CONTENTS

CHAPTER **TOPIC** **PAGE NUMBER**

Chapter 1

Vehicle Models

1.1 Unicycle model

$$\dot{x} = u_s \cos\theta \tag{1.1}$$
$$\dot{y} = u_s \sin\theta \tag{1.2}$$
$$\dot{\theta} = u_\omega \tag{1.3}$$

1.1.1 Reed Shepp's Car

A Unicycle model can simulate a Reed-Shepp's car with

- $u_s \in \{-1, 0, 1\}$
- Transformation $u_\omega = \frac{\tan\delta}{L}$
- restricting δ to $[-\delta_{max}, \delta_{max}]$ where δ represent the steering angle and $\frac{L}{\tan\delta}$ represents the turning radius. L is the wheel-base.

1.1.2 Dubin's Car

Dubin's car is a simplification of Reed-Shepp's car with $u_s \in \{0, 1\}$

A unicycle model can simulate a Reed-Shepp's car, Dubin's car and the Single track rear wheel model below when steering is the control input without considering the steering rate. It can also simulate tricycle and differential drive robot. The details are not discussed here explicitly

1.2 Single Track Rear Wheel Model

For a front-wheel driven vehicle with no slip

$$\dot{x} = u_s \cos\theta \tag{1.4}$$
$$\dot{y} = u_s \sin\theta \tag{1.5}$$
$$\dot{\theta} = u_s \frac{\tan\delta_r}{L} \tag{1.6}$$
$$\dot{\delta_r} = u_\delta \tag{1.7}$$

where L is the wheel base and the state represents the rear-wheel, δ_r represents the steering angle at the rear wheels and u_s is the speed of the rear-wheel

The Single track rear wheel model can be further expanded out to include acceleration as

$$\dot{x} = u_s \cos\theta \tag{1.8}$$
$$\dot{y} = u_s \sin\theta \tag{1.9}$$
$$\dot{\theta} = u_s \tan\delta_r / L \tag{1.10}$$
$$\dot{u_s} = u_a (\text{or other first order approximations/reduced order models}) \tag{1.11}$$
$$\dot{\delta_r} = u_\delta \tag{1.12}$$

the control input is $u = [u_a u_\delta]'$ If curvature is the input to the system, the model can be rewritten as

$$\dot{x} = u_s \cos\theta \tag{1.13}$$
$$\dot{y} = u_s \sin\theta \tag{1.14}$$
$$\dot{\theta} = u_s \kappa \tag{1.15}$$
$$\dot{u_s} = u_a (\text{or other first order approximations/reduced order models}) \tag{1.16}$$

where κ indicates the curvature value

1.3 Single Track Front Wheel Model

For a front-wheel driven vehicle with no side slip

$$\dot{x} = u_s \cos(\theta + \delta) \tag{1.17}$$
$$\dot{y} = u_s \sin(\theta + \delta) \tag{1.18}$$
$$\dot{\theta} = u_s \frac{\sin\delta}{L} \tag{1.19}$$
$$\dot{\delta_f} = u_\delta \tag{1.20}$$

where δ_f represents the steering angle at the front wheels wheels. This model can be augmented with acceleration commands instead of speed similar to the equations presented in the rear wheel models.

Furthermore, the models can be rewritten using curvature instead of steering angles or steering rate. Additionally, there are limits on speed, acceleration, steering, steering rate and curvature.

1.4 Model with Slip

Assuming the vehicle can be steered with both front and rear wheels, the model equations with slip β are

$$\dot{x} = u_s \cos(\theta + \beta) \tag{1.21}$$
$$\dot{y} = u_s \sin(\theta + \beta) \tag{1.22}$$
$$\dot{\theta} = u_s \frac{\cos\beta}{l_f + l_r}(\tan\delta_f - \tan\delta_r) \tag{1.23}$$
$$\dot{u_s} = u_a \tag{1.24}$$
$$\beta = \arctan(\frac{l_f \tan\delta_r + l_r \tan\delta_f}{l_f + l_r}) \tag{1.25}$$

where β denotes vehicle slip angle, l_f is the distance of the front wheels to the center of mass, l_r is the distance of the rear wheels to the center of mass, δ_f is the front wheel steer and δ_r is the rear wheel steer

1.4.1 Derivation

From left triangle, applying the sine rule

$$\frac{\sin(\frac{\pi}{2} + \delta_r)}{R} = \frac{sin(\beta - \delta_r)}{l_r} \tag{1.26}$$
$$\implies \frac{l_r}{R} = \sin\beta - \tan\delta_r \cos\beta \tag{1.27}$$

From the triangle on the right, applying the sine rule

$$\frac{\sin(\frac{\pi}{2} - \delta_f)}{R} = \frac{sin(\delta_f - \beta)}{l_f} \tag{1.28}$$
$$\implies \frac{l_f}{R} = \tan\delta_f \cos\beta - \sin\beta \tag{1.29}$$

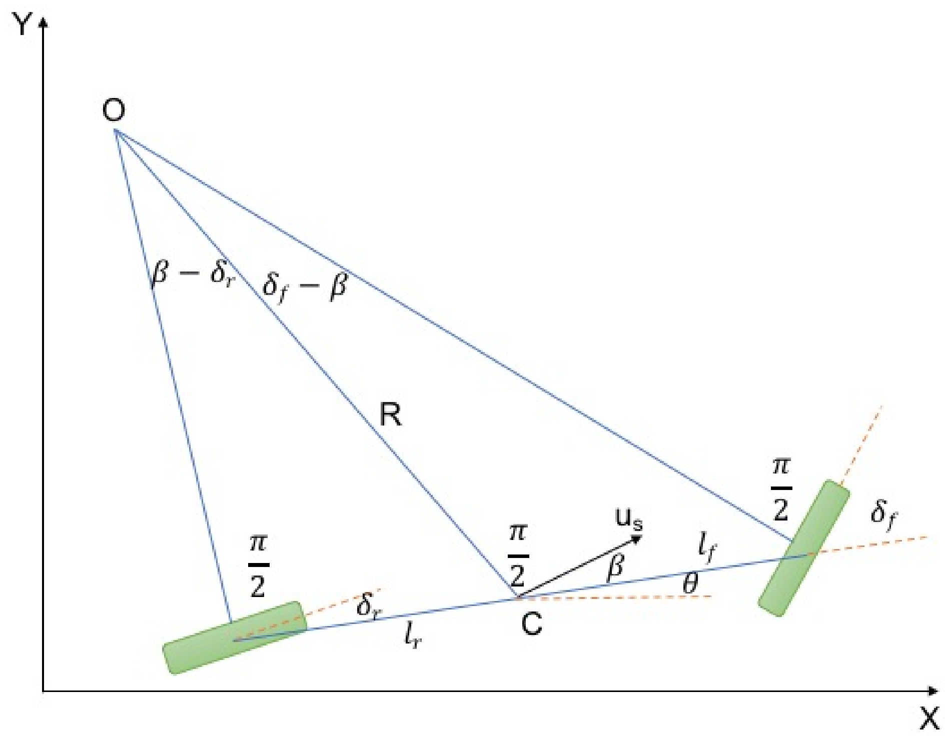

Figure 1.1: Kinematic vehicle model with Slip and front and rear wheel steer. O is the instantaneous center of rotation

Divide equations (1.27) and (1.29), we have

$$\frac{l_r}{l_f} = \frac{\sin\beta - \tan\delta_r \cos\beta}{\tan\delta_f \cos\beta - \sin\beta} \tag{1.30}$$

$$\implies \frac{l_r}{l_f} = \frac{\tan\beta - \tan\delta_r}{\tan\delta_f - \tan\beta} \tag{1.31}$$

$$\implies \tan\beta = \frac{l_r \tan\delta_f + l_f \tan\delta_r}{l_f + l_r} \tag{1.32}$$

Now we know that $\dot{\theta} = \frac{u_s}{R}$. To derive R, we use (1.27). Substituting $\tan\beta$ derived above into 1.27, we have

$$\frac{l_r}{R} = \cos\beta(\tan\beta - \tan\delta_r) \tag{1.33}$$
$$\implies \frac{l_r}{R} = \cos\beta(\frac{l_r\tan\delta_f + l_f\tan\delta_r}{l_f + l_r} - \tan\delta_r) \tag{1.34}$$
$$\implies \frac{l_r}{R} = \frac{\cos\beta}{l_f + l_r}(l_r\tan\delta_f - l_r\tan\delta_r) \tag{1.35}$$
$$\implies \frac{1}{R} = \cos\beta\frac{\tan\delta_f - \tan\delta_r}{l_f + l_r} \tag{1.36}$$
$$\implies \dot{\theta} = \frac{u_s}{R} = u_s\cos\beta\frac{\tan\delta_f - \tan\delta_r}{l_f + l_r} \tag{1.37}$$

As far the equations for x and y go, it is straightforward to see that

$$\dot{x} = u_s\cos(\theta + \beta) \tag{1.38}$$
$$\dot{y} = u_s\sin(\theta + \beta) \tag{1.39}$$
$$\tag{1.40}$$

1.5 Dynamic Lateral and Longitudinal Model

1.5.1 Lateral Dynamics

The lateral dynamics is given by

$$ma_y = m\ddot{y} + u_{sx}\dot{\theta} = F_{yf} + F_{yr} + F_{bank} \tag{1.41}$$
$$I_z\ddot{\theta} = l_fF_{yf} - l_rF_{yr} \tag{1.42}$$

where F_{yf} and F_{yr} denote the lateral tire forces at the front and rear wheels, m is the mass of the vehicle, u_sx is the longitudinal speed of the vehicle at CG.

Slip is the difference in angle between velocity vector at the wheel and the steering angle

For small slip angles (i.e. lower speeds ¡ 35-40 mph typically for nominal friction conditions i.e no sleet, water, ice etc), they are approximated by (for front wheel drive) a linearly as.

$$F_{yf} = 2C_{\alpha f}(\delta - \theta_{vf}) \tag{1.43}$$
$$F_{yr} = 2C_{\alpha r}(-\theta_{vr}) \tag{1.44}$$

where $C_{\alpha f}$ and $C_{\alpha r}$ represent the cornering stiffness of each of the front and rear wheels respectively. $F_{bank} = mg\sin\phi$, where ϕ is the bank angle, g is the acceleration due to

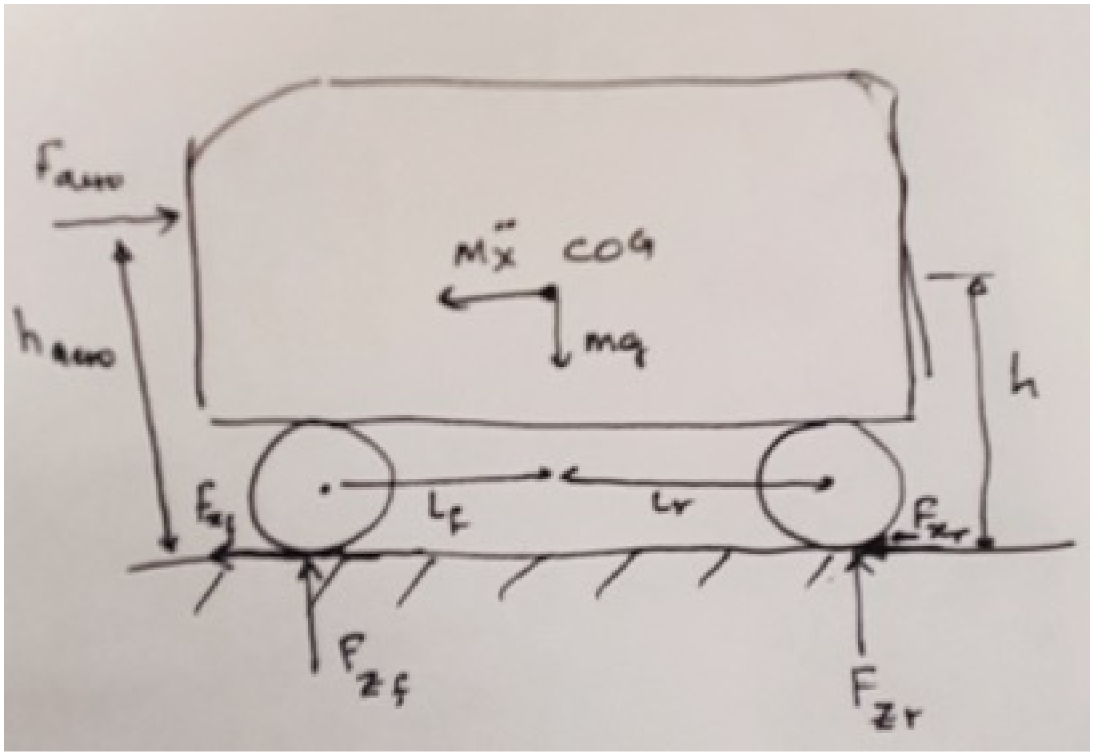

Figure 1.2: Longitudinal forces on the vehicle

gravity The velocity vector at the wheels are given by

$$\tan\theta_{vf} = \frac{u_s y + l_f\dot{\theta}}{u_s x} \tag{1.45}$$

$$\tan\theta_{vr} = \frac{u_s y - l_r\dot{\theta}}{u_s x} \tag{1.46}$$

Using small angle approximations, the above equations can be further reduced to

$$\theta_{vf} = \frac{\dot{y} + l_f\dot{\theta}}{u_s x} \tag{1.47}$$

$$\theta_{vr} = \frac{\dot{y} - l_r\dot{\theta}}{u_s x} \tag{1.48}$$

1.5.2 Longitudinal Dynamics

Ignoring the pitch of the vehicle i.e. assuming flat surface, the longitudinal dynamics is given by

$$m\ddot{x} = F_{xf} + F_{xr} - F_{aero} - R_{xf} - R_{xr} \tag{1.49}$$

where F_{xf} and F_{xr} denote the longitudinal forces at the front and rear wheels respectively, F_{aero} denotes aerodynamics drag force, R_{xf} and R_{xr} denote rolling resistance at the front and rear wheels Aerodynamic drag force is given by

$$F_{aero} = \frac{1}{2}\rho C_d A_F (V_x + V_{wind})^2 \tag{1.50}$$

where ρ is the density of air, C_d is the aerodynamic drag coefficient, V_x is the longitudinal speed of the vehicle, V_wind is the wind velocity and A_F is the frontal area of the vehicle. The longitudinal tire forces are a function of slip, normal load on the tire and the friction coefficient. The longitudinal slip ratio is defined as

$$\sigma_{long} = \frac{r_{eff}\omega_{wheel} - V_x}{V_x} \text{BRAKING} \tag{1.51}$$

$$\sigma_{long} = \frac{r_{eff}\omega_{wheel} - V_x}{r_{eff}\omega_{wheel}} \text{ACCELERATION} \tag{1.52}$$

where ω_{wheel} is the rotational speed of the wheel and r_{eff} is the effective tire radius. When the slip is small, the longitudinal forces are given by

$$F_{xf} = C_{\sigma f}\sigma_{xf}$$
$$F_{xr} = C_{\sigma r}\sigma_{xr}$$

where $C_{\sigma f}$ and $C_{\sigma r}$ represent longitudinal stiffness at front and rear wheels.

The rolling resistance R_{xf} and R_{xr} are approximated using the coefficient of friction and the normal loading force at the front wheel F_{zf} and rear wheel F_{zr} as

$$R_{xf} + R_{xr} = f(F_{zf} + F_{zr})$$

. When the vehicle is travelling along a surface with no gradient, F_{zf} and F_{zr} are in turn calculated by taking moments about the contact points of the front and rear wheels as (see Figure 1.2)

$$F_{zf}(l_f + l_r) + F_{aero}h_{aero} - m\ddot{x}h - mgl_r = 0 \tag{1.53}$$

$$F_{zr}(l_f + l_r) - F_{aero}h_{aero} + m\ddot{x}h - mgl_f = 0 \tag{1.54}$$

The above equations can be straightforwardly extended to the case when the vehicle is pitching while maintaining contact with the ground i.e when the vehicle is moving on an inclined surface. (the terms that correspond mg will have additional component along and perpendicular to the surface)

1.5.3 Error models

With respect to the road

Let's say the objective is to follow the center line of the road. At any given point let θ_{des} be the headong of the centerline. The lateral acceleration desisred at this point would be

$V_x\theta_{des}$. The lateral acceleration of the vehicle in the inertail coordinates is $a_y = \ddot{y} + V_x\theta$ where V_x is the longitudinal speed of the vehicle in the body frame and θ is the heading of the vehicle. The error equations under the assumption of constant longitudinal speed can be written as

$$\dot{e_1} = \ddot{y} + V_x(\dot{\theta} - \dot{\theta_{des}}) \tag{1.55}$$

$$\dot{e_2} = \dot{\theta} - \dot{\theta_{des}} \tag{1.56}$$

where $e_1 = \dot{y} + V_x(\theta - \theta_{des})$ and $e_2 = \theta - \theta_{des}$ The error dynamics can now be obtained using (1.41). The model now can be used for developing steering control law the objective of which would be to stabilize the above system.

1.6 Linearization

Let's say a dynamic system is defined by

$$\dot{x} = f(x, u) \tag{1.57}$$

where x is the state and u is the control input The linearized system is given by

$$\dot{\delta x} = \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial u}\delta u + H.O.T \tag{1.58}$$

H.O.T (Higher order terms) are generally neglected under the assumption that δx and δu are small. Linearization is generally used for stability analysis and for designing controllers for reference and trajectory tracking (example LQR, MPC etc)

First define $\delta x = x - x^*$ and $\delta u = u - u^*$ where x^* and u^* represent the state and control inputs at equilibrium or define the operating points i.e. trim conditions. The derivation proceeds as follows

$$\dot{\delta x} = \dot{x} - \dot{x^*} \tag{1.59}$$

$$\implies \dot{\delta x} = f(x, u) - f(x^*, u^*) \tag{1.60}$$

$$\implies \dot{\delta x} = f(x^* + \delta x, u^* + \delta u) - f(x^*, u^*) \tag{1.61}$$

Taylor series expansion can now be used to derive the equations above.

H.0.T some times are used to a second degree in methods like DDP (Differential Dynamic Programming).We will discuss more about this in another chapter.

Example

Let's take the example of the simplified single track rear wheel model. The equations for the simple car model are given by

$$\dot{x} = u_1 \cos\theta$$

$$\dot{y} = u_1 \sin\theta$$

$$\dot{\theta} = u_1 \frac{\tan u_2}{L}$$

where u_1 is the speed control input and u_2 is the steering control input with L as the wheel base.

Using the derivation, let's linearize the system about the state $[0, 0, pi/4]$ and control $[1, 0]$

$$\dot{\delta x} = 0 \times \delta x + 0 \times \delta y - (u_1 = 1) \times \sin(\theta = \frac{pi}{4}) \times \delta\theta + \delta u_1 \tag{1.62}$$

$$\dot{\delta y} = 0 \times \delta x + 0 \times \delta y + (u_1 = 1) \times \cos(\theta = \frac{pi}{4}) \times \delta\theta + \delta u_1 \tag{1.63}$$

$$\dot{\delta\theta} = 0 \times \delta x + 0 \times \delta y + 0 \times \delta\theta + \frac{\tan(u_2 = 0)}{L} \times \delta u_1 + (u_1 = 1) \times \frac{\sec^2(u_2 = \frac{pi}{4})}{L} \times \delta u_2 \tag{1.64}$$

In matrix form it can be represented as

$$\begin{bmatrix} \dot{\delta x} \\ \dot{\delta y} \\ \dot{\delta\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ \delta\theta \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & \frac{2}{L} \end{bmatrix} \begin{bmatrix} \delta u_1 \\ \delta u_2 \end{bmatrix} \tag{1.65}$$

The controllability rank condition is not met by the linearized system. However the original nonlinear model i.e. the simple car model is Small Time Locally Controllable (STLC) - we will talk about this later

Chapter 2

Control

2.1 PID

2.2 LQR

Linear Quadratic Regulator is an optimal controller for linear systems and a quadratic cost function. It drives the states to zeros while keeping the control input small. It can be extended to

1. Regularization of non-linear system to a non-zero fixed point
2. Penalize rate of change of control inputs i.e for example jerk
3. Trajectory tracking for non-linear systems
4. Linear Time Varying (LTV) systems
5. Stochastic Systems
6. Affine systems

2.2.1 Problem formulation

The finitie horizon LQR problem is to find control inputs that minimizes

$$J = \sum_{k=0}^{N} (x_k^T Q x_k + u_k^T R u_k) \tag{2.1}$$

with $Q \succ 0$ and $R \succ 0$, Q and R are positive semi-definite matrices, subject to

$$x_{k+1} = Ax_k + Bu_k$$

When N is inf, the problem is infinite horizon problem. Since Q and R are positive semi-definite, any x or u that are not zeros will yield a positive cost by definition.

2.2.2 Examples

Leader following

For this problem, let's consider kinematic models in 1D. (The 2D motion can be reduced to under the assumption that the leader and the ego do not have any lateral movement and move on a straight road and appropriately attaching a coordinate system). Assuming our prediction module is perfect, we will model the leader as

$$s_{k+1}^{leader} = s_k^{leader} + v_k^{leader}\delta t \tag{2.2}$$
$$v_{k+1}^{leader} = v_k^{leader} \tag{2.3}$$

v_k is the longitudinal speed of all vehicles and is known at all time steps (from an on-board perception module). The ego or the self vehicle that is controlled is modeled as

$$s_{k+1}^{ego} = s_k^{ego} + v_k^{ego}\ \delta t + 0.5\ a_k^{ego}\ \delta t^2 \tag{2.4}$$
$$v_{k+1}^{ego} = v_k^{ego} + a_k^{ego}\ \delta t \tag{2.5}$$
$$a_{k+1}^{ego} = u_k^{ego} \tag{2.6}$$

where u_k is the control input to our system. The goal is to follow the leader while maintaining a distance greater than or equal to L_{sep}. Let's redefine the state of our system as

$$s_{k+1} = s_{k+1}^{leader} - s_{k+1}^{ego} - L_{sep} \tag{2.7}$$
$$v_{k+1} = v_{k+1}^{leader} - v_{k+1}^{ego} \tag{2.8}$$
$$a_{k+1} = -u_k^{ego} \tag{2.9}$$

Rewriting the matrix form with $x_k = \begin{bmatrix} s_k \\ v_k \\ a_k \end{bmatrix}$, the system is defined by

$$x_{k+1} = Ax_k + Bu_k \tag{2.10}$$

with

$$A = \begin{bmatrix} 1 & \delta t & 0.5\delta t^2 \\ 0 & 1 & \delta t \\ 0 & 0 & 0 \end{bmatrix}$$

and

$$B = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

. The goal is to drive the system states to zeros while minimizing the control effort. In this formulation, the control effort gets penalized twice as it is part of the state as well. So Q and R should be chosen carefully.

Notice that in the formulation, there is nothing preventing ego vehicle from crossing the leader and then maintaining L_{sep} apart from the control cost. There is also nothing preventing the system from commanding high acceleration or deceleration between consecutive steps $\implies$ uncomfortable ride. (which could happen based on the initial conditions)

One way to address the second problem is to increase entries in R i.e. penalize control effort a lot but this will also imply slow convergence and still nothing prevents it from giving large δu. To address this issue one can penalize δu in the cost function. The problem can be reformulated with δu as the control input i.e rate of change of acceleration as control input. The equations are now

$$s_{k+1} = s_{k+1}^{leader} - s_{k+1}^{ego} - L_{sep} \tag{2.11}$$

$$v_{k+1} = v_{k+1}^{leader} - v_{k+1}^{ego} \tag{2.12}$$

$$a_{k+1} = a_k + u_{k-1}^{ego} - u_k^{ego} \tag{2.13}$$

$$\implies a_{k+1} = a_k - \Delta u_k \tag{2.14}$$

where $\Delta u_k = u_k^{ego} - u_{k-1}^{ego}$ In matrix form the A and B matrices are transformed to

The typical way to go about penalizing change in control inputs for this problem is to initially start with x as two states (position and speed), then augmenting the state vector with control input as the third state (position, speed and acceleration/control input) and a change in the control input (change in acceleration) as the control input.

Trajectory tracking

Problem The trajectory tracking problem is to compute a sequence of control input $u_1, u_2, ..., u_{H-1}$ so that a state sequence $x_0^*, x_1^*, ..., x_H^*$ is attainable such that $x_{t+1} = f(x_t, u_t)$ while minimizing the cost function $\sum_{i=0}^{H-1}(x_i - x_i^*)^T Q(x_i - x_i^*) + (u_i - u_i^*)^T R(u_i - u_i^*)$ where $u_i^*, i = 0toH-1$ is a control input that ensures feasibility i.e. $x_{t+1}^* = f(x_t^*, u_t^*) \forall t \in \{0, 1, ..., H-1\}$

Example The trajectory tracking problem for non-linear systems can be transformed to LQR for Linear Time Varying case via linearization. If the input trajectory is not feasible, the computed solution will not visit the desired states. Let's consider the example of cruise control. The objective is to regulate the longitudinal speed of the vehicle using LQR. In this case, it is sufficient to consider longitudinal dynamics alone. Let the reference speed to be tracked be given by

$$v(t) = 10|\sin(0.2\pi t)| \tag{2.15}$$

Assuming that there is no wind and no gradient, the simplified longitudinal dynamics can be written as

$$m\dot{v} = u - F_{friction} - F_{drag} \tag{2.16}$$

$$\tag{2.17}$$

For the sake of simplicity, let's consider the RMS of speed and acceleration and to derive the nominal control input. RMS is the value that the system will see when the input is applied. The following parameters will be used

$m = 1000$

$\rho = 1$

$C_d = 0.6$

$A = 3$

$C_r = 0.3$

$V_{rms} = 10/\sqrt{2} = 7.07$

$F_{drag} = 0.5 * \rho * A * C_d * V_{rms}^2 \implies F_{drag} = 44.98N,$

$F_{friction} = C_r * V_{rms} \implies F_{friction} = 2.12N.$

RMS of acceleration $\dot{v}$ is 0.45.

The nominal control input $u = 1000 * 0.45 + 44.98 + 2.12$ i.e. $u^* = 497.1N$.

Next let's linearize the system around the nominal control input and the state at time t. Using the method discussed in Section 1.6, the linearized model is

$$m\dot{\delta v} = \delta u - C_r \delta v - \rho C_d A v \delta v \tag{2.18}$$

where v represents the operating point at which linearization is performed Let's now verify the accuracy of linearization when $v = 10 \implies \dot{v} = -0.45$ and $\delta v = 2.93$

1. Using (2.16), we get $u = 93\ N$
2. Using (2.18), we have $\dot{\delta v} = -0.45$, $\delta v = 10 - 7.07 = 2.93 \implies \delta u = -450 + 0.879 + 52.74 = -396.38$. From this $u = u^* + \delta u = 497.1 - 396.38 = 100.72\ N$
3. The estimates are close but not close enough. This is because $\delta v = 2.93$ is a large number. The contributions of the Higher Order Terms (H.O.T) which were neglected in the linearization will contribute to errors. So, when linearizing make sure that the δ's are small. Had we chosen the operating point to be $8m/s$, the estimates would be much closer. The resolution of discretization used to solve the problem affects the accuracy of linearization.

Now let's design the cost function for this example. To make life easier and avoid integrals, lets discretized the linearized system. The equations for the discretized system are given by

$$m\delta v_{k+1} = m\delta v_k - C_r \delta v_k^* \delta t - \rho C_d A v \delta v_k + u\delta t \tag{2.19}$$

$$\implies \delta v_{k+1} = \left(1 - \frac{C_r \delta t}{m} - \frac{\rho C_d A v_k^* \delta t}{m}\right)\delta v_k + \delta u \delta t \tag{2.20}$$

where δt is the sampling time and v_k^* is the target/desired speed at time step k The cost function for the above system is of the form

$$\sum_{k=1}^{N} q\left(v_k - v_k^*\right)^2 + \sum_{k=1}^{N} r\left(u_k - u_k^*\right)^2$$

where N is the horizon. For example, if the time horizon we are looking at T=5 seconds, and $dt = 0.1$, then $N = 50$, q is the penalty on state deviation and is a scalar (since the system is 1D) and r, is the penalty on control input deviation and is a scalar (1D). v_k^* is determined by (2.15) and $u_k^* = 497.1 \ \forall \ k \in \{1, 2, 3, ...N\}$.

In this specific example we chose $q = 10$ and $r = 1$. We chose a low r allow deviation in control as it is not a feasible control input for the states we want to visit. We chose a high q (compared to r) as we would like to get sufficiently close to the desired state sequences.

If the input trajectory is feasible, why do we need LQR?. For example, in the above case, open loop control inputs can be computed by inverting the dynamics as a function of the desired states. If we did this, why doe we need LQR - use the computed control input directly.

Generally, approximate trajectory (state and control) is computed using a reduced order model of the system. Such an approach gets the state and control sequences close to a feasible/optimal trajectory. Then LQR (or another approach) can be used with higher order model dynamics to improve the controls. One should also be careful in aligning the cost functions across various stages of development. The general approach is to segment the problem into path, trajectory and control modules where the solution from one module feeds into the other improving it successively via additional constraints and model order with appropriate cost functions. There are other reasons too and this is one of them.

In the example above, we could augment the system with additional equations for representing the lower level control. For example the computed control input might not be translated or transferred to the actuators instantaneously. One could use a first order model to represent the system. But using the model could complicate trajectory generation module. So, partitioning it will make life easier and could also ease compute time issues.

2.2.3 Derivation

Dynamic Programming

We will use bottom up DP approach for coming up with the solution. We will use the following identity

$$\frac{d(x^T A x)}{dx} = x^T (A^T + A) \tag{2.21}$$

If A is symmetric

$$\frac{d(x^TAx)}{dx} = x^T(A^T + A) = 2x^TA = 2A^Tx = 2Ax \tag{2.22}$$

Let's look at the equation 2.1. We can write the optimization problem as

$$J^*(x) = \underset{u_k, fork \in \{1,2,\ldots,N-1\}}{\operatorname{argmin}} \sum_{k=0}^{N-1} \left[x_k^TQx_k + u_k^TRu_k\right] + x_N^TP_Nx_N \tag{2.23}$$

assuming the control input $u_N = 0$ we have

$$J(x_N) = x_N^TP_NX_N \tag{2.24}$$
$$\implies J(x_N) = (Ax_{N-1} + Bu_{N-1})^TP_N(Ax_{N-1} + Bu_{N-1}) \tag{2.25}$$
$$J^*(x_{N-1}) = \underset{u_{N-1}}{\operatorname{argmin}} \left[x_{N-1}^TQx_{N-1} + u_{N-1}^TRu_{N-1} + J^*(x_N)\right] \tag{2.26}$$
$$\tag{2.27}$$

$$\implies J^*(x_{N-1})$$
$$= \underset{u_{N-1}}{\operatorname{argmin}} \left[x_{N-1}^TQx_{N-1} + u_{N-1}^TRu_{N-1} + (Ax_{N-1} + Bu_{N-1})^TP_N(Ax_{N-1} + Bu_{N-1})\right]$$

As u_{N-1} does not depend on x_{N-1} and $J(x_{N-1})$ denotes the cost to go from x_{N-1} to x_N and $*$ denotes the optimal cost or input corresponding to the optimal cost. For minimizing $J(x_{N-1})$, we set $\frac{\partial(x_{N-1})}{\partial u_{N-1}} = 0$ which results in

$$Ru_{N-1}^* + (Ax_{N-1} + Bu_{N-1}^*)^TP_NB = 0 \tag{2.28}$$
$$\implies Ru_{N-1}^* + B^TP_N(Ax_{N-1} + Bu_{N-1}^*) = 0 \tag{2.29}$$
$$\implies u_{N-1}^* = -(R + B^TP_NB)^{-1}B^TP_NAx_{N-1} \tag{2.30}$$

Now $J(x_{N-1})$ can be reloaded using the above equation as

$$J^*(x_{N-1}) = \left[x_{N-1}^TQx_{N-1} + (K_{N-1}x_{N-1})^TR(K_{N-1}x_{N-1}) \right.$$
$$\left. + (Ax_{N-1} - BK_{N-1}x_{N-1})^TP_N(Ax_{N-1} - BK_{N-1}x_{N-1})\right]$$

with

$$K_{N-1} = -(R + B^TP_NB)^{-1}B^TP_NA$$

$J^*(x_{N-1})$ can now be rewritten as

$$J^*(x_{N-1}) = x_{N-1}^TP_{N-1}x_{N-1} \tag{2.31}$$

with

$$P_{N-1} = Q + K_{N-1}^TRK_{N-1} + (A - BK_{N-1})^TP_N(A - BK_{N-1})$$

The above expression now can be used to compute u_{N-2}^* by looking at $J(x_{N-2})$ similar to the derivation above. This concludes the derivation using DP approach

Hamiltonian

We can also derive the equations using Hamiltonian and the associated optimality conditions. The derivation proceeds as below. Let's use the cost function defined by (2.23).

For this form of derivation, we redefine the cost function as (represent $J(x)$ as J)

$$J = \sum_{k=0}^{N-1} \left[x_k^T Q x_k + u_k^T R u_k + \lambda_{k+1}^T (A x_k + B u_k - x_{k+1})\right] + x_N^T P_N x_N$$

In essence, you are absorbing the equality constraint (the model) into the cost-function and λ_k's are called the co-state variables and are a function of time/step. The Hamiltonian is defined as

$$H_k(x_k, u_k, \lambda_{k+1}) = H_k = x_k^T Q x_k + u_k^T R u_k + \lambda_{k+1}^T (A x_k + B u_k) \tag{2.32}$$

The cost function now is

$$J = x_N^T P_N x_N + \sum_{k=0}^{N-1} \left[H_k - \lambda_{k+1}^T x_{k+1})\right] \tag{2.33}$$

Next we we compute δJ and set it to zero.

$$\delta J = \frac{\partial J}{\partial x}\delta x + \frac{\partial J}{\partial u}\delta u \tag{2.34}$$

$$\implies \delta J = P x_N - \lambda_N + \sum_{k=0}^{N-1} \left[\frac{\partial H_k}{\partial x} - \lambda_k^T + \frac{\partial H_k}{\partial u})\right] \tag{2.35}$$

For δJ to be zero (first order necessary conditions) and as λ is arbitrary, we choose them to make $\delta J = 0$ that is

$$\lambda_N = P x_N \tag{2.36}$$

$$\frac{\partial H_k}{\partial u} = 0 \tag{2.37}$$

$$\lambda_k = \frac{\partial H_k}{\partial x}^T \tag{2.38}$$

which results in

$$\lambda_N = P x_N \tag{2.39}$$

$$u_k^T R^T = -\lambda_{k+1}^T B \tag{2.40}$$

$$\implies u_k = -R^{-1} B^T \lambda_{k+1} \tag{2.41}$$

$$\lambda_k = \lambda_{k+1}^T A + x_k^T Q^T \tag{2.42}$$

$$\implies \lambda_k = A^T \lambda_{k+1} + Q x_k \tag{2.43}$$

We chose $\lambda_k = P_k x_k$ (as we expect the control to be a function of states and control as given in the above equations is a function of λ). From the above equations, we have

$$Ru_k = -B^T\lambda_{k+1} \tag{2.44}$$
$$\implies Ru_k = -B^T P_{k+1} x_{k+1} \tag{2.45}$$
$$\implies Ru_k = -B^T P_{k+1}(Ax_k + Bu_k) \tag{2.46}$$
$$\implies u_k = -(R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A x_k \tag{2.47}$$

The co-state equations are

$$\begin{aligned}\lambda_k &= A^T\lambda_{k+1} + Qx_k \\ &\implies P_k x_k \\ &= A^T P_{k+1}(Ax_k + Bu_k) + Qx_k \\ &\implies P_k x_k \\ &= A^T P_{k+1} A x_k - A^T P_{k+1} B (R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A x_k + Qx_k\end{aligned} \tag{2.48}$$

The above equation holds for $\forall k$. So

$$P_k = A^T P_{k+1} A - A^T P_{k+1} B (R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A + Q \tag{2.49}$$

This completes the derivation. The two forms derived above are equivalent - also called the algebriac Riccati equation.

2.2.4 Relation to H_2 controller

The standard setup for H_2 or H_∞ control is as follows

$$\dot{x}(t) = Ax(t) + B_w w(t) + B_u u(t) \tag{2.50}$$
$$z(t) = C_z x(t) + D_{zw} w(t) + D_{zu} u(t) \tag{2.51}$$
$$y(t) = C_y x(t) + D_{yw} w(t) + D_{yu} u(t) \tag{2.52}$$

where y is the output of the system, w is the exogenous input (disturbance), u is the control input and x is the state. z is a virtual output that captures the performance index to be optimized. Typically, H_2 and H_∞ norms over a class of disturbance inputs are used to compute the transfer function/controller K from output y to input u such that the system is stabilized while minimizing the performance index.

Look at the performance index z. Typically norm of z with H_2 criteria $= \|z\|_2$ and no exogenous input i.e. $w(t) = 0 \forall t$ and full state feedback and assuming the system is strictly causal (i.e $D_{yu} = 0$) becomes similar to a LQR controller. In particular, with $C_{zx} = [\sqrt{Q} \quad 0]^T$ and $D_{zu} = [0 \quad \sqrt{R}]^T$ and will result in H_2 norm $(z^t z)$ which is the same as the cost function $\int_0^\infty (x^T Q x + u^T R u) dt$ (infinite horizon LQR). This is also tied to LQG optimal control where the measured output is corrupted by noise and we need to estimate the states from measurements.

2.3 MPC

2.3.1 Stability

2.4 Other

2.4.1 Feedback Linearization

2.4.2 Lyapunov Controller

2.4.3 Notes on Stability

Chapter 3

Path Planning

Chapter 4

Trajectory Planning

Chapter 5

Behavior Planning

Chapter 6

Architectures

Chapter 7

General Notes

7.1 Kinematics and Dynamics

7.2 Probability

7.3 Geometry

7.4 Optimization

7.5 Control

7.6 Programming & Algorithms