EE546
Robot Manipulators

Blake Hannaford and Jacob Rosen

April 19, 2012

Contents

Chapter 1

Introduction

This text has evolved from lecture notes for EE543 at the University of Washington. It will be evident to the reader that it is still very much a work in progress. It owes a great deal to the book "Introduction to Robotics, Mechanics and Control," by John Craig.

As you read this book, please report typos or suggestions (by page and line number) to
`https://catalyst.uw.edu/gopost/board/blake/20007/`

1.1 Background and Motivational Examples

1.2 Definitions

A robot manipulator is a chain of *links* connected by *joints*.

1.2.1 Degree of Freedom

A degree of freedom (DOF) is a motion which can be completely described by a scalar and its time derivatives.

1.2.2 Joints

Joints constrain the relative position of two links in all but one degree of freedom.

Rotary joints permit revolute motion about a line in space refered to as the motion axis.

Prismatic joints permit translational motion along a line in space refered to as the motion axis.

Three key components of joints are actuators, sensors, and bearings.

Actuators do work on the joint through the application of force or torque. Most robot manipulators use electric motors as actuators but hydraulic, pneumatic, and other more exotic actuators can also be used.

Sensors measure the motion of the degree of freedom of the joint and send this information to a control system.

Bearings are devices which restrict motion in all but one degree of freedom. They act as rigid structural elements in the constrained directions and minimize friction in the degree of freedom.

1.2.3 End Effector

The end effector of a robot is a device attached to a link in the chain (typically at the end of a serial chain) designed to accomplish a task. Typical end effectors include grippers, spray guns, and tools.

1.2.4 Joint Space

1.2.5 Task Space

1.2.6 Workspace

1.3 Fundamental Problems in Manipulator Control

This book will use a large set of mathematical tools to model and analyze robot manipulators. In learning all of these tools, it will be useful to keep in mind that all of them are used to answer fundamental and straightforward questions which must be answered in order to use robot manpipulators.

- Description of a positioning task

  Q: How do we specify the position and orientation of robot arms and objects?

  A: Frames, Homogeneous Transformations, and joint-space teaching.

- Forward kinematics

  Q: What is the position and orientation (configuration) of a robot end effector if joint positions are known?

  A: A linear transformation (4x4 matrix) which is a function of the joint positions (angles, displacements) and specifies end effector configuration in the base frame. This matrix can also map a point in the end effector frame to its representation in the base frame.

- Inverse kinematics

  Q: What joint positions θ, do I need to achieve a given end effector configuration?

  A: Form an equation by setting the 4x4 matrix of the foward kinematics problem equal to one specifiing the *desired* position and orientation. Then solve this equation for the joint positions. Be sure to watch out for existence of solutions and multiple solutions.

- Velocity transformation

  Q: What joint velocities, $\dot{\theta}$, do I need to achieve a given velocity of the end effector (both linear and angular velocity) in all three dimensions of a convenient reference frame?

  A: Compute the "Jacobian Matrix", the matrix derivative, of the forward kinematic equations and invert it. Watch out for singular matrix in which case the inverse does not exist.

$$\dot{\theta} = J^{-1}\dot{x}$$

- Force transformation

  Q: What joint torques, τ, correspond to a given set of end effector forces and torques, F?

  A: $\tau = J^T F$

- Inverse dynamics

  Q: For a given motion, $\ddot{\theta}, \dot{\theta}, \theta(t)$, what are the required joint torques?

  A: We can derive a dynamic equation

$$\tau(t) = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + g(\theta)$$

  by using the Recursive Newton-Euler or Lagrangian methods.

- Position Control

  Q: How can we command joint torques to achieve a desired end effector trajectory?

  A: PID control, Feed-forward dynamic control (the method of computed torques).

- Force Control

  Q: How can we command joint torques to achieve a set of forces and torques at the end effector?

  A: Open loop we can use $\tau = J^T F$. However because of friction and other losses in many typical manipulators we must use force sensing at the end effector and closed loop force control.

- Trajectory Generation

  Q: If we know a trajectory we want to follow, how do we generate a time-function which is a quick but smooth and attainable trajectory between two configurations "A" and "B"?

  A: Methods include polynomial splines and trapezoidal velocity profiles.

- Motion Planning

  Q: How do we decide the path to take between two configurations "A" and "B"?

  A: Without considering obstacles, the most common approach is straight line interpolation. When obstacles and joint limits must be considered this becomes the AI motion planning problem.

- Sensor Based Control

  Q: How can I make use of non-joint sensor information (e.g. vision, proximity, etc.) to control robot motion (for example, to follow 1cm above an unknown surface).

  A: Robot vision, vision servoing, sensor data fusion, world modeling, operational space control.

- Telemanipulation

  Q: How can a human operator control a remote manipulator in a "natural" and productive way to achieve a task goal?

  A: Generalized bi-lateral teleoperation.

1.4 Summary of Notation

This explains the notation of chapter 1.

Chapter 2

Coordinate Transformations and Rigid Body Motions

To properly control the manipulation of objects with robots, we must have precise and useful ways to describe their position and orientation. In this chapter we start out with some definitions, and then develop the mathematical tools for this surprisingly non-straightforward task. Finally we will combine position and orientation into a clever 4x4 matrix which represents both in one object.

2.1 Problem Statement and Learning Objectives

Problem Statement The problem this chapter addresses is to represent the position and orientation of rigid bodies and frames of reference in useful ways.

Learning Objectives Upon completing this Chapter, the reader should be able to

- Work with vectors and vector products.
- Derive several mathematical represenations of 3D rigid body rotation
- Represent any displacement and rotation of a rigid body by a 4x4 matrix (the homogeneous transformation).
- Solve for unknown spatial relationships in terms of known spatial relationships using homogeneous transformations and transform graphs.

2.2 Description of Position and Orientation

2.2.1 Vectors and Translation

We use vectors to represent the location of objects and also to represent translation movements from one point to another.

Vector Cross Product

We will sometimes need the vector cross product which is briefly defined here.

The cross product of two vectors, represented

$$c = a \times b$$

is a vector who's magnitude is

$$|c| = |a||b|$$

and whos direction is orthogonal to both a and b. There are two orthogonal directions to a and b and the cross product sign is set by the right hand rule (sweeping the fingers from a to b and observing direction of thumb).

Another equivalent definition of the vector cross product can be expressed in matrix notation as

$$a \times b = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} b$$

Example 2.1
1) Find the magnitude of the cross product $a \times b$ where

$$a = [3 \quad 4 \quad 7]^T \qquad b = [2 \quad 4 \quad -3]^T$$

$$c = a \times b$$

$$|c| = |a||b| = 8.6 \times 5.39 = 46.3$$

2) Find the elements of the cross product.

To get the components, it is easiest to use the matrix definition:

$$c = \begin{bmatrix} 0 & -7 & 4 \\ 7 & 0 & -3 \\ -4 & 3 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ -3 \end{bmatrix} = \begin{bmatrix} -40 \\ 23 \\ 4 \end{bmatrix}$$

You can verify that the magnitude of this vector is also 46.3.

2.2.2 Coordinate Systems and Frames

Of course when we use them numerically, our vectors must be defined with respect to axes in space. Such a coordinate system contains three axes oriented to measure displacement in each of the three spatial dimensions. We denote these axes x, y, z and we will require that this system is *right handed* so that

$$x \times y = z$$

where $\times$ represents the vector cross product. We can have as many coordinate systems as we like and they can differ from one another in two ways. The location of their origin can be different, in which case we say they are *offset* or *displaced* from one another, and their orientation can be different, in which case we say they are *rotated* from one another.

Another term for coordinate system is *reference frame* or simply *frame*.

2.2.3 Points

The numerical value or location of a point in space is only defined with respect to a selected coordinate system. Thus if we have a point, P, and a coordinate system, A, it will have a specific numerical location with respect to A. For example:

$${}^A P = \begin{bmatrix} 1.7 \\ 0.25 \\ -3.1 \end{bmatrix}$$

While P designates the point in the abstract, we add the leading subscript, ${}^A P$ to indicate P's representation in coordinate system A.

The origin of a frame is also a point. The displacement between two frames can thus be represented by a point, the origin of one frame, expressed in a second frame. For example, imagine that M and N, are two coordinate systems not rotated with respect to each other, but their origins, O_M and O_N, are separated by 5 cm in the x direction. Then

$${}^M O_N = [0.05 \quad 0.0 \quad 0.0]^T$$

and

$${}^N O_M = [-.05 \quad 0.0 \quad 0.0]^T$$

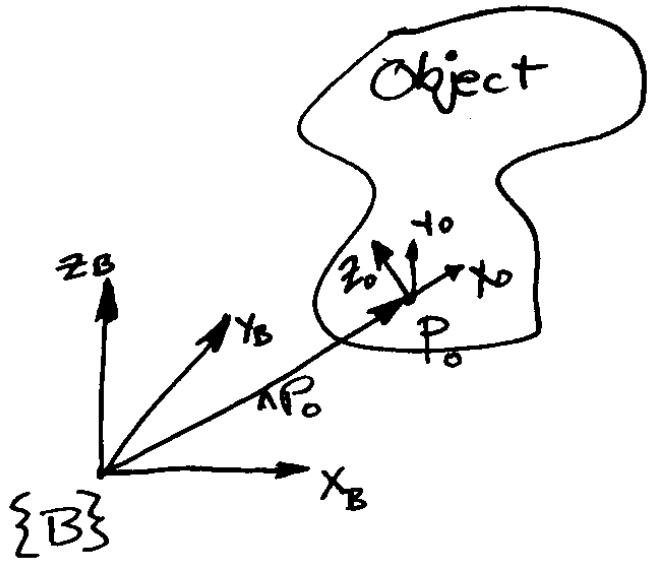


Figure 2.1: Representation of a rigid object in terms of an origin, P_0, and a frame attached to the object (frame 0).

2.2.4 Rigid Objects

We will view rigid objects as a collection of points with a fixed relationship to one another. How can we describe a rigid object's configuration in space with respect to some base frame, frame B?

First choose a point on the object, P_0 (Figure 2.1). Then create a frame, O, whos origin is P_0. The origin of P_0 does not have to be on or inside the object as long as all points in the object have a *fixed* representation in frame O. The orientation of frame O does not matter either as long as the previous condition is satisfied. We can represent the location of the object in frame B by simply the position of its origin, ${}^BP_0$.

Representation of the orientation of the object is somewhat more involved. We can imagine that if we only know the origin of object's frame, P_0, the object is still free to rotate about that point in multiple ways. Now, if we specify a second point on the object, P_1, the object is more constrained, but can still freely spin around the line connecting P_0 and P_1 without violating our assumption. If we specify a third point on the object, we fully describe the object in both position and orientation.

We will be a little bit redundant and specify the object's configuration with four points: the origin, P_0, and the tips of the three unit vectors of frame O: x_o, y_o, z_o. However we will remove the translation portion from these three points by subtracting P_0 to create

$$\hat{x_0} = x_0 - P_0, \qquad \hat{y_0} = y_0 - P_0, \qquad \hat{z_0} = z_0 - P_0$$

In particular, we represent these vectors in the base frame, frame B.

$${}^B\hat{x_0} = {}^Bx_0 - {}^BP_0 \quad \text{etc.}$$

Our representation of the object's configuration at this point is the tuple:

$$Obj = \left\{ [{}^BP_x \quad {}^BP_y \quad {}^BP_z]^T, \begin{bmatrix} {}^B\hat{x}_{Ox} & {}^B\hat{y}_{Ox} & {}^B\hat{z}_{Ox} \\ {}^B\hat{x}_{Oy} & {}^B\hat{y}_{Oy} & {}^B\hat{z}_{Oy} \\ {}^B\hat{x}_{Oz} & {}^B\hat{y}_{Oz} & {}^B\hat{z}_{Oz} \end{bmatrix} \right\}$$

we refer to the collection of $\hat{x_O}, \hat{y_O}, \hat{z_O}$ vectors into a matrix as a *rotation matrix.*.

2.3 Rotation Matrix

We will study the rotation matrix from the point of view of four definintions. Although these four definitions seem different at first, they are all simultaneously true. In this section we will consider only frames which differ in orientation. We will add translation later.

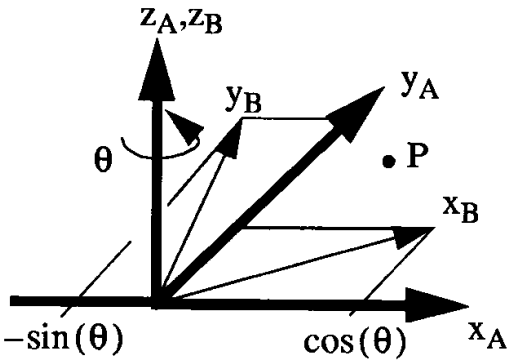


Figure 2.2: Two frames which are rotated relative to each other by the angle θ about the axis z.

The rotation matrix ${}^A_B R$ is:

1. A matrix which specifies frame B in terms of frame A. The columns of ${}^A_B R$ are the unit vectors of frame B specified in frame A.

2. A matrix which maps a point expressed in frame B, ${}^B P$ to its representation in frame A, ${}^A P$ by matrix pre-multiplication:
$$ {}^A P = {}^A_B R \, {}^B P$$

3. A description of an *operation* such as physically turning from frame A to frame B.

4. A 3×3 matrix with mathematical constraints: If $R = [a \quad b \quad c]$ where a, b, c are 3-vectors, then
$$|a| = |b| = |c| = 1$$
and
$$a \times b = c, \quad b \times c = a, \quad c \times a = b$$
where $\times$ is the vector cross product.

2.4 Descriptions of Rotations

There are many subtle aspects to rotation of objects and representing them mathematically. In this section we will consider ways to describe rotations and combinations of rotations using the rotation matrix.

2.4.1 Rotation in the Plane

Consider rotation in the x, y plane about the axis z. When we say "about," we mean that any point along the line indicated by z is unchanged by the rotation. Also, the z coordinate of any point is unchanged by rotation about the z axis. In Figure 2.2 we have two frames, A, in bold, and B. B is rotated about the z axis with respect to A. Consider the tip of the unit vector x_A. After its rotation to x_B, it has components which we can project back on to x_A and y_A. It still has no component in z. If we have rotated around z by angle θ, then these projections are
$$ {}^A x_B = \begin{bmatrix} cos(\theta) \\ sin(\theta) \\ 0 \end{bmatrix}$$

Similarly,

$$ {}^A y_B = \begin{bmatrix} -sin(\theta) \\ cos(\theta) \\ 0 \end{bmatrix} $$

These are two columns of the rotation matrix ${}^A_B R$:

$$ {}^A_B R = \begin{bmatrix} cos(\theta) & -sin(\theta) & ? \\ sin(\theta) & cos(\theta) & ? \\ 0 & 0 & ? \end{bmatrix} $$

We can get column 3 of ${}^A_B R$ two ways. First, as we have just done, we can note that

$$ {}^A z_B = {}^A z_A = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} $$

Therefore our complete matrix must be

$$ {}^A_B R = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

Our favorite point, P, has the representation $\{P_{XA}, P_{YA}\}$ in the x, y plane. P_{ZA} can have any value for this discussion since our rotation is about z. Just as it did with the unit vectors, ${}^A_B R$ maps P's representation in B to its representation in A:

$$ {}^A P = {}^A_B R \, {}^B P $$

One way we can see this is by an argument of superposition since matrix multiplication is a linear transformation and ${}^B P$ can be expressed as a linear combination of the unit vectors of B.

Rotational Operators

By similar arguments we can derive rotation matrices for rotation around the other axes, x, and y. Let's denote the unit vectors,

$$ \hat{x} \equiv [1 \quad 0 \quad 0]^T \quad \hat{y} \equiv [0 \quad 1 \quad 0]^T \quad \hat{z} \equiv [0 \quad 0 \quad 1]^T $$

These are not vectors in any particular coordinate system, just the numerical values shown above. We will define the matrices

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

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

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

to designate rotations about the three main axes of a frame. At this point let's introduce a shorter notation:

$$ c\theta \equiv cos(\theta), \quad s\theta \equiv sin(\theta) $$

and

$$ c_1 = \cos(\theta_1), \quad s_1 = sin(\theta_1) $$

for example,

$$ \mathrm{Rot}(\hat{x}, \theta_3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_3 & -s_3 \\ 0 & s_3 & c_3 \end{bmatrix} $$

Example 2.2
A point in frame B is

$$ {}^{B}P = \begin{bmatrix} -2 \\ 2 \\ 0.707 \end{bmatrix} $$

and there exists a frame A, whos rotation relative to B is given by

$$ {}^{A}_{B}R = \text{Rot}(\hat{x}, \pi/4) $$

1) What are the elements of ${}^{A}_{B}R$?
Answer:

$$ {}^{A}_{B}R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.707 & -0.707 \\ 0 & 0.707 & 0.707 \end{bmatrix} $$

2) What is ${}^{A}P$?
Answer:

$$ {}^{A}P = {}^{A}_{B}R \, {}^{B}P = \begin{bmatrix} -2.0 \\ 0.914 \\ 1.914 \end{bmatrix} $$

3) What is the change in magnitude of P when it is rotated?
Answer:

$$ \frac{|{}^{A}P|}{|{}^{B}P|} = \frac{2.915}{2.915} = 1 $$

A rotation matrix should never change the magnitude of a point.

We can verify the fourth definition of the rotation matrix with the rotation operators. For example, using $\text{Rot}(\hat{x}, \theta)$ we can easily verify that its columns have magnitude 1. We can verify the cross product constraints as well. Using $\text{Rot}(\hat{x}, \theta)$, then let a and b equal the first two columns

$$ a = [1 \quad 0 \quad 0]^T, \quad b = [0 \quad c\theta \quad s\theta]^T $$

We want to show that $a \times b = c$. Recall that one definition of the vector cross product is

$$ a \times b = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} b $$

so returning to our example,

$$ c = a \times b = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ c\theta \\ s\theta \end{bmatrix} = \begin{bmatrix} 0 \\ -s\theta \\ c\theta \end{bmatrix} $$

which is the answer we expect (i.e the third column of $\text{Rot}(\hat{x}, \theta)$) and verifies that the third column is orthogonal to the first two.

2.4.2 Inverse of a Rotation Matrix

The properties given in our fourth definition describe an *orthonormal* matrix. Any orthonormal matrix has an inverse which is equal to its transpose. Thus for rotation matrices

$$ R^{-1} = R^T $$

Clearly the inverse of a rotation matrix always exists because the transpose always exists. When we apply this to the coordinate transform application of rotation matrices:

$$ ({}^{A}_{B}R)^{-1} = ({}^{A}_{B}R)^T = {}^{B}_{A}R $$

2.4.3 Composition of Rotations

Suppose there are two rotations which create a more complex relationship between two frames. If we can transform a point from a rotated frame back to the base frame by premultiplying by the rotation matrix, then we should be able to do this twice to represent two rotations.

Example 2.3

Two frames, A, and B, start out coincident (${}^A_B R = I$). Then frame B is rotated about z_A by θ and then frame B is rotated about y_B by ϕ. Note that the second axis, y_B, is known:

$$ {}^B y_B = [0 \quad 1 \quad 0]^T = \hat{y} $$

in the most recent frame.
1) What is ${}^A_B R(\theta, \phi)$? In other words, what matrix represents the combination of both rotations?
Answer: Consider an intermediate frame, A', to represent the frame after the first rotation:

$$ {}^A_{A'} R = \text{Rot}(\hat{z}, \theta) $$

by definition 3

$$ {}^{A'}_B R = \text{Rot}(\hat{y}, \phi) $$

So now consider mapping a point in frame B, ${}^B P$, back to A using definition 2 twice:

$$ {}^A P = {}^A_{A'} R {}^{A'}_B R \, {}^B P $$

Thus,

$$ {}^A_B R = \text{Rot}(\hat{z}, \theta)\text{Rot}(\hat{y}, \phi) $$

We may combine successive rotations by *post*multiplication if each rotation is about a vector represented in the *current* rotated frame. In Example 3, the first rotation, $\text{Rot}(\hat{z}, \theta)$, is postmultiplied by the second rotation, $\text{Rot}(\hat{y}, \phi)$ to get the compound rotation ${}^A_B R$. This gives

$$ {}^A_B R(\theta, \phi) = \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\phi & 0 & s\phi \\ 0 & 1 & 0 \\ -s\phi & 0 & c\phi \end{bmatrix} = \begin{bmatrix} c\theta c\phi & -s\theta & c\theta s\phi \\ s\theta c\phi & c\theta & s\theta s\phi \\ -s\phi & 0 & c\phi \end{bmatrix} $$

When successive rotations are described with respect to a single, fixed frame (in contrast to the compound example above), we may combine these by *pre*multiplication. To illustrate this, we'll take the previous example of compound rotations and change just one thing: the second rotation will take place around y_A instead of y_B.

Example 2.4

Two frames, A, and B, start out coincident (${}^A_B R = I$). Then frame B is rotated about z_A by θ and then frame B is rotated about y_A by ϕ. Unlike the previous example, the second axis is known only in the original frame, frame A. So we know:

$$ {}^B y_A \neq [0 \quad 1 \quad 0]^T $$

1) What is ${}^A_B R(\theta, \phi)$ this time?
Answer: As before, consider the intermediate frame, A', to represent the frame after the first rotation:

$$ {}^A_{A'} R = \text{Rot}(\hat{z}, \theta) $$

by definition 3 (Section 2.3). For the second rotation let's use the point mapping interpretation, definition 2. The second rotation must be represented by a matrix which maps a point from B to A'. This rotation is *not* one of our three cannonical rotation matrices because $y_A \neq \hat{y}$ in our current frame. We can derive the proper transformation by breaking it down into three steps:

1. Transform to A using ${}^A_{A'} R$.
2. $\text{Rot}(\hat{y}, \phi)$
3. Transform back to A' using ${}^{A'}_A R$.

Thus,

$$ {}^{A'}_B R = {}^{A'}_A R \, \text{Rot}(\hat{y}, \phi) \, {}^A_{A'} R $$

Now the complete mapping from B to A is

$$ {}^A_B R = {}^A_{A'} R \, {}^{A'}_B R = {}^A_{A'} R \, {}^{A'}_A R \, \text{Rot}(\hat{y}, \phi) \, {}^A_{A'} R $$

Because ${}^A_{A'} R = {}^{A'}_A R^{-1}$, the first two rotation matrices cancel giving

$$ {}^A_B R = \text{Rot}(\hat{y}, \phi) {}^A_{A'} R = \text{Rot}(\hat{y}, \phi) \text{Rot}(\hat{z}, \theta) $$

expanding this:

$$ {}^A_B R = \begin{bmatrix} c\phi & 0 & s\phi \\ 0 & 1 & 0 \\ -s\phi & 0 & c\phi \end{bmatrix} \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\phi c\theta & -c\phi s\theta & s\phi \\ s\theta & c\theta & 0 \\ -s\phi c\theta & s\phi c\theta & c\phi \end{bmatrix} $$

Note that after the cancelation, it is as though the second rotation *pre*multiplies the first one.

Example 2.5
Two frames A and B start out superimposed. Then

B is rotated about x_A by θ_1.
B is rotated about y_B by θ_2.
B is rotated about y_A by θ_3.

What is ${}^A_B R$?
Answer:
Set up intermediate frames as above: A', A''. Then

$$ {}^A_B R = {}^A_{A'} R \, {}^{A'}_{A''} R \, {}^{A''}_{B} R $$

Considering the first rotation, since in the initial frame, $x_A = \hat{x}$ we can write

$$ {}^A_{A'} R = \mathrm{Rot}(\hat{x}, \theta_1) $$

In the second rotation, the axis is also known in the current frame: ${}^{A'} y_B = \hat{y}$, we can similarly write:

$$ {}^{A'}_{A''} R = \mathrm{Rot}(\hat{y}, \theta_2) $$

However for the third rotation we have to go all the way back to the initial frame:

$$ {}^{A''}_{B} R = \underbrace{{}^{A''}_{A'} R \, {}^{A'}_{A} R \, \mathrm{Rot}(\hat{y}, \theta_3) \, {}^{A}_{A'} R \, {}^{A'}_{A''} R}_{} $$

Combining

$$ {}^A_B R = \underbrace{{}^A_{A'} R \, {}^{A'}_{A''} R \, {}^{A''}_{A'} R \, {}^{A'}_{A} R}_{=I} \mathrm{Rot}(\hat{y}, \theta_3) \, {}^{A}_{A'} R \, {}^{A'}_{A''} R = \mathrm{Rot}(\hat{y}, \theta_3) \, {}^{A}_{A'} R \, {}^{A'}_{A''} R $$

$$ {}^A_B R = \mathrm{Rot}(\hat{y}, \theta_3) \mathrm{Rot}(\hat{x}, \theta_1) \mathrm{Rot}(\hat{y}, \theta_2) $$

Example 2.6
A robotic vehicle drives straight for 10 meters then turns right by 90°. Then it drives up a 45° incline and comes to a stop.
A frame mounted to the top of the vehicle is called F_0. The onboard computer can store the current value of F_0 at any time from navigation instruments as a 3x3 matrix. Before driving the computer stores

$$F_1 = F_0$$

Then after the drive above, it stores the new orientations

$$F_2 = F_0$$

What is the rotation matrix ${}^1_2R$ which describes the change in orientation?
Answer:
We have rotation matrices describing the three stages of the trip:
1) Drive straight for 10 meters: $\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$
2) Turn right by 90°: Rot($\hat{z}$, 90°): $\begin{bmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}$
3) Go up 45° incline: Rot($\hat{y}, -45°$): $\begin{bmatrix}0.707 & 0 & -0.707\\ 0 & 1 & 0\\ 0.707 & 0 & 0.707\end{bmatrix}$
(about the *local* frame).
Then we can combine them by postmultiplication:

$${}^1_2R = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0.707 & 0 & -0.707\\ 0 & 1 & 0\\ 0.707 & 0 & 0.707\end{bmatrix}$$

$${}^1_2R = \begin{bmatrix}0 & 1 & 0\\ -0.707 & 00.707 & \\ 0.707 & 0 & 0.707\end{bmatrix}$$

It can be confusing to remember the order of multiplication for multiple rotations. One way to think about it is we always post-multiply successive rotations. If the rotation is about the first, fixed frame, then we must use the similarity transformation as shown in Examples 4 and 5 of this chapter. The effect of this similarity transformation is to make this appear like pre-multiplication.

To summarize, if we are careful, we can use the following rules of thumb to determine the order of matrix multiplication:

1. If the rotation is about an axis in the *original fixed* frame, pre-multiply
2. If the rotation is about an axis in the *current* frame, post-multiply

2.4.4 Rotation of a Point

The rotation matrix also describes the physical act of turning something (Figure 2.3). So how can we describe the rotation of a point about an axis in a certain frame but still representing the point in the same frame?

We consider the point's rotation to be positive if the angle from the orginal point, P, to the rotated point, $\hat{P}$, is positive in the right hand sense around the axis. Consider for example, rotation of the point around the $\hat{z}$ axis. By our third definition (Section 2.3),

$${}^A\hat{P} = \text{Rot}(\hat{z}, \theta){}^AP$$

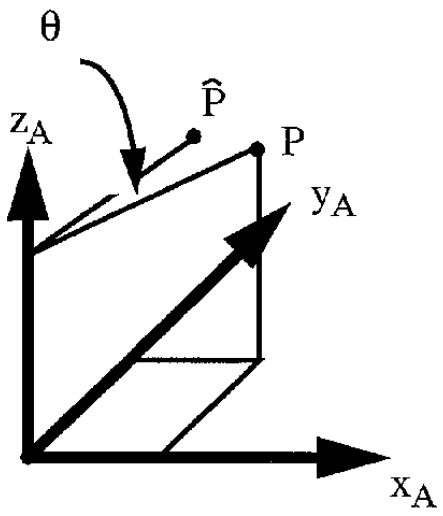


Figure 2.3: Rotation of a point around the z axis by angle θ. Here we consider the point to actually move in a single frame, A.

2.4.5 Rotations about axes other than x, y, z

Suppose we have an arbitrarily rotated frame, F:

$$F = {}^O_F R$$

Geometrically, this might look like Figure 2.4. Now we will use frame F to study more general rotations. Suppose ${}^O_F R$ is known and we want to do a rotation about z_F? How do we represent a rotation about z_F in frame O by θ? Put another way, if the point ${}^O P$ is rotated about z_F by θ to get ${}^O \hat{P}$, what is ${}^O \hat{P}$?

We solve this similarly to the last example by transforming back to frame F and doing the canonical rotation about $\hat{z}$:

$$\begin{aligned} {}^O \hat{P} &= {}^O_F R \mathrm{Rot}\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \theta \right) {}^F_O R \\ &= \mathcal{R} \mathrm{Rot}(\hat{z}, \theta) \mathcal{R}^T \, {}^O P \end{aligned}$$

where $\mathcal{R} \equiv {}^O_F R$. Another way to look at this is to define

$$\mathrm{Rot}(z_F, \theta) = \mathcal{R} \mathrm{Rot}(\hat{z}, \theta) \mathcal{R}^T$$

Mathematically, this is a "similarity transform" which can be used to rotate about an arbitrarily rotated vector, in this case, z_F.

2.5 Reduced Parameter Representations of Rotation

Although nine numbers are required to make a rotation matrix, only 3 of them are independent (this is because there are six constraints on the matrix in definition 4 of Section 2.3). This implies that there should be representations of rotation needing only 3 parameters.

2.5.1 Roll-Pitch-Yaw Angles (XYZ Fixed Angles)

In aviation, aeronautics and sailing, the terms "roll, pitch, and yaw" are commonly used. These terms describe rotations about the x, y, z axes of a fixed coordinate system in that order (Figure 2.5). When we say that the coordinate system is "fixed," we mean *not* that it is fixed to the object (i.e. the aircraft of

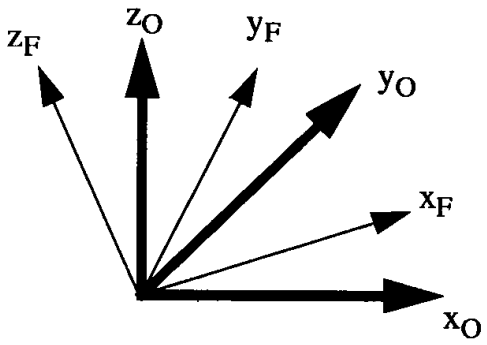


Figure 2.4: Frame F has an arbitrary orientation relative to frame O

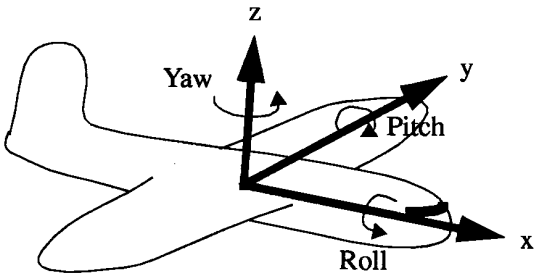


Figure 2.5: Roll-Pitch-Yaw angles (commonly used in avaition) describe rotations in a fixed order about x, y, z axes of a fixed coordinate system.

Figure 2.5), but that its orientation is fixed in space. Following an argument similar to the previous example of compound rotations in a fixed frame gives us

$$R_{RPY} = \text{Rot}(\hat{z}, Yaw)\text{Rot}(\hat{y}, Pitch)\text{Rot}(\hat{x}, Roll) \tag{2.1}$$

If we define the amounts of Roll, Pitch, and Yaw, (positive in the right hand sense) about x, y, z to be C, B, A respectively, then

$$R_{RPY} = \text{Rot}(\hat{z}, A)\text{Rot}(\hat{y}, B)\text{Rot}(\hat{x}, C)$$

$$= \begin{bmatrix} cA & -sA & 0 \\ sA & cA & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cB & 0 & sB \\ 0 & 1 & 0 \\ -sB & 0 & cB \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cC & -sC \\ 0 & sC & cC \end{bmatrix} = \begin{bmatrix} cAcB & cAsBsC - sAcC & cAsBcC + sAsC \\ sAcB & sAsBsC + cAcC & sAsBcC - cAsC \\ -sB & cBsC & cBcC \end{bmatrix}$$

2.5.2 ZYX Euler Angles

Another 3-parameter representation which is commonly used is Euler Angles. In this case the rotations are taken about the axes of a *rotating* frame. Start with $F_1 = \{x_1, y_1, z_1\}$ superimposed on F_0. Then rotate through the three steps labeled A-C in Figure 2.6.

1. Rotate about z_0 by A to create F_1.
2. Rotate about y_1 by B to create F_2.
3. Rotate about x_2 by C to create F_3.

By definition (3), step one defines ${}^0_1R$, step two defines ${}^1_2R$, etc. By definition (2)

$$ {}^0_1R = \text{Rot}(\hat{z}, A) = \text{Rot}\,(\hat{z}, A) $$

maps a point in frame 1 to frame 0.

$$ {}^1_2R = \text{Rot}(\hat{y}, B) = \text{Rot}\,(\hat{y}, B) $$

maps a point in frame 2 to frame 1. And

$$ {}^2_3R = \text{Rot}(\hat{x}, C) = \text{Rot}\,(\hat{x}, C) $$

maps a point in frame 3 to frame 2.

Using the point mapping interpretation, the complete mapping is

$$ {}^0_3R = {}^0_1R\ {}^1_2R\ {}^2_3R = \text{Rot}(\hat{z}, A)\text{Rot}(\hat{y}, B)\text{Rot}(\hat{x}, C) \tag{2.2} $$

Note that (2.1) is the same as (2.2). But, the Roll Pitch and Yaw angles are *not* equivalent to ZYX Eluler angles, because the rotations A, B, and C were applied in opposite orders in the two cases.

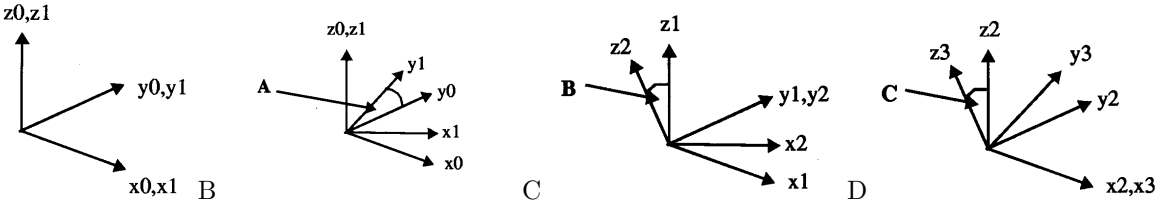


Figure 2.6: Initial Frames and rotations which make up the ZYX Euler Angles.

2.5.3 ZYZ Euler Angles

(Similar material to previous subsection on ZYX Euler Angles)

Equivalent Angle-Axis

Gimble Lock

2.6 Other Representations

2.7 Side Topic: Rotations in higher dimensions

We have seen that at least three parameters are sufficient and necessary to specify rotation in three dimensions. Is it a coincidence that 3 dimensional space has 3 rotation parameters? Any hope for this simple rule is dashed by considering that only 1 orientation parameter is needed in 2 dimensions. How many parameters are required to specify rotations of a 4 dimensional object?

We will use definition 4 to rephrase and generalize the question: How many parameters does it take to specify an $n \times n$ orthonormal matrix?

The total number of matrix elements is n^2. Then there are two constraints:

1. $C_i \cdot C_j = 0$
2. $|C_i| = 1$

where C_i is the ith column of the matrix.

The number of constraints due to the first condition is

$$ \binom{n}{2} = \frac{n!}{2(n-2)!} $$

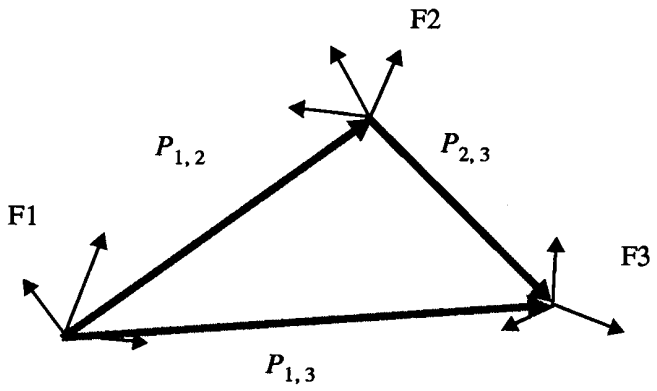


Figure 2.7: Three frames which are offset from each other in position and orientation. Vectors show displacements between the origins of the frames.

and the number of constraints of type 2 is n. So the total number of parameters is the number of matrix elements minus the number of constraints:

$$N = n^2 - \frac{n!}{2(n-2)!}$$

which can be simplified to

$$N = \frac{1}{2}(n^2 - n)$$

So for a four dimensional object we need 6 parameters and for a 10 dimensional one we need 45 parameters(!).

2.8 Homogeneous Transformation

Now we want to add translation to our description of the configuration of a rigid object. If we use the x, y, z coordinates of the origin of a frame attached to an object, and add that information to the rotation matrix of that frame, we now have its configuration completely specified relative to some "base frame," B i.e.

$$\text{configuration} = \{x, y, z, {}^B_0R\}$$

More generally, this gives us a way to relate frames to each other. Suppose we have a series of frames with different positions and orientations, and some relations between them as in Figure 2.7.

We can pose the following questions about these three frames:
If we know $P_{1,2}, P_{2,3}$, ${}^1_2R$, ${}^2_3R$, what is $P_{1,3}$, ${}^1_3R$?
Consider a point expressed in frame 3, ${}^3P$. It can be expressed in frame 2 by the affine transformation

$${}^2P = P_{2,3} + {}^2_3R\,{}^3P$$

and in frame 1 by

$${}^1P = P_{1,2} + {}^1_2R(P_{2,3} + {}^2_3R\,{}^3P)$$

Equivalently,

$${}^1P = \underbrace{P_{1,2} + {}^1_2RP_{2,3}} + \underbrace{{}^1_2R\,{}^2_3R}\,{}^3P)$$

from which we can conclude

$$P_{1,3} = P_{1,2} + {}^1_2RP_{2,3} \qquad {}^1_3R = {}^1_2R\,{}^2_3R$$

Although the preceding can be used as a general way to represent the spatial relationships between objects and robots, it is messy because we must add as well as multiply. This is a problem because we often have to represent a series of these transforms which yields ever more complicated expressions.

An alternative is presented by "homogeneous coordinates" which cleverly use an additional dimension to reduce the affine transformation to pure matrix multiplication. We define the homogeneous transform

$${}^1P = {}^1_2T^2P$$

where

$$ {}^1_2T = \begin{bmatrix} \begin{bmatrix} & {}^1_2R & \end{bmatrix} & \begin{bmatrix} {}^1P_{1,2} \end{bmatrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} $$

This 4×4 matrix compactly describes both the rotation and the position offset. When we work with homogeneous transforms, we augment our position vectors by adding a 1 in the fourth position. i.e.

$$ {}^1P = \begin{bmatrix} {}^1P_x \\ {}^1P_y \\ {}^1P_z \\ 1 \end{bmatrix} $$

in the example above,

$$ {}^1P = {}^1_2T\,{}^2P = \begin{bmatrix} \begin{bmatrix} & {}^1_2R & \end{bmatrix} & \begin{bmatrix} {}^1P_{1,2} \end{bmatrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \begin{bmatrix} {}^2P_x \\ {}^2P_y \\ {}^2P_z \\ 1 \end{bmatrix} $$

$$ = \begin{bmatrix} {}^1_2R_{[1]} \cdot {}^2P + {}^1P_{1,2_{[1]}} \\ {}^1_2R_{[2]} \cdot {}^2P + {}^1P_{1,2_{[2]}} \\ {}^1_2R_{[3]} \cdot {}^2P + {}^1P_{1,2_{[3]}} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^1P_{1,2} + {}^1_2R\,{}^2P \\ 1 \end{bmatrix} $$

where ${}^1_2R_{[i]}$ represents row i of ${}^1_2R$, and ${}^2P + {}^1P_{1,2_{[i]}}$ represents the ith element of ${}^1P_{1,2}$. Notice that the final expression is completely equivalent to multiplying by a rotation matrix and then adding a translation.

With homogeneous transformations at our disposal, a series of spatial replationships can be expressed by a series of matrix multiplications as with pure rotation. For example, solving the problem posed above for the three frames, $F1, F2, F3$,

$$ {}^1_3T = {}^1_2T\,{}^2_3T $$

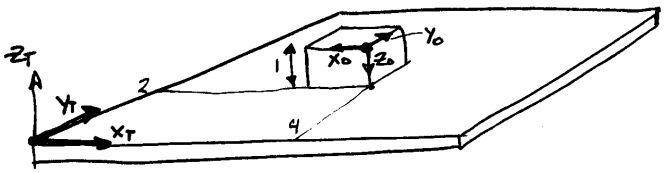
which is more compact indeed.

Homogeneous Transformation Definitions The homogeneous transformation has definitions analogous to those we made for rotation:

The homogeneous transformation ${}^A_BT$ is

1. A matrix which specifies frame B in terms of frame A.
2. A matrix which maps a point represented in frame B, ${}^BP$, to its representation in frame A, ${}^AP$.
3. A description of an operator, perhaps corresponding to a physical move, *from* frame A, *to* frame B.
4. A 4×4 matrix with the structure given above.

Example 2.7



Find the 4×4 homogeneous transformation which represents the configuration of the box on the table. Frame 0 is placed on a corner of the box and frame T is at a corner of the table such that X_T and Y_T are in the plane of the table top.
Question: What is ${}^{T}_{0}T$?
Answer: First, find the translation offset between the origin of the box frame (O) and the table frame (T):

$$ {}^{T}P_{O} = \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} $$

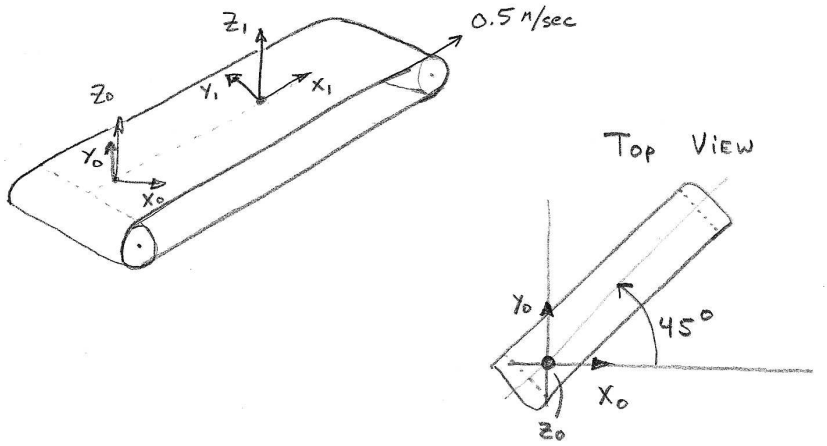
Then, identify the rotation matrix. First, align the frames by redrawing frame O without the offset superimposed on T. Then write each column of ${}^{T}_{O}R$. Each column is the vector $\{x_O, y_O, z_O\}$ written in frame T:

$$ {}^{T}_{O}R = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} $$

The homogeneous transform is thus:

$$ \begin{bmatrix} -1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Example 2.8



Two frames are on a conveyor belt. We have the following facts about the system

- The belt runs at 0.5 meters/sec.
- The belt is in the $\{X_0, Y_0\}$ plane.
- Frame F_1 is aligned with the origin of F_0 at $t = 0$.
- X_1 is pointing along the belt (same direction as belt velocity.)

Find the 4x4 homogenous transform (which is a function of time) of F_1
Answer:
O: The origin of F_1 is a function of time moving along the belt.

$$O = \begin{bmatrix} 0.5t\cos(45^\circ) \\ 0.5t\sin(45^\circ) \\ 0 \end{bmatrix} = [0.35t, \quad 0.35t, \quad 0]^T$$

Rotation $= rot(\hat{z}, 45^\circ)$

$$\begin{bmatrix} 0.707 & -0.707 & 0 \\ 0.707 & 0.707 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$^0_1T = \begin{bmatrix} 0.707 & -0.707 & 0 & 0.35t \\ 0.707 & 0.707 & 0 & 0.35t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.8.1 Inverse of Homogeneous Transforms

We will frequently need the inverse of the 4x4 homogeneous transform matrix. It seems intuitive that this inverse always exists because physical rotations and translations can always be "undone." It is straightfoward to show that the inverse of the 4x4 homogeneous transform,

Consider mapping a point in frame B to its value in frame A with both translation and rotation:

$$^AP = {}^A_BT\,{}^BP = {}^AO + {}^A_BR\,{}^BP$$

Where ${}^{A}O$ is the origin of frame B expressed in A. To invert this we seek a 4x4 matrix, $({}^{A}_{B}T)^{-1}$ such that

$$ {}^{B}P = ({}^{A}_{B}T)^{-1\,A}P $$

Solving

$$ \begin{aligned} {}^{B}P &= ({}^{A}_{B}R)^{-1}({}^{A}P - {}^{A}O) \\ &= {}^{B}_{A}R\,{}^{A}P - {}^{B}_{A}R\,{}^{A}O \end{aligned} $$

$$ {}^{A}_{B}T^{-1} = \begin{bmatrix} \begin{bmatrix} & {}^{A}_{B}R^{T} & \end{bmatrix} & \begin{bmatrix} -{}^{A}_{B}R^{T}\,{}^{A}O \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 \end{bmatrix} \end{bmatrix} $$

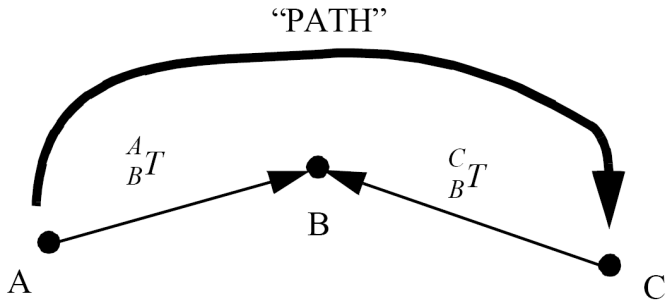
You can verify this by showing that $TT^{-1} = I_{4\times4}$.

2.9 Transform Equations

It is useful to develop a slightly more formal description of the relationships between frames. Spatial relationships can be expressed as transforms, products of transforms, or transform graphs. Transform graphs are directed graphs. Links specify transform matrices between frames. We represent a transform between two frames A, and B, by

$$ A \rightarrow B $$

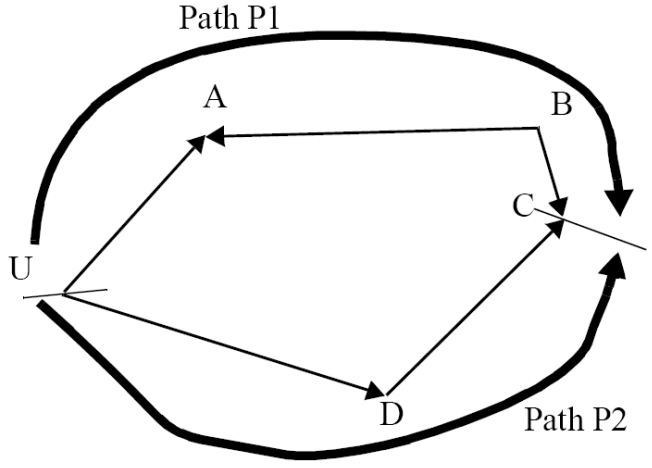
The arrow above is a graphical representation of ${}^{A}_{B}T$ and also of ${}^{B}_{A}T^{-1}$. A series of links (a path) in the transform graph corresponds to *post* multiplication of the transforms in the direction of the path.



The transform which shows the PATH above is therefore:

$$ {}^{A}_{C}T = \quad {}^{A}_{B}T({}^{C}_{B}T)^{-1} = \quad {}^{A}_{B}T\,{}^{B}_{C}T $$

In the first right hand side, ${}^{C}_{B}T$ is inverted because it represents an arrow which goes opposite to the the path. A closed path in such a graph can be cut at two nodes to form a tranform equation:



In the example above, the loop is cut at nodes U and C, and the two paths, P1, and P2 represent the same displacement. Each path corresponds to an equation

$$\text{P1}: \quad {}^{U}_{C}T = {}^{U}_{A}T \; {}^{B}_{A}T^{-1} \; {}^{B}_{C}T$$

$$\text{P2}: \quad {}^{U}_{C}T = {}^{U}_{D}T \; {}^{D}_{C}T$$

And, we can equate the two right hand sides:

$${}^{U}_{A}T \; {}^{B}_{A}T^{-1} \; {}^{B}_{C}T = {}^{U}_{D}T \; {}^{D}_{C}T$$

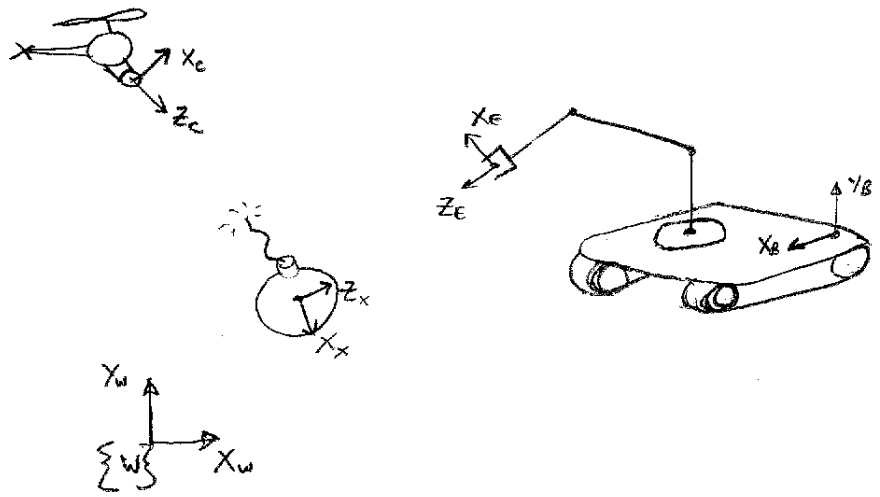
Furthermore, we can write the equation for a big path going completely around the loop in, for example, the clockwise direction:

$$P1(P2)^{-1} = {}^{U}_{C}T \; {}^{U}_{C}T^{-1} = I_{4\times 4}$$

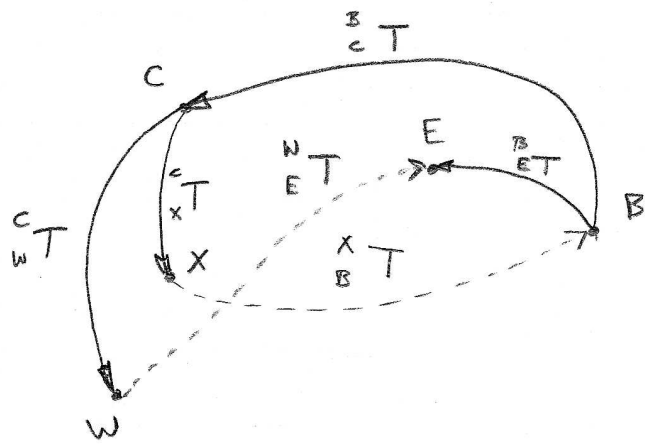
This is expected since by going completely around the loop we wind up in our original configuration or frame. Note that when we interpret it as a homogeneous transform, $I_{4\times 4}$ corresponds to

$$\text{Rot}(x, 0), \text{Trans}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right)$$

Example 2.9



A helicopter mounted camera identifies the location and orientation of a bomb. A bomb disposal robot must understand how to move its arm to pick up the bomb. The following transforms are known: ${}^{C}_{W}T, {}^{C}_{X}T, {}^{B}_{C}T, {}^{B}_{E}T$
A) Draw the transform graph
B) Solve for ${}^{X}_{B}T$
C) Solve for ${}^{W}_{E}T$
Answer:
A)



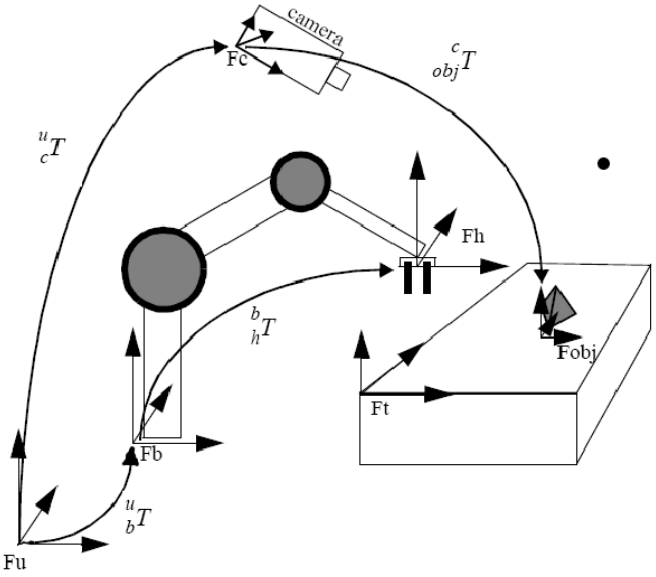
B)

$$ {}^{X}_{B}T = ({}^{C}_{X}T)^{-1}({}^{B}_{C}T)^{-1} $$

C)

$$ {}^{W}_{E}T = ({}^{C}_{W}T)^{-1}({}^{B}_{C}T)^{-1}{}^{B}_{E}T $$

Example 2.10



Lets consider a factory workstation with a robot arm, worktable, vision camera, and an object on the table. The known spatial relationships are shown by links in the diagram. Suppose we need to specify the position and orientation of the object so that the robot can pick it up. The robot controller must have the configuration in Fb, the base frame of the robot.
Q1: What is the configuration of the object in Fb?
A1: The object can be "reached" by a path in the transform graph from Fb through Fu, and Fc, to Fobj. Note that the first link has a direction opposite to the path we take to the object. The equation of this path is:

$$^{b}_{obj}T = ^{u}_{b}T^{-1}\ ^{u}_{c}T\ ^{c}_{obj}T$$

Q2: Suppose we want to find the unknown transforms $^{h}_{t}T, ^{t}_{obj}T$?
A2: If we add these two transforms to the graph (draw them in yourself), we now have a loop. We can cut this loop at any two points to make an equation. Suppose we make these cuts at the hand and at the object. Now we have

$$^{b}_{h}T^{-1}\ ^{u}_{b}T^{-1}\ ^{u}_{c}T\ ^{c}_{obj}T = ^{h}_{t}T\ ^{t}_{obj}T$$

Since the two transforms on the right hand side are unknown, we cannot solve this without another equation. Now suppose we go out to the factory floor and measure the location and orientation of the table. This gives us $^{u}_{t}T$. Drawing this link into the transform graph, we see that another loop is formed tangent to the original loop so that we have another equation. One such equation is

$$^{u}_{b}T\ ^{b}_{h}T\ ^{h}_{t}T = ^{u}_{t}T$$

We can solve this for one of our unknowns giving:

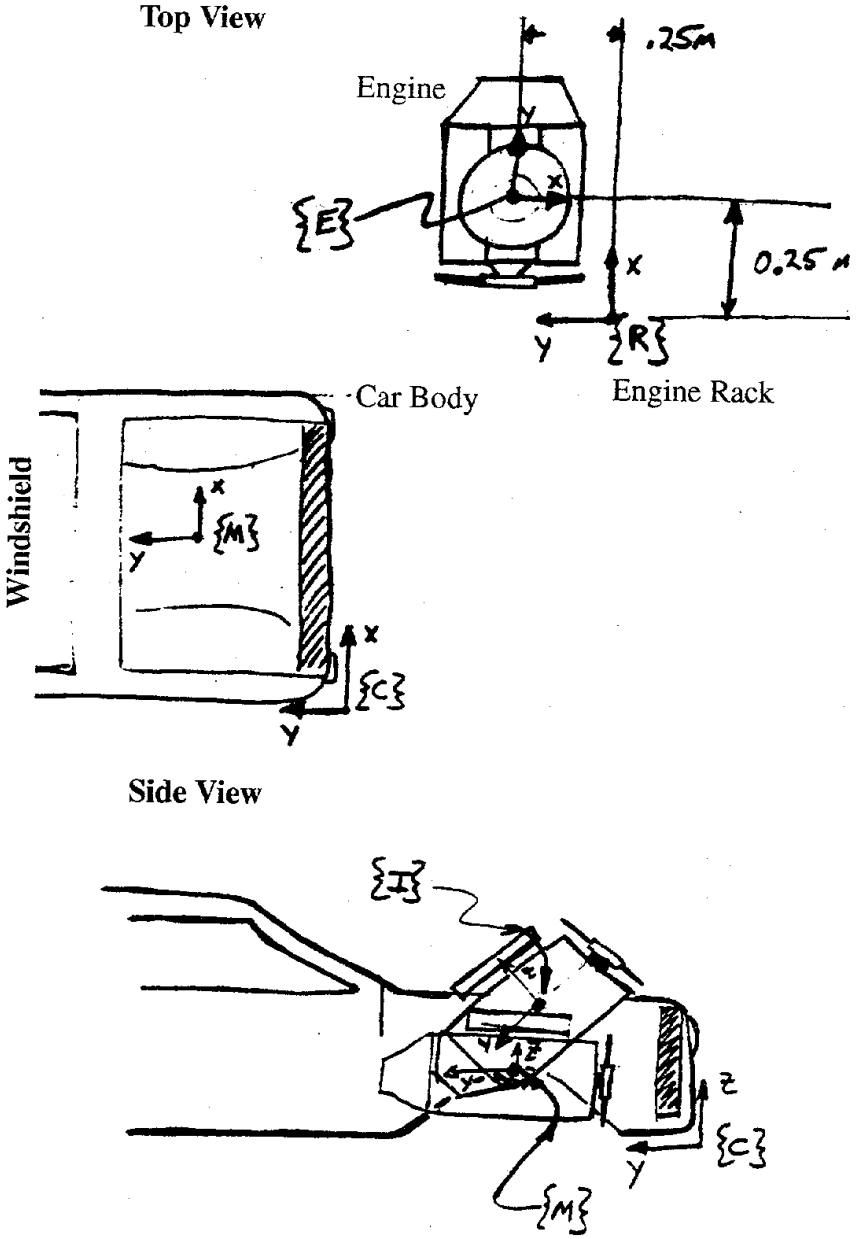
$$^{h}_{t}T = ^{b}_{h}T^{-1}\ ^{u}_{b}T^{-1}\ ^{u}_{t}T$$

Then we substitute to get

$$^{b}_{h}T^{-1}\ ^{u}_{b}T^{-1}\ ^{u}_{c}T\ ^{c}_{obj}T = ^{b}_{h}T^{-1}\ ^{u}_{b}T^{-1}\ ^{u}_{t}T\ ^{t}_{obj}T$$

$$^{c}_{obj}T = ^{u}_{c}T^{-1}\ ^{u}_{t}T\ ^{t}_{obj}T$$

Example 2.11
A revitalized GM plant has installed massive assembly robots. Such a robot arm must install the engine into a car body. Frames have been applied to the body and engine as shown below.



The following facts are known:

1. The engine must be lowered through an intermediate position in order for the transmission to clear the body. This position is ${}^{M}_{I}T$. Then the engine will be moved into its final position, ${}^{M}_{C}T$.
2. The car and engine rack are located at known tranforms ${}^{U}_{C}T$, and ${}^{U}_{R}T$ respectively.
3. GM has finally switched over to the metric system!

Questions:
Q1) Give the transform (in numerical form) which specfies the engine position in the rack, ${}^{R}_{E}T$. Assume that the z position coordinate is zero.

Example 2.11 cont.

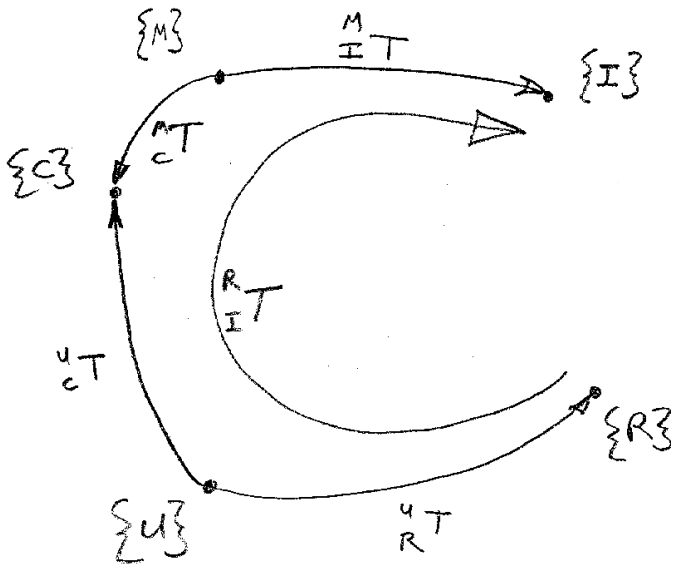
Solution

Obtain the rotation matrix by writing the unit vectors of E expressed in R, *or* evaluating $\text{Rot}(\hat{z}, -90°)$.

$$ {}^{R}_{E}T = \begin{bmatrix} 0 & 1 & 0 & 0.25 \\ -1 & 0 & 0 & 0.25 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Q2) You need to program the robot to take the engine from the rack and put it in the intermediate position. Solve for the transform which represents the intermediate engine position and orientation relative the engine rack, ${}^{R}_{I}T$ in terms of the known transforms above.

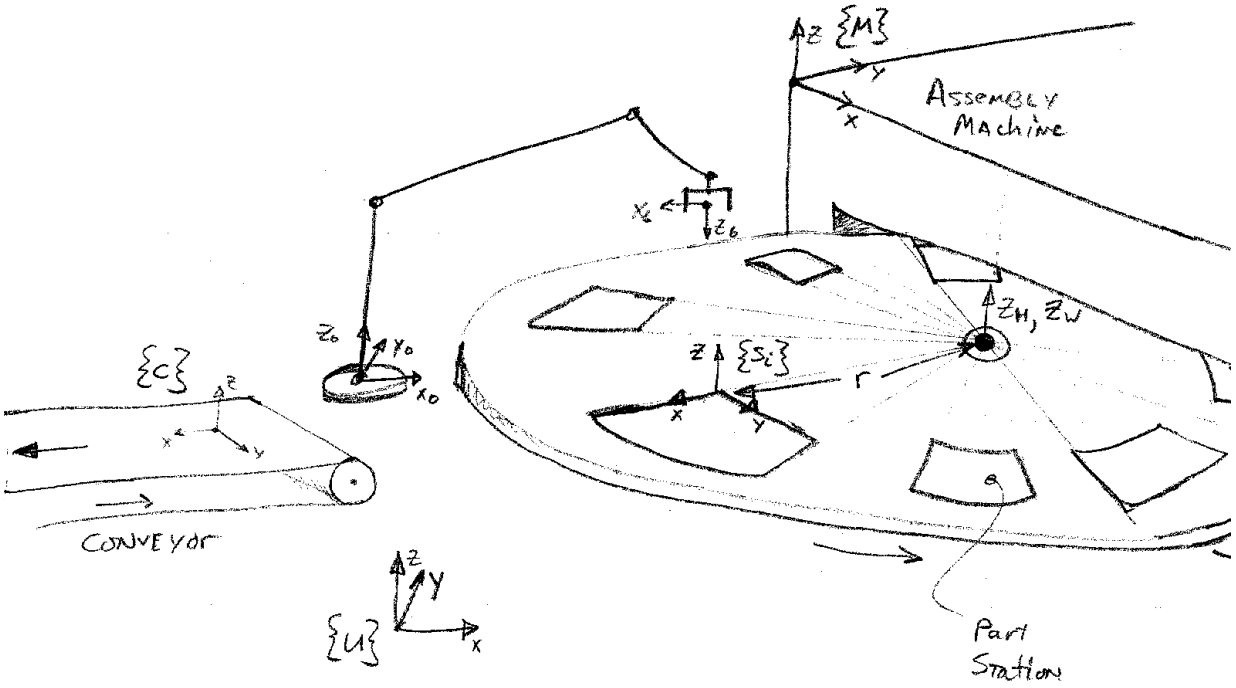
Solution



going around the transform graph in the direction shown, we have:

$$ {}^{R}_{I}T \quad = \quad {}^{U}_{R}T^{-1} \quad {}^{U}_{C}T \quad {}^{M}_{C}T^{-1} \quad {}^{M}_{I}T $$

Example 2.12



Pizza Problem

Completed assemblies (pizzas?) come out of a machine on a large rotary table. You must program a robot arm to pick up the pizzas and transfer them to a conveyer.
Use the following facts:

- The part stations are spaced every 36° on the wheel.
- The robot base frame is {0} and the end effector frame is {6}.
- The origin of frame H is located at the wheel hub:

$$ {}^{U}_{H}T = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 10 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- The i^{th} part station has frame $\{S_i\}$ located at its corner. The distance from the origin of $\{H\}$ to the origin of $\{S_i\}$ is $r = 5$.
- Another frame $\{W\}$ has the same origin as $\{H\}$, but X_W always points to the origin of $\{S_1\}$.
- The front panel of the machine is at

$$ {}^{U}_{M}T = \begin{bmatrix} 0 & 1 & 0 & 10 \\ -1 & 0 & 0 & 16 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- $\Theta_W(t)$ describes the rotation of the wheel. Θ_W is the angle from X_H to X_W.

Example 2.12 cont.
Q1: If t is in seconds, $\Theta_W(t) = \frac{\pi}{15}t$ rad or $\Theta_W(t) = 12t$ degrees, what is ${}^U_S T(i,t)$, i.e. what is the transform from frame U to frame S_i (multiply out your answer).

Solution:

$$ {}^U_{Si}T \quad = \quad {}^U_H T \quad {}^H_W T \quad {}^W_{Si} T $$

From the facts,

$$ {}^H_W T = \text{Rot}(\hat{z}, 12t^\circ $$

$$ {}^W_{Si} T = \text{Rot}(\hat{z}, (36(i-1))^\circ) $$

$$ {}^U_H T \quad = \quad \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 10 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Since the last two are both about the $\hat{z}$ axis, we can add their displacements to simplify. Let

$$ \alpha = (12t + 36(i-1))^\circ $$

Then

$$ {}^U_{Si}T \quad = \quad \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 10 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{Rot}(\hat{z}, \alpha) $$

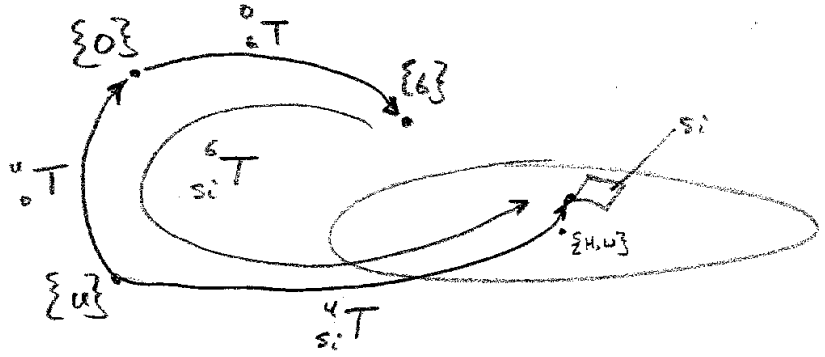
$$ {}^U_{Si}T \quad = \quad \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 10 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\alpha & -s\alpha & 0 & 0 \\ s\alpha & c\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^U_{Si}T \quad = \quad \begin{bmatrix} c\alpha & -s\alpha & 0 & 10 \\ s\alpha & c\alpha & 0 & 10 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Q2: If ${}^U_0 T, \quad {}^0_6 T, \quad {}^U_S T(i,t)$ are known, draw a transform graph, and solve for ${}^6_S T(i,t)$.

Solution:

Transform Graph:



$$ {}^6_{Si}T \quad = \quad {}^0_6 T^{-1} \quad {}^U_0 T^{-1} \quad {}^U_{Si} T $$

2.10 Quaternions

(to be written)

2.11 Exponential Coordinates

The following elegant notation and derivation is based closely on **Murray, Li, and Sastry, "A Mathematical Introduction to Robotic Manipulation", CRC Press, 1994.**

2.11.1 Rotation

Consider the velocity of a point q which is rotating about an axiz ω. We assume that $|\omega| = 1$ and its direction is the axis of rotation. We can write the velocity of the point as

$$\dot{q}(t) = \omega \times q(t)$$

Recall that the vector cross product ($\times$) can be represented as the product of a special skew-symmetric matrix,

$$\hat{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

with the vector, i.e.

$$\omega \times q = \hat{\omega} q$$

So we have

$$\dot{q}(t) = \hat{\omega} q(t)$$

This is a first order differential equation with the solution

$$q(t) = e^{\hat{\omega} t} q(0)$$

Now we need to understand what it means to take the exponential of a matrix! Using the Taylor series expansion, we have:

$$\exp(A) = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + ...$$

We are interested in the special case of

$$A = \hat{\omega} t$$

where $\hat{\omega}$ is the matrix defined as above and t is a scalar (time). One interpretation is that we can parameterize any rotation by the amount of time it takes at unit velocity ($|\omega| = 1$). Thus we can also write

$$R(\omega, \Theta) = e^{\hat{\omega}\Theta}$$

where

$$\Theta = |\omega| t$$

To evaluate the matrix exponential, we sort the series out into odd and even terms (and perform a few other tricks) to get:

$$e^{\hat{\omega}\Theta} = I + \left(\Theta - \frac{\Theta^3}{3!} + \frac{\Theta^5}{5!} + \dots\right)\hat{\omega} + \left(\frac{\Theta^2}{2!} + \frac{\Theta^4}{4!} + \frac{\Theta^6}{6!} + \dots\right)\hat{\omega}^2$$

Which simplifies to

$$e^{\hat{\omega}\Theta} = I + \hat{\omega}\sin\Theta + \hat{\omega}^2(1 - \cos\Theta)$$

So $e^{\hat{\omega}\theta}$ is the rotation matrix which expresses rotation by θ about axis ω.

This can also be inverted (See Murray, Li, Sastry, equations (2.17) and (2.18), or Craig, equations (2.81) and (2.82)):

$$\theta = cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right) \tag{2.3}$$

$$\omega = \frac{1}{2sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{2.4}$$

2.11.2 Translation and Rotation

Consider a point, $p(t)$ whos position is a function of time. Assume that the point is displaced due to rotation about an axis, ω, separate from the point. We assume that $|\omega| = 1$ and that q is a point **off** the axis. The velocity of the point is then

$$\dot{p}(t) = \omega \times (p(t) - q)$$

or

$$\dot{p}(t) = \omega \times p(t) - \omega \times q$$

Using $\hat{\omega}$ as defined above, if we define

$$\hat{\xi} = \begin{bmatrix} \hat{\omega} & -\omega \times q \\ 0 & 0 \end{bmatrix}$$

and we express $p(t)$ and its derivative in homogeneous coordinates,

$$\dot{p} = \begin{bmatrix} \hat{\omega} & -\omega \times q \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix}$$

In this special case (of rotation about an axis remote from ω) the velocity of the point p is $-\omega \times q$. More generally, we can write

$$\dot{p} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix}$$

More compactly:

$$\dot{p} = \hat{\xi} p$$

the matrix $\hat{\xi}$ is refered to as a *twist*.

This is a first order differential equation which has the solution:

$$p(t) = e^{\hat{\xi} t} p(0).$$

Recall that $\hat{\xi}$ doesn't have as simple a form as $\hat{\omega}$ and so it is not as easy to get an expression for $e^{\hat{\xi} t}$. It can be shown that (see Murray, Li, and Sastry, Proposition 2.8) that for a displacement which consists of rotation $\hat{\omega}\Theta$ and displacement $v\theta$

$$e^{\hat{\xi}\theta} = \begin{bmatrix} e^{\hat{\omega}\theta} & (I - e^{\hat{\omega}\theta})(\omega \times v) + \omega\omega^T v\theta \\ 0 & 1 \end{bmatrix} \tag{2.5}$$

As used here, θ is a general motion parameter and need not be an angle. The interpretation of $e^{\hat{\xi}\theta}$ is a description of a physical manipulation i.e.

$${}^A p(\theta) = e^{\hat{\xi}\theta} {}^A p(0)$$

Note that both ${}^A p(\theta)$ and ${}^A p(0)$ are represented in the same frame, A, which should also be the same as that used for ω and v.

To represent a known physical displacement as a twist, we first must know the 4x4 homogeneous transform defining the displacement, ${}^A_B T$ which the authors call g_{ab}. Given g_{ab}, we solve for the "twist coordinates", ω, θ, v equating (2.5) with the known g:

$$g = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} e^{\hat{\omega}\theta} & (I - e^{\hat{\omega}\theta})(\omega \times v) + \omega\omega^T v\theta \\ 0 & 1 \end{bmatrix}$$

for

$$R = e^{\hat{\omega}\theta}$$

we had equation (2.3 and (2.4) to find θ and ω. To get v, we have

$$P = (I - e^{\hat{\omega}\theta})(\omega \times v) + \omega\omega^T v\theta$$

which we can solve by hand calculation or Mathematica for v. There are some useful special cases. For example for a simple axis such as

$$\omega = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

the equation to solve is

$$\begin{bmatrix} s\theta & c\theta - 1 & 0 \\ 1 - c\theta & s\theta & 0 \\ 0 & 0 & \theta \end{bmatrix} v = P$$

In the special case of revolute motion about the axis ω, v is only due to the rotation and so it is known:

$$v = -\omega \times q$$

2.11.3 Groups

Murray, Li, and Sastry take some pains to relate these kinematic ideas to group theory. The basic properties of a group are

- It consists of a set of objects, G and a binary operation, $\times$.
- It is "closed" in the sense that if any two objects in the group are applied to the binary operation, the result is another member of the group.
- There exists an "identity element", i, which is a member of the group such that $g \times i = g$.
- For every element of the group, g, there is a unique inverse, also a an element of the group, g^{-1} such that $g \times g^{-1} = i$.
- The operation is associative: $(g_1 \times g_2) \times g_3 = g_1 \times (g_2 \times g_3)$.

For example, you can verify that these five properties hold for the set of orthonormal 3x3 rotation matrices together with the operation of matrix multiplication, and so they must form a group.

Murray, Li, and Sastry named four groups which are relevant to robotic manipulation:

$se(3)$ the group of 4x4 skew symmetric matrices (twists)

$so(3)$ the group of 3x3 skew symmetric matrices ($\hat{\omega}$).

$SE(3)$ the group of 4x4 homogeneous transforms

$SO(3)$ the group of 3x3 orthonormal rotation matrices

2.12 Summary of Notation

Chapter 3

Forward Kinematics

3.1 Problem Statement and Learning Objectives

Problem Statement This chapter addresses the problem of computing the position and orientation of an end effector, knowing the geometry of the manipulator and the position values of each joint. Joints may be rotary in which case the joint value is an angle, or prismatic, in which case the joint value is a displacement. We seek a general way to represent any serial manipulator.

Learning Objectives After completing this chapter, the student will be able to derive the forward kinematic model of a serial manipulator. Specifically to

- Identify the link and joint geometry from a picture or engineering drawing of a serial mechanism containing rotary and prismatic joints.
- Be able to assign a coordinate system to each link in a standardized manner.
- Be able to derive Denavit-Hartenberg parameters for each link
- Be able to form the 4x4 homogeneous transform for each link based on the DH parameters and be able to multiply these link matrices together to get a symbolic expression for the forward kinematic equations in the form of a 4x4 homogenous transform matrix.

The result is a computation of the position and orientation of the end effector knowing the geometric dimensions of each link and the position of each joint in the mechanism.

3.2 Serial Mechanisms

3.2.1 Links and Joints

Links

In this chapter we study the spatial relationships (transforms) between the links of a robot arm. First we define an *axis of motion* as

> A line in 3-space which contains points which are not moved by a particular rotation (a *rotary* axis), or which contains a direction describing linear motion (a *translation* axis).

We will define a *link* as

> A spatial relation, enforced by a rigid object, between two axes of motion.

In a typical robot arm, the links are rigid structural elements (most often of metal) which hold at each end the bearings for a joint. Their rigidity enforces a constant spatial relationship between axes of motion at the *proximal* (closer to the base) and *distal* (closer to the end effector) ends.

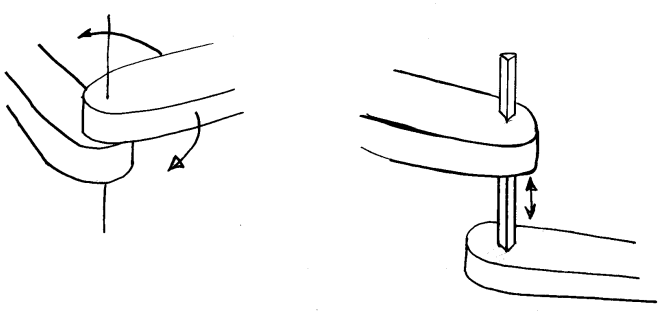


Figure 3.1: Two types of joints, rotational and translational/prismatic.

Joints

Joints connect the links in a serial kinematic chain. We define a joint as

> A structural element which allows relative motion between two bodies in only one degree of freedom. Such motion is constrained to translational motion along a line or rotation around a line.

We have two classes of joints, *rotational* and *translational*, also known as *prismatic*[1]. Combining the definitions of link and joint, we see that each axis of motion is fixed in the two links adjoining the joint. The degree of translation or rotation (the single motion freedom of the joint) is called the joint variable. Since the joint constrains relative motion of the two links to a single degree of freedom, the joint variable is sufficient to describe the spatial relationship between them.

A series of such links bridging adjacent axes is an open kinematic chain (also known as a robot arm!). The *forward kinematics* problem is to find the position and orientation of the last link in the chain (i.e. the robot hand) knowing only the robot's geometry and the angles or displacements of the joints. To make our method general, we will assume that there can be any amount of twiset, displacement, and offset between the ends of each link. This make the problem difficult unless we employ a systematic approach. We will use the approach invented by Denavit and Hartenberg in which we will construct a virtual kinematic chain which is equivalent to the original robot but is easier to analyze. The virtual kinematic chain in the "DH" method consists of the joint axes and their *common normals*. The common normal between two lines or axes is a unique[2] line which is perpendicular to both axes.

3.2.2 Modeling Links and Joints

Assignment of Frames to Links

We will define a series of frames along the virtual kinematic chain and identify the key rotations and translations which define their relationships.

Common Normal The CN will be our virtual link between two axes in space. It is worth noting that the CN might be completely outside the physical structure of the robot. However our virtual manipulator constructed using CNs will be an exact representation of the kinematics of the real one. It turns out that we can identify the CN of each link rather informally. We do not need to write the equation for each CN line segment but it will be more important to identify the two end points.

There will be a CN for each pair of joint axes (Figure 3.3). If the two joint axes are parallel, there are an infinite number of CNs so we just choose one arbitrarily. If the two axes intersect, then the CN has zero length but it still has a direction which is the cross product

$$CN = \text{Axis}_N \otimes \text{Axis}_{N+1} \tag{3.1}$$

[1] Think of a prism sliding inside a hole shaped to match its profile. It can slide in and out, but not rotate.

[2] Not always unique but see below.

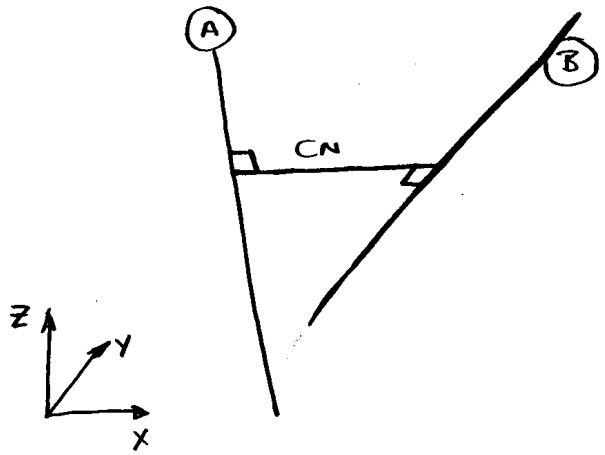


Figure 3.2: The Common Normal (CN) is the shortest line which intersects two given lines. It intersects lines **A** and **B** at a right angle.

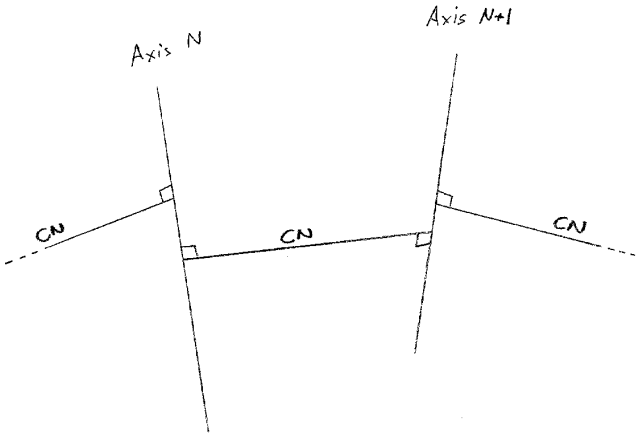


Figure 3.3: Virtual Links between axes are formed by the Common Normals.

Example 3.1
Find the Common Normal between the following lines:

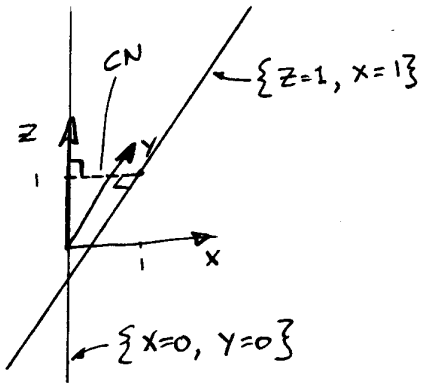
$$x = 0, y = 0$$

and

$$z = 1, x = 1$$

The common normal is also the shortest line segment which intersects both lines. The first line is a vertical along the z axis. The second goes in the y direction through the point $[1 \quad 0 \quad 1]^T$. The shortest line segment connecting these two lines is the line segement between $[0 \quad 0 \quad 1]^T$ and $[1 \quad 0 \quad 1]^T$ which is the line

$$z = 1, y = 0$$



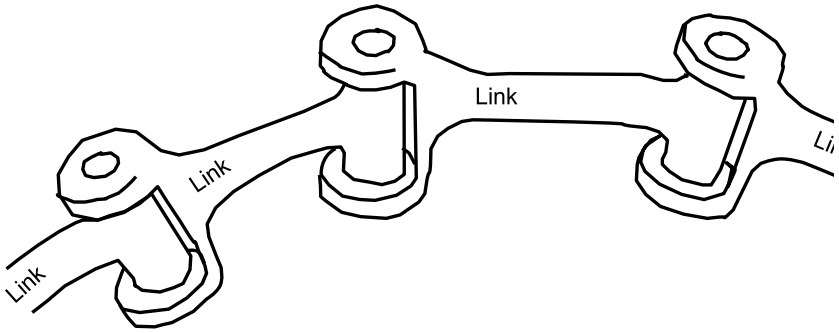
3.2.3 Fixing Frames to Links

Once we identify common normals between the manipulator axes, we are ready to locate a frame, rigidly attached to each link. Although conceptually these could be anywhere, our convention will locate them on the joint axes. We will first choose the location of the origin, and then the directions for two axes. The third axis follows by the right hand rule.

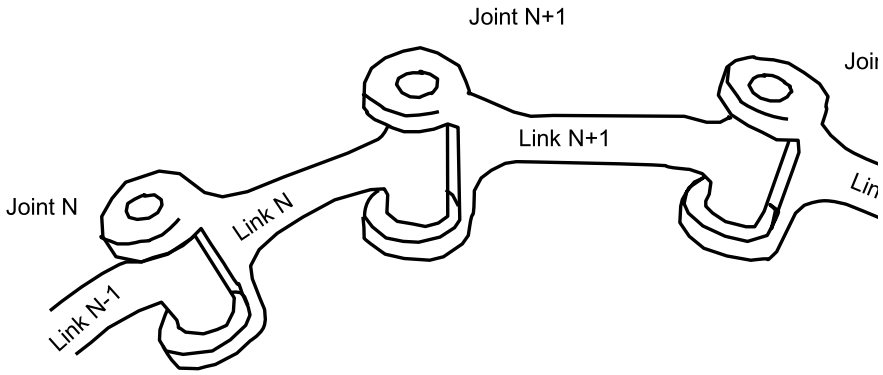
The origin of the link frame will be at the point where Axis N intersects the CN to Axis $N+1$. Let's call this point A_N. The X_N axis will point along the CN toward its intersection with the following axis. Let's call the other end of the CN B_N. Z_N will point along the joint axis. There are two ways that Z_N can point along Z_N. For rotary joints, we will require that we choose the direction of Z_N so that positive rotation of the joint is consistent with the right-hand-rule about Z_N. The points A_N and B_N are shown in Figure 3.6.

Now that X_N and Z_N are fixed, we choose Y_N simply by

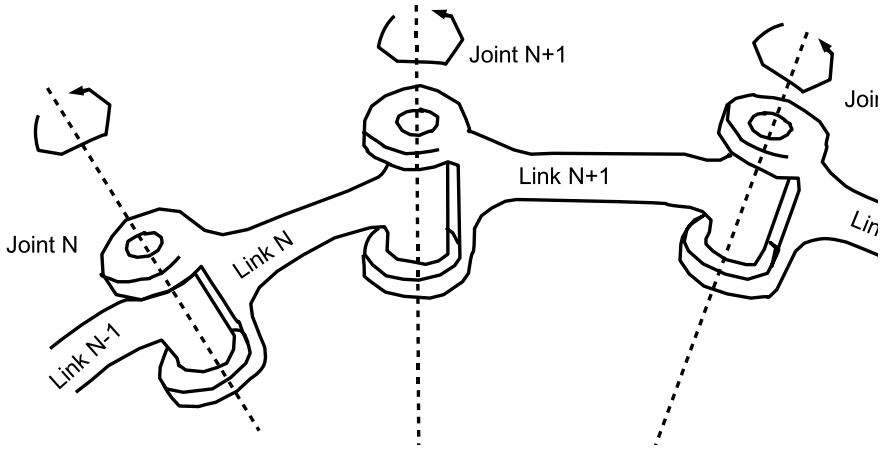
$$Y_N = Z_N \otimes X_N$$



A)

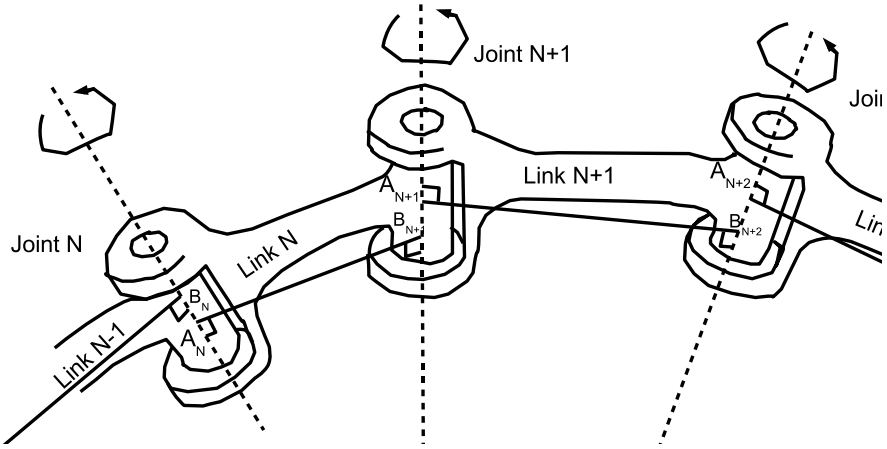


B)

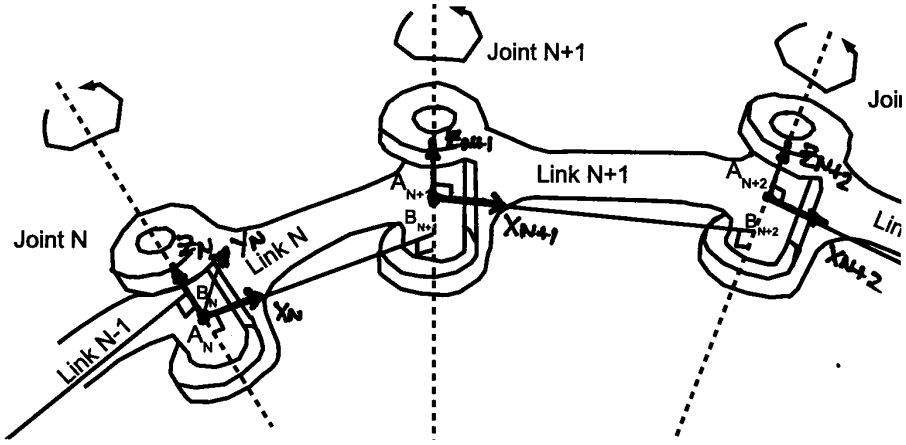


C)

Figure 3.4: Summary of steps for analyzing the DH parameters of a serial chain.



D)



E)

Figure 3.5: Figure 3.4 cont.

3.3 Summary of Link Frame Assignment Methodology

We will use the following procedure identify and assign link frames:

1. Understand the geometry and dimensions of the mechanism. Figure 3.4A shows two links (plus parts of two others) of a mechanism we will analyze.
2. Number the links and joints. The Base of the mechanism is, by convention link 0, and the first joint from the base is numbered joint 1. In Figure 3.4B we have applied numbers to the links (somewhere in the middle of the chain).
3. Identify the axes of motion and the directions which you will call positive motion (Figure 3.4C).
4. Draw common normals (CNs) and find their interesections with joint axes. Define two points on each link:

   A_N the intersection of Axis N with the CN to Axis $N+1$.

   B_N the intersection of Axis N with the CN to Axis $N-1$.

   (Figure 3.4D)
5. Assign a frame to each link as follows:

   The origin of Frame N is A_N.

   Z_N points along Axis N.

   X_N points along the CN to B_{N+1}

   Y_N completes a right handed coordinate system.

   (Figure 3.4E)

   Q: What if there is no CN because the axes intersect?

   A: Then choose X_N to be normal to Z_N and $Z_N + 1$.

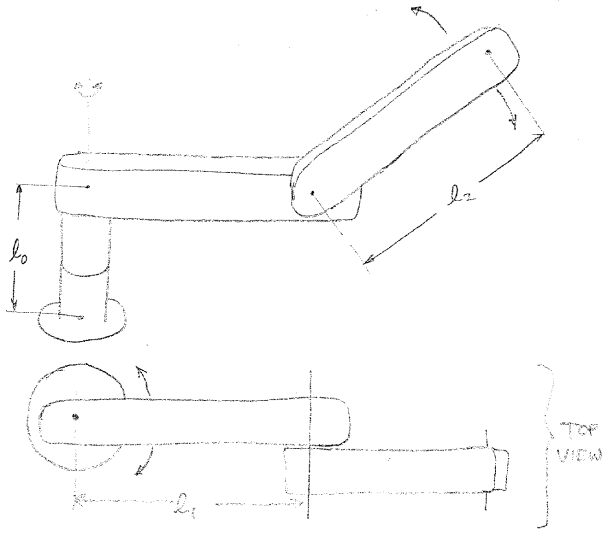
   Q: What if Z_N and $Z_N + 1$ are parallel and there is no *unique* CN?

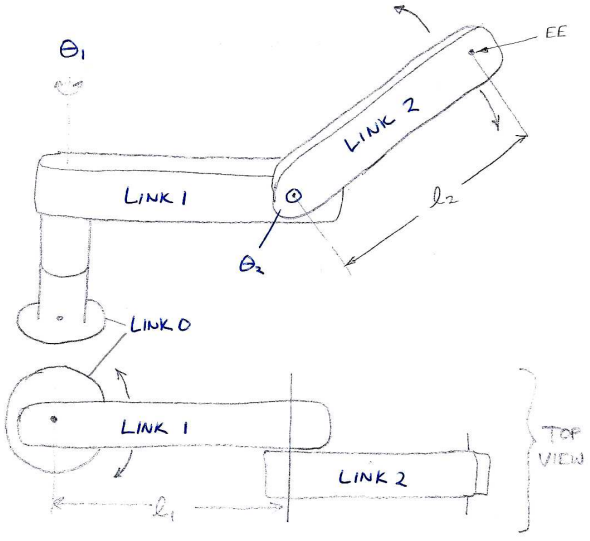
   A: Just choose one arbitrarily. Typically you can simplify subsequent analysis by choosing a CN which intersects B_N.

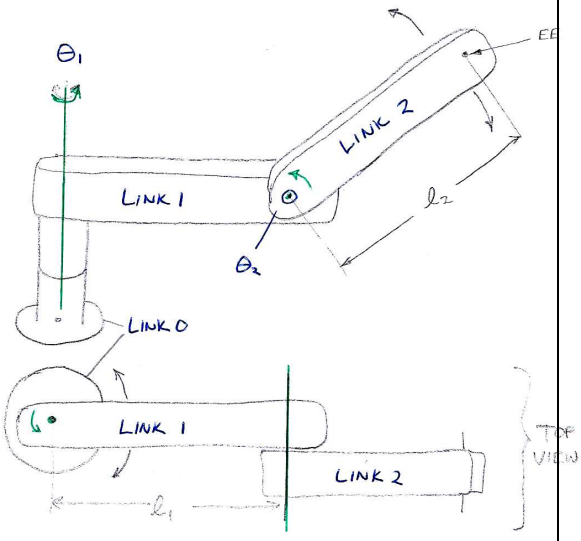
Example 3.2

Assign link frames to the arm shown below. This arm has two degrees of freedom, but we will define a third frame at the end effector as though the end effector had a degree of freedom attached.

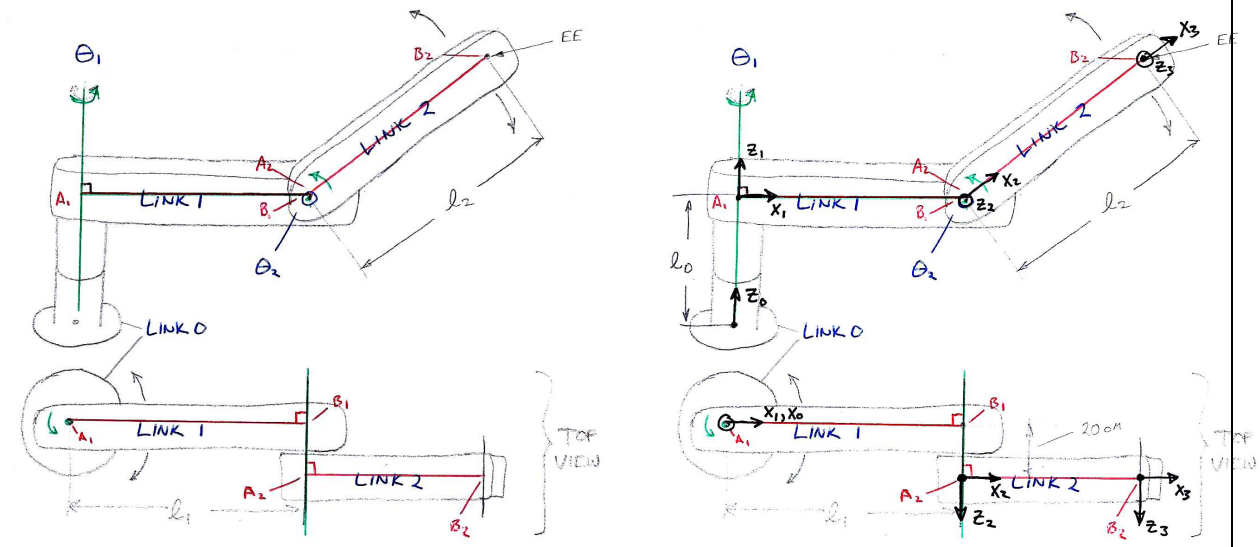
First, we identify the links and joints and number them, starting with link zero and with joint 1 between link zero and link one (blue).







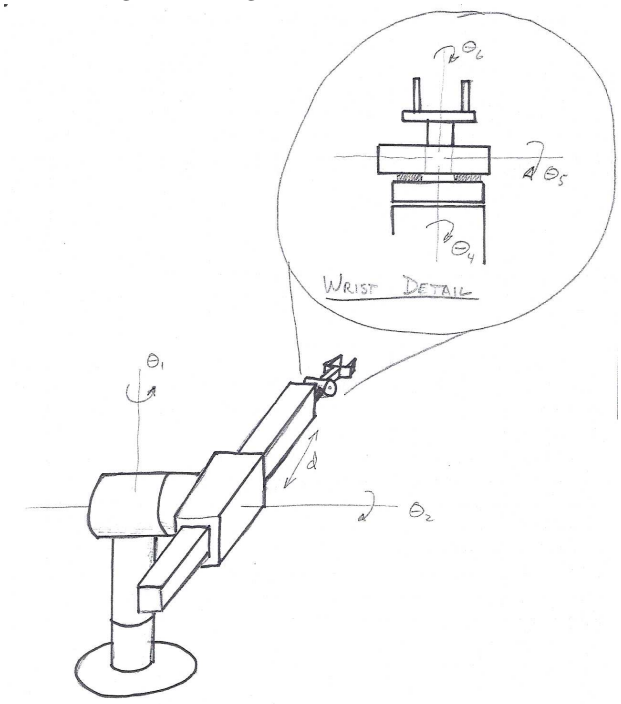
Example 3.2 cont.



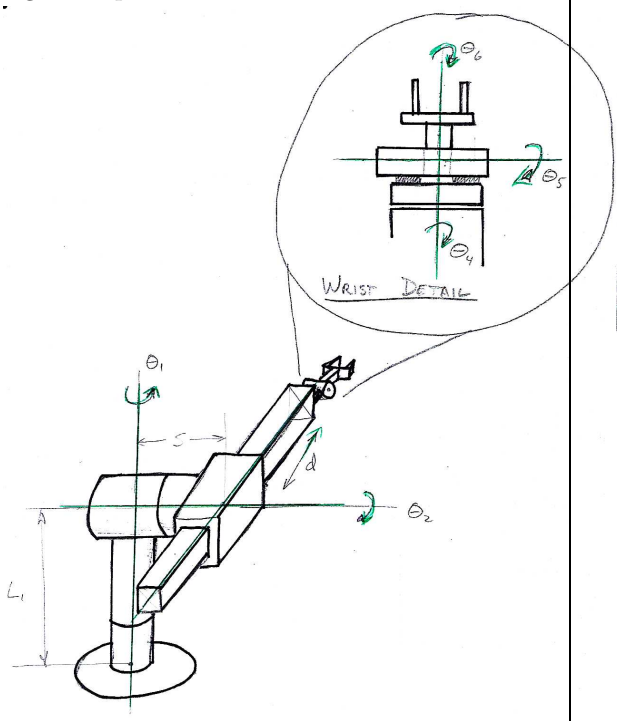
Second we identify the joint axis lines and choose directions of rotation around them (for rotary joints) which we denote to be positive (green).
Third, we draw in the common normals between the identified joint axes (red).
Finally, we identify the origins of each link frame, draw in z_i pointing so that positive rotation follows the RHR, and draw x_i along the common normal to the next axis (black).

Example 3.3

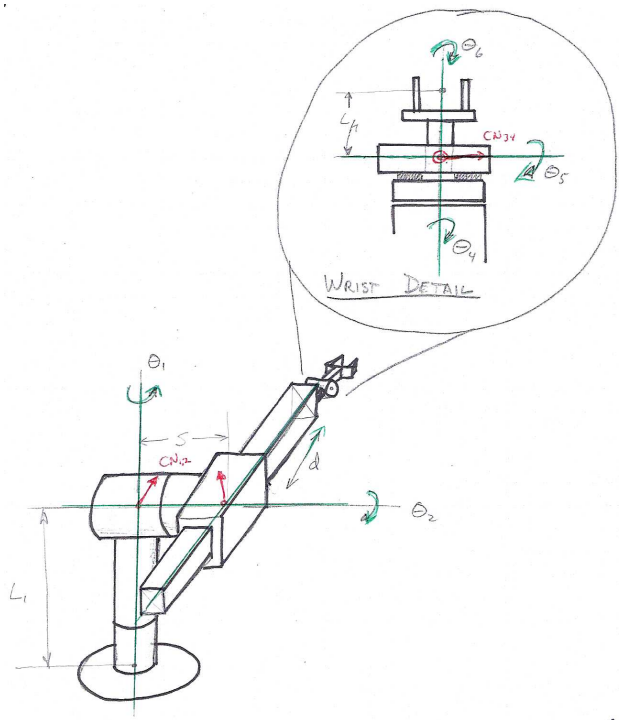
This example considers the 6 DOF arm designed by Bernard Roth, the "Stanford Arm". The series of four drawings shows the steps in analyzing and applying frames to the arm. Following the steps above in Section 3.3.
1) Understand the geometry and dimensions of the mechanism. The first drawing indicates only the shape and motions of the arm.
2) In step two we number the links and joints. It turns out that numbering the Joints is most important and we skip explicitly numbering the joints here. We add some dimensions for the shoulder height, L_1 and the shoulder offset S.
3) In green in the second drawing, we highlight the directions we will consider positive motion,
4) This arm features all intersecting axes (to simplify the equations). In drawing 3, we show in red CN vectors obtained by crossing sequential axes.
5) In drawing 4 we assign a link frame to each axis according to Step 5 of Section 3.3.



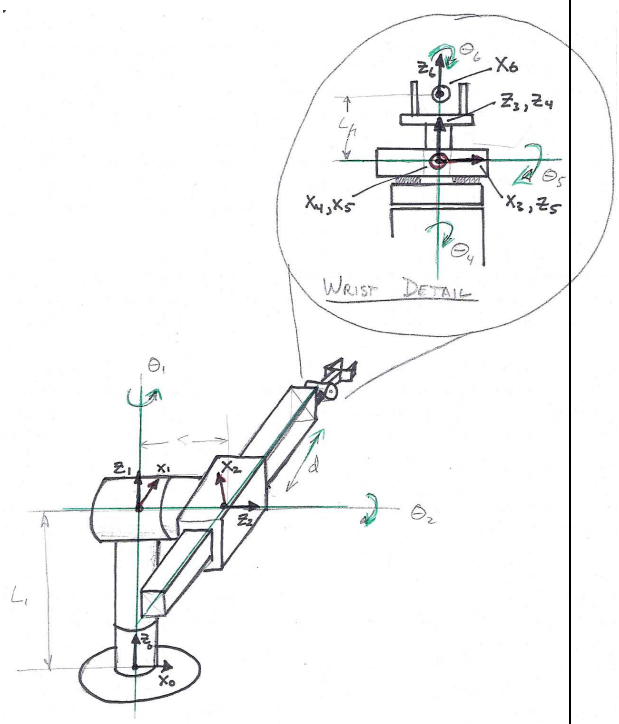
1)



2)



3)



4)

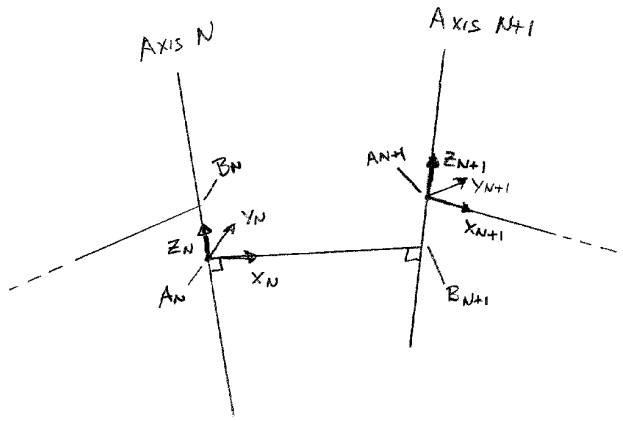


Figure 3.6: Virtual links with link frames fixed in both position and orientation.

3.4 Denavit Hartenberg Parameters

In 1955 Denavit and Hartenburg created a standardized way to derive transforms from one link to another in a serial chain. Once frames have been assigned to each link as described above, for each link, we will be able to identify two specific rotations and two translations which uniquely define the relationship between the links. Of the four Denavit-Hartenberg (DH) parameters, three are fixed constants due to the rigid body nature of the link. A fourth is a variable which specifies the relative motion between the link and the next link. For rotary joints that variable is θ_N and for prismatic joints, that variable is d_N. The definition of the four DH parameters are now given. A very nice video illustration of the DH parameters is available at the wikipedia page for "Denavit-Hartenberg Parameters".

Composition of parameters

We use the DH parameters to describe the spatial relationship between frame $N-1$ and frame N. Notice that a series of axes and CNs forms a continuous path from one link frame origin to the next. We will follow this path through its twists and turns and describe each translation and rotation with a single DH parameter. Each DH parameter represents a rotation or a translation about a single known axis. Since the origin of Frame $N-1$ is A_{N-1} we start there and work our way to the origin of the next frame. Refering to Figure 3.6, we rotate by the **link twist**: $\text{Rot}(\hat{x}, \alpha_{N-1})$, then translate by the **link length**: $\text{Trans}(\hat{x}, a_{N-1})$, then rotate by the **joint angle**: $\text{Rot}(\hat{z}, \theta_N)$, and finally translate by the **joint offset**: $\text{Trans}(\hat{z}, d_N)$. Note that the relationship between link $N-1$ and link N is described by parameters with a mix of subscripts.

For the common case of rotary joints, $a_{N-1}, \alpha_{N-1}, d_N$ will be fixed constants, dependent on the geometric design of the arm, and θ_N will be a variable joint angle. Some arms include prismatic joints in which θ_N is a constant and d_N is a variable. In evaluating the link transform we typically replace all constant parameters with their measured or design values and keep the single variable parameter in DH form. In rare cases, a joint includes two degrees of freedom such as a cylindrical shaft which can rotate as well as translate[3]. In this case it is best to abstract that joint into two separate joints, one for rotation and one for translation.

Our strategy will be to multiply together 4x4 homogeneous transforms for each of the elemental displacement and rotations above which make up the virutal link. To keep the notation under control we will use the following shorthand:

$$c\theta_j \equiv c_j \equiv cos(\theta_j)$$
$$s\theta_j \equiv s_j \equiv sin(\theta_j)$$

The complete transform is then

$$^{N-1}_{N}T = \text{Rot}(\hat{x}, \alpha_{N-1})\text{Trans}(\hat{x}, a_{N-1})\text{Rot}(\hat{z}, \theta_N)\text{Trans}(\hat{z}, d_N)$$

[3]See for example, the last two degrees of freedom of the SCARA arm.

or equivalently

$$ {}^{N-1}_{N}T = \text{Rot}(\hat{x}, \alpha_{N-1})\text{Trans}(\hat{x}, a_{N-1})\text{Rot}(\hat{z}, \theta_N)\text{Trans}(\hat{z}, d_N) $$

$$ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{N-1} & -s\alpha_{N-1} & 0 \\ 0 & s\alpha_{N-1} & c\alpha_{N-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_{N-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_N & -s\theta_N & 0 & 0 \\ s\theta_N & c\theta_N & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_N \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

or equivalently

$$ {}^{N}_{N+1}T = \text{Rot}(\hat{x}, \alpha_N)\text{Trans}(\hat{x}, a_N)\text{Rot}(\hat{z}, \theta_{N+1})\text{Trans}(\hat{z}, d_{N+1}) $$

$$ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_N & -s\alpha_N & 0 \\ 0 & s\alpha_N & c\alpha_N & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_N \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_{N+1} & -s\theta_{N+1} & 0 & 0 \\ s\theta_{N+1} & c\theta_{N+1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{N+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Each of these four matrices is a function of one of the four DH parameters. We can then multiply them together to get a single matrix which represents the link.

Generalized Link Transformation

By multiplying all four matrices together we get the following transform for any link which has DH parameters:

$$ {}^{N-1}_{N}T = \begin{bmatrix} c\theta_N & -s\theta_N & 0 & a_{N-1} \\ s\theta_N c\alpha_{N-1} & c\theta_N c\alpha_{N-1} & -s\alpha_{N-1} & -s\alpha_{N-1}d_N \\ s\theta_N s\alpha_{N-1} & c\theta_N s\alpha_{N-1} & c\alpha_{N-1} & c\alpha_{N-1}d_N \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2} $$

The reader can derive the corresponding matrix for ${}^{N}_{N+1}T$.

3.4.1 Combining links into a chain

Once we have ${}^{N-1}_{N}T$ for all of the links, we multiply them together to get the full forward kinematics model. For example in a three link robot we would have

$$ {}^{0}_{3}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T $$

Usually it is possible and desirable to multiply them together in symbolic form. We will bring this process together in the next section.

3.4.2 Summary of Forward Kinematics Analysis Part 2

1. Derive the Denavit Hartenberg parameters, $a_N, \alpha_N, d_N, \theta_N$, from your assigned link frames as follows:

   a_N is the distance from A_N to B_{N+1}. This is the distance along the CN from Z_N to Z_{N+1}. a_N is refered to as the "link length" (see Figure 3.3 and Figure 3.5 D and E).

   α_N is the angle between Axis N and Axis $N+1$ about the common normal. This is the angle between Z_N and Z_{N+1} about X_N in the RHR sense. α_N is called the "link twist".

   d_N is the distance from B_N to A_N. This is the distance from X_{N-1} to X_N along Z_N. d_N is referred to as the "joint offset".

   θ_N is the angle between CN_{N-1} and CN_N around Axis N. This is the angle between X_{N-1} and X_N around Z_N. θ_N is refered to as the "joint angle".

2. Use equation 3.2 to compute the link transform for each link

3. Multiply all the link transforms together.

Example 3.4
Find the 4x4 matrix representing the forward kinematic equations of the manipulator of Example 3.2.
Step 1. Referring to the procedure of Section 3.4.2

N	α_{N-1}	a_{N-1}	d_N	θ_N
1	0	0	l_0	θ_1
2	90°	l_1	20cm	θ_2
3	0	l_2	0	0

Step 2
Referring to Equation 3.2

link 1)

$$ {}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & l_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$c\alpha_{N-1} = c\alpha_0 = 1$
$s\alpha_{N-1} = s\alpha_0 = 0$

link 2)

$c\alpha_{N-1} = c\alpha_1 = 0$
$s\alpha_1 = 1$

$$ {}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & l_1 \\ 0 & 0 & -1 & -20cm \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

link 3)

$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$\alpha_2 = 0 \quad c\alpha_2 = 1$

Example 3.4 cont.
Step 3
Now multiplying the three matrices together we get:

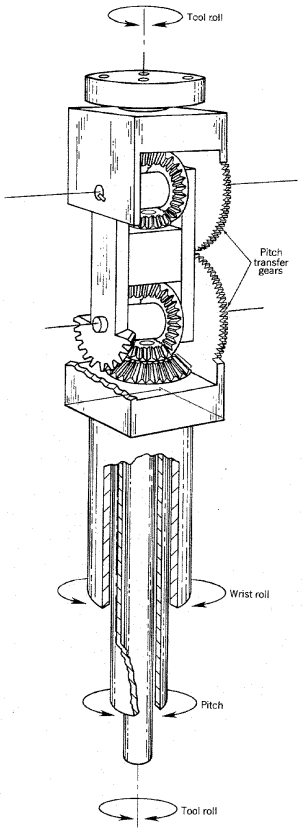
$$
{}^{0}_{2}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & l_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & l_1 \\ 0 & 0 & -1 & -20cm \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

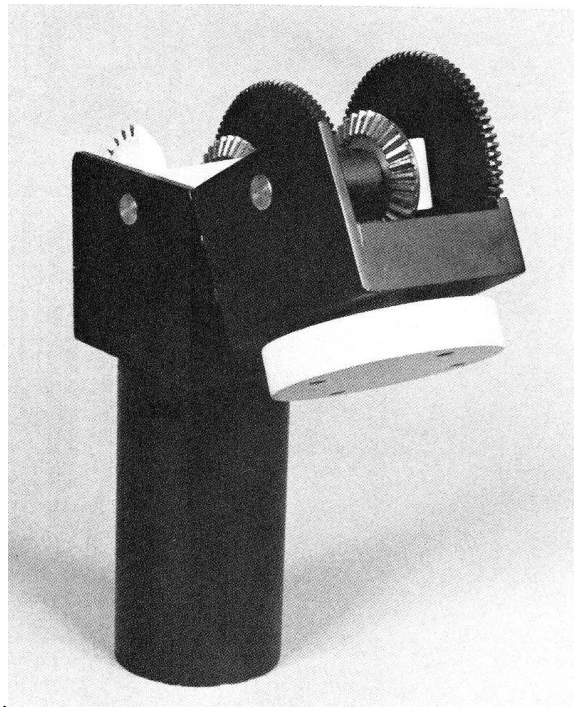
$$
\begin{bmatrix} c_1c_2 & -c_1s_2 & s_1 & l_1c_1 + 20cm\, s_1 \\ s_1c_2 & -s_1s_2 & -c_1 & l_1s_1 - 20cm\, c_1 \\ s_2 & c_2 & 0 & l_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{0}_{3}T = \begin{bmatrix} c_1c_2 & -c_1s_2 & s_1 & l_1c_1 + 20cm\, s_1 \\ s_1c_2 & -s_1s_2 & -c_1 & l_1s_1 - 20cm\, c_1 \\ s_2 & c_2 & 0 & l_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
= \begin{bmatrix} c_1c_2 & -c_1s_2 & s_1 & l_1c_1 + 20cm\, s_1 + l_2c_1c_2 \\ s_1c_2 & -s_1s_2 & -c_1 & l_1s_1 - 20cm\, c_1 + l_2s_1c_2 \\ s_2 & c_2 & 0 & l_0 + l_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

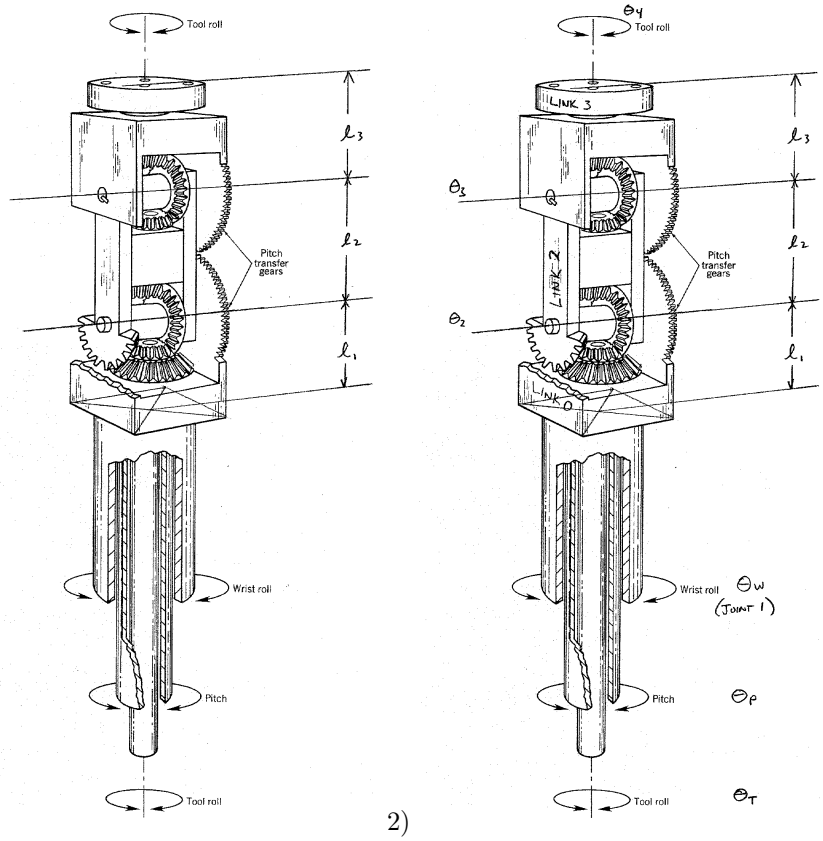
Example 3.5
Find link frame assignments and Denavit Hartenberg parameters for the Rosheim Prehensile Wrist[a]. The following interesting wrist mechanism has an extraordinary range of pitch orientations.



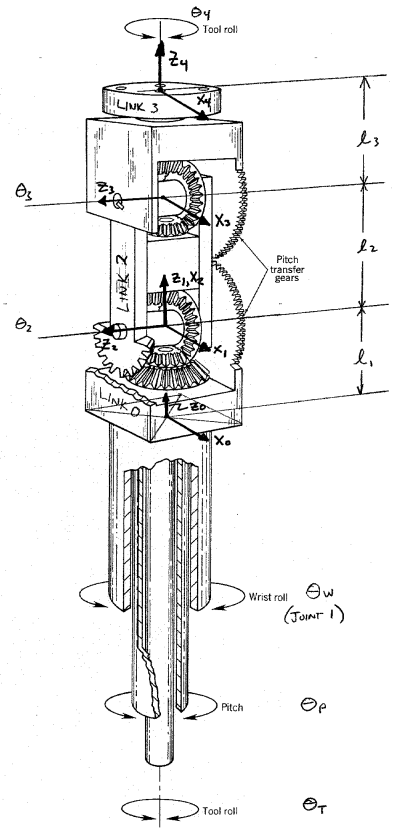


[a]Mark Rosheim, "Robotic Wrist Actuators," John Wiley & Sons, 1989.

Example 3.5 cont.
Following the steps outlined above:
1) Understand the geometry and dimensions. First, we arbitrarily identify the bottom of the rectanglar solid below the first bevel gears as a reference plane (i.e. we will locate Frame 0 there in the next step). We draw diagonals in that plane to locate its center. We also extend the joint axis lines through the joints and give names to the dimentions between them:
2) Number the Links and Joints. Furthermore, the Wrist roll axis intersects the first horizontal axis (which we will call joint 2 in the next step) so that link 1 has zero length:



Example 3.5 cont.
3) Identify axes of motion. This is a bit tricky since the "Pitch Transfer Gears" demand that some of the joints do not move independently. However for now we will ignore the pitch transfer gears and consider the two axes to be independent joints. We will ignore the sign convention which places the Z axis along the direction of positive motion and simply point them to the left and up.
4) Draw common normals. The common normal between Z_0 and Z_1 is zero length. The CN between Z_1 and Z_2 is also zero length. The CN from Z_2 to Z_3 is the line up the center of the drawing of length l_2. The CN between axis Z_3 and Z_4 is zero length.
5) Assign a frame to each link. Note that F_0 could go anywhere on the "Wrist roll" body, but we place it along the center axis for convenience. Note that when we move around the Wrist roll axis, F_0 stays where it is and F_1 moves relative to it.



6) Following the procedures of Section 3.4.2, we obtain

N	α_{N-1}	a_{N-1}	d_N	θ_N
1	0	0	l_1	θ_W
2	$\pi/2$	0	0	$\pi/2+\theta_p$
3	0	l_2	0	$-\pi/2+\theta_p$
4	$-\pi/2$	0	l_3	θ_4

Note that joints 2 and 3 form a single degree of freedom driven by θ_p because they are coupled together by the pitch transfer gears.

3.5 Spaces

We have seen that the forward kinematics analysis generates a 4x4 matrix which is a function of N variables where N is the number of joints. As the joints move to different values, the 4x4 matrix specifies different positions and orientations in the task space corresponding to a frame on the last link usually known as the end effector.

Although this matrix maps points in the end effector coordinate system to the base coordinate system, it is sometimes useful to think of this matrix as a description of a mapping from a set of joint values to an end effector configuration. For a manipulator with N joints, the joint values form a point in a N dimensional space we call the *Joint Space*.

The 4x4 matrix ${}^0_N T$ gives us a rotation matrix and X,Y,Z coordinates of the end effector. This is termed a "configuration" of the manipulator end effector. We commonly derive three orientation parameters such as roll, pitch, yaw, angles from the upper 3x3 part of ${}^0_N T$ to give six numbers which characterize the end effector configuration:

$$\begin{bmatrix} X \\ Y \\ Z \\ \text{roll} \\ \text{pitch} \\ \text{yaw} \end{bmatrix} \tag{3.3}$$

The above six numbers define a point in *End Effector Space*. End Effector Space is sometimes called "Task Space" since it is a natural space in which to represent tasks or "Configuration Space" although this term has a slightly different meaning in the context of motion planning.

As a manipulator moves around, a point moves around in joint space representing the values of its joints, and another point moves around in end effector space representing the configuration of the end effector. As we shall see in the next chapter, there may be more than one joint space point which corresponds to the same end effector point, but there is always exactly one point in end effector space for each point in joint space.

Another relevant space is *Actuator Space*. In most serial robots the actuators are located directly on each joint and actuator motion is a linear function of or is equal to the joint motion. However in certain designs, such as the Yakusawa Motoman L-3, actuators may drive the joints through linkages and therefore move somewhat differently than the joints of the serial chain. For good discussion of this consult Chapter 3 of Craig.

Chapter 4

Inverse Kinematics

4.1 Problem Statement and Learning Objectives

Problem Statement The inverse kinematics problem is to find the joint values for a given end effector configuration. The solution to this problem is needed for most practical applications of serial robot arms.

Learning Objectives Upon completing this Chapter, the reader should

- Be able to plot the workspace of a serial manipulator with or without joint limits.
- Be able to solve for the joint angles of a planar robot given the end effector x, y position and orientation.
- Be able to solve for the joint angles of a spatial 3 DOF robot given the x, y, z position of the end effector.
- Be ready to tackle full scale inverse kinematics problems if sufficient time is available.

4.2 Overview

We can express the forward kinematic problem as

$$f(\theta) = {}^0_6T$$

Then, to find the joint angles which correspond to a desired end effector configuration, we need to solve the inverse of this problem, the *Inverse Kinematic Problem*

$$\theta = f^{-1}({}^0_6T_d)$$

where the d subscript indicates the "desired" end effector configuration. Being able to solve the inverse kinematic problem is thus very important for automatic applications of robot manipulators since the control system must know the joint angles, θ, in order to move the end effector to ${}^0_6T_d$.

For serial kinematic chains, this problem is harder than the forward kinematic problem. Among the significant issues are:

- Existence of solutions
- Multiple solutions
- How to find the solution(s)

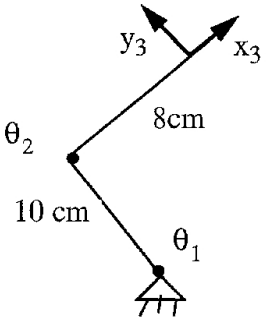
4.2.1 Workspace

A robot arm, just like the human arm, can only reach a certain number of points. Wrapped up in our inverse kinematics calculation is the question of whether or not the desired end effector configuration is actually reachable. Our mathematics should tell us this automatically when we try to compute the joint angles or displacements. Fortunately, it works out that for a solution to the inverse kinematics problem to exist, ${}^0_6T_d$ must be in the *workspace* of the manipulator. We will use two definitions of workspace:

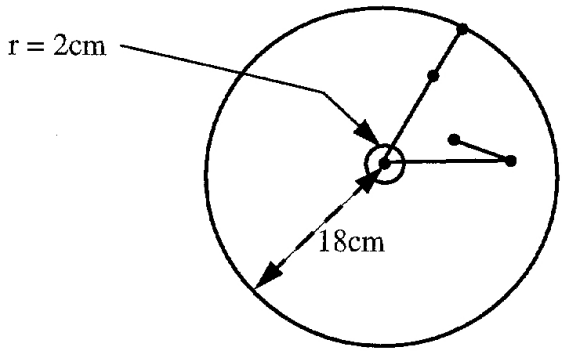
1. **Dexterous Workspace**. The subset of space in which the robot end effector (EE) can reach all orientations.
2. **Reachable Workspace** The subset of space in which the robot end effector (EE) can reach at least one orientation.

Note that the dexterous workspace is a subset of the reachable workspace.

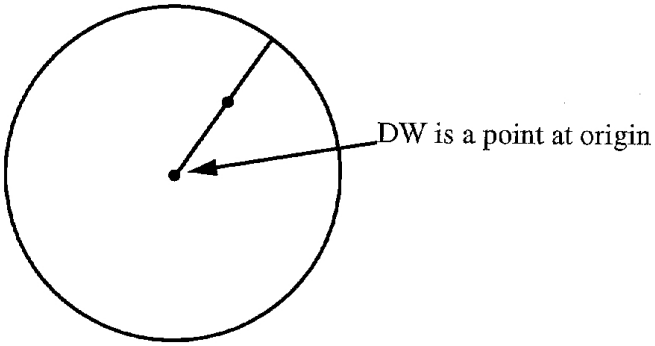
Example 4.1
Consider a two link planar robot with no joint limits:



Then the reachable workspace is:



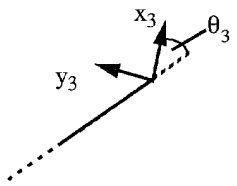
and the dexterous workspace is the empty set.
Now, if $l_1 = l_2 = 9$, the outer radius is the same, and



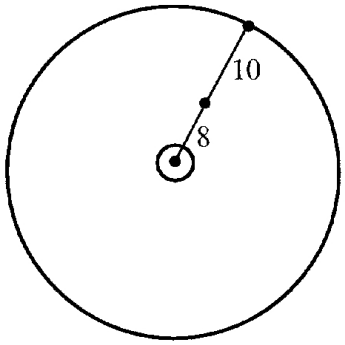
Now the RW is the same but the DW exists at a point at the origin where any orientation can be obtained.

Example 4.1 cont.

Now let's add a joint at the wrist



and set link lengths to $l_1 = 8, = l_2 = 10$.



Now the reachable and dexterous workspaces are the same.

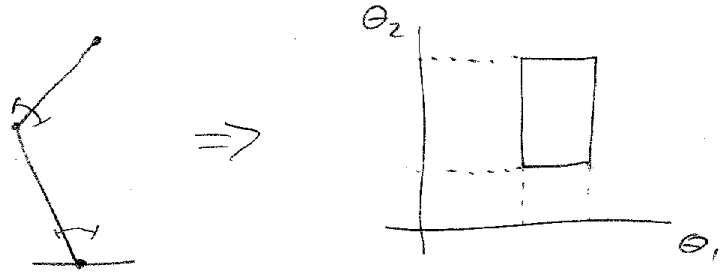
Joint Limits The joints in real mechanisms often cannot achieve rotations of $0 - 2\pi$ (or for that matter infinite displacements on prismatic joints!). Thus joint limits

$$\theta_{imin} \leq \theta_i \leq \theta_{imax} \qquad d_{imin} \leq d_i \leq d_{imax}$$

contribute additional limits to the workspace.

Rotary Joints

Without joint limits, the 2-link planar robot makes a donut shaped reachable workspace in the task space (Example 1). In joint space, any point can be attained. When joint limits are introduced, a rectangular solid is created defined by θ_{imin} and θ_{imax} for each dimension. To convert this rectangular solid back to task space, we note that each edge of the solid corresponds to varying just one joint at a time. The edges in the joint space thus become arcs in the task space. For the 2 joint manipulator, these limits allow joint configurations inside the rectangle shown below, right.



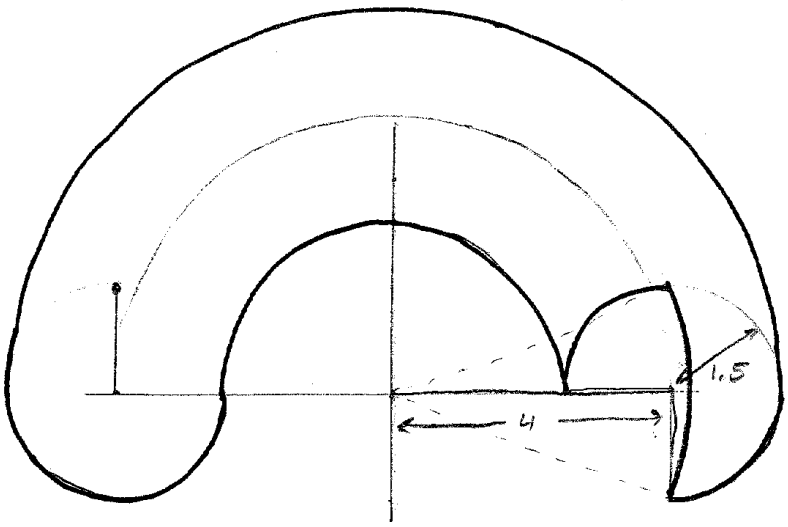
Plotting the workspace can be tricky with joint limits. One approach:

1. Identify the corners (vertices) of the joint space rectangular solid.
2. Plot a point in the task space for each vertex.
3. Connect the points with arcs (use a compass to draw) around the joint axis to connect the vertices.

4. Remember that some workspace boundaries may not occur at joint limits such as when the arm is fully stretched out.

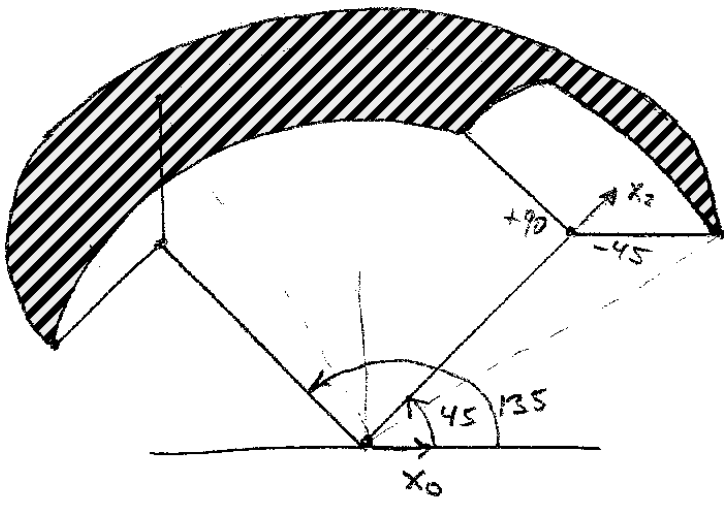
Example 4.2
Consider a 2DOF planar manipulator with link lengths $l_1 = 4, l_2 = 1.5$. Assume the joints are limited to

$$0 \leq \theta_1 \leq 180^\circ \qquad -90^\circ \leq \theta_2 \leq 180^\circ$$

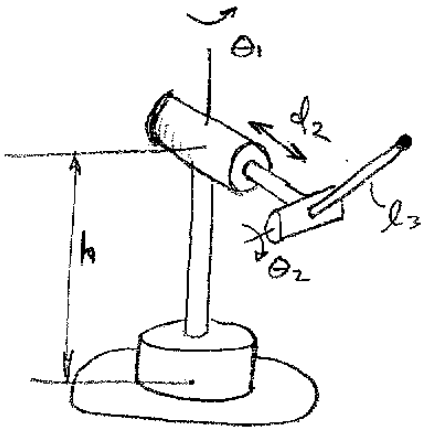


Example 4.3
Same problem as previous example: $l_1 = 3, l_2 = 1.5$. Assume the joints are limited to

$$45^\circ \leq \theta_1 \leq 135^\circ \qquad -45^\circ \leq \theta_2 \leq 90^\circ$$

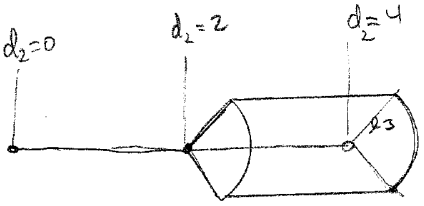


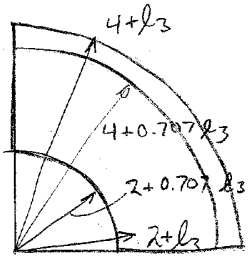
Example 4.4
Generate top view and perspective view of the 3D workspace of the manipulator shown:

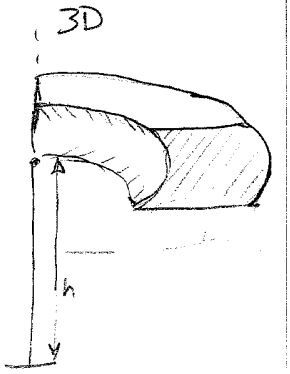


Where the joint limits are

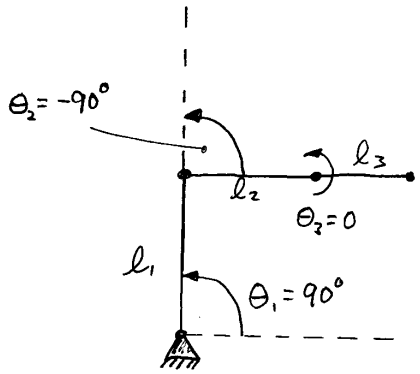
$$0 \leq \theta_1 \leq 90^\circ \qquad 2 < d_2 < 4 \qquad -45^\circ < \theta_3 < 45^\circ$$







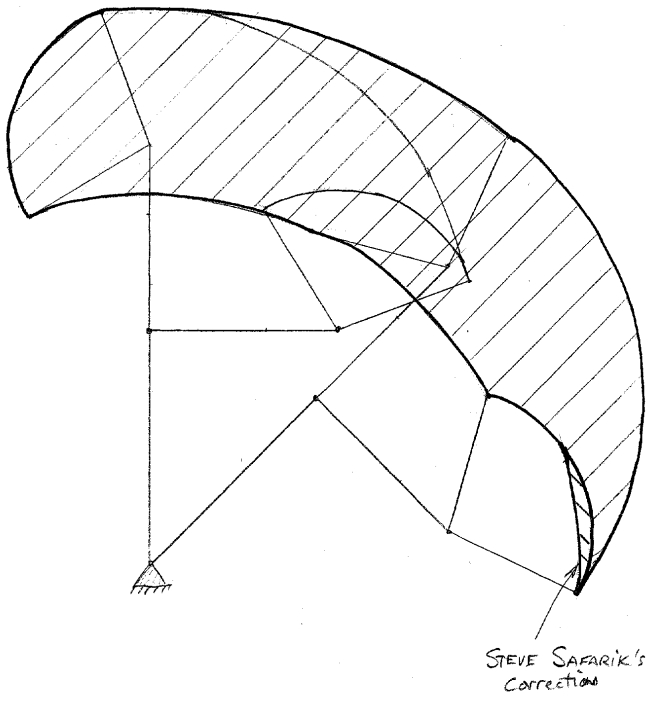
Example 4.5
Draw the 2-D workspace of the planar manipulator drawn below:



Facts:

- $45° < \theta_1 < 90°$
- $-90° < \theta_2 < 0°$
- $20° < \theta_3 < 120°$
- $l_1 = 5, \quad l_2 = 4, \quad l_3 = 3$

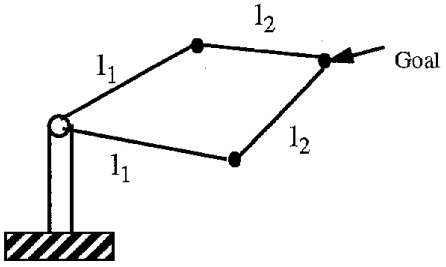
Solution:



4.2.2 Multiple Solutions

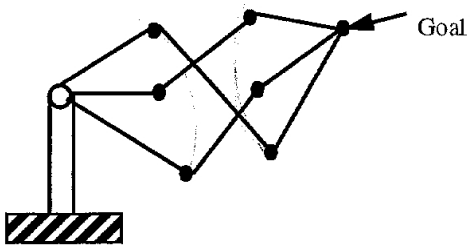
We will consider two important cases: first, when the number of degrees of freedom of the robot are equal to the number in the task, and second when the number of degrees of freedom in the robot are greater than in the task.

1) Consider a 2 link planar arm.



This arm has two joints. We consider its task to be completely specified by the position of its end effector in the plane. Thus the number of degrees of freedom and the dimensionality of the task are both 2. In this case there are a finite number of solutions (2) which are illustrated above. The planar two-joint arm can reach the goal in two configurations referred to as "elbow-up" and "elbow-down." Each of these configurations is a separate solution to the inverse kinematics problem.

2) Consider a 3-link planar arm with the same goal:



In this case there are an infinite number of solutions (three are shown). The robot can even move itself around without moving the EE from the goal. This is called self-motion. Case 2 will not be considered further in this course.

4.2.3 Methods of Solution

There are three principal ways to obtain the inverse kinematics solutions:

1. "Closed Form": An analytic expression for θ as a function of ${}^0_6T_d$ which includes all solutions.
2. Numerical methods.
3. Hybrid approach in which some degrees of freedom are solved in closed form and others numerically.

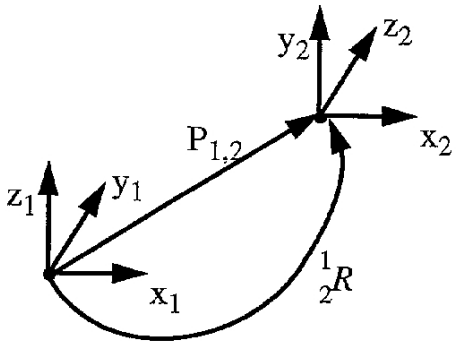
We will concentrate here on the closed form solution.

There are two standard approaches to obtaining the closed form solution, the algebraic approach, primarily involving manipulation of the forward kinematic equations, and the geometric approach, in which the inverse kinematics problem is reduced to one or more plane geometry problems. However, unlike the forward kinematics problem, there is no straightforward procedure which can be followed to get analytical solutions.

4.3 Inverse Kinematics Tools

4.3.1 Inverse of a Homogeneous Transform

Let's define two frames which are related by a translation $P_{1,2}$ and a rotation ${}^1_2R$.



If we represent this spatial relationship by a homogeneous transform, T, then it is reasonable to assume that an inverse, T^{-1} must always exist

Q: Why?

A: One reason: a physical move can always be "undone." Another reason[1], eigenvalues of $T = 1, e^{j\theta}, e^{-j\theta}$, so their product, (the determinant) is always non-zero.

If the homogeneous transform is

$$ {}^1_2T = \begin{bmatrix} \begin{bmatrix} & {}^1_2R & \end{bmatrix} & \begin{bmatrix} {}^1P_{1,2} \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 \end{bmatrix} \end{bmatrix} $$

then it is easy to show that its inverse is

$$ {}^1_2T^{-1} = {}^2_1T = \begin{bmatrix} \begin{bmatrix} & {}^2_1R & \end{bmatrix} & \begin{bmatrix} -{}^2P_{1,2} \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 \end{bmatrix} \end{bmatrix} $$

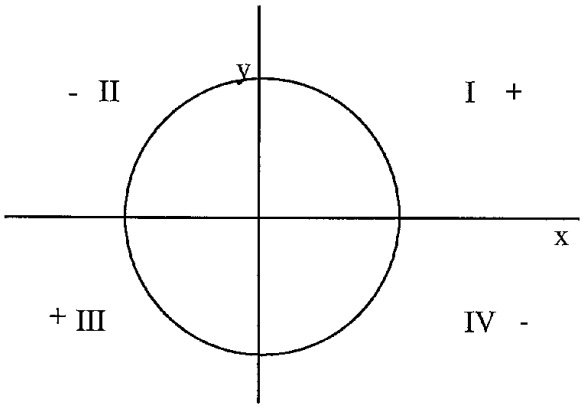
where ${}^2_1R = {}^1_2R^T$ and $-{}^2P_{1,2} = {}^2_1R\left(-{}^1P_{1,2}\right)$

4.3.2 Atan2(y,x)

We need a slightly more sophisticated form of the arctan() function to find angles. Consider a trivial robot arm with a single link of unit length with a rotary joint at the origin. If we pick a point on the unit circle, then the inverse kinematics problem is simply solved by the arctangent. However, the arctan function domain is limited to the first and fourth quadrants:

$$ -\frac{\pi}{2} \leq \arctan(\frac{y}{x}) \leq \frac{\pi}{2} $$

In real-world problems, we need to solve the inverse kinematics problem for any quadrant.



The **atan2(y,x)** function, also known as the four quadrant arctangent, returns an angle between 0 and 2π allowing the solution to occupy any of the four quadrants.

[1] J.M. McCarthy, "Introduction to Theoretical Kinematics," MIT Press, 1990.

4.3.3 Key Trigonometric Identities

Some trig identities (which may have faded from your mind) will be very useful in solving inverse kinematics problems.

Sum of Angles

If $c_{12} = \cos(\theta_1 + \theta_2)$ and $s_{12} = \sin(\theta_1 + \theta_2)$, then

$$c_{12} = c_1 c_2 - s_1 s_2$$

and

$$s_{12} = c_1 s_2 + s_1 c_2$$

This offers a way to solve the following problem which frequently comes up in inverse kinematics:

"Given k_1, k_2, and x, solve $x = k_1 \cos(\theta_1) + k_2 \sin(\theta_1)$ for θ_1."

Here is one way:

Let $r = \sqrt{k_1^2 + k_2^2}$ and let $\theta_2 = \text{atan2}(k_1, k_2)$. In other words

$$k_1 = r s_2, \quad k_2 = r c_2$$

Rewriting the problem with this change of variables and applying sum-of-angles:

$$x = r s_2 c_1 + r c_2 s_1 = r s_{12}$$

$$\frac{x}{r} = \sin(\theta_1 + \theta_2) = s_{12}$$

We can solve this with the inverse sine function, $\sin^{-1}()$, but there is a preference in the robotics literature to transform it further into an atan2() solution by:

$$c_{12} = \pm\sqrt{1 - s_{12}^2} = \pm\sqrt{1 - \left(\frac{x}{r}\right)^2}$$

$$\theta_1 + \theta_2 = \text{atan2}(s_{12}, c_{12})$$

$$\theta_1 = \text{atan2}\left(\frac{x}{r}, \pm\sqrt{1 - \left(\frac{x}{r}\right)^2}\right) - \text{atan2}(k_1, k_2)$$

Note that there are two solutions to this equation corresponding to the $\pm$ choice of the square root.

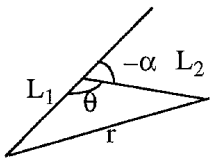
A joint angle can only be a real number for a reachable pose. For the real solution to exist, the argument of the square root must be real giving

$$\frac{x^2}{r^2} \leq 1, \quad x^2 \leq r^2, \quad |x| \leq |r|, \quad |x| \leq +\sqrt{k_1^2 + k_2^2}$$

So, if $|x| > \sqrt{k_1^2 + k_2^2}$, no solution exists. This is a mathematical test we can apply during computation to check whether or not the manipulator is capable of reaching the desired point. However this is not the only condition which must be met in practice because we are still considering the idealized case where the joints can take on any value. In real manipulators, joint limits prohibit some points from being reached even if the inverse kinematics solution exists.

4.3.4 Law of Cosines

This old workhorse is a way to solve a frequently arising problem in inverse kinematics when an arm has two parallel axes



The law states:

$$r^2 = L_1^2 + L_2^2 - 2L_1L_2\cos(\theta)$$

for an alternate form, we can use the fact that $\theta = \pi + \alpha$ to get

$$r^2 = L_1^2 + L_2^2 + 2L_1L_2\cos(\pm\alpha)$$

This has two values since $\cos(\alpha) = \cos(-\alpha)$.

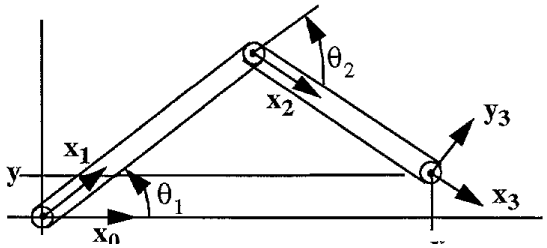
4.3.5 Manipulator Sub-Space

When the robot has fewer than six DOF, but the task has six DOF, then there are no solutions to the general problem, but only special cases. However, it is often useful to specify a subspace of the full 6 DOF configuration space which has a dimensionality which matches the robot arm. For example, if we restrict our desired EE positions to the plane, then a planar arm is capable of reaching them. This plane, embedded in the six DOF space of possible rigid body configurations, in an example of a manipulator sub-space.

Alternatively, we can think of a planar world in which the configuration of any object consists of the x and y positions, and α, the orientation of the object.

Example 4.6

Consider the following arm in a planar world which can reach various x, y positions, but has a fixed wrist so that frame 3 has a particular orientation depending on the angles θ_1 and θ_2. Because of this dependence, the arm can only reach two specific orientations corresponding to the "elbow-up" and "elbow-down" solutions.



Suppose our task is described by

$${}^0_3T = \begin{bmatrix} \begin{bmatrix} & {}^0_3R & \end{bmatrix} & \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & 1 \end{bmatrix}$$

Since there is no joint at the wrist of this robot, ${}^0_3R = {}^0_3R(x, y)$ is a multivalued function of x, y. We call

$${}^0_3T_D(x, y) = \begin{bmatrix} \begin{bmatrix} & {}^0_3R(x, y) & \end{bmatrix} & \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & 1 \end{bmatrix}$$

the "Manipulator subspace." In more detail, we can only reach orientations in the x, y plane so ${}^0_3R$ has the form

$${}^0_3R(x, y) = \begin{bmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where α is a two-valued function of x, y. If we use the algebraic approach, we would set up the problem:

$${}^0_3T_D(x, y) = {}^0_1T(\theta_1){}^1_2T(\theta_2){}^2_3T$$

The manipulator sub-space is not always easy to find. In the case of Example 6, the inverse kinematics problem is more easily solved with a geometric approach and does not require a manipulator subspace anyway. Sometimes 6 DOF robots are easier to solve than 5DOF robots because they do not require a manipulator subspace to be derived.

4.4 Algebraic Solution

4.4.1 Strategy

The forward kinematics analysis of an all rotary 6DOF arm yields an equation

$$ {}^0_6T \quad = \quad {}^0_1T(\theta_1) \quad {}^1_2T(\theta_2) \quad {}^2_3T(\theta_3) \quad {}^3_4T(\theta_4) \quad {}^4_5T(\theta_5) \quad {}^5_6T(\theta_6) $$

In the inverse kinematics version of the problem, ${}^0_6T$ is known and we can call it ${}^0_6T_D$ (meaning desired), and the θ_i are unknowns. The equation

$$ {}^0_6T_D = {}^0_6T $$

is really 12 individual equations, one for each element of the first three rows. On inspecting those 12 equations we might find one which can easily be solved for θ_1 or perhaps a pair of equations which could jointly be solved for θ_1. But all of the other unknowns are mixed into complicated equations with no apparent solution. However, at that point ${}^0_1T(\theta_1)$ becomes a known matrix. We can then write

$$ {}^0_1T(\theta_1)^{-1} \quad {}^0_6T \quad = {}^1_2T(\theta_2) \quad {}^2_3T(\theta_3) \quad {}^3_4T(\theta_4) \quad {}^4_5T(\theta_5) \quad {}^5_6T(\theta_6) $$

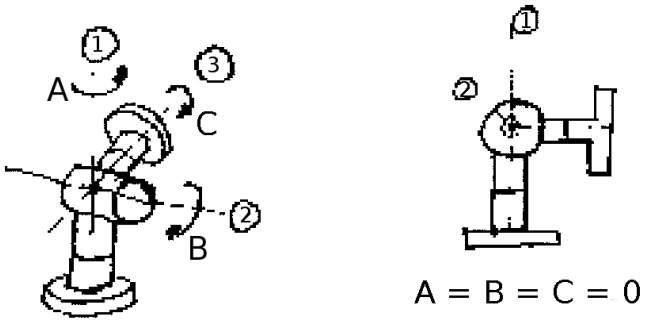
which generates a fresh set of 12 equations in which all the elements of the left hand side are knowns. We may now find a new equation or set of equations which we can solve for θ_2. When θ_2 is known, we can write

$$ {}^1_2T(\theta_2)^{-1} \quad {}^0_1T(\theta_1)^{-1} \quad {}^0_6T = {}^2_3T(\theta_3) \quad {}^3_4T(\theta_4) \quad {}^4_5T(\theta_5) \quad {}^5_6T(\theta_6) $$

Although there are several "might"s in this description, because of the serial nature of the arms, this method works as described more often than one might think.

4.4.2 Examples

Example 4.7
Consider a mechanism which corresponds to the z, y, x Euler angles (Section 2.5.2).



This mechanism has three axes of rotation which intersect at a single point. The axes are arranged so that the first rotation about z, moves the x and y axes for subsequent rotations which corresponds to the definition of Euler angles. The last frame, frame 3, has no offset from the point where the axes intersect, so the homogeneous transform consists only of rotation with no translation. The amounts of rotation about z, y, x are A, B, C. Expanding the 3x3 rotation matrix of Section 2.5.2 into a 4x4 homogeneous transform with zero translation, we get

$$ {}^0_3T(A,B,C) = \begin{bmatrix} cAcB & cAsBsC - sAcC & cAsBcC + sAsC & 0 \\ sAcB & sAsBsC + cAcC & sAsBcC - cAsC & 0 \\ -sB & cBsC & cBcC & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The manipulator subspace for this manipulator is

$$ {}^0_3T_D = \begin{bmatrix} \multicolumn{3}{c}{\multirow{3}{*}{R}} & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ & & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Where R can be any rotation matrix because we can reach nearly any orientation with ZYX Euler angles (see below). The inverse kinematics problem can then be set up to solve

$$ {}^0_3T_D = {}^0_3T(A,B,C) $$

for A, B, C, given ${}^0_3T_D$, Expanding this equation gives

$$ \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} cAcB & cAsBsC - sAcC & cAsBcC + sAsC \\ sAcB & sAsBsC + cAcC & sAsBcC - cAsC \\ -sB & cBsC & cBcC \end{bmatrix} $$

Remember that everything on the left hand side is known as part of the desired EE configuration, and the unknowns are A, B, C.

Example 4.7 cont.
Looking over these 9 equations, we find fairly simple forms in the upper left hand corner:

$$r_{11} = cAcB, \qquad r_{21} = sAcB$$

We can add the squares of these equations together to get

$$r_{11}^2 + r_{21}^2 = cB^2(cA^2 + sA^2)$$

or

$$cB = \pm\sqrt{r_{11}^2 + r_{21}^2}$$

We can use more information from ${}^0_3T_D$ since $r_{31} = -sB$ and use atan2:

$$B = \text{atan2}(-r_{31}, \pm\sqrt{r_{11}^2 + r_{21}^2})$$

if $B \neq \{\pi/2, -\pi/2\}$ then cB is a non-zero constant and we can get

$$A = \begin{cases} \text{atan2}(r_{21}, r_{11}) & cB > 0 \\ \text{atan2}(-r_{21}, -r_{11}) & cB \leq 0 \end{cases}$$

Note that atan2(ay, ax) = atan2(y,x) if $a > 0$ and that $\text{atan2}(y, x) \neq \text{atan2}(-y, -x)$.
Similarly

$$r_{32} = cBsC, \qquad r_{33} = cBcC$$

if $B \neq \{\pi/2, -\pi/2\}$ then cB is a non-zero constant and we can get

$$C = \begin{cases} \text{atan2}(r_{32}, r_{33}) & cB > 0 \\ \text{atan2}(-r_{32}, -r_{33}) & cB \leq 0 \end{cases}$$

Alternatively we can express the multiple solutions slightly differently:

$$C = \begin{cases} \text{atan2}(r_{32}, r_{33}) & cB > 0 \\ \text{atan2}(r_{32}, r_{33}) + \pi & cB \leq 0 \end{cases}$$

Example 4.8
Solve for A, B, C in the wrist of Example 7 where:

$$ {}^{0}_{3}T_{D} = \begin{bmatrix} 0.925 & 0.018 & 0.379 \\ 0.163 & 0.883 & -0.441 \\ -0.342 & 0.470 & 0.814 \end{bmatrix} $$

Solution:

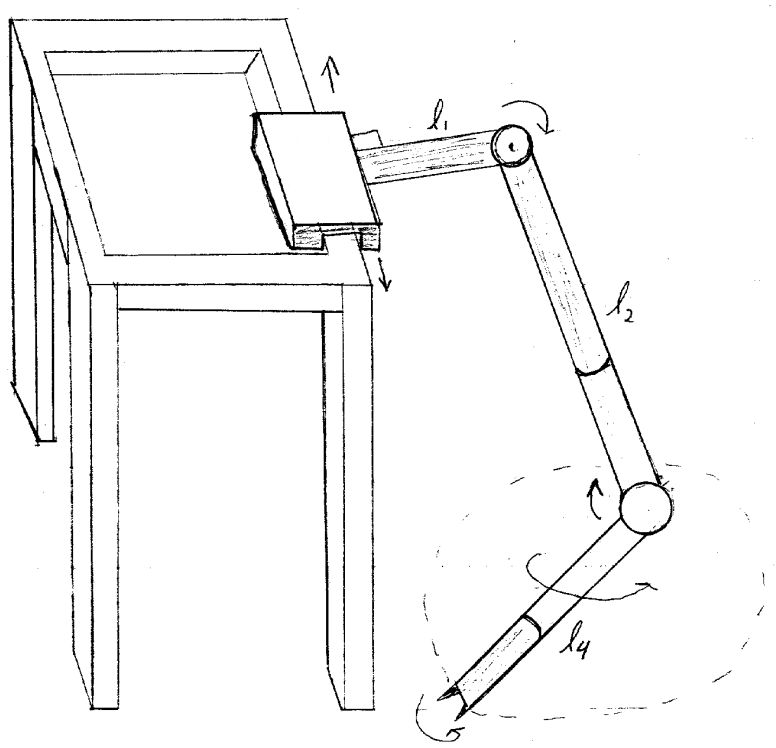
$$ B = \text{atan2}(0.342, \pm\sqrt{0.856 + 0.027} $$
$$ B = \{20^\circ, 160^\circ\} $$

$$ A = \text{atan2}(0.163, 0.925) = \arctan(0.176) \quad (B = 20^\circ) $$
$$ A = \arctan(0.176) + 180^\circ \quad (B = 160^\circ) $$
$$ A = \{10^\circ, 160^\circ\} $$

$$ C = \text{atan2}(0.470, 0.814) = \arctan(0.577) \quad (B = 20^\circ) $$
$$ C = \arctan(0.577) + 180^\circ \quad (B = 160^\circ) $$
$$ C = \{30^\circ, 210^\circ\} $$

Inverse Kinematics Example: "Chair Helper" 5-DOF Robot

Problem and solution by Melani Shoemaker (former EE543 student).



Here is an example of solving the inverse kinematics equations for a 5-DOF robot embedded in 6-DOF space. This is a robot designed to assist a wheelchair user. By application of link frames, derivation of Denavit Hartenberg parameters, and applying them to the link transform matrix,

$$ {}^{N-1}_{N}T = \begin{bmatrix} c\theta_N & -s\theta_N & 0 & a_{N-1} \\ s\theta_N c\alpha_{N-1} & c\theta_N c\alpha_{N-1} & -s\alpha_{N-1} & -s\alpha_{N-1}d_N \\ s\theta_N s\alpha_{N-1} & c\theta_N s\alpha_{N-1} & c\alpha_{N-1} & c\alpha_{N-1}d_N \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

we can get the transform for each link as follows:

$$ {}^{0}_{1}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{1}_{2}T = \begin{bmatrix} c_2 & -s_2 & 0 & l_1 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}T = \begin{bmatrix} c_3 & -s_3 & 0 & 0 \\ 0 & 0 & -1 & -l_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{3}_{4}T = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}T = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 1 & l_4 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Now we create the forward kinematic equations, by multiplying the link matrices together as:

$$ {}^{0}_{5}T = \quad {}^{0}_{1}T \quad {}^{1}_{2}T \quad {}^{2}_{3}T \quad {}^{3}_{4}T \quad {}^{4}_{5}T $$

It turns out that if we start multiplying from ${}^{4}_{5}T$ and work our way backwards, we will get highly useful intermediate results. i.e.

$$ {}^{0}_{5}T = \quad {}^{0}_{1}T \quad {}^{1}_{2}T \quad {}^{2}_{3}T \begin{bmatrix} c_4c_5 & -c_4s_5 & -s_4 & -s_4l_4 \\ s_5 & c_5 & 0 & 0 \\ s_4c_5 & -s_4s_5 & c_4 & c_4l_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The matrix at the right, ${}^3_5T$, will be useful later. Similarly,

$$ {}^0_5T = \quad {}^0_1T \quad {}^1_2T \begin{bmatrix} c_3c_4c_5 - s_3s_5 & -c_3c_4s_5 - s_3c_5 & -s_4c_3 & -c_3s_4l_4 \\ -s_4c_5 & s_4s_5 & -c_4 & -c_4l_4 - l_2 \\ s_3c_4c_5 + c_3s_5 & -s_3c_4s_5 + c_3c_5 & -s_4s_3 & -s_3s_4l_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

This time the matrix at the right is ${}^2_5T$, and it will also be useful. Continuing this process gives us the complete forward kinematic model:

$$ {}^0_5T = \begin{bmatrix} c_2c_3c_4c_5 - c_2s_3s_5 + s_2s_4c_5 & -c_2c_3c_4s_5 - c_2s_3c_5 - s_2s_4s_5 & -c_2s_4c_3 + s_2c_4 & -c_2c_3s_4l_4 + s_2c_4l_4 + s_2l_2 + l_1 \\ s_2c_3c_4c_5 - s_2s_3s_5 - c_2s_4c_5 & -s_2c_3c_4s_5 - s_2s_3c_5 + c_2s_4s_5 & -s_2s_4c_3 - c_2c_4 & -s_2c_3s_4l_4 - c_2c_4l_4 - c_2l_2 \\ s_3c_4c_5 + c_3s_5 & -s_3c_4s_5 + c_3c_5 & -s_4s_3 & -s_3s_4l_4 + d \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

We desire the robot to reach the configuration ${}^0_5T_D$, were

$$ {}^0_5T_D = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_x \\ r_{21} & r_{22} & r_{23} & P_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Remember, every element of ${}^0_5T_D$ is *known*. We generate equations to solve by equating like terms between ${}^0_5T_D$ and the forward kinematic model. For example, to pick a small one,

$$ r_{33} = -s_4s_3 $$

So far so good, but how do we attack this mess? Unfortunately, there is no substitute for just going to a quiet room, getting a cup of coffee, and going at it. Here is what Melani did. Let's look in particular at two equations we can write from the above:

$$ r_{13} = -c_2s_4c_3 + s_2c_4 $$

and

$$ P_x = -c_2c_3s_4l_4 + s_2c_4l_4 + s_2l_2 + l_1 $$

If we re-write the second one as

$$ P_x = (-c_2s_4c_3 + s_2c_4)l_4 + s_2l_2 + l_1 $$

we now have the right-hand-side of r_{13} embedded inside the RHS of P_x. Therefore we can substitute to get rid of the unknowns s_2, c_2, s_4, c_4.

$$ P_x = r_{13}l_4 + s_2l_2 + l_1 $$

This can be solved (because *only* s_2 is unknown) to give

$$ s_2 = \frac{P_x - r_{13}l_4 - l_1}{l_2} $$

We can use the exact same logic on the second row of the matrix to get

$$ c_2 = \frac{-P_y - r_{23}l_4}{l_2} $$

Now we have the first result:

$$ \theta_2 = \text{atan2}(P_x - r_{13}l_4 - l_1, -P_y - r_{23}l_4) $$

(note that we could take out the $\frac{1}{l_2}$ term because it is common to both arguments of atan2.)

What else can we solve? Let's use this same trick on the third row.

$$ r_{33} = -s_4s_3 $$

$$ P_z = -s_3s_4l_4 + d $$

This gives our second result:

$$ d = P_z + r_{33}l_4 $$

Now it seems like we have run out of such tricks, but this is where our intermediate results come to the rescue. We had

$$ {}^0_5T = \quad {}^0_1T \quad {}^1_2T \begin{bmatrix} c_3c_4c_5 - s_3s_5 & -c_3c_4s_5 - s_3c_5 & -s_4c_3 & -c_3s_4l_4 \\ -s_4c_5 & s_4s_5 & -c_4 & -c_4l_4 - l_2 \\ s_3c_4c_5 + c_3s_5 & -s_3c_4s_5 + c_3c_5 & -s_4s_3 & -s_3s_4l_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Since we know d and θ_2, ${}^0_1T$ and ${}^1_2T$ are now known. Thus we can write

$$ ({}^1_2T)^{-1}({}^0_1T)^{-1} \quad {}^0_5T = \begin{bmatrix} c_3c_4c_5 - s_3s_5 & -c_3c_4s_5 - s_3c_5 & -s_4c_3 & -c_3s_4l_4 \\ -s_4c_5 & s_4s_5 & -c_4 & -c_4l_4 - l_2 \\ s_3c_4c_5 + c_3s_5 & -s_3c_4s_5 + c_3c_5 & -s_4s_3 & -s_3s_4l_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Let's define:

$$\hat{T} = ({}^1_2T)^{-1}({}^0_1T)^{-1} \quad {}^0_5T_D$$

Though we can't know the values of $\hat{T}$ until d and θ_2 are solved, after that point we can use $\hat{T}$ the same way we used ${}^0_5T_D$. In other words, we can treat $\hat{T}$ as *known*.

$$\hat{T} = \quad {}^2_5T$$

Equating elements we get

$$\hat{T}_{33} = -s_4 s_3$$
$$\hat{T}_{13} = -s_4 c_3$$

If $s_4 \neq 0$, we can treat it like a constant, but since both θ_3 and θ_4 are unknown, we get two possible solutions:

$$\theta_{3,1} = \text{atan2}(\hat{T}_{33}, \hat{T}_{13})$$

and

$$\theta_{3,2} = \text{atan2}(-\hat{T}_{33}, -\hat{T}_{13}) = \theta_{3,1} + \pi$$

Now that we have two solutions for θ_3, we have ${}^2_3T_1$ and ${}^2_3T_2$. We can generate two versions of a third *known* matrix:

$$\tilde{T1} = ({}^2_3T_1)^{-1}({}^1_2T)^{-1}({}^0_1T)^{-1} \quad {}^0_5T_D$$

$$\tilde{T2} = ({}^2_3T_2)^{-1}({}^1_2T)^{-1}({}^0_1T)^{-1} \quad {}^0_5T_D$$

These can be equated to our first intermediate result from the forward kinematics to easily get θ_4 and θ_5:

$$\tilde{T}_{13} = -s_4, \tilde{T}_{33} = c_4$$

$$\tilde{T}_{21} = -s_5, \tilde{T}_{22} = c_5$$

but remembering that there are two versions of $\tilde{T}$

$$\theta_{4,1} = \text{atan2}(-\tilde{T1}_{13}, \tilde{T1}_{33})$$

$$\theta_{4,2} = \text{atan2}(-\tilde{T2}_{13}, \tilde{T2}_{33})$$
$$\theta_{5,1} = \text{atan2}(-\tilde{T1}_{21}, \tilde{T1}_{22})$$
$$\theta_{5,2} = \text{atan2}(-\tilde{T2}_{21}, \tilde{T2}_{22})$$

Recap

To summarize, the result of our analysis is a general solution to the inverse kinematics problem *for this robot* that will work for any reachable end effector configuration, ${}^0_5T_D$. Suppose we were now to implement this as a piece of computer code, what would we have to do? Here is a rough outline:

1. Read in the desired end-effector configuration

$${}^0_5T_D = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_x \\ r_{21} & r_{22} & r_{23} & P_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. Check that this matrix represents a valid reachable pose for this robot (see below).
3. $$\theta_2 = \text{atan2}(P_x - r_{13}l_4 - l_1, -P_y - r_{23}l_4)$$
4. $$d = P_z + r_{33}l_4$$
5. Using the values of θ_2 and d just computed, compute

$$\hat{T} = ({}^1_2T)^{-1}({}^0_1T)^{-1} \quad {}^0_5T_D$$

6. Verify assumption that $\theta_4 \neq 0$ or π. To do this, we could check that $|\hat{T}_{33}|$ **and** $|\hat{T}_{13}|$ are greater than zero, or we could check $|\hat{T}_{23}| < 1$. If these tests fail, we can only solve for $\theta_3 + \theta_5$.
7. Compute

$$\theta_{3,1} = \text{atan2}(\hat{T}_{33}, \hat{T}_{13})$$
$$\theta_{3,2} = \theta_{3,1} + 3.1415926$$

8. Using the values of θ_2, θ_3, and d just computed, compute

$$\tilde{T1} = ({}^2_3T_1)^{-1}({}^1_2T)^{-1}({}^0_1T)^{-1} \quad {}^0_5T_D$$

$$\tilde{T2} = ({}^2_3T_2)^{-1}({}^1_2T)^{-1}({}^0_1T)^{-1} \quad {}^0_5T_D$$

9. Finally we can compute the joint angles:

$$\theta_{4,1} = \text{atan2}(-\tilde{T1}_{13}, \tilde{T1}_{33})$$

$$\theta_{4,2} = \text{atan2}(-\tilde{T2}_{13}, \tilde{T2}_{33})$$

$$\theta_{5,1} = \text{atan2}(-\tilde{T1}_{21}, \tilde{T1}_{22})$$

$$\theta_{5,2} = \text{atan2}(-\tilde{T2}_{21}, \tilde{T2}_{22})$$

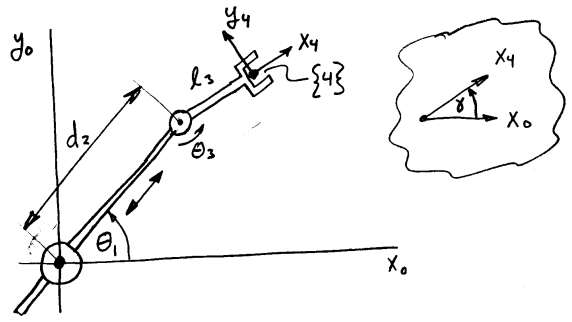
Reachability

We have skipped one last detail which makes the problem a bit more tricky. The solution above will only work if ${}^0_5T_D$ represents a valid reachable configuration for our manipulator. Since we have only 5 degrees of freedom, not all orientations are possible at a given position. It is not trivial to generate the ${}^0_5T_D$ matrix but we will leave that topic to another time.

4.5 Geometric Solution

4.5.1 Planar Examples

Example 4.9

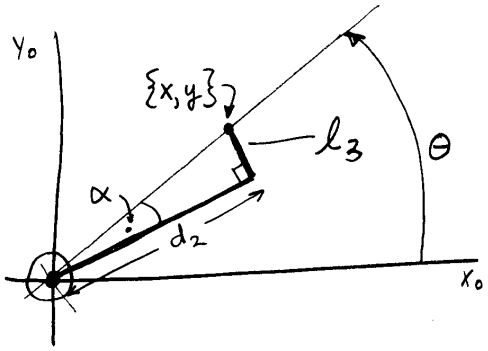


First, assume that joint 3 is locked. For $\theta_3 = 90°$, find θ_1, and d_2, given the end effector coordinates x, y:

$$^0P_4 = \begin{bmatrix} x \\ y \end{bmatrix}$$

How many solutions are there and how do you find them?

Solution:



by Pythagorean theorem and a few basic triangle facts:,

$$d_2^2 + l_3^2 = x^2 + y^2 \qquad \theta = \text{atan2}(y, x) \qquad \theta_1 = \theta - \alpha$$

$$d_2 = \pm\sqrt{x^2 + y^2 - l_3^2}$$

$$\alpha = \text{atan2}(l_3, d_2)$$

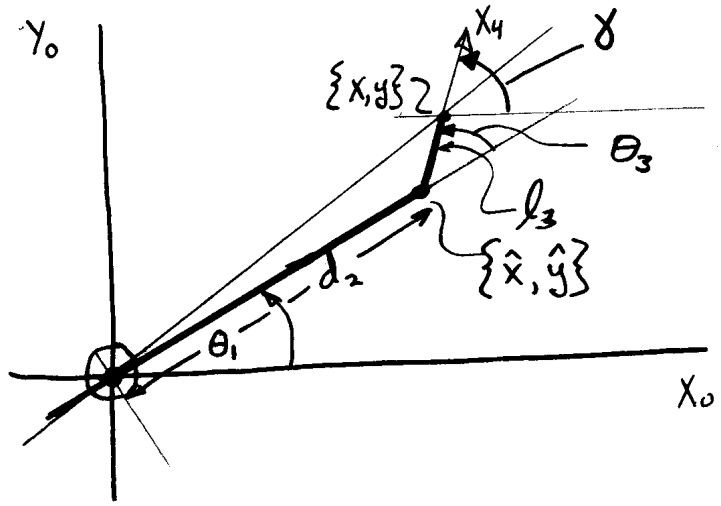
$$\theta_1 = \text{atan2}(y, x) - \text{atan2}(l_3, d_2)$$

There are two solutions, obtained by taking either positive or negative root of d_2. Choice of θ_1 is automatic.

Example 4.10

Now assume that θ_3 is variable. Given x, y and ${}^0_4R$, or equivalently, the end effector angle, γ, find θ_1, d_2, θ_3. How many solutions are there and how do you find them?

Solution:



$$ {}^0_4R = \begin{bmatrix} c\gamma & -s\gamma & 0 \\ s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ \gamma = \text{atan2}(r_{21}, r_{11}) $$

$$ \gamma = \text{direction of } x_4 = \theta_1 + \theta_3 $$

Now let's solve for the position of the elbow, labeled $\hat{x}, \hat{y}$.

$$ \hat{x} = x + l_3 \cos(\pi + \gamma) $$

$$ \hat{y} = y + l_3 \sin(\pi + \gamma) $$

$$ d_2 = \pm\sqrt{\hat{x}^2 + \hat{y}^2} $$

$$ \theta_1 = \text{atan2}(\hat{y}, \hat{x}) $$

or

$$ \theta_1 = \text{atan2}(\hat{y}, \hat{x}) + \pi $$

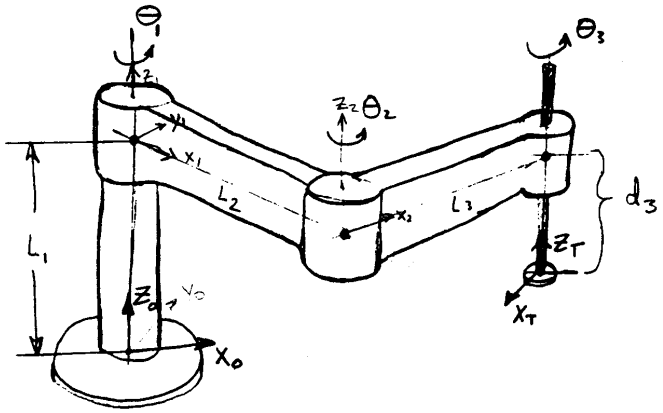
$$ \theta_3 = \gamma - \theta_1 $$

There are two solutions, obtained by taking either positive or negative root of d_2. Add π to θ_1 if $d_2 < 0$.

4.5.2 Spatial Example

Example 4.11

SCARA robot arm



This drawing illustrates a type of 4-DOF arm architecture called the SCARA arm. All three rotary axes are parallel and vertical. The manipulator subspace for this robot is

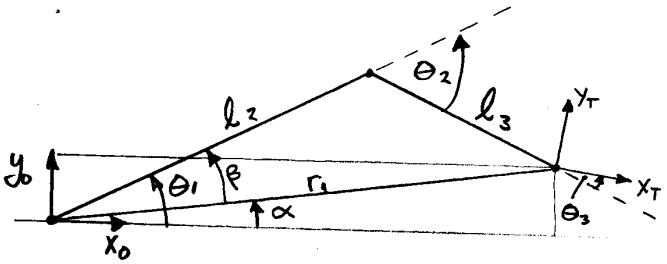
$$ {}^{O}_{T}T_d = \begin{bmatrix} c\phi & -s\phi & 0 & x_d \\ s\phi & c\phi & 0 & y_d \\ 0 & 0 & 1 & z_d \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where ϕ is the rotation of the tool frame in the X_0, Y_0 plane. Find the inverse kinematic solutions for this manipulator, i.e. find all solutions for θ_{1-3} and d_3 of

$$ {}^{O}_{T}T\left(\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ d_3 \end{bmatrix}\right) = {}^{0}_{T}T_d $$

Since forward kinematic equations are not given here, the geometric method is preferred. Make sure you find *all* solutions, and no spurious solutions.

Solution:



The approach is to solve for θ_1 and θ_3 based on the end point P_0 and then get d_3 and θ_3.

Example 4.11 cont.

$$r_1 = +\sqrt{x^2 + y^2}$$

Using law of cosines,

$$r_1^2 = l_2^2 + l_3^2 + 2l_2 l_3 \cos(\theta_2)$$

$$\cos(\theta_2) = \frac{r_1^2 - l_2^2 - l_3^2}{2l_2 l_3} = r_3$$

$$\sin(\theta_2) = \pm\sqrt{1 - r_3^2}$$

$$\theta_2 = \text{atan2}(\pm\sqrt{1 - r_3^2}, r_3)$$

$$\alpha = \text{atan2}(y_0, x_0)$$

$$l_3^2 = l_2^2 + r_1^2 - 2l_2 r_1 \cos\beta$$

$$\cos\beta = \frac{l_2^2 + x_0^2 + y_0^2 - l_3^2}{2l_2 r_1} = r_2$$

$$\beta = \text{atan2}(\pm\sqrt{1 - r_2^2}, r_2)$$

$$\theta_1 = \alpha + \beta = \text{atan2}(\pm\sqrt{1 - r_2^2}, r_2) + \text{atan2}(y_0, x_0)$$

$$\theta_1 = \text{atan2}(y_0, x_0) + \begin{cases} \text{atan2}(+\sqrt{1 - r_2^2}, r_2)\text{(elbow up config)} \\ \text{atan2}(-\sqrt{1 - r_2^2}, r_2)\text{(elbow down config)} \end{cases}$$

$$\theta_2 = \begin{cases} \text{atan2}(+\sqrt{1 - r_3^2}, r_3)\text{(elbow up config)} \\ \text{atan2}(-\sqrt{1 - r_3^2}, r_3)\text{(elbow down config)} \end{cases}$$

By inspection:

$$d_3 = L1 - z_d$$

$$\phi = \text{atan2}(t_{d21}, t_{d11})$$

where t_{dij} are the elements of ${}^0_T T_d$. And

$$\theta_3 = \phi - \theta_1 - \theta_2$$

4.6 Summary of Notation

Chapter 5

Differential Kinematics - The Jacobian Matrix

5.1 Problem Statement and Learning Objectives

Problem Statement This chapter considers the problems of mapping velocities, incremental motions, and forces and torques, between the end effector and the joint space.

Learning Objectives Upon completing this Chapter, the reader should be able to

- Be able to compute velocity with different frames of computation and representation and understand the difference between these two frames.
- Be able to define the linear and angular velocity of a rigid body.
- Be able to compute the velocity and angular velocity of a link, based on the DH parameters and the velocities of the previous link.
- Be able to propagate the velocity calculation down the serial link chain to compute linear and angular velocity of an end effector.
- Be able to extract a Jacobian Matrix from the velocity propagation calculation.
- Be able to state the relationship between joint velocities and end effector velocities using the Jacobian Matrix.
- Be able to state the relationship between joint torques and end effector forces using the Jacobian Matrix.

5.2 Velocity

5.2.1 Velocity and Acceleration of a Particle

Computation Frame vs. Representation Frame.

Velocity is based on a differential measurement or calculation of position, $\Delta X = X(t + \Delta t) - X(t)$. In order for this subtraction to be valid, both $X(t+\Delta t)$ and $X(t)$ must be represented in the same frame. If this frame is moving, we might get a very different result from use of a fixed frame. This is not a bad thing however. Depending on our needs, either a moving or fixed frame might be preferable. Another part of the velocity computation is dividing by Δt, but Δt is a scalar, and thus is the same in every frame.

The second key frame is the frame in which we represent the velocity after it is computed. Once we have $\frac{\Delta X}{\Delta T}$, we have a free vector, and this may be transformed (by rotation) into any frame we desire. This second frame is called the representation frame.

Velocity of a Vector If ${}^{A}Q$ is a point represented in frame A, then

$$ {}^{A}V_Q = lim_{\Delta t \to 0} \left(\frac{{}^{A}Q(t + \Delta t) - {}^{A}Q(t)}{\Delta t} \right) $$

We have just *computed* the velocity in frame A and it is also *represented* in frame A.

Later we may wish to represent the velocity in another frame e.g.:

$$ {}^{B}({}^{A}V_Q) = {}^{B}_{A}R\,{}^{A}V_Q $$

If we use the notation:

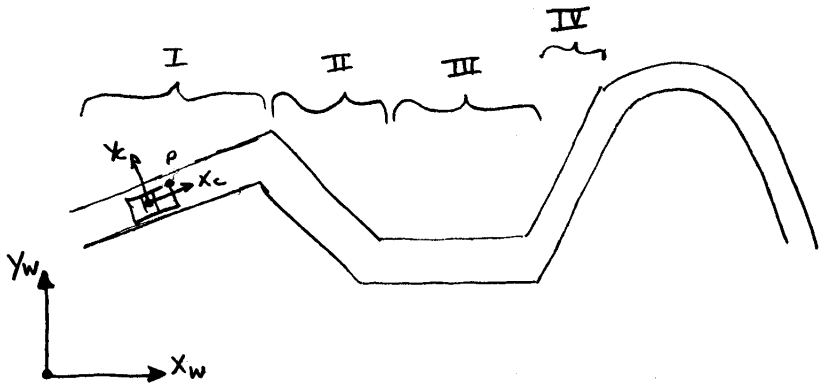
$$^A(^B V_P)$$

We mean
A is the *representation* frame.
B is the *computation* frame.
P is the point whos velocity we are talking about.

If a velocity is computed and represented in the same frame, we use just one superscript:

$$^A(^A V_P) \rightarrow ^A V_P$$

Example 5.1
To illustrate the difference between *computation* and *representation* of velocity vectors, consider the following example. A car is driving along a road. Its speedometer reads a constant 55 mph. The road is divided into four regions according to the map below.



C is a frame fixed to the car. W is a frame fixed to the earth. P is a point on the car.
What are some different velocity vectors for each of the regions I ... IV?

$$^C(^C V_P) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

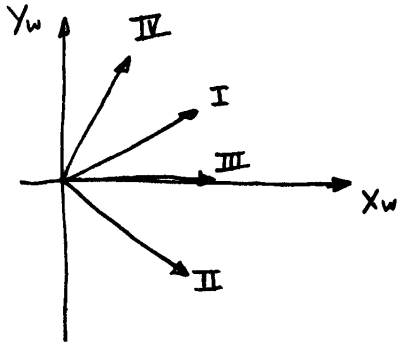
In this example, the velocity of point, P is computed in the Car frame and represented in the car frame. However since P is a fixed point on the car, it's velocity in the car frame is zero. In fact this is true for all regions of the map. Also, note that

$$^W(^C V_P) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Why? Because if we rotate this velocity, it's still zero.

$$^C(^W V_P) = \begin{bmatrix} 55mph \\ 0 \\ 0 \end{bmatrix}$$

In this example, the velocity of P is computed in the world frame. This must have a magnitude of 55mph and it must point in the positive X_C direction because we assume the car is going in forward gear! Since the velocity is represented in the car frame, it must be constant and independent of the region.
Supposed we are asked to diagram $^W(^W V_P)$ on the coordinate system $\{X_C, Y_C\}$: What would that look like?



We know $|^W(^W V_P)| = 55mph$ since the speed is a constant 55. Only the direction in frame W changes in each region.

5.3 The Jacobian Matrix

In the forward kinematics model, we saw that it was possible to relate joint angles, θ, to the configuration of the robot end effector, ${}^0_6T$. In this chapter we will work on the relationship between the joint rates, $\dot{\theta}$, and the velocity of the end effector, $\dot{x}$ with a matrix as follows:

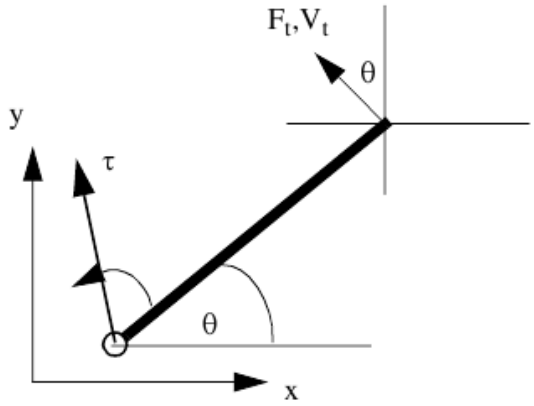
$$\dot{x} = J(\theta)\dot{\theta}$$

Here the velocity $\dot{x}$ describes both linear and rotational components. An expanded version of the previous equation is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \\ \quad J(\theta) \quad \\ \\ \end{bmatrix} \begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \vdots \\ \dot{\theta_N} \end{bmatrix}$$

ω_i are the components of angular velocity, and $J(\theta)$ is a matrix of size 6xN where N is the number of joints in the robot. We will derive $J(\theta)$ by calculating $\dot{x}$ as a function of $\dot{\theta}$ and factoring out $J(\theta)$. But first, as a simple illustration let us consider a simple planar example:

Example 5.2
Compute the velocity of the end effector of this basic planar robot as a function of its joint velocity $\dot{\theta}$.



Looking at the figure, we can resolve the velocity of the tip into x and y components as follows:

$$v_x = -r\dot{\theta}\sin(\theta) \qquad v_y = r\dot{\theta}\cos(\theta)$$

This can be expressed as a trivial matrix equation as follows:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} -r\sin(\theta) \\ r\cos(\theta) \end{bmatrix} \dot{\theta}$$

or

$$J(\theta) = \begin{bmatrix} -r\sin(\theta) \\ r\cos(\theta) \end{bmatrix}$$

Alternatively, we can look at the tip force, and the torque around the joint:

$$\tau = -rF_x\sin(\theta) + rF_y\cos(\theta)$$

This again gives a trivial matrix equation:

$$\tau = \begin{bmatrix} -r\sin(\theta) & r\cos(\theta) \end{bmatrix} \begin{bmatrix} F_x \\ F_y \end{bmatrix}$$

Note that the first term happens to be the transpose of the matrix $J(\theta)$ computed above. As we will show below, this is *not* a coincidence, so we can write

$$\tau = J^T(\theta)F$$

This chapter will cover the Jacobian Matrix and how to compute it in detail, but first we want to give the bird's eye view of this powerful matrix. This section aims to show what kinds of problems we can solve once we solve the Jacobian Matrix.

Spaces Consider three spaces relevent to robot arm motion (Figure 5.1).

- Joint Space
  A point in this space is located by the values of the N joint variables of the arm.
- Configuration Space
  A point in this space is located by the X, Y, Z position of the end effector plus three rotation variables which describe its orientation such as roll, pitch, yaw angles.
- Cartesian Task Space
  A point in this space is determined by X, Y, Z. The configuration of a rigid object in Cartesian task space is defined by its 4×4 homogeneous transform.

In Chapters 2 and 3, we've seen the static mappings between Joint Space and Cartesian Task Space, the Forward Kinematic Mapping (Fkin) and the Inverse Kinematic Mapping (Kin^{-1}) (Figure 5.2).

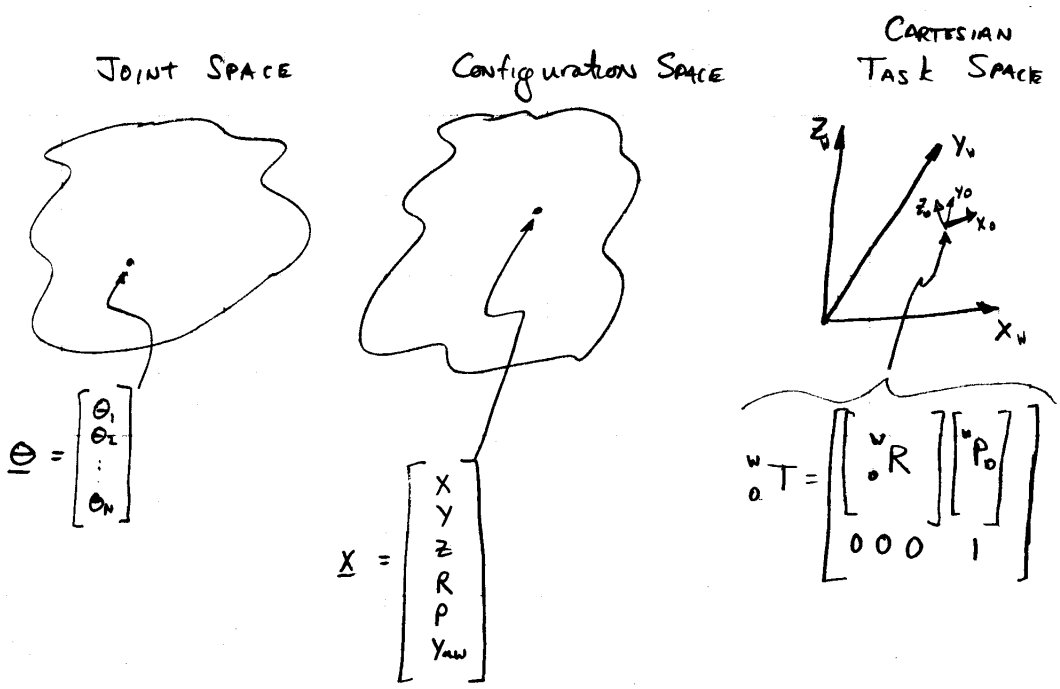


Figure 5.1: Three spaces relevent to robot arm motion: Task space, Joint Space, Configuration Space.

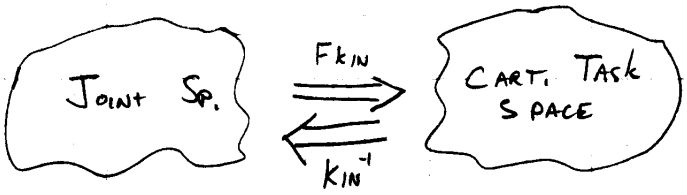


Figure 5.2: Static mappings between Joint space and Cartesian Task Space.

Now we introduce *incremental* mappings between small changes in joint space and small changes in task space. i.e.

$$\Delta X = J(\theta)\Delta\theta$$

$$\dot{x} = J(\theta)\dot{\theta}$$

and

$$\dot{\theta} = J^{-1}(\theta)\dot{x}$$

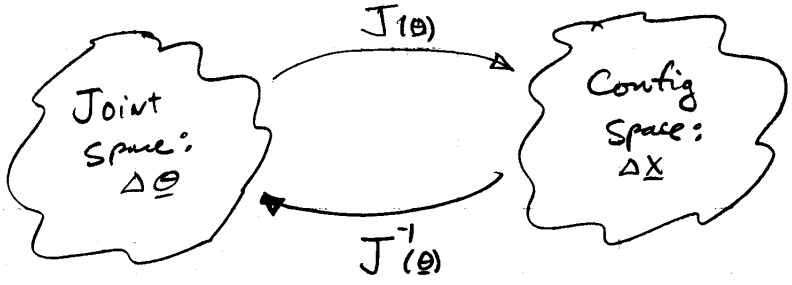


Figure 5.3: Incremental Mapping between spaces

We can apply these mappings to answer several interesting questions:

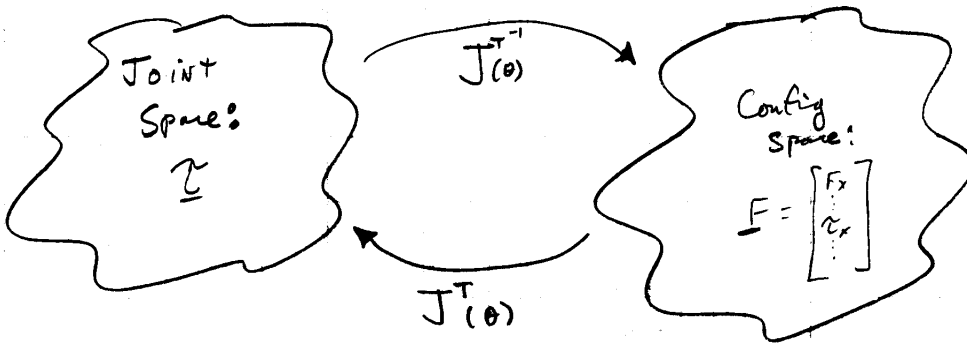


Figure 5.4: Mappings between joint torques and end effector forces.

Velocity Mapping **Q:** What velocity and angular velocity will be generated at the tip of the robot if all joints are driven at 0.1 rad/sec?
A:

$$\dot{x} = J(\theta) \begin{bmatrix} 0.1 \\ 0.1 \\ \cdots \\ 0.1 \end{bmatrix}$$

note that this solution is a function of θ.
Q: How should we drive the joints so that the end effector tracks a target moving with velocity

$$\dot{x}_t = \begin{bmatrix} 1 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

A:

$$\dot{\theta} = J^{-1}(\theta) \begin{bmatrix} 1 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

notes: Does $J(\theta)^{-1}$ exist? What about 'conditioning' of $J(\theta)$?

Force Mapping We also can consider mapping Forces and Torques between the same spaces:

$$\tau = J(\theta)^T F$$

$$F = J(\theta)^{-T} \tau$$

Notes: 1) $J(\theta)^{-T}$ indicates the inverse of $J(\theta)^T$. 2) Mappings are a function of θ. 3) Does $J(\theta)^{-T}$ exist?

Virtual Work It's not obvious that the Jacobian matrix should relate *both* incremental motions and also forces and torques between the joint space and configuration space. The concept of virtual work can be used to derive the force and torque mapping role of the Jacobian Matrix from its incremental motion mapping role. We consider the mechanism to be purely kinematic. This means that it has no ability to dissipate or store energy:

- no friction (energy loss)
- no inertia (kinetic energy)
- no mass (potential energy due to gravity)
- no compliance (potential energy due to elastic energy of deformation)

These assumptions might seem quite unrealistic. In fact many motions involve low enough amounts of energy that the assumptions are quite useful.

Let's review some definitions:

Power $\frac{d}{dt}E$ (where E is energy)

Power $F.V = F^T V$

Power $\tau.\omega = \tau^T \omega$

Work $\Delta E = P \Delta t$

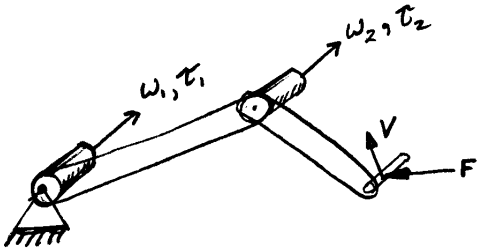


Figure 5.5: Mechanism to illustrate Virtual Work.

Work is ΔE. Because of the properties above, if we apply some work to part of the mechanism, it must come out of another part instantaneously for energy to be conserved. The locations ("ports") at which we will consider energy flow are the end effector and the joints.

Let's consider a planar 2-link mechanism with a handle applied to the end (Figure 5.5). If it helps, think of ideal generators attached to the joints which can extract energy from the joints. We crank on the handle and compare the force and velocities of the handle with the joint torques and joint angular velocities:

$$F^T V = \tau^T \omega$$

if we multiply both sides by Δt,

$$F^T \Delta X = \tau^T \Delta\Theta$$
$$\tau^T \Delta\Theta - F^T \Delta X = 0$$

We know that

$$\Delta X = J(\Theta)\Delta\Theta$$

so

$$\left(\tau^T - F^T J\right)\Delta\Theta = 0$$

for finite $\Delta\Theta$:

$$(\tau^T - F^T J) = 0$$
$$\tau^T = F^T J$$
$$\tau = J^T F$$

(because $(AB)^T = B^T A^T$ is a property of the matrix transpose).

5.3.1 Rigid Bodies

Going beyond velocity of single points requires us to consider rigid bodies. Rigid bodies can be thought of as a collection of points, all of which have a fixed location in a frame called an object frame. A rigid body has two types of velocity: linear (translational) velocity which describes its rate and direction of translation, and angular velocity which describes its rotation.

Linear Velocity Linear velocity of a rigid body is represented the same as velocity of a point, having a computation frame and a representation frame. However, nonzero angular velocity of an object will cause each point in the object to have a different linear velocity.

Angular Velocity Angular velocity can be understood through the following properties

- Angular velocity, Ω, is a vector whos direction is the axis of rotation and whos magnitude is the rate of rotation, ω.
- If a point is displaced from the axis of rotation, that point will have a linear velocity component corresponding to the rotation:
$$V = r \otimes \Omega$$
where r is a vector from a point on the axis of rotation to the point.
- The vector cross product $\otimes$ can be defined as
$$A = B \otimes C$$
$$|A| = |B||C| \sin\theta$$
$$A \perp B, A \perp C$$
where θ is the angle between the two vectors by the Right Hand Rule.

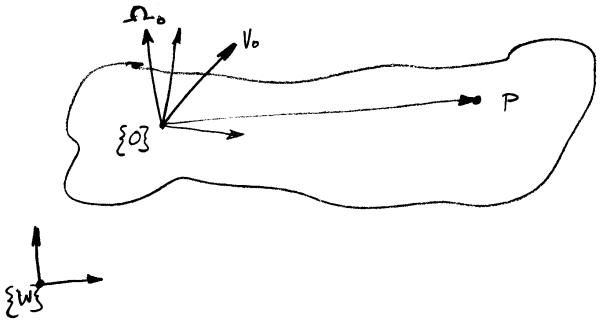
- Another way to compute $\otimes$ is using a skew symmetric matrix: If a and b are 3 dimensional vectors,

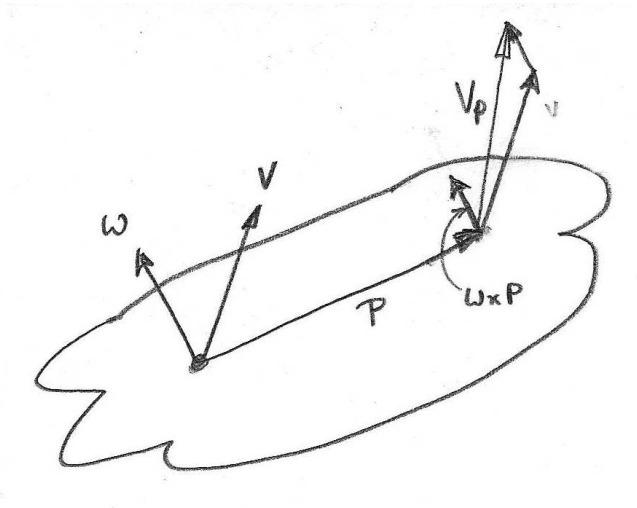
$$a \otimes b = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- The angular velocity of a rigid object is the same for all points in that object.

Angular Velocity Generates Linear In this section we consider the effects of velocity on different parts of an object or robot arising from the second point above.

First, consider a rigid object with a frame, O, attached to it:





The origin of frame 0, {O}, has velocity V_O and the object has angular velocity ω_O which are computed in the world frame, W, and represented in *any* frame. What is the velocity of a point, P, on the object?

The answer has two components

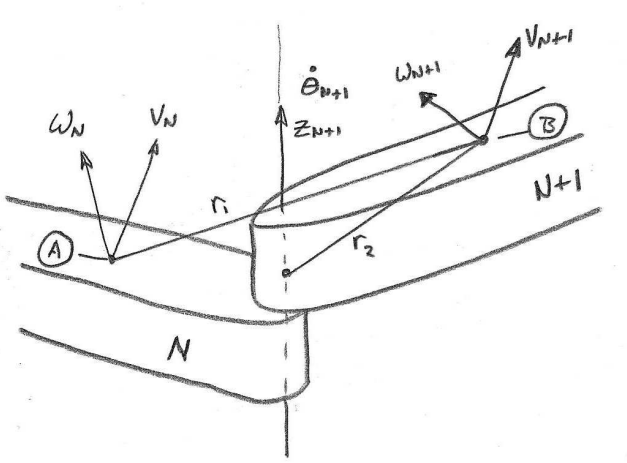
- The velocity of the object, V_O.
- Additional velocity due to the rotation of the object: $V = \omega_O \otimes P$

So its total velocity is

$$^*({}^W V_P) = V_O + \omega_O \otimes P$$

Where the * is meant to indicate that this equation is true in any frame, so long as all terms are represented in that same frame.

Velocity Propagation Now we investigate robot links arranged in a serial chain. Remember that robot links are rigid bodies, and also that adjacent robot links are rigidly linked to each other *except* for motion about/along the Z_N axis. For now, let's consider only rotary joints.



This graphic shows two adjacent robot links. An arbitrary point is selected on each link at which we define the linear and angular velocity for that link. r_1 is a vector connecting the two points. r_2 is the unique vector connecting the point on link N to the Z_N axis. Assume that all velocities are computed in the world frame, W, and represented anywhere.

Then, assume they are a single rigid object (i.e. $\dot{theta}_{N=1} = 0$). In that case, using the result above, we have

$$\omega_{N+1} = \omega_N \qquad V_{N+1} = V_N + \omega_N \times r_1$$

Now, assume that the joint velocity, $\dot{\theta}_{N+1} \neq 0$. This adds a new component to both equations:

$$\omega_{N+1} = \omega_N + Z_{N+1}\dot{\theta}_{N+1} \qquad V_{N+1} = V_N + \omega_N \times r_1 + \dot{\theta}_{N+1} Z_{N+1} \times r_2$$

Finally, we can apply this to robot links and their DH parameters as follows:

1. Let point (A) be the origin of frame N, (B) be the origin of frame $N+1$
2. In this case, $r_1 = [T_{14}, T_{24}, T_{34}]^T$
3. Because (B) is the origin of frame $N+1$, $r_2 = 0$.
4. From our DH analysis, we know ${}^{N}_{N+1}R, {}^{N}_{N+1}T$.

Then we get

$${}^{N+1}\omega_{N+1} = \quad {}^{N+1}_{N}R \quad {}^{N}\omega_N + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{N+1} \end{bmatrix}$$

and

$${}^{N+1}V_{N+1} = \quad {}^{N+1}_{N}R \left[{}^{N}V_N + {}^{N}\omega_N \otimes \quad {}^{N}P_{N,N+1} \right]$$

OK Professor, you said that all the velocities were known! What have we accomplished? Well, in reality we only know one pair:

$${}^{0}\omega_0 = 0$$

and

$${}^{0}V_0 = 0$$

because the base of the robot is (hopefully) bolted down. However, we also know the joint velocities, $\dot{\theta}_N$ (from sensors for example), and the rotation matrices ${}^{N+1}_{N}R$ and the link position offsets, ${}^{N}P_{N,N+1}$, from the forward kinematics. Therefore, we can use the equations above to compute each velocity one at a time.

Exercise: Derive similar equations for prismatic joints `Solution:`
If we identify that joint $N+1$ is prismatic, then we replace the above equations with the following. A prismatic joint does not add *any* rotation, so

$${}^{N+1}\omega_{N+1} = \quad {}^{N+1}_{N}R \quad {}^{N}\omega_N$$

but it does add a component of linear velocity in the Z_{N+1} direction:

$${}^{N+1}V_{N+1} = \quad {}^{N+1}_{N}R \left[{}^{N}V_N + {}^{N}\omega_N \otimes {}^{N}P_{N,N+1} \right] + \begin{bmatrix} 0 \\ 0 \\ \dot{d}_{N+1} \end{bmatrix}$$

Once the linear and angular velocities of the last link (end effector) of the manipulator are computed by this method of "velocity propagation," it is a quick step to get the Jacobian matrix as illustrated by the following example:

Example 5.3
Compute the Jacobian Matrix by the Velocity Propogation method. The manipulator is described by the following Denavit Hartenberg parameters:

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	$\pi/2$	L_1	0	θ_2
3	0	L_2	0	θ_3
4	0	L_3	0	0

Solution: First we need to generate the link transforms from the DH parameters, ${}_i^{i-1}T$ (hopefully we have them around from our forward kinematics step).

$$ {}_1^0T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}_2^1T = \begin{bmatrix} c_2 & -s_2 & 0 & L_1 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}_3^2T = \begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}_4^3T = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Now we propagate the angular velocities first (since they are needed for the linear step):

$$ {}^1\omega_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} $$

$$ {}^2\omega_2 = {}_1^2R\ {}^1\omega_1 + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} $$

$$ = \begin{bmatrix} c_2 & 0 & s_2 \\ -s_2 & 0 & c_2 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} $$

$$ {}^2\omega_2 = \begin{bmatrix} s_2\dot{\theta}_1 \\ c_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} $$

$$ {}^3\omega_3 = {}_2^3R\ {}^2\omega_2 + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{bmatrix} $$

$$ = \begin{bmatrix} c_3 & s_3 & 0 \\ -s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_2\dot{\theta}_1 \\ c_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{bmatrix} $$

$$ {}^3\omega_3 = \begin{bmatrix} s_2c_3\dot{\theta}_1 + c_2s_3\dot{\theta}_1 \\ c_2c_3\dot{\theta}_1 - s_2s_3\dot{\theta}_2 \\ \dot{\theta}_2 + \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} s_{23}\dot{\theta}_1 \\ c_{23}\dot{\theta}_1 \\ \dot{\theta}_2 + \dot{\theta}_3 \end{bmatrix} $$

$$ {}^4\omega_4 = {}_3^4R\ {}^3\omega_3 + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = {}^3\omega_3 $$

Now we propagate linear velocity using the previously calculated ${}_i^{i+1}T$ and ${}^i\omega_i$:

$$ {}^0V_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} $$

$$ {}^1V_1 = {}_0^1R({}^0V_0 + {}^0\omega_0 \times {}^0P_1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} $$

$$ {}^2V_2 = {}_1^2R({}^1V_1 + {}^1\omega_1 \times {}^1P_2) = $$

$$ \begin{bmatrix} c_2 & 0 & s_2 \\ -s_2 & 0 & c_2 \\ 0 & -1 & 0 \end{bmatrix} \left[\begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \times \begin{bmatrix} L_1 \\ 0 \\ 0 \end{bmatrix} \right] $$

Example 5.3 cont.

Here we can take advantage of the rule we identified in Section 2.11.1 in which the cross product can be represented as a skew symmetric matrix for faster computation. In each stage of the linear velocity propagation, we will repeat the computation

$$a \times b$$

where $b = [L \quad 0 \quad 0]^T$. For this case,

$$a \times b = \begin{bmatrix} 0 \\ a_3 L \\ -a_2 L \end{bmatrix}$$

Applying this we get

$${}^2V_2 = \begin{bmatrix} c_2 & 0 & s_2 \\ -s_2 & 0 & c_2 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ L_1\dot{\theta_1} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -L_1\dot{\theta_1} \end{bmatrix}$$

$${}^3V_3 = \begin{bmatrix} c_3 & s_3 & 0 \\ -s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{bmatrix} 0 \\ 0 \\ -\dot{\theta_1}L_1 \end{bmatrix} + \begin{bmatrix} s_2\dot{\theta_1} \\ c_2\dot{\theta_1} \\ \dot{\theta_2} \end{bmatrix} \times \begin{bmatrix} L_2 \\ 0 \\ 0 \end{bmatrix} \right]$$

$$= {}^3_2R \begin{bmatrix} 0 \\ \dot{\theta_2}L_2 \\ -(c_2L_2 - L_1)\dot{\theta_1} \end{bmatrix}$$

$${}^3V_3 = \begin{bmatrix} s_3L_2\dot{\theta_2} \\ c_3L_2\theta_2 \\ -(c_2L_2 - L_1)\dot{\theta_1} \end{bmatrix}$$

$${}^4V_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[{}^3V_3 + \begin{bmatrix} s_{23}\dot{\theta_1} \\ c_{23}\dot{\theta_1} \\ \dot{\theta_2} + \dot{\theta_3} \end{bmatrix} \times \begin{bmatrix} L_3 \\ 0 \\ 0 \end{bmatrix} \right]$$

$${}^4V_4 = \begin{bmatrix} s_3L_2\dot{\theta_2} \\ (c_3L_2 + L_3)\dot{\theta_2} + L_3\dot{\theta_3} \\ -\dot{\theta_1}(L_1 + c_2L_2 + c_{23}L_3) \end{bmatrix}$$

5.3.2 Jacobian Matrix from velocity expressions.

Now that we have calculated angular and linear velocity of the end effector, we can create the Jacobian matrix by factoring out all of the $\dot{\theta}$ terms:

$$\begin{bmatrix} {}^NV_N \\ \\ {}^N\omega_N \end{bmatrix}_{6\times 1} = \begin{bmatrix} \quad J \quad \end{bmatrix}_{6\times N} \begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \cdots \\ \dot{\theta_N} \end{bmatrix}_{N\times 1}$$

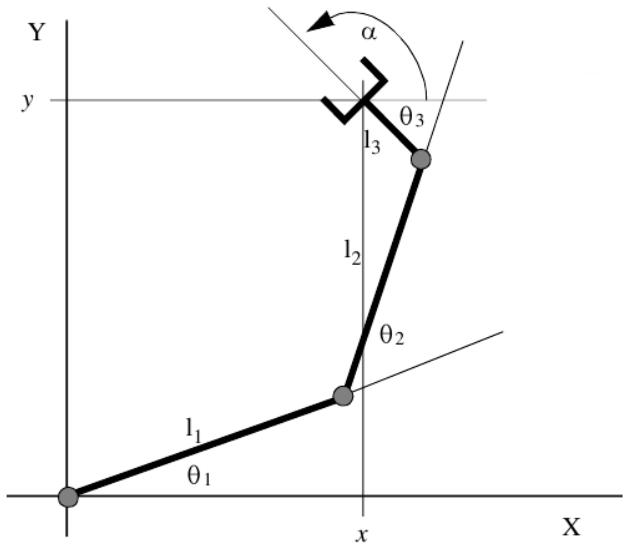


Figure 5.6: 3 Link planar manipulator.

Example 5.3 cont.

$$\begin{bmatrix} {}^4V_4 \\ {}^4\omega_4 \end{bmatrix}_{6\times1} = \begin{bmatrix} \quad J \quad \end{bmatrix}_{6\times4} \begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \cdots \\ \dot{\theta_4} = 0 \end{bmatrix}_{4\times1}$$

$${}^4J = \begin{bmatrix} 0 & s_3L_2 & 0 \\ 0 & c_3L_2 + L_3 & L3 \\ -(L_1 + C_2L_2 + C_{23}L3) & 0 & 0 \\ s_{23} & 0 & 0 \\ c_{23} & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

5.4 Jacobian Matrix by Differentiation

In addition to the method of velocity propagation, the Jacobian matrix can also be obtained by differentiation of the forward kinematic equations. Considering the 3-link planar manipulator of Figure 5.6, define the configuration vector, $x = [x, y, \alpha]^T$. The forward kinematic equations of this arm are:

$$x = l_1c_1 + l_2c_{12} + l_3c_{123} \qquad y = l_1s_1 + l_2s_{12} + l_3s_{123} \qquad \alpha = \theta_1 + \theta_2 + \theta_3$$

Differentiating the first expression gives:

$$\dot{x} = -l_1s_1\dot{\theta}_1 - l_2s_{12}(\dot{\theta}_1 + \dot{\theta}_2) - l_3s_{123}(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3)$$

or

$$\dot{x} = -(l_1s_1 + l_2s_{12} + l_3s_{123})\dot{\theta}_1 - (l_2s_{12} + l_3s_{123})\dot{\theta}_2 - (l_3s_{123})\dot{\theta}_3$$

Similarly,

$$\dot{y} = (l_1c_1 + l_2c_{12} + l_3c_{123})\dot{\theta}_1 - (l_2c_{12} + l_3c_{123})\dot{\theta}_2 - (l_3c_{123})\dot{\theta}_3$$

and

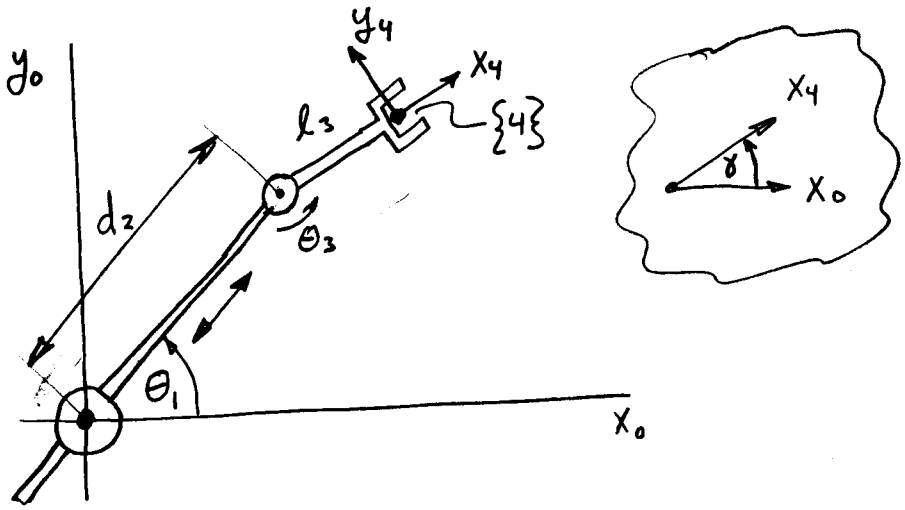
$$\dot{\alpha} = \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3$$

Thus,

$$ {}^0J\theta = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} - l_3 s_{123} & -l_2 s_{12} - l_3 s_{123} & -l_3 s_{123} \\ l_1 c_1 + l_2 c_{12} + l_3 c_{123} & l_2 c_{12} + l_3 c_{123} & l_3 c_{123} \\ 1 & 1 & 1 \end{bmatrix} $$

Remember that we started with the expression for the end effector configuration in frame 0 so the resulting Jacobian is still in frame 0. We have indicated this with a leading superscript above. In contrast, the Jacobian computed by velocity propagation leaves us with a Jacobian in frame N (we omitted the superscript in the examples above). Both results are valid. In general the Jacobian can be transformed into any desired frame as described in Section 5.6.

Example 5.4



Compute the Jacobian matrix of the arm above where all three joints $\theta_1, \dot{d}_2, \theta_3$ are variables. Consider $\dot{x}, \dot{y}$, and the rate of change of orientation of x_4, $\dot{\gamma}$. The Jacobian matrix should have 3 rows.
Solution:
The forward kinematic equations are:

$$ x = d_2 \cos\theta_1 + l_3 \cos(\theta_1 + \theta_3) $$
$$ y = d_2 \sin\theta_1 + l_3 \sin(\theta_1 + \theta_3) $$
$$ \gamma = \theta_1 + \theta_3 $$

Differentiating:

$$ \dot{x} = -d_2 \sin\theta_1 \dot{\theta}_1 + \dot{d}_2 \cos\theta_1 - l_3 \sin(\theta_1 + \theta_3)(\dot{\theta}_1 + \dot{\theta}_3) $$
$$ \dot{y} = d_2 \cos\theta_1 \dot{\theta}_1 + \dot{d}_2 \sin\theta_1 + l_3 \cos(\theta_1 + \theta_3)(\dot{\theta}_1 + \dot{\theta}_3) $$
$$ \dot{\gamma} = \dot{\theta}_1 + \dot{\theta}_2 $$

Factoring:

$$ J = \begin{bmatrix} -d_2 s_1 - l_3 s_{13} & c_1 & -l_3 s_{13} \\ d_2 c_1 + l_3 c_{13} & s_1 & l_3 c_{13} \\ 1 & 0 & 1 \end{bmatrix} $$

5.5 Inverting the Jacobian Matrix

We saw earlier that we will make frequent use of the Jacobian matrix for relating forces and velocities between the joints and the end effector spaces using the equations

$$\dot{x} = J\dot{\theta} \qquad \tau = J^T F$$

Just as with static kinematics of Chapter 2, these equations are often more useful if inverted. Thus we frequently need a form such as

$$\dot{\theta} = J^{-1}\dot{x}$$

Computation of J^{-1} is not a very time consuming job for today's processors. As we have seen however, the elements of the Jacobian matrix are functions of the joint variables θ so if we use J^{-1} we must compute it frequently as the manipulator moves around. Since the numerical values of J are always changing, we can't make a blanket statement about whether this inverse exists. We often evaluate the ability to invert a matrix using its determinant. In fact, in many practical robotic problems, there are configurations of the joints inside or throughout the workspace where

$$\mathrm{Det}(J) = 0$$

and as a result, any computation, say inside a controller, which relies on a numerical method to find J^{-1} will have problems. The configurations of the joints of a manipulator at which $\mathrm{Det}(J) = 0$ are called *singularities*.

5.5.1 Interpretations of the Jacobian Inverse: Singularities

Here we will look at some implications of basic linear algebra for the Jacobian Inverse. We will make reference to some basic facts about matrices which are given in Appendix B.

Consider the case of a square, 6x6, Jacobian matrix. If $\mathrm{rank}(J(\theta)) < 6$, then there is no solution to

$$\dot{\theta} = J^{-1}(\theta)\dot{x}$$

(see Facts 1 and 2). Also, by Fact 8, if the rank, $r = 5$, there is one non-zero solution to

$$\dot{x} = J(\theta)\dot{\theta} = 0$$

The interpretation of this fact is that there is a direction in joint velocity space for which non-zero joint motion produces no end effector motion. We call any joint configuration, $\theta = Q$ for which

$$\mathrm{rank}(J(Q)) < 6$$

a **singular configuration**.

Furthermore, there are certain directions of end effector motion, $\dot{x}_i$, which are eigenvectors of $J(\theta)$. If J is full rank, we have

$$\dot{x}_i = J(\theta)\dot{\theta} = \lambda_i(\theta)\omega_i(\theta)$$

where λ_i are the eigenvalues of J and ω_i are the eigenvectors of J. If J is full rank, we have

$$\omega_i = J^{-1}(\theta)\dot{x}_i = \lambda_i^{-1}\dot{x}_i$$

(Fact 5). As the manipulator approaches the singular configuration, $\theta \to Q$, there is at least one eigenvalue for which $\lambda_j \to 0$. Thus

$$\omega_j \to \frac{\dot{x}_j}{0} = \infty$$

In words, as the manipulator approaches the singular configuration, motion in the particular direction $\dot{x}_j$ causes

joint velocities to approach infinity.

Example 5.5
We will look at the specifics of singularities through an example, the planar 3-link manipulator of Figure 5.6. Let's look at Jacobian in the following configuration:

$$Q = \begin{bmatrix} \pi/4 \\ 0 \\ \pi \end{bmatrix}$$

When the manipulator is posed such that $\theta = Q$, we have

$$s_{12} = s_1 \qquad s_{123} = -s_1, \qquad c_{12} = c_1, \qquad c_{123} = -c_1$$

This gives

$${}^0J(Q) = \begin{bmatrix} (-l_1 - l_2 + l_3)s_1 & (-l_2 + l_3)s_1 & l_3 s_1 \\ (l_1 + l_2 - l_3)c_1 & (l_2 - l_3)c_1 & -l_3 c_1 \\ 1 & 1 & 1 \end{bmatrix}$$

If we denote the rows of J by $\{r_1, r_2, r_3\}$, and recall that for $\theta_1 = \pi/4$, then $s_1 = c_1 = 0.707$, then we note that

$$r_1 = -r_2$$

and therefore $\det[J(Q)] = 0$ (Fact 3). This means that Q is a singular configuration. Note that on closer examination, we can see that θ_1 does not have to be $\pi/4$ for this to happen. This is because, for any θ_1, we still have

$$r1 = r2 \times -\tan(\theta_1)$$

$-\tan(\theta_1)$ is a constant which relates the two rows so that the matrix must be singular. The proper way to describe the singular configuration is thus

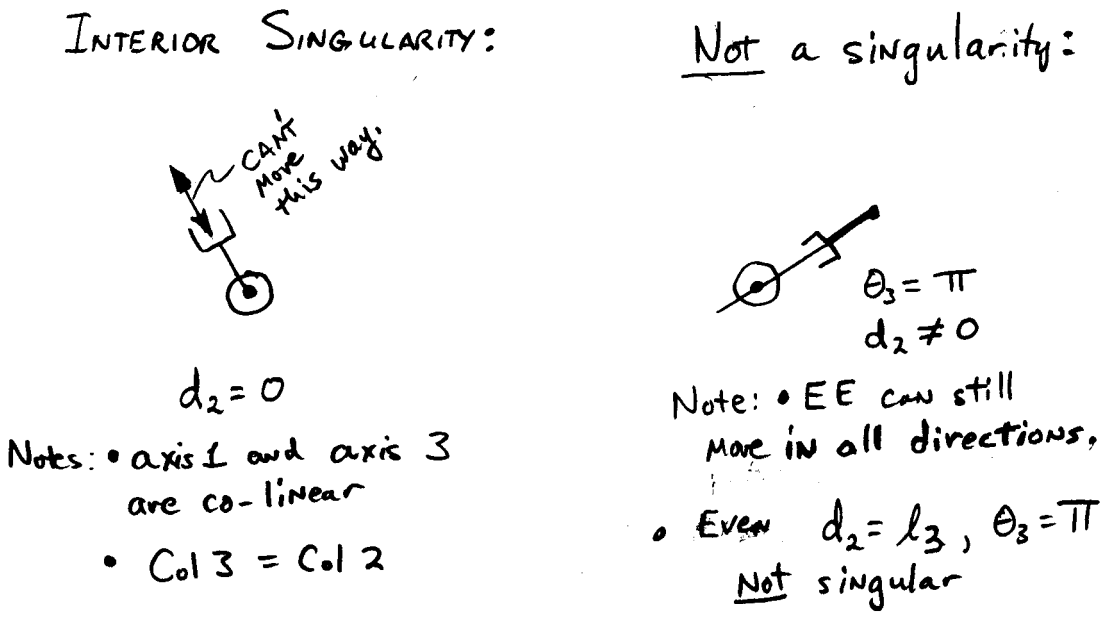
$$Q = \begin{bmatrix} \theta_1 \\ 0 \\ \pi \end{bmatrix}$$

for any θ_1.

Example 5.6
Find any singular configurations of the manipulator of Example 5.4 For each singular configuration:
1) Draw it and indicate which direction of motion cannot be achieved.
2) State whether it is an interior singularity or workspace boundary singularity.
Use analysis of the Jacobian to derive the result or confirm your graphical analysis.
Solution:
For $\theta_3 = 0$, $s_{13} = s_1$, $c_{13} = c_1$.

$$J = \begin{bmatrix} -(d_2 + l_3)s_1 & c_1 & -l_3 s_1 \\ (d_2 + l_3)c_1 & s_1 & l_3 c_1 \\ 1 & 0 & 1 \end{bmatrix}$$

This is only singular when $d_2 = 0$, making two collumns equal.



Note that there are no joint limits, infinite workspace, no workspace boundary singularities.

5.5.2 Force and torque at a kinematic singularity

Since $\tau = J^T F$, at a singular point, Q, we can expect non-zero forces F_j such that

$$J^T(Q)F_j = 0$$

In words, there will be some force vector or vectors (F_j) which can be applied to the end effector, which generate no torques at the joints. So, in a singular configuration, the mechanism can "lock up" with respect to tip forces or torques in certain directions becuase these directions allow no torque to develop at the joints. For example, suppose Q is defined as in Example 5 and

$$F_1 = \begin{bmatrix} -F\cos\theta_1 \\ -F\sin\theta_1 \\ 0 \end{bmatrix}$$

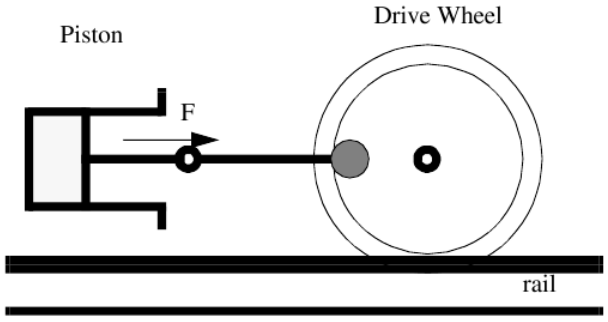
Note that this corresponds to a force applied to the tip in the direction opposite to the outstretched arm, and that no external torque is applied to the tip. Now, writing $J^T(Q)$ in a simplified form:

$$\tau = J^T(Q)F_1 = \begin{bmatrix} as_1 & -ac_1 & 1 \\ bs_1 & -bc_1 & 1 \\ cs_1 & -cc_1 & 1 \end{bmatrix} = \begin{bmatrix} -Fc_1 \\ -Fs_1 \\ 0 \end{bmatrix}$$

where $a = -l_1 - l_2 + l_3$, $b = -l_2 + l_3$, $c = l_3$. And

$$\tau = \begin{bmatrix} -aFs_1c_1 + aFs_1c_1 + 0 \\ -bFs_1c_1 + bFs_1c_1 + 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

The end effector force is finite, but the torque on all joints is zero. This situation is an old and famous one in mechanical engineering. For example, in an old railroad steam engine, "top dead center" refers to the following condition:



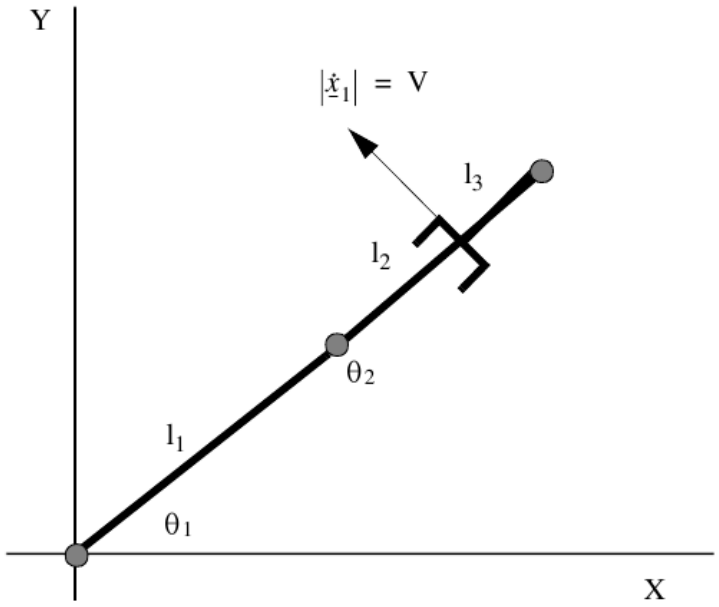
The piston force, F, cannot generate any torque around the drive wheel axis because the linkage is in a singular configuration as shown.

5.5.3 Velocity at a kinematic singularity

Note that although $\dot{\theta} = J^{-1}(Q)\dot{x}$ has no solution in general, if we assume that J looses rank by only one (i.e. $r - N = 1$), then there are $N - 1$ eigenvectors in the task velocity space, $\dot{x}_j$, for which solutions *do* exist. However, there will be multiple solutions. For example, if

$$\dot{x}_1 = \begin{bmatrix} -Vs_1 \\ Vc_1 \\ 0 \end{bmatrix}$$

Assume the manipulator is posed as below. The velocity vector we have just described is also plotted below.



We can see (by looking at the lever arms between joints and the end effector) that the valid solutions for the resulting joint velocities will be

$$\dot{\theta_1} = \begin{bmatrix} \frac{V}{(l_1+l_2-l_3)} \\ 0 \\ 0 \end{bmatrix}$$

$$\dot{\theta_2} = \begin{bmatrix} 0 \\ \frac{V}{(l_2-l_3)} \\ 0 \end{bmatrix}$$

$$\dot{\theta_3} = \begin{bmatrix} 0 \\ 0 \\ \frac{V}{(-l_3)} \end{bmatrix}$$

where the bold faced variable, $\dot{\theta}_i$ is a vector of the individual joint velocities. The end effector velocity is also satisified by linear combinations of these solutions such as

$$\dot{\theta}_{\mathbf{n}} = a_1\dot{\theta}_1 + a_2\dot{\theta}_2 + a_3\dot{\theta}_3$$

where $a_1 + a_2 + a_3 = 1$. This can be verified by multiplying any of the solutions by $J(Q)$ to obtain $\dot{x}_1$.

However, if any component of the tip velocity in the direction corresponding to the zero eigenvalue of J is non-zero, then at least one joint rate will go to infinity.

5.5.4 Kinematic Conditioning

So far we have looked at kinematic singularities from the point of view of specific joint values which cause the determinant to be zero. For example, in Example 5, when

$$\theta = Q = \begin{bmatrix} \theta_1 \\ 0 \\ \pi \end{bmatrix}$$

The manipulator was singular. In practical cases and computational implementations however, we have to be concerned about situations where the manipulator is *almost* singular. Suppose

$$\theta = \begin{bmatrix} \theta_1 \\ 0.01 \\ 3.15 \end{bmatrix}$$

Technically this is not singular. However J^{-1} may be difficult to compute. A measure of this difficulty is the *condition number* of the Jacobian, designated $\kappa(J)$. The condition number is

$$\kappa(J) = \frac{\sigma_{max}(J)}{\sigma_{min}(J)}$$

where σ_{max} is the maximum singular value J. When the Jacobian matrix looses rank, at least one singular value becomes zero. As we approach the singualrity, the condition number becomes very large.

As a practical matter, when using J^{-1} in a computation, it is not the exact singular configurations which must be avoided, but rather the regions around and including the singular configurations where the condition number of J exceeds a threshold.

5.6 Transformation of Jacobian between Representation Frames

Recall that in both the velocity propagation and the differentiation methods, we computed a velocity first and then factored out $\dot{\theta}$ to create the Jacobian matrix. Like any velocity, the one we base our Jacobian on could be represented in any frame and could be transformed by rotation between frames. Factoring each of these representations of end effector velocity will thus give us a different Jacobian matrix. The Jacobian matrix can thus be transformed among different frames by rotation as follows:

$${}^{A}J(\theta) = \begin{bmatrix} \begin{bmatrix} & {}^{A}_{B}R & \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} & {}^{A}_{B}R & \end{bmatrix} \end{bmatrix} {}^{B}J(\theta)$$

The special 6×6 matrix contains two identical submatrices which are ${}^{A}_{B}R$. These transform the linear and rotational components of the end-effector configuration space velocity separately but identically. Importantly, since the above 6×6 matrix contains two orthonormal sub-matrices, the eigenvalues of J are unchanged by this transformation. Thus singularities and condition number are not changed by rotation of the Jacobian into different frames.

5.7 Summary of Notation

Chapter 7

Trajectory Generation

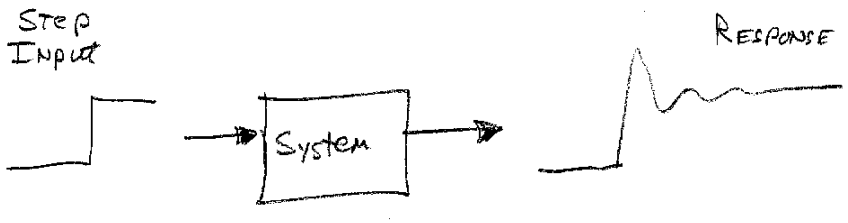
7.1 Problem Statement and Learning Objectives

Problem Statement If a manipulator is asked to move from one pose to another, it must do so smoothly and with speed and acceleration which is under the control of the programmer. This chapter describes techniques for generating smooth trajectories between two positions of the robot manipulator. Mathematically this problem can be thought of as that of finding a function of time which meets specified boundary conditions and other constraints.

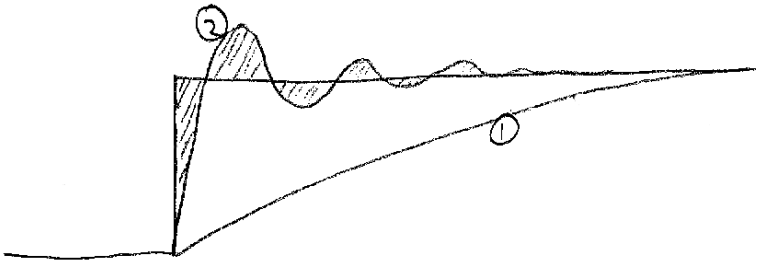
Learning Objectives After completing this chapter, the student will be able to

- Design or program linear trajectories with parabolic start and stop periods.
- Design or program trajectories based on cubic polynomials.
- Design or program trajectories which are constrained to move through via points.
- Design trajectories for the joints of a robot arm which result in straight trajectories in Cartesian space.

7.2 Trajectory Generation



Much of control theory focuses on the response of the system to a step input. There are good historical and contemporary reasons for this focus, but it is often inappropriate for motion control systems including robotics. Although no physical system can accurately follow the instantaneous position change of a step input, when viewed through the lens of control systems theory, the object is to design a controller which achieves desireable characteristics in the error. For example, the designer may adjust controller parameters to get fast response with overshoot or slow response without overshoot.



In both cases however, there is a large error between input and output. In this chapter we will avoid the problem of tracking a step input by changing to a smoother input. Such an input will demand lower velocities and accelerations and thus generate smaller tracking errors. Overshoot can be particularly bad when a robot operates in a crowded environment.

7.3 Joint Space Trajectory Generation

Suppose one joint of a robot arm needs to go from

$$\theta_A \rightarrow \theta_B$$

in an amount of time t_f. The problem of trajectory generation boils down to specifying a time function $\theta(t)$ such that

$$\theta(0) = \theta_A \quad \text{and} \quad \theta(t_f) = \theta_B$$

In addition, we will initially assume the manipulator should be at rest before and after the move giving

$$\dot{\theta}(0) = \dot{\theta}(t_f) = 0$$

The shortest distance between two points is a straight line. This naive approach to generating a trajectory gives us

$$\theta(t) = \theta_A + t \times \frac{\theta_B - \theta_A}{t_f}$$

However, because we have assumed the velocity at $t = 0^-$ and $t_f^+ = 0$, our acceleration must be infinite at the start and end of the trajectory. No robot can accelerate that fast and so we will have significant tracking error. Before we start looking at some candidate functions then, we should think about what we want in the function $f(t)$. We will consider a trajectory a "good" one if it satisifies two criteria: "Smoothness" and "Attainability".

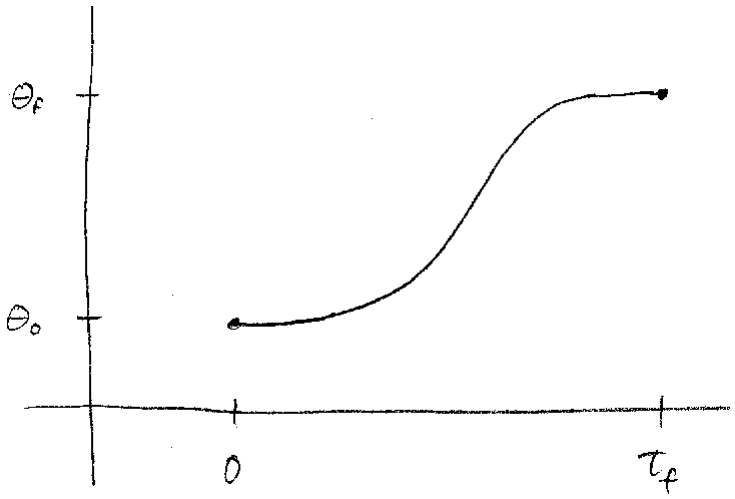
We will consider the function smooth if $\theta(t)$ and $\dot{\theta}(t)$ are continuous.

We will consider the function attainable if $|\ddot{\theta}(t)| \leq \ddot{\theta}_{MAX}$.

There are many possible functions which could meet these criteria. We will consider two in depth,

1. A polynomial
2. A linear function with parabolic start and end regions to limit maximum acceleration.

7.3.1 3rd Order Polynomial



A simple trajectory to start with is the 3rd order polynomial. We need a function of the form

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

with the following boundary conditions

$$\theta(0) = \theta_A \qquad \dot{\theta}(0) = 0$$
$$\theta(t_f) = \theta_B \qquad \dot{\theta}(t_f) = 0$$

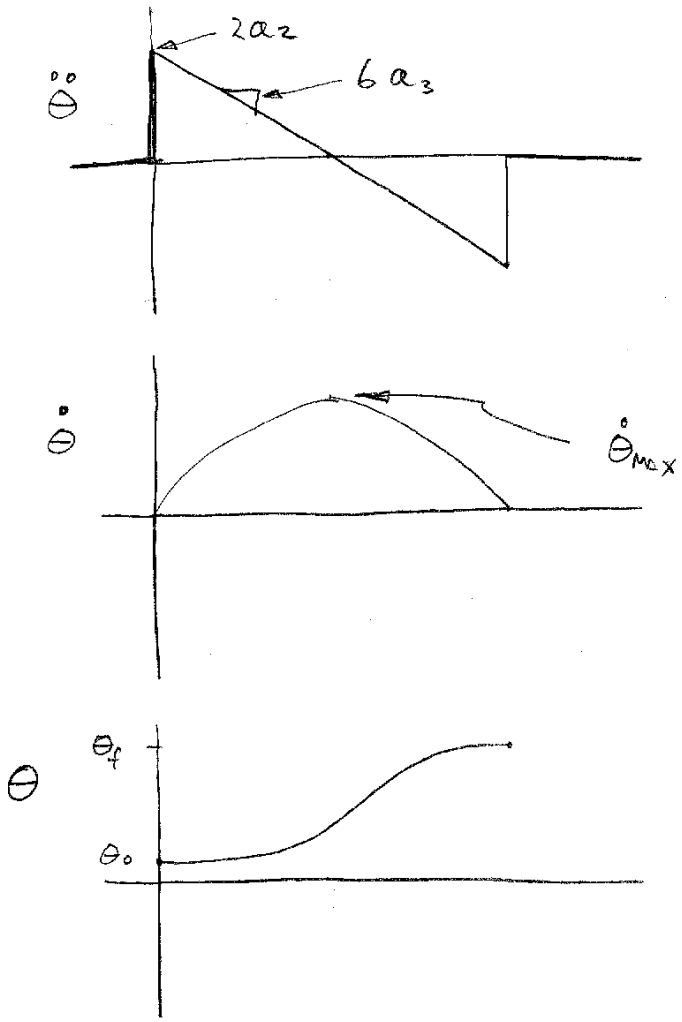
There are four unknowns ($a_0 \rightarrow a_3$) and the four equations above. By differentiating and applying the boundary conditions we can obtain

$$a_0 = \theta_A \qquad a_1 = 0$$
$$a_2 = \frac{3(\theta_B - \theta_A)}{t_f^2} \qquad a_3 = \frac{-2(\theta_B - \theta_A)}{t_f^3}$$

Note that the derivatives are easy to compute analytically:

$$\dot{\theta}(t) = 2a_2 t + 3a_3 t^2 \qquad \ddot{\theta}(t) = 2a_2 + 6a_3 t$$

The resulting trajectory looks like



We can also easily get:

$$\ddot{\theta}_{MAX} = \ddot{\theta}(0) = 2a_2 = \frac{6(\theta_B - \theta_A)}{t_f^2}$$

$$\dot{\theta}_{MAX} = \dot{\theta}(t_f/2) = \frac{3}{2}\frac{(\theta_B - \theta_A)}{t_f}$$

This function meets our requirements for smoothness and attainability if

$$\ddot{\theta}_{MAX} \geq 2a_2$$

Example 7.1
Design a 3rd order polynomial trajectory with $\theta_A = 40°$, $\theta_B = 120°$, $t_f = 2.4$sec. Also derive the velocity and acceleration of the trajectory:
ANS:

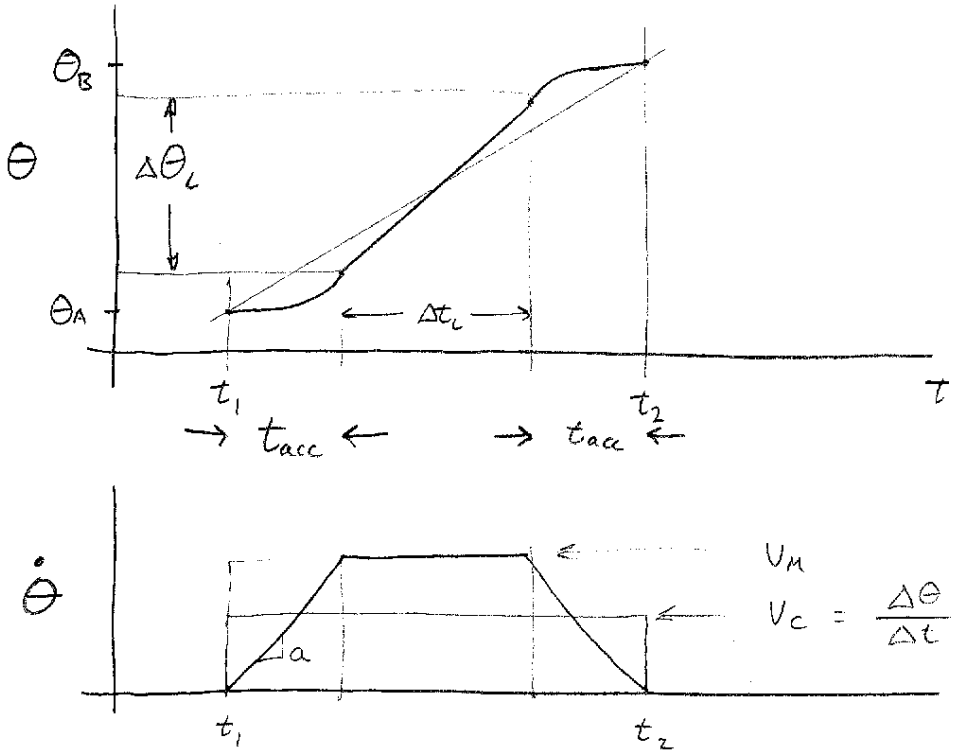
$$a_0 = 40°$$
$$a_2 = \frac{240°}{2.4^2} = 41.67$$
$$a_3 = \frac{-2 \times 80°}{13.824} = -11.57$$
$$\theta(t) = (40 + 41.67t^2 - 11.57t^3)°$$

by differentiation:

$$\dot{\theta}(t) = 83.34t - 34.71t^2$$
$$\ddot{\theta}(t) = 83.34 - 69.42t$$

7.3.2 Linear with Parabolic Blends

To understand this trajectory, it is easiest to look first at the velocity curve of the figure below:



Using this trajectory type, the velocity increases linearly for a time interval t_{acc}, then continues at a constant velocity, V_M, and then decreases linearly to zero for t_{acc} seconds prior to the end of the trajectory. The key constraint is that the area under the velocity curve must be equal to the displacement, $\theta_B - \theta_A = \Delta\theta$. The area of the velocity curve is $V_M(t_2 - t_1 - t_{acc})$. Thus:

$$V_M(t_2 - t_1 - t_{acc}) = \Delta\theta$$
$$V_M = \frac{\Delta\theta}{t_2 - t_1 - t_{acc}}$$

We can further obtain the acceleration of the velocity ramp, a

$$a = \frac{V_M}{t_{acc}} = \frac{\Delta\theta}{t_{acc}(t_2 - t_1 - t_{acc})}$$

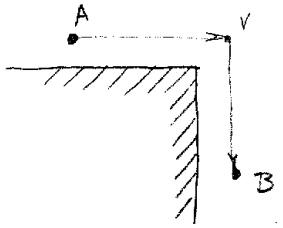
Implementation of this trajectory in software is a bit more complex than the polynomial. We must divide the trajectory time into three phases: acceleration, cruising, and deacceleration.

Example 7.2
Write pseudo code to generate a linear trajectory with parabolic blends. Divide your code into two functions, one which plans the trajectory and returns the constants V_M, a etc. and a second one which is called every 0.001 seconds and computes $\theta(t)$. Make sure that acceleration is not greater than a predefined `A_MAX`.
ANS:

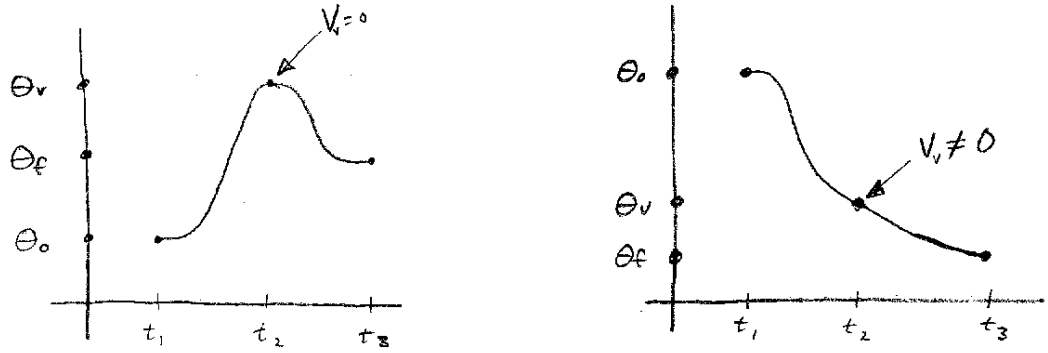
```
\\ generate constants
lin_poly_init(float delta_theta, float delta_t)
 {
 t_acc = 0.25 * delta_t; // arbitrary, other ways could be better
 VM = delta_theta/(delta_t - t_acc);
 a = VM / tacc;
 if(abs(a) > A_MAX) { error("Acceleration too high"); }
 }

\\run time call
lin_poly_run(float t, float t_0, float theta_A, float delta_t,
                                  float t_acc, float a, float VM)
 {
 dt = t-t_0;
 if(dt < t_acc) { // acceleration phase
   return(0.5 * a * (t-t_0)^2 + theta_A);
   }
 else if (dt > t_acc && dt < delta_t-t_acc) { // cruise
   return(0.5 * t_acc*VM + VM*(t-t_0-t_acc) + theta_A);
   }
 else if (dt > delta_t - t_acc) {  // decelleration
   return((theta_A+delta_theta)- 0.5*a*(t-delta_t)^2);
   if(t == delta_t)  stop;
   }
 }
```

7.3.3 Via Points

We may need to specifiy an intermediate point between a start and a goal. For example, in the case below, starting at the point A, we must go to the point V before B in order to avoid the obstacle.



If we translate the three points to joint configurations, we need a trajectory like one of the two trajectories shown below.



In the example at left, we need to go through zero velocity at point V but in the case at right it would break up the motion to require zero velocity at the via point. We should specify some non zero velocity at the via point for smoother motion. We could allow the user to specify this velocity. Alternatively, to automatically generate it, we can use some heurisitics such as

1. if the difference in θ changes sign, $V_V = 0$.
2. if not, average the slopes:

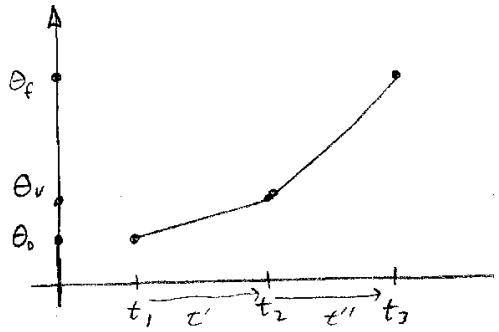
$$V_V = 1/2 \left(\frac{\Delta\theta_1}{\Delta_{t1}} + \frac{\Delta\theta_2}{\Delta_{t2}} \right)$$

Once we have derived a value for the velocity in the via point, V_V, we can compute the new trajectory as follows: First, redo the derivation of Section 7.3.1 but alter the boundary conditions:

$$\dot{\theta}_1(t_2) = \dot{\theta}_2(t_2)$$

Where the left hand side is the final velocity of the first sub-trajectory, and the right hand side is the initial velocity of the second sub-trajectory.

Alternatively, we can require that the acceleration, $\ddot{\theta}(t)$ is continuous at $t = t_2$. Consider two example trajectories shown below:



Let's introduce intermediate time variables so that we can keep each polynomial simple:

$$t' = t - t_1 \qquad t'' = t - t_2$$

Then the two segments have the equations:

$$\theta_1(t) = a_{10} + a_{11}t' + a_{12}t'^2 + a_{13}t'^3$$
$$\theta_2(t) = a_{20} + a_{21}t'' + a_{22}t''^2 + a_{23}t''^3$$

There are thus 8 unknowns and we need eight equations to solve them:

1. $\theta_0 = a_{10}$
2. $\theta_f = a_{20} + a_{21}(t_{f2}) + a_{22}t_{f2}^2 + a_{23}t_{f2}^3$
3. $\theta_V = a_{20}$
4. $\theta_V = a_{10} + a_{11}t_2 + a_{12}t_2^2 + a_{13}t_2^3$
5. $\dot{\theta}_1(t_1) = 0$
6. $\dot{\theta}_2(t_3) = 0 = a_{21} + 2a_{22}t_{f2} + 3a_{23}t_{f2}^2$

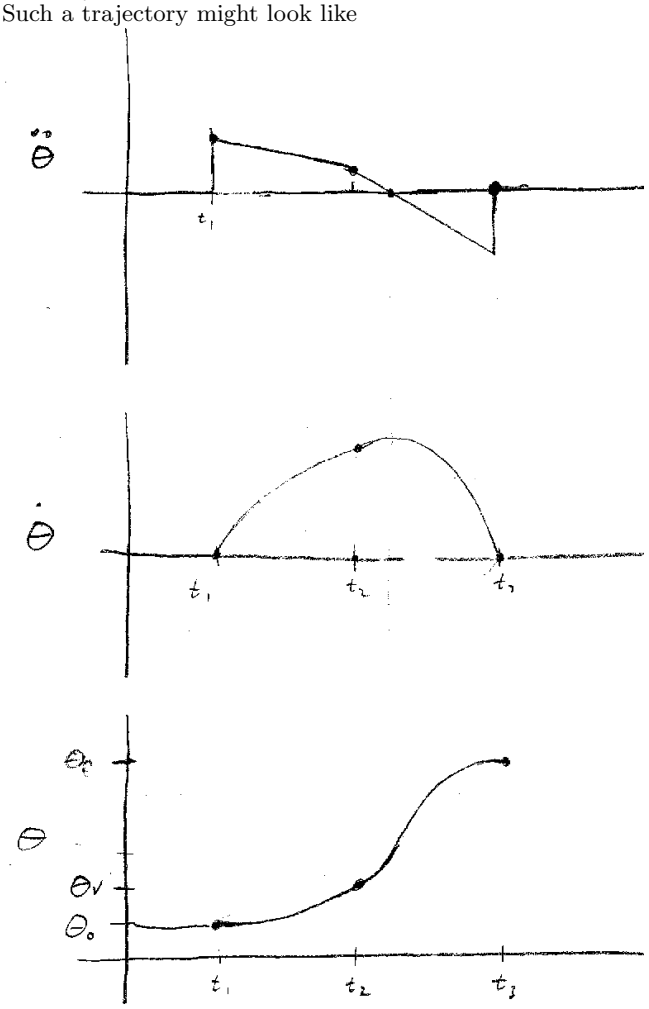
7. continuous velocity:

$$\dot{\theta}_1(t_2) = \dot{\theta}_2 t_2 \rightarrow a_{11} + 2a_{12}t_2' + 3a_{13}(t_2')^2 = a_{21}$$

8. continuous acceleration:

$$\ddot{\theta}_1(t_2) = \ddot{\theta}_2 t_2 \rightarrow 2a_{12} + 6a_{13}t_2' = 2a_{22}$$

where $t_{f2} = t_3 - t_2$. This can be solved readily and coded into a subroutine. Such a trajectory might look like

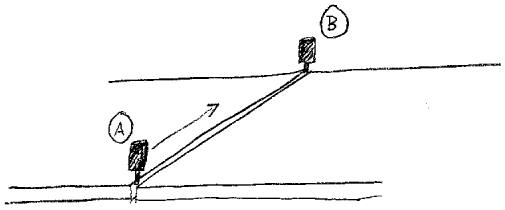


The methods above could generate the trajectory for a single joint, $\theta_i(t)$. To move a robot arm from one pose to another, we work out the joint angles for the starting and ending poses (using inverse kinematics) and then plan a trajectory for each joint from its value in pose A to its value in pose B where each joint uses the same t_f. This produces nicely synchronized and smooth motion.

7.4 Cartesian Straight Line Motion

7.4.1 Position

Sometimes it is not enough to plan the trajectory in the manipulator's joint space. While the joint space trajectory may be smooth and attainable, we do not know where it goes between the start and end points in Cartesian space. In the welding task shown below, we need to make sure that the end effector motion goes in a straight line at a constant speed.

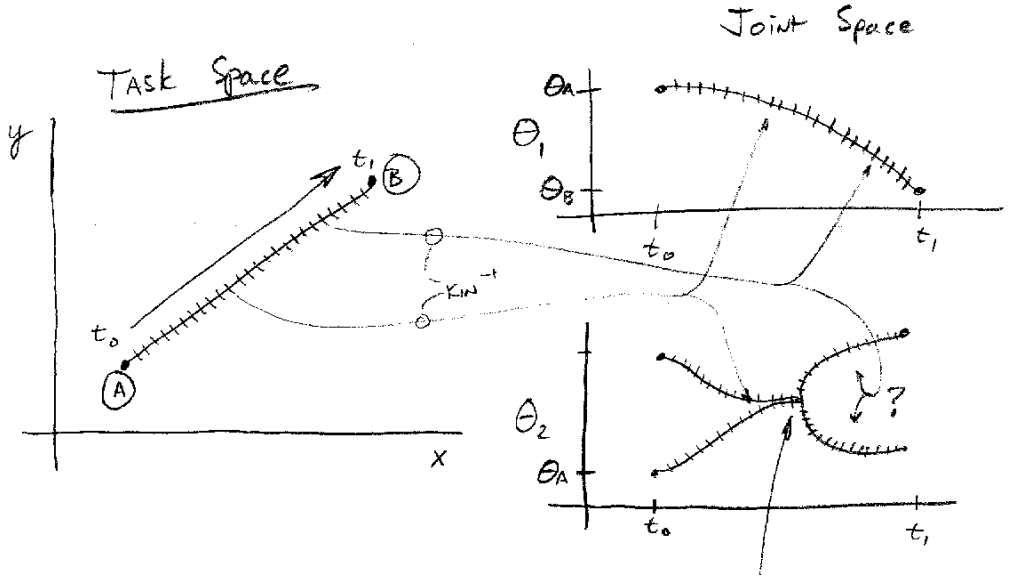


First, consider only the Cartesian end effector position and neglect orientation. If we have 3×1 start position P_A and end position P_B, we can generate a straight line path by linear interpolation. Letting the time variable, t be referenced to the start of the trajectory,

$$P(t) = P_A + \frac{t}{t_f}(P_B - P_A)$$

Then, at each sample of time, we can compute the joint positions with inverse kinematics. Note that for now we have neglected the starting and stopping accelerations.

Now, we know where the end effector is at each point along the Cartesian trajectory. This is very important when obstacles might be present or when the trajectory is important to the task itself such as welding. Two important issues however which must be considered with this method are choosing correctly between multiple solutions of the inverse kinematics, and dealing with trajectories which go through singular points. The drawing below illustrates these points



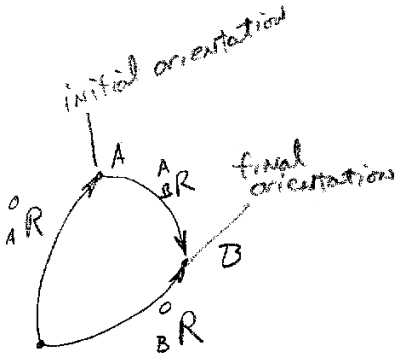
On the left, a straight line trajectory is shown in the Cartesian end effector space of the robot. The path is divided into sample points corresponding the linear interpolation function at several points in time. On the right are the corresponding inverse kinematics solutions for two joints of the robot. For θ_1 (top) the inverse kinematics equations produce a single solution at each point for an unambiguous path. However for θ_2, the inverse kinematic equations produce two solutions. Furthermore, at a singular point (see Chapter 5), the two solutions merge into one and afterwards diverge into two separate solutions again. When controlling an actual robot, it is important that the algorithms consistently pick solutions which are close to each other in the joint space and make an intelligent choice of solutions when coming out of the singularity. If control algorithms rely on the Jacobinan matrix inverse, they must be robust to the numerical problems which arise both at and in the neighborhood of the singularity.

Not all paths will go through singular points, but trajectory generation algorithms which rely on inverse kinematics must be prepared to encounter them.

7.4.2 Orientation

What if the start and end points have different orientations of the end effector? How do we interpolate orientation? A naive approach might be to interpolate the orientation matrices from R_A to R_B. However it is easily shown that the interpolated matrices produced are not valid rotation matrices. The answer is to convert the rotation matrices to one of the parameterized forms such as equivalent angle-axis form. First, derive a rotation matrix which maps from the initial to final orientation. Using the transform graph below, we can solve for ${}^A_B R$ as

$${}^A_B R = {}^0_A R^{-1} {}^0_B R$$



where ${}^0_A R$ is the orientation at the initial point A etc. Using equation () from Chapter 2, we can convert ${}^A_B R$ to a fixed axis, K, and a rotation angle around that axis, θ_{AB}. Now we can keep K fixed, and use linear interpolation for the angle $\theta_{AB}(t)$:

$$\theta_{AB}(t) = 0 + \left(\frac{t}{t_f}\right)\theta_{AB}$$

the first term is zero because ${}^A_B R$ was defined relative to position A. Then, the current rotation change is reconstructed from K and $\theta_{AB}(t)$ and combined with ${}^0_A R$ to get the current rotation matrix

$${}^A_B R = {}^0_A R\, {}^A_B R(t)$$

7.4.3 Parameterized Cartesian Trajectories

In the previous treatment of linear interpolated trajectories, we have ignored the inifinite acceleration at the start. Full consideration of this problem requires consideration of dynamics in future chapters, but we can at least address the concern by doing the linear interpolation with a parameter, $0 \leq p \leq 1$, and generating a smooth time function for p. Considering position, we can convert our interpolation function to p by

$$P(t) = P_A + p(t)(P_B - P_A)$$

where, for example,

$$p(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

We solve for a_0 to a_3 as in Section 7.3.1 where

$$p(0) = 0 \qquad p(t_f) = 1$$

$$\dot{p}(0) = \dot{p}(t_f) = 0$$

giving

$$a_0 = 0 \qquad a_1 = 0 \qquad a_2 = \frac{3}{t_f^2} \qquad a_3 = \frac{-2}{t_f^3}$$

While this gives us a smoother straight line trajectory, we do not know which trajectories are attainable. Also, a task like welding may require constant velocity over the entire path. To properly perform the weld then we should use the linear-parabolic function instead of the polynomial for $p(t)$ and extend the length of the trajectory to allow space for the manipulator to accelerate and decelerate before and after the weld line.

7.5 Summary of Notation

Chapter 10

Mechatronics and Design of Manipulators

10.1 Problem Statement and Learning Objectives

Problem Statement This chapter considers some aspects of design of serial link mechanisms for manipulation.

Learning Objectives Upon completing this Chapter, the reader should be able to

-

I'm worried that this chapter will have to be too superficial. Maybe an example with one or two DOF but actual implementation??

10.2 Sensors

10.2.1 Force Sensors

Load Cell A **"Load Cell"** is a structure which supports the load and deflects a known amount in response to applied forces and torques. The deflections are measured to characterize the applied forces and torques.

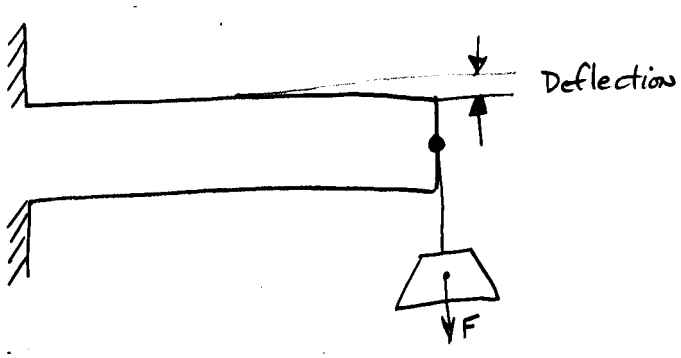


Figure 10.1: Cantilever Beam. When load F is applied to the end of the beam, the beam deflects a known amount.

Example: Cantilever Beam

Stress and Strain

Stress

- Stress (σ): Force per unit area in one dimension.

$$\sigma = \frac{F}{A}$$

  Units: Pascals = $\frac{N}{M^2}$, psi.

- Stress can be refered to as "Compressive" (tending to compress the material)and "Tensile" (tending to elongate it).

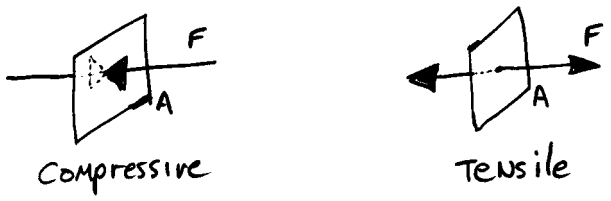


Figure 10.2: Illustrations of Compressive and Tensile stress through a patch of area A.

- Shear stress refers to forces *in* the plane. Shear stress can cause material to tear.

Figure 10.3: Shear stress. Shear stress is due to forces in the plane.

- 3-Dimensions: Full analysis of stress in 3-dimensions is complex an requires notion of *Stress Tensor* — a 3×3 matrix giving the complete state of stress at a given point in the material.

Strain

- Strain, (ϵ), is relative deformation.
- 1 Dimension: Elongation Ratio

$$\epsilon = \frac{\Delta L}{L}$$

- if $\frac{\Delta L}{L} = 10^{-5}$ we say "10 micro-strain".
- Strains of typical metal structures are very small (i.e. micro-strain).
- Strains in biological tissues can be much larger ($\epsilon \approx 1$).
- Advanced: Several Types of Strain

Hooke's Law (Elastic Deformation)

$$\sigma = \mathrm{E}\epsilon$$

where E is Young's Modulus.

- Biological materials rarely obey Hooke's Law.
- Metals obey Hooke's Law very well for stresses below the *elastic limit.* If stress is greater, material permanently deforms.

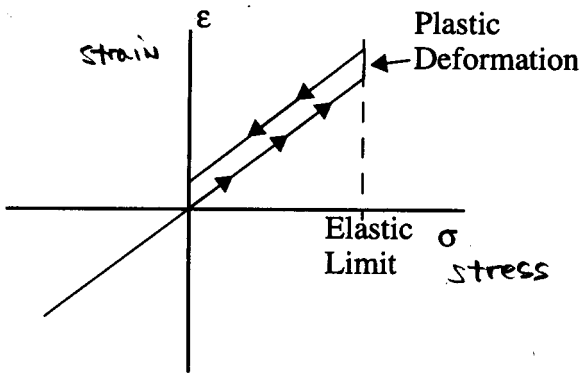


Figure 10.4: Hookean behavior with Elastic Limit. If stress increases beyond Elastic Limit, material is permanently deformed.

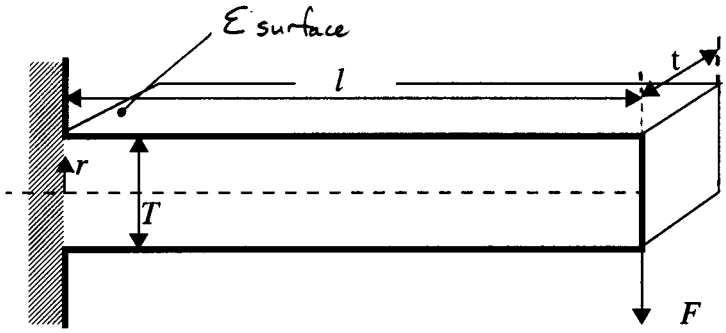


Figure 10.5: Rectangular cantilever beam is a simple load cell structure for basic analysis.

Cantilever Beam We want to derive $\epsilon_{surface}$ as a function of the applied load, the dimensions of the beam, and the beam material's stiffness (Young's modulus). First, we assume that the beam is at static equilibrium around the centerline along the wall. For this we equate the moment developed in the structure around the center line with the moment applied by the external load.

$$Fl = \int_{-\frac{T}{2}}^{\frac{T}{2}} r\sigma(r)tdr$$

where r, t, F, l are defined in the figure, and $\sigma(r)$ is the internal stress where the beam joins the wall. Let us assume that $\sigma(r)$ varies linearly through the beam so that

$$\sigma(r) = \alpha r$$

where α is some positive constant.

Then we have

$$Fl = \alpha t \int_{-\frac{T}{2}}^{\frac{T}{2}} r^2 dr = \frac{\alpha t T^3}{12}$$

Solving for α,

$$\alpha = \frac{12l}{tT^3}F$$

$$\sigma(r) = \frac{12lr}{tT^3}F$$

Finally, we can evaluate $\sigma(r)$ at the surface and use Young's modulus to convert stress to strain:

$$\sigma(\frac{T}{2}) = \frac{12l\frac{T}{2}}{tT^3}F$$

$$= \frac{6l}{tT^2} F$$

$$\epsilon_{surface} = \frac{6l}{EtT^2} F$$

Strain Gages (see `http://www.measurementsgroup.com/guide/index.htm`)

A strain gage is a device which transduces strain on a surface. There are two types in common use, resistive metal foil, and semiconductor. Both transduce strain by taking advantage of the variation of electrical resistance with strain.

Metal Foil Strain Gages Metal foil gages are bonded to the surface and their resistance changes in proportion to the surface strain.

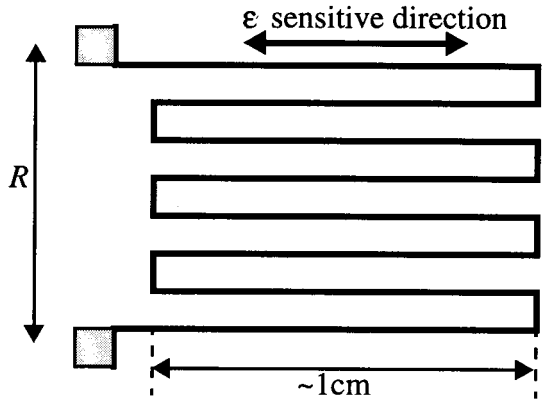


Figure 10.6: Diagram of a metal foil strain gage. Conductor in the shape shown is attached to surface. As surface deforms, conductive pattern changes length which changes resistance between contact pads (left).

In simplified form,

$$R = R_0(1 + \epsilon g)$$

where $g = \frac{\Delta R}{R_0 \epsilon}$ is called the "gage factor". For metal foil gages, $g \approx 3$.

Semiconductor strain gages These are small chips of silicon with a single region of doping to create a resistor. The chip is bonded to the load cell and its resistance varies with strain. With a semiconductor strain gage, the designer can expect signficantly higher gage factor of around $g \approx 150$.

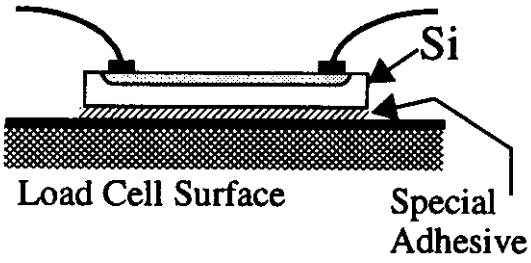


Figure 10.7: Diagram of a semiconductor strain gage.

Properties of Strain Gages

1. Very linear. 1% is achievable due to Hooke's law. Elastic limit must be avoided with a safety margin.
2. Wide dynamic range. Example: Force/Torque sensor designed by JPL for Space Shuttle RMS robot arm design range: 1N to 4×10^5N. 112db.
3. Temperature Sensitivity. Both types are highly sensitive to temperature.
4. Interface between gage and structure requires care due to different coefficients of thermal expansion between gage and structure.
5. Practicalities. Foil gages are robust and cheap. Semiconductor gages are delicate and expensive. Foil gages are fixed with an adhesive which usually requires curing in an oven. Semiconductor gages require an elaborate process for attachment which must be done by a specialist.

Temperature Sensitivity Careful thermal analysis is important for accurate force sensing with strain gages. Two effects account for the temperature sensitivity of metal foil strain gages:

- Thermal Coefficient of Resistance of gage material.
- Difference in thermal coefficient of expansion of gage and load cell structure.

The first causes resistance to change only as a function of temperature and not of strain. The second causes strains in the gage as a function of temperature only.

The temperature dependence of the gage resistance can be described by:

$$r = \frac{\frac{\Delta R}{R_0}}{\Delta T} = \beta_g + g\left(\frac{1+K_t}{1+\nu_0 K_t}\right)(\alpha_s - \alpha_g)$$

Where

β_g = thermal coefficient of resistance of the gage material.

K_t = transverse sensitivity of the gage: the gage factor for strains orthogonal to the sensitive direction (extremely small for most gages and will be neglected).

ν_0 = Poisson's ratio (typically 0.3)

α_s, α_g = coefficients of thermal expansion of substrate and gage respectively.

The first step to reduce temperature sensitivity is to make sure the thermal expansion coefficients of the gage (α_g) and substrate (α_s) are as close as possible.

We can then relate resistance to strain and temperature through

$$R = R_0(1+g\epsilon)(1+r\Delta T)$$

The second step to compensating for the remaining temperature sensitivity is to make a differential measurement of two gages which have the same temperature but opposite strains. In the case of a cantilever beam, if we assume that the beam is at a uniform temperature, this can be arranged by putting gages on either side of the beam. By symmetry, $\epsilon_1 = -\epsilon_2$.

Dual Gage beam By symmetry, $\epsilon_1 = -\epsilon_2$.

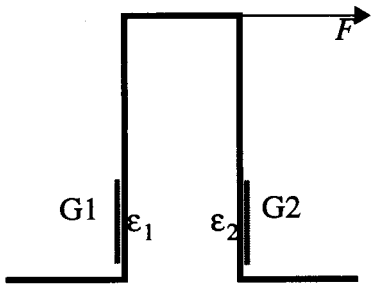


Figure 10.8: Cantilever beam load cell with gages on either side. Strains ϵ_1 and ϵ_2 will be equal and opposite. If temperature is same on both sides of beam, we can correct for temperature sensitivity.

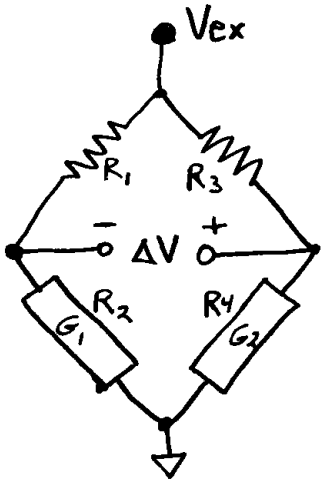


Figure 10.9: Whetstone Bridge circuit used to subtract resistance changes of the two strain gages.

Whetstone Bridge Subtraction of the two strain signals is usually accomplished right in the load cell itself using a Whetstone Bridge Circuit.

A very nice javascript animation illustrating this principle is available at `http://www.rdpe.com/us/hiw-sglc.htm`. Be sure to drag the beam tip with your mouse!

Analysis of the bridge circuit yields:

$$\frac{\Delta V}{V_{ex}} = \frac{R_4}{R_3 + R_4} - \frac{R_2}{R_1 + R_2} = \frac{R_4 R_1 - R_2 R_3}{(R_1 + R_2)(R_3 + R_4)}$$

In this example, we place the two strain gages into the bottom legs of the bridge, (R_2, R_4). The other two legs of the bridge are non-strain-sensitive resistors (but still temperature sensitive).

We define $\hat{R} = R_0(1 + r\Delta T)$.

$$R_1, R_3 = \hat{R}$$
$$R_2 = \hat{R} + (1 + r\Delta T) g R_0 \epsilon_1$$
$$R_4 = \hat{R} + (1 + r\Delta T) g R_0 \epsilon_2$$

From the load cell design, we have $\epsilon_2 = -\epsilon_1$.

Computing the output voltage we get:

$$\frac{\Delta V}{V_{ex}} = \frac{\hat{R}(1 + r\Delta T) 2 g R_0 \epsilon_2}{4\hat{R}^2 - (1 + r\Delta T)^2 g^2 R_0^2 \epsilon_1^2}$$

Expanding $\hat{R}$ an canceling terms:

$$\frac{\Delta V}{V_{ex}} = \frac{2g\epsilon_2}{4 - g^2\epsilon_1^2}$$

Ignoring the high order terms:

$$\frac{\Delta V}{V_{ex}} = \frac{g\epsilon_2}{2}$$

Temperature variation is eliminated! Non-linear term contributes insignificant errors because ϵ is very small.

Multi-Axis Sensing In robotics and surgery it is rarely of interest to measure a single force. In general we want to measure the 6 components of force and torque,

F_X
F_Y
F_Z

τ_X
τ_Y
τ_Z

We want to characterize these force/torque components at some interface between two objects. Examples of these interfaces: robot wrist — robot hand, surgical tool handle — surgical tool body, etc. So that the sensor does not distort the forces and torques present, it should be as rigid as possible. Therefore the two objects connected by the sensor should normally be rigidly connected.

Design of load cells for this measurement takes substantial effort and is beyond the scope of this course. However it is useful to consider two example designs.

"Maltese Cross" Sensor

Variations of this design have been produced in a number of laboratories and are available commercially (Fig. 10.10). In this design, the spokes of a wheel are instrumented with gages and calibrated to measure forces and torques applied to the hub relative to the rim.

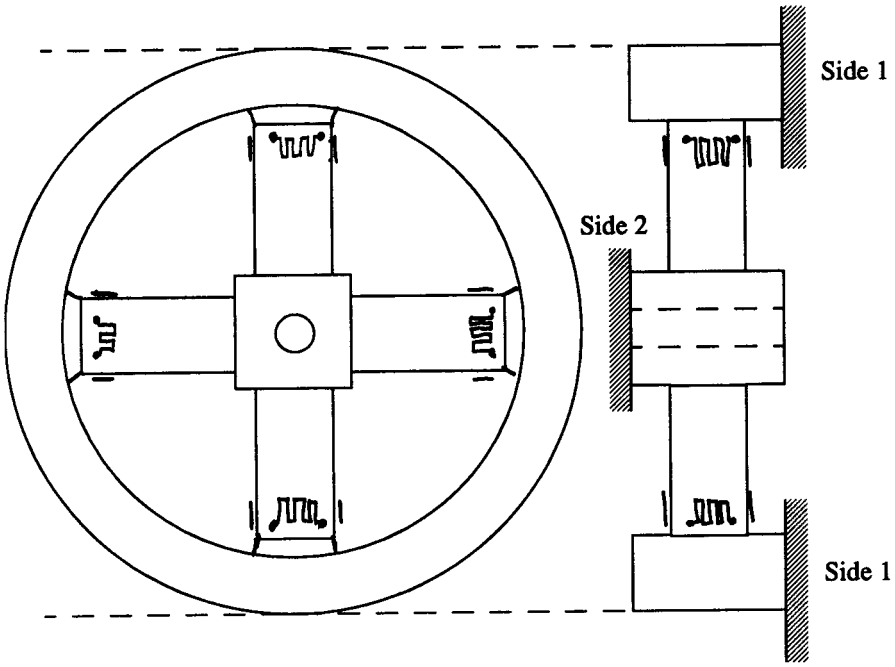


Figure 10.10: "Maltese Cross" force-torque sensor. Forces and torques applied to the surface labeled 'side 1' cause strains in the spokes which are picked up by 8 attached gages.

Four gages are applied to each beam for a total of 8 differential measurements.

A 6x8 calibration matrix C is computed to relate the strain measurements to forces/torques

$$\begin{bmatrix} F_x \\ F_Y \\ F_Z \\ \tau_X \\ \tau_Y \\ \tau_Z \end{bmatrix} = \begin{bmatrix} \qquad C \qquad \end{bmatrix}_{6\times 8} \begin{bmatrix} \epsilon \end{bmatrix}_{8\times 1}$$

In practice, manufacturing tolerances are not adequate to compute C from the load cell design with sufficient accuracy so it must be obtained from measurements of strain outputs as a function of applied forces and torques for each device using regression analysis.

Parallel-Plate Structure

(See T. Yoshikawa and T. Miyazaki, "A six-axis force sensor with three dimensional cross shape structure," Proc. Intl. Conference on Robotics and Automation, pp. 249-254, 1989.)

An alternative to the bending beam load cell is the parallel plate structure (also known as a flexure). In this case, two thin beams are arranged in parallel (Figure 10.11).

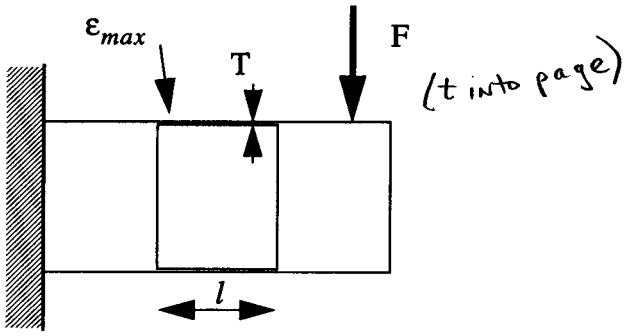


Figure 10.11: Parallel plate flexure load cell for a single axis.

Analysis At the point of maximum strain concentration, labeled ϵ_{max}, the strain is

$$\epsilon_{max} = \frac{3lF}{\mathrm{E}tT^2}$$

where t is the width of the parallel plate structure, and E is Young's Modulus for the material.

Overload Protection We saw that above a certain critical strain, the "elastic limit," a material will permanently (plastically) deform. This has two important consequences. First, if the material deforms plastically, the zero reading of the load cell will have to be recalibrated. Second, the load cell will be damaged and will eventually break or suffer reduced life.

In most realistic force sensing applications, it is difficult or impossible to predict the maximum forces likely to be experienced by the sensor. This is because collisions between the sensor and the environment often occur, either deliberately or as a result of errors. The transient forces which occur in these collisions (especially with hard objects) are hard to control and may exceed the design force limit for the sensing beam.

To prevent damage from these types of events, overload protection is sometimes designed into force/torque sensors. An overload protection device must allow safe deflection of the load cell in all active directions without disturbance forces, but must provide greatly increased stiffness and strength for deflections above the safe operating point (which may be a factor of two or more *below* the elastic limit.)

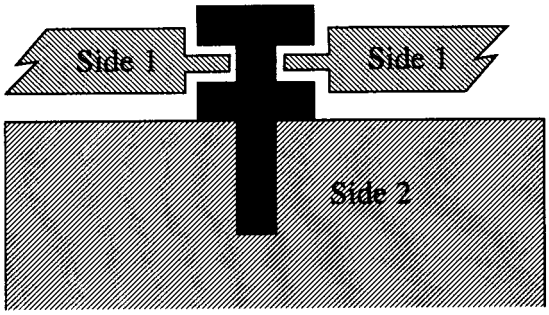


Figure 10.12: Overload protection scheme for a force/torque sensor. a high-strength structural element connects the two sides when deflection exceeds the size of its gap to protect the sensing structure from excessive strain.

In the example illustrated in Figure 10.12, a pin of strong material (i.e. steel in an aluminum load cell design) is affixed between the two sides of the load cell so that there is a calibrated gap between the two sides. When the strain reaches the maximum safe operating point, the two sides touch and the stiffness of the overall loadcell is greatly increased. Once the two sides come into contact through the pin, the amount of force required to reach the elastic limit is greatly increased. Any further deflection is thus reduced or eliminated.

10.3 Actuators

10.3.1 DC Motors

Introduction

This is a quick overview of DC motors as applied to robot manipulators and haptic devices. Although DC motors are a very mature technology, the companies which produce them are mainly focused on a non-robotics market. Many commercial applications are aimed at high velocities and moderate torques. Robots often require low velocities and high torques.

The most common way to match available motors to robot needs is to use gears. A gearbox with ratio n will reduce motor velocity by n^{-1} and increase torque by n. Unfortunately, some undesirable properties, such as the effective inertia of the motor and any friction present in the motor are increased by n^2. For high performance robotics, including force controlled robots, teleoperators, and haptic interfaces, these negative effects of gears are significant and so n must be minimized or made equal to 1.

When $n = 1$ in a robotic application, the motor "sees" a load characterized by velocities near zero and "high" torques. When torques are adequate for the application motors tend to get hot. The designer can select among different motors, but there is a trade-off which cannot be avoided between heat capacity and the mass of the motor. Furthermore, large currents can be required which in turn require large power supplies and thick wires. Many of the standard techniques for selecting motors (such as those discussed in motor catalogs) can be too conservative and result in too much weight for your robot.

This section explores how to optimize the selection of a motor for robotic requirements and how to better understand the thermal behavior of motors.

Basic Theory

DC motors are one of the most common actuators for robotics. Althought there are many important electric motor technologies for robot manipulators, here we consider the most basic, brushed DC motors.

Two basic laws describe the operation of DC motor in terms of the pair of electrical input variables, V and I, and mechanical output variables, torque, τ and angular velocity, ω.

$$\tau = K_m \times I \tag{10.1}$$

$$V = IR + K_m\omega + L\dot{I} \tag{10.2}$$

where R is the resistance of the armature windings, K_m is the motor constant[1] and L is the motor inductance. We will consider constant currents (i.e. $\dot{I} = 0$) and ignore the motor inductance.

To illustrate the use of these equations, let us pose some problems:

Stall Torque

A Maxon RE-25/118752 DC motor[2] has a motor constant, K_m of .0234 Nm/A, Coil resistance, R of 2.32 Ω, and winding inductance, I, of 0.24 mH. In a haptic application, assume $\omega = 0$ (this is referred to as "stall"). Assume $\frac{dI}{dt} = 0$. Find the current and voltage required to generate a torque of 0.05 Nm.

Solution: Using Equation (10.1),

$$0.05Nm = 0.0234Nm/A \times I$$

$$I = \frac{0.05}{0.0234} = 2.14A$$

Using Equation 10.2 for $\omega = 0$, we have

$$V = IR$$

In other words, for $\omega = 0$, the motor acts like a resistor.

$$V = 2.14A \times 2.32\Omega + K_M \times 0 + L \times 0$$

$$V = 4.96V$$

[1] Note that when MKS units are used there is only one such constant. When other unit systems are used, two different constants must be used in the two laws.

[2] `http://www.maxonmotorusa.com`

Figure 10.13: Maxon brushed DC Motor with gearhead and optical encoder (http://www.maxonmotorusa.com).

Free Running / Viscous Load

Using the same motor as in problem 10.3.1 find the speed the motor will run with no load ($\tau = 0$) when 5V is applied. What will be the current and power dissipation? Assume $\frac{dI}{dt} = 0$ and that there is no friction in the motor.

Solution: Using Equation (10.2),

$$5.0V = I \times 2.32\Omega + 0.0234Vsec \times \omega$$

Using Equation (10.1)

$$I = 0(!)$$

Solving:

$$\omega = 213.7rad/sec = 2041rpm$$

One way to think about this is that the term $K_m \times \omega$ is a voltage (called "back EMF") generated internally by the motor which opposes the applied voltage. At no-load, this voltage exactly cancels the applied voltage and the current is zero.

Since the current is zero, the power is also zero! This is not a perpetual motion machine, because we have assumed there is no friction. Electrical Power in = mechanical power out:

$$V \times I = \tau \times \omega = 0$$

Now, let us consider frictional load to the motor. The load will be viscous friction with coefficient $B = 1.2 \times 10^{-4} nMsec/rad$. Physically, this load could be due to 1) the motor's internal bearing friction and air friction, 2) the friction of the transmission mechanism such as gears, belt drives, cable drives, or 3) the external load (such as a hole being drilled, drink being blended, etc). Find ω, the current, and the power dissipation when the viscous load is applied.

Solution: The viscous friction load forms another equation which much be solved simultaneously with the motor equations:

$$\tau = B \times \omega$$

Bringing in Equations (10.2) and (10.1),

$$5.0V = I \times 2.32\Omega + .0234 \times \omega$$

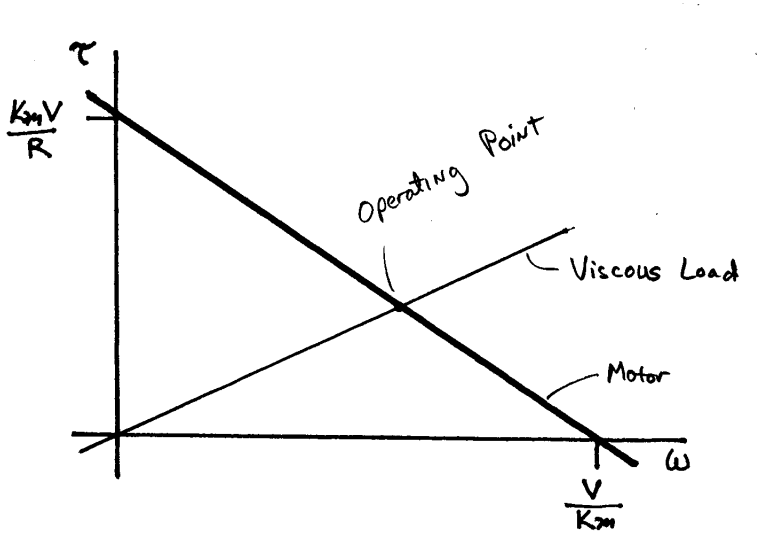
$$\tau = 0.0234 \times I$$

$$\tau = 1.2 \times 10^{-4} \times \omega$$

$$I = \frac{1.2 \times 10^{-4}}{.0234}\omega \quad = 5.128 \times 10^{-3}\omega$$

$$5.0 = (5.128 \times 10^{-3} + .0234)\omega$$
$$\omega = 175.2 rad/sec = 1674 rpm, \quad I = 0.898A$$
$$P = 5.0V \times 0.898A = 4.49 Watts$$

We can also solve this graphically:



Power Supply / Winding Selection

A DC motor is rated at 150 Watts continuous power (this means that the elecrical power cannot exceed 150 Watts in the steady state or the coils will burn out). The manufacturer has a variety of 150 Watt motors available, each with various values of stall torque (τ when $\omega = 0$), coil resistance, motor constant, etc, which are listed in a table. The manufacturer obtains all these different motors by winding the coils with different diameter wire. For small wire, the number of turns which fit into the space is greater (greater K_m), but the resistance becomes higher. We want to use a 30Volt power supply. **Problem:** Find the value of R which will give us 150 Watt power dissipation at 30V when $\omega = 0$. This way we can get the most stall current and therefore the most stall torque. Find the current I_{max} and torque τ_{max} at 30V.

Solution: We want our 30V power supply to give the maximum permissable power in this case so:

$$P = \frac{E^2}{R}$$
$$150 = \frac{900}{R}$$
$$R = 6\Omega$$

Now we look through the catalog table for the motor who's coil resistance is closest to 6Ω. In this case, the stall current is

$$I = \frac{30}{6} = 5A$$

The torque constant for this motor (made up 150W/ 6Ω motor) is 0.1Nm/A.

$$\tau_{max} = 0.1 \times 5A = 0.5Nm$$

Maximizing Power through duty cycle

In advanced applications like haptic interfaces, using the motor's continuous power output can be too conservative. This is because in haptics the maximum torque output is rarely needed 100% of the time. If we need higher stall torque, we might have to go up to a motor with a higher wattage rating. The problem is these can be very heavy and bulky. In many haptics applications, this high power capacity is wasted 90% of the time. If we can guarantee that the peak torque output (i.e. peak current) will be present for less than 100% of the time, we can calculate a higher corresponding electrical power. If this power were applied to the motor continuously, it would burn out, but

transiently it will cool off between peaks. As a rule of thumb, if we can guarantee an *average* torque value is no more than 10% of the peak, then we can drive the motor at a peak power level 10× its continuous rating. The percentage of time that peak current is applied is called the "duty cycle" of the application.

So if the motor (and its resistance) is fixed, we need to use a higher voltage power supply to get a greater power:

$$\frac{V^2}{R} = 150 \times 10$$

$$R = 6\Omega$$

$$V = 95V$$

$$I_{max} = \frac{95V}{6\Omega} = 15.8A$$

$$\tau_{max} = 0.1 \times 15.8A = 1.58Nm$$

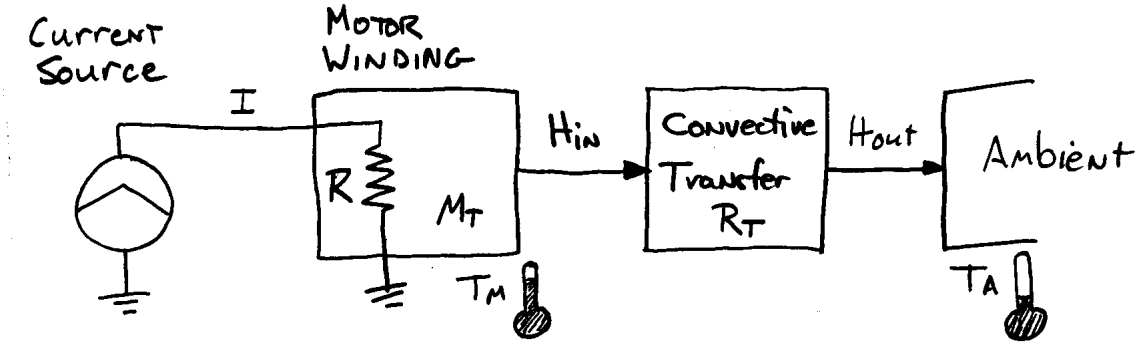
To summarize, by ASSUMING that the duty cycle is 10% or less, we can increase the supply voltage from 30V to 95V and get a peak torque which is 3.16 times higher. Since I_{max} increased from 5A to 15.8A, our power supply's peak output goes from $30V \times 5A = 150W$ to $95V \times 15.8A = 1500W$. So we tripled peak force output but our power supply cost (proportional to Watts) went up by a factor of 10!

We must be very careful when using this strategy because if we are wrong about the average duty cycle, we can burn an expensive motor.

The same duty cycle trick can be used to reduce the power supply size and cost. Capacitors can be used to provide the peak current output required for haptic peaks. Then the power supply rating can be reduced back to match the motor's steady state rating.

Thermal Analysis

In many applications, the ultimate limitation on DC motor performance is dissipation of heat coming from resistive losses in the motor coils. When the temperature gets too high (typically above 180°F) the insulation on the coil wires melts or burns and the coil shorts out. The block diagram illustrates the heat generation process from the input of electrical current, to heat generation in the motor winding (always equal to I^2R), to convective heat transfer out of the motor coils and into the environment.



Let us analyze this system:

In robotics we are most likely starting with some algorithm (usually a control law) which would like to command a certain torque to the motor. For a given motor, we need a current

$$I = \frac{\tau}{K_m}$$

Heat produced in the coils is always I^2R regardless of τ and ω.

$$H_{in} = P = I^2R$$

We can derive the steady state temperature as a function of H_{in}:

$$T_{in} = PR_T$$

where R_T is the thermal resistance of the motor (often listed in the catalog units: $^\circ/Watt$).

Heat flows through a thermal resistance in a manner analogous to current flowing through an electrical resistance. Temperature plays the role of voltage. Thus

$$H_{out} = \frac{T_M - T_A}{R_T}$$

where T_M is the motor internal temperature, T_A is the ambient temperature.

In addition to thermal resistance, objects have a "thermal mass" which determines how much thermal energy they can store, and also how fast temperature builds up when there is a net flow of heat into the object:

$$T_M = \frac{1}{M_T}\int_0^t (H_{in} - H_{out})dt + T_A$$

where M_T is the thermal mass, and we assume $T_M = T_A$ at $t = 0$.

Substituting:

$$T_M = \frac{1}{M_T}\int_0^t (P - \frac{T_M - T_A}{R_T})dt + T_A$$

Let $T = T_M - T_A$

$$T = \frac{1}{M_T}\int_0^t (P - \frac{T}{R_T})dt$$

Let $\hat{T} = PR_T = I^2 R R_T$. This is the equilibrium temperature if the current were held constant.

$$T = \frac{1}{R_T M_T}\int_0^t (\hat{T} - T)dt$$

Taking the LaPlace Transform:

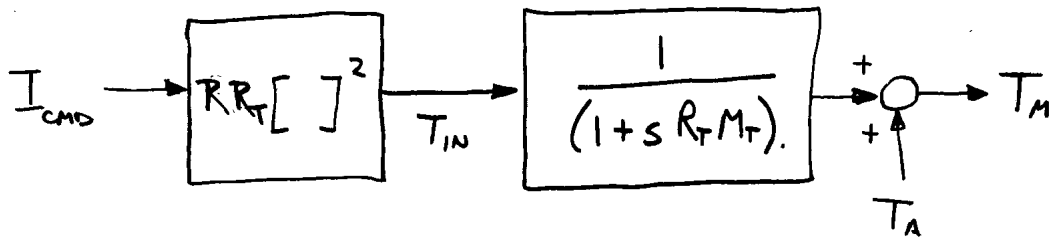
$$T(s) = \frac{1}{sR_T M_T}(\hat{T}(s) - T(s))$$

$$\frac{T(s)}{\hat{T}(s)} = \frac{1}{(1 + sR_T M_T)}$$

Or equivalently:

$$\frac{T(s)}{I^2(s)} = \frac{RR_T}{(1 + sR_T M_T)}$$

This transfer function describes a thermal system whos input is current squared (proportional to power) and whos output is motor temperature. This system is first order and the time constant is $R_T R_M$. Thus our model of motor winding temperature becomes:



Future Material:

[Use of this model to estimate motor winding temperature on-line. Use of winding temperature sensor to adapt model parameters.]

[Real time control of motor current based on measured or estimated winding temperature.]

[Brushed vs. Brushless Motors]

10.4 Transmissions

10.5 Joints

10.6 Links

10.7 End Effectors

10.8 Software Architectures

10.9 Summary of Notation

Appendix A

Mathematical Fundamentals

Affine Transformation An Affine Transformation is one of the form

$$y = Ax + b$$

Where x,y,b are N vectors, and A is an $N \times N$ matrix. Such transformations perserve straight lines and include an offset (b) so that they can represent displacement as well as other types of linear transformations.

A.1 Trigonometric Identities and Formulas

A.2 Euler Angle Sets

A.3 The Atan2 Function

Appendix B

Linear Algebra Basics

Source: "Introductino to Matrices and Determinants," F. Max Stein, 1967.

Assume A is a square matrix.

Facts:

1. A^{-1} exists if and only if
$$\det[A] \neq 0$$
if $\det[A] = 0$, we say that A is a *singular* matrix and that its inverse does not exist.
2. The *rank* of A is the size of the largest square sub-matrix, S, for which
$$\det[S] \neq 0$$
3. If two rows or columns of A are equal, or related by a constant, then $\det[A] = 0$.
4. Any singular matrix has at least one eigenvalue equal to zero.
5. If A is non-singular, and λ is an eigenvalue of A corresponding to the eigenvector, x, then
$$A^{-1}x = \lambda^{-1}x$$
6. If A is square, then A and A^T have the same eigenvalues. I.e.
$$Ax_i = \lambda_i x_i \qquad A^T y_i = \lambda_i y_i$$
7. If the nxn matrix A is of full rank (i.e. $r = n$) then the only solution to $Ax = 0$ is the trivial one, $x = 0$.
8. If $r < n$, then there are $n - r$ linearly independent (i.e. orthogonal) solutions
$$x_j \qquad 1 < j < n - r$$
for which $x_j \neq 0$ and $Ax_j = 0$.

B.1 Matrix Multiplication

Conceptual Definition:

A *matrix* is a mapping of a vector from one space to another which preserves straight lines. The matrix thus represents a linear transformation.

Algebraic Definition:
If A is a 3x3 matrix and B is a 3x1 vector,

$$AB = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_{11}b_1 + a12b_2 + a_{13}b_3 \\ a_{21}b_1 + a22b_3 + a_{23}b_3 \\ a_{31}b_1 + a32b_3 + a_{33}b_3 \end{bmatrix}$$

The rule is to multiply **row by column** and add component wise. This can also be written:

$$AB_i = \sum_{j=1}^{N} a_{ij}b_\beta$$

For matrix - matrix multiplication, this becomes

$$AC_{ij} = \sum_{m=1}^{N} a_{im} c_{mj}$$

where C is a 3x3 matrix with components c_{ij}.

(Did you know that Albert Einstein invented the idea of dropping the $\sum$ and the subscripts to simpilify the notation of matrix multiplication?) Putting it another way,

if

$$A = \begin{bmatrix} a^T \\ \alpha^T \\ \aleph^T \end{bmatrix}, \text{and} \qquad B = [b \quad \beta \quad \lambda],$$

then

$$AB = \begin{bmatrix} a \cdot b & a \cdot \beta & a \cdot \lambda \\ \alpha \cdot b & \alpha \cdot \beta & \alpha \cdot \lambda \\ \aleph \cdot b & \aleph \cdot \beta & \aleph \cdot \lambda \end{bmatrix}$$

where $\cdot$ is the vector dot product.

B.2 Matrix Inverse

The inverse of a matrix A, A^{-1} is most clearly defined by $AA^{-1} = I$.

We can find A^{-1} by

$$A^{-1} = \frac{adjA}{detA} = \begin{bmatrix} \frac{a_{11}}{|A|} & \frac{a_{12}}{|A|} & \frac{a_{13}}{|A|} \\ \cdots & & \frac{a_{ij}}{|A|} \end{bmatrix}$$

$adjA$ is a matrix of *cofactors* A_{ij} of A where

$$A_{ij} = (-1)^{n(j)} Det(A\text{-}(i,j))$$

where $A\text{-}(i,j)$ is the matrix A when row i and col j are removed and

$$|A| = Det(A) = \sum_{\{j\}} (-1)^{n(j)} a_{1j_1} a_{2j_2} a_{3j_3} \ldots a_{nj_n}$$

where $\{j\}$ is the set of all permutations of the integers $i \ldots n$, and $n(j)$ is the number of interchanges of the order of these permutations. (This detail is included simply for completeness of the review. For more information see your favorite Linear Algebra textbook.)

B.3 Interpretations of Matrix Vector Multiplication

The following are helpful ways to think about the meaning or application of matrix-vector multiplication. We will use a notation in which leading superscripts denote coordinate systems in which quantities are represented. For example the point x may be represented by different coordinates in two different coordinate systems 1 and 2. We denote these representations of x as ${}^1x, {}^2x$.

1. Stretched Sheet

   The matrix transformation ${}^2b = A^1b$ can be visualized as the transformation of point ${}^1b$ on a rubber sheet which is deformed to give a new point ${}^2b$. Suppose a rubber sheet has three points a, b, c printed on it and is streched as shown in Figure B.1:

   In this example, the stretching of the sheet can be described by the matrix A. Looking at the figure we note that the vector b is longer after the stretch and points in a slightly different direction. We also note that although a and c have had their lengths changed, they still point in the same direction. We can express this mathematically as

   $$^2a = \lambda_1 a = A^1 a$$

   and

   $$^2c = \lambda_2 c = A^1 c$$

   where λ_1 and λ_2 are scalar constants. In other words, for this particular example, for certain vectors (a, c), the linear transformation A is eqivalent to multiplication by a constant, but not for general vectors such as b. We call the scalars λ_1 and λ_2 the *eigenvalues* of A, and we call the vectors a and c the *eigenvectors* of A.

2. Rotation

   We will cover this in considerable detail in Chapter XXXXXXX.

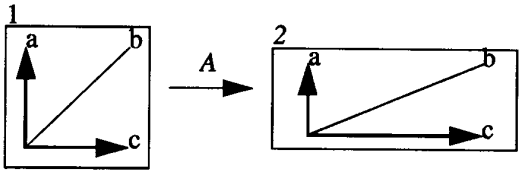


Figure B.1: Matrix multiplication can be used to represent linear transformations such as the stretching of a rubber sheet.

3. Magnification
   Many aspects of optics including image magnification and some distortions such as keystoning can be represented by matrix vector multiplication.
4. Perspective Transformation
   The mapping of 3D space to 2D such as is done in perspective drawings and camera imaging can be represented by matrices.

B.4 Inspiration

> The engineer generalizes the 1,2, or 3 dimensional problem of the physicist into k dimensions."

Gabriel Kron, "Tensor Analysis of Networks," 1938.