**Top Level Design Summary**

The Formula SAE Suspension Project focuses on the design, analysis, and optimization of a high-performance suspension system for a single-seat Formula-style race car. The system is intended to maximize tire contact, vehicle stability, and handling precision under dynamic racing conditions. The project's objective is to create lightweight, tunable, and manufacturable suspension and steering subsystems that integrate effectively with the vehicle chassis and powertrain assemblies while meeting FSAE competition requirements. Through copious research and reference from past submissions, we have compiled different options for each component of our subsystem. These options were compared against each other to find the best possible configuration. The current state of our design includes a steering assembly employing a 33% Ackermann percentage with a double wishbone suspension geometry. This includes control arms and pushrod, rocker arm, and coilover shock assembly. All of these will attach to our suspension knuckles to interface with the wheels along with our braking system which will include independent front and rear systems.

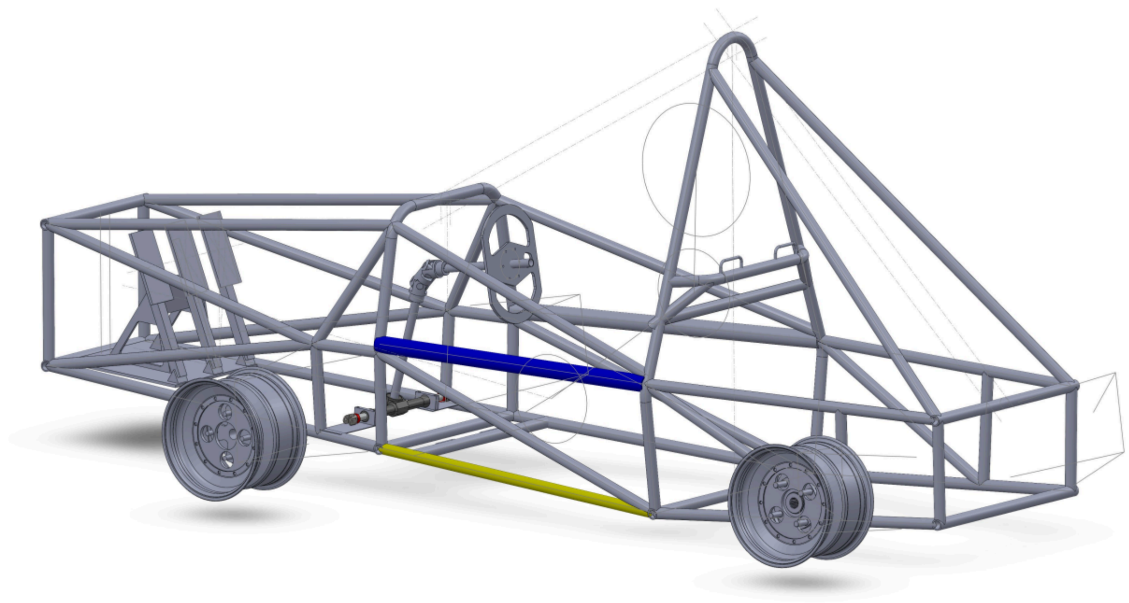
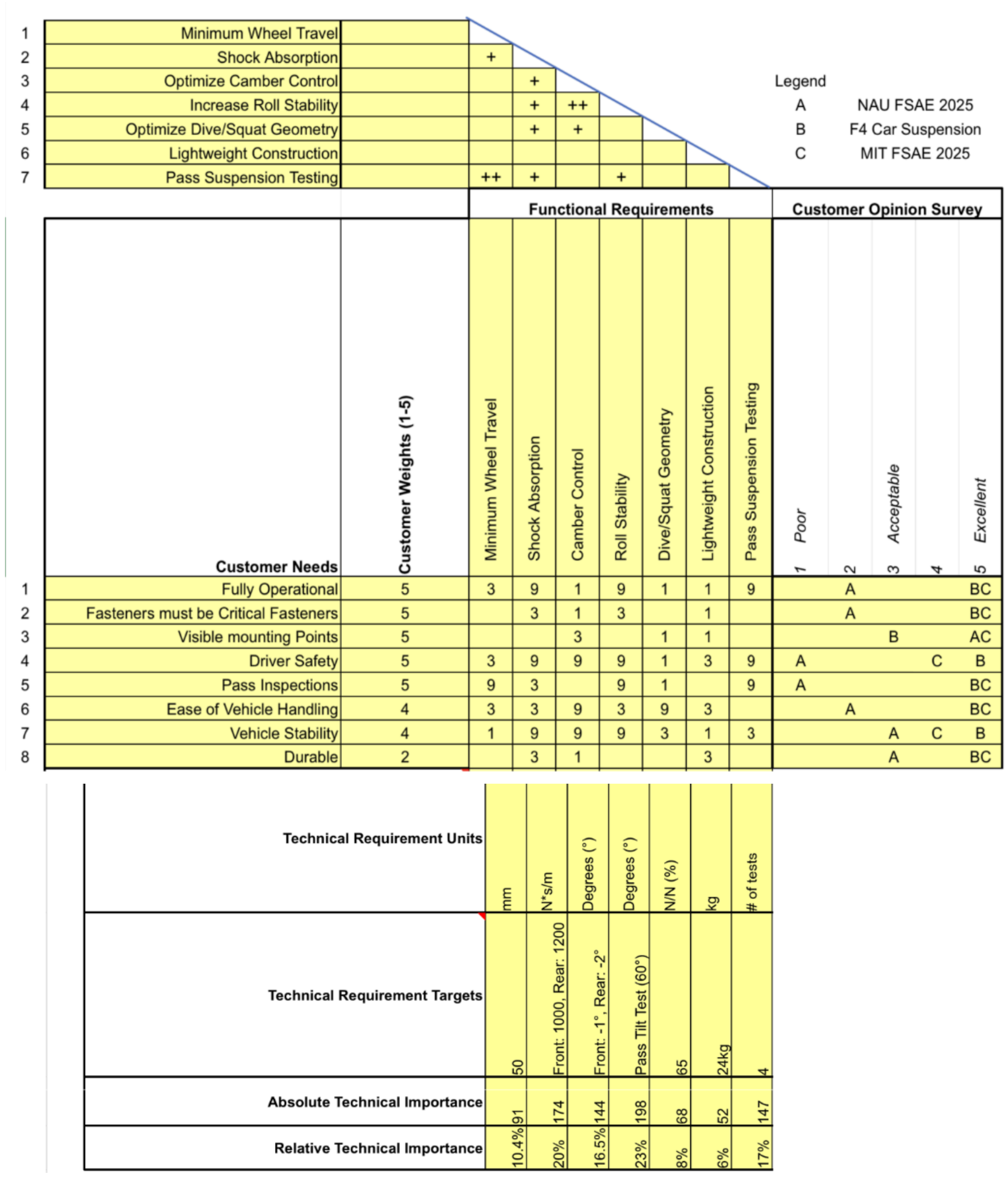


Figure 1: CAD

Shown is the current full frame with the pedal assembly placed at the front of the vehicle to accommodate a 95% male driver as dictated by the rules. Next along the chassis is the steering assembly to allow the driver to steer the vehicle through the courses and testing that is required.  To the sides are the front and rear wheels placed at the end of the suspension points that follow the contours of the double wishbone suspension.  Inside the front wheel is the brake assembly including the rotors and calipers attaching to the hub and knuckle respectively.  The rear hub differs from the front as it has a splined shaft to accommodate the powertrain.   All of these components work together to allow the vehicle to have a stable platform to ride on and tolerate uneven surfaces and the forces of cornering.  These sub-assemblies also allow the driver to finely control the direction, and speed of the vehicle.



| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | Minimum Wheel Travel | | | | | | | |
| 2 | Shock Absorption | + | | | | | | |
| 3 | Optimize Camber Control | | + | | | | | |
| 4 | Increase Roll Stability | | + | ++ | | | | |
| 5 | Optimize Dive/Squat Geometry | | + | + | | | | |
| 6 | Lightweight Construction | | | | | | | |
| 7 | Pass Suspension Testing | ++ | + | | + | | | |

Legend

| | |
|---|---|
| A | NAU FSAE 2025 |
| B | F4 Car Suspension |
| C | MIT FSAE 2025 |

| | | | Functional Requirements | | | | | | | Customer Opinion Survey | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Customer Needs | Customer Weights (1-5) | Minimum Wheel Travel | Shock Absorption | Camber Control | Roll Stability | Dive/Squat Geometry | Lightweight Construction | Pass Suspension Testing | 1 *Poor* | 2 | 3 *Acceptable* | 4 | 5 *Excellent* |
| 1 | Fully Operational | 5 | 3 | 9 | 1 | 9 | 1 | 1 | 9 | | A | | | BC |
| 2 | Fasteners must be Critical Fasteners | 5 | | 3 | 1 | 3 | | 1 | | | A | | | BC |
| 3 | Visible mounting Points | 5 | | | 3 | | 1 | 1 | | | | B | | AC |
| 4 | Driver Safety | 5 | 3 | 9 | 9 | 9 | 1 | 3 | 9 | A | | | C | B |
| 5 | Pass Inspections | 5 | 9 | 3 | | 9 | 1 | | 9 | A | | | | BC |
| 6 | Ease of Vehicle Handling | 4 | 3 | 3 | 9 | 3 | 9 | 3 | | | A | | | BC |
| 7 | Vehicle Stability | 4 | 1 | 9 | 9 | 9 | 3 | 1 | 3 | | | A | C | B |
| 8 | Durable | 2 | | 3 | 1 | | | 3 | | | | A | | BC |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Technical Requirement Units** | mm | N*s/m | Degrees (°) | Degrees (°) | N/N (%) | kg | # of tests |
| **Technical Requirement Targets** | 50 | Front: 1000, Rear: 1200 | Front: -1°, Rear: -2° | Pass Tilt Test (60°) | 65 | 24kg | 4 |
| **Absolute Technical Importance** | 91 | 174 | 144 | 198 | 68 | 52 | 147 |
| **Relative Technical Importance** | 10.4% | 20% | 16.5% | 23% | 8% | 6% | 17% |

Figure 2: QFD

**Engineering Requirements**

**Minimum Wheel Travel (50 mm) -** This is the least allowable vertical displacement of our vehicle's wheel relative to the chassis.

**Shock Absorption (Front: 1000 N*s/m, Rear: 1200 N*s/m) -** The ability of our suspension subsystem to dissipate kinetic energy from any impact or vibration present at the racecourse.

**Camber Control (Front: -1°, Rear: -2°) -** This is the amount of camber present in the front and rear wheels.

**Roll Stability (Pass the tilt test: 60°) -** A requirement provided by the rule book, where rule IN.11.2.2 states, "Vehicle does not roll when tilted at an angle of 60° to the horizontal, corresponding to 1.7 g."

**Dive Squat Geometry (65%) -** This is the ratio between the force returned by our suspension system and the amount of force placed on the front or rear axle. This value was found to be a recommended value during research and is subject to change as we continue with the design process.

**Lightweight Construction (24 kg) -** Our suspension subsystem will be made from lightweight materials to optimize performance.

**Pass Suspension Testing (All 4) -** We aim to pass all 4 suspension tests performed by the SAE staff before competition.

**Track Width (<1550mm) -** The track width needs to be below the 1550 mm mark by FSAE rules.

**Wheelbase (>1525mm) -** The wheelbase must be greater than 1525 mm by FSAE rules.

**Steering free play (<7°) -** The steering assembly must have less than 7° of steering wheel turn before the system is affected at the wheels by FSAE rules.

**Customer Requirements**

**Fully Operational –** Create a car that completes all standardized testing from SAE and passes all inspections before competition.

**Pass Inspections –** Similarly, must pass inspections made by SAE staff before competition.

**Fasteners must be Critical Fasteners -** This is a rule directly pulled from the Formula SAE Rules 2026, where rule V.3.1.4 states, "Fasteners in the Suspension system are Critical Fasteners." Critical fasteners are explained in T.8.2.

**Visible mounting points –** All mounting points connecting the suspension system to the frame must be visible for the inspectors.

**Driver Safety –** Install components that have the primary function of keeping the driver safe during competition events.

**Ease of Vehicle Handling –** Steering and suspension systems will be calibrated to allow the driver to easily and comfortably control the vehicle for optimum performance.

**Vehicle Stability -** Suspension subsystems will be built to provide stability to the car during all aspects of driving.

**Durable –** The suspension subsystem and all its components will be composed of reliable materials that will physically withstand all events during competition.

**Packaging –** The packaging used to secure our suspension subsystem's components to the frame will ensure that all components perform their purpose without interfering with other components.

**Sufficient braking system -** Part of the inspections and tests are that the braking system must be shown to be strong enough to lock all four tires at a reasonable speed.

**Summary of Standards, Codes, and Regulations**

The Formula SAE Rules 2026 define all inspection, safety, mechanical, and dimensional requirements and is the primary standard use for this design.

Suspension and Steering Regulation

- ➢ Fasteners must be Critical Fasteners
  - This is a rule directly pulled from the Formula SAE Rules 2026, where rule V.3.1.4 states, "Fasteners in the Suspension system are Critical Fasteners." Critical fasteners are explained in T.8.2.

$$\sigma = \frac{F}{A} \tag{1}$$

- ➢ Visible mounting points
  - All mounting points connecting the suspension system to the frame must be visible for the inspectors.
- ➢ Minimum Wheel Travel (50 mm)
  - This is the least allowable vertical displacement of our vehicle's wheel relative to the chassis.
- ➢ Shock Absorption (Front: 1000 N*s/m, Rear: 1200 N*s/m)
  - The ability of our suspension subsystem to dissipate kinetic energy from any impact or vibration present at the racecourse.
- ➢ Camber Control (Front: -1°, Rear: -2°)
  - This is the amount of camber present in the front and rear wheels.
- ➢ Dive Squat Geometry (65%)
  - This is the ratio between the force returned by our suspension system and the amount of force placed on the front or rear axle. This value was found to be a recommended value during research and is subject to change as we continue with the design process.
- ➢ Steering free play (<7°)
  - The steering assembly must have less than 7° of steering wheel turn before the system is affected at the wheels by FSAE rules.

Vehicle Stability and Safety

- ➢ Roll Stability (Pass the tilt test: 60°)
  - A requirement provided by the rule book, where rule IN.11.2.2 states, "Vehicle does not roll when tilted at an angle of 60° to the horizontal, corresponding to 1.7 g."

$$tan(\theta) = \frac{h_{cg}}{\frac{t}{2}} \tag{2}$$

- ➢ Sufficient braking system
    - ○ Part of the inspections and tests are that the braking system must be shown to be strong enough to lock all four tires at a reasonable speed.
    - ○ DOT has standards used for braking fluids on a numerical system with DOT4 being rated for uses where thermal aspects are expected.

$$T = F_{tire} \bullet r \tag{3}$$

Dimensional Constraints

- ➢ Track Width (<1550mm)
    - ○ The track width needs to be below the 1550 mm mark by FSAE rules.
- ➢ Wheelbase (<1500mm)
    - ○ The wheelbase must be below 1500 mm by FSAE rules.

**Summary of Equations and Solutions**

**Rocker Analysis (Chloe)**

The conditions in which the rocker was tested include simultaneous braking and bump. This is a condition in which the rocker will experience a high amount of force. The forces on the rocker were calculated using 2.5g braking deceleration and 2g bump acceleration, which is higher than our car will likely experience.

The loads calculated to be used in this simulation were found using the following calculations.

$$F_{z,\,wheel} = \frac{W_f mg(1+a_{bump})}{2}$$

The vertical wheel load was calculated using the assumed worst case bump acceleration to the front axle and dividing by the two wheels.

$$F_{pushrod} = \frac{F_{wheel}}{MR}$$

The wheel load was then converted to the force of the pushrod using the wheel-to-pushrod ratio. The same approach was used to find the force from the shock.

Results: $F_{pushrod} = 2600N$

$F_{shock} = 2920N$

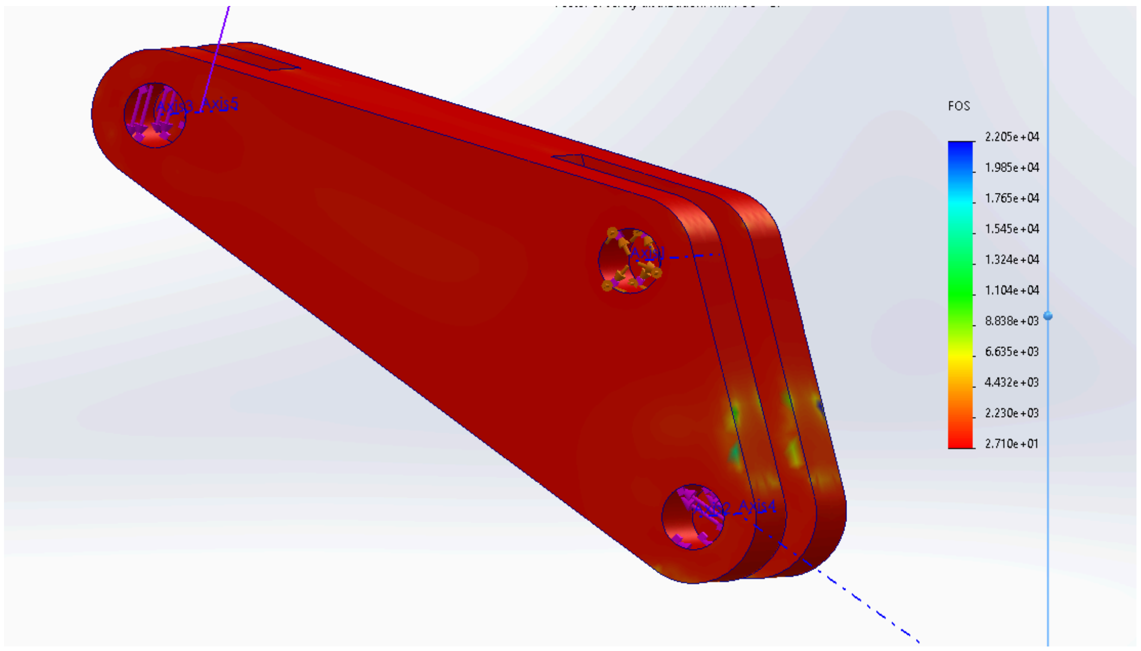


Figure 3: FEA on Front Right Rocker

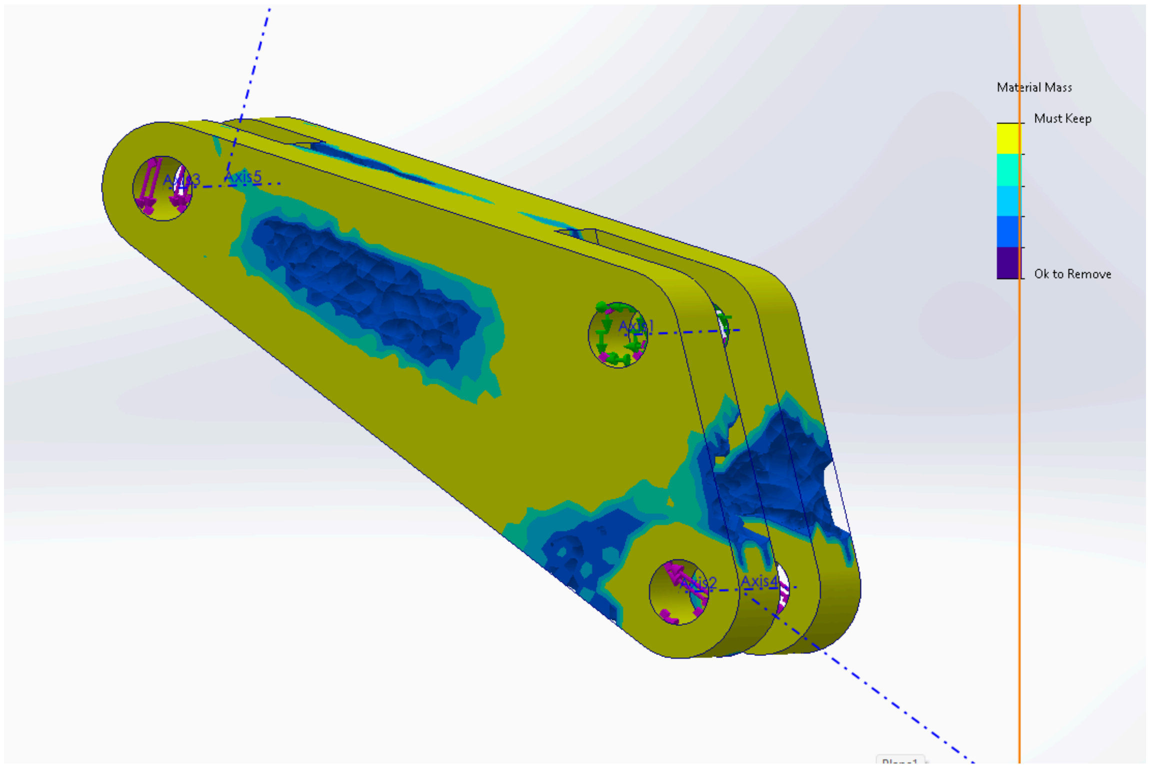


Figure 4: Topology Optimization on Front Right Rocker

A Finite Element Analysis and a topology optimization were run based on these loads. The minimum factor of safety is 27.1. This is far higher than necessary, so the topology optimization will prove to be incredibly useful. This will allow us to save weight and still ensure a reasonable factor of safety. The topology optimization was deemed necessary to ensure proper placement of supports between the two triangular plates as well as to remove any unnecessary material within them. More testing will be done.

Table 1: FoS of Rocker

| Subsystem | Part | Load Case Scenario | Material | Method of Calculating FoS | Minimum FoS |
|---|---|---|---|---|---|
| Rocker | Front Right | 2600N force applied along pushrod attachment point; 2920N force applied at coilover attachment point | Al 6061 | Solidworks FEA | 27.1 |

**Motion Analysis (Tanner)**

The conditions set for the experiments were limited heave, roll, pitch, and steering cases to check for instabilities within the suspension geometry, and effects that these motions would have on the wheels' vertical displacement and the motion ratios for pitch, heave, and roll. Drastic changes in these motion ratios as a result of small motions would display faults in the suspension system. As recommended by research, a heave of $\pm$1mm, and a pitch and roll of $\pm$0.2 degrees were the baseline motions for useful study on the suspension system within Optimum Kinematics.

**Heave**

For running a simulation on heave, higher values resulted in a solution that would not converge. Research suggested that this is because our suspension system has a "physically impossible" geometry, or that the simulation solver was simply not able to provide a numerical solution to the motion study. The motion study was reduced to $\pm$1mm of heave to provide a sanity check on the motion ratios. Below is a graph of the results.

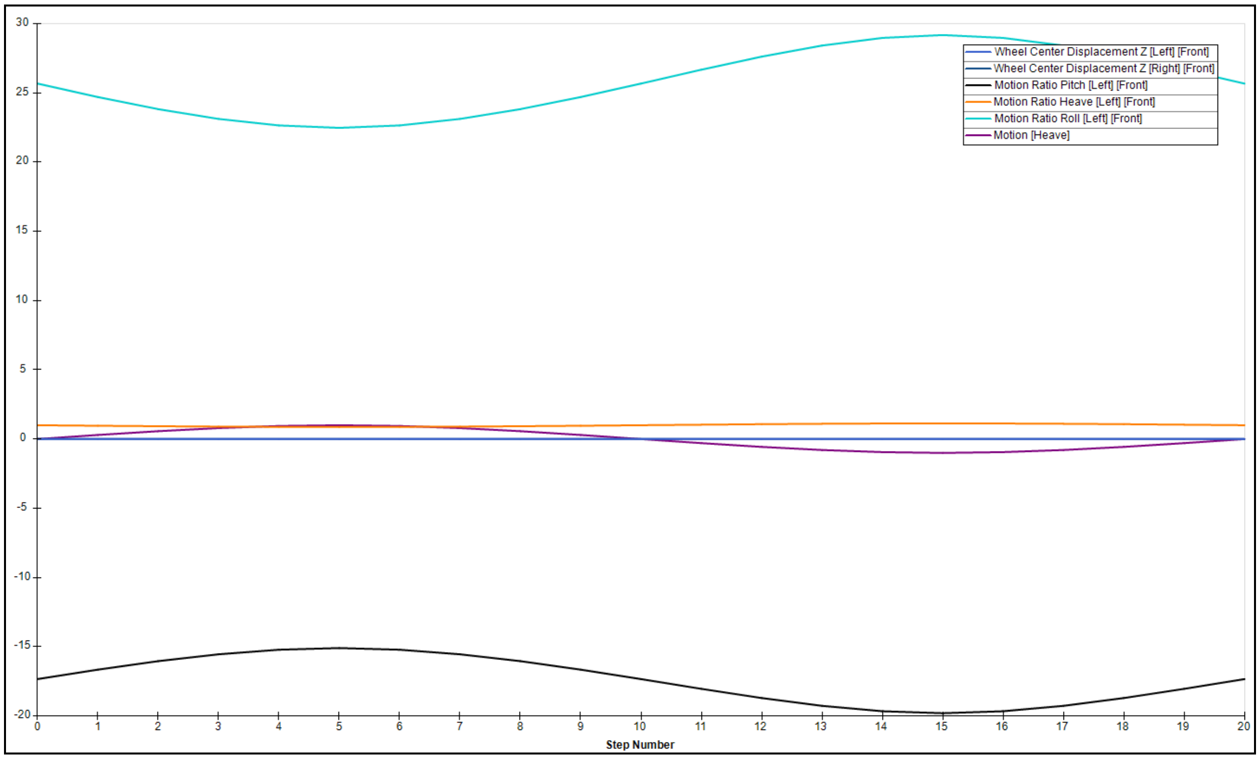


Figure 5. Heave Motion Results

As seen above, the motion ratios for pitch and roll did not experience drastic changes with the change in heave.

## Roll

When attempting to perform a motion analysis with a $\pm$0.2 degree change in roll, the solver software gave an error once again saying the solution would not converge. After learning more about the motion analysis software, I found that when there is perfect symmetry between the left and right suspension, the solver runs into a numerical issue when calculating under roll. After making attempts to reduce left-right symmetry of the system without destroying the motion analysis, there was still no solution convergence at the baseline roll values. The roll was then reduced to $\pm$0.05 degrees, and the simulation was run. Similar to heave, the effect of roll on the motion ratios and the vertical displacement of the wheels was minimal. This will be repeated in the Moving Forward section as well, but there are going to be further simulations run to display the effects of more intense motions on the suspension system. The graph below shows the results of the

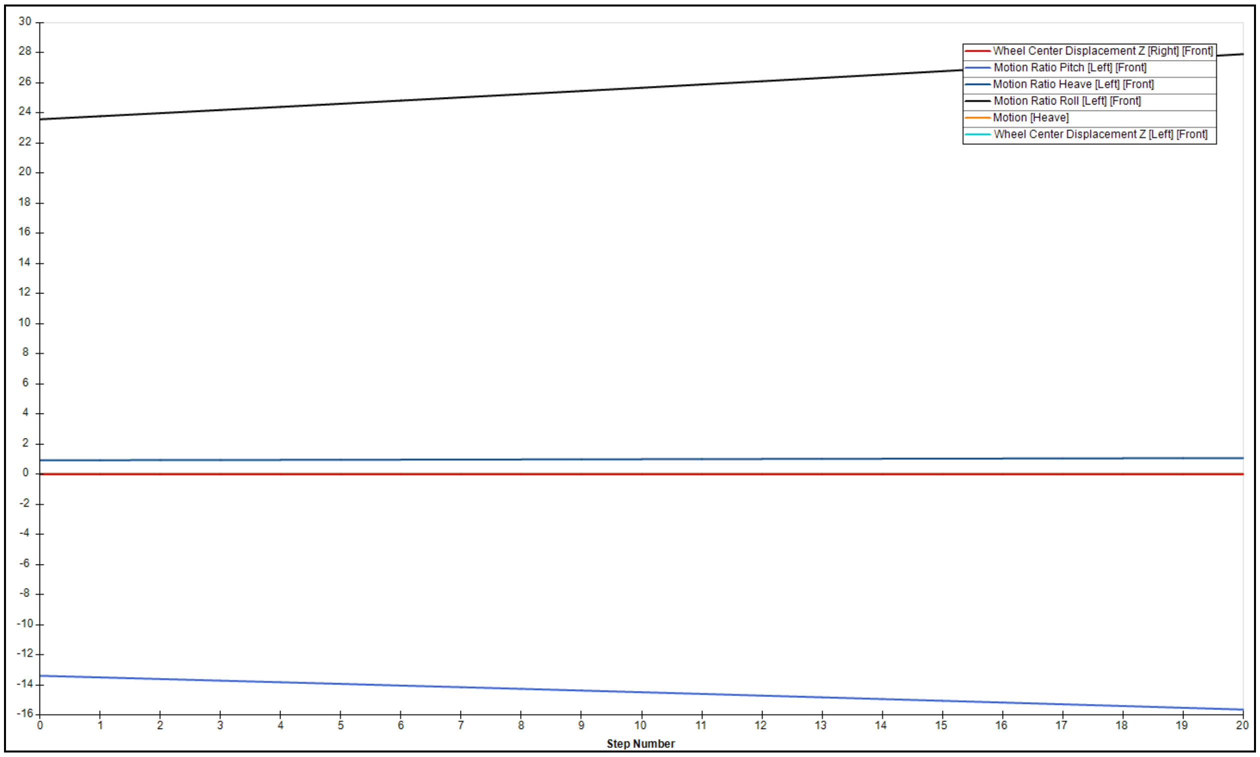


Figure 6. Roll Motion Results

Once again, the motion ratios stayed in their acceptable ranges, with negligible wheel center displacement.

**Pitch**

Similar to roll, a motion of $\pm 0.2$ degrees of pitch was tested on the vehicle. Below is the graph for the results of this motion analysis. The graph shows promising results in terms of motion ratio fluctuations in pitch and heave. However, the motion ratio for roll is a little too high when the vehicle is at +0.2 degrees of pitch.

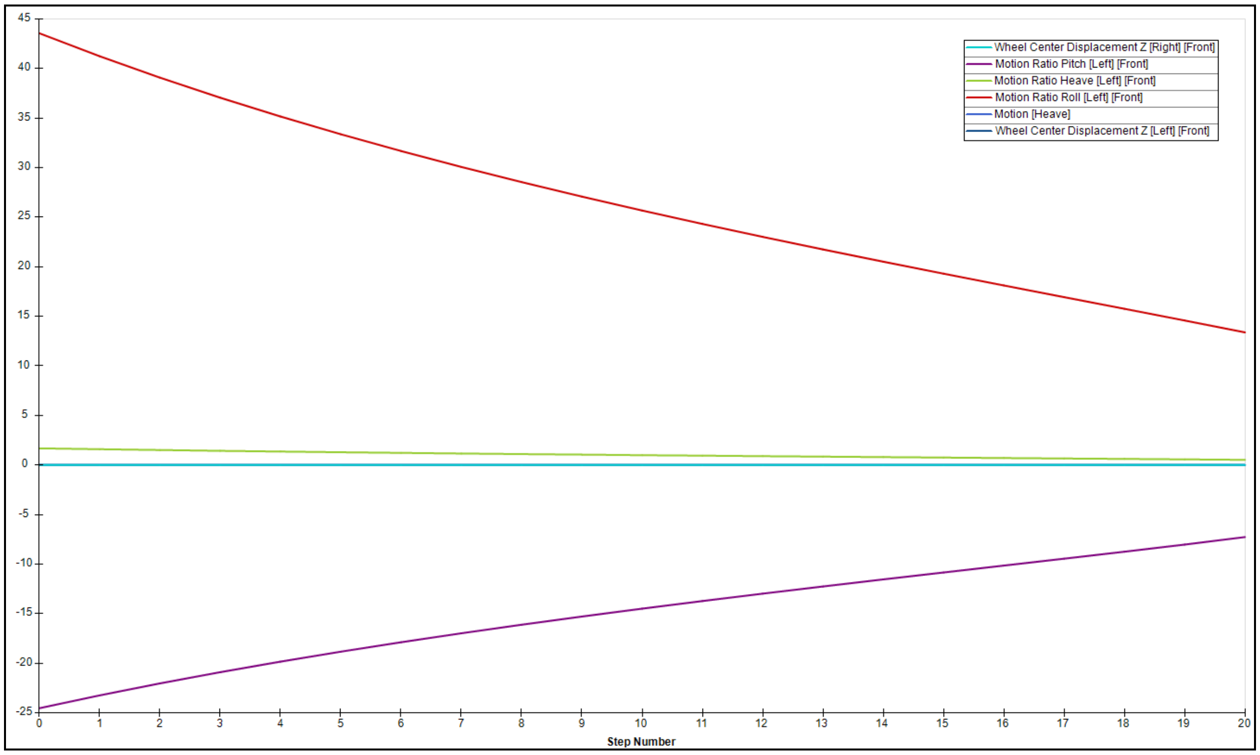


Figure 7. Pitch Motion Results

**Steering Analysis (Reuben)**

Need to navigate the given competition course layout which includes many differing types of corners, which includes an estimated 3m radius, 270° turn towards the end of the endurance lap. For this, using a MATLAB code to test for differing minimum radii, it was determined that for reasonably achievable steering angles of 22° and 33° for the outer and inner tires respectively, a 3m minimum turning radius is designed for. From this a graphical model of the front geometry was made to determine that to achieve the desired turning relationships, an offset of 55mm was found. This offset will be in front of the central axis of the front wheels with a NARCO steering rack to get an agreed upon steering travel of under 270° of steering wheel travel from full lock to full lock.

**Brakes Analysis (Reuben)**

Need to meet the requirement of full locking from average speed. Given weight of the vehicle, brake calipers, rotor diameter, material properties, and values for feasible driver input, need to determine master cylinder sizing for adequate braking performance.

Using fundamental equations for pressure and forces in a MATLAB code, it was determined that a master cylinder diameter combination of 1/2" for the front and 5/8" for the rear loop will be sufficient to lock the wheels from an average speed with a driver input force of around 430N which can be decreased with a beneficial pedal ratio.

These values have been determined with a minimum factor of safety of 1.5 to remain on the lower pressure side to ensure that the brake lines will not burst, but the line pressure is still adequate to slow the vehicle down.

Table 2: FoS Table for Brakes Sub-Assembly

| Sub-System | Part | Load Case | Material | Method of Calc | Minimum FoS |
|---|---|---|---|---|---|
| Brakes | Master Cylinders | Tire lock up | DOT4 Brake Fluid | MATLAB Code | 1.5 |

These values have informed a recommended pedal ratio to apply to the pedal assembly, nominal master cylinder diameters to accompany the chosen brake calipers and rotors, and a nominal pressure rating for all components of the brake lines to avoid pressure related failure.

**[Steering Knuckles and Uprights] Analysis (Austin )**

The knuckles/uprights of the vehicle are intended to transmit all lateral and longitudinal movements exerted by the ground onto the tires, as well as braking and bump forces, into the chassis. Uprights should be designed strong, although light, so as to minimize unsprung weight and carry the weight of the vehicle in abrasive conditions without structural failure.

Matlab was used to determine all the forces that our front and rear uprights would face in the scenarios of maximum vehicle cornering, braking, 2.5g bump, and a combination of cornering and braking. In this simulation, each scenarios' maximum calculated outputs were applied to the upright at the same time. This takes into account the worst possible scenario our uprights could face, regardless of front or rear position, and helps to design a single upright that can be machined for both front and rear use. This code can be viewed in the appendix. The following calculated forces were applied to the upright:

Table 3: Calculated Forces

| Maximum Calculated Forces Across All Scenarios | |
|---|---|
| Force Location/Type | Calculated Force/Moment |
| Upper A-Arm Mount | 1496.6 N |
| Lower A-Arm Mount | 4409.1 N |
| Tie Rod Mount | 890.4 N |
| Pushrod | 2692.5 N |
| Brake Caliper Mount | 3242.3 N |
| Wheel Bearing Force | 2202.5 N |
| Wheel Bearing Moment | 1527.2 (N*m) |

FEA Simulation Results on Upright

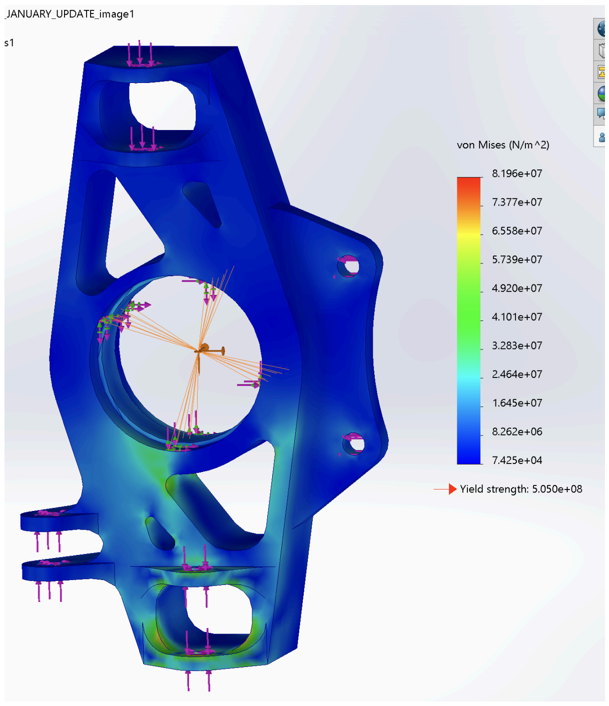


Figure: 8 Von Mises Stress Plot

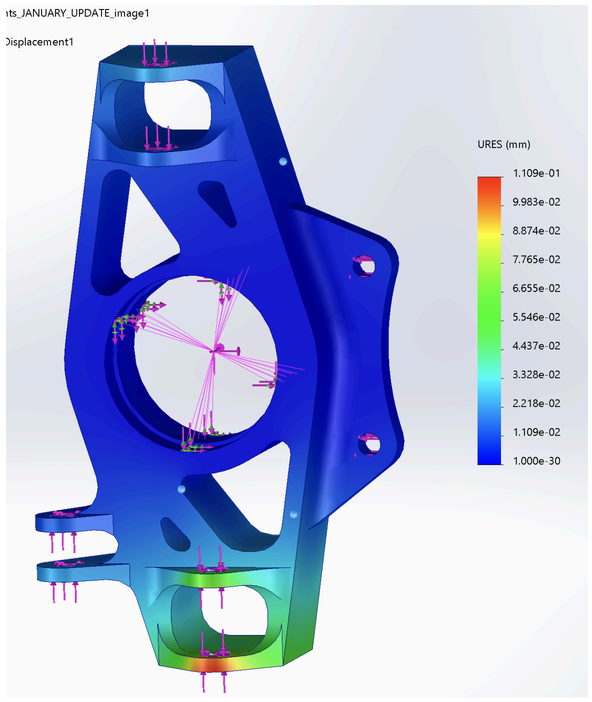


Fig 9: Displacement Heat Map (1:1 Deformation Scale)

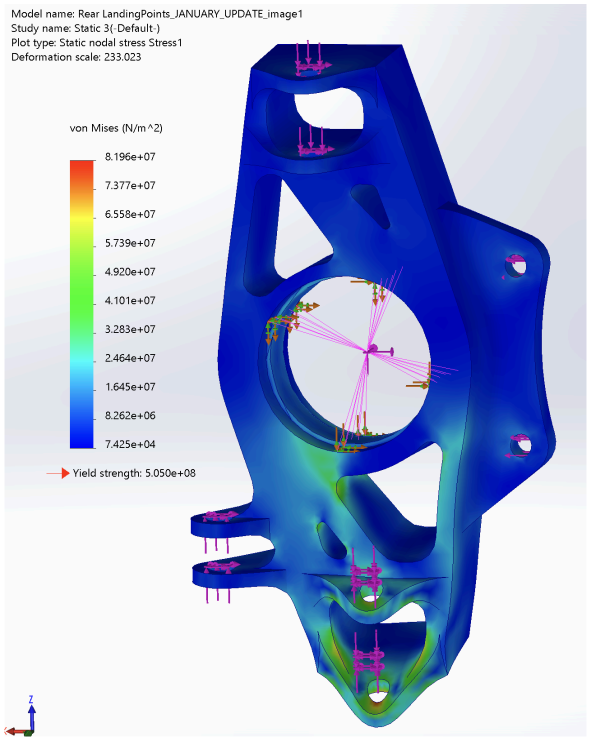

Model name: Rear LandingPoints_JANUARY_UPDATE_image1
Study name: Static 3(-Default-)
Plot type: Static nodal stress Stress1
Deformation scale: 233.023

Fig 10: Stress Plot (233:1 Deformation Scale)

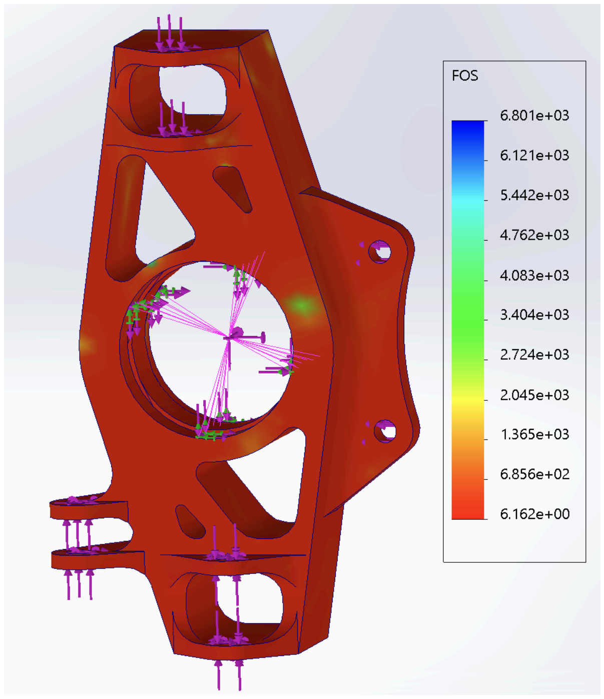


Figure 11: Factor Of Safety Plot

It should be noted that the calculated stresses appear to be somewhat low for the possible forces a Formula SAE vehicle upright could experience. 6000 to 9000 N of force are typical measurements for extreme scenarios, although these are highly unlikely. This might explain the high factor of safety measurement for the upright.

**Dampers & Spring Analysis (Maeve)**

The damper and spring system was designed around worst-case dynamic loading scenarios typical of FSAE competition, including heavy braking, aggressive cornering, rapid acceleration, and events such as curb strikes and surface irregularities. These are the conditions that our selected damper must safely and consistently accommodate.

Two damper velocities were used to capture the operating range of suspension:

Low-speed (0.05 m/s): Governs control during steady-state cornering, braking, and acceleration.

High-speed(0.25 m/s): Governs response during bump and curb impacts, where peak damper forces occur.

Target ride frequencies of 2.7 Hz (front) and 3.1 Hz (rear) were selected to balance aerodynamic platform stability, mechanical grip, and driver comfort. Target damping ratios of 0.3 in compression and 0.8 in rebound were selected to provide bump compliance while maintaining strong rebound control for rapid oscillation decay. These targets were selected to align within the tuning range of the selected Cane Creek IL dampers.

Sprung Mass per Corner:

Sprung mass per corner was computed using total mass, static weight distribution, and estimated unsprung mass. This value serves as the primary input for determining both spring stiffness and damper coefficients.

$$m_{front/rear} = m_{total} \bullet W_{front/rear}$$
$$m_{sprung} = \frac{m_{front/rear}}{2}$$

Results: front = 43.2 kg, rear = 46.8 kg

Ride Frequency Targeting:

Wheel rates were computed using the standard single-degree-of-freedom ride frequency relationship:

$$k_{wheel} = (2\Pi f)^2 m_{sprung}$$

This equation directly links suspension stiffness to desired response, ensuring that spring selection is driven by ride performance objectives rather than packaging constraints.

Results: front = 12.4 N/mm, rear = 17.7 N/mm

<u>Spring Rate:</u>

Installed spring rates were calculated using suspension motion ratios to account for rocker and pushrod geometry.

$$k_{spring} = \frac{k_{wheel}}{MR^2}$$

Results: front = 71 lb/in., rear = 160 lb/in.

These values were used to select compatible coil springs that fall within the optimal operating range of the Cane Creek IL dampers, ensuring proper preload range, stroke utilization, and adjustment authority.

<u>Critical Damping & Damping Ratios:</u>

Critical damping was computed to establish baseline damper sizing

$$c_{crit} = 2\sqrt{km}$$

Target damping coefficients were obtained using prescribed damping ratios

$$c = \zeta \bullet c_{crit}$$

These coefficients define the required damper force-velocity relationship necessary to meet dynamic performance targets.

Results: front compression = 438 Ns/m, rear compression = 543 Ns/m, front rebound = 1168 Ns/m, rear rebound = 1448 Ns/m

These values fall within the tunable operating range of the Cane Creek IL dampers, verifying that the selected dampers can achieve the required compression and rebound forces through adjustment rather than revalving.

<u>Damper Force Estimation:</u>

Damper forces were computed using linear viscous damper relationship:

$$F = c \bullet v$$

Using the high-speed damper velocity of 0.25 m/s to represent worst-case impact loading

Table 4. Damper Forces

| Location | Compression (N) | Rebound (N) | Max Load (N) |
|---|---|---|---|
| Front | 110 | 292 | 292 |
| Rear | 136 | 362 | 362 |

These peak forces confirm that the Cane Creek IL dampers operate well below their mechanical load capacity, providing sufficient safety margin for durability while maintaining high sensitivity at low shaft speeds.

Using MATLAB provided a consistent tool for developing spring and damper requirements along with an easy way to modify calculations and quickly conduct several iterations of vehicle development. The model automatically calculates:

- Sprung mass per corner
- Target wheel and spring rates
- Critical damping coefficients
- Compression and rebound damping coefficients
- Damper forces

This modeling approach allows rapid retuning of spring and damper parameters if vehicle mass, aerodynamic loading, or performance targets change. The complete MATLAB script is included in the Appendix and serves as a tuning and validation tool throughout the season.

Summary of MATLAB Outputs:

Table 5. Outputs

| | Sprung Mass (kg) | Spring Rates (lb/in.) | Critical Damping (Ns/m) | Max Damper Forces (N) | Rocker Design Loads (N) |
|---|---|---|---|---|---|
| Front | 43.20 | 71.6 | 1466 | 293 | 294 |
| Rear | 46.80 | 253.8 | 1823 | 365 | 577 |

Overall these calculations support the conclusion that the Cane Creek IL dampers fit our suspension requirements. The adjustable properties of the dampers allow customization of the ride and handling characteristics depending on how much adjustable range is needed. As calculated, the maximum amount of force expected to be applied to the dampers is much less than what they can safely support mechanically. To avoid premature failure, the maximum force on the dampers were calculated and the chosen damping system operated within its maximum mechanical limit. The provided calculations provide support to conclude that this damping system will function as intended and have an appropriate level of durability and compatibility within the suspension system.

## Flow Charts and other Diagrams

| Suspension System | | |
|---|---|---|
| Support & Load Transmission | Control Wheel Motion & Geometry | Ride & Roll Dynamics |
| Interfaces | Adjustability | Serviceability |

Figure 12: Suspension Decomposition

| | | |
|---|---|---|
| Driver Input | | |
| Bias Control | | Pedal Ratio |
| Safety & Redundancy | Heat Management | Hydraulic Lines |
| | | Calipers, Pads, & Rotors |

Figure 13: Brake Decomposition

| | | |
|---|---|---|
| Feedback & Feel | | Column & Quick Release |
| Sensors & Data Logging | | Rack & Pinion |
| | Kinematic Geometry | Tie Rods |
| | Packaging & Safety | Steering, Knuckles & Uprights |

Figure 14: Steering Decomposition

**Moving Forward**

The Optimum Kinematics motion analyses need to be re-run to perform more sanity checks on more intense motions. So far high values for heave and roll have led to issues in the solver software despite our geometry being acceptable in other softwares like Solidworks. Performing motion studies in solidworks with the entire suspension system is a good option for more analysis.

Finite Element Analysis and topology optimization will need to be executed on the rear rocker and iteration will be done on both front and rear rockers. This will be done before the next staff meeting. Similar simulations will need to be done on the control arms as well, which will also be executed before the next staff meeting.

A full CAD assembly is also in the works and will need to be completed by the next staff meeting as requested by Professor Willy and the Aero team.

## Appendix

### Damper Analysis

```
clear; clc;
% --- Motion Ratios ---
MR.front = 0.996;
MR.rear  = 0.632;
% --- Vehicle Mass ---
m_total = 230;                  % kg
weight_dist.front = 0.48;
weight_dist.rear  = 0.52;
% --- Unsprung Mass ---
m_unsprung.front = 12;          % kg per corner
m_unsprung.rear  = 13;
% --- Target Ride Frequencies (Aero) ---
f_target.front = 2.7;           % Hz
f_target.rear  = 3.1;
% --- Target Damping Ratios ---
zeta.comp  = 0.30;              % Compression
zeta.reb   = 0.80;              % Rebound
% --- Assumed Damper Shaft Velocities ---
v_damper.low  = 0.05;           % m/s (platform control)
v_damper.high = 0.25;           % m/s (bumps/curbs)
% --- Constants ---
g = 9.81;
% --- Sprung Mass per Corner ---
m_front_axle = m_total * weight_dist.front;
m_rear_axle  = m_total * weight_dist.rear;
m_sprung.front = (m_front_axle/2) - m_unsprung.front;
m_sprung.rear  = (m_rear_axle /2) - m_unsprung.rear;
% --- Wheel Rates ---
k_wheel.front = (2*pi*f_target.front)^2 * m_sprung.front; % N/m
k_wheel.rear  = (2*pi*f_target.rear )^2 * m_sprung.rear;
k_wheel.front = k_wheel.front / 1000; % N/mm
k_wheel.rear  = k_wheel.rear  / 1000;
% --- Spring Rates ---
k_spring.front = k_wheel.front / MR.front^2;
k_spring.rear  = k_wheel.rear  / MR.rear^2;
Nmm_to_lbin = 5.71;
k_spring.front_lbin = k_spring.front * Nmm_to_lbin;
k_spring.rear_lbin  = k_spring.rear  * Nmm_to_lbin;
% --- Critcal Damping ---
% c_crit = 2*sqrt(k*m)
c_crit.front = 2 * sqrt((k_wheel.front*1000) * m_sprung.front);
c_crit.rear  = 2 * sqrt((k_wheel.rear *1000) * m_sprung.rear);
% --- Target Damping ---
c.comp.front = zeta.comp * c_crit.front;
c.comp.rear  = zeta.comp * c_crit.rear;
c.reb.front  = zeta.reb  * c_crit.front;
```

```
c.reb.rear   = zeta.reb  * c_crit.rear;
% Compression forces
F_comp.front.low  = c.comp.front * v_damper.low;
F_comp.front.high = c.comp.front * v_damper.high;
F_comp.rear.low   = c.comp.rear  * v_damper.low;
F_comp.rear.high  = c.comp.rear  * v_damper.high;
% Rebound forces
F_reb.front.low   = c.reb.front * v_damper.low;
F_reb.front.high  = c.reb.front * v_damper.high;
F_reb.rear.low    = c.reb.rear  * v_damper.low;
F_reb.rear.high   = c.reb.rear  * v_damper.high;
% --- Rocker/Mount Loads ---
F_damper.front.max = max([F_comp.front.high, F_reb.front.high]);
F_damper.rear.max  = max([F_comp.rear.high,  F_reb.rear.high]);
% Approximate force transmitted to rocker
F_rocker.front = F_damper.front.max / MR.front;
F_rocker.rear  = F_damper.rear.max  / MR.rear;
fprintf('\n--- SPRUNG MASS PER CORNER ---\n');
fprintf('Front: %.2f kg\n', m_sprung.front);
fprintf('Rear : %.2f kg\n', m_sprung.rear);
fprintf('\n--- SPRING RATES ---\n');
fprintf('Front: %.1f lb/in\n', k_spring.front_lbin);
fprintf('Rear : %.1f lb/in\n', k_spring.rear_lbin);
fprintf('\n--- CRITICAL DAMPING ---\n');
fprintf('Front: %.0f N·s/m\n', c_crit.front);
fprintf('Rear : %.0f N·s/m\n', c_crit.rear);
fprintf('\n--- MAX DAMPER FORCES (HIGH SPEED) ---\n');
fprintf('Front: %.0f N\n', F_damper.front.max);
fprintf('Rear : %.0f N\n', F_damper.rear.max);
fprintf('\n--- ROCKER DESIGN LOADS ---\n');
fprintf('Front Rocker Load: %.0f N\n', F_rocker.front);
fprintf('Rear Rocker Load : %.0f N\n', F_rocker.rear);
```

## Brake Analysis

```
%%%%%%%%%%%%%
%Model for Brake Selection
%
% Variables:
% Tire size
% Pedal ratio
% Vehicle mass
% weight distribution
% Driver force
% required torque
% desired deceleration rate
% desired braking split(difference between front and rear force
clc; clear; close all;
W=250+45; %weight of vehicle + expected aero. force (kg)
Decel=2; %deceleration rate, in (g)
```

```
N=1.5;     %Factor of Safety
Forcereq=W*Decel*N; %required braking force (N)
%%%%%%%%%
BrkSplt=.6; %brake split, (%)towards front
FrontForce=Forcereq*BrkSplt; %front required stopping force (N)
BackForce=Forcereq*(1-BrkSplt); %Back required stopping force (N)
%%%%%%%%input for variables based on tire and rotor diameters
%Caliper specs
Numpist=2;  %Number of pistons
PistDiam=1; %Diameter of brake piston (in)
PistArea=pi*(PistDiam*25.4)^2/4; %Area of brake piston (mm^2)
RotorDiam=7*25.4; %Diameter of brake rotor (mm)
Padwidth=25.4;  %Width of brake pad (mm)
TireSLR=200;  %Static loaded radius (mm)
RotRadeff=(RotorDiam-Padwidth)/2;
FrontbForce=FrontForce*TireSLR/RotRadeff; %Braking force Front (N)
BackbForce=BackForce*TireSLR/RotRadeff; %Braking force Back (N)
%%%%%%%%%%%%%%insert pad friction coefficient
Mupad=0.35; %Friction coefficient of the pad and rotor materials
FrontClmp=FrontbForce/(Mupad*Numpist);   % Front and rear clamping force on the
BackClmp=BackbForce/(Mupad*Numpist);     %calipers from each piston   (N)
BrkLnPrsFront=FrontClmp/PistArea  %Pressure in the front brake line N/mm^2  MPa
BrkLnPrsBack=BackClmp/PistArea   %Pressure in the Back brake line  MPa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%insert driver force and pedal ratio
DrivForce=300;   %Force applied by the driver (N)
PedRatio=2;      %Pedal ratio
BrakeFrc=DrivForce*PedRatio;
FrontMSTCylDiam=sqrt(4*BrakeFrc/(pi*BrkLnPrsFront))/25.4 %Front Master cylinder
diameter (in)
BackMSTCylDiam=sqrt(4*BrakeFrc/(pi*BrkLnPrsBack))/25.4   %Back Master cylinder
diameter  (in)
%% Tourque Calcs
%set Master cylinder diam off of market availability
FrontMSTCylDiam=1/2;
BackMSTCylDiam=5/8;
MuStat=2; %Static friction of tire/road (slicks)
Tire=8*25.4/1000;  %Tire radius in (m)
TWeight=W*9.81;    %Weight in N
WWeightF=TWeight*.60/2;  %weight on single front tire
WWeightR=TWeight*.40/2;  %weight on single rear tire
SkidF=MuStat*WWeightF/2; %Skid force (N)
LockTorqueF=SkidF*Tire %Torque to lock tires Front
%Calculate force from pedal to create torque
LockFF=LockTorqueF/(RotRadeff/1000); %required brake force
FrontClmp=LockFF/(Mupad*Numpist);
LockPrsF=FrontClmp/PistArea
BrkFrcLockF=(FrontMSTCylDiam*25.4)^2*pi*LockPrsF/4
DrivForceLockF=BrkFrcLockF/PedRatio %driver force required to lock front tires
```

```
SkidR=MuStat*WWeightR/2; %Skid force (N) Rear
LockTorqueR=SkidR*Tire %Torque to lock tires Rears
%Calculate force from pedal to create torque
LockFR=LockTorqueR/(RotRadeff/1000); %required brake force
RearClmp=LockFR/(Mupad*Numpist);
LockPrsR=RearClmp/PistArea
BrkFrcLockR=(BackMSTCylDiam*25.4)^2*pi*LockPrsR/4
DrivForceLockR=BrkFrcLockR/PedRatio %driver force required to lock front tires
```

## Steering Analysis

```
%%%%%%%%%%%%%%%%%%%%%
% Model for Steering ranges
%
% Used to find optimal combination of ackerman percentage,
% turning angle, and predicted minimum turning radius
%
% Assumes minimal impact of slip angle
%
% Inputs:Wheelbase, Track width, and the range of wanted turning radius
%
% Outputs:Inner and outer angles needed for turning radius,
% ackerman percentage
clc; clear; close all;
%% Input Variables
R=[2:0.25:10];  %Turning Radius
L=1530/1000; %Wheelbase
T=1400/1000; %Track width
%% Calculate
InAngle=atand(L./(R-T/2)) %inner required turning angle (deg)
OutAngle=atand(L./(R+T/2)) %outer required turning angle (deg)
figure(1)
subplot(2,1,1)
plot(R,InAngle)
hold on
plot(R,OutAngle)
legend('inner angle','outer angle')
Ackermann=(InAngle-OutAngle)./InAngle;  %Ackerman percentage
subplot(2,1,2)
plot(R,Ackermann)
legend('Ackerman percentage')
```

## Upright Calculation Matlab Code:

```
% Vehicle Parameters
veh.mass = 250;                % Total vehicle mass [kg]
veh.weight_dist = 0.40;        % Front weight distribution (0.40 = 40% front)
veh.cg_height = 260;           % CG height from ground [mm]
veh.wheelbase = 1530;          % Wheelbase [mm]
veh.track_front = 1350;      % Front track width [mm]
veh.track_rear = 1250;       % Rear track width [mm]
% Tire Parameters
tire.mu_long = 1.8;          % Longitudinal friction coefficient
tire.mu_lat = 1.5;           % Lateral friction coefficient
tire.radius = 185;           % Loaded tire radius [mm]
% Brake Parameters
brake.bias = 0.55;           % Front brake bias (0.55 = 55% front)
brake.front_caliper_offset = [-57.92; -43.54; -0.15]; % Caliper mounting offset
from wheel center [X;Y;Z] mm
brake.rear_caliper_offset = [-57.92; -43.54; -0.15];
brake.front_rotor_radius = 88.9; % Front brake rotor effective radius [mm]
brake.rear_rotor_radius = 88.9;  % Rear brake rotor effective radius [mm]
% Loading Conditions to Analyze
load_cases = {
   'Max Lateral (Cornering)'
   'Max Braking'
   'Combined Braking + Cornering'
   'Bump (2.5g vertical)'
};
%% ======================================================================
%  SECTION 2: SUSPENSION GEOMETRY - FRONT UPRIGHT (RIGHT SIDE)
%% ======================================================================
% CRITICAL: For RIGHT side upright, Y should be NEGATIVE (inboard)
% Enter coordinates relative to wheel center in mm [X; Y; Z]
% X = forward(+) / rearward(-)
% Y = outboard(+) / inboard(-) -- RIGHT SIDE SHOULD BE NEGATIVE!
% Z = up(+) / down(-)
front_geom.name = 'Front Right Upright';
% OUTER pickup points on upright (relative to wheel center)
% NOTE: If you copy-pasted from Optimum K left side, flip the Y signs!
front_geom.UCA_outer = [7.5; -83.8; 90];      % Upper control arm outer pickup
front_geom.LCA_outer = [-7.5; -83.8; -90];    % Lower control arm outer pickup
front_geom.tie_rod_outer = [47.5; -68.8; -65]; % Tie rod outer pickup
front_geom.pushrod_lower = [-7.5; -133.8; -75]; % Pushrod lower pickup
% INNER pickup points (chassis side, relative to wheel center)
% These MUST be your actual values from Optimum K!
% Typical values: Y around -400mm (way inboard), measure YOUR car!
front_geom.UCA_inner = [-20; -413.8; 60];      % UCA chassis pickup (averaged)
front_geom.LCA_inner = [-20; -428.8; -110];    % LCA chassis pickup (averaged)
front_geom.tie_rod_inner = [87.5; -443.8; -75];  % Tie rod rack pickup
front_geom.pushrod_upper = [-7.5; -438.8; 425];
% Validate geometry
front_geom = validate_geometry(front_geom, 'front');
%% ======================================================================
%  SECTION 3: SUSPENSION GEOMETRY - REAR UPRIGHT (RIGHT SIDE)
%% ======================================================================
rear_geom.name = 'Rear Right Upright';
% OUTER pickup points on upright (relative to wheel center)
```

```matlab
rear_geom.UCA_outer = [7.5; -83.8; 90];
rear_geom.LCA_outer = [-7.5; -83.8; -90];
rear_geom.toe_link_outer = [47.5; -68.8; -65];
rear_geom.pushrod_lower = [-7.5; -133.8; -75];
% INNER pickup points (chassis side, relative to wheel center)
rear_geom.UCA_inner = [-22.52; -353.8; 54.85];
rear_geom.LCA_inner = [9.98; -363.8; -115.15];
rear_geom.toe_link_inner = [47.48; -363.49; -65.15];
rear_geom.pushrod_upper = [-7.52; -318.18; 194.85];
% Validate geometry
rear_geom = validate_geometry(rear_geom, 'rear');
%% =========================================================================
%  SECTION 4: CALCULATE FORCES FOR EACH LOAD CASE
%% =========================================================================
fprintf('\n========================================\n');
fprintf('FSAE UPRIGHT FORCE ANALYSIS\n');
fprintf('========================================\n\n');
results = struct();
for case_idx = 1:length(load_cases)
   fprintf('\n--- LOAD CASE %d: %s ---\n', case_idx, load_cases{case_idx});

   % Calculate tire forces for this load case
   [tire_forces, load_transfer] = calculate_tire_forces(case_idx, veh, tire);

   % Calculate forces on FRONT upright
   fprintf('\nFRONT UPRIGHT:\n');
   results.front(case_idx) = analyze_upright(front_geom, tire_forces.front, ...
       brake.front_caliper_offset, brake.front_rotor_radius, ...
       brake.bias, case_idx, tire, veh, 'front');

   % Calculate forces on REAR upright
   fprintf('\nREAR UPRIGHT:\n');
   results.rear(case_idx) = analyze_upright(rear_geom, tire_forces.rear, ...
       brake.rear_caliper_offset, brake.rear_rotor_radius, ...
       brake.bias, case_idx, tire, veh, 'rear');

   fprintf('\n');
end
%% =========================================================================
%  SECTION 5: EXPORT RESULTS FOR FEA
%% =========================================================================
fprintf('\n========================================\n');
fprintf('EXPORTING RESULTS\n');
fprintf('========================================\n\n');
export_to_fea(results, load_cases, 'front');
export_to_fea(results, load_cases, 'rear');
fprintf('Results exported to CSV files for FEA import.\n\n');
%% =========================================================================
%  FUNCTION DEFINITIONS
```

```
%% ========================================================================
function geom = validate_geometry(geom, position)
   % Validate and diagnose geometry issues

   fprintf('\n--- Validating %s geometry ---\n', upper(position));

   % Check that outer points are inboard (negative Y for right side)
   if geom.UCA_outer(2) > 0 || geom.LCA_outer(2) > 0
       warning('OUTER pickups have POSITIVE Y! For right side, Y should be
negative (inboard)');
       fprintf('  Did you export LEFT side coordinates? If so, flip all Y
signs!\n');
   end

   % Check that inner points are far inboard
   if strcmp(position, 'front')
       inner_points = [geom.UCA_inner(2), geom.LCA_inner(2), ...
                       geom.tie_rod_inner(2), geom.pushrod_upper(2)];
   else
       inner_points = [geom.UCA_inner(2), geom.LCA_inner(2), ...
                       geom.toe_link_inner(2), geom.pushrod_upper(2)];
   end

   if any(inner_points > -200)
       warning('INNER pickups are not far enough inboard (should be <
-200mm)');
       fprintf('  Check your chassis pickup coordinates!\n');
   end

   % Calculate and display member lengths
   if strcmp(position, 'front')
       L_UCA = norm(geom.UCA_outer - geom.UCA_inner);
       L_LCA = norm(geom.LCA_outer - geom.LCA_inner);
       L_tie = norm(geom.tie_rod_outer - geom.tie_rod_inner);
       L_pushrod = norm(geom.pushrod_upper - geom.pushrod_lower);

       fprintf('  UCA length: %.1f mm\n', L_UCA);
       fprintf('  LCA length: %.1f mm\n', L_LCA);
       fprintf('  Tie rod length: %.1f mm\n', L_tie);
       fprintf('  Pushrod length: %.1f mm\n', L_pushrod);

       % Check for reasonable lengths
       if L_UCA < 200 || L_UCA > 600
           warning('UCA length (%.1f mm) seems unusual. Expected 250-500mm',
L_UCA);
       end
       if L_LCA < 200 || L_LCA > 600
           warning('LCA length (%.1f mm) seems unusual. Expected 250-500mm',
L_LCA);
```

```
        end
    else
        L_UCA = norm(geom.UCA_outer - geom.UCA_inner);
        L_LCA = norm(geom.LCA_outer - geom.LCA_inner);
        L_toe = norm(geom.toe_link_outer - geom.toe_link_inner);
        L_pushrod = norm(geom.pushrod_upper - geom.pushrod_lower);

        fprintf('  UCA length: %.1f mm\n', L_UCA);
        fprintf('  LCA length: %.1f mm\n', L_LCA);
        fprintf('  Toe link length: %.1f mm\n', L_toe);
        fprintf('  Pushrod length: %.1f mm\n', L_pushrod);
    end

    % Check for collinearity issues
    v_UCA = geom.UCA_outer - geom.UCA_inner;
    v_LCA = geom.LCA_outer - geom.LCA_inner;

    u_UCA = v_UCA / norm(v_UCA);
    u_LCA = v_LCA / norm(v_LCA);

    angle_deg = acosd(dot(u_UCA, u_LCA));
    fprintf('  Angle between UCA and LCA: %.1f degrees\n', angle_deg);

    if angle_deg < 15 || angle_deg > 165
        warning('UCA and LCA are nearly parallel (%.1f deg)! This creates
numerical issues.', angle_deg);
    end
end
function [tire_forces, load_transfer] = calculate_tire_forces(case_idx, veh,
tire)
    % Calculate tire contact patch forces for each load case

    g = 9.81; % Gravity [m/s^2]

    % Static wheel loads
    Fz_front_static = veh.mass * g * veh.weight_dist / 2; % Per wheel [N]
    Fz_rear_static = veh.mass * g * (1 - veh.weight_dist) / 2; % Per wheel [N]

    switch case_idx
        case 1 % Max Lateral (Cornering)
            ay = tire.mu_lat * g; % Lateral acceleration [m/s^2]

            % Lateral load transfer
            delta_Fz_lat_front = (veh.mass * ay * veh.cg_height /
veh.track_front) * veh.weight_dist / 2;
            delta_Fz_lat_rear = (veh.mass * ay * veh.cg_height / veh.track_rear)
* (1 - veh.weight_dist) / 2;

            % Outside wheel (loaded in corner)
            tire_forces.front.Fx = 0;
            tire_forces.front.Fy = -(Fz_front_static + delta_Fz_lat_front) *
tire.mu_lat;
            tire_forces.front.Fz = -(Fz_front_static + delta_Fz_lat_front);

            tire_forces.rear.Fx = 0;
            tire_forces.rear.Fy = -(Fz_rear_static + delta_Fz_lat_rear) *
tire.mu_lat;
            tire_forces.rear.Fz = -(Fz_rear_static + delta_Fz_lat_rear);

            load_transfer.lateral_front = delta_Fz_lat_front;
            load_transfer.lateral_rear = delta_Fz_lat_rear;

        case 2 % Max Braking
            ax = -tire.mu_long * g;

            % Longitudinal load transfer
            delta_Fz_long = veh.mass * abs(ax) * veh.cg_height / veh.wheelbase;

            Fz_front_brake = Fz_front_static + delta_Fz_long / 2;
            Fz_rear_brake = Fz_rear_static - delta_Fz_long / 2;

            tire_forces.front.Fx = Fz_front_brake * tire.mu_long;
            tire_forces.front.Fy = 0;
            tire_forces.front.Fz = -Fz_front_brake;

            tire_forces.rear.Fx = Fz_rear_brake * tire.mu_long;
            tire_forces.rear.Fy = 0;
            tire_forces.rear.Fz = -Fz_rear_brake;

            load_transfer.longitudinal = delta_Fz_long;

        case 3 % Combined Braking + Cornering
            ax = -0.7 * tire.mu_long * g;
            ay = 0.7 * tire.mu_lat * g;

            delta_Fz_long = veh.mass * abs(ax) * veh.cg_height / veh.wheelbase;
            delta_Fz_lat_front = (veh.mass * ay * veh.cg_height /
veh.track_front) * veh.weight_dist / 2;
            delta_Fz_lat_rear = (veh.mass * ay * veh.cg_height / veh.track_rear)
* (1 - veh.weight_dist) / 2;

            Fz_front_combined = Fz_front_static + delta_Fz_long / 2 +
delta_Fz_lat_front;
            Fz_rear_combined = Fz_rear_static - delta_Fz_long / 2 +
delta_Fz_lat_rear;

            tire_forces.front.Fx = 0.7 * Fz_front_combined * tire.mu_long;
            tire_forces.front.Fy = -0.7 * Fz_front_combined * tire.mu_lat;
            tire_forces.front.Fz = -Fz_front_combined;

            tire_forces.rear.Fx = 0.7 * Fz_rear_combined * tire.mu_long;
            tire_forces.rear.Fy = -0.7 * Fz_rear_combined * tire.mu_lat;
            tire_forces.rear.Fz = -Fz_rear_combined;

            load_transfer.longitudinal = delta_Fz_long;
            load_transfer.lateral_front = delta_Fz_lat_front;
            load_transfer.lateral_rear = delta_Fz_lat_rear;

        case 4 % Bump (2.5g vertical)
            bump_factor = 2.5;

            tire_forces.front.Fx = 0;
            tire_forces.front.Fy = 0;
            tire_forces.front.Fz = -Fz_front_static * bump_factor;

            tire_forces.rear.Fx = 0;
            tire_forces.rear.Fy = 0;
            tire_forces.rear.Fz = -Fz_rear_static * bump_factor;

            load_transfer.bump_factor = bump_factor;
    end
end
function results = analyze_upright(geom, tire_forces, caliper_offset, ...
    rotor_radius, brake_bias, case_idx, tire, veh, position)
    % Analyze forces on upright using proper equilibrium equations

    % Contact patch forces (applied at ground contact)
    contact_point = [0; 0; -tire.radius]; % Relative to wheel center

    % Determine if this is a braking case
    is_braking = (case_idx == 2 || case_idx == 3);

    % Tire force vector at contact patch
    F_tire = [tire_forces.Fx; tire_forces.Fy; tire_forces.Fz];

    % Moment from tire forces about wheel center
    M_tire = cross(contact_point, F_tire);

    % Add brake torque moment if braking
    if is_braking
        % Brake torque = braking force * tire radius
        brake_torque = abs(tire_forces.Fx) * tire.radius; % N-mm

        % This creates a moment about wheel center in Y direction (pitch)
        M_brake = [0; brake_torque; 0]; % Torque opposes wheel rotation
        M_total = M_tire + M_brake;
    else
        M_total = M_tire;
    end

    % Suspension member unit vectors (from inner to outer)
    if strcmp(position, 'front')
        v_UCA = geom.UCA_outer - geom.UCA_inner;
        v_LCA = geom.LCA_outer - geom.LCA_inner;
        v_tie = geom.tie_rod_outer - geom.tie_rod_inner;
    else
        v_UCA = geom.UCA_outer - geom.UCA_inner;
        v_LCA = geom.LCA_outer - geom.LCA_inner;
        v_tie = geom.toe_link_outer - geom.toe_link_inner;
    end
    v_pushrod = geom.pushrod_upper - geom.pushrod_lower;

    u_UCA = v_UCA / norm(v_UCA);
    u_LCA = v_LCA / norm(v_LCA);
    u_tie = v_tie / norm(v_tie);
    u_pushrod = v_pushrod / norm(v_pushrod);

    % Moment arms from wheel center to pickup points (outer points on upright)
    r_UCA = geom.UCA_outer;
    r_LCA = geom.LCA_outer;
    if strcmp(position, 'front')
        r_tie = geom.tie_rod_outer;
    else
        r_tie = geom.toe_link_outer;
    end
    r_pushrod = geom.pushrod_lower;

    % Build equilibrium equations
    % Set up force equilibrium matrix (3 equations)
    A_force = [u_UCA, u_LCA, u_tie, u_pushrod];
    b_force = -F_tire;

    % Set up moment equilibrium matrix (3 equations) about wheel center
    A_moment = [cross(r_UCA, u_UCA), cross(r_LCA, u_LCA), ...
                cross(r_tie, u_tie), cross(r_pushrod, u_pushrod)];
    b_moment = -M_total;

    % Combine into 6x4 system
    A = [A_force; A_moment];
    b = [b_force; b_moment];

    % Solve using least squares
    F_members = A \ b;

    results.F_UCA = F_members(1);
    results.F_LCA = F_members(2);
```

```
    results.F_tie = F_members(3);
    results.F_pushrod = F_members(4);

    % Force vectors on upright at each pickup point
    results.F_UCA_vec = results.F_UCA * u_UCA;
    results.F_LCA_vec = results.F_LCA * u_LCA;
    results.F_tie_vec = results.F_tie * u_tie;
    results.F_pushrod_vec = results.F_pushrod * u_pushrod;

    % Calculate brake caliper force if braking
    if is_braking
        brake_torque = abs(tire_forces.Fx) * tire.radius;
        results.F_caliper_mag = brake_torque / rotor_radius;
        results.F_caliper = [results.F_caliper_mag; 0; 0];
    else
        results.F_caliper_mag = 0;
        results.F_caliper = [0; 0; 0];
    end

    % Wheel bearing reactions
    F_members_total = results.F_UCA_vec + results.F_LCA_vec + ...
                      results.F_tie_vec + results.F_pushrod_vec;
    results.F_bearing = -F_tire - F_members_total;

    % Bearing moments
    M_members = cross(r_UCA, results.F_UCA_vec) + cross(r_LCA,
results.F_LCA_vec) + ...
               cross(r_tie, results.F_tie_vec) + cross(r_pushrod,
results.F_pushrod_vec);
    results.M_bearing = -M_total - M_members;

    % Check equilibrium
    residual_F = norm(F_tire + F_members_total + results.F_bearing);
    residual_M = norm(M_total + M_members + results.M_bearing);

    if residual_F > 10 || residual_M > 1000
        warning('Poor equilibrium! Force residual: %.1f N, Moment residual: %.1f
N-mm', ...
            residual_F, residual_M);
    end

    % Tire forces for reference
    results.F_tire = F_tire;

    % Store geometry points for FEA export
    results.points.UCA = geom.UCA_outer;
    results.points.LCA = geom.LCA_outer;
    if strcmp(position, 'front')
        results.points.tie = geom.tie_rod_outer;
    else
        results.points.tie = geom.toe_link_outer;
    end
    results.points.pushrod = geom.pushrod_lower;
    results.points.wheel_center = [0; 0; 0];
    results.points.caliper = caliper_offset;

    % Print results
    fprintf('  Tire Forces [N]: Fx=%.1f, Fy=%.1f, Fz=%.1f (Mag: %.1f)\n', ...
        F_tire(1), F_tire(2), F_tire(3), norm(F_tire));
    fprintf('  Upper A-arm: %.1f N (%s) | Vector: [%.1f, %.1f, %.1f]\n', ...
        abs(results.F_UCA), get_member_type(results.F_UCA), ...
        results.F_UCA_vec(1), results.F_UCA_vec(2), results.F_UCA_vec(3));
    fprintf('  Lower A-arm: %.1f N (%s) | Vector: [%.1f, %.1f, %.1f]\n', ...
        abs(results.F_LCA), get_member_type(results.F_LCA), ...
        results.F_LCA_vec(1), results.F_LCA_vec(2), results.F_LCA_vec(3));
    fprintf('  Tie/Toe Rod: %.1f N (%s) | Vector: [%.1f, %.1f, %.1f]\n', ...
        abs(results.F_tie), get_member_type(results.F_tie), ...
        results.F_tie_vec(1), results.F_tie_vec(2), results.F_tie_vec(3));
    fprintf('  Pushrod: %.1f N (%s) | Vector: [%.1f, %.1f, %.1f]\n', ...
        abs(results.F_pushrod), get_member_type(results.F_pushrod), ...
        results.F_pushrod_vec(1), results.F_pushrod_vec(2),
results.F_pushrod_vec(3));
    if is_braking
        fprintf('  Brake Caliper: %.1f N | Vector: [%.1f, %.1f, %.1f]\n', ...
            abs(results.F_caliper_mag), ...
            results.F_caliper(1), results.F_caliper(2), results.F_caliper(3));
    end
    fprintf('  Wheel Bearing Force: [%.1f, %.1f, %.1f] N (Mag: %.1f)\n', ...
        results.F_bearing(1), results.F_bearing(2), results.F_bearing(3), ...
        norm(results.F_bearing));
    fprintf('  Wheel Bearing Moment: [%.1f, %.1f, %.1f] N-mm (Mag: %.1f)\n', ...
        results.M_bearing(1), results.M_bearing(2), results.M_bearing(3), ...
        norm(results.M_bearing));
end
function type_str = get_member_type(force)
    if force > 0
        type_str = 'Tension';
    else
        type_str = 'Compression';
    end
end
function export_to_fea(results, load_cases, position)
    % Export force data to CSV for FEA import

    if strcmp(position, 'front')
        data = results.front;
    else
        data = results.rear;
    end

    filename = sprintf('upright_forces_%s.csv', position);
    fid = fopen(filename, 'w');

    fprintf(fid, '%s Upright Forces for FEA\n', upper(position));
    fprintf(fid, 'All forces in Newtons, coordinates in mm\n\n');

    for i = 1:length(load_cases)
        fprintf(fid, '\nLoad Case %d: %s\n', i, load_cases{i});
        fprintf(fid, 'Point,X,Y,Z,Fx,Fy,Fz,F_magnitude\n');

        fprintf(fid, 'Upper_A-arm,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f\n', ...
            data(i).points.UCA(1), data(i).points.UCA(2), data(i).points.UCA(3),
...
            data(i).F_UCA_vec(1), data(i).F_UCA_vec(2), data(i).F_UCA_vec(3),
...
            abs(data(i).F_UCA));

        fprintf(fid, 'Lower_A-arm,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f\n', ...
            data(i).points.LCA(1), data(i).points.LCA(2), data(i).points.LCA(3),
...
            data(i).F_LCA_vec(1), data(i).F_LCA_vec(2), data(i).F_LCA_vec(3),
...
            abs(data(i).F_LCA));

        fprintf(fid, 'Tie_Toe_Rod,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f\n', ...
            data(i).points.tie(1), data(i).points.tie(2), data(i).points.tie(3),
...
            data(i).F_tie_vec(1), data(i).F_tie_vec(2), data(i).F_tie_vec(3),
...
            abs(data(i).F_tie));

        fprintf(fid, 'Pushrod,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f\n', ...
            data(i).points.pushrod(1), data(i).points.pushrod(2),
data(i).points.pushrod(3), ...
            data(i).F_pushrod_vec(1), data(i).F_pushrod_vec(2),
data(i).F_pushrod_vec(3), ...
            abs(data(i).F_pushrod));

        fprintf(fid, 'Wheel_Bearing,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f\n', ...
            data(i).points.wheel_center(1), data(i).points.wheel_center(2),
data(i).points.wheel_center(3), ...
            data(i).F_bearing(1), data(i).F_bearing(2), data(i).F_bearing(3),
...
            norm(data(i).F_bearing));

        fprintf(fid, 'Brake_Caliper,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f\n', ...
            data(i).points.caliper(1), data(i).points.caliper(2),
data(i).points.caliper(3), ...
            data(i).F_caliper(1), data(i).F_caliper(2), data(i).F_caliper(3),
...
            norm(data(i).F_caliper));

        fprintf(fid, '\n');
    end

    fclose(fid);
    fprintf('Exported %s upright forces to: %s\n', position, filename);
end
```