# AAE 590: Lie Group Methods for Control and Estimation

## Motivation: Why Lie Groups?

Dr. James Goppert

Purdue University
School of Aeronautics and Astronautics

Spring 2026

# Welcome to AAE 590

**Lie Group Methods for Control and Estimation**

**What you'll learn:**

- How to represent rotations and rigid body poses correctly
- Why traditional methods fail (and when)
- Modern geometric approaches used in aerospace research
- Practical algorithms: IEKF, Equivariant Filter, geometric control

**Prerequisites assumed:**

- Calculus (derivatives, Taylor series)
- Basic linear algebra (matrices, eigenvalues)
- Programming experience (Python for homework)

# The Central Problem

## Central Question

How do we represent the orientation of a rigid body?

**Seems simple...**

- We have 3 degrees of freedom (roll, pitch, yaw)
- Just use 3 numbers, right?

## The Catch

- Rotations don't behave like vectors in $\mathbb{R}^3$
- You can't just add/subtract orientations
- The space of rotations is **curved**

# Why Should You Care?

**These problems appear everywhere in aerospace:**

**Spacecraft:**

- Attitude determination
- Slew maneuvers
- Pointing control

**Aircraft:**

- Flight control systems
- Upset recovery
- Aerobatic maneuvers

**Robotics:**

- SLAM and localization
- Manipulation
- Path planning

**Autonomous vehicles:**

- Sensor fusion (IMU + GPS)
- State estimation
- Motion planning

# A Motivating Failure

**Scenario:** You're designing an attitude estimator using Euler angles.

**Your approach:**

1. Propagate: $\phi_{k+1} = \phi_k + \dot{\phi} \cdot \Delta t$
2. Update with sensor measurements
3. Works great in simulation!

**Flight test:** Aircraft pitches to 85°...

- Euler rates explode: $\dot{\phi}, \dot{\psi} \to \infty$
- Estimator diverges
- Control system fails

**What happened?**

Gimbal lock! Your 3-parameter representation has a **singularity**.

# Let's Start Simple: 2D Rotations

**Rotation in 2D:** Rotate point $(x, y)$ by angle $\theta$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

**This is a rotation matrix:** $R(\theta) \in \mathrm{SO}(2)$

**Key properties:**

- $R^T R = I$ (orthogonal)
- $\det(R) = \cos^2\theta + \sin^2\theta = 1$
- $R(\theta_1)R(\theta_2) = R(\theta_1 + \theta_2)$ (composition = addition!)

**2D is Special**

In 2D, angles just add. In 3D, life gets complicated!

# The 3D Rotation Problem

**In 3D, we need to specify:**

- Which axis to rotate around
- How much to rotate
- The **order** of multiple rotations

**Degrees of freedom:** 3 (same as 2D + 1)

**But now:**

- Rotations don't commute: $R_x R_y \neq R_y R_x$
- Composition is matrix multiplication, not addition
- The space has non-trivial topology

**The Challenge**

How do we parameterize this 3-DOF space without problems?

# Rotation Representations: A Comparison

| Representation | Parameters | Singularities | Redundancy |
|---|---|---|---|
| Euler angles | 3 | Yes (gimbal lock) | No |
| Axis-angle | 3 | Yes ($\theta = 0$) | No |
| Rodrigues vector | 3 | Yes ($\theta = \pi$) | No |
| MRP | 3 | Yes ($\theta = 2\pi$) | No |
| Quaternions | 4 | No | Yes ($q = -q$) |
| Rotation matrix | 9 | No | Yes (6 constraints) |

## A Fundamental Trade-off

**Minimal parameters $\Rightarrow$ Singularities**
**No singularities $\Rightarrow$ Redundancy**

# Euler Angles: Definition

**Idea:** Compose three rotations about coordinate axes

**3-2-1 (Aerospace) convention:**

$$R(\phi, \theta, \psi) = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi)$$

- $\phi =$ roll (rotation about $x$-axis)
- $\theta =$ pitch (rotation about $y$-axis)
- $\psi =$ yaw (rotation about $z$-axis)

**Other conventions exist:**

- 3-1-3 (classical Euler angles)
- 1-2-3 (roll-pitch-yaw)
- 12 total possible sequences

# Euler Angles: The Rotation Matrices

**Elementary rotations:**

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

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

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# Euler Angle Kinematics

**Problem:** Given body angular velocity $\omega = [p, q, r]^T$, find Euler rates.

**The relationship:**

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

**Notice the problem terms:**

- $\tan\theta$ appears in row 1
- $\sec\theta$ appears in row 3

**When $\theta = \pm 90^\circ$**

$\tan(\pm 90^\circ) = \pm\infty$ and $\sec(\pm 90^\circ) = \pm\infty$
The kinematic equations become **singular**!

# What is Gimbal Lock?

**Physical picture:** Three nested gimbals

**Normal operation:**

- Outer gimbal: yaw
- Middle gimbal: pitch
- Inner gimbal: roll
- 3 independent axes

**At $\theta = 90°$:**

- Outer and inner axes align!
- Yaw and roll do the same thing
- Only 2 independent axes
- Lost 1 degree of freedom

## Key Insight

This is a **coordinate singularity**, not a physical limitation.
The vehicle can still rotate in all directions!

# Gimbal Lock: Mathematical View

**At $\theta = 90°$:**
The Euler angle to rotation matrix becomes:

$$R = \begin{bmatrix} 0 & -\sin(\phi - \psi) & \cos(\phi - \psi) \\ 0 & \cos(\phi - \psi) & \sin(\phi - \psi) \\ -1 & 0 & 0 \end{bmatrix}$$

**Notice:** Only $(\phi - \psi)$ appears, not $\phi$ and $\psi$ separately!

**Consequences:**

- Infinitely many Euler angle combinations give the same rotation
- Cannot recover $\phi$ and $\psi$ individually from $R$
- Jacobian of $(\phi, \theta, \psi) \mapsto R$ loses rank

# Gimbal Lock: A Historical Note

## Apollo Program

The Apollo spacecraft IMU used physical gimbals to track orientation.

**Engineering reality:**

- Gimbal lock was a **known constraint**, not a failure
- Astronauts trained to avoid certain orientations
- Mission procedures kept spacecraft away from singular configurations
- Apollo successfully managed this through operational discipline

**The lesson:**

- Gimbal lock is a **mathematical** limitation of 3-parameter representations
- Can be managed with careful procedures, but adds operational burden
- Modern systems use quaternions or rotation matrices to avoid the issue entirely

# Axis-Angle Representation

**Idea:** Any rotation is a rotation by angle $\theta$ about some axis $\hat{a}$

**Representation:** $(\hat{a}, \theta)$ where $\|\hat{a}\| = 1$
**Or as a vector:** $\omega = \theta\hat{a}$ (angle encoded in magnitude)

**Rodrigues' rotation formula:**

$$R = I + \sin\theta\, \hat{a}^{\wedge} + (1 - \cos\theta)(\hat{a}^{\wedge})^2$$

where $\hat{a}^{\wedge}$ is the skew-symmetric matrix of $\hat{a}$.

**Singularity at $\theta = 0$**

When $\theta = 0$, the axis $\hat{a}$ is **undefined**!
Any axis gives the identity rotation.

# Rodrigues Vector (Classical)

**Definition:**

$$r = \hat{a} \tan(\theta/2)$$

**Conversion to rotation matrix:**

$$R = I + \frac{2}{1 + \|r\|^2} \left( r^\wedge + (r^\wedge)^2 \right)$$

**Advantages:**

- Only 3 parameters
- Rational functions (no trig in formulas)

**Singularity at $\theta = \pi$**

$\tan(\pi/2) = \infty$
Cannot represent $180^\circ$ rotations!

# Modified Rodrigues Parameters (MRP)

**Definition:**

$$\sigma = \hat{a} \tan(\theta/4)$$

**Advantages over classical Rodrigues:**

- Singularity moved from $180^\circ$ to $360^\circ$
- Well-behaved for most practical rotations
- Used in some spacecraft attitude systems

**Singularity at $\theta = 2\pi$**

$\|\sigma\| = \tan(\theta/4) \to \tan(\pi/2) = \infty$
Still blows up at one full rotation.

**Shadow MRP:** $\sigma_s = -\sigma/\|\sigma\|^2$ can extend range, but adds complexity.

# Understanding "Curved" Spaces: Intuition

**Key question:** What makes rotation space different from translation space?

**Translation ($\mathbb{R}^3$):**

- Move in any direction as far as you want
- You never come back to where you started
- Space is "flat" — like an infinite plane

**Rotation ($\mathrm{SO}(3)$):**

- Rotate around any axis
- After 360°, you're back where you started!
- Space "wraps around" — it's **compact**

**The Problem**

You can't flatten a compact space into $\mathbb{R}^n$ without tearing or overlapping.

# Key Topology Terms (For Engineers)

**Compact:**

- A space is **compact** if it is "bounded and closed"
- Intuition: you can't go to infinity; everything fits in a finite region
- Example: a sphere is compact; a plane is not

**Homeomorphic:**

- Two spaces are **homeomorphic** if you can stretch/bend one into the other (no tearing or gluing)
- Intuition: same "shape" from a topology perspective
- Example: a coffee cup and donut are homeomorphic (both have one hole)

**Diffeomorphic:**

- Like homeomorphic, but the stretching must be smooth (differentiable)
- Intuition: same smooth structure, so calculus works the same way
- For this course: "diffeomorphic" $\approx$ "smoothly equivalent"

# Topology Example: The Cylinder

**Consider:** A cylinder $S^1 \times \mathbb{R}$

**One axis is "flat":**

- Move up/down forever
- Never return to start
- Like $\mathbb{R}$

**One axis is "curved":**

- Go around the cylinder
- After one loop, back to start
- Like $S^1$

**Key insight:** You can parameterize the flat direction with a single coordinate that works everywhere. But the curved direction needs angle wrapping or multiple charts!

# Why Minimal Representations Fail: Topology

## Theorem (Topological Obstruction)

$\mathrm{SO}(3)$ *is diffeomorphic to* $\mathbb{RP}^3$ *(real projective 3-space), not* $\mathbb{R}^3$*.*
*Any 3-parameter global chart must have at least one singularity.*

**Why?**

- $\mathrm{SO}(3)$ is **compact**: all rotations fit in a bounded region
- $\mathbb{R}^3$ is **not compact**: extends to infinity
- No continuous bijection between them can exist!

## This Is Fundamental

Gimbal lock, MRP singularities, etc. are **unavoidable** for any 3-parameter representation. It's topology, not a failure of cleverness!

# Quaternions: Definition

**Unit quaternion:**

$$q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$$

with $\|q\|^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$

**Quaternion multiplication rules:**

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$$

**From axis-angle:**

$$q = \cos(\theta/2) + \sin(\theta/2)(a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k})$$

**Compact notation:** $q = [q_0, \vec{q}]$ where $\vec{q} = [q_1, q_2, q_3]^T$

# Quaternion Operations

**Quaternion multiplication:**

$$p \otimes q = \begin{bmatrix} p_0 q_0 - \vec{p} \cdot \vec{q} \\ p_0 \vec{q} + q_0 \vec{p} + \vec{p} \times \vec{q} \end{bmatrix}$$

**Conjugate:** $q^* = [q_0, -\vec{q}]$
**Inverse:** $q^{-1} = q^* / \|q\|^2 = q^*$ for unit quaternions

**Rotating a vector:**

$$v' = q \otimes v \otimes q^*$$

where $v = [0, \vec{v}]$ is a "pure" quaternion.

**Composition:** $(q_1 \otimes q_2)$ represents $R_1 R_2$

# Quaternions: Advantages

**Why quaternions are popular:**

1. **No gimbal lock:** Smooth everywhere
2. **Compact:** Only 4 parameters (vs 9 for matrices)
3. **Efficient:** Quaternion multiply is cheaper than matrix multiply
4. **Interpolation:** SLERP gives smooth rotations
5. **Numerical stability:** Easy to re-normalize

**Industry adoption:**

- Computer graphics (game engines, animation)
- Aerospace (spacecraft attitude)
- Robotics (motion planning)

# The Quaternion Double Cover

## The Problem

$q$ and $-q$ represent the **same rotation**!

**Mathematically:**

$$(-q) \otimes v \otimes (-q)^* = (-1)^2 \cdot q \otimes v \otimes q^* = q \otimes v \otimes q^*$$

**Why does this happen?**

- Unit quaternions live on $S^3$ (the 3-sphere in 4D)
- $\mathrm{SO}(3)$ is topologically $\mathbb{RP}^3$ (projective space)
- The map $S^3 \to \mathbb{RP}^3$ is 2-to-1

## Not a Bug, a Feature (of Topology)

This is exactly how $S^3$ and $\mathrm{SO}(3)$ are related.

# Quaternion Unwinding Problem

**Scenario:** Attitude control using quaternion feedback

**Naive error:** $q_e = q_d^* \otimes q$
**Problem:** Current $q$ is near $-q_d$ (same physical orientation as $q_d$)

- Error quaternion $q_e \approx [-1, 0, 0, 0]$
- Controller interprets this as "rotate 360°"!
- Vehicle spins unnecessarily ("unwinding")

**Standard workaround:**

1. Check if $q_0 < 0$
2. If so, use $-q$ instead of $q$

**This Creates Discontinuities**

Switching signs introduces jumps in the quaternion trajectory.

# Rotation Matrices: Definition

**Rotation matrix:**

$$R \in \mathrm{SO}(3) = \{R \in \mathbb{R}^{3\times 3} : R^T R = I, \det(R) = 1\}$$

**Properties:**

- Columns are orthonormal basis vectors
- $R^{-1} = R^T$ (inverse equals transpose)
- $\det(R) = 1$ (preserves orientation)
- $\|Rv\| = \|v\|$ (preserves lengths)

**Action on vectors:**

$$v' = Rv$$

Direct matrix-vector multiplication!

# Rotation Matrices: Advantages and Disadvantages

**Advantages:**

- No singularities (globally defined)
- No ambiguity (unique representation)
- Direct action: $v' = Rv$
- Composition: $R_{total} = R_1 R_2$

**Disadvantages:**

- 9 parameters for 3 DOF
- Must maintain 6 constraints: $R^T R = I$, $\det R = 1$
- Numerical drift: $R$ may become non-orthogonal
- How do we do calculus with matrix constraints?

## The Key Question

How do we differentiate, integrate, and optimize over $\mathrm{SO}(3)$?

# The Lie Group Solution

**Instead of fighting the constraints...**

**Key Insight**

Work with the natural mathematical structure!

**The Lie group approach:**

1. Recognize $\mathrm{SO}(3)$ as a **Lie group**
2. Do calculus in the **Lie algebra** $\mathfrak{so}(3) \cong \mathbb{R}^3$
3. Map between them via the **exponential map**

**Benefits:**

- Work directly with rotation matrices (no singularities)
- All calculus happens in a vector space (linear!)
- Constraints are automatically satisfied

# Preview: The Lie Algebra

**The tangent space at identity:** $\mathfrak{so}(3) = T_I \mathrm{SO}(3)$

**What is it?**

$$\mathfrak{so}(3) = \{\Omega \in \mathbb{R}^{3\times 3} : \Omega^T = -\Omega\}$$

The space of $3 \times 3$ skew-symmetric matrices!

**Isomorphism with $\mathbb{R}^3$:**

$$\omega^\wedge = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \quad \leftrightarrow \quad \omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

**Physical meaning:** $\omega$ is angular velocity!

# Preview: The Exponential Map

**From Lie algebra to Lie group:**

$$\exp : \mathfrak{so}(3) \to \mathrm{SO}(3)$$

**Definition (matrix exponential):**

$$\exp(\omega^\wedge) = I + \omega^\wedge + \frac{(\omega^\wedge)^2}{2!} + \frac{(\omega^\wedge)^3}{3!} + \cdots$$

**Closed form (Rodrigues):**

$$\exp(\omega^\wedge) = I + \frac{\sin\|\omega\|}{\|\omega\|}\omega^\wedge + \frac{1-\cos\|\omega\|}{\|\omega\|^2}(\omega^\wedge)^2$$

**The Bridge**

This maps angular velocities to rotation matrices!

# Research Example: UAM Safety Verification

**Application:** Urban Air Mobility (eVTOL aircraft)

## The Problem

Verify safety of control systems for vehicles with **coupled rotation and translation**.

**Lie Group Approach on SE(2):**

- State: pose $X \in \mathrm{SE}(2)$ (position + heading)
- Dynamics: $\dot{X} = X\xi^{\wedge}$ (left-invariant)
- **Flow pipes**: Over-approximate reachable sets on the manifold
- Provable collision avoidance guarantees

## Why Lie Groups?

Traditional Euclidean methods fail to capture the geometric structure.
Lie group formulation enables **global safety certificates**.

# Research Example: Inertial Navigation

**Application:** Strapdown Inertial Navigation Systems (SINS)

## The Problem

Integrate high-rate IMU data efficiently while maintaining accuracy.

**Lie Group Preintegration on $\mathrm{SE}_n(3)$:**

- Extended pose group: rotation, velocity, position in one matrix
- Closed-form preintegration (no iterative updates)
- **45% computational savings** over iterative methods
- Exact propagation of uncertainty

## Key Insight

When dynamics are **group affine**, the Lie group structure enables analytical solutions that would be impossible in Euclidean coordinates.

# Research Example: Invariant Kalman Filtering

**Application:** Robot localization and SLAM

## Traditional EKF Problems

- Linearization errors depend on state estimate
- Inconsistency: covariance doesn't match actual errors
- Poor performance with large initial uncertainty

**Invariant EKF (IEKF):**

- Error defined using group structure
- Linearization errors are **state-independent**
- Guaranteed consistency properties
- Works well even with poor initialization

# What You'll Learn in This Course

**Foundations (Weeks 1-4):**

- Groups, manifolds, Lie groups
- SO(2) as the simplest example
- Exponential/logarithm maps
- Wedge/vee, Ad/ad operators

**Core Groups (Weeks 5-12):**

- SE(2): planar motion
- SO(3): 3D rotations
- SE(3): 3D rigid body poses

**Applications (Weeks 13-17):**

- Invariant Extended Kalman Filter
- Equivariant Filter
- Geometric control, dynamics, your projects

# Lecture 1 Summary

**The Problem:**

- Rotations live on a curved 3D space, not $\mathbb{R}^3$
- Any 3-parameter representation has singularities
- Quaternions avoid singularities but have double cover
- Rotation matrices have no singularities but are redundant

**The Solution:**

- Work with the natural Lie group structure
- Do calculus in the Lie algebra (a vector space!)
- Use exponential/logarithm maps to go between them

**Next:** Mathematical foundations — groups and manifolds