ME 599/699 Robot Modeling & Control

Hasan Poonawala

Contents

Preface

This document collects and polishes hand-written notes I created for this class on Robotics. These notes are primarily based on the textbook written by Spong, Vidyasagar, and Hutchinson [**?**].

Chapter 1

Space and Motion

1.1 Introduction

This chapter lays the mathematical foundations for describing physical two and three dimensional space. The central message is that there are **infinite** ways to mathematically describe physical space. For example, two separate LIDAR sensors on a robot may not describe locations of the same object in space using the same coordinates. When we perform mathematical computations for robot motion, we need to be careful that we account for such differences.

Section 1.2 connects inner product spaces with a mathematical characterization of physical space as a Euclidean space. Section **??** describes graphs as an alternative description of space, which often serves as a useful abstraction for certain problems.

1.2 Euclidean Space

Euclidean Space is a model for physical space. Mathematically, this model turns out be that of an affine space. An affine space consists of elements called points. The main idea is that these **points are not vectors**, however differences between points become vectors. What this means is that we can't assign numbers to a point without using another (reference) point.

Definition 1 (Affine Space)**.** An affine space is a set A together with a vector space $\overrightarrow{A}$, and a transitive and free action of the additive group of $\overrightarrow{A}$ on the set A. The elements of the affine space A are called points, and the elements of the associated vector space $\overrightarrow{A}$ are called vectors, translations, or sometimes free vectors. Explicitly, the definition above means that the action is a mapping, generally denoted as an addition

$$A \times \overrightarrow{A} \to A \tag{1.1}$$

$$(a, v) \mapsto a + v \tag{1.2}$$

Free implies that the only the 0 element of a vector space maps a point back to itself. Transitive means any two points define a unique element of the vector space.

Definition 2 (Euclidean Space)**.** A Euclidean space is an affine space with the vector space given by an inner product space.

To reiterate:

Remark 1. An point in n-dimensional Euclidean space is not a vector.

When we describe points as numbers, what we are doing is implicitly using a reference point (origin) and a vector space with a basis to describe points. This process is natural to us because an n-dimensional Euclidean space is *isomorphic* to $\mathbb{R}^n$. That is, we can always create a one-to-one correspondence between a

Euclidean space and the inner product space $\mathbb{R}^n$. Concretely, after choosing a point in Euclidean space to be the origin, Euclidean space is indistinguishable from $\mathbb{R}^n$. The **problem** is that we can create **infinite** such correspondences.

Definition 3 (Cartesian Coordinates)**.** Identifying a point in Euclidean space with the zero vector of $\mathbb{R}^n$, and defining an orthogonal basis for $\mathbb{R}^n$ equips Euclidean space with Cartesian coordinates.

The fact that Euclidean space may be numerically handled through a Euclidean vector space $\mathbb{R}^n$ gives us something else: a notion of distance. This distance is known as **Euclidean distance**, and is the usual distance derived from the dot product (see Section A.1). Once we have a notion of distance, we are able to define a topology (see Appendix B.1) on Euclidean space, which leads to **mathematical descriptions of motion** in Euclidean space through the tools of calculus.

1.3 Summary & Preview

1. Points in three dimensional space (or the two dimensional plane) do not have intrinsic coordinates. These points form a real affine space.
2. Every cartesian coordinate frame assigns its own unique coordinate to a point in n-dimensional space. These coordinates form a real coordinate space $\mathbb{R}^n$ that possesses an inner product, a norm, and a metric.
3. The same point in space can have multiple coordinates, each corresponding to a different frame.
4. We can relate descriptions of the same point in space in different coordinate frames via rigid coordinate transformations.
5. We can describe the motion of multiple points on a moving rigid body occurs by describing the motion of a body-fixed coordinate frame.

Definition 4 (Group)**.** A group G is a set together with a binary operation $\cdot$ that satisfies the following properties for all $a, b, c \in G$:

(i) Closure: $a \cdot b \in G$;

(ii) Associativity: $a \cdot (b \cdot c) = (a\cdot)b \cdot c$;

(iii) Existence of identity element $e \in G$ such that $a \cdot e = e \cdot a = a$;

(iv) Existence of inverse element $d \in G$ such that $d \cdot a = a \cdot d = e$.

1.4 Cartesian Coordinates

We've seen the an n-dimensional Euclidean space consists of a collection of points, together with the notion of translation as implied by a inner product space $\mathbb{R}^n$.

This inner product helps identify whether two translations are collinear or not, in effect defining parallel lines in Euclidean space.

Definition 5 (Cartesian Coordinates)**.** Identifying a point in Euclidean space with the zero vector of $\mathbb{R}^n$, and defining an orthogonal basis for $\mathbb{R}^n$ equips Euclidean space with Cartesian coordinates.

1.4.1 Body-Fixed Frames

We may define a coordinate frame that moves with a rigid body in $\mathbb{R}^n$ by choosing $n+1$ non-trivial points on the rigid body. One point becomes the origin, the remaining points define n independent basis vectors. For 3D, we need four points. Every point on the rigid body can then be assigned a unique coordinate relative to this frame which is constant for all time.

This frame is known as a body-fixed frame. Unless specified, we assume that the n independent basis vectors are orthogonal and normal, so that the frame is a cartesian frame.

1.5 Homogenous Transformations

Example 1 (Robot And Camera)**.** A robot needs to pick something up, and a camera tells it where it is. If the robot and camera are using different reference frames, how do you convert the position from the camera into a position that makes sense for the robot?

Let p^A and p^B be the coordinates of a point p in frames A and B respectively. We want a to find a map $g\colon \mathbb{R}^n \mapsto \mathbb{R}^n$ such that $p^A = g(p^B)$ for any point p in Euclidean space. The following theorem says that such a map must necessarily be affine.

Theorem 1 (Ulam-Mazur)**.** *Let U, V be normed spaces over $\mathbb{R}$. If mapping $g\colon U \mapsto V$ is a bijective isometry, then g is affine.*

Corollary 2. *Any coordinate transformation g between a pair of three dimensional cartesian coordinate spaces X and X' with the same orientation is parametrized by a pair (d, R) where $d \in \mathbb{R}^3$ and $R \in SO(3)$. Thus, $g(p) = Rp + d$.*

Proof. Assignment. □

Problem 1 (HW 2)**.** Prove Corollary 2
Hint: Given two coordinates p^A and q^A, and a map g that maps them to coordinates p^B and q^B respectively, what properties of coordinates p^B and q^B hold independent of the map g?

Problem 2 (HW 2)**.** Show that for two given cartesian coordinate frames, the parameters (d, R) of the coordinate transformation are unique.

Problem 3 (HW 2)**.** Let the affine transformation from frame A to frame B be parametrized by (d, R). Express the affine transformation that maps coordinates in frame B to coordinates in frame A in terms of R and d?
[Aside: Why is the derivation of this expression valid?]

The unique transformation that maps a point's coordinates in one frame to its coordinates in another frame is an affine map. We can convert this affine map between two Euclidean spaces of dimension 3 into a linear map between two subsets of $\mathbb{R}^4$.

Define a homogenization $h\colon \mathbb{R}^3 \mapsto \mathbb{R}^4$

$$h\left(p^A\right) = \begin{bmatrix} p^A \\ 1 \end{bmatrix}. \tag{1.3}$$

We refer to the vector $h\left(p^A\right)$ as the homogenous representation of coordinate p^A. The transformation between homogenous representations of coordinates in different frames is linear. Mathematically, if $p^A = Rp^B + d$, then

$$h\left(p^A\right) = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} h\left(p^B\right). \tag{1.4}$$

The matrix $\begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}$ represents a homogenous transformation, and forms a group.

Definition 6 (Special Euclidean Group)**.** A rigid motion is a pair (d, R) where $d \in \mathbb{R}^3$ and $R \in SO(3)$. The group of all rigid motions is known as the **Special Euclidean Group** and is denoted by $SE(3)$. We see that $SE(3) = \mathbb{R}^3 \times SO(3)$.

1.5.1 Notation

The superscript of a coordinate denotes the frame it is defined in. The coordinate transformation that takes points in frame B to frame A is denoted as (d_B^A, R_B^A), so that

$$p^A = R_B^A p^B + d_B^A.$$

1.6 Rigid Body Pose

We have shown that given two frames A and B, there's a unique affine transformation (d, R) that maps coordinates of a point in frame B to its coordinates in frame A, where $R \in SO(3)$ and $d \in \mathbb{R}^3$. Given any other frame C, the transformation that maps coordinates of frame C into coordinates in frame A is given by a transformation (d', R') where $(d', R') \neq (d, R)$. Since the affine transformation is unique for any frame, the pair (d, R) serves as a configuration of frame B in frame A.

Since we can associate a coordinate frame to a rigid body, we can associate the configuration of that frame to the rigid body. Therefore, the configuration of any rigid body in some coordinate frame, also known as its pose, is described by a pair (d, R) where $R \in SO(3)$ and $d \in \mathbb{R}^3$. This pair is associated with the rigid body, and it comes from the body-fixed coordinate frame.

1.6.1 Rotations

The matrix R is an element of $SO(3)$, which is a subset of the more general linear group $GL(\mathbb{R}^3)$. The group $GL(\mathbb{R}^3)$ is the space of linear bijective transformations between $\mathbb{R}^3 \mapsto \mathbb{R}^3$ with functional composition as the group operation. The matrix R, which is part of the pose of a rigid body, is known as the orientation matrix, and exactly gives the orientation of one rigid body with respect to another.

We have seen one interpretation of R as a map from one frame to another frame rotated with respect to the first. Since the map is unique, R also serves to represent the orientation of the second frame with respect to the first. A rotation matrix can also represent a rotation within the same (or **current**) frame.

Most important property: $R^T = R^{-1}$

The rotation matrix is effectively defining a basis for $\mathbb{R}^3$.

Definition 7 (Basis)**.** A basis B of a vector space V over a field $\mathbb{F}$ is a linearly independent subset of V that spans V.

An orthonormal bases has unit elements and mutually perpendicular vectors.

The relationships for change of bases from linear transformation has direct interpretations in terms of rotation operations.

Similarity Transform

A linear map defined in a coordinate frame has a matrix representation in that frame. A similarity transform maps the representation of that linear transformation into another coordinate frame.

Suppose T^A represents a linear map defined in frame A. Let the orientation of frame B with respect to frame A be R^A_B. Then the same linear map in frame B is given by matrix T^B

$$\begin{aligned} T^B &= \left(R^A_B\right)^{-1} T^A R^A_B \\ \implies T^A &= R^A_B T^B \left(R^A_B\right)^{-1} \\ \implies T^B &= R^B_A T^A \left(R^B_A\right)^{-1} \\ \implies T^A &= \left(R^B_A\right)^{-1} T^B R^B_A \end{aligned}$$

1.6.2 Basic Rotations

Consider three frames rotated about each one of the world frame axes by an angle θ.

Each rotation is given by

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$R_{y,\theta} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_{z,\theta} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1.6.3 Composition of Rotations

We can consider a rotation matrix to represent a rotation relative to a frame. Consider a rigid body in frame A, where its frame is coincident with that of A. We perform a rotation corresponding to rotation matrix R_1. The rigid body's frame is now different from frame A, and we call it frame B. We perform a second rotation corresponding to a matrix R_2, say $R_2 = R_{x,\pi/3}$. However, this rotation can be applied in two ways to our rigid body, depending on which frame R_2 is relative to: the original frame A, or the frame B that is coincident with the body. We get two different final poses for the rigid body depending on which one we choose.

Current Frame

If the rotation R_2 is relative to the current frame of the rigid body (frame B), then the combined effect of the two successive rotations in frame A is a post-multiplication of the sequence of rotations: R_1 then R_2. That is, $R' = R_1 R_2$.

Fixed Frame

If the rotation R_2 is relative to the fixed frame A, then the combined effect of the two successive rotations in frame A is a pre-multiplication of the sequence of rotations: R_1 then R_2. That is, $R'' = R_2 R_1$.

How to derive: We have a rotation R_2 in frame A. We can express it in frame B via a similarity transform

$$R_3 = R_1^{-1} R_2 R_1$$

. We've converted the rotation in frame A to its representation in frame B, so that the sequence of rotations are with respect to the current frame.

$$\begin{aligned} R'' &= R_1 R_3 && (1.5) \\ &= R_1 \left(R_1^{-1} R_2 R_1 \right) && (1.6) \\ &= R_2 R_1 && (1.7) \end{aligned}$$

Non-commutation

Since matrix multiplication is non-commutative, in general $R' \neq R''$.

1.6.4 Parametrizations of SO(3)

Although the representation R has nine elements, the space $SO(3)$ is three dimensional. One way to see this is to note that $R^T R = I$, which introduces six constraints on the elements of R. We now look at two popular ways to parametrize R as a three-dimensional vector.

Euler Angle Representation

Euler angles consist of three angles corresponding to three consecutive basic rotations. These rotations either use two axes (proper Euler) or three axes (Tait-Bryan). Furthermore, we may designate the rotations to be with respect to a world frame (extrinsic) or the body frame (intrinsic).

Proper Euler: There are six proper Euler conventions:

1. X-Y-X (Rotate about X, then Y, then X again)
2. X-Z-X
3. Y-X-Y

4. Y-Z-Y
5. Z-X-Z
6. Z-Y-Z (Common in astrophysics)

These are doubled when considering intrinsic (body-frame) and extrinsic (world-frame) rotations.

Tait-Bryan:

1. X-Y-Z
2. X-Z-Y
3. Y-X-Z
4. Y-Z-X
5. Z-X-Y
6. Z-Y-X

Also double if you allow both intrinsic and extrinsic rotations. This representation includes the yaw-roll-pitch method common in aerospace literature. Instead, if someone says roll-pitch-yaw, then we have different numerical values.

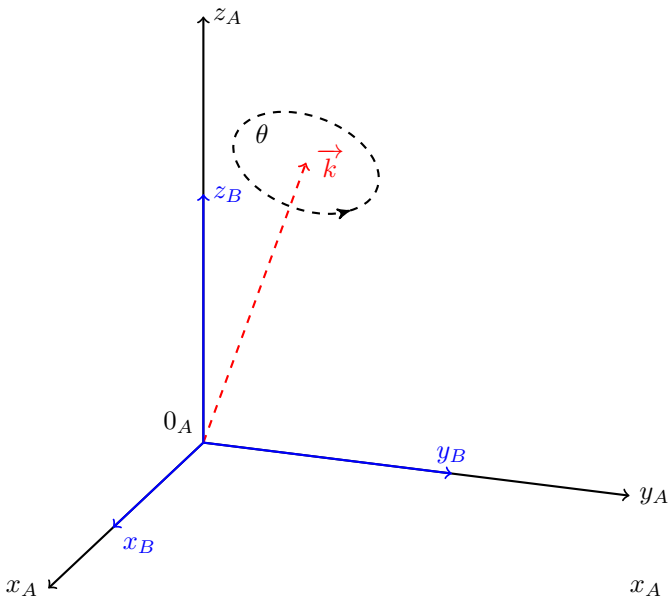
The main drawback of Euler-angles: non-uniqueness of values of three angles at singular points.

Axis/Angle Representation

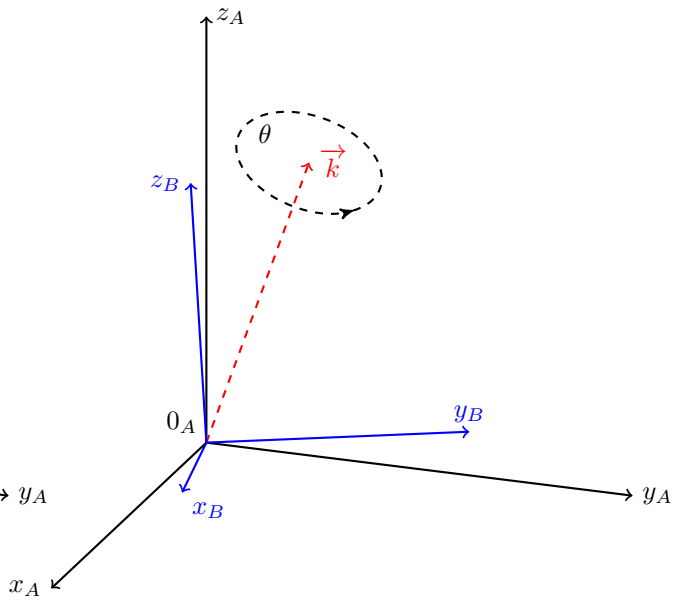
This representation is related to quaternions. The idea is that any orientation in a frame can be reached by rotating a coordinate frame by some angle $\theta \in [0, 2\pi)$ around some vector $\overrightarrow{k} \in \mathbb{R}^3$ in that frame. How do we represent that orientation?

Consider a frame C identical to the world frame A. Define β, rotation of C about world y, then α, rotation of C about world z that aligns world z_C with $\overrightarrow{k}$. In effect, β and α parametrize the unit-norm 3-dimensional vector $\overrightarrow{k}$. Then $R_C^A = R_{z,\alpha} R_{y,\beta}$. We want to find the rotation matrix $R_{k,\theta}$ in frame A corresponding to a rotation about $\overrightarrow{k}$, given that it represents a rotation about z_C in frame C by θ.

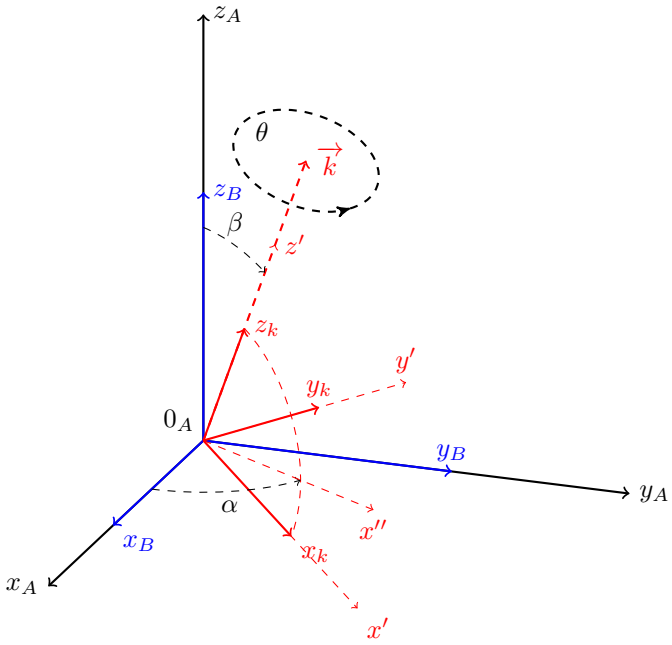
$$
\begin{aligned}
R_C^A &= R_{z,\alpha} R_{y,\beta}. \\
R_{k,\theta} &= R_C^A R_{z,\theta} \left(R_C^A\right)^{-1} && \left(\text{using } T^A = R_B^A \; T^B \; \left(R_B^A\right)^{-1} \right) \\
R_{k,\theta} &= R_{z,\alpha} R_{y,\beta} R_{z,\theta} R_{y,-\beta} R_{z,-\alpha}.
\end{aligned}
$$



(a) Axis-Angle Rotation: Before



(b) Axis-Angle Rotation: After



(c) Axis-Angle Rotation: β and α. Rotate a copy of the world frame about world y by β, then rotate the result about world x by α. The final result is the $x' - y' - k$ frame. The vector x'' lies in both the world $x - y$ plane and the plane formed by vectors k and x'.

Figure 1.1: Axis/Angle Rotations

Chapter 2

Kinematics

2.1 Kinematic Chains

Kinematic chains consist of a set of rigid bodies connected to each other through joints. Start with a base rigid body that's so massive it's fixed. We typically call this the world or inertial frame.

Types of Kinematic Chains:

- Open / Closed
- Serial / Parallel

Simple Joints:

- Prismatic
- Revolute

2.1.1 Serial Kinematic Chains

- We look at serial kinematic chains where all joints are simple.
- We number links as 0 for base to n in sequence.
- The assumption of single-parameter joints means we can use basic transformations to handle coordinate transformations.
- These basic transformation are denoted $A_i(q_i)$, where $q_i \in \mathbb{R}$ is the joint variable.
- q_i is either an angle θ_i or a distance d_i, depending on the type of simple joint.

Given link i and $i-1$,

$$A_i = \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

Transformations between links i and j is T_j^i, where we are expressing frame j in frame i.

$$T_j^i = \begin{cases} A_{i+1}A_{i+2}\cdots A_{j-1}A_j & i < j \\ I & i = j \\ \left(T_j^i\right)^{-1} & i > j \end{cases} \tag{2.2}$$

Since we don't want to manually compute R_i^{i-1} for each q_i, we use our previous tricks of composing rotations etc.

2.1.2 Denavit-Hartenberg Convention

The Denavit-Hartenberg (D-H) Convention is a popular convention that facilitates consistent communication of robot manipulator information. In this convention

- All motion happens along the z axis
- Four numbers are enough to define relative link transformations (instead of 6 or 12).

When introducing the D-H convention, we typically consider a n-link serial manipulator. Recall that the n moving links are numbered 1 to n, with the reference frame 0 attached to the base (typically not moving). Therefore, we may define $n+1$ frames where

- Frame i is rigidly attached to link i
- Frame i moves relative to link $i-1$ about joint i
- The frame $i-1$ is located such that the axis of motion of link i at joint i is defined by z_{i-1}

A consequence of this choice is that

- The location of the frame i that is rigidly attached to link i depends on how and where link $i+1$ is attached[1] to it!
- The final link has no 'successor' or 'child' link. It's frame is called the end-effector frame, or the tool frame. Without a $(n+1)^{\text{th}}$ link, we define this frame based on the application or end-effector. For two-fingered grippers, the z-axis, or approach axis is parallel to the fingers, since we approach objects to be gripped by moving along this direction. the y axis is the sliding axis, since fingers slide along y to open or close the grip.

The D-H convention is based on two restrictions:

(DH1) The x_1 axis intersects the z_0 axis.
(DH2) The x_1 axis is orthogonal to the z_0 axis.

This restriction makes the transformation matrix between link i and $i-1$ given in (2.1) reduce to

$$A_i = \text{Rot}_{z,\theta_i}\text{Trans}_{z,d_i}\text{Trans}_{x,a_i}\text{Rot}_{x,\alpha_i} \tag{2.3}$$

While the numbering of frames and joints seems a bit arbitrary, we need to get it straight when constructing the geometric Jacobian (next section).

2.2 Geometric and Analytic Jacobians

Forward and inverse kinematics are about creating the map $T_n^0(q)$ that provides the end effector pose $\left(o_n^0(q), R_n^0(q)\right)$.

In a similar way, when the link variables q change with time as $\dot{q}$, what is the 'velocity' of the end effector?

2.2.1 Geometric Jacobian

We represent the end effector velocity as (v_n^0, ω_n^0), where

$$v_n^0 = \dot{o}_n^0 \tag{2.4a}$$
$$S(\omega_n^0) = \dot{R}_n^0 \left(R_n^0\right)^T \tag{2.4b}$$

We saw that the desired characterization of the velocity ξ of the end effector is linear in the rate of change of q. That is,

$$\xi = \begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = J\dot{q} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q}, \tag{2.5}$$

where J is the kinematic Jacobian.

We compute the i^{th} column J_{v_i} of J_v as

$$J_{v_i} = \begin{cases} z_{i-1} & , \text{ if joint } i \text{ is prismatic} \\ z_{i-1} \times (o_n - o_{i-1}) & , \text{ if joint } i \text{ is revolute} \end{cases} \tag{2.6}$$

[1] Note that the book "Modern Robotics" uses a different convention for locating link frames.

We compute the i^{th} column J_{ω_i} of J_ω as

$$J_{\omega_i} = \begin{cases} 0 & , \quad \text{if joint } i \text{ is prismatic} \\ z_{i-1} & , \quad \text{if joint } i \text{ is revolute} \end{cases} \tag{2.7}$$

Here, we see that when constructing the i^{th} column of j, which corresponds to the i^{th} joint (and, so, i^{th} joint variable), we need to look at frame $i-1$, which is located along joint i. Note that J is actually a function $J(q)$, since the axes z_i, $i \in \{1, \ldots, n\}$ depend on q.

2.2.2 Analytic Jacobian

The geometric Jacobian $J(q)$ is not the partial derivative of any map from q to $\left(o_n^0(q), R_n^0(q)\right)$. In particular, the angular velocity ω_n^0 is usually not the derivative of the coordinates representing the configuration.

Suppose we represent the position and orientation of the end effector using vectors $d(q) \in \mathbb{R}^3$ and $\alpha(q) \in \mathbb{R}^3$, so that

$$X = \begin{bmatrix} d(q) \\ \alpha(q) \end{bmatrix}, \tag{2.8}$$

and

$$\dot{X} = \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} = J_a(q)\dot{q}, \tag{2.9}$$

where $J_a(q)$ is known as the analytic Jacobian. It maps the rates of change of the link angles, $\dot{q}$ to the rates of change of the chosen configuration X of the end-effector.

To derive it, we use the geometric Jacobian. To do so, note that we can define the angular velocity ω in terms of the rates of change of a parametrization such as Euler angles. For example, let α be $Z-Y-Z$ Euler angles (ϕ, θ, ψ) such that $R = R_{z,\psi} R_{y,\theta} R_{z,\phi}$ and $\dot{R} = S(\omega)R$, we can obtain a map $\omega = B(\alpha)\dot{\alpha}$. Then,

$$J(q)\dot{q} = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{d} \\ B(\alpha)\dot{\alpha} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & B(\alpha) \end{bmatrix} J_a(q)\dot{q}, \tag{2.10}$$

When $\det B(\alpha) \neq 0$, we can derive

$$J_a(q) = \begin{bmatrix} I & 0 \\ 0 & B^{-1}(\alpha) \end{bmatrix} J(q). \tag{2.11}$$

2.2.3 Singularities

Once we have an expression for the J, we can use it to find $\dot{q}$ given some desired values for v_0, ω_0. To do so, we must solve

$$\xi = J\dot{q} \tag{2.12}$$

If J is an invertible 6×6 matrix, we are done. Often, J is not invertible, because it is not square of because it does not have full rank when square.

Singularities Suppose $J \in \mathbb{R}^{m \times n}$, where n is the number of simple joints in a kinematic chain. The rank of $J(q)$, denoted $\text{rank}(J)$, is less than or equal to $\min(m, n)$, and the set of reachable velocities is a subspace with dimension $\text{rank}(J)$. Ideally, we want $\text{rank}(J) = m$ so that any arbitrary task velocity can be achieved at any configuration.

This situation fails when J has rank less than the dimension of ξ. The matrix J depends on q, and at some configurations, J may lose rank. These configurations are known as singularities or singular configurations.

It's not just that the singular point prevents calculation of $\dot{q}$, but that J is ill-conditioned near it.

- Some singular points become unreachable under perturbations of the system mechanical parameters.

2.2.4 Decoupling Singularities

For a manipulator comprising a 3-DOF arm and a 3-DOF wrist, the Jacobian $J(q)$ can be made to exhibit a block diagonal structure that makes studying its singular configurations easier. The main step that achieves this is to ensure that o_6 coincides with the already coincident origins o_3, o_4, o_5. Then,

$$J = \begin{bmatrix} J_{11} & 0 \\ J_{12} & J_{22} \end{bmatrix}, \tag{2.13}$$

and so

$$\det J = \det J_{11} \det J_{22}. \tag{2.14}$$

For the three-link articulated manipulator, we can derive that

$$\det J_{11} = a_2 a_3 \sin\theta_3 \left(a_2 \cos\theta_2 + a_3 \cos(\theta_2 + \theta_3)\right). \tag{2.15}$$

2.2.5 Inverse Velocity

When $n = m$, except for singular configurations, there is a 1-1 relationship between joint and task velocities. In this case, we may directly compute $\dot{q}$ from $\dot{x} = \xi$ as

$$\dot{q} = J(q)^{-1}\xi. \tag{2.16}$$

When $n > m$, there are more degrees of freedom available than the velocities we want to generate. This situation is known as *redundant manipulation*. It is not necessary that $n > 6$ for redundancy in manipulation. We can have redundancy in a 6 DoF manipulator ($n = 6$) when $m < 6$. For example, when the manipulator's gripper has a mounted camera, and we only want to orient the camera to observe an object, and the position is not important. When $\text{rank}(J(q)) = m$, and $m < n$, we can reach any velocity ξ for the end-effector frame at configuration q, through some choice of $\dot{q}$. However the matrix J is not invertible, so we may not use (2.16).

It can be shown that since $\text{rank}(J) = m$, the matrix product JJ^T is non-singular. Therefore, to compute $\dot{q}$ from ξ, we use the right pseudo-inverse J^+ of J, given by

$$J^+ = J^T(JJ^T)^{-1}.$$

Clearly, $JJ^+ = I$. In general, however, $J^+J \neq I$.

We may now solve for $\dot{q}$ given ξ in the redundant manipulation case as

$$\dot{q} = J^+\xi + (I - J^+J)b,$$

where $b \in \mathbb{R}^n$ is an arbitrary vector that does not affect ξ. If we want to minimize the norm of $\dot{q}$, we choose $b = 0$. Note that if J is an invertible square matrix, then the expression above reduces to (2.16).

Example 2 (Elbow Manipulator)**.** For example, let the task X be the position of the end effector of a planar elbow manipulator. The configuration is $q_1, q_2 = \theta_1, \theta_2$ (both angles). We can compute

$$\dot{X} = J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}. \tag{2.17}$$

We find that $\mu = |\det J| = a_1 a_2 |s_2|$ ($|\det J|$ for non-redundant manipulators). When θ_2 is small, we have the least ability to move in all directions.

2.2.6 Manipulability

Suppose that $\xi \in \mathbb{R}^m$, so that $J(q) \in \mathbb{R}^{m \times n}$. We use the minimum norm solution $\dot{q} = J^+\xi$ to obtain link variable velocities from end-effector velocities.

We can derive

$$\|\dot{q}\|^2 = \xi^T(JJ^T)^{-1}\xi \tag{2.18}$$

If $\text{rank}(J) = m$, so that J is full rank, then we can define a manipulability ellipsoid in $\mathbb{R}^m$ as follows. Let $J = U\Sigma V$, the singular value decomposition.

Then

$$\xi^T (JJ^T)^{-1} \xi = (U^T \xi)^T \Sigma_m^{-2} (U^T \xi), \tag{2.19}$$

in which

$$\Sigma_m^{-2} = \begin{bmatrix} \sigma_1^{-2} & & & \\ & \sigma_2^{-2} & & \\ & & \ddots & \\ & & & \sigma_m^{-2} \end{bmatrix} \tag{2.20}$$

Substituting $w = U^T \xi$, we finally get that

$$\|\dot{q}\|^2 = w^T \Sigma_m^{-2} w = \sum_{i=1}^{m} \frac{w_i^2}{\sigma_i^2} \tag{2.21}$$

If we look at unit norm velocities in the joint space, these vectors form an ellipsoid defined by σ_i^2 in the space w which is a rotated version of ξ.

The manipulability μ is then given by

$$\mu = \Pi_{i=1}^m \sigma_i \tag{2.22}$$

Example 3 (Planar Elbow Manipulator)**.** For example, let the task X be the position of the end effector of a planar elbow manipulator. The configuration is $q_1, q_2 = \theta_1, \theta_2$ (both angles). We can compute

$$\dot{X} = J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}. \tag{2.23}$$

We find that $\mu = |\det J| = a_1 a_2 |s_2|$ ($|\det J|$ for non-redundant manipulators). When θ_2 is small, we have the least ability to move in all directions.

2.3 Static Force/Torque Relationships

Let $F = \begin{bmatrix} F_x & F_y & F_x & n_x & n_y & n_z \end{bmatrix}$ be the vector of forces and moments at the end effector. Let τ be the corresponding vector of joint torques. Then F and τ are related by

$$\tau = J^T(q) F \tag{2.24}$$

2.3.1 Derivation

One way to derive this relationship is the principle of virtual work. The idea is to imagine infinitesmal displacements δX and δq which satisfy the system constraints on motion. These displacements are called virtual displacements,. The total work done by F and τ when achieving these virtual displacements is

$$\delta w = F^T \delta X - \tau^T \delta q. \tag{2.25}$$

Since $\delta X = J(q)\delta q$, which is one constraint on the displacements due to the system, we obtain that

$$\delta w = (F^T J - \tau^T)\delta q. \tag{2.26}$$

The principle of virtual work says for a system in equilibrium, the total work done under any virtual displacement satisfying the constraints must be zero. Thus, (2.24) holds.

2.3.2 Manipulability Revisited

When the end-effector is moving in free space, the manipulability ellipsoid indicates the The map from the force to the torque is characterized by $J(q)^T$. Manipulability determines our ability to apply force (or accelerate) in certain directions.

In this case, it is sometimes good to have near singular configurations, since we get more force in some directions for the same input energy (norm of torque).

Chapter 3

Dynamics

So far, our control was based of an independent joint model that isolated each link and used robustness to account for this huge assumption. Gravity and dynamic coupling will limit the success of this approach in highly complex or high-energy motions. Therefore, we need to start considering a coupled dynamics model.

3.1 Lagrangian

We use the Euler-Lagrange framework to obtain the robot dynamics model given by

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + \tau_{friction} + \tau_e, \tag{3.1}$$

where q is the configuration, $\dot{q}$ the joint rates of change, τ_e are torques due to the external forces, and τ are the motor torques. Note that conservative forces and torques due to joint friction $\tau_{friction}$ are not viewed as external forces applied to the robot. The terms D, C, J, and G are the mass or inertia matrix, Coriolis matrix, geometric Jacobian, and conservative force vector (usually gravity) respectively.

The vector q is often referred to as a generalized coordinate in dynamics. This coordinate represents degrees of freedoms after accounting for all holonomic constraints in the system.

We can derive these equations by defining the Lagrangian $\mathcal{L}$ of the system, which is the difference between the kinetic and potential energies of the system. This derivation leads to all terms on the left hand side of (3.1). The terms on the right are essentially non-conservative external forces acting on the system.

3.1.1 Kinetic Energy

The kinetic energy of the robot is the sum of the kinetic energies of its link. Since each link is a rigid body, we know how to calculate its kinetic energy. Each link has a mass m and principal inertia I about its center-of-mass q that has velocity v and angular velocity ω in the world frame. The kinetic energy is given by

$$\mathcal{K} = \frac{1}{2}mv^T v + \frac{1}{2}\omega^T \mathcal{I}\omega, \tag{3.2}$$

where $\mathcal{I}$ is the inertia of the link with respect to the world frame given by RIR^T where R is orientation of principal axes in world frame. The equation for I is

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}, \tag{3.3}$$

where

$$I_{xx} = \int\int\int (y^2 + z^2)\rho(x, y, z)dxdydz \tag{3.4}$$

$$I_{yy} = \int\int\int (x^2 + z^2)\rho(x, y, z)dxdydz \tag{3.5}$$

$$I_{zz} = \int\int\int (x^2 + y^2)\rho(x, y, z)dxdydz \tag{3.6}$$

and

$$I_{xy} = I_{yx} = -\int\int\int xy\ \rho(x, y, z)dxdydz \tag{3.7}$$

$$I_{xz} = I_{zx} = -\int\int\int xz\ \rho(x, y, z)dxdydz \tag{3.8}$$

$$I_{yz} = I_{zy} = -\int\int\int yz\ \rho(x, y, z)dxdydz \tag{3.9}$$

When we combine these kinetic energies, we get

$$\begin{aligned} K &= \sum_i \frac{1}{2} m_i v_i^T v_i + \frac{1}{2}\omega_i \mathcal{I} \omega_i \\ &= \sum_i \frac{1}{2} m_i \dot{q}^T J_{v_i}^T(q) J_{v_i}(q)\dot{q} + \frac{1}{2}\dot{q}^T J_{\omega_i}^T(q) R_i(q) I_i R_i^T(q) J_{\omega_i}\dot{q} \\ &= \frac{1}{2}\dot{q}^T D(q)\dot{q} \\ &= \sum_{i,j} d_{ij}(q)\dot{q}_i\dot{q}_j \end{aligned} \tag{3.10}$$

3.1.2 Potential Energy

Potential energy P due to gravity is

$$P = \sum_{i=1}^{n} m_i g \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} c_i^0(q) \tag{3.11}$$

where $c_i^0(q)$ is the location of the center of mass of link i in the world frame (and not the origin of the i^{th} frame).

3.2 Euler-Lagrange Equations

See the following online resource

Given a Lagrangian $K - P$, we can derive an equation of motion for each generalized coordinate q_k as

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} = \tau_j \tag{3.12}$$

Given the expressions for K and P above, we end up with

$$\sum_{j=1}^{n} d_{kj}(q)\ddot{q}_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k(q) = \tau_k, \quad k \in \{1, \ldots, n\} \tag{3.13}$$

where

$$c_{ijk} = \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\} \tag{3.14}$$

$$g_k = \frac{\partial P}{\partial q_k} \tag{3.15}$$

The compact representation of these equations is

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau, \tag{3.16}$$

where $c_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i$. Adding non-conservative forces such as viscous friction and externally applied forces yields (3.1).

3.2.1 Example

Planar Elbow Manipulator Use the DH joint variables as generalized coordinates. Compute

$$v_{c1} = J_{v_{c1}}\dot{q} = \begin{bmatrix} -l_{c1}\sin q_1 & 0 \\ l_{c1}\cos q_1 & 0 \\ 0 & 0 \end{bmatrix}\dot{q} \tag{3.17}$$

$$v_{c1} = J_{v_{c2}}\dot{q} = \begin{bmatrix} -l_1\sin q_1 - l_{c2}\sin(q_1+q_2) & -l_{c2}\sin(q_1+q_2) \\ l_{c1}\cos q_1 + l_{c2}\cos(q_1+q_2) & l_{c2}\cos(q_1+q_2) \\ 0 & 0 \end{bmatrix}\dot{q} \tag{3.18}$$

Also,

$$J_{\omega 1} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad J_{\omega 2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}, \quad R_{\omega 1} = R_{z,q_1}, \qquad R_{\omega 2} = R_{z,q_1+q_2}. \tag{3.19}$$

Therefore, $J_{\omega 1}^T R_{z,q_1} = J_{\omega 1}$ and $J_{\omega 2}^T R_{z,q_1+q_2} = J_{\omega 2}$. In turn,

$$J_{\omega 1}^T R_{z,q_1} I_1 R_{z,q_1}^T J_{\omega 1} = \begin{bmatrix} I_{zz,1} & 0 \\ 0 & 0 \end{bmatrix}, \quad J_{\omega 2}^T R_{z,q_1+q_2} I_2 R_{z,q_1+q_2}^T J_{\omega 2} = \begin{bmatrix} I_{zz,2} & I_{zz,2} \\ I_{zz,2} & I_{zz,2} \end{bmatrix}. \tag{3.20}$$

Therefore

$$\begin{aligned} D(q) &= m_1 J_{v_{c1}}^T J_{v_{c1}} + m_2 J_{v_{c2}}^T J_{v_{c2}} + \begin{bmatrix} I_{zz,1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} I_{zz,2} & I_{zz,2} \\ I_{zz,2} & I_{zz,2} \end{bmatrix} \\ &= \begin{bmatrix} m_1 l_{c1}^2 + m_2(l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos q_2) + I_{zz,1} + I_{zz,2} & m_2(l_{c2}^2 + 2l_1 l_{c2}\cos q_2) + I_{zz,2} \\ m_2(l_{c2}^2 + 2l_1 l_{c2}\cos q_2) + I_{zz,2} & m_2 l_{c2}^2 + I_{zz,2} \end{bmatrix} \end{aligned} \tag{3.21}$$

Let $h = -m_2 l_1 l_{c2}\sin q_2$. Then,

$$c_{111} = c_{222} = c_{122} = 0, \quad c_{121} = c_{211} = c_{221} = h, \quad c_{112} = -h. \tag{3.22}$$

$$C(q,\dot{q}) = \begin{bmatrix} h\dot{q}_2 & h\dot{q}_2 + h\dot{q}_1 \\ -h\dot{q}_1 & 0 \end{bmatrix} \tag{3.23}$$

We have that

$$P = m_1 g l_{c1}\sin q_1 + m_2 g(l_1\sin q_1 + l_{c2}\sin(q_1+q_2)) \tag{3.24}$$

Therefore,

$$g_1(q) = m_1 g l_{c1}\cos q_1 + m_2 g l_1\cos q_1 + m_2 g l_{c2}\cos(q_1+q_2) \tag{3.25}$$

$$g_2(q) = m_2 g l_{c2}\cos(q_1+q_2) \tag{3.26}$$

3.3 Properties of the Euler-Lagrange Equations

3.3.1 Skew Symmetry and Passivity

Proposition 3. *The matrix $\dot{D}(Q) - 2C$ is skew symmetric.*

Proof. The $(k, j)^{\text{th}}$ element of the matrix $N = \dot{D}(Q) - 2C$ is

$$\begin{aligned} n_{kj} &= \dot{d}_{kj} - 2c_{kj} \\ &= \sum_{i=1}^{n} \frac{\partial d_{kj}}{\partial q_i}\dot{q}_i - 2\sum_{i=1}^{n} c_{ijk}\dot{q}_i \\ &= \sum_{i=1}^{n} \left[\frac{\partial d_{kj}}{\partial q_i} - c_{ijk}\right]\dot{q}_i \\ &= \sum_{i=1}^{n} \left[\frac{\partial d_{kj}}{\partial q_i} - 2\frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}\right]\dot{q}_i \\ &= \sum_{i=1}^{n} \left[\frac{\partial d_{ij}}{\partial q_k} - \frac{\partial d_{ki}}{\partial q_j}\right]\dot{q}_i \end{aligned} \tag{3.27}$$

The expression for n_{jk} will be

$$n_{jk} = \left[\frac{\partial d_{ik}}{\partial q_j} - \frac{\partial d_{ji}}{\partial q_k}\right]\dot{q}_i \tag{3.28}$$

Since $d_{ij} = d_{ji}$ and $d_{ik} = d_{ki}$, we see that $n_{jk} = -n_{kj}$. Therefore, N is skew symmetric

□

This skew symmetry property is related to a concept known as passivity, discussed in Section 3.4.

3.3.2 Bounds on Inertia Matrix

For a system with revolute joints, there exist λ_m and λ_M such that

$$\lambda_m I_{n\times n} \leq D(q) \leq \lambda_M I_{n\times n} < \infty \tag{3.29}$$

If the joints are not revolute, then the upper bound by ∞ goes away. The remaining inequalities are still valid, since the matrix $D(q)$ is always real, symmetric, and positive definite.

3.3.3 Linearity in Parameters

Unsurprisingly, we can derive a function $Y(q, \dot{q}, \ddot{q})$ and parameter set θ such that

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Y(q, \dot{q}, \ddot{q})\theta \tag{3.30}$$

The key idea is that for the same robot (same mechanism), Y is unchanging, and the equations are linear in the parameters θ.

3.4 Passivity

Passivity is a property that arises in several types of systems. In the case of dynamical systems, one interpretation is that a passive system (the system exhibits passivity) doesn't produce energy of its own. Electrical circuits made with passive components behave this way, which is where the term passivity comes from. Passive systems have some well-understood behaviors such as

1. Stability
2. L_2 gain stability

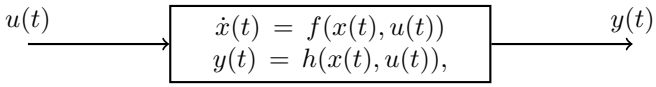
3. Stable behavior under feedback interconnections

Consider a system

$$\dot{x}(t) = f(x(t), u(t)) \tag{3.31}$$
$$y(t) = h(x(t), u(t)), \tag{3.32}$$

where

- $x \in \mathbb{R}^n$: state
- $y \in \mathbb{R}^m$: output
- $u \in \mathbb{R}^p$: input



This system interacts with the environment through time-varying inputs $u(t)$ and outputs $y(t)$, and this interaction influences the state $x(t)$.

Remark 2. For many systems, the input and output variables match the **effort** and **flow** variables of one-port models used in network-impedance-based modeling.

One way to define this interaction is through two functions known as the **supply rate** $S(y, u)$ and the **storage function** $V(x)$.

Definition 8. Supply Rate. The supply rate is a function $S: \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ that quantifies the amount of interaction with the environment.

Definition 9. Storage Function. A storage function is a non-negative function $V: \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ of the state.

Example 4. The state of a capacitor is the charge q_c contained in it. A good choice of the storage function is typically the electrical energy stored in the component

$$V(q_c) = \frac{1}{2C} q_c^2,$$

where C is the capacitance of the capacitor.

The input is the current flow i_c and the output is the voltage e_c across the component, however some circuit applications involve switching the two definitions. The supply rate is the power provided/taken from the capacitor:

$$S(e_c, i_c) = e_c i_c.$$

Note that a capacitor satisfies

$$q_c = C e_c, \quad \frac{d}{dt} q_c = i_c.$$

Definition 10 (Passivity.)**.** A system with state x is said to be *passive* with respect to an input-output pair y, u if there exists a supply rate $S(y, u)$ and storage function $V(x)$ such that

$$V(t_1) - V(t_0) \leq \int_{t_0}^{t_1} S\left(y(\tau) u(\tau)\right) d\tau, \quad \forall t_0, t_1 \geq t_0.$$

A passive system is one where the change in its storage function is always less than the integral of the supply rate.

Example 1. Continued Let's compute the time derivative of the storage function:

$$\frac{d}{dt}V(q_c) = \left(\frac{\partial V}{\partial q_c}\right)\dot{q}_c = \left(\frac{1}{C}q_c\right)\dot{q}_c \tag{3.33}$$

$$= \frac{1}{C}q_c\dot{q}_c = \frac{1}{C}\left(Ce_c\right)\left(i_c\right) \tag{3.34}$$

$$= e_c i_c \tag{3.35}$$

$$\implies \dot{V}(t) = S\left(e_c(t), i_c(t)\right) \tag{3.36}$$

$$\implies V(t_1) - V(t_0) \leq \int_{t_0}^{t_1} S\left(e_c(\tau), i_c(\tau)\right) d\tau \tag{3.37}$$

Therefore, we conclude that a capacitor is passive with respect to input e_c and output i_c, with storage function given by the energy and supply rate by the electrical power. Passivity here means that the capacitor doesn't create energy, its total energy is always no greater than the energy contained in the total power supplied to the capacitor.

3.4.1 Passivity in Robots

For robotic mechanisms, we consider the output to be the joint velocities $\dot{q}$, the input is the non-conservative torques τ applied at the joints by the environment, and the state is $(q, \dot{q})$.

A candidate storage function is the total energy of the system

$$V(q, \dot{q}) = KE(q, \dot{q}) + PE(q),$$

where K is the kinetic energy

$$K = \frac{1}{2}\dot{q}^T D(q)\dot{q},$$

and $PE(q)$ is the potential energy. Note that potential energy may be zero at many configurations, so that **a storage function is usually NOT a Lyapunov function**.

The dynamics become

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + \frac{\partial PE(q)}{\partial q} = \tau,$$

where we have so far assumed that the potential energy contains only the gravitational energy, so that $\frac{\partial PE(q)}{\partial q} = G(q)$. Note that technically the gradient $\frac{\partial PE(q)}{\partial q}$ is not a vector, but a dual vector (co-vector), but we don't stress this difference here. Since this term depends on potential energy, it represents conservative forces, of which gravitational force is an example. The term τ contains the non-conservative external forces, where . These external forces usually include:

- Motor torques τ_m
- Damping at joints $-B\dot{q}$
- Contact forces $J^T(q)F$, where $J(q)$ is the Jacobian from q to coordinates of a frame defined at the point of contact.

Other forces are possible, but we focus on these.

The power supplied to the robot is $\dot{q}^T\tau$, which serves as the **supply rate**.

Let's calculate the time-derivative of the storage function:

$$\dot{V} = \dot{q}^T D(q)\ddot{q} + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \left(\frac{\partial PE(q)}{\partial q}\right)^T \dot{q} \tag{3.38}$$

$$= \dot{q}^T D(q) D^{-1}(q) \left(\tau - C(q,\dot{q})\dot{q} - \frac{\partial PE(q)}{\partial q}\right) + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \left(\frac{\partial PE(q)}{\partial q}\right)^T \dot{q} \tag{3.39}$$

$$= \dot{q}^T \tau + \frac{1}{2}\dot{q}^T \left(\dot{D}(q) - 2C(q,\dot{q})\right)\dot{q} \tag{3.40}$$

$$= \dot{q}^T \tau \tag{3.41}$$

$$= S(y,u) \tag{3.42}$$

Again, we can conclude that a robot is passive from externally applied non-conservative forces/torques to the joint velocity.

3.4.2 Applications

Bipedal Robots

Potential-energy shaping is frequently used to dictate the behavior of a robot by modifying the potential energy of that robot. Gravity compensated PD control is an example of potential-energy shaping. A non-trivial example is low-power walking of bipedal robots on flat ground and uphill [**?**]. The idea was that some mechanisms walk steadily down slopes with no energy inputs. The energy lost at foot-strike balanced out the energy gained from potential energy. By comping up with a suitable potential energy function, those same motions could be achieved on flat ground, and do not require much additional energy inputs.

Teleoperation

Remote physical interaction used to perform poorly due to the issue of time-delay in the medium transferring real-time physical signal information between the master device and the remote device. Passivity theory suggested that the instability was due to a fake added energy resulting from the delay in signals. Loosely speaking, the power-content of out-of-phase signals is different from the real power present in synched versions. This power-mismatch built up energy, causing instability. In effect, the time-delay in the medium made it become a fake source of power that flowed into the master and remote devices. The solution was to make the medium itself passive, so that the three systems together: master, medium, and remote, formed a interconnected system that was passive by construction [**?**].

Power System Control

Electrical systems are also networks of components, and passivity naturally applies to the analysis and control of such systems [**?**].

3.5 Actuator Models

We have to understand the physical implementation of the torques τ that act on the links to move them.

3.5.1 Electric Actuators

To control a joint i, corresponding to link angle θ_i, we typically rigidly attach links $i-1$ and i to the housing/stator and shaft/rotor of a rotary actuator respectively. For prismatic joints, these links $i-1$ and i are rigidly attached to the housing and piston of a linear actuator, respectively. The output of the motor becomes the force/torque τ, unless a gear-like mechanism is introduced, at which point the torque τ on the link is some multiple of the motor's output, say τ_m.

We consider permanent magnet DC motors. Other kinds include *AC* motors and brushless *DC* motors.

A model for the motor torque is $\tau = K_m i_a$, if the flux in the motor is constant. The current is generated by a voltage source, and has dynamics

$$L\frac{d}{dt}i_a + Ri_a = V - V_b, \tag{3.43}$$

where L is the motor inductance, R is the winding resistance, V_b is the back EMF and is proportional to ω_m, the motor speed.

3.5.2 SISO Joint Model

This section justifies the most naive approach: independent servo control for each motor attached to each joint. For slow motions, the SISO model we derive is adequate, especially when the gear reductions are large. The gear ratio is typically in the range of 20 to 200 or more.

We have the actuator inertia J_a and gear inertia J_g driven by the motor torque τ_m, with gear friction coefficient B_m. The gear reduces θ_m to θ_s by ratio r. The second shaft is connected to an intertia J_l and driven by load torques τ_l.

The dynamics governing θ_m are

$$\begin{aligned} J_m\ddot{\theta}_m + B_m\dot{\theta}_m &= \tau_m - \tau_l/r \\ &= K_m i_a - \tau_l/r \end{aligned} \tag{3.44}$$

We can rewrite (3.43) and (3.45b) as

$$(Ls + R)I_a(s) = V(s) - K_b\ s\Theta_m(s), \tag{3.45a}$$
$$(J_m s^2 + B_m s)\Theta_m(s) = K_m I_a(s) - \tau_l(s)/r \tag{3.45b}$$

We combine these equations to obtain

$$s\left((Ls + R)(J_m s + B_m) + K_b K_m\right)\Theta_m(s) = K_m V(s) - \frac{(Ls + R)}{r}\tau_l(s). \tag{3.46}$$

The electrical time constant L/R is much smaller than the mechanical time constant J_m/B_m. So, we can divide by R and set L/R to zero, obtaining.

$$s\left((J_m s + B_m) + \frac{K_b K_m}{R}\right)\Theta_m(s) = \frac{K_m}{R}V(s) - \frac{1}{r}\tau_l(s). \tag{3.47}$$

Setting $u = K_m V/R$ and $d = -\tau_l/r$, we obtain the motor equation as

$$J\ddot{\theta}_m + B\dot{\theta}_m = u(t) - d(t). \tag{3.48}$$

Alternatively,

$$(Js^2 + Bs)\Theta_m(s) = U(s) - D(s) \tag{3.49}$$

This model views the joint angle through the motor angle, and the load due to the other links are a disturbance.

3.5.3 Flexible Joint Models

Harmonic gear mechanisms have low backlash, high torque transmission, and compact size. However, they use a flexspline that introduces flexibility. The effect of joint flexibility is to introduce oscillatory modes, which restricts the bandwidth of the control behavior so that these modes are not excited to resonance.

We model the flexibility as a single spring with spring constant $k[N/m^2]$. Consider such a flexible spring between the motor and the load. The equations are given by

$$J_l\ddot{\theta}_l + B_l\dot{\theta}_l + k(\theta_l - \theta_m) = 0, \text{ and} \tag{3.50}$$
$$J_m\ddot{\theta}_l + B_m\dot{\theta}_l + k(\theta_m - \theta_l) = u, \tag{3.51}$$

where u is input torque applied to the motor shaft.

$$p_l(s) = J_l s^2 + B_l s + k \tag{3.52}$$
$$p_m(s) = J_m s^2 + B_m s + k \tag{3.53}$$
$$\frac{\Theta_l(s)}{U(s)} = \frac{k}{p_l(s)p_k(s) - k^2} \tag{3.54}$$
$$\frac{\Theta_m(s)}{U(s)} = \frac{p_l(s)}{p_l(s)p_k(s) - k^2} \tag{3.55}$$
$$\tag{3.56}$$

The open-loop characteristic equation is

$$J_l J_m s^4 + (J_l B_m + J_m B_l)\, s^3 + (k(J_l + J_m) + B_l B_m)\, s^2 + k(B_l + B_m)s \tag{3.57}$$

To understand this equation, consider the case where viscous friction is absent. The open-loop characteristic equation is $s^2(J_l J_m s^2 + k(J_l + J_m))$, which has a double pole at the origin and two complex conjugate poles of the imaginary axis. That's a neutrally stable system. The frequency of the imaginary poles are proportional to $\sqrt{(}k)$. For systems with small values of B_l and B_m and high stiffness k, we expect the poles to be close to this situation, indicating that this system is hard to control.

The flexible joint model can be given a state $x \in \mathbb{R}^4$ where

$$x_1 = \theta_l, \quad x_2 = \dot{\theta}_l, \quad x_3 = \theta_m, \quad x_4 = \dot{\theta}_m. \tag{3.58}$$

The state space model is

$$\dot{x}_1 = x_2 \tag{3.59a}$$
$$\dot{x}_2 = -\frac{k}{J_l}x_1 - \frac{B_l}{J_l}x_2 + \frac{k}{J_l}x_3 \tag{3.59b}$$
$$\dot{x}_3 = x_4 \tag{3.59c}$$
$$\dot{x}_4 = \frac{k}{J_m}x_1 - \frac{k}{J_m}x_3 - \frac{B_m}{J_m}x_4 + \frac{1}{J_m}u \tag{3.59d}$$

which we can represent compactly as the linear system

$$\dot{x} = Ax + Bu \tag{3.60}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{J_l} & -\frac{B_l}{J_l} & \frac{k}{J_l} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{J_m} & 0 & -\frac{k}{J_m} & -\frac{B_m}{J_m} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J_m} \end{bmatrix}. \tag{3.61}$$

Chapter 4

Control

We first look at methods that treat the each joints as a single-input-single-output that only experiences the presence of the other links in the robots as a disturbance. These methods can often do quite well. For highly dynamic motion involving significant changes in energy, we need to switch to multi-joint approaches.

4.1 Independent Joint Control

For each link i, we construct a trajectory $q_i^d(t)$, implying $\dot{q}_i^d(t)$. The problem then becomes how to make our link angles track those desired angles and rate of change of angles. Mathematically, we want

$$\lim_{t\to\infty} q_i(t) \to q_i^d(t).$$

4.1.1 Routh Hurwitz Criterion

- The second-degree polynomial, $P(s) = s^2 + a_1 s + a_0$ has both roots in the open left half plane (and the system with characteristic equation $P(s) = 0$ is stable) if and only if both coefficients satisfy $a_i > 0$.
- The third-order polynomial $P(s) = s^3 + a_2 s^2 + a_1 s + a_0$ has all roots in the open left half plane if and only if a_2, a_0 are positive and $a_2 a_1 > a_0$.

4.1.2 P Control

The simplest control strategy is Proportional control:

$$u(t) = -k_p(\theta_m(t) - \theta_d(t)). \tag{4.1}$$

The closed-loop model becomes

$$(Js^2 + Bs)\Theta_m(s) = -k_p\Theta_m(s) + k_p\Theta_d(s) - D(s), \tag{4.2}$$

or

$$\Theta_m(s) = \frac{k_p}{Js^2 + Bs + k_p}\Theta_d(s) - \frac{1}{Js^2 + Bs + k_p}D(s). \tag{4.3}$$

The error, is therefore

$$E(s) = \Theta_d(s) - \Theta_m(s) = \frac{Js^2 + Bs}{Js^2 + Bs + k_p}\Theta_d(s) - \frac{1}{Js^2 + Bs + k_p}D(s). \tag{4.4}$$

As long as $k_p > 0$ and disturbances are bounded we get stability. For step disturbance $d(t) = D$ and reference $\theta_d(t) = \theta_d$, we apply the final value theorem to see that

$$e_{ss} = \lim_{s\to 0} sE(s) = -\frac{D}{K_p} \tag{4.5}$$

So, we can make the errors zero and the effect of disturbance small. What we can't do is shape the transient response, since we have limited control over the closed loop poles. To rectify this, we must use a PD control.

4.1.3 PD Control

To gain more control over the response, we use a Proportional-Derivative controller:

$$u(t) = -k_p(\theta_m(t) - \theta_d(t)) - K_d\dot{\theta}_m(t). \tag{4.6}$$

The closed-loop model becomes

$$(Js^2 + Bs)\Theta_m(s) = -k_p\Theta_m(s) + k_p\Theta_d(s) + k_d s\theta_m(s) - D(s), \tag{4.7}$$

or

$$\Theta_m(s) = \frac{k_p}{Js^2 + (B + k_d)s + k_p}\Theta_d(s) - \frac{1}{Js^2 + (B + k_d)s + k_p}D(s). \tag{4.8}$$

As long as $k_p > 0$, $k_d > 0$, and disturbances are bounded, the closed loop system is stable. Moreover, we can move both poles arbitrarily.

In practice, we can't move the poles wherever due to actuator saturation. An easier way to at least get e_{ss} to be zero is to use integral action.

4.1.4 PID Control

$$u(t) = -k_p(\theta_m(t) - \theta_d(t)) - K_d\dot{\theta}_m(t) - k_I\int_0^t(\theta_m(\eta) - \theta_d(\eta))d\eta, \tag{4.9}$$

$$U(s) = \left(k_p + \frac{k_I}{s}\right)(\Theta_d(s) - \Theta_m(s)) - K_d\Theta_m(s). \tag{4.10}$$

$$\Theta_m(s) = \frac{k_p s + k_I}{Js^3 + (B + k_d)s^2 + k_p s + k_I}\Theta_d(s) - \frac{s}{Js^3 + (B + k_d)s^2 + k_p s + k_I}D(s). \tag{4.11}$$

$$E(s) = \frac{Js^3 + (B + k_d)s^2}{Js^3 + (B + k_d)s^2 + k_p s + k_I}\Theta_d(s) - \frac{s}{Js^3 + (B + k_d)s^2 + k_p s + k_I}D(s). \tag{4.12}$$

Same final value theorem test gives us that $e_{ss} = 0$ for all constant step reference and constant disturbances.

Applying Routh-Hurwitz criterion, we get that $k_p, k_d, k_I > 0$ and

$$k_I < \frac{k_p(B + k_d)}{J} \tag{4.13}$$

4.1.5 FeedForward Control

The control approaches so far worked for constant references and disturbances. What happens when the reference is time varying? One approach is to use a feedforward control input.

Let the plant be $G(s)$, the controller be $H(s)$, and feedforward transfer function be $F(S)$. Then,

$$U(s) = F(s)\Theta_d(s) + H(s)\Theta_d(s) \tag{4.14}$$

One can show that if $F(s) = 1/G(s)$ and $F(s)$ is stable, then $E(s) = \Theta_d(s) - Y(s) = 0$.

Let

$$G(s) = \frac{q(s)}{p(s)}, \qquad H(s) = \frac{c(s)}{d(s)}, \text{ and} \qquad F(s) = \frac{a(s)}{b(s)}. \tag{4.15}$$

Then

$$T(s) = \frac{\Theta_m(s)}{\Theta_d(s)} = \frac{q(s)\left(c(s)b(s) + a(s)d(s)\right)}{b(s)\left(p(s)d(s) + q(s)c(s)\right)}, \tag{4.16}$$

and

$$\frac{E(s)}{\Theta_d(s)} = \frac{d(s)\left(q(s)a(s) - b(s)p(s)\right)}{b(s)\left(p(s)d(s) + q(s)c(s)\right)}. \tag{4.17}$$

We can get $E(s) \equiv 0$ if $q(s)a(s) - b(s)p(s) = 0$, or

$$\frac{q(s)}{p(s)} = \frac{b(s)}{b(a)} \tag{4.18}$$

$$\implies F(s) = \frac{1}{G(s)} \tag{4.19}$$

The effect of this choice when $H(s)$ is a PD control is that

$$\left(Js^2 + (B + K_d)s + k_p\right) E(s) = -D(s). \tag{4.20}$$

The closed loop system can be modeled as

$$J\ddot{e}(t) + (B + K_d)\dot{e}(t) + k_p e(t) = -d(t). \tag{4.21}$$

If $d(t) = 0$ then $e(t) \to 0$ for $k_p > 0$, $k_d > 0$.

4.1.6 Control Of Flexible Joints

There are two issues that affect the performance of the independent joint control strategies derived so far. The first is the issue of actuator saturation, the second is the effect of flexibility in the actuator or link.

The effect of saturation is to create integrator wind-up, and reduce the rise time in response to step functions. Furthermore, ramps may be untrackable.

The effectiveness of a PD control will depend on whether the signal is from the motor or the load. The short message is:

1. If you use θ_m, you can control θ_m well but are then letting θ_l be driven by passive dynamics.
2. If you use θ_l you can control θ_l but you have to be less aggressive to not excite a resonant feedback due to the spring.

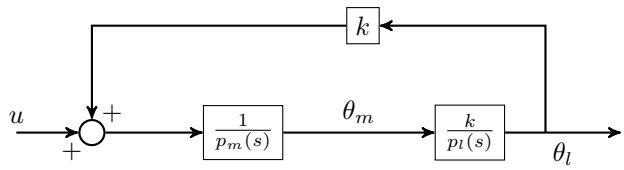


Figure: Model from voltage ($u = K_m V/R$) to load angle θ_l.

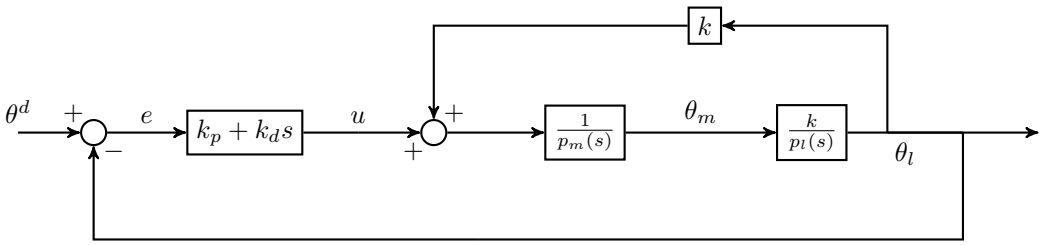


Figure: Regulation using PD Control when sensor reads load angle θ_l.

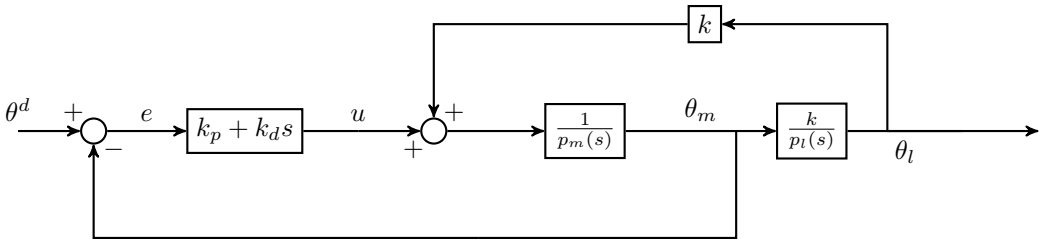


Figure: Regulation using PD Control when sensor reads motor angle θ_m.

A single measurement of only one out of θ_m or θ_l might not be suitable. If you use feedback from both θ_m and θ_l, you create four gains that are hard to design using typical frequency-domain techniques. To formulate this control, we use the state space models in (3.61).

Note that the input u is a scalar. If we know all elements of x, we can choose a linear feedback

$$u = -k^T x + r \tag{4.22}$$

where r is the reference. This system is controllable, we can choose k to assign the poles of $A - Bk^T$ however we place, as long as complex poles have their conjugates as poles.

4.2 Multivariable Control

We derived an independent joint model for the robot dynamics, encompassing link and actuators, and designed controllers based on this model. Instead, we may consider the full dynamics model which combines the Euler-Lagrangian model for the robot link dynamics with the actuator models to obtain an equation that looks like

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B\dot{q} + G(q) = u, \tag{4.23}$$

where u denotes the input due to the voltage, whereas τ was the torque acting on the link joint. In particular,

$$u_k = r_k \frac{K_{m_k}}{R_k} V_k,$$

where $\theta_{m_k} = r_k q_k$, and $M(q) = D(q) + J$, and J is diagonal with $r_k^2 J_{m_k}$ as k^{th} diagonal element.

4.2.1 PD+ Control

In the absence of gravity, a PD control for set-point tracking works for the full dynamics too! However, we show this using a Lyapunov function instead of the final value theorem. Let $u = -K_P(q - q_d) - K_D\dot{q} + G(q)$. The Lyapunov function we use is

$$V = \frac{1}{2}\dot{q}^T M(q)\dot{q} + \frac{1}{2}(q - q_d)^T K_P (q - q_d) \tag{4.24}$$

$$\begin{aligned}
\dot{V} &= \dot{q}^T M(q)\ddot{q} + \frac{1}{2}\dot{q}^T \dot{M}(q)\dot{q} + \dot{q}^T K_P(q - q_d) \\
&= \dot{q}^T \left(u - C(q,\dot{q})\dot{q} + K_P(q - q_d) - G(q) - B\dot{q}\right) + \frac{1}{2}\dot{q}^T \dot{M}(q)\dot{q} \\
&= \dot{q}^T \left(u + K_P(q - q_d) + G(q) - B\dot{q}\right) + \frac{1}{2}\dot{q}^T \left(\dot{M}(q) - 2C\right)\dot{q} \\
&= -\dot{q}^T (K_D + B)\dot{q}
\end{aligned} \tag{4.25}$$

Using La Salle's invariance principle, the fact that $\dot{V} \leq 0$ enables us to conclude that $\dot{V} \to 0$ so that $q \to q_d$ and $\dot{q} \to 0$.

La Salle's invariance principle states that for a candidate Lyapunov function $V(x)$ (continuous, positive definite) if $\dot{V} \leq 0$ then solutions $x(t)$ from all initial conditions will approach the largest set invariant set $M = \{x \in \mathbb{R}^n : V(t) \equiv 0\}$.

For (4.23) with a PD control feedback and Lyapunov function (4.24), $\dot{V} \equiv 0 \implies \dot{q} \equiv 0 \implies \ddot{q} \equiv 0 \implies u = -K_P(q - q_d) \equiv 0 \implies q \equiv q_d$.

4.2.2 Inverse Dynamics Control

Consider

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u, \tag{4.26}$$

Let $u = M(q)a_q + C(q,\dot{q})\dot{q} + G(q)$. The closed loop simply becomes

$$M(q)\ddot{q} = M(q)a_q \implies \ddot{q} = a_q \tag{4.27}$$

We can choose a_q to be $a_q = \ddot{q}(t) + K_P(q_d(t) - q(t)) + K_D(\dot{q}_d(t) - \dot{q}(t))$. If $e(t) = q(t) - q_d(t)$, then $\ddot{e} + K_D\dot{e} + K_P e = 0$. It is straightforward to choose gains K_D and K_P so that $e(t) \to 0$ as $t \to \infty$.

4.2.3 Task Space Inverse Dynamics Control

Let X be the end-effector pose with orientation given by a minimal representation of $SO(3)$. Then,

$$\ddot{X} = J_a(q)\ddot{q} + \dot{J}_a(q)\dot{q} \tag{4.28}$$

If we choose

$$a_q = J_a(q)^{-1}\left(a_X - \dot{J}_a(q)\dot{q}\right) \tag{4.29}$$

then the joint space inverse dynamics control implies a task space dynamics of

$$\ddot{X} = a_X \tag{4.30}$$

and we can now track task space trajectories $X_d(t)$. The caveat is that $J_a(q)$ must be non-singular. If the task is not the full end-effector pose, but coordinates of smaller size, Jacobian pseudoinverses may be used.

4.2.4 Robust Inverse Dynamics Control

The issue with inverse dynamics control is that the guarantees assume perfect cancellation of nonlinear dynamics to obtain the linearized system. When the model is not perfectly known, we want our control performance to be robust to the errors, and perhaps adapt to them as well.

Consider the true system

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u. \tag{4.31}$$

We design our control based on the assumed system Consider

$$\hat{M}(q)a_q + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) = u. \tag{4.32}$$

We model the closed loop as

$$\ddot{q} = a_q + \eta(q,\dot{q},\ddot{q},a_q) \tag{4.33}$$

where

$$\begin{aligned}\eta(q,\ddot{q},a_q) &= M(q)^{-1}\left((M(q) - \hat{M}(q))a_q + (C(q,\dot{q}) - \hat{C}(q,\dot{q}))\dot{q} + G(q) - \hat{G}(q)\right)\\ &= M^{-1}\left(\tilde{M}a_q + \tilde{C}\dot{q} + \tilde{G}\right)\\ &= Ea_q + M^{-1}\left(\tilde{C}\dot{q} + \tilde{G}\right)\end{aligned} \tag{4.34}$$

Let $e = (\tilde{q}, \tilde{\dot{q}})$. Selecting $a_q = \ddot{q}_d(t) - K_0\tilde{q} - K_1\tilde{\dot{q}} + \delta a$, where the last term is to be designed, we get

$$\dot{e} = \begin{bmatrix} 0 & I \\ -K_0 & -K_1 \end{bmatrix} e + \begin{bmatrix} 0 \\ I \end{bmatrix}(\delta a + \eta) \tag{4.35}$$

Suppose we can bound η as $\|\eta\| \le \rho(e,t)$, we can then design δa to guarantee ultimate boundedness of e.

Let $V = e^T P e$ where $A^T P + PA = -Q$. Since A can be made Hurwitz by choosing K_0 and K_1, we know that for each $Q > 0$ there exists $P > 0$ that satisfies the Lyapunov equation.

We have that

$$\begin{aligned}\dot{V} &= e^T PAe + e^T A^T Pe + 2e^T PB(\delta a + \eta) \\ &= -e^T Qe + 2e^T PB(\delta a + \eta)\end{aligned} \tag{4.36}$$

We choose

$$\delta a = \begin{cases} -\rho(e,t)\frac{B^T Pe}{\|B^T Pe\|} & , \quad \text{if } \|B^T Pe\| \neq 0 \\ 0 & , \quad \text{if } \|B^T Pe\| = 0 \end{cases} \tag{4.37}$$

Let $w = B^T Pe$. Then the second term in (4.36) is

$$\begin{aligned} w^T\left(-\rho\frac{w}{\|w\|} + \eta\right) &\leq -\rho\|w\| + \|w\|\|\eta\| && (w^T\eta \leq \|w\|\|\eta\|) \\ &\leq \|w\|(-\rho + \|\eta\|) \\ &\leq 0 && (\|\eta\| \leq \rho(e,t)) \end{aligned}$$

So, $\dot{V} \leq -e^T Qe < 0$.
All of this works if $\|\eta\| \leq \rho(e,t)$. To define such a bound, consider

$$\begin{aligned} \eta &= Ea_q + M^{-1}\left(\tilde{C}\dot{q} + \tilde{G}\right) \\ &= E\delta a + E(\ddot{q}_d(t) - K_0\tilde{q} - K_1\tilde{\dot{q}}) + M^{-1}\left(\tilde{C}\dot{q} + \tilde{G}\right) \\ &\implies \|\eta\| \leq \alpha\|\delta a\| + \gamma_1\|e\| + \gamma_2\|e\|^2 + \gamma_3 \end{aligned} \tag{4.38}$$

If we can find $\|E\| = \alpha < 1$, and constants γ_i above, we can define

$$\rho(e,t) = \frac{1}{1-\alpha}\left(\gamma_1\|e\| + \gamma_2\|e\|^2 + \gamma_3\right). \tag{4.39}$$

For $E = M^{-1}\tilde{M} = M^{-1}\hat{M} - 1$ we can always ensure $\alpha < 1$ by defining $\hat{M}$ as

$$\hat{M} = \frac{2}{\overline{M} + \underline{M}} I \tag{4.40}$$

where $\underline{M} \leq \|M^{-1}\| \leq \overline{M}$.

Continuous Robust Control

The robust controller (4.37) is discontinuous, but ensures that $e(t) \to 0$. We can use a continuous approximation at the cost of only being able to show that the errors are uniformly ultimately bounded (UUB). An open ball $B_r(y) \in \mathbb{R}^n$ is a set $\{x \in \mathbb{R}^n : \|x - y\| < r\}$.

A system is UUB with respect to ball $B_r(0)$ if for every initial condition the error $e(t)$ there exists $T < \infty$ such that $e(t) \in B_r(0)\ \forall t \geq T$. The trouble is this ultimate bound becomes large when $\rho(e,t)$ is large.

$$\delta a = \begin{cases} -\rho(e,t)\frac{B^T Pe}{\|B^T Pe\|} & , \quad \text{if } \|B^T Pe\| > \epsilon \\ -\frac{\rho(e,t)}{\epsilon}B^T Pe & , \quad \text{if } \|B^T Pe\| \leq \epsilon \end{cases} \tag{4.41}$$

So, $\dot{V} = -e^T Qe + 2w^T(\delta a + \eta)$. When $\|w\| \leq \epsilon$

$$\begin{aligned} \dot{V} &= -e^T Qe + 2w^T(\delta a + \eta) \\ &\leq -e^T Qe + 2w^T(-\frac{\rho}{\epsilon}w + \rho\frac{w}{\|w\|}) \\ &\leq -e^T Qe - 2\frac{\rho}{\epsilon}\|w\|^2 + 2\rho\|w\| \end{aligned} \tag{4.42}$$

which is clearly maximized at $\|w\| = \epsilon/2$

Thus

$$\dot{V} \leq -e^T Q e + \epsilon \frac{\rho}{2} \tag{4.43}$$

We want to find the smallest ball in error coordinates e outside of which $\dot{V} < 0$. Clearly, $\dot{V} < 0$ when $e^T Q e > \epsilon\rho/2$. Since $e^T Q e \geq \lambda_{min}(Q)\|e\|^2$, $\dot{V} \geq 0$ when $\lambda_{min}(Q)\|e\|^2 \geq \epsilon\rho/2$. So, $\dot{V} < 0$ outside of the set

$$\delta = \left(\frac{\epsilon\rho}{2\lambda_{min}(Q)} \right)^{1/2} \tag{4.44}$$

The UUB ball comes from the smallest ball containing the smallest level set that contains $B_\delta(0)$.

4.2.5 Adaptive Inverse Dynamics Control

The error in model estimate affects $\rho(\epsilon, t)$ which ruins the lowest achievable error. Ideally, we want smaller model errors to achieve lower error. Luckily, we can learn models on-the-fly using adaptive control theory.

The idea is to create a dynamical system whose state is the parameters we want to estimate. We feed it an input that makes the estimated parameters reach a set that permits the state errors to converge.

We want to control the system

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Y(q, \dot{q}, \ddot{q})\theta = u. \tag{4.45}$$

Choosing $u = Y(q, \dot{q}, a_q)\hat{\theta}$, where $a_q = \ddot{q}_d(t) - K_0\tilde{q} - K_1\dot{\tilde{q}}$ we get

$$\ddot{\tilde{q}} + K_1\dot{\tilde{q}} + K_0\tilde{q} = M^{-1}Y(q, \dot{q}, \ddot{q})\tilde{\theta} = \Phi\tilde{\theta}, \tag{4.46}$$

where $\tilde{\theta} = \hat{\theta} - \theta$.

We get the ODE

$$\dot{e} = Ae + B\Phi\tilde{\theta} \tag{4.47}$$

with same matrices as in the robust case.

How should we pick $\hat{\theta}$, given that we don't know θ? Consider a function of $e, \hat{\theta}, \theta$ given by

$$V = e^T P e + \tilde{\theta}^T \Gamma \tilde{\theta}. \tag{4.48}$$

For $P > 0$ and $\Gamma > 0$, $V = 0$ when $e = 0$ and $\theta = \hat{\theta}$.

Let K_0 and K_1 be chosen so that A is Hurwitz. Implies there exists $Q > 0$ such that $A^T P + PA = -Q$. We have

$$\dot{V} = -e^T Q e + 2\tilde{\theta}^T \left(\Phi^T B^T P e + \Gamma\dot{\hat{\theta}} \right) \tag{4.49}$$

If we knew θ we'd ensure that the second term was negative definite. However, since we don't, we set the second term to zero, by choosing

$$\dot{\hat{\theta}} = -\Gamma^{-1}\Phi^T B^T P e \tag{4.50}$$

It's like a nonlinear integral control!

Analysis We have that $\dot{V}$ is non-positive and is the square of a term. Therefore $V(t) - V(t_0) \leq \int_0^T e^T(s) Q e(s) ds < \infty$. This makes $e(t)$ a square integrable function. Now, if we can show that its derivative $\dot{e}(t)$ is bounded, we can show that $e(t) \to 0$.

Lemma 1 (Barabalat). *Suppose $f : \mathbb{R} \mapsto \mathbb{R}$ is a square integrable function and further suppose that its derivative $\dot{f}$ is bounded. Then $f(t) \to 0$ as $t \to \infty$.*

So, why is $\dot{e}$ bounded? Since $V(t)$ is bounded so are $e(t)$ and $\tilde{\theta}(t)$. Since $e(t)$ is bounded so are $\tilde{q}$ and $\dot{\tilde{q}}$. This implies that $\ddot{\tilde{q}}$ is bounded, so that $\dot{e}(t)$ is bounded. Requires bounded $\ddot{q}_d(t)$.

4.3 Passivity-Based Control

The fact that robots, when viewed as rigid n-link mechanisms, are passive enables some nonlinear control approaches that have advantages over PD control in joint space.

4.3.1 Potential-Shaping Control

Passivity provides an easy way to achieve a certain type of set-point regulation. If the torques τ include a term of the form $-B\dot{q}$, where $B > 0$, either due to motor friction or controlled damping, we can conclude that

$$\dot{V} = -\dot{q}^T B\dot{q} \leq 0.$$

Where does $q(t)$ reach? If the potential energy $PE(q)$ has a local minimum $q^\star$, then we can show that $q(t) \to q^\star$ at least locally (when $q(0)$ is close to $q^\star$).

If we want $q^\dagger$ to be the equilibrium, where $q^\dagger \neq q^\star$, we just need to define a new potential energy $PE^\dagger(q)$ which has only one minimum at $q^\dagger$, and then use the motor control

$$\tau_m = -B\dot{q} + \frac{\partial PE(q)}{\partial q} - \frac{\partial PE^\dagger(q)}{\partial q},$$

and the storage function

$$V(q, \dot{q}) = KE(q, \dot{q}) + PE^\dagger(q),$$

to arrive at the conclusion that $q(t) \to q^\dagger$.

Note that this analysis does not account for the presence of any interaction force F.

4.3.2 Passivity-based Tracking

Suppose we want to track the trajectory $q_d(t)$. Let

$$e = \begin{bmatrix} q(t) - q_d(t) \\ \dot{q}(t) - \dot{q}_d(t) \end{bmatrix} = \begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{bmatrix}.$$

Assume that the model of the robot is

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + \frac{\partial PE(q)}{\partial q} = \tau_m,$$

We define a controller as

$$\tau_m = M(q)a + C(q, \dot{q})v + \frac{\partial PE(q)}{\partial q} - Kr, \text{ where} \tag{4.51}$$

$$v = \dot{q}_d - \Lambda\tilde{q} \tag{4.52}$$

$$a = \dot{v} = \ddot{q}_d - \Lambda\dot{\tilde{q}} \tag{4.53}$$

$$r = \dot{q} - v = \dot{\tilde{q}} + \Lambda\tilde{q}, \tag{4.54}$$

with K, Λ diagonal matrices of positive gains.

The closed loop becomes

$$M(q)\dot{r} + C(q, \dot{q})r + Kr = 0 \tag{4.55}$$

Choose a storage function $V(q, \dot{q})$, which in this case is also a Lyapunov function:

$$V = \frac{1}{2}r^T M(q)r + \tilde{q}^T \Lambda K\tilde{q} \tag{4.56}$$

$$\begin{aligned}\dot{V} &= -r^T K r + 2\tilde{q}^T \Lambda K \dot{\tilde{q}} + \frac{1}{2} r^T \left(\dot{M}(q) - 2C \right) r \\ &= -\tilde{q}^T \Lambda^T K \Lambda \tilde{q} - \dot{\tilde{q}}^T K \dot{\tilde{q}} \\ &= -e^T Q e\end{aligned} \tag{4.57}$$

If $M(q)$ is bounded then we can conclude that $e = 0 \iff V = 0$ so that the origin is GAS. Again, our analysis assumes that $F = 0$.

The conclusion of global asymptotic stability seems to suggest passivity-based control has no advantage over inverse dynamics control, which achieves the same behavior. The advantage appears in the robust and adaptive control approaches, where the constraints on $M(q)$ are relaxed. Robust control required bounds on the error and mass matrix terms. Implementing the adaptive controller requires knowledge of acceleration (due to Φ) and invertible $M(q)$.

4.3.3 Passivity-Based Robust Control

The control is

$$u = \hat{M}(q)a + \hat{C}(q, \dot{q})v + \hat{G}(q) - Kr = Y(q, \dot{q}, a, v)\hat{\theta} - Kr \tag{4.58}$$

The closed-loop becomes

$$M(q)r + C(q, \dot{q})r + Kr = Y(\hat{\theta} - \theta) \tag{4.59}$$

Let $\hat{\theta} = \theta_0 + \delta\theta$ where $\|\tilde{\theta}\| = \|\theta - \theta_0\| \leq \rho$. The same Lyapunov candidate as in for passivity gives

$$\dot{V} = -e^T Q e + r^T Y(\delta\theta + \tilde{\theta}) \tag{4.60}$$

The same UUB analysis goes through where $w = r^T Y$, $\delta\theta = \delta a$ and $\tilde{\theta} = \eta$. Note that the uncertainty characterization is simpler, and prior knowledge is easily baked in.

4.3.4 Passivity-Based Adaptive Control

The closed-loop is again

$$M(q)r + C(q, \dot{q})r + Kr = Y(\hat{\theta} - \theta) \tag{4.61}$$

The Lyapunov function is

$$V = \frac{1}{2} r^T M(q) r + \tilde{q}^T \Lambda K \tilde{q} + \frac{1}{2} \tilde{\theta}^T \Gamma \tilde{\theta} \tag{4.62}$$

and the update law becomes

$$\dot{\hat{\theta}} = -\Gamma^{-1} Y(q, \dot{q}, a, v) r \tag{4.63}$$

Again we see that

$$\dot{V} = -\tilde{q}^T \Lambda^T K \Lambda \tilde{q} - \dot{\tilde{q}}^T K \dot{\tilde{q}} - \tilde{\theta}^T (Y^T r + \Gamma \dot{\hat{\theta}}) \tag{4.64}$$

As in the previous adaptive control analysis, we use Barbalat's Lemma to conclude $e(t) \to 0$ and $\|\tilde{\theta}\|$ remains bounded.

4.3.5 Passivity-based Interaction

When $F \neq 0$, we cannot ensure tracking using the previous arguments, because

$$\begin{aligned}\dot{V} &= \dot{q}^T \tau_m + \dot{q}^T J^T(q) F && (4.65)\\ &= \dot{q}^T \tau_m + \xi^T F, && (4.66)\\ & && (4.67)\end{aligned}$$

where ξ is the velocity of the contact frame, in which the contact force is F.

Like impedance control, if we know how the environment behaves (the force it generates at the contact in response to motion of the contact), we may be able to choose τ_m intelligently to achieve force tracking or position tracking.

Passivity control allows us a different type guarantee: if the environment is passive, then the robot-environment can be made stable by choosing τ_m to make the robot passive, **without knowing anything else about the environment, like its impedance**.

Let the robot and environment storage functions be V_r and V_e respectively. If the only interaction is at the robot-environment contact, we get the the supply rate to the robot is

$$S_r = \dot{q}^T \tau_m + \xi^T F.$$

The important idea is that the supply rate of the environment is

$$S_e = -\xi^T F.$$

Therefore, we can define a new storage function

$$V_{re} = V_r + V_e,$$

and see that

$$\dot{V}_{re} = \dot{q}^T \tau_m.$$

Assuming that the storage functions are bounded when the state is bounded, we can make the system stable by choosing $\tau_m = 0$.

Saying more than that requires us to know the storage function and supply rate of the environment. If we know that, for example, that $F = K\xi$, which would destabilize the system, we may choose $\tau_m = -\rho(\xi) K \dot{q}$ where $\rho(\xi) \geq \|\xi\|^2$.

4.4 Force Control

The robot control tasks we have focused on involve a desired, possibly time-varying, position of the end effector. This end-effector trajectory (task-space trajectory) dictates a trajectory $q(t)$ for the robot joint coordinates q. Many tasks are sufficiently characterized by the position of the end-effector.

In some cases, the task for the robot involves generating desired forces rather than just positions. Simple examples involve pushing delicate objects on a table-top, polishing or grinding, assembly tasks, throwing a ball.

We may achieve force control in two ways:

1. Directly through measurement of the applied force and error-based feedback
2. Indirectly through changes in static configuration.

4.4.1 Force-based Force Control

Consider the usual equations $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + J^T F_{tip}(t)$. If we can measure $F_{tip}(t)$ through some sensor, and have a desired force F_d at the end-effector, we can construct an error $F_e = F_d - F_{tip}(t)$.

Force Sensors. There are typically three locations for placing sensors that measure the forces acting on the robot. They are the wrist, the joints, and the end-effector. The wrist sensor is usually a force-and torque sensor placed between the end-effector and the final robot link. A force sensor measures the torques about the actuator shaft. The end-effector sensors are often tactile sensors placed on the fingers of grippers.

Let's choose the control

$$\tau = G(q) - J^T F_d - J^T \left(K_p F_e + K_i \int F_e(s) ds \right) \tag{4.68}$$

The closed-loop equations become

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + J^T F_{tip}(t) \tag{4.69}$$

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = G(q) - J^T F_d - J^T \left(K_p F_e + K_i \int F_e(s)ds \right) + J^T F_{tip}(t) \tag{4.70}$$

In general, this system is hard to analyze. For quasi-static motions, where $\ddot{q} \approx 0$ and $\dot{q} \approx 0$, we get

$$(\approx 0) + (\approx 0) + G(q) = G(q) - J^T F_d - J^T \left(K_p F_e + K_i \int F_e(s)ds \right) + J^T F_{tip}(t) \tag{4.71}$$

$$\implies 0 = J^T \left((I + K_p)F_e + K_i \int F_e(s)ds \right) \tag{4.72}$$

$$\implies 0 = (I + K_p)\dot{F}_e + K_i F_e \quad \text{(if } J(q) \text{ is non-singular)} \tag{4.73}$$

So, if $\ddot{q}(t) \approx 0$, $\dot{q}(t) \approx 0$, $J(q)$ is non-singular, $K_p \geq 0$ and $K_d > 0$, then $F_e \to 0$. As you might guess, direct force control is difficult to achieve in practice.

Problems: There's a potential contradiction where we apply a time-varying torque $\tau(t) \neq 0$ at the robot's joints but need $\ddot{q}, \dot{q} = 0$. This situation might exist when the end-effector is in contact with something that doesn't move much. By contrast, what happens when the end-effector loses contact? The measured force drops to zero, and the end-effector accelerates due to F_e! Unexpected changes in contact turn out to be disastrous for force controllers intended to work on a particular contact configuration. These issues make force control unpopular except for very structured situations, requiring advanced methods.

Partial Solution: Add damping $-K_d\dot{q}$ to achieve slow motion

4.4.2 Configuration-based Force Control

Consider the usual equations $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + J^T F_{tip}(t)$. When the robot is stationary, and the end-effector is in contact with a surface, we obtain

$$G(q) = \tau + J^T(q) F_{tip}(t) \tag{4.74}$$

The idea is to then solve this equation, say for q^* and τ^*, when $F_{tip}(t) = F_d$, a desired force, and then design a controller τ that stabilizes the configuration and torque at these values. That is, $q(t) \to q^*$ and $\tau(t) \to \tau^*$. For example,

$$\tau = \underbrace{G(q^*) - J^T(q^*)F_d}_{\tau^*} + \underbrace{K_p(q - q^*) + K_d(\dot{q})}_{\text{stabilization}}.$$

Task: Analyze this control law.

Note that this approach does not monitor F_{tip}, but focuses on configuration q.

4.4.3 Coordinate Frames and Constraints

The forces acting on a robot often come about due to contact with the environment. This contact occurs at specific surfaces and points, and represent position constraints. The forces acting on the robot then arise as reactions to the existence of these constraints. For example, your finger moving in free space would not sense any pressure at the finger tip, until it is pressed against a surface. Even when you sense a pressure due to a force, the magnitude depends on whether the surface is pushing on you, or whether you are pushing on the object, versus maintaining a light stationary contact.

Reciprocal Bases

In more formal descriptions of mechanics, the linear and angular velocity $\xi = (v, \omega)$ and force and moment $F = (f, n)$ are considered dual to each other. The quantities ξ and F are called twists and wrenches respectively, and are both six-dimensional.

Given a configuration space corresponding to a mechanical system, twists belong to the tangent space $\mathcal{M}$ and wrenches belong to the co-tangent space $\mathcal{F}$, and their product corresponds to power. The numerical

value of the power depends on the bases we choose for $\mathcal{M}$ and $\mathcal{F}$. For consistency, the power we calculate must be the same for any pair of bases we choose, if these two pairs of bases are linearly related. This consistency is achieved by using reciprocal bases.

Definition 11 (Reciprocal Bases)**.** If $\{e_1, \ldots, e_6\}$ is a basis for $\mathcal{M}$ and $\{f_1, \ldots, f_6\}$ is a basis for $\mathcal{F}$, these two bases are *reciprocal* if

$$e_i^T f_j = \begin{cases} 1 & , \quad \text{if } i = j \\ 0 & , \quad \text{if } i \neq j. \end{cases} \tag{4.75}$$

Definition 12. A twist $\xi \in \mathcal{M}$ and wrench $\mathbb{F} \in \mathcal{F}$ are called *r*eciprocal if

$$\xi^T F = v^T f + \omega^T n = 0 \tag{4.76}$$

The advantage of using reciprocal bases for $\mathcal{M}$ and $\mathcal{F}$ is that the product $\xi^T F$ has the invariance we want. Therefore, the reciprocity condition (12) is invariant with respect to the choice of reciprocal bases.

Natural and Artificial Constraints

We discuss natural constraints which are defined using (12). The intuition is that the power consumed by a twist and wrench to satisfy a natural constraint is zero. A natural constraint typically comes from the environment the robot is interacting with. The total power consumed by such a twist and wrench may be non-zero, but none of that power consumption is due to the natural constraint.

Natural constraints in turn define artificial constraints, which arise due to specifying reference values for input motion and force control tasks. These are constraints we impose on the motion to complete a given task. The natural constraints together with the artificial constraint provide a complete reference for ξ and F.

Compliance frame A compliance frame $o_c x_c y_c z_c$ (also called a constraint frame) is a frame in which the task is easily described. For example, consider

1. inserting a peg into a hole, or
2. turning a crank.

Peg We choose an orthonormal bases for both $\mathcal{M}$ and $\mathcal{F}$, with respect to a compliant frame with origin at the bottom and center of peg, z parallel to peg axis.

4.5 Network Models and Impedance

Introduction The reciprocity condition $\xi^T F = 0$ mean that the forces of constraint do no work in directions of motion compatible with motion constraints. This property holds under ideal conditions of no friction and perfectly rigid robot and environment. In practice, compliance and friction alter the nature of constraints and constraint forces.

For example, when a robot pushes against a compliant surface, the contact experiences both non-zero normal reaction forces and non-zero motion, so that the work $\xi^T F$ is non-zero. If the stiffness of the surface is k, then the force is kx, and the total work done is

$$\begin{aligned} W &= \int_0^t \dot{x}(u) k x(u) du \\ &= \int_0^t \frac{d}{du} \frac{1}{2} k x(u)^2 du \\ &= \frac{1}{2} k (x(t)^2 - x(0)^2) \end{aligned} \tag{4.77}$$

The work done increases with k, as does the force required to induce a velocity $\dot{x}$. This impact of stiffness on the relationship between force, velocity and energy is captured by the more general notion of impedance.

4.5.1 One-Port Model

The robot and environment are modeled as one-port nodes in a network (See main text for details). Each node has two interaction ports and corresponding port variables **effort** F_i and **flow** V_i. The relationship between these port variables depend on the dynamics of the system.

The interaction between the robot and environment is then modeled as a network of such one-ports, where the connections between nodes occur at the interaction ports of the nodes, and flow and effort are transmitted between interacting nodes. For a network with a robot and environment, the flow and effort variables are V_r, V_e and F_r, F_e respectively. The power consumed or dissipated by the network is $V^T F$.

4.5.2 Impedance

The relationship between flow and effort for a system is given by the impedance operator of that system. For linear systems, we have the definition below

Definition 13. For a one-port network, the impedance $Z(s)$ is

$$Z(s) = \frac{F(s)}{V(s)}.$$

Example 2. For a S-M-D with dynamics $M\ddot{x} + B\dot{x} + Kx = F$, we have

$$Z(s) = Ms + B + \frac{K}{s}. \tag{4.78}$$

The task space inverse dynamics control approach allows us to think of impedance of a robot in terms of (4.78), even though the relationship between velocity and force for the full model (4.23) is quite complex.

Classification of Impedance Operators

1. Inertial: iff $|Z(0)| = 0$
2. Resistive: iff $0 < |Z(0)| < \infty$
3. Capacitive: iff $|Z(0)| = \infty$

See (4.78) for example of each kind (mass, damper, spring, respectively).

The impedance of the robot determines the force response of the end-effector to a velocity input at the end-effector. That is,

$$F(s) = Z(s)V(s). \tag{4.79}$$

Suppose we move the end-effector with a constant reference.

$$F(s) = MsV_r\frac{1}{s} \tag{4.80}$$

Admittance

The reciprocal relationship is often called admittance $Y(s)$

$$Y(s) = \frac{V(s)}{F(s)}.$$

Thevenin and Norton Equivalents

We can represent any one-port network consisting of multiple nodes as an equivalent network containing just one impedance and a source. In a Thevenin equivalent network, the impedance $Z(s)$ is placed in series with a source of effort F_s. In a Norton equivalent network, the impedance $Z(s)$ is placed in parallel to a source of flow V_s. These sources represent references (see artificial constraints) or external disturbances.

4.5.3 Task Space Dynamics

The interaction between manipulator and environment is easier to describe in the task space rather than the joint space. Given a system

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B\dot{q} + G(q) + J^T(q)F_e = u, \tag{4.81}$$

we can impose the control

$$u = M(q)a_q + C(q,\dot{q})\dot{q} + B\dot{q} + G(q) + J^T(q)a_f, \tag{4.82}$$

to obtain the closed loop

$$\ddot{q} = a_q + M(q)^{-1}J^T(q)(F_e - a_f). \tag{4.83}$$

We can choose a_q, a desired acceleration in the joint space, as

$$a_q = J_a(q)^{-1}\left(a_X - \dot{J}_a(q)\dot{q}\right). \tag{4.84}$$

where a_X is the desired acceleration in the task space X e. Since

$$\ddot{X} = J(q)\ddot{q} + \dot{J}(q)\dot{q}, \tag{4.85}$$

the closed-loop task space dynamics are

$$\ddot{X} = a_X + W(q)(F_e - a_f), \tag{4.86}$$

where $W(q) = J(q)M^{-1}(q)J(q)$ is the mobility tensor.

If $W(q)$ is non-singular, then we set $a_f = F_e$, and achieve any force regulation task by modifying a_X appropriately. Therefore, we design impedance controllers for the system

$$\ddot{X} = a_X. \tag{4.87}$$

4.5.4 Impedance Control

When a robot interacts with physical objects, defining the interaction by impedance of the robot is often easier than specifying the exact motion of the robot end-effector relative to the environment.

Example 3 (Apparent Inertia)**.** we can control a system $M\ddot{x} = u - F$ so that it appears to us as a lighter mass $m' < M$ by using the control $u = -mF$, $m > 0$. The new apparent mass is $m' = \frac{M}{1+m}$.

In general, suppose we want the system (4.23) to behave like a system with inertia, damping, and stiffness in the **task space** given by

$$M_d\ddot{x} + B_d\dot{x} + K_d x = F. \tag{4.88}$$

We would need to implement a task space inverse dynamics control, and choose a_X as

$$a_x = \ddot{x} - M_d^{-1}(B_d\dot{x} + K_d x + F) \tag{4.89}$$

Note that the term F corresponds to the earlier point of setting $a_f = F_e$ and using a_X to implement any force feedback.

Our closed-loop model (4.88) is a spring-mass-damper model with desired parameters. When there is no external force ($F = 0$), clearly $x \to 0$ for $M_d, B_d, K_d > 0$. Instead, we can derive an impedance control relative to a trajectory $x^d(t)$, so that our impedance control achieves trajectory tracking, however the reaction of the system to disturbance forces F behaves like our target system.

4.5.5 Hybrid Impedance Control

When we fix the robot impedance, the velocity or force at the contact depends on what the environment does. Instead, we'd like to fix one of the robot's force or velocity to a reference value.

To achieve this goal, we need to exploit the environment's characteristics when possible.

The main achievement of impedance control is that we do not have to know anything about the environment beyond one simple thing: is $Z_e(0) = 0$ or is $Z_e(0) = \infty$? That knowledge is enough to design a controller that achieved our goals (position or force-tracking) no matter what the interaction force happens to be.

4.6 Optimal Control

If we keep seeing that the control isn't able to follow the trajectory returned by the previous methods, perhaps we need to constrain trajectories using control. This approach leads to an optimal control problem.

In continuous time, we have

$$\begin{aligned} \min \quad & J(q(t), u(t)) \\ \text{subject to} \quad & q(t) \text{ satisfies dynamics and state constraints} \\ & u(t) \text{ satisfies input constraints} \end{aligned}$$

We may also formulate discrete time versions of this problem.

4.6.1 Linear Quadratic Regulator

This section is inspired by Sergey Levine's slides. When time is discrete, the dynamics are linear, and the cost function is quadratic in state and control, the optimal control problem may be solved in a straightforward way. Consider a finite time horizon $t \in \{0, 1, 2, \ldots, T\}$.

At each time $t \in \{0, 1, 2, \ldots, T\}$, we have

$$\mathbf{x}_{t+1} = A_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + a_t; \quad c_t(\mathbf{x}_t, \mathbf{u}_t) = \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{C}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{c}_t$$

Let $J = \sum_{t=0}^{T} c_t(x_t, u_t)$

Now, at time T, we have only one decision to make: pick u_T. The cost of doing so is exactly $c_T(\mathbf{x}_T, \mathbf{u}_T)$ The cost for the first $T-1$ time steps are some value that is effectively constant at time T, so that the total cost will be $\mathbf{Q}_T(x_T, u_T)$

$$\mathbf{Q}_T(\mathbf{x}_T, \mathbf{u}_T) = \text{const} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_T \\ \mathbf{u}_T \end{bmatrix}^T \mathbf{C}_T \begin{bmatrix} \mathbf{x}_T \\ \mathbf{u}_T \end{bmatrix} + \begin{bmatrix} \mathbf{x}_T \\ \mathbf{u}_T \end{bmatrix}^T \mathbf{c}_T$$

To find the best u_T, we minimize this expression. It's gradient w.r.t. u_T is

$$\nabla_{u_T} \mathbf{Q}_T(x_T, u_T) = x_T^T \mathbf{C}_{\mathbf{x}_T, \mathbf{u}_T} + u_T^T \mathbf{C}_{\mathbf{u}_T, \mathbf{u}_T} + \mathbf{c}_{\mathbf{u}_T}^T, \text{ where}$$

$$\mathbf{C}_T = \begin{bmatrix} \mathbf{C}_{\mathbf{x}_T, \mathbf{x}_T} & \mathbf{C}_{\mathbf{x}_T, \mathbf{u}_T} \\ \mathbf{C}_{\mathbf{x}_T, \mathbf{u}_T} & \mathbf{C}_{\mathbf{u}_T, \mathbf{u}_T} \end{bmatrix}, \quad \mathbf{c}_T = \begin{bmatrix} \mathbf{c}_{\mathbf{x}_T} \\ \mathbf{c}_{\mathbf{u}_T} \end{bmatrix}.$$

Setting $\nabla_{u_T} Q_T(x_T, u_T) = 0$ we obtain

$$\mathbf{u}_T = -\mathbf{C}_{\mathbf{u}_T, \mathbf{u}_T}^{-1} \left(\mathbf{C}_{\mathbf{x}_T, \mathbf{u}_T} \mathbf{x}_T + \mathbf{c}_{\mathbf{u}_T} \right) = \mathbf{K}_T \mathbf{x}_T + \mathbf{k}_T,$$

which is a linear (well, affine) feedback control.

To cut a long story short, we may substitute for $\mathbf{u}_T$ in $\mathbf{Q}_T(\mathbf{x}_T, \mathbf{u}_T)$, and we will see that

$$\mathbf{Q}_T(\mathbf{x}_T, \mathbf{K}_T \mathbf{x}_T + \mathbf{k}_T) = V(\mathbf{x}_T) = \mathbf{x}_T^T \mathbf{V}_T \mathbf{x}_T + \mathbf{x}_T^T \mathbf{v}_T,$$

for some appropriate matrix $\mathbf{V}_T$ and $\mathbf{v}_T$ that depends on the problem's parameters.

The main idea is that we can repeat the same step at time $T-1$, with the convenient result that linear dynamics makes $\mathbf{Q}_{T-1}(x_{T-1}, u_{T-1})$ is also a quadratic function of its arguments. So, at time $T-1$ we can expect a linear feedback in $\mathbf{x}_{T-1}$ to be optimal, and the value function $V(x_{T-1})$ will be quadratic in x_{T-1}, and this convenient structure persists backwards in time till $t = 0$. This convenience is easily broken when any of the cost functions are not quadratic, or the dynamics are nonlinear.

This procedure nicely illustrates some of the core ideas in optimal control of dynamical systems. We solve for the best control by moving backwards in time from the final time, by building up an estimate of the cost-to-go (V). The function $Q_t(\mathbf{x}_t, \mathbf{u}_t)$ is known as the Q-function in reinforcement learning, and V is the value function (which is being maximized there, not minimized as we did here).

Chapter 5

Motion Planning

Definition 14 (Path)**.** A path in $\mathbb{R}^n$ is a continuous function γ from the unit interval $I = [0, 1]$ to $\mathbb{R}^n$.

Definition 15 (Trajectory)**.** A trajectory $q(t)$ in $\mathbb{R}^n$ is a a continuous function q from the an interval of time $[t_0, t_f]$ to $\mathbb{R}^n$.

Definition 16 (Graph)**.** A graph G is an ordered pair $G = (V, E)$ where

- V is a set
- E is a set of (ordered) pairs of elements in V

The interpretation of G is that V is a set of nodes, and E describes directed edges that indicate an 'immediate' operation from one node to another.

5.1 Path And Trajectory Planning

The trajectory planning problem can be cast as an optimization problem. Suppose we can measure the 'goodness' of a trajectory by a function J. We want to find a solution $q^*(t)$ of the problem:

$$\min_{q(t)} J(q(t)) \tag{5.1}$$

$$\text{subject to} \quad \text{Robot doesn't destroy itself or things} \tag{5.2}$$

$$\text{Other concerns} \tag{5.3}$$

This version of the problem doesn't worry about control, unlike optimal control formulations. Out pops $q^*(t)$ and we then try and use the trajectory tracking controllers developed in previous controllers which make $q(t) \to q^*(t)$.

These trajectory tracking problems are typically solved using heuristics. One alternative to finding a trajectory $q(t)$ is to start by finding a path. Then, attach time to the path to get a trajectory.

A prototypical approach, with a description of the complexity:

- Figure out the Obstacle-free configuration space (can be very difficult)
- Sample points in free space (easy)
- Connect points in free space (can be difficuly)
  - Potential Field + random walk
  - Probabilistic Road Maps
  - Rapidly-exploring Random Trees
- Attach time to the path formed by connecting points.
  - Use polynomial function of sufficient degree to specify initial point, final point, initial velocity, final velocity, and add accelerations. Finding coefficients given start and end times and values is a linear optimization problem.

Some of these methods are described in the next sections.

Algorithm 1 Gradient Descent

Require: q_0, q_f, $U(q)$, $\epsilon > 0$. $\{U(q)$ must be positive demi-definite$\}$
Ensure: $q_k \approx \min_q U(q)$
  $q^0 \leftarrow q_s$
  $i \leftarrow 0$ {Loop counter}
  while $U(q) > \epsilon$ **do**
    $q^{i+1} \leftarrow q^i - \frac{\alpha^i}{\|\nabla U(q)\|}\nabla U(q)$
    $i \leftarrow i + 1$
  end while
  return $\{q^j\}_{j\in\{1,\dots,i\}}$

5.2 Potential Field Planning

The aim is to define a continuous real-valued function $U\colon \mathcal{Q} \to \mathbb{R}$ where $\mathcal{Q}$ is the configuration space such that configuration q_f is the unique minimum of the U. A gradient descent algorithm for minimizing U generates a sequence of points in the state space corresponding to a path from an initial point q_0 to the final point q_f. The function U is called a potential function, and its gradient defines a force F as

$$F = -\nabla U(q) \tag{5.4}$$

The idea for this approach comes from the behavior of particles with inertia moving in potential fields that imply certain forces acting on the particle.

The simplest way to construct U is to express it as a sum of the form

$$U(q) = U_{att}(q) + U_{rep}(q), \tag{5.5}$$

where U_{att} is an term designed to 'attract' q to q_f. For example,

$$U_{att}(q) = \begin{cases} \frac{1}{2}\|q - q_f\|_2^2 & , \text{ if } \|q - q_f\|_2 \le d \\ d\|q - q_f\|_2 - \frac{1}{2}d^2 & , \text{ if } \|q - q_f\|_2 > d \end{cases} \tag{5.6}$$

The terms U_{rep} will 'repel' the point q from obstacles during the gradient descent. Let $\rho(q)$ be the shortest distance of a point q from any obstacle. One example of such a term is

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2 & , \text{ if } \rho(q) \le \rho_0 \\ 0 & , \text{ if } \rho(q) > \rho_0 \end{cases} \tag{5.7}$$

The function $\rho(q)$ can be discontinuous, unless we define ρ_0 to be small enough.

5.2.1 Gradient Descent

Algorithm 1 implements gradient descent for a function $U(q)$.

5.2.2 Task Space Potentials

Sometimes defining $\rho(q)$ is hard, but defining the obstacles in the task space is easy. In this situation, one may prefer to define a potential field in the task space, compute that gradient, and map the gradient to the configuration space.

We represent the robot by the frame origins $o_i^0(q)$ for $i \in 1, \dots, 6$. Sometimes, we also include additional points to represent parts of the robots not close to these frame origins, to account for the physical space occupied by the robots.

We define potential functions for each point $o_i^0(q)$ and compute the gradient. These gradients represent 'forces' F_i in task space that should act on the frame origins. To obtain the corresponding force in coordinate space, we use the transformation $\tau_i = J_v^T F_i$.

5.3 Probabilistic Road Maps

The probabilistic roadmap planner is a motion planning algorithm in robotics, which solves the problem of determining a path between a starting configuration of the robot and a goal configuration while avoiding collisions.

Motion planning using a PRM involves a construction phase and a query phase.

5.3.1 Construction

The construction phase involves building a graph, which is a set of nodes and a set of edges.

1. Take random samples from the configuration space of the robot
2. Test for membership in free space
3. If in free space, add node to PRM graph
4. select subset of existing nodes based on proximity
5. Connect new node to selected nodes by straight line paths, check for collisions by sampling along these paths
6. If there are no collisions for a path, add corresponding edge to PRM

5.3.2 Query

Once a sufficiently dense PRM is available, one can query the algorithm to connect any initial and goal points. Connect initial and goal points to nearest nodes creating a graph corresponding to that query. Use Djikstra's shortest path algorithm to connect the initial and final nodes in the graph. The paths along the edges of the shortest path yield the planned path.

5.3.3 Analysis

Given certain relatively weak conditions on the shape of the free space, PRM is provably probabilistically complete, meaning that as the number of sampled points increases without bound, the probability that the algorithm will not find a path if one exists approaches zero. The rate of convergence depends on certain visibility properties of the free space, where visibility is determined by the local planner. Roughly, if each point can "see" a large fraction of the space, and also if a large fraction of each subset of the space can "see" a large fraction of its complement, then the planner will find a path quickly.

5.4 RRT

First, a quick summary of RRT(*)

1. A fundamental assumption to these planning methods is that we can represent the task as a sequence of nodes.
2. RRT is about computing (growing) an under-approximating abstraction of a complex state space with a tree topology.
3. Two important questions:
   (a) How to find a new leaf
   (b) Where to connect new leaf to the tree?
4. Loosely, RRT answers 3a by random sampling, and 3b by nearest-node in underlying space (more accurate details below).
5. RRT* improves RRT by using weights for the edges, and ensuring that all paths to leaves are minimum-weight.

6. Informed RRT* answers 3a for a subclass of edge weights, namely, length. For shortest-path problems between two given points, to provably reduce length of current path by sampling, it is necessary to sample a particular ellipsoid defined by points and the current optimal path.

RRT(*) works off of a transition system $TS = (S, Act, \rightarrow, I, AP, L)$ that abstracts the underlying dynamical system whose motion we wish to plan for. The states S are a finite set of points in the state space of the dynamical system. For RRT(*), an action is to attempt to reach another state. Therefore, $Act = S$. The available transitions are encoded in $\rightarrow$. The key point is that TS models a tree, therefore only one action actually results in a transition in each state, to a unique successor node not identical to the current state. I is the root node/state of the tree. For RRT(*), an obvious simple definition of AP is $AP = \{goal, waypoint\}$.

The goal of RRT*is to find a minimum cost path from the initial state I to the goal state in the dynamical system. The goal states are $s \in S$ such that $L(s) = goal$. Note the identification of states in TS and the dynamical system's state space. TS is not fixed, but grown incrementally, one state at a time. These states come from the underlying dynamical system. To justify that TS simulates the dynamical system, RRT*also needs the following functions.

- *sample*
- *nearest neighbor*
- (*local*) *steer*
- *collision check*
- *nearest vertex*
- *cost* or *distance*

The *cost* is required when finding an optimal path, though it can trivially be taken as 0 to make any found path optimal.

We start the growth of TS by obtaining a sample z from X using *sample*. The point of z is to become a temporary goal for the nodes in the tree, to help it extend into unexplored regions. We find the nearest neighbor s_v to candidate z in S, using *nearest neighbor*. The steer function in RRT returns a point y that minimizes distance from y to z while being only so far away from v.

The reason for using y and not z is that for a new point to accepted, there must be a transition from S to it. In RRT without dynamics, the assumption is that the free space is connected, so there must be such a transition. However, if z is far from v, the connection is likely to pass through an obstacle. Therefore, we first find y, and attempt to connect it to the tree after it passes a collision check. Since y is not too far from v, it is more likely that the path from y to v is collision-free.

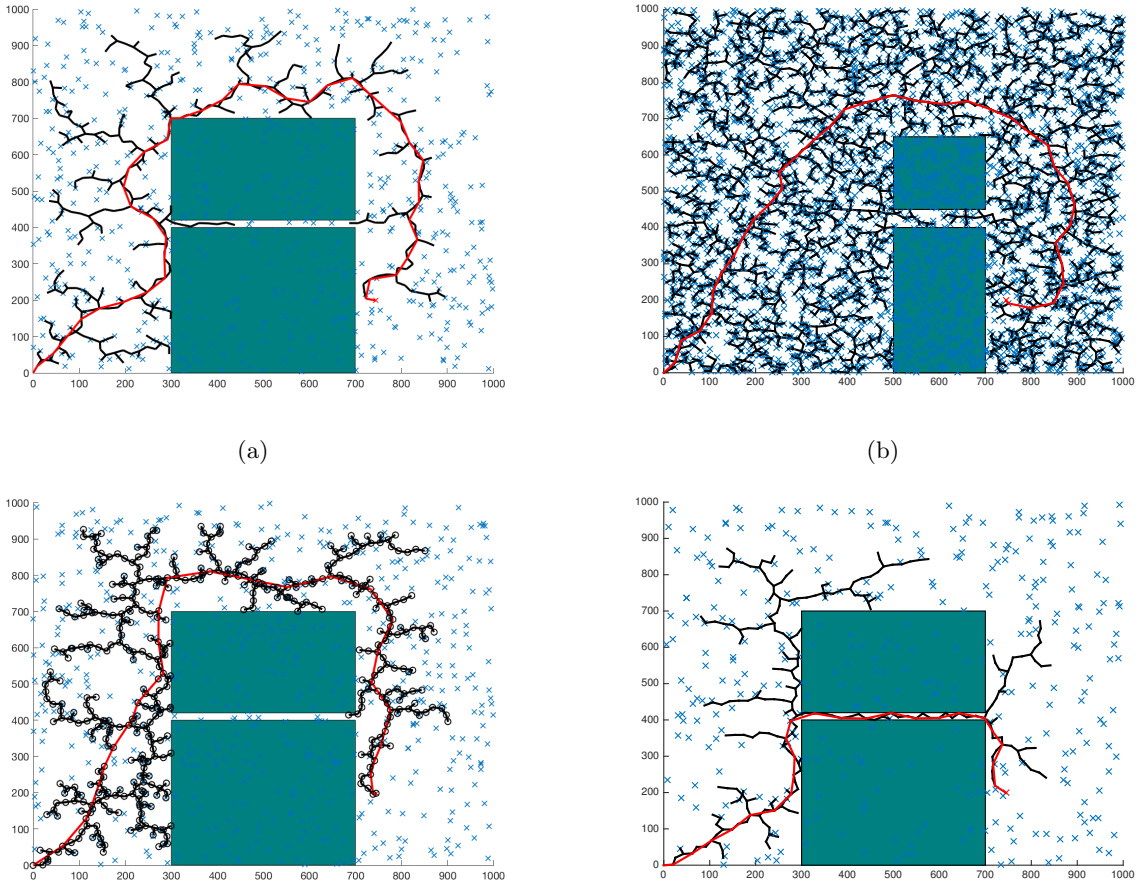
In RRT or RRT*with dynamics, the *steer* function finds an (optimal) trajectory between y and v, where again y is closest possible to z, but only so far from v. The trajectory returned by the *steer* function confirms that we can connect y to v. Again, there may be an obstacle in the way, and *collision check* determines if the entire trajectory is safe from collision. Once the trajectory is shown to be collision-free, we can add s_y with a transition from its nearest neighbor. Again, to prevent wasted collision checks, we aim to add y and not z, since the latter could be far from v.

In RRT*we then rewire the connections between the outputs of *nearest vertex* to s_y using cost-information. This rewiring makes sure that the transitions in TS encodes optimal paths.

5.4.1 RRT* Simulation

This section focuses on the example provided, and the concern that RRT* is not likely to find the shorter path through the gap. The folder RRT_Star_2D contains code that implements the RRT* algorithm for motion planning in two dimensions. There are no dynamic constraints. This code was downloaded from MATLAB's forums, and modified. There might be some errors in the re-wiring step, but it runs well enough to see how the RRT tree grows as samples are added.

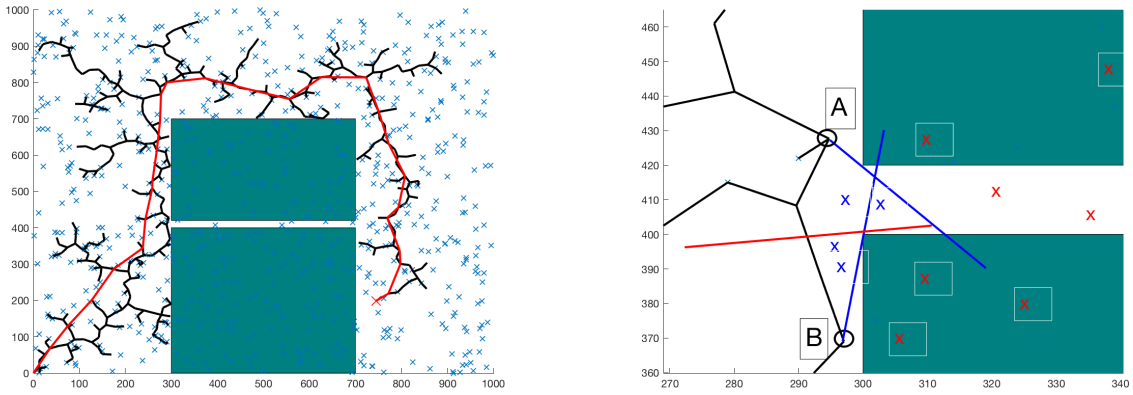
Figure 5.1 shows the tree without rewiring (in black), with the optimal path in red. The initial state is the origin in the lower left corner, the goal state is the red check mark. Figure 5.2 shows a branch of the tree that successfully passed through the narrower gap.



(a) (b)

(c) (d)

Figure 5.1: A 2D path planning scenario with two obstacles, with a red check mark indicating the goal at $(750, 200)$. The origin $(0, 0)$ is the root node. The tree obtained using RRT* is in black. (a) The shortest path in the tree found by RRT* does not pass through the narrow gap. (b) RRT* may ignore shorter and wider gaps. (c) The gap is not even explored, unlike in a) (d) The optimal path goes through the gap.



(a) (b)

Figure 5.2: Another run of RRT* on the 2D path planning example where the gap is never explored. (a) The entire tree and optimal path. (b) A blow-up of nodes close to the left entrance. Node A and B interact with the obstacles to create a very small region within which samples can successfully grow the tree through the gap. Examples of successful samples are the blue cross marks. The samples marked in red (even the sample in free space) would be rejected, since any sufficiently long path from points A or B passes through an obstacle.

Observations. The outputs of the `nearest-neighbor`, the `steer`, and the `collision_check` functions interact with the existing nodes to promote or inhibit growth of the tree through the gap. For a path to be found through the gap, the following two situations seem to be necessary

- Nodes near the entrance should be 'in front' of the gap.
- The tree in the longer path must not be well developed.

Figure 5.2 shows two two nodes at points A and B that inhibit growth of the tree through the gap since they create a very small region that a sample must lie in to successfully create a new node of the tree lying within the gap. For any sample to create a branch through the gap, it will have to select A or B as the nearest node. Unfortunately, the obstacle intersects paths between most of such samples and the nodes at A and B, causing a rejection of those samples. This inhibition allows the other branch to grow longer while no progress is made on the branch through the gap most of the time (unless the small region is sampled).

Figure 5.3 depicts the location of a node near the left end of the gap that would enable growth of the tree through the gap. The growth is possible because a vey large set of samples can lead to successfully placing new nodes within the gap. However, if the alternate branch has grown significantly, then again the path through the gap is not discovered. The new samples that could have promoted growth from C instead select D as their closest node, and grow the branch emanating from D.

Importance Sampling This mechanism hopefully convinces the reader that the issue is not that samples will rarely fall within the gap. The issue is that the components of the RRT* algorithm interact in a way that makes the usefulness of a sample non-deterministic.

One simple way to overcome the second necessary condition above is to use new samples to grow all 'branches'. The technical problem is characterizing how many 'branches' there are, and which set of nodes belong to which branch. There are probably algorithms for versions of this problem already, if not the exact problem. Solving this problem is a way of incorporating some notion of topology into the RRT* algorithm.

The first necessary condition is perhaps a place for important sampling, where one identifies these bottleneck node placement situations and *avoids* placing nodes there, or promotes placement of nodes in 'extendable' positions.

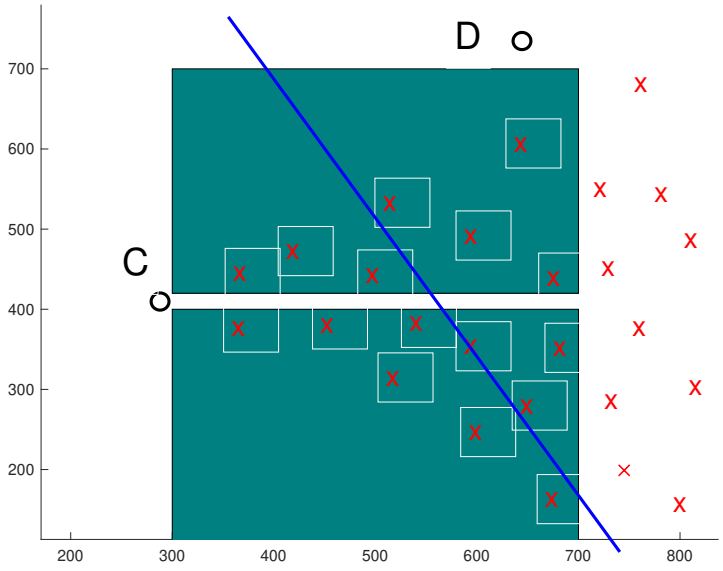


Figure 5.3: Point C, in the absence of point D, will be extended through the gap by samples in a large region, potentially represented by the red check marks. Note that if a one of the blue check marks closer to point A in Figure 5.2b were sampled, they would become points similar to point C here. If the alternate branch grows, and a node exists at point D, then the samples above the blue line would be closer to node D, inhibiting growth of the branch through the gap.

5.5 Trajectories From Paths

The algorithms we've mentioned so far provide a sequence of points that belong to a path joining q_0 and q_f. We can connect any two points by a straight line, thereby defining a path between the two points.

What we want is a trajectory, which means that we associate each point on the path with a time in a continuous manner. We may also want the path to be sufficiently smooth, so that the derivatives are bounded. Since each point corresponds to a unique time, the first and second derivatives of a trajectory correspond to velocity and acceleration respectively.

The full problem is known as trajectory optimization.

Typically, we are converting line segments defined by end points into trajectories. The start and end points are fixed, and we assume that the task define the velocities and accelerations only at the end points. We will search for trajectories that satisfy these requirements on the end points from the set of polynomial trajectories.

5.5.1 Polynomials

We consider each joint coordinate separately, since they are independent scalars. Suppose we know that

$$q(t_0) = q_0, \qquad q(t_f) = q_f \tag{5.8}$$
$$\dot{q}(t_0) = v_0, \qquad \dot{q}(t_f) = v_f \tag{5.9}$$

We have four constraints, and so we use a cubic polynomial to generate $q(t)$ satisfying these constraints:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3. \tag{5.10}$$

We rewrite this into the linear equation

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 \tag{5.11}$$
$$v_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2 \tag{5.12}$$
$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \tag{5.13}$$
$$v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 \tag{5.14}$$

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} \tag{5.15}$$

The determinant of the matrix is $(t_f - t_0)^4$, so that a solution exists on all non-trivial time intervals.

If we want to specify accelerations, we need quintic polynomials, and obtain similar equations. Why specify accelerations? So that there are no discontinuities when combining multiple segments, which would require infinite jerk.

5.5.2 Parabolic Blends

Divide the time interval $[t_0, t_f]$ into three segments where the velocity on the middle segment is specified, and the first and last segment represent smooth transitions between zero velocity and the specified middle segment velocity.

Minimum Time Trajectories Final time is not fixed, just need to when to switch from positive maximum acceleration to negative maximum acceleration.

Appendix A

Vector Spaces

A.1 Vector Spaces

Definition 17 (Group)**.** A group G is a set together with a binary operation $\cdot$ that satisfies the following properties for all $a, b, c \in G$:

(i) Closure: $a \cdot b \in G$;

(ii) Associativity: $a \cdot (b \cdot c) = (a\cdot)b \cdot c$;

(iii) Existence of identity element $e \in G$ such that $a \cdot e = e \cdot a = a$;

(iv) Existence of inverse element $d \in G$ such that $d \cdot a = a \cdot d = e$.

Example 4. Real numbers form a group under addition.

Example 5. Real numbers without 0 form a group under multiplication.

Definition 18 (Field)**.** A field $\mathbb{F}$ is a set together with two operations – addition $+\colon \mathbb{F} \times \mathbb{F} \mapsto \mathbb{F}$ and multiplication $\cdot\colon \mathbb{F} \times \mathbb{F} \mapsto \mathbb{F}$ – that satisfy the eight axioms listed below.

(i) Addition and multiplication are associative

(ii) Addition and multiplication are commutative

(iii) Existence of additive and multiplicative identity elements

(iv) Existence of inverse element for addition for each $v \in V$

(v) Existence of inverse element for multiplication for each $v \in V$ except for the additive identity

(vi) Distributivity of multiplication with respect to addition

Example 6. Real numbers are a field under usual addition and multiplication.

Definition 19 (Vector space)**.** A vector space over a field $\mathbb{F}$ is a set V together with two operations – vector addition $+\colon V \times V \mapsto V$ and scalar multiplication $\cdot\colon \mathbb{F} \times V \mapsto V$ – that satisfy the eight axioms listed below, for all $u, v, w \in V$ and $a, b \in \mathbb{F}$.

(i) Addition is associative: $u + (v + w) = (u + v) + w$;

(ii) Addition is commutative: $u + v = v + u$;

(iii) Existence of identity element $0 \in V$ such that $v + 0 = v$;

(iv) Existence of inverse element $x \in V$ such that $v + x = 0$;

(v) Compatibility of scalar multiplication with respect to field multiplication: $a \cdot (bv) = (a \cdot b)b$;

(vi) Existence of identity element $e \in \mathbb{F}$ under scalar multiplication such that $ev = v$;

(vii) Distributivity of scalar multiplication with respect to vector addition: $a \cdot (u + v) = a \cdot u + b \cdot v$;

(viii) Distributivity of scalar multiplication with respect to field addition: $(a + b) \cdot u = a \cdot u + b \cdot u$.

Example 7. The set of n-tuples of real numbers, denoted $\mathbb{R}^n$, over the field of real numbers form a vector space when addition and scalar multiplication of these n-tuples are taken to be element-wise addition and scalar multiplication. The 0 vector is the vector with all elements 0, and the inverse of $v \in \mathbb{R}^n$ is $-v = (-1)\cdot v$.

Definition 20 (Vector Space Basis)**.** A basis B of a vector space V is a set of vectors in V such that all other vectors can be written as a finite linear combination of the elements of B.

Remark 3 (Basis for $\mathbb{R}^n$)**.** A basis for vector space $\mathbb{R}^n$ contains exactly n linearly independent vectors.

Remark 4 (Coordinates for $\mathbb{R}^n$)**.** A basis for $\mathbb{R}^n$ equips each point $x \in \mathbb{R}^n$ with a coordinate given by the n coefficients of the basis vectors in the linear combination that yields x.

Definition 21 (Inner Product Space)**.** An inner product on a vector space V defined over a field $\mathbb{F}$ is a function $\langle \cdot, \cdot \rangle \colon V \times V \mapsto \mathbb{F}$ with the following properties

(i) $\langle x, y \rangle = \overline{\langle y, x \rangle}$ for all $x, y \in V$;

(ii) $\langle ax + by, z \rangle = a\langle x, z \rangle + b\langle y, z \rangle$, for all $x, y, z \in V$ and $a, b \in \mathbb{F}$;

(iii) $\langle x, x \rangle \geq 0$, for all $x \in V$, and $\langle x, x \rangle = 0 \iff x = 0$.

An inner product space is a vector space equipped with a suitable inner product.

An inner product defines the notion of angle between two vectors, specifically defining when two vectors are orthogonal (perpendicular) to each other.

Example 8. Vector space $\mathbb{R}^n$ equipped with the usual dot product forms an inner product space. Two vectors in $\mathbb{R}^n$ are orthogonal when the angle between them is $90°$.

Definition 22 (Norm)**.** A norm on a vector space V defined over field $\mathbb{F}$ (which is a subfield of the complex numbers $\mathbb{C}$) is a function $p \colon V \mapsto \mathbb{R}$ with the following properties:

For all $a \in \mathbb{F}$ and $x, y \in V$,

(i) $p(x + y) \leq p(x) + p(y)$;

(ii) $p(ax) = |a|p(x)$;

(iii) If $p(x) = 0$ then $x = 0$.

A **norm** defines a notion of **size** of vectors.

Example 9. An inner product space V with field $\mathbb{R}$ may be equipped with a norm p as follows:

$$p(u) = \sqrt{\langle u, u \rangle}.$$

Remark 5. For real vector spaces defined over $\mathbb{R}$, the symbol $\|\cdot\|$ is often used to denote the norm, instead of $p(\cdot)$.

Definition 23 (Metric)**.** A metric on a space X is a function $d \colon X \times X \mapsto \mathbb{R}$ with the following properties

(i) $d(x, y) \geq 0$ for all $x, y \in X$, and $d(x, y) = 0 \iff x = y$;

(ii) $d(x, y) = d(y, x)$, for all $x, y \in X$;

(iii) $d(x, y) \leq d(x, z) + d(z, y)$, for all $x, y, z \in X$.

A **metric** defines a notion of **distance** on a space.

Example 10. An inner product space V may be equipped with a norm $\|\cdot\|$, which then defines a metric $d \colon V \times V \to \mathbb{R}$ as

$$d(u, v) = \|u - v\|.$$

A.2 A Concept Chart

In Figure A.1, try to use directed arrows to indicate how different concepts lead to one another.
Hint: the 'Point in Space' may be viewed as the root node.

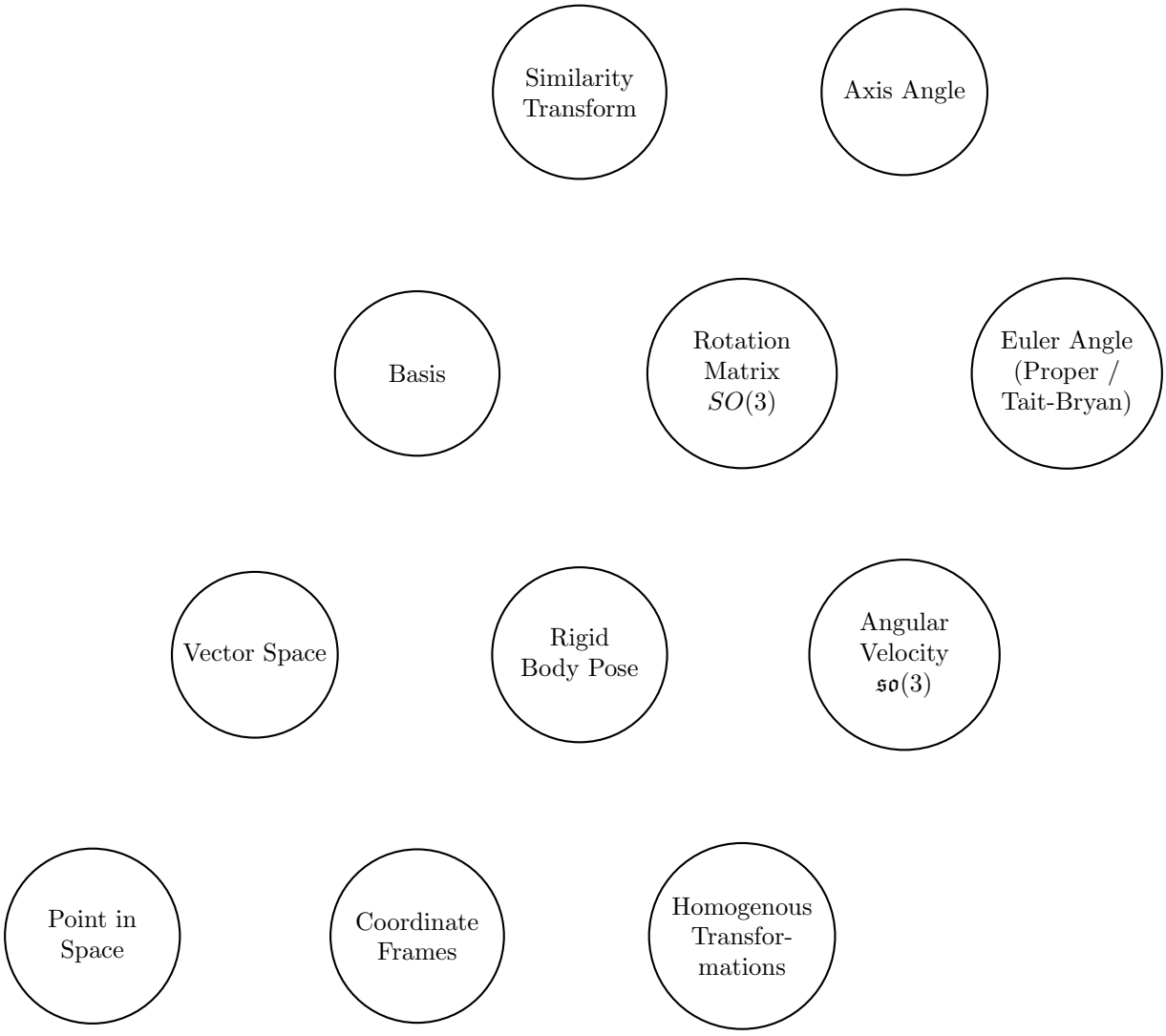


Figure A.1: Topics as nodes in a graph.

Appendix B

Analysis

B.1 Topology

A set X is a collection of distinct objects. When we define a concept of closeness between elements of a set, we equip the set with a topology. The importance of an abstract concept like topology in practice is that it allows us to predict the effect of imprecision and approximation; of being close but not quite exact.

B.1.1 Neighborhoods

More concretely, we define neighborhoods that are subsets of X with specific properties.

Definition 24 (Topological Space via Neighborhoods)**.** Let $\mathbf{N}(x)$ be the neighborhood function that provides the neighborhoods of $x \in X$. X with $\mathbf{N}(x)$ defines a topological space if it satisfies the following axioms:

i) If N is a neighborhood of x ($N \in \mathbf{N}(x)$), then $x \in N$

ii) If $N \subset X$ contains an element of $\mathbf{N}(x)$, then $N \in \mathbf{N}(x)$

iii) The intersection of two neighborhoods of x is also a neighborhood of x

iv) Any neighborhood N of x includes a neighborhood M of x such that N is a neighborhood of each point of M.

Example 11. The real number line $\mathbb{R}$, with N being a neighborhood of x if it contains an open interval containing x, forms a topological space. An open interval of the real line is a connected line segment that does not contain the end points.

The set of neighborhoods of x, which is $\mathbf{N}(x)$, is mind-bogglingly large. For example, we could create a countably infinite set of disjoint intervals of the real line where only one of these intervals contains x, and this infinitely large set would still be called a neighborhood of x.

To see that this set $X = \mathbb{R}$ with $\mathbf{N}(x)$ as given satisfies the requirements of being a topological space, we mainly need to apply the definition and keep in mind the properties of open intervals.

i) By definition, if $N \in \mathbf{N}(x)$, then $x \in N$

ii) If a set M contains a subset N that belongs to $\mathbf{N}(x)$, then there's an open interval U such that $x \in U \subset N$. Since $N \subset M$, in turn we may say that $x \in U \subset M$, which implies that $M \in \mathbf{N}(x)$

iii) If N_1 and N_2 are two neighborhoods, they contain open intervals U_1 and U_2 such that $x \in U_1$, $x \in U_2$. Clearly $U_3 = U_1 \cap U_2 \neq \emptyset$, since it contains x. Moreover, U_3 is an open interval since it is a non-empty intersection of two open intervals. Since $x \in U_3 \subset N_1 \cap N_2$, and U_3 is an open interval, we conclude that $N_1 \cap N_2$ is a neighborhood of x, so $N_1 \cap N_2 \in \mathbf{N}(x)$.

iv) Any neighborhood N of x contains at least one open interval U that contains x. For each y in U, we may treat N as a neighborhood of y, since $y \in U \subset N$, and U is an open interval. Thus, any neighborhood N of x contains a set U such that N is a neighborhood of each point in U.

Remark 6. Several mathematical proofs are as tedious but straightforward as this. Apply the recent definitions, using a few arguments involving mathematical properties/facts/definitions that are considered to be widely known (properties of open intervals of the real line, in this case).

B.1.2 Open Sets

An alternate characterization of topological spaces can be given in terms of a collection τ of subsets of X that satisfy specific properties:

Definition 25 (Topological Space via Open Sets)**.** A topological space if an ordered space (X, τ), where X is a set and τ is a collection of subsets of X satisfying

1. The empty set and X belong to τ
2. Any arbitrary union of members of τ still belongs to τ
3. The intersection of **finite** number of members of τ still belongs to τ.

The elements of τ that satisfy the conditions above are called the **open sets** and the collection τ is called a **topology** on X.

To make things confusing, one can use neighborhoods to define open sets:

Definition 26 (Open Sets via Neighborhoods)**.** Given a set of neighborhoods, a subset $U \subset X$ is *open* if U is a neighborhood for all points in U.

Example 12. Returning to Example 11, the closed interval $I = [0, 1]$ is a neighborhood of any point in I **except** for 0 and 1, since you can't fit an open interval containing 0 (or 1) into I. By contrast, the open interval $I' = (0, 1)$ is a neighborhood for all points in I'. Therefore, it is an open set. In general, unions of open intervals are precisely the open sets in the real line.

What is the corresponding version in $\mathbb{R}^n$ of an open interval in $\mathbb{R}$? Euclidean distance allows us to come up with one answer. An open set is a ball $B_r(x)$, where $x \in \mathbb{R}^n$, $r > 0$ is a radius, and

$$B_r(x) = \{y \in \mathbb{R}^n : \|y - x\| < r\}.$$

It is important to use $< r$ rather than $\leq r$ to ensure that $B_r(x)$ is open, much like $[0, 1]$ is not open in $\mathbb{R}$ but $(0, 1)$ is.

Example 13 (Topology Of $\mathbb{R}^n$)**.** The Euclidean vector space $\mathbb{R}^n$ equipped with a metric forms a topology where the open sets are balls centered at any point and with any radius.

Appendix C

Dynamical Systems & Control

C.1 Dynamical Systems

A general dynamical system has a state $x(t) \in X$ at time t, where X is the state space of the system. For mechanical system, the state often comprises both the configuration q and its derivative $\dot{q}$.

Many dynamical systems have an n-dimensional state space, and the evolution of the state with time is given by an ordinary differential equation:

$$\dot{x}(t) = f(x, t). \tag{C.1}$$

For many systems, f only depends on the current state, not on time. These systems are known as autonomous systems.

A controlled dynamical system typically also allows an additional signal u known as the control signal:

$$\dot{x}(t) = f(x, u, t). \tag{C.2}$$

Suppose f is time-invariant, and we choose a state-based feedback control $u = k(x)$. The dynamics becomes

$$\dot{x}(t) \ = f(x, k(x)) = f_{cl}(x), \tag{C.3}$$

which is again an autonomous system.

C.1.1 Solutions Of ODEs

A solution $x^*\colon [t_0, t_f] \mapsto X$ is a function that maps each time t in the interval $[t_0, t_f]$ to a unique state $x \in X$, denoted $x^*(t)$. The state at time t_0 is called the initial condition.

If one is given a map $\bar{x}\colon [t_1, t_2] \mapsto X$, then $\bar{(}t)$ is a solution to the ODE if for all $\bar{t} \in [t_1, t_2]$

$$\frac{d}{dt}\bar{x}(t)|_{\bar{t}} = f(\bar{x}(\bar{t}), \bar{t}) \tag{C.4}$$

C.1.2 Stability

The study of dynamical systems is often concerned with the existence of equilibria and the properties of these equilibria. For now, we focus on autonomous systems.

Definition 27 (Equilibrium)**.** An *equilibrium* point $x_e \in X$ is a point such that $f(x_e, t) = 0$ for all t.

The main property of equilibria is their stability.

Definition 28 (Stable)**.** An equilibrium point x_e is *stable* if for every $\epsilon > 0$, there exists $\delta > 0$ such that every solution $x(t)$ with initial condition $x(t_0) \in B_\delta(x_e)$ is such that $x(t) \in B_\epsilon(x_e)$.

In other words, solutions that start close stay close, no matter how small you define staying close to be.

Definition 29 (Asymptotically Stable)**.** An equilibrium point x_e is *asymptotically stable* if it is stable and for every solution $x(t)$ with initial condition $x(t_0) \in B_\delta(x_e)$ for some $\delta > 0$

$$\lim_{t \to \infty} x(t) = x_e \tag{C.5}$$

In other words, solutions not only stay close, they return back to x_e in a long enough time frame.

Definition 30 (Exponentially Stable)**.** An equilibrium point x_e is *asymptotically stable* if it is stable and for every solution $x(t)$ with initial condition $x(t_0) \in B_\delta(x_e)$ for some $\delta > 0$

$$\lim_{t \to \infty} x(t) = x_e \tag{C.6}$$

In other words, solutions not only stay close, they return back to x_e in a long enough time frame.

C.2 Linear Dynamical Systems

Consider the linear dynamical system

$$\dot{x}(t) = Ax(t) + Bu. \tag{C.7}$$

If $u \equiv 0$, then the dynamical system is stable if $Re(\lambda) \leq 0$ where λ is any eigenvalue of A. If $u \equiv 0$, then the dynamical system is **asymptotically** stable if $Re(\lambda) < 0$.

Suppose that there is an eigenvalue $\lambda^{us} \in \mathbb{C}$ of A such that $Re(\lambda^{us}) > 0$. The system is unstable when run in open loop ($u \equiv 0$). Suppose we choose $u = -Kx$. Then

$$\dot{x}(t) = (A - BK)x(t). \tag{C.8}$$

Naturally, we want to choose K so that all eigenvalues of $A - BK$ have non-negative real part.

C.2.1 Transfer Functions

The state-based representation focuses on the system state. Instead, we can define an output $y = Cx + Du$, and understand the system not through its (internal) state x, but merely through the relationship between its input u and output y. That is, given some input signal $u(t)$, what will the output signal $y(t)$ be?

As seen in introductory courses (ME 340), working in the frequency domain is a much easier way to answer this question, leading to a transfer function representation:

$$Y(s) = G(s)U(s), \tag{C.9}$$

where

$$G(s) = C(sI - A)^{-1}B + D. \tag{C.10}$$

For a single-input single-output system, the transfer function $G(s)$ is the quotient of two real polynomial functions of the complex variable $s = \sigma + j\omega$, i.e., $G(s) = n(s)/d(s)$. For multi-input multi-output systems, the transfer function $G(s)$ is a matrix of such SISO transfer functions.

The roots $s_i = \sigma_i + j\omega_i$ of the equation $d(s) = 0$ are known as the poles of the transfer function. Stability requires that all poles are in the closed left half plane, i.e., $\sigma_i \leq 0$ for all poles s_i.

C.2.2 Controllability and Observability

Given the linear time-invariant system

$$\dot{x} = Ax + Bu \tag{C.11}$$

$$y = Cx + Du \tag{C.12}$$

, we can reduce out ability to acquire information about the system state and the ability to influence the system's evolution to two concepts: observability and controllability.

C.2.3 Controllability

If we know all elements of x, we can choose a linear feedback

$$u(t) = -Kx(t) + r(t) \tag{C.13}$$

where r is the reference for $x(t)$. How useful is our ability to choose K? Can we always find a K that will make $x(t)$ behave in some desired way?

A LTI system (A, B, C, D) is **controllable** if and only if the matrix

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$$

is full rank.

If our system is controllable, we can choose K to assign the poles of $A - BK$ however we like, as long as complex poles have their conjugates as poles.

The flip side of this unfettered power is that we need to know every element of x, which means we need some sensors to measure them. Mathematically, we need C to be the identity, or at least some some non-singular square matrix.

C.2.4 Observability

What if we only measure a subset of x? In other words, what happens if C is not a matrix of rank n? Are we still able to figure out x and use it our idea of x in the feedback control rule $u = -Kx + r$?

The notion of observability describes whether we can do this. A LTI system (A, B, C, D) is **observable** if and only if the matrix

$$\mathcal{O} = \begin{bmatrix} C \\ CA & CA^2 \\ \vdots & CA^{n-1} \end{bmatrix}$$

is full rank.