# Intelligent Unmanned System Control Practice

## Chapter 4 Vehicle Motion Modeling and Simulation

Author: Xunhua Dai, Central South University

Website: https://faculty.csu.edu.cn/daixunhua

Email: dai.xh@csu.edu.cn

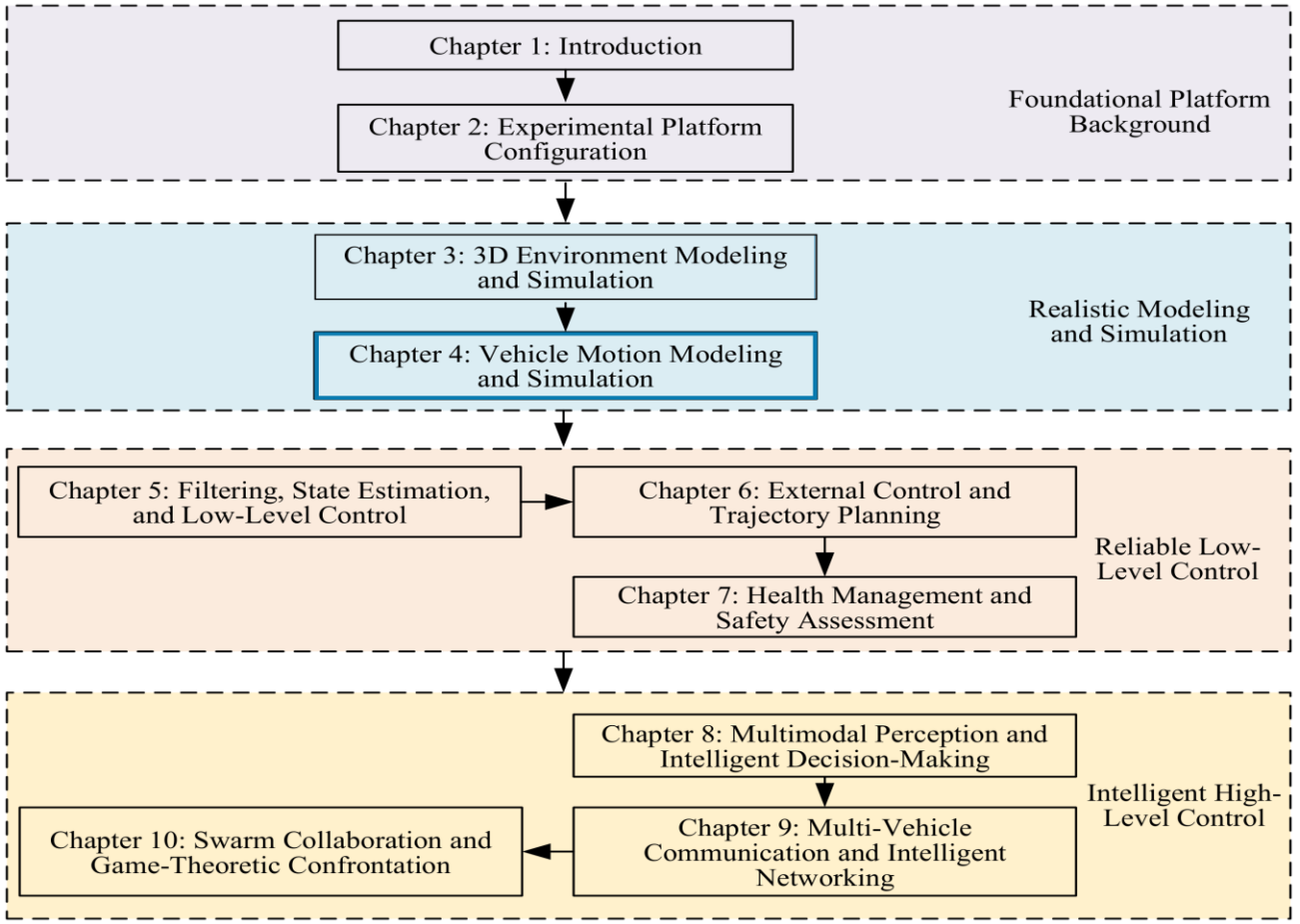
RflySim: http://rflysim.com

# Companion Paper



Dai, Xunhua, Jinhu Tu, and Quan Quan. "RflySim ToolChain: a rapid development and validation toolchain for intelligent unmanned swarm systems." Journal of Systems Engineering and Electronics 36.4 (2025): 1077-1093.

Textbook is coming soon...



## Chapter Learning Objectives

# What You Can Do After Chapter 4

1. Describe vehicle motion with coordinates, states, and mechanical equations
2. Build actuator, environment, and sensor submodels
3. Implement models; configure, drive, and read data with a software development kit (SDK)
4. Measure and identify parameters; validate with independent data and transfer models

**Implement physical equations as a model, then use data to establish where it applies.**

## Chapter Outline

# Modeling Fundamentals and Kinematics

Chapter 4
Vehicle Motion Modeling and Simulation

**4.1 Modeling Fundamentals and Kinematics**
4.2 Rigid-Body Dynamics and Solvers
4.3 Propulsion, Forces, and Observations
4.4 Trim, Linearization, and Dynamics
4.5 Multicopter Modeling and Simulation
4.6 Measurement and Static Identification
4.7 Dynamic Identification and Refinement
4.8 Typical Vehicles and Extensions
4.9 Modeling Tools and Deployment
4.10 API and SDK Modeling Interfaces
4.11 Examples and Integrated Practice
4.12 Exercises and Course Projects

Unit 1 textbook: 4.1.1.1–2, 4.1.2.1–4, 4.1.3.1–2; model types: also 4.2.2.1

## 5 Units in This Section

**4.1.1 Modeling Methods and a Unified Framework**
Textbook: see Unit 1 details below the chapter list

**4.1.2 Frames, Rotation Matrices, and Euler Angles**
Textbook: Sec. 4.1.4.1

**4.1.3 Quaternions and Representation Conversion**
Textbook: Sec. 4.1.4.1

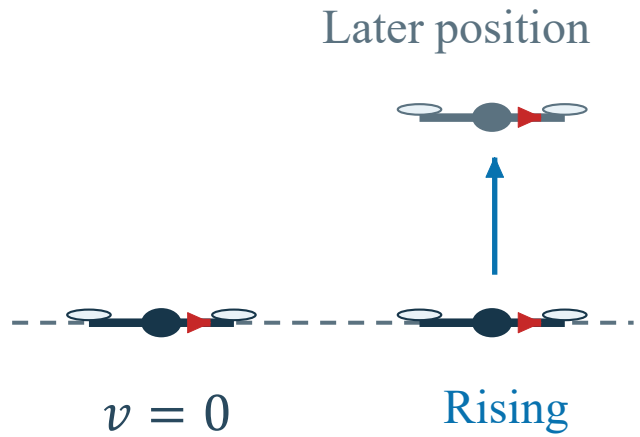
**4.1.4 Attitude Kinematics**
Textbook: Sec. 4.1.4.1

**4.1.5 Position, Mounting Points, and Frame Conventions**
Textbook: Sec. 4.1.4.1

### 4.1.1 Modeling Methods and a Unified Framework: Core Concepts

# A Motion Model Connects Inputs, States, and Outputs

- **Model and parameters:** given thrust, initial state, and $\boldsymbol{\lambda}$ (e.g., mass and inertia), compute position, velocity, and attitude. The trajectory is a result.
- **State:** at equal height, one vehicle rests and another rises. Their later heights differ. Prediction must retain velocity and other needed state information.
- **Inputs and outputs:** $\mathbf{u}$ contains four specified rotor thrusts. State $\mathbf{x}$ gives outputs $\mathbf{y}$ (height and attitude); measurements also require an observation model.



Equal height; different velocities

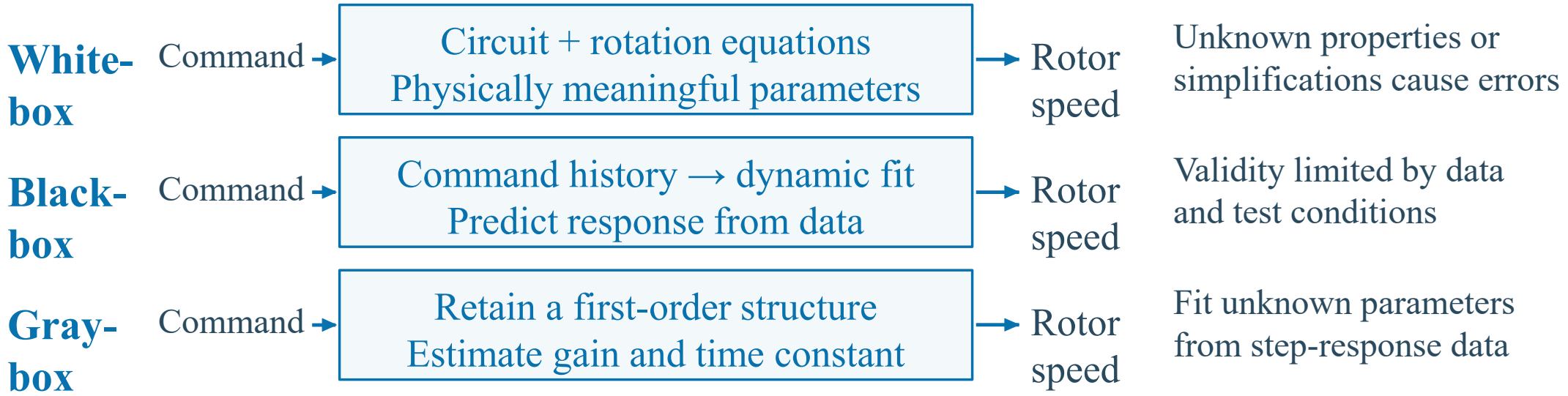
$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}; \boldsymbol{\lambda}), \qquad \mathbf{y} = h(\mathbf{x}, \mathbf{u}; \boldsymbol{\lambda})$$

- **Notation:** a dot over $\mathbf{x}$ denotes a time derivative; $f$ gives the state rate and $h$ the output.

**The state carries the past; inputs drive change; the model links both to outputs.**

### 4.1.1 Modeling Methods and a Unified Framework: System Model Types

## How White-, Black-, and Gray-Box Models Combine Physics and Data




- **Choosing for the same plant:** known structure, available data, and target conditions determine the method. White-, black-, and gray-box describe model sources, not increasing accuracy levels.

Same motor, same inputs and outputs, different sources of knowledge

**The modeling method determines how knowledge and data combine, not an accuracy grade.**

### 4.1.1 Modeling Methods and a Unified Framework: Tasks and Boundaries

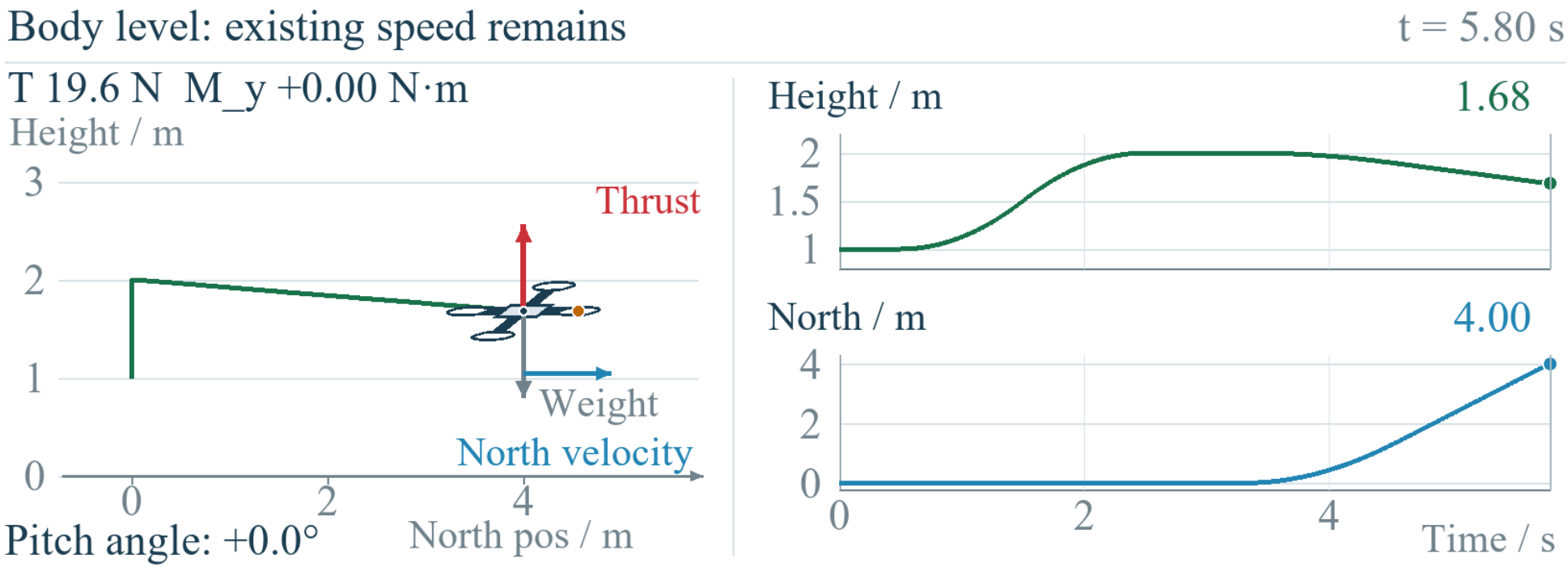
# Playback, Motion Prediction, and Closed-Loop Validation

- **Geometry and playback:** given position, attitude, and time, 3D software displays flight. The predefined trajectory does not explain why throttle changes cause acceleration.
- **Motion prediction:** actuator inputs determine rotor speed, thrust, net force and torque, and rigid-body response. Position and attitude are outputs, so trajectories need not be predefined.
- **Closed-loop validation:** the controller keeps adjusting inputs from observation errors. Motion depends jointly on controller, vehicle, environment, and observations; the body model alone does not determine it.
- **Chapter roles:** this section gives thrust to study vehicle response; actuators and control come later. Chapter 5 develops controllers, filters, and control allocation.

**The engineering question determines which dynamics a model retains.**

### 4.1.1 Modeling Methods and a Unified Framework: Motion Preview

# How Four Rotor Thrusts Become Vehicle Motion




Specified rotor thrusts produce climb, descent, tilt, and translation of the same vehicle.

**Given inputs and an initial state, the model advances position, velocity, and attitude.**

## 4.1.1 Modeling Methods and a Unified Framework: Model-Based Design

# Models Connect Design, Implementation, and Physical Tests

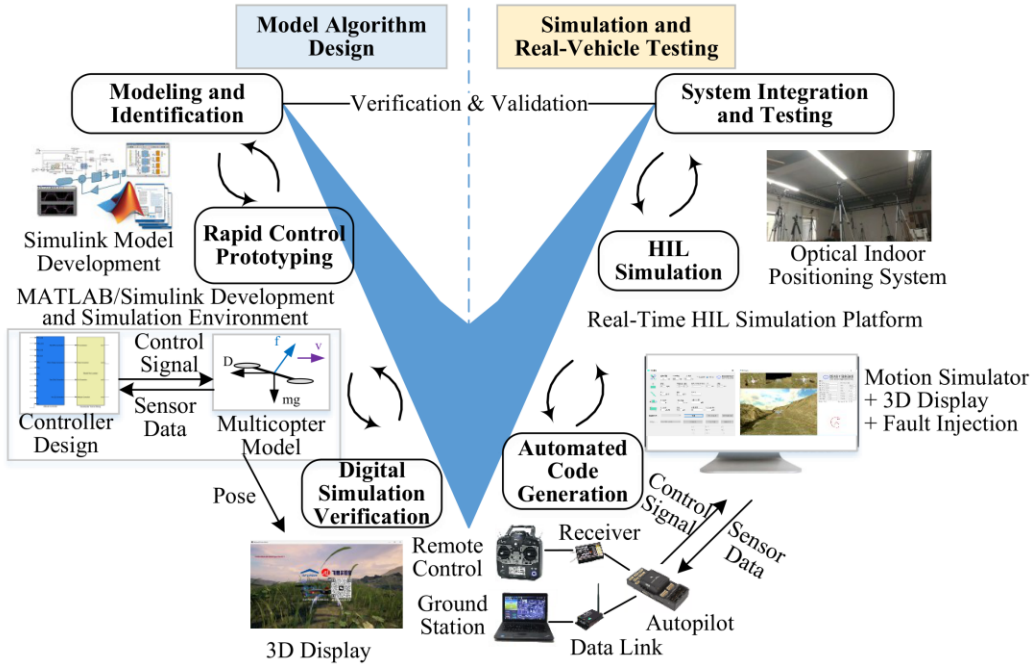



Fig. 4.1 RflySim MBD workflow

- **Model-based design (MBD):** link design and testing through models; use discrepancies to refine structure and parameters.
- **Model stages, left:** modeling and identification → rapid control prototyping → digital simulation verification.
- **Physical stages, right:** automatic code generation → hardware-in-the-loop (HITL; HIL in the textbook) → integration and testing. Feed same-input test discrepancies back to the model.

**Predict → Test the same input → Trace discrepancies**

**Models support design and testing; physical evidence feeds back to refine them.**

## 4.1.1 Modeling Methods and a Unified Framework: System Decomposition and Interconnection

# One System Structure for Different Vehicles

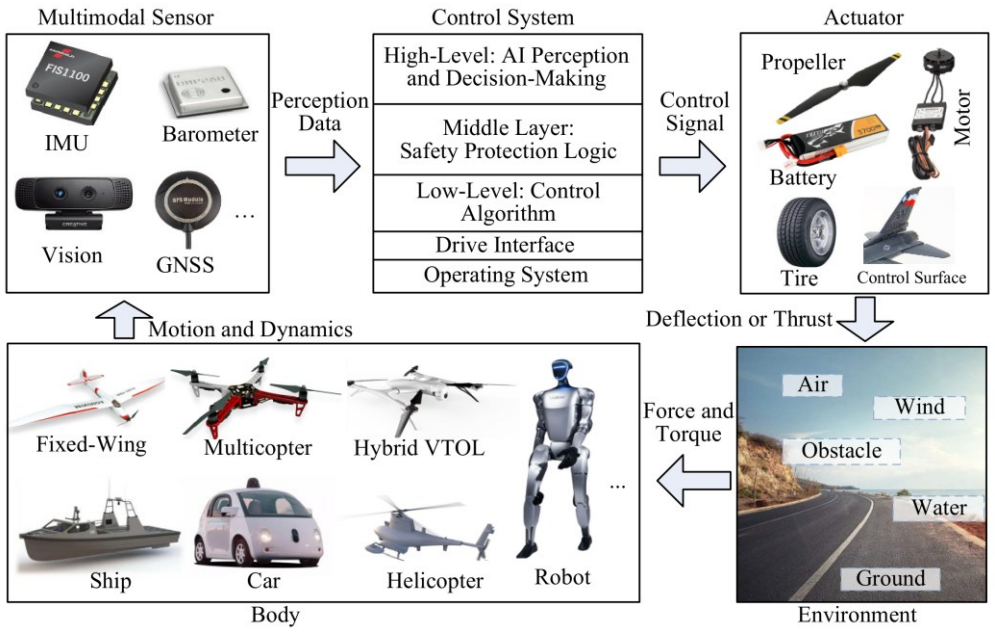



Fig. 4.2 Shared unmanned-vehicle structure

- **Command path:** in Fig. 4.2, control commands reach actuators, producing rotor speeds or deflections, then forces on the body.
- **Feedback path:** trace body motion through environment and sensors. Inflow, contact, and observations change with motion; the same command can give different responses.
- **Common roles, different physics:** aircraft, cars, and boats use the same subsystem roles with lift, tire-force, and hydrodynamic equations. Decisions and control stay in the control system.

**A unified framework fixes subsystem roles; vehicle-specific physics supplies the differences.**

## 4.1.1 Modeling Methods and a Unified Framework: Components and the Equivalent Rigid Body

# From Physical Components to Models and Parameters

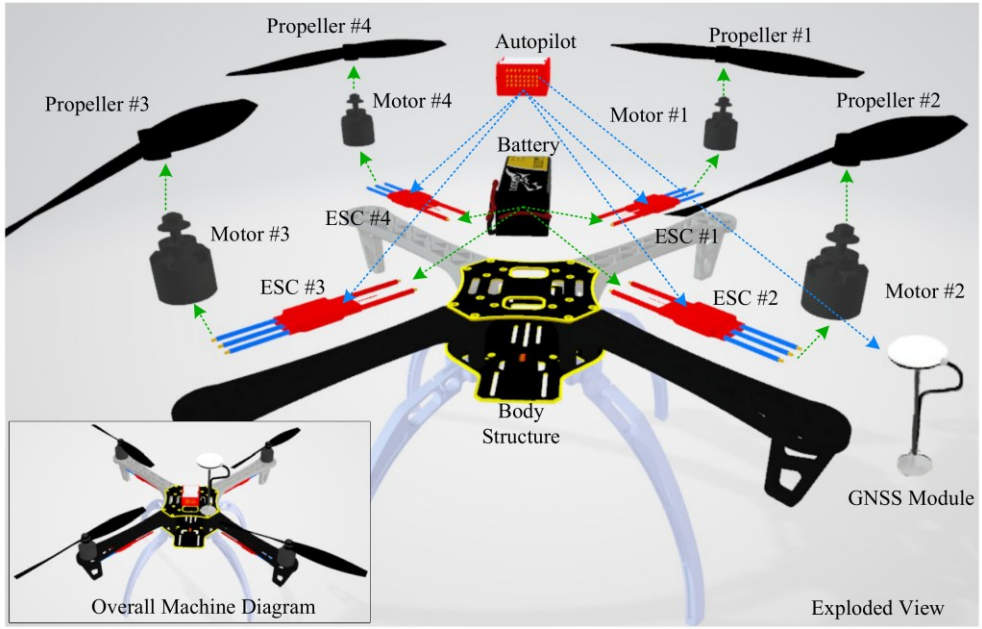



Fig. 4.4 Multicopter components: topological decomposition and connections

- **Propulsion chain:** battery → electronic speed controller (ESC) → motor → propeller. Power and commands set speed and thrust; connections define component inputs and outputs.
- **Rigid-body equivalent:** frame, battery, and payload fixed together give total mass, center of mass, and inertia. Mounting sets moment arms; add states for relative motion.
- **Battery change:** mass and mounting alter inertia; voltage and internal resistance alter propulsion. The same command may give different motion.

**Physical connections define interfaces; the task determines the required component detail.**

### 4.1.1 Modeling Methods and a Unified Framework: Propulsion Chain Components

# Read the Arm: Battery, ESC, Motor, and Propeller

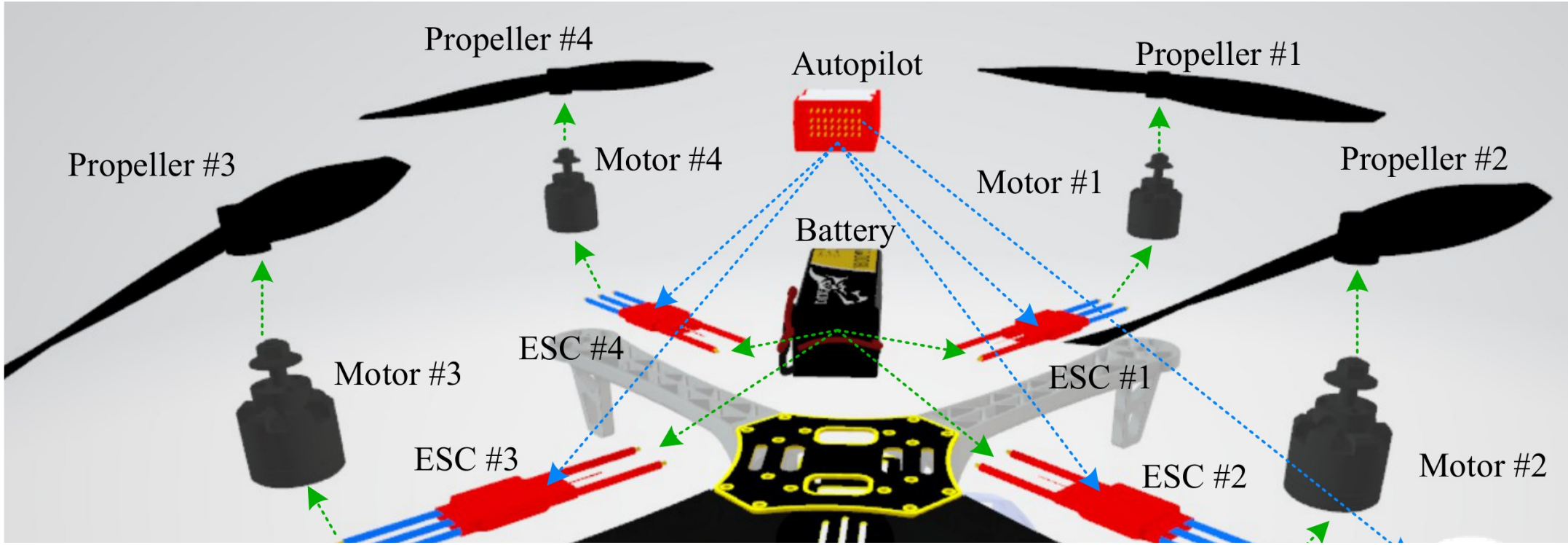



Fig. 4.4 (detail) Blue dashed lines: autopilot commands to ESCs; green dashed lines: power supply and transmission

## 4.1.1 Modeling Methods and a Unified Framework: Standardized Component Models

# How Motion, Energy Use, and Faults Couple Within a Component

**Energy use ↔ Motion ← Faults**

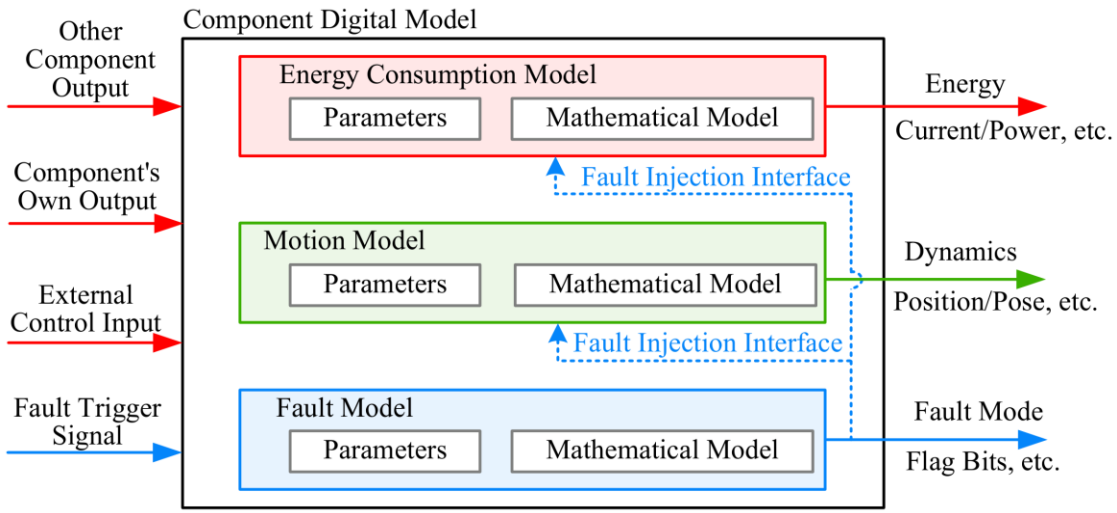



Fig. 4.5 Example standardized component model

- **Motion and energy:** the motion model gives speed or pose; the energy model gives current, power, and energy state. Tasks change loads; the supply limits propulsion.
- **Fault effects:** faults change parameters or physical relations through interfaces. Lower motor efficiency reduces thrust and changes power needed for the same motion.
- **Read shared variables:** energy depletion lowers voltage and changes motion response. Full energy and service-life analysis can be added.

**Shared variables couple a component's motion, energy use, and faults.**

### 4.1.1 Modeling Methods and a Unified Framework: From Components to a Complete Vehicle

# One Topology Assembles Three Kinds of Vehicle Models

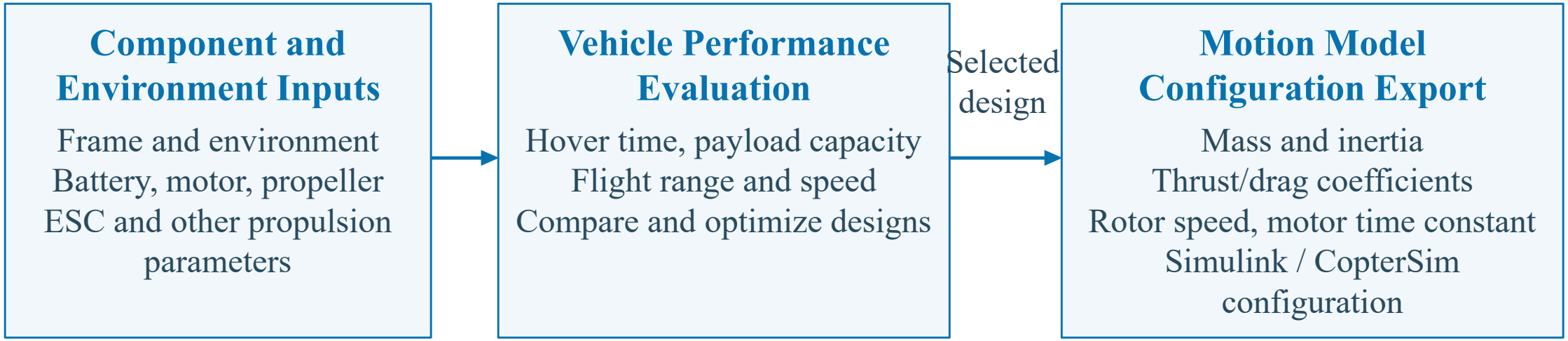
- Standard component models record parameters, inputs/outputs, operating conditions, and validation data. Whole-vehicle response requires connections matching the actual frame, battery, ESC, motor, and propeller topology.
- Topology + energy models: connect component currents, powers, and energy states to assess endurance and supply limits. Changing a battery requires more than updating mass.
- Topology + motion models: connect rotor actions, environment, and rigid-body equations to predict states and support control simulation. This is the chapter's focus.
- Topology + fault models: propagate efficiency loss or failure into forces and energy use to create test conditions. Offline component self-checks compare specified models and data; whole-vehicle certification also needs simulation and test evidence.

**Retain component interfaces and evidence, then assemble the vehicle model needed for the task.**

### 4.1.1 Modeling Methods and a Unified Framework: The FlyEval Textbook Example

# FlyEval: Component Selection, Vehicle Evaluation, and Export

FlyEval supports vehicle design with a standardized component database. Textbook: Sec. 4.1.2.3.

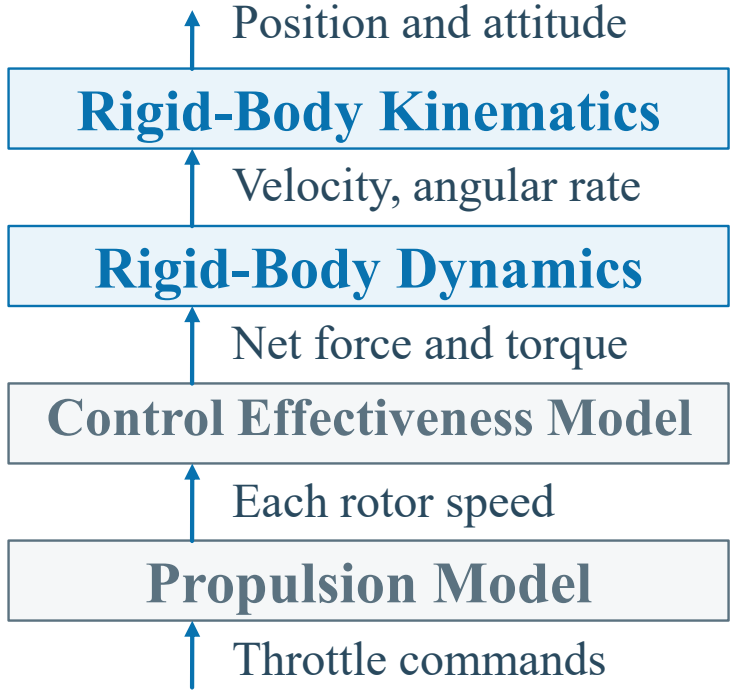



**Select components from constraints:** given target hover time, payload, temperature/altitude, and battery parameters, solve inversely and recommend combinations meeting performance thresholds; then evaluate the designs.

### 4.1.1 Modeling Methods and a Unified Framework: Layered Computation

# Trace Throttle to Motion Through Four Model Layers

- **Propulsion:** throttle sets target motor speed; rotational inertia and motor control make actual speed change gradually.
- **Control effects:** rotor speeds determine thrust; mounting locations and spin directions determine torque about the center of mass.
- **Rigid-body motion:** dynamics determines velocity and angular-rate changes; kinematics uses them to update position and attitude.



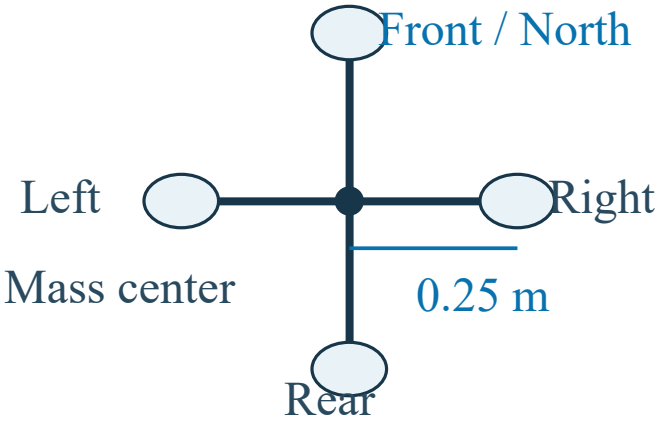

Four layers of a motion model

**Each conversion from throttle to position needs an explainable model layer.**

### 4.1.1 Modeling Methods and a Unified Framework: Parameters and Initial Conditions

# The Running Example: Vehicle, Parameters, and Assumptions

- **Initial state:** ideal "+" quadcopter, level, nose north; height 1 m; zero velocity and angular rate; body origin at mass center.
- **Teaching parameters:** $m = 2\,\mathrm{kg}$, $g = 9.8\,\mathrm{m/s^2}$; all rotor arms are 0.25 m; pitch inertia $J_y = 0.05\,\mathrm{kg\,m^2}$.
- **Inputs and outputs:** specify front, right, rear, and left rotor thrusts; compute position, velocity, attitude, and angular rate. Sec. 4.3 covers command-to-thrust dynamics.



Top view: rotors + moment arms

- **Simplifications:** set the textbook's drag and rotational damping to zero for analytical checks. Rigid frame; fixed mass and inertia; no rotor lag or ground contact; symmetric inputs excite pitch only.
- **Frame:** NED (north-east-down), positive down; origin below the start. $z_D = -1\,\mathrm{m}$, $H = -z_D = 1\,\mathrm{m}$.

**Define the vehicle and conditions, then predict its response to the specified input.**

### 4.1.1 Modeling Methods and a Unified Framework: One-Step Calculation

# Given Thrust, Calculate Acceleration and Update the State

- Given: 1D vertical motion; m=2 kg, g=9.8 m/s², T=21.6 N. NED is positive down. Initial height 1 m, velocity 0; step 0.01 s. Omit motor dynamics, drag, contact, and control.

$$a_D = g - \frac{T}{m} = 9.8 - \frac{21.6}{2} = -1 \text{ m/s}^2$$

$$z_{D,k+1} = z_{D,k} + \Delta t\, v_{D,k} = -1 + 0.01 \times 0 = -1 \text{ m}$$

$$v_{D,k+1} = v_{D,k} + \Delta t\, a_{D,k} = 0 + 0.01 \times (-1) = -0.01 \text{ m/s}$$

- **Synchronous update:** both right-hand sides use the old state at the same instant.
- **This step:** height remains 1 m; velocity becomes −0.01 m/s. Position changes next step.
- **Scope:** vertical motion only; this is the first step of the 1D climb/descent class example.

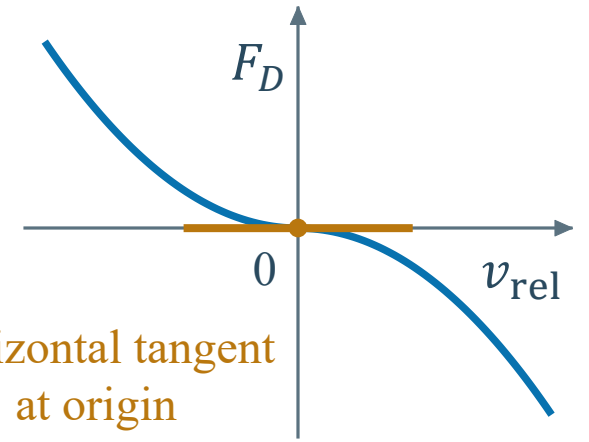
### 4.1.1 Modeling Methods and a Unified Framework: Local Approximation of Quadratic Drag

# Why Quadratic Drag Has No First-Order Term at Calm Hover

- **Operating point:** stationary hover in still air gives zero air-relative velocity. Use 1D relative airspeed $v_{\mathrm{rel}}$ to explain drag.

$$F_D = -k\, v_{\mathrm{rel}} |v_{\mathrm{rel}}| \qquad (k > 0)$$

- **Quadratic law:** for finite constant $k$, drag magnitude scales with airspeed squared; halving airspeed gives one-quarter drag.



1D quadratic drag

- **First order:** use the origin's horizontal tangent; neglect this drag for small airspeed perturbations.
- **Limits:** linear aerodynamic drag can have a first-order term at zero airspeed, and rotational damping at zero angular rate. This example sets both to zero.

**Quadratic drag has no linear term at zero airspeed; assess other aerodynamic effects separately.**

## Chapter Outline

# Modeling Fundamentals and Kinematics

Chapter 4
Vehicle Motion Modeling and Simulation

**4.1 Modeling Fundamentals and Kinematics**
4.2 Rigid-Body Dynamics and Solvers
4.3 Propulsion, Forces, and Observations
4.4 Trim, Linearization, and Dynamics
4.5 Multicopter Modeling and Simulation
4.6 Measurement and Static Identification
4.7 Dynamic Identification and Refinement
4.8 Typical Vehicles and Extensions
4.9 Modeling Tools and Deployment
4.10 API and SDK Modeling Interfaces
4.11 Examples and Integrated Practice
4.12 Exercises and Course Projects

Unit 1 textbook: 4.1.1.1–2, 4.1.2.1–4, 4.1.3.1–2; model types: also 4.2.2.1

## 5 Units in This Section

**4.1.1 Modeling Methods and a Unified Framework**
Textbook: see Unit 1 details below the chapter list

**4.1.2 Frames, Rotation Matrices, and Euler Angles**
Textbook: Sec. 4.1.4.1

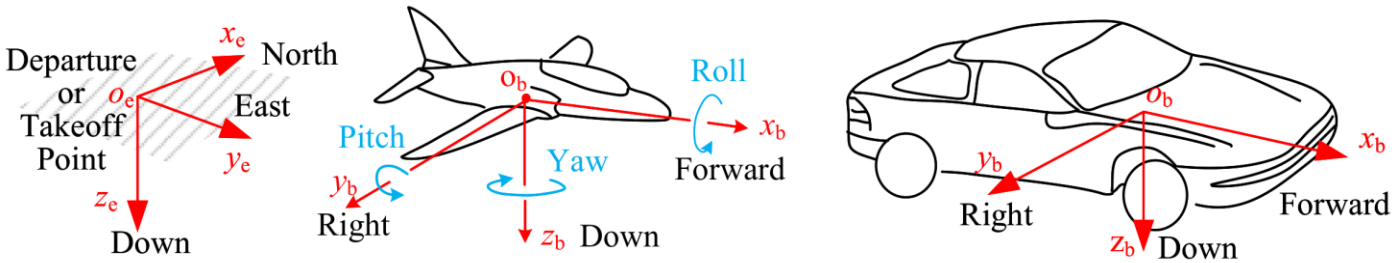
**4.1.3 Quaternions and Representation Conversion**
Textbook: Sec. 4.1.4.1

**4.1.4 Attitude Kinematics**
Textbook: Sec. 4.1.4.1

**4.1.5 Position, Mounting Points, and Frame Conventions**
Textbook: Sec. 4.1.4.1

## 4.1.2 Frames, Rotation Matrices, and Euler Angles: Axes and Origins

# A Coordinate Vector Needs an Origin, Axes, and Units




(a) Earth frame (b) UAV body frame (c) Ground vehicle body frame

Fig. 1.9 Earth and body frames

- **Body frame:** front-right-down (FRD), centered at the center of mass and rotating with the vehicle. Aircraft and cars can share it; forward is north only at certain attitudes.
- **Prerequisite:** Fig. 1.9 is from textbook Chapter 1; slide section 1.4 introduced these frames. Here we calculate and check.

- **Earth frame:** north-east-down (NED), with a specified reference origin and axes north, east, down. At 10 m above the origin, component 3 is −10 m.

- **Scale and units:** the local Earth frame is approximately fixed. The Earth-centered, Earth-fixed (ECEF) origin is at Earth's center. Latitude/longitude are angles; their differences are not displacements in meters.

## 4.1.2 Frames, Rotation Matrices, and Euler Angles: Right-Handed Frames

# The Right-Hand Rule Defines Axes and Positive Rotation

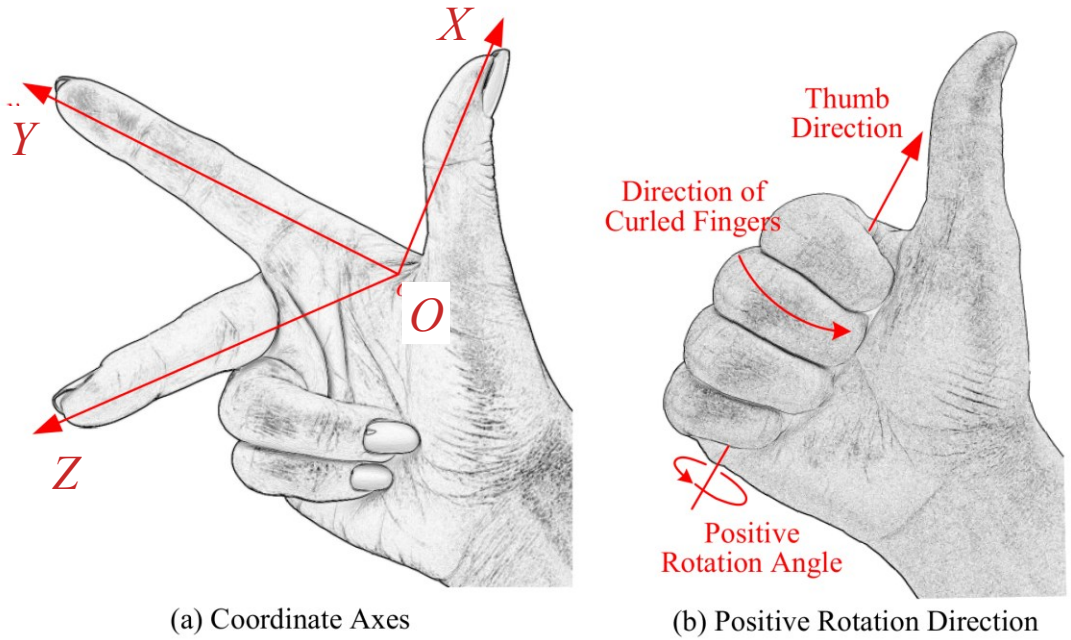



Fig. 1.10 Right-hand axes and positive rotation

- **Right-handed axes:** NED and FRD use axes 1, 2, 3 along thumb, index, and middle fingers. Point the thumb along a positive axis; curled fingers give positive rotation.
- **Attitude motions:** positive roll lowers the right side; positive pitch raises the nose. With the nose north, positive yaw turns it east.
- **Computation:** trigonometric functions use radians; convert degrees with the equation below.

$$\alpha_{\text{rad}} = \alpha_{\text{deg}} \frac{\pi}{180}$$

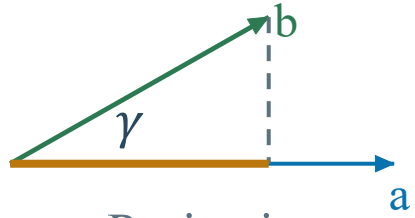
**Once positive axes are fixed, the right-hand rule uniquely determines positive rotation.**

### 4.1.2 Frames, Rotation Matrices, and Euler Angles: Vector Operations

# Dot and Cross Products: Projection and Oriented Area

- **Dot product:** a scalar encoding projection; $\gamma$ is the angle between nonzero vectors in one frame. Perpendicular gives zero; order has no effect.

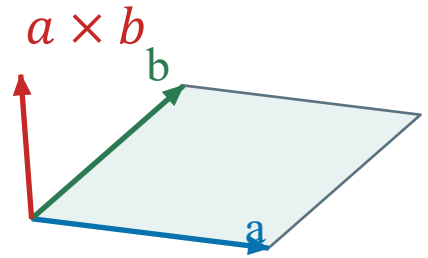
$$\mathbf{a} \cdot \mathbf{b} = a_x b_x + a_y b_y + a_z b_z = \| \mathbf{a} \| \| \mathbf{b} \| \cos\gamma$$



Projection

- **Cross product:** normal to the vectors' plane by the right-hand rule; magnitude = parallelogram area. Parallel gives zero; swapped order reverses direction. Forward × right = down.

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix}, \qquad \mathbf{e}_x \times \mathbf{e}_y = \mathbf{e}_z$$



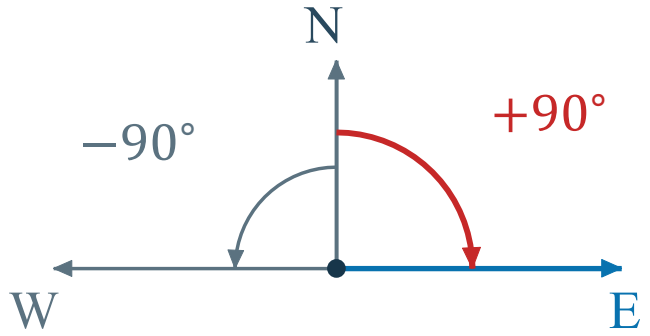
Normal to the plane

**The dot product gives projection; the cross product gives right-hand direction and oriented area.**

### 4.1.2 Frames, Rotation Matrices, and Euler Angles: Yaw Matrix and Direction

# Yaw Turns a Level Nose from North Toward East

- Yaw: about down. Column 3 of $R_z$ and the vertical component stay fixed; north/east mix through sine and cosine.
- **Direction check:** $\psi = 90^\circ$ maps the initially north-pointing forward axis east; $\psi = -90^\circ$ maps it west. The sign sets the physical turn.
- **Inverse:** transposing reverses yaw. If forward maps west, the transformation is reversed; orthogonality alone cannot detect this.




D into the page

Top view: positive yaw turns east

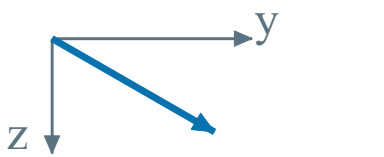
$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad R_z(90^\circ)\mathbf{e}_x = \mathbf{e}_y$$

**Positive yaw follows the right-hand rule about down, turning a north-facing nose toward east.**

### 4.1.2 Frames, Rotation Matrices, and Euler Angles: Single-Axis Rotations

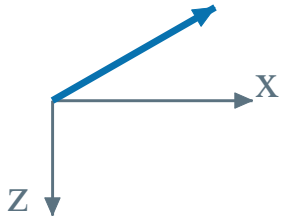
# Roll and Pitch Matrices Follow from Planar Rotation

- **Common structure:** the rotation axis stays fixed; the other two directions rotate in a plane. Diagonal terms use cosine; the right-hand direction determines cross-term signs.
- **Roll:** $R_x$ preserves the forward axis. Positive rotation gives the right axis a positive down component, lowering the right arm.



Right side lowers

- **Pitch:** $R_y$ preserves the right axis. Positive rotation gives the forward axis a negative down component, raising the nose.



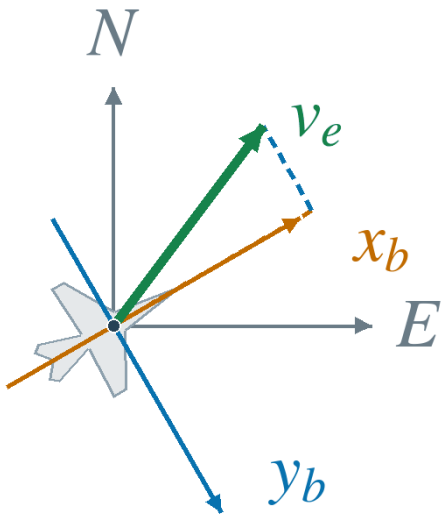
Nose rises

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

**Check single-axis signs with the right-hand rule and one unit axis.**

## 4.1.2 Frames, Rotation Matrices, and Euler Angles: Body to Earth

# Rotation Matrix Columns Give Body-Axis Directions




$$\psi = 60.00^\circ$$

$$v_{x,b} = 4.60$$

$$v_{y,b} = -1.96$$

$$v_{z,b} = 0$$

$$\|v_e\| = \|v_b\| = 5 \text{ m/s}$$

Fixed velocity in changing coordinates (animation)

- **Conditions:** yaw 0°→60°; Earth velocity $\mathbf{v}_e = [4,3,0]^\mathsf{T}$ m/s is fixed. Body components vary; their squared sum stays 25.
- **Inverse:** $R^\mathsf{T}$ gives body components of Earth velocity; $R$ recovers the original vector. Columns of $R$ give body forward, right, and down in Earth coordinates.
- **Shared use:** $R$ transforms velocity and force. Put forces in one frame before adding them.

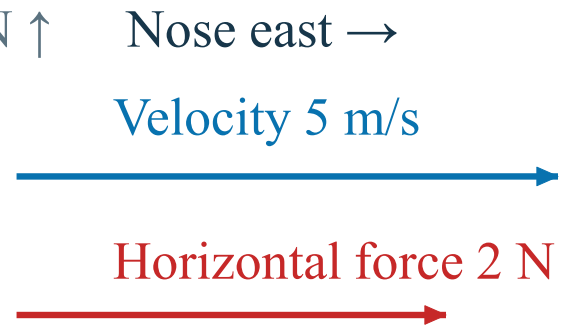
$$\mathbf{a}_e = R\mathbf{a}_b, \qquad R = [\hat{\mathbf{x}}_{b,e} \quad \hat{\mathbf{y}}_{b,e} \quad \hat{\mathbf{z}}_{b,e}]$$

**The body-to-Earth matrix consists of the three body-axis directions in the Earth frame.**

### 4.1.2 Frames, Rotation Matrices, and Euler Angles: A Vector Transformation Example

## One Attitude Matrix Transforms Velocity, Thrust, and Other Vectors

- **Shared mapping:** $R$ takes velocity and force from body to Earth components. Changing coordinates neither changes speed nor increases thrust.
- **Example:** nose east. Body-forward 5 m/s maps to 5 m/s east; 2 N forward plus 10 N upward maps to 2 N east plus 10 N upward.
- **Addition:** first express gravity and thrust in one frame. Force magnitude stays $\sqrt{104}$ N; its units differ from velocity.



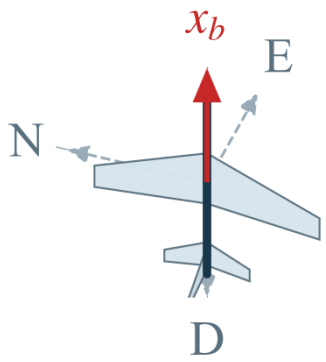

$$R = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{v}_e = (0,5,0)^{\mathrm{T}}\mathrm{m/s}, \quad \mathbf{F}_e = (0,2,-10)^{\mathrm{T}}\mathrm{N}$$

**Use a common frame before summing forces; the same attitude matrix handles velocity and force.**

### 4.1.2 Frames, Rotation Matrices, and Euler Angles: Rotations Do Not Commute

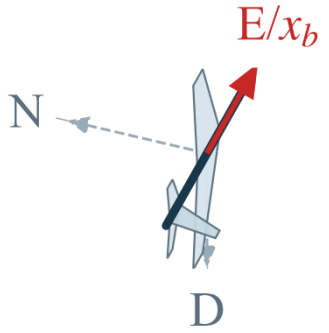
# Exchanging Two Rotations Changes the Final Attitude

$R_z(\alpha)R_y(\beta)$




Nose $x_b = -D$

$R_y(\alpha)R_z(\beta)$




Nose $x_b = E$

$\alpha = 90^\circ, \quad \beta = 90^\circ$

Rotation order changes attitude (animation)

- **Intrinsic turns:** left, about initial $z$ then current $y$; right, about initial $y$ then current $z$. Each turn is 90°.
- **Result:** the first turn moves the second axis; the left nose ends up, the right nose east.
- **Matrix convention:** $R$ maps body components to Earth; factors act right to left on column vectors.

$$R_z(90^\circ)R_y(90^\circ)\mathbf{e}_x = -\mathbf{e}_z$$

$$R_y(90^\circ)R_z(90^\circ)\mathbf{e}_x = \mathbf{e}_y$$

**Rotations about different axes cannot be freely reordered; multiaxis checks reveal order errors.**

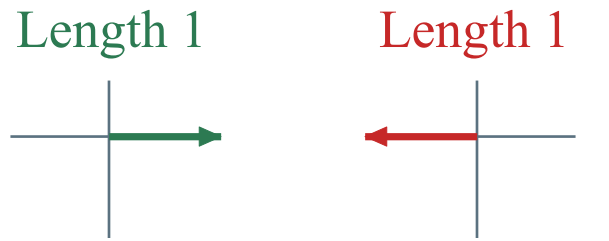
### 4.1.2 Frames, Rotation Matrices, and Euler Angles: Orthogonality and Inverse Transformation

# A Valid Rotation Matrix Can Still Be Used Backward

- **Geometry:** body axes are mutually perpendicular, each of length 1, so $R^{\mathsf{T}}R = I$. Condition $\det R = 1$ preserves handedness and excludes reflections.

$$R^{\mathsf{T}}R = I, \quad \det R = 1, \quad R^{-1} = R^{\mathsf{T}}, \quad \mathbf{a}_b = R^{\mathsf{T}}\mathbf{a}_e, \quad \| \mathbf{a}_e \| = \| \mathbf{a}_b \|$$

- **Inverse:** an orthogonal matrix's inverse is its transpose. Multiply Earth components by $R^{\mathsf{T}}$ to recover body components without changing length.
- **Direction:** $R$ and $R^{\mathsf{T}}$ both pass orthogonality and round-trip tests. For a nose facing east, forward must map east. Both right-hand results have length 1; only one direction is correct.

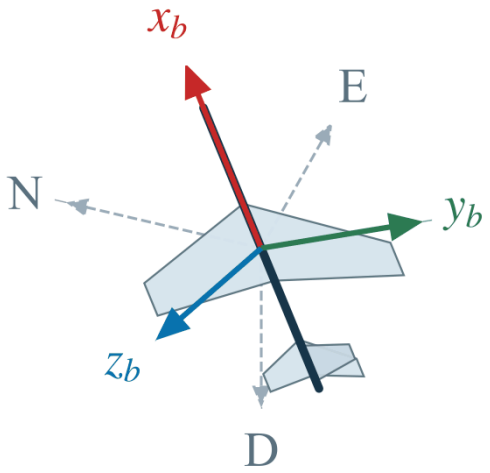



Equal length can hide reversal

**Orthogonality checks matrix validity; a known direction checks how it is used.**

## 4.1.2 Frames, Rotation Matrices, and Euler Angles: The Z–Y–X Convention

# Define Euler Angles Together with Their Rotation Order




Successive rotations about current axes (animation)

**Complete**

1. Yaw $\psi = 45.0°$
   Current $z_b$
2. Pitch $\theta = 30.0°$
   Current $y_b$
3. Roll $\phi = 20.0°$
   Current $x_b$

- **Intrinsic ZYX:** yaw $\psi$ about current down, then pitch $\theta$ about current right, then roll $\phi$ about current forward.
- **Animation:** yaw 45°, pitch 30°, roll 20°; earlier turns move the axes for later turns.
- **Reading the equation:** positive follows the right-hand rule; $R$ maps body components to Earth. The rightmost factor acts first on a column vector.

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

**Euler angles need a rotation order, positive directions, and a transformation direction.**

### 4.1.2 Frames, Rotation Matrices, and Euler Angles: The Full Z–Y–X Matrix

# Read the Full Attitude Matrix Column by Column

- **Derivation:** multiply the single-axis matrices as $R_z R_y R_x$ to map body directions to Earth. Write $c_\alpha = \cos\alpha$, $s_\alpha = \sin\alpha$, using full angles, not half-angles.

Nose direction

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

Down projections

Read columns and rows

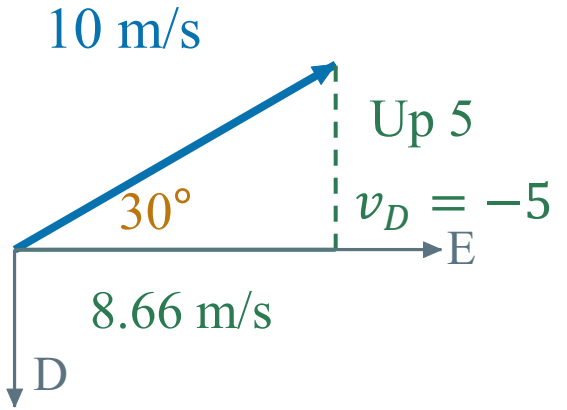
- **Column 1:** the forward direction depends only on yaw and pitch. Rolling about the nose does not change its direction, but does change right and down.
- **Row 3:** down components of the three body axes. Term $-\sin\theta$ keeps the nose-up sign for positive pitch. Zero attitude gives identity; pure yaw gives $R_z(\psi)$ as before.

### 4.1.2 Frames, Rotation Matrices, and Euler Angles: A Complete Calculation

# A Two-Angle Example Links Attitude and Velocity Direction

- **Given:** $\phi = 0, \theta = 30°, \psi = 90°$, nose east and 30° up. The center-of-mass velocity relative to Earth is specified as 10 m/s along the nose.

$$R = \begin{bmatrix} 0 & -1 & 0 \\ \sqrt{3}/2 & 0 & 1/2 \\ -1/2 & 0 & \sqrt{3}/2 \end{bmatrix}, \qquad \mathbf{v}_e = R \begin{bmatrix} 10 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 5\sqrt{3} \\ -5 \end{bmatrix} \text{m/s}$$




Velocity projection: nose east and up

- **Components:** $5\sqrt{3} \approx 8.66$ m/s east and 5 m/s upward; speed remains 10 m/s.
- **Physical scope:** velocity along the nose is given in this problem. Attitude only transforms components; initial conditions and dynamics still determine actual velocity.

**Two nonzero angles check nose direction, vertical sign, and velocity magnitude together.**

### 4.1.2 Frames, Rotation Matrices, and Euler Angles: Inverse Conversion and Quadrants

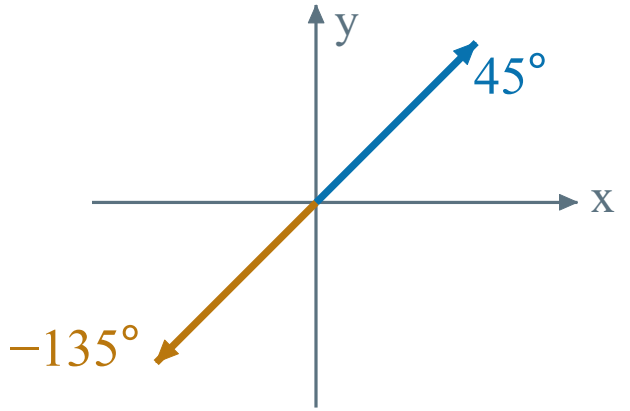
# Recovering Euler Angles Requires a Defined Range

- **Principal pitch:** invert $R_{31} = -\sin\theta$ with arcsine. Restrict $-\pi/2 < \theta < \pi/2$ to recover angles consistent with Z–Y–X in a nonsingular interval.

$$\theta = \arcsin(-R_{31}), \quad \phi = \operatorname{atan2}(R_{32}, R_{33}), \quad \psi = \operatorname{atan2}(R_{21}, R_{11})$$

- **Quadrants:** both arrows have y/x=1, but differ by 180°. atan2 retains the signs of both x and y.

- **Limits:** the pitch range excludes $\pm 90°$. Euler angles become error-sensitive near these endpoints; periodic equivalents also represent the same rotation.



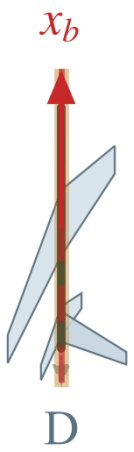
Same ratio, different quadrants

**Choose angle ranges before recovering Euler angles from matrix entries.**

### 4.1.2 Frames, Rotation Matrices, and Euler Angles: Pitch Singularity

# Euler Singularity Is a Loss of Coordinate Independence

Collinear axes, opposite directions




Yaw axis D

Roll axis $x_b$

$\psi = 30.00^\circ$

$\theta = 90.00^\circ$

$\phi = 30.00^\circ$

Line gap 0.00°

$x_b = -D$

Same attitude

$$R(0^\circ, 90^\circ, 0^\circ) = R(30^\circ, 90^\circ, 30^\circ)$$

Collinear rotation axes at 90° pitch (animation)

- **Singularity:** at pitch $\theta = \pm 90^\circ$, roll and yaw lose independent meaning. Attitude still exists; the three-angle coordinates are nonunique.
- **Animation:** the ocher yaw axis and red-dashed roll axis are collinear at 90° pitch, with opposite positive directions. Increase both angles by 30°: the body stays fixed.

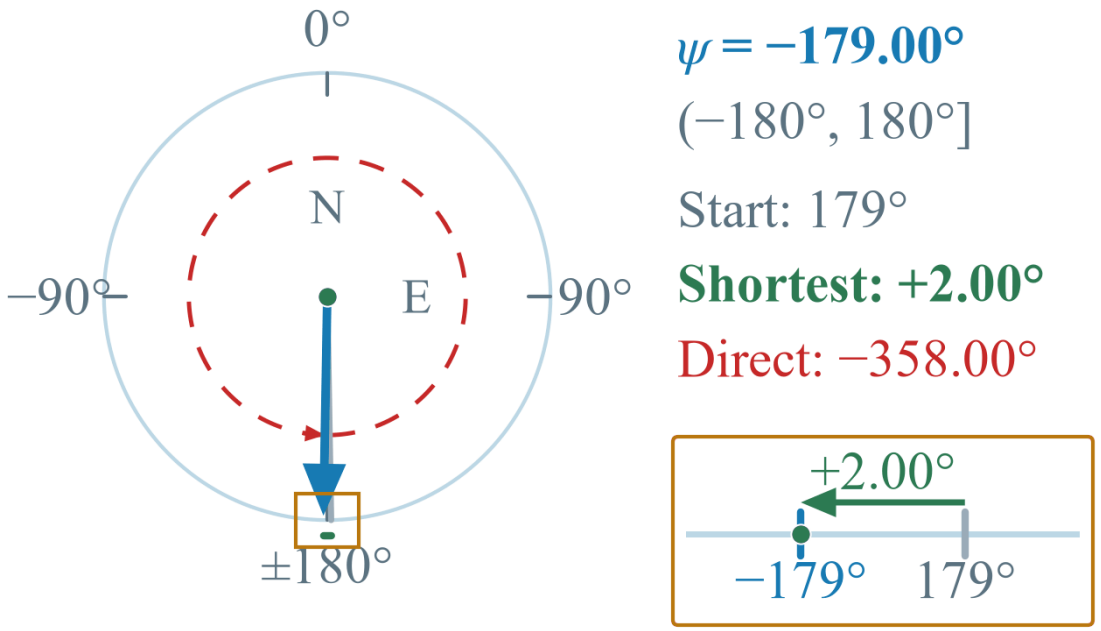
$$R(0^\circ, 90^\circ, 0^\circ) = R(30^\circ, 90^\circ, 30^\circ)$$

- **Nearby** angles may jump; use a unit quaternion or rotation matrix for continuous propagation.

**Euler singularity belongs to the representation; the aircraft's physical attitude still exists.**

### 4.1.2 Frames, Rotation Matrices, and Euler Angles: Periodic Angles and Shortest Differences

# Angle Wrapping and Euler Singularity Are Different Issues




Yaw wraps during a continuous turn (animation)

- **Periodicity:** in a $\pm 180^\circ$ display interval, yaw changes sign at the boundary while physical direction can remain continuous.
- **Read the figure:** $179^\circ$ to $-179^\circ$ is a positive $2^\circ$ turn; direct subtraction gives $-358^\circ$.
- **Computation:** atan2 uses sine and cosine for the shortest signed angle difference. Interval wrapping is not Euler singularity.

$$\Delta\psi = \operatorname{atan2}(\sin d,\ \cos d)$$

$$d = \psi_2 - \psi_1$$

**Compute angle differences periodically; a displayed jump may just be interval wrapping.**

## Chapter Outline

# Modeling Fundamentals and Kinematics

Chapter 4
Vehicle Motion Modeling and Simulation

**4.1 Modeling Fundamentals and Kinematics**
4.2 Rigid-Body Dynamics and Solvers
4.3 Propulsion, Forces, and Observations
4.4 Trim, Linearization, and Dynamics
4.5 Multicopter Modeling and Simulation
4.6 Measurement and Static Identification
4.7 Dynamic Identification and Refinement
4.8 Typical Vehicles and Extensions
4.9 Modeling Tools and Deployment
4.10 API and SDK Modeling Interfaces
4.11 Examples and Integrated Practice
4.12 Exercises and Course Projects

Unit 1 textbook: 4.1.1.1–2, 4.1.2.1–4, 4.1.3.1–2; model types: also 4.2.2.1

## 5 Units in This Section

**4.1.1 Modeling Methods and a Unified Framework**
Textbook: see Unit 1 details below the chapter list

**4.1.2 Frames, Rotation Matrices, and Euler Angles**
Textbook: Sec. 4.1.4.1

**4.1.3 Quaternions and Representation Conversion**
Textbook: Sec. 4.1.4.1

**4.1.4 Attitude Kinematics**
Textbook: Sec. 4.1.4.1

**4.1.5 Position, Mounting Points, and Frame Conventions**
Textbook: Sec. 4.1.4.1

### 4.1.3 Quaternions and Representation Conversion: From Representation to Kinematics

# Three Attitude Representations Share the Work

**Euler angles $(\phi, \theta, \psi)$**

- 3 intuitive angles
- Set and display attitude
- Opening animation: pitch $\approx -10.3°$
- Singular at pitch $\pm 90°$

**Rotation matrix $R$**

- 9 elements
- Orthogonal; determinant 1
- Map velocity and thrust
- Only 3 degrees of freedom

**Unit quaternion q**

- 4 components; unit norm
- $\mathbf{q}$ and $-\mathbf{q}$ give one attitude
- Integrate without Euler singularity
- Keep product order

Initial Euler angles → Quaternion $\mathbf{q}$ → Rotation matrix $R$ → Earth-frame thrust/velocity

Quaternion $\mathbf{q}$ → Euler-angle display

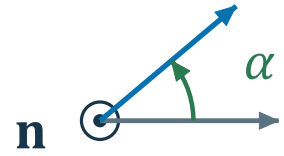
One attitude: storage, computation, and display

**Read attitude with Euler angles, transform coordinates with matrices, and propagate with unit quaternions.**

### 4.1.3 Quaternions and Representation Conversion: Axis–Angle and Unit Norm

# A Unit Quaternion Encodes an Axis and a Half-Angle

- **Convention:** scalar-first Hamilton quaternion $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$, representing the body-to-Earth attitude just like $R$.




Rotation axis and angle

$$\mathbf{q} = \begin{bmatrix} q_0 \\ \mathbf{q}_v \end{bmatrix} = \begin{bmatrix} \cos(\alpha/2) \\ \mathbf{n}\sin(\alpha/2) \end{bmatrix}, \quad \|\mathbf{n}\| = 1,\ q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$$

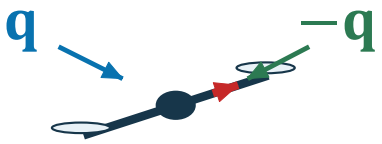
- **Axis–angle:** $\mathbf{n}$ is a unit rotation axis and $\alpha$ the right-hand angle. The scalar is the half-angle cosine; the vector is the half-angle sine along $\mathbf{n}$.
- **Constraint:** four components encode three independent degrees of freedom; $[1,0,0,0]^T$ is zero rotation. Unit quaternions stay continuous across Euler singularities; product order still matters.

**Four constrained quaternion components represent three attitude degrees of freedom.**

### 4.1.3 Quaternions and Representation Conversion: Quaternion Constraints and Continuity

# Sign Equivalence and Normalization Solve Different Problems

- **Two values, one attitude:** $\mathbf{q}$ and $-\mathbf{q}$ describe the same orientation. Handle this sign equivalence before interpolation or adjacent-sample comparison; a full sign flip is not a sudden rotation.



Equivalent signs

$$R(\mathbf{q}) = R(-\mathbf{q}), \quad \mathbf{q}_{\text{unit}} = \mathbf{q}_{\text{raw}}/\|\mathbf{q}_{\text{raw}}\| \ (\|\mathbf{q}_{\text{raw}}\| > 0), \quad (1,0,0,1)^{\mathsf{T}} \mapsto (1,0,0,1)^{\mathsf{T}}/\sqrt{2}$$

- **Restore unit length:** divide a nonzero quaternion by its norm. A zero quaternion cannot be normalized or represent attitude. Normalizing $[1,0,0,1]^{\mathsf{T}}$ gives positive 90° yaw.
- **Limits:** normalization restores the constraint, not frame direction, product order, or angle accuracy. Account for sign equivalence; a direct four-component difference does not measure attitude error.

**Changing every quaternion sign preserves attitude; normalization only restores unit length.**

### 4.1.3 Quaternions and Representation Conversion: Quaternion Multiplication

## Hamilton Products Combine Scalars, Dot Products, and Cross Products

- **Notation:** $p_0$, $q_0$ are scalars; $\mathbf{p}_v$, $\mathbf{q}_v$ are 3D vectors. Symbol $\otimes$ denotes the Hamilton product, not componentwise multiplication.

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

- **Structure:** scalar part = scalar product minus dot product. Vector part = two scalar–vector products plus $\mathbf{p}_v \times \mathbf{q}_v$.
- **Order:** reversing a cross product changes its sign, so generally $\mathbf{p} \otimes \mathbf{q} \neq \mathbf{q} \otimes \mathbf{p}$. Ordered unit-quaternion products compose rotations in the same order as their rotation matrices.

**Hamilton multiplication combines scalars, dot products, and cross products in a defined order.**

### 4.1.3 Quaternions and Representation Conversion: Rotation Action and Inverse

# Quaternion Sandwiching Matches Matrix Rotation

- **Pure-vector quaternion:** write $\mathbf{a}$ as $[0, a_x, a_y, a_z]^\mathsf{T}$, adding a zero scalar as a calculation container. Vector $\mathbf{a}$ need not have unit length.

$$\mathbf{a}_e = \mathbf{q} \otimes \mathbf{a}_b \otimes \mathbf{q}^*, \quad \mathbf{q}^* = (q_0, -q_1, -q_2, -q_3)^\mathsf{T} = \mathbf{q}^{-1} \ (\|\mathbf{q}\| = 1)$$

- **Convention:** sandwich with $\mathbf{q}$ on the left and $\mathbf{q}^*$ on the right to obtain Earth components of the same vector, equivalent to $\mathbf{a}_e = R\mathbf{a}_b$.
- **Conjugate and inverse:** the conjugate equals the inverse only for unit $\mathbf{q}$. Exchange the two sides for the reverse transformation; vector length is preserved.




$$\mathbf{a}_b = [1,0,0]^T$$

$$\mathbf{a}_e = [0,1,0]^T$$

One vector, two component sets

**Keep the active-rotation product order consistent with the body-to-Earth mapping.**

### 4.1.3 Quaternions and Representation Conversion: Quaternion to Matrix

# Expand a Quaternion into the Full Rotation Matrix

- **Derivation:** expand $\mathbf{q} \otimes [0, \mathbf{a}_b] \otimes \mathbf{q}^*$ and collect coefficients of the three components of $\mathbf{a}_b$ to obtain $R(\mathbf{q})$, mapping body to Earth.

$$R(\mathbf{q}) = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix}$$

- **Condition:** the formula assumes $q_0^2+q_1^2+q_2^2+q_3^2=1$. Substituting a quaternion with norm drift may produce a nonorthogonal matrix.
- **Reading and use:** squared terms give same-axis projections; cross terms combine axes. Zero rotation $[1,0,0,0]^\top$ gives identity. Use this same mapping for velocity, thrust, and observation directions.

**Quaternion and matrix describe one attitude; their transformation directions must agree.**

### 4.1.3 Quaternions and Representation Conversion: Euler Angles to Quaternion

## Compose Euler Rotations into a Quaternion in the Defined Order

- **Composition:** multiply the three single-axis quaternions as $\mathbf{q} = \mathbf{q}_z(\psi) \otimes \mathbf{q}_y(\theta) \otimes \mathbf{q}_x(\phi)$, matching matrix order $R_z R_y R_x$.

$$\mathbf{q} = \begin{bmatrix} C_\psi C_\theta C_\phi + S_\psi S_\theta S_\phi \\ C_\psi C_\theta S_\phi - S_\psi S_\theta C_\phi \\ C_\psi S_\theta C_\phi + S_\psi C_\theta S_\phi \\ S_\psi C_\theta C_\phi - C_\psi S_\theta S_\phi \end{bmatrix}$$

- **Half-angles:** $C_\alpha = \cos(\alpha/2)$, $S_\alpha = \sin(\alpha/2)$. Four rows: $q_0$, $q_1$, $q_2$, $q_3$, in order.
- **Use:** set an intuitive initial attitude with Euler angles, then store it as a quaternion. Three zero angles give $[1,0,0,0]^\mathsf{T}$. Using full-angle functions instead gives the wrong attitude.

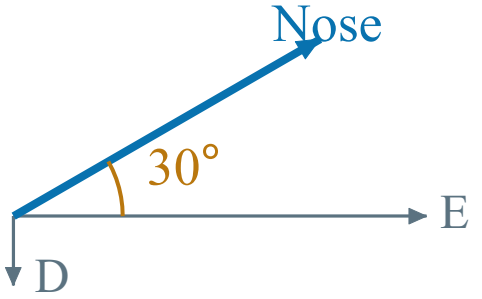
**Half-angle sines and cosines form the quaternion in the established product order.**

### 4.1.3 Quaternions and Representation Conversion: Substitution

# The Quaternion for the Same Two-Angle Attitude

- **Continue the example:** $\phi = 0, \theta = 30°, \psi = 90°$, so the half-angles are 0, 15°, and 45°. Substitution into the full half-angle formula gives:

$$\mathbf{q} = \frac{1}{4}\begin{bmatrix} \sqrt{3}+1 \\ -(\sqrt{3}-1) \\ \sqrt{3}-1 \\ \sqrt{3}+1 \end{bmatrix} \approx \begin{bmatrix} 0.683013 \\ -0.183013 \\ 0.183013 \\ 0.683013 \end{bmatrix}$$



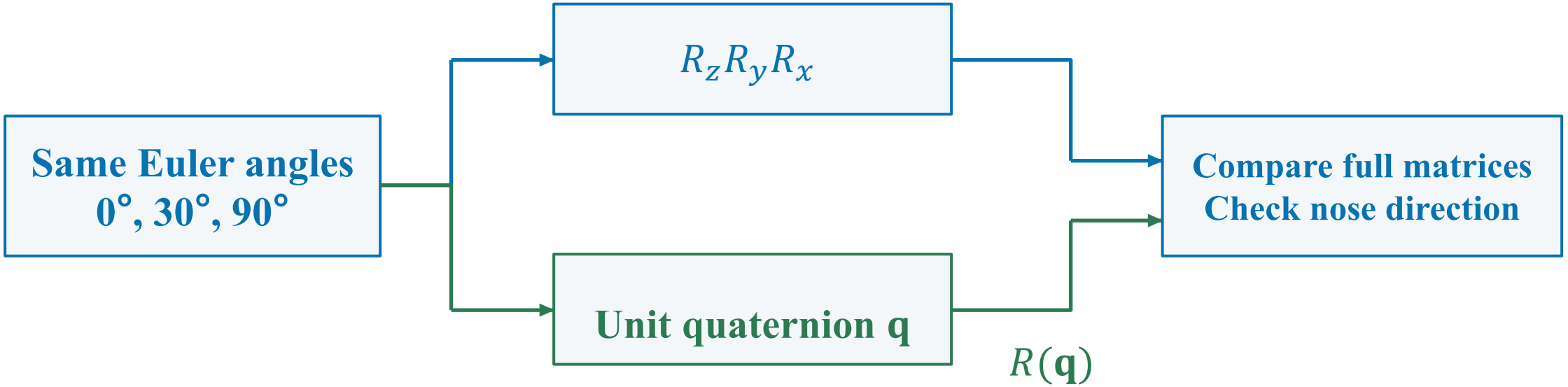
Same attitude: east and nose up

- **Components:** first the scalar, then three vector entries. Negative $q_1$ and positive $q_2$ follow from product order. Exact radicals have squared sum 1; decimals are approximations.
- **Geometry:** nose east and 30° up. Convert back to a matrix to verify agreement with Euler angles.

**Two nonzero angles jointly check half-angle substitution and quaternion product order.**

### 4.1.3 Quaternions and Representation Conversion: Diagram and Checks

# One Input, Two Paths to the Same Attitude




Norm and orthogonality check validity; matrix and direction checks establish the same attitude.

Two-path cross-check of attitude conversion functions

**Matching full matrices and physical directions establishes consistent attitude conventions.**

### 4.1.3 Quaternions and Representation Conversion: Substitution

## Cross-Check One Attitude with Euler Angles and Quaternions

- **Same conditions:** $\phi = 0, \theta = 30°, \psi = 90°$, nose east and 30° up. Substitute half-angles 0°, 15°, and 45°:

$$\mathbf{q} = 1/4\,[\sqrt{3}+1,\ -(\sqrt{3}-1),\ \sqrt{3}-1,\ \sqrt{3}+1]^{\mathsf{T}}$$

Euler angles → $R_z R_y R_x$

$$\mathbf{q} \approx [0.683013,\ -0.183013,\ 0.183013,\ 0.683013]^{\mathsf{T}}$$

Euler angles → $\mathbf{q}$ → $R$

$$R(\mathbf{q}) = \begin{bmatrix} 0 & -1 & 0 \\ \sqrt{3}/2 & 0 & 1/2 \\ -1/2 & 0 & \sqrt{3}/2 \end{bmatrix} = R_z(90°)R_y(30°)R_x(0)$$

- **Cross-check:** norm and orthogonality establish validity. Matching matrices and forward-axis directions establish compatible conventions. Forward 10 m/s maps to 8.66 m/s east and 5 m/s up.

## Chapter Outline

# Modeling Fundamentals and Kinematics

Chapter 4
Vehicle Motion Modeling and Simulation

**4.1 Modeling Fundamentals and Kinematics**
4.2 Rigid-Body Dynamics and Solvers
4.3 Propulsion, Forces, and Observations
4.4 Trim, Linearization, and Dynamics
4.5 Multicopter Modeling and Simulation
4.6 Measurement and Static Identification
4.7 Dynamic Identification and Refinement
4.8 Typical Vehicles and Extensions
4.9 Modeling Tools and Deployment
4.10 API and SDK Modeling Interfaces
4.11 Examples and Integrated Practice
4.12 Exercises and Course Projects

Unit 1 textbook: 4.1.1.1–2, 4.1.2.1–4, 4.1.3.1–2; model types: also 4.2.2.1

## 5 Units in This Section

**4.1.1 Modeling Methods and a Unified Framework**
Textbook: see Unit 1 details below the chapter list

**4.1.2 Frames, Rotation Matrices, and Euler Angles**
Textbook: Sec. 4.1.4.1

**4.1.3 Quaternions and Representation Conversion**
Textbook: Sec. 4.1.4.1

**4.1.4 Attitude Kinematics**
Textbook: Sec. 4.1.4.1

**4.1.5 Position, Mounting Points, and Frame Conventions**
Textbook: Sec. 4.1.4.1

### 4.1.4 Attitude Kinematics: Rotation Axes and Rate Mapping

# Why Body Angular Rates Differ from Euler-Angle Rates

- **Meaning:** $p,\ q,\ r$ are angular-velocity components on current body forward/right/down axes. ZYX roll $\phi$, pitch $\theta$, yaw $\psi$ do not all rotate about the final body axes.

- **Why map:** project and add Euler-rate terms in one body frame to get angular velocity. Invert the attitude-dependent map; do not integrate gyro components directly into Euler angles.

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

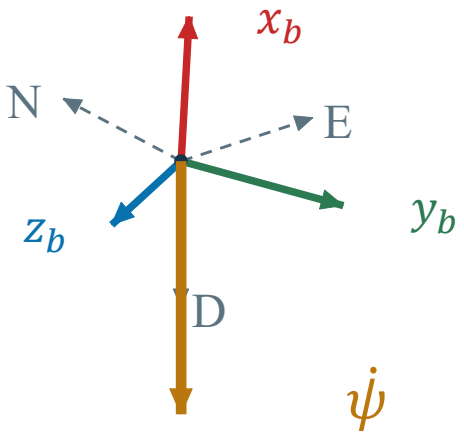
- **Conditions:** radians and nonzero $\cos\theta$. Approximate identity requires both roll and pitch to be small. A level single-axis case does not establish direct correspondence at general attitudes.

**Euler-angle rates describe attitude parameters; map actual angular velocity through the geometry.**

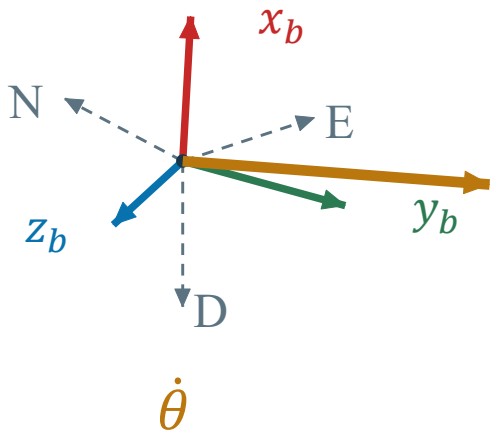
### 4.1.4 Attitude Kinematics: Diagram and Checks

# Euler-Rate Axes Differ from Gyro Body Axes

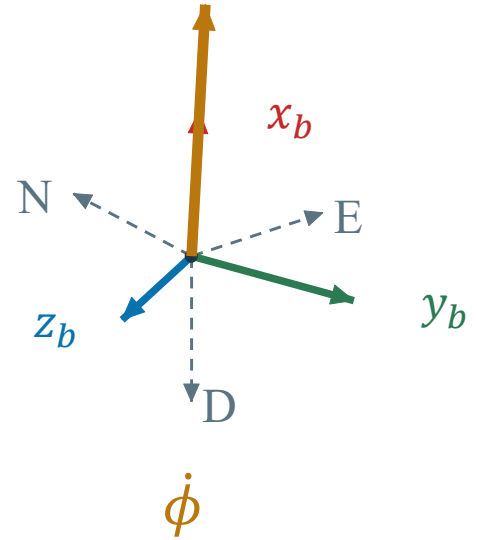
**Yaw rate: Earth down axis**




**Pitch rate: intermediate right axis**




**Roll rate: final forward axis**




Legend: Euler rotation axis  Final body forward/right/down  NED (gray dashed)

Three instantaneous axes for ZYX Euler-angle rates

**Express rotation contributions in one frame before summing actual angular velocity.**

### 4.1.4 Attitude Kinematics: Euler-Rate Example and Singularity

# The Same Angular Rate at Level and Tilted Attitudes

- **Common input:** body rate $[p, q, r] = [0,0,1]\,\mathrm{rad/s}$. At level attitude $\phi = \theta = 0$, Euler rates are $[0,0,1]\,\mathrm{rad/s}$. This is only the special case where the mapping becomes identity.
- **Tilted case:** $\phi = 30°, \theta = 60°$ give roll/pitch/yaw rates 1.5, −0.5, 1.732051 rad/s by row substitution. All Euler angles change; angular velocity is only along body down.

$$\dot{\phi} = \cos 30° \tan 60° = 1.5, \quad \dot{\theta} = -\sin 30° = -0.5$$

$$\dot{\psi} = \cos 30° / \cos 60° = 1.732051\ \mathrm{rad/s}$$

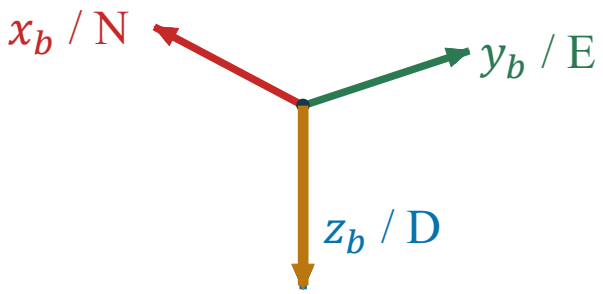
- **Singularity:** as $\theta$ nears $\pm 90°$, tangent/cosine denominators make Euler rates sensitive or undefined; body rates stay finite. Integrate large attitudes with quaternions; output Euler angles as needed.

**Tilted cases reveal the error in directly integrating $p$, $q$, $r$; singularity belongs to the representation.**

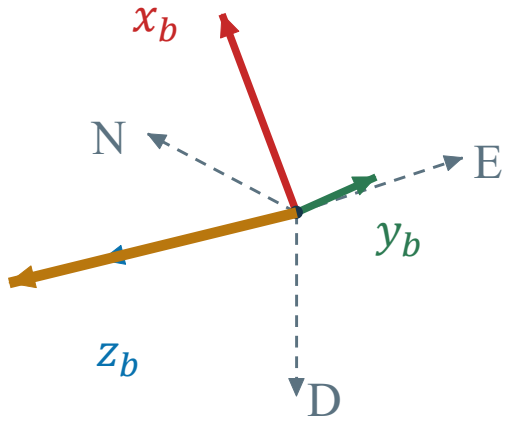
### 4.1.4 Attitude Kinematics: Diagram and Checks

# One Body-Frame Input, Different Euler Rates

**Level:** $\phi = 0, \theta = 0$




**Tilted:** $\phi = 30°, \theta = 60°$




**Both inputs:** $[p, q, r] = [0,0,1]$ **rad/s**

$[\dot{\phi}, \dot{\theta}, \dot{\psi}] = [0,0,1]$

$[\dot{\phi}, \dot{\theta}, \dot{\psi}] = [1.5, -0.5, 1.732]$

Same angular-rate components along each body's down axis

**The same $p, q, r$ give different Euler rates through an attitude-dependent mapping.**

### 4.1.4 Attitude Kinematics: Continuous Rotation with the Same Input

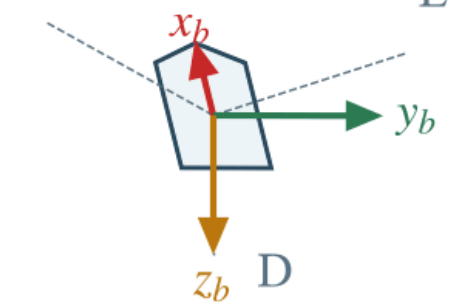
# Constant Body Rates Still Give Attitude-Dependent Euler Rates

Full result

$t = 0.600$ s

Same body-frame input $[p, q, r] = [0,0,1]$ rad/s

A: $\phi_0 = 0°, \theta_0 = 0°$




Current Euler rates

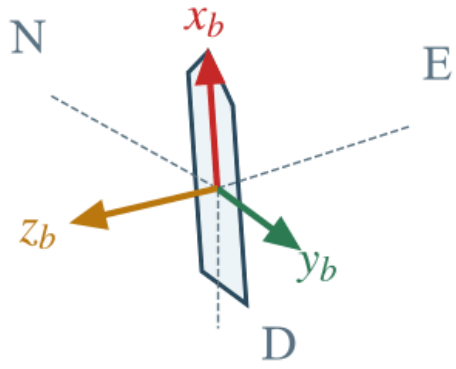
$\dot{\phi} = 0.000$ rad/s

$\dot{\theta} = 0.000$ rad/s

$\dot{\psi} = 1.000$ rad/s

Initial rates [0.0, 0.0, 1.000] rad/s

B: $\phi_0 = 30°, \theta_0 = 60°$




Current Euler rates

$\dot{\phi} = 0.370$ rad/s

$\dot{\theta} = -0.849$ rad/s

$\dot{\psi} = 0.645$ rad/s

Initial rates [1.5, -0.5, 1.732] rad/s

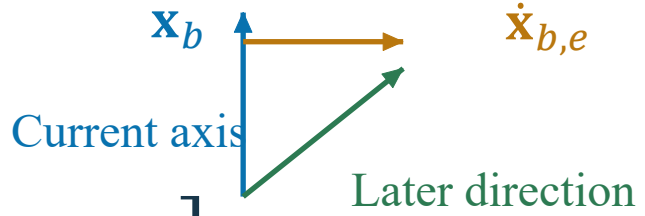
Two initial attitudes rotate about their own down axes; recompute Euler rates from the current attitude

**The same $p, q, r$ give different Euler rates through an attitude-dependent mapping.**

### 4.1.4 Attitude Kinematics: Cross-Product Matrix and Attitude Derivative

# Rotating Body Axes Determine the Matrix Derivative

- **Geometry:** a rotating body-fixed axis has derivative angular velocity crossed with the axis. Positive yaw makes the forward-axis derivative point along current right.



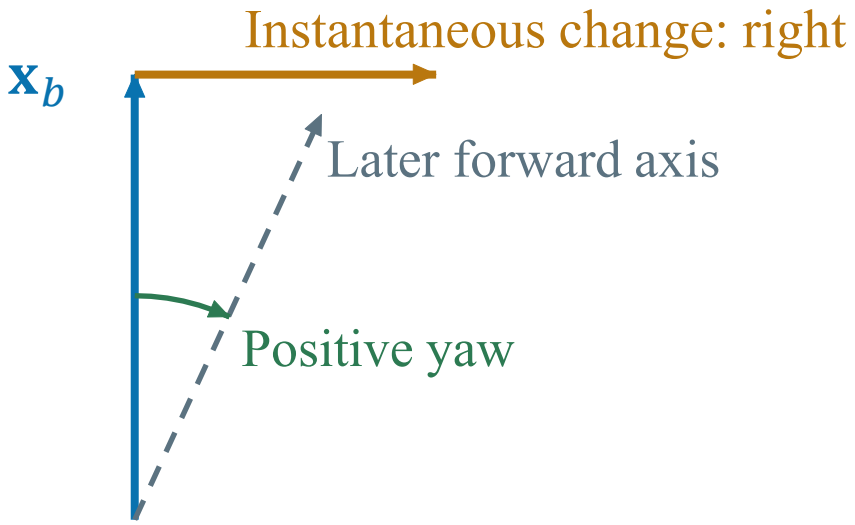

$$\dot{R} = R[\boldsymbol{\omega}_b]_\times, \qquad [\boldsymbol{\omega}_b]_\times = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}$$

- **Product direction:** angular velocity uses body components. Compute body-axis change first, then apply $R$ for Earth components; this requires right multiplication.
- **Constraints:** $R^\mathsf{T}R = I$, $\det R = 1$. Nine entries encode only three rotational degrees of freedom. Numerical integration error may destroy orthogonality.

**Attitude derivatives turn angular velocity into continuous axis change while preserving rotation constraints.**

### 4.1.4 Attitude Kinematics: Diagram and Checks

## A Rotating Axis Has an Instantaneously Tangential Derivative




- **Geometry:** $\boldsymbol{\omega}_b = r\mathbf{z}_b$, hence $\boldsymbol{\omega}_b \times \mathbf{x}_b = r\mathbf{y}_b$.
- **Coordinates:** compute axis change in the body frame, then multiply by $R$ to obtain $\dot{R} = R[\boldsymbol{\omega}_b]_\times$.

Top view; FRD and NED align at this instant

Forward axis and tangential change under positive yaw

**The cross product gives axis-direction change; R then maps it to Earth coordinates.**

### 4.1.4 Attitude Kinematics: Hamilton Convention and Component Formula

## Quaternion Derivatives Convert Angular Velocity to Attitude Change

- **Convention:** $\mathbf{q} = [q_0, \mathbf{q}_v]$ maps body to Earth; scalar $q_0$ comes first and $\mathbf{q}_v$ is a 3D vector. Use Hamilton multiplication and body-frame $\boldsymbol{\omega}_b$, with attitude increments multiplied on the right.

$$\mathbf{p} \otimes \mathbf{q} = (p_0 q_0 - \mathbf{p}_v \cdot \mathbf{q}_v,\ p_0 \mathbf{q}_v + q_0 \mathbf{p}_v + \mathbf{p}_v \times \mathbf{q}_v)$$

- **Meaning:** scalar part uses a dot product; vector part a cross product. Substitute pure angular-velocity quaternion $[0, \boldsymbol{\omega}_b]$ for four attitude derivatives. Bold $\mathbf{q}$ is not rate $q$.

$$\dot{\mathbf{q}} = 1/2\, \mathbf{q} \otimes [0, \boldsymbol{\omega}_b^{\mathsf{T}}]^{\mathsf{T}} = 1/2 \begin{bmatrix} -\boldsymbol{\omega}_b^{\mathsf{T}} \mathbf{q}_v \\ q_0 \boldsymbol{\omega}_b + \mathbf{q}_v \times \boldsymbol{\omega}_b \end{bmatrix}$$

- **Constraint and use:** $\mathbf{q}^{\mathsf{T}}\mathbf{q} = 1$ leaves three rotational degrees of freedom; $\mathbf{q}$ and $-\mathbf{q}$ give one attitude. No Euler pitch singularity; code must maintain norm and verify rotation direction.

**Fix rotation direction, component order, and multiplication convention to define the quaternion derivative.**

### 4.1.4 Attitude Kinematics: Single-Axis Rotation and Finite Increments

# Exact Constant-Rate Attitude and a One-Step Update

- **Single-axis problem:** start at identity and rotate about body down at constant $\pi/2$ rad/s. After 1 s the nose turns north to east: an actual 90° rotation.

$$\mathbf{q}(t) = [\cos(\pi t/4), 0, 0, \sin(\pi t/4)]^{\mathsf{T}}$$

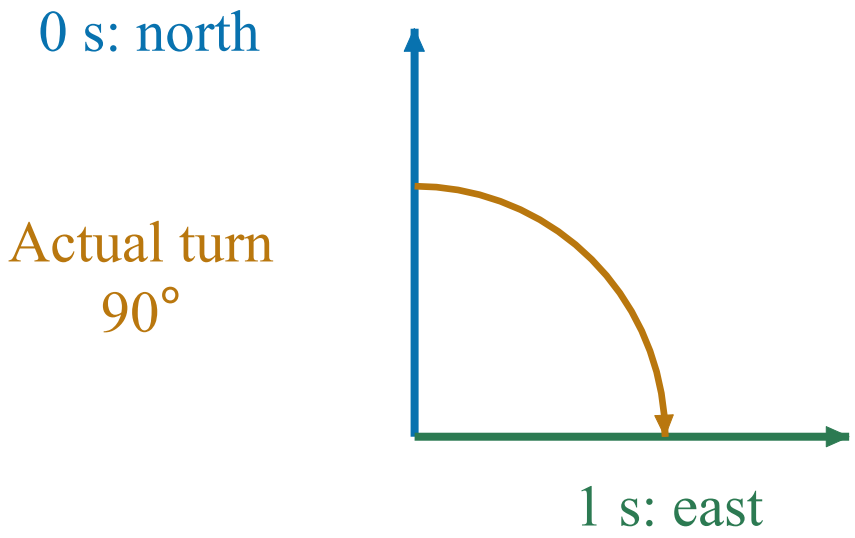
- **One step:** if body angular velocity is constant over step $h$, use $\mathbf{q}_{k+1} = \mathbf{q}_k \otimes \delta\mathbf{q}$. At zero angular velocity use the identity increment, avoiding normalization of a zero vector.

$$\delta\mathbf{q} = [\cos(\| \boldsymbol{\omega}_b \| h/2),\ \mathbf{n}^{\mathsf{T}}\sin(\| \boldsymbol{\omega}_b \| h/2)]^{\mathsf{T}}, \quad \mathbf{n} = \boldsymbol{\omega}_b/\| \boldsymbol{\omega}_b \|$$

**The rotation angle sets the quaternion half-angle; constant rate gives a unit attitude increment directly.**

### 4.1.4 Attitude Kinematics: Diagram and Checks

# The Nose Turns 90°; the Quaternion Uses a 45° Half-Angle




**Substitute half-angle: 90° / 2 = 45°**

$$\mathbf{q}(1) = \left[\frac{\sqrt{2}}{2}, 0, 0, \frac{\sqrt{2}}{2}\right]^{\mathrm{T}}$$

- **Constant rate:** $\omega_z = \pi/2$ rad/s. Use this unit-increment rule for nearly constant angular velocity within a step.

Actual rotation angle and half-angle play different roles

**The half-angle belongs to the representation formula; the physical nose still turns the full 90°.**

### 4.1.4 Attitude Kinematics: Why Body-Frame Increments Multiply on the Right

# Two Rotations About Different Axes Cannot Be Reordered

- **Sequence:** roll 90° about $x$, then 90° about current body $z$. The first turn changes the second axis; right-multiply the body-frame increment, matching $R_x R_z$.

$$\mathbf{q}_x = [c, c, 0, 0]^\mathsf{T}, \quad \mathbf{q}_z = [c, 0, 0, c]^\mathsf{T}, \quad c = \sqrt{2}/2$$

$$\mathbf{q}_x \otimes \mathbf{q}_z = [0.5, 0.5, -0.5, 0.5]^\mathsf{T}, \quad R_x R_z \mathbf{e}_x = \mathbf{e}_z$$

- **Compare:** the correct product has third component −0.5 and final forward axis down. Reversing the product gives +0.5 and forward east. Both have unit norm but different attitudes.

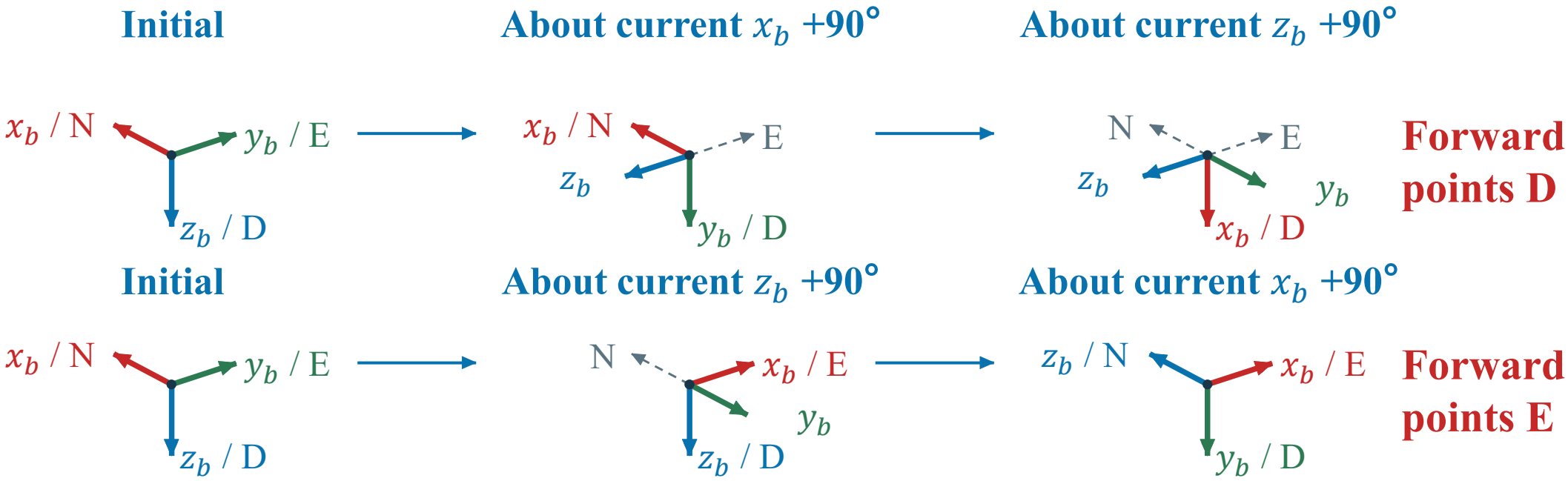
$$\mathbf{q}_z \otimes \mathbf{q}_x = [0.5, 0.5, 0.5, 0.5]^\mathsf{T}, \quad R_z R_x \mathbf{e}_x = \mathbf{e}_y$$

- **Why this detects errors:** a single-axis turn from identity may give identical left and right products. Two nonparallel axes expose a valid formula used in the wrong rotation order.

**Product order determines the axes of an increment; correct norm does not guarantee correct direction.**

### 4.1.4 Attitude Kinematics: Diagram and Checks

# Each Turn Changes the Next "Body Axis"




Two body-frame rotation sequences; colors mark forward, right, and down axes

**Exchanging body-frame increments changes the final forward direction from down to east.**

## Chapter Outline

# Modeling Fundamentals and Kinematics

Chapter 4
Vehicle Motion Modeling and Simulation

**4.1 Modeling Fundamentals and Kinematics**
4.2 Rigid-Body Dynamics and Solvers
4.3 Propulsion, Forces, and Observations
4.4 Trim, Linearization, and Dynamics
4.5 Multicopter Modeling and Simulation
4.6 Measurement and Static Identification
4.7 Dynamic Identification and Refinement
4.8 Typical Vehicles and Extensions
4.9 Modeling Tools and Deployment
4.10 API and SDK Modeling Interfaces
4.11 Examples and Integrated Practice
4.12 Exercises and Course Projects

Unit 1 textbook: 4.1.1.1–2, 4.1.2.1–4, 4.1.3.1–2; model types: also 4.2.2.1

## 5 Units in This Section

**4.1.1 Modeling Methods and a Unified Framework**
Textbook: see Unit 1 details below the chapter list

**4.1.2 Frames, Rotation Matrices, and Euler Angles**
Textbook: Sec. 4.1.4.1

**4.1.3 Quaternions and Representation Conversion**
Textbook: Sec. 4.1.4.1

**4.1.4 Attitude Kinematics**
Textbook: Sec. 4.1.4.1

**4.1.5 Position, Mounting Points, and Frame Conventions**
Textbook: Sec. 4.1.4.1

### 4.1.5 Position, Mounting Points, and Frame Conventions: State and Position Updates

# What the Motion State Stores and How Position Updates

- **Rigid-body state:** position $\mathbf{r}_e$ and velocity $\mathbf{v}_e$ use Earth coordinates; angular velocity $\boldsymbol{\omega}_b$ uses body coordinates; $\mathbf{q}$ stores body-to-Earth attitude. Initial state and forces determine subsequent motion.

$$\mathbf{x} = [\mathbf{r}_e^\top, \mathbf{v}_e^\top, \mathbf{q}^\top, \boldsymbol{\omega}_b^\top]^\top, \quad \mathbf{q}^\top\mathbf{q} = 1$$

| Position: 3 values | Velocity: 3 values | Quaternion: 4 values | Angular rate: 3 values |
|---|---|---|---|

13 stored values; unit-norm constraint leaves 12 independent states

- **Position update:** nose east, moving only body-forward at constant $2\ \mathrm{m/s}$ gives $4\ \mathrm{m}$ eastward in two seconds. After turning, update $R$ before transforming velocity. Actuator dynamics adds further states.

$$\dot{\mathbf{r}}_e = \mathbf{v}_e = R\mathbf{v}_b$$

**Six degrees of freedom describe position and attitude; prediction also needs velocity and angular rate.**

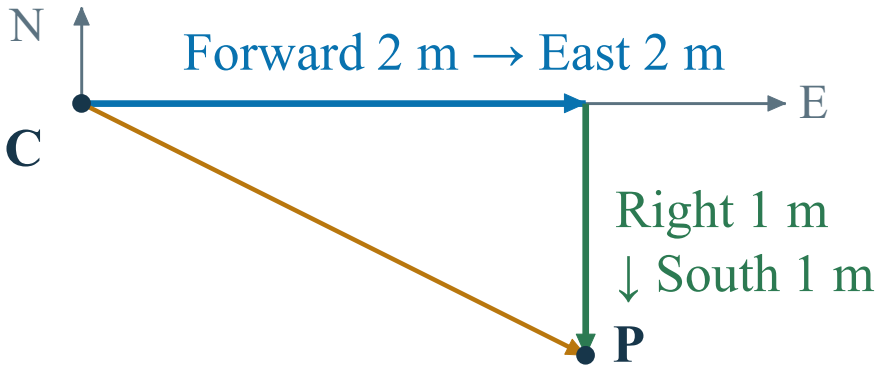
### 4.1.5 Position, Mounting Points, and Frame Conventions: Position Transformation Example

# Rotate a Mounting Offset, Then Add Center-of-Mass Position

- **Given:** nose east; positions in m.

$$\mathbf{r}_{C,e} = [100,200,-10]^T$$

$$\boldsymbol{\rho}_{CP,b} = [2,1,0]^T$$




Mounting-point offset near C

- **Positions:** $C$ is the center of mass and $P$ the mounting point; $\mathbf{r}_{C,e}$ and $\mathbf{r}_{P,e}$ share one Earth-frame origin.

$$\mathbf{r}_{P,e} = \mathbf{r}_{C,e} + R\boldsymbol{\rho}_{CP,b}$$

$$R\boldsymbol{\rho}_{CP,b} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix} \text{m}$$

- **Result:** 2 m forward and 1 m right give 2 m east and 1 m south. Add center-of-mass position: $[99,202,-10]^{\mathrm{T}}$ m.

**Rotation converts directions; translation accounts for the displacement between origins.**

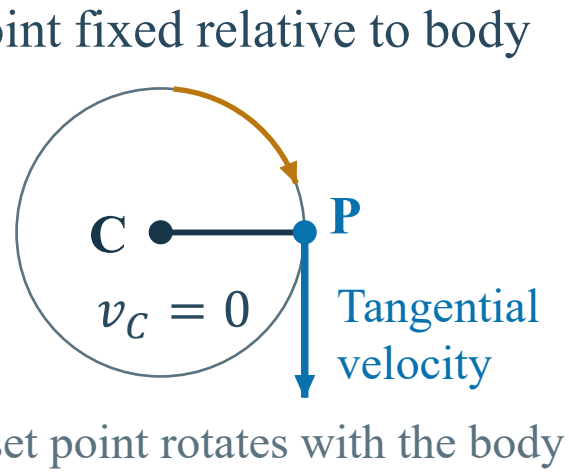
### 4.1.5 Position, Mounting Points, and Frame Conventions: Velocity and Relative Motion

## Changing Velocity Coordinates Differs from Changing the Point

- **Same velocity:** $\mathbf{v}_{C,b}$ and $\mathbf{v}_{C,e}$ both describe center-of-mass velocity relative to Earth, resolved along different axes. Do not add origin position.

$$\mathbf{v}_{C,e} = R\mathbf{v}_{C,b}, \qquad \mathbf{v}_{P,e} = \dot{\mathbf{r}}_{C,e} + \dot{R}\boldsymbol{\rho}_{CP,b} + R\dot{\boldsymbol{\rho}}_{CP,b}$$

- **Different points:** differentiating mounting-point position gives center-of-mass translation, body rotation, and mounting-point motion relative to the body.
- **Fixed mounting:** $\dot{\boldsymbol{\rho}}_{CP,b} = 0$, but $\dot{R}\boldsymbol{\rho}_{CP,b}$ can be nonzero. With a stationary center of mass, an offset sensor can still move along an arc.

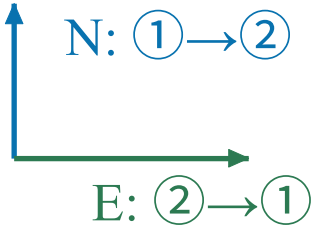



**One velocity needs only rotation to change coordinates; different points also require rigid-body motion.**

### 4.1.5 Position, Mounting Points, and Frame Conventions: Converting Interface Conventions

# Frame Conversion Changes Both Axis Order and Signs

- **Earth axes:** east-north-up (ENU) orders components east, north, up. NED→ENU swaps the first two and reverses the third.




$$\mathbf{a}_{ENU} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \mathbf{a}_{NED}, \quad \mathbf{a}_{FLU} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \mathbf{a}_{FRD}$$

- **Body axes:** front-left-up (FLU) keeps forward and reverses both right and down. Reversing two axes preserves handedness.
- **Scope:** these matrices transform vector components. Positions also need any origin offset; attitude matrices require a basis change at both ends. A northward 1 m is $[0,1,0]^{\mathsf{T}}$ m in ENU.

**When conventions change, transform axis permutations and signs together.**

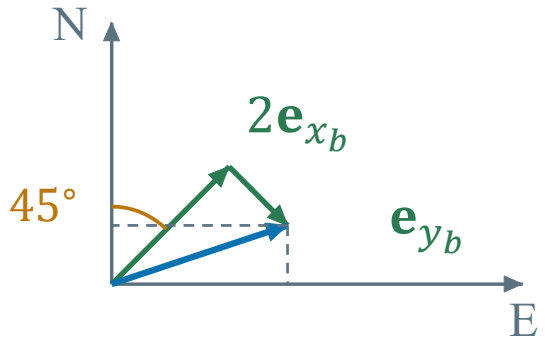
Class Exercise

# Does a Coordinate Change Alter Speed?

- **Given:** level vehicle, yaw $\psi = 45°$, body velocity $\mathbf{v}_b = [2,1,0]^T$ m/s. Find Earth-frame velocity, check for a downward component, and compare speeds before and after conversion.

$$\mathbf{v}_e = [2\cos 45° - \sin 45°,\ 2\sin 45° + \cos 45°,\ 0]^{\mathrm{T}}$$

$$\mathbf{v}_e = [0.707107, 2.121320, 0]^{\mathrm{T}}\ \mathrm{m/s}, \quad \| \mathbf{v}_e \| = \| \mathbf{v}_b \|$$



Velocity-vector addition

- **Answer:** project forward/right onto north/east; vertical velocity is zero. Eastward terms add; northward terms partly cancel. Both speeds remain $\sqrt{5} \approx 2.236068$ m/s.
- **Reverse check:** $R^T \mathbf{v}_e$ should recover $[2,1,0]^T$ m/s. Round trips test forward/inverse consistency; check physical direction against north and east as well.

**Coordinates change components; orthogonal rotation preserves vector length and physical motion.**

Class Exercise

# Self-Check: Direction, Half-Angles, and Rate Mapping

- **Q1:** level attitude, $\psi = 135°$, body-forward speed 3 m/s. Find Earth velocity and direction.
- **Q2:** constant rotation about body down at $\omega_z = \pi/3$ rad/s, step 0.5 s. Find the one-step increment $\delta\mathbf{q}$.
- **Q3:** $\phi = 30°$, $\theta = 0$, gyro reading $[p, q, r] = [0,0.2,0]$ rad/s. What are the three Euler-angle rates?
- **Reference:** Q1 $\mathbf{v}_e = [-2.121,2.121,0]^\mathsf{T}$ m/s, southeast. Q2 actual turn 30°, half-angle 15° gives $\delta\mathbf{q} = [0.9659,0,0,0.2588]^\mathsf{T}$. Q3 $[\dot{\phi}, \dot{\theta}, \dot{\psi}] = [0,0.173,0.1]$ rad/s; both pitch and yaw rates are nonzero.

**Check known directions, half-angles, and rate mapping before trusting program output.**

## Class Exercise: 1D Vertical-Motion Example

# Reproduce Thrust-to-Height Computation with the 2 kg Model

- **Directory:** [installation directory]/RflySimAPIs/4.RflySimModel/1.BasicExps/ e6_ModelingLearningLab/01_VerticalRigidBody/; read Readme.pdf first

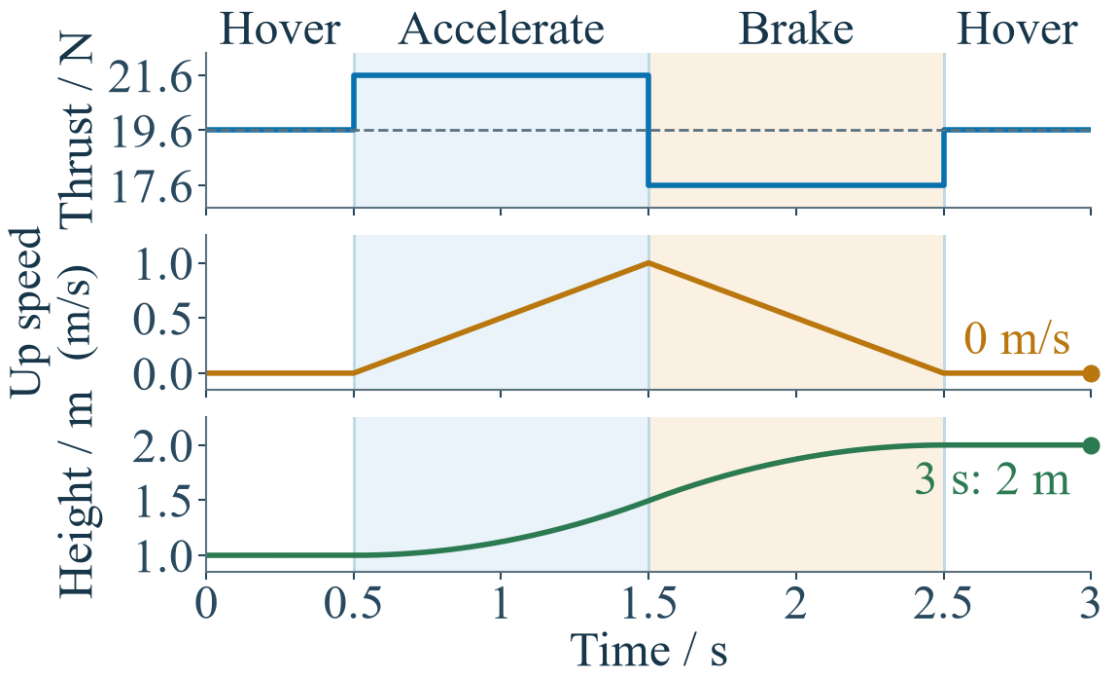
```
r = vertical_beginner;
```

- Run offline with MATLAB only. Set this folder as current, check mass, gravity, thrust segments, and initial height in vertical_beginner.m, then execute the command above.
- **Success:** at 3 s the command window reports height = 2 m, vD = 0, and Default-case checks passed. The outputs folder contains a CSV table, JSON errors, and PNG curves.
- **Common failures:** for a read-only install folder, pass a writable first argument, e.g., vertical_beginner(fullfile(tempdir,'vb')). The time step must divide 0.5 s exactly; otherwise assert fails.

**Check inputs and initial conditions, then explain each phase using the output table and curves.**

## Class Exercise: Interpreting Offline Reference Results

# Thrust Accelerates; Active Braking Is Needed to Stop




1D vertical-motion reference curves (teaching example, not platform measurements)

- **Read:** top panel shows thrust segments. During acceleration, upward speed reaches 1 m/s; braking returns it to 0.
- **Result:** at 3 s, height is 2 m and velocity is 0. Returning directly to 19.6 N after acceleration would give about 3 m height at 3 s.
- **Step check:** halving dt cuts maximum position error from about 0.005 m to 0.0025 m.
- **Mass change:** at 2.2 kg, 19.6 N is below the 21.56 N weight; height keeps decreasing.

**Interpret results by input interval, coordinate signs, errors, and model limits.**

Section Summary

# Keep One Convention from Model Boundary to Motion Results

- Define vehicle and boundary: the complete textbook model connects actuators, environment, forces, rigid body, and observations. The 2 kg example specifies thrust or zeros selected effects only under its stated assumptions.
- Keep NED/FRD, units, and transformation direction consistent. Rotate vector coordinates; account for origins at mounting points. Angular velocity needs kinematics to update attitude.
- For the same input, check both one-step substitution and the full response. Verify matrix orthogonality, quaternion unit norm, and physical direction. Valid numbers do not guarantee correct direction.

- **Common errors:** integrating p, q, r directly into Euler angles; reversing R and its transpose despite passing orthogonality; using full quaternion angles; forgetting upward velocity is negative in NED; left-multiplying body increments. Next: net force and inertia determine acceleration.

**Consistent boundaries, coordinates, and result evidence make predictions explainable.**