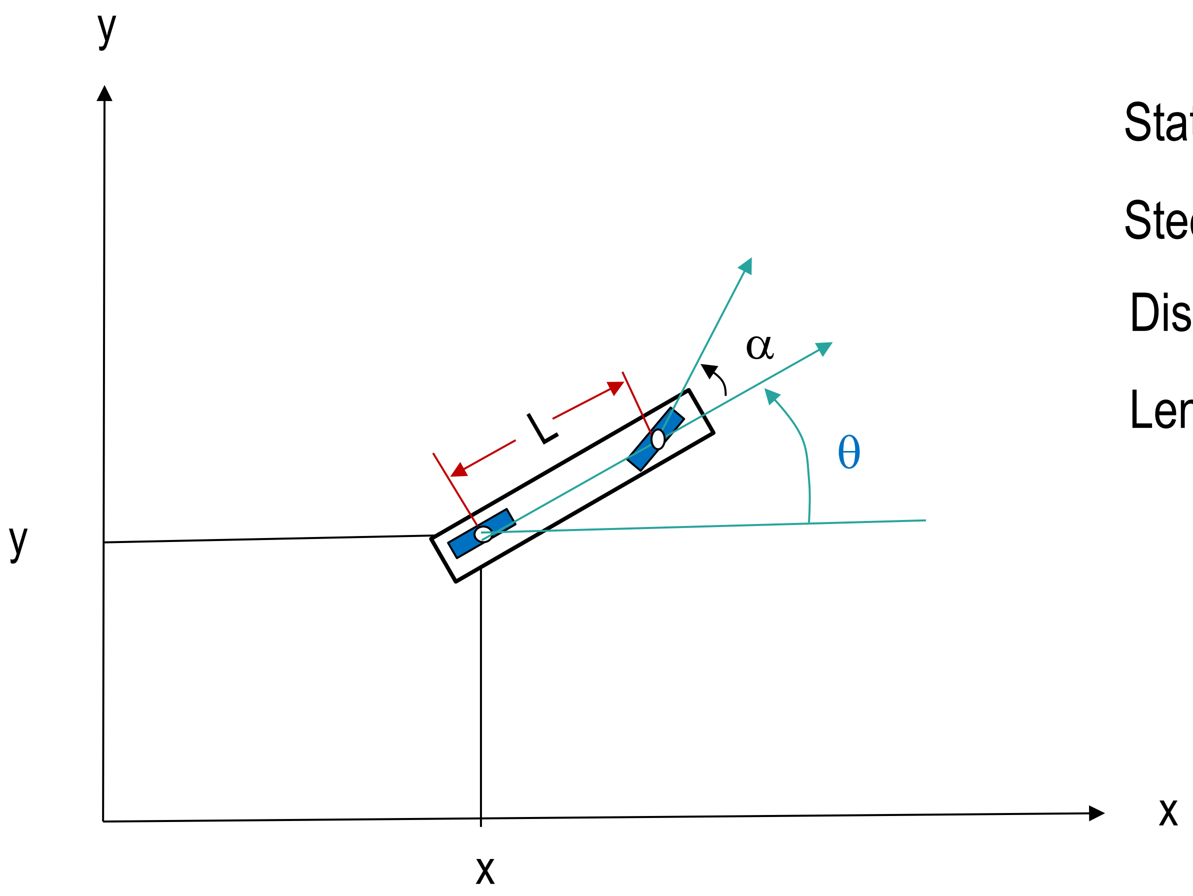
Mobile Robotics

Robot Motion: PID Control

Programming Assignments and Projects

Recall: Bicycle Model

- The bicycle robot shown below will be used in this project



State: (x, y, θ)

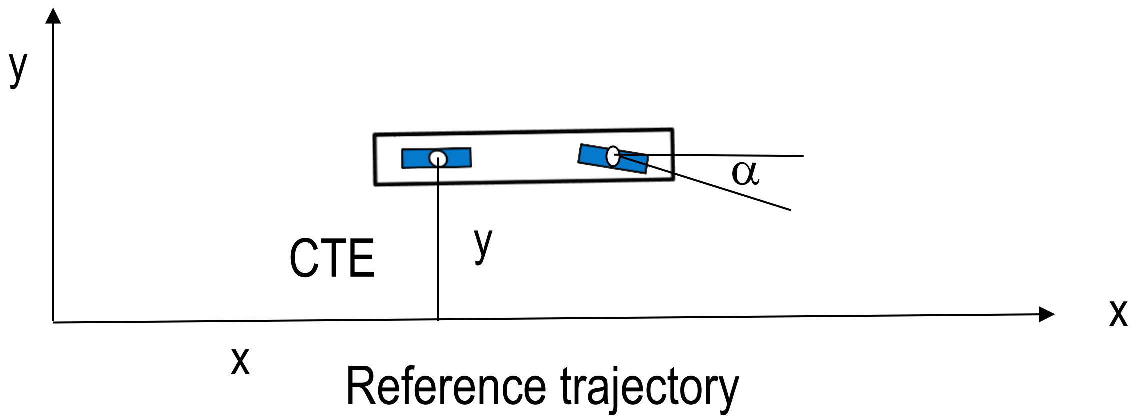
Steering angle : α

Distance: d

Length: L

PA5A: P (Proportional) Control

- Reference trajectory and Cross track error (CTE)

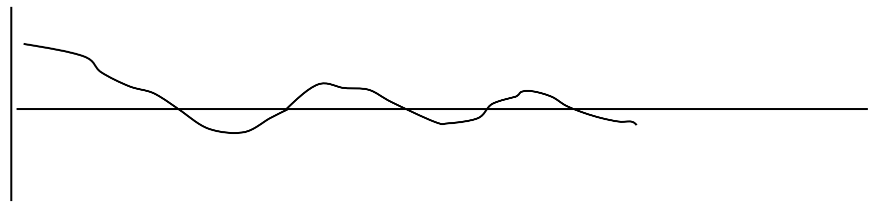


State: (x, y, θ)

Steering angle : α

Distance: d

- To steer in proportional to CTE: $\alpha = -\tau * CTE$
- Problem: Overshoot and oscillation



Marginally stable

- What will happen? When
  - τ is bigger

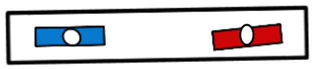
PA5B : PD (Proportional and Differential) Control

- To steer in proportional to CTE and difference of CTE:

$$\alpha = -\tau_P * CTE - \tau_D * \frac{d}{dt} CTE$$

$$\frac{d}{dt} CTE = \frac{CTE_t - CTE_{t-1}}{\Delta t} \qquad \Delta t = 1$$

- Problem
  - Systematic Bias causes big CTE



  - P-term and D-term cannot solve this problem

PA5C : PID Control

- PID means Proportional, Integral, and Differential
- To steer in proportional to CTE and integral and difference of CTE:

$$\alpha = -\tau_P * CTE - \tau_D * \frac{d}{dt}CTE - \tau_I * \int CTF$$

$$\frac{d}{dt}CTE = \frac{CTE_t - CTE_{t-1}}{\Delta t} \qquad \Delta t\text{=1}$$

$$\int CTE = \sum CTE_t * \Delta t \qquad \Delta t\text{=1}$$

- Question: how can we find good control gains τ_P, τ_D, τ_I?
- The answer is called Twiddle or Coordinate ascent

Twiddle or Coordinate Ascent Algorithm

- "Twiddle" refers to an optimization algorithm that is used to fine-tune the parameters of a system or a model. It is a heuristic algorithm that is often used in control theory and machine learning. The basic idea of twiddle is to iteratively adjust the parameters of the system or model until a satisfactory result is achieved. At each iteration, the algorithm tweaks the parameters and measures the effect on the system's output. If the output is improved, the algorithm keeps the new parameter values; otherwise, it reverts to the old parameter values and tries a different set of adjustments.
- "Coordinate ascent algorithm" is a type of optimization algorithm that is often used in machine learning, particularly in the context of reinforcement learning. The basic idea of coordinate ascent is to optimize a function by successively maximizing it with respect to each of its variables, while holding the other variables fixed. The algorithm iteratively updates the value of one variable, then fixes it and updates the value of another variable, and so on, until convergence is achieved.
- Note: the above two points are stolen from ChatGPT.

Twiddle Algorithm for PID Controller

- Initialize the PID controller with some initial parameter values
- Set the twiddle factors (values) and calculate initial error as the best error so far
- Loop until the sum of the twiddle values is less than the tolerance
  - Try each parameter in turn
    - Add the twiddle factor to the parameter value and calculate the new error.
    - If the new error is less than the best error so far, update the best error and increase the twiddle factor for that parameter by 10%.
    - else subtract twice the twiddle factor from the parameter value, and calculate the new error
      - If the new error is less than the best error so far, update the best error and increase the twiddle factor for that parameter by 10%.
      - else reset the parameter value to its original value and decrease the twiddle factor for that parameter by 10%.

PA5D : Implement Twiddle Function

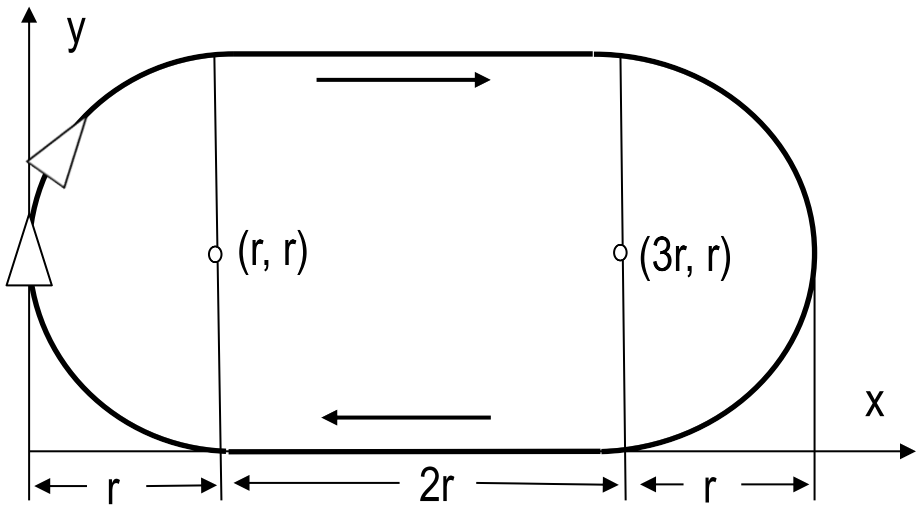
- The run(p) function in PA5C has three parameters.
- Twiddle algorithm can generate a set of optimized parameters p so that the function run(p) will minimize the average CTE.

```
p = [0, 0, 0]
dp = [1, 1, 1]
best_err = run(p)
while sum(dp) > tol:
    for i in range(len(p)):
        p[i] += dp[i]
        err = run(p)
        if err < best_err:
            best_err = err
            dp[i] *= 1.1
        else:
            p[i] -= 2*dp[i]
            err = run(p)
            if err < best_err:
                best_err = err
                dp[i] *= 1.1
            else:
                p[i] += dp[i]
                dp[i] *= 0.9

return p
```

PP5A: Racetrack Control

- Drive a robot along the following racetrack
- The robot starts at (0, r) and faces north (pi/2)
- A run function is provided to drive the robot
- You need to define the cte function which is called inside the run function to determine the steering angle.
- Four cases should be considered when compute cte
    - ❖ $x \leq r$
    - ❖ $r < x \leq 3r$ and $y < r$
    - ❖ $r < x \leq 3r$ and $y > r$
    - ❖ $r > 3r$

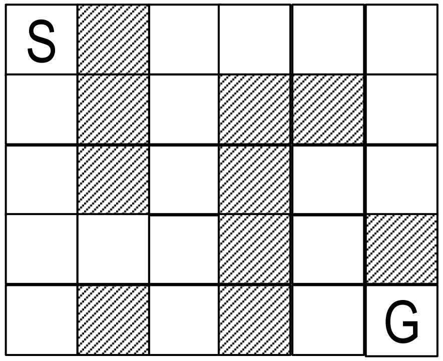


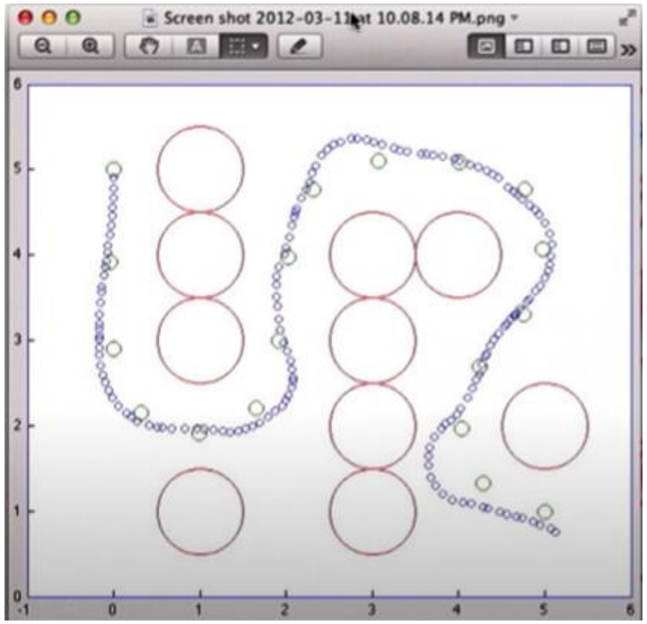
Put It Together

- We have studied
  - Localization:
    - Markov: Discrete, Multimodal
    - Kalman: Continuous, Unimodal
    - Particle: Exponential, Multimodal
  - Path Planning and Smoothing:
    - Breadth First, Dijkstra, A*: Optimal
    - Smoothing: Continuous, local
  - PID Control
    - P: Minimize error
    - I: Compensate system error such as steering drifting
    - D: Avoids overshoot

Put It Together (Cont.)

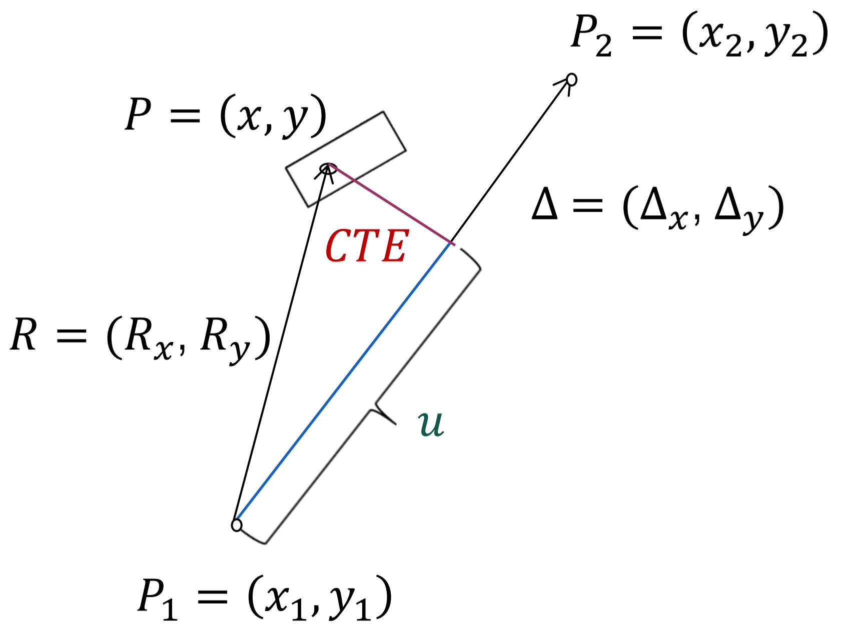
- Path Planning: A* Planner
- Path Smoothing: Initial and goal points are fixed
- Path Following: PD controller
  - Repeat the following until the goal is reached or timeout
    - Particle filter estimates the robot pose
    - Compute CTE and then steering
    - Move robot and particles
    - Resample particles
    - Check robot collision





PP5B: Segmented CTE

- When a path composed of a sequence of line segments, which segment on the path will be selected to compute CTE?



$$CTE = \frac{-\Delta_y R_x + \Delta_x R_y}{\sqrt{\Delta_x \Delta_x + \Delta_y \Delta_y}}$$

$$u = \frac{\Delta_x R_x + \Delta_y R_y}{\sqrt{\Delta_x \Delta_x + \Delta_y \Delta_y}}$$

$\Delta_x = x_2 - x_1$

$\Delta_y = y_2 - y_1$

$R_x = x - x_1$

$R_y = y - y_1$

- If $|u| > \sqrt{\Delta_x \Delta_x + \Delta_y \Delta_y}$, then the next line segment should be selected