RBOT101: Mathematical Foundations of Robotics

Instructor: Dr. Lekan Molu

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

PREAMBLE

Consider this the roadmap for this course. Please read through the syllabus posted on Moodle2 carefully and feel free to share any questions that you may have. Please print a copy of the Syllabus for reference. Some relevant parts of the Syllabus are repeated here but the Moodles reference should serve as your guide throughout the ten weeks of this course.

1.1 Course Description

This course focuses on the algorithmic and mathematical concepts with respect classical and recent methods for solving real-world problems in robotics. While some students may have encountered some of the concepts we will be treating in past courses or avenues of study, we will provide the breadth and depth necessary for equipping students to be world-class roboticists. The topics covered by this course shall include the configuration space, rigid bodies, semi-rigid soft bodies, as well as their motions in $\mathbb{R}^n$, wrenches, homogeneous transformations, optimal algorithms for rigid body rotations, linear systems theory, probability theory, the Kalman filter. The course will begin and end with a self-assessment to allow students to gauge their strengths and weaknesses in these topics. References for further, in-depth study in each topic are provided at the end of this course.

1.2 Course Outcomes

After taking this course, each student will be able to

- Develop mathematical tools for solving fundamental kinematic problems in robot operation;
- Formulate optimal state estimation tools for solving real-time smoothing and filtering operations in robotics;
- Integrate state estimation with rigid and semi-rigid soft bodies to solve real-world automation problems; and 4. Use open-source Python, and C++ tools to solve classical and emerging problems in robotics in our day.

1.3 Prerequisites

An undergraduate-level understanding of linear algebra, analytical mechanics, Python and C++ programming.

1.4 Recommended Texts

- Main Texts

- Simon, Dan. (2007). Optimal state estimation: Kalman, $H - \infty$, and nonlinear approaches. Choice Reviews Online, Vol. 44, pp. 44-3334-44–3334. https://doi.org/10.5860/choice.44-3334
- Murray, R. M., Li, Z., and Sastry, S. S. (1994). A Mathematical Introduction to Robotic Manipulation. Book (Vol. 29). Free PDF preprint downloadable from, Murray's website.
- Theory of Screws: A Study in the Dynamics of a Rigid Body by Robert Stawell Ball, Dublin: Hodges, Foster, and Co., Grafton-Street. a. Textbooks:

- Secondary Text

  - Modern Robotics: Mechanics, Planning, and Control. Free PDF preprint downloadable from Author's Northwestern University Website.

- Auxiliary Text:

  - Theory of Screws: A Study in the Dynamics of a Rigid Body by Robert Stawell Ball, Dublin: Hodges, Foster, and Co., Grafton-Street (Should be downloadable via Interlibrary Loan).

1.5 Recommended Journals

- IEEE Transactions on Robotics.
- The International Journal of Robotics Research.
- The IEEE International Conference on Robotics and Automation (ICRA).
- IEEE/Robotics Society of Japan International Conference on Intelligent Robots and Systems (IROS).
- Robotics and Autonomous Systems, An Elsevier Journal.

1.6 Required Software

- A working knowledge of python and the anaconda environment.
- ROS 1.x Installation Instructions: ros 1.x website.
- ROS 2 installation ros 2.0 website.

1.7 Online Course Content

This course will be conducted completely online using Brandeis' LATTE site. The site contains the course syllabus, assignments, our discussion forums, links/resources to course-related professional organizations and sites, and weekly checklists, objectives, outcomes, topic notes, self-tests, and discussion questions. Access information is emailed to enrolled participants before the start of the course. To begin participating in the course, review the "Welcoming Message" and the "Week 1 Checklist."

1.8 Errata

If in the course of using these notes, you find sentence errors, errata or mistakes in equations, please annotate them and upload it to the discussion forum. Points will awarded, at the discretion of the instructor, for such help.

CHAPTER 2

INTRODUCTION TO MATRIX ANALYSIS.

Our goal here is to introduce the student to the study of matrix theory. Matrices are symbolism of the important transformations in everyday life; these transformations lie at the heart of mathematics and robotics. The contents of this topic are thus positioned toward the aspiration of roboticists, engineers of all stripes and scientists. Specifically, we are concerned with the *theory of symmetric matrices*, which is important for all fields, *matrices and differential equations*, necessary for engineering and robotics, as well as *positive matrices*, necessary for probability theory. Most of the texts im this chapter are drawn from Richard Bellman's Matrix Analysis Book given in the Syllabus.

2.1 Maximization and Minimization

Of importance to us in this section is to ascertain the range of values of *homogeneous quadratic functions* of two variables and how it is connected to the determination of the maximum or minimum of a general function of two variables.

2.1.1 Maximization of Functions of a Variable

Suppose $f(x)$ is a real function of the real variable x for $x \in [a, b]$, and let us suppose that it is a Taylor series of the form

$$f(x) = f(c) + f'(x - c) + f'' \frac{(x - c)^2}{2!} + \dots \tag{2.1.1}$$

around every point in the open interval (a, b). We define a *stationary point* of $f(x)$ to be a point where $f'(x) = 0$ and it is the point that determines if c is a point at which $f(x)$ is a relative maximum, a relative minimum, or a stationary point of a subtle characteristic. If c is a stationary point, we must have

$$f(x) = f(c) + f'' \frac{(x - c)^2}{2!} + \dots \tag{2.1.2}$$

If $f''(c) > 0$, then $f(x)$ has a relative minimum at $x = c$. Otherwise, if $f''(c) < 0$, $f(x)$ has a relative maximum at $x = c$. Whereas, if $f''(c) = 0$, we must needs consider further terms in the expansion.

Quiz 1. Suppose that $f''(c) = 0$, what are the sufficient conditions that c must furnish to be a relative minimum?

2.1.2 Maximization of Functions of Two Variables

Now, suppose that we have two variables x, y as arguments of a function f, defined over the rectangle $a_1 \leq x \leq b_1$, $a_2 \leq y \leq b_2$, and possessing a convergent Taylor series around each point

(c_1, c_2) within the region. Then, for sufficiently small $|x - c_1$ and $|y - c_2|$, we have

$$f(x,y) = f(c_1,c_2) + (x-c_1)\frac{\partial f}{\partial c_1} + (y-c_2)\frac{\partial f}{\partial c_2} + \frac{(x-c_1)^2}{2}\frac{\partial^2 f}{\partial c_1^2} + (x-c_1)(y-c_2)\frac{\partial^2 f}{\partial c_1 \partial c_2} + \frac{(y-c_2)^2}{2}\frac{\partial^2 f}{\partial c_2^2} + \dots \tag{2.1.3}$$

where

$$\begin{aligned} \frac{\partial f}{\partial c_1} &= \frac{\partial f}{\partial x} \text{ at } x = c_1, \qquad y = c_2 \\ \frac{\partial f}{\partial c_2} &= \frac{\partial f}{\partial y} \text{ at } x = c_1, \qquad y = c_2 \text{ e.t.c.} \end{aligned} \tag{2.1.4}$$

As before, the stationary point of $f(x, y)$ is defined to be (c_1, c_2) so that $\frac{\partial f}{\partial c_1} = 0$ and $\frac{\partial f}{\partial c_2} = 0$; and the behavior of $f(x, y)$ in the immediate neighborhood of (c_1, c_2) depends on the nature of the quadratic terms in the expansion of (2.1.3),

$$Q_2(x,y) = a(x-c_1)^2 + 2b(x-c_1)(y-c_2) + c(y-c_2)^2 \tag{2.1.5}$$

where $a = \frac{1}{2}\frac{\partial^2 f}{\partial c_1^2}$, $2b = \frac{\partial^2 f}{\partial c_1 \partial c_2}$, and $c = \frac{1}{2}\frac{\partial^2 f}{\partial c_2^2}$.

Suppose we set $x - c_1 = u$ and $y - c_2 = v$, then we can write a quadratic expression in variables u and v $i.e.$

$$Q(u,v) = au^2 + 2buv + cv^2 \tag{2.1.6}$$

whereupon we are interested in the behavior of $Q(u, v)$ in the vicinity of $u = v = 0$ and the fact that $Q(u, v)$ is homogeneous allows us to examine the range of values of $Q(u, v)$ for the set of values on $u^2 + v^2 = 1$.

If $Q(u, v) > 0$ for all u and v distinct from $u = v = 0$, $f(x, y)$ will have a relative minimum at $x = c_1$, $y = c_2$; and if $Q(u, v) < 0$ for all u and v distinct from $u = v = 0$, $f(x, y)$ will have a relative maximum at $x = c_1$, $y = c_2$; The stationary point is a *saddle point* if $Q(u, v)$ can take on both positive and negative values.

2.1.3 Algebraic Approach

How do we determine which of the three situations described in the foregoing occur for any given quadratic form, $au^2 + 2buv + cv^2$, with real coefficients. To determine the sign of $Q(u, v)$, we complete the square in $au^2 + 2buv$ and write $Q(u, v)$ as

$$Q(u,v) = a\left(u + \frac{bv}{a}\right)^2 + \left(c - \frac{b^2}{a}\right)v^2 \tag{2.1.7}$$

provided that $a \neq 0$.

If $a = c = -$, then $Q(u, v) \equiv 2buv$. If $b \neq 0$, then $Q(u, v)$ can be positive or negative. If however, $b = 0$, the quadratic form is eliminated.

If $a \neq 0$, from (2.1.7), we must have a $Q(u, v) > 0$ for all unique u and v different from the pair $(0, 0)$ provided that $a > 0$ and $c - \frac{b^2}{a} > 0$.

In the same vein, $Q(u, v) < 0$ for all nontrivial u and v, provided that we have the inequalities, $a < 0$ and $c - \frac{b^2}{a} < 0$.

Positivity Requirement

A set of *necessary and sufficient* conditions that $Q(u, v)$ be positive for all nontrivial u and v is that

$$a > 0, \qquad \begin{vmatrix} a & b \\ b & c \end{vmatrix} > 0. \tag{2.1.8}$$

2.1.4 Analytic Approach

To find the range of values of $Q(u, v)$, we can examine the set of values that $Q(u, v)$ occupies on the circle $u^2 + v^2 = 1$. If Q is to be positive for all nontrivial values of u and v, we must have

$$\min_{u^2+v^2=1} Q(u, v) > 0 \tag{2.1.9}$$

and to have $Q(u, v)$ negative for all u and v on the unit circle, we must have

$$\max_{u^2+v^2=1} Q(u, v) < 0. \tag{2.1.10}$$

Introducing a Lagrange multiplier, λ, we can rewrite the problem as

$$R(u, v) = aU^2 + 2buv + cv^2 - \lambda(u^2 + v^2). \tag{2.1.11}$$

At the stationary points, we must have $\frac{\partial R}{\partial u} = \frac{\partial R}{\partial v} = 0$ so that

$$\begin{aligned} au + bv - \lambda u &= 0 \\ bu + cv - \lambda v &= 0 \end{aligned} \tag{2.1.12}$$

whereupon, we see that λ satisfies

$$\begin{vmatrix} a - \lambda & b \\ b & c - \lambda \end{vmatrix} = 0 \tag{2.1.13}$$

$$\lambda^2 - (a + c)\lambda + ac - b^2 = 0. \tag{2.1.14}$$

The roots of (2.1.14) are real seeing that the discriminant is non-negative $i.e.$

$$(a + c)^2 - 4\left(ac - b^2\right) = (a - c)^2 + 4b^2, \tag{2.1.15}$$

and as long as $a \neq 0$ and $b \neq 0$, the roots are distinct.

If $b = 0$, the roots of the quadratic in (2.1.14) becomes $\lambda_1 = a$, $\lambda_2 = c$. For $\lambda_1 = a$, the linear set of equations from (2.1.12) becomes

$$(a - \lambda_1)\, u = 0 \qquad (c - \lambda_1)\, v = 0 \tag{2.1.16}$$

which leaves u arbitrary and $v = 0$, if $a \neq c$.

Whereas if $b \neq 0$, we obtain the nontrivial solutions of (2.1.12) by using one equation and discarding the other. Therefore, u and v are connected by the relation

$$(a - \lambda_1)\, u = -bv. \tag{2.1.17}$$

For the exact solution, we can add the normalization requirement that $u^2 + v^2 = 1$ so that the values of u and v are

$$\begin{aligned} u_1 &= -b/\left(b^2 + (a - \lambda_1)^2\right)^{1/2} \\ v_1 &= (a - \lambda_1)/\left(b^2 + (a - \lambda_1)^2\right)^{1/2} \end{aligned} \tag{2.1.18}$$

with another set (u_2, v_2) determined in a similar fashion when λ_2 is used in place of λ_1.

CHAPTER 3

VECTORS AND MATRICES

In the previous chapter, we looked into the problem of the minima and maxima (locally) of a function of a single and two variables. Suppose that we have N variables, and proceed in a similar manner as before, we see that finding basic necessary and sufficient conditions that ensure the positivity of a quadratic form of N variables are of the form

$$Q(x_1, x_2, \ldots, x_N) = \sum_{i,j=1}^{N} a_{ij} x_i x_j \tag{3.0.1}$$

We will thus develop a notation that allows us to solve the problem *analytically* using a minimum of arithmetic or analytical calculation. In this light, we will develop a notation that allows us to study linear transformations such as

$$y_i = \sum_{j=1}^{N} a_{ij} x_j \qquad i = 1, 2, \ldots, N \tag{3.0.2}$$

3.1 Vectors

We shall define a set of N complex-valued numbers as a *vector*, written as

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \tag{3.1.1}$$

The vector $\boldsymbol{x}$ in (3.1.1) shall be called a *column vector*. If the elements of the vector are stacked horizontally, $i.e.$

$$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \ldots & x_N, \end{bmatrix} \tag{3.1.2}$$

then we shall call it a *row vector*.

Going forward, we shall use the notation of (3.1.1) to represent all forms of vectors we shall be using. When we mean a row vector, we shall use the notation of a transpose of (3.1.1), $i.e.$ $\boldsymbol{x}^T$. Bold font letters such as $\boldsymbol{x}$, or $\boldsymbol{y}$ shall denote vectors and lower-case letters with subscripts i such as x_i, y_i, z_i or p_i, q_i, r_i shall denote the components of a vector. When discussing a particular set of vectors, we shall use the superscripts $\boldsymbol{x}^1, \boldsymbol{x}^2$ e.t.c. N shall denote the dimension of a vector $\boldsymbol{x}$.

One-dimensional vectors are called *scalars* and shall be our quantities of analysis. When we write $\bar{\boldsymbol{x}}$, we shall mean the vector whose components are the complex conjugates of the elements of $\boldsymbol{x}$.

3.1.1 Addition of Vectors

Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are said to be equal if all of their components, (x_i, y_i) are equal for $i = 1, 2, \ldots, N$. Addition is the simplest of the arithmetic operations on vectors. We shall write the sum of two vectors as $\boldsymbol{x} + \boldsymbol{y}$ so that

$$\boldsymbol{x} + \boldsymbol{y} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_N + y_N, \end{bmatrix} \tag{3.1.3}$$

whereupon we note that the " $+$ " sign connecting $\boldsymbol{x}$ and $\boldsymbol{y}$ is different from the one connecting x_i and y_i.

Homework 1. Prove that we have the *commutativity*, $\boldsymbol{x} + \boldsymbol{y} = \boldsymbol{y} + \boldsymbol{x}$, and the associativity $\boldsymbol{x} + (\boldsymbol{y} + \boldsymbol{z}) = (\boldsymbol{x} + \boldsymbol{y}) + \boldsymbol{z}$

Homework 2. Just as we showed the addition property of two vectors above, show the subtraction property of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

3.1.2 Scalar Multiplication

When a vector is multiplied by a scalar, we shall write it out as follows

$$c_1\boldsymbol{x} = \boldsymbol{x}c_1 = \begin{bmatrix} c_1x_1 \\ c_1x_2 \\ \vdots \\ c_1x_N \end{bmatrix} \tag{3.1.4}$$

3.1.3 The Inner Product of Two Vectors

This is a scalar function of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{N} x_i y_i. \tag{3.1.5}$$

Further to the above, we define the following properties for inner product

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle \tag{3.1.6a}$$

$$\langle \boldsymbol{x} + \boldsymbol{y}, \boldsymbol{u} + \boldsymbol{v} \rangle = \langle \boldsymbol{x}, \boldsymbol{u} \rangle + \langle \boldsymbol{x}, \boldsymbol{v} \rangle + \langle \boldsymbol{y}, \boldsymbol{u} \rangle + \langle \boldsymbol{y}, \boldsymbol{v} \rangle \tag{3.1.6b}$$

$$\langle c_1\boldsymbol{x}, \boldsymbol{y} \rangle = c_1 \langle \boldsymbol{x}, \boldsymbol{y} \rangle \tag{3.1.6c}$$

The above is an easy way to *multiply* two vectors. The inner product is important because $\langle \boldsymbol{x}, \boldsymbol{x} \rangle$ can be considered as the square of the "length" of the real vector $\boldsymbol{x}$.

Homework 3. Prove that $\langle a\boldsymbol{x}+b\boldsymbol{y}, a\boldsymbol{x}+b\boldsymbol{y}\rangle = a^2\langle \boldsymbol{x},\boldsymbol{x}\rangle + 2ab\langle \boldsymbol{x},\boldsymbol{x}\rangle + b^2\langle \boldsymbol{y},\boldsymbol{y}\rangle$ is a non-negative quadratic form in the scalar variables a and b if $\boldsymbol{x}$ and $\boldsymbol{y}$ are real.

Homework 4. Hence, show that for real-valued vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, that the Cauchy-Schwarz Inequality $\langle \boldsymbol{x},\boldsymbol{y}\rangle^2 \leq \langle \boldsymbol{x},\boldsymbol{x}\rangle\langle \boldsymbol{y},\boldsymbol{y}\rangle$ holds.

Homework 5. Using the above result, show that for any two complex vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, $|\langle \boldsymbol{x},\boldsymbol{y}\rangle|^2 \leq \langle \boldsymbol{x},\bar{\boldsymbol{x}}\rangle\,\langle \boldsymbol{y},\bar{\boldsymbol{y}}\rangle$

Homework 6. Show that the *triangle inequality*

$$\langle \boldsymbol{x}+\boldsymbol{y}, \boldsymbol{x}+\boldsymbol{y}\rangle^{\frac{1}{2}} \leq \langle \boldsymbol{x},\boldsymbol{x}\rangle^{\frac{1}{2}} + \langle \boldsymbol{y},\boldsymbol{y}\rangle^{\frac{1}{2}}$$

holds for any two real-valued variables.

3.1.4 Orthogonality

Two vectors are said to be orthogonal if their inner product is 0 $i.e.$

$$\langle \boldsymbol{x},\boldsymbol{y}\rangle = 0 \tag{3.1.7}$$

When the set of real vectors $\{\boldsymbol{x}^i\}$ possess the property that $\langle \boldsymbol{x}^i,\boldsymbol{y}^i\rangle = 1$, then we say they are *orthonormal.*

Homework 7. show that $\boldsymbol{x}^i$ are mutually orthogonal and normalized $i.e.$ orthonormal for the following N-dimensional Euclidean basis coordinate vectors

$$\boldsymbol{x}^1 = \begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} \qquad \boldsymbol{x}^2 = \begin{bmatrix}0\\1\\\vdots\\0\end{bmatrix} \qquad \boldsymbol{x}^N = \begin{bmatrix}0\\0\\\vdots\\1\end{bmatrix} \tag{3.1.8}$$

3.2 Matrices

We can write an array of complex numbers in the form

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1N} \\ x_{21} & x_{22} & \dots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{NN} \end{bmatrix} \tag{3.2.1}$$

The matrix of (3.2.1) shall be called a *square matrix.* The quantities x_{ij} are the *elements* of the matrix X; the quantities $x_{i1}, x_{i2}, \dots, x_{iN}$ are the ith *rows* of the matrix X and the quantities

$x_{1j}, x_{2j}, \ldots, x_{Nj}$ are the jth columns of X. We denote matrices with upper case letters or the lower-case subscript notations

$$X = (x_{ij}) \tag{3.2.2}$$

while the *determinant* of the array associated with (3.2.1) shall be denoted $|X|$ or $|x_{ij}$.

Similar to the equality definition between vectors, two matrices are said to be equal if and only if their elements are equal $i.e.$

$$A + B = (a_{ij} + b_{ij}) \tag{3.2.3}$$

Scalar multiplication of a matrix can be expressed as

$$c_1 X = X c_1 = (c_1 x_{ij}) \tag{3.2.4}$$

Lastly, by $\bar{X}$ we shall mean the matrix whose elements are the complex conjugates of X. X is a real matrix if the elements of X are real.

3.2.1 Vector by Matrix Multiplication

Recall the linear transformation

$$y_i = \sum_{j=1}^{N} a_{ij} x_j \qquad i = 1, 2, \ldots, N \tag{3.2.5}$$

where a_{ij} are complex quantities. For two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ related as above, we have

$$\boldsymbol{y} = A\boldsymbol{x} \tag{3.2.6}$$

to describe the multiplication of a vector $\boldsymbol{x}$ by a matrix X.

Homework 8. Consider the identity matrix I, so defined

$$I = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix} \tag{3.2.7}$$

$i.e.$ $I = (\delta_{ij})$, where δ_{ij} is the Kronecker delta symbol, defined as

$$\delta_{ij} = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{if } i = j \end{cases} \tag{3.2.8}$$

Show that

$$\delta_{ij} = \sum_{k=1}^{N} \delta_{ik} \delta_{kj} \tag{3.2.9}$$

Homework 9. Show that

$$\langle Ax, Ax \rangle = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} a_{ij} x_j \right)^2 \tag{3.2.10}$$

3.2.2 Matrix by Matrix Multiplication

Consider (3.2.6). Now, suppose our goal is to generate a second-order linear transformation so defined

$$\boldsymbol{z} = B\boldsymbol{y} \tag{3.2.11}$$

which converts the components of $\boldsymbol{y}$ into components of $\boldsymbol{z}$. To express the components of $\boldsymbol{z}$ in terms of the components of $\boldsymbol{x}$ this, we write

$$z_i = \sum_{k=1}^{N} b_{ik} y_k = \sum_{k=1}^{N} b_{ik} \left(\sum_{j=1}^{N} a_{kj} x_j \right) \tag{3.2.12}$$

$$= \sum_{j=1}^{N} \left(\sum_{k=1}^{N} b_{ik} a_{kj} \right) \boldsymbol{x}_j \tag{3.2.13}$$

Introducing $C = (c_{ij})$ defined as

$$c_{ij} = \sum_{k=1}^{N} b_{ik} a_{kj} \qquad i, j = 1, 2, \ldots, N \tag{3.2.14}$$

we may write

$$\boldsymbol{z} = C\boldsymbol{x} \tag{3.2.15}$$

Since, formally

$$\boldsymbol{z} = B\boldsymbol{y} = B(A\boldsymbol{x}) = B(A\boldsymbol{x}) = (BA)\boldsymbol{x} \tag{3.2.16}$$

so that

$$C = BA \tag{3.2.17}$$

Note the ordering of the matrix product above.

Homework 10. Show that

$$f(\theta_1) f(\theta_2) = f(\theta_2) f(\theta_1) = f(\theta_1 + \theta_2) \tag{3.2.18}$$

where

$$f(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{3.2.19}$$

Homework 11. Show that

$$\left(a_1^2 + a_2^2\right)\left(b_1^2 + b_2^2\right) = \left(a_2 b_1 + a_1 b_2\right)^2 + \left(a_1 b_1 - a_2 b_2\right)^2 \tag{3.2.20}$$

Hint: $|AB| = |A||B|$,

3.2.3 Non-Commutativity

Matrix multiplication is not commutative, *i.e.* $AB \neq BA$. For an example, consider the following 3×3 matrices

$$A = \begin{bmatrix} 5 & 6 & 9 \\ 2 & 1 & 6 \\ 3 & 6 & 9 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 4 & 13 \\ 23 & 6 & 24 \\ 8 & 3 & 9 \end{bmatrix} \tag{3.2.21}$$

where

$$AB = \begin{bmatrix} 215 & 83 & 290 \\ 73 & 32 & 104 \\ 213 & 75 & 264 \end{bmatrix} \qquad \text{and } BA = \begin{bmatrix} 52 & 88 & 150 \\ 199 & 288 & 459 \\ 73 & 105 & 171 \end{bmatrix} \tag{3.2.22}$$

so that $AB \neq BA$. If, however, $AB = BA$, we say A and B *commute*. Note that

$$(AB)^{-1} = B^{-1}A^{-1}. \tag{3.2.23}$$

3.2.4 Associativity

Associativity of matrix multiplication gets preserved unlike the commutativity. So for matrices A, B, and C, we have

$$(AB)\,C = A\,(BC) \tag{3.2.24}$$

that is, the product ABC is unambiguously defined without the parentheses. To prove this, we write the ijth element of AB as

$$a_{ik}b_{kj} \tag{3.2.25}$$

so that the definition of multiplication implies that

$$(AB)\,C = [(a_{ik}b_{kl})\,c_{lj}] \tag{3.2.26}$$

$$A\,(BC) = [a_{ik}\,(b_{kl}c_{lj})] \tag{3.2.27}$$

which establishes the equality $(AB)\,C$ and $A\,(BC)$.

3.2.5 Invariant Vectors

The problem of finding the minimum or maximum of $Q = \sum_{i,j=1}^{N} a_{ij}\boldsymbol{x}_i\boldsymbol{x}_j$ for $\boldsymbol{x}_i$ satisfying the relation $\sum_{i=1}^{N} \boldsymbol{x}_i^2 = 1$ can be reduced to the problem of finding the values of the scalar λ that satisfies the set of linear homogeneous equations

$$\sum_{j=1}^{N} a_{ij}\boldsymbol{x}_j = \lambda\boldsymbol{x}_i, \qquad i = 1, 2, \ldots, N \tag{3.2.28}$$

which possesses nontrivial solutions. Vectorizing, we have

$$A\boldsymbol{x} = \lambda\boldsymbol{x} \tag{3.2.29}$$

Here, $\boldsymbol{x}$ signifies the direction indicated by the N direction numbers $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, and we are searching for the directions that are invariant.

3.2.6 The Matrix Transpose

We define the transpose of the matrix $A = (a_{ij})$ as $A^T = (a_{ji})$ $i.e.$ the rows of A^T are the columns of A and vice versa. An important consequence of this is that the transformation A on the set of a vector $\boldsymbol{x}$ is same as the transformation of the matrix A^T on the set $\boldsymbol{y}$. This is shown in the following

$$\langle A\boldsymbol{x}, \boldsymbol{y}\rangle = y_1 \sum_{j=1}^{N} a_{1j}\boldsymbol{x}_j + y_2 \sum_{j=1}^{N} a_{2j}\boldsymbol{x}_j + \ldots + y_N \sum_{j=1}^{N} a_{Nj}\boldsymbol{x}_j \tag{3.2.30}$$

which becomes upon rearrangement,

$$\langle A\boldsymbol{x}, \boldsymbol{y}\rangle = x_1 \sum_{i=1}^{N} a_{i1}\boldsymbol{y}_i + x_2 \sum_{i=1}^{N} a_{i2}\boldsymbol{y}_i + \ldots + x_N \sum_{i=1}^{N} a_{iN}\boldsymbol{y}_i \tag{3.2.31}$$

$$= \langle \boldsymbol{x}, A^T\boldsymbol{y}\rangle \tag{3.2.32}$$

We can then regard A^T as the *induced or adjoint transformation* of A. An interesting property of the transpose of amatrix product is that

$$(AB)^T = B^T A^T \tag{3.2.33}$$

3.2.7 Symmetric Matrices

Matrices that satisfy the relation

$$A = A^T \tag{3.2.34}$$

play a crucial role in the study of quadratic forms and such matrices are said to be *symmetric*, with the property that

$$a_{ij} = a_{ji} \tag{3.2.35}$$

Homework 12. Prove that $(A^T)^T = A$

Homework 13. Prove that $\langle A\boldsymbol{x}, B\boldsymbol{y}\rangle = \langle \boldsymbol{x}, A^T B\boldsymbol{y}\rangle$

3.2.8 Hermitian Matrices

The scalar function for complex vectors is the expression $\langle \boldsymbol{x}, \bar{\boldsymbol{y}}\rangle$. Suppose we define $\boldsymbol{z} = \bar{A^T}\boldsymbol{y}$, then

$$\langle A\boldsymbol{x}, \bar{y}\rangle = \langle \boldsymbol{x}, \bar{\boldsymbol{z}}\rangle \tag{3.2.36}$$

$i.e.$ the induced transformation is now $\bar{A^T}$, the complex conjugate of A. Matrices for which

$$A = \bar{A^T} \tag{3.2.37}$$

are called Hermitian. Note that in some literature, the Hermitian matrix is often written as $A^\star$.

3.2.9 Orthogonal Matrices

This section has to do with the invariance of distance between matrices, that is, taking the Euclidean measure of distance as the measure of the magnitude of the real-valued vector $\boldsymbol{x}$. The prodding question of interest is to figure out the linear transformation $\boldsymbol{y} = H\boldsymbol{x}$ that leaves the inner product $\langle \boldsymbol{x}, \boldsymbol{z} \rangle$. Mathematically, we express this problem such that

$$\langle \boldsymbol{x}, \boldsymbol{x} \rangle = \langle H\boldsymbol{x}, H\boldsymbol{x} \rangle \tag{3.2.38}$$

is satisfied for *all* $\boldsymbol{x}$. We know that

$$\langle H\boldsymbol{x}, H\boldsymbol{x} \rangle = \langle \boldsymbol{x}, H^T H\boldsymbol{x} \rangle \tag{3.2.39}$$

and that $H^T H$ is symmetric so that (3.2.38), gives

$$H^T H = I. \tag{3.2.40}$$

Orthogonal Matrix

A real matrix H for which $H^T H = I$ is called *orthogonal*.

3.2.10 Unitary Matrices

This is the measure of the distance of a complex vector, akin to the invariance condition of real-valued matrices (3.2.40). We define the unitary property as follows:

$$H^\star H = I. \tag{3.2.41}$$

Matrices defined as in the foregoing play a crucial role in the treatment of Hermitian matrices, such as the role that orthogonal matrices play in symmetric matrices theory.

3.2.11 Matrix Determinant

The determinant of a scalar is same as the scalar while the determinant of a matrix shall be inductively defined for square matrices. Suppose we have an $n \times n$ matrix A, its determinant is defined as

$$|A| = \sum_{j=1}^{N} (-1)^{i+j} a_{(i,j)} |a^{(i,j)}| \tag{3.2.42}$$

for any value of $i \in [1, n]$, where (3.2.42) is called the Laplace expansion of $|A|$ along its ith row. Equation (3.2.42) shows us that the determinant of the square matrix A is found in terms of the determinants of the $(n-1) \times (n-1)$ matrices. Similarly, the determinants of $(n-1) \times (n-1)$

matrices are defined by $(n-2)\times(n-2)$ and so on until we get to the determinant of 1×1 matrices which are scalars. We can also define the determinant of A as

$$|A| = \sum_{i=1}^{N}(-1)^{i+j}a_{(i,j)}|a^{(i,j)}| \tag{3.2.43}$$

for any value of $j \in [1, n]$. This is termed the Laplace expansion of A along its jth column. It follows that

$$|A_{11}| = A_{11} \tag{3.2.44a}$$

$$\det\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = A_{11}A_{22} - A_{12}A_{21} \tag{3.2.44b}$$

and that

$$\det\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = A_{11}(A_{22}A_{33} - A_{23}A_{32}) - \tag{3.2.44c}$$

$$A_{12}(A_{21}A_{33} - A_{23}A_{31}) + \tag{3.2.44d}$$

$$A_{13}(A_{21}A_{32} - A_{22}A_{31}) \tag{3.2.44e}$$

3.2.12 Properties of the Matrix Determinant

1. $|AB| = |A||B|$, where A and B are assumed to be of equal dimensions.
2. $|A| = \prod_{i=1}^{N}\lambda_i$, where λ_i are the eigenvalues of A.
3. The inverse of A is said to exist if $AA^{-1} = I$. Such a matrix is said to be *non-singular*. Note that A must be a square matrix in order for it to have a determinant. A square matrix whose inverse does not exist is said to be *singular*.

Take for example,

$$\begin{bmatrix} 3 & 0 \\ 2,1 & \end{bmatrix}\begin{bmatrix} 1/3 & 0 \\ -2/3, 1 & \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0, 1 & \end{bmatrix}. \tag{3.2.45}$$

Then we say that the two matrices on the left are inverses of one another. Among other ways of stating the nonsingularity of A are that

- A's rows or columns are linearly independent.
- $|A| \neq 0$.
- $Ax = b$ has a unique solution x for all b.
- The rank of $A = n$.
- 0 is not an eigenvalue of A.
- A^{-1} exists.

3.2.13 The Matrix Trace

The trace of a matrix exists if and only of the matrix is square. It is defined as the sum of its diagonal elements.

$$\text{Tr}(A) = \sum_{i=1} A_{ii} \tag{3.2.46}$$

Also, the trace can be expressed in terms of the sum of the matrix's eigenvalues,

$$\text{Tr}(A) = \sum_{i=1} \lambda_i. \tag{3.2.47}$$

The trace of a matrix product is not dependent in the order of multiplication of the matrices:

$$\text{Tr}(AB) = \text{Tr}(BA). \tag{3.2.48}$$

3.2.14 Eigenvectors and Eigenvalues of a Matrix

A square $n \times n$ matrix A has n eigenvalues and n eigenvectors. If

$$A\boldsymbol{x} = \lambda\boldsymbol{x} \tag{3.2.49}$$

for a scalar λ and an $n \times 1$ vector $\boldsymbol{x}$ then we say the matrix A has eigenvalues λ and eigenvectors $\boldsymbol{x}$. Together, λ and $\boldsymbol{x}$ are called *eigendata*, the *characteristic roots*, *latent roots*, or *proper numbers and vectors* of the matrix.

Homework 14. If A has eigendata $(\lambda, \boldsymbol{x})$, show that A^2 has eigendata $(\lambda^2, \boldsymbol{x})$.

Homework 15. Show that A^{-1} exists if and only if none of the eigenvalues of A are zero.

Homework 16. Show that the eigenvalues of A are real numbers if A is symmetric.

3.2.15 Other Matrix Properties

A *symmetric* $n \times n$ matrix A can be characterized as either positive definite, positive semidefinite, negative definite, negative semidefinite, or indefinite if matrix A is

- *Positive definite* if $\boldsymbol{x}^T A\boldsymbol{x} > 0$ for all nonzero $n \times 1$ vectors $\boldsymbol{x}$. That is, all the eigenvalues of A are positive real numbers. If A is positive definite, then so is A^{-1}.
- *Positive semidefinite* if $\boldsymbol{x}^T A\boldsymbol{x} \geq 0$ for all nonzero $n \times 1$ vectors $\boldsymbol{x}$. That is, all the eigenvalues of A are non-negative real numbers. A positive semidefinite matrices are sometimes called nonnegative definite.
- *Negative definite* if $\boldsymbol{x}^T A\boldsymbol{x} < 0$ for all nonzero $n \times 1$ vectors $\boldsymbol{x}$. That is, all the eigenvalues of A are negative real numbers. If A is negative definite, then so is A^{-1}.

- *Negative semidefinite* if $\boldsymbol{x}^T A\boldsymbol{x} \leq 0$ for all nonzero $n \times 1$ vectors $\boldsymbol{x}$. That is, all the eigenvalues of A are non-negative real numbers. A positive semidefinite matrices are sometimes called non positive definite.
- When some of the eigenvalues of A are positive and some are negative, then the matrix is said to be *indefinite*.

The singular values of matrix A are defined as

$$\begin{aligned} \sigma^2(A) &= \lambda(A^T A) \\ &= \lambda(A^T A) \end{aligned} \tag{3.2.50}$$

For an $n \times n$ matrix A, we have a $\min(n, m)$ singular values. If $n > m$, then AA^T will have the same eigenvalues as $A^T A$ and an additional $n - m$ zeroes. We do not consider the additional zeroes to be singular values of A because A always has $\min(n, m)$ singular values.

Quiz 2. If A is $n \times m$, what are number of eigenvalues of $A^T A$ and AA^T respectively?

3.2.16 The Matrix Inversion Lemma

This is sometimes called the *Woodbury matrix identity*, named after Max A. Woodbury., *Sherman-Morrison formula*, or the *modified matrices formula*. It a tool frequently used in statistics, system identification, state estimation and control theory. Assume we have a blockwise matrix $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$ where A and D are invertible square matrices, and B and C are not necessarily square. We can define the following matrices

$$\begin{aligned} E &= D - CA^{-1}B \\ F &= A - BD^{-1}C. \end{aligned} \tag{3.2.51}$$

If E is invertible, it follows that

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} A^{-1} + A^{-1}BE^{-1}CA^{-1} & -A^{-1}BE^{-1} \\ -E^{-1}CA^{-1} & E^{-1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}. \tag{3.2.52}$$

Also, if F were invertible, it follows that

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} F^{-1} & -A^{-1}BE^{-1} \\ -D^{-1}CF^{-1} & E^{-1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}. \tag{3.2.53}$$

It follows that (3.2.52) and (3.2.53) are two expressions for the inverse of $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$. We must therefore have the upper-left partitions of the two matrices equal so that

$$F^{-1} = A^{-1} + A^{-1}BE^{-1}CA^{-1} \tag{3.2.54}$$

and from the definition of F, we have

$$\left(A - BD^{-1}C\right)^{-1} = A^{-1} + A^{-1}B\left(D - CA^{-1}B\right)^{-1}CA^{-1} \tag{3.2.55}$$

An alternative statement of the matrix inversion lemma is

$$\left(A - +BD^{-1}C\right)^{-1} = A^{-1} + A^{-1}B\left(D + CA^{-1}B\right)^{-1}CA^{-1} \tag{3.2.56}$$

Quiz 3. Verify the expressions in (3.2.52) and (3.2.53).

Example 1. Suppose that at Brandeis, you took three courses in your Freshman year namely, RBOT 101, RBOT 103, and RBOT 105 where you got 90%, 85%, and 86% respectively. In your Sophomore year, you took RBOT 201, RBOT 203, and RBOT 205, where you got 65%, 68%, and 92% respectively, and in your junior year, you decide to retake your Sophomore classes, where your scores increased by 10%, 5%, on the first two courses and decreased by 8% in the last course. Your GPA each year increased by 4%, 3.5% and 2.5% respectively. Given your analytical prowess, you decide to model each year's GPA changes with the equation $z = au + bv + cw$, where u and v and w are the scores/grades you got as percentages and a, b, and c are unknown constants. To find the unknown constants, you figure you need to invert the matrix

$$A = \begin{bmatrix} 90 & 85 & 86 \\ 65 & 68 & 92 \\ 71.5 & 71.4 & 84.64 \end{bmatrix} \tag{3.2.57}$$

so that

$$A^{-1} = \begin{bmatrix} 23/20 & 155/104 & -145/52 \\ -207/136 & -569/274 & 6725/1768 \\ 5/16 & 205/416 & -175/208 \end{bmatrix} \tag{3.2.58}$$

It follows that the unknown constants are

$$X = A^{-1}\begin{pmatrix} 4 \\ 7/2 \\ 5/2 \end{pmatrix} = \begin{pmatrix} 2959/1040 \\ -2693/700 \\ 725/832 \end{pmatrix} \tag{3.2.59}$$

As a result, you are able to determine a model which allows you to predict future GPA changes based on how hard you work, sleep, engage in social activities. You can better allocate your time resource and improve your grades in the following years.

Suppose that in the aftermath of generating this model, you now realize that your grade in RBOT 201 the second year was 86% rather than 65%, this means that in order to find the constants, you want to invert

$$\bar{A} = \begin{bmatrix} 90 & 85 & 86 \\ 86 & 68 & 92 \\ 71.5 & 71.4 & 84.64 \end{bmatrix}. \tag{3.2.60}$$

Rather than invert all the matrix all over, you decide to apply a mathematical trick leveraging the inversion lemma. You write $\bar{A} = A + BD^{-1}C$, where

$$B = \begin{bmatrix} 0 & 21 & 3.44 \end{bmatrix}^T C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} D = 1 \tag{3.2.61a}$$

so that

$$\bar{A}^{-1} = \left(A + BD^{-1}C\right)^{-1} \tag{3.2.62}$$

$$= A^{-1} - A^{-1}B\left(D + CA^{-1}B\right)^{-1}CA^{-1} \tag{3.2.63}$$

The $(D + CA^{-1}B)^{-1}$ term turns out to be a scalar so that

$$\bar{A}^{-1} = \begin{bmatrix} 0.0506 & 0.0656 & -0.1228 \\ 0.0239 & -0.073 & 0.0551 \\ -0.065 & 0.0035 & 0.0741 \end{bmatrix} \tag{3.2.64}$$

$$\begin{aligned} X &= \bar{A}^{-1} \begin{pmatrix} 4 \\ 7/2 \\ 5/2 \end{pmatrix} \\ &= \begin{pmatrix} 51/407 \\ -134/6037 \\ -148/2361 \end{pmatrix} \end{aligned} \tag{3.2.65}$$

Here, the matrix inversion lemma may not be necessary since the size of the matrix is small. However, if the matrix had a larger size, the computational savings of using the matrix inversion lemma becomes appreciated.

Homework 17. Using the same linear model employed in 1, suppose that on a typical weekend, you go to your local Farmers' market and bought tomatoes, bell peppers, and blue berries for $\$35\%$, $\$18\%$, and $\$32\%$ respectively; on your way home, you drove to your local grocery store and found that the prices for each item were actually increased by 10%, 25% for each of tomatoes, and bell peppers, and decreased by 68% for the blue berries. You decide to buy more blueberries that cost a total of $\$50$ at your local grocery; and discovered that $\$5$ worth of tomatoes bought at the Farmers' market was defective and had to be discarded. Can you compute a model that allows you to predict future prices for good tomatoes, blue berries and bell peppers at your local Farmers' market?

CHAPTER 4

REGISTRATION OF OBJECTS IN ROBOTICS.

In this chapter, we are concerned with the problem of optimally aligning two vectors, a model point/shape to a “sensed" or measured point/shape in space e.g. $\nu_1, \nu_2 \in \mathbb{R}^n$ to one another with the minimal amount of errors. To transform between two pints in the Cartesian coordinate system is akin to the problem of solving a rigid body motion problem where that yields a rotation and a translation. In addition, the scaling factor may be unknown. For translation, there are three degrees of freedom, while rotation has another three viz., the direction of the axis about which we are rotating, the angle of rotation itself, and the scaling. Three points in either coordinate systems give us nine constraints (with each contributing three coordinates), more than enough to find the seven unknowns. If we discard two of the constraints, we end up with seven equations in seven unknowns that can be developed to allow us to recover the parameters.

There exists many methods of solving this problem. Most of them leverage clever optimization methods and we will be looking into these in this chapter. We could follow the homogeneous transformation scheme we presented in Chapter 1, but we would not have an optimal solution. A popular technique in computer geometry and computer vision is to use the iterative closest point algorithm(ICP), an algorithm by Paul Besl and Neil McKay developed out of General Motors Laboratory in the 1990’s (Besl and McKay, 1992). This is more appropriate for 3D tasks and it describes a generic, representation method for the accurate and computationally efficient registration of three-dimensional (3-D) shapes. The ICP algorithm always converges monotonically to the nearest local minimum of a mean-square distance metric such as an l_2 distance, and this convergence rate is of the order of a few iterations. An important property of the ICP algorithm is that it can register data from unfixtured rigid objects with an ideal geometrical model prior to shape inspection. So, if we want to figure out that two geometric representations are congruent, estimate the motion between them in real-time where the correspondences are not known, ICP tends to be really good for such operations.

Now, suppose our dataset is not a complex geometric primitive[1], but rather a set of two vectors such that we are tasked with the problem of determining the best *unconstrained transformation* between the two sets of coordinates. We can formulate the problem into a constrained optimization problem and thereafter, through clever factorization, turn the problem into a simple one of factorizing the unconstrained transformation into a symmetric and orthogonal matrix by which we may solve for the optimal rotation and translation. The algorithm we shall be looking into will be the one that was invented in crystallography in 1976 and updated in 1978 by Wolfgang Kabsch, today dubbed the Kabsch algorithm (Kabsch, 1978). Kabsch showed that a direct solution was possible, irrespective of the non-linear character of the problem.

While other newer algorithms exist, these are the two popular algorithms that we shall be concerning ourselves with in this chapter.

[1]We shall refer to a geometric primitive as a primitive 3D shape such as a cylinder, square, prism and the likes.

4.1 Preliminaries

We will denote the real line by $\mathbb{R}$. An example of a **metric space** is the **Euclidean n-space** $\mathbb{R}^n$, which consists of $n-$tuples $x = (x_1, x_2, \ldots, x_n)$ where each $x_i \in \mathbb{R}$. We shall mean an $\mathbb{R}^n$ metric space to have the metric

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}. \tag{4.1.1}$$

If $n = 0$, then $\mathbb{R}^0$ is taken to be a single point $0 \in \mathbb{R}$.

A manifold is "locally" similar to one of the example metric spaces $\mathbb{R}^n$. Precisely, a **manifold** is a metric space $\boldsymbol{M}$ with the property that, *if $x \in M$., then there is some neighborhood U of x and some integer $n \geq 0$ such that U is homeomorphic*[2] *to $\mathbb{R}^n$.*

A simple example of a manifold is $\mathbb{R}^n$: for each $x \in \mathbb{R}^n$ we can take U to be everything in $\mathbb{R}^n$.

Quiz 4. Suppose we supply $\mathbb{R}^n$ with an equivalent metric, which makes it homeomorphic to $\mathbb{R}^n$, would it also be a manifold?

Another example of a metric space is an open ball in $\mathbb{R}^n$, wherein one can take U to be the entire open ball since an open ball in $\mathbb{R}^n$ is homeomorphic to $\mathbb{R}^n$. Similarly, an open subset V of $\mathbb{R}^n$ is a manifold, *i.e.* for each $x \in V$ we can choose U to be some open ball with $x \in U \subset V$.

The **Euclidean distance** $d(\boldsymbol{r}_1, \boldsymbol{r}_2)$ between two points $\boldsymbol{r}_1 = (x_1, y_1, z_1)$ and $\boldsymbol{r}_2 = (x_2, y_2, z_2)$ is given by

$$d(\boldsymbol{r}_1, \boldsymbol{r}_2) = \|\boldsymbol{r}_1 - \boldsymbol{r}_2\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}. \tag{4.1.2}$$

Suppose that P is a point set with N_p points denoted as $\boldsymbol{p}_i : P = \{p_i\}$ for $i = 1, \ldots, N_p$. The distance between the point $\boldsymbol{q}$ and the point set P is

$$d(\boldsymbol{q}, P) = \min_{i \in \{1, \ldots, N_p\}} d(\boldsymbol{q}, \boldsymbol{p}_i). \tag{4.1.3}$$

We find that the closest point $\boldsymbol{p}_j$ of P satisfies $d(\boldsymbol{q}, \boldsymbol{p}_j) = d(\boldsymbol{q}, P)$.

Suppose that we have a **line segment** that connects the points, $(\boldsymbol{r}_1, \boldsymbol{r}_2)$, the distance between the point $\boldsymbol{r}$ and the line segment l is

$$d(\boldsymbol{p}, l) = \min_{x+y=1} \|x\boldsymbol{r}_1 + y\boldsymbol{r}_2 - \boldsymbol{p}\| \tag{4.1.4}$$

where $x, y \in [0, 1]$.

[2] A homeomorphic mapping means intrinsic topological equivalence between e.g. objects. Two objects are homeomorphic if they can be deformed into each other by a continuous, invertible mapping. Such a homeomorphism ignores the space in which surfaces are embedded, so the deformation can be completed in a higher dimensional space than the surface was originally embedded. Mirror images are homeomorphic, as are Möbius strip with an even number of half-twists, and Möbius strip with an odd number of half-twists (Weisstein, Weisstein).

Homework 18. Find a closed-form expression for the solution to (4.1.4).

Now, if instead of a line segment, suppose we have a set of N_l line segments denoted l_i, and let $L = \{l_i\}$ for $i = 1, \ldots, N_l$. The **distance between the point $\boldsymbol{p}$ and the line segment set** L is

$$d(\boldsymbol{p}, L) = \min_{i \in \{1,\ldots,N_l\}} d(\boldsymbol{p}, l_i). \tag{4.1.5}$$

The closest point y_j on the line segment set L satisfies $d(\boldsymbol{p}, y_j) = d(\boldsymbol{p}, L)$. Let g be a triangle with the following coordinates $\boldsymbol{r} = (x_1, y_1, z_1), \boldsymbol{r}_2 = (x, y, z)$, and $\boldsymbol{r}_3 = (x_3, y_3, z_3)$. The **distance between the point $\boldsymbol{p}$ and the triangle** g is

$$d(\boldsymbol{p}, g) = \min_{x+y+z=1} \|x\boldsymbol{r}_1 + y\boldsymbol{r}_2 + z\boldsymbol{r}_3 - \boldsymbol{p}\| \tag{4.1.6}$$

where $x \in [0, 1],\ y \in [0, 1]$, and $z \in [0, 1]$.

Homework 19. Find a closed-form expression for the problem in (4.1.6).

Now, if we have a collection of N_g triangles G, denoted by g_i such that $G = \{g_i\}$ for $i = 1, \ldots, N_g$. The **distance between the point $\boldsymbol{p}$ and the triangle set** G is

$$d(\boldsymbol{p}, G) = \min_{i \in \{1,\ldots,N_g\}} d(\boldsymbol{p}, g_i), \tag{4.1.7}$$

and the closest point y_j on the triangle set G satisfies the equality $d(\boldsymbol{p}, y_j) = d(\boldsymbol{p}, G)$.

4.1.1 Distance between a Point and a Parameterized Entity

We define a parametric curve and a parametric surface as single parametric entities $\boldsymbol{r}(\boldsymbol{u})$, where $\boldsymbol{u} = u \in \mathbb{R}^1$ denotes a parameterized curve, and $\boldsymbol{u} = (u, v) \in \mathbb{R}^2$ denotes parametric surfaces. We will evaluate a curve within an interval domain e.g. $[x, y]$ while the evaluation domain of a surface can be an arbitrarily closely-connected region in a plane.

We will take the distance from a given point $\boldsymbol{p}$ to a parametric entity E to be

$$d(\boldsymbol{p}, E) = \min_{\boldsymbol{r}(\boldsymbol{u}) \in E} d(\boldsymbol{p}, \boldsymbol{r}(\boldsymbol{u})) \tag{4.1.8}$$

To compute the point-to-curve and point-to-surface distances, let F be the set of N_e parametric entities denoted by E_i, and let $F = \{E_i\}$ for $i = 1, N_e$> The distance between a point $\boldsymbol{p}$ and the parametric entity set F is

$$d(\boldsymbol{p}, F) = \min_{i \in \{1,\ldots,N_e\}} d(\boldsymbol{p}, \boldsymbol{E}_i). \tag{4.1.9}$$

To find the distance from a point to a parametric entity, we can create a simplex-based approximation for e.g. a line segment or triangle. For a parametric space curve $C = \{\boldsymbol{r}(u)\}$, we can compute a polyline $L(C, \delta)$ such that the piecewise-linear approximation never deviates from the space curve by more than a prespecified distance δ. If we tag every point of the polyline with a

corresponding u argument values of the parametric curve, we can obtain an estimate of the closest point from the line segment set.

In a similar vein, for a parametric surface $S = \{\boldsymbol{r}(u, v)\}$, one can compute a triangle set $G(S, \delta)$ such that the piecewise triangular approximation never deviates from the surface by more than a prespecified distance δ. If we tag each truangle vertex with the corresponding (u, v) argument values of the parametric surface, we can find the $U_a, v_a)$ of the argument values of the closest point from the triangle set. The initial value of $\boldsymbol{u}_a$ is assumed to be available such that $\boldsymbol{r}(\boldsymbol{u}_a)$ is very close to the closest point on the parametric entity.

We can employ a Newtonian minimization approach for solving the point to parametric entity problem when a reliable starting point $\boldsymbol{u}_a$ is available. The scalar objective function to be minimized is

$$f(\boldsymbol{u}) = \|\boldsymbol{r}(\boldsymbol{u}) - \boldsymbol{p}\|^2. \tag{4.1.10}$$

Suppose $\Delta = [\partial/\partial \boldsymbol{u}]^T$ is the vector differential gradient operator, the minimum of f must occur at $\Delta f = 0$. If we have a surface, then we must have $\Delta f = [f_u, f_v]^T$, with the 2-D Hessian matrix is given by

$$\Delta\Delta^T(f) = \begin{bmatrix} f_{uu} & f_{uv} \\ f_{uv} & f_{vv} \end{bmatrix} \tag{4.1.11}$$

where the partial derivatives of the objective function is

$$f_u(\boldsymbol{u}) = 2\boldsymbol{r}_u^T(\boldsymbol{u})(\boldsymbol{r}(\boldsymbol{u}) - \boldsymbol{p}) \tag{4.1.12a}$$
$$f_v(\boldsymbol{u}) = 2\boldsymbol{r}_v^T(\boldsymbol{u})(\boldsymbol{r}(\boldsymbol{u}) - \boldsymbol{p}) \tag{4.1.12b}$$
$$f_{uu}(\boldsymbol{u}) = 2\boldsymbol{r}_{uu}^T(\boldsymbol{u})(\boldsymbol{r}(\boldsymbol{u}) - \boldsymbol{p}) + 2\boldsymbol{r}_u^T(\boldsymbol{u})\boldsymbol{r}_u(\boldsymbol{u}) \tag{4.1.12c}$$
$$f_{vv}(\boldsymbol{u}) = 2\boldsymbol{r}_{vv}^T(\boldsymbol{u})(\boldsymbol{r}(\boldsymbol{u}) - \boldsymbol{p}) + 2\boldsymbol{r}_v^T(\boldsymbol{u})\boldsymbol{r}_v(\boldsymbol{u}) \tag{4.1.12d}$$
$$f_{uv}(\boldsymbol{u}) = 2\boldsymbol{r}_{uv}^T(\boldsymbol{u})(\boldsymbol{r}(\boldsymbol{u}) - \boldsymbol{p}) + 2\boldsymbol{r}_u^T(\boldsymbol{u})\boldsymbol{r}_v(\boldsymbol{u}). \tag{4.1.12e}$$

And the update relation for the curve and surface case is

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k - \left[\Delta\Delta^T(f)(\boldsymbol{u}_k)\right]^{-1} \Delta f(\boldsymbol{u}_k) \tag{4.1.13}$$

where $\boldsymbol{u}_0 = \boldsymbol{u}_a$.

4.1.2 Distance between a Point and an Implicit Entity

An implicit geometric entity is the zero set of a possibly vector-valued multivariate function $\boldsymbol{g}(\boldsymbol{r}) = 0$. Examples of this distance could be a point-to-curve or point-to-surface distance. The important thing to bear in mind is that the distance metric for an individual entity, once defined, makes the sets of implicit entities straightforward to implement. The distance from a given point $\boldsymbol{p}$ to an implicit entity I is given by

$$d(\boldsymbol{p}, I) = \min_{\boldsymbol{g}(\boldsymbol{r})=0} d(\boldsymbol{p}, \boldsymbol{r}) = \min_{\boldsymbol{g}(\boldsymbol{r})=0} \|\boldsymbol{r} - \boldsymbol{p}\|. \tag{4.1.14}$$

It is helpful to note that when computing the implicit entity distance from a point, the solution is never closed-form and are usually involved. Suppose that J is the set of N_I parametric entities, represented by I_k and $J = \{I_k\}$ for $k = 1, N_I$. The distance between a point $\boldsymbol{p}$ and the implicit entity set J is given by

$$d(\boldsymbol{p}, J) = \min_{k \in \{1,\ldots,N_I\}} d(\boldsymbol{p}, \boldsymbol{I}_k), \tag{4.1.15}$$

and the closest point $\boldsymbol{y}_j$ on the implicit entity I_j satisfies the equality $d(\boldsymbol{p}, \boldsymbol{y}_j) = d(\boldsymbol{p}, J)$. In order to compute the distance from a point to an implicit entity, we can create a simplex-based approximation such as line segments or triangles. The point-to-line or point-to-triangle set distance yields an approximate closest point $\boldsymbol{r}_a$ which can be used to compute the exact distance.

Typically, we must solve a constrained optimization problem when finding the closest point on an implicit entity, say $\boldsymbol{g}(\boldsymbol{r}) = 0$ to a point $\boldsymbol{p}$ in order to minimize a quadratic objective function that is subject to a nonlinear constraint

$$\min f(\boldsymbol{r}) = \|\boldsymbol{r} - \boldsymbol{p}\|^2 \tag{4.1.16}$$

where $\boldsymbol{g}(\boldsymbol{r}) = 0$ We can form the augmented Lagrange multiplier system of equations to solve the above, $i.e.$

$$\begin{aligned} \Delta f(\boldsymbol{r}) + \boldsymbol{\lambda}^T \Delta \boldsymbol{g}(\boldsymbol{r}) &= 0 \\ \boldsymbol{g}(\boldsymbol{r}) &= 0 \end{aligned} \tag{4.1.17}$$

where $\Delta - [\partial/\partial \boldsymbol{r}]^T$.

4.1.3 Quaternions

The unit quaternion is a four vector $\boldsymbol{q}_R = [q_0, q_1, q_2, q_3]^T$, where $q_0 \geq 0$, and $q_0^2 + q_1^2 + q_2^2 + q_3 = 1$, used to parameterize a rotation matrix. The 3×3 rotation matrix generated by a unit rotation quaternion is given by

$$R = \boldsymbol{q}_R^T \boldsymbol{q}_R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3 & 2(q_1 q_2 - q_0 q_3) & 2\,(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 + q_1^2 - q_2^2 - q_3 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 + q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 + q_3^2 - q_1^2 - q_2 \end{bmatrix} \tag{4.1.18}$$

For more on unit quaternions, see **??**.

4.2 Closed-form Solution using Least Sum of Squares Errors

As we will see from our sensors, measurements are often inexact, which means we need a way to enforce greater accuracy when determining the transformation parameters. Therefore, we will need more than three points. One approach is to minimize the sum of squares of residual errors using various empirical, graphical, and numerical procedures. Because these are iterative in nature, they

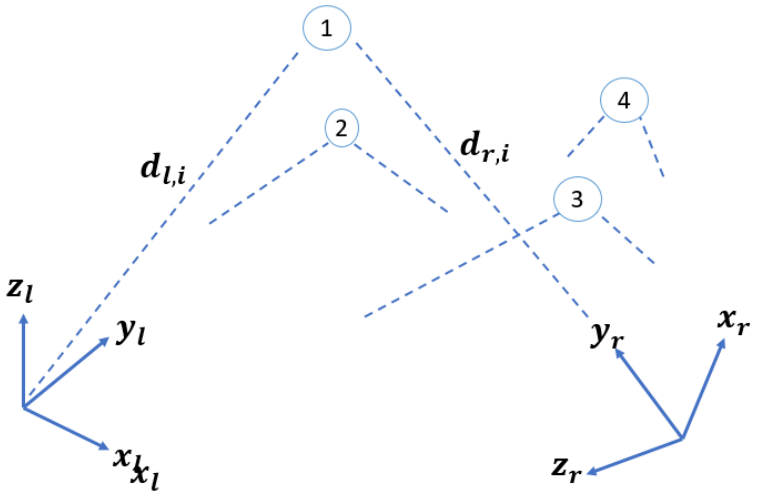

Figure 4.1: Given two coordinate systems, we measure a number of points in the two different coordinate systems. The goal is to find the transformation between the two points.

lead to an approximate solution and while the answer is better, it is imperfect.Iterative methods are repeatedly applied until the residual error is negligible.

There are closed-form solutions which present the absolute orientation in a single step with the best possible transformation given the measurements of the points in the two coordinate systems (Horn, 1987; Kabsch, 1978). With these closed-form least sum of squares methods, we do not need to find an initial good guess as is the case for iterative methods.

4.2.1 Kabsch Algorithm

Suppose we have two sets of vectors $\boldsymbol{x_n}$ and $\boldsymbol{y_n}$ where $n = 1, \ldots, N$, and weight w_n that corresponds to each pair $\boldsymbol{x_n}$ and $\boldsymbol{y_n}$. Our goal is to find an orthogonal matrix $\cup = (u_{ij})$ which minimizes the cost function

$$C = \frac{1}{2}\Sigma_n w_n \left(\cup \boldsymbol{x_n} - \boldsymbol{y_n}\right)^2 \tag{4.2.1}$$

subject to

$$\sum_k u_{ki}u_{kj} - \delta_{ij} = 0 \tag{4.2.2}$$

where δ_{ij} are the elements of a unit matrix. When there is a translation, we can find the centroid of the vector sets to the origin.

In order to solve the problem, we may introduce a symmetric Lagrangian matrix of multipliers, $L = (l_{ij})$ and an auxiliary function as follows

$$D = \frac{1}{2}\Sigma_{i,j} l_{ij} \left(\Sigma_k u_{kl} u_{kj} - \delta_{ij}\right) \tag{4.2.3}$$

so that we can form the Lagrangian, $E = C + D$. For each condition in eq. 4.2.2, we have an independent number l_{ij} so that the constrained minimum of C is part of the free minima of D. A free minimum of D can occur if

$$\frac{\partial E}{\partial u_{ij}} = \sum_k u_{ik} \left(\Sigma_n w_n x_{nk} x_{nj} + l_{k,j}\right) - \sum_n w_n y_{nl} x_{nj} = 0 \tag{4.2.4}$$

and

$$\frac{\partial^2 E}{\partial u_{mk} \partial u_{ij}} = \delta_{mi} \left(\Sigma_n w_n x_{nk} x_{nj} + l_{kj}\right) \tag{4.2.5}$$

are elements of a positive definite matrix x_{nk} and y_{nk} are the kth elements of $\boldsymbol{x_n}$ and $\boldsymbol{y_n}$. Now, suppose we have a matrix $R = (r_{ij})$ and a symmetric matrix $\boldsymbol{S} = (s_{ij})$, such that

$$r_{ij} = \sum_n w_n y_{ni} x_{nj} \tag{4.2.6}$$

and

$$s_{ij} = \sum_n w_n x_{ni} x_{nj}. \tag{4.2.7}$$

If the matrix (4.2.5) has 1 along its diagonal, we must have the minimum of the Lagrangian E to mean that $S + L$ is positive definite, and (4.2.4) translates to

$$U.\,(S + L) = R. \tag{4.2.8}$$

Our goal would be to find a matrix L of Lagrange multipliers so that $\cup$ is orthogonal. We can do this by multiplying both sides of (4.2.8) by their transposed matrices so that we can get rid of matrix $\cup$ as follows:

$$\begin{aligned} U(S + L)^T (S + L) &= (S + L)^T U^T U (S + L) \\ &= (S + L)(S + L) = R^T R. \end{aligned} \tag{4.2.9}$$

Now, we know that $R^T R$ is a symmetric positive definite matrix so that we can find the eigenvalues λ_k and eigenvectors $\boldsymbol{v}_k$ using standard procedures e.g. single value decomposition. Thus, since $S + L$ is symmetric and positive definite, it must have normalized eigenvectors, $\boldsymbol{v}_k$ and positive eigenvalues $\sqrt{\lambda_k}$ so that the Lagrange multipliers are

$$l_{ij} = \Sigma_k \sqrt{\lambda_k}; \qquad \boldsymbol{v}_{ki} \boldsymbol{v}_{ki} - s_{ij} \tag{4.2.10}$$

where $\boldsymbol{v}_{ki}$ signifies the ith component of $\boldsymbol{v}_k$ and the effect of the orthogonal matrix U on these eigenvectors $\boldsymbol{a}_k$ is determined from (4.2.8) which defines the unit vectors $\boldsymbol{q}_k$ as

$$\boldsymbol{q}_k = U.\boldsymbol{v}_k = \frac{1}{\sqrt{\lambda_k}} U(S+L)\boldsymbol{v}_k = \frac{1}{\sqrt{\lambda_k}} R\boldsymbol{v}_k. \tag{4.2.11}$$

The solution to find the constraint minimum of the minimum of the proposed cost function in (4.2.1) is then given by,

Kabsch's Optimal Rotation

$$u_{ij} = \Sigma_k b_{kl} a_{kj}. \tag{4.2.12}$$

4.2.2 Examples

There are clever ways of solving the optimal rotation between two vectors.

There is a jupyter notebook at the following link: Kabsch Algorithm and Implementation. For your convenience, it is included as a pdf file below.

Kabsch

December 15, 2020

0.1 Overview

Throughout this course, we will be leveraging Google's Colab Notebooks to reinforce the concepts we have been learning in class. For an introduction into how to use collab, in case you are not already familiar with it, have a go at this overview of Colaboratory features.

0.1.1 Kabsch's Algorithm

As stated in the course notes, the Kabsch algorithm is a very versatile tool for optimally aligning two vectors to one another. In this example, we are provided with two point sets - a model set and a point (measured) set, and our goal would be to compute the optimal rotation matrix U that allows us to efficiently rotate the point set into the model set.

0.1.2 Load the Measured Point Set

For the example we are interested in, we have measured the position of an object in 3D space using a Northern Digital Inc's Polaris Camera. The points are collected as a set of three-dimensional (3D) points in space, arranged in rows of `(x,y,z)` tuples and they are as given by the `measured_points_full` function below:

```
In [9]: # Here, we are importing all the libraries we will be using in these notebook
        import os
        import numpy as np
        from os.path import join, expanduser
        import scipy.linalg as LA
```

```
In [8]: def measured_points_full():
            # these are the (x,y,z) tuples
            pre_calib = {
                '0,0.0': [-369.88531494140625, 101.30087280273438, -1960.3780517578125],
                '200,0': [-369.8937683105469, 101.32111358642578, -1960.302734375],
                '0,0.1': [-369.8780212402344, 101.32646942138672, -1960.353271484375],
                '220,0': [-369.8780212402344, 101.32646942138672, -1960.353271484375],
                '0,0.2': [-367.74957275390625, 101.65080261230469, -1953.7960205078125],
                '240,0': [-370.8532409667969, 101.074951171875, -1942.255126953125],
                '0,0.3': [-366.7646484375, 101.17594909667969, -1949.628173828125],
                '255,0': [-381.33837890625, 97.10205078125, -1920.667236328125],
                '0,0.4': [-368.0609436035156, 100.83153533935547, -1953.857177734375],
                '0,220': [-382.8047790527344, 100.34918975830078, -1944.807373046875],
```

```
    '0,0.5': [-369.7981262207031, 100.01362609863281, -1958.6396484375],
    '0,240': [-382.71600341796875, 99.87244415283203, -1945.184326171875],
    '0,0.6': [-370.24237060546875, 98.66026306152344, -1957.2281494140625],
    '0,255': [-382.71600341796875, 99.87244415283203, -1945.184326171875],
    '0,0.7': [-370.1295166015625, 98.33242797851562, -1956.1732177734375],
    }
def exp(x):
    'This function expands the array along the second dimension so that '
    return np.expand_dims(x, 1)

# sort pre-recorded points in the order.
measured_calib = np.array(([[
                            pre_calib['0,0.0'],
                            pre_calib['0,220'],
                            pre_calib['0,0.1'],
                            pre_calib['0,240'],
                            pre_calib['0,0.2'],
                            pre_calib['0,255'],
                            pre_calib['0,0.3'],
                            pre_calib['200,0'],
                            pre_calib['0,0.4'],
                            pre_calib['220,0'],
                            pre_calib['0,0.5'],
                            pre_calib['240,0'],
                            pre_calib['0,0.6'],
                            pre_calib['255,0'],
                            pre_calib['0,0.7'],
                        ]]))
"""
 As it is currently, our array has 3 dimensions. We need to reduce the size of the
 array alongthe singleton dimention for efficient matrix manipulations, hence why we
 are squeezing the matrix
"""
measured_calib_zero_centered = np.array(([0, 0, 0]))
for i in range(len(measured_calib)):
    ' find the centroid of the points '
    centered = measured_calib[i] - np.min(measured_calib, 0)
    measured_calib_zero_centered = np.vstack((measured_calib_zero_centered, centered
measured_calib_zero_centered = measured_calib_zero_centered[1:]

return measured_calib_zero_centered
```

0.1.3 Load the model set

It now behooves us to load the model set so we can begin our Kabsch computation. For this, we have them saved in a numpy array. Therefore, we will import numpy as well as associated and needed libraries necessary for our computation.

```
In [12]: model_points = np.array((
                                  [[-1755.87720294,   866.87898685,   283.0353811 ],
                                   [-1755.76266696,   866.8540598,    282.9782946 ],
                                   [-1758.9453555,    857.8363267,    296.13326449],
                                   [-1759.02853104,   865.92774874,   283.52951211],
                                   [-1777.42772925,   826.8692224,    293.38292356],
                                   [-1784.34737705,   836.7521396,    281.74652354],
                                   [-1777.77335781,   826.96331701,   292.88727602],
                                   [-1783.56649649,   836.45510137,   281.56390533],
                                   [-1783.53245361,   836.46510174,   281.54257437],
                                   [-1783.6947516,    836.55773878,   281.52364873],
                                   [-1783.58522171,   836.46979064,   281.55684051],
                                   [-1783.66230977,   836.54098015,   281.50709046],
                                   [-1783.52724697,   836.44927943,   281.56064662],
                                   [-1783.59681243,   836.52118858,   281.52347799],
                                   [-1783.44129296,   836.40624764,   281.5671847 ]]
         ))
```

0.1.4 Get the point set from the function above.

```
In [13]: point_set = measured_points_full()
```

0.2 Now, let us calculate the transformation as we described in our notes

```
In [23]: def Kabsch(P=None, Q=None, augment_Q=True, center=True):
             '''P and Q must be nX3. This rotation is accurate.
              Rotates points in P optimally to measured reference points in Q

              Params
              ======
              Q: Points to be rotated into
              augment_Q: Whether Q was recorded without the zero/home points embedded between su

              '''
             if not isinstance(P, np.ndarray) or not isinstance(Q, np.ndarray):
                 P, Q = prepro()

             # calculate the centroids
             if center:
                 'This only for computed old points'
                 q0 = np.mean(Q, 1)
                 p0 = np.mean(P, 1)

                 Q_ctr = Q - np.expand_dims(q0, 1)
                 P_ctr = P - np.expand_dims(p0, 1)
             else:
                 Q_ctr, P_ctr = Q, P
```

```
    # add the zero points to preconputed control points
    if augment_Q:
        'This only for computed old points'
        Q_aug = np.array(([[0,0,0]]))
        for i in range(Q_ctr.shape[0]-1):
                Q_aug = np.append(Q_aug, np.expand_dims(Q_ctr[i+1], 0),0)
                Q_aug=  np.append(Q_aug, np.expand_dims(Q_ctr[0], 0),0)
        Q_ctr = Q_aug[1:]


    Hmat = P_ctr.T@Q_ctr
    U, S, V = LA.svd(Hmat)
    d = np.sign(np.linalg.det(V@U.T))
    M = np.eye(3); M[-1][-1] =d
    opt_rot = V@M@U.T
    opt_trans = Q_ctr.T- opt_rot@P.T

    return opt_rot, np.mean(opt_trans, 1)
```

0.2.1 Test the algorithm

Remember that we are rotating the points in `point_set` into `model_points`. So we would go ahead and call the Kabsch function above as follows:

```
In [24]: Rot, Trans = Kabsch(model_points, point_set, augment_Q=False, center=False)

In [25]: print(Rot)

[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]


In [26]: print(Trans)

[1775.85125374 -842.66314863 -284.40256961]
```

Homework 20. For the following model points P and measured poinyts Q, compute the optimal rotation matrices for moving points Q into point P. For the three assignments below, report your results within a colab notebook, download the colab notebook as a pdf and upload on Latte.

1.

$$P = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \qquad Q = \begin{bmatrix} 0 & -1 & -1 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \tag{4.2.13}$$

2.

$$P = \begin{bmatrix} 3172.79468418 & 727.52462347 & 7122.70450243 \\ 165.28953155 & -3552.32467068 & -2045.15346584 \\ 5292.45250241 & -1748.52037006 & -6181.40300009 \\ 1893.07584225 & 5897.19719625 & 3130.41287776 \end{bmatrix}, \tag{4.2.14}$$

$$Q = \begin{bmatrix} 1774.11606309 & -4241.11341178 & 5259.04277742 \\ 6079.70499031 & -98.14197972 & -3442.0914569 \\ 813.07069876 & 3334.26289147 & -6112.55652513 \\ 1856.72080823 & 2328.86927901 & 6322.16611888 \end{bmatrix}$$

3. For a toy problem, measure the coordinates of an object in the world using your favorite measuring instrument (a 3D camera sensor, iPhone app (e.g. ArkIt), android app e.t.c.). Be sure to record the position of the object at multiple points in world coordinates and make sure that the physical locations of these points are known (these are your model points). Then compute the optimal rotation and translation between the model and measured points.

4.2.3 Corresponding Point Set Registration with Quaternions

While the Kabsch algorithm does yield an optimal solution for the rotation of two sets of points that correspond to one another, it leverages the orthonormal rotation matrix with positive determinant in its computations. This suffers from the non-uniqueness of solutions that arise from reflections. Using matrices straightforward is problematic because we need six nonlinear constraints to guarantee the orthonormality of the rotation matrix. To yield a least squares rotation, and translation, we will generally avoid singular value decomposition (SVD) methods in two and three dimensions since we generally do not want reflections. For $n > 3$ in any n-dimensional application, the SVD approach, based on the cross-covariance matrix of two point distributions, does generalize easily to n dimensions.

Let $\boldsymbol{t} = [t_x, t_y, t_z]^T$ denote the translation vector and $\boldsymbol{q}_R = [q_0, q_1, q_2, q_3]^T$ denote the unit quaternion. Suppose further that the complete registration set of vectors is $\boldsymbol{H} = [\boldsymbol{q}_R|\boldsymbol{t}]^T$. Now, let $D_l = \{\boldsymbol{d}_{l_i}\}$ be the measured set of points which we want to align with the model point set

$D_r = \{\boldsymbol{d}_{r_i}\}$, where the cardinality, N_l of D_l is same as that of D_r, N_r, and where each point $\boldsymbol{d}_{l_i}$ corresponds to point $\boldsymbol{d}_{r_i}$ with the same index. We are looking for a transformation of the form

$$D_r = a\boldsymbol{R}(D_l) + \boldsymbol{t} \tag{4.2.15}$$

from the left to the right coordinate system as shown in **??**, where a is a scale factor, and $\boldsymbol{t}$ is the translation vector offset. $R(D_l)$ denotes the rotated version of D_l. Since we do not expect to have a perfect data, it will be difficult to find a scale factor, a translation and a rotation so that the transformation equation is satisfied for every point. Thus, there will be a residual error,

$$\boldsymbol{e}_i = \boldsymbol{d}_{r,i} - a\boldsymbol{R}(\boldsymbol{d}_{l,i}) - \boldsymbol{t} \tag{4.2.16}$$

and the cost function will minimize the sum of squares is given as,

$$f(\boldsymbol{s}) = \min \|\boldsymbol{e}_i\|^2. \tag{4.2.17}$$

Finding Translation

We can find the translation, scale and finally rotation by systematically varying the total error.

Consider the centroids of the measured and point sets,

$$\bar{D}_l = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{d}_{l,i}, \qquad \bar{D}_r = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{d}_{r,i}, \tag{4.2.18}$$

so that the new coordinates are

$$\boldsymbol{d}'_{l,i} = \boldsymbol{d}_{l,i} - \bar{\boldsymbol{d}}_l, \qquad \boldsymbol{d}'_{r,i} = \boldsymbol{d}_{r,i} - \bar{\boldsymbol{d}}_r. \tag{4.2.19}$$

If we write $\boldsymbol{t}' = \boldsymbol{t} - \bar{\boldsymbol{t}} + a\boldsymbol{R}(\boldsymbol{d}_l)$, it follows that we can write the error as

$$\boldsymbol{e}_i = \boldsymbol{d}'_{r,i} - a\boldsymbol{R}(\boldsymbol{d}'_{l,i}) - \boldsymbol{t}' \tag{4.2.20}$$

and the sum of squares of errors becomes

$$\sum_{i=1}^{n} \|\boldsymbol{d}'_{r,i} - a\boldsymbol{R}(\boldsymbol{d}'_{l,i}) - \boldsymbol{t}'\|^2 \equiv \sum_{i=1}^{n} \|\boldsymbol{d}'_{r,i} - a\boldsymbol{R}(\boldsymbol{d}'_{l,i})\|^2 - 2\boldsymbol{t}'.\sum_{i=1}^{n}\left[\boldsymbol{d}'_{r,i} - a\boldsymbol{R}(\boldsymbol{d}'_{l,i})\right] + n\|\boldsymbol{t}'\|^2. \tag{4.2.21}$$

The middle term on the right hand side vanishes since the measurements are referred to the centroid and we are left with the first and the third terms. The first term is independent of $\boldsymbol{t}'$ and the last term cannot be negative given the squared norm. Thus, the total error to be minimized with $\boldsymbol{t}' = 0$ is

Optimal Translation

$$\boldsymbol{t} = \bar{\boldsymbol{d}}_r - a\ R(\bar{\boldsymbol{d}}_l) \tag{4.2.22}$$

In other words, *the translation is the difference between the right centroid and the scaled and rotated left centroid.*

We can now rewrite the error term from (4.2.20) as

$$\boldsymbol{e}_i = \boldsymbol{d}'_{r,i} - a\boldsymbol{R}(\boldsymbol{d}'_{l,i}) \tag{4.2.23}$$

since $\boldsymbol{t}' = 0$. So the total error to be minimized is

$$\sum_{i=1}^{n} \|\boldsymbol{d}'_{r,i} - a\boldsymbol{R}(\boldsymbol{d}'_{l,i})\|^2. \tag{4.2.24}$$

Finding Scale

Expanding (4.2.24), we find that

$$\sum_{i=1}^{n} \|\boldsymbol{d}'_{r,i}\|^2 - 2a\sum_{i=1}^{n} \boldsymbol{d}'_{r,i}.R(\boldsymbol{d}'_{l,i}) + s^2\sum_{i=1}^{n} \|\boldsymbol{d}'_{l,i}\|^2, \tag{4.2.25}$$

and since rotation preserves distances, $\|R(\boldsymbol{d}'_{l,i})\|^2 = \|\boldsymbol{d}'_{l,i}\|^2$, we can write the foregoing as $S_r - 2sD + s^2S_l$, where S_r and S_l are the sums of the squares of the measurement vectors (relative to their centroids), while D is the sum of the dot products of corresponding coordinates in the right system with the rotated coordinates in the left system. Completing the square in s, we find that

$$\left(a\sqrt{S_l} - D/\sqrt{S_l}\right)^2 + \left(S_rS_l - D^2\right)/S_l. \tag{4.2.26}$$

If we minimize with respect to scale a when the first term is 0 or $a = D/S_l$, we find that

$$s = \frac{\sum_{i=1}^{n} \boldsymbol{d}'_{r,i} \cdot R(\boldsymbol{d}'_{l,i})}{\sum_{i=1}^{n} \|\boldsymbol{d}'_{l,i}.\|^2.} \tag{4.2.27}$$

Finding rotation

To find the optimal rotation, we note that the cross-covariance matrix Σ_{lr} between the sets D_l and D_r is given by

$$\Sigma_{lr} = \frac{1}{N_l}\sum_{i=1}^{N_l} \left[(\boldsymbol{d}_{l,i} - \bar{\boldsymbol{d}}_l)(\boldsymbol{d}_{r,i} - \bar{\boldsymbol{d}}_r)^T\right] \tag{4.2.28}$$

$$= \frac{1}{N_l}\sum_{i=1}^{N_l} \left[\boldsymbol{d}_{l,i}\boldsymbol{d}_{r,i}^T\right] - \bar{\boldsymbol{d}}_l\bar{\boldsymbol{d}}_r^T. \tag{4.2.29}$$

The cyclic components of the skew symmetric matrix $Q_{ij} = \left(\Sigma_{lr} - \Sigma_{lr}^T\right)_{ij}$ are used to construct the column vector $\Delta = [Q_{23} \quad Q_{31} \quad Q_{12}]^T$, so that the vector is then used to form the symmetric matrix

$$Q(\Sigma_{lr}) = \begin{bmatrix} \text{tr}(\Sigma_{lr}) & \Delta^T \\ \Delta & \Sigma_{lr} + \Sigma_{lr}^T - \text{tr}(\Sigma_{lr})\boldsymbol{I}_3 \end{bmatrix} \tag{4.2.30}$$

where $\boldsymbol{I}_3$ is the 3×3 identity matrix and the unit eigenvector $\boldsymbol{q}_R = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T$ that corresponds to the maximum eigenvalue of $Q(\Sigma_{lr})$ is chosen as the optimal rotation.

4.3 Iterative Closest Point

The Iterative Closest Point (ICP) algorithm applies to the following sets of problems (i) sets of points, (ii) sets of line segments, (iii) sets of parametric curves, (iv) sets of implicit curves, (v) sets of triangles, (vi) sets of parametric surfaces, and (vii) sets of implicit surfaces. To properly describe the algorithm, we choose a data, P, which is to be moved or registered/positioned to best align with a "model" data X. It is best if the data and model shape are decomposed into a point set if they are not already in point set form. For triangles and line segments, we use their vertices and endpoints respectively; while for curves and surfaces, an approximation to the vertices and endpoints of triangles and lines are used. Suppose we denote, as before, the number of points in the data shape as N_p and N_x as the number of points, line segments, or triangles in the model shape. The distance metric d between an individual data point $\boldsymbol{p}$ and a model shape X will be denoted

$$d(\boldsymbol{p}, X) = \min_{\boldsymbol{x}(X)} \|\boldsymbol{x} - \boldsymbol{p}\|. \tag{4.3.1}$$

The closest point in X that yields the minimum distance is denoted $\boldsymbol{y}$ such that $d(\boldsymbol{p}, \boldsymbol{y}) = d(\boldsymbol{p}, X)$, where $\boldsymbol{y} \in X$.

Quiz 5. 1. What is the worst case asymptotic computation for the closest point in X and why?

2. What is the expected worst case computation time?

When the closest point computation from $\boldsymbol{p}$ to X is performed for each point P, that process is worst case $O(N_p, N_x)$. Let Y denote the resulting set of closest points, and $\mathcal{C}$ the closest point operator, $i.e.$

$$Y = \mathcal{C}(P, X). \tag{4.3.2}$$

For the resultant corresponding point set Y, the least squares registration can be computed as

$$(\boldsymbol{q}, d) = \mathcal{Q}\left(P, Y\right). \tag{4.3.3}$$

and the positions of the data shape point set are] then updated via $P = \boldsymbol{q}(P)$.

Algorithm 1 ICP Algorithm

1: Given point set P with N_p points $\{\boldsymbol{p}_i\}$ from the data shape and the model shape X with N_x supporting geometric primitives: points, lines, or triangles
2: Start the iteration with P_0 set to P, $\boldsymbol{q}_0 = [1, 0, 0, 0, 0, 0, 0]^T$ and $k = 0$ and define the registration vector relative to the initial data set P_0 so that the final registration denotes the complete transformation.
3: Given a mean-square error with preset threshold $\tau > 0$, and a desired registration accuracy, d
4: **while** $\tau > d_k - d_{k+1}$ **do**
5: Compute the closest points, $Y_k = \mathcal{C}(P_k, X)$ (cost: $O(N_o, N_x)$, worst-case: $O(N_p \log N_x)$ average).
6: Compute the registration: $(\boldsymbol{q}_k, d_k) = \mathcal{Q}(P_0, Y_k)$ (cost: $O(N_p)$).
7: Apply the registration, $P_{k+1} = \boldsymbol{q}(P_0)$ (cost: $O(N_p)$).
8: **end while**

CHAPTER 5

STATE ESTIMATION

The next few topics in this course shall involve the quantification of uncertainty in order to enable a robot navigate, move, or understand its environment via visual or audio sensors. In order to do justice to this topic, we shall soon find out that the concept of putting a value or percentage on how sure we are about a robot's environment shall be very helpful in effective control of our robots. Thus the concept of probability shall greatly aid us in quantifying uncertainty. Even so, we introduce the concept of states, grounded in a mathematical theory that allows the engineer to implement a state through discrete-time systems (since we assume that most implementations shall be done on digital computers). By the *state* of a system, we shall loosely mean "those variables that provide a complete representation of the internal condition or status of the system at a given time instant." In this sentiment, the states of a motor system may mean currents that flow through the inductive coils, the position and speed of its motor shaft, or the voltage across the coils of a solenoid valve. The states of a military power may include the number of its aircraft carriers, the size and horsepower of its nuclear submarines, the number of enlisted servicemen in its forces e.t.c. For a biological system, the states might include blood sugar levels, heart and respiration rates, or body temperature.

Robot systems may include mobile platforms for extraterrestrial navigation, robotics arms in assembly lines, autonomous cars, or actuated surgical devices that assist surgeons. Our goal is to treat uncertainty. Uncertainty occurs if the robot lacks important information that hinders it from carrying out assigned tasks. We may classify this uncertainty into five different factors, viz.,

1. **Environments.** The physical world is inherently unpredictable. While the degree of uncertainty in well-structured environments such as assembly lines is small, environments such as highways and private homes are highly dynamic and unpredictable.

2. **Sensors.** Most sensors have limitations in their perceptual ability arising from noise and the range and the resolution of the sensors. For example, environmental disturbances, weather, lighting conditions limit the information that can be extracted from sensors. Secondly, as to range and resolution, cameras cannot see through walls despite the perceptual range that the spatial resolution of the camera is limited.

3. **Models.** In general, models are at best an approximation or a mathematical representation or abstraction of the physical world. As such, model errors are a source of uncertainty that need to be incorporated in modeling robotics problems.

4. **Computation.** Being real-time systems, robots require a lot of computation in order to be able to achieve timely-response through sacrificing accuracy.

We will estimate states as they shall represent latent or underlying variables that influence the physical or chemical or financial properties of the system. And in motivating the study of a system's state, we can resolve to many weapons in our estimation arsenal which may include linear state filtering (the simple Kalman filter), nonlinear state filters (the extended Kalman filter,

unscented Kalman filter e.t.c.), Bayesian estimation, and *frequentist/classical* estimation approaches. In general, state estimation is an important topic to the engineer because:

- We may need to implement a feedback controller in order to regulate a system's behavior. If the application was for a surgeon to regulate blood pH levels, we may need to estimate the system's state. Or if the challenge is to adequately position a patient's head to a position in 3D space during cancer stereotactic radiosurgery, we may need to estimate the position and orientation of the patient's head and neck in the inertial frame.
- If the states in question are curious enough, we may want to measure these states to understand the faults tolerance of the system in order to perform a good fault identification and prognosis. For example, we might want to estimate the internal states of an aircraft system in flight such that if an aircraft engine fails during flight, we can safely monitor system states in real-time in order to determine how long we can continue flying the aircraft or if we should quickly find a near-by airport where we could land the aircraft for maintenance.

In our treatment, therefore, we shall give a brief introduction to linear systems theory, touch upon standard linear filters and then proceed to treat probability theory before we treat nonlinear systems, and decision-making.

5.1 Linear Systems

State-space systems are very important in engineering systems because they allow us (i) to gain insight into the characteristics of the system, (ii) be able to predict future behaviors of the system, (iii) identify the controllable and observable states of the system. The mathematical model of the process allows us to infer the information about the process. State-space models can be classified into linear and nonlinear systems. While most real-world systems are nonlinear, the tools that exist for analyzing and synthesizing nonlinear systems are well-developed and sophisticated that most nonlinear systems can be approximated by linear systems in order to exercise good control and estimation for real-world applications.

A continuous-time, deterministic linear system can be described by the equations

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{5.1.1}$$

where x is the *state vector* in $\mathbb{R}^n \times 1$, u is the *control vector* in $\mathbb{R}^p \times 1$, and y is an $\mathbb{R}^n \times 1$ vector. Matrices A, B, and C are respectively $n \times n$, $n \times p$ and $n \times 1$ in dimension. The matrix A is often called the system matrix, B the input or control matrix, while C is often called the output matrix. A, B, and C can be time-varying matrices, in which case the system is linear. Otherwise, the solution to the linear system of equations above is

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau \tag{5.1.2}$$

$$y(t) = Cx(t) \tag{5.1.3}$$

where t_0 is the initial time of the system. If the input control law is zero, then we have a *non-autonomous system* i.e.

$$x(t) = e^{A(t-t_0)}x(t_0) \tag{5.1.4}$$

and because of this, e^{AT} is called the state-transition matrix $i.e.$ it describes how the state moves between transitions at different times regardless of external inputs. At $t = t_0$, we have that

$$e^{A\,0} = I, \tag{5.1.5}$$

which is similar to the scalar exponential of a zero. What happens if x is an n-element vector? The solution in (5.1.3) still remains valid but we must note that the exponential of the matrix becomes interpreted as

$$\begin{aligned} e^{At} &= \sum_{j=0}^{\infty} \frac{(At)^j}{j!} \\ &= \mathcal{L}^{-1}\left[sI - A\right]^{-1} = Qe^{\hat{A}t}Q^{-1} \end{aligned} \tag{5.1.6}$$

where the symbol $\mathcal{L}^{-1}$ is the symbol for the inverse Laplace transform and "s" is the Laplace operator. We see that A must be square in order for e^{At} to exist. Q contains the eigenvectors of A and $\hat{A}$ are the Jordan form of A.

Quiz 6. Write a note about the Jordan form. Also, explain how it can be determined from (5.1.6).

Quiz 7. Does the matrix A commute with its exponential i.e. does $A\,e^{At} = e^{At}\,A$?

The matrix $\hat{A}$ is often diagonal, so that case $e^{\hat{A}t}$ can be computed as

$$\hat{A} = \begin{bmatrix} \hat{A}_{11} & 0 & \dots & 0 \\ 0 & \hat{A}_{22} & \dots 0 & \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \hat{A}_{nn} \end{bmatrix} \quad e^{\hat{A}t} = \begin{bmatrix} e^{\hat{A}_{11}} & 0 & \dots & 0 \\ 0 & e^{\hat{A}_{22}} & \dots 0 & \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & e^{\hat{A}_{nn}} \end{bmatrix} \tag{5.1.7}$$

From (5.1.6), we can write

$$\left[e^{At}\right]^{-1} = e^{-At} = Qe^{-\hat{A}t}Q^{-1} \tag{5.1.8}$$

Since A and $-A$ have eigenvalues that are negative of each other, e^{At} is always invertible.

Example 2. Suppose we are controlling angular heading of a mobile robot (for example, using voltage applied to its wheels' rotor windings in order to generate command velocity along the x, y, and z heading, i.e. θ, ω and α respectively). The derivative of the angular velocity vector can be written as

$$\begin{aligned} \dot{\theta} &= \omega + \alpha + 3.5\omega_1 + 6\theta_2 \\ \dot{\omega} &= u + 0.1\theta + 2.5\alpha + \omega_1 + \omega_2^2 \\ \dot{\alpha} &= \theta_1 + 2u \end{aligned} \tag{5.1.9}$$

The scalars ω_1, ω_2, θ_1 and θ_2 are acceleration noise terms such as gear backlash, friction, and modeling errors. If our measurement consists of the θ and ω states, it follows that we can write the state space equation as

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 2.5 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} u + \begin{bmatrix} 3.5\omega_1 + 6\theta_2 \\ \omega_1 + \omega_2^2 \\ \theta_1 \end{bmatrix}$$
$$y = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} \theta \\ \omega \\ \alpha \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{5.1.10}$$

where $v = [v_x, v_y, v_z]^T$ is the linear velocity vector for the robot.

Example 3. Suppose that

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{5.1.11}$$

It follows that

$$\begin{aligned} e^{At} &= \sum_{j=0}^{\infty} \frac{(At)^j}{j!} \\ &= (At)^0 + (At)^1 + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \dots \\ &= I + At \end{aligned} \tag{5.1.12}$$

where the last term follows from the fact that $A^k = 0$ for $k > 1$ so that

$$e^{At} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & t \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \tag{5.1.13}$$

Using the expression for the inverse Laplace transform earlier, we have

$$\begin{aligned} e^{At} &= \mathcal{L}^{-1}\left[(sI - A)^{-1}\right] \\ &= \mathcal{L}^{-1}\left(\begin{bmatrix} s & -1 \\ 0 & s \end{bmatrix}^{-1}\right) \\ &= \mathcal{L}^{-1}\begin{bmatrix} 1/s & 1/s^2 \\ 0 & 1/s \end{bmatrix} \\ &= \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{5.1.14}$$

Homework 21. Find the eigendata of the matrix A in (5.1.14). Then determine the following terms using the eigenvector and eigenvalue that you may find: $\hat{A}, Q$ and e^{At}.

Homework 22. Produce a one-page report on a control system transfer function.

5.2 State Space Standard Forms

For a linear system, there are many possible state space models that can result in the same *transfer function dynamics*. Therefore, standardizing state space model structures is relevant for solving problems in a conformal way. For consider the following input-output system's *linear difference equation*[1]

$$y_n + a_1 y_{n-1} + \ldots + a_{n-1} y_1 + a_n y = b_0 u_n + b_1 u_{n-1} + \ldots + b_{n-1} u_1 + b_n u \tag{5.2.1}$$

with u and y serving respectively as the input and output, and y_n serving as the nth derivative of y with respect to time. If we take the Laplace transform of both sides, we have

$$Y(s)\left(s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n\right) = U(s)\left(b_0 s^n + b_1 s^{n-1} + \ldots + b_{n-1} s + b_n\right) \tag{5.2.2}$$

so that the transfer function from the input u to the output y can be written as

$$\frac{Y(s)}{U(s)} = \frac{b_0 s^n + b_1 s^{n-1} + \ldots + b_{n-1} s + b_n}{s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n} \tag{5.2.3}$$

5.2.1 Companion form

In *companion form* representation, the coefficients of the transfer function in (5.2.3) are arranged along its far rows or columns. An example would be

$$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & 0 & \cdots & 0 & -a_2 \\ 0 & 0 & 1 & \cdots & 0 & -a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -a_{n-1} \end{bmatrix} \tag{5.2.4}$$

or

$$\begin{bmatrix} -a_{n-1} & -a_{n-2} & -a_{n-3} & \cdots & -a_1 & -a_0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{5.2.5}$$

In general, we use the convenient *observable* and *controllable* canonical forms in control theory. They are exactly the transpose of one another and using either for control design simplifies the system structure so that it can be readily manipulated for a desired control.

---

[1]

Quiz 8. What is the difference between a *linear difference equation* and a *linear ordinary differential equation?*

5.2.2 Modal Form

The modal form is the dual to the companion form. In the modal form, the state matrix is a diagonal matrix with non-repeating eigenvalues such that the control has a unitary influence on each eigenspace, and the output is a linear combination of the contributions from the eigenspaces. That is,

$$A = \begin{bmatrix} -p_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -p_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -p_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -p_n \end{bmatrix} \tag{5.2.6a}$$

$$B = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \quad C = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} \tag{5.2.6b}$$

Homework 23. Write out the solution to eq. 24 in modal form.

5.2.3 Controllable Canonical Form

When we want to design a controller that leverages the full state of the system (assuming this is known), often the *controllable canonical form* will come in handy. It is expressed as follows:

$$A = \begin{bmatrix} -a_1 & -a_2 & -a_3 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{5.2.7a}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} b_1 & b_2 & b_3 & \cdots & b_n \end{bmatrix} \quad D = \begin{bmatrix} b_0 \end{bmatrix} \tag{5.2.7b}$$

Example 4. For the system

$$\frac{Y(s)}{U(s)} = \frac{5s^2 - s + 8}{s^2 + 4s - 2} \tag{5.2.8}$$

we can realize the state space representation in canonical form as follows:

1. Observe that n from (5.2.3) is 3, *i.e.* the highest s exponent in the given transfer function.

2. It follows that we have $a_0 = -2$ and $a_1 = 4$; and $b_0 = 5,\ b_1 = -1,\ \ b_2 = 8$, so that we can write the state space model as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 1 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} u_1 & u_2 \end{bmatrix}$$
$$y = \begin{bmatrix} -1 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \boldsymbol{u} \tag{5.2.9}$$

Homework 24. Derive the companion form for the system:

$$\frac{Y(s)}{U(s)} = \frac{3s^2 - 2s + 1}{s^2 - 8s + 5} \tag{5.2.10}$$

The controllable canonical form is helpful in when using the pole placement method for controller design. However, the system's transformation to companion form is based on the controllability matrix which is almost always numerically singular for mid-range orders. It should be avoided for computation when possible.

5.2.4 Observable Canonical Form

In observable canonical form, the transfer function coefficients of (5.2.3) are written in the rightmost column of the A matrix similar to the companion canonical form but the B matrix takes a different form. It is given as follows:

$$A = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & 0 & \cdots & 0 & -a_2 \\ 0 & 0 & 1 & \cdots & 0 & -a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -a_{n-1} \end{bmatrix} \tag{5.2.11}$$

$$B = \begin{bmatrix} b_n - a_n b_0 \\ b_{n-1} - a_{n-1} b_0 \\ b_{n-2} - a_{n-2} b_0 \\ \vdots \\ b_1 - a_1 b_0 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix}, \quad D = b_0 \tag{5.2.12}$$

This observable canonical form is ill-conditioned for most state-space computation. It should be avoided for computation when possible as its controllability matrix is almost always numerically singular for mid-range orders.

The observable and controllable canonical forms' matrices are respectively transposes of one another.

Homework 25. Transform the exercise of 24 to observable canonical form.

5.3 Nonlinear Systems

All the world is a nonlinear system. He linearized to the right. He linearized to the left. Till nothing was right. And nothing was left. – Stephen Billings.

Our treatment of dynamical so far has involved linear systems. These are optimistic models of the real world as in the reality, nothing is really linear. In general, a nonlinear system is a system which is not linear $i.e.$, does not satisfy the *principle of superposition*. Even a simple resistor exhibits nonlinearity. However, we utilize Ohm's law in approximating the dynamics of a resistor. This is because the equation is valid over a wide enough operating range. In this light, while we may say linear systems do not exist in real life, linear systems are a useful tool for describing nonlinear systems. We will write a general nonlinear system with the equation

$$\begin{aligned} \dot{x} &= f(x, u, w) \\ y &= h(x, v) \end{aligned} \tag{5.3.1}$$

where $f(\cdot)$ and $h(\cdot)$ are arbitrary vector valued functions, w denotes the process noise, and v denotes the measurement noise. We have a *time-varying* system if $f(\cdot)$ and $h(\cdot)$ are explicit functions of t, otherwise, the system is termed *time-invariant*. Suppose that

$$f(x, u, w) = Ax + Bu + w; \text{ and} \tag{5.3.2}$$

$$h(x, v) = Hx + v, \tag{5.3.3}$$

then the system is linear. Otherwise, the system is nonlinear.

Often, we will need to linearize a nonlinear system in order to properly analyze its stability properties or synthesize its parameters for a particular control application. Suppose we have a nonlinear vector function $f(\cdot)$ of a scalar x, we can expand $f(x)$ in a Taylor series around some nominal operating point, $x = \bar{x}$ i.e.

$$f(x) = f(\bar{x}) + \frac{\partial f}{\partial x}\bigg|_{\bar{x}} \tilde{x} + \frac{1}{2!}\frac{\partial^2 f}{\partial x^2}\bigg|_{\bar{x}} \tilde{x}^2 + \frac{1}{3!}\frac{\partial^3 f}{\partial x^3}\bigg|_{\bar{x}} \tilde{x}^3 + \dots \tag{5.3.4}$$

where $\tilde{x} = x - \bar{x}$. For a 2×1 vector x, we can write $f(x)$ as follows:

$$\begin{aligned} f(x) = f(\bar{x}) + \frac{\partial f}{\partial x_1}\bigg|_{\bar{x}} \tilde{x}_1 + \frac{\partial f}{\partial x_2}\bigg|_{\bar{x}} \tilde{x}_2 + \frac{1}{2!}\left(\frac{\partial^2 f}{\partial x_1^2}\bigg|_{\bar{x}} \tilde{x}_1^2 + \frac{\partial^2 f}{\partial x_2^2}\bigg|_{\bar{x}} \tilde{x}_2^2 + 2\frac{\partial^2 f}{\partial x_1 x_2}\bigg|_{\bar{x}} \tilde{x}_1\tilde{x}_2\right) + \\ \frac{1}{3!}\left(\frac{\partial^3 f}{\partial x_1^3}\bigg|_{\bar{x}} \tilde{x}_1^3 + \frac{\partial^3 f}{\partial x_2^3}\bigg|_{\bar{x}} \tilde{x}_2^3 + 3\frac{\partial^3 f}{\partial x_1^2 x_2}\bigg|_{\bar{x}} \tilde{x}_1^2\tilde{x}_2 + 3\frac{\partial^3 f}{\partial x_1 x_2^2}\bigg|_{\bar{x}} \tilde{x}_1\tilde{x}_2^2\right) + \dots \end{aligned} \tag{5.3.5}$$

which can be compactly written as

$$\begin{aligned} f(x) = f(\tilde{x}) + \left(\tilde{x}_1\frac{\partial}{\partial x_1} + \tilde{x}_2\frac{\partial}{\partial x_2}\right) f\bigg|_{\bar{x}} + \frac{1}{2!}\left(\tilde{x}_1\frac{\partial}{\partial x_1} + \tilde{x}_2\frac{\partial}{\partial x_2}\right)^2 f\bigg|_{\bar{x}} + \\ \frac{1}{3!}\left(\tilde{x}_1\frac{\partial}{\partial x_1} + \tilde{x}_2\frac{\partial}{\partial x_2}\right)^3 f\bigg|_{\bar{x}} + \dots \end{aligned} \tag{5.3.6}$$

And when n is an $n \times 1$ vector, the vector $f(x)$, expanded in a Taylor series becomes

$$f(x) = f(\tilde{x}) + \left(\tilde{x}_1 \frac{\partial}{\partial x_1} + \ldots + \tilde{x}_n \frac{\partial}{\partial x_n}\right) f\Big|_{\bar{x}} + \frac{1}{2!}\left(\tilde{x}_1 \frac{\partial}{\partial x_1} + \ldots + \tilde{x}_n \frac{\partial}{\partial x_n}\right)^2 f\Big|_{\bar{x}} + \frac{1}{3!}\left(\tilde{x}_1 \frac{\partial}{\partial x_1} + \ldots + \tilde{x}_n \frac{\partial}{\partial x_n}\right)^3 f\Big|_{\bar{x}} + \ldots \tag{5.3.7}$$

Suppose we define the operation $D^k_{\tilde{x}} f$ as

$$D^k_{\tilde{x}} = \left(\sum_{i=1}^{n} \tilde{x}_i \frac{\partial}{\partial x_i}\right)^k f(x)\Big|_{\bar{x}} \tag{5.3.8}$$

so that we can define $f(x)$ in Taylor series form as

$$f(x) = f(\bar{x}) + D_{\tilde{x}} f + \frac{1}{2!} D^2_{\tilde{x}} f + \frac{1}{3!} D^3_{\tilde{x}} f + \ldots \tag{5.3.9}$$

$$= f(\bar{x}) + D_{\tilde{x}} f + o(\delta). \tag{5.3.10}$$

If $f(x)$ is "sufficiently smooth", it is not far fetched to see that the above equation turns to

$$f(x) \approx f(\bar{x}) + D_{\tilde{x}} f \approx f(\bar{x}) + \frac{\partial f}{\partial x}|_{\bar{x}} \tilde{x} \approx f(\bar{x}) + A\tilde{x}. \tag{5.3.11}$$

since $o(\delta)$ implies that higher order terms satisfy $\lim_{\delta \to 0} \frac{o(\delta)}{\delta} = 0$, and $A = \frac{\partial f}{\partial x}\Big|_{\bar{x}}$.

Recall (5.3.1), if we choose a nominal operating point $(\bar{x}, \bar{u}, \bar{w})$ and carry out a Taylor series expansion about this nominal point of the nonlinear system of equations, for the state part, we have

$$\begin{aligned}
\dot{x} &= f(x, u, w) \\
&\approx f(\bar{x}, \bar{u}, \bar{w}) + \frac{\partial f}{\partial x}\Big|_{(\bar{x},\bar{u},\bar{w})} (x - \bar{x}) + \frac{\partial f}{\partial u}\Big|_{(\bar{x},\bar{u},\bar{w})} (u - \bar{u}) + \frac{\partial f}{\partial w}\Big|_{(\bar{x},\bar{u},\bar{w})} (w - \bar{w}) + o(\delta) \qquad (5.3.12) \\
&= f(\bar{x}, \bar{u}, \bar{w}) + \frac{\partial f}{\partial x}\Big|_{(\bar{x},\bar{u},\bar{w})} \tilde{x} + \frac{\partial f}{\partial u}\Big|_{(\bar{x},\bar{u},\bar{w})} \tilde{u} + \frac{\partial f}{\partial w}\Big|_{(\bar{x},\bar{u},\bar{w})} \tilde{w} + o(\delta) \\
&= \dot{\bar{x}} + A\tilde{x} + B\tilde{u} + L\tilde{w}
\end{aligned}$$

Since $\tilde{w}$ is a noise term, it suffices that $\tilde{w} = \bar{w} = w$ so that we can write

$$\dot{x} - \dot{\bar{x}} = A\tilde{x} + B\tilde{u} + Lw \qquad \text{or}$$

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u} + Lw. \tag{5.3.13}$$

In other words, we have a linear equation for the deviations of the nonlinear system from the nominal system. It is therefore reason that as long as the deviations are minute enough, the linearization will be valid and the linear equation of (5.3.13) will describe the nonlinear system (5.3.1) well enough.

In a similar vein, the measurement equation from (5.3.1) will be approximated with the Taylor series expansion about the nominal operating point $(\bar{x}, \bar{u})$ as follows:

$$\begin{aligned} y &= h(x,u) \\ &\approx h(\bar{x},\bar{u}) + \left.\frac{\partial h}{\partial x}\right|_{(\bar{x},\bar{\nu})} \tilde{x} + \left.\frac{\partial h}{\partial \nu}\right|_{(\bar{x},\bar{\nu})} \tilde{\nu} + o(\delta) \qquad (5.3.14) \\ &= \bar{y} + C\tilde{x} + D\tilde{\nu} \qquad \text{or} \\ \tilde{y} &= C\tilde{x} + D\tilde{\nu}. \qquad (5.3.15) \end{aligned}$$

It follows that we can "solve" a nonlinear control problem by finding linear operating regions whereby we can solve the control problem, after which we can obtain locally linear solutions for the nonlinear control problem.

Example 5. Consider the longitudinal flight control of a hypersonic aircraft cruising at a Mach number of 15 at an altitude of $110,000ft$. The dynamic equations are (Wang and Stengel, 2000)

$$\dot{V} = (T\cos\alpha - D)/m - \mu\sin\gamma/r^2 \qquad (5.3.16a)$$
$$\dot{\gamma} = (L + T\sin\alpha)/mV - \left[(\mu - V^2 r)\cos\gamma\right]/(Vr^2) \qquad (5.3.16b)$$
$$\dot{h} = V\sin\gamma \qquad (5.3.16c)$$
$$\dot{\alpha} = q - \dot{\gamma} \qquad (5.3.16d)$$
$$\dot{q} = M_{yy}/I_{yy} \qquad (5.3.16e)$$

where

$$L = \frac{1}{2}\rho V^2 S C_L \qquad (5.3.17a)$$
$$D = \frac{1}{2}\rho V^2 S C_D \qquad (5.3.17b)$$
$$T = \frac{1}{2}\rho V^2 S C_T. \qquad (5.3.17c)$$

Here, α is the angle of attack, γ is the flight path angle, rad, r is the radial distance from the center of the Earth, $20,903,500$ft, C_T is the thrust coefficient, C_D is the drag coefficient, C_L is the lift coefficient, L is the lift, D is the drag in lbf, h is the altitude, T is the thrust in lbf, V is the velocity in ft/sec, m is the mass, 9375 slugs, q is the pitch rate in rad/sec, S is the reference area, $3603ft^2$, I_{yy} is the moment of inertia, 7×10^6 slug-ft^2, M_{yy} is the pitching moment in lbf-ft, μ is the gravitational constant, $1.39 \times 10^16 ft^3/s^2$.

We can write the state space vector of the dynamics as follows:

$$\dot{x} = \begin{bmatrix} \dot{x}_1 & \dot{x}_2 & \dot{x}_3 & \dot{x}_4 & \dot{x}_5 \end{bmatrix} = \begin{bmatrix} \dot{V} & \dot{\gamma} & \dot{h} & \dot{\alpha} & \dot{q} \end{bmatrix}$$

.

So that the *nonlinear* dynamics of the hypersonic aircraft at the specified cruising altitude of $110,000$ ft and Mach number 15 becomes

$$\dot{x}_1 = \frac{1}{2m}\rho S\left(C_T\cos x_4 - C_D\right)x_1^2 - \frac{\mu}{r^2}\sin x_2 \tag{5.3.18a}$$

$$\dot{x}_2 = \left(\frac{1}{2m}\rho S C_L - \frac{\mu}{x_1^2 r^2}\right)x_1 + \frac{x_1}{r}\cos x_2 + \frac{1}{2m}\rho S C_D \sin x_4 \tag{5.3.18b}$$

$$\dot{x}_3 = x_1 \sin x_2 \tag{5.3.18c}$$

$$\dot{x}_4 = -\left(\frac{1}{2m}\rho S C_L - \frac{\mu}{x_1^2 r^2}\right)x_1 - \frac{x_1}{r}\cos x_2 - \frac{1}{2m}\rho S C_D \sin x_4 + x_5 \tag{5.3.18d}$$

$$\dot{x}_5 = M_{yy}/I_{yy} \tag{5.3.18e}$$

Following our earlier argument, we proceed to linearize the dynamics by first finding the Jacobian with respect to the state transition matrix, x:

$$\begin{aligned} A &= \frac{\partial f}{\partial x} \\ &= \begin{bmatrix} \frac{1}{m}\rho S\left(C_T\cos x_4 - C_D\right)x_1 & -\frac{\mu}{r^2}\cos x_2 & 0 & -\frac{1}{2m}\rho S C_T \sin x_4 x_1^2 & 0 \\ \frac{1}{2m}\rho S C_L - \frac{\mu}{r^2 x_1^2} + \frac{1}{r}\cos x_2 & -\frac{x_1}{r}\sin x_2 & 0 & \frac{1}{2m}\rho S C_D \cos x_4 & 0 \\ \sin x_2 & x_1\cos x_2 & 0 & 0 & 0 \\ -\frac{1}{2m}\rho S C_L - \frac{1}{r}\cos x_2 - \frac{\mu}{r^2 x_1^2} & \frac{x_1}{r}\sin x_2 & 0 & -\frac{1}{2m}\rho S C_D \cos x_4 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{5.3.19}$$

Similarly, the input matrix can be obtained by finding the Jacobian with respect to the lift, L, drag D, and thrust, T, are

$$B = \frac{\partial f}{\partial u} \tag{5.3.20}$$

$$= \begin{bmatrix} 0 & -1/m & 0 \\ 1/mV & 0 & \frac{\sin\alpha}{mV} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.3.21}$$

so that the linear system

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u} \tag{5.3.22}$$

approximately describes the nonlinear hypersonic aircraft's deviation from its nominal value $\bar{x}$. We can simulate the lift, drag and thrust with the following nominal control values: $L = \sin 2\pi t$, $D = \cos 2\pi t$, $T = 2L - D$ to find the nominal state trajectory $\bar{x}$.

REFERENCES

Besl, P. J. and N. D. McKay (1992). Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, Volume 1611, pp. 586–606. International Society for Optics and Photonics.

Horn, B. K. (1987). Closed-form solution of absolute orientation using unit quaternions. *Josa a 4*(4), 629–642.

Kabsch, W. (1978). A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A 34*(5), 827–828.

Wang, Q. and R. F. Stengel (2000). Robust nonlinear control of a hypersonic aircraft. *Journal of guidance, control, and dynamics 23*(4), 577–585. 47

Weisstein, E. W. Homeomorphic.