February 1, 2023

Differential Swerve State Space Controller Design

Henry LeCompte
FRC Team 2383, The Ninjineers

Contents

1 Introduction

1.1 What is Swerve?

Swerve is a type of drivetrain that is characterized by having at least two wheels that can each rotate and spin independently of each other (here rotate is define as a rotation around the robots up axis and spin is defined as a rotation around the modules y axis). This allows the robot to move in any direction and turn in place. Differential Swerve is a type of swerve drivetrain in which the rotation of the wheel is controlled by the difference of the motor velocities and the spin of the wheel is controlled by the average of the motor velocities. This allows for the combination of the torque produced by both motors when driving and turning. This is the main advantage over conventional (coaxial) swerve drivetrains in which rotation and spin are controlled by separate motors. The main disadvantage of differential swerve is that it is significantly more complex to control.

1.2 Purpose of this paper

By reading this paper you should be able to gain an understanding of how a relatively complex mechanism is analyzed and a state space model is created. You should then learn the steps to implement a simple Linear Quadratic Regulator (LQR) and then a Kalman Filter to filter out measurement noise.

1.3 Who should read this paper

This paper was designed for High School students currently competing in First Robotics Competition (FRC) although nothing in this paper is specific to this competition, swerve is commonly used in this competition and it provides many tools for state-space controls.

2 Designing the State Space Model

We need to design a continuous, time-invariant state space model of the system. This will allow us to estimate the response of the system to create a feedforward and feedback controller. The following is the general form of a state space model.

$$\dot{x} = Ax + Bu \tag{1}$$

$$y = Cx + Du \tag{2}$$

What should we choose as our inputs, outputs, and states?

Inputs

In robotics you can commonly only input one value, voltage, but because differential swerve has two motors that work together to control the module we can pass in two inputs, the top motor voltage and the bottom motor voltage. In differential swerve we refer to the motors as top and bottom because there is one motor that drives the top gear of the differential and there is one motor that drives the bottom gear of the differential not because one of the motors is physically on top of the other.

Outputs

Outputs in a state-space model are things that you can physically measure. Although, this makes it a little confusing as to what we should chose because there are many different things we can measure with each encoder (displacement, velocity, acceleration) and we have 3 of them. To figure out what to output we need to think about which of these is it actually important for us to measure. In swerve there are really two things that we need to know: the velocity of the wheel, and the rotation of the wheel. Even though it would be nice to have these as outputs we only have a way of directly measuring one of these with the module rotation encoder. For the wheel velocity we can instead output the velocity of both motors, we can then calculate the wheel velocity as the average of the motor speeds. So we are going to have three outputs: top motor velocity [radians/second], bottom motor velocity [radians/second], and wheel rotation [radians]

States

To calculate the change in the state of the system we can only use the states and the inputs. We might be able to see that we can calculate how the output variables will change just by knowing the current outputs and the inputs, this means that we can use these as the states. We now have the same states as outputs which means that our C matrix will be the identity matrix and our D matrix will be 0.

2.1 Inputs, Outputs, and States

The States will be in the form of

$$x = \begin{bmatrix} v_t \\ v_b \\ \theta \end{bmatrix} \tag{3}$$

Where θ is the angle of the wheel and v_t and v_b are the top and bottom motor velocities respectively.

The velocity of the wheel is not included as a state because it is a linear combination of the motor velocities so it can be calculated from the current state.

The inputs will be in the form of

$$u = \begin{bmatrix} V_t \\ V_b \end{bmatrix} \tag{4}$$

Where V_t is the top motor voltage and V_b is the bottom motor voltage.

Our output matrix is the same as the state matrix as we can measure all states.

$$y = \begin{bmatrix} v_t \\ v_b \\ \theta \end{bmatrix} \tag{5}$$

Where v_t is the top motor velocity, v_b is the bottom motor velocity, and θ is the angle of the wheel.

2.2 DC Motor Model

We know that a permanent magnet DC motor follows the general equation of

$$V = K_v \dot{x} + K_a \ddot{x} \tag{6}$$

And we can rewrite this as

$$\begin{aligned} V &= K_v\dot{x} + K_a\ddot{x} \\ V - K_a\ddot{x} &= K_v\dot{x} \\ -K_a\ddot{x} &= K_v\dot{x} - V \\ \ddot{x} &= \frac{-K_v\dot{x} + V}{K_a} \\ \ddot{x} &= \frac{-K_v\dot{x}}{K_a} + \frac{V}{K_a} \end{aligned}$$

We can also substitute v as $\dot{x}$ to create

$$\dot{v} = \frac{-K_v v}{K_a} + \frac{V}{K_a} \tag{7}$$

This equation can then be written in state space form as

$$\dot{x} = \left[\frac{-K_v}{K_a}\right] x + \left[\frac{1}{K_a}\right] u \tag{8}$$

$$y = 1x + 0u \tag{9}$$

2.3 A and B Matrices

Now that we know how to calculate the angular velocity of a motor based on its constants and the input voltage we can start to create the formulas that describe the changes in state of a swerve module.

$$\dot{v_t} = \frac{-K_{v_{drive}}}{K_{a_{drive}}} v_t + \frac{1}{K_{a_{drive}}} V_t \tag{10}$$

$$\dot{v_b} = \frac{-K_{v_{drive}}}{K_{a_{drive}}} v_b + \frac{1}{K_{a_{drive}}} V_b \tag{11}$$

$$\dot{\theta} = \frac{v_t + v_b}{2} * K_{turn\_ratio} \tag{12}$$

We can now rewrite these equations as a system that has coefficients for every state and input.

$$\dot{v_t} = \frac{-K_{v_{drive}}}{K_{a_{drive}}} v_t + 0v_b + 0\theta + \frac{1}{K_{a_{drive}}} V_t + 0V_b \tag{13}$$

$$\dot{v_b} = 0v_t + \frac{-K_{v_{drive}}}{K_{a_{drive}}} v_b + 0\theta + 0V_t + \frac{1}{K_{a_{drive}}} V_b \tag{14}$$

$$\dot{\theta} = \frac{K_{turn\_ratio}}{2} v_t + \frac{K_{turn\_ratio}}{2} v_b + 0\theta + 0V_t + 0V_b \tag{15}$$

Now that we have these linear equations we can turn them into the A and B matrix.

$$A = \begin{bmatrix} \frac{-K_{v_{drive}}}{K_{a_{drive}}} & 0 & 0 \\ 0 & \frac{-K_{v_{drive}}}{K_{a_{drive}}} & 0 \\ \frac{K_{turn\_ratio}}{2} & \frac{K_{turn\_ratio}}{2} & 0 \end{bmatrix} \tag{16}$$

$$B = \begin{bmatrix} \frac{1}{K_{a_{drive}}} & 0 \\ 0 & \frac{1}{K_{a_{drive}}} \\ 0 & 0 \end{bmatrix} \tag{17}$$

2.4 C and D matrices

With these, we just need the C and D matrices. Because the output matrix is the same as the state matrix C is just the identity matrix and D = 0.

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{18}$$

$$D = 0 \tag{19}$$

2.5 Full model

And now will all of these matrices we can write the full continuous model as

$$\begin{bmatrix} \dot{v_t} \\ \dot{v_b} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{-K_{v_{drive}}}{K_{a_{drive}}} & 0 & 0 \\ 0 & \frac{-K_{v_{drive}}}{K_{a_{drive}}} & 0 \\ \frac{K_{turn\_ratio}}{2} & \frac{K_{turn\_ratio}}{2} & 0 \end{bmatrix} \begin{bmatrix} v_t \\ v_b \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{K_{a_{drive}}} & 0 \\ 0 & \frac{1}{K_{a_{drive}}} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_t \\ V_b \end{bmatrix} \tag{20}$$

$$\begin{bmatrix} v_t \\ v_b \\ \theta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_t \\ v_b \\ \theta \end{bmatrix} + 0 \begin{bmatrix} V_t \\ V_b \end{bmatrix} \tag{21}$$

Now we can also add the following constants to the model (These will be different for each robot)

$$K_{v_{drive}} = 0.023 \tag{22}$$

$$K_{a_{drive}} = 0.001 \tag{23}$$

$$K_{turn\_ratio} = \frac{1}{28} \tag{24}$$

3 Validating the State Space Model

Now that we have a state space model we need to validate that it is controllable, observable and stable.

3.1 Controllability

Controllability is a property of the system that allows any initial state to be driven to any desired state using control inputs in a finite time. The Controllability matrix for continuous time-invariant systems is defined as

$$R = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \tag{25}$$

And the system is said to be controllable if the rank of the controllability matrix is equal to the number of states (n).

$$rank(R) = n \tag{26}$$

We can now calculate the rank of the controllability matrix for our system and find that it is 3. This means that out system is controllable and we can move on to the next step.

3.2 Observability

A system is said to be observable if, for every possible state, the current state can be estimated using only the information from outputs. The observability matrix for continuous time-invariant systems is defined as

$$O = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{27}$$

And the system is said to be observable if the rank of the observability matrix is equal to the number of states (n).

$$rank(O) = n \tag{28}$$

Calculating this for our current system we find that the rank is 3. This means that our system is observable and we can move on to the next step.

3.3 Stability

A system is said to be stable if the eigenvalues of the A matrix are all negative. Calculating the eigenvalues of the A matrix we can find that the real parts of the eigenvalues are 0 and -23. This means that our system is marginally stable but we can fix this with the LQR controller in the next section.

3.4 Step Response

To validate that our system is stable we can also look at the step response of the system. The step response is the response of the system to a step input. This can be calculated with the python package control. We can graph the step response values for the system and find that the system is indeed stable. This graph can be seen in figure 1.

3.5 Forced Response

We also want to make sure that the wheel is turning correctly and is still stable with different inputs. To do this we can create a graph of forced inputs. This graph uses 5 time steps of 12 volts on each motor. 495 more time steps of 12 and -12 volts and 500 more time steps of no voltage. This graph can be seen in figure 2.

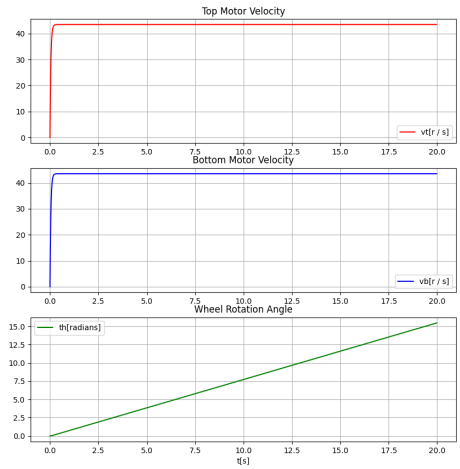

Figure 1: Step response of the system

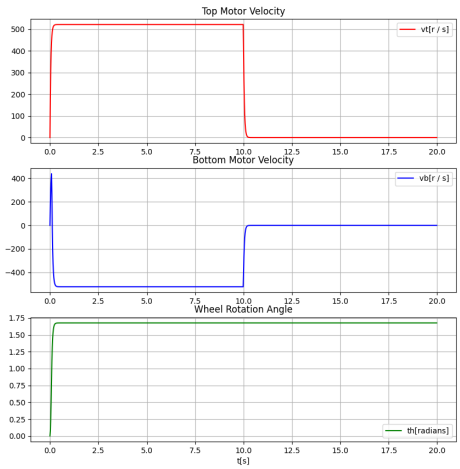

Figure 2: Forced response of the system

4 Controller Design

Now that we have a state space model that accurately describes our plant we can start to design a controller to allow the modules to track a reference velocity and rotation. Here we don't have very strict requirements other than minimal oscillation and a fast response time. For this reference tracking control we need a feedforward controller to calculate the plant inputs for a specific reference and we need a feedback controller to account for un-modeled dynamics. We should also use a state estimator to reduce the impact of measurement noise on the system: for this we will use a Kalman filter.

4.1 Feedforward Controller

The feedforward controller we will be using is Plant Inversion. This controller is a linear controller that uses the state space model to calculate the inputs to the system to achieve a desired state. The controller is defined as

$$u_k = B^+(r_{k+1} - Ar_k) \tag{29}$$

Where B^+ is the Moore-Penrose pseudoinverse of the B matrix, r_k is the reference at the current timestep, and r_{k+1} is the reference at the next timestep.

A simple system could use this controller to track the reference. For example we can pass in the current state that we get from our sensors and the next reference which is what we want they system to do. This will then give us inputs which we can pass directly to the motors. The issue with this approach is that we don't have a way to account for un-modeled dynamics. This means that the system will not be able to track the reference perfectly. To account for this we add a feedback controller.

4.2 Feedback Controller

The feedback controller we will be using is the Linear Quadratic Regulator (LQR). The LQR controller works by minimizing a quadratic cost function defined by the equation

$$J = \sum_{k=0}^{\infty}(Q(r_k - x_k)^2 + Ru_k^2) \tag{30}$$

The goal is to minimize this cost function while obeying the system dynamics. To create the Q and R matrices we use Bryson's rule to create the matrices in the form of

$$Q = \begin{bmatrix} \frac{1}{x_{1,max}^2} & 0 & \dots & 0 \\ 0 & \frac{1}{x_{2,max}^2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{x_{n,max}^2} \end{bmatrix} R = \begin{bmatrix} \frac{1}{u_{1,max}^2} & 0 & \dots & 0 \\ 0 & \frac{1}{u_{2,max}^2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{u_{n,max}^2} \end{bmatrix} \tag{31}$$

We can now define the maximum state errors as $x = [0.1, 0.1, 0.01]$ and the maximum input values as $u = [12, 12]$. Using this you can now solve for the K matrix which is the feedback gain matrix.

4.3 Kalman Filter

The Kalman filter is a state estimator that uses the system dynamics and the sensor measurements to estimate the current state of the system. Because our sensor measurements are noisy approximations of the system state we need to use a state estimator to reduce the impact of this noise. The Kalman filter uses the system dynamics to predict the next state and then uses the sensor measurements to correct the prediction.

The Kalman filter uses the state and measurement standard deviations which we can estimate as $[0.1, 0.1, 0.1]$ for the state and $[0.01, 0.01, 0.01]$ for the measurement.

4.4 Putting it all together

Now that we have a controller and a state estimator we can put them together to create a controller that can track a reference. To do this we go through a loop of setting the new reference, correcting the state estimation, predicting the next state, and calculating the new inputs.