L1 Introduction & Fundamentals

A **robot** is a machine that can perform a complicated series of tasks automatically

Control is a process of managing the behavior of devices or systems

Kinematics is a study of motion with disregard to forces/torques that cause motion

Dynamics is a study of motion and forces/torques that cause motion

1.1 Fundamentals

Degrees of freedom of a mechanical system define the number of independent parameters that describe its configuration

A rigid body in space has 6 DoF (3 positions, 3 rotations)

$$\text{DOF of robot} = \text{total possible} - \text{number of constraints} \tag{1.1}$$

Mechanical joints add constraints to the configuration space

Joints can be actuated (e.g. by a motor in a robotic arm) or non-actuated (e.g. by a ball joint in a driving simulator)

Common types of mechanical joints:

1. Revolute (1R)
2. Prismatic (1T)
3. Universal (2R)
4. Ball (3R)

1.1.1 Velocity Constraints

A constraint that reduces the total DoF of a ridged body is called a **holonomic constraint**

A constraint that reduces the total possible velocities but does not reduce the total DoF is called a **non-holonomic constraint**

A non-holonomic system is a physical system whose configuration depends on the path taken in order to achieve that configuration

Typically, in a non-holonomic system, the number of actuated DoF is less than the total DoF

1.1.2 Pose

To control the robot's behavior. we must be able to describe the configuration of its parts

The position and orientation of the part's coordinate frame are called **pose**

The position in 3D Cartesian space can be described by a vector containing x, y, and z-axis components with respect to the origin of the base frame

The orientation in 3D space can be described by various mathematical tools:

1. Euler angles
2. Axis-angle
3. Quaternions
4. Rotation matrix

1.1.3 Euler Angles

Euler angles describe the orientation by three subsequent rotations along the principal axes of the coordinate frame

These rotations can be **extrinsic** or **intrinsic**

In the extrinsic case, the rotations are applied with respect to the original starting frame. In the intrinsic case, the rotations are applied with respect to the currently rotated frame

+ simplicity
+ few parameters (3)
- potential loss of DoF
- subsequent rotations around multiple axes
- difficult to interpolate

1.1.4 Axis-Angle

Axis-angle describes the orientation by a unit vector e that indicates the direction of an axis of rotation and a rotation θ

+ no gimbal lock
+ rotation around single axis
+ simplicity
+ few parameters (4)
- Non-unique solutions

1.1.5 Unit Quaternions

Unit quaternions give a simple way to encode this axis-angle representations in four parameters:

$$\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k \tag{1.2}$$

+ no gimbal lock
+ rotation around a single axis
+ more computationally efficient
- non-unique solutions
- hard to imagine and visualize

1.1.6 Rotation Matrix

A rotation matrix is an orthogonal 3x3 matrix describing a rotation between two Cartesian coordinate frams:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{1.3}$$

+ unique representation
- more parameters (9)
- computationally more expensive

A homogeneous transformation matrix is a way to describe a pose

It is a 4x4 matrix that combines the rotation matrix and the position vector:

$$\mathbf{T} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

Multiplication of several transformation matrices produces subsequent transformations between coordinate frames (useful for multiple joints)

L2 Recap of Control Theory

2.1 Mathematical Model of Systems

We mainly focus on continuous time dynamical systems:

$$\frac{dx(t)}{dt} = f(x,t,u) \tag{2.1}$$
$$y(t) = g(x,t,u) \tag{2.2}$$

With x the internal state of the system, u the input, y the output, and t the time. If the time derivative and the output depend linearly on the states and the input (f and g are linear functions) then:

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t) \tag{2.3}$$
$$y(t) = Cx(t) + Du(t) \tag{2.4}$$

This system is linear.

2.1.1 State Space Model

The state space of an LTI system is:

$$\dot{x} = Ax + Bu \tag{2.5}$$
$$y = Cx + Du \tag{2.6}$$

With the state vector x, control input vector u, state matrix A, control input matrix B, output vector y, output matrix C, and feedthrough matrix D.

The state vector and matrix are defined by physics, the output vector by the user.

2.1.2 Trajectory of Linear Systems

Given an initial state $x(0)$ and an input $u(t)$, the trajectory of a linear system is unique:

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\xi)}Bu(\xi)d\xi \tag{2.7}$$

The first part is the free movement, the integral the forced movement.

$$e^{At} = I + At + \frac{1}{2!}A^2t^2 + \frac{1}{3!}A^3t^3 + \dots \tag{2.8}$$

2.1.3 Stability of Linear Systems

The internal stability of linear systems depends only on the free movement. Consider the first order system:

$$\dot{x} = ax \tag{2.9}$$
$$x(t) = e^{at}x(0) \tag{2.10}$$

Then:

1. $a < 0$: asymptotically stable
2. $a = 0$: simply stable
3. $a > 0$: unstable

The above classsification can be generalized to higher order systems with the concep of eigenvalues of the state matrix A:

$$\Delta_A(\lambda) = \det(\lambda I - A) \tag{2.11}$$
$$\Delta_A(\lambda_i) = 0 \tag{2.12}$$

In general, an eigenvalue can be complex. If all eigenvalues are in the left-hand plane ($\Re(\lambda_i) < 0$), the system is asymptotically stable. If at least one eigenvalue is in the right-hand plane, the system is unstable. If there are eigenvalues on the imaginary axis ($\Re(\lambda_i) = 0$), check multiplicity (the number of times the eigenvalue appears as a root of the characteristic polynomial).

The position of the eigenvalues also determines the free movement.

2.2 Laplace Transform

The Laplace transform is particularly convenient to study dynamical systems because it is a linear operator that transforms differential relations into algebraic relations.

$$y(t) \to Y(s) \tag{2.13}$$
$$y'(t) \to sY(s) \tag{2.14}$$

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t) \qquad y(t) = Cx(t) + Du(t) \tag{2.15}$$
$$sX(s) = AX(s) + BU(s) \qquad Y(s) = CX(s) + DU(s) \tag{2.16}$$
$$(sI - A)X(s) = BU(s) \tag{2.17}$$
$$X(s) = (sI - A)^{-1}BU(s) \qquad Y(s) = (C(sI - A)^{-1}B + D)U(s) \tag{2.18}$$

$$Y(s) = \frac{N(s)}{D(s)}U(s) = \frac{\beta_0 s^n + \beta_1 s^{n-1} + \cdots + \beta_n}{s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n}U(s) \tag{2.19}$$

$\frac{N(s)}{D(s)}$ is called the transfer function, it takes an input $U(s)$ and gives an output $Y(s)$. The transfer function defines the input-output behavior of the system.

2.2.1 Transfer Function

From the transfer function, the following properties are defined:

1. poles: the values of s for which $D(s) = 0$
2. zeros: the values of s for which $N(s) = 0$
3. static gain: $N(0)/D(0)$ if it exists finite

In the most general case, the transfer function can be a transfer matrix.

2.2.2 Poles

If there is no numerator-denominator cancellation, the poles are the same at the system eigenvalues. They determine the stability properties as before.

2.2.3 Frequency Response

Given $u(t) = A\sin(\omega t)$, $y(t) = A|G(j\omega)|\sin(\omega t + \arg(G(j\omega)))$. It becomes very handy to study the behavior of the transfer function G evaluated along the imaginary axis $s = j\omega$. This is frequency response.

2.3 Mass-Spring-Damper System

Given:

1. Spring force is proportional to the displacement of the mass, x
2. Viscous damping force is proportional to the velocity of the mass, $\dot{x}$
3. Both forces oppose the motion of the mass, m
4. Note also that x_0 corresponds to the position of the mass when the spring is unstretched
5. F is an external input

The system dynamic can be written as:

$$\sum F = F(t) - c\dot{x} - kx = m\ddot{x} \tag{2.20}$$

Assuming $f(t) = 0$, the characteristic equation is:

$$m_b s^2 + cs + k = 0 \tag{2.21}$$

Then:

1. Natural frequency: $\omega_n = \sqrt{\frac{k}{m}}$
2. Critical damping: $c_c = 2\sqrt{km} = 2m\sqrt{\frac{k}{m}} = 2m\omega_n$
3. Damping ratio: $\xi = \frac{c}{c_c}$

This system can be analyzed in the time domain for different values of the damping ratio (assuming a certain value for $x(0)$). Then:

1. Underdamped: $\xi < 1$ (poles in imaginary plane)
2. Critically damped: $\xi = 1$ (pole (1) on real axis in imaginary plane)
3. Overdamped: $\xi > 1$ (poles on real axis in imaginary plane)
4. Undamped: $\xi = 0$ (poles on imaginary axis)

2.3.1 State Space Model

We choose the position and velocity as state variables:

$$x = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{2.22}$$

The system dynamics can be rewritten as:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F \tag{2.23}$$

Since we are primarily interested in displacement, the output equation is:

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{2.24}$$

The transfer finction is:

$$\frac{x(s)}{F(s)} = \frac{1}{ms^2 + cs + k} \tag{2.25}$$

This can be examined using a pole map.

The poles location defines the system response.

2.3.2 Step Response

The step response provides important information regarding system response. The steady state value is $1/k$. The settling time is the time required for the system output to fall within a certain percentage of the steady state value for a step input. The rise time is the time required for the system output to rise from some lower level of 10% to some higher level of 90% of the final steady state value. The percent overshoot is the percent by which a system's step response exceeds its final steady state value.

2.3.3 Bode Plot

The bode plot helps understand the system response in the frequency domain. At low frequencies, the input and output are equal in magnitude, and in phase shown by a constant magnitude and phase of zero.

At the frequency crossed to the resonance frequency, the output gets larger than the input, with some growing phase lag.

As frequency increases further, the output decreases. At high frequencies (near -180 degrees) the two waveforms are completely out of phase, when one is at a maximum, the other is at a minimum.

2.4 Control System Design

1. Identify signals and elements, and the system structure (y, r, u, d). Also, there is the plant, controller, actuator, sensor, etc.
2. Develop a control-design model of the plant.
3. Identify the controller structure and algorithm, e.g. P, PD, PI, PID, etc.
4. Establish specifications for the closed-loop system (settling time, steady-state error, pole locations)
5. Determine the controller gains to meet specifications (step response, bode, etc.)
6. Evaluate the closed-loop system using simulation

Every plant is characterized by three groups of variables: control inputs, disturbances and outputs.

Controller design requires a dynamic control-design model that relates the output y to the inputs u and d.

The goal of the controller design is that the outputs take on desired values, despite the disturbances that might be acting on the plant.

2.4.1 Feedback Control

Feedback Control is the most effective way. The idea is to measure the output, determine the deviation of the output from the desired value, calculate a control signal that eliminates the deviation, and enforce this control signal at the input of the plant.

$$y = P(Ce + d) = P[C(r - y) + d] \tag{2.26}$$

$$y = \frac{PC}{1 + PC} r + \frac{P}{1 + PC} d \tag{2.27}$$

If we assume that $PC = L = \frac{N(s)}{D(s)}$, then $y = \frac{L}{1+L} r = \frac{N(s)}{N(s)+D(s)} r$.

Stability depends on both the numerator and denominator of L and we need to evaluate the poles.

A large loop gain ensures:

+ The output closely follows the reference signal
+ Desensitization of output due to disturbance
+ Desensitization of the I/O map from r to y due to plant variations

but it also can lead to:

- Amplification of measurement noise

- Instability of the closed-loop system

2.4.2 PID Controller

- **P**roportional
- **I**ntegral
- **D**erivative

PID is the most widely used feedback algorithm in the industry. It is a simple, non-model-based algorithm. The P term acts on the error between the actual and reference value of the controlled variable. The I term eliminates a residual error of the proportional term in a steady state. The D term can anticipate the changes and reduces overshoots.

$$u = K_p(x_r - x_m) + K_I \int (x_r - x_m)dt + K_D \frac{d(x_r - x_m)}{dt} \tag{2.28}$$

With higher P values, the poles move up, natural frequency is increased, and the damping ratio is decreased, resulting in an increase in oscillations.

An increase of the D term leads to left shifted and damped poles, decrease of system settling time, and sensitivity to noise.

With higher I values, a third slower pole is added to the system, and the asymptote of the complex poles is closer to the imaginary axis. The system poles are shifted to the right, possibly resulting in instability.

PID tuning is simplest done using the Ziegler-Nichols step response method: examine the open-loop behavior under a step input, and the output is required to be roughly S-shaped.

$$H_c(s) = K_p(1 + \frac{1}{\tau_1 s} + \tau_D s) \tag{2.29}$$

2.4.3 LQR Controller

The state feedback control design approach is a fundamental tool in the control of systems, but is not always useful: the translation from design spceifications to desired poles is not always direct, and in the MIMO system, the state feedback gains that achieve a given pole configuration are not unique.

LQR is a structured way to design state feedback control by the trade-off between convergence rate and cost of control input.

LQR provides optimally controlled feedback gains to enable the closed-loop stable and high performance design of systems.

Before starting a search, the costs must be defined, and a mathimatical scale must be used to quantify what best means:

$$\dot{x} = Ax + Bu \tag{2.30}$$

$$J = \int_0^\infty (x^T Qx + u^T Ru)dt \tag{2.31}$$

The optimal state feedback is given as the minimum J, which can be solved by:

$$PA + A^T P - PBR^{-1}B^T P + Q = 0 \tag{2.32}$$

2.4.4 MPC Controller

The concept of Model Predictive Control (MPC):

1. Identify reference to be tracked
2. Measure system's state
3. Predict the system evolution for the next Np steps
4. Optimize

1. Define the prediction model
2. Define the prediction horizon T_p
3. Define state and actuator constraints
4. Minimize the objective cost function

L3 Robotic Manipulator Control

A robotic manipulator is an actuated mechanical structure that is governed by a controller with the purpose of manipulating the environment.

The robot endpoint is the point where the robot interacts with the environment.

The robot base is the point where the robot is attached to the ground.

Mobile robots have bases which are not attached to the ground.

Position control refers to the action of deriving actuation inputs to minimize the difference between the desired reference position and the actual measured position.

Force control refers to the action of deriving actuation inputs to minimize the difference between the desired reference force and the actual measured force.

Position controller can be applied when we need to move the manipulated object. Force controller can be applied when we need to maintain stable contact with the environment.

Robot control can be divided into endpoint control in Cartesian space, and joint control in joint space.

Tasks are typically performed by the endpoint in Cartesian space.

Since robots are actuated in joints by motors, the robots require additional controllers for each joint.

3.1 Basic Control Theory

3.1.1 Open-Loop Control

Open-loop control is the simplest type of control. The control command is derived by a feed-forward controller, and does not care whether the actual controlled variable becomes equal to the reference variable.

+ A sensor to measure the controlled variable is not required
- Can not guarantee that the desired reference value is achieved
- Requires a model of the plant

3.1.2 Closed-Loop Control

Closed-loop control incorporates feedback about the state of the plant. The control command is derived by a feedback controller that tried to make the actual controlled variable equal to the desired reference variable.

+ Makes adjustments so that the desired reference value is achieved
+ Does not require a model of the plant
- A sensor to measure the controlled variable is required
- If control parameters are not properly set, it can make the system unstable

3.1.3 Combined Control

Feedback and feed-forward controllers can be combined for improved performance.

+ Uses a model of the plant to predict the rough desired control value, making an easier job for the feedback controller
+ Feedback controller makes fine adjustments and compensates for model inaccuracies and external disturbances
- More complex, requires model of plant and a sensor

3.2 Stability Analysis

Again assess the Bode plots. They are a powerful and visually intuitive tool for designing and analyzing systems across the operational range.

The system's output amplitude can decrease with increasing frequency of the input, and the output phase can converge to -180 degrees with increasing frequency of the input. This makes the system's output less responsive to the input at higher frequencies. The 180 degree delay essentially inverts the output compared to the input, and can make a stable negative feedback loop unstable positive if it is fed back in a closed-loop control.

The closed-loop instability condition is given by:

$$\frac{x}{x_r} = \frac{KG}{1+KG} = \infty \rightarrow 1 + KG = 0 \tag{3.1}$$

$$KG = -1 \tag{3.2}$$

This is dominated by the open-loop transfer function, and shows the inverted phase (negative). In the Bode plot, check how much gain margin until the condition is satisfied. Mark the gain 1 point (0 dB). Check how much phase margin at the condition.

Bode plots of open-loop transfer function should be used to analyze closed-loop stability where phase and gain margins are employed. An overview of the whole system can be seen by looking at Bode plots derived by the closed-loop transfer function.

Closed-loop stability can also be analyzed by the Nyquist plot using an open-loop transfer function of the system, however Bode plots can be visually more intuitive, and derived empirically through measurements.

3.3 Position-Controlled and Torque-Controlled

The main usefulness of robots comes from their ability to manipulate the environment with their endpoint and perform various tasks. Nevertheless, the robots are actuated in the joints by motors. Most of the robots are actuated by electric motors, but others also exist. The type of actuation system in the joints also affects the endpoint control.

3.3.1 Position-Controlled Motor

A position-controlled motor controls the reference position or velocity by an electrical current based on the position feedback from an encoder. It does not have a torque sensor and therefore cannot control the torque.

$$I = K(q_c - q_m) \tag{3.3}$$

If you try to move the load with your hand, the motor will rigorously oppose the movement, since the controller tries to maintain the reference position. Essentially, the controller does not care about the dynamics of the system, the load, or any external perturbation.

The motors in joints control joint positions or joint velocities by an electrical current based on the position feedback from the encoders. The input is the commanded joint position or velocity and the output is the measured joint position.

3.3.2 Torque-Controlled Motor

A torque-controlled motor controls the reference torque by an electrical current based on feedback from a torque sensor.

Since a specific reference torque can be controlled, the system and the load dynamics can be compensated.

$$I = K(\tau_c - \tau_m) \tag{3.4}$$

$$\tau_c = \tau_{load} \tag{3.5}$$

Imagine commanding the torque that is equal to the torque required to compensate the gravity of the load. If you try to move the load with your hand, the motor will not oppose the movement. When you stop moving, the load will remain at that position.

We can add torque for joint position control on top of the torque for gravity compensation of the load. Unlike in a position-controlled motor, the gravity compensation and position control can be decoupled.

$$\tau_c = \tau_{load} + K_q(q_r - q) \tag{3.6}$$

The motors in joints control joint torques based on feedback from torque sensors. The input is the commanded joint torque and the output is the measured joint position. Additionally, the torque that compensates the robot dynamics can be added to the commanded torque.

3.4 Endpoint Position Control - Position-Controlled

We have a scenario where we want the robot's actual endpoint position x to follow a reference position x_r. The measured joint position q is obtained from the encoders in the joints and transformed into endpoint position x by forward kinematics:

$$x = fk(q) \tag{3.7}$$

The actual endpoint position is compared to the desired reference endpoint position x_r and multiplied by a proportional gain K_p to obtain a velocity that moves the actual position toward the reference:

$$\dot{x} = K_p(x_r - x) \tag{3.8}$$

Finally, since the robot takes joint velocity as an input, the endpoint velocity is transformed into the joint velocity:

$$\dot{q} = J^{-1}\dot{x} \tag{3.9}$$

3.4.1 Step Response

A step response test is a standard tool to investigate the properties of a controller. Overshoot is an occurrence when the controller moves the actual value beyond the reference value. Steady-state error is the difference between the reference value and the actual value after the controller settles down. Instability is a condition when oscillations fail to decrease and are amplified with time. Additional properties are rise time, settling time, and time constant.

You need to select the right gains for given conditions. Improper selection of PID parameters can make the system unstable. General guidelines for manual tuning are given by Ang (2005), see slide 48.

3.5 Endpoint Position Control - Torque Controlled

We have the same scenario as before, but now we have a torque-controlled robot. The actual endpoint position is compared to the desired reference endpoint position x_r and multiplied by a stiffness K to obtain an external force that moves the actual position towards the reference:

$$F = K(x_r - x) \tag{3.10}$$

Since the robot takes joint torque as an input, the endpoint force is transformed into the joint torque:

$$\tau = J^T F \tag{3.11}$$

Additionally, the dynamics of the robot can be compensated by a torque obtained by the dynamical model. The total torque vector that is sent to the motor controllers is:

$$\tau_c = \tau + \tau_{dyn} \tag{3.12}$$

3.6 Endpoint Force Control - Torque Controlled

We have a scenario where we want the robot's actual endpoint force F to follow a reference force F_r. The actual endpoint force is compared to the desired reference endpoint force F_r and multiplied by a proportional gain K_F to obtain an external force that produces the desired reference force:

$$F = F_r + K_F(F_r - F_m) \tag{3.13}$$

The external force is then controlled by the joint torque:

$$\tau = J^T F \tag{3.14}$$

As before, the dynamics of the robot can be compensated by a torque obtained by the dynamical model in order to improve the endpoint controller performance:

$$\tau_c = \tau + \tau_{dyn} \tag{3.15}$$

3.7 Endpoint Force Control - Position-Controlled

We have a scenario where we want the robot's actual endpoint force F to follow a reference force F_r. The actual endpoint force is compared to the desired reference endpoint force F_r and multiplied by a proportional gain K_F o obtain an endpoint velocity that produces the desired reference force:

$$\dot{x} = K_F(F_r - F)m \tag{3.16}$$

The external force is then controlled by the joint velocity:

$$\dot{q} = J^{-1}\dot{x} \tag{3.17}$$

Unlike torque-controlled robots, position-controlled robots cannot decouple dynamics compensation from the endpoint task control.

L4 Robotic Manipulator Kinematics (1)

Kinematics is a study of motion with disregard for forces/torques that cause that motion.

Forward kinematics: calculate endpoint position/motion in Cartesian space from a given joint position/motion.

> **Application**: Get the endpoint state from the measured joint state
>
> **Intuitively**: Where is the endpoint if we know where the joints are?

Inverse kinematics: calculate joint position/motion from a given endpoint position/motion in Cartesian space.

> **Application**: Kinematic control of robot
>
> **Intuitively**: What is the joint configuration if we know where the endpoint is?

4.1 Forward Kinematics

Robots are controlled by joint movements or joint torques, but the task is usually performed at the endpoint.

How does one find the endpoint position x in Cartesian space from the joint configuration q in joint space, and vice-versa?

Forward kinematics transforms the joint configuration in joint space into the endpoint position in Cartesian space. For two DoF, we can derive the endpoint position:

$$x = x_1 + x_2 \qquad x_1 = l_1 \cos q_1 \qquad x_2 = l_2 \cos(q_1 + q_2) \tag{4.1}$$

$$y = y_1 + y_2 \qquad y_1 = l_1 \sin q_1 \qquad y_2 = l_2 \sin(q_1 + q_2) \tag{4.2}$$

The forward kinematics in vector form:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \\ l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) \end{bmatrix} \tag{4.3}$$

The most common more general method to derive the forward kinematics is the Denavit-Hartenberg convention.

4.2 Jacobian Matrix

The Jacobian matrix transforms joint velocities into the endpoint velocity. Elements of the Jacobian matrix are defined as partial derivatives of endpoint variables by joint variables:

$$J = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} \end{bmatrix} \tag{4.4}$$

For the 2 DoF arm:

$$J = \begin{bmatrix} -l_1 \sin(q_1) - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) \\ l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) & l_2 \cos(q_1 + q_2) \end{bmatrix} \tag{4.5}$$

Since J depends on the configuration, it is indicated as $J(q)$.

The Jacobian transforms joint velocities into the endpoint velocity:

$$\dot{x} = J(q)\dot{q} \tag{4.6}$$

The inverse of the Jacobian transforms the endpoint velocity into joint velocities:

$$\dot{q} = J^{-1}(q)\dot{x} \tag{4.7}$$

The transpose of the Jacobian transforms the endpoint force into joint torques:

$$\tau = J^T(q)F \tag{4.8}$$

The inverse of the transpose of the Jacobian transforms joint torques into the endpoint force:

$$F = J^{-T}(q)\tau \tag{4.9}$$

Imagine a one-joint pendulum with a long lever arm, where we are only interested in the magnitude of endpoint velocity or endpoint force. With this long lever arm, a small motion in the joint produces a large motion at the endpoint $\dot{x} = l\dot{q}$, while a small force at the endpoint produces a large torque in the joint $\tau = lF$. In both cases, the relationship depends on the length of the lever arm l, which is essentially what the Jacobian represents.

If the Jacobian is not a square matrix, it can not be inverted. A solution is to use the pseudo-inverse:

$$J^+ = J^T(JJ^T)^{-1} \tag{4.10}$$

for $n > m$, else:

$$J^+ = (J^TJ)^{-1}J^T \tag{4.11}$$

Remember as TNT: Transposed - Not transposed - Transposed.

4.3 Singularity-Robust Position Control

In singular configurations, the robot Jacobian does not have full rank ($\det(J) = 0$), therefore it is not invertible. Pseudo-inverse makes a single invertible matrix invertible, however does not solve the control problem below. The endpoint reference position can be set outside the reachable endpoint workspace. In a singular configuration, no combination of joint velocities can produce an endpoint velocity in the direction of the reference. The endpoint position controller nevertheless tries to reduce the error by increasing joint velocities:

$$\dot{q} = J^{-1}K_p(x_r - x) \tag{4.12}$$

Result is that the robot starts to dangerously oscillate around the singular configuration with high joint speeds.

A solution is to carefully preplan or constrain the robot motion, so that it always avoids singular configurations. A more elegant and general solution is to use the damped least-squares method to make a singularity-robust pseudo-inverse:

$$J^+ = J^T(JJ^T + \lambda I)^{-1} \tag{4.13}$$

$$J^+ = (J^TJ + \lambda I)^{-1}J^T \tag{4.14}$$

The damping coefficient λ has a small value. Singularity-robust pseudo-inverse is then used instead of a normal inverse or pseudo-inverse in the endpoint controller:

$$\dot{q} = J^+K_p(x_r - x) \tag{4.15}$$

L5 Robotic Manipulator Kinematics (2)

5.1 Kinematic Task-Priority Control

Robots with redundant DoF have more joint DoF than endpoint DoF. For example, an industrial robot with 7 joints that uses its endpoint to work in 3D Cartesian space is a robot with redundant DoF. The Jacobian for such a robot has more columns than rows.

The redundancy condition depends on number of robot joint DoF and how we define endpoint space. Joint DoF are defined by the mechanical design, but we can still decrease their number by locking certain joints and not using them. We can define the number of endpoint DoF in Cartesian space based on the task.

Pseudo-inverse of the Jacobian by itself does not give us a direct control over the redundant joint DoF. Direct control over it can be achieved by a task-priority control approach using null space:

$$\dot{q} = J^{+}\dot{x} + (I - J^{+}J)\dot{q}_N \tag{5.1}$$

The expression in parentheses is called null space and does not project any joint motion into the endpoint motion. Null space joint velocity can be anything and will be produced as long as it does not affect the endpoint movement. For example, it can be the movement of the elbow as a secondary endpoint using the elbow Jacobian $q_N = \begin{bmatrix} J_E^{+}\dot{x}_E \\ 0 \end{bmatrix}$.

5.2 Robots with Parallel Kinematic Structure

Robots with parallel kinematic structure have multiple serial kinematic structures linked in a parallel between the base and endpoint. Stewart platform is one of the most common types of parallel robot and is often used for driving or flight simulators. It has six legs and each leg has only one actuated DoF. The other DoFs in each leg are non-actuated and are made of universal joints and spherical joints.

Given a planar example of a parallel robot with two legs. Given then the desired endpoint pose x, derive the leg lengths. x is expressed in the Cartesian coordinate system that has its origin in the base point.

Draw vectors that connect the base to the endpoint through the ground b, leg l, and platform p. The vectors:

$$\vec{x} = \vec{b_n} + \vec{l_n} + \vec{p_n} \tag{5.2}$$

Note that the $\vec{x}$ also implies a constraint on the orientation of the platform.

Since non-actuated DoF in each leg are typically not sensorized, the full state is unknown. Only the leg lengths are known through the actuators of the translational DoFs. Hence, without the vectors for leg and platform, the calculation is not trivial. Multi-legged platforms typically have multiple solutions for a given set of leg lengths. Various optimization or machine learning techniques can be used to derive a specific solution for direct kinematics.

5.3 Serial vs Parallel Robots

Overview of pros and cons on slide 51.

A general equation (Grübler's formula) to calculate the number of endpoint DoF:

$$M = m(N - 1 - J) + \sum_{i=1}^{J} f_i \tag{5.3}$$

Where:

M	is the number of endpoint DoF
m	is the DoF of the space
N	is the number of bodies
J	is the number of joints
f_i	is the DoF of the ith joint

Assumed that all constraints are independent.

L6 Robotic Manipulator Dynamics

Dynamics is a study of motion and forces/torques that cause that motion.

Forward dynamics: calculate motion from given forces/torques.

Application: Simulation of robot plant

Intuitively: How will the simulated robot move, if I apply certain forces/torques?

Inverse dynamics: calculate forces/torques from a given motion.

Application: Dynamic control of robot

Intuitively: What forces/torques do I need to apply to move the robot in the desired way?

6.1 Formulation of Robot Dynamics

The Lagrangian method is a good alternative to the Newton-Euler method to describe the dynamics of a robot. The Lagrangian function is defined as:

$$L(q, \dot{q}) = T(q, \dot{q}) - V(q) \tag{6.1}$$

It can also be derived as a sum of contributions of individual segments:

$$L(q, \dot{q}) = \sum_{i=1}^{n} (T_i(q, \dot{q}) - V_i(q)) \tag{6.2}$$

T kinetic en
V potential

The joint torque in a vector form is defined by:

$$\tau = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} \tag{6.3}$$

Individual joint torques can also be derived separately:

$$\tau_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q_i} \tag{6.4}$$

Example: 2 DoF rigid robotic arm moving in 2D space. Assumptions:

1. The segments are point masses with center of mass at each joint
2. The endpoint also has a mass
3. Links are massless
4. No friction in the joints
5. No external forces on the robot apart from gravity

First, the position and velocity of each segment's center of mass:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} l_1 \cos(q_1) \\ l_1 \sin(q_1) \end{bmatrix} \quad \begin{bmatrix} \dot{x_1} \\ \dot{y_1} \end{bmatrix} = \begin{bmatrix} -l_1 \sin(q_1) \\ l_1 \cos(q_1) \end{bmatrix} \dot{q_1}$$

Similarly for $\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$ and $\begin{bmatrix} \dot{x_2} \\ \dot{y_2} \end{bmatrix}$.

Next, derive the kinetic energy for both segments: $T = \frac{1}{2} m V^2$.

Then, derive the potential energy for both segments: $V = mgh$.

Next, calculate the Lagrangian function and then the individual joint torques. Rigid body dynamics equation:

$$\tau = M(q)\ddot{q} + C(q, \dot{q}) + G(q) \tag{6.5}$$

For longer derivation, see slides 9-14.

6.2 Application of Dynamics in Control

6.2.1 Inverse Dynamics

Standard formulation (above equation) only accounts for the robot dynamics itself. If we want to control a specific part of the robot, like a hand, to perform some manipulation tasks, we need to add endpoint force to the formulation:

$$\tau_c = \tau_{dyn} + J^T(q)F \quad \tau_{dyn} = M(q)\ddot{q} + C(q,\dot{q}) + G(q) \tag{6.6}$$

Endpoint force can then be used to design a Cartesian impedance controller for position-tracking tasks:

$$F = K(x_r - x) - D\dot{x} \tag{6.7}$$

6.2.2 Forward Dynamics

We want to use the derived dynamics to simulate robot movements. Invert the inverse dynamics equation to get the forward dynamics equation:

$$\ddot{q} = M^{-1}(q)(\tau - C(q,\dot{q}) - G(q) - J^T(q)F_{ext}) \quad \tau = J^T(q)F \quad F = K(x_r - x) - D\dot{x} \tag{6.8}$$

To get a new state of the simulated robot, we integrate the calculated acceleration at each step:

$$\dot{q} = \int \ddot{q}dt \quad q = \int \dot{q}dt \tag{6.9}$$

The updated state is then again used to calculate a new acceleration in the next step.

6.3 Controller and Robot Relationship

Impedance takes a measured position/motion as input and controls the force as an output:

$$F = D\dot{x} \tag{6.10}$$

Admittance takes a measured force as input and controls the position/motion as an output:

$$\dot{x} = \frac{1}{D}F \tag{6.11}$$

When controlling a robot with an impedance controller, the robot acts as an admittance, and vice-versa.

6.4 Dynamic Task-Priority Control

Similarly, as in kinematic task-priority control, redundant joint DoF can be controlled by using null space. For the dynamic case, the control can be achieved by:

$$\tau = J^T(q)F + (I - J^T\bar{J}^T)\tau_N \tag{6.12}$$

$$\bar{J} = M^{-1}J^T(JM^{-1}J^T)^{-1} \tag{6.13}$$

The expression in the top equation between parentheses is the null space and does not project any joint torques into the endpoint forces. M is the robot inertia matrix and is used to weigh the joints based on inertia. Null space joint torque can be anything and will be produced as long as it does not affect the endpoint movement. For example, null space joint torque can control the movement of the elbow as a secondary endpoint using a secondary impedance controller.

An alternative approach for dynamic task-priority control using null space is given as:

$$\tau = J^T(q)F + M(I - J^+J)(M(I - J^+J))^+\tau_N \tag{6.14}$$

6.5 Dynamic Simulation and Programming

The main loop in the program refers to the code that is executed repetitively throughout time. For a dynamic simulation, where the total time of the simulation is known, we typically use a for loop. For a robot controller, where the robot operates indefinitely, we typically use a while loop.

Sample time or step time `dt` is the time between the two calculation steps or the duration of one loop. Intuitively, sample time determines how fast the main loop in the program will run, and the sample frequency is $1/dt$. In a simulation, we can pick a sample time as long as the computer is fast enough to handle the required speed of calculation. In real robot control, the sample time is typically regulated by a real-time clock.

Sample time is important for numerical integration inside the main loop. It also is important for the controller inside the main loop.

L7 3D Kinematics

As we know, forward kinematics transforms the joint configuration in joint space into the endpoint position in Cartesian space. We got the 2 DoF forward kinematics in vector form as $\begin{bmatrix} x \\ y \end{bmatrix}$.

Again, with FK we know the joint angles, and determine the endpoint frame. With IK, we have the Cartesian pose for the endpoint frame, and determine the joint angles. For the latter, many combinations exist.

When moving to 3D, we are now in 6 DoF (3T + 3R). We can have many joints, which can rotate, translate, and their axes may be offset and misaligned. As a result, we need some standard methods for organizing and analyzing these complex systems.

7.1 Denavit-Hartenberg Convention

The Denavit-Hartenberg (DH) Convention is one for assigning coordinate frames to a robot manipulator. It works for any serial, open-chain manipulator. It allows us to perform coordinate transformations between each joint. It is used to do FK and IK, and for finding the Jacobian.

It is composed of two translations and two rotations:

1. A translation in Z and a rotation about Z
2. A translation in X and rotation about X

$$ {}^{i-1}_{i}T = Trans_{z_{i-1}}(d_i) \cdot Rot_{z_{i-1}}(\theta_i) \cdot Trans_{x_i}(a_i) \cdot Rot_{x_i}(\alpha_i) \tag{7.1}$$

Or:

$$ Z \cdot X =^{i-1}_{i} T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & \cos\theta_i a_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & \sin\theta_i a_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.2}$$

Applying the convention:

1. Assign coordinate frames
2. Complete the DH-Table
3. Calculate transforms between adjacent joints
4. Calculate transform from base frame to endpoint (FK)
5. Use transforms to calculate the Jacobian

Two different joints exist:

1. Revolute: Rotational
2. Prismatic: Translational

To assign DH Frames:

1. Draw an axis of rotation (or direction of translation) for each join, your z-axis for each joint will be in the direction of these axes.
2. Starting with the first joint frame/base frame, find the line that is mutually perpendicular to the first and next axes.
3. The connection point at the next axis will be the origin for that joint frame with the z-axis pointing along the joint axis and the x-axis pointing along the perpendicular line you found in step 2.
4. If the axes are parallel you may choose the most convenient point of origin, if the axes intersect, the origin is the point of intersection.

To find DH Parameters:

1. First find d the distance along the initial z-axis needed to align it with the new origin.
2. Second, find θ the angle needed to align the previous x-axis with the perpendicular line that wil become the new x-axis.

3. Third, find a the distance along the mutual perpendicular new x-axis needed to put the old frame at the new origin.

4. Finally, find α the angle needed to rotate the frame about the new x-axis so that the z-axis aligns with the axis of rotation for the new joint.

7.1.1 DH Table

The DH table is a table with the links in the rows and the parameters in the columns. It can be filled in using the procedure for finding the parameters for each link.

Homogeneous transformations

The Special Euclidean Group (SE3):

$$ {}^A_BT = \begin{bmatrix} a_1 & b_1 & c_1 & p_x \\ a_2 & b_2 & c_2 & p_y \\ a_3 & b_3 & c_3 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3}$$

Invertable:

$$ {}^A_BT^{-1} = \begin{bmatrix} a_1 & a_2 & a_3 & -(p_xa_1+p_ya_2+p_za_3) \\ b_1 & b_2 & b_3 & -(p_xb_1+p_yb_2+p_zb_3) \\ c_1 & c_2 & c_3 & -(p_xc_1+p_yc_2+p_zc_3) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4}$$

Associative:

$$(T_1T_2)T_3 = T_1(T_2T_3) \tag{7.5}$$

Not Commutative:

$$T_1T_2 \neq T_2T_1 \tag{7.6}$$

The transform ${}^{i-1}_{i}T$ is composed of 2 translations and 2 rotations. We typically combine one rotation and translation into a single homogeneous transform (${}^{i-1}_{i}Z, {}^{i-1}_{i}X$). We can multiply these to get a single homogeneous transform that moves from one frame to the next. Weh we have several frames we can multiply our adjacent frames to give us transformations between frames that are further apart:

$$ {}^0_3T = {}^0_1T \cdot {}^1_2T \cdot {}^2_3T \tag{7.7}$$

We can input the values from the DH table into the equations in the transformation matrices.

Transforms allow us to move between frames using the above equation(s). Find the transforms from the base frame to each of the joint frames. Since we know individual ${}^{i-1}_{i}T$, we can find any ${}^0_iT$ using the multiplications.

Other approaches are the modified DH parameters. We don't consider it, but it should be known that it exist, and that other use it. It is quite similar to the classical DH equations. Another is the product of exponentials.

7.2 Jacobian

The Jacobian transforms joint angle velocities to the velocities of the end effector frame. We can also find Jacobians between other frames that may be of interest to us. We will use the same Jacobian to transform end effector forces to joint torques.

The Jacobian can be calculated as the partial derivatives of the kinematics matrix. This direct calculation is typically infeasible for robotic systems with more than 2 DoF. We will use an indirect approach, velocity propagation, to calculate it for more complex systems.

The Jacobian is a $6 \times n$ matrix comprising the 3T and 3R velocities for each joint. Jacobians can include n number of joints and are typically not square.

Velocity propagation can be thought of as stepping through each joint to consider how the previous joints velocity will impact the current joint. We will do this with some standard equations using the DH table and the corresponding transformation matrices.

For revolute joints, the translational velocity will be calculated as:

$$v_i = z_{i-1} \times (t_n - t_{i-1}) \cdot \dot{q}_i \tag{7.8}$$

The rotational velocity will be calculated as:

$$\omega_i = z_{i-1} \cdot \dot{q}_i \tag{7.9}$$

For prismatic joints:

$$v_i = z_{i-1} \cdot \dot{q}_i \tag{7.10}$$

$$\omega_i = 0 \tag{7.11}$$

From that:

$$J_v = \begin{cases} z_{i-1} \times (t_n - t_{i-1}) \cdot \dot{q}_i & \text{revolute} \\ z_{i-1} \cdot \dot{q}_i & \text{prismatic} \end{cases} \tag{7.12}$$

$$J_\omega = \begin{cases} z_{i-1} \cdot q_i & \text{revolute} \\ 0 & \text{prismatic} \end{cases} \tag{7.13}$$

L8 3D Trajectory Generation & Tracking

From the previous lecture, we know how to calculate the Jacobian using the velocity propagation approach. Also:

$$v = \omega \times r \tag{8.1}$$

Path planning concerns the physical path (set of poses) the robot will follow in space with no consideration for time. It can be in task or joint space.

Trajectory planning considers time information (velocity, acceleration) as well as position to schedule the motion along a path.

Joint space trajectory planning:

1. xyz waypoints
2. IK
3. Joint space trajectory
4. Command joint positions

In the joint space, paths will typically not be straight lines in the Cartesian space. IK is only needed for determining joint angles at the start and end points. Paths can more easily satisfy the workspace requirements and joint limits of the robot.

In Cartesian space, paths shape can be intuitively designed in Cartesian space as desired. IK is needed at every point along a path to determine the robot's joint angles. Joint or workspace violations are possible and the IK solutions should be checked to ensure valid robot configurations.

Important considerations:

1. Start and ending positions
2. Start and ending velocities
3. Start and ending accelerations
4. Jerk
5. Waypoints

Cubic polynomial: define velocity and position at the start and end points:

$$q(\tau) = \alpha_3\tau^3 + \alpha_2\tau^2 + \alpha_1\tau + \alpha_0 \tag{8.2}$$
$$\dot{q}(\tau) = 3\alpha_3\tau^2 + 2\alpha_2\tau + \alpha_1 \tag{8.3}$$

$$q(0) = 0 \qquad v(0) = 0 \tag{8.4}$$
$$q(T) = \pi/4 \qquad v(T) = 0 \tag{8.5}$$

$$q(0) = \alpha_0 \tag{8.6}$$
$$q(T) = \alpha_3T^3 + \alpha_2T^2 + \alpha_1T + \alpha_0 \tag{8.7}$$
$$\dot{q}(0) = \alpha_1 \tag{8.8}$$
$$\dot{q}(T) = 3\alpha_3T^2 + 2\alpha_2T + \alpha_1 \tag{8.9}$$

Thus:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & T & T^2 & T^3 \\ 0 & 1 & 2T & 3T^2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \pi/4 \\ 0 \end{bmatrix} \tag{8.10}$$

This can be solved for $\vec{\alpha}$ for a given T.

It can also be done with a 5th order polynomial.

On the other hand, trapezoidal trajectories can be used. The velocity curve looks like a trapezoid. Ramp-up and ramp-down velocity states with a constant velocity state in the middle of the trajectory. The position is parabolic for the ramp-up and ramp-down and linear for the constant velocity state. Acceleration will have discontinuities as the velocity transitions between states.

S-curve trajectories have 7 different parts of the curve. Parts 1, 3, 5, and 7 have a constant jerk, parts 2 and 6 have a constant acceleration, and part 4 has a constant velocity.

Handling via points helps when we want to move to more than a single point in space without coming to a complete stop. We can specify the velocity at each via point and connect with cubic trajectories. It may be difficult to choose these velocities, and we will end up with discontinuities in our accelerations. Alternatively, we can match our velocities and accelerations at each point. Joint constraints may not always be satisfied, but we get smooth trajectories. B-splines allow us to get very close, but we will not hit our via points exactly.

8.1 Orientation

We want to control orientation and even plan orientation trajectories. Like position control, we want to calculate the error of our rotational degrees of freedom. Calculating error between rotation matrices however is not so straightforward. We need to move from rotation matrices to some other expression of our orientation.

Controlling Cartesian position is fairly intuitive. We have some transform from the base to the endpoint. And with that, some known endpoint position. If we have a desired position and a known position, calculating our error is very straightforward.

To control our orientation both statically or during a trajectory, we also need a some way to calculate our error. We can get our rotation matrices from our kinematics. But we cannot subtract rotation matrices to calculate the error. Instead, we need to consider other methods for express rotation and calculate our orientation error.

Axis and Angle representation is another way to express rotation/orientation of a body. We consider and axis r and an angle of rotation θ. We can move from rotation matrices to axis and angle expression using the equations:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{22} & r_{33} \end{bmatrix} \tag{8.11}$$

$$r = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{8.12}$$

$$\theta = \cos^{-1}(\frac{r_{11} + r_{22} + r_{33} - 1}{2}) \tag{8.13}$$

We can also move back. Long equations, see slide 66.

Axis and angle representation is another way to express rotation/orientation of a body. It is not subject to the same limitation as rotation matrices or Euler angles. We do not have to worry about gimbal lock. One limitation is that any axis and angle can be flipped, since $R(-\theta, -r) = R(\theta, r)$.

Now, the orientation error between two frames can be expressed as:

$$e_O = r\sin\theta \text{ for } -\pi/2 < \theta < \pi/2 \tag{8.14}$$

Where r and θ represent the axis and angle of the rotation matrix:

$$R(\theta, r) = R_{ref}R_{act}^T \tag{8.15}$$

We consider our rotation matrices to be built from 3 vectors:

$$R_{ref} = \begin{bmatrix} n_{ref} & s_{ref} & a_{ref} \end{bmatrix} \tag{8.16}$$

$$R_{act} = \begin{bmatrix} n_{act} & s_{act} & a_{act} \end{bmatrix} \tag{8.17}$$

And with some derivation from our axis angle form rotation matrix definition we can come up with a functional expression of our orientation error:

$$e_O = \frac{1}{2}(n_{act} \times n_{ref} + s_{act} \times s_{ref} + a_{act} \times a_{ref}) \tag{8.18}$$

With this expression for orientation error, we can calculate our error so long as:

$$n_{act}^T n_{ref} \geq 0 \tag{8.19}$$
$$s_{act}^T s_{ref} \geq 0 \tag{8.20}$$
$$a_{act}^T a_{ref} \geq 0 \tag{8.21}$$

The error is important for orientation control. It can also be useful in planning trajectories. We can interpolate between axis and angle expressions, but not rotation matrices. Interpolation is limited to the constraints as noted in the equations.

We would like to improve our orientation planning to allow us to interpolate between any possible orientations. Quaternions allow us to do this in a very nice way.

$$q = (q_0, q_1, q_2, q_3) \tag{8.22}$$

$$q_0 = \frac{1}{2}\sqrt{r_{11} + r_{22} + r_{33} + 1} \tag{8.23}$$
$$q_1 = \frac{1}{2} sgn(r_{32} - r_{23})\sqrt{r_{11} - r_{22} - r_{33} + 1} \tag{8.24}$$
$$q_2 = \frac{1}{2} sgn(r_{13} - r_{31})\sqrt{r_{22} - r_{33} - r_{11} + 1} \tag{8.25}$$
$$q_3 = \frac{1}{2} sgn(r_{21} - r_{12})\sqrt{r_{33} - r_{11} - r_{2} + 1} \tag{8.26}$$

Again, reverse is also possible. Slide 77 for equations.

It is important to make sure that any quaternion you are using has been normalized. This can be done exactly the same way we normalize a vector: find the magnitude and divide each part of the quaternion by it. Quaternions that do not have a magnitude of 1 will not accurately represent a 3D rotation, they may result in scaling/skewing errors.

Spherical Linear Interpolation (SLERP) can allow us to more easily interpolate between orientations. It interpolates between quaternions about a sphere. It is not subject to singularities (gimbal lock) or other constraints as seen in axis and angle approach, and maintains constant angular velocity.

Given two quaternions q_0 and q_1, we can find an arc on the unit sphere and interpolate along this arc. If we have some unit of time T and the half angle distance θ between our quaternions, we can use the following equation to find another quaternion q_2 between them:

$$q_2 = \frac{\sin(1-T)\theta}{\sin(\theta)} q_0 + \frac{\sin(T\theta)}{\sin(\theta)} q_1 \text{ where } \theta = \cos^{-1}(q_0 \cdot q_1) \tag{8.27}$$

L9 Automated Vehicle Kinematics & Dynar

9.1 Wheeled Mobile Robots

Wheels are the most common solution for the majority of applications. Three wheels are sufficient to guarantee stability, which is improved with four. There are many possibilities of wheel configurations when considering possible techniques for mobile robot locomotion. The selection of wheels depends on the application.

Wheel types:

1. Standard wheel: Two DoF, rotation around the (motorized) wheel axle and the contact point
2. Castor wheel: Three DoF, rotation around the wheel axle, the contact point, and the castor axle
3. Swedish / Mecanum wheel: Three DoF, rotation around the (motorized) wheel axle, rollers, and the contact point
4. Ball or spherical wheel

Stability is guaranteed with three wheels if the center of gravity is within the triangle formed by the ground contact point of the wheels. It is improved by 4 and more wheels. These arrangements are hyper-static and require a flexible suspension system.

Bigger wheels allow overcoming of obstacles, but require higher torques or reductions in the gearbox. Most arrangements are non-holonomic, requiring high control effort.

As such, there is a trade-off between maneuverability and controllability. Combining actuation and steering on one wheel increases complexity.

Two wheel configurations generally have one steering wheel and one traction wheel (bike), or two-wheel differential drive (hoverboard). Three wheel configurations include two-wheel differential drive with one unpowered omnidirectional wheel, two connected traction wheels and one steering wheel, and many more. Even more options exist for four-wheel configurations.

9.2 Kinematic Model and Constraints

Each wheel contributes to the robot's motion and imposes constraints on robot motion. Wheels are tied together based on robot chassis geometry.

Main assumptions for kinematics:

1. The robot is built from rigid mechanisms
2. No slip occurs in the orthogonal direction of rolling (no-slipping)
3. No translational slip occurs between the wheel and the floor (pure rolling)
4. All steering axes are perpendicular to the floor

The position P in the global reference frame (X_I-Y_I) is specified by coordinates x and y, and the angular difference between the global and local (X_R-Y_R) reference frames is given by θ.

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{9.1}$$

We need to map motion along the axes of the robot's local reference frame.

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.2}$$

The forward kinematic model describes the motion of a mobile robot in the global reference frame:

$$\dot{\xi}_R = R(\theta)\dot{\xi}_I = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{9.3}$$

The first constraint enforces the concept of rolling contact: the wheel must roll when motion takes place in the appropriate direction. The second constraint enforces the concept of no lateral slippage: the wheel must not slide orthogonal to the wheel plane.

Defining:

- I as the distance from CoG P to wheel center point A
- α as the angle between CoG and wheel center
- β as the wheel orientation angle
- r as the wheel radius

9.2.1 Rolling Constraint

The rolling constraint enforces that the motion along the direction of the wheel plane must be accompanied by the appropriate amount of wheel spin so that there is pure rolling at the contact point:

$$v = \dot{\phi} r = \dot{x}_R \sin(\alpha+\beta) - \dot{y}_R \cos(\alpha+\beta) - \dot{\theta} l \cos\beta \tag{9.4}$$

$$\dot{\phi} r = \begin{bmatrix} \sin(\alpha+\beta) & -\cos(\alpha+\beta) & -l\cos\beta \end{bmatrix} \begin{bmatrix} \dot{x}_R & \\ \dot{y}_R & \dot{\theta} \end{bmatrix} \tag{9.5}$$

Considering the relation between the ξ, the rolling constant can be written as:

$$\begin{bmatrix} \sin(\alpha+\beta) & -\cos(\alpha+\beta) & -l\cos\beta \end{bmatrix} R(\theta)\dot{\xi}_I - r\dot{\phi} = 0 \tag{9.6}$$

9.2.2 Sliding Constraint

The sliding constraint for this wheel enforces that the component of the wheel's motion orthogonal to the wheel plane must be zero:

$$\dot{x}_R \cos(\alpha+\beta) + \dot{y}_R \sin(\alpha_\beta) + \dot{\theta} l \sin\beta = 0 \tag{9.7}$$

The sliding constraint can be written as:

$$\begin{bmatrix} \cos(\alpha+\beta) & \sin(\alpha+\beta) & l\sin\beta \end{bmatrix} R(\theta)\dot{\xi}_I = 0 \tag{9.8}$$

9.2.3 Generalized Wheel Constraints

Given a robot with N wheels:

- Each wheel imposes zero or more constraints on the robot's motion
- Only fixed and steerable standard wheels impose constraints

Suppose we have a total of $N = N_f + N_s$ standard wheels. We can develop the equation for the constraints in matrix forms. Rolling (J_1 - matrix with projections for all wheels to their motions along their individual wheel planes, J_2 - constant diagonal matrix):

$$J_1(\beta_s)R(\theta)\dot{\xi}_I + J_2\dot{\phi} = 0 \tag{9.9}$$

With:

$$\phi(t) = \begin{bmatrix} \phi_f(t) \\ \phi_s(t) \end{bmatrix} \tag{9.10}$$

$$J_1(\beta_s) = \begin{bmatrix} J_{1f} \\ J_{1s}(\beta_s) \end{bmatrix} \tag{9.11}$$

$$J_2 = \text{diag}(r_1, \ldots, r_N) \tag{9.12}$$

For lateral movement:

$$C(\beta_s)R(\theta)\dot{\xi}_I = 0 \tag{9.13}$$

With:

$$C_1(\beta_s) = \begin{bmatrix} C_{1f} \\ C_{1s}(\beta_s) \end{bmatrix} \tag{9.14}$$

9.3 Differential Drive Mobile Robot

Assume:

- Wheel A:
  - $\alpha = \pi/2$
  - $\beta = 0$
- Wheel B:
  - $\alpha = -\pi/2$
  - $\beta = \pi$

The rolling constraints are:

$$\begin{bmatrix}1 & 0 & -l\end{bmatrix} R(\theta)\dot{\xi}_I - r\dot{\phi}_A = 0 \tag{9.15}$$

$$\begin{bmatrix}1 & 0 & l\end{bmatrix} R(\theta)\dot{\xi}_I - r\dot{\phi}_B \tag{9.16}$$

The sliding constraints are:

$$\begin{bmatrix}0 & 1 & 0\\ 0 & 1 & 0\end{bmatrix} R(\theta)\dot{\xi}_I = 0 \tag{9.17}$$

Fusing the two equations yields:

$$\begin{bmatrix}J_1\\ C_1\end{bmatrix} R(\theta)\dot{\xi}_I = \begin{bmatrix}J_2\dot{\phi}\\ 0\end{bmatrix} \tag{9.18}$$

The final kinematic equation of the mobile robot is:

$$\dot{\xi}_I = R^T(\theta)\begin{bmatrix}1 & 0 & -l\\ 1 & 0 & l\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}r\dot{\phi}_A & r\dot{\phi}_B & 0\end{bmatrix} \tag{9.19}$$

9.4 Mobile Robot Maneuverability

The maneuverability of a mobile robot is the combination of the mobility available based on the sliding constraints plus additional freedom contributed by the steering. To avoid any lateral slip the motion has to satisfy:

$$C(\beta_s) = \begin{bmatrix}C_f\\ C_s(\beta_s)\end{bmatrix} R(\theta)\dot{\xi}_I = 0 \tag{9.20}$$

The rank of $C(\beta_s)$ is the number of independent constraints. The degree of mobility is defined by the dimensionality of the null space of $C(\beta_s)$. It is restricted by no-sliding constraints. The greater the rank, the more constrained the mobility:

$$\delta_m = \dim N[C(\beta_s)] = 3 - \text{rank}[C(\beta_s)] \tag{9.21}$$

- No standard wheels: rank is zero
- All motion directions constrained: rank is 3

The degree of steerability is defined as:

$$\delta_s = \text{rank}[C_s(\beta_s)] \tag{9.22}$$

Increase in the rank implies more degrees of steering freedom and greater eventual maneuverability. Since $C(\beta_s)$ includes $C_s(\beta_s)$, this means that a steered standard wheel can both decrease mobility and increase steerability. The range of δ_s can be specified as $[0,\ 2]$. The overall degrees of freedom that a robot can manipulate, called the degree of maneuverability, can be defined in terms of mobility and steerablility:

$$\delta_M = \delta_m + \delta_s \tag{9.23}$$

9.5 Outdoor Mobile Robot / Automated Vehicle

Vehicle handling is the lateral response of a vehicle to control input. The steady-state lateral motion is a core part of handling. Motion regions:

1. Kinematic (low speed)
2. Linear dynamics
3. Nonlinear dynamics
4. Near friction limits

The heading angle ψ is the angle between the trace on the X-Y pane of the vehicle x-axis and the X-axis of the earth fixed system.

The velocity vector V is the vehicle in earth-fixed coordinates, it is always tangent to the path of the vehicle CoG.

The forward velocity u is the component of the vehicle velocity in the x-direction.

The lateral or side velocity v is the component of the vehicle velocity in the y-direction.

The sideslip angle β is the angle between the vehicle x-axis and the vehicle velocity vector.

In stable driving situations, the sideslip angle is small ($< 5°$).

$$\beta = \arctan\left(\frac{v}{u}\right) \approx \left(\frac{v}{u}\right) \tag{9.24}$$

The yaw rate r is the angular rate about the vertical axis.

The lateral acceleration a_y is the component of the vector acceleration of a point in the vehicle perpendicular to the vehicle's x-axis.

9.5.1 Turning at Low Speed

Assuming a bicycle. Inertial forces and centrifugal acceleration are negligible. The problem transforms into pure geometrical. We have an instantaneous center of rotation. A longer wheelbase L results in a wider turn radius R, or more steer input. This geometric steering angle is known as the Ackermann angle.

$$\arctan\left(\frac{L}{R}\right) \approx \frac{L}{R} \tag{9.25}$$

For a four-wheel vehicle, we have the same steer angle for inner and outer tires. This creates tire scrub as the wheels tend to fight each other's motion. Thus, it is necessary to allow different steer angles on inner and outer wheels. The wheels should follow curved paths with radii originating from a common center ICR. For proper steering, the steer angles are given by:

$$\delta_o \approx \frac{L}{R + 0.5B} < \delta_i \approx \frac{L}{R - 0.5B} \tag{9.26}$$

The Ackermann geometry, ignoring kingpin offsets, is defined as:

$$\cot\delta_o - \cot\delta_i = \frac{B}{L} \tag{9.27}$$

9.5.2 Kinematic Bicycle Model

The equations of motion are purely based on geometric relationships governing the system. It is intensively used in motion planning due to its low complexity. It is valid in the case of very small lateral forces.

Using the sine rule to the triangle origin, center, wheel A:

$$\frac{\sin(\delta - \beta)}{l_f} = \frac{\sin(0.5\pi - \delta)}{R} \tag{9.28}$$

And wheel B:

$$\frac{\sin\beta}{l_r} = \frac{\sin(0.5\pi)}{R} \tag{9.29}$$

Expanding for both equations:

$$\frac{\sin\delta\cos\beta - \sin\beta\cos\delta}{l_f} = \frac{\cos\delta}{R} \tag{9.30}$$

$$\frac{\sin\beta}{l_r} = \frac{1}{R} \tag{9.31}$$

Multiply the first by $\frac{l_f}{\cos\beta}$, and combine to obtain:

$$\tan\delta\cos\beta = \frac{l_f + l_r}{R} \tag{9.32}$$

Knowing that $\dot{\psi} = \frac{V}{R}$, we can obtain:

$$\dot{\psi} = \frac{V\cos\beta}{l_f + l +_r}\tan\beta \tag{9.33}$$

And since R is large, we can get:

$$\tan\beta = \frac{l_r}{l_f + l_r}\tan\delta \tag{9.34}$$

Then, the overall equations of motion for the kinematic bicycle model are:

$$\dot{X} = V\cos(\psi + \beta) \tag{9.35}$$

$$\dot{Y} = V\sin(\psi + \beta) \tag{9.36}$$

$$\dot{\psi} = \frac{V\cos\beta}{l_f + l_r}\tan\delta \tag{9.37}$$

$$\beta = \arctan\left(\frac{l_r}{l_f + l_r}\tan\delta\right) \tag{9.38}$$

9.5.3 Tire Slip

The tire slip angle α is the angle between the expected/desired direction and actual direction of travel of the center of tire contact:

$$\alpha = \arctan\left(\frac{V_{yw}}{V_{xw}}\right) \tag{9.39}$$

Assuming a linear relationship between the lateral forces and the slip angle:

$$F_{yf} = C_{\alpha f}\alpha_f \tag{9.40}$$

$$F_{yr} = C_{\alpha r}\alpha_r \tag{9.41}$$

$C_{\alpha f}$ and $C_{\alpha r}$ are cornering stiffness of front and rear axles respectively.

9.6 Steering Characteristics

Assume a vehicle speeds up in a constant radius turn.

1. Neutral steer: The driver should maintain the same steering wheel position. That is, when accelerated with the steering wheel fixed, the turning radius remains the same.
2. Understeer: The driver must increase the steering angle. That is, when accelerated with the steering wheel fixed, the turning radius increases.
3. Oversteer: The driver must decrease the steering angle. That is, when accelerated with the steering wheel fixed, the turning radius decreases.

9.6.1 Steady-State Cornering

Assume front and rear wheels turn about the same turn center. The sum of the inner angles inside the triangle give:

$$\frac{\pi}{2} + \alpha_f - \delta + \frac{\pi}{2} - \alpha_r + \frac{L}{R} = \pi \tag{9.42}$$

Now, substitute for the slip angles:

$$\delta = \frac{L}{R} + \alpha_f - \alpha_r \tag{9.43}$$
$$\delta = \frac{L}{R} + \frac{F_{yf}}{C_{\alpha f}} - \frac{F_{yr}}{C_{\alpha r}} \tag{9.44}$$

The lateral acceleration:

$$a_y = \frac{u^2}{R} \tag{9.45}$$

The yaw rate:

$$r = \frac{u}{R} \tag{9.46}$$

The path curve:

$$\rho = \frac{1}{R} \tag{9.47}$$

Performing a force balance maintaining steady state motion:

$$\sum F_y = F_{yf} + F_{yr} = ma_y = \frac{mu^2}{R} \tag{9.48}$$

Also, the moment balance yields:

$$\sum M_z = l_f F_{yf} - l_r F_{yr} = 0 \tag{9.49}$$
$$\rightarrow F_{yf} = \frac{l_r}{l_f} F_{yr} \tag{9.50}$$

Substituting into the first equation yields:

$$F_{yr} = \frac{l_f}{L}\frac{mu^2}{R} \tag{9.51}$$
$$F_{yf} = \frac{l_r}{L}\frac{mu^2}{R} \tag{9.52}$$

Putting the equations together:

$$\delta = \frac{L}{R} + K_{us}\frac{a_y}{g} \tag{9.53}$$

with the understeer gradient:

$$K_{us} = \frac{mg}{L}\left(\frac{l_r}{C_{\alpha f}} - \frac{l_f}{C_{\alpha r}}\right) \tag{9.54}$$

This has two main contributions:

1. Kinematic
2. Speed or acceleration dependent

To achieve a certain vehicle response, determination of the magnitude and direction of the steering inputs is required.

1. Neutral steer:
   - $K_{us} = 0$

- $\alpha_f = \alpha_r$
- No change in steer angle is required to follow the path

2. Understeer:
   - $K_{us} > 0$
   - $\alpha_f > \alpha_r$
   - Angle difference required: to keep trajectory, the steering angle should be increased
3. Neutral steer:
   - $K_{us} < 0$
   - $\alpha_f < \alpha_r$
   - Angle difference required: to keep trajectory, the steering angle should be decreased

9.6.2 Steady-State Yaw Rate Response

Considering steady-state vehicle motion:

$$r = \frac{u}{R} \tag{9.55}$$

$$a_y = \frac{u^2}{R} \tag{9.56}$$

We get:

$$\delta = \frac{r}{u}\left(L + K_{us}\frac{u^2}{g}\right) \tag{9.57}$$

Then, the yaw velocity response gain can be formulated as:

$$G_{yaw}^{ss} = \frac{r}{\delta}|_{ss} = \frac{u}{L + \frac{K_{us}u^2}{g}} \tag{9.58}$$

E.g., this is defined as the ratio of the steady state yaw velocity to the steer angle.

1. Neutral steer: linear variation with speed
2. Understeer: increases to a maximum and then drops
3. Oversteer: increases with speed

A car with oversteer has the most sensitive handling characteristics, while the understeer vehicle is the least responsive.

9.6.3 Characteristic Speed

The characteristic speed is the speed at which the steering input required to negotiate the turn is twice the angle needed at speeds approaching zero. It is only for an understeer vehicle.

$$V_{char} = \sqrt{\frac{gL}{K_{us}}} \tag{9.59}$$

It is the maximum steady-state steering sensitivity.

9.6.4 Critical Speed

The critical speed is the speed at which the steering input required to negotiate the turn is zero. It is only for an oversteer vehicle.

$$V_{crit} = \sqrt{\frac{gL}{-K_{us}}} \tag{9.60}$$

Beyond this, the vehicle is unstable.

9.7 Dynamic Bicycle Model

Assumptions and limitations:

1. Constant forward velocity
2. No longitudinal / lateral load transfer
3. Linear range tires
4. No roll. pitch, or vertical motion
5. Ideal steering dynamics
6. No suspension
7. No compliance effects

With front wheel steering, there is a front slip angle, causing a front lateral force, causing lateral acceleration. More lateral acceleration leads to more yaw velocity but less yaw moment.

For vehicle motion by fixed coordinates on the vehicle, the velocity vector is:

$$\dot{R} = u\hat{i} + v\hat{j} \tag{9.61}$$

The acceleration vector is:

$$\ddot{R} = \dot{u}\hat{i} + u\dot{\hat{i}} + \dot{v}\hat{j} + v\dot{\hat{j}} \tag{9.62}$$

The changes in the unit vectors over time:

$$\Delta\hat{i} = r\Delta t\hat{j} \tag{9.63}$$

$$\Delta\hat{j} = -r\Delta t\hat{i} \tag{9.64}$$

Thus:

$$\dot{\hat{i}} = r\hat{j} \tag{9.65}$$

$$\dot{\hat{j}} = -r\hat{i} \tag{9.66}$$

From that:

$$\ddot{R} = (\dot{u} - vr)\hat{i} + (\dot{v} + ur)\hat{j} \tag{9.67}$$

With Newton-Euler equations for the vehicle motion:

$$ma_x = 0 \tag{9.68}$$

$$ma_y = m(\dot{v} + ur) = \sum F_y \tag{9.69}$$

$$I_z\dot{r} = \sum M_z \tag{9.70}$$

Assuming a small steering angle δ:

$$\sum F_y \approx F_{yf} + F_{yr} \tag{9.71}$$

$$\sum M_z = l_f F_{yf} - l_r F_{yr} \tag{9.72}$$

Then:

$$u = \text{ constant} \tag{9.73}$$
$$m(\dot{v} + ur) = F_{yf} + F_{yr} \tag{9.74}$$
$$I_z\dot{r} = l_f F_{yf} - l_r F_{yr} \tag{9.75}$$

where:

$$F_{yf} = C_{\alpha f}\alpha_f \tag{9.76}$$
$$F_{yr} = C_{\alpha r}\alpha_r \tag{9.77}$$

the front slip angle turns the vehicle out of the turn:

$$\alpha_f = \delta - \frac{v + l_f r}{u} = -\beta + \delta - \frac{l_f r}{u} \tag{9.78}$$

The rear slip angle turns the vehicle into the turn:

$$\alpha_r = -\frac{v - l_r r}{u} = -\beta + \frac{l_r r}{u} \tag{9.79}$$

Considering the derived equations of wheel slip angle and lateral forces, and putting them together:

$$\dot{v} = -\left(\frac{C_{\alpha f} + C_{\alpha r}}{mu}\right)v + \left(-u + \frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{mu}\right)r + \frac{C_{\alpha f}}{m}\delta \tag{9.80}$$
$$\dot{r} = \left(\frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{I_z u}\right)v + \left(-\frac{l_f^2 C_{\alpha f} + l_r^2 C_{\alpha r}}{mu}\right)r + \left(\frac{l_f C_{\alpha f}}{I_z}\right)\delta \tag{9.81}$$

Due to no constraints in the linear tire model, acceleration is linearly proportional to steering input resulting in unrealistic values.

L10 Automated Vehicle Control

10.1 Motion Analysis

Rewriting the final equations from the last lecture in matrix form:

$$\dot{x} = Ax + Bu \tag{10.1}$$

the steady-st
model

We also have the output vector containing lateral acceleration, yaw rate and sideslip angle:

$$y = Cx + Du \tag{10.2}$$

The eigenvalues show:

1. Neutral and oversteer vehicles yield to real eigenvalues
2. Oversteer vehicles have a positive real eigenvalue once vehicle speed is larger than critical, therefore becomes unstable
3. Understeer vehicles have complex conjugate eigenvalues

For understeer, the damping ratio decreases with the increase n forward velocity. For neural steer, it is overdamped and stable. For Oversteer, it is overdamped and unstable beyond the critical speed.

The steady-state yaw amplification factor increases at high speeds. At higher frequency, yaw response is very damped. Near the yaw natural frequency, overshoot is observed indicating reduced yaw damping and causing vehicle oscillations in a rapid steering.

The yaw rate response becomes high for an oversteered vehicle.

With a step input:

1. Understeer: response reaches to steady-state (stable)
2. Oversteer: response exceeds steady-state and diverges (unstable)

10.2 Control

10.2.1 Path-Following

General architecture: tracking of the reference path. It is usually expressed in terms of reference steering angle and long acceleration. We will only consider lateral path-following control.

y_e is the distance of the vehicle CoG from the lane centerline.

ψ_e is the orientation error of the vehicle with respect to the road.

$$y_e = y_{des} - y \tag{10.3}$$

$$\psi_e = \psi_{des} - \psi \tag{10.4}$$

10.2.2 Stanley Algorithm

The Stanley algorithm is a geometry-based path tracking method suitable for medium-high speed driving conditions. Steering control is defined as:

$$\delta = \psi_e + \arctan \frac{ky_e}{u} \tag{10.5}$$

Considering lateral acceleration:

$$\ddot{y}_e = \ddot{y}_{des} - a_y \tag{10.6}$$

Knowing that $\ddot{y}_{des} = u^2/R$ and $a_y = (\ddot{y} + u\dot{\psi})$:

$$\ddot{y}_e = \frac{u^2}{R} - (\ddot{y} + u\dot{\psi}) \tag{10.7}$$

Consider that $u = \dot{\psi}R$. Then we can define the acceleration error as:

$$\ddot{y}_e = u\dot{\psi}_{des}.\ddot{y} - u\dot{\psi} = u(\dot{\psi}_{des} - \dot{\psi}) - \ddot{y} \tag{10.8}$$

By integration, the rate of lateral offset can be defined as:

$$\dot{y}_e = -\dot{y} + u(\psi_{des} - \psi), \text{ if } u \text{ is constant} \tag{10.9}$$

By differentiation, the rate of the heading angle offset can be defined as:

$$\dot{\psi}_e = \dot{\psi}_{des} - \dot{\psi} = \frac{u}{R} - \dot{\psi} \tag{10.10}$$

This can be put into state-space representation. Instead of the vehicle state, we can use the lateral offset and heading angle error, and their rates.

10.3 Vehicle-Road Model

An open loop model has four polse:

1. Two due to vehicle dynamics
2. Two at the origin due to the road error mode, making the system marginally stable

It is a linear parameter-varying system as the vehicle dynamics related poles change with changing longitudinal velocity. Matrices A and B_1 depend on vehicle mass and inertia, and axle cornering stiffness. In practice these parameters change with payload, road, and environmental conditions, tire wear. etc.

10.4 Industrial SOA Control: Path vs Single Preview

The problem can be tackled by providing appropriate feedback and feedforward terms:

$$\delta = \delta_{fb} + \delta_{ff} \tag{10.11}$$

The following control law is applied:

$$\delta = \begin{bmatrix} k_{y_e} & k_{\dot{y}_e} & k_{\psi_e} & k_{\dot{\psi}_e} \end{bmatrix} x + d_{ff} \tag{10.12}$$

The usage of gains related to offset/error rates leads to poor performance due to high noise and needs to be estimated, and are therefore set to 0.

The feedforward steering angle to make the lateral error equal to zero and determining the steady-state solution:

$$\delta_{ff} = \delta_{ssc} - k_{\psi_e}\beta_{ssc} \tag{10.13}$$

The steady state body sideslip is often ignored due to no robust way to accurately measure or estimate in normal driving, and inclusion of the sideslip angle in the feedback control law leads to significantly reduced stability margins.

Feedforward contribution is calculated based on the reference yaw rate or turn radius R from curvature estimation using cameras:

$$\delta_{ff} = \delta_{ssc} = \left(\frac{L}{u} + K_{us}u\right)\dot{\psi}_{ref} = \frac{L + K_{us}u^2}{R} \tag{10.14}$$

10.4.1 Path Control vs Single Preview Point

If the lanes can be well detected by an on-board camera, information about the path to follow is available, e.g. lateral and heading errors of the vehicle with respect to the center of the lane. For car following, longitudinal vehicle following at a small headway distance is desired in favor of a reduction in air drag. As a result, the preceding vehicle blocks the line of sight of the camera and usable lane information is unavailable. In this case, by a fusion of camera and radar measurements, the longitudinal and lateral distance to the preceding vehicle can be accurately obtained, i.e. a single point information is available instead of path information. For this reason, two feedback control laws are applied. Switching between these is done depending on the available measurements.

10.4.2 Path Controller

The goal is to minimize y_e and ψ_e. The control law is defined as:

$$\delta = k_{y_e} y_e + k_{\psi_e} \psi_e + \delta_{ff} \tag{10.15}$$

Combined feedback error y_e can be defined at a virtual look-ahead point, located at a distance x_{la} in front of the vehicle CoG.

$$k_{y_e}(y_e + x_{la}\psi_e) + \delta_{ff} \tag{10.16}$$

10.4.3 Single Point Preview Controller

If only one measurement is available of the path, the goal is to minimize the lateral error in the look-ahead distance y_m.

$$\delta = k_{y_m} y_m \tag{10.17}$$

The control law is defined as:

$$\delta = k_{y_m} y_m = k_{y_m}(y_e + x_{la}\psi_e + y_{road}) \tag{10.18}$$

No feedforward term is applied.

10.4.4 Comparison

Path control:

+ No cutting of the corners in ideal world
+ Two gains that can be tuned independently
- Path information required to obtain the road curvature and measurements of the lateral position and heading angle error

Single point preview control:

+ Directly use sensors
- Errors for curvature transients