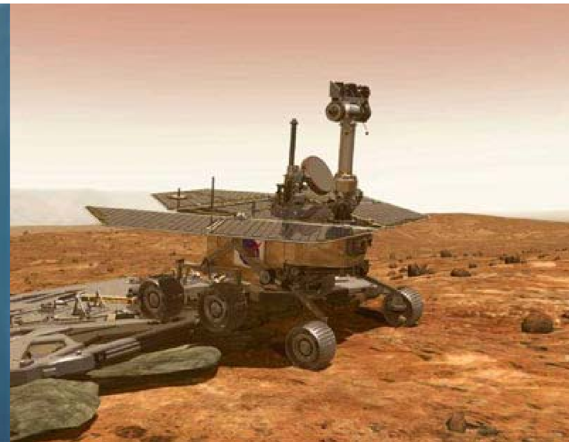
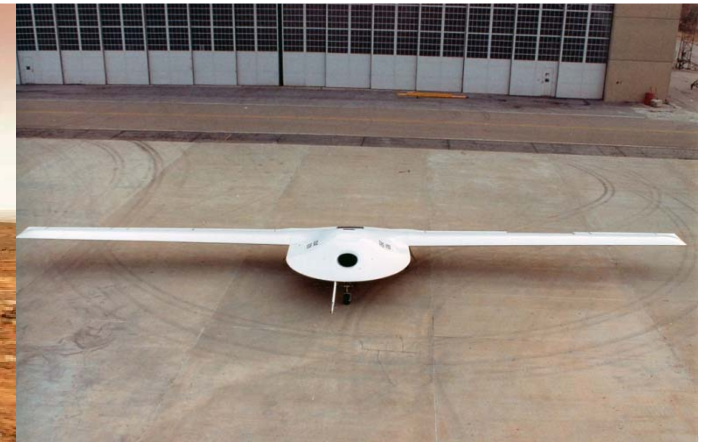
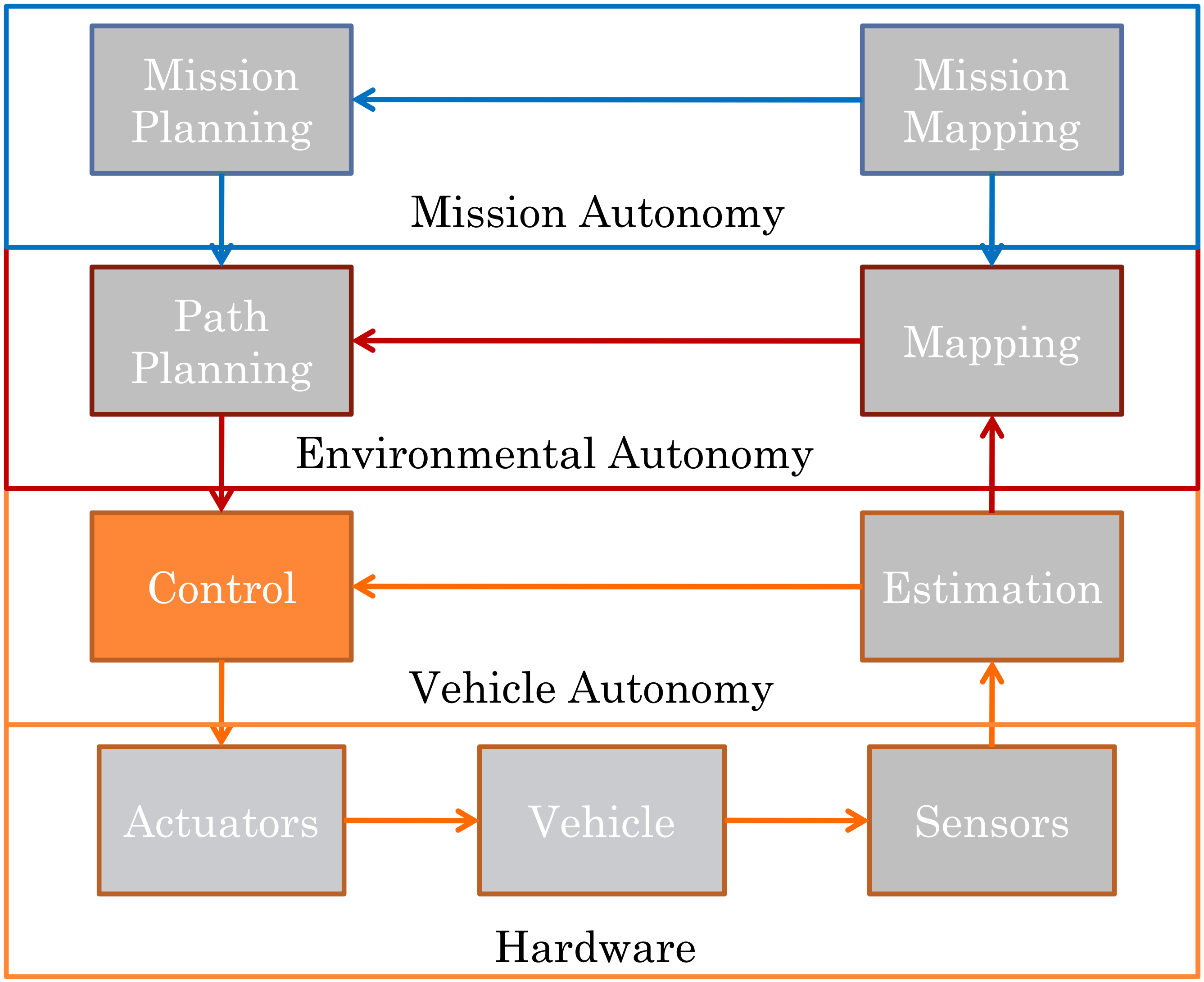





ME 597: Autonomous Mobile Robotics
Section 4 – Control

Prof. Steven Waslander

Components


Mission Planning
Mission Mapping
Mission Autonomy
Path Planning
Mapping
Environmental Autonomy
Control
Estimation
Vehicle Autonomy
Actuators
Vehicle
Sensors
Hardware

Outline

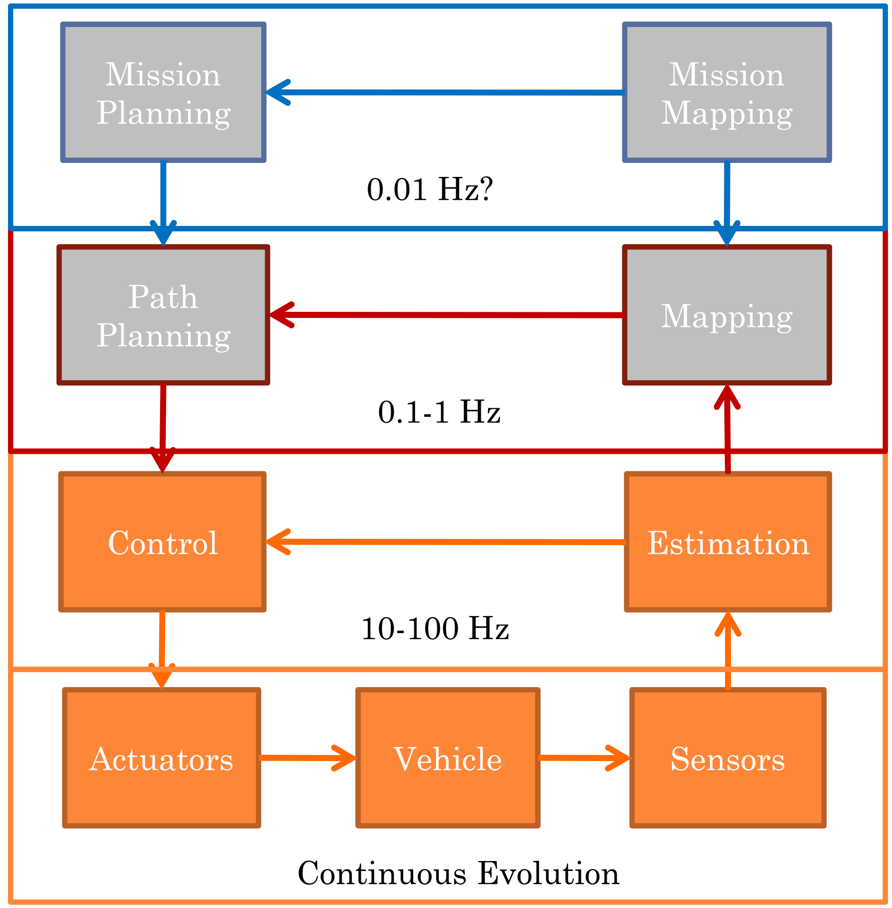
- Control Structures
- Linear Motion Models
  - PID Control
  - Linear Quadratic Regulator
  - Tracking
- Nonlinear Motion Models
  - Description of main methods
  - Geometric driving controller

Control Structures

- Regulation
  - Maintaining a constant desired state.
- Path Following
  - Tracking a state trajectory defined in state only, but not restricted in time.
- Trajectory Tracking
  - Tracking a state trajectory with explicit timing.

Control Structure

- Time-Scale Separation
  - Using multi-loop feedback analogy
  - Estimation and control performed much more quickly than mapping and planning
  - Possible to ignore inner loops when developing higher levels of control
  - Abstractions must be consistent



Typical Timescales

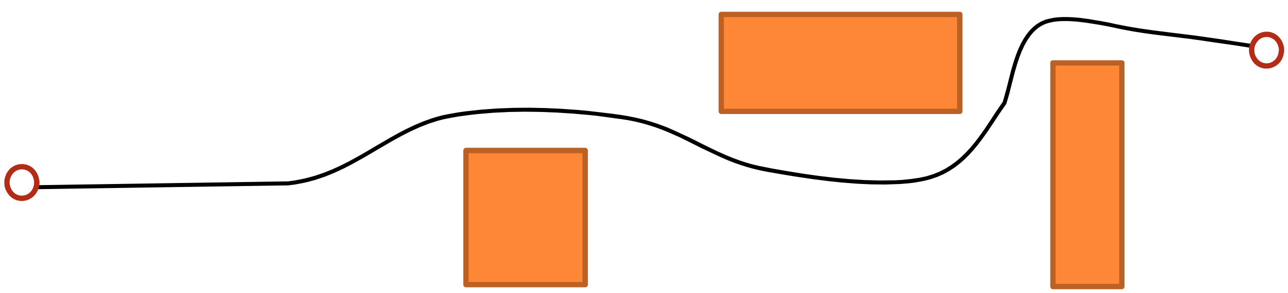
Control Structure

- Separating planning and control timescales
  - Pros
    - Simplified planning, often to make it real-time
    - Guarantees on stability
    - Can operate without plan, through human-in-the-loop
  - Cons
    - Planning interval may require use of old state information
    - Resulting trajectories may not be optimal
    - Trajectories may collide with environment
    - Planner may not be able to consider dynamic constraints
      - Provide infeasible paths

CONTROL STRUCTURES

- Planner Outputs
  - Full trajectory defined by open loop inputs
    - At each time step, desired inputs specified
    - Pre-computed open loop control
    - May still require feedback for disturbance rejection
    - Often not at frequency of controller
    - Superscript t for trajectory

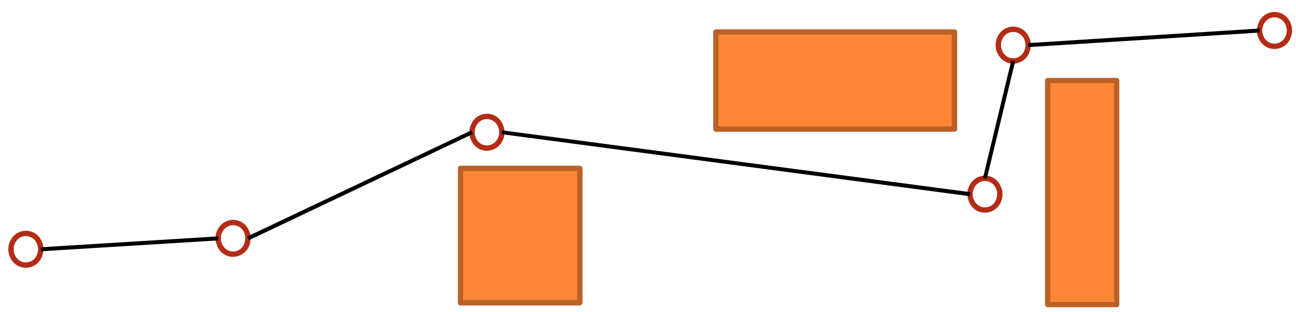
$$\pi_t^t = \{u_t^t, \ldots, u_{t+N}^t\}$$



Control Structures

- Planner Outputs
  - Waypoints
    - Position coordinates to achieve
      - With/without timing constraints
    - Joined by straight line segments to create a path

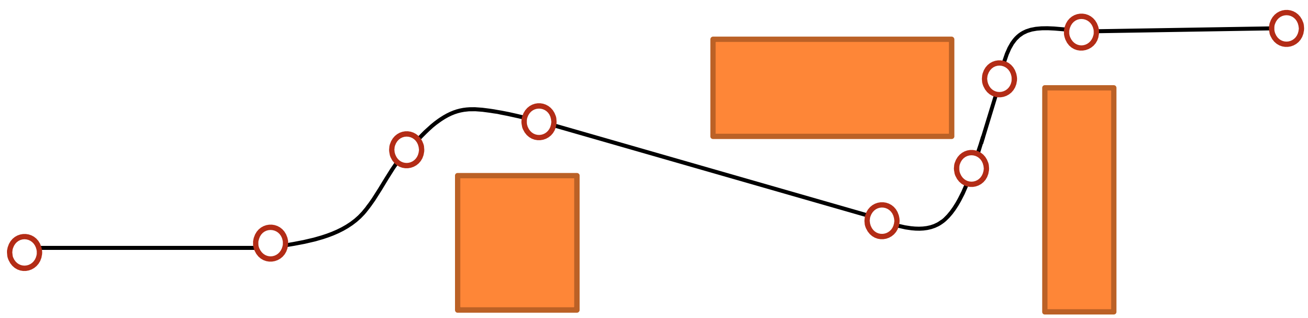
$$\pi_t^{wp} = \{x_t^{wp}, \ldots, x_{t+N}^{wp}\}$$



CONTROL STRUCTURES

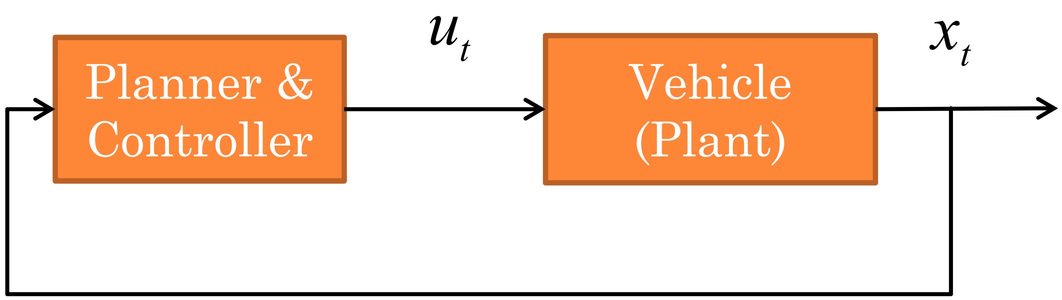
- Planner Outputs
  - Motion primitives
    - A sequence of predefined motions
      - E.g. Straight lines and curves of defined radius
      - End point of each segment easily calculated
      - Often parameterized to admit an array of options

$$\pi_t^{mp} = \{m_1^{mp}, \ldots, m_M^{mp}\}$$



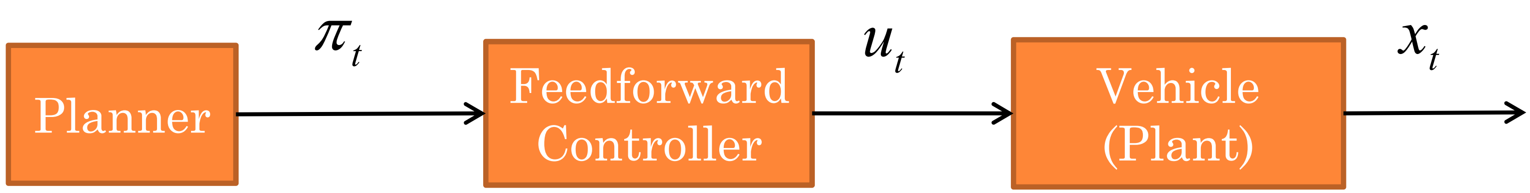
CONTROL STRUCTURES

- Block Diagrams
  - Combined Planner and Controller
    - Planner generates desired state and inputs at every time step
    - Replan given new information at each time step



CONTROL STRUCTURES

- Block Diagrams
  - Planner with Feedforward control
    - Planner generates a desired plan, π_t
      - Direction to head in
      - Speed of travel etc.
    - Feedforward controller converts it into inputs
      - Inverse dynamics needed to make conversion



Control Structure

- Open loop often works
  - e.g. Open loop on RC steering
    - Steering has embedded position control in servo
    - From robot perspective, commanded angles are achieved

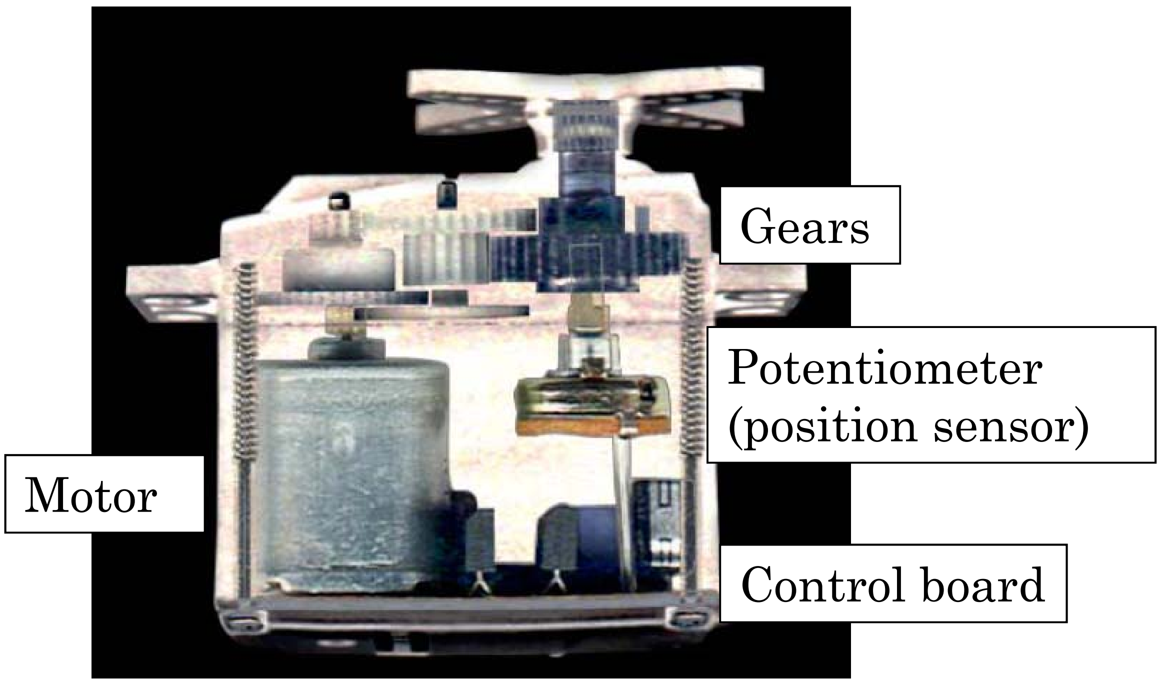
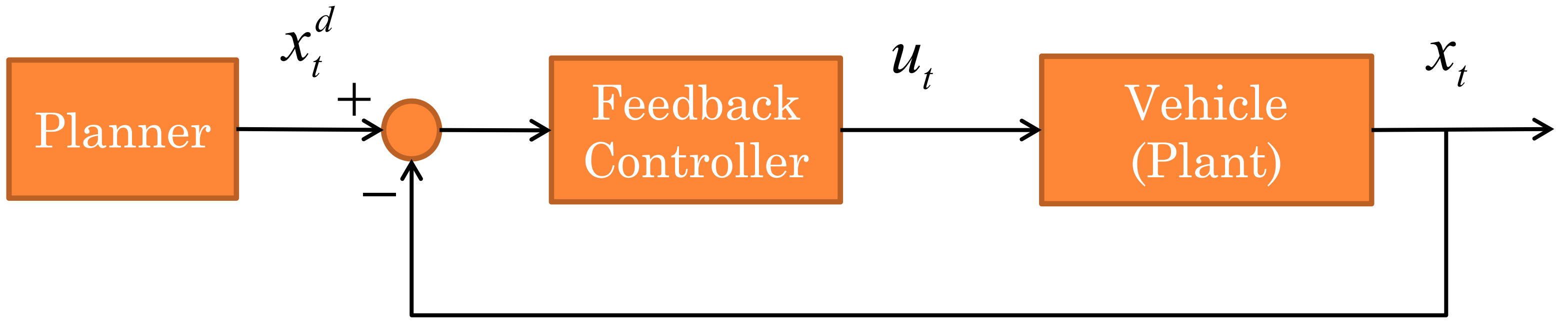


Image courtesy of Darren Sawicz

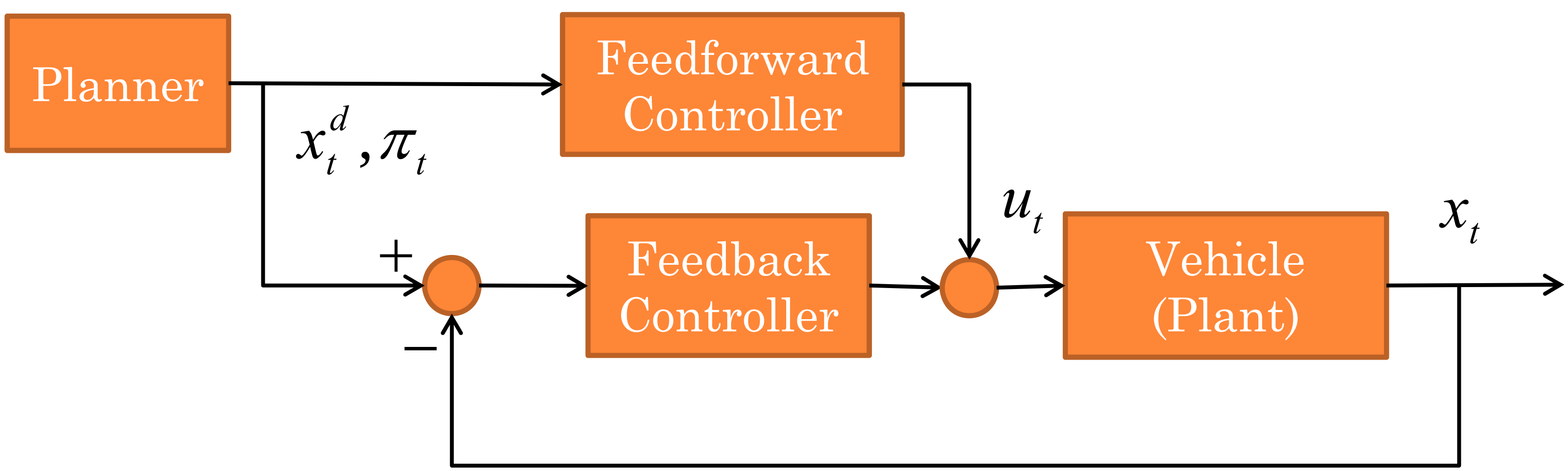
Control Structures

- Block Diagrams
  - Planner with Feedback control for regulation
    - Planner generates instantaneous desired state
      - Rely on timescale separation for control design
      - Used with high frequency inner loop control



Control Structures

- Block Diagrams
  - Planner with Feedback & Feedforward control
    - Planner generates desired state
    - Feedforward controller generates required open loop input
    - Feedback controller eliminates errors from disturbances, unmodeled dynamics

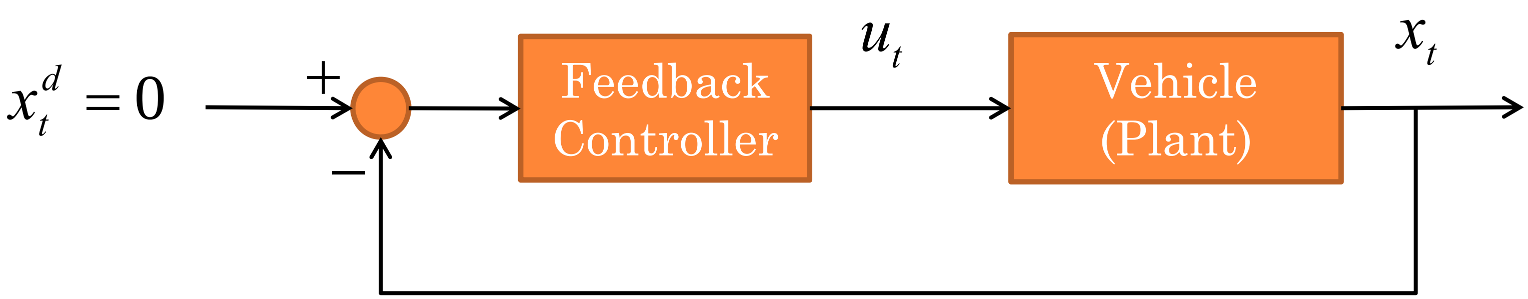


OUTLINE

- Control Structures
- Linear Motion Models
  - PID Control
  - Linear Quadratic Regulator
  - Tracking
- Nonlinear Motion Models
  - Description of main methods
  - Geometric driving controller

Linear Control Design

- Assume linear dynamics
- Start with regulation problem
- Adapt to tracking afterwards
- Control Structure:
  - Pure Feedback for regulation
  - Feedforward/feedback for tracking



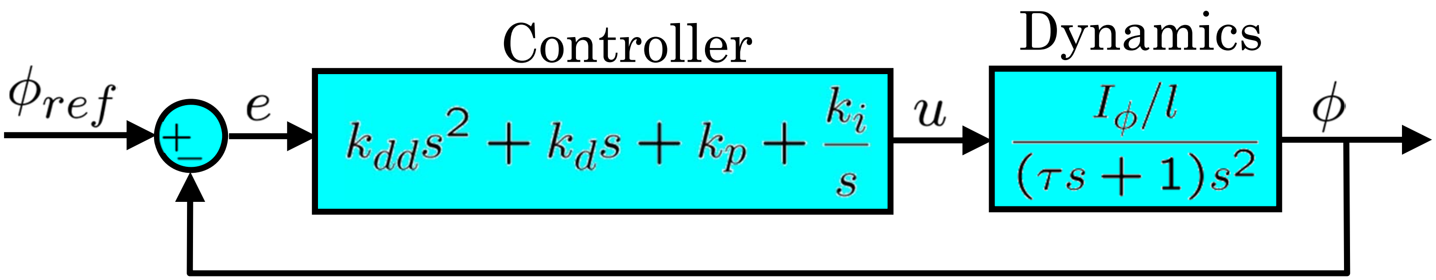
PID CONTROL

- Proportional-Integral-Derivative control
  - e.g. for velocity control of ground robots

$$u_t = K_p e_t + K_i \sum_{t=0}^{t} e_t dt + K_d \dot{e}_t$$

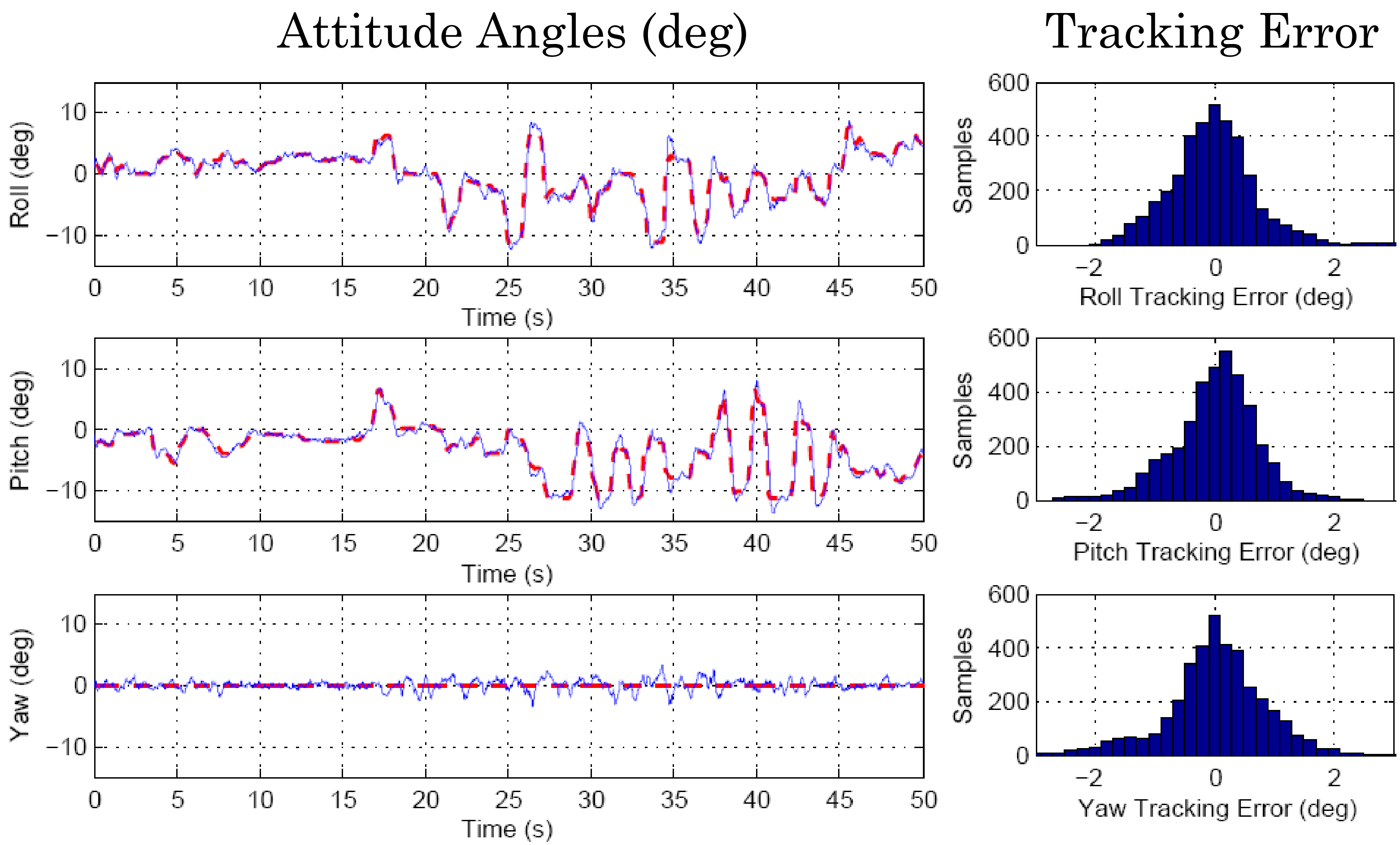
  - Particularly effective for SISO linear systems, or systems where inputs can be actuated in a decoupled manner
  - Proportional and derivative govern time response, stability
  - Integral eliminates steady state errors, sensor biases and constant disturbances
  - Can be used to track reference signals (up to bandwidth of closed loop system)

Quadrotor Attitude Control



- Key Developments
  - Angular Accel. Feedback (specific thrust)
  - Command Tracking
  - Frame Stiffness
  - Tip Vortex Impingement

Tracking Reference Commands



Root mean square error of 0.65°

Linear Control Design

- Linear Quadratic Regulator
  - Linear Plant Model
  - Quadratic penalty on deviation from desired state and on control input usage
  - The controller optimally regulates all state errors to 0
- Derivation of optimal control will rely on backward induction
  - Recall Dynamic Programming

Linear Quadratic Regulator

- Discrete time version
  - Same notation as Thrun, Fox
  - Define initial and final times

$$t_0, \quad t_f$$

  - Linear motion model

$$x_t = A_t x_{t-1} + B_t u_t$$

    - Disturbances can be ignored, leads to same result

  - Assume we know the state at each timestep, including initial state

$$x(t_0) = x_0$$

Linear Quadratic Regulator

- Goal: Drive all states to zero!
  - Regulation, not tracking

- Cost Definition:
  - Tradeoff between error in states and use of control

$$J\left(x_{t_0:t_f}, u_{t_0+1:t_f}\right) = \frac{1}{2} x_{t_f}^T Q_{t_f} x_{t_f} + \frac{1}{2} \sum_{t=t_0+1}^{t_f} \left(x_{t-1}^T Q_{t-1} x_{t-1} + u_t^T R_t u_t \right)$$

Final Cost | State Cost | Control Cost

- LQR Problem: Find sequence of inputs that minimizes *J*
  - subject to dynamics, boundary conditions

LINEAR QUADRATIC REGULATOR

- A note on "Quadratic Cost"
  - Since state and input are vectors, quadratic penalties are written as

$$x^T Q x$$

    - Where x_t is an nX1 vector, and Q is an nXn weighting matrix that decides how to penalize each state separately
  - For example, suppose x_t = [N E D], the position of a vehicle in North, East and Down coordinates.
  - If we care more about errors in the horizontal than the vertical plane, we might pick a Q as follows:

$$Q = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Longrightarrow x_t^T Q x_t = \begin{bmatrix} N & E & D \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} N \\ E \\ D \end{bmatrix}$$

$$= 10N^2 + 10E^2 + 1D^2$$

Linear Quadratic Regulator

- Derivation
  - Aim to formulate as a backward induction problem, and solve for minimum at each backward time step

$$J_t = \min_{u_t}\left[L(x_{t-1}, u_t) + J_{t+1}\right]$$

  - End condition is known
    - Defined to have this quadratic form

$$J_{t_f} = \frac{1}{2} x_{t_f}^T Q_{t_f} x_{t_f}$$

LINEAR QUADRATIC REGULATOR

- Derivation
  - Assume J_t is of specific quadratic form

$$J_t = \frac{1}{2} x_t^T P_t x_t$$

  - Find J_{t-1} in the same form
  - Done by rewriting the optimal cost as

$$J_{t-1} = \min_{u_t} \left[\frac{1}{2} x_{t-1}^T Q_{t-1} x_{t-1} + \frac{1}{2} u_t^T R_t u_t + J_t \right]$$

Stage cost

Cost to Go

LINEAR QUADRATIC REGULATOR

- Derivation
  - Substituting in for J_t

$$J_{t-1} = \min_{u_t} \frac{1}{2}\left[x_{t-1}^T Q_{t-1} x_{t-1} + u_t^T R_t u_t + x_t^T P_t x_t \right]$$

  - Incorporating dynamic constraints

$$J_{t-1} = \min_{u_t} \frac{1}{2}\Big[x_{t-1}^T Q_{t-1} x_{t-1} + u_t^T R_t u_t \\ + (A_t x_{t-1} + B_t u_t)^T P_t (A_t x_{t-1} + B_t u_t) \Big]$$

Linear Quadratic Regulator

- Derivation
  - Expanding

$$J_{t-1} = \min_{u_t} \frac{1}{2}\Big[x_{t-1}^T Q_{t-1} x_{t-1} + u_t^T R_t u_t$$
$$+ x_{t-1}^T A_t^T P_t A_t x_{t-1} + x_{t-1}^T A_t^T P_t B_t u_t$$
$$+ u_t^T B_t^T P_t A_t x_{t-1} + u_t^T B_t^T P_t B_t u_t \Big]$$

  - Now J_{t-1} is a function of only u_t, x_{t-1} and P_t, but neither of the last two depend on u_t
  - The minimization over u_t can be performed
    - Set derivative to zero and solve for u_t

LINEAR QUADRATIC REGULATOR

- Derivation
  - We rely on matrix derivatives

$$\frac{\partial J_{t-1}}{\partial u_t} = u_t^T R_t + x_{t-1}^T A_t^T P_t B_t + u_t^T B_t^T P_t B_t = 0$$

  - Transposing and grouping like terms together yields

$$\left(B_t^T P_t B_t + R_t \right) u_t = -B_t^T P_t A_t x_{t-1}$$

  - Next, an inverse is applied to define the control law

$$\begin{aligned} u_t^* &= -\left(B_t^T P_t B_t + R_t \right)^{-1} B_t^T P_t A_t x_{t-1} \\ &= -K_t x_{t-1} \end{aligned}$$

Linear Quadratic Regulator

- Derivation
  - Now we must complete the backward induction and demonstrate that

$$J_{t-1} = \frac{1}{2} x_{t-1}^T P_{t-1} x_{t-1}$$

  - To do so, we substitute in the optimal control input and simplify

$$J_{t-1} = \min_{u_t} \frac{1}{2} \Big[x_{t-1}^T Q_{t-1} x_{t-1} + u_t^{*T} R_t u_t^* + x_{t-1}^T A_t^T P_t A_t x_{t-1} + x_{t-1}^T A_t^T P_t B_t u_t^* + u_t^{*T} B_t^T P_t A_t x_{t-1} + u_t^{*T} B_t^T P_t B_t u_t^* \Big]$$

LINEAR QUADRATIC REGULATOR

- Derivation
  - Substituting

$$J_{t-1} = \frac{1}{2}\Big[x_{t-1}^T Q_{t-1} x_{t-1} + x_{t-1}^T K_t^T R_t K_t \, x_{t-1}^T$$
$$+ x_{t-1}^T A_t^T P_t A_t x_{t-1} - x_{t-1}^T A_t^T P_t B_t K_t \, x_{t-1}^T$$
$$- x_{t-1}^T K_t^T B_t^T P_t A_t x_{t-1} + x_{t-1}^T K_t^T B_t^T P_t B_t K_t \, x_{t-1} \Big]$$

  - Regrouping, we see J_{t-1} is of the right form

$$J_{t-1} = \frac{1}{2} x_{t-1}^T \Big[Q_{t-1} + K_t^T R_t K_t$$
$$+ A_t^T P_t A_t - A_t^T P_t B_t K_t$$
$$- K_t^T B_t^T P_t A_t + K_t^T B_t^T P_t B_t K_t \Big] x_{t-1}$$

LINEAR QUADRATIC REGULATOR

- Derivation
  - Finally, substituting in for K_t yields a simplified form for defining the relation from P_t to P_{t+1}
    - Will spare you the details

$$J_{t-1} = \frac{1}{2} x_{t-1}^T \left[Q_{t-1} + A_t^T P_t A_t - A_t^T P_t B_t (B_t^T P_t B_t + R_t)^{-1} B_t^T P_t A_t \right] x_{t-1}$$

  - As a result, we can define an update for P_{t-1}

$$P_{t-1} = Q_{t-1} + A_t^T P_t A_t - A_t^T P_t B_t (B_t^T P_t B_t + R_t)^{-1} B_t^T P_t A_t$$

  - The *costate* update does not depend on the state.
    - If you assume you will arrive at the desired end goal, can compute in advance

Linear Quadratic Regulator

- Summary of controller
  - Control
    - Depends on previous state and next costate

$$
\begin{aligned}
u_t &= -K_t x_{t-1} \\
&= -(B_t^T P_t B_t + R)^{-1} B_t^T P_t A_t x_{t-1}
\end{aligned}
$$

  - Costate update
    - Requires evolution backward in time from end state

$$
P_{t-1} = Q_{t-1} + A_t^T P_t A_t - A_t^T P_t B_t (B_t^T P_t B_t + R_t)^{-1} B_t^T P_t A_t
$$

Linear Quadratic Regulator

- Implementation of algorithm
  - Set final costate based on terminal cost matrix

$$\left.\begin{aligned} J_{t_f} &= \frac{1}{2} x_{t_f}^T Q_{t_f} x_{t_f} \\ J_t &= \frac{1}{2} x_t^T P_t x_t \end{aligned}\right\} \qquad P_{t_f} = Q_{t_f}$$

  - Solve for costate backward in time to initial time

$$P_{t-1} = Q_{t-1} + A_t^T P_t A_t - A_t^T P_t B_t (B_t^T P_t B_t + R_t)^{-1} B_t^T P_t A_t$$

  - Note: Both steps depend only on problem definition, not initial or final conditions

Linear Quadratic Regulator

- Implementation of algorithm
  - Next, find controller to use at each time step
    - Use pre-calculated costate to determine gain at time t

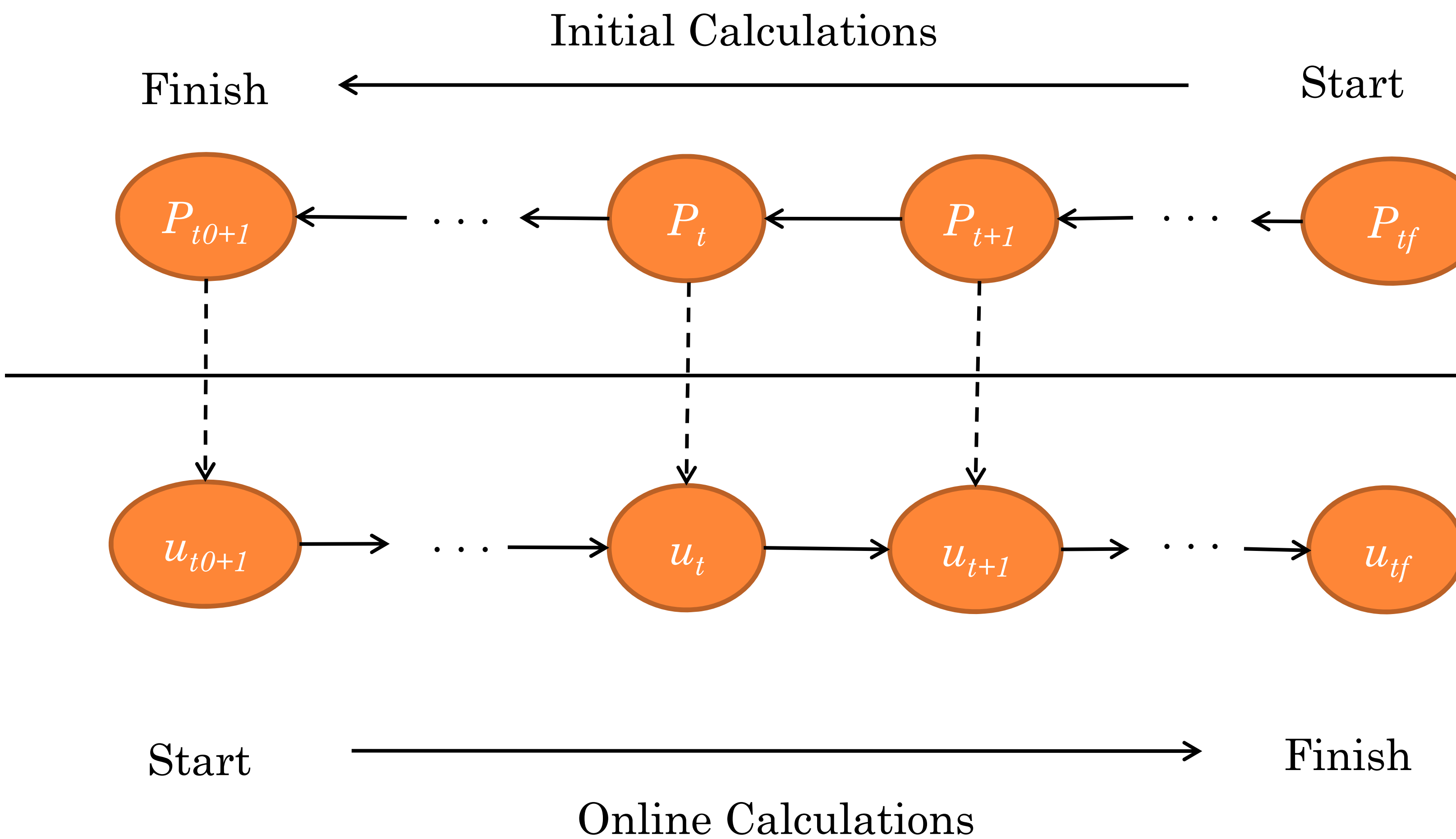
$$K_t = (B_t^T P_t B_t + R)^{-1} B_t^T P_t A_t$$

    - Implement controller at time t using LQR gain and current state

$$u_t = -K_t x_{t-1}$$

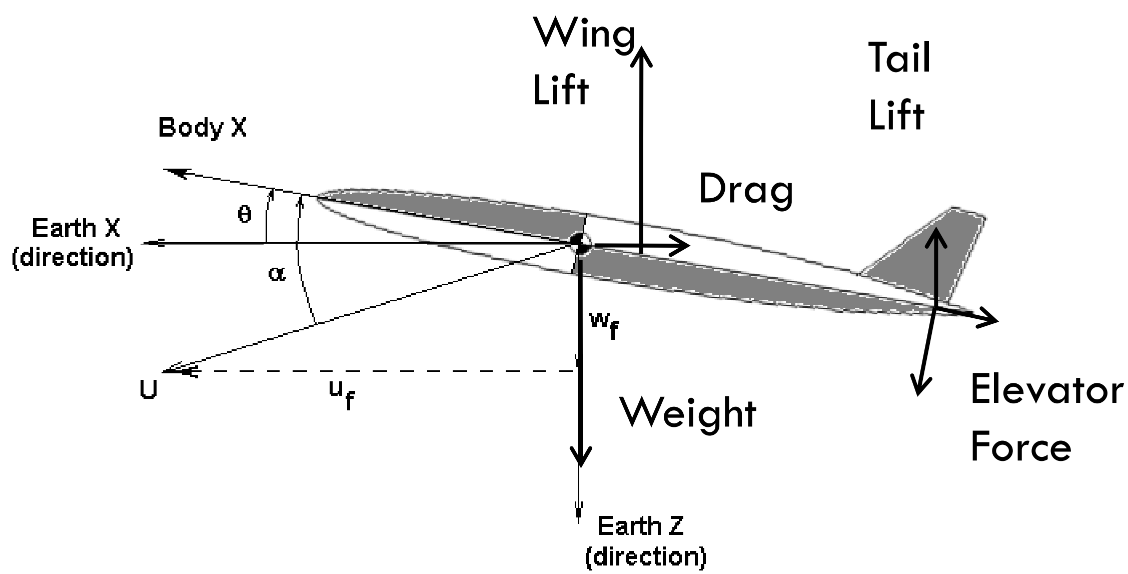
Linear Quadratic Regulator

- Pictorially



LINEAR QUADRATIC REGULATOR

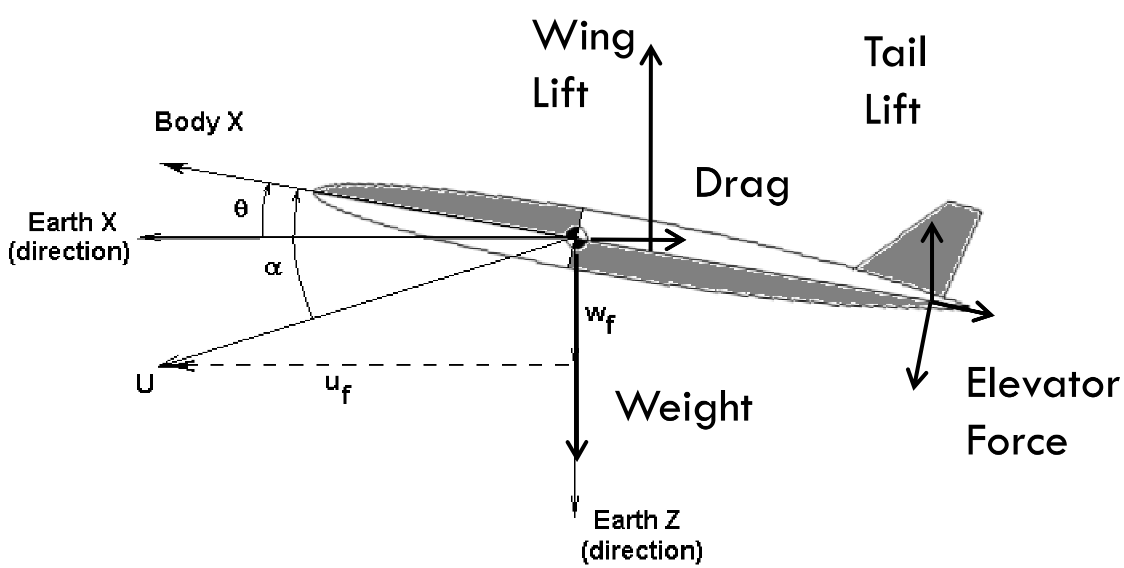
- Example: LQR
  - Linear pitch controller for an aircraft
    - Linearized about constant speed and altitude



Longitudinal Equations of Motion

LINEAR QUADRATIC REGULATOR

- Example: LQR
  - Elevator causes moment about cg
  - Tail resists rotation about cg (damping)
  - Total lift and weight approximately balance
  - Drag increases with elevator deflection



Longitudinal Equations of Motion

Linear Quadratic Regulator

- Example
  - Dynamics
    - State defined as
      - Angle of attack, α
      - Pitch angle, Θ
      - Pitch rate, q
    - Input is elevator deflector, δ
    - If velocity and altitude are held constant, continuous dynamics are

$$\begin{bmatrix} \dot{\alpha} \\ \dot{\theta} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -0.313 & 0 & 56.7 \\ 0 & 0 & 56.7 \\ -0.0139 & 0 & -0.426 \end{bmatrix} \begin{bmatrix} \alpha \\ \theta \\ q \end{bmatrix} + \begin{bmatrix} 0.232 \\ 0 \\ 0.0203 \end{bmatrix} \delta$$

Linear Quadratic Regulator

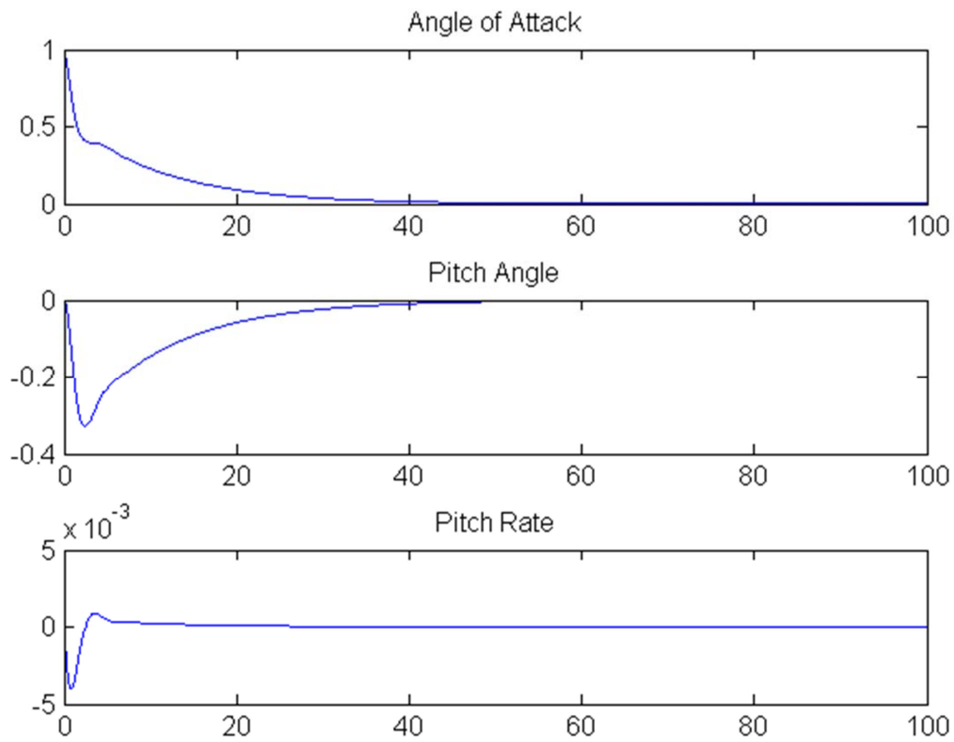
- Example
  - Sample Code (discretized dynamics):

```
% Solve for costate
for t=length(T)-1:-1:1
    P = Q+Ad'*Pn*Ad - Ad'*Pn*Bd*inv(Bd'*Pn*Bd+R)*Bd'*Pn*Ad;
    P_S(:,:,t)=P;
    Pn=P;
end

% Solve for control and simulate
for t=1:length(T)-1
    K = inv(Bd'*P_S(:,:,t+1)*Bd + R)*Bd'*P_S(:,:,t+1)*Ad;
    u(:,t)=-K*x(:,t);
    x(:,t+1) = Ad*x(:,t)+Bd*u(:,t);
end
```

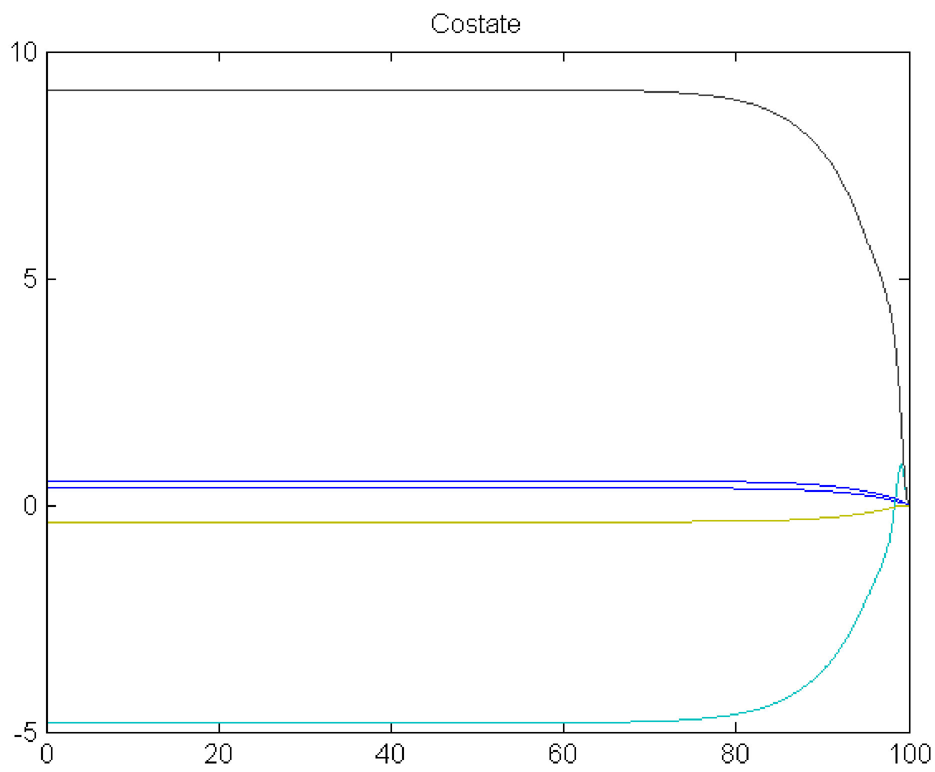
Linear Quadratic Regulator

- Example
  - Cost Matrices, Q, R = I



Linear Quadratic Regulator

- Example
  - Costate values
    - All but (2,3) element for easy viewing



LINEAR QUADRATIC REGULATOR

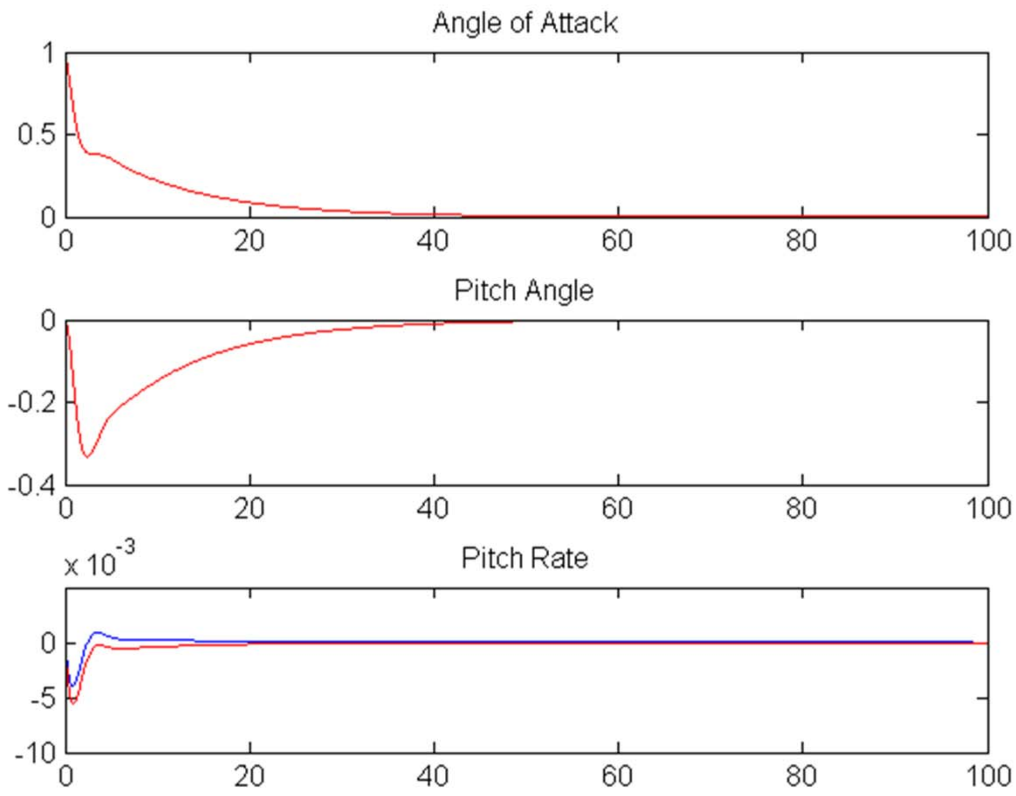
- Steady state linear quadratic regulator (SS LQR)
  - If end goal is far away, steady state solution can be used
    - Almost always the case, infinite horizon formulation

$$P = Q + A^T P A - A^T P B \; (B^T P B + R)^{-1} B^T P A$$

    - Algebraic Ricatti Equation
    - Can be solved two ways
      - Through iteration
        - Set Q_f to Q and run backward in time until convergence
      - Analytically
        - Ask Matlab (lqr(A,B,Q,R))

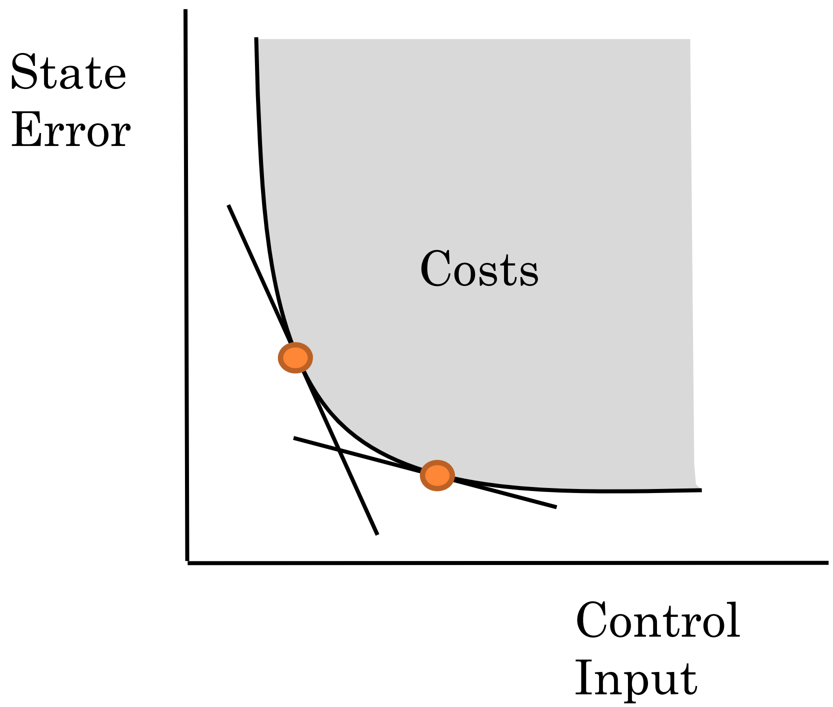
LINEAR QUADRATIC REGULATOR

- Example: SS LQR



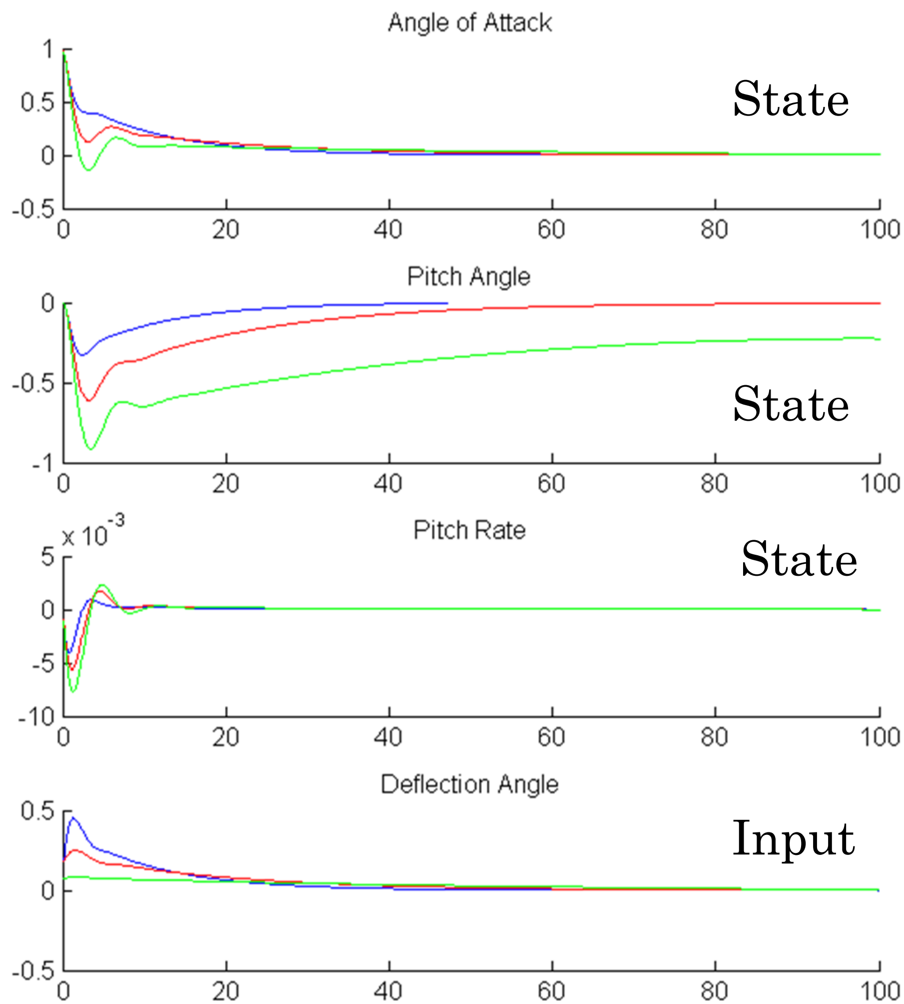
LINEAR QUADRATIC REGULATOR

- Q, R trade off (ignoring terminal condition)
  - Large inputs will drive state to zero more quickly
  - Can define Q, R relative to each other
  - Absolute value defines rate of convergence



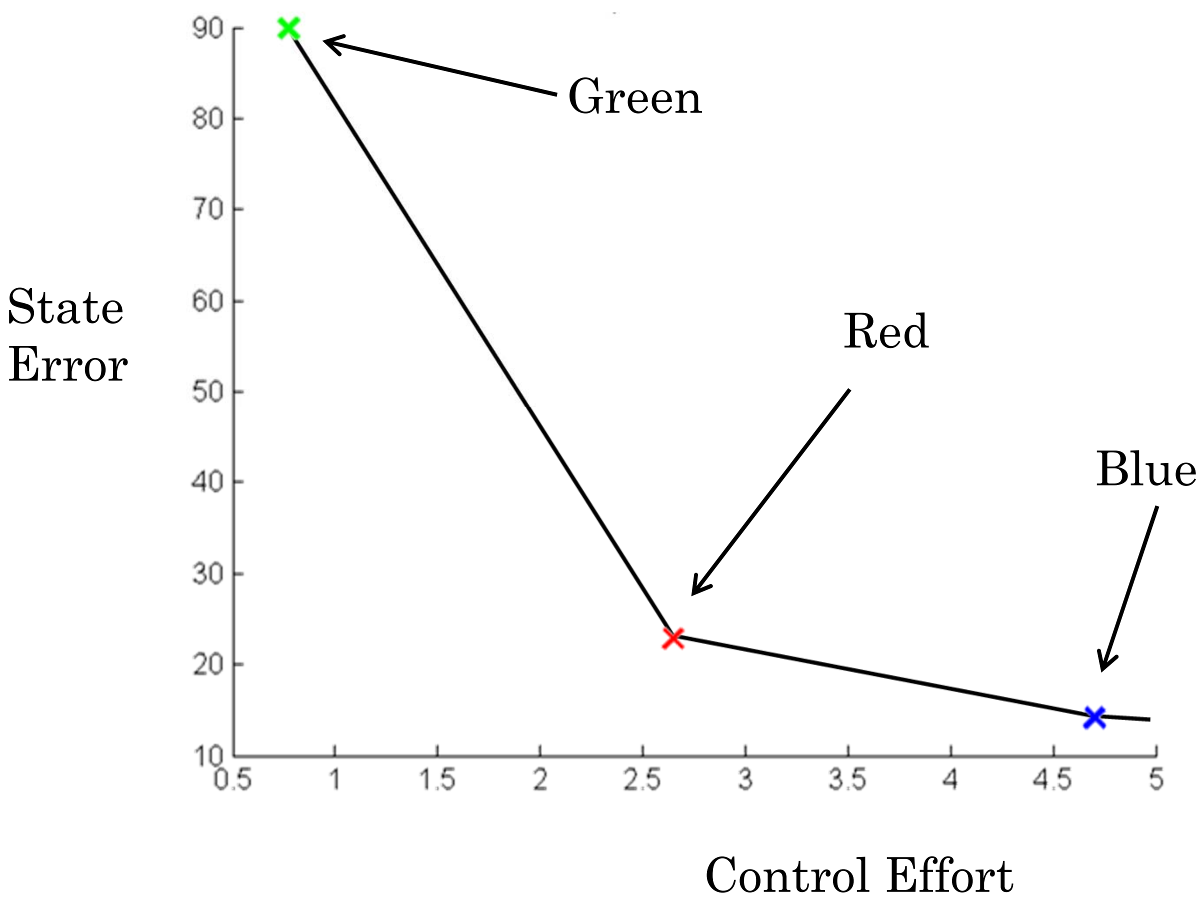
LINEAR QUADRATIC REGULATOR

- Example: LQR Tradeoff
  - Blue
    - Q = 0.01I
    - R = 0.01I
  - Red
    - Q = 0.01I
    - R = 0.1I
  - Green
    - Q = 0.01I
    - R = I



Linear Quadratic Regulator

- Example
  - Comparison of costs from three controllers



Linear Quadratic Regulator

- Stochastic formulation
  - Zero mean additive Gaussian noise has no effect on result
    - Kind of surprising, but very nice

- Separation of Estimation and Control
  - Can be proven to be optimal solution
  - Linear Quadratic Gaussian controller
    - LQR Combined with Kalman Filter
    - LQR uses mean of Kalman belief as current state estimate

Linear Quadratic Tracking

- Tracking
  - LQR control used with state and input offsets
    - Includes LQR regulation to non-zero quantities
  - Desired trajectory can be defined by inputs

$$\pi^t = \{\{x_{t_0}^t, u_{t_{0+1}}^t\}, \ldots, \{x_{t_f-1}^t, u_{t_f}^t\}\}$$

  - State and input deviations used in LQR

$$\delta x_t = x_t - x_t^t, \quad \delta u_t = u_t - u_t^t$$

  - Dynamics are the same, and control is now $u_t^t + \delta u_t$

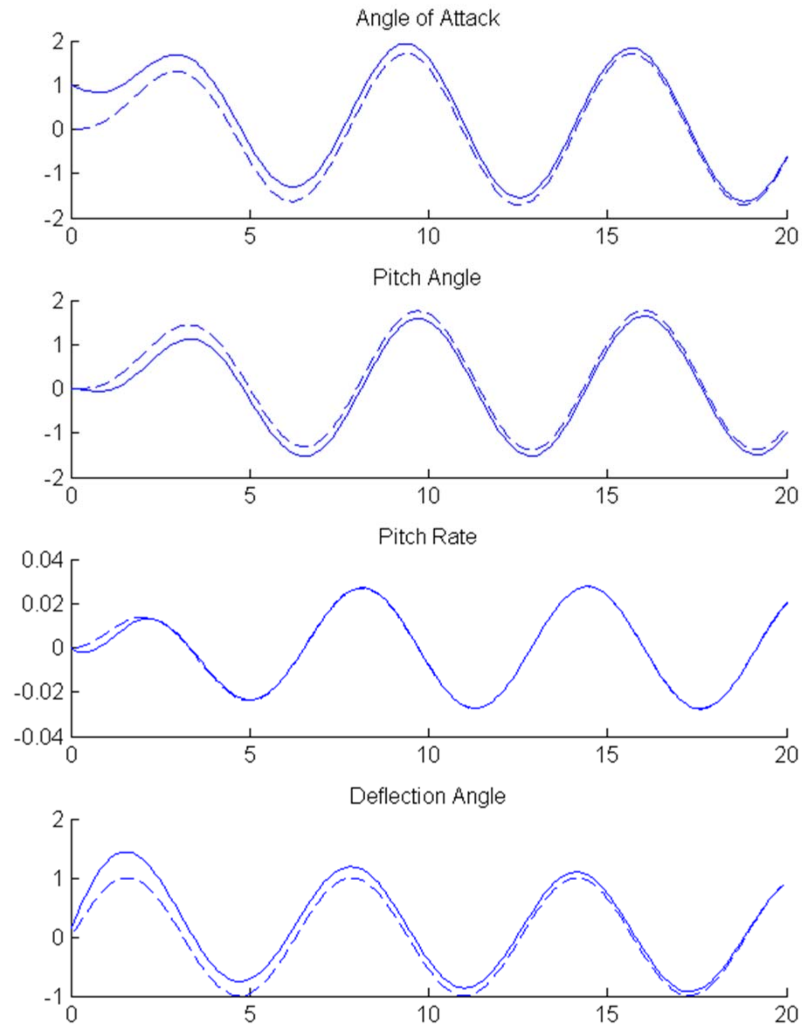
$$x_t = A_t x_{t-1} + B_t u_t$$

$$-x_t^t = A_t x_{t-1}^t + B_t u_t^t$$

$$\delta x_t = A_t \delta x_{t-1} + B_t \delta u_t$$

Linear Quadratic Tracking

- Example: LQR Tracking
  - Sinusoidal variation
    - Trajectory driven by desired control input selection
    - Initial angle of attack error of 1 degree
    - Tracking achieved on identical timescale to LQR
  - Hardest part is defining desired trajectory
  - Example of superposition



Outline

- Control Structures
- Linear Motion Models
  - PID Control
  - Linear Quadratic Regulator
  - Tracking
- Nonlinear Motion Models
  - Description of main methods
  - Geometric driving controller

NONLINEAR CONTROL

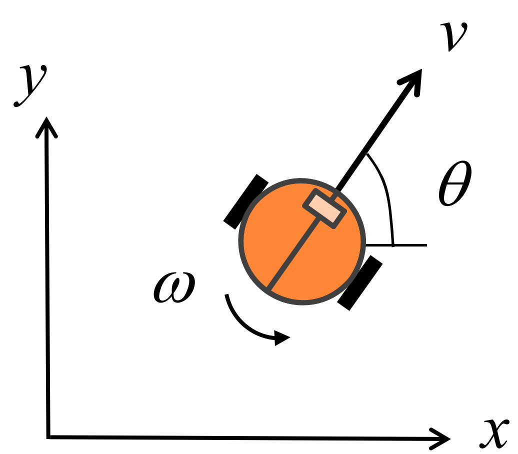
- A field dominated by continuous time domain
  - Nonlinear systems (ECE 688)
- Consider continuous nonlinear dynamics without disturbances

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

- Rely on timescale assumption
  - Discrete output commands occur much more quickly than variation in system dynamics
  - Estimation also fast enough and accurate enough to ignore

NONLINEAR CONTROL

- Let's take a test case
  - Two wheeled robot



$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix}$$

$$\Downarrow$$

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

$$\Downarrow$$

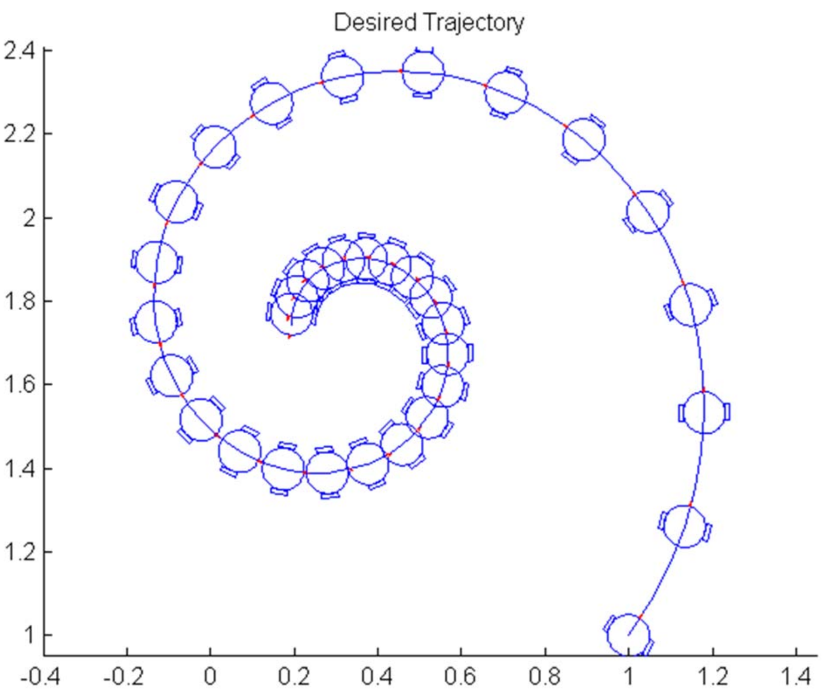
$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} u_1\cos x_3 \\ u_1\sin x_3 \\ u_2 \end{bmatrix}$$

NONLINEAR CONTROL

- Desired trajectory
  - Selected to have same dynamics as system
  - Specify desired inputs, and path results

$$\dot{x}^t = \begin{bmatrix} u_1^t \cos x_3^t \\ u_1^t \sin x_3^t \\ u_2^d \end{bmatrix}$$

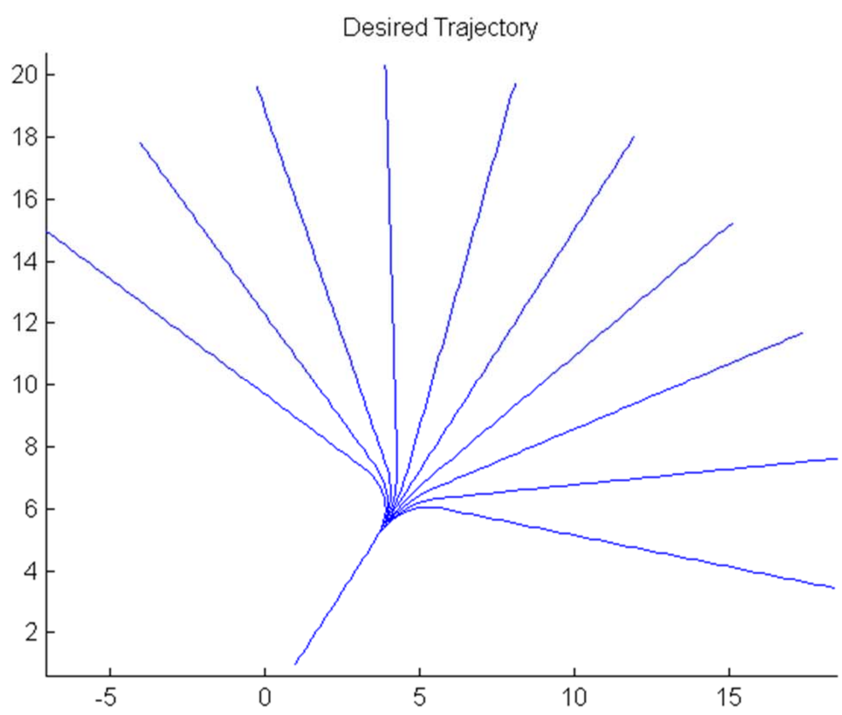
$$u^t = \begin{bmatrix} e^{-0.2t} \\ 1 \end{bmatrix}$$



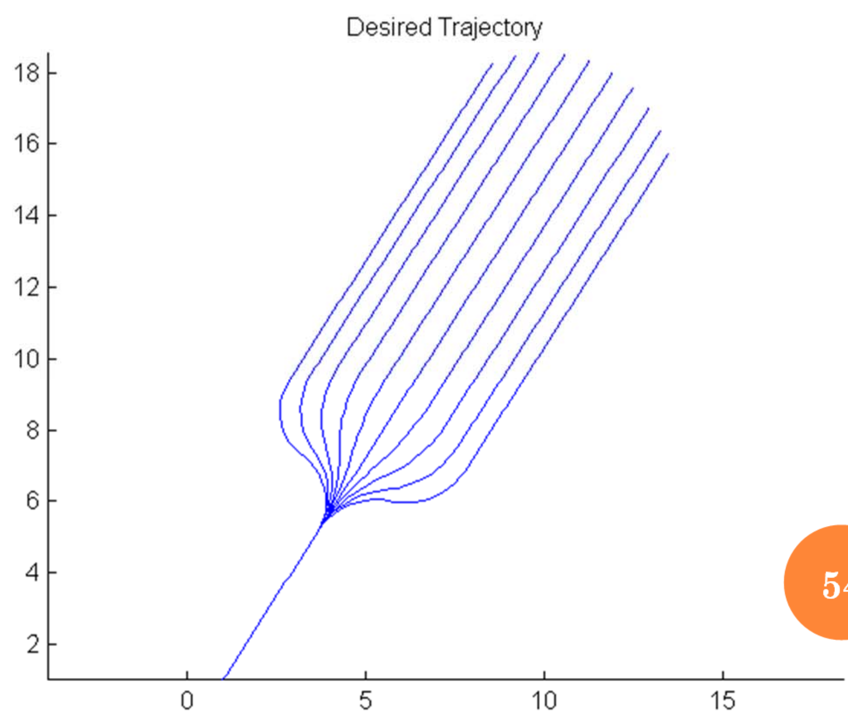
Nonlinear Control

- Desired trajectory as Motion Primitive
  - Can be used to generate a family of trajectories that can be used to reduce planning problem

Curved Trajectory



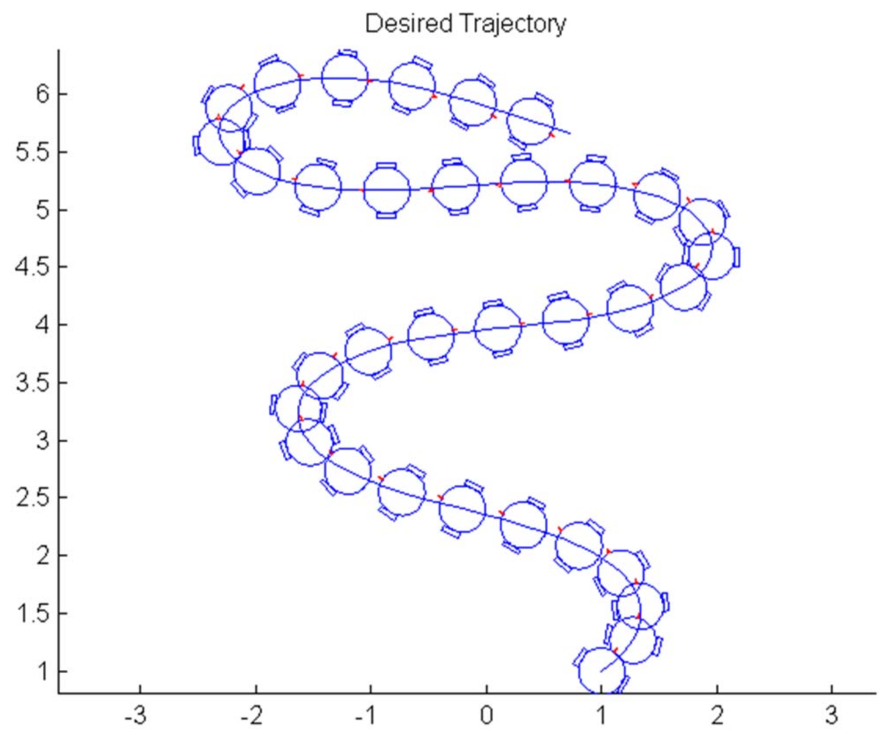
Swerve Trajectory



Nonlinear Control

- Desired trajectory
  - Track arbitrary nonlinear curve
  - Specify desired states, and control must be determined

$$\dot{x}^t = \begin{bmatrix} 2\cos x_3^t \\ \sin x_3^t \\ x_1^t \end{bmatrix}$$



  - Careful: example violates forward motion constraint
    - Not possible to track exactly

Nonlinear Control

- Option 1: Feedback Linearization
  - If motion is of the form

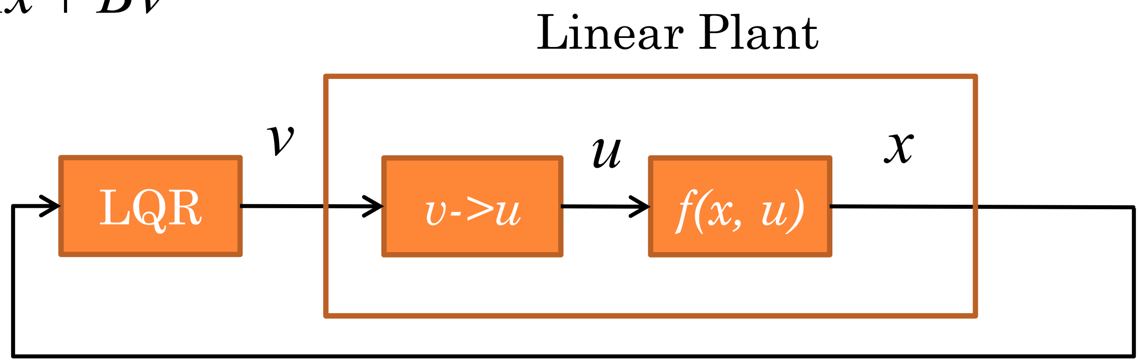
$$\dot{x} = f(x) + g(x)u$$

    - It is sometimes possible to find a controller which makes the map from v and x to dx/dt linear

$$u = a(x) + b(x)v \qquad f(x) + g(x)a(x) = Ax$$

$$\dot{x} = f(x) + g(x)\big(a(x) + b(x)v\big) \qquad g(x)b(x)v = Bv$$

$$= Ax + Bv$$



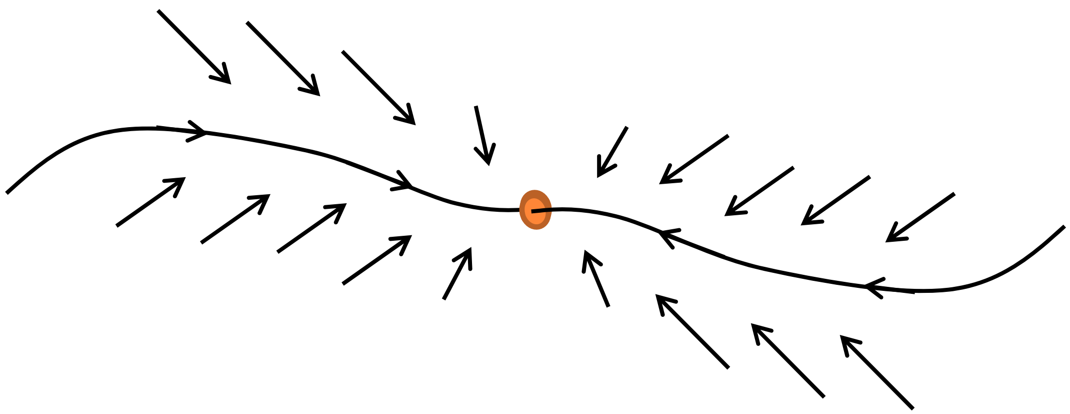
  - Not possible for two-wheeled robot

Nonlinear Control

- Option 2: Backstepping control
  - If we have a feedback linearizable system for which the inversion results in large inputs, can elect to leave some of the nonlinearity in the plant
  - If a control is known for a subsystem of derivative terms, then a controller for the full system can be developed one derivative at a time
  - Relies on Lyapunov stability argument to construct each successive controller and ensure stability
    - Not always easy to do!
- Not possible for two-wheeled robot

NONLINEAR CONTROL

- Option 3: Sliding Mode Control
  - If a trajectory is known to converge to a desired equilibrium, regulation is possible
  - Find a control law that drives the system to the trajectory
  - Follow the trajectory to the equilibrium



  - Is possible for two-wheeled robot
  - Issues relating to control chattering can be addressed

Nonlinear Control

- Many nonlinear control methods exist
  - Can work very well if the system is of the right form
  - Usually rely on knowing dynamics and derivatives exactly
  - Smooth derivatives required
  - Modeling issues, robustness of inversion
  - In practice, each nonlinear system is analyzed individually

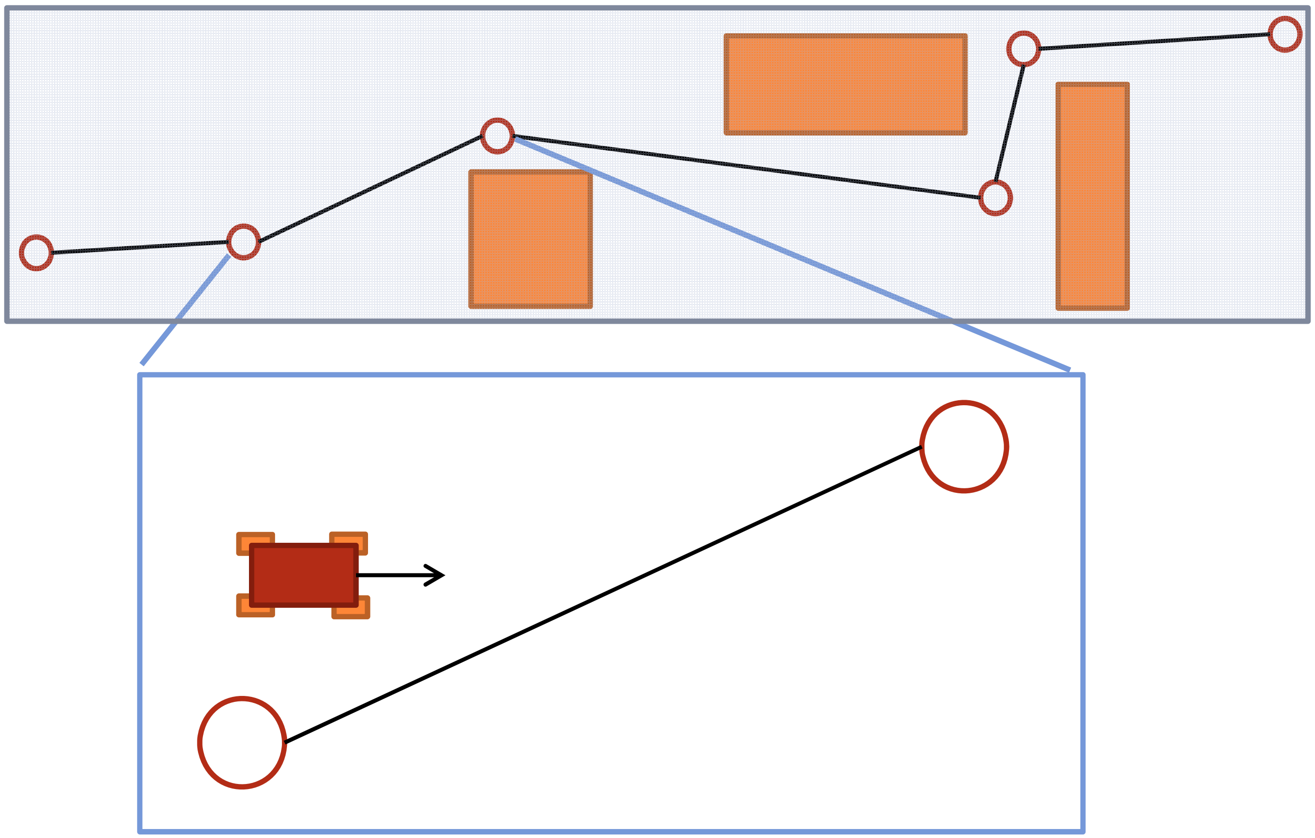
- Continue with ground vehicle example
  - Slightly more complicated kinematics

Driving Controller

- Motion Control for an automobile
  - Define error dynamics relative to desired path
  - Select a control law that drives errors to zero and satisfies input constraints
  - Prove stability of controller
  - Add dynamic considerations to manage unmodeled effects

DRIVING CONTROLLER

- Goal of controller
  - To track straight line trajectories
    - from one waypoint to the next
    - Also works on corners, smooth paths

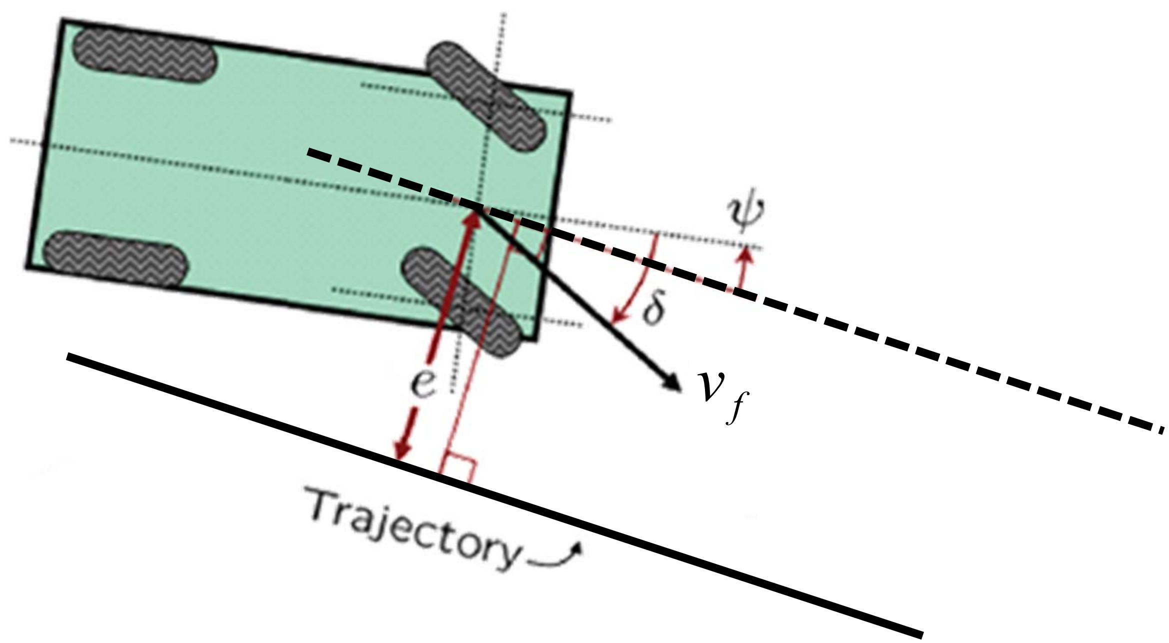


Driving Controller

- Approach
  - Look at both the error in heading and the error in position relative to the closest point on the path
    - Perpendicular distance for straight line segments
    - Can become ambiguous for curves, usually well defined
  - Use the center of the front axle as a reference point
  - Define an intuitive steering law to
    - Correct heading error
    - Correct position error
    - Obey max steering angle bounds

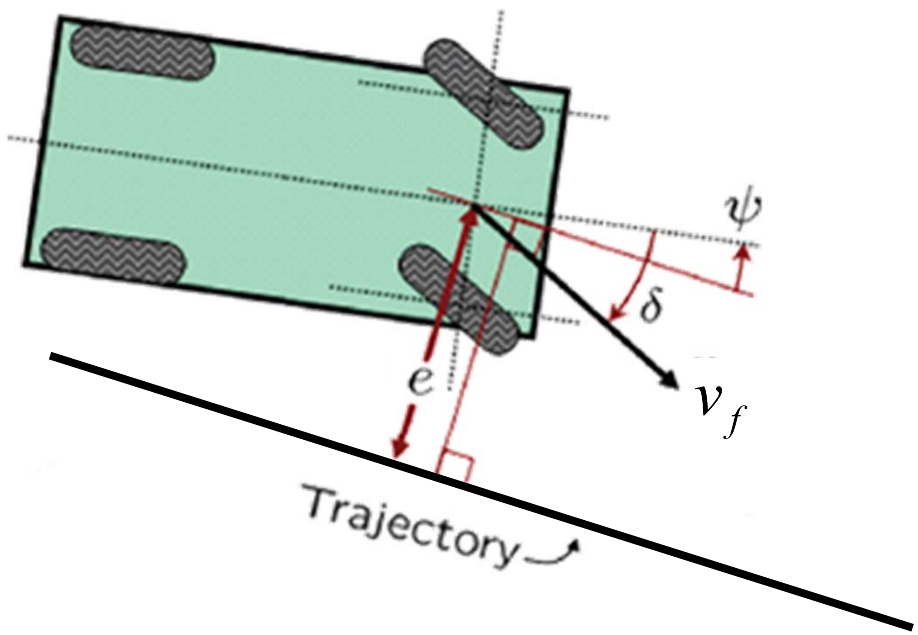
Driving Controller

- Description of vehicle
  - All state variables and inputs defined relative to center point of front axle
  - Steering relative to heading (in opposite direction): δ
  - Velocity in direction of front wheels: v_f
  - Heading relative to trajectory: ψ



Driving Controller

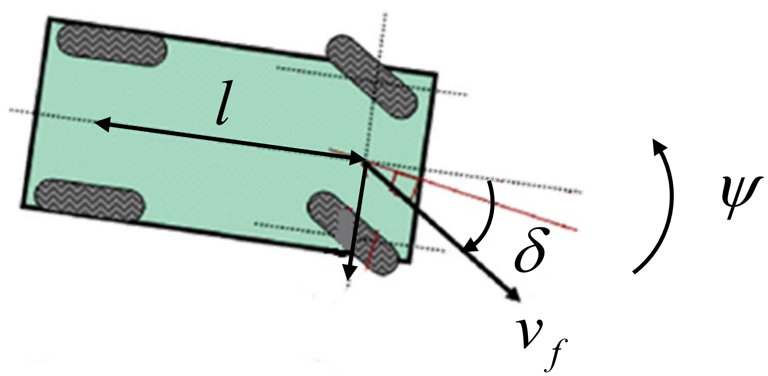
- Description of vehicle
  - Crosstrack error: e
    - Distance from center of front axle to closest point on trajectory



Driving Controller

- Error Dynamics
  - Heading error
    - Rotation about rear wheel center point (ICR, again)
    - Component of velocity perpendicular to trajectory
    - Desired heading is 0

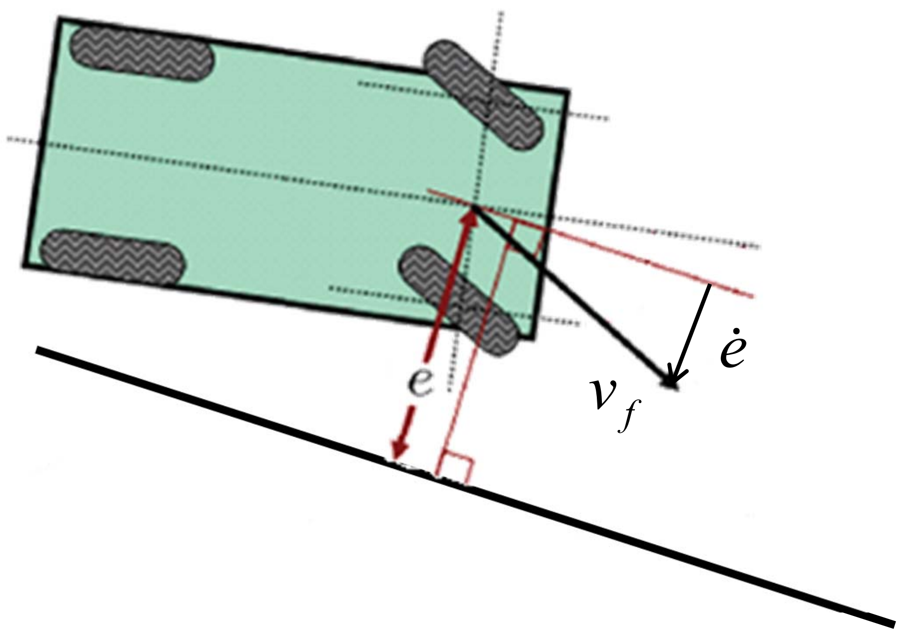
$$\dot{\psi}(t) = \frac{-v_f(t)\sin(\delta(t))}{l}$$



DRIVING CONTROLLER

- Error Dynamics
  - Rate of change of cross track error
    - Component of velocity perpendicular to trajectory

$$\dot{e}(t) = v_f(t)\sin(\psi(t) - \delta(t))$$



Driving Controller

- Proposed heading control law
  - Combine three requirements
    - Steer to align heading with desired heading
      - Proportional to heading error

$$\delta(t) = \psi(t)$$

    - Steer to eliminate crosstrack error
      - Also essentially proportional to error
      - Inversely proportional to speed
      - Gain *k* determined experimentally
      - Limit effect for large errors with inverse tan

$$\delta(t) = \tan^{-1}\left(\frac{ke(t)}{v_f(t)}\right)$$

    - Maximum and minimum steering angles

$$\delta(t) \in [\delta_{\min}, \delta_{\max}]$$

DRIVING CONTROLLER

- Combined steering law

$$\delta(t) = \psi(t) + \tan^{-1}\left(\frac{ke(t)}{v_f(t)}\right) \qquad \delta(t) \in [\delta_{\min}, \delta_{\max}]$$

  - For large heading error, steer in opposite direction
    - The larger the heading error, the larger the steering correction

DRIVING CONTROLLER

- Combined steering law

$$\delta(t) = \psi(t) + \tan^{-1}\left(\frac{ke(t)}{v_f(t)}\right) \qquad \delta(t) \in [\delta_{\min}, \delta_{\max}]$$

  - For large positive crosstrack error

$$\tan^{-1}\left(\frac{ke(t)}{v_f(t)}\right) \approx \frac{\pi}{2} \implies \delta(t) \approx \psi(t) + \frac{\pi}{2}$$

    - The larger the crosstrack error, the larger the steering angle required by this part of the control
    - As heading changes due to steering angle, the heading correction counteracts the crosstrack correction, and drives the steering angle back to zero

Driving Controller

- Combined steering law
  - The error dynamics when not at maximum steering angle are

$$\dot{e}(t) = -v_f(t)\sin(\psi(t) - \delta(t))$$

$$= -v_f(t)\sin\left(\tan^{-1}\left(\frac{ke(t)}{v_f(t)}\right)\right)$$

$$= \frac{-ke(t)}{\sqrt{1+\left(\frac{ke(t)}{v_f}\right)^2}}$$

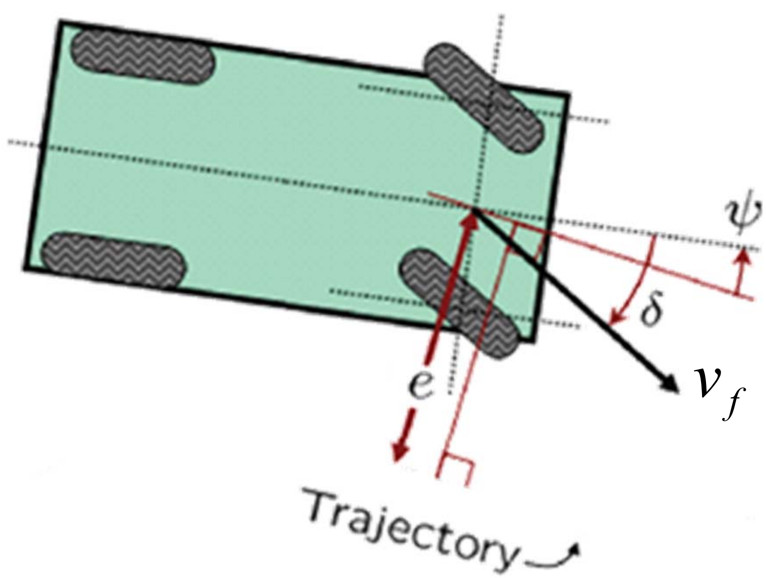
  - For small crosstrack errors

$$\dot{e}(t) \approx -ke(t)$$

    - Exponential decay of error

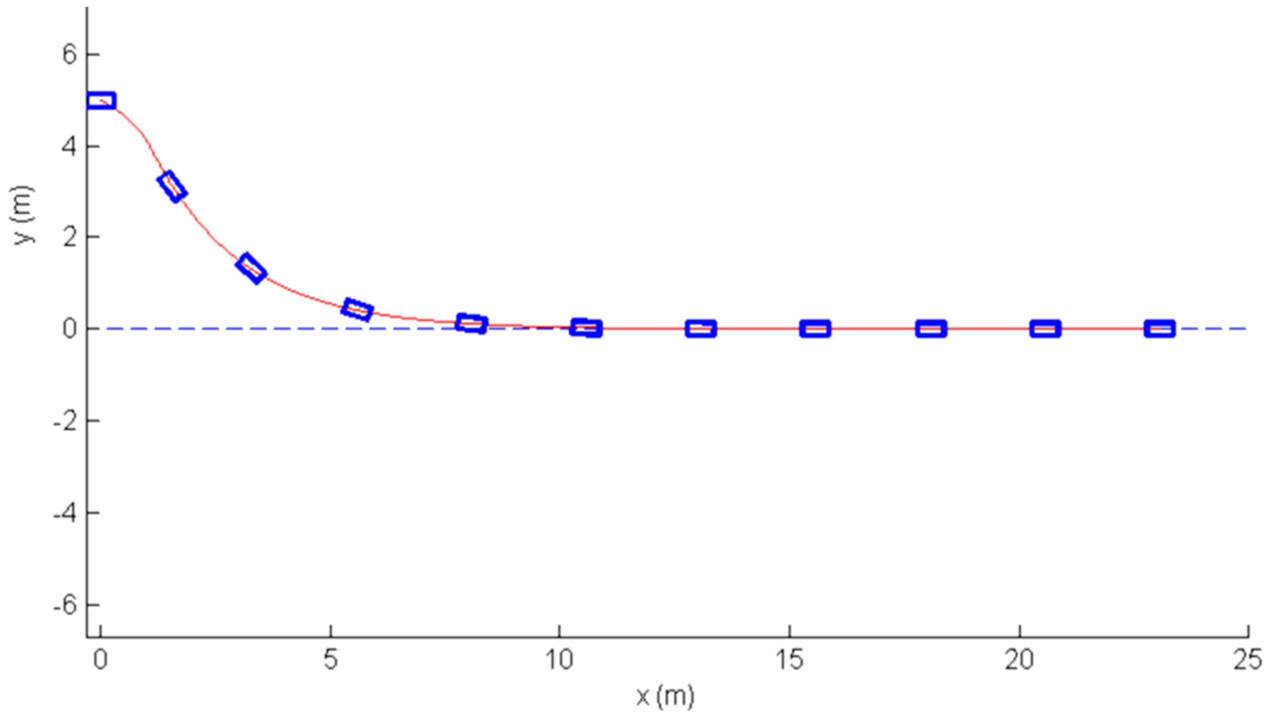
Driving Controller

- Example code
  - Implement the error dynamics directly.
  - Explore various initial conditions to understand how the controller works.
  - Add in noise/disturbances and assess how the controller reacts.



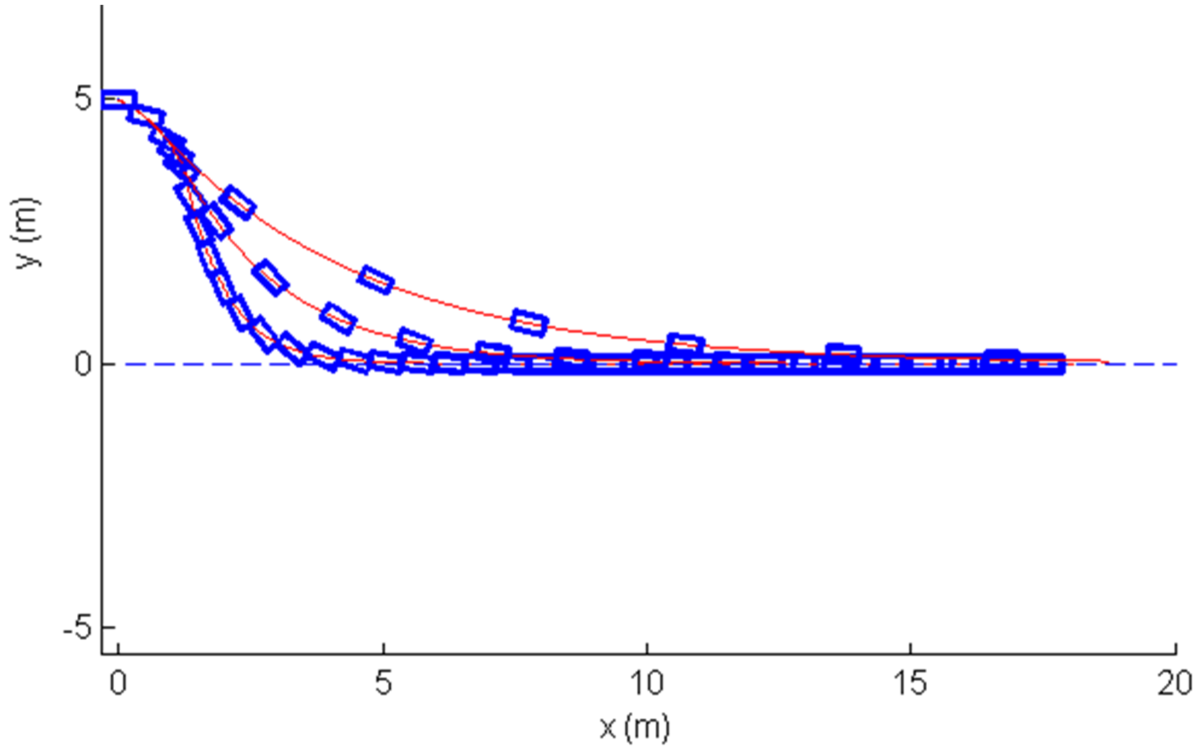
DRIVING CONTROLLER

- Example – Large initial crosstrack error
  - Crosstrack error of 5 meters
    - Max steer 25°, speed 5 m/s
    - Gain k = 2.5, Length l = 1 m



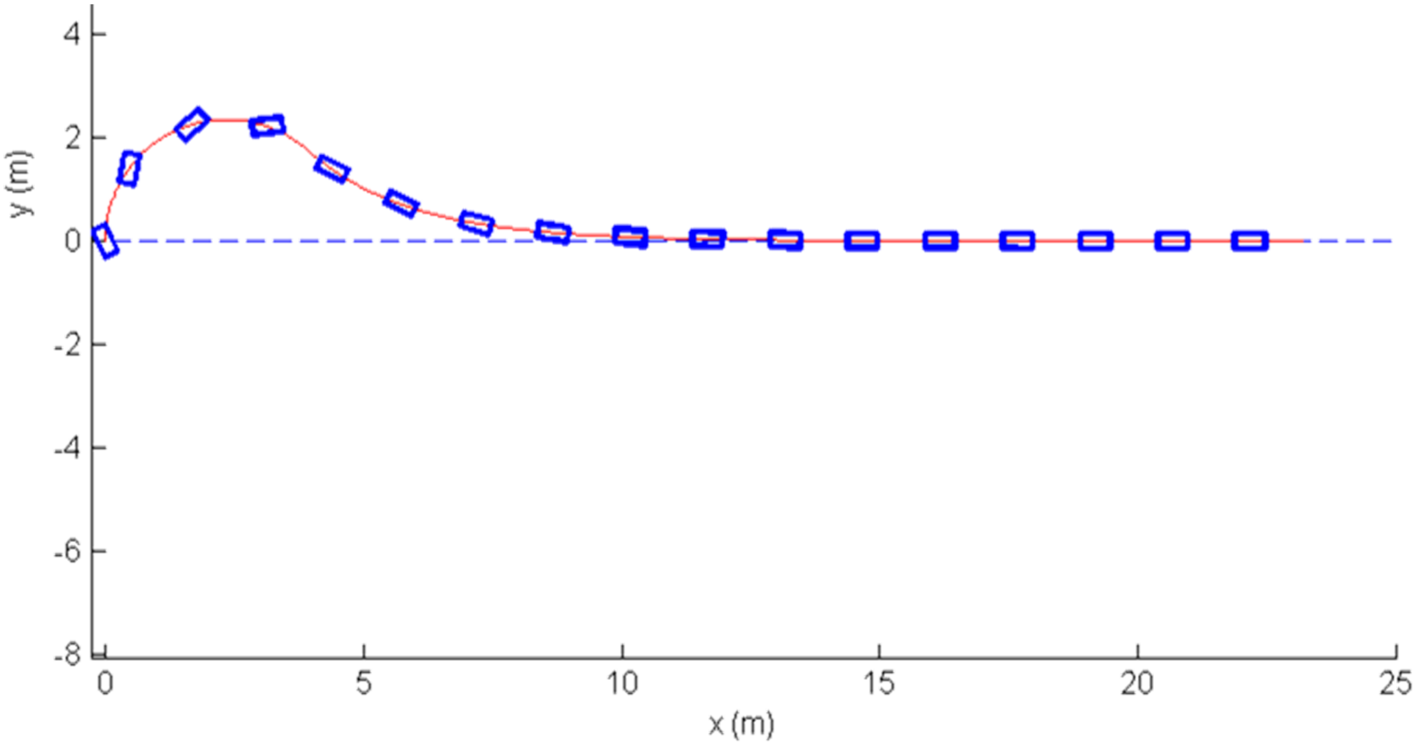
Driving Controller

- Example – Effect of speed variation
  - Crosstrack error of 5 meters
    - Speeds 2, 5, 10 m/s



DRIVING CONTROLLER

- Example – Large Error in Heading
  - Max steer 25°, speed 5 m/s
  - Gain k = 2.5, Length l = 1 m



Driving Controller

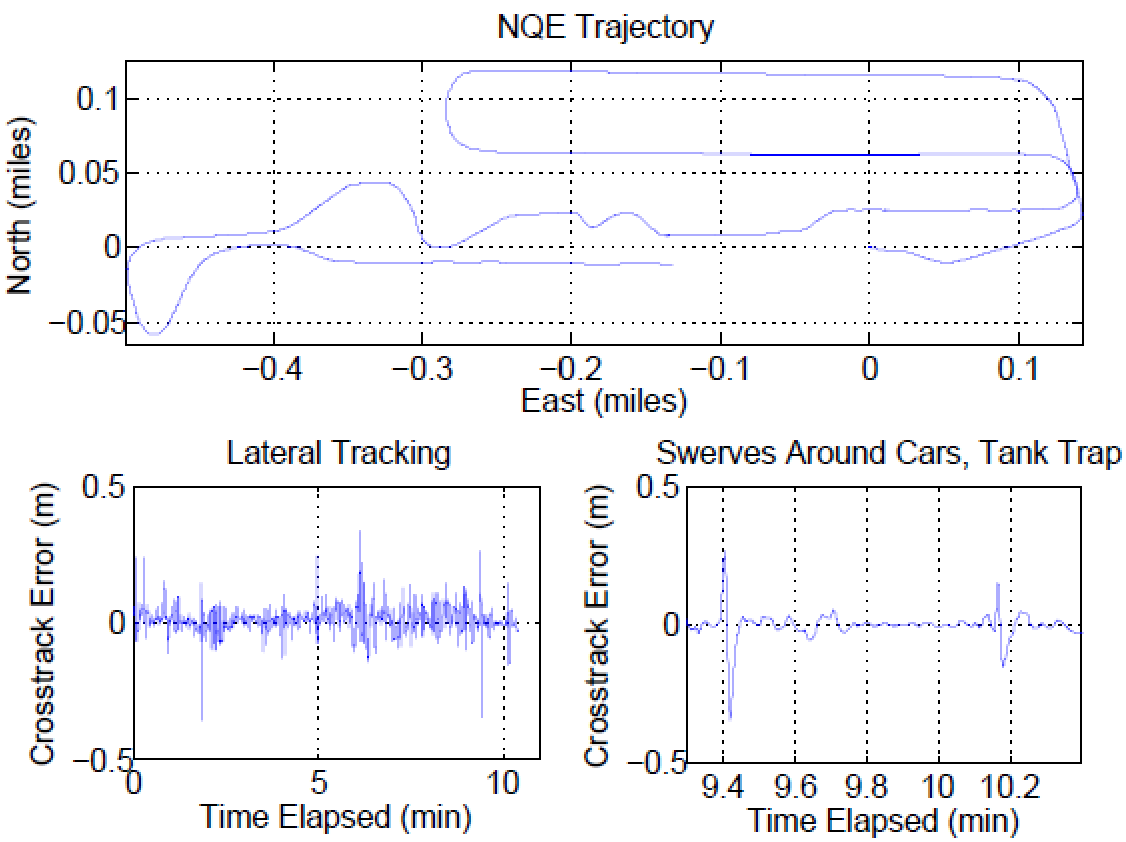
- Adjustments
  - Low speed operation
    - Inverse speed can cause numerical instability
    - Add softening constant to controller

$$\delta(t) = \psi(t) + \tan^{-1}\left(\frac{ke(t)}{k_s + v_f(t)}\right)$$

  - Extra damping on heading
    - Becomes an issue at higher speeds in real vehicle
  - Steer into constant radius curves
    - Improves tracking on curves by adding a feedforward term on heading

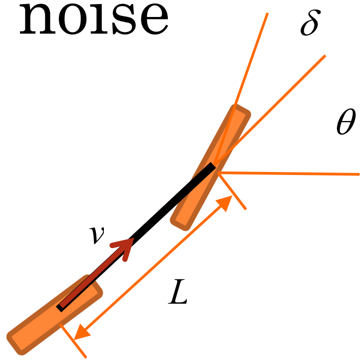
DRIVING CONTROLLER

- Results
  - National Qualifying event



Exercise – Challenge Problem

- Create a simulation of bicycle model with noise on steering angle and speed inputs



- Add Stanley controller

$$\delta(t) = \psi(t) + \tan^{-1}\left(\frac{ke(t)}{v_f(t)}\right)$$

$$\delta(t) \in [\delta_{\min}, \delta_{\max}]$$

- Experiment with low speed and damping issues

$$\delta(t) = \psi(t) + \tan^{-1}\left(\frac{ke(t)}{k_s + v_f(t)}\right)$$

- Identify feedforward term for tracking curves

Extra Slides

NONLINEAR CONTROL

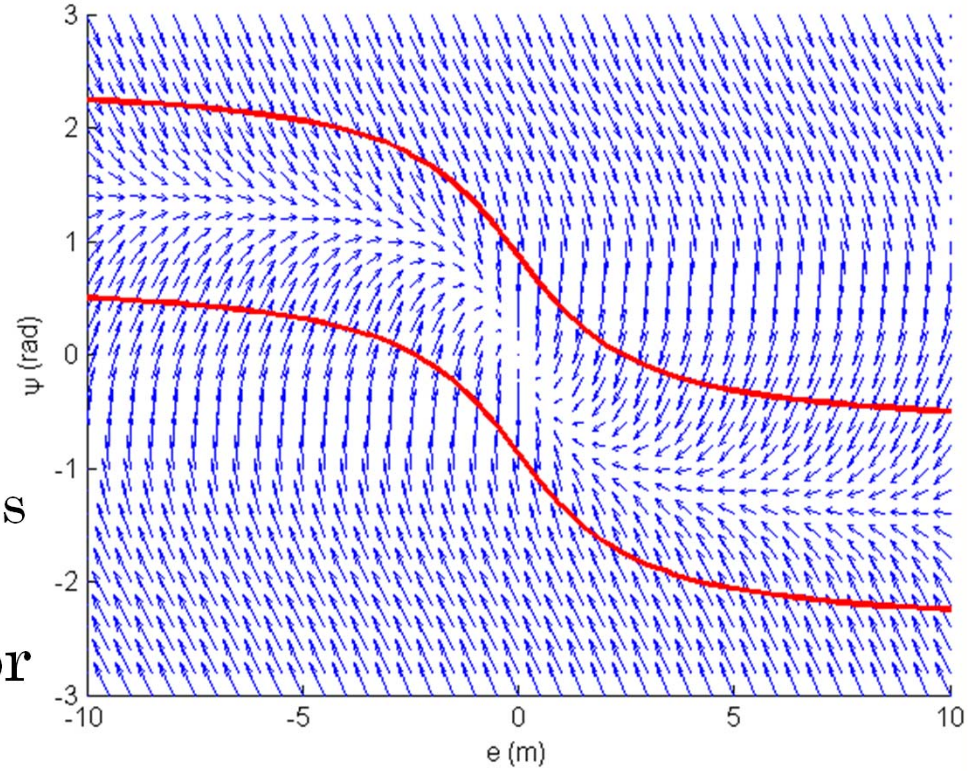
- Option 1: Linearize about current state, control and apply LQR
  - "Extended Linear Quadratic Regulator"

$$A_t = \frac{\partial f}{\partial x} = \begin{bmatrix} 0 & 0 & -v\sin(\theta)\omega \\ 0 & 0 & v\cos(\theta)\omega \\ 0 & 0 & 0 \end{bmatrix} \qquad B_t = \frac{\partial f}{\partial u} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}$$

  - Both matrices linearized about current control inputs, but are used to find the control to apply
  - Therefore, must iterate solution to be linearizing about correct point
    - Inefficient, poor convergence

DRIVING CONTROLLER

- Phase portrait
  - $v_f = 5$ m/s, k = 2.5, $l = 1$ m
  - Allows comparison of crosstrack and heading error evolution
  - Arrows represent derivatives of axes
  - Red lines are boundaries of regions
- All arrows enter interior
- Only one equilibrium
- Crosstrack error decreasing in interior



Driving Controller

- Global Convergence Proof
  - Split into three regions
    - Max steering angle
    - Min steering angle
    - Interior
  - Show trajectory always exits min/max regions
  - Show unique equilibrium exists at origin
  - Show interior dynamics always strictly decrease crosstrack error magnitude
  - Show that heading converges to crosstrack error
  - Show that if trajectory exits interior and enters min/max regions, it returns to interior with smaller errors

Driving Controller

- Velocity control law
  - PI control to match planner speed recommendations
    - Curve limitations
      - Side force constraints to avoid wheel slip
    - Terrain knowledge
  - Combined command of brake and throttle
    - Brake cylinder pressure command
    - Throttle position command
    - Susceptible to chatter
  - More interesting problem: deciding what speed to drive