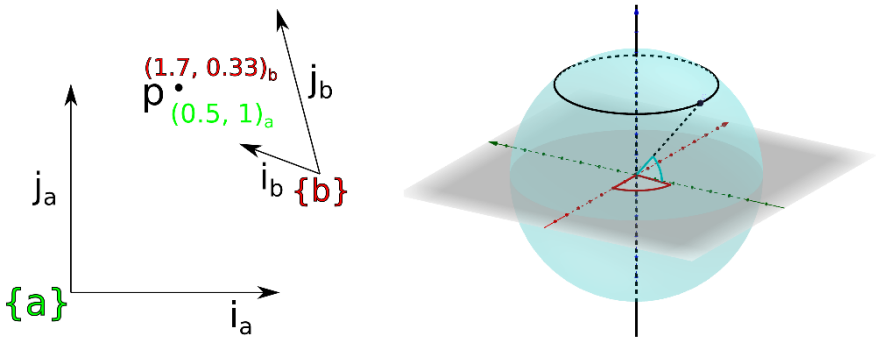
Module specification	**Explanation**		
Teacher Name			
Training Topic	Robotics application in Virtual Laboratory		
Training Code	UNIOULU_03 Kinematics		
Module Name	Kinematics		
Module duration	100 minutes		
Module objective	•Setup •Controlling robot via ROS •Kinematics •Sensor interaction		
Mode of provision	Classroom		
Laboratory structure	Time (min)	Objective	Performed by?
	5	Indrotuction	Teacher
	35	Kinematics and sensor interaction	Teacher
	60	Exercise with froward kinematics	Students

Introduction

Kinematics is a wide and deep topic whose surface can only be barely scratched by one tutorial. The purpose of this tutorial is to teach one modern way of doing forward position kinematics with serial manipulators with enough mathematical background to keep the tutorial as self-contained and readable as possible. The mathematical descriptions are quite loose, with a main purpose of giving some intuition about their "physical" meaning. This tutorial will not use the common Denavit-Hartenberg parameters to describe the manipulator, or Euler angles to describe orientations, but instead focuses on exponential coordinates, screws, and matrix representations. After the tutorial you should understand how to construct the following equation and how to use it to switch between sensor and global perspectives;

$$T(\theta) = e^{[S_1]\theta_1} e^{[S_2]\theta_2} ... e^{[S_n]\theta_n} M$$

Object vs. Representation



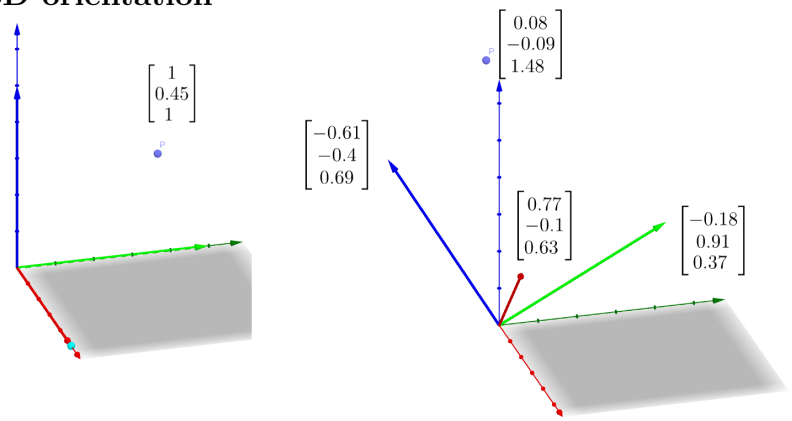
We often write points in 2D space as $p = (x, y)$. Instead of thinking (x,y) as the point itself, we can think of it as shorthand for $p = xi + yj$, i.e. combine x-units of vector i and y-units of vector j, stick the root of this composed arrow to a fixed starting location and the point will lie at the point p. The numbers x and y therefore are the *representation*, or *coordinates*, of the point p in the coordinate system that is determined by the origin and *basis vectors* i and j. Usually we choose basis vectors that are unit length, orthogonal to each other (in this context meaning they have a 90 degree angles to each other), and follow the right hand rule for cross-products. Such *orthonormal* basis is indeed very convenient in many cases, but nothing prevents us from using a different coordinate system (different starting location, different basis vectors), in which the coordinates of p (numbers x and y) would be different, but equally correct. Above picture is an example of this and point p can be reached equally by the two coordinate systems. The example should clarify the difference between a point, and the numbers we use to represent it.

Just like positions on 2D planes, locations on the surfaces of spheres (such as idealized planet earth), can be represented in different ways. Even though spheres are 3D objects embedded in 3D space, their surfaces are 2D objects embedded in 3D space, so just like positions in a 2D plane can be represented by a minimum of two numbers, the location on the surface of a sphere can also be represented by just two numbers, such as the latitude and longitude angles commonly used in navigation. However, the 2D surface of a sphere is *topologically* different to a 2D plane as a plane continues to infinity (you can walk forever in any direction), and the surface of a sphere wraps unto itself (you can only walk limited distance on a given direction before returning where you started). This causes some unfortunate issues when we try to represent each point on the surface of a sphere with just two numbers. For example, walking some time along the equator of Earth would change our longitude much less than walking the same time around a nearby geographical North Pole (NP), i.e. the rate how fast the two-parameter representation of our position changes is dependent on not only our actual walking speed, but also our location. For example, just by looking at the rate of change of longitude we couldn't tell if we were actually walking very fast or just near the NP. The longitude changes faster and faster the closer we circle NP, and at the exact position of NP this latitude-longitude representation "breaks down" as any value of longitude is equally valid there. These breakdowns are more formally called *representation singularities* and they are troublesome for computer applications because additional code is needed to handle them (analogous how we want to avoid divisions by zero) as even a "proximity" to them causes issues with numerical accuracy. Furthermore, this two-parameter representation clearly does not match our intuition as our walking speed does not in reality dramatically increase the closer we are to the NP, and the world does not break if we reach it.

The examples above are *explicit parameterizations'* that use as few numbers as possible. An alternative is to use *implicit parameterizations'* which use more numbers than strictly necessary, but also give constraints to them. For example we could use three numbers instead of two to represent a point a spheres surface, but also give them the following constraint with fixed radius r; $x^2+y^2+z^2 = r^2$. We still only have 2 *Degrees of Freedom* (DOF), because even though we now have three numbers instead of two, we can choose only two of them "freely". In general DOF $= \#numbers - \#constraints$. In some cases this not true, but those cases are not discussed in this tutorial. Implicit parameterizations need more memory than explicit parameterizations, and are sometimes arguably less intuitive, but they generally behave "nicely" everywhere. For example, the above implicit parametrization does not have any representation singularities and the overall rate that the number triplet (x, y, z) changes is only dependent on your actual physical velocity, and not on location as was the case with latitude-longitude parameterization. This tutorial uses implicit parameterizations.

The key points are:
- We can represent things in multiple ways.
- A number is just a number and needs extra rules to give it a meaning.
- We choose those rules.
- Some rules are more convenient than others (e.g. they match our intuition or have other nice properties).

3D orientation



As with all things, the orientation of a frame floating in a 3D space can represented in many ways, the most popular being Euler Angles, Rotation matrices, and quaternions. Euler angles are a form of explicit parameterization and thus use a minimal amount of three numbers (usually roll, pitch, yaw). Euler angles suffer from similar issues as latitude-longitude of the previous example, with "gimbal lock" being the most famous problem. As this tutorial uses implicit parameterizations, Euler angles are not discussed further. Quaternions are 4 number implicit parameterizations with nice mathematical properties, but are also not discussed in this tutorial.

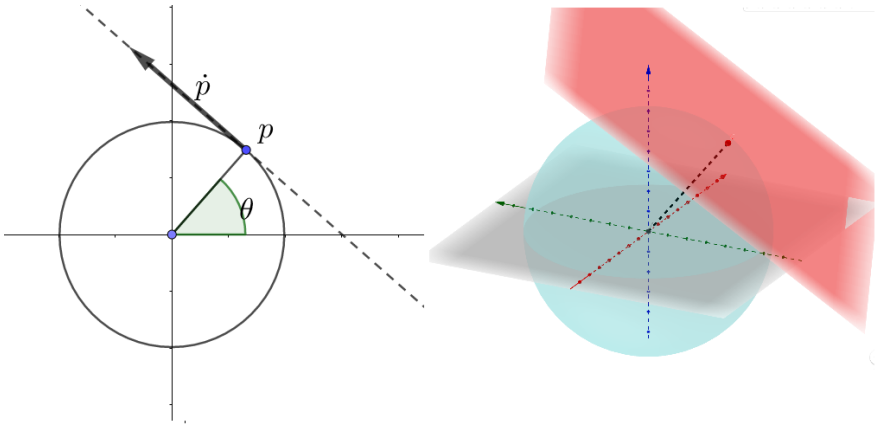
3x3 rotation matrices R are implicit representations of 3D orientation. Even as R has nine elements, it only has three degrees of freedom as the elements are subjected to six constraints. Each column must be unit length (3 constraints), and orthogonal to each other (another 3 constraints). These constraints are succinctly expressed as the property $R^{-1} = R^T$. This is a significant property as matrix inversions are usually costly operations, and transposing a matrix is cheap. Rotation matrices have no representation singularities and their columns have clear geometric meanings. Each column of R describes where the tips basis vector would end up after the rotation that R represents. The first column tells where the tip of $\begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ gets rotated to, the second column where $\begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$ and the third column where $\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$.

R can be viewed as a representation of a *linear transformation* that somehow rotates everything in space around some unchanging frame of reference. As previously discussed, the coordinates x, y, z, of point p just tell how many units of basis vectors are needed to reach the specific location in space. For example if the space, and thus also the basis vectors of some frame of reference initially co-incident with the unchanging frame, has been modified by a linear transformation represented as $R = \begin{bmatrix} 0.77 & -0.18 & -0.61 \\ -0.1 & 0.91 & -0.4 \\ 0.63 & 0.37 & 0.69 \end{bmatrix}$, the new location of $p = (1, 0.45, 1)$, as viewed in the unchanged reference frame, is then $p' = 1 \begin{bmatrix} 0.77 \\ -0.1 \\ 0.63 \end{bmatrix} + 0.45 \begin{bmatrix} -0.18 \\ -0.91 \\ 0.37 \end{bmatrix} + 1 \begin{bmatrix} 0.77 \\ -0.1 \\ 0.63 \end{bmatrix} = \begin{bmatrix} 0.08 \\ -0.09 \\ 1.48 \end{bmatrix}$

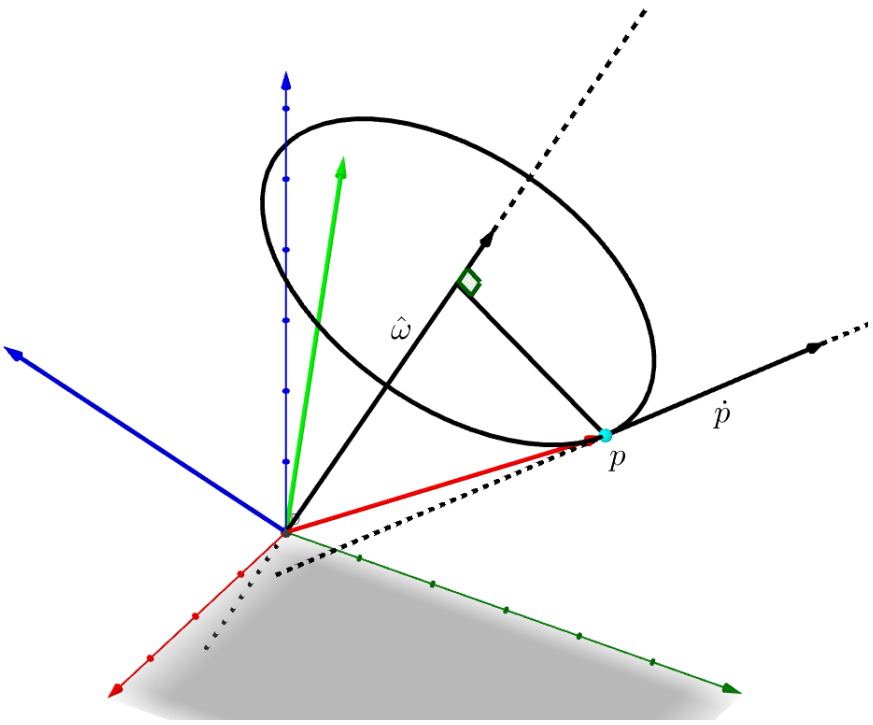
3D rotational velocity

Any orientation can be achieved by picking a rotation axis and turning around it for a specific amount. As the orientations of physical objects can smoothly change, so should the values of our representation R change smoothly. As rotating an object around a fixed axis way long enough returns it to its original orientation, so should the values of our representations "wrap around".
Given these two conditions it would initially make sense to imagine all possible orientations as somehow "living" on a surface of a sphere (imagine dotting the whole surface of a ball with a pen end then giving each dot writing an unique numerical description R). Each point on the sphere would be a specific orientation that can be uniquely represented with R. Moving from a starting orientation to a final location would be like drawing a path with a finger so that the orientation, and its representation R smoothly changing without any gaps (if we had dotted the ball with infinitely small intervals). Picking a specific direction and sticking to it would eventually cause the finger to travel around the ball and return to the starting location.

The above analogy works well enough, but it has a major flaw. The problem is that as the surface of a 3D sphere we can touch is actually 2D, and thus each point would give two independent values, no three as required. The solution is therefore to consider the orientations as "living" on a 3D surface of a 4D sphere. We can't imagine a 4D sphere visually, but nevertheless the mathematics work.



Just like any orientation can be achieved by picking a rotation axis and turning around it for a specific amount, so can any instantaneous rotational velocity be achieved by picking a rotation axis and rotating around it with specific velocity. The instantaneous velocity of a point constrained to a ring (a 1D sphere) is constrained to the tangential vector, and the instantaneous velocity of a point constrained to a 2D sphere is constrained to a tangential 2D plane. Similarly the velocity on the surface of a 3D sphere (which we can't imagine) is constrained to a 3D (hyper)plane. The 3D hyperplane is just the Euclidean space we live in, so we can visualize it. Each point in this 3D space, that is somehow tangential to a 4D object and has a origin at the contact point, represents a specific unique rotational velocity. Let's call the coordinate a specific point as $\omega = \begin{bmatrix}\omega_1 & \omega_2 & \omega_3\end{bmatrix}^T$. Note that even though orientations have a limited range of values (because they "live" on the surface of a 4D sphere), ω has no such restriction (as the point it represents lives in 3D space that does not wrap back to itself). You can imagine the rotational velocity represented by ω as a combination of static axis of rotation and some angular velocity around it, i.e a specific rotational velocity is $\omega = \dot{\theta}\hat{\omega}$ where $\dot{\theta}$ is the rotational velocity and $\hat{\omega}$ is a unit length axis of rotation.



Frames are composed of three orthonormal vectors. The rate of change of one vector tip p is given by the cross product between the rotation axis and vector pointing to p that is scaled by the rotational velocity

$$\dot{p} = \dot{\theta}(\hat{\omega} \times p)$$

We can express the cross product operation as a matrix multiplication $\hat{\omega} \times p = [\hat{\omega}]p$ where

$$[\hat{\omega}] = \begin{bmatrix} 0 & -\hat{\omega}_3 & \hat{\omega}_2 \\ \hat{\omega}_3 & 0 & -\hat{\omega}_1 \\ -\hat{\omega}_2 & \hat{\omega}_1 & 0 \end{bmatrix}$$

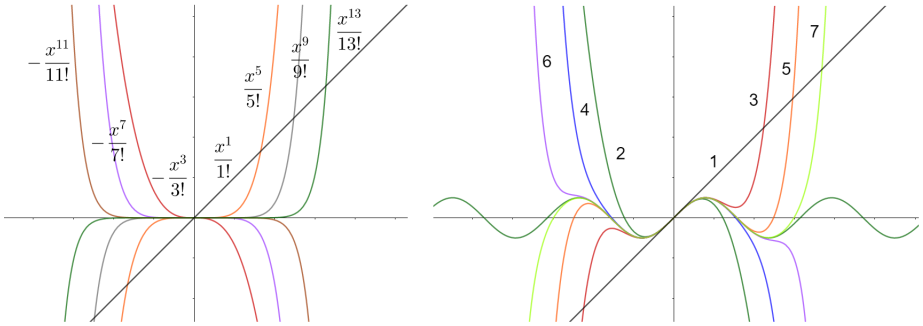
If we now collect all three vectors to a single matrix R we have the following equation for the rate of change of in the elements of R

$$\dot{R} = \dot{\theta}[\hat{\omega}]R$$

Note $\dot{R}$ is now dependent also on the current value of R, but unlike the latitude-longitude case, is not scaled by it. Also note that on the right side of the equation

all the values have clear physical meaning of angluar velocity, rotation axis and a combination of current locations of frame tips.

Exponential coordinates for rotations



Functions can be written as an infinite sum of its derivatives evaluated at some fixed point b. This infinite sum is known as *Taylor Series* of the function.

$$f(x) = f(b) + \frac{f'(b)}{1!}(x-b) + \frac{f''(b)}{2!}(x-b)^2 + + \frac{f'''(b)}{3!}(x-b)^3 + ...$$

A classic example of Taylor series is $sin\theta$ evaluated at zero.It is

$$\sin\theta = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - ...$$

Above and left is a picture of the first 7 elements of the sum, on the right are the sum of 1,2,3,5,6, and 7 first elements and actual sine wave. Note that as the new elements are added the closer the sum is to the sine wave locally. Infinite sums do not sound very practical, but in many useful cases these seemingly infinite sums can be condensed to short closed form solutions by noticing some well known repeating pattern, such as the sine Taylor series are the cosine Taylor series

$$\cos\theta = 1 - \frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - ...$$

The solution to first order differential equation $x \in \mathbb{R} : \dot{x} = ax(t) \Rightarrow x(t) = e^{at}x(0)$ is in expressed in Taylor series as

$$f(t) = e^{at}, f'(t) = ae^{at}, f''(t) = a^2e^{at}, f'''(t) = a^3e^{at}, ...$$
$$e^{at} = e^{a0} + \frac{ae^{a0}}{1!}(t-0) + \frac{a^2e^{a0}}{2!}(t-0)^2 + \frac{a^3e^{a0}}{3!}(t-0)^3 + ...$$
$$e^{at} = 1 + at + \frac{(at)^2}{2!} + \frac{(at)^3}{3!} + ...$$

The above is valid even when the constant is a matrix instead of a scalar.

$$x \in \mathbb{R}^n : \dot{x} = Ax(t) \Rightarrow x(t) = e^{At}x(0)$$
$$e^{At} = I + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + ...$$

We can therefore find solution for

$$x \in \mathbb{R}^n : \dot{R} = \dot{\theta}[\hat{\omega}]R(t) \Rightarrow R(t) = e^{\dot{\theta}[\hat{\omega}]t}R(0)$$

where $R(0)$ is the initial orientation and $R(t)$ the orientation after rotating with $\hat{\omega}$ angular velocity around unit axis of direction $\hat{\omega}$ for t seconds. By setting $\dot{\theta} = 1$ we can write $R(\theta) = e^{[\hat{\omega}]\theta}R(0)$ which we can view as a function for final orientation after rotating around $\hat{\omega}$ for θ radians. $e^{[\hat{\omega}]\theta}$ is therefore a rotation matrix R which rotates the initial configuration $R(0)$ to final configuration $R(\theta)$. Solving $e^{[\hat{\omega}]\theta}$ via Taylor series gives

$$\begin{aligned}
e^{[\hat{\omega}]\theta} &= I + [\hat{\omega}]\theta + \frac{([\hat{\omega}]\theta)^2}{2!} + \frac{([\hat{\omega}]\theta)^3}{3!} + ... \\
&= I + [\hat{\omega}]\theta + [\hat{\omega}]^2\frac{\theta^2}{2!} + [\hat{\omega}]^3\frac{\theta^3}{3!} + ... \\
&= I + (\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - ...)[\hat{\omega}] + (\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - ...)[\hat{\omega}]^2 \\
&= I + \sin\theta[\hat{\omega}] + (1 - \cos\theta)[\hat{\omega}]^2 \\
&= R
\end{aligned}$$

The key to expressing an infinite series a short closes form solution by first noticing that there are actually only four different matrices that get repeated, namely $-[\hat{\omega}], [\hat{\omega}], -[\hat{\omega}]^2$, and $[\hat{\omega}]^2$:

$$\hat{\omega} = [\hat{\omega}_1, \hat{\omega}_2, \hat{\omega}_3]^T \in \mathbb{R}^3, ||\hat{\omega}|| = 1, [\hat{\omega}] = \begin{bmatrix} 0 & -\hat{\omega}_3 & \hat{\omega}_2 \\ \hat{\omega}_3 & 0 & -\hat{\omega}_1 \\ -\hat{\omega}_2 & \hat{\omega}_1 & 0 \end{bmatrix}$$

$$[\hat{\omega}]^2 = \begin{bmatrix} -\hat{\omega}_2^2 - \hat{\omega}_3^2 & \hat{\omega}_1\hat{\omega}_2 & \hat{\omega}_1\hat{\omega}_3 \\ \hat{\omega}_1\hat{\omega}_2 & -\hat{\omega}_1^2 - \hat{\omega}_3^2 & \hat{\omega}_2\hat{\omega}_3 \\ \hat{\omega}_1\hat{\omega}_3 & \hat{\omega}_2\hat{\omega}_3 & -\hat{\omega}_1^2 - \hat{\omega}_2^2 \end{bmatrix}$$

$$[\hat{\omega}]^3 = \begin{bmatrix} 0 & \hat{\omega}_3(\hat{\omega}_1^2 + \hat{\omega}_2^2 + \hat{\omega}_3^2) & -\hat{\omega}_2(\hat{\omega}_1^2 + \hat{\omega}_2^2 + \hat{\omega}_3^2) \\ -\hat{\omega}_3(\hat{\omega}_1^2 + \hat{\omega}_2^2 + \hat{\omega}_3^2) & 0 & \hat{\omega}_1(\hat{\omega}_1^2 + \hat{\omega}_2^2 + \hat{\omega}_3^2) \\ \hat{\omega}_2(\hat{\omega}_1^2 + \hat{\omega}_2^2 + \hat{\omega}_3^2) & -\hat{\omega}_1(\hat{\omega}_1^2 + \hat{\omega}_2^2 + \hat{\omega}_3^2) & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \hat{\omega}_3 & -\hat{\omega}_2 \\ -\hat{\omega}_3 & 0 & \hat{\omega}_1 \\ \hat{\omega}_2 & -\hat{\omega}_1 & 0 \end{bmatrix} = -[\hat{\omega}]$$

$$\begin{aligned} [\hat{\omega}]^4 &= [\hat{\omega}]^3[\hat{\omega}] = -[\hat{\omega}]^2 \\ [\hat{\omega}]^5 &= [\hat{\omega}]^3[\hat{\omega}]^2 = -[\hat{\omega}][\hat{\omega}]^2 = -[\hat{\omega}]^3 = [\hat{\omega}] \\ [\hat{\omega}]^6 &= [\hat{\omega}]^3[\hat{\omega}]^3 = -[\hat{\omega}](-[\hat{\omega}]) = [\hat{\omega}]^2 \\ &... \end{aligned}$$

So the key discovery is that

$$[\hat{\omega}]^3 = -[\hat{\omega}].$$

Then, after re-arrigning the the sum as

$$I + (\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - ...)[\hat{\omega}] + (\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - ...)[\hat{\omega}]^2$$

we notice that $(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - ...)$ is simply $sin\theta$ and $(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - ...)$ is simply $(1 - cos\theta)$. We now have a simple solution for calculating rotation matrix R from the given product unit length rotation axis and rotation angle $\theta\hat{\omega}$:

$$R = I + \sin\theta[\hat{\omega}] + (1 - \cos\theta)[\hat{\omega}]^2$$

This a is called the *Rodrigues' formula*. We can view $\hat{\omega}$ either as a static rotation axis around which we rotate θ degrees, or view $\hat{\omega}$ is an active motor that rotates things around at a rotational velocity $1\frac{rad}{s}$ for θ seconds. Both views are equally valid and the viewpoint can be chosen based on the problem at hand. We can construct R by exponentiating the product $\theta\hat{\omega}$ (the representation of angular velocity) and thus call $\theta\hat{\omega}$ the *exponential coordinates for rotation*. Note that the product $\theta\hat{\omega}$ can be viewed as being constructed from four numbers and one constraint $||\hat{\omega}|| = 1$. Writing θ and $\hat{\omega}$ separately would be the *axis-angle* representation. There is also an algorithm for finding $\theta\hat{\omega}$ for given R (the "matrix logarithm" of the rotation R) but it will not be discussed in this tutorial.

3D pose representation

For general 6D description of an object's pose (3 paramaters for location, 3 for orientation), a 4x4 *homogeneous transformation matrix* T is often used:

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$$

where R is the 3x3 rotation matrix and p the 3x1 location vector. When T is viewed as linear transformation the, it first rotates the space with R and then translates everything according to p.

$$T = Trans(p)Rot([\hat{\omega}], \theta)$$

The inverse of T is,

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

If a space is first transformed by T, then transforming with T^{-1} would return the space back to its original state, and thus returning all points and frames to their original positions and orientations, i.e

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

Twist, general motion in 3D

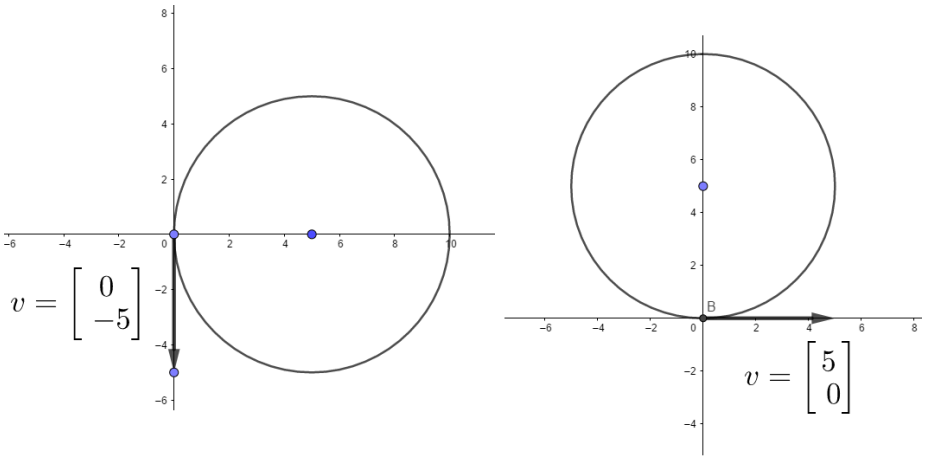
Any general instantaneous velocity can be viewed as rotation about an axis with simultaneous translation along it, i.e a screwing motion. Think of as placing the rotation axis anywhere in space (with pure rotation the axis could go only through the origin of the unchanging frame). The axis then rotates everything around it while at the same time sliding along it with the ratio between rotation and sliding given by the *pitch*. Previously we derived

$$\dot{R} = \dot{\theta}[\hat{\omega}]R \Rightarrow \dot{R}R^{-1} = \dot{\theta}[\hat{\omega}] = [\omega]$$

for orientations, with $\dot{\theta}[\hat{\omega}]$ having very clear physical meaning of angular velocity and rotation axis. You could imagine that the right side $\dot{\theta}[\hat{\omega}]$ is therefore somehow the "actual" ongoing rotation, and the left side $\dot{R}R^{-1}$ is more involved about our chosen method of representing orientation. It just happens so that similar relationship applies also to general motion

$$\dot{T}T^{-1} = \begin{bmatrix} \dot{R} & \dot{p} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R^T & -R^T p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \dot{R}R^T & \dot{p} - \dot{R}R^T p \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} = [\mathcal{V}]$$

with the right side somehow being the "actual" general motion. $[\omega] = \dot{\theta}[\hat{\omega}]$ is the familiar representation of total rotational velocity and v the velocity of body-fixed points that go through the origin. v therefore embeds onto itself both the translation along the axis, and the location of the axis (the farther away the axis is, the larger v would be with given $\dot{\theta}$. For example, with $\omega = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ the value of v would change if we moved the rotation axis from a location where it pierces the xy-plane at [5,0] to a location where the piercing happens at [0,5].



More formally $v = \dot{p} - \omega \times p$. Imagine that point p is fixed to a turntable that is turned by the rotation axis. In the image above p was at the rotation axis and thus $\dot{p}$ was zero. We can combine all the relevant information of $[\mathcal{V}]$ to a single 6-element *twist* vector $\mathcal{V} = \begin{bmatrix} \omega \\ v \end{bmatrix} = \begin{bmatrix} \hat{\omega}\dot{\theta} \\ h\hat{\omega}\dot{\theta} - \hat{\omega}\dot{\theta} \times p \end{bmatrix}$ where h is the pitch and p some point on the rotation axis. In the 2D the example above the pitch was zero. If it had not been non-zero then the point at the origin would have move in a helical motion away from the plane, i.e having addtional velocity in the z-direction. For the rest of the tutorial we assume that the pitch is always zero. The twist $\mathcal{V}$ can expressed as $\mathcal{V} = \dot{\theta}\mathcal{S}$ where $\mathcal{S}$ is the *screw* and it is analogous to the unit length rotation axis $\hat{\omega}$. Just like the unit rotation axis has an alternative from $[\hat{\omega}]$, so does the screw with $[\mathcal{S}] = \begin{bmatrix} [\hat{\omega}] & h\hat{\omega} - \hat{\omega} \times p \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} [\hat{\omega}] & v' \\ 0 & 0 \end{bmatrix}$.

Exponential coordinates for poses

Just like we could use the exponential coordinates $[\hat{\omega}]\theta$ to construct rotation matrix R, we can construct general pose matrix T from $[\mathcal{S}]\theta$. The basic principle is the same and θ is still the rotation in radians. The only difference is that with $[\mathcal{S}]$ the rotation axis does not need to go through the origin and that there is the translation caused by sliding along the axis.

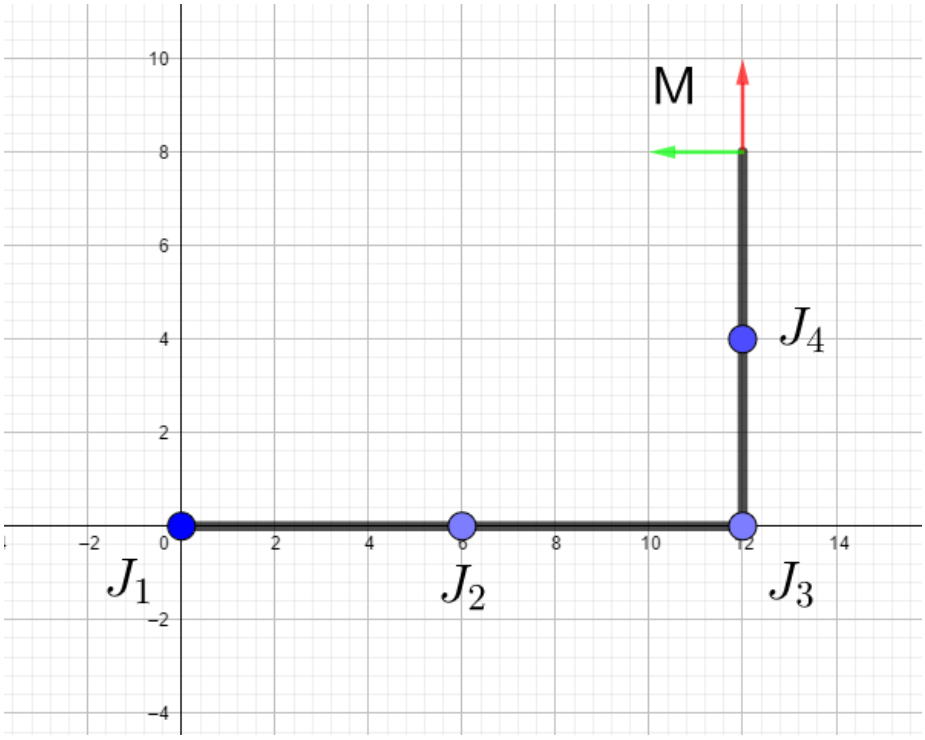
$$\begin{aligned} T = e^{[\mathcal{S}]\theta} &= I + [\mathcal{S}]\theta + [\mathcal{S}]^2\frac{\theta^2}{2!} + [\mathcal{S}]^3\frac{\theta^3}{3!} + ... \\ &= \begin{bmatrix} e^{[\hat{\omega}]\theta} & G(\theta)v' \\ 0 & 1 \end{bmatrix} \end{aligned}$$

where

$$
\begin{aligned}
G(\theta) &= I\theta + [\hat{\omega}]^2\frac{\theta^2}{2!} + [\hat{\omega}]^3\frac{\theta^3}{3!} + ... \\
&= I\theta + (\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - ...)[\hat{\omega}] + (\frac{\theta^3}{3!} - \frac{\theta^5}{5!} + \frac{\theta^7}{7!} - ...)[\hat{\omega}]^2 \\
&= I\theta + (1 - cos\theta)[\hat{\omega}] + (\theta - sin\theta)[\hat{\omega}]^2
\end{aligned}
$$

Basically, if we know how to construct $[\mathcal{S}]$ from a technical drawing, then the above equation is basically everything we need to calculate the forward kinematics of serial manipulators.

Forward kinematics algorithm



Given the above (crude) technical drawing we can by visual inspection determine that the tip's home pose is

$$
M = \begin{bmatrix} 0 & -1 & 0 & 12 \\ 1 & 0 & 0 & 8 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

and the four joints are located at $[0,0,0]^T, [6,0,0]^T, [12,0,0]^T$ and $[12,4,0]^T$ (so we know p for each joint), and that each joint is rotational with joint axis $\hat{\omega} = [0,0,1]^T$. As the joints are rotational their pitch is zero. This is all the information we need to construct the forward kinematics equation that calculates the tip's pose with given joint values $\theta_1, \theta_2, \theta_3$ and θ_4. We combine these four values as vector θ. We can view the effects of each manipulator as a linear transformation $T_n = e^{[S_n]\theta_n}$

$$\begin{aligned} T(\theta) &= T_1 T_2 T_3 T_4 M \\ &= e^{[S_1]\theta_1} e^{[S_2]\theta_2} e^{[S_3]\theta_3} e^{[S_4]\theta_4} M \end{aligned}$$

At home position all angles are zero and thus

$$\begin{aligned} T(0) &= IIIIM \\ &= M \end{aligned}$$

If only the third joint has been turned then

$$\begin{aligned} T(\theta) &= II e^{[S_3]\theta_3} IM \\ &= e^{[S_3]\theta_3} M \end{aligned}$$

and similarly for other joints. We can imagine going through the joints from tip to base. The joint closest to the tip changes the pose of M, then the second-to-last joint changes this modified pose, and so on until the first joint at the base. The geneal forward kinematics formula for n joints is

$$T(\theta) = e^{[S_1]\theta_1} e^{[S_2]\theta_2} ... e^{[S_n]\theta_n} M$$

The above equation, the equation for $e^{[\mathcal{S}]\theta}$, and how to visually inspect a technical drawing for $[\mathcal{S}] = \begin{bmatrix} [\hat{\omega}] & h\hat{\omega} - \hat{\omega} \times p \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} [\hat{\omega}] & v' \\ 0 & 0 \end{bmatrix}$, is evertyhing we need to calculate the forward kinematics of serial manipulator.

Sensor Interaction

If we set the tip frame as the frame of some sensor and measure the pose of some object of interest in that frame, then we can calculate the pose of that point in some other frame of reference by using the forward kinematics code

$$M_{pose-in-base-frame} = T(\theta) M_{pose-in-sensor-frame}$$

Further Reading

This tutorial skipped and condensed lot of relevant material and is not mathematically rigorous. For those who want deeper understanding of the material

a good foundation of linear algebra is crucial. The book "Modern Robotics: Mechanics, Planning, and Control" (and the associated videos) by Kevin M. Lynch and Frank C. Park offers a good and far more extensive introduction to exponential coordinates and screws. Even more exhaustive and mathematically rigorous explanations for the material of this tutorial can be found in the literature of Lie Groups and Lie Algebras. Numerical optimization methods are also important as they are often used in calculating the inverse kinematics.

Exercise

Setup
Open the simulator scene, code template, and technical drawing.

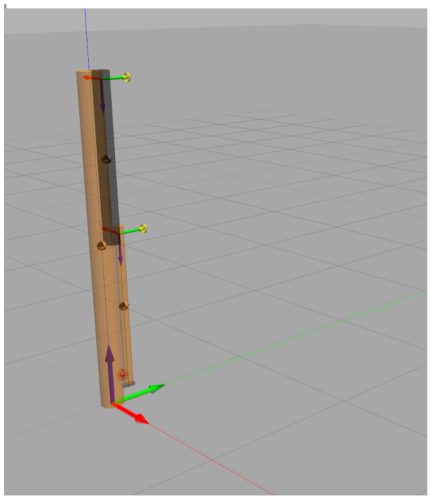
Task

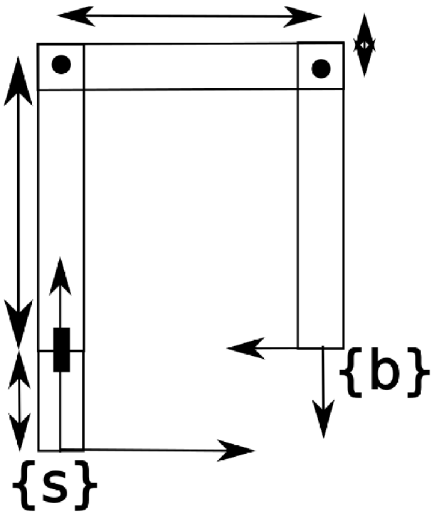
First task

Students are give a technical drawing of an arm, a simulation scene where this arm is already constructed, and a code template where the communication with the simulator is already implemented. The students' task is to look at the technical drawing, and write the forward kinematics code (in python) for it. In the code template there are hints how to perform relevant mathematical operations. A random number generator that gives values to joints. The students' implementation must take these numbers, calculate the forward kinematics, set a frame to that pose, "release" the arm and see if the tip goes to that same pose.

Second Task

The sensor a the tip of the serial manipulator locates a point of interest in its own frame of reference. Use the the previously developed forward kinematics algorithm to calculate where the point is in another frame of reference.




{b}
{s}