ECE 493 Lecture Notes 2

# 8-1 State Estimation: Probabilistic Models

## State-Space System – Probabilistic Representation

* Recall the non-linear model in functional form:
* The equivalent probabilistic model is:  
  + Represents output of each functional equation as a conditional probability function
  + Don’t need to specify disturbances, since uncertainty is modeled by distribution

## Types of Probabilistic Models

### Markov Chain

* No control inputs (autonomous system)
* No hidden states, so no measurement model

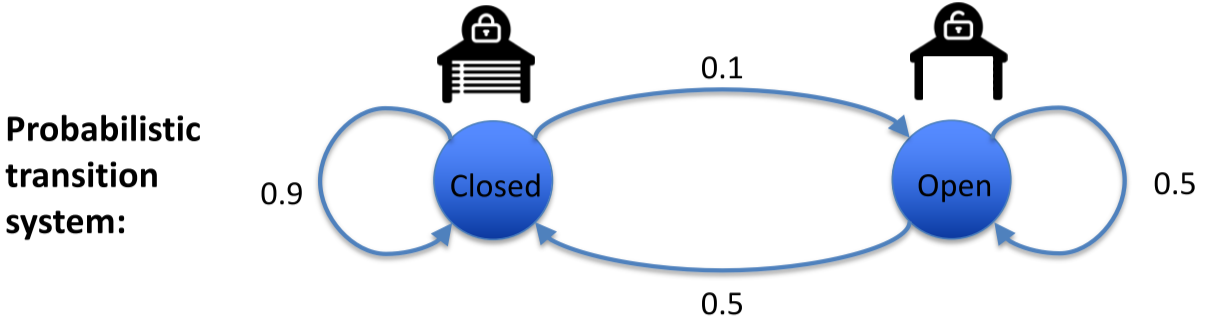
### Hidden Markov Chain

* No control inputs (autonomous system)
* Has hidden states – uses observation model

### Dynamic Bayesian System

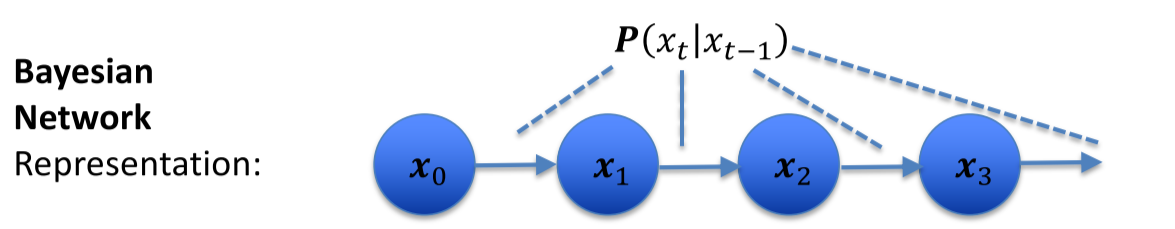
* Probabilistic state transition model with control inputs
* Has hidden states – uses observation model

## Discrete Markov Chain – Garage Door Example



Probabilistic distribution represented as a vector:

**Left stochastic matrix** (each column adds to 1):



Computing the state probability distribution in each step, assuming the door is closed at :

## Steady-State of System Modelled by Markov Chain

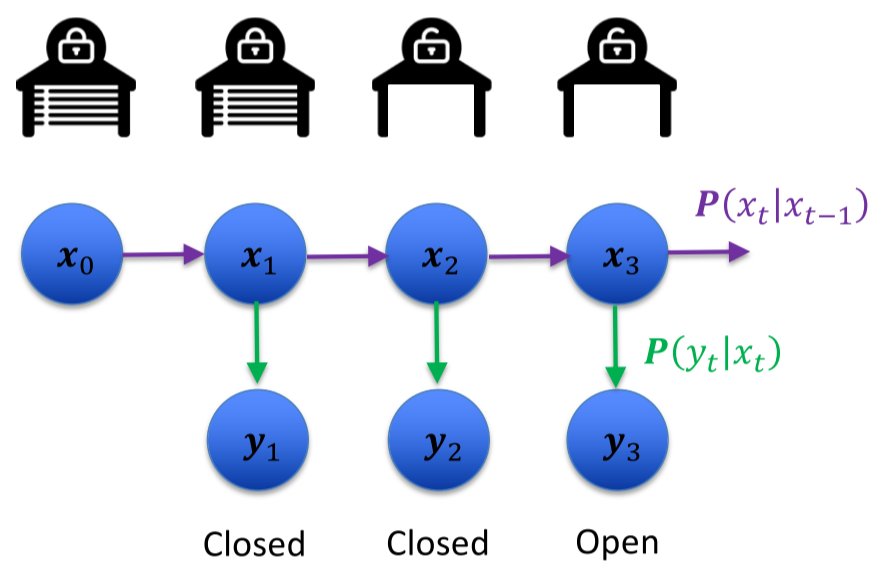
* After a long time, what is the probability distribution of the state?
* Since is the steady state, it won’t change when multiplied by :
  + Recall that the eigenvector and eigenvalue of of a matrix are defined by:
  + Thus, is the eigenvector (eigenvalue = 1) of

Going back to the garage door example, need to solve equation:

Hence,

## Hidden Markov Model – Garage Door Example

We now add a noisy sensor:

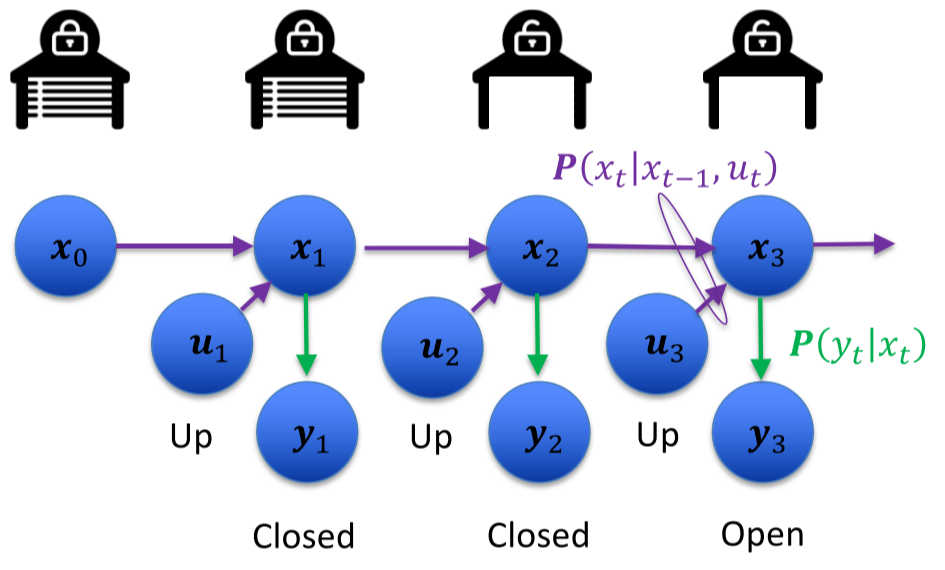


## Dynamic Bayesian Network

* Add control inputs
  + They are noisy, so they affect the probability of state transitions, but do not make them deterministic
* To model the influence of control inputs on state transitions, need separate state transition matrix for each input value

### Garage Door Example

We add a noisy remote controller:



# 8-2 State Estimation: Bayes Filter

## Bayes Filter

* Foundation for all other filters
* **Bayes Rule**: right way to incorporate new information into existing estimate
* Resulting filter definition can be implemented directly for discrete state systems
* For continuous state systems, need additional structures and assumptions to solve the update equations analytically

Inputs:

* – state of vehicle and environment that can impact the future, assumed to be **complete**
* – control inputs – all elements of vehicle/environment that can be controlled
* – measurements – all elements of vehicle/environment that can be sensed

Notation:

* – discrete time index
* – initial state
* First, apply control action
* Move to state
* Take measurement

### State Update Modelling

At each time is a sufficient summary of all previous inputs and measurements

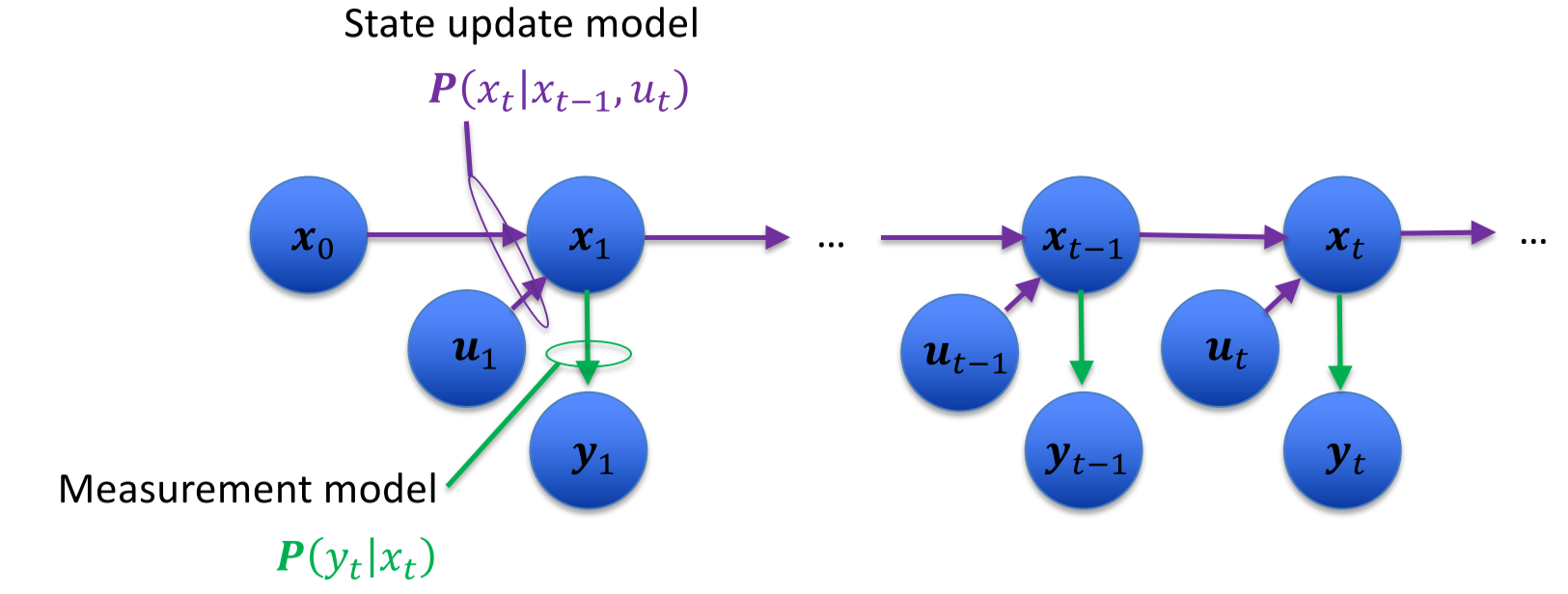
* Apply **Markov Assumption**: no additional information needs to be dependent on previous inputs/measurements (conditional independence)

### Measurement Modelling

Current state is sufficient to model all previous states, measurements, and inputs

* Again, assume Markov Assumption (conditional independence)
* In standard LTI state space, measurement may also depend on current input

## Combined Model (Dynamic Bayes Network)



### State Update Model

For discrete states, where can take on one of values , the state update model can be written as an matrix

For each input value , the state update model is:

* Each row defines the probability of transitioning to state from all possible states
* Each column defines the probability of transitioning to all possible states from a specific state
* All columns sum to 1

### Measurement Update Model

For discrete states, where can take on one of values , and can take on one of values , the state update model can be written as an matrix

* Each row defines the probability of for all possible states
* Each column defines the probability of all possible values of for a specific state
* All columns sum to 1

## Bayes Filter

* Aim: estimate current state based on all known inputs and measurements, by defining a **belief** about the current state using all available information:
  + Depends on every piece of information available up until time
* Can also define a **prediction**, a belief prior to measurement :

### Problem Statement

* Given:
  + A prior system state
  + State model and measurement model
  + Sequence of inputs and measurements
* Form a belief about/estimate the current state

### Bayes Filter Algorithm

At each time step , for all possible values of state :

1. Prediction update (total probability)
2. Measurement update (Bayes Theorem)  
   * is a normalizing constant that doesn’t depend on the state

## Garage Door Example

Problem: detect if door is open/closed with robot that can sense door position and request door open

State:

Initial state:

**State model**

Inputs:

* If , do nothing:
* If , door opens with high probability:

**Measurement model (noisy door sensor)**

Measurements:

Example: at time step 1, input = none

1. Perform state prediction update – calculate belief prediction for each possible state:
2. Perform measurement update:

At time step 1,

Thus:

Example: at time step 2, input = up, measurement

1. Perform state prediction update:
2. Perform measurement update:

# 8-3 State Estimation: Occupancy Grid Mapping

## Mapping

* Process of determining the environment relative to a known position
* Given:
  + Vehicle location model
  + Sensor measurements and inverse measurement model
* Find: environment map

### Types of Maps

* **Location-based**
  + Map is defined by occupancy of each location
  + Works well in 2D, but scales poorly
* **Feature-based**
  + A feature is defined at a specific location, and may have a signature
  + Map is defined by set of all features
  + Scales well to large dimensions, but hard to use for collision detection

## Occupancy Grid Mapping

* Find the probability at time that each grid cell contains an obstacle
* Assumptions:
  + Static environment
  + Cells are independent
  + Vehicle state is known at each time step
  + Sensor model is known

### Bayes Filter with Static States

* Since cell contents don’t move, the motion model is trivial
  + Each predicted belief is the belief from the previous time step:
  + Since prediction step is no longer needed, update with each new measurement regardless of vehicle motion
* **Log odds ratio**: instead of tracking probability, track log odds ratio for each cell  
  + Provides numerical stability for low and high values of probability
  + Allows update rule to only involve addition
  + To recover the probability:
* The Bayesian log odds update rule is:  
  + Inverse measurement ratio + ratio at , subtract initial ratio
* To get the inverse measurement ratio, need **inverse measurement mode**l:
  + Probability of a state given a certain measurement:
  + Inverse conditional probability:

## Laser Scanner Example

* Returns range to the closest objects at a set of bearings relative to the vehicle heading
* For more accurate results, use **Bresenham’s line algorithm**
  + Instead of updating each cell once per complete scan, perform one update per range measurement

### Computational Issues

* Calculation grows as grid size grows
* Measurement model pre-caching – entire model can be pre-calculated, since state does not change
* Sensor subsampling – may be significant overlap in scans
* Selective updating – only update cells when significant new information is available

# 9-1 State Estimation: Kalman Filtering

## Kalman Filter Model Assumptions

* Continuous state, inputs, and measurements
* The state prior is Gaussian:
* Motion model is **linear**, with **additive Gaussian** process/motion noise:  
  + Robotic systems are often more easily described in the continuous domain
* Measurement model is linear, with additive Gaussian measurement noise:
  + Can also add control input dependence:

## Kalman Filter

* Assume that the belief is Gaussian at time :
  + – best estimate of current state (at time )
  + – covariance (certainty in the current estimate)
* Goal: minimize the mean square error of the estimate

### Problem Formulation

Related to Bayes filter:

* State prior:
* Motion model:
* Measurement model:
* Beliefs:

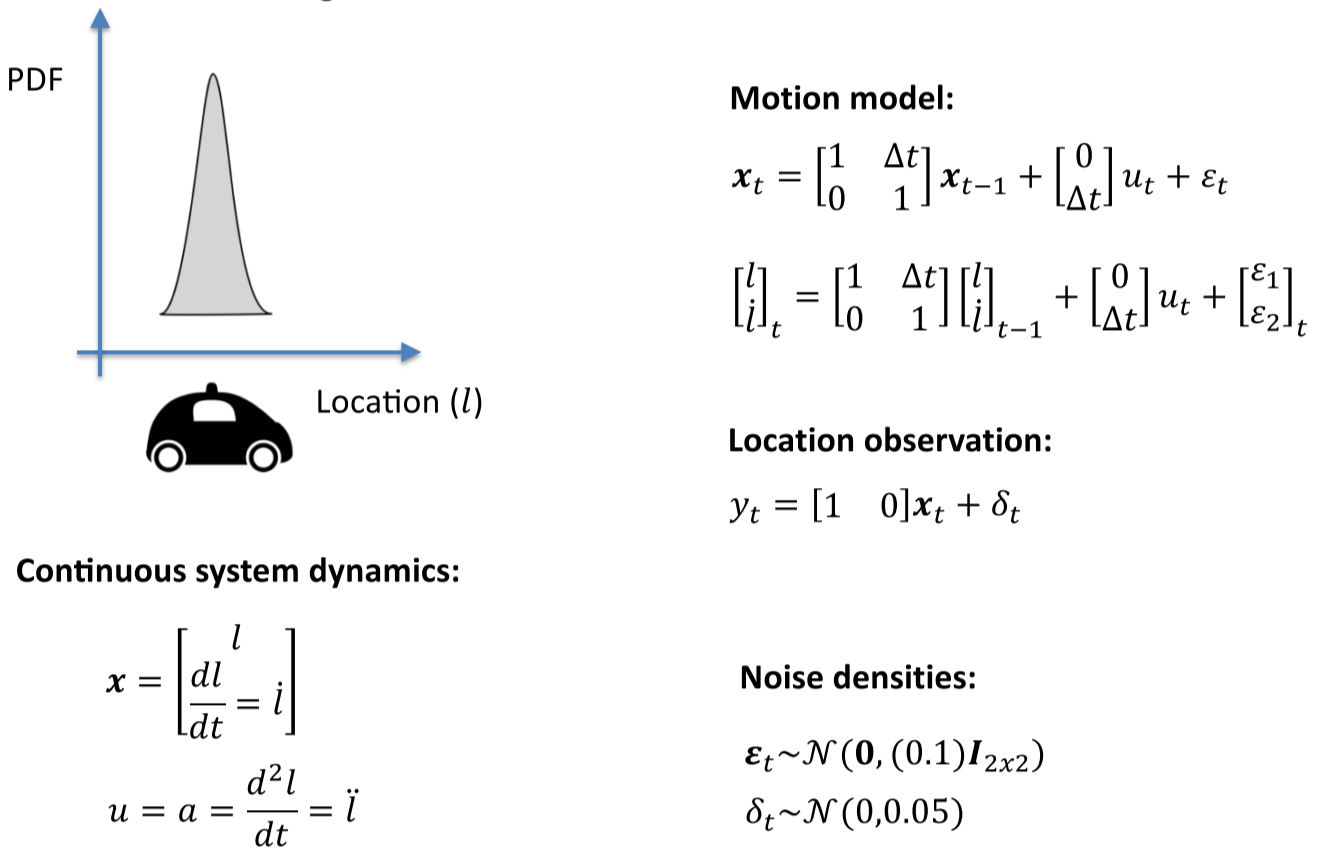
### Kalman Filter Algorithm

1. Prediction update:
2. Measurement update:

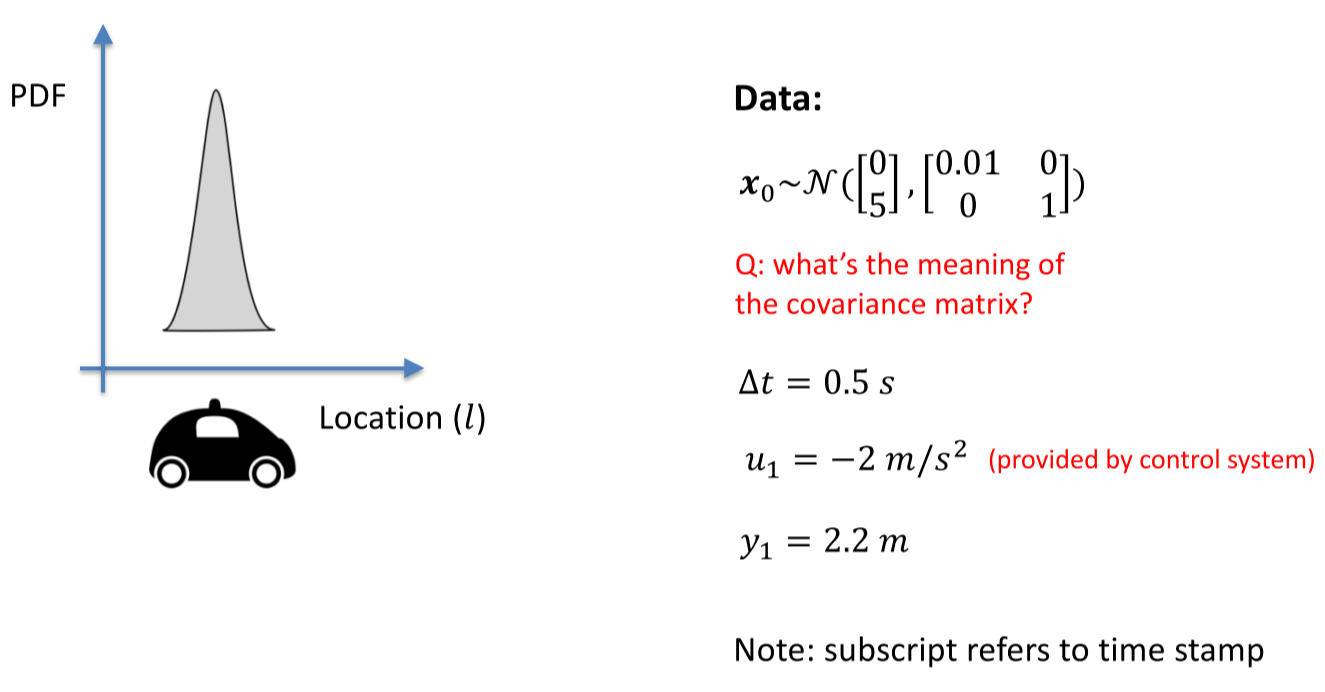
* – Kalman Gain – blending factor between prediction and measurement innovation
  + **Innovation**: the difference between the measurement and the expected measurement, given the predicted state and measurement model
* – prediction uncertainty
* – measurement matrix
* – measurement noise
* – measurement innovation
* When is low, is the inverse of , so the filter pays attention to measurement innovation
* When is low, is low, so the filter pays attention to the prediction
* If is large, the Kalman gain (inverse) is small – when measurement covariance is high, don’t trust it
* If is large, so is the predicted belief’s covariance, so the Kalman gain is high – when model is affected by unknown disturbances, don’t trust the prediction

## Vehicle Localization Example











1. Prediction update:
2. Measurement update:

## Multi-Rate Kalman Filter

* At each time step, it’s possible for and to vary
* Identify a base update rate, then create a discretized update motion model at base rate
* At each time step:
  1. Perform prediction update
  2. If new measurements exist, perform measurement update for those measurements only, selecting appropriate and

## Steady-State Kalman Filter

* For constant noise models, it is possible to use steady-state values for Kalman gain
  + **Discrete Algebraic Riccati Equation (DARE)**: set in the Kalman filter update equations, solve for
* Can also run Kalman filter until convergence, then eliminate gain update step

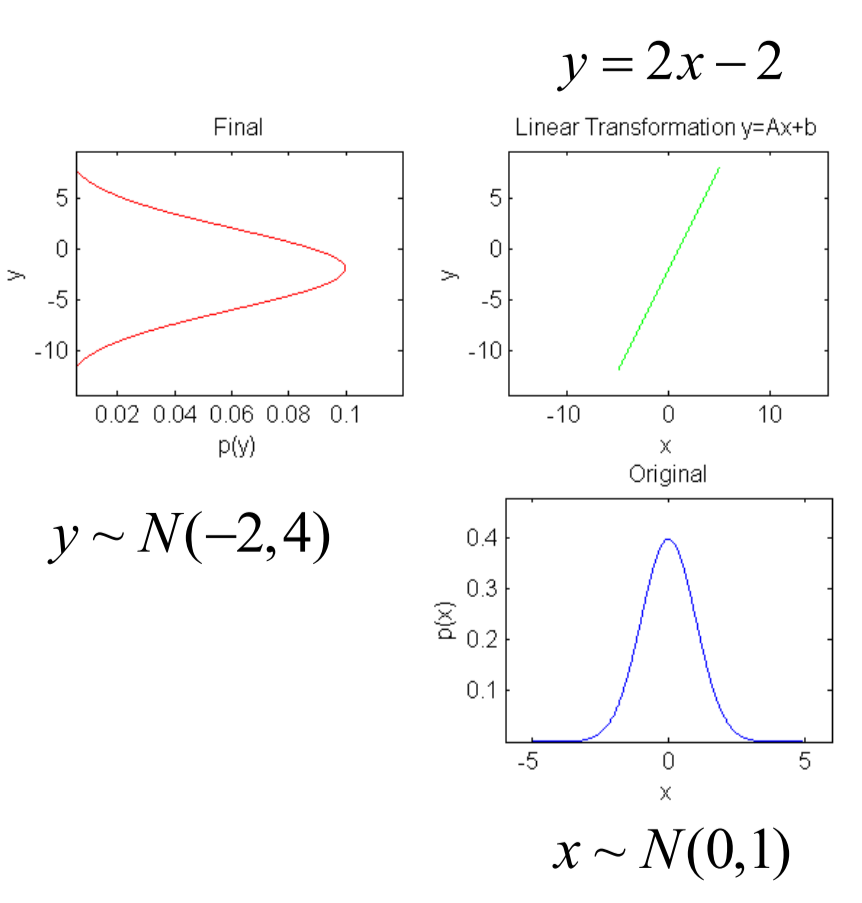
# 9-2 State Estimation: Extended Kalman Filter

## Kalman Filter vs. Non-Linear Models

* Kalman Filter requires **linear** motion and measurement models
  + Results in compact, recursive estimation technique
  + However, not realistic for most applications
* With non-linear models, belief distributions aren’t guaranteed to be Gaussian
  + No longer able to track mean and covariance
  + There isn’t a closed-form solution to the Bayes filter algorithm, for non-linear models

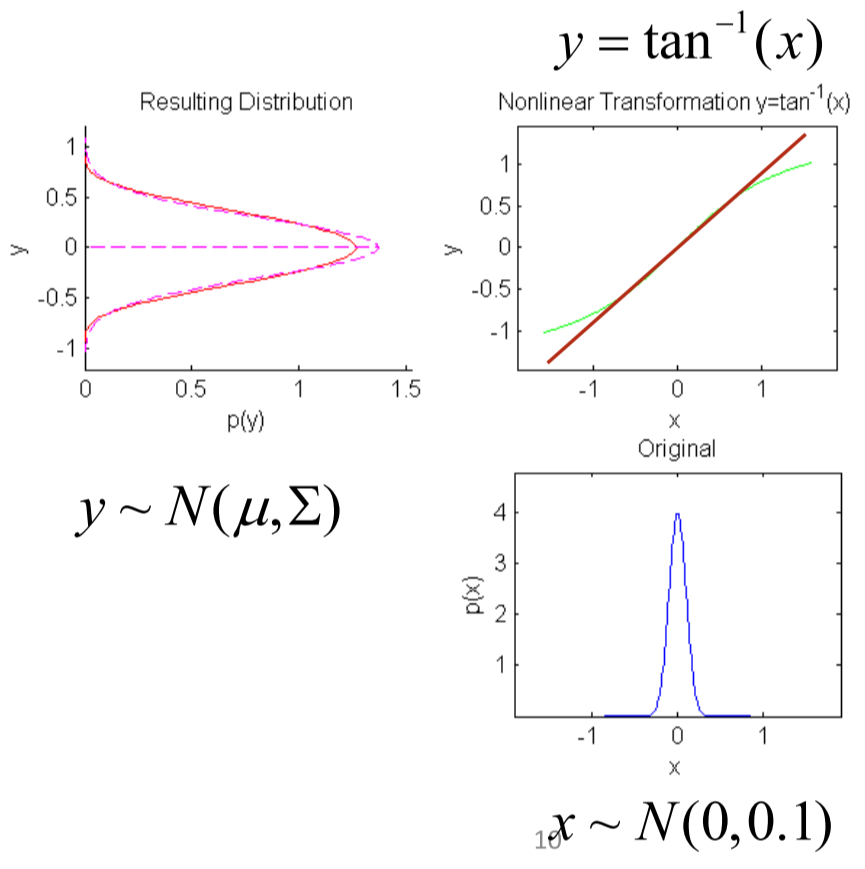
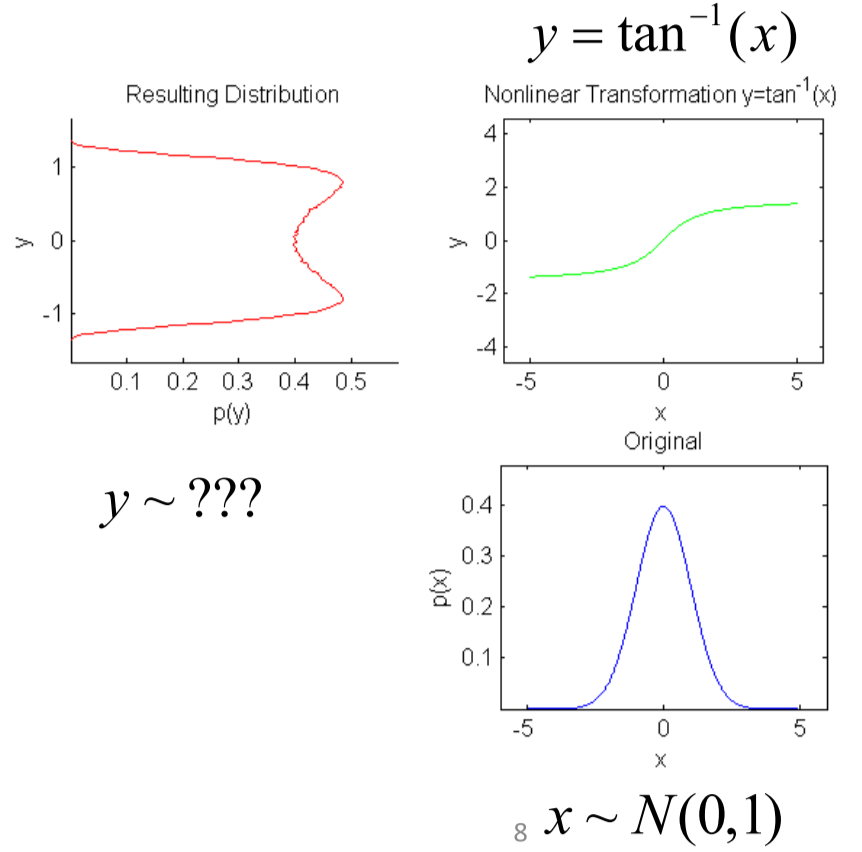
## Linear Transformations on Gaussian

Result in another Gaussian!



### Effect of Non-Linearity on Gaussian Distribution

* Take 5,000,000 samples of original Gaussian
* Apply nonlinear transformation to each sample
* Create histogram with 100 bins and normalize counts
* Calculate the mean and covariance of the 5,000,000 samples, to get best Gaussian fit



## Extended Kalman Filter (EKF)

Uses general non-linear discrete-time state space model:

Key idea of maintaining Gaussian distribution with EKF:

1. Mean can be propagated through non-linear model
2. Covariance can be updated with a locally linear approximation to the model

## Linearizing a Non-Linear System

* Choose an operating point , and approximate the non-linear function at point by the tangent line
* For the EKF, choose the operating point as the most recent state estimate, known input, and zero noise:

**Linearized motion model**:



**Linearized measurement model**:



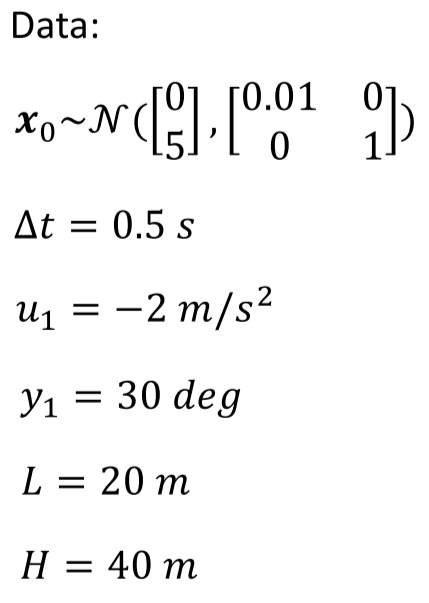
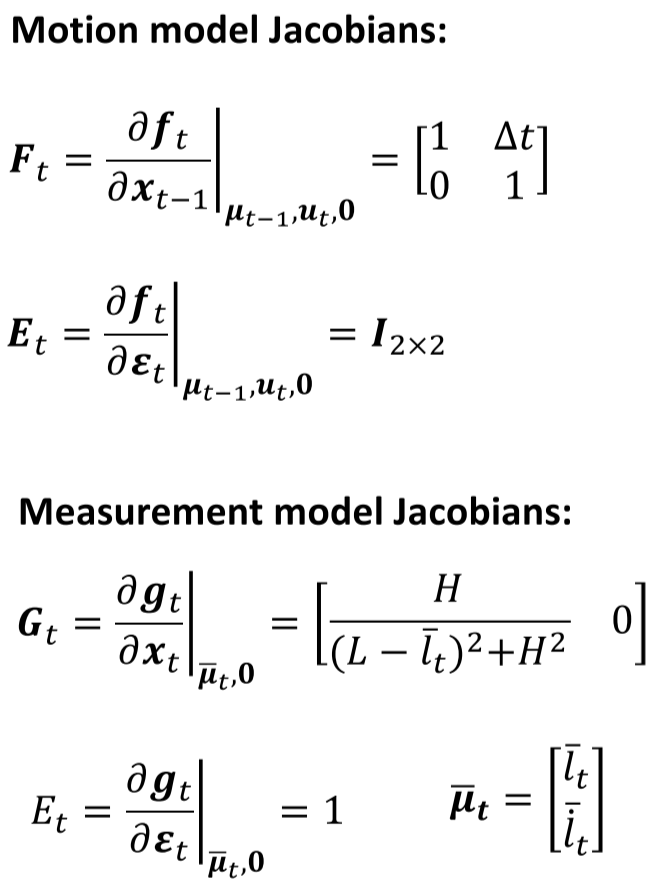
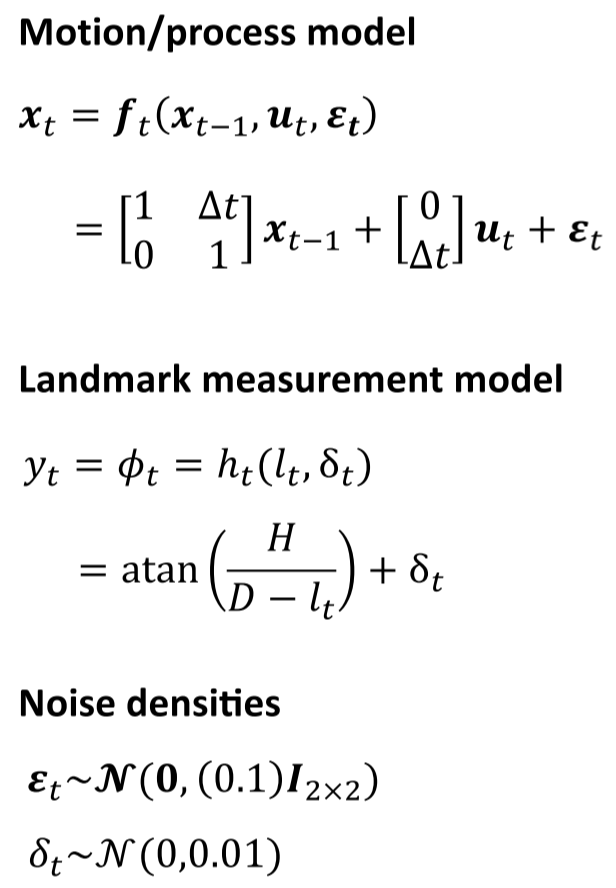
* The matrices are the **Jacobian matrices** of the system
  + To compute a Jacobian matrix, take all first-order partial derivatives of the function
  + Intuitively, the Jacobian matrix tells you how fast each output of the function is changing along the input dimension
* Hence, the update equations become:

**Prediction**:

**Optimal gain**:

**Correction**:

## EKF Example





**Prediction step:**

**Correction step**:

# 10-1 Vehicle Control: Vehicle Modelling – Kinematics

## Kinematic vs. Dynamic Models

* At low speeds, it is often sufficient to look at only kinematic models of a vehicle
* Dynamic models capture vehicle behaviour more precisely over a larger range of operation, but they are more involved
* Kinematic models disregard inertia and mass, take linear and angular velocities as inputs
* Dynamic models take inertia and mass into consideration, and take forces and torques as inputs

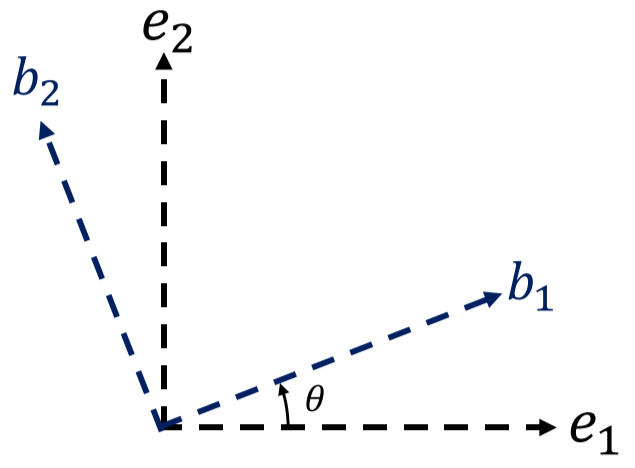
## Coordinate Frames

* By convention, right-handed
* Fixed **inertial frame**, usually relative to Earth
* **Body frame** attached to vehicle, origin is at vehicle’s center of gravity or center of rotation
* **Sensor frame** attached to sensor, convenient for expressing sensor measurements

## Coordinate Transformation

* Conversion between inertial frame and body frame coordinates is done with a translation vector and rotation matrix
* Every translation has two interpretations – translation of object, or translation of frame

### Rotation Matrices in 2D

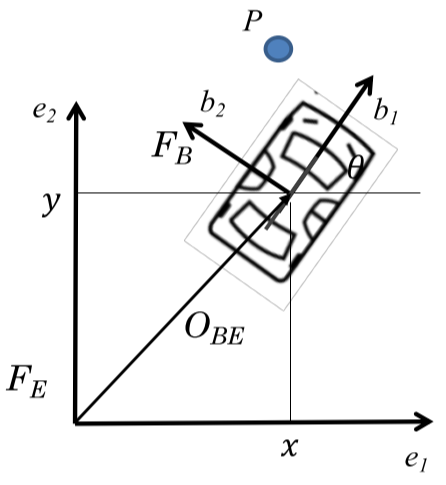


**Clockwise Rotation Matrix (E to B):**

* CW rotation matrix, but corresponding coordinate transformation is counter-clockwise

**Counter-Clockwise Rotation Matrix (B to E):**

* CCW rotation matrix, but corresponding coordinate transformation is clockwise



To find the location of point in body frame :

To find the location of point in inertial frame

* is the translation term – is it expressed in body frame, is it expressed in inertial frame

## Homogenous Coordinate Form

* Allows rotation, translation, scaling, and projection to be represented by one **matrix multiplication**
* Extra coordinate allows representing points at infinity by setting to 0
* A 2D vector in homogenous form:
* To transform a point from body to inertial coordinates with homogenous coordinates:

Examples:

**Translation:**

**Rotation (CCW):**

**Rotation followed by Translation (Matrix Multiplication):**

* Note that

### Properties of Rotation

* Rotation matrices are **orthogonal**:
* The determinant of every rotation matrix is 1

### Scaling

Combining with translation and rotation:

## Bicycle Kinematic Model

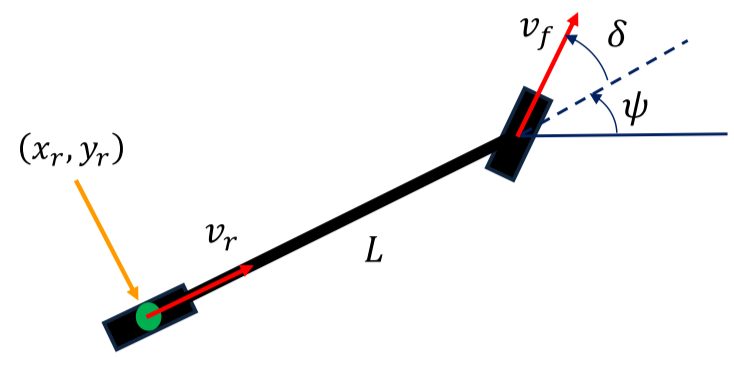
* Provides simple model for steering at low speeds (40-50 kph) and small lateral acceleration (max 0.5 g)
* **Yaw rate** is proportional to product of vehicle speed and steering angle
* For a fixed steering angle, vehicle travels in a circular path:

First, pick a reference point on the body:

* Front axle:
* Center of gravity:
* Rear axle:

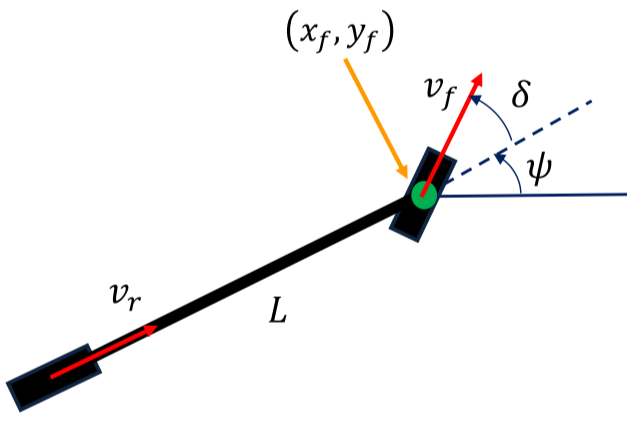
### Rear Axle Bicycle Model

* Direct and simplest formulation

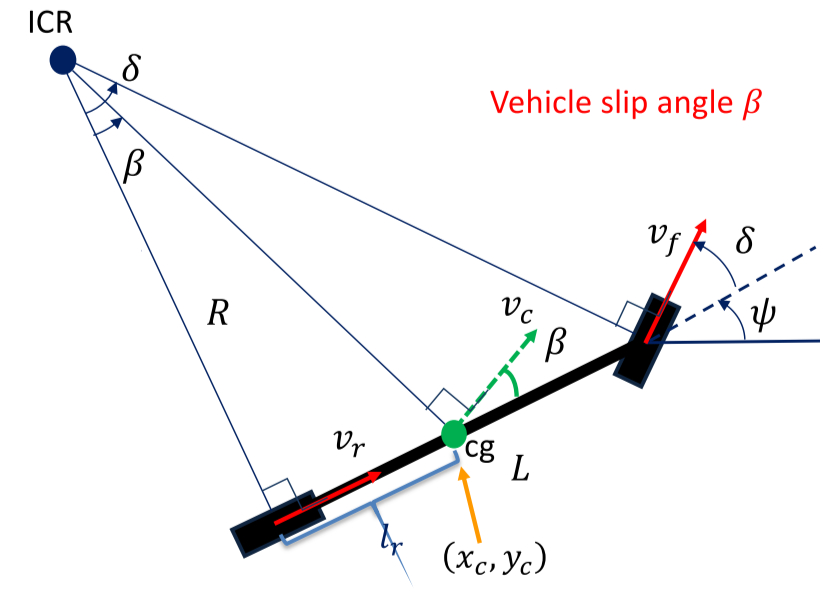


Modifying the model to use inputs steering rate and longitudinal acceleration :

### Front Axel Bicycle Model



### Center of Gravity Bicycle Model





* Existence of is needed for dynamic model, no significant role in kinematics

Modifying the model to use steering rate input :

# 10-2 Vehicle Control: Vehicle Modelling – Dynamics

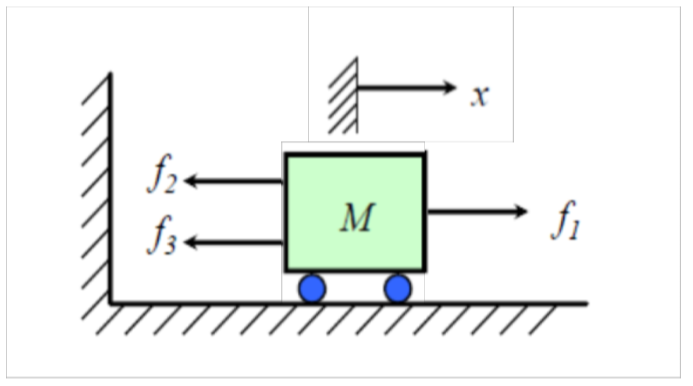
## Dynamic Modelling

* With higher vehicle speeds or slippery roads, vehicles don’t satisfy the no-slip condition
* Forces including drag and road friction govern the throttle inputs

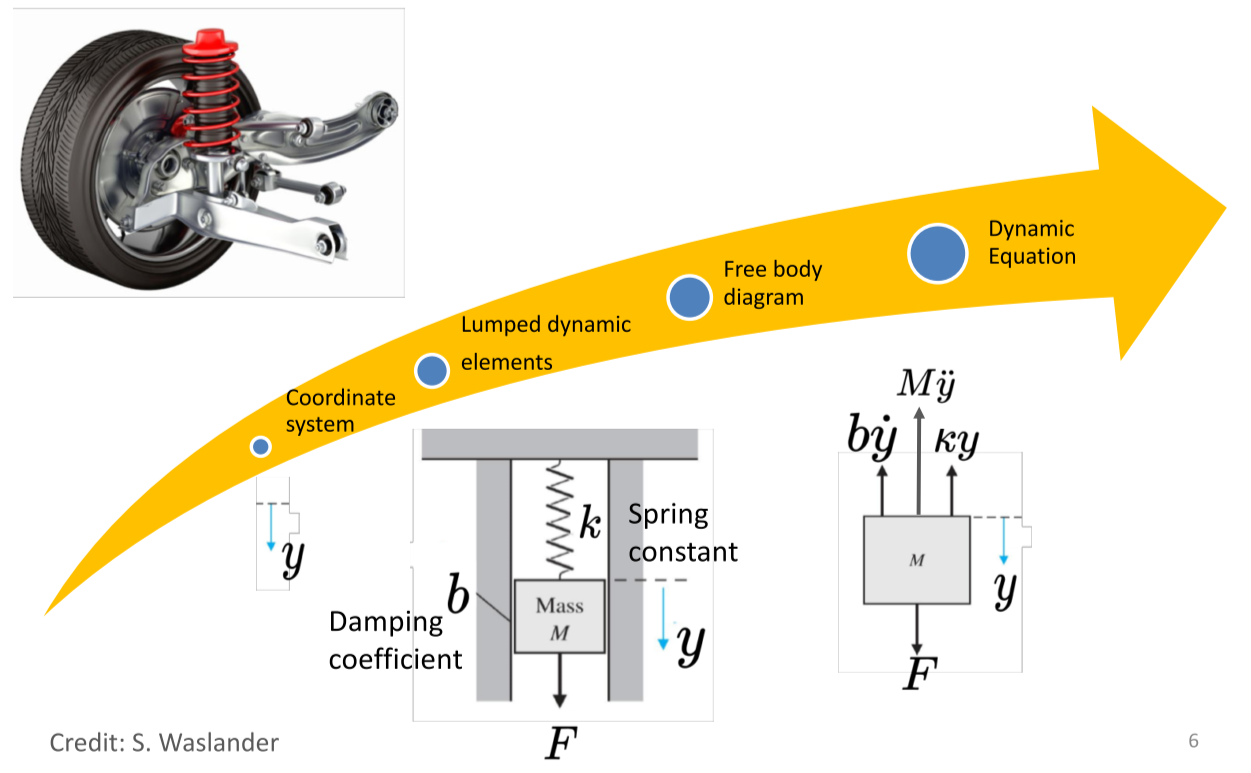
Steps to building a dynamic model:

Coordinate frames 🡪 Lumped dynamic elements 🡪 Free body diagram 🡪 Dynamic equations

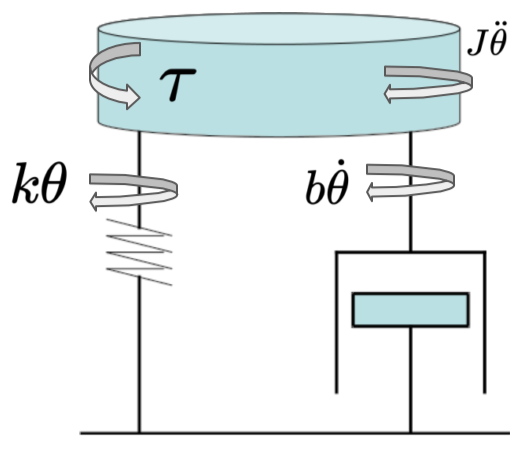
### Translational Systems

* Deals with forces and torques
* Roughly, equate all forces
* Governed by Newton’s second law – the acceleration of an object depends on the net force acting upon the object and the mass of the object  
  

Example: Vehicle shock absorber

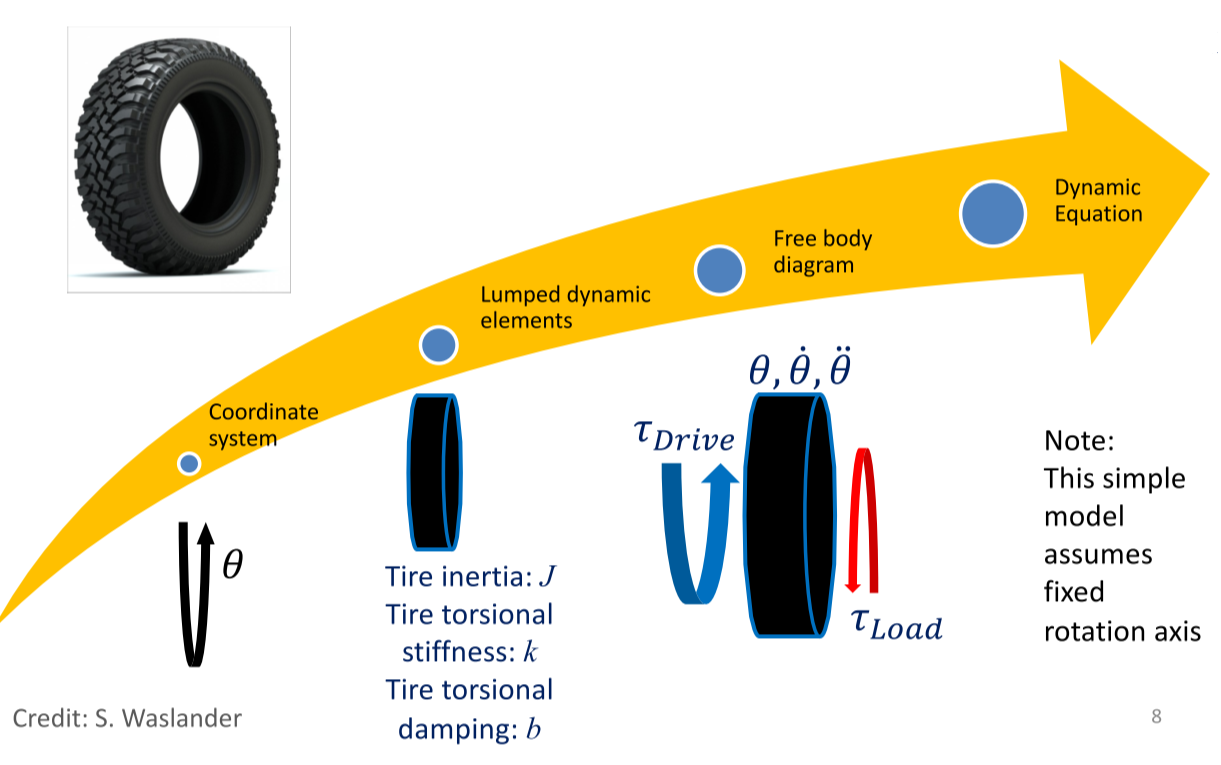


### Rotational Systems



* Inertial
* Torsional force
* Forces that resist the torsional force:
  + Spring force
  + Damping force
  + Inertia force

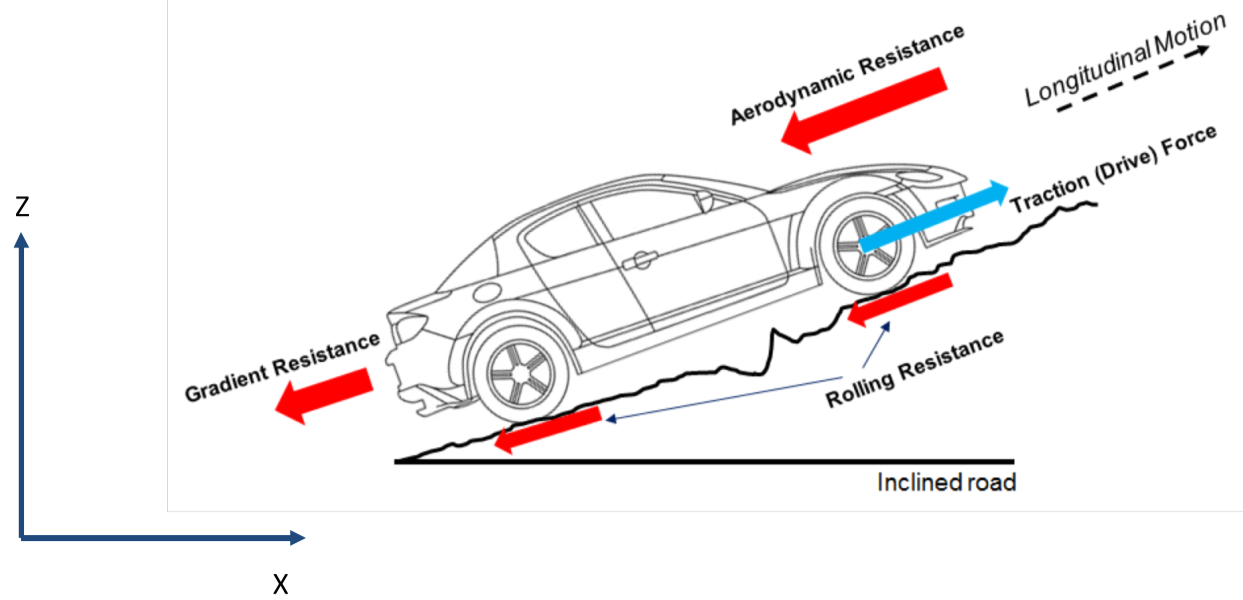
Example: Tire model



### Full Vehicle Modelling

* All components, forces, and moments are in 3D
  + Pitch, roll, normal forces
  + Suspension, drivetrain, component models

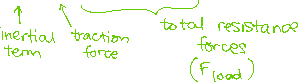
## 2D Dynamics – Vehicle Longitudinal Motion



### Simplified Longitudinal Dynamics

* is the **total longitudinal force** –
* is the **total rolling resistance** –
* Assume small angle:

Then:



### Simple Resistance Force Models

The total resistance load is:

* The aerodynamic force can depend on air density , frontal area , and vehicle speed :
* The rolling resistance can depend on the tire normal force, tire pressures, and vehicle speed :

## Powertrain Modelling

Rotational coupling:

* – wheel angular speed
* – turbine angular speed
* – engine angular speed
* – combined gear ratios

Longitudinal velocity:

* – effective radius of tire

Longitudinal acceleration:

* – engine angular acceleration

### Power Flow in Powertrain

Wheel 🡪 Transmission 🡪 Torque Converter 🡪 Engine

🡨 🡨 🡨

**Wheel**:

**Transmission**:

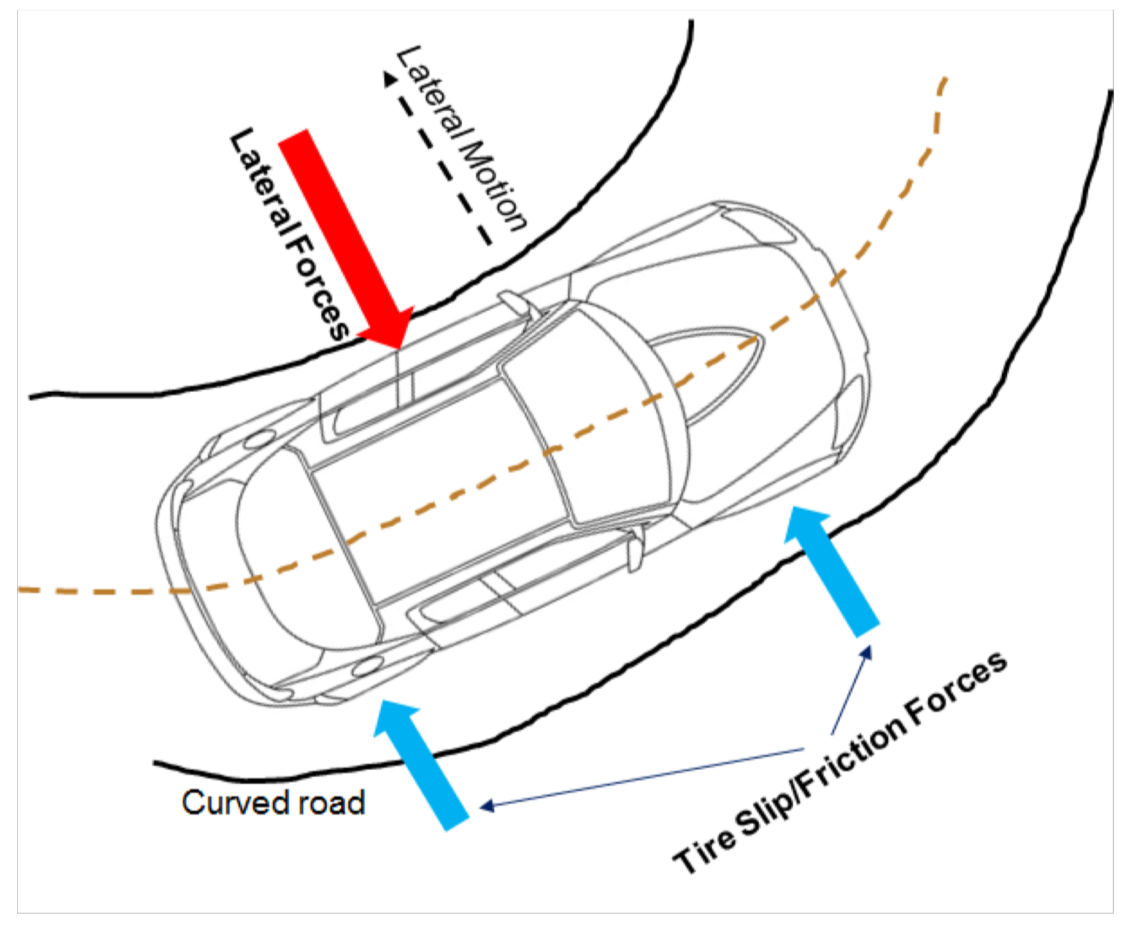
**Torque converter**:

**Engine**:

## Engine Dynamics

* Tire force in terms of inertia and load force:
* Combining with engine dynamics:
* Finally, the engine dynamic model simplifies to:  
  + – Total torque load

### 2D Dynamics – Vehicle Lateral Motion



## Simplified Lateral Dynamics: Bicycle Dynamic Model

* Assumptions:
  + Constant longitudinal velocity
  + Left and right axle act as single wheel
  + Neglect suspension movement, road inclination, and aerodynamic influence

### Lateral Dynamics

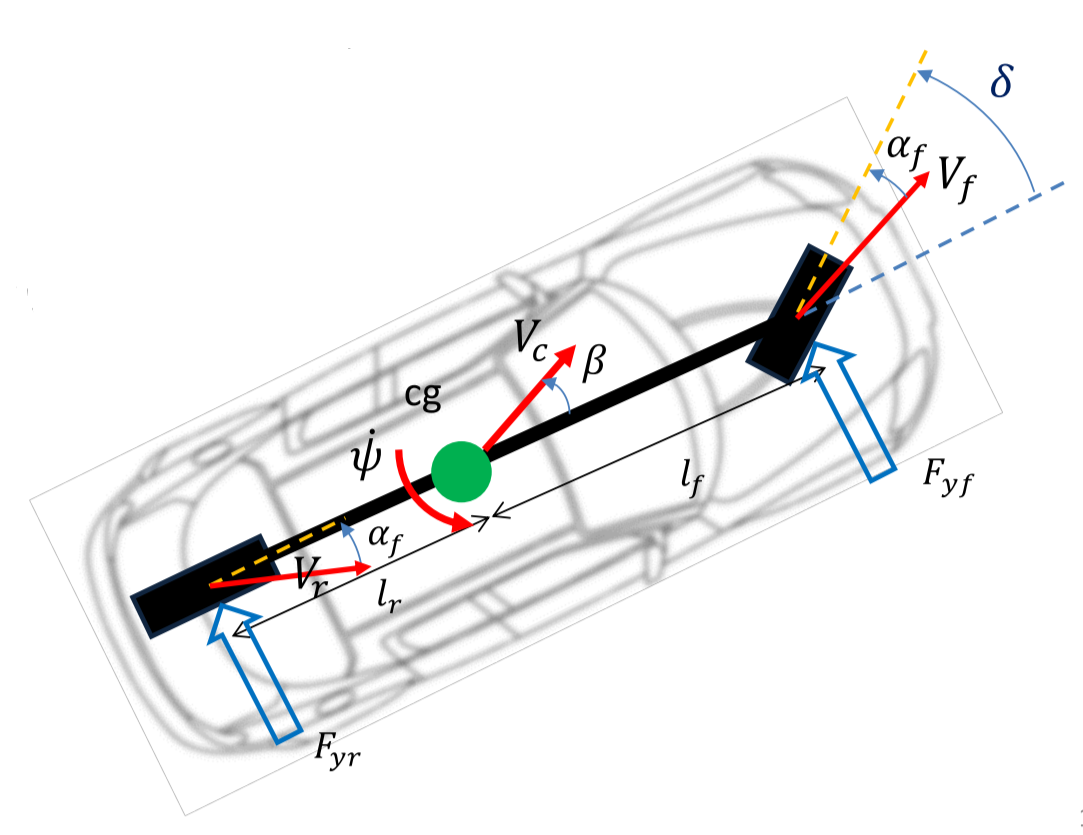


Lateral acceleration:

Lateral forces:

* – vehicle mass
* – longitudinal velocity
* – slide slip rate
* – yaw rate
* – front and rear tire forces
* – vehicle inertia
* – center of gravity distance from front and rear tires

### Tire Slip Angles



For small tire slip angles, lateral tire forces are approximated as a linear function of tire slip angle

* Front tire slip angle:
* Rear tire slip angle:

### Front and Rear Tire Forces

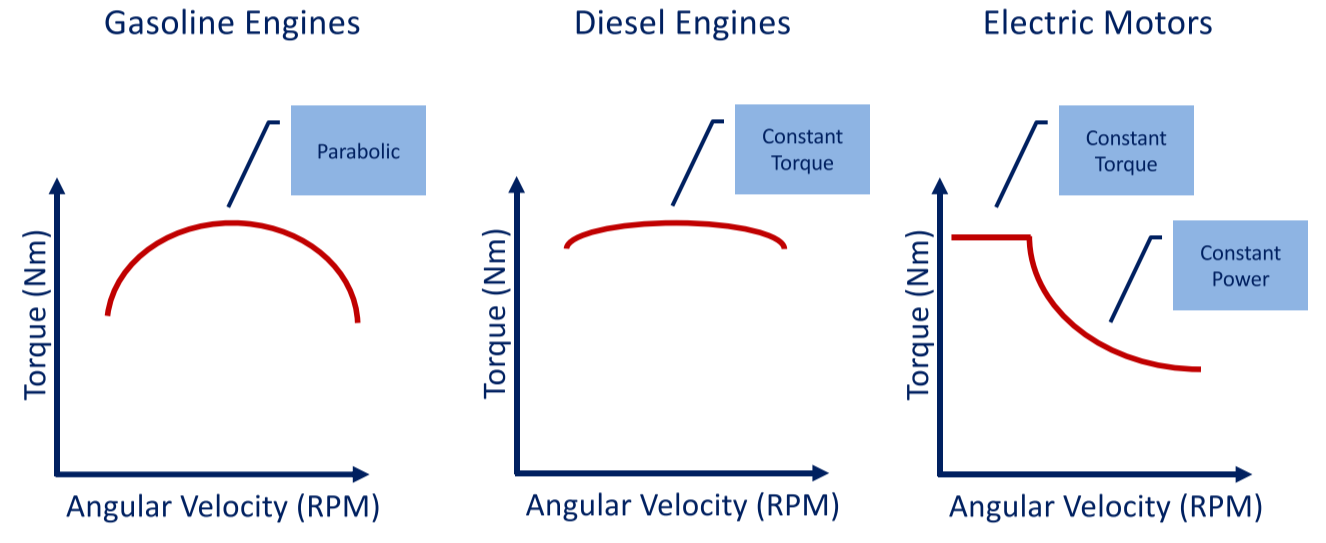
* – linearized cornering stiffness of front wheel
* – linearized cornering stiffness of rear wheel

## Lateral and Yaw Dynamics

Substituting the lateral forces and rearranging the equations:

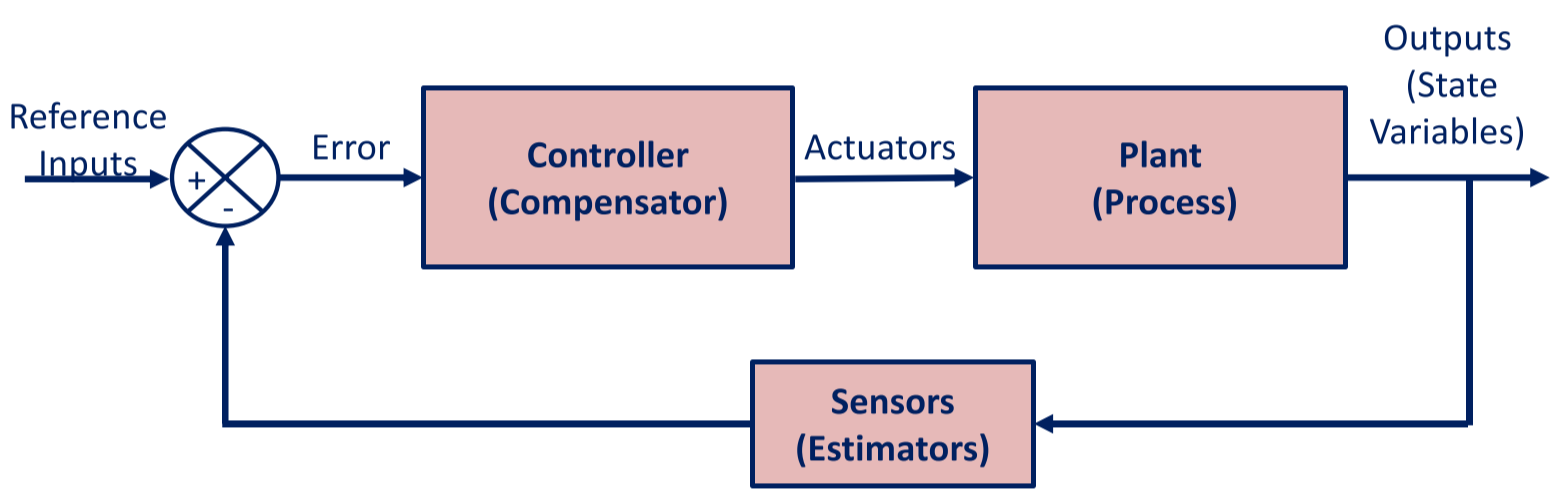
## Engine Characteristics Plots

Relates key engine performance features, mainly angular velocity and torque, as a function of throttle



# 11-1 Vehicle Control: Longitudinal Control

## Typical Feedback Control Loop



### Plant System/Process

* Either linear or non-linear system
* Represent by either state-space form or transfer functions
  + Linear time-invariant systems can be expressed with transfer functions

### Controller (Compensator)

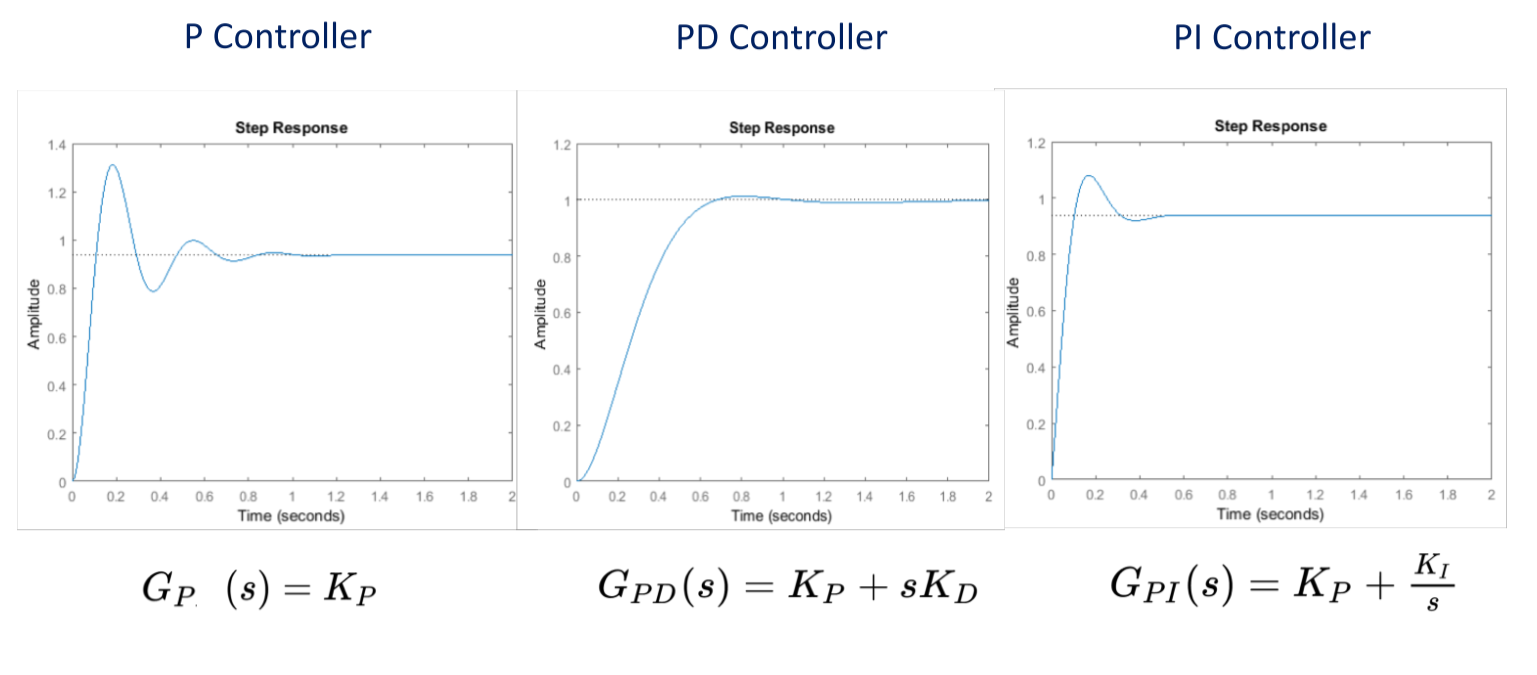
* Simple algorithms vs. more complex algorithms
* Simple:
  + Lead-lag controllers
  + PID controllers
* Complex:
  + Non-linear methods: feedback linearization, backstepping, sliding mode
  + Optimization methods: model predictive control

## PID Controller

**Time domain**:

* – proportional gain
* – integral gain
* – derivative gain

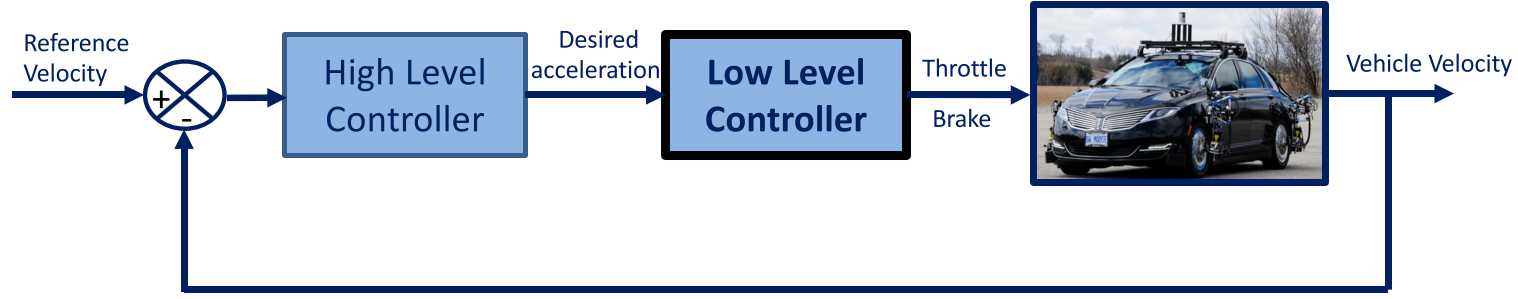
**Laplace domain**:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Closed Loop Response** | **Rise Time** | **Overshoot** | **Settling Time** | **Steady-State Error** |
| Increase | Decrease | Increase | Small change | Decrease |
| Increase | Decrease | Increase | Increase | Eliminate |
| Increase | Small change | Decrease | Decrease | Small change |

## Longitudinal Speed Control

* **Cruise control**: vehicle speed is controlled by throttling or braking, to be kept at the reference speed



### Upper Level Controller

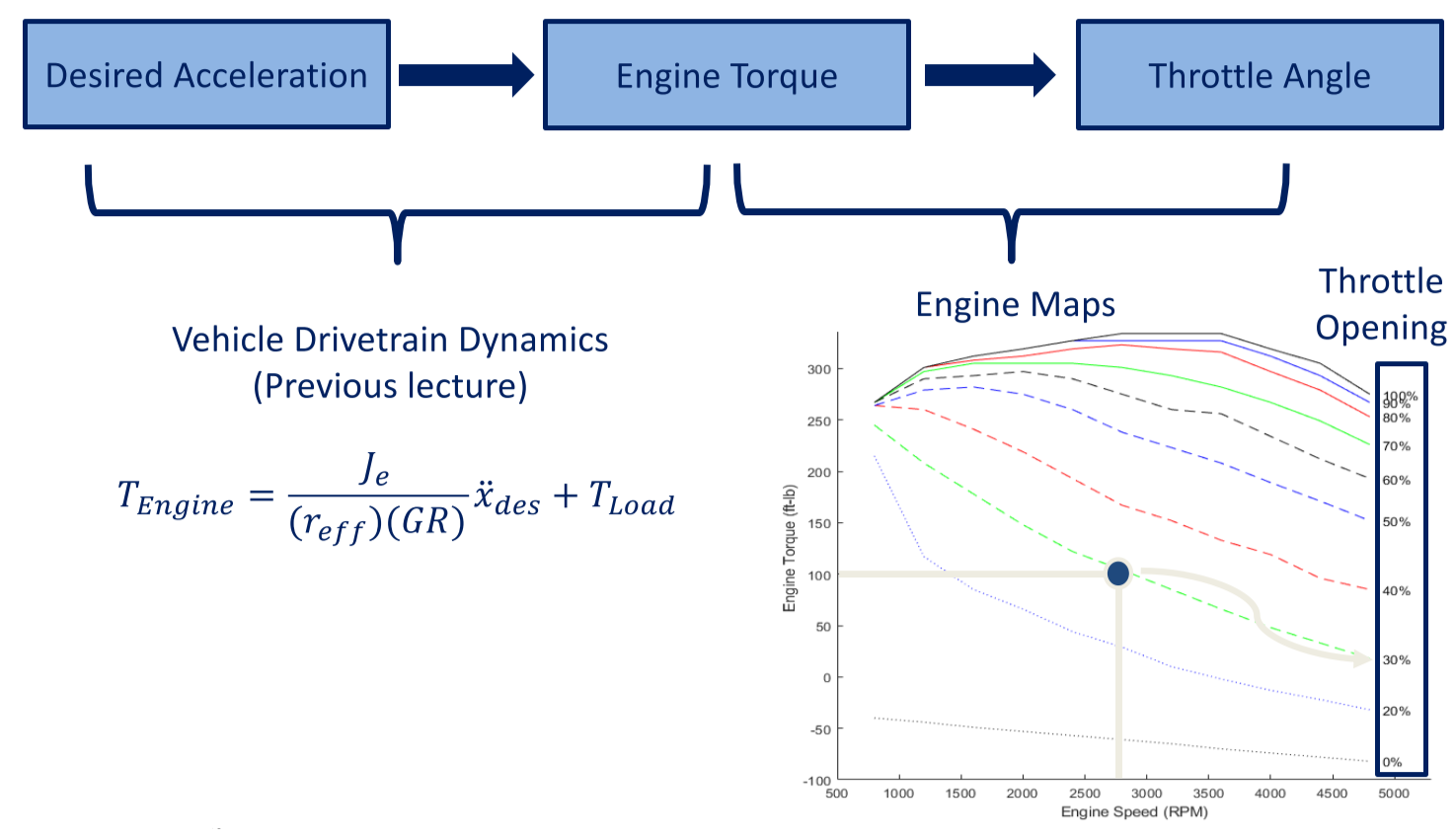
Determines desired acceleration , based on reference and actual velocity

### Lower Level Controller

Calculates throttle input such that the vehicle can track the desired acceleration determined by the upper level controller

Assumptions:

* Only throttle actuations are considered – no braking
* The torque converter is locked
* Small tire slip

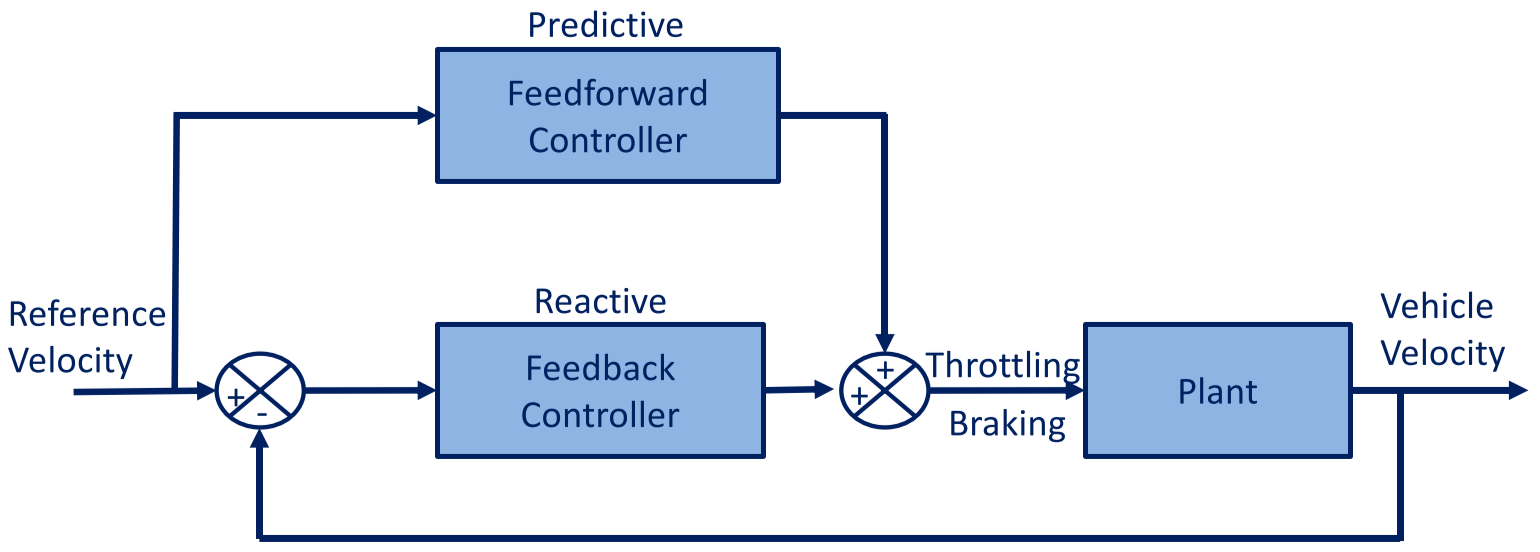


**For braking**:

* Set a **dead band** – no braking within the dead band, vehicle decelerates by itself
* Below the dead band, apply brake
  + Map vehicle deceleration to brake torque, using vehicle mass and tire effective radius
  + Use brake map to look up brake pedal position for a given torque request

## Feedback vs. Feedforward Control

* They are often used together
  + Feedforward controller provides predictive response, non-zero offset
  + Feedback controller corrects the prediction, compensating for disturbances and errors



* Output of feedforward and feedback control blocks are the throttling/braking signals to accelerate/decelerate the plant (vehicle), in order to maintain the reference velocity

### Feedforward Table

Feedforward component makes the vehicle converge to the reference speed faster, by determining the throttle needed to overcome the resistance forces at the given speed

**Reference velocity 🡪 Wheel angular speed**

**🡪 Engine angular speed**

**🡪 Engine torque**

**🡪 Throttle load**

# 11-2 Vehicle Control: Lateral Control

## Lateral Control Design

1. Define error relative to desired path
2. Select control law that drives error to zero and satisfies input constraints
3. Add dynamic consideration to account for forces/moments acting on the vehicle

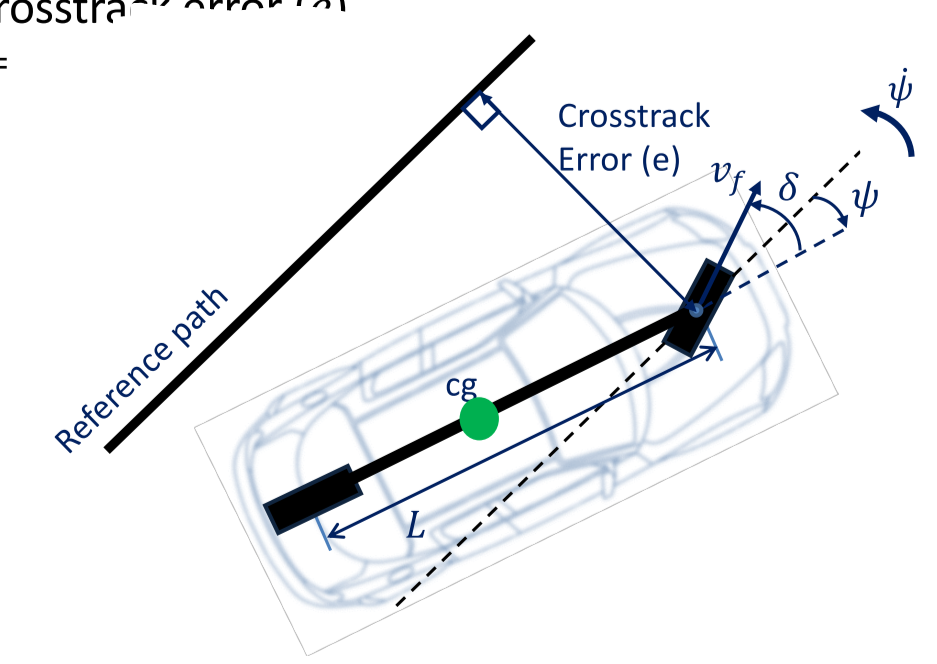
### Reference Path

* Track: straight line segments, waypoints, or parameterized curves
* Main control goals:
  + Align heading path
  + Eliminate offset to path

### Types of Control Design

* **Geometric controllers**: pure pursuit, Stanley
* **Dynamic controllers**: MPC control, sliding mode, feedback linearization

## Lateral Controller Input

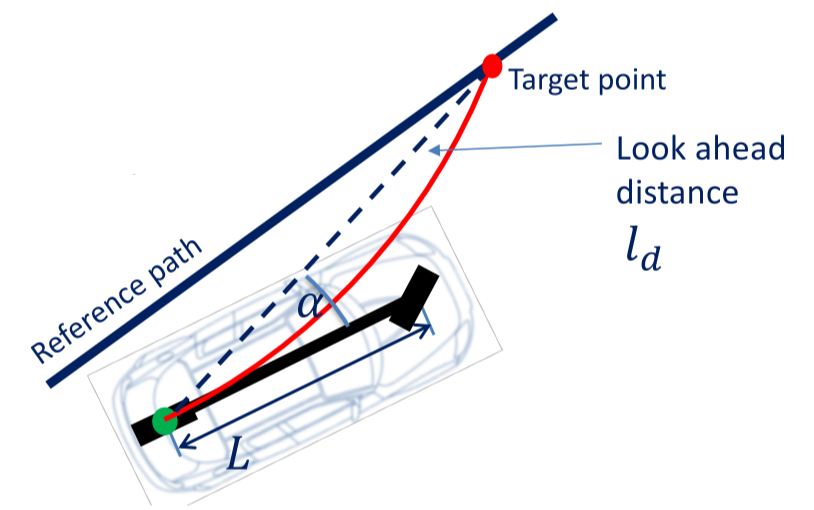


* **Heading** **error** – heading relative to the reference path; want it to be zero
* **Cross-track error** – distance from center of front axle to closest point on the reference path
* **Rate of change of cross-track error**

## Geometric Path Tracking

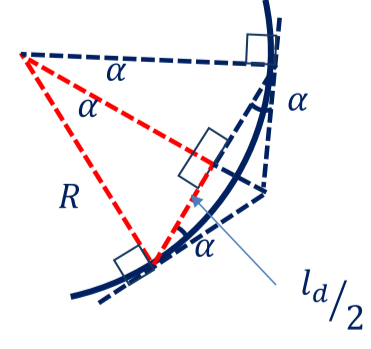
* Exploits geometric relationship between vehicle and path, resulting in compact control law solutions
* Uses reference point on path to measure error of vehicle
* Reference point can be ahead of vehicle

## Pure Pursuit

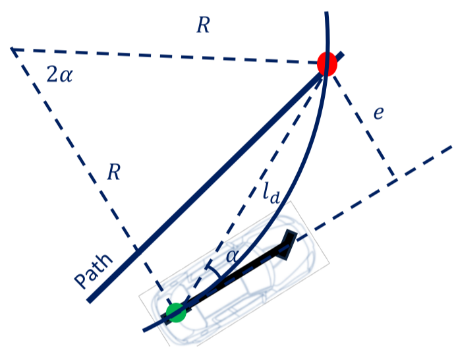


* Geometrically calculate path curvature
* Connect center of rear axle to a target point on the path ahead of the vehicle, with a lookahead distance
* Drive in a circular arc from the current position to the target point
  + As vehicle drives forward, target point also moves forward, and arc curvature is updated

**Curvature** is calculated by the lookahead distance and the angle between the vehicle’s heading direction and the lookahead direction:



**Cross-track error**  isdefinedas the lateral distance between the heading vector and the target point:



### Pure Pursuit Formulation

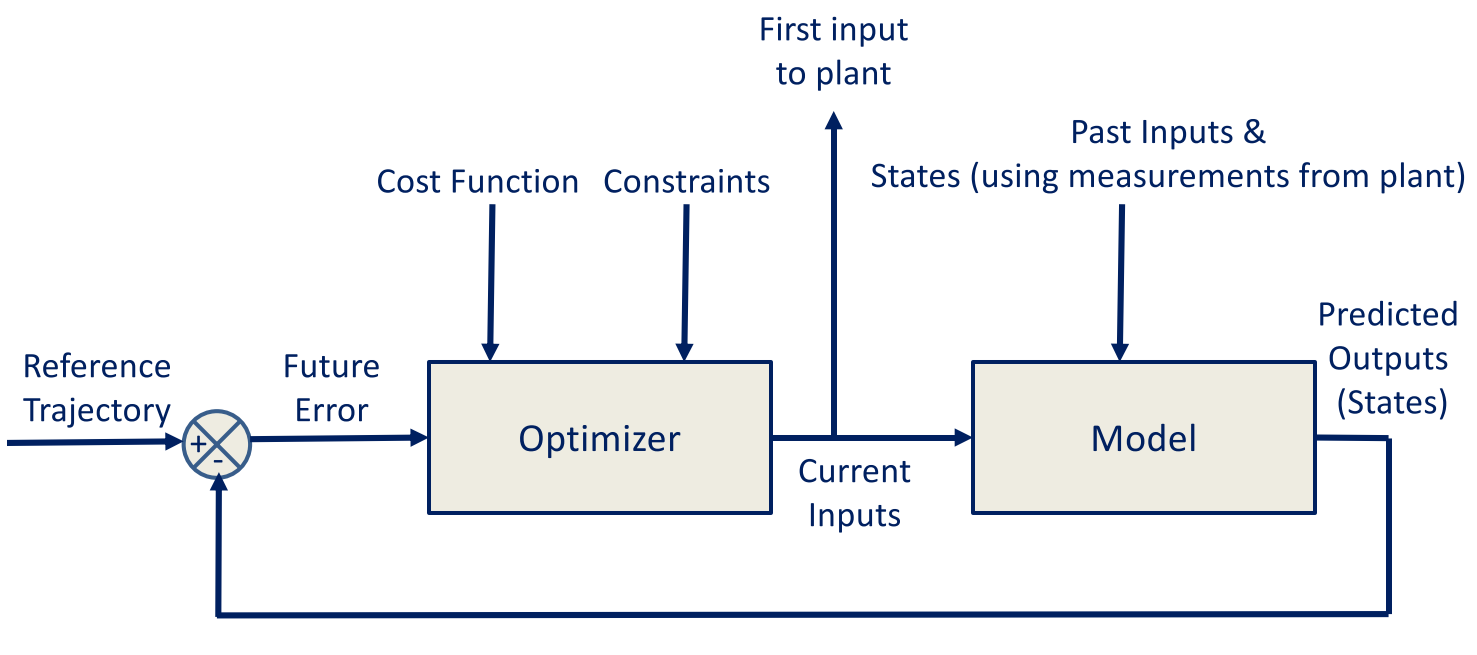
* **Proportional** controller of the steering angle
* Operates on **cross-track error** and a **lookahead distance** in front of the vehicle
* The proportional gain can be tuned at different speeds ( is a function of vehicle speed)
* – minimum lookahead distance
* – longitudinal velocity

## Model-Predictive Control (MPC)

* Solves a numerical optimization problem at each timestep
* Advantages:
  + Straightforward formulation
  + Explicitly handles constraints
  + Applicable to linear or non-linear models
* Disadvantages:
  + Computationally expensive

### Receding Horizon Approach

* Pick receding horizontal length
* For each timestep :
  + Set initial state to predicted state
  + Perform optimization from to , while travelling from to
  + At time , apply first control command



### Linear MPC Formulation

* Linear time-invariant discrete time model:
* MPC seeks to find a control policy :
* Objective function – regulation (achieving zero output)
* Objective function – tracking a reference

### Model-Predictive Controller

* Minimizes errors:
  + Deviation from desired trajectory
  + Control command magnitude
* Subject to:
  + Longitudinal and lateral dynamic models
  + Tire force limits

# 12-1 Motion Planning: Problem Definition

## Atonomous Driving Mission – Strategic Decisions

* **Mission**: navigate from point A to point B on the map
* **Mission planning**: higher-level planning, abstracts away lower-level details
  + Goal: find most efficient path (time or distance)

## Tactical Behaviour – Road Structure Scenarios

* Road structure influences driving scenario through lane boundaries and regulatory elements
* Simplest case: **driving straight**, following center of lane
  + Minimize deviation from center line
  + Attain reference speed for efficiency
* **Lane changes** are more complex
  + Different shapes for different situations
  + Shape depends on vehicle speed and acceleration limitations
  + Time horizon of execution affects aggressiveness of change
* **Left and right turns** are common
  + Shape of turn varies
  + State of surrounding environment affects ability to make turns
* **U-turns** are useful for efficient direction changes
  + Shape of U-turn depends on car’s speed and acceleration limits
  + Not always allowed/possible at intersections

## Tactical Behaviour – Obstacle Scenarios

* Static and dynamic obstacles impact driving scenario
  + **Static** **obstacles** impact which locations the path can occupy
  + **Dynamic obstacles** are most often the vehicle in front of the ego vehicle – need to maintain safe distance
    - They also impact turns and lane changes
    - Need to use estimation and prediction to calculate windows of opportunity

## Maneuvers

* Speed tracking
* Deceleration to stop
* Stay stopped
* Yield
* Emergency stops

### Challenges

* Only covers a small subset of scenarios – focus on common cases
* Edge cases (e.g. lane splitting, jaywalking) make driving task complex

## Hierarchical Planning

Mission Planner 🡪 Behavioural Planner 🡪 Local Planner (Path Planner + Velocity Profile Generator) 🡪 Vehicle Control

* Hierarchy of optimization problems
* Higher up = more abstraction
* Each optimization problem has constraints and objective functions

# 12-2 Motion Planning: Trajectory Planning Constraints and Objectives

## Main Constraints

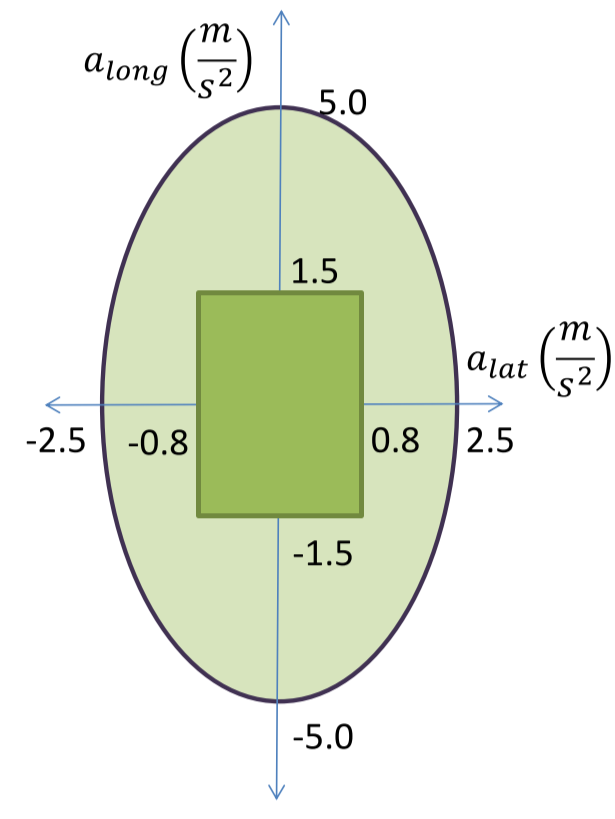
### Vehicle Model

* Trajectory planner solutions must obey vehicle model equations
* Reasonable model for low speeds is bicycle kinematic model:
  + Imposes non-holonomic curvature constraint on path planning process
    - Reduces the directions that the vehicle can travel at any point

### Curvature

* For a planar curve, curvature is the rate of change of the tangent unit vector with respect to arc length
* Assuming bicycle kinematic model, maximum steering angle of a vehicle imposes a maximum curvature constraint

### Vehicle Dynamics



* **Friction ellipses** denotes maximum magnitude of tire forces before skidding
  + Friction forces are the extreme limit – for planning emergency maneuvers
  + For normal driving, limit lateral and longitudinal acceleration to “**comfort rectangle**”
* Turning limits the maximum available braking
* Braking limits the maximum available lateral acceleration
* Assuming constant longitudinal speed, lateral acceleration is **centripetal** – function of instantaneous turning radius of path + velocity
  + Velocity is constrained by path curvature and lateral acceleration

### Static and Dynamic Obstacles

* Static obstacles are encoded in occupancy grid, blocks portions of the workspace
* Satisfy static obstacle constraints by **collision checking**
  + Can check for collisions using swath of the vehicle’s path
  + Can check for closest obstacles along the vehicle’s path
* Static obstacles affect paths
* Dynamic obstacles affect trajectories
  + Simplest case: same path, modified speed profile

### Rules of the Road / Regulatory Elements

* Lane constraints restrict path locations
* Signs and traffic lights influence vehicle behaviour

## Optimization Objectives

* System level:
  1. Safety/reduce risk of crashes
  2. Progress
  3. Energy efficiency
  4. Comfort
* Trajectory generation level: implement objectives using objective functions

### Efficiency

* **Path length** – minimize arc length of path, to generate shortest path to goal
* **Travel time** – minimize time to destination, while following planned path

### Reference Tracking

* Penalize deviation from reference path or speed profile
* Hinge loss – penalize speed limit violations severely

### Smoothness

**Jerk**:

* High jerk means faster change in vehicle acceleration, which doesn’t allow muscles in passengers’ bodies enough time to adjust to the change in forces

**Curvature**:

* Penalizes high, concentrated curvature – prefers low, distributed curvature for smoothness and comfort

# 13-2 Motion Planning: Local Planning Methods

## Trajectory Rollout Planner

* + - 1. Uses **trajectory propagation** to generate a set of candidate trajectories
      2. Among collision-free trajectories, select the one that makes most progress towards goal

Steps:

1. Discretely sample vehicle’s control space
2. **Trajectory propagation**: for each sampled input, perform forward simulation from robot’s current state to predict its trajectory
   * Use **dynamic windowing** to eliminate infeasible control inputs
3. **Trajectory evaluation**:
   * Check collisions on each trajectory, discard ones that collide with obstacles
   * Apply cost function to remaining trajectories, pick one with lowest cost
4. **Execute** **initial portion** of the selected trajectory
5. Repeat

## Vehicle Model for Trajectory Propagation – Bicycle Kinematic



* and are rear-axle positions of the robot
* is the heading of the chassis with respect to the x-axis
* is the steering angle input, is the velocity input
* **Discretization** of differential equations allows for efficient computation of trajectories
* Recursive definition also saves computation time

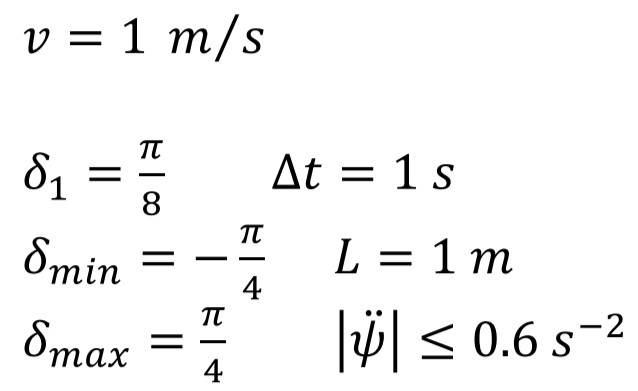
### Trajectory Set Generation

* Holding the velocity constant and varying the steering angle gives a candidate set of trajectories
* Each trajectory corresponds to a fixed control input
  + Typically, uniformly sampled across range of possible inputs
  + More sampled trajectories lead to more maneuverability
  + Fewer sampled trajectories improve computation time

### Dynamic Windowing

* Add dynamic constraints as control input parameters
* Angular acceleration constraint may prevent selection of certain maneuvers based on current angular velocity
* Change in steering angle between cycles is also bounded

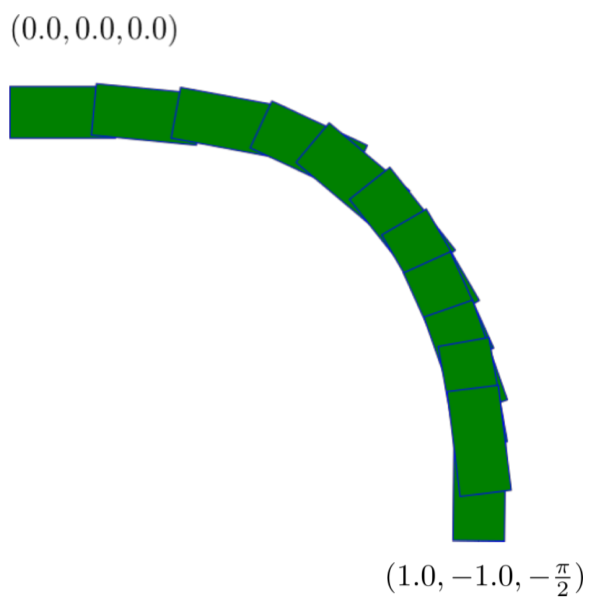
Example: Given current steering angle and angular acceleration bound, which candidate trajectories do not violate the bound?



## Collision Checking Challenges

* Computationally intensive
* Requires perfect information to guarantee safety
* Needs to be approximated but must be robust to noise

### Swath Computation



* Get area occupied by car along path by rotating the car’s footprint by each along the path
* Swath is the union of all the rotated/translated footprints
* Use swath to check for collisions

## Speed and Robustness

* Improve speed
* Be robust to noise
* Use conservative approximations to solve these two problems
* Want algorithmic speedup without sacrificing path quality

### Conservative Approximations

* May report a collision even if there isn’t one, but will never miss a collision if there is one
* Completely encapsulate car in 3 circles
  + Circle approximation is effective because it’s fast to check if an occupancy grid point likes within a circle with radius centered at
  + If obstacle in occupancy grid lies within circle, a collision is reported; otherwise, no collision is possible
  + Can also apply a **distance transform** on the occupancy grid, marking each cell with distance to closest occupied cell
    - Collision occurs if circle center is annotated with a distance
* Collision checking accuracy is affected by **resolution of discretization**
* Higher fidelity collision checking requires finer resolution for occupancy grids and path points, and thus more computation resources

## Objective Function

Rewards progress towards goal point



Distance to goal Curvature Deviation from centerline

# 13-3 Motion Planning: Local Planning Methods II

## Conformal Lattice Planner on Autonomoose

1. Sample set of goal points across the road, at a planning location
2. Use variational planning to generate drivable and smooth paths to each goal point
3. Eliminate paths that collide with obstacles
4. From the remaining paths, select one closest to the reference point

## Variational Planners

* Optimize trajectory according to cost function   
  + Contains penalties for collision avoidance and robot dynamics
  + Generates high quality paths, but can be slower and less likely to converge to feasible solution

## Parametric Curves

* Described by a set of parameterized equations
* Parameter denotes path traversal - can be arc length or unitless
* **Boundary conditions** must hold on either endpoint of the path
* **Maximum curvature** cannot be exceeded – ensures that car can drive along path

### Path Optimization

* Parametric curves allow for optimization (according to cost function ) over parameter space, which simplifies optimization formulation
* Two common parameterized curves: **quintic splines** and **cubic spirals**
  + Both allow satisfaction of boundary conditions and can be optimized parametrically

## Quintic Splines

* and defined by 5th order splines
* Closed-form solution available for boundary conditions
* Challenging to constrain curvature, due to potential discontinuities in curvature or its derivatives

## Cubic Spirals

* Spirals defined by their curvature as a function of arc length
* Closed-form curvature definition allows simple curvature constraint checking

`

* Boundary conditions specify starting state and required ending state
* Spiral end position requires approximation, as it lacks closed-form solution

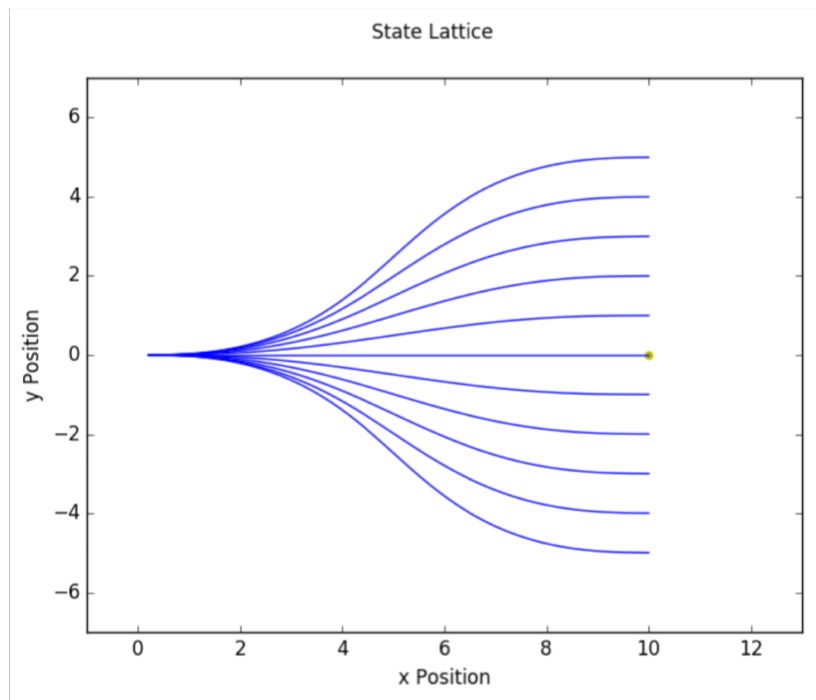
### Position Integrals and Simpson’s Rule

* **Simpson’s rule** has improved accuracy over other methods
* Divide the integral into regions, evaluate the function at the region boundaries
* Using Simpson’s approximation, can write boundary conditions in terms of spiral parameters
* Can generate a spiral that satisfies boundary conditions, by optimizing its spiral parameters and its length

### Bending Energy Objective

* Distributes curvature more evenly along spiral, to promote comfort
* Take integral of square of curvature along path

## Conformal Lattice



* Exploits road structure to speed up planning feasible collision-free path to goal
* Lattice paths are **laterally offset** from a goal point along the road

## Goal Horizon

* Short lookahead improves computation time, but reduces ability to avoid obstacles
* Long lookahead is more accurate, but more computationally expensive
* Goal point is dynamically calculated based on factors including vehicle speed
* Endpoints are sampled laterally offset from goal, according to heading along the road

## Generating Spirals

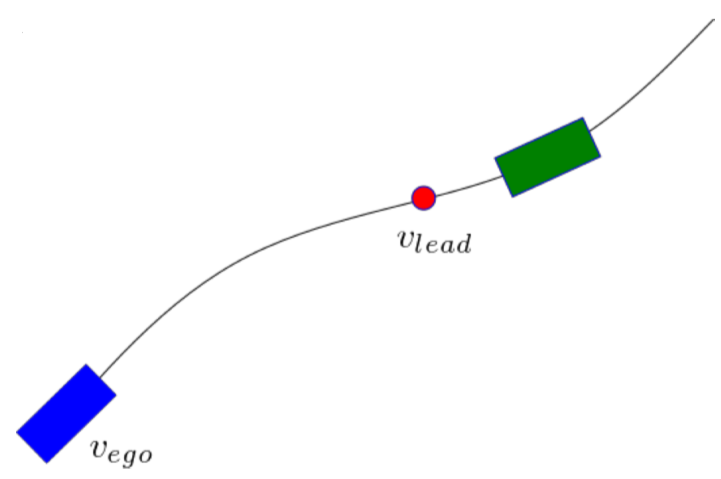
* Compute cubic spirals to each goal point
* If goal point cannot be reached by spiral, discard it
* **Trapezoid rule integration**: uses numerical integration to generate positions along path
  + Faster than Simpson’s rule at generating entire path
* Do collision-checking using swath-based or circle-based method

# 13-4 Motion Planning: Local Planning III

## Behavioural Planner Reference Velocity

* Need to compute a reference velocity
* Can use speed limit as a starting point
* Behavioural maneuver also influences reference velocity

### Dynamic Obstacles



* Lead dynamic obstacles regulate speed of vehicle, to prevent collisions
* **Time to collision** is important metric to preserve when driving with lead vehicles
* Need to reach red point at lead vehicle speed to ensure no collision

## Curvature and Lateral Acceleration

### Linear Ramp Profile

* Simplest shape: linear ramp to desired velocity
* We know total arc length of path , and initial/final velocities
* Calculate acceleration:
* For a given acceleration, we can calculate velocity at each point by using the accumulated arc length up to that point

### Trapezoidal Profile





* Car decelerates to slower speed before stopping
* Useful for scenarios like stop signs
* Deceleration chosen well within comfort rectangle, to maximize passenger comfort

**Step 1**: determine distance required to reach transit velocity using gentle deceleration

**Step 2**: repeat process to reach a stop from , using gentle deceleration