sqrt((x1o - shock(1)).^2 + (y1o - shock(2)).^2);
s_trav = delta_o(1:1) - delta_o(2:length(delta_o));
delta_o = delta_o(1:length(delta_o)-1) - delta_o(2:length(delta_o));

w_trav = [0;w_trav];
s_trav = [0;s_trav];
lFit = fitlm(w_trav,s_trav,'Intercept',false);
lFit = lFit.Coefficients{1,'Estimate'};
yfit = polyval([lFit 0],w_trav);
```

```
figure(2)
plot(w_trav,s_trav)
hold on
plot(w_trav,yfit,'ro')
hold off
txt = strcat("Approx. Leverage Ratio: ",num2str(lFit));
text(0.01,0.038,txt)
title("Leverage Ratio")
xlabel('Wheel Travel (m)')
ylabel('Shock Travel (m)')
legend('Actual','Linear Approximation','Location',"Best")
grid
%%

% Generate the small bump function
bump = 0.07;        % bump height (meters)
speed = 33;         % travelling speed (m/s)
r = 0.36195;        % Outer wheel radius (meters)
bumpTime = (2*sqrt(r^2 - (r - bump)^2))/speed
t1 = 0:bumpTime/50:bumpTime;
baseFunc = bump*sin(pi*t1/bumpTime);
baseFunc(1,length(t1)+1:1/(bumpTime/50)) = 0;
baseFunc = baseFunc + 0.143;
t1 = 0:bumpTime/50:1;
figure(3)
plot(t1(1:length(t1)-1),baseFunc)
xlabel('Time (s)')
ylabel('Vertical Axle Displacement (m)')
title('Desired Small Bump Function')

% Use the Fourier analysis to generate a continuous bump function
baseFourier = fSeries(t1(1:length(t1)-1),baseFunc,10);
vpa(baseFourier)
d_baseFourier = diff(baseFourier);
vpa(d_baseFourier)
t=0:0.01:5;
y1 = subs(baseFourier);
y2 = subs(d_baseFourier);
figure(4)
plot(t,y1)
hold on
plot(t,y2)
hold off
xlabel('Time (s)')
ylabel('Vertical Axle Displacement / Velocity (m / m/s^2)')
legend('Displacement', 'Velocity')
title('Fourier Series - Vertical Axle Displacement and Velocity Estimations')

%%
```

```
% Solve the system response to small bumps
tspan = [0 5];      % Time span (s)
x1_0 = 0.143;       % x(0) (m)
x2_0 = 0;           % dx(0) (m/s)
[T,X] = ode45(@odes_bump,tspan,[x1_0 x2_0]);    % c = 200 N-m/s
[T1,X1] = ode45(@odes1_bump,tspan,[x1_0 x2_0]); % c = 3,000 N-m/s
[T2,X2] = ode45(@odes2_bump,tspan,[x1_0 x2_0]); % c = 20,000 N-m/s
[T3,X3] = ode45(@odes3_bump,tspan,[x1_0 x2_0]); % c = 500,000 N-m/s
figure(5)
plot(T,X(:,1))
hold on
plot(T1,X1(:,1))
plot(T2,X2(:,1))
plot(T3,X3(:,1))
hold off
legend('200 N-m/s','3,000 N-m/s','20,000 N-m/s','500,000 N-m/s','Location',"Best")
xlabel('Time (s)')
ylabel('Vertical Axle Displacement (m)')
title('System Bump Response by Damping Coefficient')


%%

% Solve the system response to a large drop-off
tspan = [0 5];
x1_0 = 0;
x2_0 = sqrt(2*9.81*1);
[T,X] = ode45(@odes_drop,tspan,[x1_0 x2_0]);    % c = 200 N-m/s
[T1,X1] = ode45(@odes1_drop,tspan,[x1_0 x2_0]); % c = 3,000 N-m/s
[T2,X2] = ode45(@odes2_drop,tspan,[x1_0 x2_0]); % c = 20,000 N-m/s
[T3,X3] = ode45(@odes3_drop,tspan,[x1_0 x2_0]); % c = 500,000 N-m/s
figure(6)
plot(T,X(:,1))
hold on
plot(T1,X1(:,1))
plot(T2,X2(:,1))
plot(T3,X3(:,1))
hold off
legend('200 N-m/s','3000 N-m/s','20,000 N-m/s','500,000 N-m/s','Location',"Best")
xlabel('Time (s)')
ylabel('Vertical Axle Displacement (m)')
title('System Drop Response by Damping Coefficient')


%%

% Functions

function dxdt = odes_bump(t,x)
    dxdt = zeros(2,1);
```

```
    ks = 61294;     % Shock spring rate (N/m)

    cs = 200;       % Damping coefficient (N-m/s)
    m = 92;         % Bike + Rider mass (kg)
    levRatio = 0.3646;

    k = ks*levRatio;
    c = cs*levRatio;

    % Base excitation function
    f = 0.001145073619136069377130726643088l*cos(18.850086945100215984894020948559*t) +↙
...
        0.001114650488709262093553475914120 6*cos(37.700173890200431969788041897118*t) +↙
...
        0.001100153086173946649292676447373 7*cos(43.983536205233832561134477145 97*t) +↙
...
        0.001144711049545555458720066862809 97*sin(18.850086945100215984894020948559*t) +↙
...
        0.001113925358192237790797007867093 3*sin(37.700173890200431969788041897118*t) +↙
...
        0.001154169559296512343610219275547*cos(6.2833623150334050322385337494779*t) +↙
...
        0.001150754188331065729425373156402 7*cos(12.566724630066810064477067498956*t) +↙
...
        0.001137145378723375573060128118640 9*cos(25.133449260133620128954134997912*t) +↙
...
        0.001083546244697229611261835025004 5*cos(50.266898520267240257908269995824*t) +↙
...
        0.001099307105437543783874043690218 5*sin(43.983536205233832561134477145 97*t) +↙
...
        0.001064880899881493359765416428786*cos(56.550260835300640849254705244675*t) +↙
...
        0.001154048702338486597288036428210 5*sin(6.2833623150334050322385337494779*t) +↙
...
        0.001150512474735905120090673214861*sin(12.566724630066810064477067498956*t) +↙
...
        0.001136661954100175566176322838885*sin(25.133449260133620128954134997912*t) +↙
...
        0.001082579415987635577250935980941 9*sin(50.266898520267240257908269995824*t) +↙
...
        0.001126993911895809909878840571195 7*cos(31.416811575167024273014249047264*t) +↙
...
        0.001044214190390645646475120678076 1*cos(62.833623150334048546028498094529*t) +↙
...
        0.001063793225765763449455758582473 7*sin(56.550260835300640849254705244675*t) +↙
...
        0.001126389633523479328822136658061 5*sin(31.416811575167024273014249047264*t) +↙
...
        0.001043005673756694159926006193472 9*sin(62.833623150334048546028498094529*t) +↙
...
        0.14357765459458612933296706160036;
```

```
    % Base excitation function derivative
    df = 0.021577902810950625052225463965936*cos(18.850086945100215984894020948559*t) +...
...
        0.041995179704551167015015958672258*cos(37.700173890200431969788041897118*t) +...
...
        0.048351413872683013180234485379094*cos(43.983536205233832561134477145597*t) - ...
        0.021584737279255478224984074078911*sin(18.850086945100215984894020948559*t) - ...
        0.042022517251136074173628690931966*sin(37.700173890200431969788041897118*t) +...
...
        0.007251306125986850093638684477027 9*cos(6.2833623150334050322385337494779*t) +...
...
        0.014458173453462817430605347235486*cos(12.566724630066810064477067498956*t) +...
...
        0.028568235549301092466468506819671*cos(25.133449260133620128954134997912*t) +...
...
        0.054417909643580651938749139571148*cos(50.266898520267240257908269995824*t) -...
...
        0.048388623097031518990682925248 05*sin(43.983536205233832561134477145597*t) + ...
        0.060157784391879785378458108144106*cos(56.550260835300640849254705244675*t) -...
...
        0.007252065514042418642109817129697 8*sin(6.2833623150334050322385337494779*t) -...
...
        0.014461211001652544257638406568617*sin(12.566724630066810064477067498956*t) -...
...
        0.028580385677539389053574769123179*sin(25.133449260133620128954134997912*t) -...
...
        0.054466509124212296172326135422532*sin(50.266898520267240257908269995824*t) +...
...
        0.035387570876628587821682579617671*cos(31.416811575167024273014249047264*t) +...
...
        0.065535825448488380163414402761414*cos(62.833623150334048546028498094529*t) -...
...
        0.060219292646828116782150280373465*sin(56.550260835300640849254705244675*t) -...
...
        0.035406555376391046309507147596182*sin(31.416811575167024273014249047264*t) -...
...
        0.065611760927236998067848179001919*sin(62.833623150334048546028498094529*t);

    dxdt(1) = x(2);
    dxdt(2) = (-c*x(2) - k*x(1) + k*f + c*df)/m;
end

function dxdt = odes1_bump(t,x)
    dxdt = zeros(2,1);

    ks = 61294;      % Shock spring rate (N/m)

    cs = 3000;       % Damping coefficient (N-m/s)
    m = 92;          % Bike + Rider mass (kg)
    levRatio = 0.3646;
```

```
    k = ks*levRatio;
    c = cs*levRatio;

    % Base excitation function
    f = 0.001145073619136069377130726643088l*cos(18.850086945100215984894020948559*t) +...
        0.001114650488709262093553475914120 6*cos(37.700173890200431969788041897118*t) +...
        0.001100153086173946649292676447373 7*cos(43.983536205233832561134477145 97*t) +...
        0.001144711049545555458720066862809 97*sin(18.850086945100215984894020948559*t) +...
        0.001113925358192237790797007868093 3*sin(37.700173890200431969788041897118*t) +...
        0.001154169559296512343610219275547*cos(6.2833623150334050322385337494779*t) +...
        0.001150754188331065729425373156402 7*cos(12.566724630066810064477067498956*t) +...
        0.001137145378723375573060128118640 9*cos(25.133449260133620128954134997912*t) +...
        0.001083546244697229611261835025004 5*cos(50.266898520267240257908269995824*t) +...
        0.001099307105437543783874043690218 5*sin(43.983536205233832561134477145 97*t) +...
        0.001064880899881493359765416428786*cos(56.550260835300640849254705244675*t) +...
        0.001154048702338486597288036428210 5*sin(6.2833623150334050322385337494779*t) +...
        0.001150512474735905120090673214861*sin(12.566724630066810064477067498956*t) +...
        0.001136661954100175566176322838885*sin(25.133449260133620128954134997912*t) +...
        0.001082579415987635577250935980941 9*sin(50.266898520267240257908269995824*t) +...
        0.001126993911895809909878840571195 7*cos(31.416811575167024273014249047264*t) +...
        0.001044214190390645646475120678076 1*cos(62.833623150334048546028498094529*t) +...
        0.001063793225765763449455758582473 7*sin(56.550260835300640849254705244675*t) +...
        0.001126389633523479328822136658061 5*sin(31.416811575167024273014249047264*t) +...
        0.001043005673756694159926006193472 9*sin(62.833623150334048546028498094529*t) +...
        0.14357765459458612933296706160036;

    % Base excitation function derivative
    df = 0.021577902810950625052225463965936*cos(18.850086945100215984894020948559*t) +...
        0.041995179704551167015015958672258*cos(37.700173890200431969788041897118*t) +...
```

```
...
        0.048351413872683013180234485379094*cos(43.983536205233832561134477145597*t) - ...
        0.021584737279255478224984074078911*sin(18.850086945100215984894020948559*t) - ...
        0.042022517251136074173628690931966*sin(37.700173890200431969788041897118*t) +↙
...
        0.007251306125986850093638684477027 9*cos(6.2833623150334050322385337494779*t) +↙
...
        0.014458173453462817430605347235486*cos(12.566724630066810064477067498956*t) +↙
...
        0.028568235549301092466468506819671*cos(25.133449260133620128954134997912*t) +↙
...
        0.054417909643580651938749139571148*cos(50.266898520267240257908269995824*t) -↙
...
        0.048388623097031518990682925248 05*sin(43.983536205233832561134477145597*t) + ...
        0.060157784391879785378458108144106*cos(56.550260835300640849254705244675*t) -↙
...
        0.007252065514042418642109817129697 8*sin(6.2833623150334050322385337494779*t) -↙
...
        0.014461211001652544257638406568617*sin(12.566724630066810064477067498956*t) -↙
...
        0.028580385677539389053574769123179*sin(25.133449260133620128954134997912*t) -↙
...
        0.054466509124212296172326135422532*sin(50.266898520267240257908269995824*t) +↙
...
        0.035387570876628587821682579617671*cos(31.416811575167024273014249047264*t) +↙
...
        0.065535825448488380163414402761414*cos(62.833623150334048546028498094529*t) -↙
...
        0.060219292646828116782150280373465*sin(56.550260835300640849254705244675*t) -↙
...
        0.035406555376391046309507147596182*sin(31.416811575167024273014249047264*t) -↙
...
        0.065611760927236998067848179001919*sin(62.833623150334048546028498094529*t);

    dxdt(1) = x(2);
    dxdt(2) = (-c*x(2) - k*x(1) + k*f + c*df)/m;
end

function dxdt = odes2_bump(t,x)
    dxdt = zeros(2,1);

    ks = 61294;      % Shock spring rate (N/m)

    cs = 20000;      % Damping coefficient (N-m/s)
    m = 92;          % Bike + Rider mass (kg)
    levRatio = 0.3646;

    k = ks*levRatio;
    c = cs*levRatio;

    % Base excitation function
```

```
    f = 0.001145073619136069377130726643088l*cos(18.850086945100215984894020948559*t) +↙
...
        0.001114650488709262093553475914120 6*cos(37.700173890200431969788041897118*t) +↙
...
        0.001100153086173946649292676447373 7*cos(43.983536205233832561134477145 97*t) +↙
...
        0.001144711049545555458720066862809 97*sin(18.850086945100215984894020948559*t) +↙
...
        0.001113925358192237790797007868093 3*sin(37.700173890200431969788041897118*t) +↙
...
        0.001154169559296512343610219275547*cos(6.2833623150334050322385337494779*t) +↙
...
        0.001150754188331065729425373156402 7*cos(12.566724630066810064477067498956*t) +↙
...
        0.001137145378723375573060128118640 9*cos(25.133449260133620128954134997912*t) +↙
...
        0.001083546244697229611261835025004 5*cos(50.266898520267240257908269995824*t) +↙
...
        0.001099307105437543783874043690218 5*sin(43.983536205233832561134477145 97*t) +↙
...
        0.001064880899881493359765416428786*cos(56.550260835300640849254705244675*t) +↙
...
        0.001154048702338486597288036428210 5*sin(6.2833623150334050322385337494779*t) +↙
...
        0.001150512474735905120090673214861*sin(12.566724630066810064477067498956*t) +↙
...
        0.001136661954100175566176322838885*sin(25.133449260133620128954134997912*t) +↙
...
        0.001082579415987635577250935980941 9*sin(50.266898520267240257908269995824*t) +↙
...
        0.001126993911895809909878840571195 7*cos(31.416811575167024273014249047264*t) +↙
...
        0.001044214190390645646475120678076 1*cos(62.833623150334048546028498094529*t) +↙
...
        0.001063793225765763449455758582473 7*sin(56.550260835300640849254705244675*t) +↙
...
        0.001126389633523479328822136658061 5*sin(31.416811575167024273014249047264*t) +↙
...
        0.001043005673756694159926006193472 9*sin(62.833623150334048546028498094529*t) +↙
...
        0.143577654594586129332967061600 36;

    % Base excitation function derivative
    df = 0.021577902810950625052225463965936*cos(18.850086945100215984894020948559*t) +↙
...
        0.041995179704551167015015958672258*cos(37.700173890200431969788041897118*t) +↙
...
        0.048351413872683013180234485379094*cos(43.983536205233832561134477145 97*t) - ...
        0.021584737279255478224984074078 91*sin(18.850086945100215984894020948559*t) - ...
        0.042022517251136074173628690931966*sin(37.700173890200431969788041897118*t) +↙
...
```

```
        0.007251306125986850093638684477027 9*cos(6.2833623150334050322385337494779*t) +
...
        0.014458173453462817430605347235486*cos(12.566724630066810064477067498956*t) +
...
        0.028568235549301092466468506819671*cos(25.133449260133620128954134997912*t) +
...
        0.054417909643580651938749139571148*cos(50.266898520267240257908269995824*t) -
...
        0.048388623097031518990682925248 05*sin(43.983536205233832561134477145 97*t) + ...
        0.060157784391879785378458108144106*cos(56.550260835300640849254705244675*t) -
...
        0.0072520655140424186421098171296978*sin(6.2833623150334050322385337494779*t) -
...
        0.014461211001652544257638406568617*sin(12.566724630066810064477067498956*t) -
...
        0.028580385677539389053574769123179*sin(25.133449260133620128954134997912*t) -
...
        0.054466509124212296172326135422532*sin(50.266898520267240257908269995824*t) +
...
        0.035387570876628587821682579617671*cos(31.416811575167024273014249047264*t) +
...
        0.065535825448488380163414402761414*cos(62.833623150334048546028498094529*t) -
...
        0.060219292646828116782150280373465*sin(56.550260835300640849254705244675*t) -
...
        0.035406555376391046309507147596182*sin(31.416811575167024273014249047264*t) -
...
        0.065611760927236998067848179001919*sin(62.833623150334048546028498094529*t);

    dxdt(1) = x(2);
    dxdt(2) = (-c*x(2) - k*x(1) + k*f + c*df)/m;
end

function dxdt = odes3_bump(t,x)
    dxdt = zeros(2,1);

    ks = 61294;      % Shock spring rate (N/m)

    cs = 500000;     % Damping coefficient (N-m/s)
    m = 92;          % Bike + Rider mass (kg)
    levRatio = 0.3646;

    k = ks*levRatio;
    c = cs*levRatio;

    % Base excitation function
    f = 0.0011450736191360693771307266430881*cos(18.850086945100215984894020948559*t) +
...
        0.0011146504887092620935534759141206*cos(37.700173890200431969788041897118*t) +
...
        0.0011001530861739466492926764473737*cos(43.983536205233832561134477145 97*t) +
```

```
...
    0.0011447110495455554587200668628099 7*sin(18.850086945100215984894020948559*t) +↙
...
    0.0011139253581922377907970078680933*sin(37.700173890200431969788041897118*t) +↙
...
    0.0011541695592965123436102192755 47*cos(6.2833623150334050322385337494779*t) +↙
...
    0.0011507541883310657294253731564027*cos(12.566724630066810064477067498956*t) +↙
...
    0.0011371453787233755730601281186409*cos(25.133449260133620128954134997912*t) +↙
...
    0.0010835462446972296112618350250045*cos(50.266898520267240257908269995824*t) +↙
...
    0.0010993071054375437837840436902185*sin(43.983536205233832561134477145 97*t) +↙
...
    0.0010648808998814933597654164287 86*cos(56.550260835300640849254705244675*t) +↙
...
    0.0011540487023384865972880364282105*sin(6.2833623150334050322385337494779*t) +↙
...
    0.0011505124747359051200906732148 61*sin(12.566724630066810064477067498956*t) +↙
...
    0.0011366619541001755661763228388 85*sin(25.133449260133620128954134997912*t) +↙
...
    0.0010825794159876355772509359809419*sin(50.266898520267240257908269995824*t) +↙
...
    0.0011269939118958099098788405711957*cos(31.416811575167024273014249047264*t) +↙
...
    0.0010442141903906456464751206780761*cos(62.833623150334048546028498094529*t) +↙
...
    0.0010637932257657634494557585824737*sin(56.550260835300640849254705244675*t) +↙
...
    0.0011263896335234793288221366580615*sin(31.416811575167024273014249047264*t) +↙
...
    0.0010430056737566941599260061934729*sin(62.833623150334048546028498094529*t) +↙
...
    0.1435776545945861293329670616 0036;

    % Base excitation function derivative
    df = 0.021577902810950625052225463965936*cos(18.850086945100215984894020948559*t) +↙
...
    0.0419951797045511670150159586722 58*cos(37.700173890200431969788041897118*t) +↙
...
    0.0483514138726830131802344853790 94*cos(43.983536205233832561134477145 97*t) - ...
    0.0215847372792554782249840740789 1*sin(18.850086945100215984894020948559*t) - ...
    0.0420225172511360741736286909319 66*sin(37.700173890200431969788041897118*t) +↙
...
    0.0072513061259868500936386844770279*cos(6.2833623150334050322385337494779*t) +↙
...
    0.0144581734534628174306053472354 86*cos(12.566724630066810064477067498956*t) +↙
...
    0.0285682355493010924664685068196 71*cos(25.133449260133620128954134997912*t) +↙
```

```
...
        0.054417909643580651938749139571148*cos(50.266898520267240257908269995824*t) -↙
...
        0.048388623097031518990682925248 05*sin(43.983536205233832561134477145 97*t) + ...
        0.060157784391879785378458108144106*cos(56.550260835300640849254705244675*t) -↙
...
        0.007252065514042418642109817129 6978*sin(6.283362315033405032238533749 4779*t) -↙
...
        0.014461211001652544257638406568617*sin(12.566724630066810064477067498956*t) -↙
...
        0.028580385677539389053574769123179*sin(25.133449260133620128954134997912*t) -↙
...
        0.054466509124212296172326135422532*sin(50.266898520267240257908269995824*t) +↙
...
        0.035387570876628587821682579617671*cos(31.416811575167024273014249047264*t) +↙
...
        0.065535825448488380163414402761414*cos(62.833623150334048546028498094529*t) -↙
...
        0.060219292646828116782150280373465*sin(56.550260835300640849254705244675*t) -↙
...
        0.035406555376391046309507147596182*sin(31.416811575167024273014249047264*t) -↙
...
        0.065611760927236998067848179001919*sin(62.833623150334048546028498094529*t);

    dxdt(1) = x(2);
    dxdt(2) = (-c*x(2) - k*x(1) + k*f + c*df)/m;
end

function dxdt = odes_drop(t,x)
    dxdt = zeros(2,1);

    ks = 61294;     % Shock spring rate (N/m)

    cs = 200;       % Damping coefficient (N-m/s)
    m = 92;         % Bike + Rider mass (kg)
    levRatio = 0.3646;

    k = ks*levRatio;
    c = cs*levRatio;

    f = 0.143;      % Base excitation function
    df = 0;         % Base excitation function derivative

    dxdt(1) = x(2);
    dxdt(2) = (-c*x(2) - k*x(1) + k*f + c*df)/m;
end

function dxdt = odes1_drop(t,x)
    dxdt = zeros(2,1);

    ks = 61294;     % Shock spring rate (N/m)
```

```
    cs = 3000;      % Damping coefficient (N-m/s)
    m = 92;         % Bike + Rider mass (kg)
    levRatio = 0.3646;

    k = ks*levRatio;
    c = cs*levRatio;

    f = 0.143;      % Base excitation function
    df = 0;         % Base excitation function derivative

    dxdt(1) = x(2);
    dxdt(2) = (-c*x(2) - k*x(1) + k*f + c*df)/m;
end

function dxdt = odes2_drop(t,x)
    dxdt = zeros(2,1);

    ks = 61294;     % Shock spring rate (N/m)

    cs = 20000;     % Damping coefficient (N-m/s)
    m = 92;         % Bike + Rider mass (kg)
    levRatio = 0.3646;

    k = ks*levRatio;
    c = cs*levRatio;

    f = 0.143;      % Base excitation function
    df = 0;         % Base excitation function derivative

    dxdt(1) = x(2);
    dxdt(2) = (-c*x(2) - k*x(1) + k*f + c*df)/m;
end

function dxdt = odes3_drop(t,x)
    dxdt = zeros(2,1);

    ks = 61294;     % Shock spring rate (N/m)
    cs = 500000;    % Damping coefficient (N-m/s)
    m = 92;         % Bike + Rider mass (kg)
    levRatio = 0.3646;

    k = ks*levRatio;
    c = cs*levRatio;

    f = 0.143;      % Base excitation function
    df = 0;         % Base excitation function derivative

    dxdt(1) = x(2);
    dxdt(2) = (-c*x(2) - k*x(1) + k*f + c*df)/m;
end
```

```
function fourier = fSeries(x,y,n)
    syms t
    a0 = 2*mean(y);
    a = zeros(n,1);
    b = zeros(n,1);
    tau = length(x)*(x(2)-x(1));

    fourier = a0/2;

    for j = 1:1:n
        for i = 1:1:length(x)
            a(j) = a(j) + (2/length(x))*y(i)*cos((2*j*pi*x(i))/tau);
            b(j) = a(j) + (2/length(x))*y(i)*sin((2*j*pi*x(i))/tau);
        end
        fourier = fourier + a(j)*cos(j*2*pi*(1/tau)*t) + b(j)*sin(j*2*pi*(1/tau)*t);
    end

end
```