Robotics: Science & Systems
[Topic 5: Dynamics]

Prof. Sethu Vijayakumar

Course webpage: http://wcms.inf.ed.ac.uk/ipab/rss
Dynamics vs. Kinematics

- So far we always assumed we could arbitrarily set the next robot state $q_{t+1}$ (or an arbitrary velocity $\dot{q}_t$). What if we can’t?

Examples:

- **A flying airplane**: You cannot command it to hold still in the air or move straight up
- **A car**: You cannot command it to move sideways
- **Your arm**: You cannot command it to throw a ball with arbitrary velocity (force limits)
- **A torque controlled robot**: You cannot command it to instantaneously change velocity (infinite acceleration/torque)

What all examples have in common:

- One can set **controls** $u_t$ (airplane’s control stick, car’s steering wheel, your muscles activations, torque/voltage/current send to a robot’s motors)
- But these controls only indirectly influence the **dynamics of state**, $x_{t+1} = f(x_t, u_t)$
Dynamics

• The **dynamics** of a system describes how the controls \( u_t \) influence the change-of-state of the system

\[
x_{t+1} = f(x_t, u_t)
\]

– The notation \( x_t \) refers to the **dynamic state** of the system: e.g., joint positions *and velocities* \( x_t = (q_t, \dot{q}_t) \).
– \( f \) is an arbitrary function, often smooth

• We define a **nonholonomic system** as one with **differential constraints**:

\[
\dim(u_t) < \dim(x_t)
\]

\[\Rightarrow \text{Not all degrees of freedom are directly controllable} \]
Outline

• We discuss three different basic examples
  – 1D point mass
  – A general dynamic robot
  – A non-holonomic car model

• We discuss how previous methods can be extended to the dynamic/non-holonomic case:
  – Inv. Kinematics Control → Optimal Operational Space Control
  – Trajectory Optimisation → Trajectory Optimisation
  – RRTs → RRTs with special tricks
1-D Mass

- Start with the simplest possible example: 1D point mass (no gravity, no friction, just a single mass)

\[ x(t) = (q(t), \dot{q}(t)) \] is described by:
- position \( q(t) \in \mathbb{R} \)
- velocity \( \dot{q}(t) \in \mathbb{R} \)

- The **controls** \( u(t) \) is the force we apply on the mass point

- The **system dynamics** is:

\[ \ddot{q}(t) = u(t)/m \]
1D Mass: Proportional Control

Assume current position is $q$.
Goal is to move to $q^*$.  
What can we do?

- **IDEA 1**
  
  "Always pull the mass towards the goal $q^*$:"

  $$u = K_p (q^* - q)$$
1D Mass: Proportional Control

What is the effect of IDEA 1? \( m \ddot{q} = u = K_p (q^* - q) \)

\( q = q(t) \) is a function of time, this is a second order differential eq.

- Solution: assume \( q(t) = a + be^{\omega t} \)
  (an “non-imaginary” alternative would be \( q(t) = a + b e^{-\lambda t} \cos(\omega t) \))

\[
m b \omega^2 e^{\omega t} = K_p q^* - K_p a - K_p b e^{\omega t} \\
(m b \omega^2 + K_p b) e^{\omega t} = K_p (q^* - a) \\
\Rightarrow (m b \omega^2 + K_p b) = 0 \land (q^* - a) = 0 \\
\Rightarrow \omega = i \sqrt{K_p/m} \\
q(t) = q^* + b e^{i \sqrt{K_p/m} t}
\]

This is an oscillation around \( q^* \) with amplitude \( b = q(0) - q^* \) and frequency \( \sqrt{K_p/m} \)!
1D Mass: Proportional Control

What’s the effect of IDEA 1?

\[ m \ddot{q} = u = K_p (q^* - q) \]

\[ q(t) = q^* + b e^{i \sqrt{K_p/m} t} \]

This is an oscillation around \( q^* \) with amplitude \( b = q(0) - q^* \) and frequency \( \sqrt{K_p/m} \)!
That didn’t solve the problem!

- **IDEA 2**
  
  "Pull less, when we’re heading the right direction already:"
  "Damp the system:"

  \[ u = K_p (q^* - q) + K_d (\dot{q}^* - \dot{q}) \]

  \( \dot{q}^* \) is a desired goal velocity

  For simplicity we set \( \dot{q}^* = 0 \) in the following.
1D Mass: Derivative Feedback

What is the effect of IDEA 2?

$$m \ddot{q} = u = K_p (q^* - q) + K_d (0 - \dot{q})$$

- Solution: again assume $q(t) = a + be^{\omega t}$

$$m b \omega^2 e^{\omega t} = K_p q^* - K_p a - K_p b e^{\omega t} - K_d b \omega e^{\omega t}$$

$$(m b \omega^2 + K_d b \omega + K_p b) e^{\omega t} = K_p (q^* - a)$$

$$\Rightarrow (m \omega^2 + K_d \omega + K_p) = 0 \land (q^* - a) = 0$$

$$\Rightarrow \omega = \frac{-K_d \pm \sqrt{K_d^2 - 4mK_p}}{2m}$$

$$q(t) = q^* + b e^{\omega t}$$

The term $-\frac{K_d}{2m}$ in $\omega$ is real $\Leftrightarrow$ exponential decay (damping)
1D Mass: Derivative Feedback

What’s the effect of IDEA 2?

\[ q(t) = q^* + b e^{\omega t}, \quad \omega = \frac{-K_d \pm \sqrt{K_d^2 - 4mK_p}}{2m} \]

- Effect of the second term \( \sqrt{K_d^2 - 4mK_p}/2m \) in \( \omega \):

  \[
  \begin{align*}
  K_d^2 < 4mK_p & \implies \omega \text{ has imaginary part} \\
  & \text{oscillating with frequency } \sqrt{K_p/m - K_d^2/4m^2} \\
  q(t) = q^* + be^{-K_d/2m t} e^{i\sqrt{K_p/m - K_d^2/4m^2} t}
  \end{align*}
  \]

  \[
  \begin{align*}
  K_d^2 > 4mK_p & \implies \omega \text{ real} \\
  & \text{strongly damped}
  \end{align*}
  \]

  \[
  \begin{align*}
  K_d^2 = 4mK_p & \implies \text{second term zero} \\
  & \text{only exponential decay}
  \end{align*}
  \]
1D Mass: Derivative Feedback

- **Under-damped** or **Oscillatory Damped**
- **Over-damped**
- **Critically Damped**
Alternate Parameterisation

Instead of specifying the gains $K_p$ and $K_d$

- “wave length” $\lambda = \frac{1}{\omega_0} = \frac{1}{\sqrt{K_p/m}}$, $K_p = m/\lambda^2$
- damping ratio $\xi = \frac{K_d}{\sqrt{4mK_p}} = \frac{\lambda K_d}{2m}$, $K_d = 2m\xi/\lambda$

$\xi > 1$: over-damped
$\xi = 1$: critically damped
$\xi < 1$: oscillatory-damped

$q(t) = q^* + be^{-\xi/\lambda t} e^{i\omega_0 \sqrt{1-\xi^2}} t$
1D Mass: Integral Feedback

- **IDEA 3**
  
  "Pull if the position error accumulated large in the past:"

  \[
  u = K_p(q^* - q) + K_d(\dot{q}^* - \dot{q}) + K_i \int_{s=0}^{t} (q^*(s) - q(s)) \, ds
  \]

- This is *not a linear control law* (not linear w.r.t. \((q, \dot{q})\))
1D Mass: PID Control

\[ u = K_p(q^* - q) + K_d(\dot{q}^* - \dot{q}) + K_i \int_{s=0}^{t} (q^*(s) - q(s)) \, ds \]

- **PID control**
  - Proportional Control ("Position Control")
    \[ f \propto K_p(q^* - q) \]
  - Derivative Control ("Damping")
    \[ f \propto K_d(\dot{q}^* - \dot{q}) \quad (\ddot{x}^* = 0 \rightarrow \text{damping}) \]
  - Integral Control ("Steady State Error")
    \[ f \propto K_i \int_{s=0}^{t} (q^*(s) - q(s)) \, ds \]
1D Mass: Lessons Learnt

• Proportional and derivative feedback (PD control) are like adding a spring and damper to the point mass

• PD control is a linear control law

\[ \pi : (q, \dot{q}) \mapsto u = K_p(q^* - q) + K_d(q^* - \dot{q}) \]

(linear in the dynamic system state \( x = (q, \dot{q}) \))

• With such linear control laws, we can design approach trajectories (by tuning the gains)
  – but no optimality criterion behind such motions yet
1D Mass: Lessons Learnt

• The point mass is a basic example of a non-holonomic system

  – The state is $x = (q, \dot{q}) \in \mathbb{R}^2$
  – The control $u \in \mathbb{R}$ is the force
  – The dynamics are $\begin{pmatrix} \ddot{q} \\ \ddot{q} \end{pmatrix} = f(u, \begin{pmatrix} q \\ \dot{q} \end{pmatrix}) = \begin{pmatrix} \dot{q} \\ u/m \end{pmatrix}$
Outline

- We discuss three different basic examples
  - 1D point mass
    - A general dynamic robot
    - A non-holonomic car model
- We discuss how previous methods can be extended to the dynamic/non-holonomic case:
  - Inv. Kinematics Control $\rightarrow$ Optimal Operational Space Control
  - Trajectory Optimisation $\rightarrow$ Trajectory Optimisation
  - RRTs $\rightarrow$ RRTs with special tricks
General Robot System Dynamics

- **State** $x = (q, \dot{q}) \in \mathbb{R}^{2n}$
  - joint positions $q \in \mathbb{R}^n$
  - joint velocities $\dot{q} \in \mathbb{R}^n$

- **Controls** $u \in \mathbb{R}^n$ are the *torques* generated in each motor.

- The **system dynamics** are:

\[
M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + G(q) = u
\]

- $M(q) \in \mathbb{R}^{n \times n}$ is positive definite inertia matrix
  (can be inverted \(\rightarrow\) forward simulation of dynamics)

- $C(q, \dot{q}) \in \mathbb{R}^n$ are the centripetal and coriolis forces

- $G(q) \in \mathbb{R}^n$ are the gravitational forces

- $u$ are the joint torques

- More compact:

\[
M(q) \ddot{q} + F(q, \dot{q}) = u
\]
Computing $M(q)$ and $F(q, \dot{q})$

$$M(q) \ddot{q} + F(q, \dot{q}) = u$$

- **Recall:**
  We have an algorithm to compute all positions and orientations given $q$:
  $$T_{W \rightarrow i}(q) = T_{W \rightarrow A} T_{A \rightarrow A'}(q) T_{A' \rightarrow B} T_{B \rightarrow B'}(q) T_{B' \rightarrow C} T_{C \rightarrow C'}(q) T_{C' \rightarrow i}$$

  This can be similarly extended to compute velocities and accelerations (including angular) for any link. Then, one can employ the Newton-Euler formulation to compute the M and F terms for any $q$.

  **Basic idea of Newton-Euler Algorithm:**
  - Force and Torque balance at every link
  - Recursion through all links

- **Bottom line:**
  There exist algorithms to efficiently compute $M(q)$ and $F(q, \dot{q})$ for any dynamic state $(q, \dot{q})$. 
Controlling a Robot: Joint Space Method

- If we know the desired $\ddot{q}^*$ for each joint, the eqn.
  \[ M(q) \ddot{q}^* + F(q, \dot{q}) = u^* \]
gives the desired torques.

- Where could we get the desired $\ddot{q}^*$ from?
  Assume we have a nice smooth reference trajectory $q_{0:T}^{ref}$ (generated with some motion profile or alike), we can at each $t$ read off the desired acceleration as
  \[ \ddot{q}_t^{ref} := \frac{1}{\tau} \left[ (q_{t+1} - q_t)/\tau - (q_t - q_{t-1})/\tau \right] = \left( q_{t-1} + q_{t+1} - 2q_t \right)/\tau^2 \]

  However, tiny errors in acceleration will accumulate greatly over time and this makes this an unstable approach!
Controlling a Robot: Joint Space Method

- If we know the desired $\ddot{q}^*$ for each joint, the eqn.
  \[ M(q) \ddot{q}^* + F(q, \dot{q}) = u^* \]
gives the desired torques.

Choose a desired acceleration $\dddot{q}_t^*$ that implies a PD-like behavior around the reference trajectory!

\[ \dddot{q}_t^* = \dddot{q}_t^\text{ref} + K_p (q_t^\text{ref} - q_t) + K_d (\dddot{q}_t^\text{ref} - \dddot{q}_t) \]

This is a standard and convenient way of tracking a reference trajectory when the \textbf{robot dynamics are known}: all the joints will behave exactly like a 1D point mass around the reference trajectory!
Controlling a Robot: Operational Space

- Recall the inverse kinematics problem:
  - We know the desired step $\delta y^*$ (or velocity $\dot{y}^*$) of the endeffector.
  - Which step $\delta q$ (or velocities $\dot{q}$) should we make in the joints?

- Equivalent dynamic problem:
  - We know how the desired acceleration $\ddot{y}^*$ of the endeffector.
  - What controls $u$ should we apply?
Optimal Operational Space Control

- Given current state \((q_t, \dot{q}_t)\) and \(y_t = \phi(q_t), \dot{y}_t = J\dot{q}_t\)
  - given desired \(\ddot{y}^*\)
  - compute a torque \(u\) such that

1) \(\|u\|^2\) is small \(\iff\) be lazy

2) \(\ddot{y}\) is close to \(\ddot{y}^*\) \(\iff\) accelerate endeffector

- We translate this to the objective function:

\[
 f(u) = \|u\|_H^2 + \|JM^{-1}(u - F) + \dot{J}\dot{q} - \ddot{y}^*\|_C^2
\]

note: \(\dddot{y} = \frac{d}{dt}\dot{y} = \frac{d}{dt}(J\ddot{q}) = J\dddot{q} + J\ddot{q} = JM^{-1}(u - F') + J\ddot{q}\)
Optimal Operational Space Control

\[ f(u) = \|u\|_H^2 + \|JM^{-1}(u - F) + \dot{J}\dot{q} - \ddot{y}^*\|_C^2 \]

- Optimum: (analogous derivation to kinematic case)

\[ u^* = T\#(\ddot{y}^* - \dot{J}\dot{q} + TF) \]

with \( T = JM^{-1} \), \( T\# = (T^TC + H)^{-1}T^TC \)

\( (C \to \infty \Rightarrow T\# = H^{-1}T^{\top}(TH^{-1}T^{\top})^{-1}) \)

Can be interpreted as a generalization of the kinematic pseudo-inverse to the dynamic case, taking into account the intrinsic forces (Coriolis, Gravity) as well as \( \dot{J} \).
Controlling a Robot: Operational Space

- Where could we get the desired $\ddot{y}^*$ from?
  - **Reference trajectory** $\dot{y}_{0:T}^{\text{ref}}$ in operational space!
  - **PD-like behavior** in each operational space:
    \[
    \ddot{y}_t^* = \dot{y}_t^{\text{ref}} + K_p (y_t^{\text{ref}} - y_t) + K_d (\dot{y}_t^{\text{ref}} - \dot{y}_t)
    \]

Operational space control: *Let the system behave as if we could directly “apply 1D point mass behavior” to the endeffector.*
Outline

• We discuss three different basic examples
  – 1D point mass
  – A general dynamic robot
  – A non-holonomic car model

• We discuss how previous methods can be extended to the dynamic/non-holonomic case:
  – Inv. Kinematics Control → Optimal Operational Space Control
  – Trajectory Optimisation → Trajectory Optimisation
  – RRTs → RRTs with special tricks
Example: Car

• The previous examples were all ‘dynamic’ systems in the sense \( x = (q, \dot{q}) \)

• The car example is just kinematic (no mass/inertia/forces)

• But the treatment is very much the same because a car is also a non-holonomic system!
  – (Just like a car cannot instantly move sideward, a dynamic system cannot instantly change its position \( q \): the current change in position is constrained by the current velocity)
Example: Car

Systems Equation:

\[
\begin{pmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{pmatrix} =
\begin{pmatrix}
v \cos \theta \\
v \sin \theta \\
(v/L) \tan \varphi
\end{pmatrix}
\]

State \( q = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \)

Controls \( u = \begin{pmatrix} v \\ \varphi \end{pmatrix} \)
Example: Car

• The car is a *non-holonomic* system: not all DOFs are controlled
  \[ \dim(u) < \dim(q) \]

• We have a differential constraint \( \dot{q} \):
  \[ \dot{x} \sin \theta - \dot{y} \cos \theta = 0 \]

  “A car cannot move directly laterally”

• General definition of a differential constraint:

  For any given state \( x \), let \( U_x \) be the tangent space that is generated by controls:
  \[ U_x = \{ \dot{x} : \dot{x} = f(x, u), u \in U \} \]

  (non-holonomic \( \iff \) \( \dim(U_x) < \dim(x) \))
Path finding with nonholonomic systems

• Could a car follow this trajectory?

• This is generated with a PRM in state space $q=(x,y,\theta)$ ignoring the *differential constraints*
Path finding with nonholonomic systems

• This is the solution we would like to have...

• This path respects the **differential constraints**

• Each step in the path corresponds to a setting a certain control
Path finding with nonholonomic systems

• **Control based sampling**: fulfils none of the nice exploration properties of RRTs – but respects the differential constraints
  
  1. Select a $q$ from $T$ of current configurations
  2. Pick a control vector $u$ at random
  3. Integrate equation of motion over short duration
  4. If the motion is collision free, add end point to the tree

1) Select a $q \in T$
2) Pick $\nu, \phi, \text{and } \tau$
3) Integrate motion from $q$
4) Add result if collision-free
Car Parking

Car Parking

Parking with only left steering

...with a trailer

Better control-based exploration

• RRTs with differential constraints:

  **Input:** $q_{\text{start}}$, number $k$ of nodes, time interval $\tau$
  **Output:** tree $T = (V, E)$

1: initialize $V = \{q_{\text{start}}\}$, $E = \emptyset$
2: for $i = 0 : k$ do
3:   $q_{\text{target}} \leftarrow$ random sample from $Q$
4:   $q_{\text{near}} \leftarrow$ nearest neighbor of $q_{\text{target}}$ in $V$
5:   use local planner to compute controls $u$ that steer $q_{\text{near}}$ towards $q_{\text{target}}$
6:   $q_{\text{new}} \leftarrow q_{\text{near}} + \int_{t=0}^{\tau} \dot{q}(q, u)\,dt$
7:   if $q_{\text{new}} \in Q_{\text{free}}$ then $V \leftarrow V \cup \{q_{\text{new}}\}$, $E \leftarrow E \cup \{(q_{\text{near}}, q_{\text{new}})\}$
8:   end for

• Crucial Questions:
  – How to measure near in non-holonomic systems?
  – How to find controls $u$ to steer towards target?
Summary

• We discussed three different basic examples
  – 1D point mass
  – A general dynamic robot
  – A non-holonomic car model

• We discussed how previous methods can be extended to the \textit{dynamic}/non-holonomic case:
  – Inv. Kinematics Control $\rightarrow$ Optimal Operational Space Control
  – Trajectory Optimisation $\rightarrow$ Trajectory Optimisation
  – RRTs $\rightarrow$ RRTs with special tricks

• Trajectory optimization is as before:

$$f(u_{0:T}) = \sum_t \|u_t\|_H^2 + \Phi_t(x_t)^T\Phi_t(x_t)$$

where $x_t$ depends on the controls $u_{0:t-1}$, and $\Phi_t(x_t)$ can be any task vector containing positional (as in the kinematic case) or velocity-type task error.
Only kinematic control has $\text{dim}(u)=\text{dim}(x)$, all others $\text{dim}(u)<\text{dim}(x)$