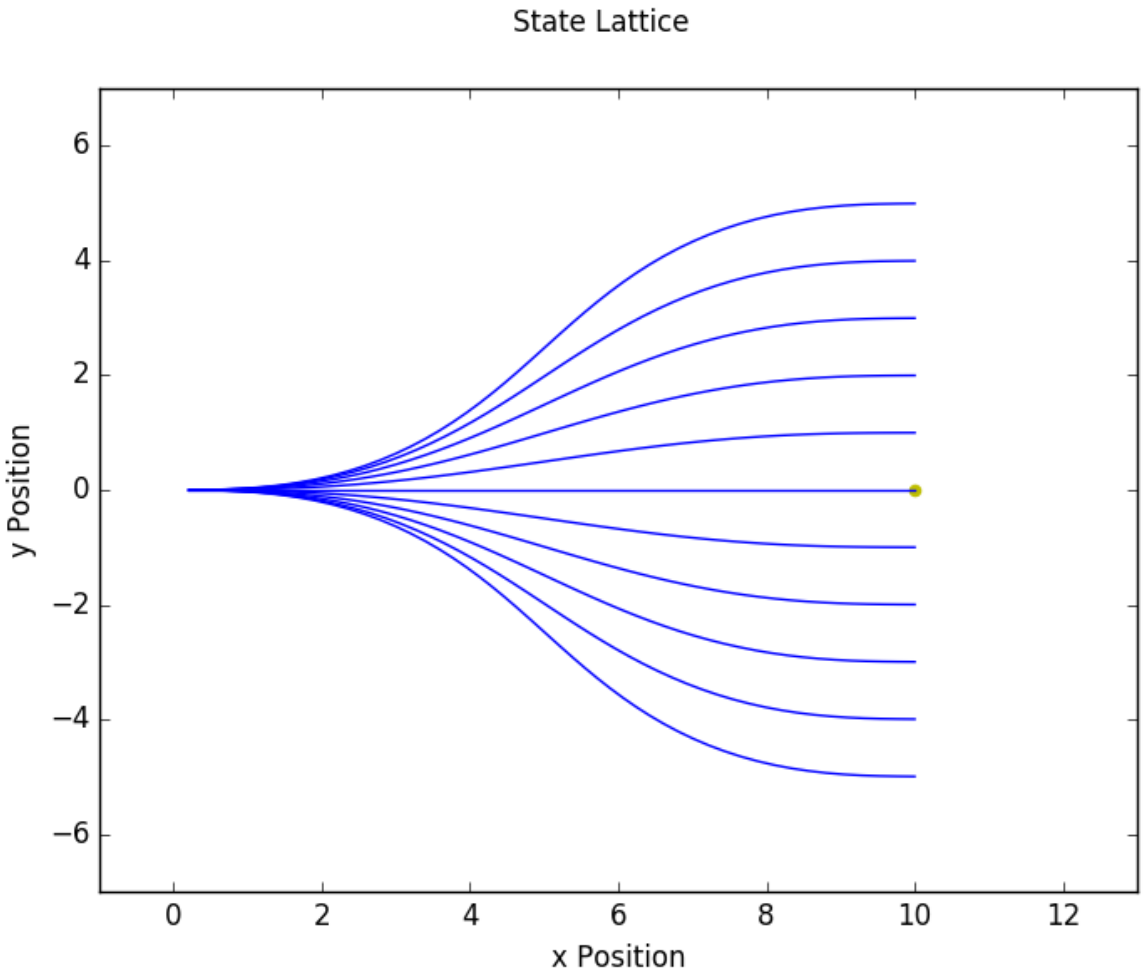
Conformal Lattice Planning

Course 4, Module 7, Lesson 4

UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

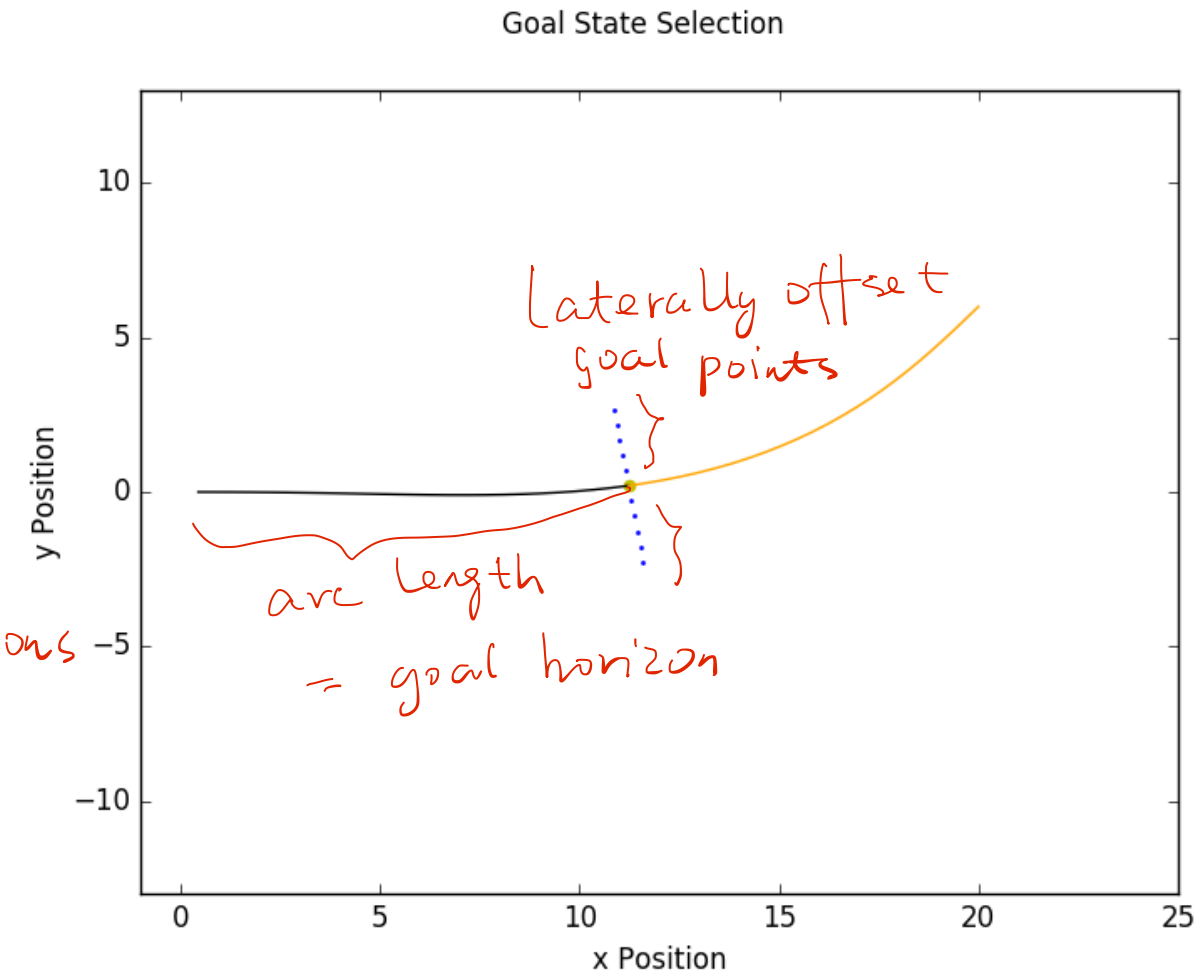
Conformal Lattice

- Goal is to plan a feasible collision-free path to goal
- Conformal lattice exploits road structure to speed up planning
- Lattice paths are laterally offset from a goal point along road



Goal Horizon

- Short lookahead improves computation time, but reduces ability to avoid obstacles
- Goal point is dynamically calculated based on speed and other factors weather conditions
- Endpoints are sampled laterally offset from goal according to the heading along the road



Generating Spirals

- Can then compute cubic spirals to each goal point
- Focus on kinematic feasibility for now, collision-checking comes later
- If a goal point cannot be reached with a spiral under the kinematic constraints, discard that goal point

$$\min f_{be}(a_0, a_1, a_2, a_3, s_f) + \alpha\left(x_S(p_4) - x_f\right) + \beta\left(y_S(p_4) - y_f\right) + \gamma\left(\theta_S(p_4) - \theta_f\right)$$

$$\text{s. t.} \begin{cases} |p_1| \leq \kappa_{max} \\ |p_2| \leq \kappa_{max} \end{cases}$$

Getting Spiral Parameters

- Convert optimization variables back into spiral parameters
- Can then use spiral coefficients to sample points along the spiral

$$a_0 = p_0$$

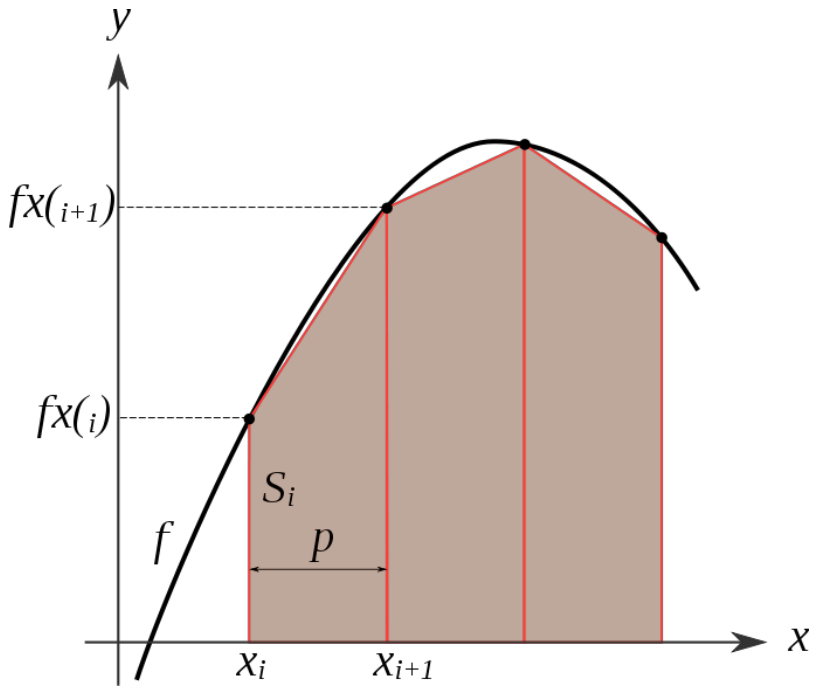
$$a_1 = -\frac{11p_0/2 - 9p_1 + 9p_2/2 - p_3}{p_4}$$

$$a_2 = \frac{9p_0 - 45p_1/2 + 18p_2 - 9p_3/2}{p_4^2}$$

$$a_3 = -\frac{9p_0/2 - 27p_1/2 + 27p_2/2 - 9p_3/2}{p_4^3}$$

Trapezoidal Rule Integration

- Use numerical integration to generate positions along path
- Trapezoidal rule is faster for generating entire path than Simpson's rule
- Discrete representation generated using cumulative_trapezoid() function in Python



$$x(s) = x_0 + \int_0^s \cos\big(\theta(s')\big)ds'$$

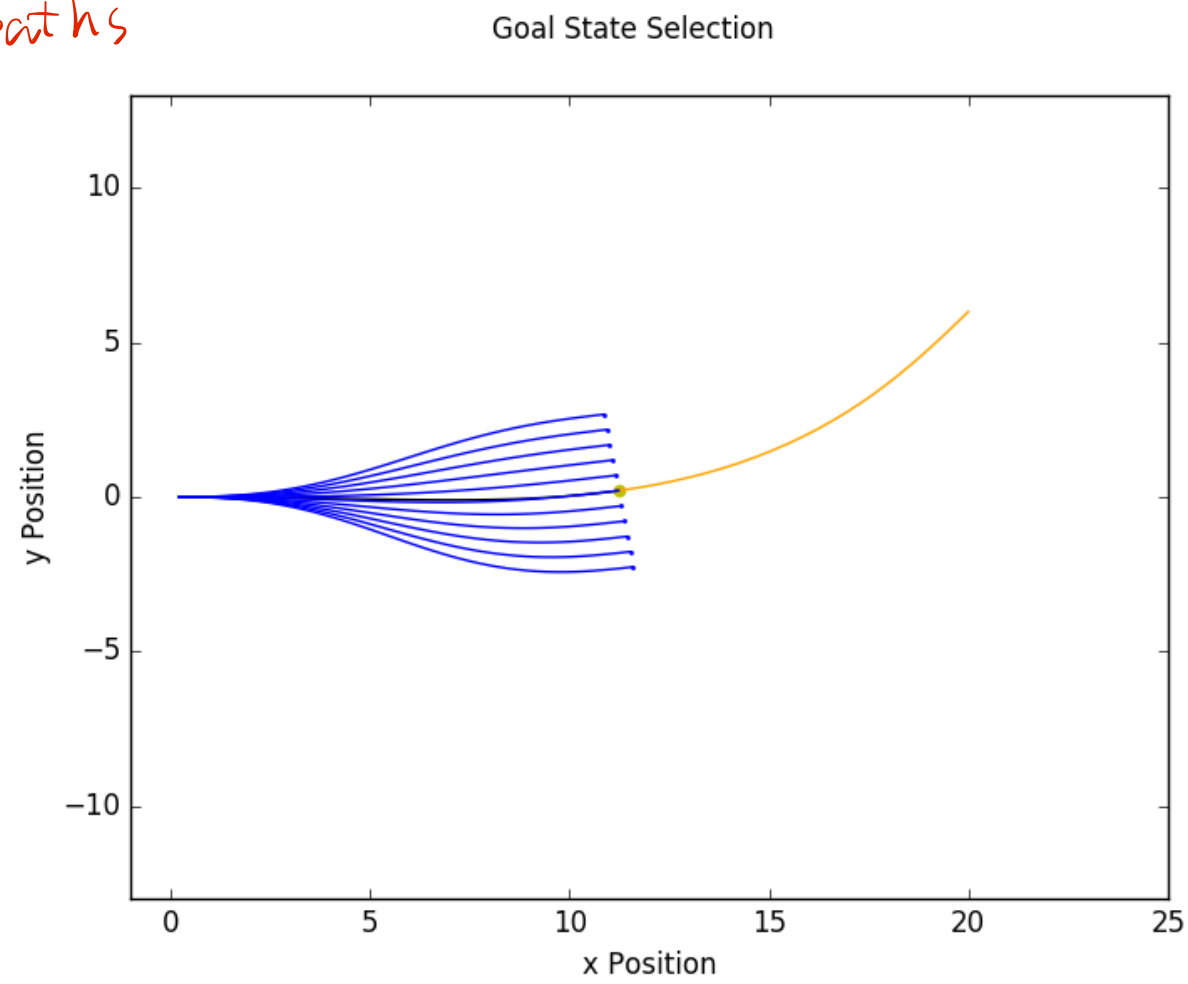
$$y(s) = y_0 + \int_0^s \sin\big(\theta(s')\big)ds'$$

$$\Rightarrow \int_0^s f(x)dx \approx \sum_{i=1}^{N-1} \frac{f(x_{i+1}) + f(x_i)}{2}(x_{i+1} - x_i)$$

Generated Path Set

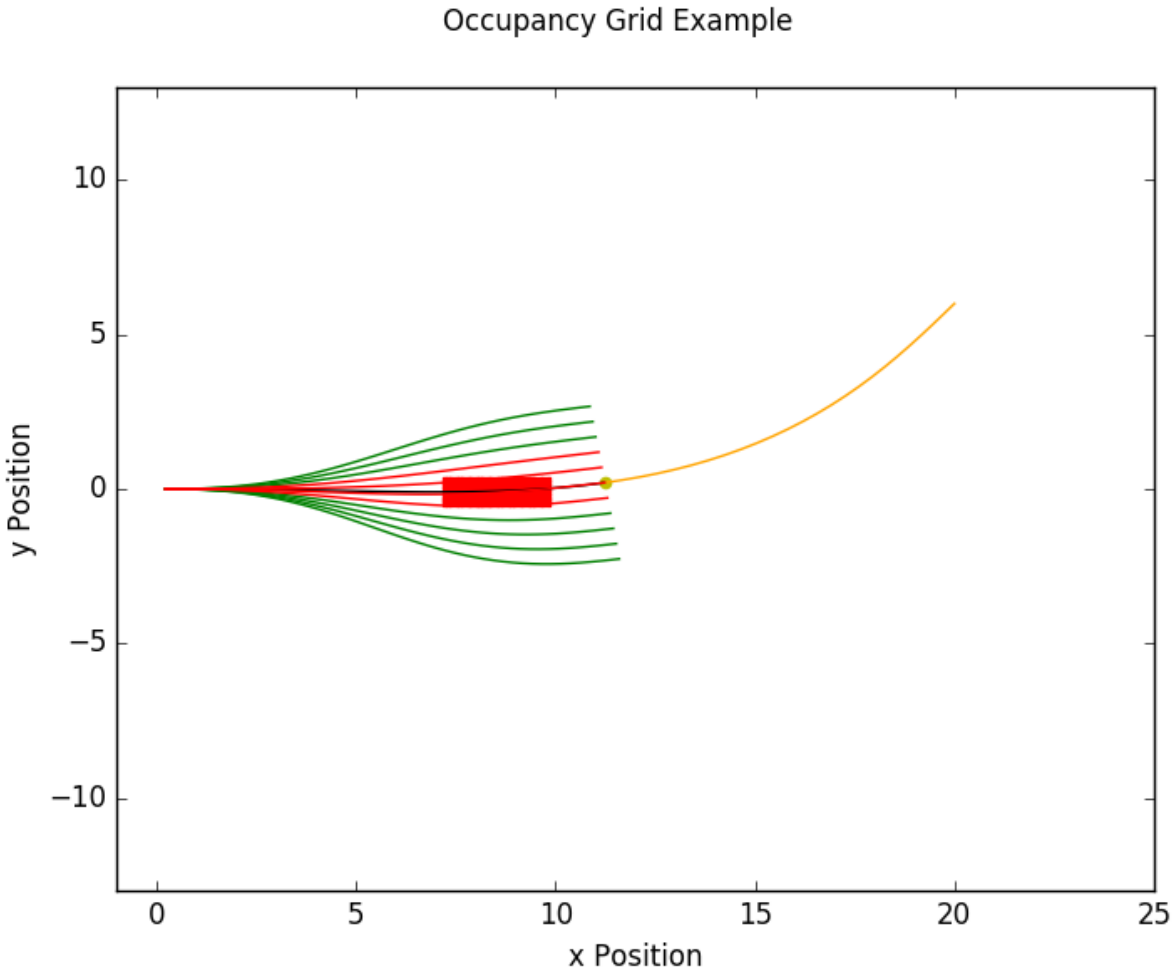
generate set of paths

- For each of our goal states, we can optimize a spiral to the goal point
- Using trapezoidal numerical integration we get a discrete path representation
- Now need to see which are collision-free



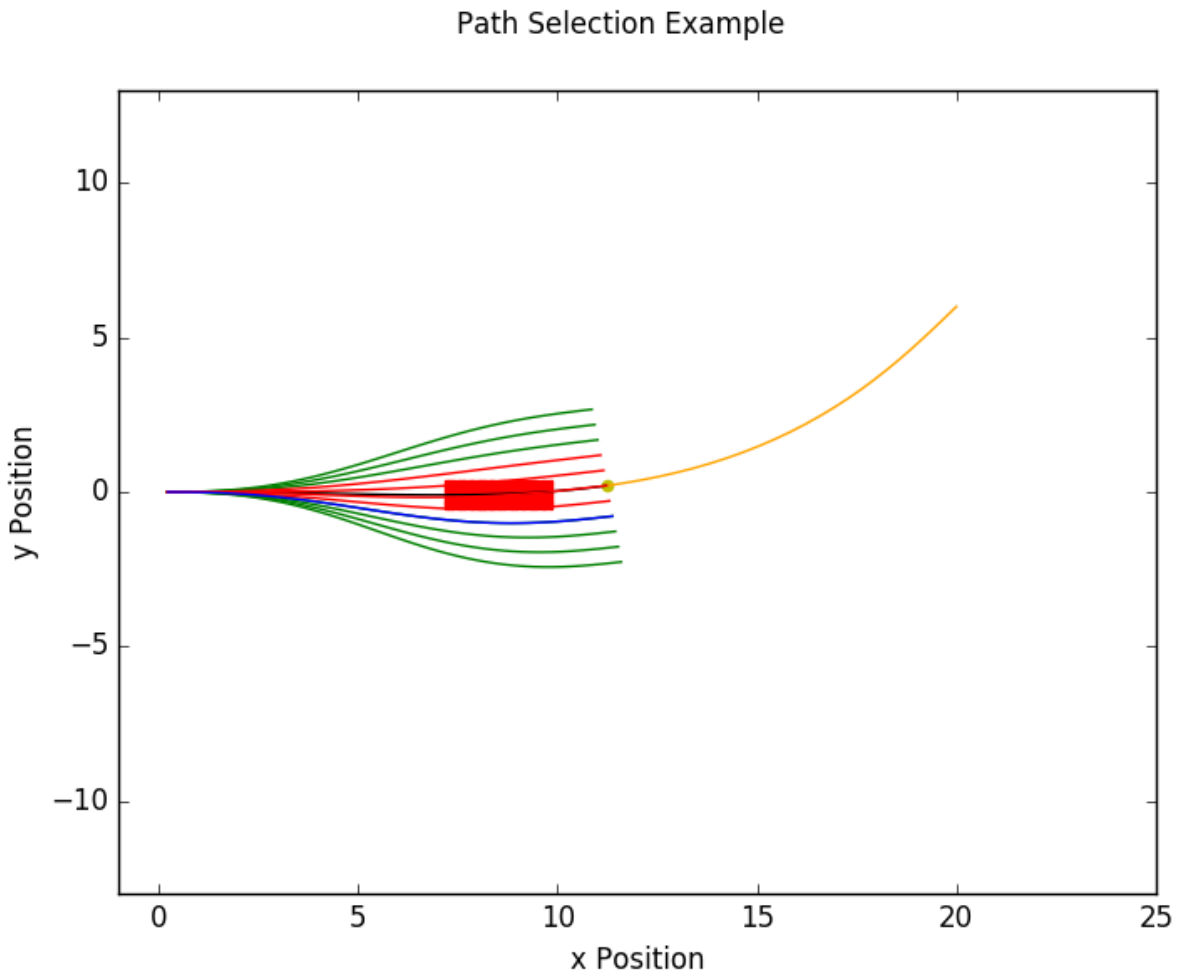
Collision Checking

- Can do this through circle-based or swath-based collision checking for each point along each path (as in Module 4)
- Parked vehicle (in red) represents obstacle, paths that would result in a collision with it are marked red



Path Selection

- Need to select best path among collision-free paths
- Objective function for selection is a design choice
- Can reward paths that track the center of the road, and penalize paths that come too close to obstacles
- Best path highlighted in blue



Full Path

- Can repeat this process for each planning step as car moves along road
- Path will converge to the centerline of the road, even when obstacles are present

in a receding horizon fashion