Motion Planning and State Estimation in Robotics

Arun Kumar Singh

University of Tartu *arun.singh@ut.ee, aks1812@gmail.com*

November 9, 2021

Overview

Course Repo

Link to Course Repo
`https://github.com/arunkumar-singh/Motion_Planning_Lecture_Codes`

Global Trajectory Planning

- Till now we have considered only incremental trajectory planning. That is, we were only considered about planning a trajectory for say next 0.1s or less.
- Incremental trajectory planning is myopic and greedy. The overall trajectory quality is sub-optimal
- We now move to global trajectory planning. That is, we plan the trajectory taking into account all the costs and constraints for all time steps into the future
- We will do this by modeling trajectories as polynomials.

Recall Triple Integrator System

If the input jerk j_x, j_y is constant, we get the following by integrating between time t_0 and t

$$\dddot{x} = \jmath_x, \dddot{y} = \jmath_y \tag{1}$$

$$\ddot{x}(t) = \ddot{x}_0 + \jmath_x(t - t_0), \ddot{y} = \ddot{y} + \jmath_y(t - t_0) \tag{2}$$

$$\dot{x}(t) = \dot{x}_0 + \ddot{x}_0(t - t_0) + \frac{1}{2}\jmath_x(t - t_0)^2, \dot{y}(t) = \dot{y}_0 + \ddot{y}_0(t - t_0) + \frac{1}{2}\jmath_y(t - t_0)^2 \tag{3}$$

$$x(t) = x_0 + \dot{x}_0 t + \frac{1}{2}\ddot{x}_0(t - t_0)^2 + \frac{1}{6}\jmath_x(t - t_0)^3 \tag{4}$$

$$y(t) = y_0 + \dot{y}_0 t + \frac{1}{2}\ddot{y}_0(t - t_0)^2 + \frac{1}{6}\jmath_y(t - t_0)^3 \tag{5}$$

What if the input is not constant?

Polynomial Trajectories

Let

$$\dddot{x}(t) = 6a_1 + 24a_2 t + 60a_3 t^2, \dddot{y}(t) = 6b_1 + 24b_2 t + 60b_3 t^2 \tag{6}$$

Integrating three times, we get

$$\ddot{x}(t) = \ddot{x}_0 + 6a_1 t + 12a_2 t^2 + 20a_3 t^3 \tag{7}$$
$$\dot{x}(t) = \dot{x}_0 + \ddot{x}_0 t + 3a_1 t^2 + 4a_2 t^3 + 5a_3 t^4 \tag{8}$$
$$x(t) = x_0 + \dot{x}_0 t + \frac{1}{2}\ddot{y}_0 t^2 + a_1 t^3 + a_2 t^4 + a_3 t^5 \tag{9}$$

$$\ddot{y}(t) = \ddot{y}_0 + 6b_1 t + 12b_2 t^2 + 20b_3 t^3 \tag{10}$$
$$\dot{y}(t) = \dot{y}_0 + \ddot{y}_0 t + 3b_1 t^2 + 4b_2 t^3 + 5b_3 t^4 \tag{11}$$
$$y(t) = y_0 + \dot{y}_0 t + \frac{1}{2}\ddot{y}_0 t^2 + b_1 t^3 + b_2 t^4 + b_3 t^5 \tag{12}$$

Polynomial Trajectories

Let

$$\dddot{x}(t) = 6a_1 + 24a_2 t + 60a_3 t^2 + 120a_4 t^3 + 210a_5 t^4 \tag{13}$$

Integrating three times, we get the following, where $x_0, \dot{x}_0, \ddot{x}_0$ are the initial position, velocity and acceleration of the robot.

$$\ddot{x}(t) = \ddot{x}_0 + 6a_1 t + 12a_2 t^2 + 20a_3 t^3 + 30a_4 t^4 + 42a_5 t^5 \tag{14}$$

$$\dot{x}(t) = \dot{x}_0 + \ddot{x}_0 t + 3a_1 t^2 + 4a_2 t^3 + 5a_3 t^4 + 6a_4 t^5 + 7a_5 t^6 \tag{15}$$

$$x(t) = x_0 + \dot{x}_0 t + \frac{1}{2}\ddot{x}_0 t^2 + a_1 t^3 + a_2 t^4 + a_3 t^5 + a_4 t^6 + a_5 t^7 \tag{16}$$

We can write similar expression for y as well

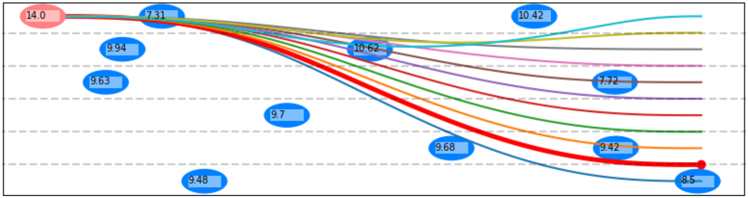
$$\ddot{y}(t) = \ddot{y}_0 + 6b_1 t + 12b_2 t^2 + 20b_3 t^3 + 30b_4 t^4 + 42b_5 t^5 \tag{17}$$

$$\dot{y}(t) = \dot{y}_0 + \ddot{y}_0 t + 3b_1 t^2 + 4b_2 t^3 + 5b_3 t^4 + 6b_4 t^5 + 7b_5 t^6 \tag{18}$$

$$y(t) = y_0 + \dot{y}_0 t + \frac{1}{2}\ddot{y}_0 t^2 + b_1 t^3 + b_2 t^4 + b_3 t^5 + b_4 t^6 + b_5 t^7 \tag{19}$$

Computing the right a's and b's

- Different a's and b's lead to different trajectories.
- Global trajectory planning can be thought as the problem of computing the right a's and b's for your problem setting.



(a)

Figure: Each trajectory corresponds to a different set of a's and b's

Polynomial Trajectories in Matrix Form

Lets start with the 5th order polynomial.
During numerical computations, we are mostly interested in knowing trajectory values at specific time instants like $t_0, t_1, t_2 \dots t_f$.

$$x(t_0) = x_0 \tag{20}$$

$$x(t_1) = x_0 + \dot{x}_0 t_1 + \frac{1}{2}\ddot{x}_0 t_1^2 + a_1 t_1^3 + a_2 t_1^4 + a_3 t_1^5 \tag{21}$$

$$x(t_2) = x_0 + \dot{x}_0 t_2 + \frac{1}{2}\ddot{x}_0 t_2^2 + a_1 t_2^3 + a_2 t_2^4 + a_3 t_2^5 \tag{22}$$

$$\dots\dots\dots\dots\dots \tag{23}$$

$$x(t_f) = x_0 + \dot{x}_0 t_f + \frac{1}{2}\ddot{x}_0 t_f^2 + a_1 t_f^3 + a_2 t_f^4 + a_3 t_f^5 \tag{24}$$

Polynomial Trajectories in Matrix Form

In matrix form, we have

$$x(t) = \begin{bmatrix} x(t_1) \\ x(t_2) \\ x(t_3) \\ \dots \\ x(t_f) \end{bmatrix} = \mathbf{A}_x \widetilde{\mathbf{c}_x} + \mathbf{P}_x \mathbf{c}_x, \tag{25}$$

where,

$$\mathbf{A}_x = \begin{bmatrix} 1.0 & t_1 & 0.5t_1^2 \\ 1.0 & t_2 & 0.5t_2^2 \\ 1.0 & t_3 & 0.5t_3^2 \\ \dots\dots & & \\ 1.0 & t_f & 0.5t_f^2 \end{bmatrix}, \widetilde{\mathbf{c}}_x = \begin{bmatrix} x_0 \\ \dot{x}_0 \\ \ddot{x}_0 \end{bmatrix} \tag{26}$$

$$\mathbf{P}_x = \begin{bmatrix} t_1^3 & t_1^4 & t_1^5 \\ t_2^3 & t_2^4 & t_2^5 \\ t_3^3 & t_3^4 & t_3^5 \\ \dots\dots & & \\ t_f^3 & t_f^4 & t_f^5 \end{bmatrix}, \mathbf{c}_x = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{27}$$

Polynomial Trajectories in Matrix Form

Similarly, we get

$$\dot{x}(t) = \begin{bmatrix} \dot{x}(t_1) \\ \dot{x}(t_2) \\ \dot{x}(t_3) \\ \dots \\ \dot{x}(t_f) \end{bmatrix} = \dot{\mathbf{A}}_x \widetilde{\mathbf{c}_x} + \dot{\mathbf{P}}_x \mathbf{c}_x, \tag{28}$$

where,

$$\dot{\mathbf{A}}_x = \begin{bmatrix} 0.0 & 1.0 & t_1 \\ 0.0 & 1.0 & t_2 \\ 0.0 & 1.0 & t_3 \\ \dots\dots & & \\ 0.0 & 1.0 & t_4 \end{bmatrix}, \widetilde{\mathbf{c}}_x = \begin{bmatrix} x_0 \\ \dot{x}_0 \\ \ddot{x}_0 \end{bmatrix} \tag{29}$$

$$\dot{\mathbf{P}}_x = \begin{bmatrix} 3t_1^2 & 4t_1^3 & 5t_1^4 \\ 3t_2^2 & 4t_2^3 & 5t_2^4 \\ 3t_2^2 & 4t_2^3 & 5t_2^4 \\ \dots\dots & & \\ 3t_f^2 & 4t_f^3 & 5t_f^4 \end{bmatrix}, \mathbf{c}_x = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{30}$$

Polynomial Trajectories in Matrix Form

Similarly, we get

$$\ddot{x}(t) = \begin{bmatrix} \ddot{x}(t_1) \\ \ddot{x}(t_2) \\ \ddot{x}(t_3) \\ \dots \\ \ddot{x}(t_f) \end{bmatrix} = \ddot{\mathbf{A}}_x \widetilde{\mathbf{c}_x} + \ddot{\mathbf{P}}_x \mathbf{c}_x, \tag{31}$$

where,

$$\ddot{\mathbf{A}}_x = \begin{bmatrix} 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 1.0 \\ \dots\dots & & \\ 0.0 & 0.0 & 1.0 \end{bmatrix}, \widetilde{\mathbf{c}_x} = \begin{bmatrix} x_0 \\ \dot{x}_0 \\ \ddot{x}_0 \end{bmatrix} \tag{32}$$

$$\ddot{\mathbf{P}}_x = \begin{bmatrix} 6t_1 & 12t_1^2 & 20t_1^4 \\ 6t_2 & 12t_2^2 & 20t_2^4 \\ 6t_3 & 12t_3^2 & 20t_3^4 \\ \dots\dots & & \\ 6t_f & 12t_f^2 & 20t_f^4 \end{bmatrix}, \mathbf{c}_x = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{33}$$

Point to Point Trajectory in Matrix Form

$$\overbrace{\begin{bmatrix} x(t_f) \\ \dot{x}(t_f) \\ \ddot{x}(t_f) \end{bmatrix}}^{\mathbf{s}_{end}} = \overbrace{\begin{bmatrix} {}^{n}\mathbf{A}_x \\ {}^{n}\dot{\mathbf{A}}_x \\ {}^{n}\ddot{\mathbf{A}}_x \end{bmatrix}}^{\mathbf{G}_{end}} \widetilde{\mathbf{c}}_x + \overbrace{\begin{bmatrix} {}^{n}\mathbf{P}_x \\ {}^{n}\dot{\mathbf{P}}_x \\ {}^{n}\ddot{\mathbf{P}}_x \end{bmatrix}}^{\mathbf{H}_{end}} \mathbf{c}_x \tag{34}$$

Solution is given by

$$\mathbf{c}_x = \mathbf{H}_{end}^{-1}(\mathbf{s}_{end} - \mathbf{G}_{end}\widetilde{\mathbf{c}}_x) \tag{35}$$

Polynomial Form for a 7^{th} degree polynomial

$$x(t) = \begin{bmatrix} x(t_1) \\ x(t_2) \\ x(t_3) \\ \dots \\ x(t_f) \end{bmatrix} = \mathbf{A}_x \widetilde{\mathbf{c}_x} + \mathbf{P}_x \mathbf{c}_x, \tag{36}$$

where,

$$\mathbf{A}_x = \begin{bmatrix} 1.0 & t_1 & 0.5t_1^2 \\ 1.0 & t_2 & 0.5t_2^2 \\ 1.0 & t_3 & 0.5t_3^2 \\ \dots\dots & & \\ 1.0 & t_f & 0.5t_f^2 \end{bmatrix}, \widetilde{\mathbf{c}_x} = \begin{bmatrix} x_0 \\ \dot{x}_0 \\ \ddot{x}_0 \end{bmatrix} \tag{37}$$

$$\mathbf{P}_x = \begin{bmatrix} t_1^3 & t_1^4 & t_1^5 & t_1^6 & t_1^7 \\ t_2^3 & t_2^4 & t_2^5 & t_2^6 & t_2^7 \\ t_3^3 & t_3^4 & t_3^5 & t_3^6 & t_3^7 \\ \dots\dots & & & & \\ t_f^3 & t_f^4 & t_f^5 & t_f^6 & t_f^7 \end{bmatrix}, \mathbf{c}_x = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} \tag{38}$$

Solving for the coefficients of a 7^{th} order polynomial

We can follow a similar procedure to derive the $\dot{\mathbf{P}}_x, \ddot{\mathbf{P}}_x$ for the 7^{th} order polynomial.
Lets now re-write the linear equation for solving the 7^{th} order polynomial.

$$\overbrace{\begin{bmatrix} x(t_f) \\ \dot{x}(t_f) \\ \ddot{x}(t_f) \end{bmatrix}}^{\mathbf{s}_{end}} = \overbrace{\begin{bmatrix} {}^n\mathbf{A}_x \\ {}^n\dot{\mathbf{A}}_x \\ {}^n\ddot{\mathbf{A}}_x \end{bmatrix}}^{\mathbf{G}_{end}} \widetilde{\mathbf{c}}_x + \overbrace{\begin{bmatrix} {}^n\mathbf{P}_x \\ {}^n\dot{\mathbf{P}}_x \\ {}^n\ddot{\mathbf{P}}_x \end{bmatrix}}^{\mathbf{H}_{end}} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} \tag{39}$$

We cannot solve the above equation since the matrix $\mathbf{H}_{end}$ is not square.

Incorporating intermediate way-point trajectory

We need to append the rows corresponding to intermediate points. Suppose, the intermediate points happen at time t_i, t_j. So, we need to extract row i and j from matrix $\mathbf{A}_x$ and $\mathbf{P}_x$.

$$\overbrace{\begin{bmatrix} \mathbf{s}_{mid} \\ \mathbf{s}_{end} \end{bmatrix}}^{\mathbf{s}} = \overbrace{\begin{bmatrix} \mathbf{G}_{mid} \\ \mathbf{G}_{end} \end{bmatrix}}^{\mathbf{G}} \widetilde{\mathbf{c}}_x + \overbrace{\begin{bmatrix} \mathbf{H}_{mid} \\ \mathbf{H}_{end} \end{bmatrix}}^{\mathbf{H}} \mathbf{c}_x \tag{40}$$

$$\mathbf{G}_{mid} = \begin{bmatrix} {}^{i}\mathbf{A}_x \\ {}^{j}\mathbf{A}_x \end{bmatrix}, \mathbf{H}_{mid} = \begin{bmatrix} {}^{i}\mathbf{P}_x \\ {}^{j}\mathbf{P}_x \end{bmatrix}, \mathbf{s}_{mid} = \begin{bmatrix} x(t_1) \\ x(t_2) \end{bmatrix} \tag{41}$$

Solution is given by

$$\mathbf{c}_x = \mathbf{H}^{-1}(\mathbf{s} - \mathbf{G}\widetilde{\mathbf{c}}_x) \tag{42}$$

Linear Equation Solving is not Enough

- Computing trajectory coefficients through solving linear equations cannot be used when the matrix $\mathbf{H}$ is not invertible. This in turn can happen when the trajectory polynomial has more coefficients than specified equations.
- Using higher degree polynomial with more trajectory coefficients gives us a great deal of flexibility but then we have to find out of way of computing those coefficients.
- One way of computing trajectory coefficients is through the optimization route.

Introducing Optimality

- We want the robots to behave in a particular way and not just go from point A to point B
- The most intuitive way of describing robot behavior is in terms of cost functions. For example, move with a particular reference velocity as much as possible.
- Most trajectory planning problems are framed as optimization problems

Computing minimum of functions

$$\min f(s) = s^2 - 6s + 4 \tag{43}$$

To solve this minimization, we take the derivative with respect to x and equate it to zero

$$2s - 6 = 0 \tag{44}$$

Multi variable function

$$\min f(s_1, s_2) = 2s_1^2 + 2s_1 s_2 + 2s_2^2 - 6s_1 \tag{45}$$

We compute partial derivatives and equate them to zero

$$\bigtriangledown f_{s_1} = 4s_1 + 2s_2 - 6 = 0 \tag{46}$$
$$\bigtriangledown f_{s_2} = 2s_1 + 4s_2 = 0 \tag{47}$$
$$\tag{48}$$

Multi-variable quadratic function in Matrix form

$$f(s_1, s_2) = 2s_1^2 + 2s_1 s_2 + 2s_2^2 - 6s_1$$

$$\Rightarrow f(s_1, s_2) = \frac{1}{2}\begin{bmatrix} s_1 \\ s_2 \end{bmatrix}^T \overbrace{\begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}}^{\mathbf{Q}} \overbrace{\begin{bmatrix} s_1 \\ s_2 \end{bmatrix}}^{\mathbf{s}} + \underbrace{\begin{bmatrix} -6 \\ 0 \end{bmatrix}^T}_{\mathbf{q}} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$$

$$f(s_1, s_2) = \frac{1}{2}\mathbf{s}^T \mathbf{Q}\mathbf{s} + \mathbf{q}^T \mathbf{s} \tag{49}$$

The derivative (or gradient) in matrix form is given by

$$\bigtriangledown f = \mathbf{Q}\mathbf{s} + \mathbf{q} = 0 \Rightarrow \mathbf{s} = -\mathbf{Q}^{-1}\mathbf{q} \tag{50}$$

Multiple Quadratic Costs

$$\begin{aligned}\frac{1}{2}\mathbf{s}^T\mathbf{Q}_1\mathbf{s} + \mathbf{q}_1^T\mathbf{s} + \frac{1}{2}\mathbf{s}^T\mathbf{Q}_2\mathbf{s} + \mathbf{q}_2^T\mathbf{s}\\ = \frac{1}{2}\mathbf{s}^T\mathbf{Q}\mathbf{s} + \mathbf{q}^T\mathbf{s}\\ \mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2, \mathbf{q} = \mathbf{q}_1 + \mathbf{q}_2\end{aligned} \tag{51}$$

The derivative (or gradient) in matrix form is given by

$$\mathbf{Q}\mathbf{s} + \mathbf{q} = 0 \Rightarrow \mathbf{s} = -\mathbf{Q}^{-1}\mathbf{q} \tag{52}$$

Least Squares Cost

Consider the following cost function where a_{ij}, d_i are given constants.

$$\min f(s_1, s_2) = \frac{1}{2}((a_{11}s_1 + a_{12}s_2 - d_1)^2 + (a_{21}s_1 + a_{22}s_2 - d_2)^2 + (a_{31}s_1 + a_{32}s_2 - d_3)^2) \tag{53}$$

$$\begin{aligned} f(s_1, s_2) = \frac{1}{2} \| \overbrace{\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}}^{\mathbf{A}} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} - \overbrace{\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}}^{\mathbf{b}} \|_2^2 \\ f(s_1, s_2) = \mathbf{s}^T\mathbf{Q}\mathbf{s} + \mathbf{q}^T\mathbf{s} \\ \mathbf{Q} = \mathbf{A}^T\mathbf{A}, \mathbf{q} = -\mathbf{A}^T\mathbf{b} \end{aligned} \tag{54}$$

Again, solution is given by

$$\mathbf{Q}\mathbf{s} + \mathbf{q} = 0 \Rightarrow \mathbf{s} = -\mathbf{Q}^{-1}\mathbf{q} \tag{55}$$

Constructing Cost Functions

Suppose, you want to choose s_1, s_2 such that their sum is as close as possible to 1 and s_2 is as close as possible to 0.5

Constructing Cost Functions

Suppose, you want to choose s_1, s_2 such that their sum is as close as possible to 1 and s_2 is as close as possible to 0.5

$$f(s_1, s_2) = (s_1 + s_2 - 1)^2 + (s_2 - 0.5)^2 \tag{56}$$

Constructing Cost Functions

Suppose, you want to choose s_1, s_2 such that their sum is as close as possible to 1 and magnitude of s_2 is as small as possible

Constructing Cost Functions

Suppose, you want to choose s_1, s_2 such that their sum is as close as possible to 1 and magnitude of s_2 is as small as possible

$$f(s_1, s_2) = (s_1 + s_2 - 1)^2 + (s_2)^2 \tag{57}$$

Constructing Cost function

Construct a cost function such that the end position, velocity, acceleration is as close as possible to $x_f, \dot{x}_f, \ddot{x}_f$

$$f_{end} = \frac{1}{2}((x(t_f) - x_f)^2 + (\dot{x}(t_f) - \dot{x}_f)^2 + (\ddot{x}(t_f) - \ddot{x}_f)^2) \tag{58}$$

Constructing Cost function

Construct a cost function to keep the acceleration as small as possible at all times

$$f_{acc} = \frac{1}{2}\sum_t \ddot{x}(t)^2 \tag{59}$$

$$f_{acc} = \frac{1}{2}\sum_k ({}^k\ddot{\mathbf{A}}_x \widetilde{\mathbf{c}}_x + {}^k\ddot{\mathbf{P}}_x \mathbf{c}_x)^2$$

$$f_{acc} = \frac{1}{2}\|\ddot{\mathbf{A}}_x \widetilde{\mathbf{c}}_x + \ddot{\mathbf{P}}_x \mathbf{c}_x\|_2^2 \tag{60}$$

$$f_{acc} = \frac{1}{2}\mathbf{c}_x^T \mathbf{Q}_{acc} \mathbf{c}_x + \mathbf{q}_{acc}^T \mathbf{c}_x, \tag{61}$$

where

$$\mathbf{Q}_{acc} = \ddot{\mathbf{P}}_x^T \ddot{\mathbf{P}}_x, \mathbf{q}_{acc} = \ddot{\mathbf{P}}_x^T (\ddot{\mathbf{A}}_x \widetilde{\mathbf{c}}_x) \tag{62}$$

Constructing Cost function

Construct a cost function to ensure trajectory passes as close as possible to the intermediate points x_1 and x_2

$$\begin{aligned} f_{mid} &= \frac{1}{2}((x(t_1) - x_1)^2 + (x(t_2) - x_2)^2) \\ f_{mid} &= \frac{1}{2}\|\mathbf{H}_{mid}\mathbf{c}_x + \mathbf{G}_{mid}\widetilde{\mathbf{c}}_x - \mathbf{s}_{mid}\|_2^2 \end{aligned} \tag{63}$$

$$f_{mid} = \frac{1}{2}\mathbf{c}_x^T\mathbf{Q}_{mid}\mathbf{c}_x + \mathbf{q}_{mid}^T\mathbf{c}_x, \tag{64}$$

where

$$\mathbf{Q}_{mid} = \mathbf{H}_{mid}^T\mathbf{H}_{mid}, \mathbf{q}_{mid} = -\mathbf{H}_{mid}^T(\mathbf{s}_{mid} - \mathbf{G}_{mid}\widetilde{\mathbf{c}}_x) \tag{65}$$

Constructing Cost Functions

Adding the final goal cost, acceleration cost and mid-point cost together.

$$f = \frac{1}{2}((x(t_f) - x_f)^2 + (\dot{x}(t_f) - \dot{x}_f)^2 + (\ddot{x}(t_f) - \ddot{x}_f)^2) + \frac{1}{2}\sum_t \ddot{x}(t)^2 + \frac{1}{2}((x(t_1) - x_1)^2 + (x(t_2) - x_2)^2) \tag{66}$$

$$\begin{aligned} f = f_{end} + f_{mid} + f_{acc} &= \frac{1}{2}\mathbf{c}_x^T\mathbf{Q}_{end}\mathbf{c}_x + \mathbf{q}_{end}^T\mathbf{c}_x + \frac{1}{2}\mathbf{c}_x^T\mathbf{Q}_{mid}\mathbf{c}_x + \mathbf{q}_{mid}^T\mathbf{c}_x + \frac{1}{2}\mathbf{c}_x^T\mathbf{Q}_{acc}\mathbf{c}_x + \mathbf{q}_{acc}^T\mathbf{c}_x \\ f &= \frac{1}{2}\mathbf{c}_x^T\mathbf{Q}\mathbf{c}_x + \mathbf{q}^T\mathbf{c}_x, \mathbf{Q} = \mathbf{Q}_{end} + \mathbf{Q}_{mid} + \mathbf{Q}_{acc}, \mathbf{q} = \mathbf{q}_{end} + \mathbf{q}_{mid} + \mathbf{q}_{acc} \end{aligned} \tag{67}$$

Constructing Cost Functions Contd..

Construct a cost function to ensure that (i) $x(t)$ and $y(t)$ are as close *as possible* to the desired path $(\mathbf{x}_d(t), \mathbf{y}_d(t))$ *at all times* and (ii) acceleration is as small as possible at all times

$$f(\mathbf{c}_x) = \frac{1}{2}w_1\sum_t(\mathbf{x}(t) - \mathbf{x}_d(t))^2 + \frac{1}{2}w_2\sum_t \ddot{x}(t)^2 \tag{68}$$

$$= \frac{1}{2}w_1\|\mathbf{A}_x\widetilde{\mathbf{c}}_x + \mathbf{P}_x\mathbf{c}_x - x_d(t)\|_2^2 + \frac{1}{2}w_2\|\ddot{\mathbf{A}}_x\widetilde{\mathbf{c}}_x + \ddot{\mathbf{P}}_x\mathbf{c}_x\|_2^2 \tag{69}$$

We get similar expressions for y component

$$f(\mathbf{c}_y) = \frac{1}{2}w_1\sum_t(\mathbf{y}(t) - \mathbf{y}_d(t))^2 + \frac{1}{2}w_2\sum_t \ddot{y}(t)^2$$

$$\frac{1}{2}\|\mathbf{A}_y\widetilde{\mathbf{c}}_y + \mathbf{P}_y\mathbf{c}_y - y_d(t)\|_2^2 + \frac{1}{2}\|\ddot{\mathbf{A}}_y\widetilde{\mathbf{c}}_y + \ddot{\mathbf{P}}_y\mathbf{c}_y\|_2^2 \tag{70}$$

The trajectories are computed by solving the following two optimization problem

$$\arg\min f(\mathbf{c}_x), \arg\min f(\mathbf{c}_y) \tag{71}$$

Modeling Hard/Strict constraints in Trajectory Planning

- Modeling robot behaviors through cost functions has a downside: the weights needs to be tuned to achieve the desired behavior.
- Strict requirements like stopping at the final position or bounds in velocity, acceleration can be handled alternately through hard constraints leading to constrained optimization problem.

Minimizing a quadratic function with Linear Constraints

$$\min s_1^2 - s_2 \tag{72}$$
$$s_1 + s_2 = 3.0 \tag{73}$$

Steps

- Eliminate s_1 from (73), i.e $s_1 = 3 - s_2$
- Substituting it back to (72), we get $\min(3 - s_2)^2 - s_2$

For functions of several variables, we essentially need a general matrix version of the above two steps.

Constrained Optimization with Equality Constraints

$$\arg\min \frac{1}{2}\mathbf{c}_x^T\mathbf{Q}\mathbf{c}_x + \mathbf{q}^T\mathbf{c}_x \tag{74}$$

$$\mathbf{M}\mathbf{c}_x = \mathbf{n} \tag{75}$$

The solution to the above optimization problem is given in two parts

$$\boldsymbol{\lambda} = -(\mathbf{M}\mathbf{Q}^{-1}\mathbf{M}^T)^{-1}(\mathbf{n} + \mathbf{M}\mathbf{Q}^{-1}\mathbf{q}) \tag{76}$$

$$\mathbf{c}_x = -\mathbf{Q}^{-1}(\mathbf{M}^T\boldsymbol{\lambda} + \mathbf{q}) \tag{77}$$

Constrained Optimization with Equality Constraints: Using Solver

$$\arg\min \frac{1}{2}\mathbf{c}_x^T\mathbf{Q}\mathbf{c}_x + \mathbf{q}^T\mathbf{c}_x \tag{78}$$

$$\mathbf{M}\mathbf{c}_x = \mathbf{n} \tag{79}$$

We can just give the matrices and vectors to the solver to get a solution

$$\mathbf{c}_x = \text{CVXOPT}(\mathbf{Q}, \mathbf{q}, \mathbf{M}, \mathbf{n}) \tag{80}$$

Hard Constraints

Example: End position, velocity and acceleration constraints

$$x(t_f) = x_f \tag{81}$$

$$^n\mathbf{A}_x\widetilde{\mathbf{c}}_x + {}^n\mathbf{P}_x\mathbf{c}_x = x_f \tag{82}$$

$$\dot{x}(t_f) = \dot{x}_f \tag{83}$$

$$^n\dot{\mathbf{A}}_x\widetilde{\mathbf{c}}_x + {}^n\dot{\mathbf{P}}_x\mathbf{c}_x = \dot{x}_f \tag{84}$$

$$\ddot{x}(t_f) = \ddot{x}_f \tag{85}$$

$$^n\ddot{\mathbf{A}}_x\widetilde{\mathbf{c}}_x + {}^n\ddot{\mathbf{P}}_x\mathbf{c}_x = \ddot{x}_f \tag{86}$$

$$\begin{bmatrix} ^n\mathbf{A}_x\widetilde{\mathbf{c}}_x + {}^n\mathbf{P}_x\mathbf{c}_x \\ ^n\dot{\mathbf{A}}_x\widetilde{\mathbf{c}}_x + {}^n\dot{\mathbf{P}}_x\mathbf{c}_x \\ ^n\ddot{\mathbf{A}}_x\widetilde{\mathbf{c}}_x + {}^n\ddot{\mathbf{P}}_x\mathbf{c}_x \end{bmatrix} = \begin{bmatrix} x_f \\ \dot{x}_f \\ \ddot{x}_f \end{bmatrix} \tag{87}$$

$$\mathbf{M} = \mathbf{H}_{end} = \begin{bmatrix} ^n\mathbf{P}_x \\ \dot{\mathbf{P}}_x \\ ^n\ddot{\mathbf{P}}_x \end{bmatrix}, \mathbf{n} = \mathbf{s}_{end} - \mathbf{G}_{end}\widetilde{\mathbf{c}}_x = \begin{bmatrix} x_f \\ \dot{x}_f \\ \ddot{x}_f \end{bmatrix} - \begin{bmatrix} ^n\mathbf{A}_x\widetilde{\mathbf{c}}_x \\ ^n\dot{\mathbf{A}}_x\widetilde{\mathbf{c}}_x \\ ^n\ddot{\mathbf{A}}_x\widetilde{\mathbf{c}}_x \end{bmatrix} \tag{88}$$

Constrained Optimization with Hard Inequality Constraints

$$\arg\min \frac{1}{2}\mathbf{c}_x^T\mathbf{Q}\mathbf{c}_x + \mathbf{q}^T\mathbf{c}_x \tag{89}$$

$$\mathbf{M}_{ineq}\mathbf{c}_x \leq \mathbf{n}_{ineq} \tag{90}$$

Example: bounds on velocity, acceleration

$$v_x^{min} \leq \dot{\mathbf{A}}_x\widetilde{\mathbf{c}}_x + \dot{\mathbf{P}}_x\mathbf{c}_x \leq v_x^{max} \tag{91}$$

$$a_x^{min} \leq \dot{\mathbf{A}}_{\dot{x}}\widetilde{\mathbf{c}}_x + \ddot{\mathbf{P}}_x\mathbf{c}_x \leq a_x^{max} \tag{92}$$

$$\tag{93}$$

$$\mathbf{M}_{ineq} = \begin{bmatrix} \dot{\mathbf{P}}_x \\ -\dot{\mathbf{P}}_x \\ \ddot{\mathbf{P}}_x \\ -\ddot{\mathbf{P}}_x \end{bmatrix}, \mathbf{n}_{ineq} = \begin{bmatrix} v_x^{max} - \dot{\mathbf{A}}_x\widetilde{\mathbf{c}}_x \\ -v_x^{min} + \dot{\mathbf{A}}_x\widetilde{\mathbf{c}}_x \\ a_x^{max} - \ddot{\mathbf{A}}_x\widetilde{\mathbf{c}}_x \\ -a_x^{min} + \ddot{\mathbf{A}}_x\widetilde{\mathbf{c}}_x \end{bmatrix} \tag{94}$$