Motion Planning

A Brief Overview:
-Trajectory Generation
-Collision Avoidance
-Optimal Control

Trajectory Generation

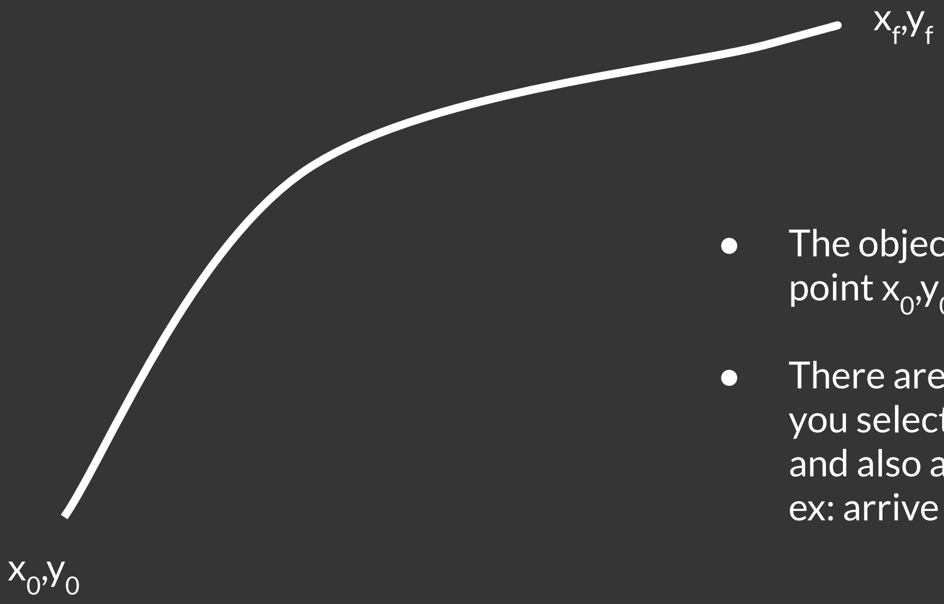
Trajectory Generation

Two types of robots:

Type I : Non-Holonomic

Type II : Holonomic

Trajectory Generation



- The objective is to get the robot from initial point x_0, y_0 to final point x_f, y_f
- There are several such paths, but the path you select has to executable by the robot and also adhere to user imposed constraints ex: arrive at x_t, y_t at some time ‘t’

Trajectory Generation

- Non-holonomic trajectory generation

  Consider the robot a robot in 2d “configuration space” with state q=(x,y,vx,vy,θ)
  A Trajectory is the evolution of the state with time: i.e. a function q(t).

  Objective find q(t):
  with respect to some trajectory constraints i.e.
  $x(t_c) = x_{tc}$, $y(t_c) = y_{tc}$, $vx(t_c)= vx_{tc}$, $vy(t_c)= vy_{tc}$, $\theta(t_c)=\theta_{tc}$

  With nonholonomic constraint vy = vx*tan(θ)

  $$\dot{x}\sin\theta - \dot{y}\cos\theta = 0 \implies \dot{y} = \dot{x}\tan\theta$$

  Lets derivative of state q as

  $$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}$$

Trajectory Generation

- Non-holonomic trajectory generation

Lets represent the function we need to find q(t) with a Bernstein basis polynomial

$$\mathbf{B_n(f(x);t) = \sum_{i=0}^{n} f(\frac{i}{n}) {}^nC_i t^i (1-t)^{n-i}}$$

Parameterize x(t) and tan(θ(t))

$$x(t) \approx B_n(x(t)) = B_x(\mu(t)) = \sum_{i=0}^{5} W_{x_i} B_i(\mu(t)),$$

$$\tan\theta(t) = k(t) \approx B_n(k(t)) = B_k(\mu(t)) = \sum_{i=0}^{5} W_{k_i} B_i(\mu(t))$$

Trajectory Generation

where,

$B_i(\mu(t)) =^n C_i(1-\mu)^i(\mu)^{n-i}$

$\mu(t) = \frac{t-t_0}{t_f-t_0}$

y is represented in terms of vx: $$y = \int \dot{x}\tan\theta\, dt$$

$$y(t) = y_0 + \int_{t_0}^{t} (\sum_{i=0}^{5} W_{x_i}\dot{B}_i(\mu(t))(\sum_{i=0}^{5} W_{k_i} B_i(\mu(t))dt$$

Trajectory Generation

-The problem remains now to determine the weights of the polynomial basis:

-For instance if you choose n=5 for the bernstein the resulting expression for x(t):

$$x(t_0) = W_{x_0}B_0(\mu(t_0)) + W_{x_1}B_1(\mu(t_0)) + W_{x_2}B_2(\mu(t_0)) + W_{x_3}B_3(\mu(t_0)) + W_{x_4}B_4(\mu(t_0)) + W_{x_5}B_5(\mu(t_0)) \tag{9}$$

$$x(t_f) = W_{x_0}B_0(\mu(t_f)) + W_{x_1}B_1(\mu(t_f)) + W_{x_2}B_2(\mu(t_f)) + W_{x_3}B_3(\mu(t_f)) + W_{x_4}B_4(\mu(t_f)) + W_{x_5}B_5(\mu(t_f)) \tag{10}$$

Format these equations into the standard form AX=B

$$\begin{bmatrix} x_{t_c} - W_{x_0}B_0(\mu(t_c)) - W_{x_5}B_5(\mu(t_c)) \\ \dot{x}_{t_0} - W_{x_0}\dot{B}_0(\mu(t_0)) - W_{x_5}\dot{B}_5(\mu(t_0)) \\ \dot{x}_{t_f} - W_{x_0}\dot{B}_0(\mu(t_f)) - W_{x_5}\dot{B}_5(\mu(t_f)) \\ \dot{x}_{t_c} - W_{x_0}\dot{B}_0(\mu(t_c)) - W_{x_5}\dot{B}_5(\mu(t_c)) \end{bmatrix} = \begin{bmatrix} B_1(\mu(t_c)) & B_2(\mu(t_c)) & B_3(\mu(t_c)) & B_4(\mu(t_c)) \\ \dot{B}_1(\mu(t_0)) & \dot{B}_2(\mu(t_0)) & \dot{B}_3(\mu(t_0)) & \dot{B}_4(\mu(t_0)) \\ \dot{B}_1(\mu(t_f)) & \dot{B}_2(\mu(t_f)) & \dot{B}_3(\mu(t_f)) & \dot{B}_4(\mu(t_f)) \\ \dot{B}_1(\mu(t_c)) & \dot{B}_2(\mu(t_c)) & \dot{B}_3(\mu(t_c)) & \dot{B}_4(\mu(t_c)) \end{bmatrix} \begin{bmatrix} W_{x_1} \\ W_{x_2} \\ W_{x_3} \\ W_{x_4} \end{bmatrix} \tag{13}$$

Trajectory Generation

-The coefficients of y(t) are:

$$y(t) = y_0 + \int_{t_0}^{t} (W_{k_0} . f_0(t, t_0, t_f, W_{x1}, W_{x2}, ...W_{x5}) + (W_{k_1} . f_1(t, t_0, t_f, W_{x1}, W_{x2}, ...W_{x5}) + ...$$

The terms in the integral have to be integrated numerically (`integrate` function in Matlab can be used)

$$y(t) = y_0 + W_{k_0} F_0(t) + W_{k_1} F_1(t) + W_{k_2} F_2(t) + W_{k_3} F_3(t) + W_{k_4} F_4(t) 0 + W_{k_5} F_5(t) \quad (17)$$

$$where F_i(t) = \int_{t_0}^{t} f_i(t, t_0, t_f, W_{x_1}, W_{x_2}, ...W_{x_5}) dt \quad (18)$$

Our objective is to get weights, $W_{K_0}, W_{K_1}, W_{K_2}, W_{K_3}, W_{K_4}, W_{K_4}$

Note tan(θ(t)) is represented by

$$k(t_0) = k_0 = W_{k_0} B_0(\mu(t_0)) + W_{k_1} B_1(\mu(t_0)) + W_{k_2} B_2(\mu(t_0)) + W_{k_3} B_3(\mu(t_0)) + W_{k_4} B_4(\mu(t_0)) + W_{k_5} B_5(\mu(t_0)) \quad (19)$$

Trajectory Generation

-Combine the tan(θ(t)), y(t), vy(t) constraints to form

$$\begin{bmatrix} \dot{k}_{t_0} - W_{k_0}\dot{B}_0(\mu(t_0)) - W_{k_5}\dot{B}_5(\mu(t_0)) \\ \dot{k}_{t_f} - W_{k_0}\dot{B}_0(\mu(t_f)) - W_{k_5}\dot{B}_5(\mu(t_f)) \\ y_0 - W_{k_0}F_0 - W_{k_5}F_5 \\ y_f - W_{k_5}F_0 - W_{k_5}F_5 \end{bmatrix} = \begin{bmatrix} \dot{B}_{k_1}(\mu(t_0)) & \dot{B}_2(\mu(t_0)) & \dot{B}_3(\mu(t_0)) & \dot{B}_4(\mu(t_0)) \\ \dot{B}_1(\mu(t_f)) & \dot{B}_2(\mu(t_f)) & \dot{B}_3(\mu(t_f)) & \dot{B}_4(\mu(t_f)) \\ F_1(t_0) & F_2(t_0) & F_3(t_0) & F_4(t_0) \\ F_1(t_f) & F_2(t_f) & F_3(t_f) & F_4(t_f) \end{bmatrix} \begin{bmatrix} W_{k_1} \\ W_{k_2} \\ W_{k_3} \\ W_{k_4} \end{bmatrix} \quad (23)$$

-Solve the equation by taking the pseudo inverse:

$$A_k = B_k W_k$$

$$W_k = pinv(B_k)A_k$$

-Note:you solve for the x,y-weights separately

Trajectory Generation

-Assignment 1: Consider the curve

(x(t) ,y(t)) = (at^3 + bt^2+ct+d, et^3 + ft^2+gt+h)

Passing through points (x,y,t): (0,0,0) , (4,3,4), (8,7,9) and (vx,t) = (0,0), (0,9) and (vy,t) = (0,0),(0,9)

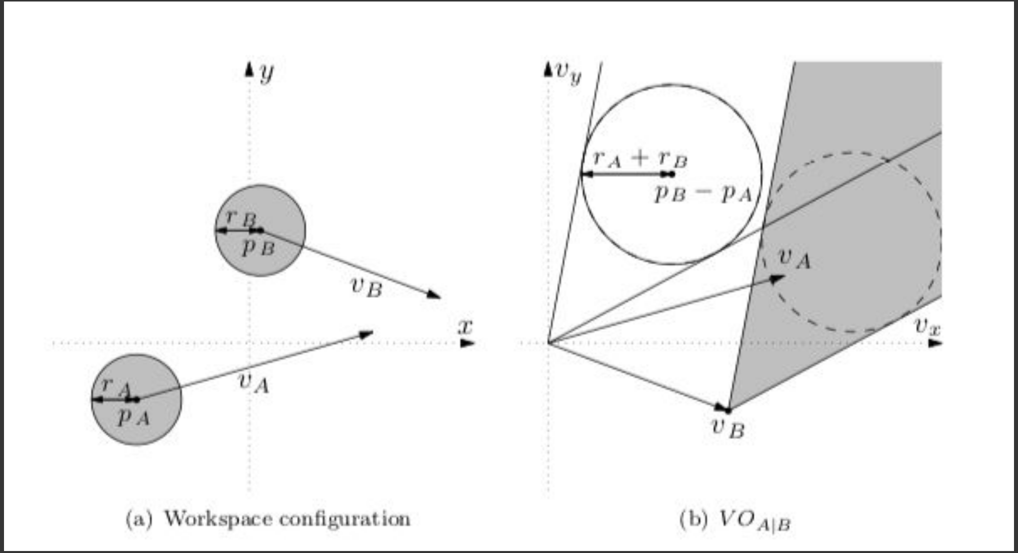
Find the coefficients of function x(t),y(t)

Collision Avoidance

Collision Avoidance

There are several methods for collision avoidance between two robots, one such important method is called the Velocity Obstacle

Consider the two robots shown in (a)



(a) Workspace configuration (b) $VO_{A|B}$

Question:

-Would they collide?

-How can you select a velocity that avoids collision?

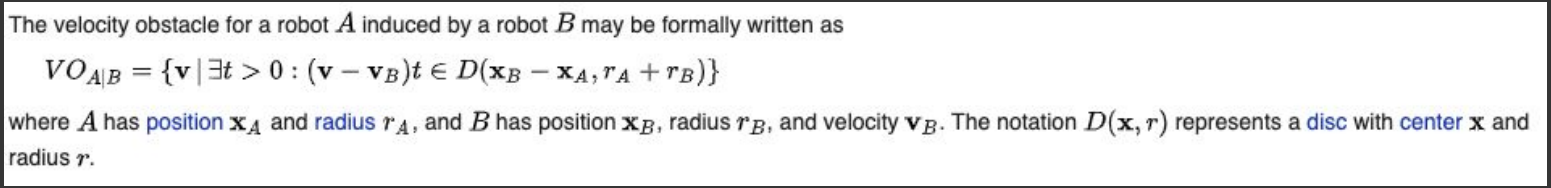
Collision Avoidance

Question 1: Would they collide?

-If the future position of the robot falls in the enlarged disk of radius $r_A + r_B$ then the robots do collide:

- The VO is a geometric representation of all velocities that will eventually result in a collision given the dynamic obstacle maintains the observed velocity.

From wiki :



The velocity obstacle for a robot A induced by a robot B may be formally written as

$$VO_{A|B} = \{\mathbf{v} \mid \exists t > 0 : (\mathbf{v} - \mathbf{v}_B)t \in D(\mathbf{x}_B - \mathbf{x}_A, r_A + r_B)\}$$

where A has position $\mathbf{x}_A$ and radius r_A, and B has position $\mathbf{x}_B$, radius r_B, and velocity $\mathbf{v}_B$. The notation $D(\mathbf{x}, r)$ represents a disc with center $\mathbf{x}$ and radius r.

Therefore A shouldn't be taking the velocities in $VO_{A|B}$ else it risks collision with B

Collision Avoidance

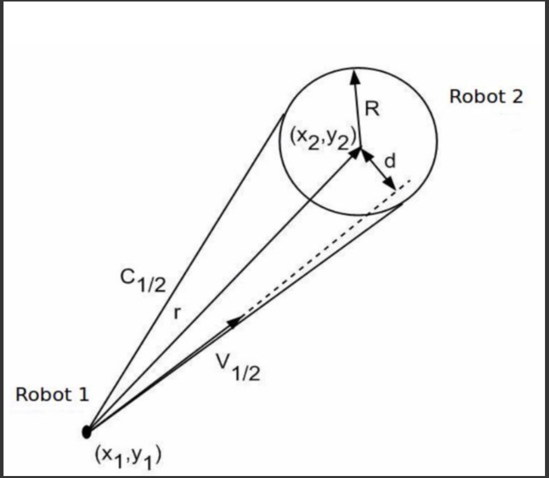
Question 2: Obtaining a equation for VO :

-Take the minkowski sum of the A, B to get the disk of radius $R = r_a + r_b$ (switch to the velocity space)

-The Relative velocity is represented by $V_{1/2}$ the tangent to the disk is $C_{1/2}$

-The velocity you pick should be outside the collision cone i.e. $d^2 >= R^2$

$$d^2 = |\vec{r}|^2 - \frac{\vec{r}.\vec{V}_{1/2}}{|\vec{V}_{1/2}|^2} \geq R^2$$



Collision Avoidance

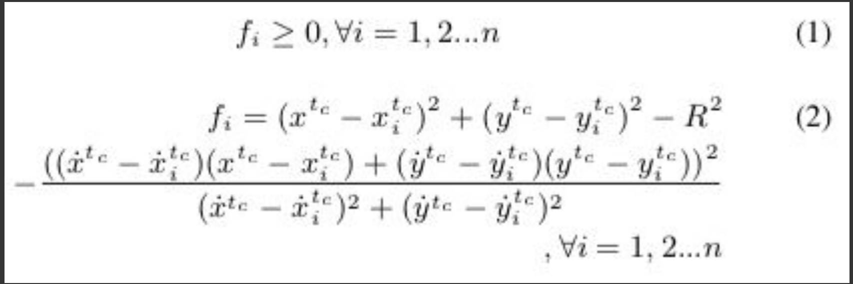
Scaling:
-Scaling allows you to increase or decrease the velocity along a fixed direction. i.e. v_A is transformed as s^*v_A
(If s>1 the robot velocity is increased s<1 then it slows down)

This requires a transformation of time as :

$dx(t_n)/dt = x(t)*dt/dt_n$

Where $s = dt/dt_n$

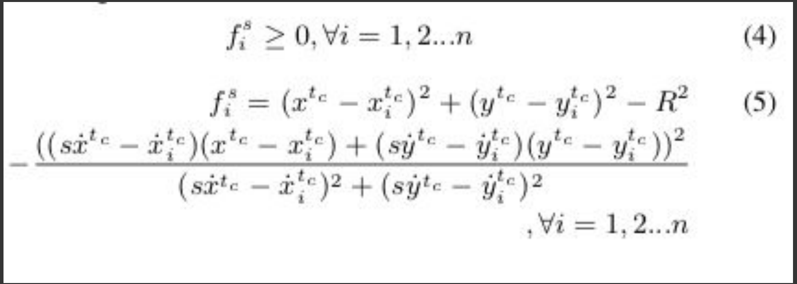
The original equation :



$$f_i \geq 0, \forall i = 1, 2 ... n \tag{1}$$

$$f_i = (x^{t_c} - x_i^{t_c})^2 + (y^{t_c} - y_i^{t_c})^2 - R^2 - \frac{((\dot{x}^{t_c} - \dot{x}_i^{t_c})(x^{t_c} - x_i^{t_c}) + (\dot{y}^{t_c} - \dot{y}_i^{t_c})(y^{t_c} - y_i^{t_c}))^2}{(\dot{x}^{t_c} - \dot{x}_i^{t_c})^2 + (\dot{y}^{t_c} - \dot{y}_i^{t_c})^2}, \forall i = 1, 2 ... n \tag{2}$$

Can be rewritten into the form:



$$f_i^s \geq 0, \forall i = 1, 2 ... n \tag{4}$$

$$f_i^s = (x^{t_c} - x_i^{t_c})^2 + (y^{t_c} - y_i^{t_c})^2 - R^2 - \frac{((s\dot{x}^{t_c} - \dot{x}_i^{t_c})(x^{t_c} - x_i^{t_c}) + (s\dot{y}^{t_c} - \dot{y}_i^{t_c})(y^{t_c} - y_i^{t_c}))^2}{(s\dot{x}^{t_c} - \dot{x}_i^{t_c})^2 + (s\dot{y}^{t_c} - \dot{y}_i^{t_c})^2}, \forall i = 1, 2 ... n \tag{5}$$

Collision Avoidance

The equation can be further simplified and rewritten into a form

$f^s = a*s^2+b*s+c >= 0$. This allows us to compute a closed form solution for s.

$$\begin{aligned} s \in [0, -b - \frac{\sqrt{b^2-4ac}}{2a}] \cup [-b + \frac{\sqrt{b^2-4ac}}{2a}, \infty) \\ , or [-b + \frac{\sqrt{b^2-4ac}}{2a}, \infty), if\, a > 0 \end{aligned} \tag{12}$$

$$\begin{aligned} s \in [-b - \frac{\sqrt{b^2-4ac}}{2a}, -b + \frac{\sqrt{b^2-4ac}}{2a}] \\ or [0, -b + \frac{\sqrt{b^2-4ac}}{2a}], if\, a < 0 \end{aligned} \tag{13}$$

Collision Avoidance

Demo:
-(Documents ▸ MATLAB ▸ scaleuncertainity ▸ determinsticcodemastifsp2)

Assignment implement scaling. For a robot.

Optimal Control

Optimal Control

So far we can observe that the major ingredients for a control problem are :

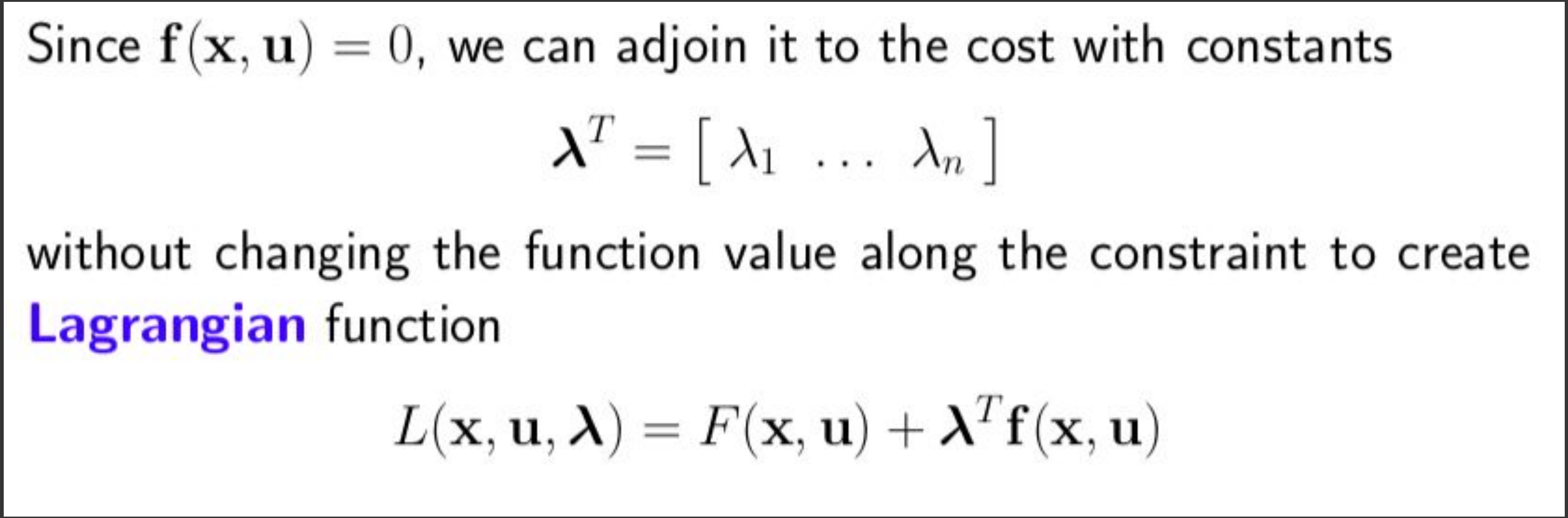
1) A cost function
2) A motion model
3) Any other constraints

Ex: Cost function F(y): constraints: f(y) can be considered as motion model.

$$\min_{\mathbf{y}} F(\mathbf{y})$$

$$\text{such that } \mathbf{f}(\mathbf{y}) = 0$$

Optimal Control



Since $\mathbf{f}(\mathbf{x},\mathbf{u}) = 0$, we can adjoin it to the cost with constants

$$\boldsymbol{\lambda}^T = \left[\begin{array}{ccc} \lambda_1 & \dots & \lambda_n \end{array}\right]$$

without changing the function value along the constraint to create **Lagrangian** function

$$L(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) = F(\mathbf{x},\mathbf{u}) + \boldsymbol{\lambda}^T\mathbf{f}(\mathbf{x},\mathbf{u})$$

- Given values of $\mathbf{x}$ and $\mathbf{u}$ for which $\mathbf{f}(\mathbf{x},\mathbf{u})=0$, consider differential changes to the Lagrangian from differential changes to $\mathbf{x}$ and $\mathbf{u}$:

$$dL = \frac{\partial L}{\partial \mathbf{x}} d\mathbf{x} + \frac{\partial L}{\partial \mathbf{u}} d\mathbf{u}$$

where $\frac{\partial L}{\partial \mathbf{u}} = \left[\begin{array}{ccc} \frac{\partial L}{\partial u_1} & \cdots & \frac{\partial L}{\partial u_m} \end{array} \right]$ (row vector)

- Since $\mathbf{u}$ are the decision variables it is convenient to choose $\boldsymbol{\lambda}$ so that

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{x}} &\triangleq \frac{\partial F}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \equiv 0 && (2.1) \\ \Rightarrow \boldsymbol{\lambda}^T &= -\frac{\partial F}{\partial \mathbf{x}} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)^{-1} && (2.2) \end{aligned}$$

- To proceed, must determine what changes are possible to the **cost** keeping the equality constraint satisfied.
  - Changes to $\mathbf{x}$ and $\mathbf{u}$ are such that $\mathbf{f}(\mathbf{x},\mathbf{u})=0$, then

$$\begin{aligned} d\mathbf{f} &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}} d\mathbf{x} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}} d\mathbf{u} \equiv 0 && (2.3) \\ \Rightarrow d\mathbf{x} &= -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} d\mathbf{u} && (2.4) \end{aligned}$$

- Then the allowable cost variations are

$$\begin{aligned} dF &= \frac{\partial F}{\partial \mathbf{x}} d\mathbf{x} + \frac{\partial F}{\partial \mathbf{u}} d\mathbf{u} && (2.5) \\ &= \left(-\frac{\partial F}{\partial \mathbf{x}} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} + \frac{\partial F}{\partial \mathbf{u}} \right) d\mathbf{u} \\ &= \left(\frac{\partial F}{\partial \mathbf{u}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right) d\mathbf{u} && (2.6) \\ &\equiv \frac{\partial L}{\partial \mathbf{u}} d\mathbf{u} && (2.7) \end{aligned}$$

- So the gradient of the cost F with respect to $\mathbf{u}$ while keeping the constraint $\mathbf{f}(\mathbf{x}, \mathbf{u}) = 0$ is just

$$\frac{\partial L}{\partial \mathbf{u}}$$

and we need this gradient to be zero to have a stationary point so that $dF = 0 \; \forall \; d\mathbf{u} \neq 0$.

- Thus the necessary conditions for a stationary value of F are

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{x}} &= 0 && (2.8) \\ \frac{\partial L}{\partial \mathbf{u}} &= 0 && (2.9) \\ \frac{\partial L}{\partial \boldsymbol{\lambda}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}) = 0 && (2.10) \end{aligned}$$

Assignment problem:

Min $(x_{t+1}-x_g)^2+(x_{t+1}-y_g)^2$
x,u

s.t $\quad x_{t+1} = x_t+v_x dt$
$\quad y_{t+1} = y_t+v_y dt$

Use Matlab function `**fmincon**` to solve this problem

Assignment with an
Min $(x_{t+1}-x_g)^2+(x_{t+1}-y_g)^2$
x,u

s.t $\quad x_{t+1} = x_t+v_x dt$
$\quad y_{t+1} = y_t+v_y dt$
$\quad (x_a-x_b)^2+(y_a-y_b)^2 >= R^2$

References

Trajectory Generation:

- Bezier Curve MR Scribe - Enna & Gourav

Collision Avoidance:

- Reciprocal Velocity Obstacles for real-time multi-agent navigation - Jur Vandenberg
- Reactive collision avoidance for multiple robots by non linear time scaling - Arun Singh

Optimal Control:

- Optimal Control - Syrmose
- Model predictive control for autonomous driving based on time scaled collision cone - Mithun

Reference

Papers by Bharath Gopalakrishnan, Bhargav @ https://robotics.iiit.ac.in/publications.html