1 Overview

The problem of autonomous driving can be divided into four main components, each representing a crucial aspect of the system.

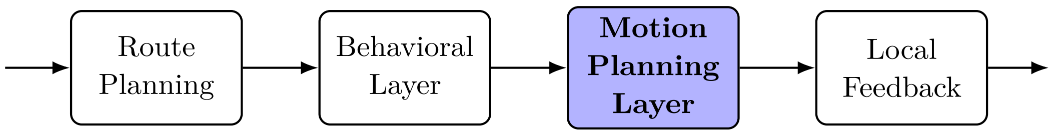


Figure 1.1: Overview of Autonomous Driving Problem Decomposition

The process begins with the user providing a travel destination, which serves as the input to the first component, Route Planning. In this phase, the system generates a sequence of waypoints through a predefined road network.

Next, the Behavioral Layer refines the waypoints by considering environmental factors such as other vehicles, obstacles, and road signs. This layer defines the driving behavior at any given time, ensuring the vehicle adapts to dynamic traffic conditions.

With a behavior strategy in place, the Motion Planning Layer generates a trajectory that adheres to strict physical and safety constraints, ensuring feasibility and compliance with rules of the road.

Finally, the Local Feedback component executes the plan by generating precise control commands—steering, throttle, and brake inputs—based on real-time vehicle and environmental feedback.

1.1 Motion Planning

Finding an exact solution to the motion planning problem is computationally intractable in most cases. Therefore, numerical approaches are employed. These methods fall into three main categories:

1. Graph-Based Algorithms: These algorithms discretize the vehicle's possible states and connect valid state transitions with edges. A graph search algorithm can then be used to find an optimal trajectory.

2. Incremental Tree Approaches: This category involves generating branches of feasible trajectories by randomly applying control commands and simulating the resulting states. The tree expands incrementally, exploring potential paths until a suitable trajectory is found.
3. Optimization-Based Methods: These methods formulate the problem as an optimization task over a function space. Optimization techniques aim to minimize an objective function (e.g., minimizing travel time or maximizing safety) while respecting constraints. This is the approach we focus on.

Our objective is to design a motion planner that consistently provides near real-time solutions. We achieve this by leveraging modern, reliable solvers that can efficiently handle the optimization problem.

1.2 Convex Optimization

A set $K \subset \mathbb{R}^n$ is called convex if, for all $x, y \in K$ and $\lambda \in [0, 1]$, the following condition holds

$$\lambda x + (1 - \lambda)y \in K \tag{1.1}$$

A real-valued function f defined over a convex subset X of a vector space is called convex if, for all $x, y \in X$ and $\lambda \in [0, 1]$, the inequality below is satisfied:

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) \tag{1.2}$$

A function f is said to be concave if $-f$ is convex.

An optimization problem is defined by a feasible set $X \subset \mathbb{R}^n$ and an objective function $f : X \to \mathbb{R}$. The goal is to find:

$$min_{x \in X} f(x)$$

1.3 Disciplined Convex Programming (DCP)

For an optimization problem to be efficiently and reliably solvable, it must adhere to the principles of Disciplined Convex Programming (DCP). These rules can be summarized as follows:

An optimization problem (X, f) satisfies the DCP rules if the feasible set X is defined by a series of equality and inequality constraints, where each constraint conforms to one of the following patterns:

- $affine = affine$
- $convex \leq concave$
- $concave \geq convex$

Additionally, the objective function f must be convex. By adhering to these rules, we ensure that state-of-the-art solvers can efficiently find optimal solutions.

To be more precise, each constraint in a convex optimization problem consists of a left-hand side (LHS) and a right-hand side (RHS), both of which can be expressed as functions $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$. The DCP rules can be interpreted as follows:

- $affine = affine$ as f, g are real-valued affine functions
- $convex \leq concave$ as f is convex over $\mathbb{R}^n$ and g is concave over $\mathbb{R}^n$
- $concave \geq convex$ as f is concave over $\mathbb{R}^n$ and g is convex over $\mathbb{R}^n$

If a problem adheres to these rules, it is considered a convex optimization problem. However, these rules are not comprehensive. It is possible to create valid convex expressions and models that fall outside the scope of these rules. In such cases, additional analysis may be needed to verify convexity.

Advanced solvers, such as 'CVX', apply these rules to systematically determine whether an expression is affine, convex, or concave. For further details on how these solvers handle disciplined convex programming, refer to their documentation: https://web.cvxr.com/cvx/beta/doc/dcp.html

2 Problem Framework

The problem of motion planning using an optimization-based approach is defined as follows:

The objective is to find a function $\pi(t) : [0, T] \to \mathcal{X}$, where T represents the planning horizon and $\mathcal{X}$ is the configuration space, which defines the set of feasible configurations for the vehicle.

The vehicle starts in an initial configuration $x_{\text{initial}} \in \mathcal{X}$, and the trajectory should end in a set of goal configurations $X_{\text{goal}} \subset \mathcal{X}$.

Additionally, a constraint is defined over the derivatives of π with respect to $t \in [0, T]$, denoted as $D(\pi(t), \pi'(t), \pi''(t), \dots)$.

To formulate an optimization problem, we define an objective function and a feasible set.

Let $\Pi[\mathcal{X}, T]$ represent the set of all possible functions mapping $[0, T]$ to $\mathcal{X}$. Further, let $J(\pi) : \Pi[\mathcal{X}, T] \to \mathbb{R}$ be the objective function.

Problem Definition: Optimal Trajectory Planning

Given a 6-tuple $(\mathcal{X}, x_{\text{initial}}, X_{\text{goal}}, D, J, T)$, the objective is to find:

$$\begin{aligned}
x^* = \; & \underset{\pi \in \Pi(\mathcal{X}, T)}{\arg\min}\, J(\pi) && && (2.1)\\
\text{s.t.} \quad & \pi(0) = x_{\text{initial}} && && (2.2)\\
& \pi(T) \in X_{\text{goal}} && && (2.3)\\
& \pi(t) \in \mathcal{X}, && \text{for all} \quad t \in [0, T] && (2.4)\\
& D(\pi(t), \pi'(t), \pi''(t), \dots), && \text{for all} \quad t \in [0, T] && (2.5)
\end{aligned}$$

2.1 Complexity

Finding an exact solution in a dynamic environment is highly challenging. The problem is inherently non-convex, and solvers cannot directly operate over a function space.

2.2 Numerical Approach

To address this problem numerically, we first define the constraints by modeling the vehicle and its environment. We then reformulate the problem, discretize it, and approximate it. Our goal is to obtain a solution that is both computationally efficient and reliable.

To achieve this, we employ a convex solver, necessitating adherence to disciplined convex programming (DCP) rules.

2.3 Vehicle Models

A vehicle model describes the position and orientation of the vehicle in the real world and predicts how these states change over time. Different models vary in complexity and accuracy. Generally, higher complexity results in increased accuracy.

Here, we introduce two fundamental models.

2.3.1 State and Control Variables

The state variables x_i represent the current position and pose of the vehicle, potentially including velocity, steering angle, or other relevant parameters. We group these variables into a state vector x.

The control inputs u_i are external influences that modify the state. These are grouped into a control vector u. Both x and u are time-dependent.

2.3.2 Point Mass Model

The point mass model (PM) consists of four state variables:

$$x = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} \tag{2.6}$$

Here, p_x and p_y represent the vehicle's position in a global fixed coordinate system. Instead of an explicit orientation, velocity is divided into its components v_x and v_y.

The model has two control inputs:

$$u = \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{2.7}$$

These correspond to accelerations in the x and y directions.

The future position and velocity are determined by the following system of differential equations:

$$\dot{x} = Ax + Bu \tag{2.8}$$

where

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.9}$$

The control inputs are bounded by a circular constraint, where the radius $a_{\max}$ is a model parameter:

$$\sqrt{u_1^2 + u_2^2} \leq a_{\max} \tag{2.10}$$

This model is the simplest commonly used for motion planning.

2.3.3 Kinematic Single Track Model

The kinematic single track model (KST) consists of five state variables:

$$x = \begin{bmatrix} p_x \\ p_y \\ \delta \\ v \\ \psi \end{bmatrix} \tag{2.11}$$

Similar to the point mass model, the first two state variables p_x and p_y define the vehicle's global position in a two-dimensional coordinate system. The vehicle is now modeled with an orientation ψ relative to the global x-axis. The velocity vector describes the velocity v of the rear wheel, aligning with the orientation. The front wheels can rotate around the yaw axis, and their angle relative to the orientation is represented by the steering angle δ.

Two control inputs modify the velocity and steering, directly affecting the state variables:

$$u = \begin{bmatrix} v_\delta \\ a_{\text{long}} \end{bmatrix} \tag{2.12}$$

where v_δ is the steering velocity, and a_{long} is the longitudinal acceleration.

The future state follows these differential equations:

$$\dot{p}_x = v\cos(\psi) \tag{2.13}$$
$$\dot{p}_y = v\sin(\psi) \tag{2.14}$$
$$\dot{\delta} = v_\delta \tag{2.15}$$
$$\dot{v} = a_{\text{long}} \tag{2.16}$$
$$\dot{\psi} = \frac{v}{l_{wb}}\tan(\delta) \tag{2.17}$$
$$\tag{2.18}$$

The single-track name originates from simplifying front and rear wheels into single contact points, assuming no wheel slip, leading to a kinematic model abstraction. The following figure comprehends the whole model nicely.

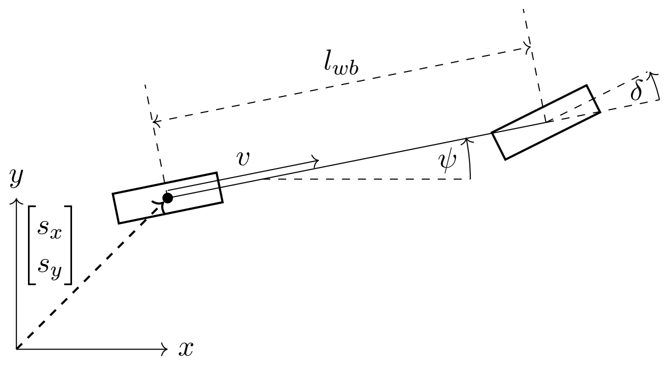


Figure 2.1: Bicycle model representation of a vehicle.

The following additional constraints are part of the model, a parameter a_{max} is introduced:

$$\sqrt{u_2^2 + (x_4\dot{x}_5)^2} \leq a_{\max} \tag{2.19}$$

For both models the vehicle is additional constrained on its velocity range, steering angle range and how fast the steering angle can change. Which are quite natural constraints, which should not be forgotten.

2.4 Constraints

Our constraints on the derivatives 2.5 using one of the models can be then expressed as:

$$\pi'(t) = f(\pi(t), u(t)) \tag{2.20}$$

2.5 Reformulation of the Problem

Since Solver cannot operate on a function space, the first task is to project our infinite-dimensional function space of trajectories to a finite-dimensional vector space. Additionally, we convert the constraints, which so far consist of set membership and predicates, into a set of equalities and inequalities, which is the standard input format for the solver.

Outlook: Penalty or Barrier functions, which integrate the constraints as part of the objective.

Direct vs Indirect Methods

2.5.1 Direct Methods

To project the function space to a vector space one defines a set of basis function which than span a subspace of the original function space. A function of the subspace is then defined by a linear combination of the basis function.

$$\pi\tilde{(t)} = \sum_{i=1}^{N} \pi_i \phi_i(t) \tag{2.21}$$

Of course, we may lose our original optimal solution and restrict our search space by the basis function, but our problem now reduces to finding finite-dimensional subspace. One of the most common numerical method to approximate are Numerical Integrators with Collocation. Collocation means that we require the trajectories to satisfies the constraints only in a set of discrete points $\{t_i\}_{i=1...m}$. Then numerical integrations techniques are used to approximate the trajectory between those points.

2.5.2 New Formulation

Given the models and using direct method, we can formulate our optimization problem over a finite-dimensional vector space.

Let $\mathcal{X}$ be the set of valid state the vehicle model can be in and $\mathcal{U}$ bet set of all valid control inputs of the model.

We define our trajectory over discrete time points $\{t_i\}_{i=1...m}$ as $\pi(t_i) = x_i$ and our new objective over $\mathcal{X} \times \mathcal{U}$, as $J : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$.

Problem Definition: Discrete Time Optimal Trajectory Planning

Given a 7-tuple $(\mathcal{X}, \mathcal{U}, x_{\text{initial}}, X_{\text{goal}}, f, J, \{t_i\}_{i=1...m})$, the objective is to find:

$$
\begin{aligned}
u^* = \arg\min_{u \in \mathcal{U}^{T-1}} & \sum_{i=1}^{T-1} J(x_{i+1}, u_i) && && \text{(2.22)}\\
\text{s.t.} \quad & x_1 = x_{\text{initial}} && && \text{(2.23)}\\
& x_T \in X_{\text{goal}} \subseteq \mathcal{X} && && \text{(2.24)}\\
& x_i, u_i, \in \mathcal{C} \subseteq \mathcal{X} \times \mathcal{U} && \text{for all} \quad i \in \{1, \dots, m-1\} && \text{(2.25)}\\
& x_{i+1} = x_i + (t_{i+1} - t_i) f(x_i, u_i) && \text{for all} \quad i \in \{1, \dots, m-1\} && \text{(2.26)}
\end{aligned}
$$

Given an initial state x_{initial} and a control sequence over the whole sequence of time points, we can use our model's dynamics f and the numerical integration approach to compute the state for each time point. Our problem formulation introduces a new coupling constraint C on a state and the control input applied to this state.

We will face the problem that the problem does so far not apply to the DCP Problems. We will present how one can come around this problem with both of the models we introduced. Doing so we will explain some modeling techniques, that come along with some approximations and investigate them. We will first focus on the point mass model.

2.5.3 Frenet Frame

So far, our models used a global Cartesian coordinate system to describe the vehicle positions and pose. This is fine for the dynamics of the vehicle, but once one starts to model the constraints which arise from the road topology it gets quite complex. Given the fact that road topology is known in advance we will use it to model or vehicles states that will empower us to model constraints based on the road in a more natural way, which also leads to convex road topology constraints.

3 McCormick Relaxation

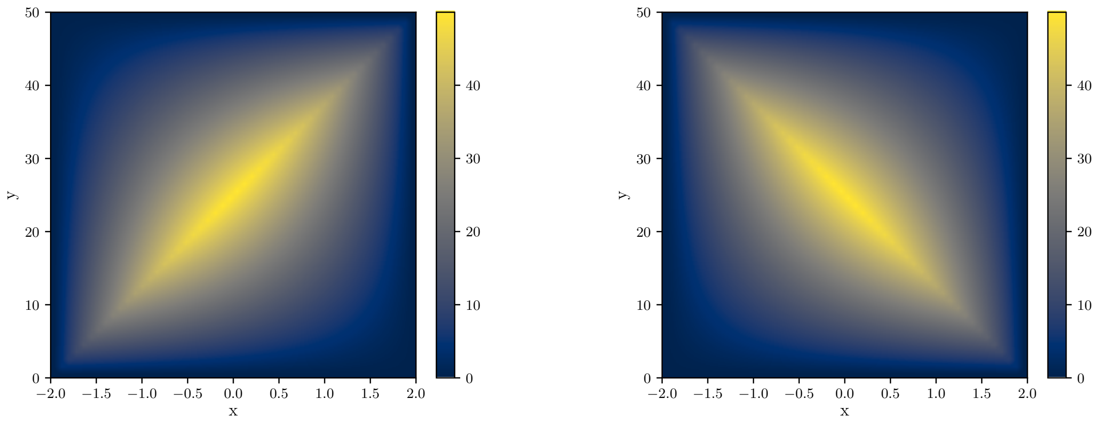
To address bilinear terms of the form $w = xy$, we introduce the following constraints based on the bounds of x and y:

$$x^L \leq x \leq x^U, \qquad y^L \leq y \leq y^U.$$

The resulting McCormick relaxation constraints for w are:

$$\begin{aligned} w &\geq x^L y + x y^L - x^L y^L, \\ w &\geq x^U y + x y^U - x^U y^U, \\ w &\leq x^U y + x y^L - x^U y^L, \\ w &\leq x^L y + x y^U - x^L y^U. \end{aligned}$$

These constraints establish an overestimation and underestimation of the bilinear term w, which can be visualized to assess their accuracy compared to the actual bilinear relationship.

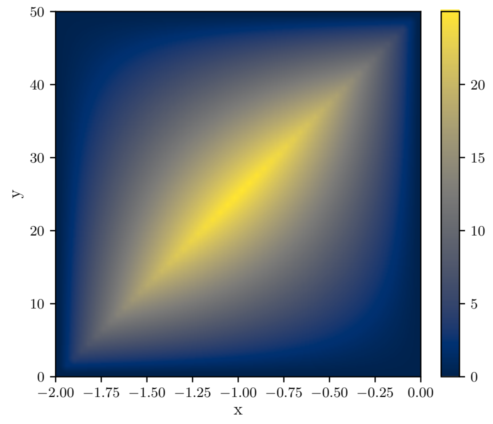


(a) Difference to the upper bound

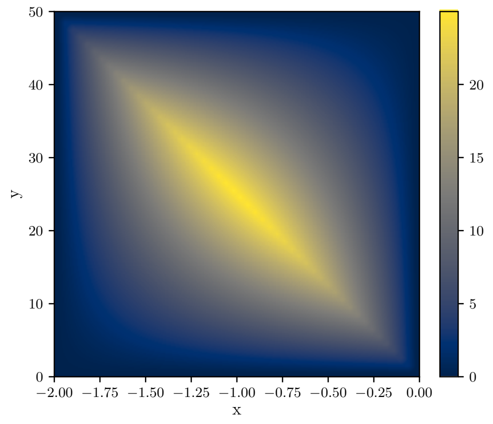
(b) Difference to the lower bound

Figure 3.1: McCormick relaxation bounds for the bilinear term $w = xy$.

Figure 3.1a illustrates the deviation between the actual bilinear term $w = xy$ and the smallest upper bound provided by the relaxation constraints. Similarly, Figure 3.1b shows the deviation to the greatest lower bound. For the range $-2 \leq x \leq 2$ and $0 \leq y \leq 50$. It is evident that the bounds improve as x and y approach their respective limits.



(a) Difference to the upper bound



(b) Difference to the lower bound

Figure 3.2: McCormick relaxation bounds for the bilinear term $w = xy$ with stricter bounds on x.

Figures 3.2a and 3.2b present the results when x is more tightly bounded, specifically $-2 \leq x \leq 0$. One can observe that the maximum deviation is considerably reduced compared to the previous scenario, indicating that tighter bounds yield a more accurate relaxation.

To illustrate the application of these relaxations in practice, consider a path-planning scenario with $v_{min} = 1$, $v_{max} = 4$, and $v_{start} = 1$. In this scenario, the bilinear term $v\xi$, which appears in the equation of motion for $\dot{n} = v \sin \xi \approx v\xi$, is approximated using McCormick relaxations.

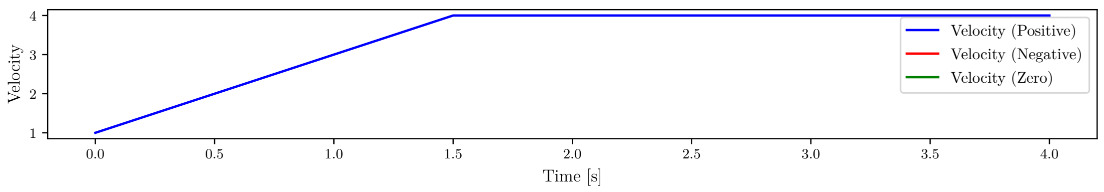


Figure 3.3: Planned velocity profile.

Figure 3.3 shows the planned velocity profile, which quickly reaches its upper limit.

Figure 3.4 depicts the alignment error ξ at each planned time point. Here, ξ is bounded within $-45° \leq \xi \leq 45°$. It is noteworthy that ξ does not reach these bounds.

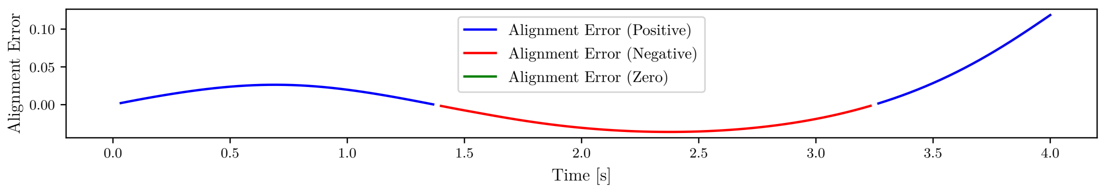


Figure 3.4: Alignment error ξ over time.

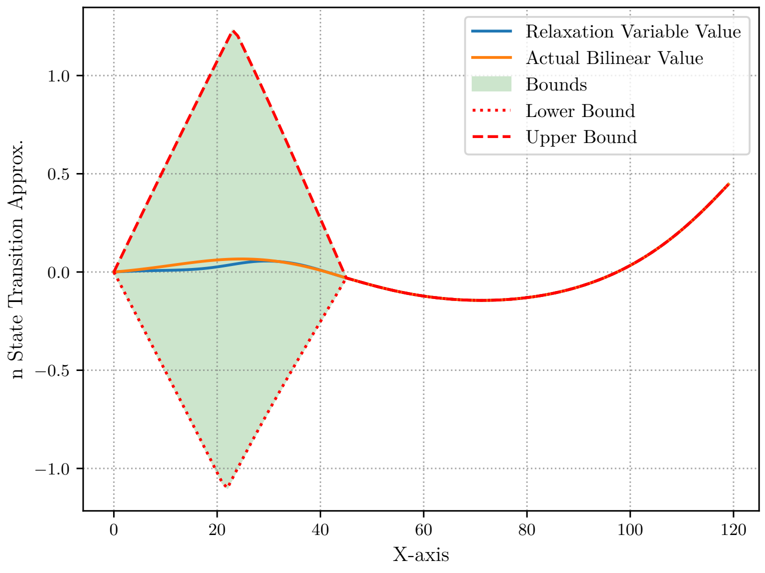


Figure 3.5: State transition approximation for n using bilinear term relaxation.

Figure 3.5 compares the actual bilinear value $v\xi$ with the relaxation variable w introduced via McCormick envelopes. This comparison highlights the accuracy of the relaxation approach in approximating the bilinear interaction and its effect on the state transition of n. Once the velocity reaches its limit, the approximation becomes increasingly accurate.

Note: The x-axis does not represent time directly but instead shows discrete time points. Here, 30 time points per second were chosen, resulting in a range from 0 to 120 for this figure, whereas the other figures range from 0 to 4.

4 Single Track Model

4.1 Model

The state variables and control inputs for the single-track model are defined as follows:

$$x = \begin{bmatrix} s \\ n \\ \xi \\ v \\ \delta \end{bmatrix} \quad \text{(state variables)}, \qquad u = \begin{bmatrix} a_{x,b} \\ v_\delta \end{bmatrix} \quad \text{(control inputs)}.$$

,where the state consist of Frenet Frame Coordinates (s,n), a Heading Alignment Error ξ, a longitude vehicle velocity v and a steering angle δ. The state evolution is given by:

$$\dot{x} = \begin{bmatrix} \dfrac{v\cos\xi}{1 - nC(s)} \\ v\sin\xi \\ \dfrac{1}{l_{wb}} v\tan\delta - C(s)\dot{s} \\ a_{x,b} \\ v_\delta \end{bmatrix}.$$

4.2 Assumptions

To simplify the model, the following assumptions are made:

- $C(s)$ is constant.
- $nC(s)$ is close to zero.

4.3 Non-Linear Terms

The following approximations are applied to linearize the model:

- $\frac{v\cos\xi}{1-nC(s)} \approx v\cos\xi \approx v$
- $v\sin\xi \approx v\xi$
- $v\tan\delta \approx v\delta$

4.4 Handling Bilinear Terms

To handle bilinear terms of the form $w = xy$, the following constraints are applied based on the bounds of x and y:

$$x^L \leq x \leq x^U, \qquad y^L \leq y \leq y^U.$$

The resulting constraints for w are:

$$\begin{aligned}
w &\geq x^L y + x y^L - x^L y^L, \\
w &\geq x^U y + x y^U - x^U y^U, \\
w &\leq x^U y + x y^L - x^U y^L, \\
w &\leq x^L y + x y^U - x^L y^U.
\end{aligned}$$

5 Derivation of the Integrator Model

5.1 Assumptions

1. External accelerations can be directly controlled:

$$u = \begin{bmatrix} a_{x,b}, & a_{y,b}, & a_{\psi} \end{bmatrix}, \quad a_b = \begin{bmatrix} a_{x,b}, & a_{y,b} \end{bmatrix} \tag{5.1}$$

2. Orientation of the vehicle ψ equals the angle of the road θ:

$$\xi = \psi - \theta = 0 \tag{5.2}$$

which implies:

- $a_b = a_t$
- $\dot{\psi} = \dot{\theta} = \frac{d\theta}{ds} \cdot \frac{ds}{dt} = C(s)\dot{s}$
- $a_{\psi} = \ddot{\psi} = \ddot{\theta} = C'(s)\dot{s}^2 + C(s)\ddot{s}$

3. $C'(s) = C'$ is constant.

5.2 Further Simplification

1. Define artificial input variables:

$$\tilde{u} := \begin{bmatrix} u_t \\ u_n \end{bmatrix} = \begin{bmatrix} \dfrac{a_{x,tn} + 2\dot{n}C(s)\dot{s} + nC'(s)\dot{s}^2}{1 - nC(s)} \\ a_{y,tn} - C(s)\dot{s}^2(1 - nC(s)) \end{bmatrix} \tag{5.3}$$

5.3 Resulting Integrator Model

State:

$$x_{tn} = \begin{bmatrix} s, & n, & \dot{s}, & \dot{n} \end{bmatrix}$$

Input:

$$\tilde{u} = \begin{bmatrix} u_t, & u_n \end{bmatrix}$$

The integrator model is given by:

$$\dot{x}_{tn} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} x_{tn} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tilde{u} \tag{5.4}$$

5.4 Constraints

The constraints are defined by:

$$g(x_{tn}, \tilde{u}) := \begin{bmatrix} (1 - nC(s))u_t - (2\dot{n}C(s)\dot{s} + nC'\dot{s}^2) \\ u_n + C(s)\dot{s}^2(1 - nC(s)) \end{bmatrix} \tag{5.5}$$

Define the constraint set $\mathcal{Z}$ as:

$$\mathcal{Z} = \left\{ \begin{bmatrix} x_{tn} \\ \tilde{u} \end{bmatrix} \middle| \begin{array}{l} C_{min} \leq C(s) \leq C_{max}, \\ n_{min} \leq n \leq n_{max}, \\ \begin{bmatrix} v_{xmin} \\ v_{ymin} \end{bmatrix} \leq \begin{bmatrix} \dot{s}(1 + nC(s)) \\ \dot{n} \end{bmatrix} \leq \begin{bmatrix} v_{ymax} \\ v_{xmax} \end{bmatrix}, \\ \dot{\psi}_{min} \leq C(s)\dot{s} \leq \dot{\psi}_{max}, \\ a_{\psi,b,min} \leq C'\dot{s}^2 + C(s)u_t \leq a_{\psi,b,max}, \\ a_{b,min} \leq g(x_{tn}, \tilde{u}) \leq a_{b,max}, \\ ||g(x_{tn}, \tilde{u})||^2 \leq \text{const} \end{array} \right\} \tag{5.6}$$

5.5 Constraint Approximation Problem

Given $\mathcal{Z}$, find an inner approximation $\underline{\mathcal{Z}}$ of $\mathcal{Z}$ such that $\underline{\mathcal{Z}}$ can be described with a set of constraints following the Disciplined Convex Programming (DCP) rules:

- affine == affine
- convex $\leq$ concave
- concave $\geq$ convex

5.6 $\forall$-Elimination

For a constraint not following the DCP rules, of the form:

$$c_{min} \leq f(x, y) \leq c_{max} \tag{5.7}$$

with $x \in \mathbb{R}$, $y \in \mathbb{R}^n$, and $f : \mathbb{R}^{n+1} \to \mathbb{R}$, where $c_{min}, c_{max} \in \mathbb{R}$ are constants. Further, if f is affine in x, represented by:

$$f(x, y) = a(y)x + b(y) \tag{5.8}$$

with $a, b : \mathbb{R}^n \to \mathbb{R}$, bounds on $a(y)$ and $b(y)$ can be chosen:

$$a_{min} \leq a(y) \leq a_{max}, \quad b_{min} \leq b(y) \leq b_{max}$$

Thus, an inner approximation of the set Z defined by $c_{min} \leq f(x, y) \leq c_{max}$ can be given by:

$$\begin{aligned} &\underline{Z} = \\ &\{x \in \mathbb{R} \mid \forall y \in \mathbb{R}^n : a(y) \in [a_{min}, a_{max}] \wedge b(y) \in [b_{min}, b_{max}] \implies f(x, y) \in [c_{min}, c_{max}]\} \\ &\times \{y \in \mathbb{R}^n \mid a(y) \in [a_{min}, a_{max}] \wedge b(y) \in [b_{min}, b_{max}]\} \\ &=: X \times Y \end{aligned} \tag{5.9}$$

5.6.1 Calculate X

Assumptions:

$$c_{min} \leq b_{min} \text{ and } b_{max} \leq c_{max} \text{ (or } a(y) \neq 0 \text{ TODO)} \tag{5.10}$$

Definitions:

$$x_{min} := \max\left\{\min\left\{0, \frac{c_{min} - b_{min}}{a_{max}}\right\}, \min\left\{0, \frac{c_{max} - b_{max}}{a_{min}}\right\}\right\} \tag{5.11}$$

$$x_{max} := \min\left\{\max\left\{0, \frac{c_{max} - b_{max}}{a_{max}}\right\}, \max\left\{0, \frac{c_{min} - b_{min}}{a_{min}}\right\}\right\} \tag{5.12}$$

Claim:

$$X = [x_{min}, x_{max}] \tag{5.13}$$

Sub-Claim:

$$x_{min} < 0 < x_{max} \tag{5.14}$$

Proof of Claim (5.13)**:**

Let $x \in X$.

Case Distinction for x_{min}:

- **Case 1:** $x_{min} = \dfrac{c_{min} - b_{min}}{a_{max}}$

$$a_{max}x_{min} + b_{min} = c_{min} \leq a_{max}x + b_{min} \implies x_{min} \leq x \tag{5.15}$$

- **Case 2:** $x_{min} = \dfrac{c_{max} - b_{max}}{a_{min}}$

$$a_{min}x_{min} + b_{max} = c_{max} \geq a_{min}x + b_{max} \implies x_{min} \leq x \tag{5.16}$$

Case Distinction for x_{max}:

- **Case 1:** $x_{max} = \dfrac{c_{max} - b_{max}}{a_{max}}$

$$a_{max}x_{max} + b_{max} = c_{max} \geq a_{max}x + b_{max} \implies x_{max} \geq x \tag{5.17}$$

- **Case 2:** $x_{max} = \dfrac{c_{min} - b_{min}}{a_{min}}$

$$a_{min}x_{max} + b_{min} = c_{min} \leq a_{min}x + b_{min} \implies x_{max} \geq x \tag{5.18}$$

Therefore, we have:

$$x_{min} \leq x \leq x_{max} \tag{5.19}$$

Let $x \in [x_{min}, x_{max}]$, $y \in Y$.

Case Distinction for $a(y)$:

- **Case 1:** $a(y) > 0$

$$a(y)x + b(y) \leq a(y)x_{max} + b(y) \leq a_{max}\frac{c_{max} - b_{max}}{a_{max}} + b_{max} = c_{max} \tag{5.20}$$

$$a(y)x + b(y) \geq a(y)x_{min} + b(y) \geq a_{max}\frac{c_{min} - b_{min}}{a_{max}} + b_{min} = c_{min} \tag{5.21}$$

- **Case 2:** $a(y) < 0$

$$a(y)x + b(y) \leq a(y)x_{min} + b(y) \leq a_{min}\frac{c_{max} - b_{max}}{a_{min}} + b_{max} = c_{max} \tag{5.22}$$

$$a(y)x + b(y) \geq a(y)x_{max} + b(y) \geq a_{min}\frac{c_{min} - b_{min}}{a_{min}} + b_{min} = c_{min} \tag{5.23}$$

- **Case 3:** $a(y) = 0$

$$a(y)x + b(y) = b(y) \in [c_{min}, c_{max}] \tag{5.24}$$

Therefore,

$$\forall y \in Y : a(y)x + b(y) \in [c_{min}, c_{max}] \implies x \in X \tag{5.25}$$

5.6.2 Example

Given $v_{xmin} \leq \dot{s}(1 + nC(s)) \leq v_{xmin}$.
Set:

- $x = \dot{s}$
- $y = \begin{bmatrix} n \\ C(s) \end{bmatrix}$
- $a(y) = 1 + y_1 y_2$
- $b(y) = 0$

Bounds for $a(y), b(y)$:

- $a_{min} = 1 + \min\{C_{min}n_{min}, C_{min}n_{max}, C_{max}n_{min}, C_{max}n_{max}\}$
- $a_{max} = 1 + \max\{C_{min}n_{min}, C_{min}n_{max}, C_{max}n_{min}, C_{max}n_{max}\}$
- $b_{min} = b_{max} = 0$

One can now easily calculate $[x_{min}, x_{max}] = X$ and Y can be expressed as $[C_{min}, C_{max}] \times [n_{min}, n_{max}]$

6 Mapping Controls and States

Given: A car model which has state:

$$x_{car} = \begin{bmatrix} p_x \\ p_y \\ \delta \\ v \\ \psi \\ \dot{\psi} \\ \beta \end{bmatrix}$$

p_x Global Position, p_y Global Postion, δ Steering Angle, v Velocity, ψ Orientation, $\dot{\psi}$ Yaw Rate, β Slip Angle.
and control inputs:

$$u_{car} = \begin{bmatrix} \dot{\delta} \\ a \end{bmatrix}$$

$\dot{\delta}$ Steering Angle Rate, a Longitudinal Acceleration

The used planning model has its own state definition $x_{planning}$ and control inputs $u_{planning}$.

A planning model has to provide a mapping from the car states to its states:

$$x_{car} \mapsto x_{planning}$$

and a mapping from its control inputs to the control inputs of the car:

$$u_{planning} \mapsto u_{car}$$

6.1 Single Track Planning Model

$$x_{planning} = \begin{bmatrix} s \\ n \\ \xi \\ v \\ \delta \end{bmatrix}$$

$(s,\ n)$ Frenet Frame Coordinates along the road, ξ alignment error to the road, v velocity, δ steering angl.

$$u_{planning} = \begin{bmatrix} a \\ \dot{\delta} \end{bmatrix}$$

a Longitudinal Acceleration, $\dot{\delta}$ Steering Angle Rate

$$x_{car} = \begin{bmatrix} p_x \\ p_y \\ \delta \\ v \\ \psi \\ \dot{\psi} \\ \beta \end{bmatrix} \mapsto \begin{bmatrix} self.road.get\_road\_position(x, y)[0] \\ self.road.get\_road\_position(x, y)[1] \\ \psi - \theta(s) \\ v \\ \delta \\ \end{bmatrix}$$

$$u_{planning} = \begin{bmatrix} a \\ \dot{\delta} \end{bmatrix} \mapsto \begin{bmatrix} \dot{\delta} \\ a \end{bmatrix} = \begin{bmatrix} \dot{\delta} \\ a \end{bmatrix}$$

6.2 Double Integrator Planning Model

$$x_{planning} = \begin{bmatrix} s \\ n \\ \dot{s} \\ \dot{n} \end{bmatrix}$$

$(s,\ n)$ Frenet Frame Coordinates along the road, $(\dot{s},\ \dot{n})$ Change of the Frenet Frame Coordinates along the road,

$$u_{planning} = \begin{bmatrix} u_t \\ u_n \end{bmatrix}$$

u_t, u_n artificial control inputs, which can be mapped (a_s, a_n) Acceleration of the Frenet Frame Coordinates along the road using g.

$$x_{car} = \begin{bmatrix} p_x \\ p_y \\ \delta \\ v \\ \psi \\ \dot{\psi} \\ \beta \end{bmatrix} \mapsto \begin{bmatrix} self.road.get\_road\_position(x,y)[0] \\ self.road.get\_road\_position(x,y)[1] \\ v\cos(\psi - \theta(s)) \\ v\sin(\psi - \theta(s)) \end{bmatrix}$$

For control input we additional require the current state $x = [s, n, \dot{s}, \dot{n}]^T$ of the planning model and we need to store the current steering angle of the car δ_{cur}:

6.2.1 Approach 1

$$\xi = \arctan(\frac{\dot{n}}{\dot{s}(1 - nC(s))})$$

$$v = \sqrt{(\dot{s}(1 - nC(s)))^2 + \dot{n}^2}$$

$$\dot{\psi} = \frac{u_n - \tan(\xi)u_t}{v(\tan(\xi)\sin(\xi) + \cos(\xi))}$$

$$a = \frac{u_t + v\dot{\psi}\sin(\xi)}{\cos(\xi)}$$

$$\dot{C} = \dot{C}(s) = C'(s)\dot{s} \approx \frac{C(s + \dot{s}\Delta t) - C(s)}{\Delta t}$$

$$\dot{\xi} = \frac{1}{1 + (\frac{\dot{n}}{\dot{s}(1-nC(s))})^2} \frac{u_n\dot{s}(1 - nC(s)) - \dot{n}(u_t - C(s)(u_t n + \dot{s}\dot{n}) - \dot{C}\dot{s}n)}{(\dot{s}(1 - nC(s)))^2}$$

$$\delta = \arctan((\dot{\xi} + C(s)\dot{s})\frac{l_{wb}}{v})$$

$$v_\delta = \max(\min(\frac{\delta - \delta_{cur}}{\Delta t}, \overline{v_\delta}), \underline{v_\delta})$$

$$u_{planning}, x_{planning} \mapsto \begin{bmatrix} v_\delta \\ a \end{bmatrix} = \begin{bmatrix} \dot{\delta} \\ a \end{bmatrix}$$

6.2.2 Approach 2

$$[s_0, n_0, \dot{s}_0, \dot{n}_0]^T := [s, n, \dot{s}, \dot{n}]^T$$

$$[s_1, n_1, \dot{s}_1, \dot{n}_1]^T := \begin{bmatrix} s_0 \\ n_0 \\ \dot{s}_0 \\ \dot{n}_0 \end{bmatrix} + \Delta t \begin{bmatrix} \dot{s}_0 \\ \dot{n}_0 \\ u_n \\ u_t \end{bmatrix}$$

$$v_0 = \sqrt{(\dot{s}_0^2 + \dot{n}_0^2)}$$

$$v_1 = \sqrt{(\dot{s}_1^2 + \dot{n}_1^2)}$$

$$a = \frac{v_1 - v_0}{\Delta t}$$

$$v_\delta = \arctan(l_{wb}(planned_psi_2 - (cur_psi + 1/l_wb * v_0 * np.tan(cur_steering_angle) * dt))/(v_1 * dt))/dt$$

$$x_1 = \begin{bmatrix} s_1 \\ n_1 \\ \dot{s}_1 \\ \dot{n}_1 \end{bmatrix} = x_0 + \begin{bmatrix} \dot{s}_0 \\ \dot{n}_0 \\ u_t \\ u_n \end{bmatrix} dt$$

$$\begin{bmatrix} a_s \\ a_n \end{bmatrix} = g(u_t, u_n)$$

$$\xi_0 = \arctan\left(\frac{\dot{n}_0}{\dot{s}_0}\right)$$
$$\xi_1 = \arctan\left(\frac{\dot{n}_1}{\dot{s}_1}\right)$$

$$v_0 = \sqrt{\dot{s}_0^2 + \dot{n}_0^2}$$
$$v_1 = \sqrt{\dot{s}_1^2 + \dot{n}_1^2}$$

$$\psi_0 = \xi_0 + \theta(s_0)$$
$$\psi_1 = \xi_1 + \theta(s_1)$$

$$\dot{\psi} = \frac{\psi_1 - \psi_0}{dt}$$

$$u_{planning}, x_{planning} \mapsto \frac{1}{dt} \begin{bmatrix} \arctan(l_{wb}\frac{\dot{\psi}}{v_1}) - \delta_{cur} \\ v_1 - v_0 \end{bmatrix} = \begin{bmatrix} \dot{\delta} \\ a \end{bmatrix}$$

7 Main Problem with Mccormick

The useual way how I apply McCormick is by saying:

Given a State, which is valid, find the next State which is valid with the corresponding control which is also valid under a given Objective.

Problem: This is not convex.

By Convexifying this Relationship, we decouble the interaction.

Question:

Can this approach result in two states, where no valid control input can exist? If so, how often does this happen.

Lets look at a example:

$$x_{n+1} = x_n + u_n v_n$$

With State x and Control Inputs u, v.

Applying Mccormick:

$w = xy$

Hesse-Matrix of $f(x, y) = xy$:

$$\nabla f = \begin{bmatrix} y \\ x \end{bmatrix}$$

$$\nabla^2 f = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$| \begin{bmatrix} -\lambda & 1 \\ 1 & -\lambda \end{bmatrix} | = \lambda^2 - 1 = 0 \implies \lambda = \pm 1$$

8 Convexify Equations of Motion

$$x_{tn} = \begin{bmatrix} s \\ n \\ \xi \\ \dot{s} \\ \dot{n} \\ \dot{\psi} \end{bmatrix}$$

$$\dot{x}_{tn} = \begin{bmatrix} \dot{s} \\ \dot{n} \\ \dot{\psi} - C(s)\dot{s} \\ \dfrac{a_{x,tn} + 2\dot{n}C(s)\dot{s} + nC'(s)\dot{s}^2}{1 - nC(s)} \\ a_{y,tn} - C(s)\dot{s}^2(1 - nC(s)) \\ a_{\psi} \end{bmatrix}$$

1. Equation

$$\xi_{n+1} = \xi_n + (\dot{\psi} - C(s)\dot{s})dt$$

Since dt is a constant, one only has to approximate $C(s)\dot{s}$ with a affine linear term. This can be done by guessing s using a reference velocity and evalutaing the curvature at this position, this would make $C(s)$ a constant in planning.

2. Equation

$$\dot{s}_{n+1} = \dot{s}_n + \frac{a_{x,tn} + 2\dot{n}C(s)\dot{s} + nC'(s)\dot{s}^2}{1 - nC(s)}dt$$

- One can assume that $nC(s)$ is close to Zero: $\dfrac{1}{1 - nC(s)} \approx 1$.
- We can again make $C(s)$ a constant, that will be close to the true value.
- Use McCormick for $\dot{s}\dot{n}$
- Hardest Part: $n\dot{s}^2$: Use McCormick twice

9 Investigate Main Paper

Mapping Vector p between the Coordinate Systems

$$p^T \mathbf{B}_b = p^T \begin{bmatrix} \cos\xi & \sin\xi & 0 \\ -\sin\xi & \cos\xi & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{B}_{tn} \tag{9.1}$$

Mapping Velocity

$$\begin{bmatrix} \dot{s}(1-nC) \\ \dot{n} \\ 0 \end{bmatrix}^T \mathbf{B}_{tn} = \begin{bmatrix} v_x\cos\xi - v_y\sin\xi \\ v_x\sin\xi + v_y\cos\xi \\ 0 \end{bmatrix}^T \mathbf{B}_{tn} \tag{9.2}$$

Mapping Acceleration

$$\begin{bmatrix} a_{x,tn} \\ a_{y,tn} \\ 0 \end{bmatrix}^T \mathbf{B}_{tn} = \begin{bmatrix} (\dot{v}_x - v_y\dot{\psi})\cos\xi - (\dot{v}_y + v_x\dot{\psi})\sin\xi \\ (\dot{v}_x - v_y\dot{\psi})\sin\xi + (\dot{v}_y + v_x\dot{\psi})\cos\xi \\ 0 \end{bmatrix}^T \mathbf{B}_{tn} \tag{9.3}$$

with $a_{x,tn} = \ddot{s}(1-nC(s)) - 2\dot{n}C(s)\dot{s} - nC'(s)\dot{s}^2$, $a_{y,tn} = \ddot{n} + C(s)\dot{s}^2(1-nC(s))$, it follows

$$\begin{bmatrix} \ddot{s} \\ \ddot{n} \\ 0 \end{bmatrix}^T \mathbf{B}_{tn} = \begin{bmatrix} \dfrac{(\dot{v}_x - v_y\dot{\psi})\cos\xi - (\dot{v}_y + v_x\dot{\psi})\sin\xi + 2\dot{n}C(s)\dot{s} - nC'(s)\dot{s}^2}{1-nC(s)} \\ (\dot{v}_x - v_y\dot{\psi})\sin\xi + (\dot{v}_y + v_x\dot{\psi})\cos\xi - C(s)\dot{s}^2(1-nC(s)) \\ 0 \end{bmatrix}^T \mathbf{B}_{tn} \tag{9.4}$$

since a vehicle has only longitudinal velocity, set $v_y = 0$, $\dot{v}_y = 0$

$$\begin{bmatrix} \dot{s}(1-nC) \\ \dot{n} \\ 0 \end{bmatrix}^T \mathbf{B}_{tn} = \begin{bmatrix} v_x\cos\xi \\ v_x\sin\xi \\ 0 \end{bmatrix}^T \mathbf{B}_{tn} \tag{9.5}$$

$$\begin{bmatrix} \ddot{s} \\ \ddot{n} \\ 0 \end{bmatrix}^T \mathbf{B}_{tn} = \begin{bmatrix} \dfrac{(\dot{v}_x + v_x\dot{\psi})\sin\xi + 2\dot{n}C(s)\dot{s} - nC'(s)\dot{s}^2}{1 - nC(s)} \\ (\dot{v}_x + v_x\dot{\psi})\cos\xi - C(s)\dot{s}^2(1 - nC(s)) \\ 0 \end{bmatrix}^T \mathbf{B}_{tn} \tag{9.6}$$

Choosing $\dot{v}_x$ and $\ddot{\psi}$ as control input results in the following state transition

$$x_{tn} = \begin{bmatrix} s & n & \xi & \dot{s} & \dot{n} & \dot{\psi} \end{bmatrix}^T \tag{9.7}$$

$$\dot{x}_{tn} = f_{tn}(s, n, \xi, \dot{s}, \dot{n}, \dot{\psi}, \dot{v}_x, \ddot{\psi}) = \begin{bmatrix} \dot{s} \\ \dot{n} \\ \dot{\psi} - C(s)\dot{s} \\ \dfrac{(\dot{v}_x + v_x\dot{\psi})\sin\xi + 2\dot{n}C(s)\dot{s} - nC'(s)\dot{s}^2}{1 - nC(s)} \\ (\dot{v}_x + v_x\dot{\psi})\cos\xi - C(s)\dot{s}^2(1 - nC(s)) \\ \ddot{\psi} \end{bmatrix} \tag{9.8}$$

with (9.5)

$$\dot{x}_{tn} = f_{tn}(s, n, \xi, \dot{s}, \dot{n}, \dot{\psi}, \dot{v}_x, \ddot{\psi}) = \begin{bmatrix} \dot{s} \\ \dot{n} \\ \dot{\psi} - C(s)\dot{s} \\ \dfrac{\dot{v}_x\sin\xi + \dot{\psi}\dot{n} + 2\dot{n}C(s)\dot{s} - nC'(s)\dot{s}^2}{1 - nC(s)} \\ \dot{v}_x\cos\xi + \dot{\psi}\dot{s}(1 - nC) - C(s)\dot{s}^2(1 - nC(s)) \\ \ddot{\psi} \end{bmatrix} \tag{9.9}$$

to make life easier, replace the state variables $\dot{s}$ and $\dot{n}$ with v_x, one can derive $\dot{s}$ and $\dot{n}$ using (9.5)

$$\dot{x}_{tn} = f_{tn}(s, n, \xi, v_x, \dot{\psi}, \dot{v}_x, \ddot{\psi}) = \begin{bmatrix} \dfrac{v_x\cos\xi}{1 - nC} \\ v_x\sin\xi \\ \dot{\psi} - C(s)\dot{s} \\ \dot{v}_x \\ \ddot{\psi} \end{bmatrix} \tag{9.10}$$

10 Cylindrical Algebraic Decomposition

$$\exists x, x^2 + bx + 1 \leq 0 \iff (b \leq -2) \vee (b \geq 2)$$

CAD for $f(x) = x^2 + bx + 1$:

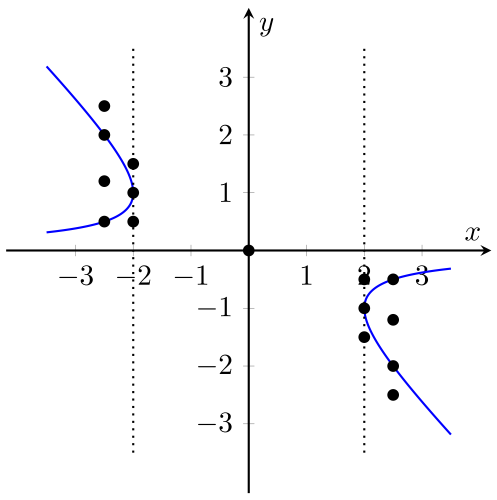


Figure 10.1: Illustrating the cells and the samples.

Project the valid cells onto the b-axis and take the disjunction of the intervals.