Differential Drive Module Design

Dennis Slobodzian

April 28, 2023

1 Introduction

This report discusses the code for a simulation of a differentially coupled drive module controlled by a linear quadratic regulator (LQR) controller. The system is modeled as a linear rotational-dynamics system, and the LQR controller is designed to stabilize the module and track a sequence of desired steering-angle and wheel-speed setpoints. This report first discusses the derivation of the dynamic model, followed by the design of the LQR controller. Finally, it presents and analyzes the simulation results, root-locus plots, and closed-loop eigenvalues.

The simulation is a module-level model. It does not include the position or global heading of a full robot. The inputs are commanded motor currents, so the model assumes an ideal or sufficiently fast inner current-control loop.

2 Derivation of Dynamic Model

The differential drive module uses two motors to control steering and wheel motion. The common and differential components of the motor motion can be described conceptually by

$$\frac{\omega_{\text{left}} + \omega_{\text{right}}}{2} = \omega_{\text{steer}}, \tag{1}$$

$$\frac{\omega_{\text{left}} - \omega_{\text{right}}}{2} = \omega_{\text{wheel}}. \tag{2}$$

Here, ω_{left} and ω_{right} represent appropriately scaled left and right motor speeds, ω_{steer} is the steering-axis angular velocity, and ω_{wheel} is the wheel angular velocity. These relationships describe the common-mode and differential-mode behavior of the module.

The dynamic model used in the simulation assumes two equivalent disk inertias driven by two motors. The system parameters are the wheel inertia J_w, steering inertia J_s, steering gear ratio k_s, wheel gear ratio k_w, motor torque constant k_t, steering and wheel friction coefficients, and wheel radius r_w.

The state vector is

$$x = \begin{bmatrix} \theta_{\text{steer}} \\ \omega_{\text{steer}} \\ \omega_{\text{wheel}} \end{bmatrix}, \tag{3}$$

where θ_{steer} is azimuth angle, ω_{steer} is azimuth angular velocity, and ω_{wheel} is wheel angular velocity. The third state is in rad/s; its linear wheel speed is computed only for plotting as

$$v_{\text{wheel}} = r_w \omega_{\text{wheel}}. \tag{4}$$

The input vector is

$$u = \begin{bmatrix} i_{\text{left}} \\ i_{\text{right}} \end{bmatrix}, \tag{5}$$

where i_{left} and i_{right} are the commanded left and right motor currents in amperes. The motor torque is modeled by $\tau_m = k_t i$. Therefore, equal motor currents create steering torque and equal-and-opposite motor currents create wheel torque:

$$\tau_{\text{steer}} = k_t k_s \left(i_{\text{left}} + i_{\text{right}} \right), \tag{6}$$

$$\tau_{\text{wheel}} = k_t k_w \left(i_{\text{left}} - i_{\text{right}} \right). \tag{7}$$

The resulting dynamic model is represented by the state-space equations

$$\dot{x} = Ax + Bu, \tag{8}$$

where A is the state matrix and B is the current-input matrix:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{\text{friction}_{\text{steer}}}{J_s} & 0 \\ 0 & 0 & -\frac{\text{friction}_{\text{wheel}}}{J_w} \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 0 \\ \frac{k_t k_s}{J_s} & \frac{k_t k_s}{J_s} \\ \frac{k_t k_w}{J_w} & -\frac{k_t k_w}{J_w} \end{bmatrix}. \tag{9}$$

The wheel-dynamics row uses J_w, not J_s, because wheel torque accelerates the wheel inertia. In the present simulation J_s and J_w are numerically equal, so the original typo did not change the numerical result; however, J_w is the physically correct denominator.

Table 1: Simulation parameters.

Parameter	Value	Description
J_w	0.001 kg m^2	Wheel inertia
J_s	0.001 kg m^2	Steering inertia
k_w	6.46875/1.2	Motor-to-wheel torque ratio
k_s	9.2/1.2	Motor-to-steering torque ratio
k_t	5.84/304 N m/A	Motor torque constant
r_w	0.04615 m	Wheel radius
$i_{\max}$	120 A	Per-motor current limit
Δt	0.01 s	Simulation time step
T	5.0 s	Total simulation time

2.1 Controllability

The controllability matrix is

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B \end{bmatrix}. \tag{10}$$

The MATLAB calculation gives $\text{rank}(\mathcal{C}) = 3$, which is equal to the number of states. Therefore, the linear model is fully controllable.

3 Benefits and Issues with the Differential Drive Module

The differential drive system used in this simulation has several benefits. One significant advantage is that it can provide high power output, making it useful for applications such as robotic vehicles. The system is also relatively simple to model because the sum and difference of the two motor currents separate steering and wheel torque in the ideal symmetric case.

There are also potential issues with differential drive systems. The motors can fight each other when there are differences in motor constants, gearbox efficiency, sensor calibration, or mechanical compliance. That behavior is not present in the ideal symmetric simulation, but it can reduce efficiency and degrade performance on hardware. The system can also be difficult to control at high speeds because wheels can slip and lose traction. Tire slip, compliance, backlash, and chassis dynamics are not included in this model.

4 LQR Controller Design

In this code, the LQR controller design technique is used to design a controller for the system. LQR minimizes a cost function that contains both state error and control input:

$$J = \int_0^\infty \left[(x - x_{\text{des}})^T Q (x - x_{\text{des}}) + u^T R u \right] dt. \tag{11}$$

The matrices Q and R determine the relative importance of state error and motor-current effort. The values are chosen empirically based on desired performance and the physical current limit. The state cost matrix and control cost matrix used in the simulation are

$$Q = \text{diag}\left(\frac{1}{0.15^2}, \frac{1}{2.0^2}, \frac{1}{2.0^2} \right) = \text{diag}(44.4444, 0.25, 0.25), \tag{12}$$

$$R = \text{diag}(1, 1). \tag{13}$$

The steering-angle error is weighted much more heavily than steering-rate and wheel-speed error. The two velocity states have equal weights, and the controller penalizes the left and right motor-current commands equally.

MATLAB computes the LQR gain matrix using `lqr()`:

$$K = \begin{bmatrix} 4.7140 & 0.3962 & 0.3536 \\ 4.7140 & 0.3962 & -0.3536 \end{bmatrix}. \tag{14}$$

The control input at each time step is calculated as

$$u = -K(x - x_{\text{des}}). \tag{15}$$

Each current command is independently clamped to ± 120 A to represent a physical motor-current limit.

4.1 Root-Locus Interpretation

The original plant has two inputs, so a single classical root locus cannot be drawn directly for the full system. To create meaningful root-locus plots, the motor currents are transformed into common-mode steering current and differential wheel current:

$$i_{\text{steer}} = \frac{i_{\text{left}} + i_{\text{right}}}{2}, \qquad i_{\text{wheel}} = \frac{i_{\text{left}} - i_{\text{right}}}{2}. \tag{16}$$

This produces a steering channel and a wheel-speed channel. Figure 1 shows MATLAB root-locus plots for these channels. The red markers and numeric annotations show the operating poles at the implemented LQR gain ($\gamma = 1$). The steering common-mode channel has poles at -13.45 and -103.27 s^{-1}, while the wheel differential-current channel has a pole at -73.23 s^{-1}. Together, these modal poles reproduce the eigenvalues of the full two-input closed-loop system. The plots provide a SISO interpretation of the two modes; the implemented controller remains a two-input LQR controller.

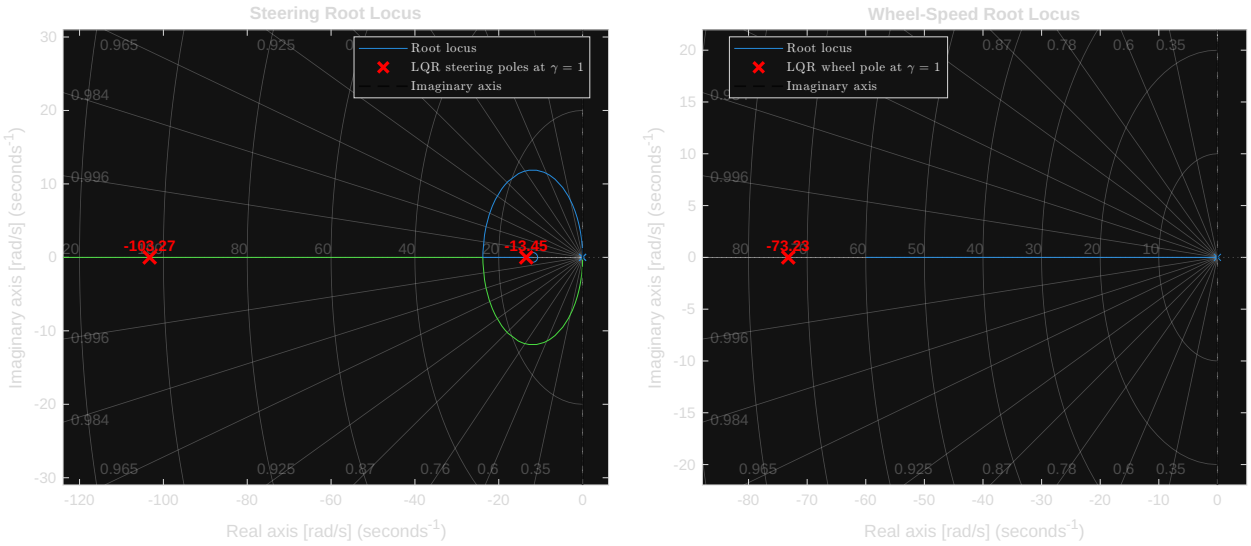

Figure 1: MATLAB root-locus plots for the steering common-mode and wheel differential-current channels. Red markers and labels show the modal closed-loop poles at the implemented LQR gain ($\gamma = 1$): steering poles at -13.45 and -103.27 s^{-1}, and the wheel pole at -73.23 s^{-1}.

4.2 Closed-Loop Stability

The closed-loop system matrix is

$$A_{\mathrm{cl}} = A - BK. \tag{17}$$

For this model, the closed-loop eigenvalues are

$$\lambda(A_{\mathrm{cl}}) = \{-13.4459,\ -73.2257,\ -103.2715\}\ \mathrm{s}^{-1}. \tag{18}$$

The modal interpretation above assigns the -13.4459 and -103.2715 s^{-1} poles to steering dynamics and the -73.2257 s^{-1} pole to wheel-speed dynamics. All three eigenvalues are real and negative. Therefore, all closed-loop poles lie in the left-half plane, and the unsaturated continuous-time linear system is asymptotically stable. Figure 2 shows the MATLAB pole plot and the real parts of the closed-loop eigenvalues.

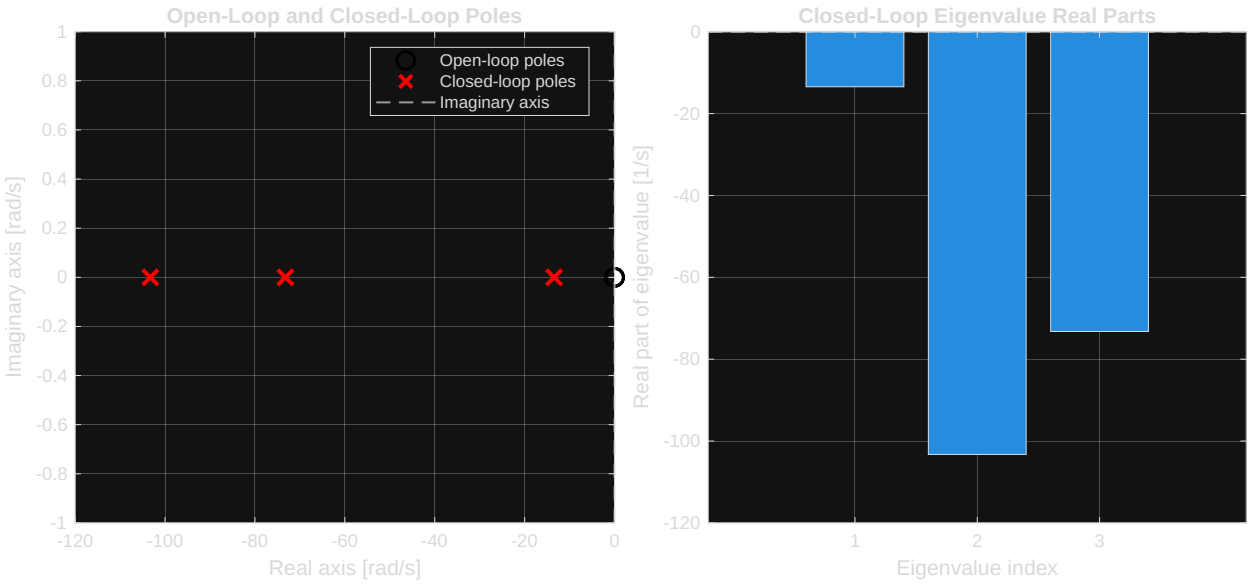

Figure 2: MATLAB open-loop and closed-loop pole plot. All closed-loop eigenvalues have negative real parts, which establishes stability for the unsaturated linear model.

5 Simulation Loop

In the simulation loop, the control input is calculated at each time step using the LQR controller gain matrix K. The motor-current command is clamped to a maximum value so that neither motor command exceeds the specified current limit. The initial state is zero steering angle, zero steering rate, and zero wheel angular velocity.

The desired state changes at different time intervals. At the beginning of the simulation, the desired state is an azimuth angle of π radians, zero azimuth angular velocity, and a wheel angular velocity of 30 rad/s. This wheel state corresponds to approximately 1.38 m/s after multiplication by the wheel radius. After 25% of the simulation time, the desired state changes to an azimuth angle of $-\pi/2$ radians and wheel angular velocity of -20 rad/s. After 50% of the simulation time, the desired wheel angular velocity is 100 rad/s. After 75% of the simulation time, it changes to -100 rad/s. The steering angular-velocity reference remains zero.

After the control input is calculated, the state is updated with forward Euler integration:

$$x_{k+1} = x_k + \Delta t \left(A x_k + B u_k \right). \tag{19}$$

The state and control input arrays are updated until the end of the simulation time.

6 Results

The MATLAB results consist of plots for azimuth angle, wheel velocity, and motor-current inputs. Figure 3 shows the azimuth angle and azimuth-angle goal over time, the wheel velocity and wheel-velocity goal, and the left and right current commands.

The simulation shows that the system tracks the desired azimuth angle and wheel-speed references. The response can show transient overshoot and oscillation following setpoint changes, especially because the references are discontinuous. The current commands remain within the specified current limit. Since friction and external load are both zero in the simulation, the commanded current

returns near zero after a setpoint is reached. A physical module would require nonzero steady-state effort to overcome losses and disturbances.

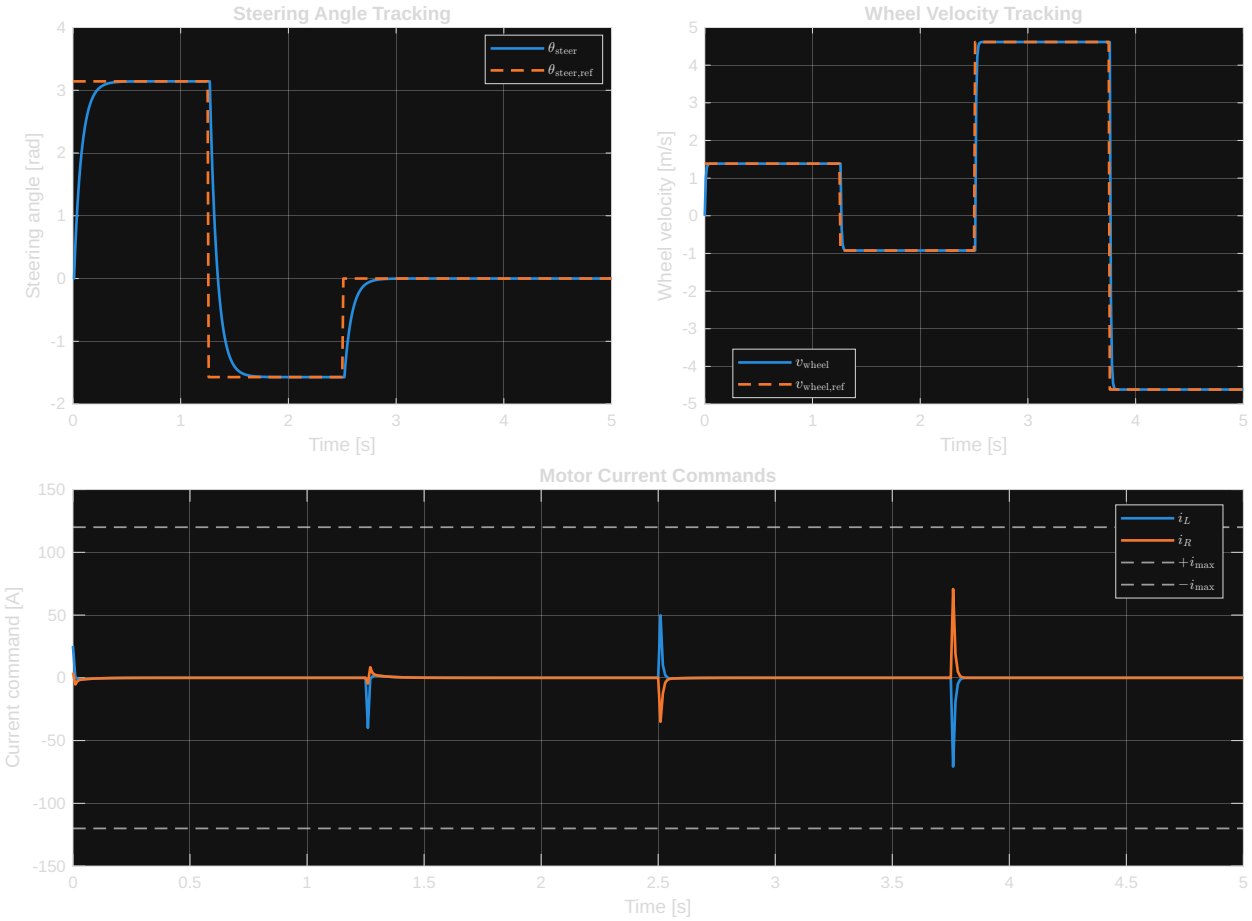

Figure 3: MATLAB simulation results. The top plots show steering-angle and wheel-speed tracking. The bottom plot shows the left and right commanded motor currents. Dashed lines indicate references.

7 Conclusion

In conclusion, the code provides a simulation of a differentially coupled drive module using an LQR controller. The dynamic model assumes two equivalent inertias driven by motors and uses current as the control input. The simulation demonstrates the effectiveness of the LQR controller in controlling steering angle and wheel angular velocity toward desired states. It also highlights the importance of a fully controllable system for LQR control.

The code provides a basic implementation of an LQR controller, and further improvements can be made to the controller and model. A state estimator such as a Kalman filter would be useful in noisy environments where sensor measurements are unreliable. The simulation can also be extended to include traction limits, wheel slip, external disturbances, gear losses, backlash, chassis dynamics, and a voltage-input motor model with back-EMF, winding resistance, inductance, and inverter limits. These additions would support later evaluation of nonlinear controllers such as sliding-mode control or model predictive control. Overall, the code provides a useful starting point for simulating a differential drive module and implementing an LQR controller for current-based steering and wheel-speed control.