ME/AER 676 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 largely based on the textbook written by Spong, Vidyasagar, and Hutchinson [5].

For the Spring of 2021, I began to incorporate more material from the textbook by Kevin Lynch and Frank Park [2]. This incorporation has been challenging, however, I am excited to continue developing a pedagogical resource that helps readers translate between the two approaches.

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 *numerically* describe physical space. For example, two separate LIDAR sensors on a robot may describe locations of the same object in space using different 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.

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. A good amount of bugs in implemented robotics algorithms boil down to erroneously assuming that two vectors are in the same reference frame.

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 C.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.

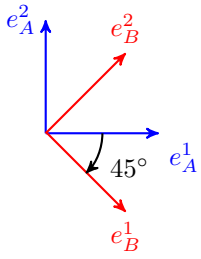
Definition 6 (Special Euclidean Group)**.** A rigid motion is a pair (d, R) where $d \in \mathbb{R}^3$ and $R \in SO(3)$. The group of all rigid motions is known as the **Special Euclidean Group** and is denoted by $SE(3)$. We see that $SE(3) = \mathbb{R}^3 \times SO(3)$.

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

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

Example 2 (Coordinate Transformation)**.** Find p^B if $p^A = (1, 1, 0)$.



Solution:

If these vectors have coordinates v_1^A, v_2^A, and v_3^A, then $R = \begin{bmatrix} v_1^A & v_2^A & v_3^A \end{bmatrix}$, or

$$
\begin{aligned}
e_B^1 &= \frac{1}{\sqrt{2}} e_A^1 - \frac{1}{\sqrt{2}} e_A^2 \\
e_B^2 &= \frac{1}{\sqrt{2}} e_A^1 + \frac{1}{\sqrt{2}} e_A^2 \\
e_B^3 &= 1 \cdot e_A^3 \\
\implies R = T_B^A &= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Therefore,

$$
p^B = \left(T_B^A\right)^{-1} p^A = T_A^B p^A = \begin{bmatrix} 0 \\ \sqrt{2} \\ 0 \end{bmatrix}
$$

□ □

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 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_B^A. Then the same linear map in frame B is given by matrix T^B

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

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.

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

Non-commutation

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

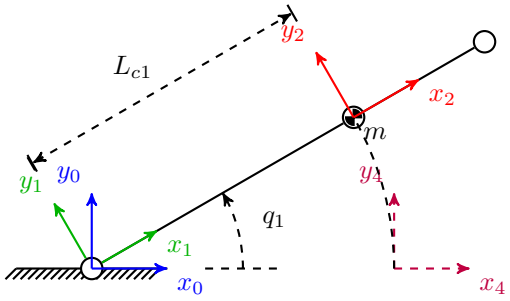


Figure 1.1: We may view frame {2} as the result of a) a rotation in frame {0} followed by translation in frame {1}, or b) same translation but in frame {0} producing {4}, followed by same rotation but in frame {0}, which moves {4} to {2}.

Example 3 (Simple Pendulum)**.** Consider the simple pendulum with mass m in Figure 1.1. Our goal will be to describe the transformation from frame {2} to frame {0}, denoted T_2^0. We may view this transformation as a sequence of two simpler transformations: a rotation and a translation. Let's refer to these transformations as H_T and H_R given by

$$H_T = \begin{bmatrix} I & d \\ 0 & 1 \end{bmatrix}, \quad H_R = \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix},$$

where

$$d = \begin{bmatrix} L_{c1} \\ 0 \\ 0 \end{bmatrix}, \quad R = R_{z,q_1}.$$

The key idea is that the frames we define them in, and the order in which we combine them, have to be consistent.

By looking at the figure, we see that

$$T_1^0 = H_R.$$

In other words, to get to frame {1} from frame {0}, we need to rotate frame {0} by q_1 radians about the z-axis. We can also see that

$$T_2^1 = H_T.$$

In words, to get to frame {2} from frame {1}, we need to translate along the x-axis by L_{c1} units (meters, usually). Therefore, we may see that to get from {0} to {2}, we can define a rigid body motion H_R relative to {0}, and then a rigid body motion relative to {1} of H_R (intrinsic motion, or motion relative to current frame). Using our rule of post-multiplication for a sequence of intrinsic motions, or motions relative to the current frame, we get that we must **post-multiply** H_R by H_T:

$$T_2^0 = \underbrace{H_R}_{\text{first}} \quad \underbrace{H_T}_{\text{(second, intrinsic/current)}} = \begin{bmatrix} R & Rd \\ 0 & 1 \end{bmatrix}.$$

We get the same result by viewing the sequence from the fixed-frame, but now **the translation happens first**, so that we are **pre-multiplying** H_T by H_R:

$$T_2^0 = \underbrace{H_R}_{\text{(second, fixed)}} \underbrace{H_T}_{\text{first}} = \begin{bmatrix} R & Rd \\ 0 & 1 \end{bmatrix}.$$

If we pre-multiplied the current-frame sequence, meaning H_T pre-multiplies H_R instead of the correct post-multiplication, we would instead get a frame whose origin is on the x-axis of $\{0\}$, at d, with the right orientation:

$$\text{(incorrect pre-multiplication by current-frame motion)} \qquad H_T H_R = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \neq T_2^0 \tag{1.8}$$

Again, this premultiplication is a reversal of the right sequence of motions, viewed from two frames.

□

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.

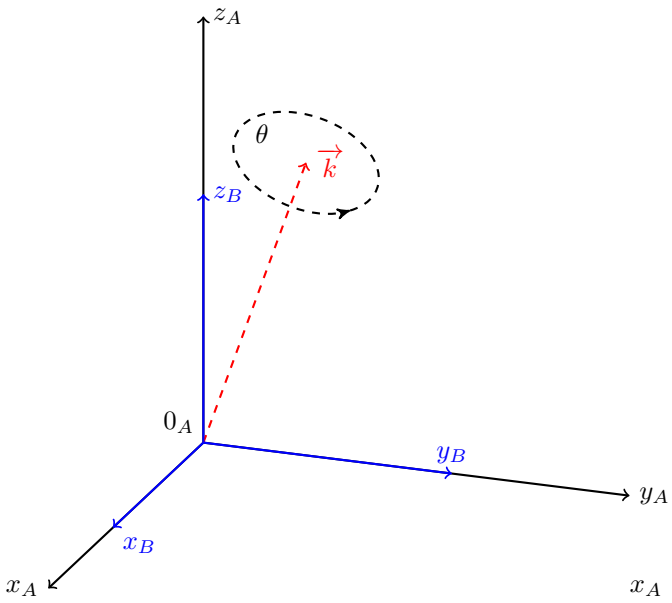
The main drawback of Euler-angles: non-uniqueness of values of three angles at singular points.

Axis/Angle Representation

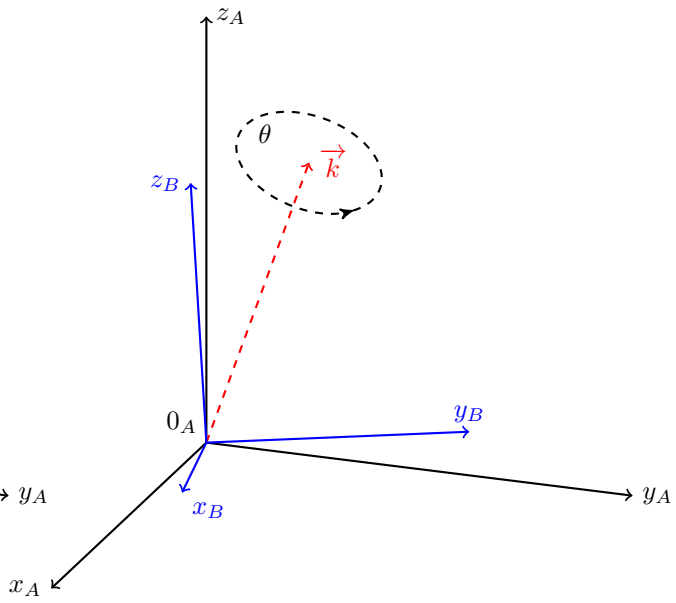
This representation is related to quaternions. The idea is that any orientation in a frame can be reached by rotating a coordinate frame by some angle $\theta \in [0, 2\pi)$ around some vector $\overrightarrow{k} \in \mathbb{R}^3$ in that frame. How do we represent that orientation?

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.

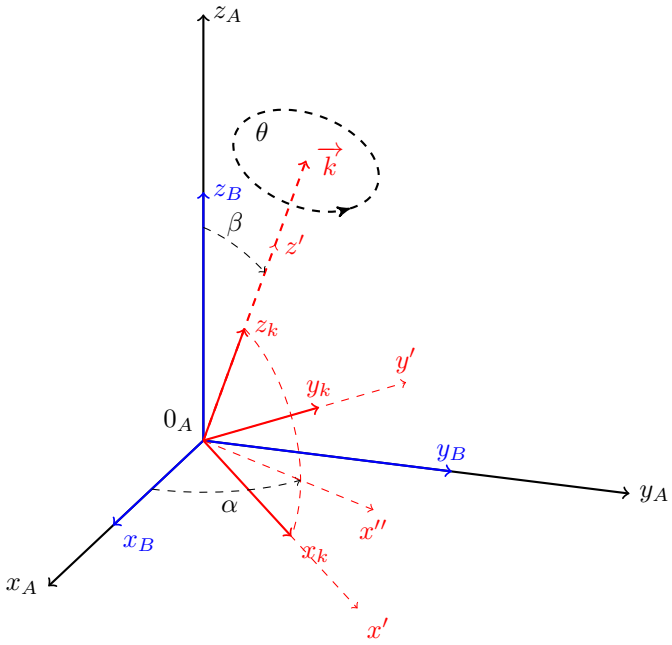
$$
\begin{aligned}
R_C^A &= R_{z,\alpha}R_{y,\beta}. \\
R_{k,\theta} &= R_C^A R_{z,\theta} \left(R_C^A\right)^{-1} && \left(\text{using } T^A = R_B^A \; T^B \; \left(R_B^A\right)^{-1} \right) \\
R_{k,\theta} &= R_{z,\alpha}R_{y,\beta}R_{z,\theta}R_{y,-\beta}R_{z,-\alpha}.
\end{aligned}
$$



(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 1.2: 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 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 in Section 2.3.1.

2.2 Velocity of Frames

We assign coordinates – aka rigid body pose – (d, R) to frame, relative to reference.
$d \in \mathbb{R}^3$, $R \in \text{SO}(3)$. If the rigid body pose tells us where a frame is located, its position, what is the rate-of-change of the position? For a position vector in $\mathbb{R}^n$, we know that the rate of change of position is another vector in $\mathbb{R}^n$, called the **velocity**. However, the coordinate (d, R) is not a vector!

2.2.1 Linear Velocity

If the position of the origin of one frame in another is given by some a time-varying quantity $x(t)$, where $x(t) \in \mathbb{R}^3$ for all t, then the linear velocity of the latter frame in the former is simply $\dot{x}(t)$, the usual notion of a velocity.

2.2.2 Angular Velocity

Unlike the rate-of-change of the position of the origin of a frame, the rate-of-change of the orientation of a frame is a more complicated object. If the orientation is time-varying, say $R(t)$, it turns out that

$$\dot{R}(t) = S(t)R(t),$$

where $S(t)$ satisfies $S(t) + S(t)^T = 0$ for each t. The matrix S is a skew-symmetric matrix, and has the form

$$S = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix},$$

for three numbers ω_1, ω_2, ω_3 Physically, the vector $\omega = \begin{bmatrix}\omega_1 & \omega_2 & \omega_3\end{bmatrix}^T$ defines the instantaneous angular velocity of the frame whose coordiantes are $(d(t), R(t))$ in the reference or base frame.

2.2.3 Task-space Velocity

There's a one-to-one relationship between a vector $\mathbb{R}^3$ and the set of 3×3 skew-symmetric matrices. Therefore, we can represent the rate of change of orientation using an angular velocity. So, when a task is $x(t) = (d(t), R(t)) \in \mathbb{R}^3 \times \mathrm{SO}(3)$, its velocity i s

$$\xi \in \mathbb{R}^6 = \underbrace{\mathbb{R}^3}_{\text{linear velocity}} \times \underbrace{\mathbb{R}^3}_{\text{angular velocity}}$$

2.2.4 Derivation

A frequent computation in robot motion is to determine the velocity of the end-effector frame ξ given the rates of change $\dot{q}$ of the joint angles q. This relationship is linear, and takes the form

$$\xi = J(q)\dot{q},$$

where $J(q)$ is a configuration-dependent matrix called the *Jacobian*.

To compute ξ, we therefore need to know $J(q)$. We now show that $J(q)$ in most cases may be derived using frame-transformations similar to the forward kinematics problem.

The core computation involves deriving the linear and angular velocity of a frame $\{2\}$ that is fixed in frame $\{1\}$, if we know the linear and angular velocity of the frame $\{1\}$ relative to a fixed frame $\{0\}$. Frame $\{0\}$ is not arbitrary; it is chosen to be coincident with $\{1\}$ at the instant of time we are considering. Specifically, for an observer in frame $\{0\}$, the linear and angular velocity of frame $\{1\}$ are v_0^1 and ω_0^1 respectively. Figure 2.1 depicts this situation.

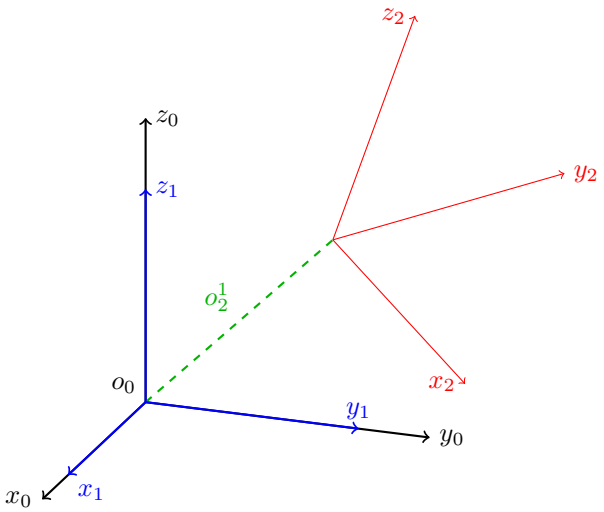


Figure 2.1: Frame $\{1\}$ moves with respect to fixed frame $\{0\}$. Frame $\{2\}$ is fixed in frame $\{1\}$.

Note two things about this setup. First, o_2^1 and R_2^1, which define the location of frame $\{2\}$ in frame $\{1\}$, are constants. Second, $o_1^0 = 0 \in \mathbb{R}^3$, and $R_1^0 = I \in \mathrm{SO}(3)$, however these my not be constants, due to the velocity of frame $\{1\}$ as seen in frame $\{0\}$.

Our goal is to compute v_2^0 and ω_2^0, the linear and angular velocity of frame $\{2\}$ as observed in fixed frame $\{0\}$. We may rewrite o_2^1 as the seemingly pointless expressions :

$$o_2^0 = o_1^0 + R_1^0 o_2^1, \tag{2.4}$$

$$R_2^0 = R_1^0 R_2^1. \tag{2.5}$$

Their value comes when we differentiate them with respect to time. Doing so, we obtain

$$v_2^0 = v_1^0 + \omega_1^0 \times R_1^0 o_2^1 \tag{2.6}$$

$$= v_1^0 + \omega_1^0 \times o_2^0, \tag{2.7}$$

and

$$[\omega_2^0]R_2^0 = [\omega_1^0]R_1^0R_2^1 \tag{2.8}$$
$$\implies [\omega_2^0] = [\omega_1^0]R_2^0 \tag{2.9}$$
$$\implies \omega_2^0 = \omega_1^0 \tag{2.10}$$

Therefore, we only need to know $o_2^1 = o_2^0$ to derive v_2^0 and ω_2^0 given v_1^0 and ω_1^0. The link to joint velocity is as follows. in our conventions, the single degree of freedom of motion q of a frame $\{1\}$ relative to $\{0\}$ always occurs about the z axis of frame $\{0\}$ (same as that of frame $\{1\}$, so that either

$$v_1^0 = \begin{bmatrix}0\\0\\1\end{bmatrix}\dot{q}, \quad \omega_1^0 = 0, \text{ or} \qquad\qquad v_1^0 = 0, \quad \omega_1^0 = \begin{bmatrix}0\\0\\1\end{bmatrix}\dot{q} \tag{2.11}$$

2.2.5 Another Derivation

Consider a fixed frame $\{s\}$. Point mass m_i is located at position $\mathbf{r}_i \in \mathbb{R}^3$ in frame $\{s\}$. These point masses all move rigidly relative to one another, but are **not** rigidly connected to the origin of $\{s\}$. Let d_{ij} be the constant distance between m_i and m_j. Then

$$(\mathbf{r}_i - \mathbf{r}_j)^T(\mathbf{r}_i - \mathbf{r}_j) = d_{ij}^2 \tag{2.12}$$
$$\implies (\mathbf{r}_i - \mathbf{r}_j)^T(\dot{\mathbf{r}}_i - \dot{\mathbf{r}}_j) = 0 \tag{2.13}$$
$$\tag{2.14}$$

Let's pick four points r_i for $i \in \{0,1,2,3\}$, where no 2D plane contains all four points. We define all positions and velocities relative to $\mathbf{r}_0$ and $\mathbf{v}_0 = \dot{\mathbf{r}}_0$, and simplify the notation by defining

$$\mathbf{r}_{i0} = \mathbf{r}_i - \mathbf{r}_0, \qquad\qquad \mathbf{v}_{i0} = \dot{\mathbf{r}}_i - \dot{\mathbf{r}}_0 = \mathbf{v}_i - \mathbf{v}_0 \tag{2.15}$$

The distance constraints between $\mathbf{r}_0$ and $\mathbf{r}_i$ for any index i then becomes

$$\mathbf{r}_{i0}^T\mathbf{v}_{i0} = 0 \tag{2.16}$$

We may rewrite the distance constraint between $\mathbf{r}_i$ and $\mathbf{r}_j$ as follows:

$$(\mathbf{r}_i - \mathbf{r}_j)^T(\mathbf{v}_i - \mathbf{v}_j) = ((\mathbf{r}_i - \mathbf{r}_0) - (\mathbf{r}_j - \mathbf{r}_0))^T((\mathbf{v}_i - \mathbf{v}_0) - (\mathbf{r}_v - \mathbf{v}_0)) \tag{2.17}$$
$$= (\mathbf{r}_{i0} - \mathbf{r}_{j0})^T(\mathbf{v}_{i0} - \mathbf{v}_{j0}) \tag{2.18}$$
$$= \underbrace{\mathbf{r}_{i0}^T\mathbf{v}_{i0} + \mathbf{r}_{j0}^T\mathbf{v}_{j0}}_{0} - \mathbf{r}_{j0}^T\mathbf{v}_{i0} - \mathbf{r}_{i0}^T\mathbf{v}_{j0} \tag{2.19}$$
$$= -\mathbf{r}_{j0}^T\mathbf{v}_{i0} - \mathbf{r}_{i0}^T\mathbf{v}_{j0} \tag{2.20}$$
$$\implies \mathbf{r}_{j0}^T\mathbf{v}_{i0} = -\mathbf{r}_{i0}^T\mathbf{v}_{j0} \tag{2.21}$$

Therefore, given the four position vectors $\mathbf{r}_i$ for $i \in \{0,1,2,3\}$, they define six relative velocity constraints, corresponding to choosing pairs from a set of four.

Since we chose $\mathbf{r}_i$ for $i \in \{0,1,2,3\}$ to be non-coplanar, we know that the three vectors $\mathbf{r}_{10}$, $\mathbf{r}_{20}$, $\mathbf{r}_{30}$ are mutually independent, and therefore form a basis for $\mathbb{R}^3$. If we define the matrix

$$M_{sb} = \begin{bmatrix}\mathbf{r}_{10} & \mathbf{r}_{20} & \mathbf{r}_{30}\end{bmatrix},$$

then

$$\dot{M}_{sb} = \begin{bmatrix}\mathbf{v}_{10} & \mathbf{v}_{20} & \mathbf{v}_{30}\end{bmatrix}.$$

We know that M_{sb} is invertible, and any point $\mathbf{r}_j$ in $\{s\}$ can be rewritten as $r_j = r_0 + M_{sb}\alpha$, for some $\alpha \in \mathbb{R}^3$. We know α to be the coordinates of r_j in a frame $\{b\}$ defined by points $\mathbf{r}_i$ for $i \in \{0,1,2,3\}$ on the rigid body.

We may compute the matrix product S given by

$$S = M_{sb}^T \dot{M}_{sb} \tag{2.22}$$
$$= \begin{bmatrix} \mathbf{r}_{10}^T \\ \mathbf{r}_{20}^T \\ \mathbf{r}_{30}^T \end{bmatrix}_{3\times 3} \begin{bmatrix} \mathbf{v}_{10} & \mathbf{v}_{20} & \mathbf{v}_{30} \end{bmatrix}_{3\times 3} \tag{2.23}$$
$$= \begin{bmatrix} \mathbf{r}_{10}^T\mathbf{v}_{10} & \mathbf{r}_{10}^T\mathbf{v}_{20} & \mathbf{r}_{10}^T\mathbf{v}_{30} \\ \mathbf{r}_{20}^T\mathbf{v}_{10} & \mathbf{r}_{20}^T\mathbf{v}_{20} & \mathbf{r}_{20}^T\mathbf{v}_{30} \\ \mathbf{r}_{30}^T\mathbf{v}_{10} & \mathbf{r}_{30}^T\mathbf{v}_{20} & \mathbf{r}_{30}^T\mathbf{v}_{30} \end{bmatrix} \tag{2.24}$$
$$= \begin{bmatrix} 0 & \mathbf{r}_{10}^T\mathbf{v}_{20} & \mathbf{r}_{10}^T\mathbf{v}_{30} \\ \mathbf{r}_{20}^T\mathbf{v}_{10} & 0 & \mathbf{r}_{20}^T\mathbf{v}_{30} \\ \mathbf{r}_{30}^T\mathbf{v}_{10} & \mathbf{r}_{30}^T\mathbf{v}_{20} & 0 \end{bmatrix} \tag{2.25}$$
$$= \begin{bmatrix} 0 & -\mathbf{r}_{20}^T\mathbf{v}_{10} & \mathbf{r}_{10}^T\mathbf{v}_{30} \\ \mathbf{r}_{20}^T\mathbf{v}_{10} & 0 & -\mathbf{r}_{30}^T\mathbf{v}_{20} \\ -\mathbf{r}_{10}^T\mathbf{v}_{30} & \mathbf{r}_{20}^T\mathbf{v}_{30} & 0 \end{bmatrix} \tag{2.26}$$
$$= \begin{bmatrix} \begin{bmatrix} \mathbf{r}_{30}^T\mathbf{v}_{20} \\ \mathbf{r}_{10}^T\mathbf{v}_{30} \\ \mathbf{r}_{20}^T\mathbf{v}_{10} \end{bmatrix} \end{bmatrix} \tag{2.27}$$

Therefore, $S = [\omega] \in \mathfrak{so}(3)$, where $\omega = \begin{bmatrix} \mathbf{r}_{30}^T\mathbf{v}_{20} & \mathbf{r}_{10}^T\mathbf{v}_{30} & \mathbf{r}_{20}^T\mathbf{v}_{10} \end{bmatrix}^T \in \mathbb{R}^3$.

We may now ask, given any point $\mathbf{r}_j \neq r_i$ for $i \in \{0,1,2,3\}$ on the rigid body we have been looking at, what will its velocity $\mathbf{v}_j$ be? Remember, all these points and vectors are in frame $\{s\}$. We derive the answer by using the relative position $\mathbf{r}_{j0}$ and velocity $\mathbf{v}_{j0}$, and applying the derived rule in (2.21):

$$\mathbf{r}_{i0}^T\mathbf{v}_{j0} = -\mathbf{v}_{i0}^T\mathbf{r}_{j0}, \quad \text{for } i \in \{1,2,3\} \tag{2.28}$$
$$\implies \begin{bmatrix} \mathbf{r}_{10}^T \\ \mathbf{r}_{20}^T \\ \mathbf{r}_{30}^T \end{bmatrix}_{3\times 3} \mathbf{v}_{j0} = -\begin{bmatrix} \mathbf{v}_{10}^T \\ \mathbf{v}_{20}^T \\ \mathbf{v}_{30}^T \end{bmatrix}_{3\times 3} \mathbf{r}_{j0} \tag{2.29}$$
$$\implies M_{sb}^T\mathbf{v}_{j0} = -\dot{M}_{sb}^T r_{j0} \tag{2.30}$$
$$\implies \mathbf{v}_{j0} = -\left(M_{sb}^{-1}\right)^T \dot{M}_{sb}^T\mathbf{r}_{j0} \tag{2.31}$$
$$\implies \mathbf{v}_j = \mathbf{v}_0 - \left(M_{sb}^{-1}\right)^T \dot{M}_{sb}^T\mathbf{r}_{j0} \tag{2.32}$$

This expression holds for any point $\mathbf{r}_j$, once we have chosen any four points $\mathbf{r}_i$ for $i \in \{0,1,2,3\}$. We can simplify this expression by picking the four points to form an orthonormal basis, in which case $M_{sb}^T = M_{sb}^{-1}$, and with the right ordering, $\det M_{sb} = 1$. In other words, $M_{sb} = R_{sb} \in \mathrm{SO}(3)$. Recall that

$$S = M_{sb}^T\dot{M}_{sb} = R_{sb}^T\dot{R}_{sb} \tag{2.33}$$
$$\implies S^T = -S = \dot{R}_{sb}^T R_{sb} \tag{2.34}$$
$$\implies \dot{R}_{sb}^T = -SR_{sb}^T \tag{2.35}$$

Then, we may rewrite the expression for $\mathbf{v}_j$ as

$$\mathbf{v}_j = \mathbf{v}_0 - R_{sb}\dot{R}_{sb}^T\mathbf{r}_{j0} \tag{2.36}$$
$$= \mathbf{v}_0 + R_{sb}SR_{sb}^T\mathbf{r}_{j0} \tag{2.37}$$
$$= \mathbf{v}_0 + R_{sb}[\omega]R_{sb}^T\mathbf{r}_{j0} \tag{2.38}$$
$$= \mathbf{v}_0 + [R_{sb}\omega]\mathbf{r}_{j0}, \tag{2.39}$$

where $\omega = \begin{bmatrix} \mathbf{r}_{30}^T\mathbf{v}_{20} & \mathbf{r}_{10}^T\mathbf{v}_{30} & \mathbf{r}_{20}^T\mathbf{v}_{10} \end{bmatrix}^T$. We now rename ω to be ω_b, the **body** angular velocity of the frame $\{b\}$ defined by R_{sb} and $\mathbf{r}_0$. Then, $\omega_s = R_{sb}\omega = R_{sb}\omega_b$ is the **spatial** angular velocity.

Putting it all together, given a set of points moving in frame $\{s\}$ with constant distance to each other, we can pick a point $\mathbf{r}_0$ whose velocity is $\mathbf{v}_0$ as a reference. Then, we pick three points whose relative position vectors with $\mathbf{r}_0$ form an orthonormal basis for $\mathbb{R}^3$. The position of these points relative to $\mathbf{r}_0$ (as viewed in $\{s\}$) points define a matrix R_{sb}. Their velocities relative to $\mathbf{v}_0$ define a body angular velocity ω_b. Given this construction, for any point with coordinates $\mathbf{p}$ in the frame $\{b\}$, we can derive its position in velocity in frame $\{s\}$ as

$$\mathbf{r}_j = \mathbf{r}_0 + R_{sb}\mathbf{p} \tag{2.40}$$

$$\mathbf{v}_j = \mathbf{v}_0 + [\omega_s]R_{sb}\mathbf{p}, \text{ where} \tag{2.41}$$

$$\omega_s = R_{sb}\omega_b = R_{sb}\begin{bmatrix}\mathbf{r}_{30}^T\mathbf{v}_{20}\\ \mathbf{r}_{10}^T\mathbf{v}_{30}\\ \mathbf{r}_{20}^T\mathbf{v}_{10}\end{bmatrix}. \tag{2.42}$$

In the equations above, all objects except $\mathbf{p}$ are defined in $\{s\}$.

The velocity $\mathbf{v}_b$ of the origin of $\{b\}$ relative to $\{s\}$ as seen in $\{b\}$ is $v_b = R_{sb}^T\mathbf{v}_0$. Note that ω_b may be derived from quantities only measured in frame $\{b\}$. In frame $\{b\}$, if the observed position and velocity of point j are $\bar{\mathbf{r}}_j$ and $\bar{\mathbf{v}}_j$ respectively, given by

$$\bar{\mathbf{r}}_j = R_{sb}^T(\mathbf{r}_j - \mathbf{r}_0) \tag{2.43}$$

$$\bar{\mathbf{v}}_j = R_{sb}^T(\mathbf{v}_j - \mathbf{v}_0), \tag{2.44}$$

then,

$$\omega_b = \begin{bmatrix}\bar{\mathbf{r}}_3^T\bar{\mathbf{v}}_2\\ \bar{\mathbf{r}}_1^T\bar{\mathbf{v}}_3\\ \bar{\mathbf{r}}_2^T\bar{\mathbf{v}}_1\end{bmatrix}. \tag{2.45}$$

Therefore, the velocity of the frame $\{b\}$ relative to frame $\{s\}$ can be observed in frame $\{b\}$, and will be given by the vectors v_b, ω_b, which form the **body** twist of $\{b\}$.

Note that we have also shown that $\dot{R}_{sb} = R_{Sb}[\omega_b] = [\omega_s]R_{Sb}$. If we knew this ahead of time, we would be able to derive

$$\mathbf{r}_j = \mathbf{r}_0 + R_{sb}p_b \tag{2.46}$$

$$\implies \mathbf{v}_j = \dot{\mathbf{r}}_0 + \dot{R}_{sb}p_b + R_{sb}\underbrace{\dot{p}_b}_{0} \tag{2.47}$$

$$= \mathbf{v}_o + [\omega_s]R_{sb}p_b \tag{2.48}$$

To summarize, we have shown that the position and velocity of all points on a rigid body may be expressed using the position and velocity of the origin, and the orientation and angular velocity of a frame rigidly fixed to this body.

2.3 Geometric and Analytic Jacobians

Forward and inverse kinematics are about creating the map $T_n^0(q)$ that provides the end effector pose $\left(o_n^0(q), R_n^0(q)\right)$. if we view the end-effector as the task coordinates x, then forward kinematics is about computing x given joint angles q, which we represent as the map f where

$$x = f(q).$$

In a similar way, when the link variables q change with time as $\dot{q}$, what is the 'velocity' ξ of the end effector? When the orientation of x is given by a vector of three numbers α, then $\xi = \dot{x}$. We may analytically derive a linear relationship $\xi = \dot{x} = \frac{\partial f}{\partial q}\dot{q} = J_a(q)\dot{q}$, and the Jacobian $J_a(q)$ is called the *analytic Jacobian*.

An alternative is to take velocity ξ as the linear velocity of the origin of the end-effector frame, together with the angular velocity of that frame, where coordinates of these velocities are in the base frame. Again, we may derive a linear relationship,

$$\xi = J(q)\dot{q},$$

however $J(q)$ is **not** the partial derivative of any function, but is derived geometrically. It is therefore called the *geometric* Jacobian.

2.3.1 Geometric Jacobian

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

$$v_n^0 = \dot{o}_n^0 \tag{2.49a}$$
$$S(\omega_n^0) = \dot{R}_n^0 \left(R_n^0\right)^T \tag{2.49b}$$

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,

$$\xi = \begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = J\dot{q} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q}, \tag{2.50}$$

where J is the geometric Jacobian. Note that some texts, like Modern Robotics, choose to put the angular velocity first, and linear velocity second. The variable $\dot{q}_i$ corresponds to the i^{th} degree of freedom (DoF). In the D-H convention, this motion along the i^{th} DoF occurs relative to the z-axis of the $i-1^{\text{th}}$ frame, which is called joint i. In the convention adopted by URDFs and the Modern Robotics text, motion along the i^{th} DoF occurs relative to the z-axis of the i^{th} frame.

This section follows the D-H convention. We derive the Jacobian column-by-column, where the i^{th} column corresponds to the i^{th} joint variable derivative $\dot{q}_i$. To derive this i^{th} column, we assume that only $\dot{q}_i \neq 0$, and all other joints are locked. As a consequence, frame $i-1$ is a fixed frame relative to the base frame, while link i is moving since $\dot{q}_i \neq 0$. Moreover, the end-effector is fixed with respect to link i, and so its motion is completely determined by $\dot{q}_i$. The velocity of this motion we consider is given by $\dot{q}_i$ relative to axis z_{i-1}, the z-axis of frame $i-1$. Therefore, applying (2.11), we may compute the linear and angular velocity of the end-effector frame due to $\dot{q}_i$, when all other joints are locked. This computation leads to finding the i^{th} column J_{v_i} of J_v to be

$$J_{v_i} = \begin{cases} z_{i-1} & , \quad \text{if joint } i \text{ is prismatic} \\ z_{i-1} \times (o_n - o_{i-1}) & , \quad \text{if joint } i \text{ is revolute} \end{cases} \tag{2.51}$$

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

$$J_{\omega_i} = \begin{cases} 0 & , \quad \text{if joint } i \text{ is prismatic} \\ z_{i-1} & , \quad \text{if joint } i \text{ is revolute} \end{cases} \tag{2.52}$$

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.

Example 4 (Planar Elbow Manipulator)**.** Consider the two-degree of freedom planar elbow manipulator shown in Figure 3.1. According to the D-H convention, Since there are two (moving) links, we have three frames:

1. Frame {0} attached to the base
2. Frame {1} attached to link 1
3. Frame {2} attached to link 2

Let $c_i = \cos q_i$, $s_i = \sin q_i$, $c_{ij} = \cos(q_i + q_j)$, $s_{ij} = \sin(q_i + q_j)$. We denote $e_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. Given any value of q_1 and q_2, we can determine that

$$T_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & L_1 c_1 \\ s_1 & c_1 & 0 & L_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_2^0 = T_1^0 T_2^1 = T_1^0 \begin{bmatrix} c_2 & -s_2 & 0 & L_{c2} \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{12} & -s_{12} & 0 & L_1 c_1 + L_{c2} c_{12} \\ s_{12} & c_{12} & 0 & L_1 s_1 + L_{c2} s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

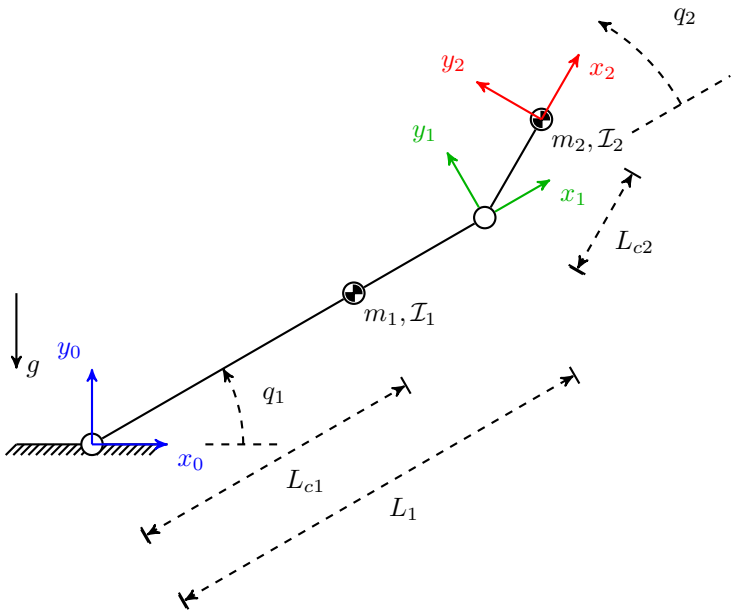


Figure 2.2: D-H Convention

Therefore, in the frame $\{0\}$, we have $z_0 = e_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, and $z_1 = R_1^0 e_3 = e_3$ for any q_1. Moreover, we have

$$o_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad o_1 = \begin{bmatrix} L_1 c_1 \\ L_1 s_1 \\ 0 \end{bmatrix}, \quad o_2 = \begin{bmatrix} L_1 c_1 + L_{c2} c_{12} \\ L_1 s_1 + L_{c2} s_{12} \\ 0 \end{bmatrix} \tag{2.53}$$

We wish to find v_2^0 and ω_2^0 as a function of $\dot{q}$. Since the joints are all revolute, We may derive the geometric Jacobian $J(q)$ as

$$J(q) = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} = \begin{bmatrix} z_0 \times (o_2 - o_0) & z_1 \times (o_2 - o_1) \\ z_0 & z_1 \end{bmatrix} \tag{2.54}$$

$$= \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} L_1 c_1 + L_{c2} c_{12} \\ L_1 s_1 + L_{c2} s_{12} \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \left(\begin{bmatrix} L_1 c_1 + L_{c2} c_{12} \\ L_1 s_1 + L_{c2} s_{12} \\ 0 \end{bmatrix} - \begin{bmatrix} L_1 c_1 \\ L_1 s_1 \\ 0 \end{bmatrix} \right) \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} \tag{2.55}$$

$$= \begin{bmatrix} \begin{bmatrix} -L_1 s_1 - L_{c2} s_{12} \\ L_1 c_1 + L_{c_2} c_{12} \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} L_{c_2} c_{12} \\ L_{c2} s_{12} \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} -L_1 s_1 - L_{c2} s_{12} & -L_{c2} s_{12} \\ L_1 c_1 + L_{c2} c_{12} & L_{c2} c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{2.56}$$

□

For comparison, note the URDF/Modern Robotics convention for assigning frames in Figure 2.4 below. Simply put, the frames are on the 'other end' of the link, which typically makes more sense. However, the Jacobian **will not change**. What will change is the label we assign to intermediate terms used to derive this same Jacobian. With this alternate numbering and location of frames, our expression for the Jacobian

would become

$$J(q) = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} = \begin{bmatrix} z_1 \times (o_3 - o_1) & z_2 \times (o_3 - o_2) \\ z_1 & z_2 \end{bmatrix}$$

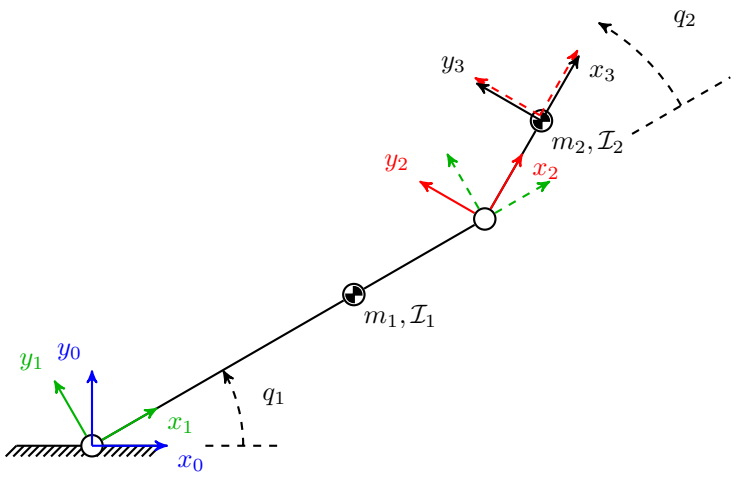


Figure 2.3: URDF frame convention, similar to Modern Robotics [2]. We need an additional frame {3} at the center of mass of the last link, to get the right Jacobian. The dashed axes of a color represent the locations of the frames attached to each link according to the D-H convention, for comparison.

2.3.2 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

$$X = \begin{bmatrix} d(q) \\ \alpha(q) \end{bmatrix}, \tag{2.57}$$

and

$$\dot{X} = \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} = J_a(q)\dot{q}, \tag{2.58}$$

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

$$R(\alpha) = R_{z,\psi} R_{y,\theta} R_{z,\phi}.$$

Each element r_{ij} of $R(\alpha)$ depends on α. Therefore, each element $\dot{r}_{ij}(t)$ of $\dot{R}$ will depend on α and $\dot{\alpha}$. Since

$$\dot{R}(\alpha, \dot{\alpha}) = S(\omega)R(\alpha),$$

we may rearrange terms to derive the relationship

$$\omega = B(\alpha)\dot{\alpha},$$

where the derivatives $\dot{\alpha}$ turn out to appear linearly in the expression (see RMC text). Then,

$$J(q)\dot{q} = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{d} \\ B(\alpha)\dot{\alpha} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & B(\alpha) \end{bmatrix} J_a(q)\dot{q}, \tag{2.59}$$

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

$$J_a(q) = \begin{bmatrix} I & 0 \\ 0 & B^{-1}(\alpha) \end{bmatrix} J(q). \tag{2.60}$$

2.3.3 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

$$\xi = J\dot{q} \tag{2.61}$$

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.

2.3.4 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,

$$J = \begin{bmatrix} J_{11} & 0 \\ J_{12} & J_{22} \end{bmatrix}, \tag{2.62}$$

and so

$$\det J = \det J_{11} \det J_{22}. \tag{2.63}$$

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

$$\det J_{11} = a_2 a_3 \sin\theta_3 \left(a_2 \cos\theta_2 + a_3 \cos(\theta_2 + \theta_3)\right). \tag{2.64}$$

2.3.5 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

$$\dot{q} = J(q)^{-1}\xi. \tag{2.65}$$

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 (2.65).

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 (2.65).

2.3.6 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

$$\|\dot{q}\|^2 = \xi^T(JJ^T)^{-1}\xi \tag{2.66}$$

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

$$\xi^T(JJ^T)^{-1}\xi = (U^T\xi)^T\Sigma_m^{-2}(U^T\xi), \tag{2.67}$$

in which

$$\Sigma_m^{-2} = \begin{bmatrix} \sigma_1^{-2} & & & \\ & \sigma_2^{-2} & & \\ & & \ddots & \\ & & & \sigma_m^{-2} \end{bmatrix} \tag{2.68}$$

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

$$\|\dot{q}\|^2 = w^T\Sigma_m^{-2}w = \sum_{i=1}^{m} \frac{w_i^2}{\sigma_i^2} \tag{2.69}$$

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

$$\mu = \Pi_{i=1}^m \sigma_i \tag{2.70}$$

Example 5 (Planar Elbow Manipulator)**.** Given the Jacobian in Example 4, we find that

$$\mu = |\det J| = L_1L_{c_2}|s_2|.$$

When q_2, the elbow angle, is small, we have the least ability to move in all directions. Another interpretation is that when the elbow angle is zero, the kinematics of the two links resemble that of a single link (simple pendulum). □

Manipulability Ellipsoid

The ellipsoid $\xi^T(JJ^T)^{-1}\xi = 1$ where x is the task-space velocity indicates the motion corresponding to different unit norm joint velocities. Effectively, the unit norm ball of joint velocities becomes this ellipsoid in task space. The ellipsoid axes with larger length indicate the directions in which motion is amplified.

2.4 Inverse Kinematics

The inverse kinematics (IK) problem is to find q given X. If the forward kinematics map f, where $X = f(q)$, is one-to-one, then we may be able to manually derive its inverse f^{-1} and solve $q = f^{-1}(X)$. Robotics texts will typically solve the problem for simple robots like the planar elbow manipulator.

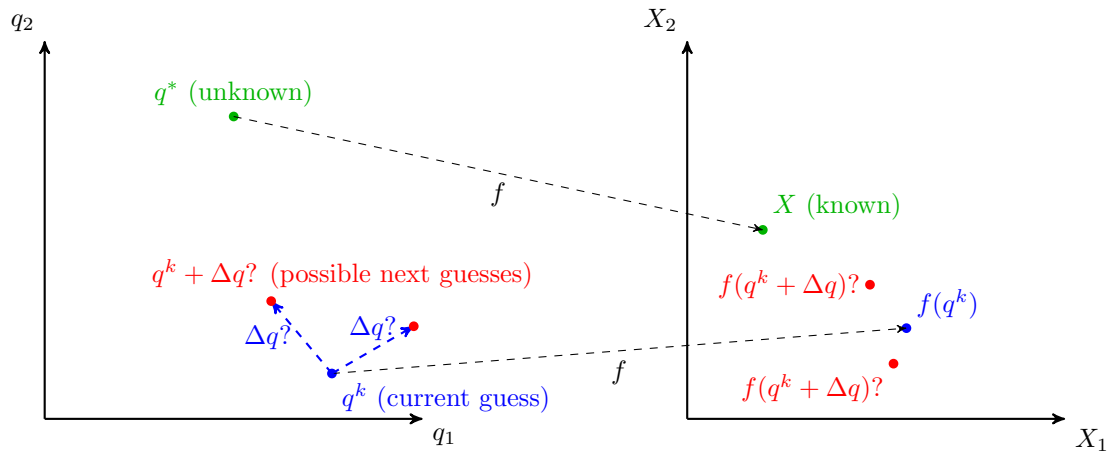


Figure 2.4: In iterative schemes to solve inverse kinematics, we try to choose a step Δq so that $f(q^k + \Delta q)$ is 'closer' to X than $f(q^k)$.

2.4.1 Optimization-Based Kinematics

Deriving the inverse of the forward kinematics function is usually hard outside the simple cases. Implementing the forward kinematics is easy, so one approach is to randomly select candidate solutions q_{cand}, compute $f(q_{cand})$, and see if it matches X. Instead of random selection, we can use ideas from optimization to *search* for the right candidate:

$$\text{IK: solve } \min_q \quad \underbrace{\|x - f(q)\|_2^2}_{L(q)} \tag{2.71}$$

In general this problem is nonlinear and non-convex, and different approaches may be implemented to solve it depending on the robot.

A general method to solve such an optimization problem is using gradient descent, an iterative scheme. Starting with a guess q^0 for the candidate, we take a small step Δq in some direction that will reduce $L(q)$, meaning we hope that $L(q^0 + \Delta q) < L(q^0)$. This direction is often take as the negative of the gradient of $L(q)$. We repeat this process until $L(q) \approx 0$, where the gradient after each step changes. So, we repeatedly solve

$$q^{k+1} = q^k + \Delta q \tag{2.72}$$
$$= q^k - \eta \, \nabla_q L(q)|_{q=q^k} \tag{2.73}$$
$$= q^k + 2\eta(X - f(q^k)) \, \nabla_q f(q)|_{q=q^k}, \tag{2.74}$$

where η is a learning rate or step-size parameter, and ∇_q is the gradient with respect to q. Assuming $\nabla_q f(q)$ exists and is easy to calculate, q^k will converge to a local solution q^* such that $X \approx f(q^*)$.

Instead of discrete steps, we can solve an ODE:

$$\dot{q}(t) = \eta(X - f(q))\nabla_q f(q). \tag{2.75}$$

We expect the ODE solution to approach a solution to the IK problem as $t \to \infty$.

2.4.2 Differential Inverse Kinematics

With recent advances in software for machine learning, it has become increasingly realistic to obtain gradients of complicated functions $f(q)$, using a technique called automatic differentiation (AD). Before AD, finding the gradient was nearly impossible, and even then it might be expensive to compute depending on $f(q)$.

For the case where $\nabla_q f(q)$ is not obtainable, how do we solve the optimization problem in (2.71)? The key insight is that we can obtain a step Δq in the joint space by applying inverse velocity kinematics to a step ΔX in the task space. A good step ΔX is easy to choose:

$$\Delta X = X - f(q^k), \tag{2.76}$$

which is the direction from our current guess $f(q^k)$ to our target X. Then,

$$\Delta q = J(q^k)^+ \Delta X = J(q^k)^+ \left(X - f(q^k)\right), \tag{2.77}$$

$$q^{k+1} = q^k + \Delta q \tag{2.78}$$

$$= q^k + \eta J(q^k)^+ \left(X - f(q^k)\right), \tag{2.79}$$

Instead of discrete steps, we can solve an ODE:

$$\dot{q}(t) = \eta J(q(t))^+ \left(X - f(q(t))\right). \tag{2.80}$$

Dropping the arguments to simplify the expression we get

$$\dot{q} = \eta J^+ \left(X - f(q)\right). \tag{2.81}$$

This ODE is easy to implement since $f(q)$ is easy to implement.

2.4.3 Trajectory Inverse Kinematics

Instead of a static target X, we may want to solve the IK problem for a trajectory $X(t)$. That is, we want to find a trajectory $q(t)$ such that $f(q(t)) = X(t)$ for all times t for which $X(t)$ is defined. In that case, we modify the inverse kinematics rule to solve

$$\dot{q}(t) = \eta J(q(t))^+ \left(\xi(t) + \left(X(t) - f(q(t))\right)\right). \tag{2.82}$$

where $\xi(t)$ is the task velocity at time t. Dropping the arguments to simplify the expression we get

$$\dot{q} = \eta J^+ \left(\xi + \left(X - f(q)\right)\right). \tag{2.83}$$

We may interpret this approach as integrating $\dot{q}(t) = J^+ \xi(t)$ to produce $q(t)$, but with an additional corrective term based on the error between $X(t)$ and $f(q(t))$.

2.5 Spatial and Body Jacobians

When the velocity of a frame is expressed using spatial or body twists, then we need a different set of Jacobians to map $\dot{q}$ to these twists. These are the spatial Jacobian J_s and the body Jacobian J_b respectively.

The definitions of twists and formulae are in the text [2], and we apply the computations to the planar elbow manipulator below.

Example 6 (Spatial and Body Jacobians)**.** We need the joint axes to be represented as screws $\mathcal{B}_i$ in the body frame of the end effector (for body Jacobian) or screws $\mathcal{S}_i$ in the base frame (for spatial Jacobian), when the joint angles are zero.

Once we have either these space vectors, a standard formula may be applied for different values of q. We start with M, the transformation T_{sb} when $q = 0$. By inspection,

$$M = \begin{bmatrix} 1 & 0 & 0 & (L_1 + L_2) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.84}$$

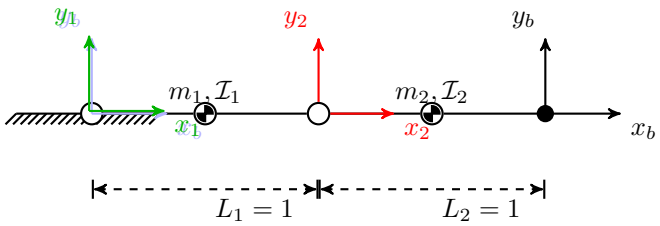


Figure 2.5: Length units are in meters. The spatial fixed frame $\{s\}$ coincides with $\{1\}$.

Because these are revolute joints with zero pitch, we can identify screw axis $\mathcal{S}_1$ and $\mathcal{S}_2$ as

$$\mathcal{S}_1 = \begin{bmatrix} z_1 \\ -z_1 \times o_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathcal{S}_2 = \begin{bmatrix} z_2 \\ -z_2 \times o_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ -L_1 \\ 0 \end{bmatrix} \tag{2.85}$$

Similarly, we may identify $\mathcal{B}_1$ and $\mathcal{B}_2$ as

$$\mathcal{B}_1 = \underbrace{\begin{bmatrix} z_1 \\ -z_1 \times (o_1) \end{bmatrix}}_{\text{coords in frame } \{b\}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ L_1 + L_2 \\ 0 \end{bmatrix}, \quad \mathcal{B}_2 = \underbrace{\begin{bmatrix} z_2 \\ -z_2 \times (o_2) \end{bmatrix}}_{\text{coords in frame } \{b\}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ L_2 \\ 0 \end{bmatrix} \tag{2.86}$$

We may check that these derivations satisfy the general relationship

$$\mathcal{B}_i = \mathrm{Ad}_{M^{-1}} \mathcal{S}_i.$$

Note that

$$M^{-1} = \begin{bmatrix} 1 & 0 & 0 & -(L_1 + L_2) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{2.87}$$

$$\implies \mathrm{Ad}_{M^{-1}} = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} = \begin{bmatrix} I & 0 \\ [p] & I \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & (L_1 + L_2) & 0 & 1 & 0 \\ 0 & -(L_1 + L_2) & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.88}$$

With the screw axes derived, we can construct the Jacobians J_s and J_b, where we need to combine the screw axis, M, and adjoint mappings. Specifically, the i^{th} column J_{si} of J_s is given by

$$J_{si} = \mathrm{Ad}_{e^{[\mathcal{S}_1]q_1} \ldots e^{[\mathcal{S}_{i-1}]q_{i-1}}}(\mathcal{S}_i), \text{ for } i = 2, \ldots, n \tag{2.89}$$

, and $J_{s1} = \mathcal{S}_1$. For our planar manipulator, we therefore just need to work through J_{s2}. To do so, note that

$$[\mathcal{S}_1] = \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} & \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.90}$$

$$\implies e^{[\mathcal{S}_1]q_1} = \begin{bmatrix} R_{z,q_1} & 0 \\ 0 & 1 \end{bmatrix} \tag{2.91}$$

Therefore,

$$J_{s2} = \mathrm{Ad}_{e^{[\mathcal{S}_1]q_1}}(\mathcal{S}_2) = \begin{bmatrix} R_{z,q_1} & 0 \\ 0 & R_{z,q_1} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ -L_1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.92}$$

$$\implies J_s = \begin{bmatrix} \mathcal{S}_1 & J_{s2} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 0 & -L_1 \sin q_1 \\ 0 & L_1 \cos q_1 \\ 0 & 0 \end{bmatrix} \tag{2.93}$$

$$\implies J_{s,v} = \begin{bmatrix} 0 & -L_1 \sin q_1 \\ 0 & L_1 \cos q_1 \\ 0 & 0 \end{bmatrix}, \quad J_{s,\omega} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{2.94}$$

Clearly,

$$J_{s_v}^T \begin{bmatrix} F_{x_0} \\ F_{y_0} \\ F_{z_0} \end{bmatrix} \neq \begin{bmatrix} -\sin\frac{\pi}{4} \\ \cos\frac{\pi}{4} \end{bmatrix}, \quad J_{s_v}^T \begin{bmatrix} F_{x_3} \\ F_{y_3} \\ F_{z_3} \end{bmatrix} \neq \begin{bmatrix} -\sin\frac{\pi}{4} \\ \cos\frac{\pi}{4} \end{bmatrix} \tag{2.95}$$

The i^{th} column J_{bi} of J_b is given by

$$J_{bi} = \mathrm{Ad}_{e^{-[\mathcal{B}_b]q_n} \ldots e^{-[\mathcal{B}_{i+1}]q_{i+1}}}(\mathcal{B}_i), \text{ for } i = 1, \ldots, n-1, \tag{2.96}$$

where $J_{bn} = \mathcal{B}_n$. Now,

$$e^{-[\mathcal{B}_2]q_2} = \begin{bmatrix} R_{z,-q_2} & \alpha \\ 0 & 1 \end{bmatrix}, \tag{2.97}$$

$$\text{where } \alpha = \left(I q_2 + (1 - \cos q_2) \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + (q_2 - \sin q_2) \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}^2 \right) \begin{bmatrix} 0 \\ -L_2 \\ 0 \end{bmatrix} \tag{2.98}$$

$$= \left(\begin{bmatrix} q_2 & 1 - \cos q_2 & 0 \\ \cos q_2 - 1 & q_2 & 0 \\ 0 & 0 & q_2 \end{bmatrix} + \begin{bmatrix} \sin q_2 - q_2 & 0 & 0 \\ 0 & \sin q_2 - q_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} 0 \\ -L_2 \\ 0 \end{bmatrix} \tag{2.99}$$

$$= \begin{bmatrix} L_2 \cos q_2 - L_2 \\ -L_2 \sin q_2 \\ 0 \end{bmatrix} \tag{2.100}$$

To calculate the adjoint, we need to compute

$$[\alpha]R_{z,-q_2} = \begin{bmatrix} 0 & 0 & -L_2 s_2 \\ 0 & 0 & L_2 - L_2 c_2 \\ L_2 s_2 & L_2 c_2 - L_2 & 0 \end{bmatrix} \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -L_2 s_2 \\ 0 & 0 & L_2 - L_2 c_2 \\ L_2 s_2 & -L_2 c_2 & 0 \end{bmatrix} \tag{2.101}$$

Therefore,

$$J_{b1} = \begin{bmatrix} R_{z,-q_2} & 0 \\ [\alpha]R_{z,-q_2} & R_{z,-q_2} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ L_1 + L_2 \\ 0 \end{bmatrix} \tag{2.102}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \\ L_1 s_2 \\ L_2 + L_1 c_2 \\ 0 \end{bmatrix} \tag{2.103}$$

Finally,

$$J_b = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ L_1 s_2 & 0 \\ L_2 + L_1 c_2 & L_2 \\ 0 & 0 \end{bmatrix} \implies J_{b,v} = \begin{bmatrix} L_1 s_2 & 0 \\ L_2 + L_1 c_2 & L_2 \\ 0 & 0 \end{bmatrix}, \quad J_{b,\omega} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{2.104}$$

□

2.6 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, expressed in the base frame. Let τ be the corresponding vector of joint torques. If the Geometric Jacobian is $J(q)$, then F and τ are related by

$$\tau = J^T(q)F \tag{2.105}$$

Note that some implementations construct $J(q)$ with the linear portion J_v as the first three rows, and others place J_ω in the first three rows. Note that if the Jacobian is a spatial Jacobian or body Jacobian, which converts $\dot{q}$ to spatial and body twists respectively (see the Modern Robotics [2] text), then using F in $J^T F$ would be incorrect.

2.6.1 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

$$\delta w = F^T \delta X - \tau^T \delta q. \tag{2.106}$$

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

$$\delta w = (F^T J - \tau^T)\delta q. \tag{2.107}$$

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, (2.105) holds.

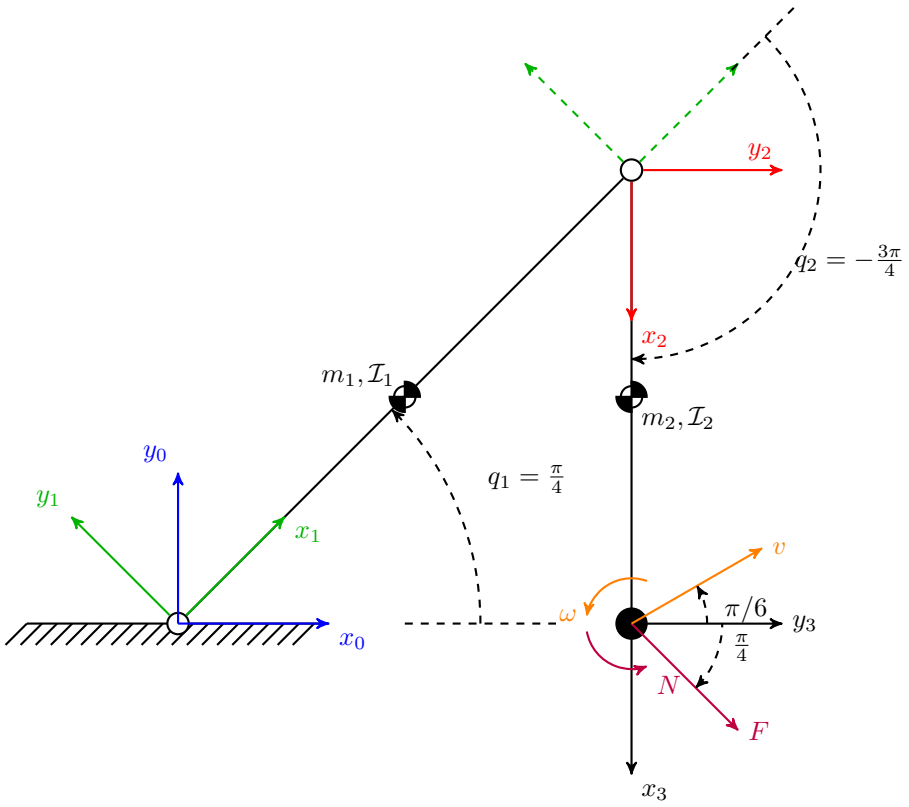


Figure 2.6: The first link has length $L_1 = \sqrt{2}$m, the second link has length $L_2 = 1$m.

Example 7 (Static Forces To Torques)**.** Consider the planar elbow manipulator in Figure 2.6. The first link has length $\sqrt{2}$m, the second link has length 1m The configuration is such that $q_1 = \pi/4$ rad, and $q_2 = -3\pi/4$. The origin of the end-effector frame $\{3\}$ (black) has velocity v, and experiences a force F. Assume that $\|v\| = 1$, and $\|F\| = 1$.

The body frame is the end-effector $\{3\}$. The linear and angular velocities v and ω may be combined to produce three quantities, based on the frame they are expressed in:

Name	Task velocity	Spatial Twist	Body Twist
angular velocity	ω_0	ω_0	ω_3
linear velocity ...	v_0	$v_0 - \omega_0 \times o_{30}$	v_3
... of point	o_3	o_0	o_3
Jacobian	Task	Spatial	Body
Symbol	$J(q)$	J_s	J_b

In Figure 2.6, the linear velocity v and angular velocity ω of frame $\{3\}$ as viewed in frames $\{3\}$ and $\{0\}$ are

$$\{3\} \quad : v_3 = \begin{bmatrix} -\sin\frac{\pi}{6} \\ \cos\frac{\pi}{6} \\ 0 \end{bmatrix}, \quad \omega_3 = \begin{bmatrix} 0 \\ 0 \\ \omega_z \end{bmatrix}, \qquad \{0\} : \quad v_0 = \begin{bmatrix} \cos\frac{\pi}{6} \\ \sin\frac{\pi}{6} \\ 0 \end{bmatrix}, \quad \omega_0 = \begin{bmatrix} 0 \\ 0 \\ \omega_z \end{bmatrix}. \tag{2.108}$$

From Example 4, we get

$$J(q) = \begin{bmatrix} J_v \\ j_\omega \end{bmatrix} = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} \\ L_1 c_1 + L_2 c_{12} & L_2 c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{2.109}$$

$$\implies v_0 = \begin{bmatrix} \dot{q}_2 \\ \dot{q}_1 \\ 0 \end{bmatrix}, \quad \omega_0 = \begin{bmatrix} 0 \\ 0 \\ \dot{q}_1 + \dot{q}_2 \end{bmatrix} \implies \dot{q}_2 = \cos\frac{\pi}{6}, \quad \dot{q}_1 = \sin\frac{\pi}{6}, \quad \omega_z = \cos\frac{\pi}{6} + \sin\frac{\pi}{6} \tag{2.110}$$

Consider F, which is a force applied at o_3, and no applied external torque ($N = 0$). In frame $\{0\}$, it has coordinates $F_0 = \begin{bmatrix} F_{x_0} & F_{y_0} & 0 & 0 & 0 & 0 \end{bmatrix}^T$. Given $\|F\| = 1$, we get

$$F_{x_0} = \cos\frac{\pi}{4}, \quad F_{y_0} = -\sin\frac{\pi}{4}.$$

We may inspect the figure to see that the torque at joint 1 due to F depends only on F_{y_0}, and is $F_{y_0} \cdot 1$, since the moment arm is 1m.
Similarly, the torque at joint 2 due to F depends only on F_{x_0}, and is $F_{x_0} \cdot 1$, since the moment arm is 1m.

We now see that because F is applied at o_3 but with components parallel to $\{0\}$, the correct transformation from F to resulting joint torques needs the Task Jacobian $J(q)$. We use the Jacobian in (2.109) to compute $\tau = J(q)^T F_0$ to get

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{x_0} \\ F_{y_0} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} F_{y_0} \cdot 1 \\ F_{x_0} \cdot 1 \end{bmatrix} = \begin{bmatrix} -\sin\frac{\pi}{4} \\ \cos\frac{\pi}{4} \end{bmatrix} \tag{2.111}$$

If we used F_3 in $\{3\}$, which would be determined by

$$F_{x_3} = \sin\frac{\pi}{4}, \quad F_{y_3} = \cos\frac{\pi}{4},$$

and all other components as 0, then using $J(q)$ would give us the incorrect result:

$$\tau = J(q)^T F_3 = \begin{bmatrix} \cos\frac{\pi}{4} \\ \sin\frac{\pi}{4} \end{bmatrix}.$$

From (2.104) When $q_2 = -3\pi/4$, $L_1 = \sqrt{2}$, $L_2 = 1$, we get

$$J_{b,v} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}. \tag{2.112}$$

Then,

$$\tau = J_b^T F_3 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \sin\frac{\pi}{4} \\ \cos\frac{\pi}{4} \\ 0 \end{bmatrix} + J_{b,\omega}^T \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -\sin\frac{\pi}{4} \\ \cos\frac{\pi}{4} \end{bmatrix}, \tag{2.113}$$

which is the correct answer.

□

If we wish to use the function `geometric_jacobian` provided by `RigidBodyDynamics` to map task forces into joint torques, we must transform the applied forces into an equivalent force and torque applied at o_0, with coordinates inframe $\{0\}$. The function `point_jacobian` at a point x defined in a frame $\{f\}$ maps $\dot{q}$ to the velocity of the origin of $\{f\}$ as seen in frame $\{f\}$, which is the body linear velocity. If we want to use the function `point_jacobian` at a point x, then the applied forces F must be converted into an equivalent force and torque applied at x, with coordinates in frame $\{f\}$. This is precisely a body wrench in $\{f\}$.

2.6.2 Force Ellipsoid

When the end-effector is moving in free space, the manipulability ellipsoid $\xi(JJ^T)^{-1}\xi = 1$ indicates the way a unit velocity in joint space will translate to motion directions in the end-effector space. This ellipsoid captures the mapping from joint space velocities to task space velocities.

When the end-effector is at a constant pose, maybe because of contact forces applied to it, then we can derive a but related ellipsoid. Since the map from the force to the torque is characterized by $J(q)^T$, the ellipsoid $F_{tip}(JJ^T)F_{tip} = 1$ now represents the set of forces generated in the task space given all possible unit-norm vector of joint torques (more accurately, generalized forces) applied at the joint axes.

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 Newton-Euler Formulations

The Newton-Euler formulation focuses on expressing expressing the dynamics of each link by treating it as a free body with forces that enforce constraints. We look at some examples that sequentially increase constraints and dimensions, starting from an constrained 2D point mass to a rigid body.

3.1.1 Two Dimensional Examples

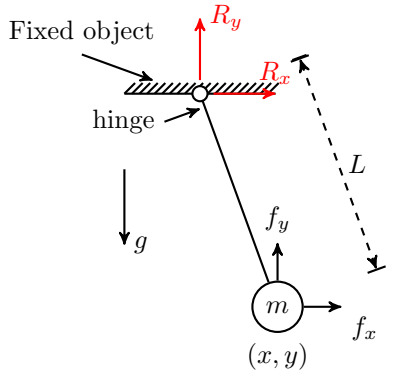
We introduce some examples to build the intuition for the general 2D and 3D rigid body equations. Note that the planar two dimension is always a subset of three dimensions when $z \equiv 0$ and all rotation axes are parallel to the z axis.

Example 8 (Simple Pendulum)**.** Let a mass m be located at position (x, y) in a Cartesian coordinate frame. This mass is constrained to stay at distance L from a hinge located at the origin of this coordinate frame, so that $x^2+y^2 = L^2$. Applying Newton's laws to this system, we obtain

$$m\ddot{x} = R_x + f_x \tag{3.1}$$
$$m\ddot{y} = R_y + f_y - mg \tag{3.2}$$

The Euler equation boils down to zero net torque about mass m at (x, y):

$$yR_x - xR_y = 0 \tag{3.3}$$



We have three equations in four unknowns: $\ddot{x}$, $\ddot{y}$, R_x, and R_y. The constraint $x^2 + y^2 = L^2$ may be differentiated twice to obtain

$$x\ddot{x} + y\ddot{y} = -\dot{x}^2 - \dot{y}^2 \tag{3.4}$$

With this fourth equation, we may solve the linear system consisting of four equations in four unknowns $\ddot{x}$, $\ddot{y}$, R_x, and R_y:

$$\begin{array}{cccc|l} m\ddot{x} & & -R_x & & = f_x \\ & m\ddot{y} & & -R_y & = f_y - mg \\ & & +yR_x & -xR_y & = 0 \\ x\ddot{x} & +y\ddot{y} & & & = -\dot{x}^2 - \dot{y}^2 \end{array} \quad \Longrightarrow \quad \begin{bmatrix} m & 0 & -1 & 0 \\ 0 & m & 0 & -1 \\ 0 & 0 & y & -x \\ x & y & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ R_x \\ R_y \end{bmatrix} = \begin{bmatrix} f_x \\ f_y - mg \\ 0 \\ -\dot{x}^2 - \dot{y}^2 \end{bmatrix} \tag{3.5}$$

□

Example 9 (Double pendulum)**.** The double pendulum consists of two masses m_1 and m_2. Mass m_1 is attached to a fixed frame through a hinge. Mass m_2 is attached to mass m_1 through another hinge. This mass is constrained to stay at distance L from a hinge located at the origin of this coordinate frame, so that $x^2 + y^2 = L^2$.

Applying Newton's laws to m_1, we obtain

$$m_1\ddot{x}_1 = R_x + F_x \tag{3.6}$$
$$m_1\ddot{y}_1 = R_y + F_y - m_1 g \tag{3.7}$$

The Euler equations for m_1 boil down to zero net torque about mass m at (x, y):

$$y_1 R_x - x_1 R_y + (x - x_1)F_y + (y_1 - y)F_x = 0 \tag{3.8}$$

Applying Newton's and Euler's laws to m_2, we get

$$m_2\ddot{x}_2 = -F_x \tag{3.9}$$
$$m_2\ddot{y}_2 = -F_y + f - m_2 g \tag{3.10}$$
$$0 = -(y_2 - y)F_x + (x_2 - x)F_y \tag{3.11}$$

We have six equations in eight unknowns: $\ddot{x}_1$, $\ddot{y}_1$, $\ddot{x}_2$, $\ddot{y}_2$, R_x, R_y, F_x and F_y. The two distance constraints $x_1^2 + y_1^2 = L_{c_1}^2$ and $(x_2 - x)^2 + (y_2 - y)^2 = L_{c_2}^2$ may be differentiated twice to obtain

$$x_1\ddot{x}_1 + y_1\ddot{y}_1 = -\dot{x}_1^2 - \dot{y}_1^2 \tag{3.12}$$
$$(x_2 - x)(\ddot{x}_2 - \ddot{x}) + (y_2 - y)(\ddot{y}_2 - \ddot{y}) = -(\dot{x}_2 - \dot{x})^2 - (\dot{y}_2 - \dot{y})^2 \tag{3.13}$$

Let $r = L_1/L_{c_1}$. Then $x = rx_1$ and $y = ry_1$. Similar to the simple pendulum, we may collect and rewrite all the equations above to obtain the linear system:

$$\begin{bmatrix} m_1 & 0 & 0 & 0 & -1 & 0 & -1 & 0 \\ 0 & m_1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & y_1 & -x_1 & -(r-1)y_1 & (r-1)x_1 \\ 0 & 0 & m_2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & m_2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -c_y & c_x \\ x_1 & y_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -rc_x & -rc_y & c_x & c_y & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{y}_1 \\ \ddot{x}_2 \\ \ddot{y}_2 \\ R_x \\ R_y \\ F_x \\ F_y \end{bmatrix} = \begin{bmatrix} 0 \\ -m_1 g \\ 0 \\ 0 \\ f - m_2 g \\ 0 \\ -\dot{x}^2 - \dot{y}^2 \\ -(\dot{x}_2 - r\dot{x}_2)^2 - (\dot{y}_2 - r\dot{y}_1)^2 \end{bmatrix} \tag{3.14}$$

where $c_x = (x_2 - rx_1)$, $c_y = (y_2 - ry_1)$. □

3.1.2 Solving Newton-Euler Equations

In the previous section, we saw that we can model point-masses with constrained motions using Newton's Laws leading to a set of linear equations in terms of the accelerations, constraint forces, and external forces.

We can rewrite the simple pendulum dynamics in (3.5) or (3.14) by splitting the matrix into blocks to obtain the block matrix equation:

$$\begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} a \\ R \end{bmatrix} = \begin{bmatrix} f \\ b \end{bmatrix}, \tag{3.15}$$

where a consists of the accelerations, R consists of the constraint forces, the variable f consists of external forces we control, and b are the remaining (known) terms that depend on x, y, $\dot{x}$, $\dot{y}$ etc. and forces that we do not control, like gravity. For example, in the case of the simple pendulum we get

$$a = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix}, \quad R = \begin{bmatrix} R_x \\ R_y \end{bmatrix}, \quad f = \begin{bmatrix} f_x \\ f_y \end{bmatrix}, \text{ and } b = \begin{bmatrix} 0 \\ -\dot{x}^2 - \dot{y}^2 - mg \end{bmatrix}. \tag{3.16}$$

There are two standard problems we solve using (3.15).

1. **Forward Dynamics** (FD): Solve for a given f
2. **Inverse Dynamics** (ID): Solve for f given a

In both cases, we know the joint configuration q, joint velocities $\dot{q}$, and terms in b, however we do not know the reaction forces R.

We will need to solve the Forward Dynamics problem whenever we want to simulate the motion of a robot given selected actuator forces and known external forces acting on the robot. We will need to solve the Inverse Dynamics problem when we want to achieve a certain motion and therefore need to know what actuator torques or forces will do so.

Forward Dynamics Solution. One (inefficient) way to solve FD is by inverting the matrix on the left hand side in (3.15):

$$\begin{bmatrix} a \\ R \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}^{-1} \begin{bmatrix} f \\ b \end{bmatrix}.$$

To avoid inverting the whole matrix, we derive closed form expressions

$$a = A_1^{-1} f - A_1^{-1} A_2 \Delta^{-1} \left(b - A_3 A_1^{-1} f \right), \tag{3.17}$$

$$R = \Delta^{-1} \left(b - A_3 A_1^{-1} f \right), \tag{3.18}$$

where

$$\Delta = A_4 - A_3 A_1^{-1} A_2.$$

Note that we may not care about R in some situations. However, computing the smaller inverses A_1^{-1} and Δ^{-1} is still inefficient, which matters when trying to compute on real embedded systems. These inverses take $\mathcal{O}(N^3)$ calculations to compute, where there are N links in a body. The Articulated Body Algorithm solves the FD problem using $\mathcal{O}(N)$ calculations.

Inverse Dynamics Solution. We can again derive closed-form expressions

$$f = \left(I + A_2 \Delta^{-1} A_3 A_1^{-1} \right)^{-1} \left(A_1 a + A_2 \Delta^{-1} b \right) \tag{3.19}$$

$$R = \left(A_4 - A_3 A_1^{-1} A_2 \right)^{-1} \left(b - A_3 A_1^{-1} f \right) \tag{3.20}$$

Again, we may not care about R in some situations. Using these closed-form equations would again result in an algorithm that takes $\mathcal{O}(N^3)$ calculations. The Recursive Newton Euler Algorithm (RNEA) solves the FD problem using $\mathcal{O}(N)$ calculations.

3.1.3 Newton-Euler Equations For A Planar Rigid Body

The two examples in Section 3.1.1 treat links in a mechanism as point masses. We can begin to understand continuum rigid body masses by constraining the double pendulum further, so that it acts like a rigid body consisting of two particles (point masses).

Note that we can constrain the distance of (x_2, y_2) to any point $(\tilde{x}, \tilde{y}) = r(x_1, y_1)$ on the line connecting the hinge to the first mass m_1. For the double pendulum, we chose $r = L_1/L_{c_1} \neq 1$, but we now allow r to be a variable. This 'generalized' view of the constraint leads to:

$$(x_2 - rx_1)(\ddot{x}_2 - r\ddot{x}_1) + (y_2 - ry_1)(\ddot{y}_2 - r\ddot{y}_1) = -(\dot{x}_2 - r\dot{x}_1)^2 - (\dot{y}_2 - r\dot{y}_1)^2 \tag{3.21a}$$

$$\begin{aligned} \implies & x_2\ddot{x}_2 - r(x_1\ddot{x}_2 + x_2\ddot{x}_1) + r^2x_1\ddot{x}_1 + y_2\ddot{y}_2 - r(y_1\ddot{y}_2 + y_2\ddot{y}_1) + r^2y_1\ddot{y}_1 \\ & = -\dot{x}_2^2 - r^2\dot{x}_1^2 - 2r\dot{x}_1\dot{x}_2 - \dot{y}_2^2 - r^2\dot{y}_1^2 - 2r\dot{y}_1\dot{y}_2 \end{aligned} \tag{3.21b}$$

$$\begin{aligned} \implies & x_2\ddot{x}_2 - r(x_1\ddot{x}_2 + x_2\ddot{x}_1) + y_2\ddot{y}_2 - r(y_1\ddot{y}_2 + y_2\ddot{y}_1) + r^2x_1\ddot{x}_1 + r^2y_1\ddot{y}_1 \\ & = -r^2\dot{x}_1^2 - r^2\dot{y}_1^2 - \dot{x}_2^2 - 2r\dot{x}_1\dot{x}_2 - \dot{y}_2^2 - 2r\dot{y}_1\dot{y}_2 \end{aligned} \tag{3.21c}$$

$$\implies x_2\ddot{x}_2 + y_2\ddot{y}_2 - r(x_1\ddot{x}_2 + x_2\ddot{x}_1 - y_1\ddot{y}_2 + y_2\ddot{y}_1) = -\dot{x}_2^2 - \dot{y}_2^2 - r(2\dot{x}_1\dot{x}_2 + 2\dot{y}_1\dot{y}_2) \tag{3.21d}$$

We derived (3.21d) without any assumptions on r. If we constrain (x_2, y_2) to be at constant distance from two points on pendulum with mass m_1, corresponding to two values of r, say r_1 and r_2, then

$$x_2\ddot{x}_2 + y_2\ddot{y}_2 - r_1(x_1\ddot{x}_2 + x_2\ddot{x}_1 - y_1\ddot{y}_2 + y_2\ddot{y}_1) = -\dot{x}_2^2 - \dot{y}_2^2 - r_1(2\dot{x}_1\dot{x}_2 + 2\dot{y}_1\dot{y}_2) \tag{3.22}$$

$$x_2\ddot{x}_2 + y_2\ddot{y}_2 - r_2(x_1\ddot{x}_2 + x_2\ddot{x}_1 - y_1\ddot{y}_2 + y_2\ddot{y}_1) = -\dot{x}_2^2 - \dot{y}_2^2 - r_2(2\dot{x}_1\dot{x}_2 + 2\dot{y}_1\dot{y}_2) \tag{3.23}$$

$$\implies x_1\ddot{x}_2 + x_2\ddot{x}_1 - y_1\ddot{y}_2 + y_2\ddot{y}_1 = 2\dot{x}_1\dot{x}_2 + 2\dot{y}_1\dot{y}_2 \tag{3.24}$$

This last expression, and the preceding calculations, allow us to conclude that if constraint (3.21a) holds for two values of r, then it holds for any value of r. In words, **when mass m_2 is constrained to be at a constant distance from two points on pendulum m_1, then it is constrained to be at a constant distance from all points on pendulum m_1**, including the hinge ($r = 0$) of the pendulum.

Before we proceed, we will reduce the number of variables we use to represent the motion of masses by exploiting the constraints.

Example 10 (Simple Pendulum Using Angle)**.** For the simple pendulum, we may manually eliminate $\ddot{y}$, R_x, and R_y to obtain

$$m(x^2 + y^2)\ddot{x} = y^2 f_x - xy(f_y - mg) - mx(\dot{x}^2 + \dot{y}^2). \tag{3.25}$$

This step is equivalent to solving the forward dynamics using the formula in (3.17). If we define an angle q made by the pendulum with the positive x axis, with counter-clockwise sense positive, then $x = L\cos q$, and $y = L\sin q$. These expressions allow us to rewrite (3.25) as

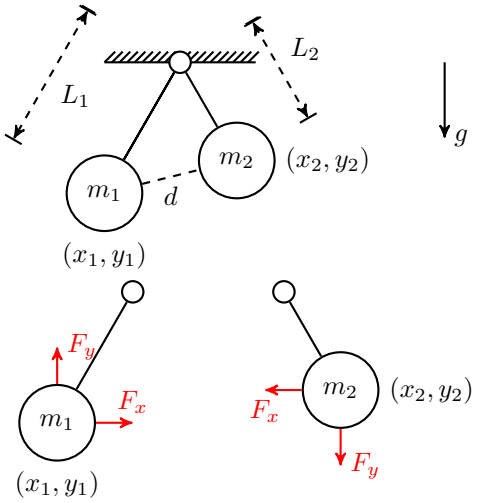
$$mL^2\ddot{q} + mgL\cos q = f_y x - f_x y = \tau, \tag{3.26}$$

where τ is the torque at the hinge due to the external force $f = [f_x\ f_y]^T$ acting on m. As we will see later, the Euler-Lagrangian approach directly would have directly led to such an equation, as would the Newton-Euler equations for a planar rigid body rotating about fixed point. We introduce it here because removing the constraints will make the development of multiple point-mass rigid bodies easier to follow. □

Example 11 (Two-Mass Pendulum)**.** Consider two simple pendula respectively with masses m_1 and m_2, lengths L_1 and L_2, and torques τ_1 and τ_2. Instead of cartesian position of the mass, since we know they are constrained, we will represent their motion using angles, as Example 10 shows how to derive.

If the angles of these pendula with the horizontal are q_1 and q_2 respectively, we obtain the independent equations for each as

$$m_1 L_1^2 \ddot{q}_1 + m_1 g L_1 \cos q_1 = \tau_1 \tag{3.27}$$
$$m_2 L_2^2 \ddot{q}_2 + m_2 g L_2 \cos q_2 = \tau_2 \tag{3.28}$$



How should we represent the motion after we add a distance constraint? One step is to observe that we can treat the constraint forces F_x and F_y on the two masses as additional external torques:

$$m_1 L_1^2 \ddot{q}_1 + m_1 g L_1 \cos q_1 = \tau_1 - y_1 F_x + x_1 F_y \tag{3.29}$$
$$m_2 L_2^2 \ddot{q}_2 + m_2 g L_2 \cos q_2 = \tau_2 + y_2 F_x - x_2 F_y \tag{3.30}$$

The idea we use is simply that:

> The forces that constrain the distance between m_1 and m_2 cannot cause any other motion.

Mathematically, the net constraint force $F = \begin{bmatrix} F_x & F_y \end{bmatrix}^T$ must lie along the line connecting the two masses. This requirement leads to

$$\frac{F_x}{x_2 - x_1} = \frac{F_y}{y_2 - y_1}$$

This expression is very convenient, since it allows us to eliminate the constraint-forces-induced torques easily:

$$m_1 L_1^2 \ddot{q}_1 + m_2 L_2^2 \ddot{q}_2 + m_1 g L_1 \cos q_1 + m_2 g L_2 \cos q_2 = \tau_1 + \tau_2 + \underbrace{(y_2 - y_1) F_x + (x_1 - x_2) F_y}_{0} \tag{3.31}$$

To reduce the two angles q_1 and q_2 to one angle, we apply the distance constraint

$$(x_2 - x_1)^2 + (y_2 - y_1)^2 = d^2 \tag{3.32}$$
$$\implies L_1^2 + L_2^2 - 2 L_1 L_2 \cos(q_2 - q_1) = d^2 \tag{3.33}$$
$$\implies q_2 = q_1 + c \tag{3.34}$$

The dynamics become

$$(m_1 L_1^2 + m_2 L_2^2) \ddot{q}_1 + m_1 g L_1 \cos q_1 + m_2 g L_2 \cos(q_1 + c) = \tau_1 + \tau_2 \tag{3.35}$$

□

Example 11 allows us to model a compound pendulum, where the mass may consist of an arbitrary number of point masses, leading to a continuous mass distribution in the limit.

Example 12 (Compound Pendulum)**.** Let each point-mass m_i be located at (x_i, y_i). As shown in Example 11, it is effectively constrained to have some constant distance L_i from the hinge, once rigidly attached to the simple pendulum rotating about that hinge. The line joining m_i to the hinge has angle $q_1 + c_i$, and the net equations are

$$\left(\sum_i m_i L_i^2 \right) \ddot{q}_1 + g \sum_i m_i L_i \cos(q_1 + c_i) = \sum_i \tau_i \tag{3.36}$$

Let the total mass be $m = \sum_i m_i$. We can define the average location $(\bar{x}, \bar{y})$ of these point-masses that all move rigidly together as

$$\bar{x}(q_1) = \frac{\sum_i m_i x_i}{\sum_i m_i} = \frac{\sum_i m_i L_i \cos(q_1 + c_i)}{m} \tag{3.37}$$

$$\bar{y}(q_1) = \frac{\sum_i m_i y_i}{\sum_i m_i} = \frac{\sum_i m_i L_i \sin(q_1 + c_i)}{m}. \tag{3.38}$$

Let's convert this average position into polar coordinates:

$$\bar{x}(q_1) = L\cos q, \qquad \bar{y}(q_1) = L\sin q \tag{3.39}$$

Some tedious algebra will show that L is independent of q_1, and that $q = q_1 + c$ for some constant c. Therefore, we get

$$\left(\sum_i m_i L_i^2\right)\ddot{q} + mgL\cos q = \tau, \tag{3.40}$$

where $\tau = \sum_i \tau_i$. We see that the effect of gravity on the system comprising all these particles is the same as a single particle located at the average mass location $(\bar{x}, \bar{y})$ with mass m. The average location is called the *center of mass*, and is related to the first moment of any continuous distribution. Given the average location, we can define the average squared distance of mass from this average, and called the *second moment of mass*, or the *moment of inertia*.

$$\mathcal{I} = \frac{1}{m}\sum_i m_i \left((x_i - \bar{x})^2 + (y_i - \bar{y})^2\right) \tag{3.41}$$

Using $x_i^2 = (\bar{x} + (x_i - \bar{x}))^2$, $y_i^2 = (\bar{y} + (y_i - \bar{y}))^2$, and some algebra will lead to the fact that

$$\left(\sum_i m_i L_i^2\right) = \mathcal{I} + mL^2 \tag{3.42}$$

Finally, we get the equation for a compound pendulum :

$$\left(\mathcal{I} + mL^2\right)\ddot{q} + mgL\cos q = \tau, \tag{3.43}$$

where m is the total mass, $\mathcal{I}$ is the moment of inertia about the center of mass, L and q are the polar coordinates of the center of mass relative to the hinge. □

The compound pendulum model in Example 12 leads to the following well-known rotational version of Newton's second law:

> Any rigid body rotating about a constant axis through its center of mass ($L = 0$) satisfies a scalar ordinary differential equation
>
> $$\mathcal{I}\ddot{q} = \tau.$$

Recall that τ is the combined torque due to all the forces acting on all the particles making up the rigid body. The equation above holds even if the center of mass is accelerating, since that acceleration does not affect the position, velocity, or acceleration of the particles *relative to the center of mass*, meaning that the stationary rotation axis derivations still apply. The only difference is that the acceleration of any point on the rigid body is not determined solely by $\ddot{q}$. The acceleration of the center of mass $(\bar{x}, \bar{y})$ is given by

$$\left(\sum_i m_i\right)\ddot{\bar{x}} = \sum_i m_i \ddot{x}_i \implies m\ddot{\bar{x}} = F_x^{ext} \tag{3.44}$$

$$\left(\sum_i m_i\right)\ddot{\bar{y}} = \sum_i m_i \ddot{y}_i \implies m\ddot{\bar{y}} = F_y^{ext}, \tag{3.45}$$

where F_x^{ext} and F_y^{ext} are the components of the sum of all the external forces acting on the particles. Every constraint force has an equal and opposite reaction constraint force, so all constraint forces cancel out under summation of mass-times-acceleration over all particles, leaving only the external forces in the sum.

We have shown three things for rigid body motion in the plane:

- The motion of **all** points on a planar rigid body can be described in terms of the position of the center of mass and an angle: $(\bar{x}, \bar{y}, q)$
- The 2D acceleration of the center of mass $\ddot{\bar{x}}, \ddot{\bar{y}}$ depends on the total mass m and the sum of forces acting on all particles
- The angular acceleration $\ddot{q}$ depends on the second moment of mass, or the moment of inertia, relative to the center of mass, and the sum of the torques at $(\bar{x}, \bar{y})$ due to all the forces on the particles

Continuous Rigid Bodies. Instead of a finite number of point-masses making up a planar rigid body, we may have a continuous distribution of mass. Consider a fixed spatial frame $\{s\}$. Instead of mass at a point, we have a *mass density* $\rho(x, y)$ at a point (x, y) in this frame. Then,

$$m = \int\int \rho(x,y) dx dy \tag{3.46}$$

$$\bar{x} = \frac{1}{m} \int\int \rho(x,y) x dx \tag{3.47}$$

$$\bar{y} = \frac{1}{m} \int\int \rho(x,y) y dy \tag{3.48}$$

$$\mathcal{I}_s = \int\int \rho(x,y)(x^2+y^2) dx dy = \mathcal{I}_b + m(\bar{x}^2+\bar{y}^2), \text{ where} \tag{3.49}$$

$$\mathcal{I}_b = \int\int \rho(x,y) \left((x-\bar{x})^2 + (y-\bar{y})^2\right) dx dy. \tag{3.50}$$

The inertia $\mathcal{I}_b$ is the inertia relative to a frame $\{b\}$ whose origin coincides with the center of mass, and whose axes are aligned with those of $\{s\}$. In general, it is always convenient to place frames associated with centers of mass.

Example 13. Three-link robot Jacobian Example 4.6 in [5]

3.1.4 Newton-Euler Equations For A 3D Rigid Body

We can apply Newton's Laws to the system of point masses m_i in 3D with coordinates $\bar{\mathbf{r}}_i$ in Cartesian frame $\{b\}$. The intuition remains the same, but the notation gets more involved.

The total linear momentum $\mathbf{p}$ of these particles in frame $\{s\}$ is

$$\mathbf{p} = \sum_i m_i \mathbf{v}_i = \sum_i m_i \left(\mathbf{v}_0 + [\omega_s] R_{sb} \bar{\mathbf{r}}_i\right) \tag{3.51}$$

$$= \left(\sum_i m_i\right) \mathbf{v}_0 + [\omega_s] R_{sb} \left(\sum_i m_i \bar{\mathbf{r}}_i\right) \tag{3.52}$$

An important choice: If we pick $\mathbf{r}_0$, the origin of $\{b\}$ as the unique point such that $\sum_i m_i \bar{\mathbf{r}}_i = 0$, and define $m = \sum_i m_i$, then

$$\mathbf{p} = m\mathbf{v}_0. \tag{3.53}$$

Newton's Second Law in $\{s\}$ then yields

$$m\dot{\mathbf{v}}_0 = \sum_i \mathbf{f}_i \tag{3.54}$$

The total angular momentum $\mathbf{h}_s$ in the frame $\{s\}$ is

$$\mathbf{h}_s = \sum_i m_i [\mathbf{r}_i]\mathbf{v}_i = \sum_i m_i \left([\mathbf{r}_i]\mathbf{v}_0 + [\mathbf{r}_i][\omega_s]R_{sb}\bar{\mathbf{r}}_i\right) \tag{3.55}$$

$$= \left(\sum_i m_i[\mathbf{r}_i]\right)\mathbf{v}_0 + \sum_i m_i \left([\mathbf{r}_i][\omega_s]R_{sb}\bar{\mathbf{r}}_i\right) \tag{3.56}$$

$$= 0 + \sum_i m_i \left([\mathbf{r}_0 + R_{sb}\bar{\mathbf{r}}_i][\omega_s]R_{sb}\bar{\mathbf{r}}_i\right) \tag{3.57}$$

$$= \sum_i m_i \left([\mathbf{r}_0][\omega_s]R_{sb}\bar{\mathbf{r}}_i\right) + \sum_i m_i \left([R_{sb}\bar{\mathbf{r}}_i][\omega_s]R_{sb}\bar{\mathbf{r}}_i\right) \tag{3.58}$$

$$= \left(-\sum_i m_i \left[R_{sb}\bar{\mathbf{r}}_i\right]^2\right)\omega_s \tag{3.59}$$

$$= \mathcal{I}_s\omega_s \tag{3.60}$$

The quantity $\mathcal{I}_s$ is the second moment of inertia of the rigid body in the frame of $\{s\}$. We may also define the second moment of inertia in the frame $\{b\}$, as

$$\mathcal{I}_b = -\left(\sum_i m_i \left[\bar{\mathbf{r}}_i\right]^2\right) \tag{3.61}$$

These definitions lead to

$$\mathbf{h}_b = \mathcal{I}_b\omega_b \tag{3.62}$$

$$\mathbf{h}_s = \mathcal{I}_s\omega_s = R_{sb}\mathbf{h}_b \tag{3.63}$$

$$\mathcal{I}_s = R_{sb}\mathcal{I}_b R_{sb}^T \tag{3.64}$$

From their definitions, $\mathcal{I}_b$ is a constant, but $\mathcal{I}_s$ varies with R_{sb}. Therefore, the Euler equation is applied as follows

$$\frac{d}{dt}\mathbf{h}_s = \sum_i m_i[\mathbf{r}_i]f_i = \tau_s \tag{3.65}$$

$$\implies \frac{d}{dt}\left(R_{sb}\mathcal{I}_b\omega_b\right) = \tau_s \tag{3.66}$$

$$\implies R_{sb}[\omega_b]\mathcal{I}_b\omega_b + R_{sb}\mathcal{I}_b\dot{\omega}_b = \tau_s \tag{3.67}$$

$$\implies [\omega_s]\mathcal{I}_s\omega_s + \mathcal{I}_s\dot{\omega}_s = \tau_s \tag{3.68}$$

We may express this equation in $\{b\}$, leading to

$$[\omega_b]\mathcal{I}_b\omega_b + \mathcal{I}_b\dot{\omega}_b = \tau_b, \tag{3.69}$$

where $\tau_b = R_{sb}^T\tau_s$

Continuous Rigid Bodies. The equation for m is

$$m = \int\int \rho(x,y,z)dxdydz \tag{3.70}$$

$$\bar{x}_s = \frac{1}{m}\int \rho_s(x,y,z)xdx \tag{3.71}$$

$$\bar{y}_s = \frac{1}{m}\int \rho_s(x,y,z)ydy \tag{3.72}$$

$$\bar{z}_s = \frac{1}{m}\int \rho_s(x,y,z)zdz \tag{3.73}$$

$$\tag{3.74}$$

where subscript s indicates quantities with respect to a fixed reference frame.

In a frame $\{b\}$ whose origin is located at the center of mass of the body, the equation for $\mathcal{I}_b$ is

$$\mathcal{I}_b = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}, \tag{3.75}$$

where

$$I_{xx} = \int\int\int (y^2 + z^2)\rho(x, y, z)dxdydz \tag{3.76}$$

$$I_{yy} = \int\int\int (x^2 + z^2)\rho(x, y, z)dxdydz \tag{3.77}$$

$$I_{zz} = \int\int\int (x^2 + y^2)\rho(x, y, z)dxdydz \tag{3.78}$$

and

$$I_{xy} = I_{yx} = -\int\int\int xy\ \rho(x, y, z)dxdydz \tag{3.79}$$

$$I_{xz} = I_{zx} = -\int\int\int xz\ \rho(x, y, z)dxdydz \tag{3.80}$$

$$I_{yz} = I_{zy} = -\int\int\int yz\ \rho(x, y, z)dxdydz \tag{3.81}$$

We don't need to solve for accelerations of all point masses on a rigid body. Instead, we derive a smaller number of rigid-body equations by constraining masses on a rigid body to have constant distance from one another. We saw in Section 3.1.4 that we may replace positions $\mathbf{r}_i = (x_i, y_i, z_i)$ and their derivatives for several particles with a position, orientation, and their derivatives involving a linear velocity and an angular velocity of a single frame.

3.1.5 Newton-Euler Equations For Rigid Body Mechanisms

For a robot modeled as inter-connected rigid bodies, the Newton-Euler Inverse Dynamics approach treats a rigid body as the basic element, but otherwise repeats the same process we applied to point-masses. Each frame has a position, velocity, and acceleration. Each frame has external forces applied to it, but also reaction forces corresponding to joints with other links.

To obtain the system dynamics, for each rigid body (previously just point-mass), we relate linear and angular accelerations (previously just linear accelerations) of the center of mass frame (previously just x_i, y_i) with the constraint forces from other rigid bodies (previously other point-masses) and external forces.

3.1.6 Recursive Newton Euler Algorithm

Once we model a set of rigid bodies using techniques from the previous sections, we are still dealing with a system of equations of the form

$$\begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} a \\ R \end{bmatrix} = \begin{bmatrix} f \\ b \end{bmatrix}, \tag{3.82}$$

however a, f, and R involve spatial (3D) velocities, accelerations, forces, and torques seen in Section 3.1.4, unlike the simpler point-mass cases we saw earlier.

One source of complexity in the Newton-Euler Inverse Dynamics formulation is that we are given information in terms of the joint angles, and not the position and orientation of link frames relative to the base frame. If we were given the latter, it would be easy to apply the rigid body equations to each links, and solve for the constraint forces and external forces as a linear system, just like we did for pendula made of point-masses.

Fortunately, we may derive the frame accelerations a from the joint angle accelerations using a recursive procedure that starts from the base frame $\{0\}$ and moves towards the end effector $\{n\}$. Once the frame accelerations a have been computed, instead of solving all the linear equations for all frames simultaneously, we may recursively solve for the external forces f required, starting from the end-effector frame $\{n\}$ to the base frame $\{0\}$. Just like for the point-mass examples, we will end up computing the constraint forces R along the way. The first recursion is called the forward pass, the second is called the backward pass. The details can be found in [5].

3.2 The Lagrangian

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

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + \tau_{friction} + \tau_e, \tag{3.83}$$

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 $D(q)$, $C(q,\dot{q})$, and $G(q)$ are the mass or inertia matrix, Coriolis matrix, 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 (3.83). The terms on the right are essentially non-conservative external forces acting on the system.

3.2.1 On Frames

As alluded to in the first chapter, a big source of confusion in modeling robots is that the same physical quantity may be described in infinitely many frames, and this infinite possibility may become a source of error.

To obtain the correct equations of motion from Newton's Laws, we must ensure that we are applying Newton's Laws in an inertial reference frame. The main consequence of this requirement is that we often formulate quantities in the body-fixed frame of a link, and then transform them into the reference/base/spatial frame.

3.2.2 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

$$\mathcal{K} = \frac{1}{2}mv^Tv + \frac{1}{2}\omega^T\mathcal{I}\omega, \tag{3.84}$$

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.

When we combine these kinetic energies, we get

$$\begin{aligned} K &= \sum_i \frac{1}{2} m_i v_i^T v_i + \frac{1}{2} \omega_i \mathcal{I} \omega_i \\ &= \sum_i \frac{1}{2} m_i \dot{q}^T J_{v_i}^T(q) J_{v_i}(q) \dot{q} + \frac{1}{2} \dot{q}^T J_{\omega_i}^T(q) R_i(q) I_i R_i^T(q) J_{\omega_i}(q) \dot{q} \\ &= \frac{1}{2} \dot{q}^T \left(\sum_i \left(m_i J_{v_i}^T J_{v_i} + J_{\omega_i}^T R_i I_i R_i^T J_{\omega_i} \right) \right) \dot{q} \qquad \text{(ignoring } (q)\text{)} \\ &= \frac{1}{2} \dot{q}^T D(q) \dot{q} \\ &= \sum_{i,j} d_{ij}(q) \dot{q}_i \dot{q}_j \end{aligned} \tag{3.85}$$

3.2.3 Potential Energy

Potential energy P due to gravity is

$$P = \sum_{i=1}^{n} m_i g \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} c_i^0(q) \tag{3.86}$$

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).

3.3 Euler-Lagrange Equations

See the following online resource for a longer description than in standard robotics texts.

Given a Lagrangian $\mathcal{L} = K - P$, we can derive an equation of motion for each generalized coordinate q_k as

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} = \tau_j \tag{3.87}$$

Given the expressions for K and P in Section 3.2, we end up with

$$\sum_{j=1}^{n} d_{kj}(q) \ddot{q}_j + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ijk}(q) \dot{q}_i \dot{q}_j + g_k(q) = \tau_k, \quad k \in \{1, \ldots, n\} \tag{3.88}$$

where

$$c_{ijk} = \frac{1}{2} \left\{ \frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right\} \tag{3.89}$$

$$g_k = \frac{\partial P}{\partial q_k} \tag{3.90}$$

These equations are combined into a single compact representation, called the Euler-Lagrangian model of a robot's dynamics, denoted as

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau, \tag{3.91}$$

where the k, j element of $C(q, \dot{q})$ is

$$c_{kj} = \sum_{i=1}^{n} c_{ijk}(q) \dot{q}_i. \tag{3.92}$$

Adding non-conservative forces such as viscous friction and externally applied forces yields (3.83).

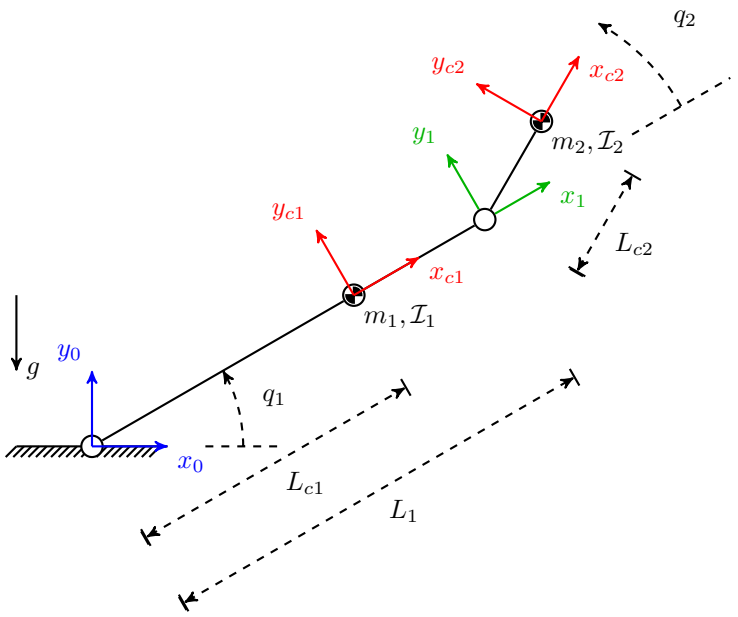


Figure 3.1: Planar Elbow Manipulator with frames according to D-H Convention

Example 14 (Planar Elbow Manipulator)**.** Refer to Example 4 for an example of the derivation of the Jacobian. Note that in that example, the Jacobian is derived for the frame corresponding to $\{c2\}$ here. Similar steps work to derive the Jacobian for for the frame $\{c1\}$ associated with $m_1, \mathcal{I}_1$. We use the DH joint variables as generalized coordinates.

We may derive velocities of $\{c1\}$ and $\{c2\}$, through Jacobians (see Example 4), as

$$v_{c1} = J_{v_{c1}}\dot{q} = \begin{bmatrix} -L_{c1}\sin q_1 & 0 \\ L_{c1}\cos q_1 & 0 \\ 0 & 0 \end{bmatrix} \dot{q} \tag{3.93}$$

$$v_{c1} = J_{v_{c2}}\dot{q} = \begin{bmatrix} -L_1\sin q_1 - L_{c2}\sin(q_1+q_2) & -L_{c2}\sin(q_1+q_2) \\ L_{c1}\cos q_1 + L_{c2}\cos(q_1+q_2) & L_{c2}\cos(q_1+q_2) \\ 0 & 0 \end{bmatrix} \dot{q} \tag{3.94}$$

Also,

$$J_{\omega 1} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad J_{\omega 2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}, \quad R_{\omega 1} = R_{z,q_1}, \qquad R_{\omega 2} = R_{z,q_1+q_2}. \tag{3.95}$$

Given these expressions for the Jacobians, and knowing the inertia tensors in the respective center-of-mass frames, we may construct the inertia matrix $D(q)$ using Equation (3.85). Since we have two links, we need to compute four terms, which we do below.

First, we have

$$J_{v1}^T J_{v1} = \begin{bmatrix} -L_{c1}\sin q_1 & 0 \\ L_{c1}\cos q_1 & 0 \\ 0 & 0 \end{bmatrix}^T \begin{bmatrix} -L_{c1}\sin q_1 & 0 \\ L_{c1}\cos q_1 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.96}$$

$$= \begin{bmatrix} L_{c1}^2 & 0 \\ 0 & 0 \end{bmatrix}. \tag{3.97}$$

Second, we compute

$$J_{v2}^T J_{v2} = \begin{bmatrix} -L_1 \sin q_1 - L_{c2}\sin(q_1+q_2) & -L_{c2}\sin(q_1+q_2) \\ L_{c1}\cos q_1 + L_{c2}\cos(q_1+q_2) & L_{c2}\cos(q_1+q_2) \\ 0 & 0 \end{bmatrix}^T \begin{bmatrix} -L_1 \sin q_1 - L_{c2}\sin(q_1+q_2) & -L_{c2}\sin(q_1+q_2) \\ L_{c1}\cos q_1 + L_{c2}\cos(q_1+q_2) & L_{c2}\cos(q_1+q_2) \\ 0 & 0 \end{bmatrix} \tag{3.98}$$

$$= \begin{bmatrix} L_1^2 + L_{c2}^2 + 2L_1 L_{c2}\cos q_2 & L_{c2}^2 + 2L_1 L_{c2}\cos q_2 \\ L_{c2}^2 + 2L_1 L_{c2}\cos q_2 & L_{c2}^2 \end{bmatrix} \tag{3.99}$$

Notice how several terms combine through standard trigonometric identities, leaving a simpler expression than what comes out of raw multiplication.

Since every column of $J_{\omega 1}$ and $J_{\omega 2}$ is aligned with the z-axis, rotation about z leaves these matrices unchanged. Therefore, $R_{z,q_1}^T J_{\omega 1} = J_{\omega 1}$ and $R_{z,q_1+q_2}^T J_{\omega 2} = J_{\omega 2}$. Therefore, we get

$$J_{\omega 1}^T R_{z,q_1} \mathcal{I}_1 R_{z,q_1}^T J_{\omega 1} = J_{\omega 1}^T \mathcal{I}_1 J_{\omega 1} \tag{3.100}$$

$$= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}^T \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & \mathcal{I}_{zz,1} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{3.101}$$

$$= \begin{bmatrix} \mathcal{I}_{zz,1} & 0 \\ 0 & 0 \end{bmatrix}, \tag{3.102}$$

and

$$J_{\omega 2}^T R_{z,q_1+q_2} \mathcal{I}_2 R_{z,q_1+q_2}^T J_{\omega 2} = J_{\omega 2}^T \mathcal{I}_2 J_{\omega 2} \tag{3.103}$$

$$= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & \mathcal{I}_{zz,2} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{3.104}$$

$$= \begin{bmatrix} \mathcal{I}_{zz,2} & \mathcal{I}_{zz,2} \\ \mathcal{I}_{zz,2} & \mathcal{I}_{zz,2} \end{bmatrix}. \tag{3.105}$$

where the terms in places marked with $*$ don't matter, so we don't assign symbols to them. Therefore

$$\begin{aligned} D(q) &= \left(m_1 J_{v_{c1}}^T J_{v_{c1}} + J_{\omega 1}^T R_{z,q_1} \mathcal{I}_1 R_{z,q_1}^T J_{\omega 1}\right) + \left(m_2 J_{v_{c2}}^T J_{v_{c2}} + J_{\omega 2}^T R_{z,q_1+q_2} \mathcal{I}_2 R_{z,q_1+q_2}^T J_{\omega 2}\right) \\ &= m_1 \begin{bmatrix} L_{c1}^2 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \mathcal{I}_{zz,1} & 0 \\ 0 & 0 \end{bmatrix} + m_2 \begin{bmatrix} L_1^2 + L_{c2}^2 + 2L_1 L_{c2}\cos q_2 & L_{c2}^2 + 2L_1 L_{c2}\cos q_2 \\ L_{c2}^2 + 2L_1 L_{c2}\cos q_2 & L_{c2}^2 \end{bmatrix} J_{v_{c2}} + \begin{bmatrix} \mathcal{I}_{zz,2} & \mathcal{I}_{zz,2} \\ \mathcal{I}_{zz,2} & \mathcal{I}_{zz,2} \end{bmatrix} \\ &= \begin{bmatrix} m_1 L_{c1}^2 + m_2(L_1^2 + L_{c2}^2 + 2L_1 L_{c2}\cos q_2) + \mathcal{I}_{zz,1} + \mathcal{I}_{zz,2} & m_2(L_{c2}^2 + 2L_1 L_{c2}\cos q_2) + \mathcal{I}_{zz,2} \\ m_2(L_{c2}^2 + 2L_1 L_{c2}\cos q_2) + \mathcal{I}_{zz,2} & m_2 L_{c2}^2 + \mathcal{I}_{zz,2} \end{bmatrix} \end{aligned} \tag{3.106}$$

For convenience, we may write

$$d_{11}(q) = m_1 L_{c1}^2 + m_2(L_1^2 + L_{c2}^2 + 2L_1 L_{c2}\cos q_2) + \mathcal{I}_{zz,1} + \mathcal{I}_{zz,2} \tag{3.107}$$

$$d_{12}(q) = d_{21}(q) = m_2(L_{c2}^2 + 2L_1 L_{c2}\cos q_2) + \mathcal{I}_{zz,2} \tag{3.108}$$

$$d_{22}(q) = m_2 L_{c2}^2 + \mathcal{I}_{zz,2} \tag{3.109}$$

Let $h = -m_2 L_1 L_{c2} \sin q_2$. Then, applying (3.89) using the mass matrix terms above, we get

$$c_{111} = c_{222} = c_{122} = 0, \quad c_{121} = c_{211} = c_{221} = h, \quad c_{112} = -h. \tag{3.110}$$

We then use (3.92) to obtain

$$C(q, \dot{q}) = \begin{bmatrix} h\dot{q}_2 & h\dot{q}_2 + h\dot{q}_1 \\ -h\dot{q}_1 & 0 \end{bmatrix} \tag{3.111}$$

Finally, we apply (3.86) to get

$$P = m_1 g L_{c1} \sin q_1 + m_2 g (L_1 \sin q_1 + L_{c2} \sin(q_1 + q_2)) \tag{3.112}$$

Therefore, we can derive the components of $G(q) = \begin{bmatrix} g_1(q) & g_2(q) \end{bmatrix}^T$ using (3.90) to obtain

$$g_1(q) = m_1 g L_{c1} \cos q_1 + m_2 g L_1 \cos q_1 + m_2 g L_{c2} \cos(q_1 + q_2) \tag{3.113}$$
$$g_2(q) = m_2 g L_{c2} \cos(q_1 + q_2) \tag{3.114}$$

Putting it all together, the equations become

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = 0 \tag{3.115}$$

We may now consider external non-conservative forces. If we consider a viscous damping coefficient b at each joint, then the external torque at a joint q_i due to damping is $-b\dot{q}_i$, so that the combined equations with damping become

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = -\begin{bmatrix} b & 0 \\ 0 & b \end{bmatrix} \dot{q} = -B\dot{q} \tag{3.116}$$

We may also apply a torque τ_i about the joint angles q_i, to obtain

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau - B\dot{q} \tag{3.117}$$

□

3.3.1 Derivation

Consider mechanism consisting of k particles, each with mass m_i, and a position $\mathbf{r}_i \in \mathbb{R}^3$ in an inertial frame. We assume that the vector $q \in \mathbb{R}^n$, called the generalized coordinates, represents a minimum number of degrees of freedom of the mechanism such that every position $\mathbf{r}_i$ is a function of q:

$$\mathbf{r}_i = \mathbf{r}_i(q_1, q_2, \ldots, q_n).$$

Simply based on this relationship, we may derive

$$\delta \mathbf{r}_i = \sum_{j=1}^{n} \frac{\partial \mathbf{r}_i}{\partial q_j} \delta q_j \tag{3.118}$$
$$\implies \dot{\mathbf{r}}_i = \mathbf{v}_i = \sum_{j=1}^{n} \frac{\partial \mathbf{r}_i}{\partial q_j} \dot{q}_j \tag{3.119}$$
$$\implies \frac{\partial \mathbf{v}_i}{\partial \dot{q}_j} = \frac{\partial \mathbf{r}_i}{\partial q_j} \tag{3.120}$$

We may also derive

$$\frac{d}{dt}\frac{\partial \mathbf{r}_i}{\partial q_j} = \sum_{l=1}^{n} \frac{\partial^2 \mathbf{r}_i}{\partial q_j \partial q_l} \dot{q}_l \tag{3.121}$$
$$= \frac{\partial}{\partial q_j} \left(\sum_{l=1}^{n} \frac{\partial \mathbf{r}_i}{\partial q_l} \dot{q}_l \right) \tag{3.122}$$
$$= \frac{\partial \mathbf{v}_i}{\partial q_j} \tag{3.123}$$

We will use these relationships between partial derivatives of $\mathbf{r}_i$ and $\mathbf{v}_i$ below.

Let a total force $\mathbf{F}_i$ act on particle m_i. Some of these forces are externally applied, and some of these forces arise due to constraints. Let the sum of external forces on m_i be $\mathbf{f}_i$, and the sum of constraint forces be $\mathbf{f}_i^{(a)}$. Then

$$\mathbf{F}_i = \mathbf{f}_i + \mathbf{f}_i^{(a)} \tag{3.124}$$

Consider an infinitesmal displacement $\delta\mathbf{r}_i$, also called a **virtual displacement** for each particle. These virtual displacements must be compatible with the constraints acting between particles.

We now focus on constrainted systems in equilibrium, meaning $\ddot{\mathbf{r}}_i = 0$. The net force on each particle is zero, so that the total work done over all particles is zero.:

$$\sum_{i=1}^{k} \mathbf{F}_i^T \delta\mathbf{r}_i = 0 \tag{3.125}$$

The main insight is that for certain types of constraints, the constraint forces cannot do any work. This statement is known as the **principle of virtual work**. This statement is another way of saying that the allowable virtual displacements have to be perpendicular to all constraint forces. In particular, for robotic systems, we focus on distance-preserving constraints, for which the principle of virtual work holds. It leads to

$$\sum_{i=1}^{k} \left(\mathbf{f}_i^{(a)}\right)^T \delta\mathbf{r}_i = 0 \tag{3.126}$$

Since the total work done is zero, we get the relationship

$$\sum_{i=1}^{k} \left(\mathbf{f}_i\right)^T \delta\mathbf{r}_i = 0 \tag{3.127}$$

So, using the principle of virtual work, we are able to relate external forces to displacements, and ignore the constraint forces.

Suppose that $\ddot{\mathbf{r}}_i \neq 0$, so that the particles are not in equilibrium. We can still obtain a relationship between external forces and particle displacements if we use D'Alembert's principle, which requires us to treat the change in momentum as an external force. Doing so, we get the new equation

$$\sum_{i=1}^{k} \left(\mathbf{f}_i - m_i\ddot{\mathbf{r}}_i\right)^T \delta\mathbf{r}_i = 0 \tag{3.128}$$

$$\implies \sum_{i=1}^{k} {\mathbf{f}_i}^T \delta\mathbf{r}_i - \sum_{i=1}^{k} m_i\ddot{\mathbf{r}}_i^T \delta\mathbf{r}_i = 0 \tag{3.129}$$

This equation is essentially an expression for the Euler-Lagrange equations, and we just need to modify and substitute terms to get the familiar expression. We start by noting that

$$\sum_{i=1}^{k} {\mathbf{f}_i}^T \delta\mathbf{r}_i = \sum_{i=1}^{k} {\mathbf{f}_i}^T \sum_{j=1}^{n} \frac{\partial\mathbf{r}_i}{\partial q_j}\delta q_j \tag{3.130}$$

$$= \sum_{j=1}^{n} \left(\sum_{i=1}^{k} \mathbf{f}_i^T \frac{\partial\mathbf{r}_i}{\partial q_j}\right)\delta q_j \qquad \text{(switch sum order)} \tag{3.131}$$

$$= \sum_{j=1}^{n} \psi_j \delta q_j, \tag{3.132}$$

where $\psi_j = \sum_{i=1}^{k} \mathbf{f}_i^T \frac{\partial\mathbf{r}_i}{\partial q_j}$ is called the **generalized force** corresponding to generalized coordinate q_j.

We now deal with the second term in (3.129), namely $\sum_{i=1}^{k} m_i \ddot{\mathbf{r}}_i^T \delta \mathbf{r}_i$. Observe that

$$\frac{d}{dt}\left[m_i \dot{\mathbf{r}}_i^T \delta \mathbf{r}_i\right] = m_i \ddot{\mathbf{r}}_i^T \delta \mathbf{r}_i + m_i \dot{\mathbf{r}}_i^T \frac{d}{dt}\delta \mathbf{r}_i. \tag{3.133}$$

Therefore,

$$\sum_{i=1}^{k} m_i \ddot{\mathbf{r}}_i^T \delta \mathbf{r}_i = \sum_{i=1}^{k}\left(\frac{d}{dt}\left[m_i \dot{\mathbf{r}}_i^T \delta \mathbf{r}_i\right] - m_i \dot{\mathbf{r}}_i^T \frac{d}{dt}\delta \mathbf{r}_i\right) \tag{3.134}$$

$$= \sum_{i=1}^{k}\left(\frac{d}{dt}\left[m_i \dot{\mathbf{r}}_i^T \sum_{j=1}^{n}\frac{\partial \mathbf{r}_i}{\partial q_j}\delta q_j\right] - m_i \dot{\mathbf{r}}_i^T \frac{d}{dt}\sum_{j=1}^{n}\frac{\partial \mathbf{r}_i}{\partial q_j}\delta q_j\right) \tag{3.135}$$

$$= \sum_{j=1}^{n}\left(\sum_{i=1}^{k}\left(\frac{d}{dt}\left[m_i \dot{\mathbf{r}}_i^T \frac{\partial \mathbf{r}_i}{\partial q_j}\right] - m_i \dot{\mathbf{r}}_i^T \frac{d}{dt}\frac{\partial \mathbf{r}_i}{\partial q_j}\right)\right)\delta q_j \tag{3.136}$$

$$= \sum_{j=1}^{n}\left(\sum_{i=1}^{k}\left(\frac{d}{dt}\left[m_i \mathbf{v}_i^T \frac{\partial \mathbf{v}_i}{\partial \dot{q}_j}\right] - m_i \mathbf{v}_i^T \frac{\partial \mathbf{v}_i}{\partial q_j}\right)\right)\delta q_j, \tag{3.137}$$

where we have used (3.120) and (3.123) in the last step. The motivation for doing so is that we can relate the equation above to the total Kinetic energy of the system, which is

$$K = \sum_{i=1}^{k}\frac{1}{2} m_i \mathbf{v}_i^T \mathbf{v}_i, \tag{3.138}$$

by taking partial derivatives:

$$\frac{\partial K}{\partial q_j} = \sum_{i=1}^{k} m_i \mathbf{v}_i^T \frac{\partial \mathbf{v}_i}{\partial q_j}, \quad \frac{\partial K}{\partial \dot{q}_j} = \sum_{i=1}^{k} m_i \mathbf{v}_i^T \frac{\partial \mathbf{v}_i}{\partial \dot{q}_j}. \tag{3.139}$$

We use these partial derivatives of K in (3.137) to get

$$\sum_{i=1}^{k} m_i \ddot{\mathbf{r}}_i^T \delta \mathbf{r}_i = \sum_{j=1}^{n}\left(\frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j}\right)\delta q_j \tag{3.140}$$

Finally, we can substitute (3.132) and (3.140) in (3.129)

$$\sum_{j=1}^{n}\left(\psi_j - \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} + \frac{\partial K}{\partial q_j}\right)\delta q_j = 0 \tag{3.141}$$

Assume that the generalized forces ψ_j consists of two terms, one due to conservative forces generalized and the other due to non-conservative generalized forces.

A conservative force is the partial derivative with respect to position of a potential energy term. Therefore, we rewrite ψ_j as

$$\psi_j = -\frac{\partial P}{\partial q_j} + \tau_j, \tag{3.142}$$

where $P(q)$ is the total potential energy of the system and τ_j is the non-conservative generalized forces corresponding to q_j.

If we define $\mathcal{L} = K - P$, then we can rewrite (3.141) as

$$\sum_{j=1}^{n}\left(\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_j} - \frac{\partial \mathcal{L}}{\partial q_j} - \tau_j\right)\delta q_j = 0 \tag{3.143}$$

Since this equation must hold for all possible virtual displacements δq_j, the coefficients of δq_j must be zero. In other words,

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_j} - \frac{\partial \mathcal{L}}{\partial q_j} = \tau_j, \quad \text{for } j \in \{1, \ldots, n\}, \tag{3.144}$$

which are the Euler-Lagrange equations. Again, these equations may be combined to form the Euler Lagrange equations of the form

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau. \tag{3.145}$$

3.4 Properties of the Euler-Lagrange Equations

3.4.1 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

$$\begin{aligned} n_{kj} &= \dot{d}_{kj} - 2c_{kj} \\ &= \sum_{i=1}^{n} \frac{\partial d_{kj}}{\partial q_i}\dot{q}_i - 2\sum_{i=1}^{n} c_{ijk}\dot{q}_i \\ &= \sum_{i=1}^{n} \left[\frac{\partial d_{kj}}{\partial q_i} - c_{ijk}\right]\dot{q}_i \\ &= \sum_{i=1}^{n} \left[\frac{\partial d_{kj}}{\partial q_i} - 2\frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}\right]\dot{q}_i \\ &= \sum_{i=1}^{n} \left[\frac{\partial d_{ij}}{\partial q_k} - \frac{\partial d_{ki}}{\partial q_j}\right]\dot{q}_i \end{aligned} \tag{3.146}$$

The expression for n_{jk} will be

$$n_{jk} = \left[\frac{\partial d_{ik}}{\partial q_j} - \frac{\partial d_{ji}}{\partial q_k}\right]\dot{q}_i \tag{3.147}$$

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 3.5.

3.4.2 Bounds on Inertia Matrix

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

$$\lambda_m I_{n\times n} \leq D(q) \leq \lambda_M I_{n\times n} < \infty \tag{3.148}$$

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.

3.4.3 Linearity in Parameters

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

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Y(q, \dot{q}, \ddot{q})\theta \tag{3.149}$$

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

3.5 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

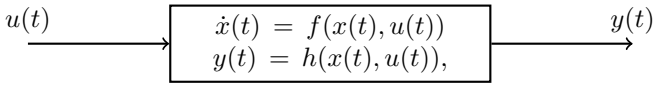
1. Stability
2. L_2 gain stability
3. Stable behavior under feedback interconnections

Consider a system

$$\dot{x}(t) = f(x(t), u(t)) \tag{3.150}$$
$$y(t) = h(x(t), u(t)), \tag{3.151}$$

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 15. 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 16. Continued Let's compute the time derivative of the storage function:

$$\frac{d}{dt}V(q_c) = \left(\frac{\partial V}{\partial q_c}\right)\dot{q}_c = \left(\frac{1}{C}q_c\right)\dot{q}_c \tag{3.152}$$

$$= \frac{1}{C}q_c\dot{q}_c = \frac{1}{C}\left(Ce_c\right)\left(i_c\right) \tag{3.153}$$

$$= e_c i_c \tag{3.154}$$

$$\implies \dot{V}(t) = S\left(e_c(t), i_c(t)\right) \tag{3.155}$$

$$\implies V(t_1) - V(t_0) \leq \int_{t_0}^{t_1} S\left(e_c(\tau), i_c(\tau)\right) d\tau \tag{3.156}$$

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.

3.5.1 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:

$$\dot{V} = \dot{q}^T D(q)\ddot{q} + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \left(\frac{\partial PE(q)}{\partial q}\right)^T \dot{q} \tag{3.157}$$

$$= \dot{q}^T D(q) D^{-1}(q) \left(\tau - C(q,\dot{q})\dot{q} - \frac{\partial PE(q)}{\partial q}\right) + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \left(\frac{\partial PE(q)}{\partial q}\right)^T \dot{q} \tag{3.158}$$

$$= \dot{q}^T \tau + \frac{1}{2}\dot{q}^T \left(\dot{D}(q) - 2C(q,\dot{q})\right)\dot{q} \tag{3.159}$$

$$= \dot{q}^T \tau \tag{3.160}$$

$$= S(y,u) \tag{3.161}$$

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

3.5.2 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 [4]. 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 [3].

Power System Control

Electrical systems are also networks of components, and passivity naturally applies to the analysis and control of such systems [1].

3.6 Actuator Models

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

3.6.1 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

$$L\frac{d}{dt}i_a + Ri_a = V - V_b, \tag{3.162}$$

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.

3.6.2 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

$$\begin{aligned} J_m\ddot{\theta}_m + B_m\dot{\theta}_m &= \tau_m - \tau_l/r \\ &= K_m i_a - \tau_l/r \end{aligned} \tag{3.163}$$

We can rewrite (3.162) and (3.164b) as

$$(Ls + R)I_a(s) = V(s) - K_b\ s\Theta_m(s), \tag{3.164a}$$
$$(J_m s^2 + B_m s)\Theta_m(s) = K_m I_a(s) - \tau_l(s)/r \tag{3.164b}$$

We combine these equations to obtain

$$s\left((Ls + R)(J_m s + B_m) + K_b K_m\right)\Theta_m(s) = K_m V(s) - \frac{(Ls + R)}{r}\tau_l(s). \tag{3.165}$$

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.

$$s\left((J_m s + B_m) + \frac{K_b K_m}{R}\right)\Theta_m(s) = \frac{K_m}{R}V(s) - \frac{1}{r}\tau_l(s). \tag{3.166}$$

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

$$J\ddot{\theta}_m + B\dot{\theta}_m = u(t) - d(t). \tag{3.167}$$

Alternatively,

$$(Js^2 + Bs)\Theta_m(s) = U(s) - D(s) \tag{3.168}$$

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

3.6.3 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

$$J_l\ddot{\theta}_l + B_l\dot{\theta}_l + k(\theta_l - \theta_m) = 0, \text{ and} \tag{3.169}$$
$$J_m\ddot{\theta}_l + B_m\dot{\theta}_l + k(\theta_m - \theta_l) = u, \tag{3.170}$$

where u is input torque applied to the motor shaft.

$$p_l(s) = J_l s^2 + B_l s + k \tag{3.171}$$
$$p_m(s) = J_m s^2 + B_m s + k \tag{3.172}$$
$$\frac{\Theta_l(s)}{U(s)} = \frac{k}{p_l(s)p_k(s) - k^2} \tag{3.173}$$
$$\frac{\Theta_m(s)}{U(s)} = \frac{p_l(s)}{p_l(s)p_k(s) - k^2} \tag{3.174}$$
$$\tag{3.175}$$

The open-loop characteristic equation is

$$J_l J_m s^4 + (J_l B_m + J_m B_l)\, s^3 + (k(J_l + J_m) + B_l B_m)\, s^2 + k(B_l + B_m)s \tag{3.176}$$

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

$$x_1 = \theta_l, \quad x_2 = \dot{\theta}_l, \quad x_3 = \theta_m, \quad x_4 = \dot{\theta}_m. \tag{3.177}$$

The state space model is

$$\dot{x}_1 = x_2 \tag{3.178a}$$
$$\dot{x}_2 = -\frac{k}{J_l}x_1 - \frac{B_l}{J_l}x_2 + \frac{k}{J_l}x_3 \tag{3.178b}$$
$$\dot{x}_3 = x_4 \tag{3.178c}$$
$$\dot{x}_4 = \frac{k}{J_m}x_1 - \frac{k}{J_m}x_3 - \frac{B_m}{J_m}x_4 + \frac{1}{J_m}u \tag{3.178d}$$

which we can represent compactly as the linear system

$$\dot{x} = Ax + Bu \tag{3.179}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{J_l} & -\frac{B_l}{J_l} & \frac{k}{J_l} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{J_m} & 0 & -\frac{k}{J_m} & -\frac{B_m}{J_m} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J_m} \end{bmatrix}. \tag{3.180}$$

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.

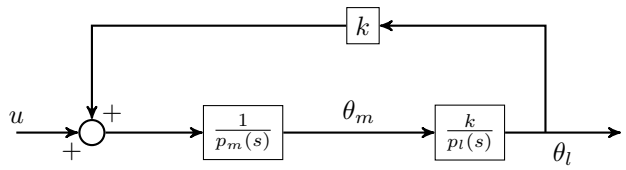


Figure: Model from voltage ($u = K_m V/R$) to load angle θ_l.

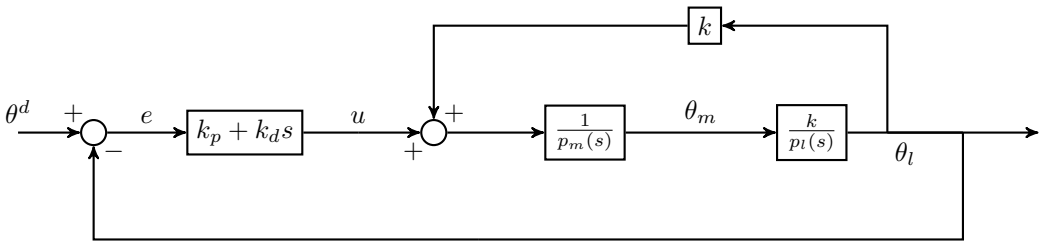


Figure: Regulation using PD Control when sensor reads load angle θ_l.

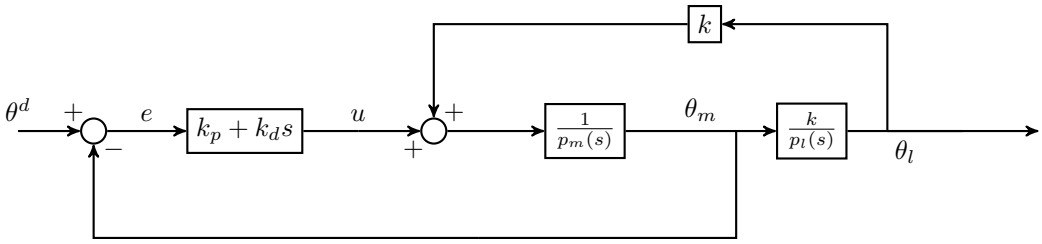


Figure: Regulation using PD Control when sensor reads motor angle θ_m.

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 (3.180).

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), $\dot{V} \equiv 0 \implies \dot{q} \equiv 0 \implies \ddot{q} \equiv 0 \implies u = -K_P(q - q_d) \equiv 0 \implies q \equiv q_d$.

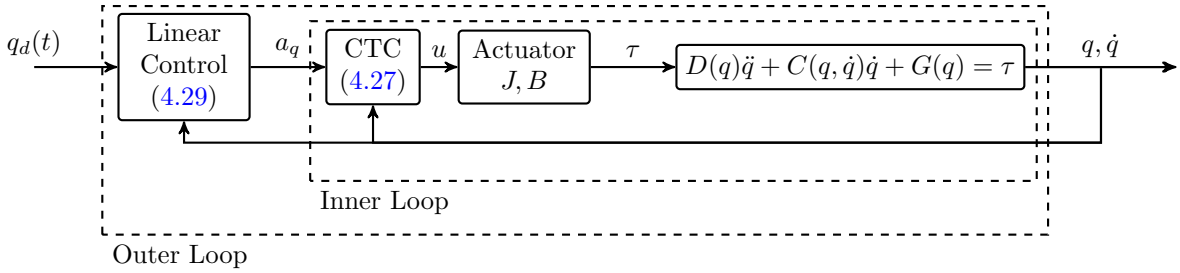


Figure 4.1: Joint space inverse dynamics control

4.2.2 Inverse Dynamics Control

Consider

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

We create an inner-loop outer-loop control as shown in Figure 4.2. Let

$$\text{CTC:} \qquad u = M(q)a_q + C(q,\dot{q})\dot{q} + G(q) + B\dot{q}. \tag{4.27}$$

The closed loop simply becomes

$$M(q)\ddot{q} = M(q)a_q \implies \ddot{q} = a_q \tag{4.28}$$

We can choose a_q to be a linear PD control:

$$a_q = \ddot{q}(t) + K_P(q_d(t) - q(t)) + K_D(\dot{q}_d(t) - \dot{q}(t)). \tag{4.29}$$

If $e(t) = q(t) - q_d(t)$, then the equation above reduces to

$$\ddot{e} + K_D\dot{e} + K_P e = 0.$$

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,

$$\ddot{X} = J_a(q)\ddot{q} + \dot{J}_a(q)\dot{q} \tag{4.30}$$

If we choose

$$a_q = J_a(q)^{-1}\left(a_X - \dot{J}_a(q)\dot{q}\right) \tag{4.31}$$

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

$$\ddot{X} = a_X \tag{4.32}$$

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

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u. \tag{4.33}$$

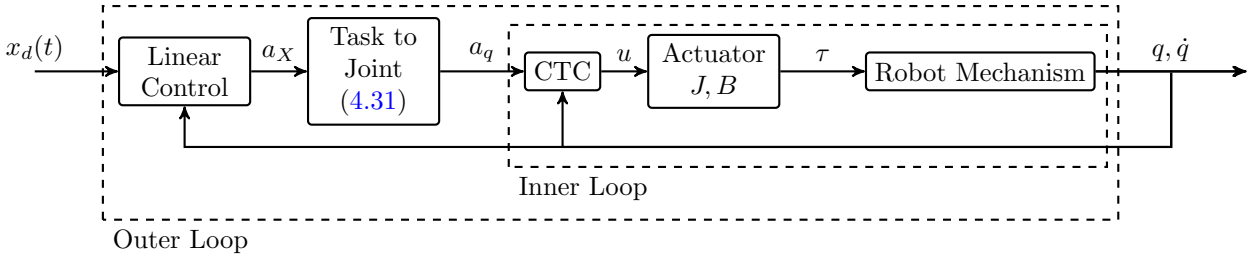


Figure 4.2: Task Space Inverse Dynamics. Notice that the inner loop is the same as for joint space inverse dynamics / computed torque control.

We design our control based on the assumed system Consider

$$\hat{M}(q)a_q + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) = u. \tag{4.34}$$

We model the closed loop as

$$\ddot{q} = a_q + \eta(q,\dot{q},\ddot{q},a_q) \tag{4.35}$$

where

$$\begin{aligned}\eta(q,\ddot{q},a_q) &= M(q)^{-1}\left((M(q)-\hat{M}(q))a_q + (C(q,\dot{q})-\hat{C}(q,\dot{q}))\dot{q} + G(q) - \hat{G}(q)\right)\\ &= M^{-1}\left(\tilde{M}a_q + \tilde{C}\dot{q} + \tilde{G}\right)\\ &= Ea_q + M^{-1}\left(\tilde{C}\dot{q} + \tilde{G}\right)\end{aligned} \tag{4.36}$$

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

$$\dot{e} = \begin{bmatrix} 0 & I \\ -K_0 & -K_1 \end{bmatrix} e + \begin{bmatrix} 0 \\ I \end{bmatrix}(\delta a + \eta) \tag{4.37}$$

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

$$\begin{aligned}\dot{V} &= e^T PAe + e^T A^T Pe + 2e^T PB(\delta a + \eta)\\ &= -e^T Qe + 2e^T PB(\delta a + \eta)\end{aligned} \tag{4.38}$$

We choose

$$\delta a = \begin{cases} -\rho(e,t)\frac{B^T Pe}{\|B^T Pe\|} & , \quad \text{if } \|B^T Pe\| \neq 0 \\ 0 & , \quad \text{if } \|B^T Pe\| = 0 \end{cases} \tag{4.39}$$

Let $w = B^T Pe$. Then the second term in (4.38) is

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

So, $\dot{V} \le -e^T Qe < 0$.

All of this works if $\|\eta\| \le \rho(e,t)$. To define such a bound, consider

$$\begin{aligned}\eta &= Ea_q + M^{-1}\left(\tilde{C}\dot{q} + \tilde{G}\right)\\ &= E\delta a + E(\ddot{q}_d(t) - K_0\tilde{q} - K_1\tilde{\dot{q}}) + M^{-1}\left(\tilde{C}\dot{q} + \tilde{G}\right)\\ &\implies \|\eta\| \le \alpha\|\delta a\| + \gamma_1\|e\| + \gamma_2\|e\|^2 + \gamma_3\end{aligned} \tag{4.40}$$

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

$$\rho(e,t) = \frac{1}{1-\alpha}\left(\gamma_1\|e\| + \gamma_2\|e\|^2 + \gamma_3\right). \tag{4.41}$$

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

$$\hat{M} = \frac{2}{\overline{M} + \underline{M}} I \tag{4.42}$$

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

Continuous Robust Control

The robust controller (4.39) 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.

$$\delta a = \begin{cases} -\rho(e,t)\frac{B^T P e}{\|B^T P e\|} & , \quad \text{if } \|B^T P e\| > \epsilon \\ -\frac{\rho(e,t)}{\epsilon} B^T P e & , \quad \text{if } \|B^T P e\| \leq \epsilon \end{cases} \tag{4.43}$$

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

$$\begin{aligned} \dot{V} &= -e^T Q e + 2w^T(\delta a + \eta) \\ &\leq -e^T Q e + 2w^T(-\frac{\rho}{\epsilon} w + \rho\frac{w}{\|w\|}) \\ &\leq -e^T Q e - 2\frac{\rho}{\epsilon}\|w\|^2 + 2\rho\|w\| \end{aligned} \tag{4.44}$$

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

Thus

$$\dot{V} \leq -e^T Q e + \epsilon\frac{\rho}{2} \tag{4.45}$$

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

$$\delta = \left(\frac{\epsilon\rho}{2\lambda_{min}(Q)}\right)^{1/2} \tag{4.46}$$

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

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Y(q,\dot{q},\ddot{q})\theta = u. \tag{4.47}$$

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

$$\ddot{\tilde{q}} + K_1\dot{\tilde{q}} + K_0\tilde{q} = M^{-1}Y(q,\dot{q},\ddot{q})\tilde{\theta} = \Phi\tilde{\theta}, \tag{4.48}$$

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

We get the ODE

$$\dot{e} = Ae + B\Phi\tilde{\theta} \tag{4.49}$$

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

$$V = e^T P e + \tilde{\theta}^T \Gamma \tilde{\theta}. \tag{4.50}$$

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

$$\dot{V} = -e^T Q e + 2\tilde{\theta}^T \left(\Phi^T B^T P e + \Gamma \dot{\hat{\theta}} \right) \tag{4.51}$$

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

$$\dot{\hat{\theta}} = -\Gamma^{-1} \Phi^T B^T P e \tag{4.52}$$

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

$$\tau_m = M(q)a + C(q,\dot{q})v + \frac{\partial PE(q)}{\partial q} - Kr, \text{ where} \tag{4.53}$$

$$v = \dot{q}_d - \Lambda\tilde{q} \tag{4.54}$$
$$a = \dot{v} = \ddot{q}_d - \Lambda\dot{\tilde{q}} \tag{4.55}$$
$$r = \dot{q} - v = \dot{\tilde{q}} + \Lambda\tilde{q}, \tag{4.56}$$

with K, Λ **diagonal matrices** of positive gains.

Note that

$$\dot{r} = \ddot{q} - a \tag{4.57}$$

Applying the control law, we get

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + \frac{\partial PE(q)}{\partial q} = M(q)a + C(q,\dot{q})v + \frac{\partial PE(q)}{\partial q} - Kr \tag{4.58}$$
$$\implies M(q)(\ddot{q} - a) + C(q,\dot{q})(\dot{q} - v) + Kr = 0, \tag{4.59}$$
$$\tag{4.60}$$

so that the closed loop becomes

$$M(q)\dot{r} + C(q,\dot{q})r + Kr = 0. \tag{4.61}$$

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

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

$$\begin{aligned} \dot{V} &= r^T M(q)\dot{r} + r^T \dot{M}(q)r + 2\tilde{q}^T \Lambda K \dot{\tilde{q}} \\ &= r^T \left(-C(q,\dot{q})r - Kr\right) + r^T \dot{M}(q)r + 2\tilde{q}^T \Lambda K \dot{\tilde{q}} \\ &= -r^T Kr + 2\tilde{q}^T \Lambda K \dot{\tilde{q}} + \frac{1}{2} r^T \left(\dot{M}(q) - 2C\right) r \\ &= -(\dot{\tilde{q}} + \Lambda\tilde{q})^T K(\dot{\tilde{q}} + \Lambda\tilde{q}^T) + 2\tilde{q}^T \Lambda K \dot{\tilde{q}} \\ &= -\tilde{q}^T \Lambda^T K \Lambda \tilde{q} - \dot{\tilde{q}}^T K \dot{\tilde{q}} \\ &= -e^T Q e \end{aligned} \tag{4.63}$$

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

$$u = \hat{M}(q)a + \hat{C}(q,\dot{q})v + \hat{G}(q) - Kr = Y(q,\dot{q},a,v)\hat{\theta} - Kr \tag{4.64}$$

The closed-loop becomes

$$M(q)r + C(q,\dot{q})r + Kr = Y(\hat{\theta} - \theta) \tag{4.65}$$

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

$$\dot{V} = -e^T Q e + r^T Y(\delta\theta + \tilde{\theta}) \tag{4.66}$$

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

$$M(q)r + C(q,\dot{q})r + Kr = Y(\hat{\theta} - \theta) \tag{4.67}$$

The Lyapunov function is

$$V = \frac{1}{2}r^T M(q)r + \tilde{q}^T \Lambda K \tilde{q} + \frac{1}{2}\tilde{\theta}^T \Gamma \tilde{\theta} \tag{4.68}$$

and the update law becomes

$$\dot{\hat{\theta}} = -\Gamma^{-1} Y(q,\dot{q},a,v)r \tag{4.69}$$

Again we see that

$$\dot{V} = -\tilde{q}^T \Lambda^T K \Lambda \tilde{q} - \dot{\tilde{q}}^T K \dot{\tilde{q}} - \tilde{\theta}^T (Y^T r + \Gamma \dot{\hat{\theta}}) \tag{4.70}$$

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

$$\dot{V} = \dot{q}^T \tau_m + \dot{q}^T J^T(q) F \tag{4.71}$$

$$= \dot{q}^T \tau_m + \xi^T F, \tag{4.72}$$

$$\tag{4.73}$$

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 Direct Force Control

Consider the usual Euler Lagrangian 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.

Direct Force Control. Let's choose the control

$$\tau = \underbrace{G(q) - J^T F_d}_{\text{feed-forward}} \underbrace{- J^T \left(K_p F_e + K_i \int F_e(s) ds \right)}_{\text{feedback}} \tag{4.74}$$

The closed-loop equations become

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + J^T F_{tip}(t) \tag{4.75}$$

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = G(q) - J^T F_d - J^T \left(K_p F_e + K_i \int F_e(s) ds \right) + J^T F_{tip}(t) \tag{4.76}$$

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

$$(\approx 0) + (\approx 0) + G(q) = G(q) - J^T F_d - J^T \left(K_p F_e + K_i \int F_e(s) ds \right) + J^T F_{tip}(t) \tag{4.77}$$

$$\implies 0 = J^T \left((I + K_p) F_e + K_i \int F_e(s) ds \right) \tag{4.78}$$

$$\implies 0 = (I + K_p)\dot{F}_e + K_i F_e \quad \text{(if } J(q) \text{ is non-singular)} \tag{4.79}$$

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

$$G(q) = \tau + J^T(q) F_{tip}(t) \tag{4.80}$$

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. In practice, this approach avoids the challenges of implementing an accurate force sensor.

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. When these quantities are defined in a frame attached to a body, the quantities ξ and F are called (body) twists and wrenches respectively, and are both six-dimensional. The usual symbols for a twist and a wrench are $\mathcal{V}$ and $\mathcal{F}$.

Given a configuration space corresponding to a mechanical system, twists belong to the tangent space $\mathbf{M}$ and wrenches belong to the co-tangent space $\mathbf{F}$, and their product corresponds to power. In fact, each configuration has its own tangent space and co-tangent space. See Chapter 1 of [2] for a description of these concepts. The main point used here is that $\mathbf{M}$ and $\mathbf{F}$ are vector spaces.

The numerical value of the power depends on the bases we choose for the spaces $\mathbf{M}$ and $\mathbf{F}$. 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 $\mathbf{M}$ and $\{f_1, \ldots, f_6\}$ is a basis for $\mathbf{F}$, these two bases are *reciprocal* if

$$e_i^T f_j = \begin{cases} 1 & , \quad \text{if } i = j \\ 0 & , \quad \text{if } i \neq j. \end{cases} \tag{4.81}$$

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

$$\mathcal{V}^T \mathcal{F} = v^T f + \omega^T n = 0 \tag{4.82}$$

The advantage of using reciprocal bases for $\mathbf{M}$ and $\mathbf{F}$ is that the product $\mathcal{V}^T \mathcal{F}$ has the invariance we want. Therefore, the reciprocity condition (4.82) is invariant with respect to the choice of reciprocal bases.

Natural and Artificial Constraints

The adjectives *natural* and *artificial* are simply a way to distinguish between restrictions on motion that naturally arise out of physical contact, and restrictions that we artificially choose to get a task done.

We discuss first discuss natural constraints which are defined using (4.82). 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 $\mathcal{V}$ and $\mathcal{F}$.

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.

(These examples are described in [5]).

4.4.4 Hybrid Force / Position Control

If we want to track a trajectory $x_d(t)$ in the task space, we may use a control

$$\tau_{acc} = D(q)a_q + C(q, \dot{q})\dot{q} + G(q), \text{ where} \tag{4.83}$$

$$a_q = J(q)^{-1}\left(a_X - \dot{J}(q)\dot{q}\right), \text{ and} \tag{4.84}$$

$$a_X = \ddot{x}_d(t) - K_P(x - x_d(t)) - K_D(\dot{x} - \dot{x}_d(t)). \tag{4.85}$$

This controller works ($x(t) \to x_d(t)$) when no external force is present on the robot (and the model is known perfectly). This requirement implies that the robot shouldn't contact anything in the environment.

If we want to achieve a force F_d at a contact point, we may use the controller

$$\tau_f = G(q) - J^T a_F = G(q) - J^T F_d - J^T \left(K_p(F_d - F_{tip}) + K_i \int (F_d - F_{tip}(s))ds\right) \tag{4.86}$$

As mentioned earlier, this controller produces complex motion in general, but for quasi-static motions ($\ddot{q} \approx 0$ and $\dot{q} \approx 0$), we see that $F_{tip}(t) \to F_d$.

The two controllers seem to only work in conditions that are incompatible: position control at end-effector requires no contact forces, and force control at the end-effector requires no motion. If we choose the spatial directions along which we want motion or force well, we may be able to still use both controllers at the same time, leading to what is known as **hybrid** force/position control.

To achieve tracking of a desired task-space position $x_d(t)$ and force $F_d(t)$, we first define task space accelerations a_X and task-space forces a_F . The core idea is to ensure that motion corresponding to acceleration $a_X \in \mathbb{R}^6$ and the force $a_F \in \mathbb{R}^6$ lie along mutually orthogonal directions, so that along each of these directions, the conditions for either force control or position control are satisfied, allowing the two controllers to work together. The next example shows a simplified case.

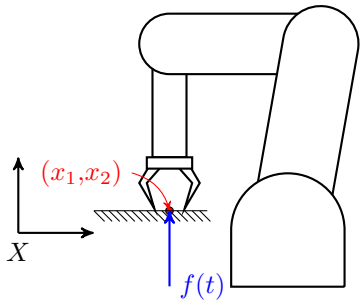


Figure 4.3: Simple contact in 2D

Example 17 (Simple Hybrid Force/Position Control)**.** Consider the state $x \in \mathbb{R}^2$, $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$. Let the dynamics be

$$m\ddot{x} = F_c + F_{ext}(t),$$

where c stands for control, and ext stands for external. We can choose F_c, but $F_{ext}(t)$ is from the environment. Assume that

$$F_{ext}(t) = \begin{bmatrix} 0 \\ f(t) \end{bmatrix}$$

. We get

$$m \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} = \begin{bmatrix} F_{c,1} \\ F_{c,2} \end{bmatrix} + \begin{bmatrix} 0 \\ f(t) \end{bmatrix} \tag{4.87}$$

The external force $f(t)$ could exist for many reasons, but we focus on the case where it is the reaction force that arises out of contact with an environment. For an illustration, see Figure 4.3.

This example is simple enough to see that we can implement a position control along x_1 to track a motion $x_{1,d}(t)$, by choosing

$$F_{pos} = m \begin{bmatrix} \ddot{x}_{1,d}(t) - K_P(x - x_{1,d}(t)) - K_D(\dot{x} - \dot{x}_{1,d}(t)) \\ 0 \end{bmatrix}. \tag{4.88}$$

To achieve an intended contact force $f_d(t)$, we choose the force controller

$$F_{force} = m \begin{bmatrix} 0 \\ -f_d(t) - k_p\left(f_d(t) - f(t)\right) - k_i\left(\int_0^t f_d(s) - f(s)ds\right) \end{bmatrix}. \tag{4.89}$$

Notice that we previously saw that F_{pos} works when there is no contact, and F_{force} works when there is no motion. The force equation is trickier, because it requires contact to exist and the reaction force to be positive. However, applying the corresponding control will maintain contact.

The key point is that implementing the combined control

$$F_c = F_{pos} + F_{force} \tag{4.90}$$

when both motion and force exist will work, because the desired motion and forces are perpendicular to each other. If we don't obey this requirement, these controllers will interfere with each other, and we may neither achieve the desired position nor achieve the desired force. □

For the general case, consider the manipulator equations

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + J^T(q)F_{tip} \tag{4.91}$$

Given any a_X and a_F, we may use the control

$$\tau_{hyb} = M(q)a_q + C(q,\dot{q})\dot{q} + G(q) - J^T(q)a_F, \text{ where} \tag{4.92}$$

$$a_q = J(q)^{-1}\left(a_X - \dot{J}(q)\dot{q}\right), \tag{4.93}$$

to obtain the closed loop

$$\ddot{X} = a_X + J(q)M(q)^{-1}J^T(q)(F_{tip} - a_F). \tag{4.94}$$

We may relate this closed-loop again to what we know about the separate controllers. If $F_{tip} = 0$, we choose $a_F = 0$ and we could choose a_X to get $x(t) \to x_d(t)$. If the robot was quasi-static, so that $\ddot{X} \approx 0$, then if $J(q)M(q)^{-1}J^T(q)$ is non-singular we could choose $a_F = F_d$, so that $F_{tip} = F_d$, or define $a_F = F_d - K_P(F_{tip} - F_d) - K_I \int (F_{tip}(s) - F_d)ds$ to get a self-correcting system where $F_{tip} \to F_d$.

If we choose a_F and a_X such that the resulting motion twist and the contact force wrenches are orthogonal, then we still get the desired convergence in position and forces, despite the robot experiencing both motion and contact. The idea is to view equation (4.94) differently:

$$\ddot{X} = a_X + J(q)M(q)^{-1}J^T(q)(F_{tip} - a_F). \tag{4.95}$$

If we choose a_X and a_F correctly, we may view this equation as the sum of two independent equations, where independence is in terms of the subsets of $\mathbb{R}^6$ in which the motion and forces reside:

$$\ddot{X} = a_X, \qquad 0 = J(q)M(q)^{-1}J^T(q)(F_{tip} - a_F). \tag{4.96}$$

If a_X and a_F are now designed appropriately, the motion stays in its subspace, and the contact forces stay in their own independent subspace, like in Example 17. The PD position control (which assumes no external forces) and PI force control (which assumes no motion) strategies will be appropriate.

An appropriate design for a_X and a_F depends on what the robot is in contact with. When the robot is in contact with flat walls, like in example 17, these are easy to design. When the robot is in contact with objects that have complex shapes or changing contacts, the design gets more complicated.

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.

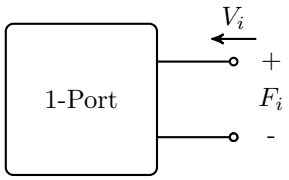
For example, when a robot pushes against a compliant surface, the contact experiences both non-zero normal reaction forces and non-zero motion, so that the work $\xi^T F$ is non-zero. If the stiffness of the surface is k, then the force is kx, and the total work done is

$$\begin{aligned} W &= \int_0^t \dot{x}(u)kx(u)du \\ &= \int_0^t \frac{d}{du}\frac{1}{2}kx(u)^2 du \\ &= \frac{1}{2}k(x(t)^2 - x(0)^2) \end{aligned} \tag{4.97}$$

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 a one port consisting of two interaction terminals and corresponding port variables **effort** F_i and **flow** V_i. The power consumed or dissipated by the node is $V_i^T F_i$. The relationship between these port variables depend on the dynamics of the system.



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

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

Electrical Circuit Analogy

The term impedance has an obvious analogue in electrical circuits. In fact, many concepts of mechanical impedance have names derived from electrical systems.

Electrical impedance may be thought of as a generalized form of electrical resistance, where we define this general electrical resistance (electrical impedance) as the ratio of the voltage produced across a component when we pass a current through it. Thus, the corresponding concept in mechanical systems is that the mechanical impedance is the force (voltage/effort) produced when we make an object move with a certain velocity (current/flow).

In Hogan's words

> Several important constraints on the behavior of physical systems can be identified. Along each degree of freedom, instantaneous power flow between two or more physical systems (e.g., a physical system and its environment) is always definable as the product of two conjugate variables, an effort (e.g., a force, a voltage) and a flow (e.g . a velocity, a current). An obvious but important physical constraint is that no one system may determine both variables. Along any degree of freedom a manipulator may impress a force on its environment or impose a displacement or velocity on it, but not both.
>
> Seen from the environment along any degree of freedom, physical systems come in only two types: admittances, which accept effort (e.g. force) inputs and yield flow (e.g. motion) outputs; and impedances, which accept flow (e.g., motion) inputs and yield effort (e.g., force) outputs. The concepts of impedance and admittance are familiar to designers of electrical systems as frequency-dependent generalizations of resistance or conductance and are usually regarded as equivalent and interchangeable representations of the same system. For a linear system operating at finite frequencies this is true, but manipulation is fundamentally a nonlinear problem, and for a nonlinear system it is not true; the two representations are in general not interchangeable.

Example 18 (Examples of Impedances)**.**

- Mass:
$$F(t) = M\dot{V}(t) \implies \hat{F}(s) = Ms\hat{V}(s) \implies Z_M(s) = Ms.$$
- Damper:
$$F(t) = BV(t) \implies \hat{F}(s) = B\hat{V}(s) \implies Z_B(s) = B.$$
- Spring:
$$F(t) = Kx(t) \implies \hat{F}(s) = \frac{K}{s}\hat{V}(s) \implies Z_K(s) = \frac{K}{s}.$$

□

(a) Thevenin equivalent circuit (b) Norton equivalent circuit

Figure 4.4: Equivalent one-port networks.

Example 19. For a S-M-D with dynamics $M\ddot{x} + B\dot{x} + Kx = F$, we have

$$Z(s) = Ms + B + \frac{K}{s}. \tag{4.98}$$

Classification of Impedance Operators

1. Inertial: iff $|Z(0)| = 0$
   Example: Mass, $Z(s) = Ms$.
2. Resistive: iff $0 < |Z(0)| < \infty$
   Example: Damper, $Z(s) = B$.
3. Capacitive: iff $|Z(0)| = \infty$
   Example: Spring, $Z(s) = K/s$.

Admittance

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

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

A physical system is an admittance if it can accept forces and produce negligible motion. A physical system is an impedance if it can accept motion and produce negligible forces. Hogan's main point is that a robot is often viewing the world as admittance, since it get's reaction forces from unmoving surfaces. Therefore, the robot needs to be controlled like an impedance. That is, we should seek to make it generate contact forces based upon the contact motion.

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. Figure 4.4 depicts these equivalent circuits. 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. Capacitive impedances are modeled as Thevenin equivalent circuits, and inertial impedances

4.5.3 Robot Impedance

The task space inverse dynamics control approach allows us to think of impedance of a robot in terms of (4.98), even though the relationship between velocity and force for the full model (4.23) is quite complex.

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

$$F(s) = Z(s)V(s). \tag{4.99}$$

Task Space Robot Impedance

Defining the impedance of a nonlinear system is difficult. Fortunately, we can focus on defining the impedance of the robot in the task space. Given a system

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B\dot{q} + G(q) = u + J^T(q)F_e, \tag{4.100}$$

we can impose the control

$$u = M(q)a_q + C(q,\dot{q})\dot{q} + B\dot{q} + G(q) - J^T(q)a_f, \tag{4.101}$$

to obtain the closed loop

$$\ddot{q} = a_q + M(q)^{-1}J^T(q)(F_e - a_f). \tag{4.102}$$

The joint acceleration $\ddot{q}$ will depend on our choices for a_q and a_f. In turn, the task acceleration is

$$\begin{aligned}
\ddot{X} &= J(q)\ddot{q} + \dot{J}(q)\dot{q} && (4.103)\\
&= J(q)\left(a_q + M(q)^{-1}J^T(q)(F_e - a_f)\right) + \dot{J}(q)\dot{q} && (4.104)\\
&= \underbrace{J(q)a_q + \dot{J}(q)\dot{q}}_{a_X} + \underbrace{J(q)M(q)^{-1}J^T(q)}_{W(q)}(F_e - a_f) && (4.105)
\end{aligned}$$

Let

$$\begin{aligned}
a_X &= J(q)a_q + \dot{J}(q)\dot{q} + J(q) && (4.106)\\
\implies a_q &= J_a(q)^{-1}\left(a_X - \dot{J}_a(q)\dot{q}\right), \text{ and} && (4.107)\\
W(q) &= J(q)M^{-1}(q)J^T(q). && (4.108)
\end{aligned}$$

The closed-loop task space dynamics are then

$$\ddot{X} = a_X + W(q)(F_e - a_f), \tag{4.109}$$

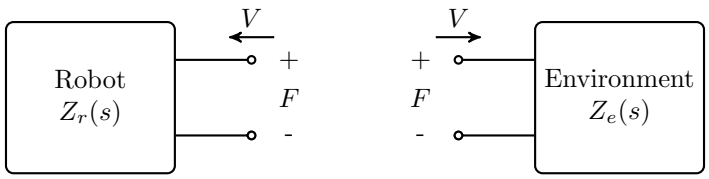
The term a_X is the desired acceleration in the task space X and $W(q)$ is the mobility tensor.

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

$$\ddot{X} = a_X. \tag{4.110}$$

4.5.4 Robot and Environment Interaction

The interaction between the robot and environment is modeled as a network of one-ports, where the connections between nodes occur at the interaction ports of the nodes, and flow and effort are transmitted between interacting nodes.



4.5.5 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 20 (Apparent Inertia)**.** Consider a mass M to which we apply a force F in order to move it around. The mass itself has an actuator that applies a force u, so that the full model is $M\ddot{x} = u - F$. When $u = 0$, the system behaves like

$$M\ddot{x} = -F.$$

We can control this system through a choice of u so that it appears to us (we are the ones applying a force F) as a lighter mass $m' < M$ by using the control $u = -mF$, $m > 0$. When we do so, we get

$$M\ddot{x} = u - F = -mF - F \tag{4.111}$$
$$\implies M\ddot{x} = -(1+m)F \tag{4.112}$$
$$\implies \frac{M}{1+m}\ddot{x} = -F \tag{4.113}$$
$$\implies m'\ddot{x} = -F \tag{4.114}$$

The new apparent mass is $m' = \frac{M}{1+m} < M$.

This kind of control is used in exoskeletons, where instead of forcing the limbs to move a specific way, we simply focus on reducing the apparent weight of the body. □

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

$$M_d\ddot{x} + B_d\dot{x} + K_d x = F. \tag{4.115}$$

We would need to implement a task space inverse dynamics control as described in Section 4.5.3 to achieve the closed-loop behavior in (4.110), and choose a_X as

$$a_X = M_d^{-1}(F - B_d\dot{x} - K_d x) \tag{4.116}$$

Our closed-loop model (4.115) is a spring-mass-damper model with desired parameters. When there is no external force ($F = 0$), clearly $x \to 0$ for $M_d, B_d, K_d > 0$.

The force F appears a bit artificial, and it is, because we assumed we are attempting to design a_X based on (4.110), which relies on knowing F_e so that we set $a_f = F_e$ in (4.109). If we remove the need to measure F_e, and just set $a_f = 0$ and $a_X = M_d^{-1}(-B_d\dot{x} - K_d x)$, we would instead obtain the closed-loop equation

$$M_d\ddot{x} + B_d\dot{x} + K_d x = M_d W(q) F_e \tag{4.117}$$

The value of admittance-based trajectory tracking is clearer now. We get a system where $x(t)$ acts like a spring-mass-damper acted upon by an unknown force, and its motion in response to these unknown forces can be tuned as we like. This idea is used a lot in **haptics**, where we want to tune M_d, B_d and K_d to produce the sensation of artificially interacting with a variety of objects, even though the actual haptic interface has almost no impedance when unpowered.

We can also define an impedance control relative to a trajectory $x_d(t)$, so that our impedance control achieves trajectory tracking when no disturbance forces exist.

$$a_X = +\ddot{x}_d(t) + M_d^{-1}\left(F(t) - B_d\left(\dot{x}(t) - \dot{x}_d(t)\right) - K_d\left(x(t) - x_d(t)\right)\right) \tag{4.118}$$

If we define $e(t) = x(t) - x_d(t)$, then the closed-loop error dynamics becomes

$$M_d\ddot{e}(t) + B_d\dot{e}(t) + K_d e(t) = F(t).$$

Again, when $F(t) = 0$, then $e(t) \to 0$, otherwise the error system behaves like an admittance, where the system error moves in reaction to $F(t)$. However, the standard choices of M_d, B_d and K_d in practice make this impedance-based trajectory tracking indistinguishable from task space inverse dynamics control with a linear controller in the outermost loop.

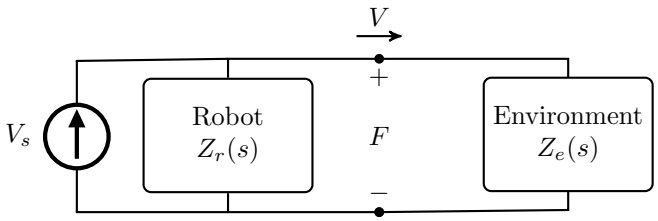
4.5.6 Hybrid Impedance Control

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 either 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.

In contrast, in position control, we required that the environment offer no force in response to the robot's motion, so that $Z_e(s) = 0$ for all s. For force control, we required that $Z_e(s) = \infty$ for all s. With impedance control, we now only require a condition on $Z_e(s)$ when $s = 0$, instead of forcing the system to be one way or another for all s.

Example 21 (Inertial Environment)**.**



We may equate the force at contact to obtain

$$Z_r(s)\left(V_s(s) - V(s)\right) = Z_e(s)V(s) \tag{4.119}$$

$$Z_r(s)V_s(s) - Z_r(s)V(s) = Z_e(s)V(s) \tag{4.120}$$

$$\frac{V(s)}{V_s(s)} = \frac{Z_r(s)}{Z_e(s) + Z_r(s)} \tag{4.121}$$

$$\frac{V(s) - V_s(s)}{V_s(s)} = -\frac{Z_e(s)}{Z_e(s) + Z_r(s)} \tag{4.122}$$

$$E_V(s) = -\frac{Z_e(s)}{Z_e(s) + Z_r(s)}V_s(s) \tag{4.123}$$

Under a step input $V_s(s) = V_d/s$, the steady state error will be zero if $Z_e(s) = 0$ and $Z_r(s) \neq 0$. Note that this happens even though there is some non-zero interaction force at the contact point between the robot and the environment!

A unit step for $V_s(s)$ corresponds to a linearly increasing desired position (a ramp). Therefore, if $Z_r(s)$ is non-inertial, and $Z_e(s)$ is inertial, we can achieve velocity tracking. Clearly, if $Z_e(s)$ was capacitive, the error would not go to zero.

To execute this observation on our manipulator, we need to choose a_X in (4.110) appropriately. To do so, notice that

$$\ddot{X} = a_X \implies sV(s) = a_X(s) \tag{4.124}$$

Our design effectively wants to enforce

$$F(s) = Z_r(s)(V_s(s) - V(s)),$$

where $Z_r(s)$ is non-inertial. Let $Z_r(s) = M_c s + Z_{rem}(s)$, where $Z_{rem}(s)$ is a proper rational transfer function. Then, we want

$$F(s) = M_c s\left(V_s(s) - V(s)\right) + Z_{rem}(s)\left(V_s(s) - V(s)\right) \tag{4.125}$$

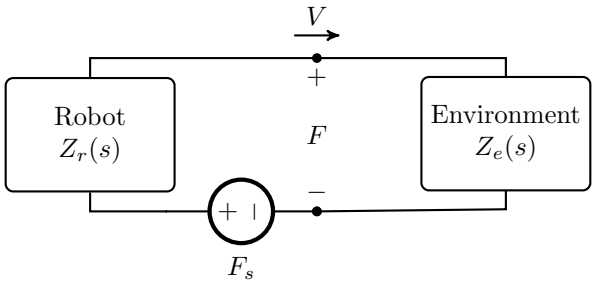
$$\implies sV(s) = sV_s(s) - \frac{F(s)}{M_c} + \frac{Z_{rem}(s)}{M_c}\left(V_s(s) - V(s)\right) \tag{4.126}$$

If we choose $Z_{rem}(s) = B + K/s$, so that it is proper and also makes $Z_r(s)$ capacitive, so that it is dual to $Z_e(s)$ (which is inertial), then the equation above implies

$$a_X(t) = \ddot{x}_d(t) - \frac{F}{M_c} + \frac{B}{M_c}\left(\dot{x}_d(t) - \dot{x}(t)\right) + \frac{K}{M_c}\left(x_d(t) - x(t)\right), \tag{4.127}$$

which reduces to impedance-based trajectory tracking (4.118) along a single direction. □

Example 22 (Capacitive Environment)**.**



We may equate the velocity at contact to obtain

$$\frac{F_s(s) - F(s)}{Z_r(s)} = \frac{F(s)}{Z_e(s)} \tag{4.128}$$

$$Z_e(s)F_s(s) - Z_e(s)F(s) = Z_r(s)F(s) \tag{4.129}$$

$$Z_e(s)F_s(s) = \left(Z_e(s) + Z_r(s)\right)F(s) \tag{4.130}$$

$$\frac{F(s)}{F_s(s)} = \frac{Z_e(s)}{Z_e(s) + Z_r(s)} \tag{4.131}$$

$$\frac{F(s) - F_s(s)}{F_s(s)} = -\frac{Z_r(s)}{Z_e(s) + Z_r(s)} \tag{4.132}$$

$$E_F(s) = -\frac{Z_r(s)}{Z_e(s) + Z_r(s)}F_s(s) \tag{4.133}$$

We can see that for step input force $F_s(s) = F_d/s$, we have that the error in the interaction force will become zero when $Z_r(0)$ is finite. Furthermore, this happens even though there may be non-zero motion of the contact point between the robot and the environment!

To implement the control, note that at the contact we need

$$F_s(s) - F(s) = Z_r(s)V(s).$$

Again, let $Z_r(s) = Ms + Z_{rem}(s)$. Then

$$F_s(s) - F(s) = MsV(s) + Z_{rem}(s)V(s) \tag{4.134}$$

$$sV(s) = \frac{F_s(s) - F(s)}{M} - \frac{Z_{rem}(s)}{M}V(s) \tag{4.135}$$

For $Z_r(s)$ to be inertial and non-capacitive, we want $Z_{rem}(s) = B$. Then,

$$a_X = M^{-1}\left(F_d(t) - F(t) - B\dot{x}\right),$$

which is force control in the task space with damping. □

These single degree-of-freedom designs lead to a hybrid impedance controller where the robot applies different control strategies along different directions in the contact frame based on the environment's impedance along those directions. This idea is a generalization of hybrid force/position control.

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).

4.7 Summary

Topic	Equation
Euler-Lagrange Eqns	$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau$
Euler-Lagrange Eqns + Actuator	$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + B\dot{q} = u$
Euler-Lagrange Eqns + Force	$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u + J^T F$
Euler-Lagrange Eqns + Actuator + Force	$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + B\dot{q} = u + J^T F$
Euler-Lagrange Eqns + Actuator + Joint elasticity	

Controller	Equations	Notes
Joint Space PD	$u = -K_p\left(q - q_d(t)\right) - K_d\left(\dot{q} - \dot{q}_d(t)\right)$	Great for high-gear ratio motors, $q_d(t)$ constant
Joint Space PD+	$u = G(q) - K_p\left(q - q_d(t)\right) - K_d\left(\dot{q} - \dot{q}_d(t)\right)$	Reduces gains required for $q_d(t)$ constant
Joint Space CTC	$u = M(q)a_q + C(q,\dot{q})\dot{q} + G(q)$ $a_q = \ddot{q}_d(t) - K_p\left(q - q_d(t)\right) - K_d\left(\dot{q} - \dot{q}_d(t)\right)$	Inner-outer loop design
Task Space CTC	$u = M(q)a_q + C(q,\dot{q})\dot{q} + G(q)$ $a_q = J^+\left(a_X - \dot{J}\dot{q}\right)$ $a_X = \ddot{x}_d(t) - K_p\left(x - x_d(t)\right) - K_d\left(\dot{x} - \dot{x}_d(t)\right)$	Tracking requires no contact force
Direct Force Control	$u = G(q) - J^T F_d - J^T\left(K_p F_{err} + K_i \int F_{err} dy\right)$	Tracking requires no contact motion
Config-based Force Control	$u = G(q^*) - J^T(q^*)F_d - K_p\left(q - q^*\right) - K_d\dot{q}$	Avoids measuring F_{tip}
Hybrid Force/Position Control	$u = M(q)a_q + C(q,\dot{q})\dot{q} + G(q) - J^T a_F$ $a_q = J^+\left(a_X - \dot{J}\dot{q}\right)$ $a_X = \ddot{x}_d(t) - K_p\left(x - x_d(t)\right) - K_d\left(\dot{x} - \dot{x}_d(t)\right)$ $a_F = F_d + K_p F_{err} + K_i \int F_{err} dy$	a_X and a_F must keep position and force directions orthogonal
Impedance Control		
Hybrid Impedance Control		
Robust Control		
Adaptive Control		
Passivity-based Control		
Robust PBC		
Adaptive PBC		

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:

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

$$\text{subject toRobot doesn't destroy itself or things} \tag{5.2}$$

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

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.

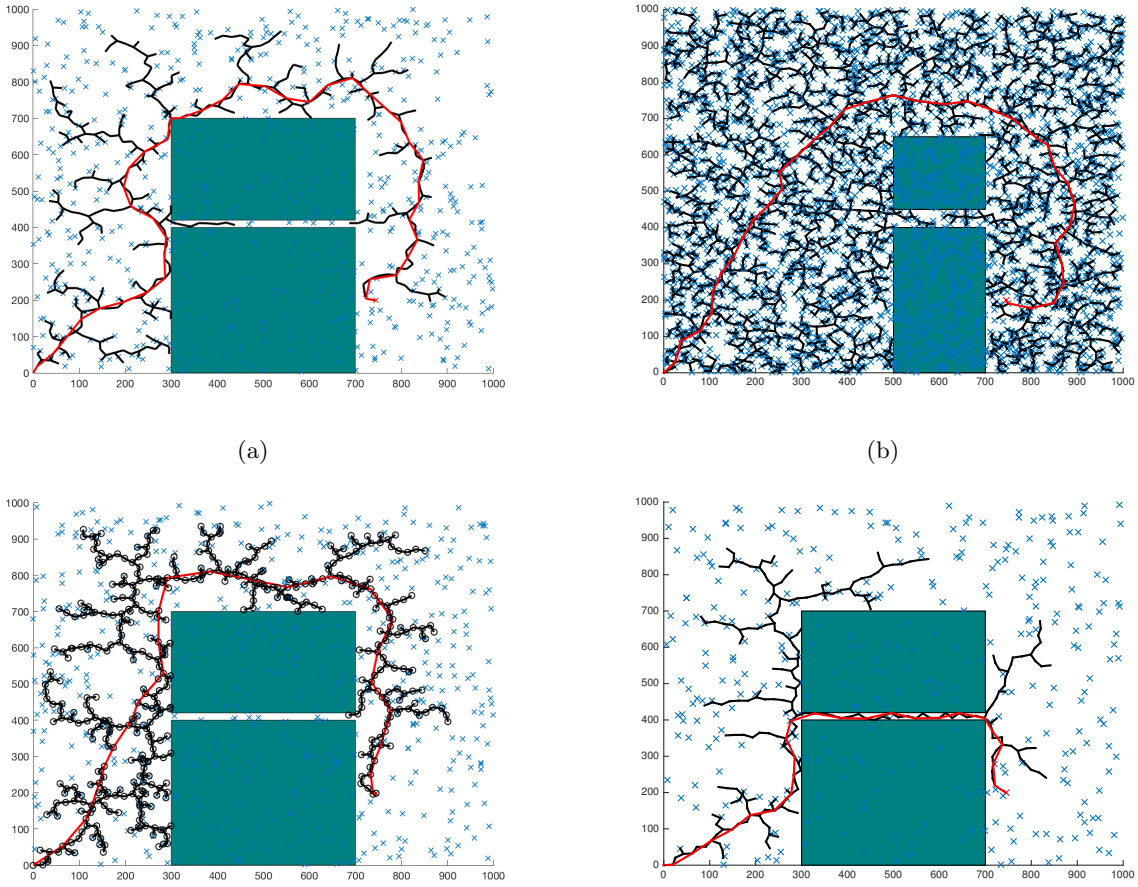
In RRT or RRT*with dynamics, the *steer* function finds an (optimal) trajectory between y and v, where again y is closest possible to z, but only so far from v. The trajectory returned by the *steer* function confirms that we can connect y to v. Again, there may be an obstacle in the way, and *collision 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.

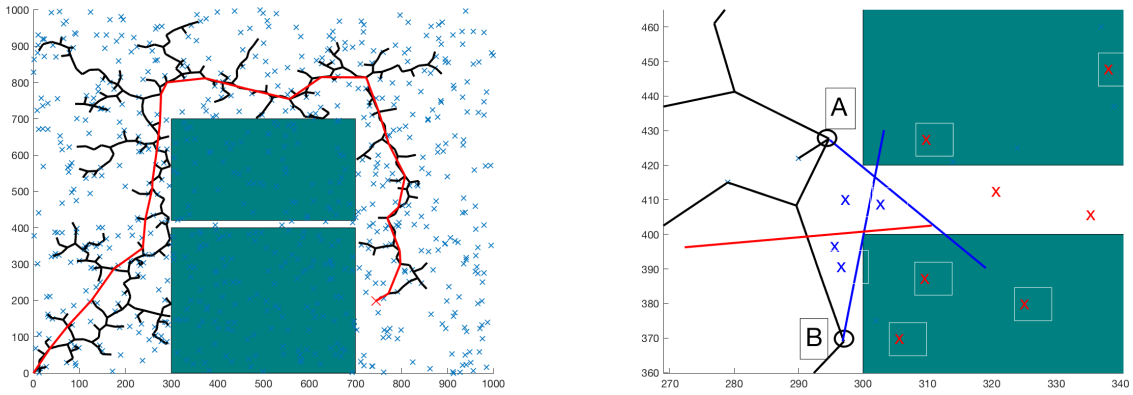
Figure 5.1 shows the tree without rewiring (in black), with the optimal path in red. The initial state is the origin in the lower left corner, the goal state is the red check mark. Figure 5.2 shows a branch of the tree that successfully passed through the narrower gap.



(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.

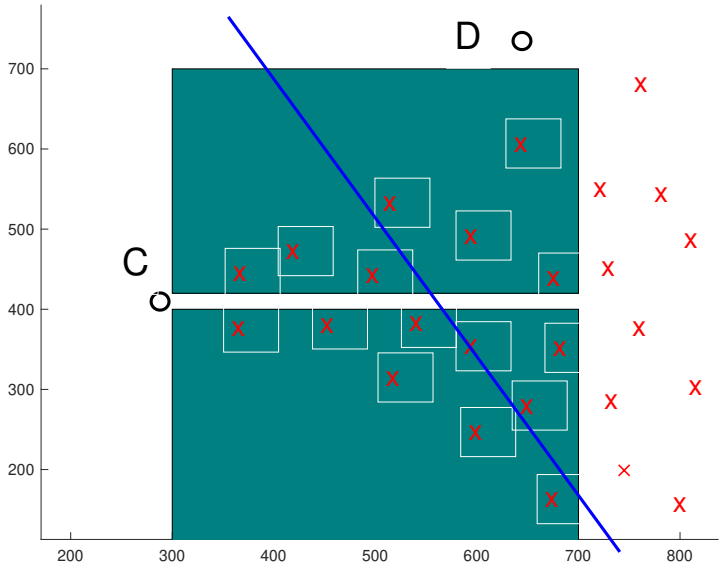


Figure 5.3: Point C, in the absence of point D, will be extended through the gap by samples in a large region, potentially represented by the red check marks. Note that if a one of the blue check marks closer to point A in Figure 5.2b were sampled, they would become points similar to point C here. If 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.

5.5.3 Cubic Splines in Hermite Form

5.5.4 Bezier Splines

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 23. Real numbers form a group under addition.

Example 24. 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 25. 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 26. 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 27. 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 28. 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 29. 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.

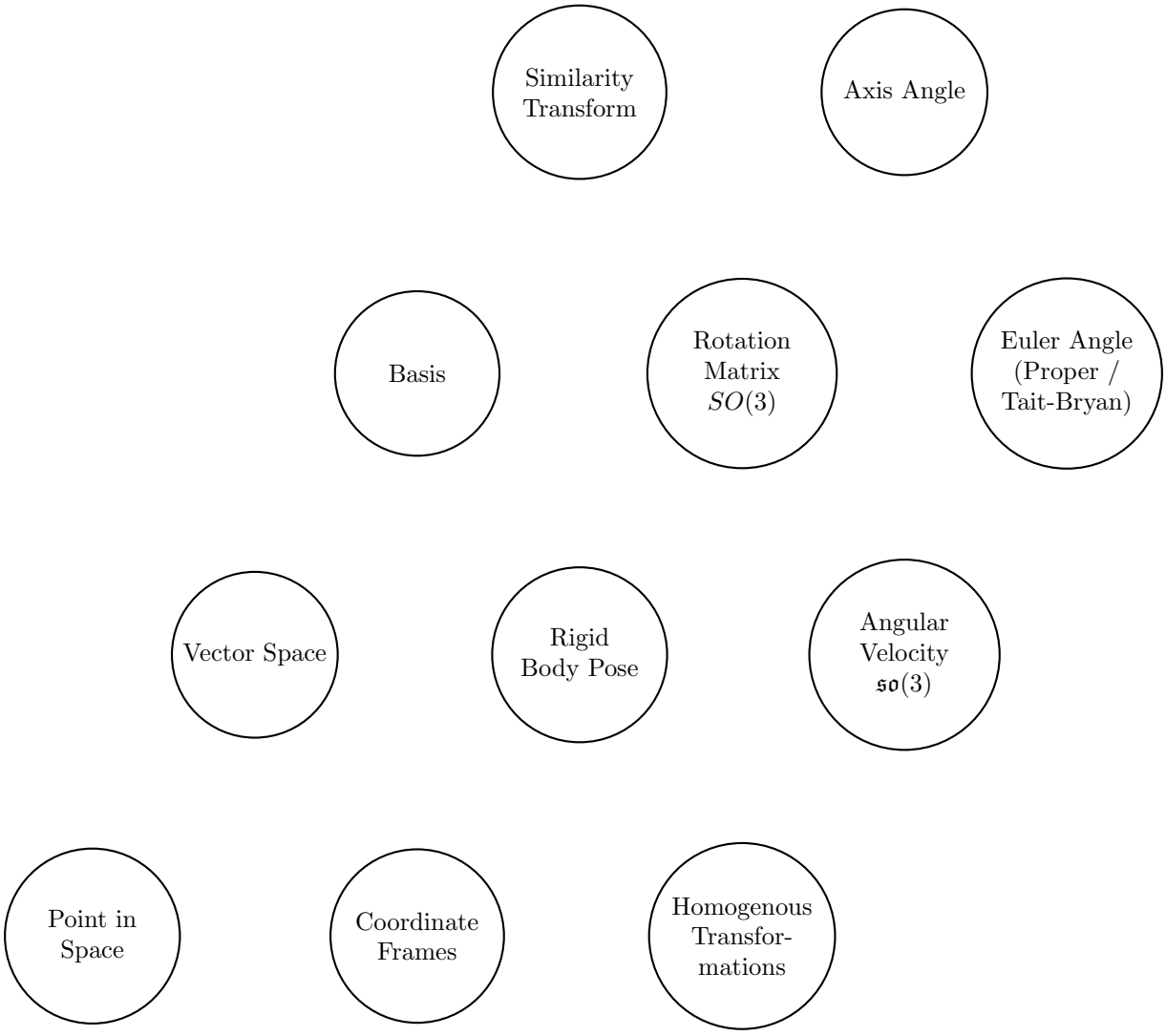


Figure A.1: Topics as nodes in a graph.

Appendix B

Linear Algebra

B.1 Matrix-Vector Products

A linear map $T: V \to W$ applied to a vector $v \in V$ is usually depicted as a matrix of numbers multiplied by a column of numbers. These quantities are representations of the matrix and vector respectively given bases for V and W.

Let's consider $W = V = \mathbb{R}^3$. The matrix-vector product is then

$$T(v) = \begin{bmatrix} a & b & c \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} d \\ e \\ f \end{bmatrix}. \tag{B.1}$$

The main point is that we are typically taught to compute this row-wise, so that we compute

$$\text{(row-wise:)} \qquad T(v) = \begin{bmatrix} ad + be + cf \\ \\ \\ \end{bmatrix} \to \begin{bmatrix} ad + be + cf \\ *d + *e + *f \\ \\ \end{bmatrix} \to \begin{bmatrix} ad + be + cf \\ *d + *e + *f \\ *d + *e + *f \end{bmatrix} \tag{B.2}$$

This approach is effective for computing, but not for understanding. As Gilbert Strang suggests, conceptually, we should think of this linear mapping of a vector by a matrix using a column-wise approach:

$$\text{(column-wise:)} \qquad T(v) = d \begin{bmatrix} a \\ * \\ * \end{bmatrix} + e \begin{bmatrix} b \\ * \\ * \end{bmatrix} + f \begin{bmatrix} c \\ * \\ * \end{bmatrix} \tag{B.3}$$

Effectively, the vector coefficients define a linear combination of the columns of the matrix. As some like to suggest, the best way to think about something may be different than the best way to compute it.

Appendix C

Analysis

C.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.

C.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 30. 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).

C.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 31. Returning to Example 30, 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 32 (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 D

Dynamical Systems & Control

D.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:

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

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:

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

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

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

which is again an autonomous system.

D.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 $\bar{t} \in [t_1, t_2]$

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

D.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 $\delta > 0$

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

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 $\delta > 0$

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

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

D.2 Linear Dynamical Systems

Consider the linear dynamical system

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

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

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

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

D.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:

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

where

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

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.

D.2.2 Controllability and Observability

Given the linear time-invariant system

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

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

, 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.

D.2.3 Controllability

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

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

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.

D.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. Once a system is observable, we may design state estimators that will use y, u, and a model of the system to estimate the full state x, where the estimate is often denoted as $\hat{x}$. For an observable LTI system with perfect knowledge of the model, we can achieve $\hat{x}(t) \to x(t)$. In other words, despite not measuring the whole state, we can eventually know what it is.

The practical application is that sensors for some physical quantities are easier to build than for others, and we may get by without sensing those hard-to-measure quantities if the easy-to-measure variables result in an observable system.

Bibliography

[1] V. H. Garcia-Rodriguez, J. R. Garcia-Sanchez, R. Silva-Ortigoza, E. Hernandez-Marquez, H. Taud, M. Ponce-Silva, and G. Saldana-Gonzalez. Passivity based control for the boost converter-inverter-dc motor system. In *2017 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE)*, pages 77–81, 2017.

[2] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, USA, 1st edition, 2017.

[3] Emmanuel Nuno, Luis Basanez, and Romeo Ortega. Passivity-based control for bilateral teleoperation: A tutorial. *Automatica*, 47(3):485 – 495, 2011.

[4] Mark W. Spong. Passivity based control of the compass gait biped. *IFAC Proceedings Volumes*, 32(2):506 – 510, 1999. 14th IFAC World Congress 1999, Beijing, Chia, 5-9 July.

[5] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot Modeling and Control*. Wiley and Sons, New York, USA, 2006.