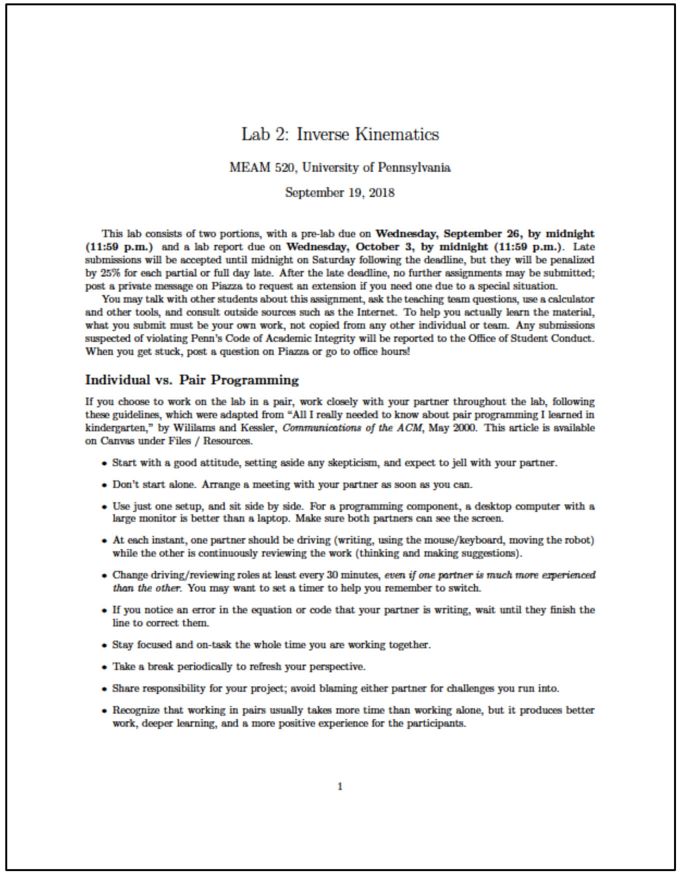
MEAM 520
Lecture 9: Quaternions

Cynthia Sung, Ph.D.
Mechanical Engineering & Applied Mechanics
University of Pennsylvania



Lab 2: Inverse Kinematics

MEAM 520, University of Pennsylvania

September 19, 2018

This lab consists of two portions, with a pre-lab due on **Wednesday, September 26, by midnight (11:59 p.m.)** and a lab report due on **Wednesday, October 3, by midnight (11:59 p.m.)**. Late submissions will be accepted until midnight on Saturday following the deadline, but they will be penalized by 25% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Piazza to request an extension if you need one due to a special situation.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you submit must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. When you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

If you choose to work on the lab in a pair, work closely with your partner throughout the lab, following these guidelines, which were adapted from "All I really needed to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000. This article is available on Canvas under Files / Resources.

- Start with a good attitude, setting aside any skepticism, and expect to jell with your partner.
- Don't start alone. Arrange a meeting with your partner as soon as you can.
- Use just one setup, and sit side by side. For a programming component, a desktop computer with a large monitor is better than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (writing, using the mouse/keyboard, moving the robot) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every 30 minutes, *even if one partner is much more experienced than the other.* You may want to set a timer to help you remember to switch.
- If you notice an error in the equation or code that your partner is writing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.
- Recognize that working in pairs usually takes more time than working alone, but it produces better work, deeper learning, and a more positive experience for the participants.

1

Lab 2 posted

Prelab due tomorrow 9/26, 11:59 p.m.

Funda, J. and Paul, R.P. "A comparison of transforms and quaternions in robotics." Proceedings of the IEEE Conference on Robotics and Automation (ICRA). 1988. pp 886-891. doi: 10.1109/ROBOT.1988.12172

A Comparison of Transforms and Quaternions in Robotics

Janez Funda   Richard P. Paul

University of Pennsylvania, Philadelphia

Abstract

1 Introduction

2 The Quaternion

Funda, J and Paul, RP. “A comparison of transforms and quaternions in robotics.” ICRA 1988.

Quick Primer on Paper Reading

Pass 1

A Comparison of Transforms and Quaternions in Robotics

What is the paper about (high level)?

Janez Funda
grad student
(CIS 1991)

Richard P. Paul
advisor

University of Pennsylvania, Philadelphia

Who wrote it?

What year was the paper published?

What are the major claims?

Methods?

Evaluation?

Conclusion?

Abstract

Three-dimensional (3-D) modeling of rotations and translations in robot kinematics is most commonly performed using homogeneous transforms. In this paper, an alternate approach, employing quaternion/vector pairs as spatial operators, is discussed and analyzed. Sequential and parallel algorithms for some of the common spatial operations are given for both homogeneous transforms and quaternion/vector pairs and compared in terms of computational efficiency. The two approaches are shown to be essentially equivalent in the absence of the need for frequent renormalization of rotational operators.

1 Introduction

A variety of different mathematical tools for expressing spatial relationships have been developed and successfully applied in robotics, computer vision, graphics, and other engineering disciplines. Most notable among the approaches to this problem is the *vectorial technique*. Although very elegant in expressing translational information, vector calculus does not lend itself naturally to representing 3-D rotations. In this system, spatial transformations have traditionally been expressed as a set of four vectors, arranged in the form of a 4×4 matrix termed a *homogeneous transform*.[9]

This paper discusses an alternate mathematical model of spatial transformations, where 3-D rotations are represented via *quaternions*

However, Hamilton soon observed that within the scope of 3-D geometry quaternions also arise as products of vectors, powers of vectors, and sums of scalars and vectors. [3][4]

For instance, multiplication of two 3-D vectors using Eq.(2) as the defining relationship between the imaginary units, gives [8]

$$\vec{v}_1 \bullet \vec{v}_2 = -(\vec{v}_1 \cdot \vec{v}_2) + (\vec{v}_1 \times \vec{v}_2) \tag{3}$$

which is of the form $s + \vec{v}$, where $s \in \mathbf{R}$ and $\vec{v} \in \mathbf{R} \times \mathbf{R} \times \mathbf{R}$ (*i.e.*, $\vec{v}$ is a vector), and thus constitutes a quaternion as defined by Eq.(1). For our purposes, it will be convenient to treat a quaternion as a sum of a scalar (s) and a 3-D vector ($\langle x, y, z \rangle$), which also seems to be the most natural interpretation of Eq.(1). Observe that vectors can thus be represented as quaternions with null scalar parts. We will abbreviate the notation of Eq.(1) as follows

$$q = [s, \langle x, y, z \rangle] = [s, \vec{v}] \tag{4}$$

Two quaternions $q_1 = [s_1, \vec{v}_1]$ and $q_2 = [s_2, \vec{v}_2]$ are added simply by adding their components, *i.e.*,

$$q = q_1 + q_2 = [s_1 + s_2, \vec{v}_1 + \vec{v}_2] \tag{5}$$

Employing the composition rule of Eq.(2), along with the result of Eq.(3), we find that the general multiplication rule for quaternions is

Quick Primer on Paper Reading

Pass 1

$$\bar{q} = [s, -\vec{v}] \quad (7)$$

$$N(q) = \|q\| = \sqrt{q * \bar{q}} = \sqrt{s^2 + |\vec{v}|^2} \quad (8)$$

$$q^{-1} = \frac{1}{q} = \frac{1}{q} * \frac{\bar{q}}{\bar{q}} = \frac{\bar{q}}{N(q)^2} \quad (9)$$

Necessary math background?

Inverse Kinematics for Puma 560				
Component	Cost			
	$*$	$+$	$\sqrt{}$	trig
θ_1	2	2	1	2
θ_2	5	4	1	4
θ_3	2	4	0	4
$[s, (x, y, z)]$	16	8	0	4
θ_4	0	1	0	2
θ_5	4	2	2	1
θ_6	0	1	0	0
Total	29	22	4	17

Quality and format of data?

6 Discussion

Major takeaways?

The main purpose of this paper was to compare the computational efficiency of some of the common spatial operations using homogeneous transforms on the one hand, and quaternion/vector pairs on the other. The table of Section 4.4 summarizes the main results, indicating that the two approaches are virtually equivalent for the case of *non-normalizing sequential* procedures, but also that the *non-normalizing* vectorial algorithms parallelize slightly better than their algebraic counterparts. If the cost of normalizing the rotational operator is included in the total cost, however, the quaternion/vector approach yields much more efficient implementations on both single and multi-processor systems. Clearly, the non-normalizing procedures (where the rotational operators are never normalized) and their normalized versions as defined above (where the rotational operators are normalized at every call) correspond to the two extremes. In practice, the need for normalization will be intermediate to the above two cases, depending mostly on the presence of chains of products, which are likely to denormalize rotational operators. Note, however, that the cost of testing for whether or not a rotational matrix needs renormalization (Eq.(32)) accounts for approximately $\frac{2}{3}$ of the cost of the normalizing process itself. Consequently, it is uncertain if normalizing homogeneous transforms "by need" will result in a more efficient overall performance.

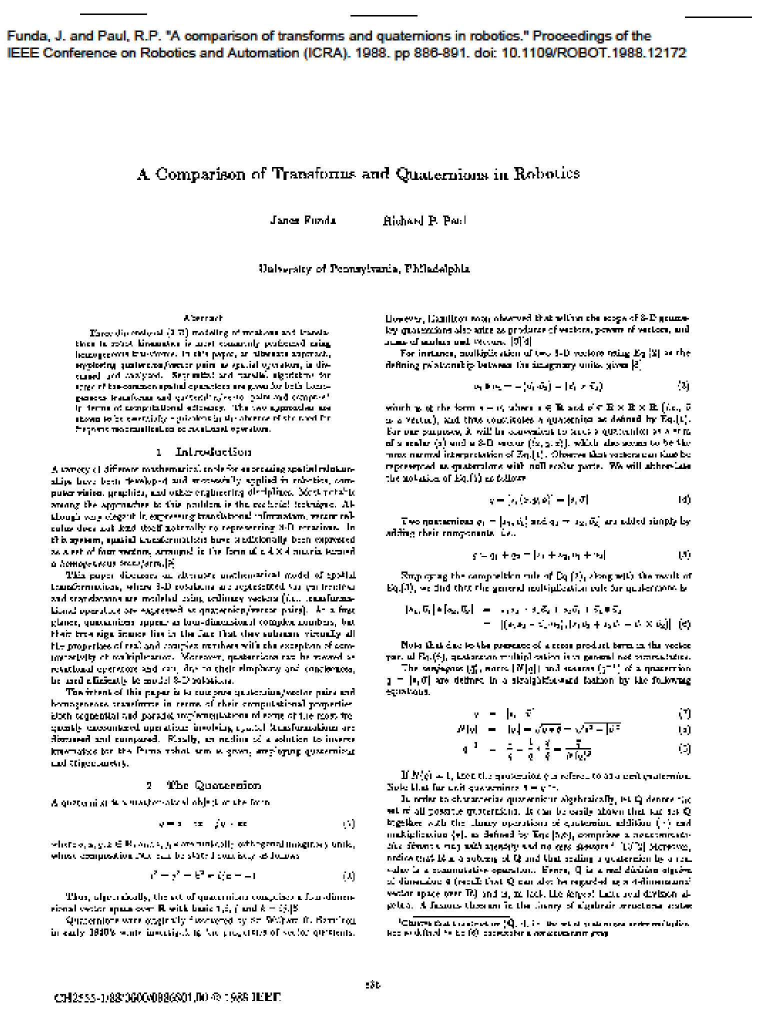
In summary, the differences in computational efficiency between the two approaches are not sufficiently significant to warrant a particular choice on that basis alone. Instead, the deciding factor may be the nature of a particular computation or the available hardware.

Ask yourself: Should I read the rest of the paper?

Pass 2

Quick Primer on Paper Reading

- Read each section in more depth
- Mark terms you do not know
- Write questions you have for the authors
- Take a careful look at tables and figures – do they make sense?


Funda, J. and Paul, R.P. "A comparison of transforms and quaternions in robotics." Proceedings of the IEEE Conference on Robotics and Automation (ICRA). 1988. pp 886-891. doi: 10.1109/ROBOT.1988.12172

A Comparison of Transforms and Quaternions in Robotics

James Funda   Richard P. Paul

University of Pennsylvania, Philadelphia

Funda, J and Paul, RP. "A comparison of transforms and quaternions in robotics." ICRA 1988.

Main contributions:

- Explanation of quaternion math as applied to rotations in 3D space
- Evaluation of computational complexity for basic rotations, inverses, and normalization

Strengths

- Well-organized
- Succinct summary of quaternions and homogeneous transformations
- Analysis supported by data tables

Weaknesses

- Requires significant math background
- Lack of intuition in what quaternions mean physically
- Evaluation on PUMA IK only on quaternions
- Conclusion is wishy-washy

Side comments:

- images/figures
- page limits

Questions from the class

- Practical applications
  (quaternions vs homogeneous transformations)
- Re-normalization
- Evaluation of computational complexity vs. running on real hardware
  (and the effect of modern hardware)

- Kudos to everyone who found the typo in eq. 18:
  $$q * v * q^{-1} = [0,2(u \cdot v)u + (s^2 - u \cdot u)v + 2(u \times v)]$$
- Thanks to your classmate who posted a video on Piazza for intuition!

We'll come back to these at the end.

a 3x3 rotation matrix

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

a 3x1 vector

$$\vec{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

the corresponding eigenvector of the matrix **R**

a scalar

$$\lambda$$

an eigenvalue of the matrix **R**

a simple equation

$$\mathbf{R}\vec{v} = \lambda\vec{v}$$

What is the geometric meaning of the vector?
It's the axis of rotation.

What will the associated eigenvalue be?
+1 (no scaling)

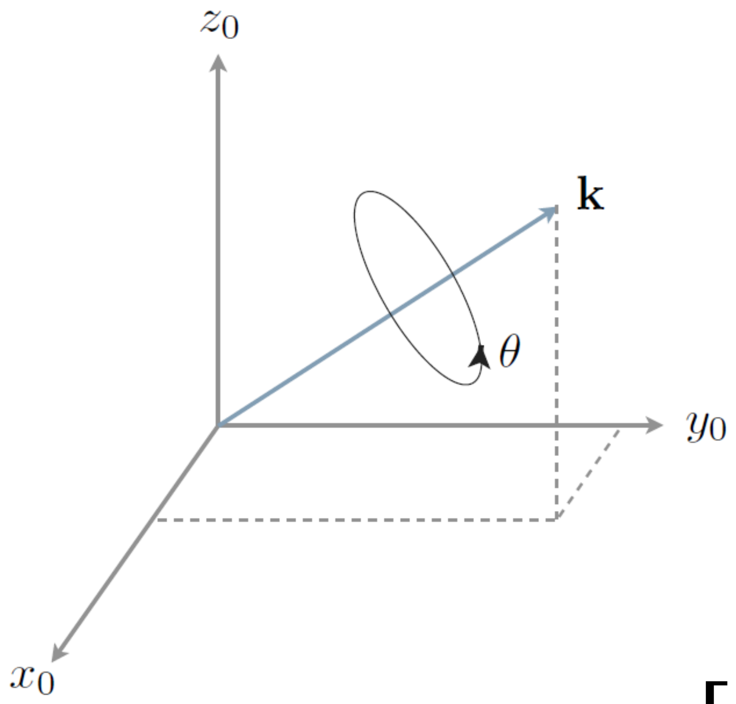
What does this equation mean?

Multiplying the vector by the rotation matrix only scales the vector;
it doesn't change the vector's direction.

This should remind you of eigenvalues and eigenvectors.

Angle/Axis Representation (Lecture 3)

Rotation by an angle about an axis in space



$$\mathbf{k} = \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} \text{ with } \|\mathbf{k}\| = 1$$

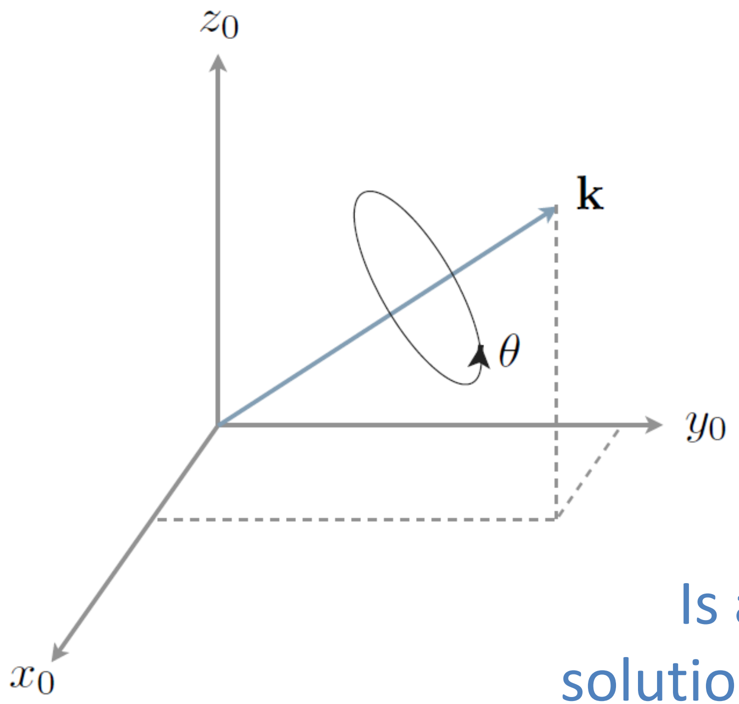
Let $v_\theta = \text{vers}\,\theta = 1 - c_\theta$

$$\mathbf{R}_{k,\theta} = \begin{bmatrix} k_x^2 v_\theta + c_\theta & k_x k_y v_\theta - k_z s_\theta & k_x k_z v_\theta + k_y s_\theta \\ k_x k_y v_\theta + k_z s_\theta & k_y^2 v_\theta + c_\theta & k_y k_z v_\theta - k_x s_\theta \\ k_x k_z v_\theta - k_y s_\theta & k_y k_z v_\theta + k_x s_\theta & k_z^2 v_\theta + c_\theta \end{bmatrix}$$

Angle/Axis Representation (Lecture 3)

Is k defined in frame 0 or frame 1?
Both.

Any rotation matrix can be represented this way!



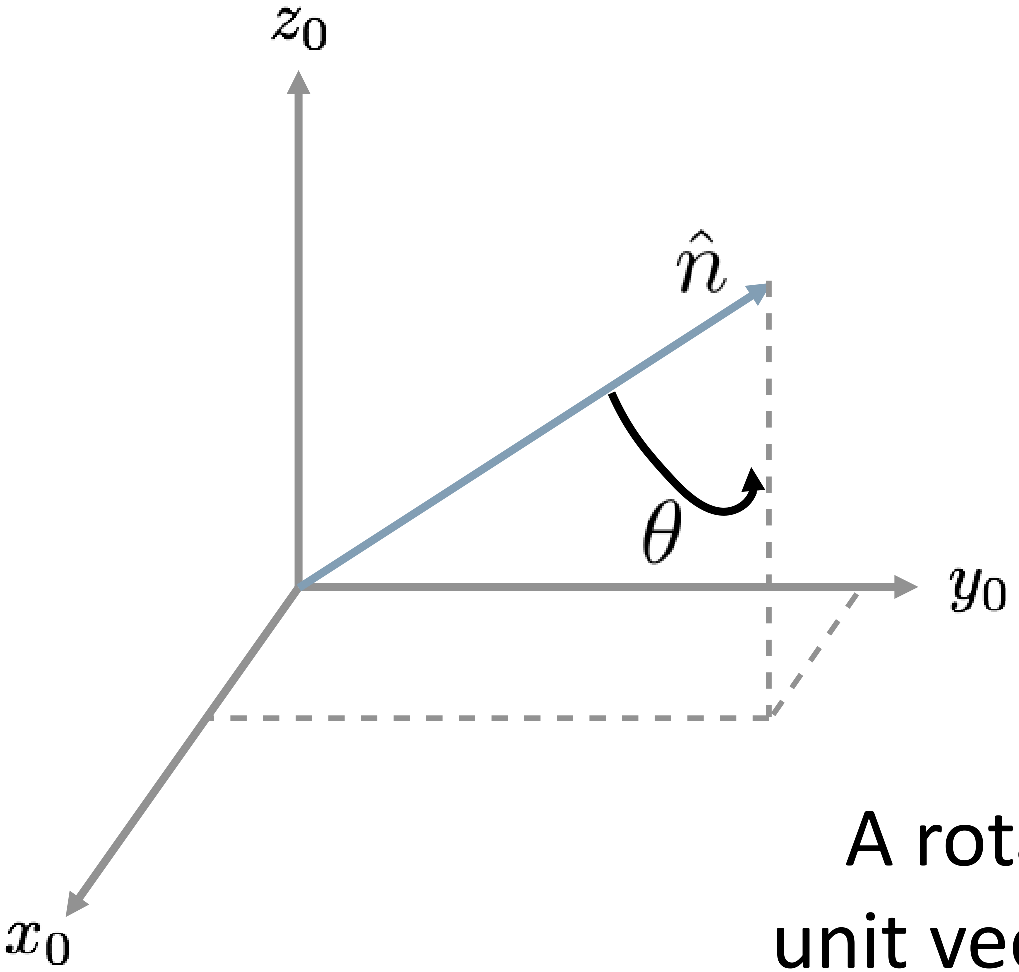
$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\mathbf{k} = \frac{1}{2s_\theta} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

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

Is axis/angle solution unique?
No: $\mathbf{R}_{k,\theta} = \mathbf{R}_{-k,-\theta}$

$\mathbf{Rk} = \mathbf{k} \Longrightarrow \mathbf{k}$ is the eigenvector of $\mathbf{R}$ corresponding to eigenvalue $\lambda = 1$



$$\hat{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \qquad |\hat{n}| = 1$$

A rotation around the
unit vector $\hat{n}$ by the angle θ
can be represented as a **quaternion** as follows:

$$Q = (\underbrace{\cos(\theta/2)}_{\text{scalar}},\ \underbrace{n_x \sin(\theta/2),\ n_y \sin(\theta/2),\ n_z \sin(\theta/2)}_{\text{three-component vector}})$$

$$Q = (q_0, q_1, q_2, q_3)$$

$$Q = (\cos(\theta/2),\ n_x \sin(\theta/2),\ n_y \sin(\theta/2),\ n_z \sin(\theta/2))$$

What is the magnitude of this quaternion?

$$(\cos(\theta/2))^2 + (n_x \sin(\theta/2))^2 + (n_y \sin(\theta/2))^2 + (n_z \sin(\theta/2))^2$$

$$\cos^2(\theta/2) + n_x^2 \sin^2(\theta/2) + n_y^2 \sin^2(\theta/2) + n_z^2 \sin^2(\theta/2)$$

$$\cos^2(\theta/2) + (n_x^2 + n_y^2 + n_z^2) \sin^2(\theta/2)$$

$$|\hat{n}| = 1$$

$$\cos^2(\theta/2) + \sin^2(\theta/2)$$

$$= 1$$ This is a *unit* quaternion.

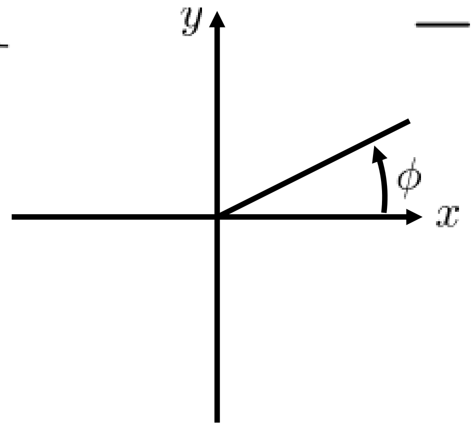
Rotations in 3D have only three degrees of freedom.

Unit quaternions use one extra number to avoid gimbal lock.

$$Q = (\cos(\theta/2),\ n_x \sin(\theta/2),\ n_y \sin(\theta/2),\ n_z \sin(\theta/2))$$

What values will the components have?

$$-1 \leq \cos\phi \leq 1$$



$$-1 \leq \sin\phi \leq 1$$

$$\text{when } -\pi \leq \theta \leq \pi, \quad 0 \leq \cos(\theta/2) \leq 1$$

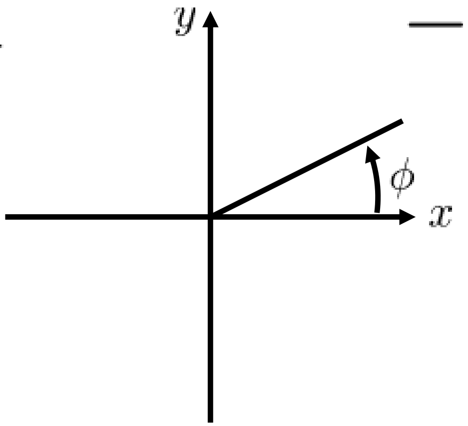
$$\text{when } \pi \leq \theta \leq 3\pi, \quad -1 \leq \cos(\theta/2) \leq 0$$

Dividing the angle by two allows you to specify more than half a rotation in either direction.

$$Q = (\cos(\theta/2),\ n_x \sin(\theta/2),\ n_y \sin(\theta/2),\ n_z \sin(\theta/2))$$

What values will the components have?

$$-1 \leq \cos\phi \leq 1$$



$$-1 \leq \sin\phi \leq 1$$

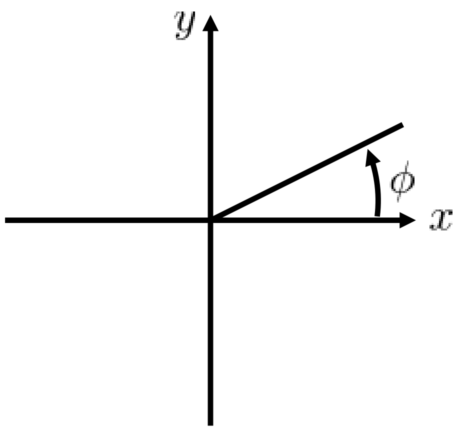
Recall the two solutions for inverse axis/angle:

$$R_{k,\theta} = R_{-k,-\theta}$$

These two solutions are indistinguishable in quaternions because sine of theta/2 is multiplied by all of the components of the unit vector showing the rotation axis.

$$Q = (\cos(\theta/2),\ n_x \sin(\theta/2),\ n_y \sin(\theta/2),\ n_z \sin(\theta/2))$$

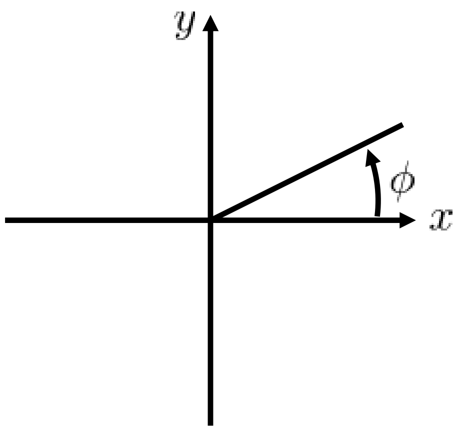
What quaternion represents zero rotation?



What quaternion represents a full 360º rotation?

$$Q = (\cos(\theta/2),\ n_x \sin(\theta/2),\ n_y \sin(\theta/2),\ n_z \sin(\theta/2))$$

What quaternion represents 180º rotation around the x-axis?



What quaternion represents 60º rotation around the z-axis?

$$Q = (\cos(\theta/2),\ n_x \sin(\theta/2),\ n_y \sin(\theta/2),\ n_z \sin(\theta/2))$$

How can we convert between quaternions and rotation matrices?

$$Q = (q_0, q_1, q_2, q_3) \qquad \theta = ? \qquad \hat{n} = ?$$

$$q_0 = \cos(\theta/2) \longrightarrow \theta = 2\cos^{-1}(q_0)$$

$$\theta = 2\,\mathrm{atan2}\left(\frac{\pm\sqrt{1-q_0^2}}{q_0}\right)$$

$$\sin(\theta/2) = \pm\sqrt{1-q_0^2}$$

$$\hat{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} q_1/\pm\sqrt{1-q_0^2} \\ q_2/\pm\sqrt{1-q_0^2} \\ q_3/\pm\sqrt{1-q_0^2} \end{bmatrix}$$

$$Q = (\cos(\theta/2),\ n_x \sin(\theta/2),\ n_y \sin(\theta/2),\ n_z \sin(\theta/2))$$

How can we convert between quaternions and rotation matrices?

where $v_\theta = \operatorname{vers}\theta = 1 - \cos\theta$

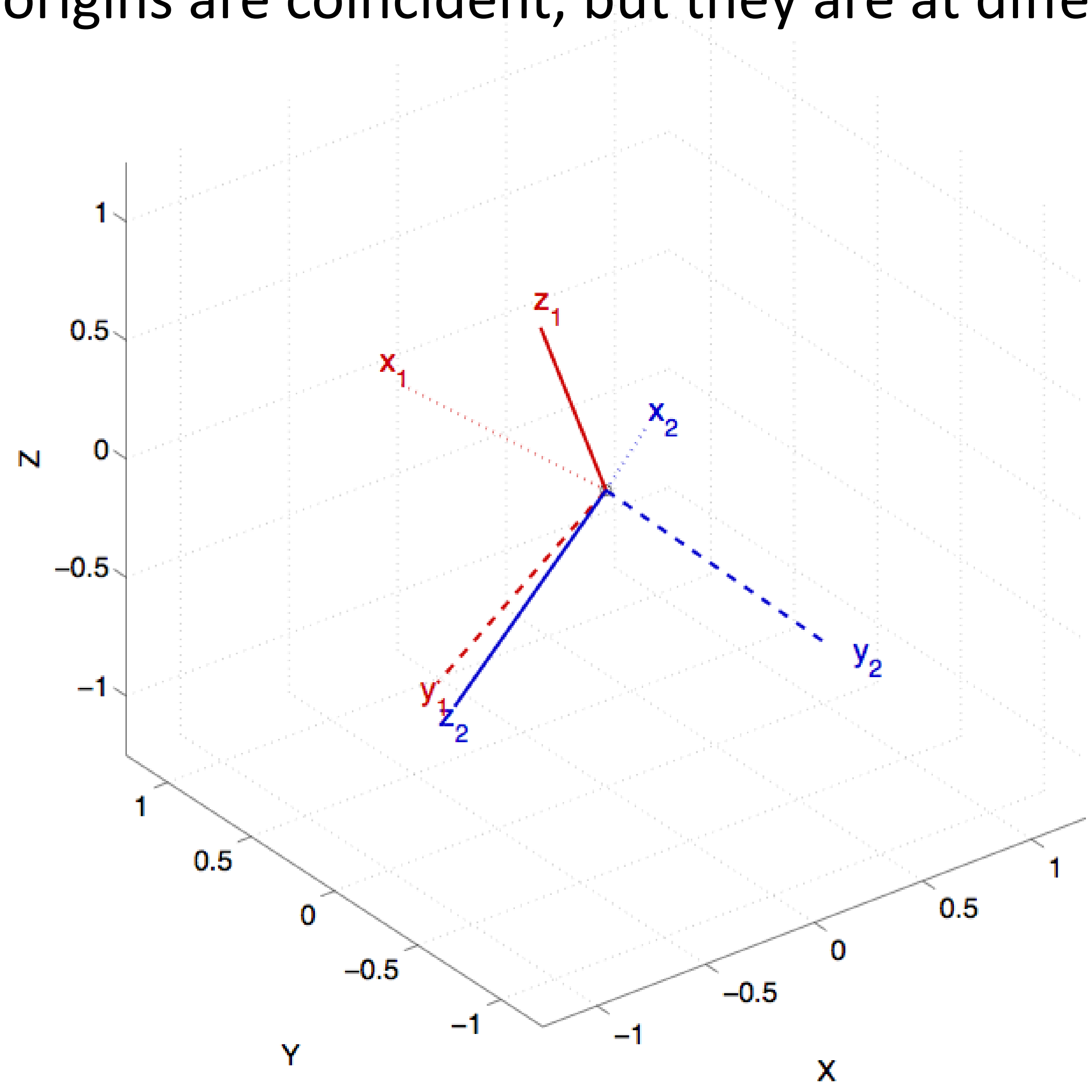
$$R_{k,\theta} = \left[\begin{array}{c|c|c} k_x^2 v_\theta + c_\theta & k_x k_y v_\theta - k_z s_\theta & k_x k_z v_\theta + k_y s_\theta \\ k_x k_y v_\theta + k_z s_\theta & k_y^2 v_\theta + c_\theta & k_y k_z v_\theta - k_x s_\theta \\ k_x k_z v_\theta - k_y s_\theta & k_y k_z v_\theta + k_x s_\theta & k_z^2 v_\theta + c_\theta \end{array}\right]$$

$$\theta = 2\,\operatorname{atan2}\left(\frac{\pm\sqrt{1 - q_0^2}}{q_0}\right)$$

Plug into axis/angle formula from SHV

$$\hat{k} = \hat{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} q_1 / \pm\sqrt{1 - q_0^2} \\ q_2 / \pm\sqrt{1 - q_0^2} \\ q_3 / \pm\sqrt{1 - q_0^2} \end{bmatrix}$$

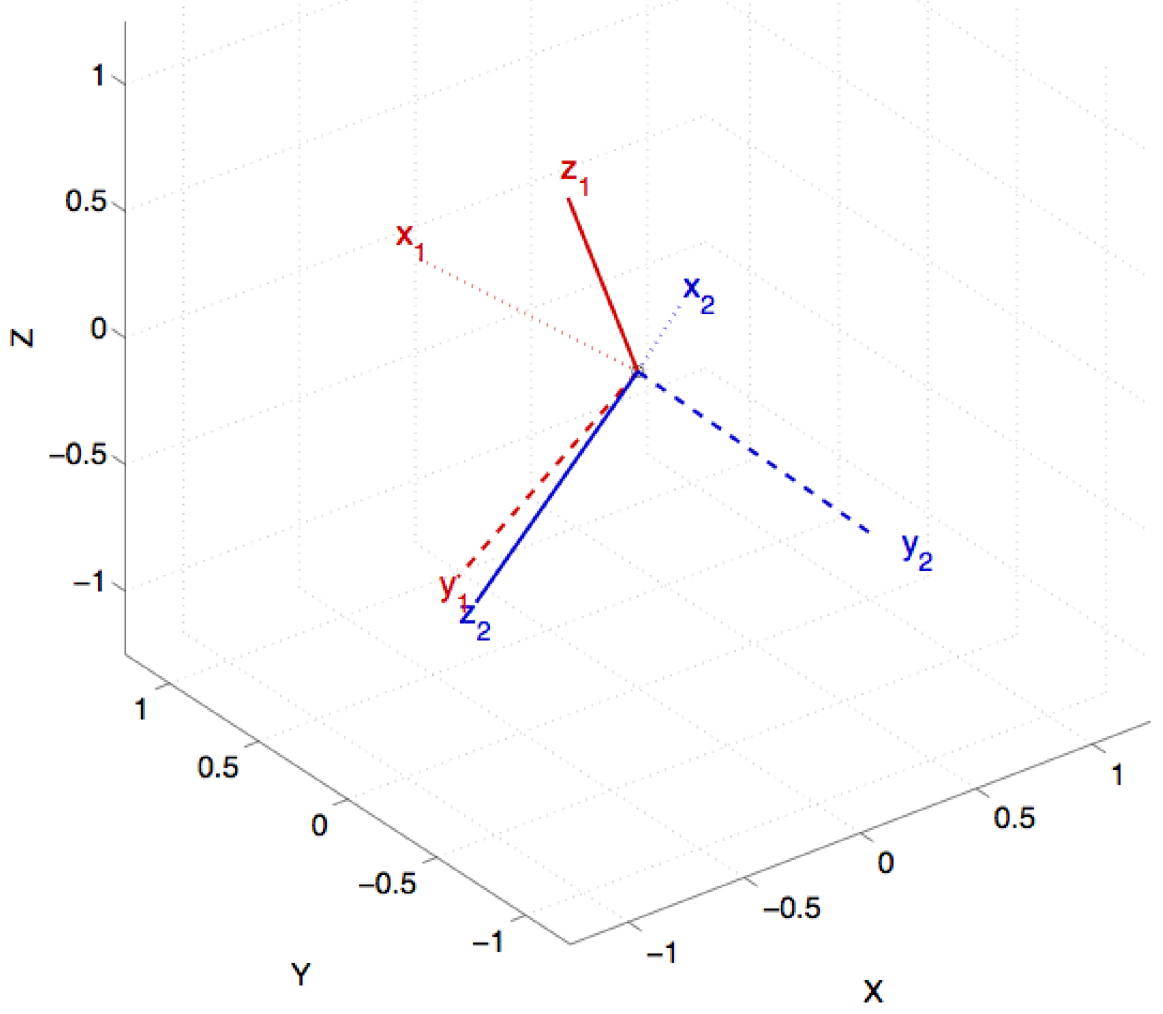
Imagine we have two right-handed frames in 3D:
their origins are coincident, but they are at different orientations.



We want to animate a frame rotating from orientation 1 to orientation 2.

What are all the ways we could animate this motion?

Imagine we have two right-handed frames in 3D:
their origins are coincident, but they are at different orientations.



We could interpret using:
Rotation matrix elements
Euler angles
Quaternions
Axis/Angle from 1 to 2

What do we care about?

We want the motion
to be intuitive.
We want each
intermediate frame
to be **valid** (three
orthogonal unit vectors).

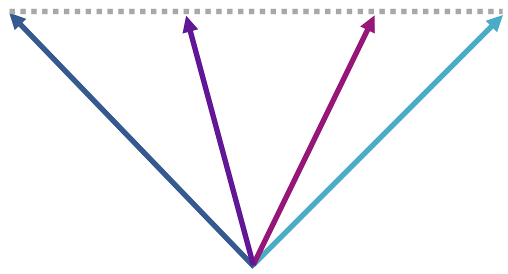
Matrix Elements

$$R_1 = \begin{bmatrix} -0.1817 & 0.7236 & 0.6658 \\ -0.6198 & -0.6100 & -0.4938 \\ 0.7634 & -0.3230 & 0.5594 \end{bmatrix}$$

Linear interpolation

$$R_2 = \begin{bmatrix} 0.6404 & 0.3577 & -0.6797 \\ -0.3121 & 0.9298 & 0.1952 \\ 0.7018 & 0.0872 & 0.7071 \end{bmatrix}$$

Each axis changes
linearly from
old to new:
from x1 to x2
from y1 to y2
from z1 to z2



$$\det(\mathbf{R}) \neq 1$$

The intermediate
frames are not valid.

Euler Angles

Each Euler angle changes
linearly from old to new:
from phi1 to phi2
from theta1 to theta2
from psi1 to psi2

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix}$$

Use atan2 to determine φ
for both θ options

Use atan2 to determine ψ
for both θ options

Two solutions for θ because sign
of s_θ is not known.

$$\phi = ? \quad \theta = ? \quad \psi = ?$$

$$\det(\mathbf{R}) = 1$$

The intermediate
frames are valid.

Inverse Euler angles yields two solutions.

Interpolating between the
worst pair often creates
wild motions.

We sometimes see
weird motion even
for the best pair.

Quaternions (SLERP)

The Quaternion changes from old to new:
from Q1 to Q2

Linear interpolation (LERP) yields non-unit quaternions,
which are not valid frames, just like interpolating rotation matrices.

Spherical linear interpolation (SLERP) yields only unit quaternions.

SLERP on quaternions yields exactly the same motion as linear
interpolation of the angle in Axis/Angle

Rotations are often interpolated using quaternions.
This makes them very useful for mobile robots.

Beyond representing the orientation of a frame in 3D,
what can we do with quaternions?

The same things you can do with a rotation matrix:
compose successive rotations,
calculate the coordinates of a vector in another frame,
and apply the rotation as an operator on a vector.

$$\mathbf{R}_2^0 = \mathbf{R}_1^0\,\mathbf{R}_2^1 \qquad \mathbf{v}_p^0 = \mathbf{R}_1^0\,\mathbf{v}_p^1 \qquad \mathbf{p}_b^0 = \mathbf{R}\,\mathbf{p}_a^0$$

But how do we multiply quaternions together
and multiply a quaternion into a vector?

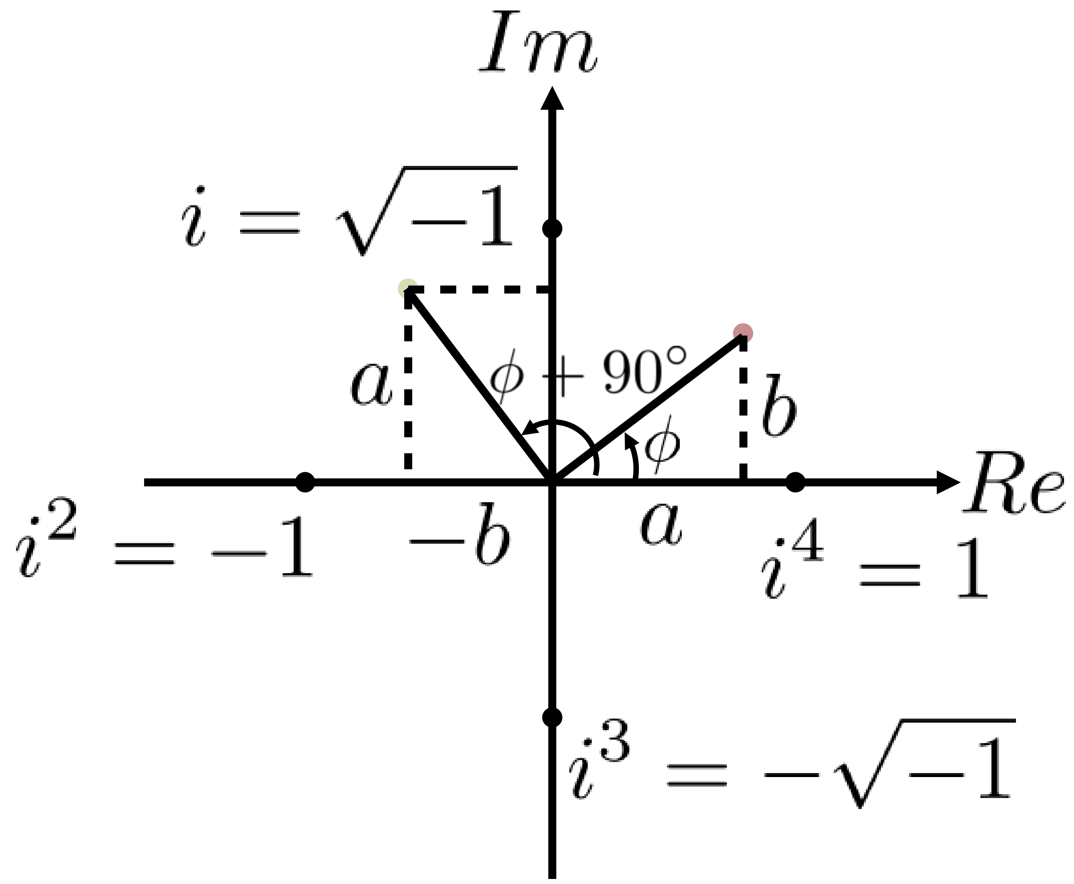
We need to define a quaternion operator.

Quaternions were discovered by Sir William Rowan Hamilton in 1843

$$i^2 = j^2 = k^2 = ijk = -1$$

A brief review of unit-magnitude complex numbers

$$a + ib$$



$$a^2 + b^2 = 1$$

$$b = \sin\phi \qquad a = \cos\phi$$

$$\phi = \text{atan2}\left(\frac{b}{a}\right)$$

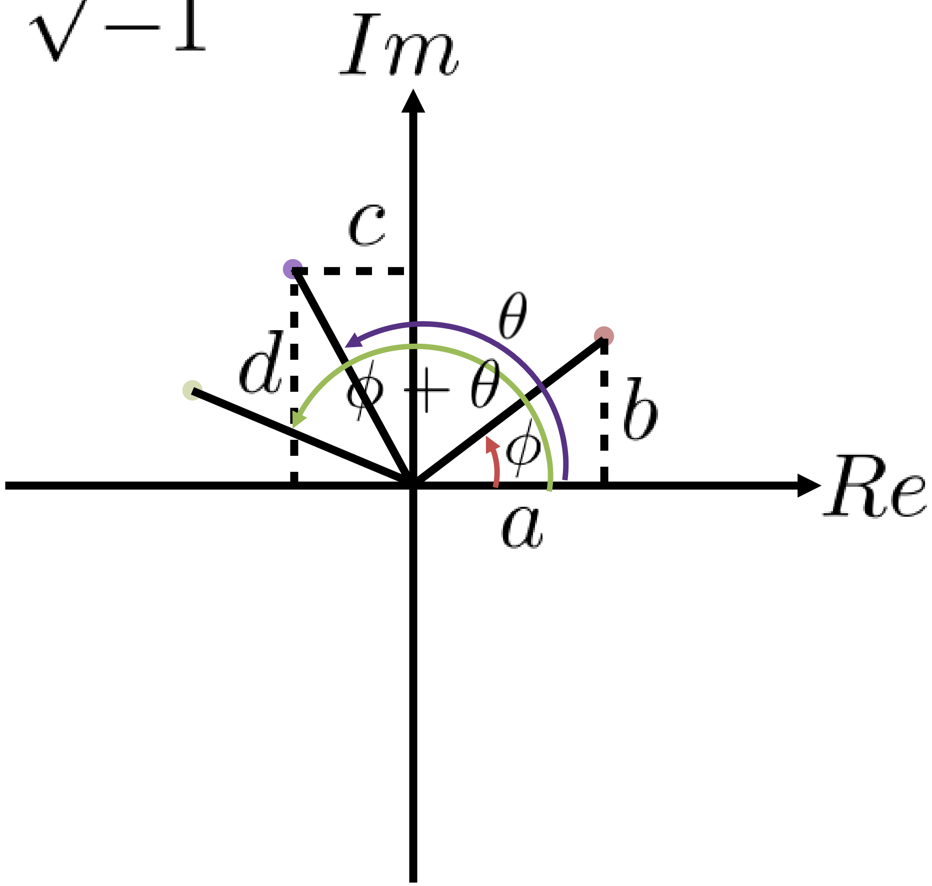
Multiplying by i causes a 90º rotation in the plane.

$$(a + ib)(i) = -b + ia$$

Multiplying complex numbers

$$a + ib \qquad a^2 + b^2 = 1 \qquad c + id \qquad c^2 + d^2 = 1$$

$$i = \sqrt{-1}$$



$$(a + ib)(c + id)$$

$$= ac + ibc + iad + i^2bd$$

$$= ac - bd + i(bc + ad)$$

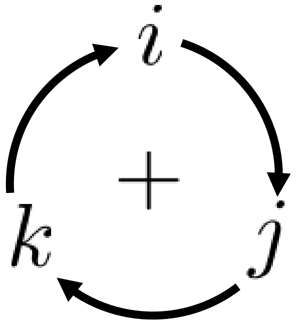
$$= \cos(\phi + \theta) + i\sin(\phi + \theta)$$

Multiplying two unit complex numbers yields a unit complex number whose angle is the sum of the original two angles.

That sounds like what we want to do with quaternions!

Hamilton defined three independent square roots for -1

$$i^2 = j^2 = k^2 = -1$$



Order of multiplication matters!

$$i = jk = -kj$$

$$j = ki = -ik$$

$$k = ij = -ji$$

This extension of complex numbers gives us a way to multiply quaternions together.

Use i, j, and k to express quaternion components algebraically.

$$Q = (\underbrace{q_0}_{\text{scalar}}, \underbrace{q_1, q_2, q_3}_{\text{three-component vector}})$$

$$Q = q_0 + iq_1 + jq_2 + kq_3$$

Multiplying Two Quaternions

$$X = (x_0, \underbrace{x_1, x_2, x_3}_{\vec{x}}) \qquad Y = (y_0, \underbrace{y_1, y_2, y_3}_{\vec{y}})$$

$$Z = XY = ?$$

$$= (x_0 + ix_1 + jx_2 + kx_3)(y_0 + iy_1 + jy_2 + ky_3)$$

$$= x_0y_0 + x_0iy_1 + x_0jy_2 + x_0ky_3 \ldots$$

$$= \underbrace{x_0y_0 - \vec{x}^T\vec{y}}_{z_0} + \underbrace{x_0\vec{y} + y_0\vec{x} + \vec{x} \times \vec{y}}_{\vec{z}}$$

$$\boxed{XY = x_0y_0 - \vec{x}^T\vec{y} + x_0\vec{y} + y_0\vec{x} + \vec{x} \times \vec{y}}$$

What is the Identity Quaternion?

$$Q_I = ?$$

$$QQ_I = Q_I Q = Q$$

$$XY = x_0 y_0 - \vec{x}^T \vec{y} + x_0 \vec{y} + y_0 \vec{x} + \vec{x} \times \vec{y}$$

$$Q_I = (1, 0, 0, 0)$$

What does it mean?
No rotation!

$$\begin{aligned} Q_I Y &= (1 + 0i + 0j + 0k)(y_0 + iy_1 + jy_2 + ky_3) \\ &= Y \checkmark \end{aligned}$$

$$\begin{aligned} XQ_I &= (x_0 + ix_1 + jx_2 + kx_3)(1 + 0i + 0j + 0k) \\ &= X \checkmark \end{aligned}$$

What is the Conjugate for a Unit Quaternion?

$$Q = (q_0, q_1, q_2, q_3)$$

Recall Complex Conjugates

$$c = a + ib \qquad c^* = ?$$

$$cc^* = c^*c = 1$$

$$c^* = a - ib$$

$$cc^* = (a + ib)(a - ib)$$

$$= a^2 - iab + iab - i^2b^2$$

$$= a^2 + b^2$$

$$= 1 \checkmark$$

$$Q^* = ?$$

$$QQ^* = Q^*Q = Q_I$$

$$Q^* = (q_0, -q_1, -q_2, -q_3)$$

$$Q = (\cos(\theta/2),\ n_x \sin(\theta/2),\ n_y \sin(\theta/2),\ n_z \sin(\theta/2))$$

Doing a rotation of the same magnitude in the opposite direction.

How do we apply a unit quaternion's rotation to a vector?

$$Q = (q_0, q_1, q_2, q_3) \qquad \vec{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

Put the vector into quaternion form
with zero as its real component.

$$Q_v = (0, v_x, v_y, v_z)$$

Calculate the new, rotated coordinates of v
by premultiplying by Q and post-multiplying by Q*

$$QQ_vQ^*$$

How do we apply a unit quaternion's rotation to a vector?

$$Q = (q_0, q_1, q_2, q_3)$$

$$\vec{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

$$QQ_vQ^*$$

$$Q = \left(\cos\frac{\theta}{2}, \left(\sin\frac{\theta}{2}\right)\mathbf{n}\right)$$

$$Q_v = (0, \mathbf{v})$$

$$Q^* = \left(\cos\frac{\theta}{2}, -\left(\sin\frac{\theta}{2}\right)\mathbf{n}\right)$$

Nonzero scalar

$$QQ_v = (\boxed{} \quad)$$

$$QQ_vQ^* = (\quad ,$$

$$(\mathbf{n}\cdot\mathbf{v})\mathbf{n} + \mathbf{v} - (\mathbf{n}\cdot\mathbf{v})\mathbf{n}$$

$$-\mathbf{v}_\perp$$

$$QQ_vQ^* = (0,$$

Rotation by theta!

$$Q = (q_0, q_1, q_2, q_3)$$

multiplication of quaternions

$$XY = x_0 y_0 - \vec{x}^T \vec{y} + x_0 \vec{y} + y_0 \vec{x} + \vec{x} \times \vec{y}$$

identity quaternion

$$Q_I = (1, 0, 0, 0)$$

$$QQ_I = Q_I Q = Q$$

conjugate quaternion

$$QQ^* = Q^* Q = Q_I$$

$$Q^* = (q_0, -q_1, -q_2, -q_3)$$

applying quaternion to a vector

$$QQ_vQ^*$$

$$Q_v = (0, v_x, v_y, v_z)$$

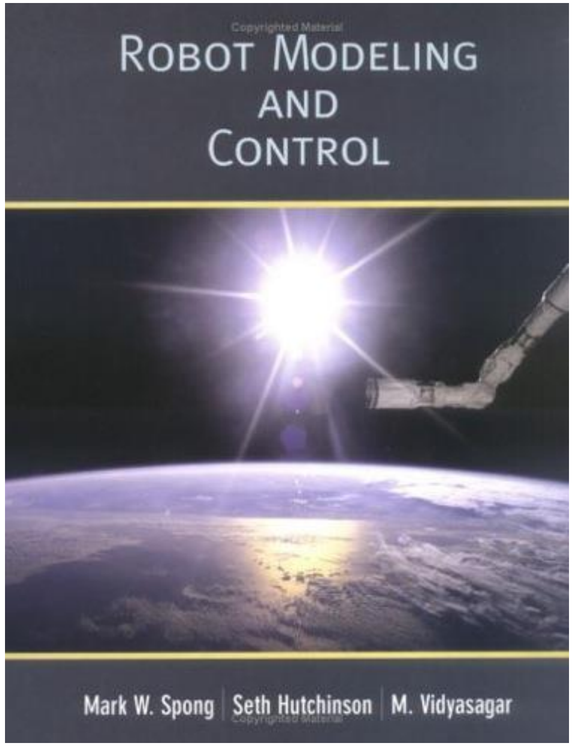
Back to your questions

- Practical applications
  (quaternions vs homogeneous transformations)
- Re-normalization
- Evaluation of computational complexity vs. running on real hardware
  (and the effect of modern hardware)

var4 = (var1 + var2) * var3

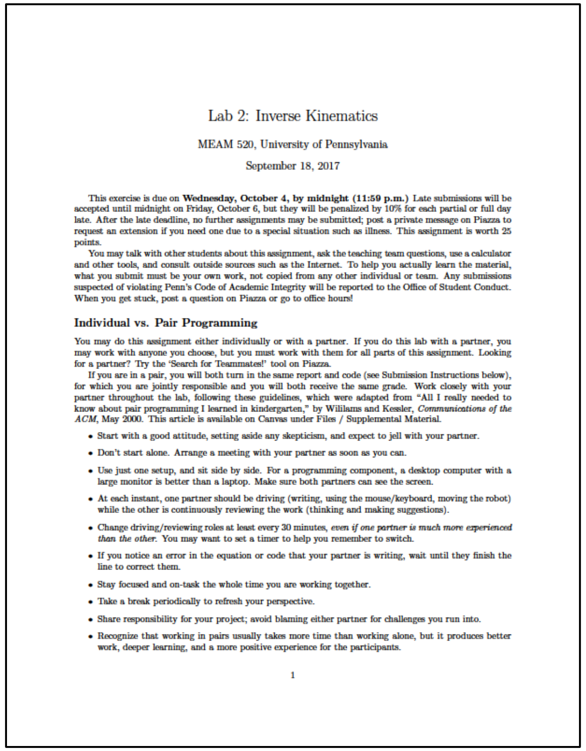
```
mov eax,var1
add eax,var2
mul var3
mov var4,eax
```

Next time: Trajectory Planning in Joint Space!



Chapter 5: Path and Trajectory Planning

- Read 5.5



Lab 2: Inverse Kinematics

MEAM 520, University of Pennsylvania

September 18, 2017

This exercise is due on **Wednesday, October 4, by midnight (11:59 p.m.)** Late submissions will be accepted until midnight on Friday, October 6, but they will be penalized by 10% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Piazza to request an extension if you need one due to a special situation such as illness. This assignment is worth 25 points.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you submit must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. When you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

You may do this assignment either individually or with a partner. If you do this lab with a partner, you may work with anyone you choose, but you must work with them for all parts of this assignment. Looking for a partner? Try the 'Search for Teammates!' tool on Piazza.

If you are in a pair, you will both turn in the same report and code (see Submission Instructions below), for which you are jointly responsible and you will both receive the same grade. Work closely with your partner throughout the lab, following these guidelines, which were adapted from "All I really needed to know about pair programming I learned in kindergarten," by Wililams and Kessler, *Communications of the ACM*, May 2000. This article is available on Canvas under Files / Supplemental Material.

- Start with a good attitude, setting aside any skepticism, and expect to jell with your partner.
- Don't start alone. Arrange a meeting with your partner as soon as you can.
- Use just one setup, and sit side by side. For a programming component, a desktop computer with a large monitor is better than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (writing, using the mouse/keyboard, moving the robot) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every 30 minutes, *even if one partner is much more experienced than the other*. You may want to set a timer to help you remember to switch.
- If you notice an error in the equation or code that your partner is writing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.
- Recognize that working in pairs usually takes more time than working alone, but it produces better work, deeper learning, and a more positive experience for the participants.

1

Lab 2: Inverse Kinematics due 10/3

- Pre-lab due tomorrow
- You can now do all the tasks