Quadratic Optimization-Based Controller for Carnegie Mellon Racing 24E

Carnegie Mellon University
16-811: Final Project

Alfred Chang
Carnegie Institute of Technology
Carnegie Mellon University
alfredc@andrew.cmu.edu

Aditti Ramsisaria
Robotics Institute
Carnegie Mellon University
aramsisa@andrew.cmu.edu

Jennifer Yang
Robotics Institute
Carnegie Mellon University
jennifey@andrew.cmu.edu

Abstract

Carnegie Mellon Racing (CMR) has participated in years of Formula SAE competitions. From building cars with combustible engines to switching to electric vehicles, CMR has gone through many iterations on the hardware and electronics components of the vehicle. However, the vehicle controls development is still relatively new and can be greatly improved upon. Although the current controls pipeline has made the vehicle more competitive in competition events, this controller is suboptimal. In this paper, we present a more optimal controller that utilizes quadratic programming to realize all of the constraints and targets. This controller outperforms the current controller and we recommend it be pursued further to implement onto the current and future generations of CMR's vehicles.

Index Terms

Traction Control, Yaw Control, FSAE, Quadratic Programming, Torque Vectoring, Electric Vehicles

I. Introduction

A. Background and Motivation

Carnegie Mellon Racing is CMU's Formula SAE EV Racing Team. Formula SAE is a collegiate league where students design and build racecars and compete at various competitions over the summer months. CMR has had many years of experience with designing and iterating on hardware and electronic components of the vehicle (1). However, the team's design of vehicle controllers is still in its infancy and can be seen as one of the final frontiers for CMR.

Vehicle controls is an area that the world's best FSAE teams have mastered, allowing them to set records in events from acceleration to skid pad to autocross. The current world record for a competition spec acceleration run is 2.999s (1.7 g-force) held by TU Delft at Formula Student Germany in 2023. This past year, CMR implemented its first version of general purpose traction control and yaw rate control. The team saw an increase in average longitudinal and average lateral acceleration of about 5% in the accel and skidpad events respectively. However, there is still significant room for improvement. For comparison, CMR's best acceleration run at Michigan in 2023 was 4.131s (0.9 g-force).

While our controllers helped the team achieve new team records in accel and skidpad and provided drivers with better control of the vehicle, the controllers in their current setup try to realize adversarial goals making them suboptimal. One solution the team is interested in pursuing is the use of a quadratic programming optimizer which accounts for all control targets— tractive capacity of the tires, desired yaw rate, and power limit requirements by FSAE rules— in one cost function. This setup creates a unified controller which optimizes over all the given constraints and goals.

Fig. 1: Carnegie Mellon Racing's 22E Vehicle

II. Methods

A. The Current Pipeline

The current pipeline is a sequential set of controllers that have conflicting goals. This pipeline is visualized in Figure 2 Each controller returns its optimal solution, but final solution isn't necessarily the optimal combination of solutions. First, the vehicle's requests for steering angle and throttle position are converted into yaw rate and linear acceleration requests. These are then fed into a yaw rate controller, which optimizes the yaw rate to maximize the driver control. The controller uses the desired yaw rate and actual yaw rate to calculate the yaw rate error, which is then fed into a PID controller. This returns a left-right bias, which can then be used to calculate the left and right motor torques. These values are then fed into the traction controller to manage the driver throttle without wheel spin. These return updated motor torques that have been limited by the maximum possible tractive forces. Finally, these values are passed into a safety filter to ensure that the vehicle is not exceeding its power limits.

For the purposes of this project, we have made some simplifying assumptions. For the yaw rate controller, rather than using desired yaw rate as our input to get the left-right bias, we interpolate the left-right bias by mapping the steering angle to half of the yaw moment range and converting the resulting value to a torque value. We also assume that the front and rear torques are the same per side. To ensure we do not exceed the power limits, if the net power exceeds the power limit then all of the torques are scaled down such that the net power draw is the maximum possible power draw.

Fig. 2: Current Controls Pipeline

B. Vehicle Dynamics

For the following constants and variables,

- d = steering angle (clockwise positive)
- l = half length of car = 0.775 m
- w = half width of car = 0.650 m
- r = tyre radius = 0.23241 m
- g = gear ratio = 15
- m = mass of car = 310 kg
- $\tau_1, \tau_2, \tau_3, \tau_4$ are the motor torques which correspond to the front left, front right, rear left, and rear right motors respectively

the vehicle dynamics assuming clockwise orientation to be positive are given by

$$a(\vec{\tau}) = (\frac{g}{m \times r})(\tau_1 + \tau_2 + \tau_3 + \tau_4)$$

$$M(\vec{\tau}) = (\frac{g}{r})(\tau_1 l \sin(d) + \tau_1 w \cos(d) + \tau_2 l sin(d) - \tau_2 w cos(d) + \tau_3 w - \tau_4 w)$$

Here, $a(\vec{\tau}), M(\vec{\tau})$ are the measured linear acceleration and yaw moment as a function of the motor torques respectively.

C. Cost Function

The cost function is given by the following equation:

$$\min_{\vec{\tau}=[\tau_{FL},\tau_{FR},\tau_{RL},\tau_{RR}]} k_{lin}(a_{req} - a(\vec{\tau}))^2 + k_{yaw}(M_{req} - M(\vec{\tau}))^2 + k_{tie}||\vec{\tau}||_2^2$$

where

- $\vec{\tau} = [\tau_{FL}, \tau_{FR}, \tau_{RL}, \tau_{RR}]$ represents the motor torques to the front-left (FL), front-right (FR), rear-left (RL) and rear-right (RR) tyres
- a_{req} and M_{req} are driver input-based requests for linear acceleration and yaw moment respectively
- $a(\vec{\tau})$ and $M(\vec{\tau})$ are the linear acceleration and yaw moment realised by the model as a function of the motor torques
- $k_{lin}, k_{yaw}, k_{tie}$ are tuneable weights
- $||\vec{\tau}||_2$ is the L2 norm of the torque vector (regularization term)

The first term in our cost function, $k_{lin}(a_{req} - a(\vec{\tau}))^2$, addresses the optimization of linear acceleration. It penalizes deviations between the requested and measured linear accelerations. This term is quadratic in nature, fostering a smooth and well-behaved optimization landscape. The second term, $k_{yaw}(M_{req} - M(\vec{\tau}))^2$, governs the optimization of yaw motion. Analogous to the linear acceleration term, it penalizes discrepancies between the requested and measured yaw moments. This term ensures the vehicle's trajectory aligns with the desired yaw characteristics, promoting stability and control. The third term, $k_{tie}||\vec{\tau}||_2^2$, introduces a regularization component. It discourages extreme variations in torque values, promoting a more balanced distribution of torques among the motors. This regularization fosters a controlled and energy-efficient application of torques.

Our cost function is inherently convex and quadratic for several reasons.

1) Squared Terms: All terms in the cost function are quadratic, involving squared deviations from the desired values. Quadratic terms result in convexity, ensuring that the optimization problem possesses a unique global minimum.

2) Linear Combinations: The cost function is a linear combination of the individual terms, each being a convex function in itself. The linearity in combination preserves convexity, facilitating efficient optimization using convex optimization solvers such as OSQP.

3) Regularization Term: The inclusion of the torque regularization term in the quadratic cost function further contributes to convexity. Regularization terms often introduce convexity by penalizing extreme values and promoting a more controlled optimization landscape.

4) Constraints: The convex nature of the cost function is complemented by convex constraints, such as motor efficiency constraints and traction limits. The interplay of convex cost functions and constraints ensures the optimization problem is not only mathematically sound but also solvable in real-time.

By formulating our cost function as a convex quadratic expression, we leverage the efficiency and reliability of convex optimization methods, such as OSQP. This strategic choice underpins the success of our Traction and Yaw Control System in achieving optimal and real-time solutions for the dynamic demands of Formula SAE racing.

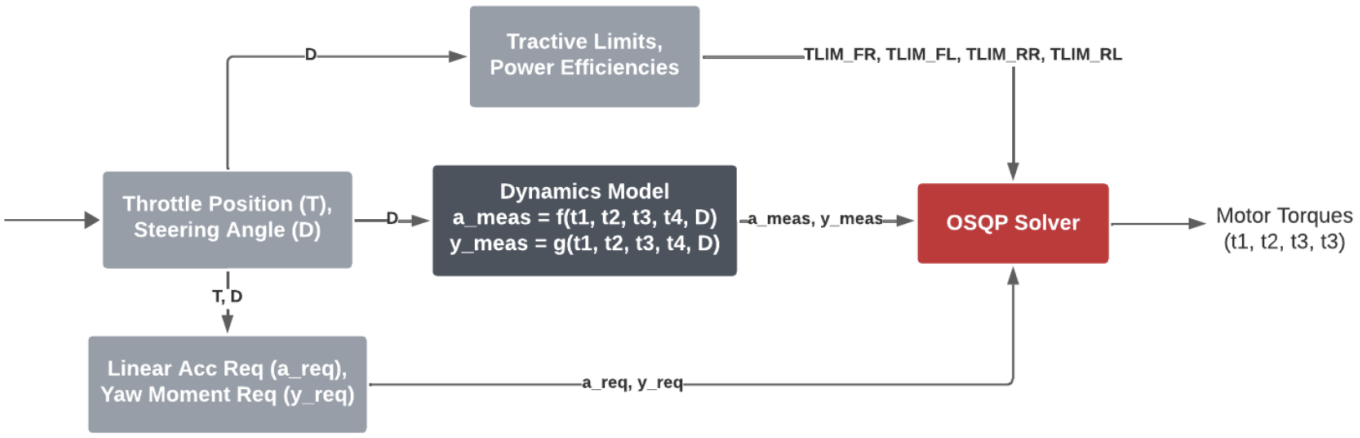


Fig. 3: OSQP Controls Pipeline

D. Constraints

The OSQP solution must account for the vehicle's physical and regulatory constraints. The constraints that the solution satisfied are as follows:

- The AMK hub motors on the vehicle have a 21Nm maximum torque and a maximum rotational speed of 19000 RPM.
- The steering angle of the front wheels has a range of ±22°.
- Finally, the FSAE rulebook mandates the high voltage power consumption of the vehicle remain below 80kW.

E. Quadratic Optimization and Pipeline

OSQP, short for Operator Splitting Quadratic Program, is a high-performance, open-source numerical optimization library designed for solving Quadratic Programming (QP) problems. Written in C and accessible through various interfaces, OSQP offers robust and efficient optimization capabilities. In our project, we utilized OSQP through its Python interface to optimize the control strategy for the 24E.

The underlying mathematics of OSQP are rooted in the alternating direction method of multipliers (ADMM) optimization algorithm. This algorithm efficiently solves convex quadratic programs by splitting the original problem into smaller, more manageable subproblems. The iterative nature of ADMM, combined with the specifics of the QP formulation, enables OSQP to find optimal solutions for large-scale problems with sparse matrices.

OSQP is useful in solving convex optimization problems of the form

$$\min \frac{1}{2} x^T P x + q^T x$$

subject to $l \leq Ax \leq u$ where, x is the optimization variable ($\vec{\tau}$) The objective function is defined by a positive semidefinite matrix P and vector q. The linear constraints are defined by matrix A and vectors l and u.

F. Testing Setup

1) Generating Driver Inputs: For the purposes of this project, we generate sine wave inputs for the steering angle and throttle position (driver requests) and linearly map these values to a request for linear acceleration and torque within the constraints of the vehicle dynamics for the 24E. These values then get updated periodically to generate desired requests for every time step.

2) Generating Results: We used the aforementioned linear acceleration and torque request waves as sweeps for evaluating both the current pipeline, which we used as a baseline, and the OSQP pipeline. We tested multiple combinations of motor RPM and downforce per wheel. The motor RPM values were 1000 RPM, 9000 RPM, and 19000 RPM, which were the low, medium, and high ends of the motor RPM range. The downforce per wheel values were 790 N and 1292 N, which correspond to when the tractive capacity is below versus at the motor torque limits respectively. From this, we generated achieved values for linear acceleration, yaw moments, power limits, and torque distributions.

III. Results and Figures

Tables I and II show the average errors for linear acceleration and yaw moment at each motor speed for both of the current pipeline and the OSQP pipeline. We used the Mean Average Error (MAE) error metric for our calculations. y_i is the requested value and x_i is the achieved value.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

When looking at one specific downforce value, at 1000 and 9000 RPM, all of the error values are similar. The OSQP errors increase slightly when going from 1000 RPM to 9000 RPM. At 19000 RPM, all of the errors other than the OSQP pipeline yaw moment error increase. The OSQP error yaw moment error decreases significantly. When comparing between downforce values, the error values at at 1000 and 9000 are lower for 790 N than the error values for 1292 N. However for 19000 RPM, the error values are similar between 790 N of downforce and 1292 N of downforce. The only difference is that the yaw moment error for the current pipeline decreases when the downforce increases.

Notably, the current pipeline demonstrates marginally lower errors than the OSQP pipeline does for linear acceleration. However, the current pipeline yaw moment errors are significantly higher than the OSQP pipeline errors, especially at 19000 RPM.

Motor RPM	**Linear Acceleration Error** (m/s^2)		**Yaw Moment Error** (Nm)	
	Current Pipeline	OSQP Pipeline	Current Pipeline	OSQP Pipeline
1000	2.271	2.464	148.9	8.538
9000	2.271	2.473	148.9	8.658
19000	5.883	5.960	218.5	0.227

TABLE I: Current Pipeline VS OSQP Pipeline Results with 790 N Downforce Per Wheel

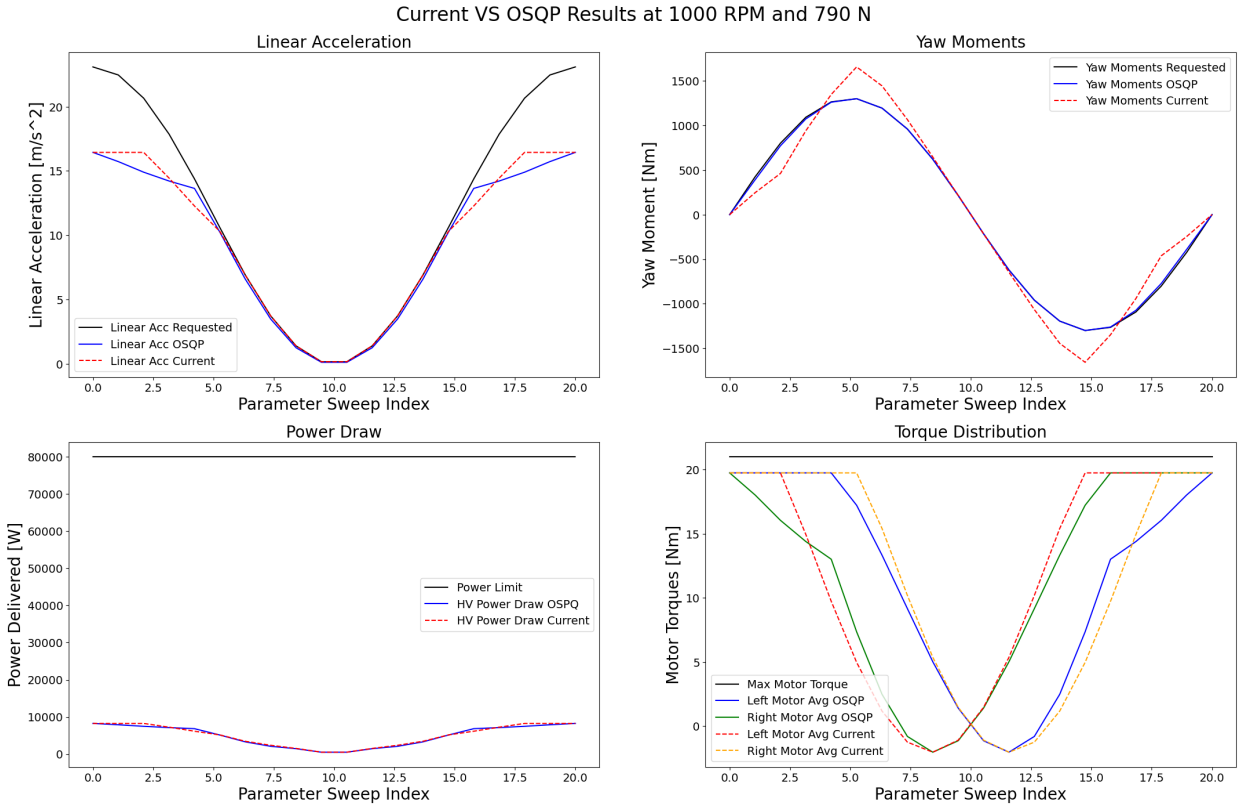
Motor RPM	**Linear Acceleration Error** (m/s^2)		**Yaw Moment Error** (Nm)	
	Current Pipeline	OSQP Pipeline	Current Pipeline	OSQP Pipeline
1000	1.864	1.974	144.9	6.448
9000	1.864	1.975	144.9	6.500
19000	5.883	5.960	207.1	0.227

TABLE II: Current Pipeline VS OSQP Pipeline Results with 1292 N Downforce Per Wheel

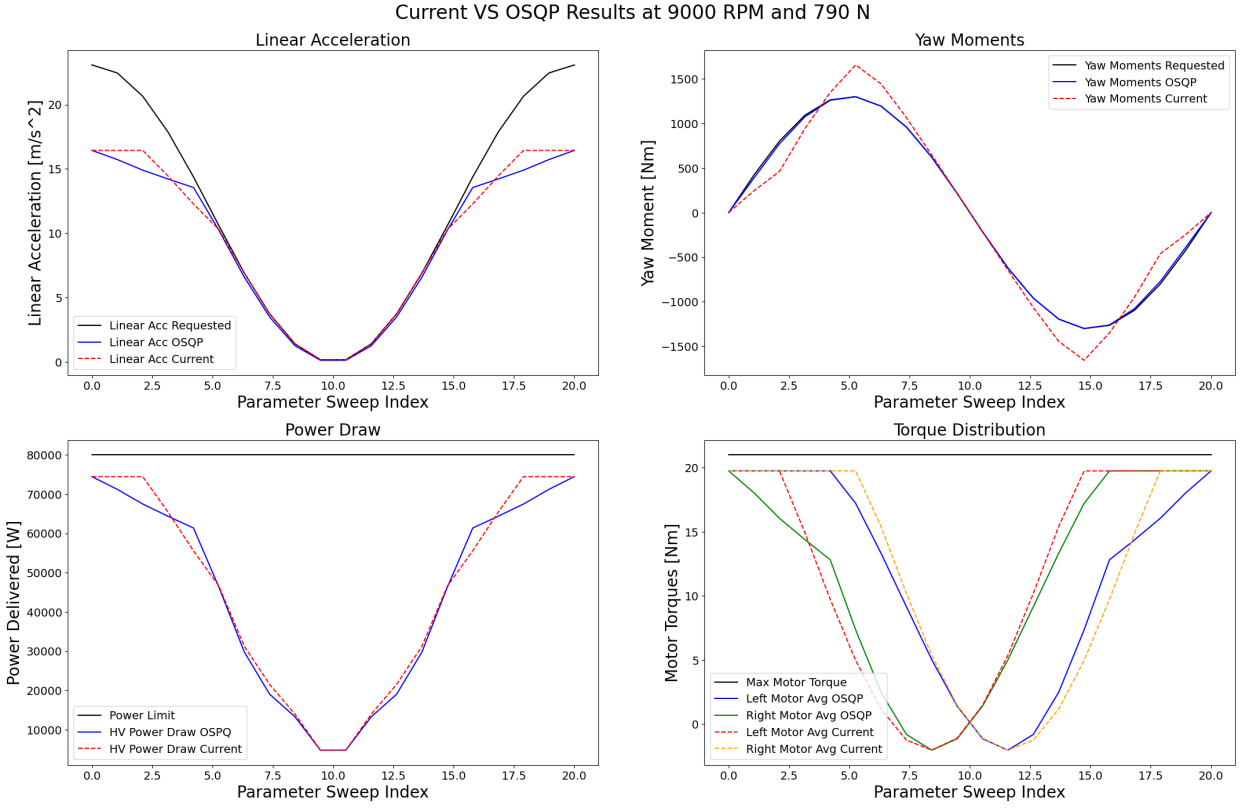
Figures 4 and 5 demonstrate the comparison of the current pipeline to the OSQP pipeline results at each motor speed and downforce value combination. The graphs shown are the linear acceleration curves, yaw moment curves, power draw curves, and torque distribution curves. For these figures, we elected to average the front and rear motor torques on the left and the right sides to provide a better visual comparsion of the two torques.

We observe that both solutions tend to follow the same trends although the OSQP solution generally produces smoother results than the current solution. The current pipeline has a tendency to overshoot the yaw moment requests whereas the OSQP pipeline generally follows the requested yaw moment curve. The overshoot is worse as the motor RPM increases. When the power draw hit its limits, the maximum achieved linear acceleration values are lower than when compared to when the power draw isn't at the limits. This happens at the 19000 RPM in each downforcce situation. The left and right torques are symmetrical across the point where the requested yaw moment is 0. This symmetry can be thought of as changing turning directions. At 1000 and 9000 RPM, the left and right values tend to decrease before increasing, creating a parabola-like curve. However at 19000 RPM, the curves created are more similar to a wave with the left motor increasing, decreasing, and then increasing again and the right motor decreasing, increasing, and then decreasing again.

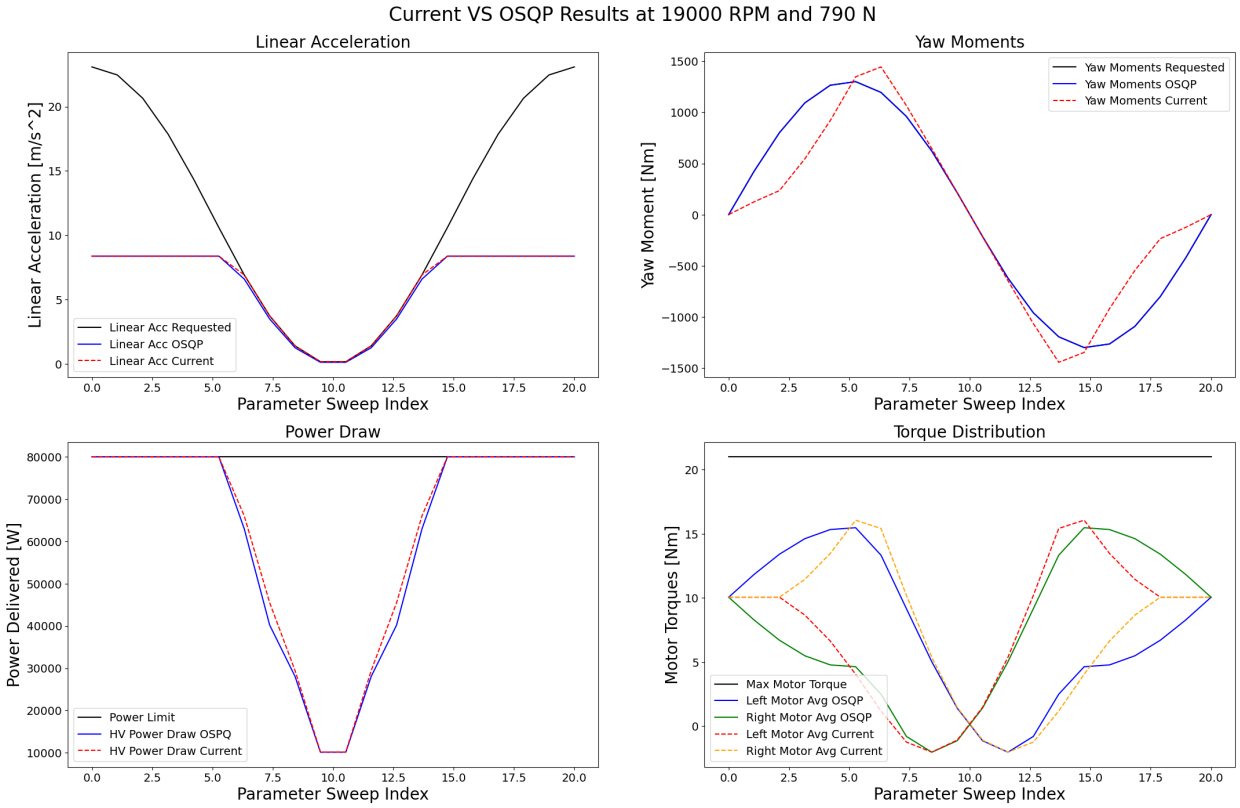
For the current pipeline, we assumed that the front and rear motor torques were equal on the left side and the right side, so the average motor torque for one side was actually the motor torque for both the front and the rear motors on that side. However, the OSQP pipeline does not make that same assumption. Figures 6 and 7 provide a more in-depth look at the torque distribution for the OSQP pipeline. We observe that in the OSQP solution, the front motors tend to have a greater torque value than the rear motors.



(a) Results at 1000 RPM

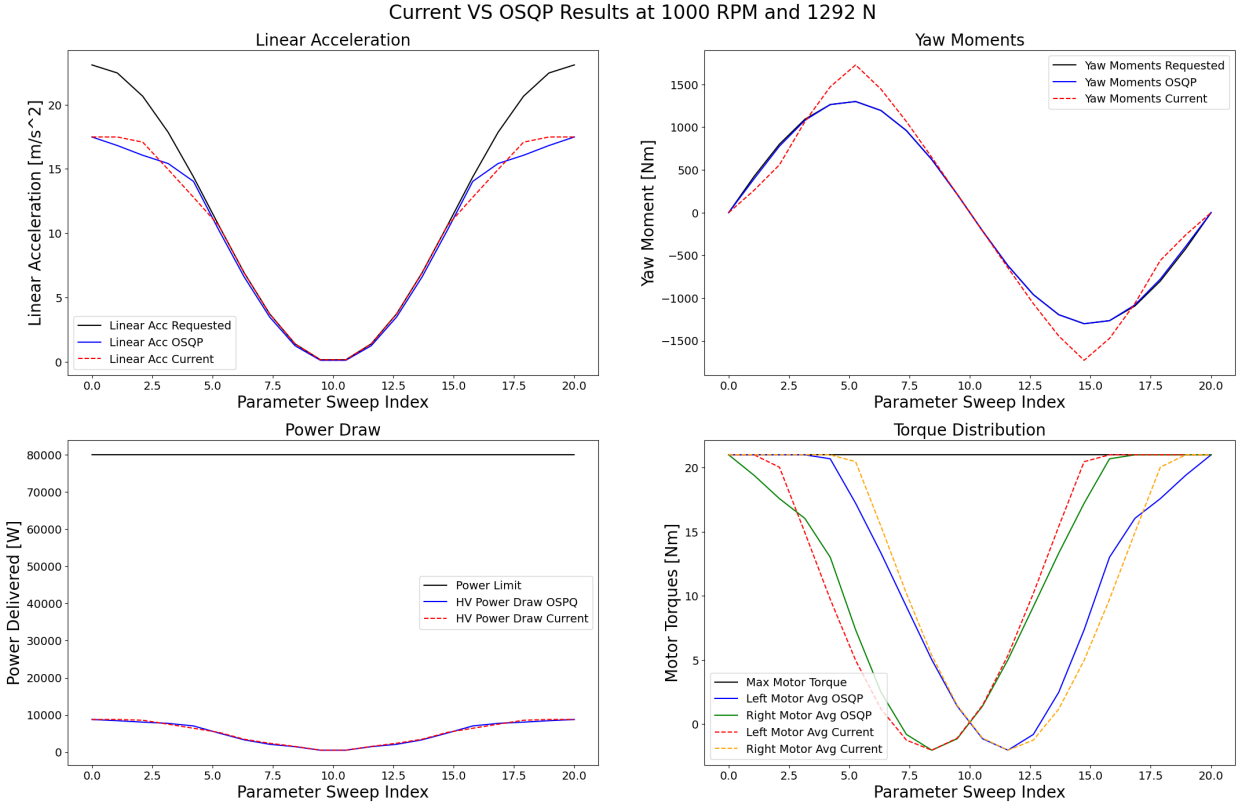


(b) Results at 9000 RPM

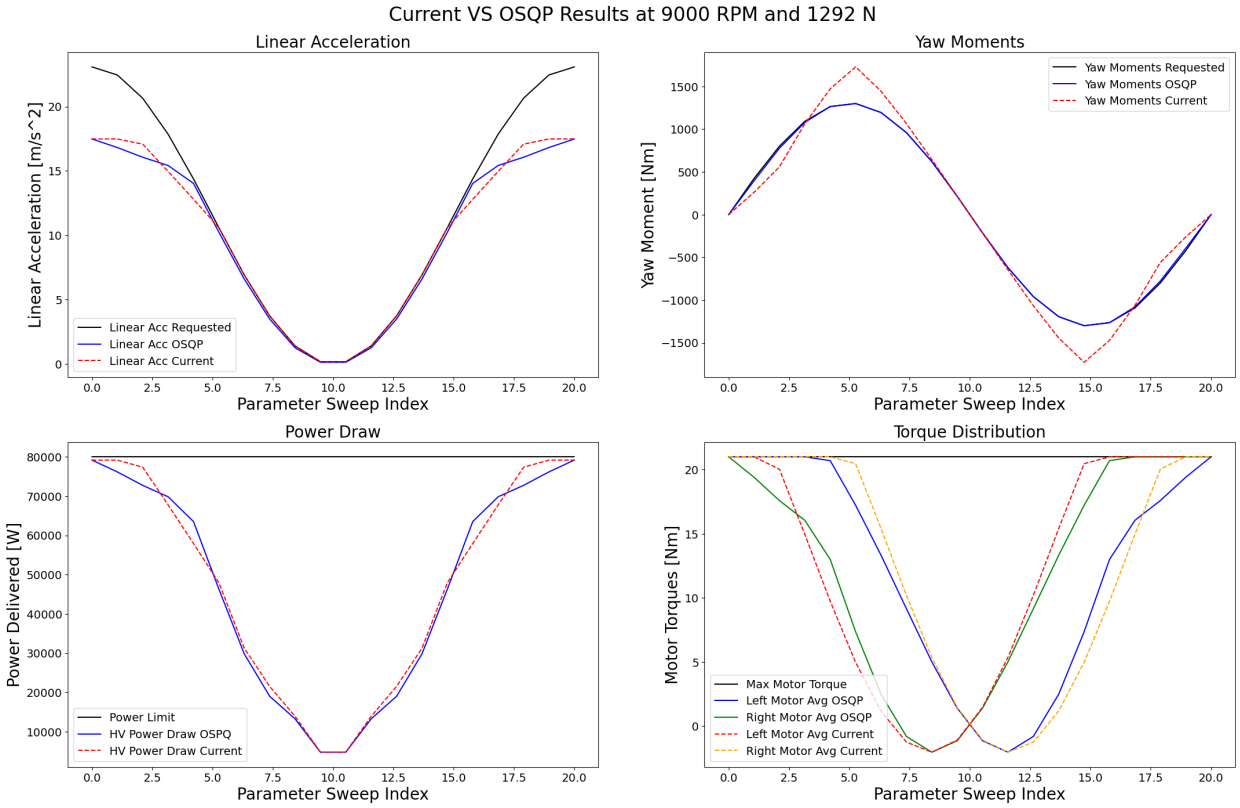


(c) Results at 19000 RPM

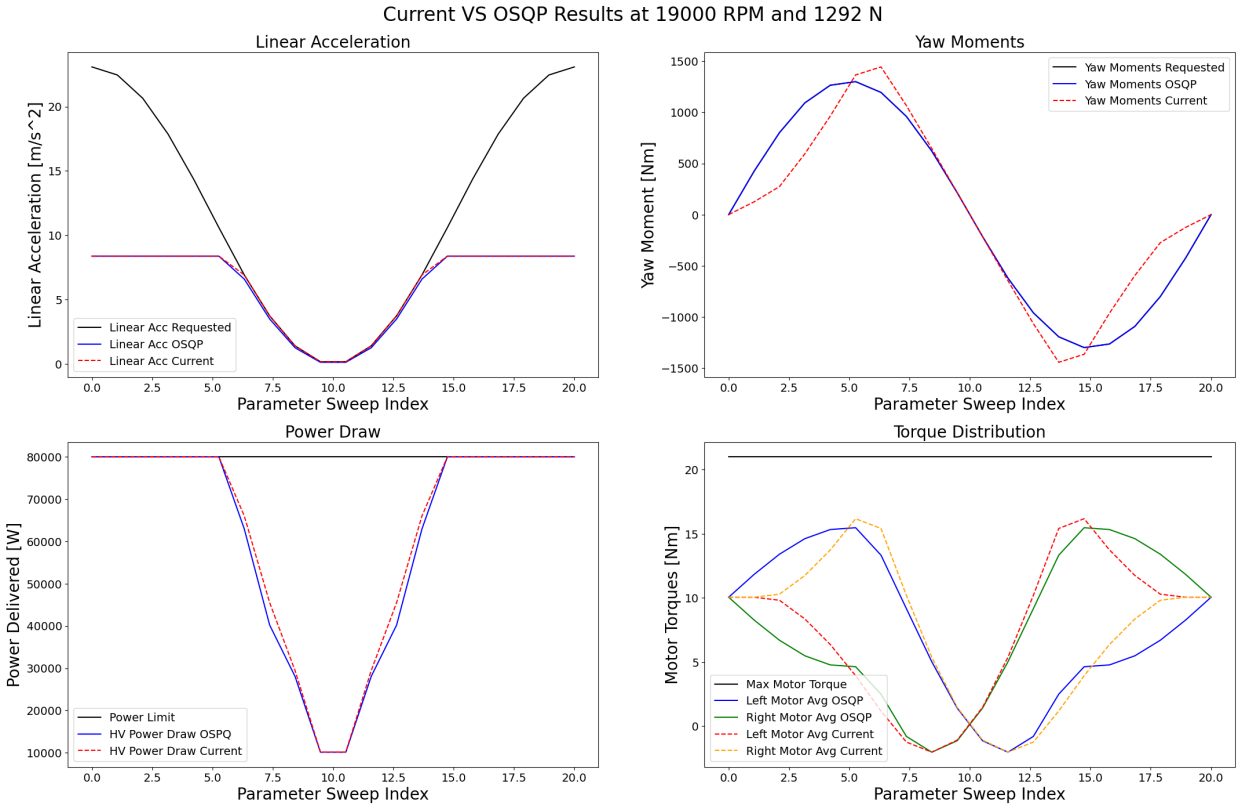
Fig. 4: Current Pipeline VS OSQP Pipeline Results with 790 N Downforce Per Wheel



(a) Results at 1000 RPM

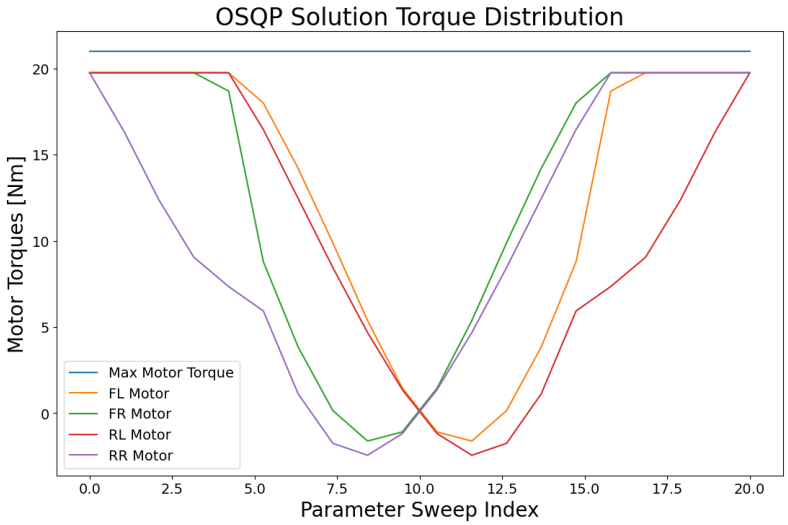


(b) Results at 9000 RPM

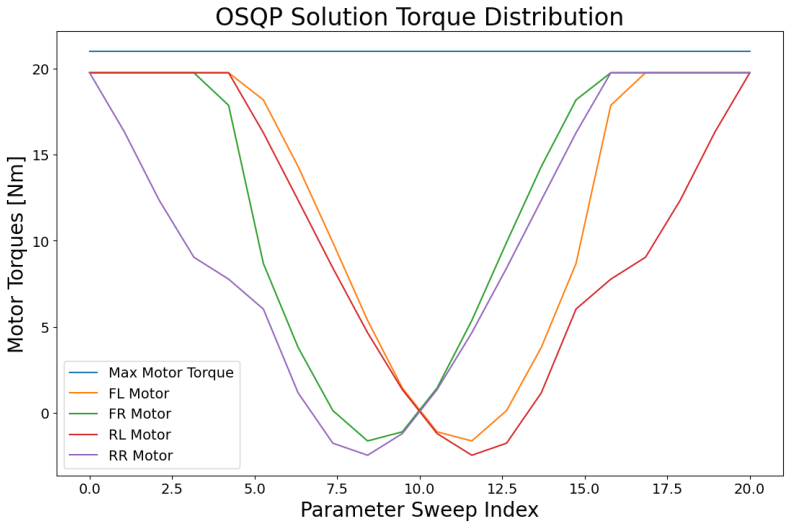


(c) Results at 19000 RPM

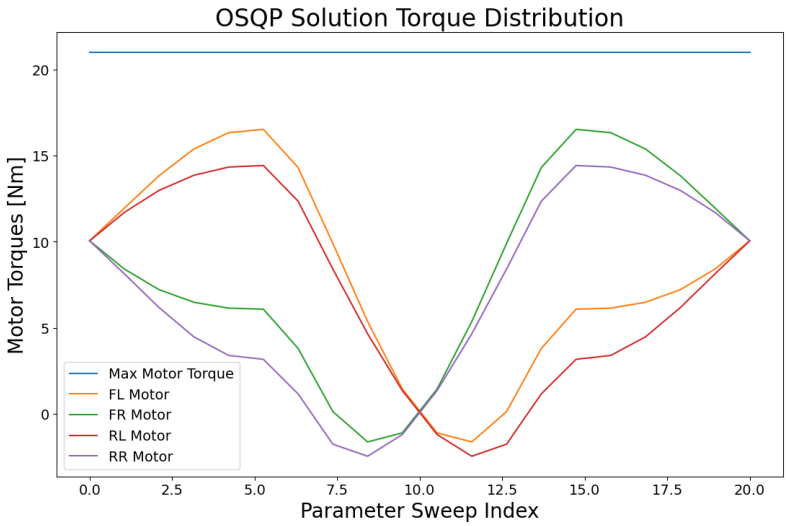
Fig. 5: Current Pipeline VS OSQP Pipeline Results with 1292 N Downforce Per Wheel



(a) Results at 1000 RPM

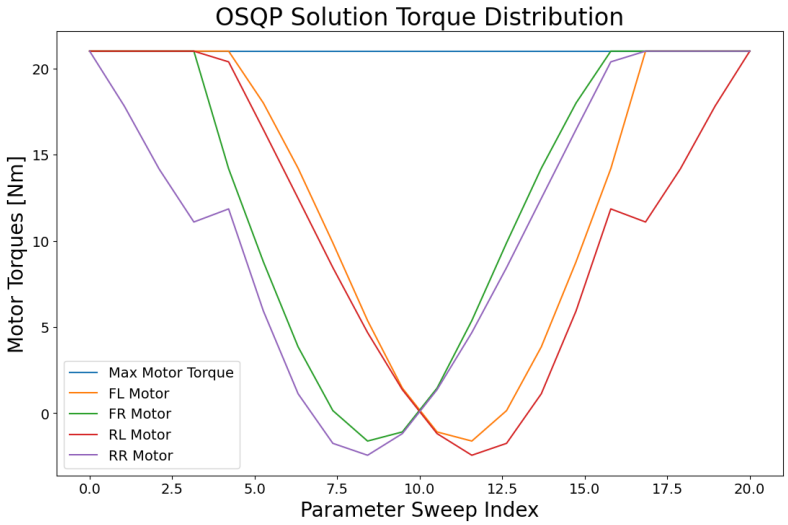


(b) Results at 9000 RPM

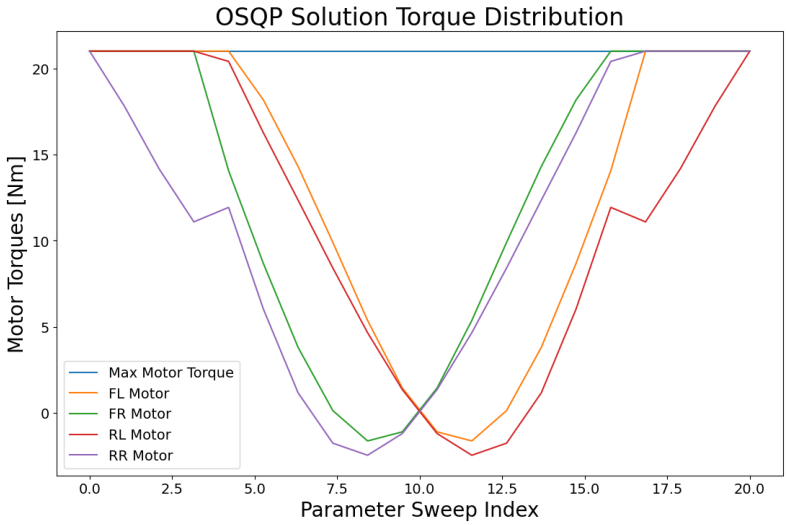


(c) Results at 19000 RPM

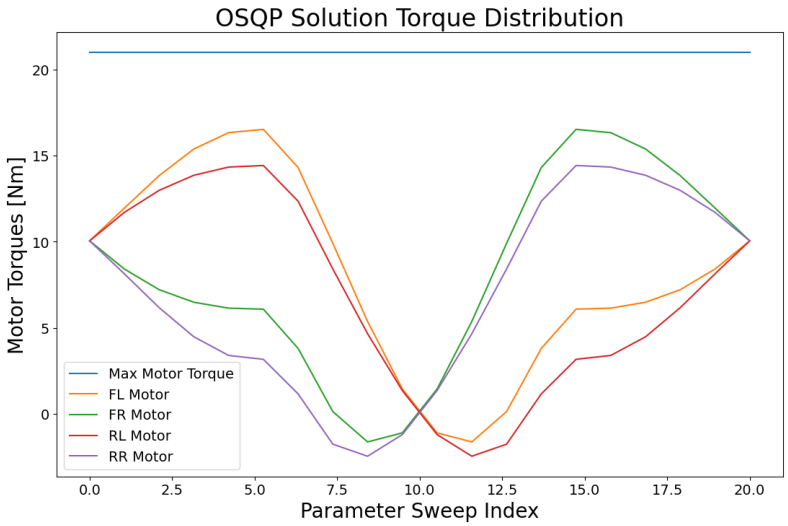
Fig. 6: OSQP Torque Vector Solution with 790 N Downforce Per Wheel



(a) Results at 1000 RPM



(b) Results at 9000 RPM



(c) Results at 19000 RPM

Fig. 7: OSQP Torque Vector Solution with 1292 N Downforce Per Wheel

IV. Discussion

A. Primary Discussion

The results suggest a significant boost in yaw moment realization with the OSQP solution over the current pipeline. In every evaluated condition, the OSQP solution yields a yaw moment realization curve (in blue) that is nearly identical to the yaw moment request. These errors are quantitatively measured in Tables I and II where a clear performance boost in Yaw Moment Error is realized. In the tested scenarios, the reduction in yaw moment error from the current pipeline to OSQP pipeline is 96%-99.9%.

Furthermore, it is worth highlighting that while the OSQP solution produces great yaw moment realizations, it additionally tracks the linear acceleration request and the total power drawn in nearly the same fashion as the current pipeline's solution. Tables I and II indicate a increase an error in linear acceleration realization of about 10%, which is acceptable given the improvement in yaw moment realization. The reasoning for this is that when the driver is turning, the primary goal of torque vectoring should be to meet the drivers yaw moment request so that they can take a tighter racing line, straighten the car, and get back on throttle sooner. In addition, because the net power consumption is the same for both pipelines, this indicates that OSQP is successfully producing a four-wheel torque vectoring solution that meets all constraints without drawing any additional power.

These result together with improvement in yaw moment realization indicate that the OSQP solution has a clear performance boost over the current pipeline.

B. Model Simplifications and Tuning

Although the model does utilize a few model simplifications – using zero tire slipangle when indexing into the traction control lookup table and using constant motor efficiency of one – these simplifications are applied to both the OSQP and current control pipeline implementations so bias is not introduced into the comparison. These model simplifications will be corrected in future work to produce more accurate results and be compared to track testing data for validation.

Furthermore,the weighting coefficients of the cost function can be tuned to achieve different request realization characteristics. The weights in this work were tuned to heavily prioritize yaw moment realization with minimal impact to linear acceleration realization.

V. Future Work

Our endeavor to optimize the Traction and Yaw Control System for our Formula SAE race car has yielded promising results, yet there are several avenues for future exploration and enhancement.

A. Adaptive Control Strategies

While our current system excels under nominal conditions, exploring adaptive control strategies could further enhance performance across a broader range of scenarios. Investigating methods that dynamically adjust controller parameters based on varying track conditions or unexpected disturbances can contribute to improved adaptability and responsiveness.

B. Machine Learning Integration

Introducing machine learning algorithms to the control framework presents an exciting opportunity. Training models on historical race data could enable the system to learn optimal control strategies, enhancing overall performance. Additionally, machine learning techniques may aid in predicting and mitigating potential issues, such as tire wear or changing track conditions.

C. Multi-Objective Optimization

Expanding the optimization framework to consider multiple, possibly conflicting objectives can be pivotal. Incorporating objectives related to energy efficiency, tire wear minimization, or even driver comfort can lead to a more holistic and well-balanced control strategy. Implementing a multi-objective optimization approach could result in a Pareto-optimal set of solutions, providing a spectrum of trade-offs for different scenarios.

D. Hardware-in-the-Loop (HiL) Testing

Transitioning from simulation to real-world hardware testing is a critical step for validating the efficacy of our control system. Conducting extensive Hardware-in-the-Loop testing, where the control algorithm interacts with physical components in a simulated environment, will provide valuable insights and ensure the system's robustness in practical race conditions.

E. Fault Tolerance and Safety Mechanisms

Incorporating fault tolerance mechanisms and safety features is essential for the reliability of any control system. Future work should focus on developing algorithms that can detect and gracefully handle faults in sensors or actuators, ensuring the safety of both the vehicle and the driver.

F. Driver-in-the-Loop Simulation

To better understand the human-machine interaction in high-performance racing scenarios, implementing a Driver-in-the-Loop simulation can provide valuable feedback. This involves integrating a human driver into the simulation loop, allowing for the evaluation of how our control system interfaces with a driver's input and reactions.

G. Real-Time Implementation Optimization

As technology evolves, exploring opportunities to optimize the real-time implementation of our control system becomes relevant. Investigating hardware accelerators, parallel processing, or dedicated real-time platforms can reduce computation times, enhancing the responsiveness of the control system during operation.

By venturing into these future avenues, we aim to continuously refine and innovate our Traction and Yaw Control System, pushing the boundaries of performance and safety in Formula SAE racing.

VI. Conclusion

Overall, we have demonstrated that the OSQP pipeline is a better controls pipeline than the current one. It generates smoother results and can better reconcile the constraints such that the solution provided is optimized across all constraints. With this in mind, we can continue investigating the possible implementation of a quadratic programming controller for CMR's 24E vehicle and improve the performance of racecars to come.

VII. Code

Due to considerations aimed at preserving our competitive advantage, we regret that the source code for our research cannot be released publicly. However, we welcome inquiries and are open to providing access to interested parties. For code access requests or further collaboration, please feel free to reach out to the authors.

VIII. Acknowledgements

We would like to thank Thomas Liao from Carnegie Mellon Racing for his guidance in this project.

References

[1] Stellato, Bartolomeo, Banjac, Goran, Goulart, Paul, Bemporad, Alberto, Boyd, Stephen. (2018). OSQP: An Operator Splitting Solver for Quadratic Programs. 339-339. 10.1109/CONTROL.2018.8516834.

[2] H. Fujimoto and K. Maeda, "Optimal yaw-rate control for electric vehicles with active front-rear steering and four-wheel driving-braking force distribution," IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society, Vienna, Austria, 2013, pp. 6514-6519, doi: 10.1109/IECON.2013.6700209.

[3] Yang K, Dong D, Ma C, Tian Z, Chang Y, Wang G. Stability Control for Electric Vehicles with Four In-Wheel-Motors Based on Sideslip Angle. World Electric Vehicle Journal. 2021; 12(1):42. https://doi.org/10.3390/wevj12010042.

[4] Smith, Edward, Velenis, Efstathios, Cao, Dongpu, Tavernini, Davide. (2016). Evaluation of Optimal Yaw Rate Reference for Electric Vehicle Torque Vectoring.