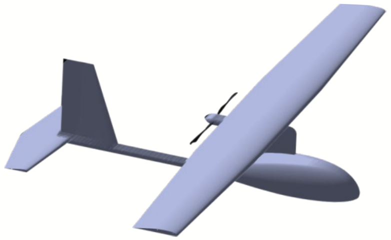
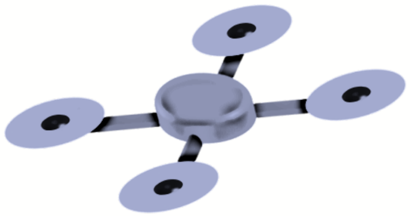
JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING





UAV Systems & Control, Module 1A

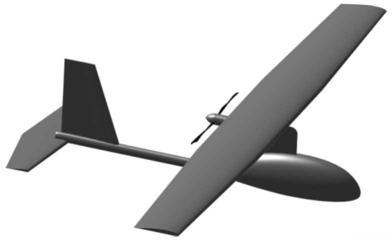
- Course Objectives and Grading
- What are UAVs, and what their components?
- UAV 6DOF Simulation
- Weekly Expectations

Course Objectives

Prerequisites:
- Some Matrix Math
- Control Systems
  - Laplace
  - Block Diagrams
  - Feedback
- MATLAB

General Objective of Course

- Gain an understanding of UAV modeling, flight control and state estimation
  - Mathematics of kinematics & dynamics
  - Force & moment modeling
  - Full nonlinear Equations of Motion (EoMs)
  - Trimming & linearization of EoMs
  - Flight control algorithms & tuning methods
  - Feedback sensor physics and modeling
  - State estimation via Extended Kalman Filtering
  - 6DOF simulation development

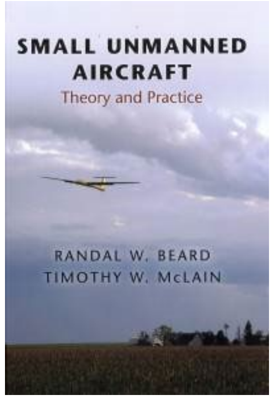


Fixed Wing

Quadcopter (multirotor)

Course Outputs (what you'll end up with)

- Ability to model UAV dynamic motion
- Ability to design, tune and implement flight control and state estimation algorithms
- A complete fixed-wing UAV 6DOF (Matlab/Simulink)
- A complete quadcopter UAV 6DOF (Matlab/Simulink)
- Demonstrated ability to utilize and manipulate an open-source quadcopter



Class generally uses and follows: Beard & McLain, "Small Unmanned Aircraft", *Princeton University Press*, 2012

Numerous images and text courtesy of book website: http://uavbook.byu.edu

Grading

- **Weekly Assignments:**
  - Homework problems, including development of fixed-wing 6DOF: 65%
  - Experimentation and reinforcing assignments with open-source quadcopter: 5%
  - Online discussions: 5%

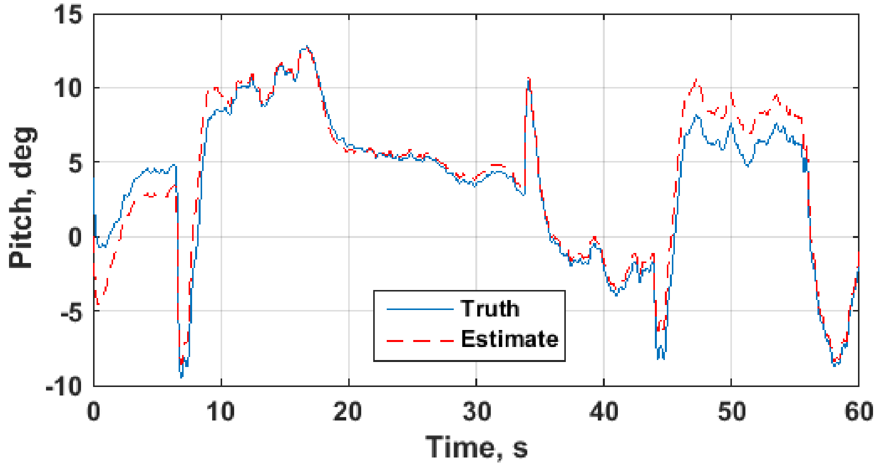
- **End-of-semester Design Projects:**
  - Project 1: Convert the fixed-wing 6DOF to a quadcopter 6DOF: 15%
  - Project 2: Implement/Test autopilot algorithms on open-source quadcopter: 10%

- **Homework and projects are expected to reflect individual work and should be turned in *on-time*.**
  - Homework builds off of previous homework, so it is essential that you keep up
  - Late homework will be docked 10% per day.
  - If you are having difficulty, contact us. We will do our best to respond within 24 hours. (So don't wait until the day before homework is due)
    - *Hardware-related questions should be asked on a Blackboard discussion*
    - *General homework questions (e.g. interpreting questions, etc.) should be asked on a Blackboard discussion thread*
    - *More specific questions ("Help! My code isn't working!") can be sent as emails*

Homework Expectations:

- Legible, and typed if possible
- Organized
- If we say "explain", we are looking for a thoughtful proof of understanding
- Figures should *always* be labeled!

Example:

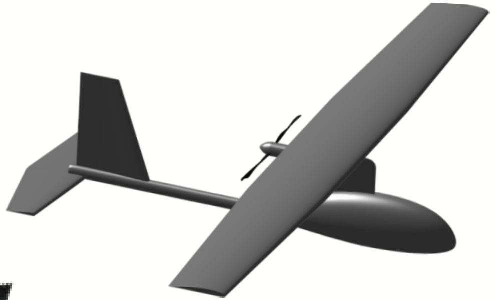


Note: xlabel, ylabel, legend, grid

What is a UAV? What types are there? Applications?

- A UAV is an Unmanned Aerial Vehicle
  - ➢ Reusable, pilotless aircraft
  - ➢ Can be
    - o *Remotely Piloted (Remote pilot “steers”)*
    - o *Autonomous (follows a flight plan without intervention)*
  - ➢ Uses on-board sensors and algorithms to:
    - o *Estimate its position & orientation (yaw, pitch, roll)*
    - o *Automatically control and/or stabilize its flight*

Multi-rotor
e.g. quadcopter
or quadrotor



Fixed-wing

- An Unmanned Aerial <u>System</u> (UA<u>S</u>) consists of a UAV along with all of its supporting communication and Ground Control Station (GCS) systems.

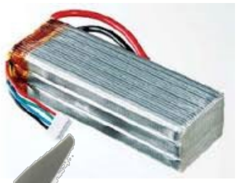
- Potential Applications:
  - ➢ Civil and Commercial
    - o *Filming, monitoring environment, precision agriculture, search and rescue, communications relay, etc.*
  - ➢ Military and Homeland Security
    - o *Situational awareness, special operations, battle damage assessment, surveillance, etc.*

http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

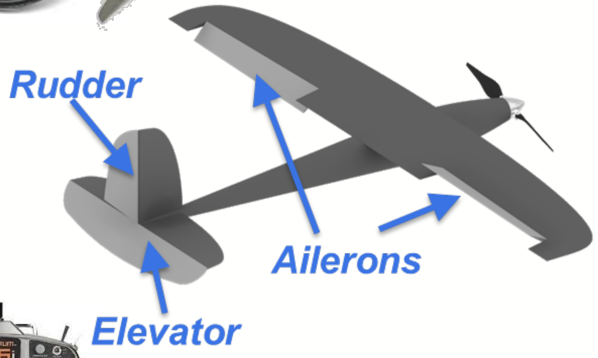
Components of a UAS (Unmanned Aerial System)

- **Airframe**
  - Main Fuselage
  - Wing
  - Tail

- **Propulsion - Electric Motor or Gas Engine**
  - Electronic speed controller & Electric motor
  - Gas Engine & Servo Actuator
  - Propeller
  - Power source (Battery/Fuel)



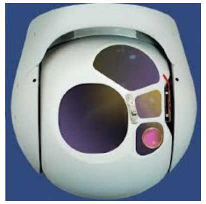
- **Control Surfaces**
  - Ailerons, Elevator, Rudder
  - V-Tail, Elevons
  - Servo actuators

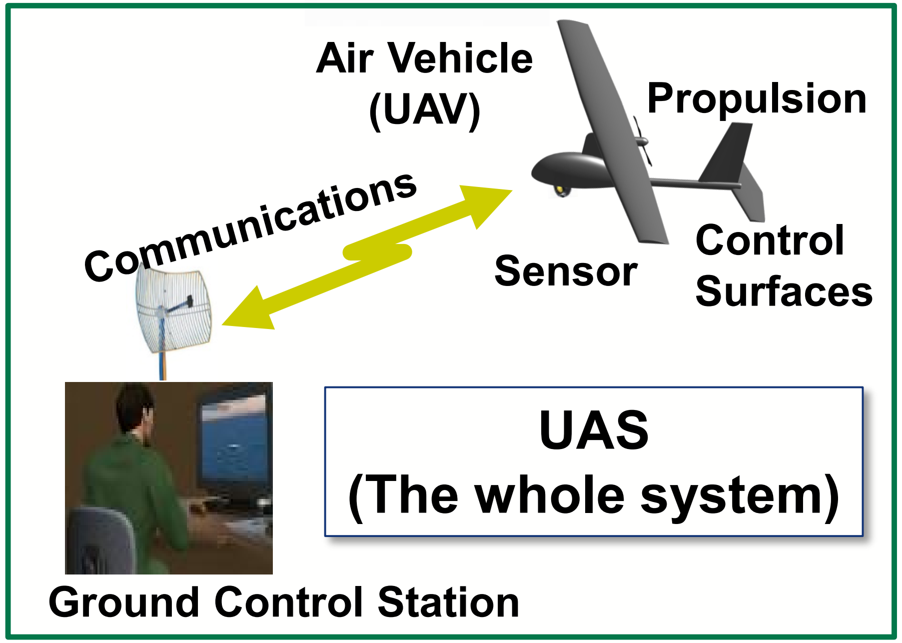


- **Control System**
  - RC Transmitter
  - Autopilot

- **Communication System**
  - Transmitter
  - Antenna
  - RC receiver (optional)

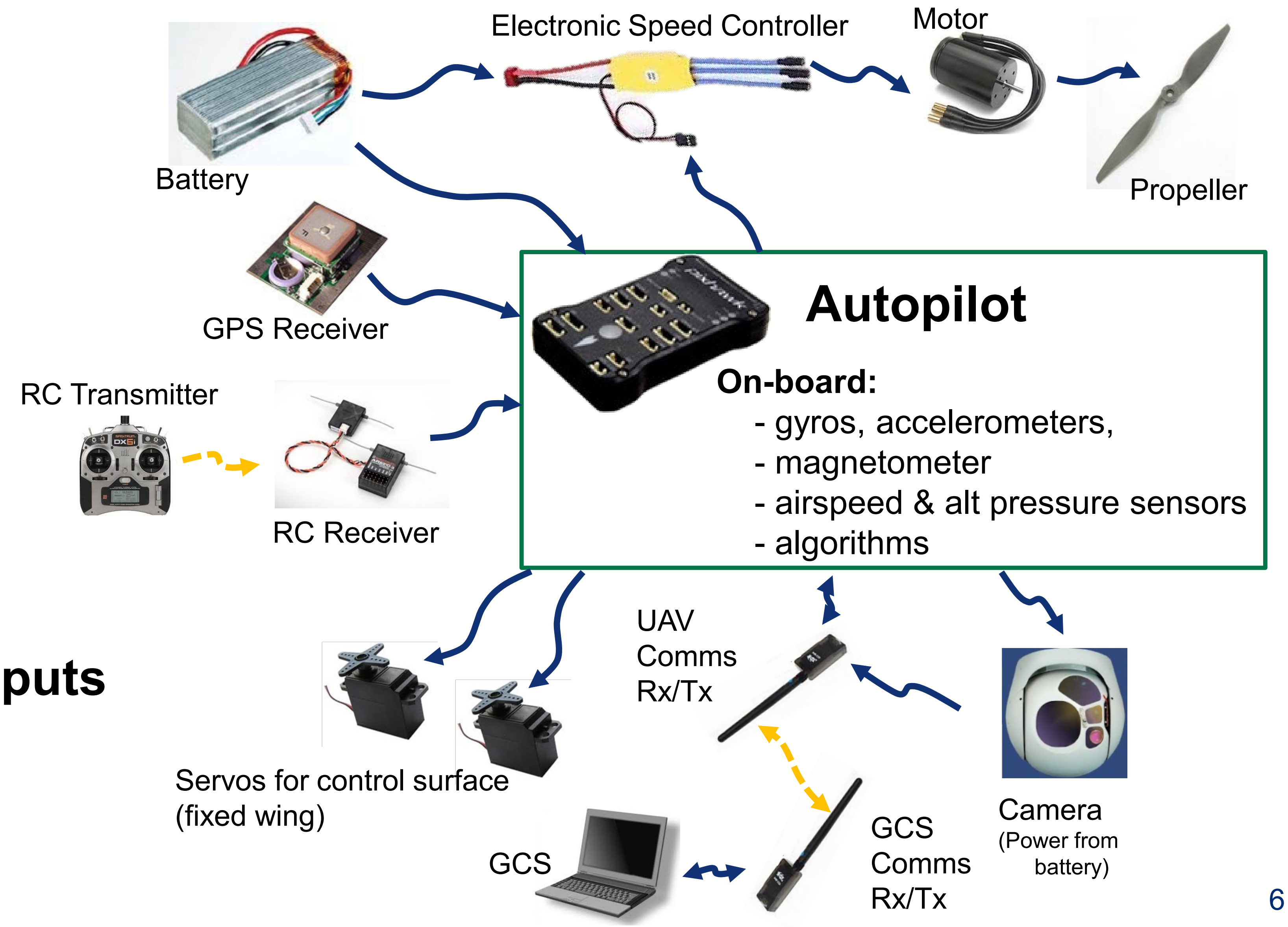
- **Payload/Sensor**
  - Camera
  - Radar



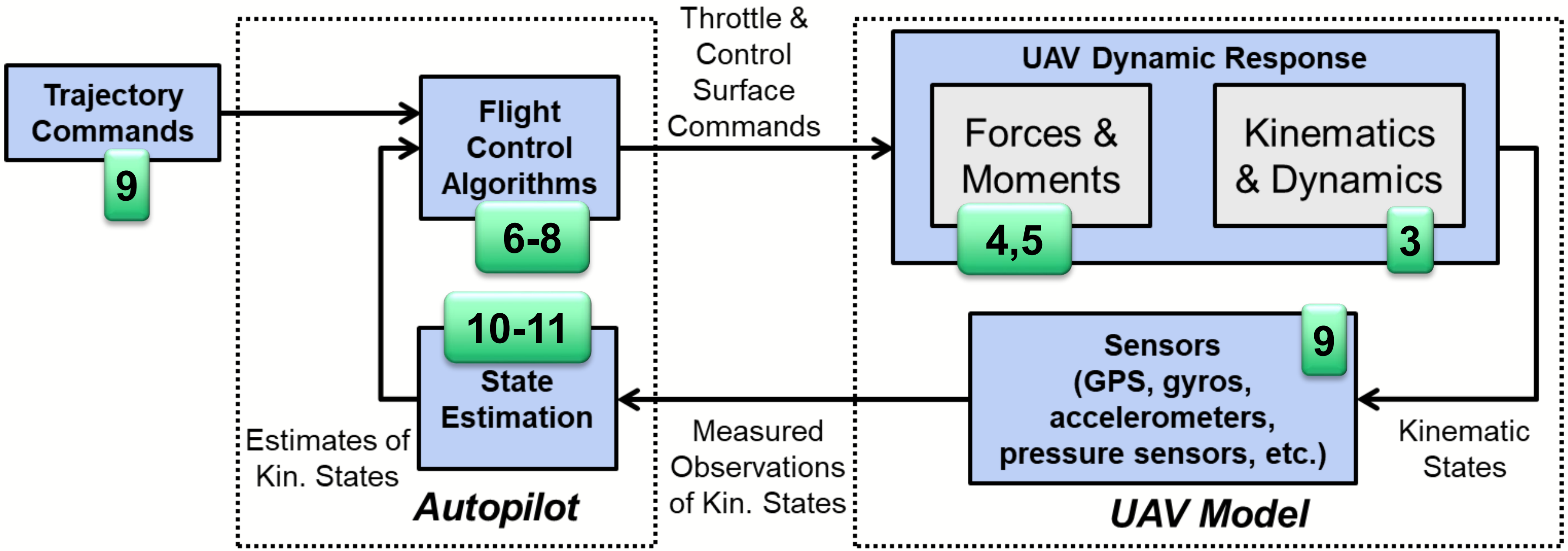


Autopilot

- **CPU**
  - State Estimator
  - Flight Controller
- **Sensors**
  - Gyros
  - Accelerometer
  - GPS
  - Magnetometer
  - Pressure
  - Other Altimeter (optional)
- **Actuator and Motor outputs**
  - Including control surfaces and the motor



UAV Flight Control System Modeling & Course Outline

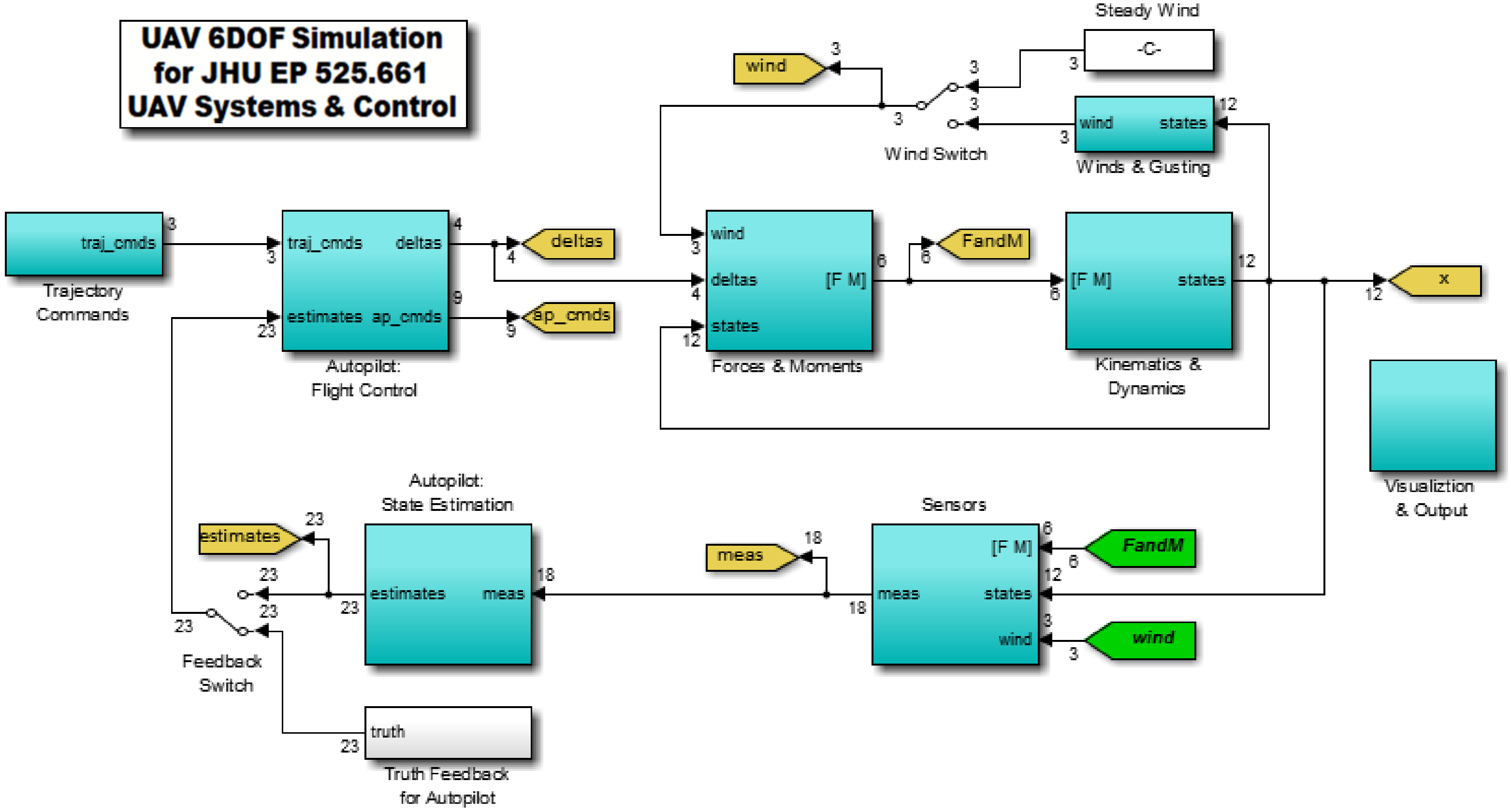


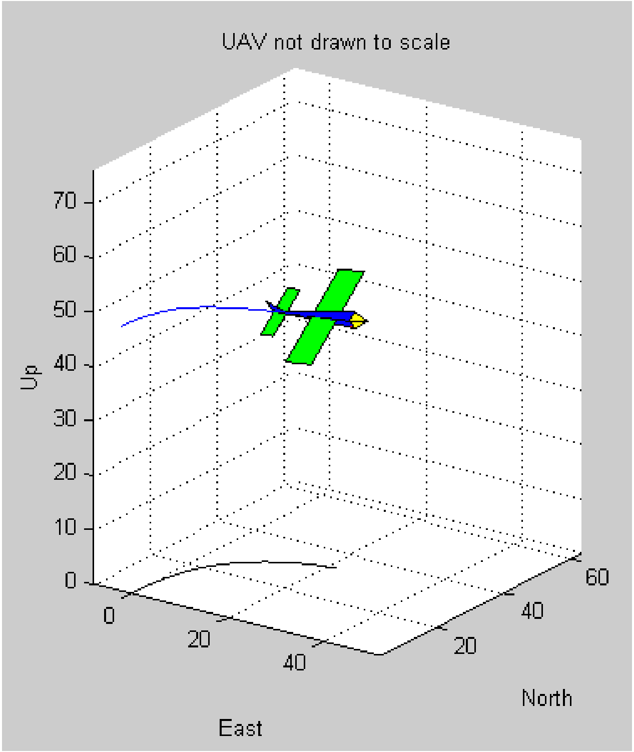
Course Outline by Week:

- **1,2** Vector Geometry
  System Modeling
  & Control
  Intro to UAVs,
  aircraft nomenclature,
  coordinates frames
  and variables
- **3-11** Fixed-wing modeling,
  flight control &
  state estimation
  Concepts reinforced using open-source quadcopter
- **12,13** Quadcopter modeling
  and Design Projects
- **14** Advanced topics
  (camera modeling,
  gimbal pointing, etc.)

- UAV flight control is an interconnection of multiple systems
  - The physical UAV consists of its dynamical response and sensors
  - Autopilot algorithms:
    - *Estimate the UAV state*
    - *Generate flight control commands (i.e. to control surfaces and motors)*

UAV 6DOF Simulation



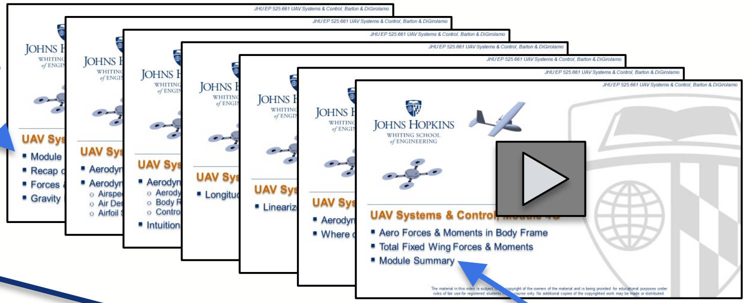


- Students will develop a full 6DOF simulation of a fixed-wing UAV using Simulink and Matlab
  - Simulink architecture is provided
  - Coding is via Matlab functions

Weekly Expectations

Academic sub-module lectures

Module Objectives



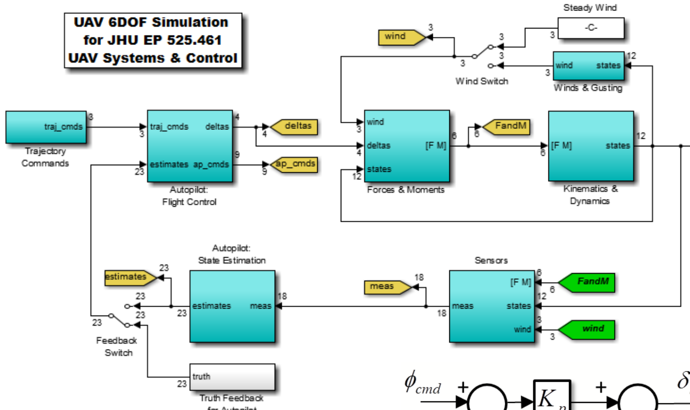
Module Summary

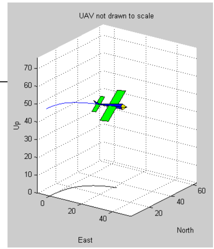
Hardware sub-module lecture(s) (some weeks)

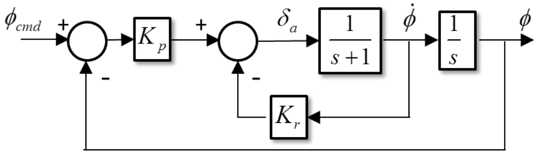
Reinforcing Quadcopter Hardware Assignments

Academic & Simulation Assignments

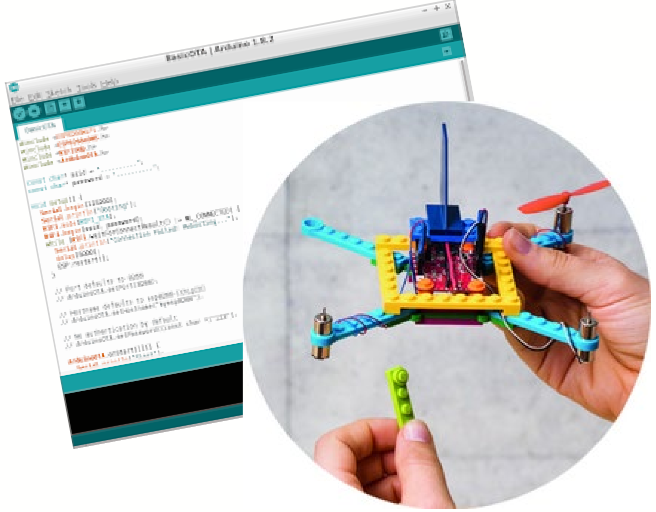
- Respond to discussion prompt by Day 4
- Comment on 2 classmates prompts by Day 6





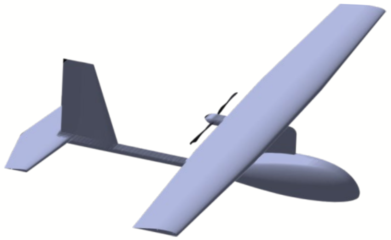


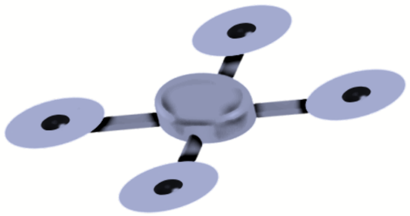
Solutions posted the following week



Two End-of-semester Quadcopter Design Projects (Weeks 12-14)	1) Develop a full 6DOF simulation of a quadcopter UAV	2) Manipulate/modify/add open-source quadcopter flight software

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING





UAV Systems & Control, Module 1B

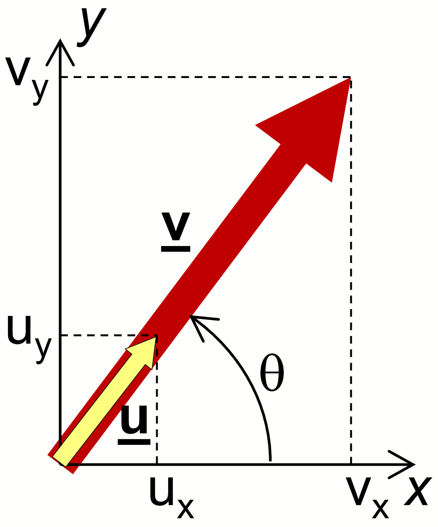
- Vectors
- Matrices
- Coordinate Frames

Vectors — Vectors are defined by magnitude and direction

$$\underline{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

$$v = |\underline{v}| = \sqrt{v_x^2 + v_y^2}$$

$$v_x = v\cos(\theta), v_y = v\sin(\theta)$$



$\underline{u}$: unit vector ($|\underline{u}| = 1$)

$$\underline{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} |\underline{u}|\cos(\theta) \\ |\underline{u}|\sin(\theta) \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}$$

$$|\underline{u}| = \sqrt{u_x^2 + u_y^2} = \sqrt{\cos(\theta)^2 + \sin(\theta)^2} = 1$$

$$\underline{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \end{bmatrix} = v\begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} = v\underline{u}$$

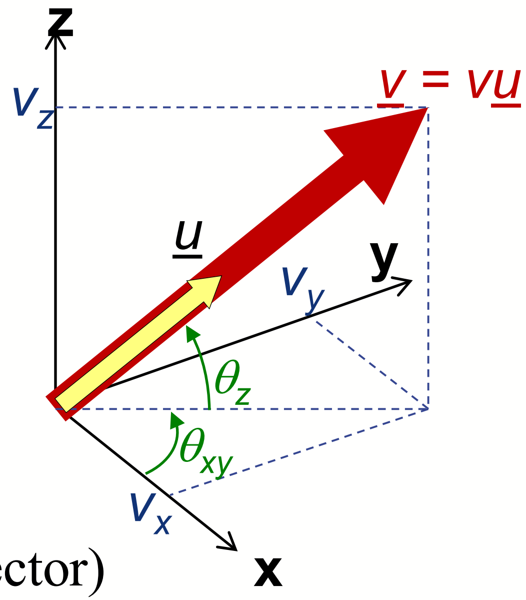
$$\underline{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T \quad (T : \text{Transpose})$$

$$v = |\underline{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

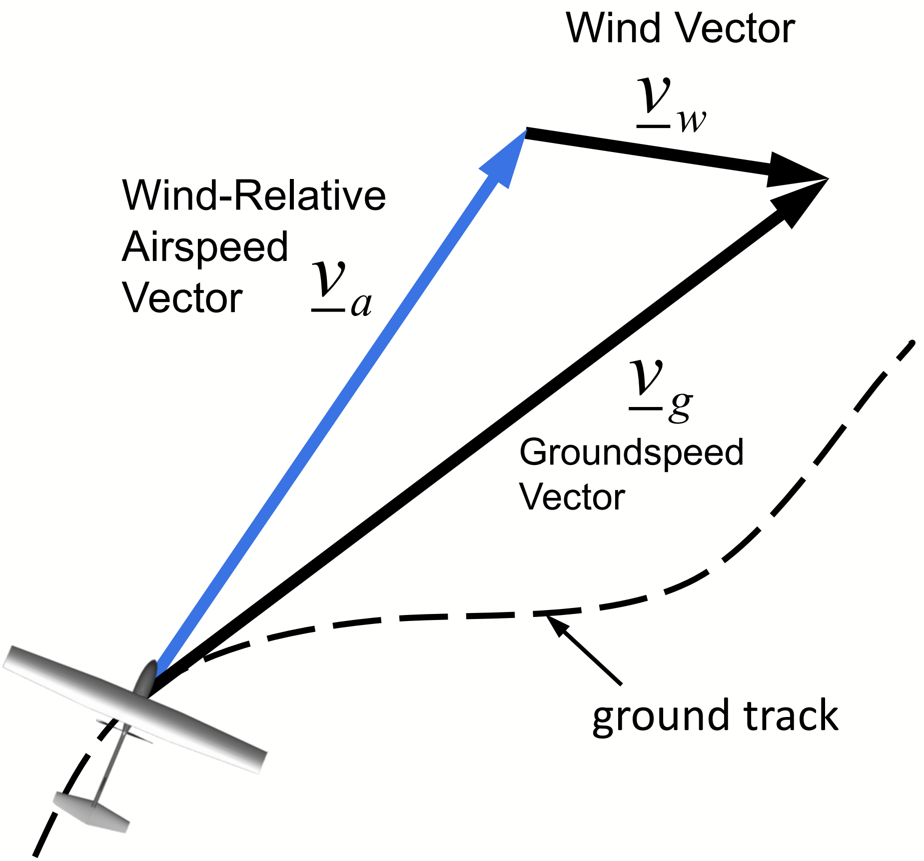
$$\underline{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = v\begin{bmatrix} \cos(\theta_{xy})\cos(\theta_z) \\ \sin(\theta_{xy})\cos(\theta_z) \\ \sin(\theta_z) \end{bmatrix} = v\underline{u}$$

v : Magnitude

$\underline{u}$: Direction (represented as a unit vector)



Vector Example: The Wind Triangle



$\underline{v}_g$: Velocity vector, relative to ground, m/s

$\underline{v}_w$: Wind Vector, relative to ground, m/s

$\underline{v}_a$: Vehicle airspeed vector, aka “Wind-Relative velocity vector”, aka “Velocity relative to moving air”, m/s

$$\boxed{\underline{v}_a = \underline{v}_g - \underline{v}_w}$$

V_a : Airspeed (speed relative to air), m/s

$$V_a = \left|\underline{v}_a\right|$$

Don’t confuse “airspeed” with “windspeed”

We will find that aerodynamic forces and moments are a function of $\underline{v}_a$, the vehicle’s velocity *__relative to__* the moving air mass (wind).

Matrices

- Matrices are 2D arrays of numbers
- Vectors are just a special case of matrices

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad \underline{x}=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad B=\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \qquad \underline{u}=\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

- Matrix sizes are denoted by the number of rows and columns
  - B is a 3x2 matrix with 3 rows and 2 columns
  - A is a square 3x3 matrix with 3 rows and 3 columns

- Matrix of similar sizes can be added and subtracted
- Matrices can be multiplied if the # of *columns* of the first matrix match the # of *rows* of the second matrix

$$c_{ij} = \sum_{k=1}^{k=\#\text{ of colums of } A} a_{ik} b_{kj}$$

$$C = AB = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix}$$

3x3 3x2

Must match

$$c_{32} = \sum_{k=1}^{k=3} a_{3k} b_{k2} = a_{31} b_{12} + a_{32} b_{22} + a_{33} b_{32}$$

c_{32} is the dot product of the 3rd row of A with the 2nd column of B

Matrices

- In general, FG is not equal to GF (where F and G are matrices)
- Transpose means switching rows with columns

$$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \qquad B^T = \begin{bmatrix} b_{11} & b_{21} & b_{31} \\ b_{12} & b_{22} & b_{32} \end{bmatrix}$$

```
MATLAB:
A = [1 2; 3 4];
A_transpose = A';
A_inv = inv(A);
I_3x3 = eye(3);
```

- "Rank" of a matrix is the number of independent rows in a matrix
  - Independent rows: rows that are not linear combinations of other rows
  - "Full Rank" means its rank equals its number of rows

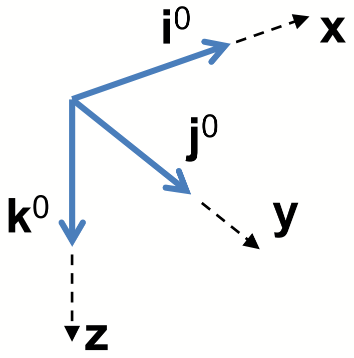
- A square matrix with full rank can be inverted:

$$A^{-1}A = AA^{-1} = I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Coordinate Frames

- In guidance and control of aircraft, reference frames are used *a lot*

- We define a frame (e.g. Frame 0) as three orthogonal unit vectors
  - e.g, $\mathbf{i}^0$, $\mathbf{j}^0$, $\mathbf{k}^0$ (along x, y & z directions)



- Frames used to describe relative position and orientation of objects
  - Aircraft relative to direction of wind
  - Camera relative to aircraft
  - Aircraft relative to inertial frame

- Some things most easily calculated or described in certain reference frames
  - Newton's law (in an *inertial*, or non-moving, frame)
  - Aircraft attitude (relative to *North-East-Down* frame)
  - Aerodynamic forces/torques (in *body-fixed* frame)
  - Accelerometers, rate gyros (in *body-fixed* frame)
  - GPS (in *Earth-fixed* frame)
  - Mission requirements (e.g. in *North-East-Down* frame)

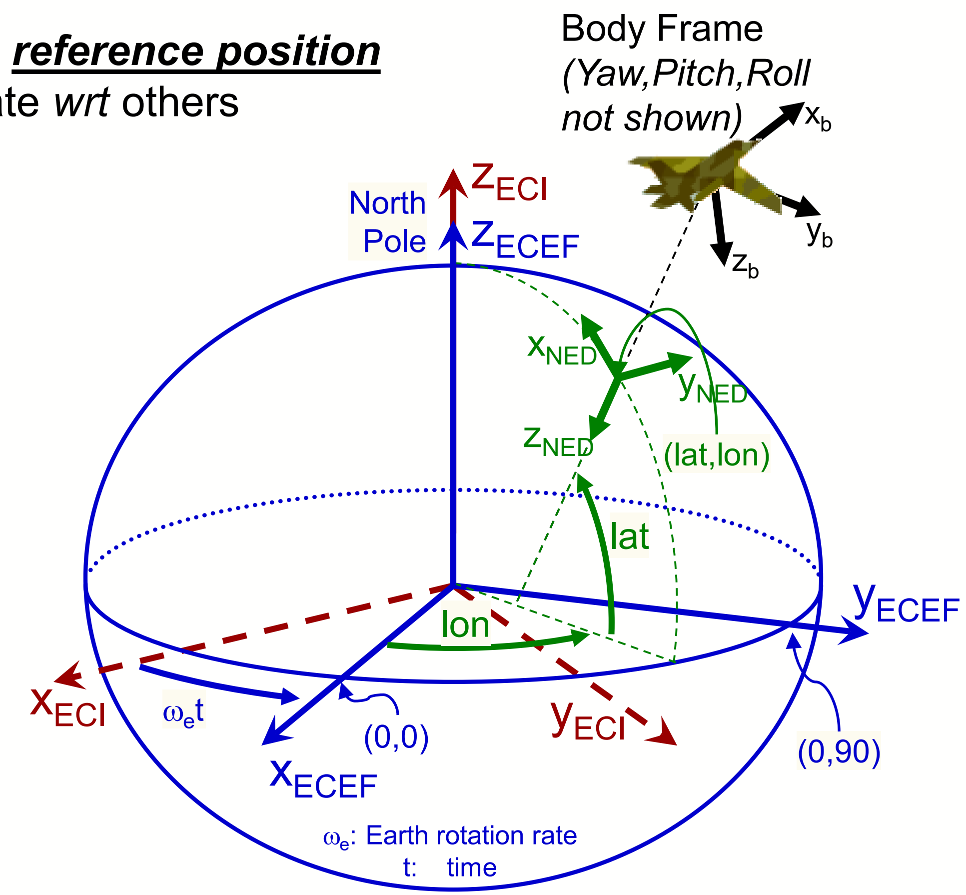
Must know how to transform between different reference frames

Coordinate Frames

Coordinate frame defined by ***orientation*** and ***reference position***
Some coordinate frames rotate and/or translate *wrt* others

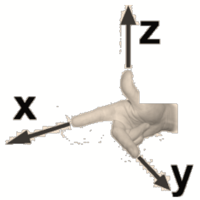
Common Coordinate Frames:

- **NED: x=North, y=East, z=Down**
  - **Referenced to some (lat,lon)**
- ENU: x=East, y=North, z=Up
  - Referenced to some (lat,lon)
- **Body, Forward-Right-Down:**
  - **x="Forward" (axially out nose)**
  - **y="Right" (out right wing)**
  - **z="Down" (out belly of aircraft)**
  - **Fixed to body (position & orientation)**
- ECEF: Earth-Centered-Earth-Fixed
  - Origin at earth center, rotates with Earth
  - x: toward (lat,lon)=(0,0)
  - z: toward north pole
  - y: completes right-hand-rule
- ECI: Earth-Centered Inertial
  - Starts as ECEF at some time reference (e.g. vernal equinox)
  - Does not rotate with Earth

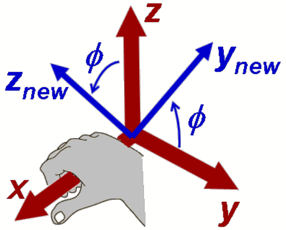


Most coordinate frames Follow right-hand-rule

Right-handed Frame



Right-handed Rotation



"Inertial" frame:

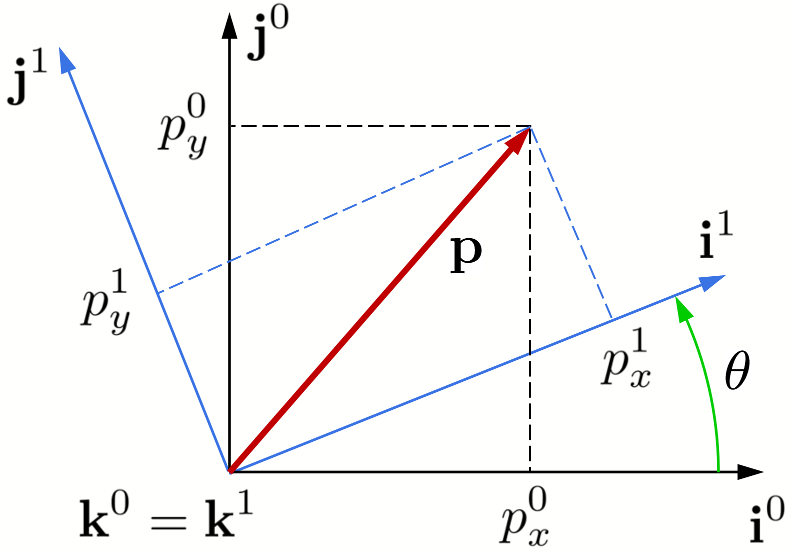
- Any frame assumed to be non-moving (rotating or translating)
- ***e.g.*: In a flat-earth simulation, NED may be considered inertial**
- *e.g.*: In a curved-earth simulation ignoring Earth rotation, ECEF may be considered inertial
- *e.g.*: In a curved-earth simulation modeling Earth rotation, ECI may be considered inertial

Rotation of a Coordinate Frame

Consider two coordinate frames.

Frame 0: i^0, j^0, k^0 Frame 1: i^1, j^1, k^1

Frame 1 is rotated θ radians (about $\mathbf{k^0}$) relative to Frame 0.



p can be expressed in Frame 0 coordinates:

$$\mathbf{p} = p_x^0\mathbf{i}^0 + p_y^0\mathbf{j}^0 + p_z^0\mathbf{k}^0$$

p can be expressed in Frame 1 coordinates:

$$\mathbf{p} = p_x^1\mathbf{i}^1 + p_y^1\mathbf{j}^1 + p_z^1\mathbf{k}^1$$

But the vector **p** hasn't changed, so:

$$p_x^1\mathbf{i}^1 + p_y^1\mathbf{j}^1 + p_z^1\mathbf{k}^1 = p_x^0\mathbf{i}^0 + p_y^0\mathbf{j}^0 + p_z^0\mathbf{k}^0$$

With manipulation (taking the dot product of both sides with $\mathbf{i^1}$, $\mathbf{j^1}$, and $\mathbf{k^1}$) a relationship can be made between the vector expressed in Frame 0 ($\mathbf{p^0}$) and the same vector expressed in Frame 1 ($\mathbf{p^1}$):

$$\mathbf{p}^1 \triangleq \begin{pmatrix} p_x^1 \\ p_y^1 \\ p_z^1 \end{pmatrix} = \begin{pmatrix} \mathbf{i}^1\cdot\mathbf{i}^0 & \mathbf{i}^1\cdot\mathbf{j}^0 & \mathbf{i}^1\cdot\mathbf{k}^0 \\ \mathbf{j}^1\cdot\mathbf{i}^0 & \mathbf{j}^1\cdot\mathbf{j}^0 & \mathbf{j}^1\cdot\mathbf{k}^0 \\ \mathbf{k}^1\cdot\mathbf{i}^0 & \mathbf{k}^1\cdot\mathbf{j}^0 & \mathbf{k}^1\cdot\mathbf{k}^0 \end{pmatrix} \begin{pmatrix} p_x^0 \\ p_y^0 \\ p_z^0 \end{pmatrix}$$

or: $\mathbf{p}^1 = \mathcal{R}_0^1\mathbf{p}^0$

where

$$\mathcal{R}_0^1 \triangleq \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

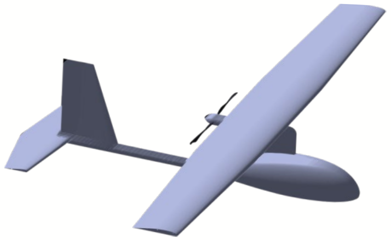
(rotation about $\mathbf{k}$ axis)

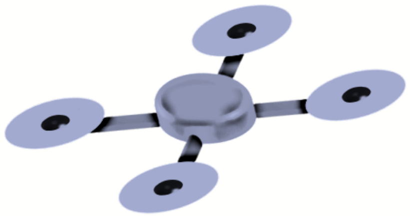
Dot Product of two unit vectors is the cosine of the angle between them, e.g.:

$$\mathbf{i}^1\cdot\mathbf{i}^0 = \cos(\theta)$$
$$\mathbf{i}^1\cdot\mathbf{j}^0 = \cos(90^o - \theta) = \sin(\theta)$$
$$\mathbf{k}^1\cdot\mathbf{k}^0 = \cos(0^o) = 1$$

$$\mathbf{i}^1\cdot\mathbf{i}^1 = \cos(0^o) = 1$$
$$\mathbf{i}^1\cdot\mathbf{j}^1 = \cos(90^o) = 0$$
$$\mathbf{i}^1\cdot\mathbf{k}^1 = \cos(90^o) = 0$$

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING



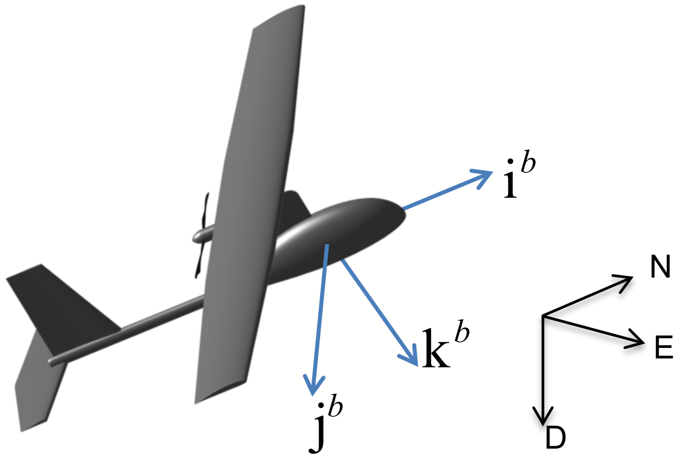


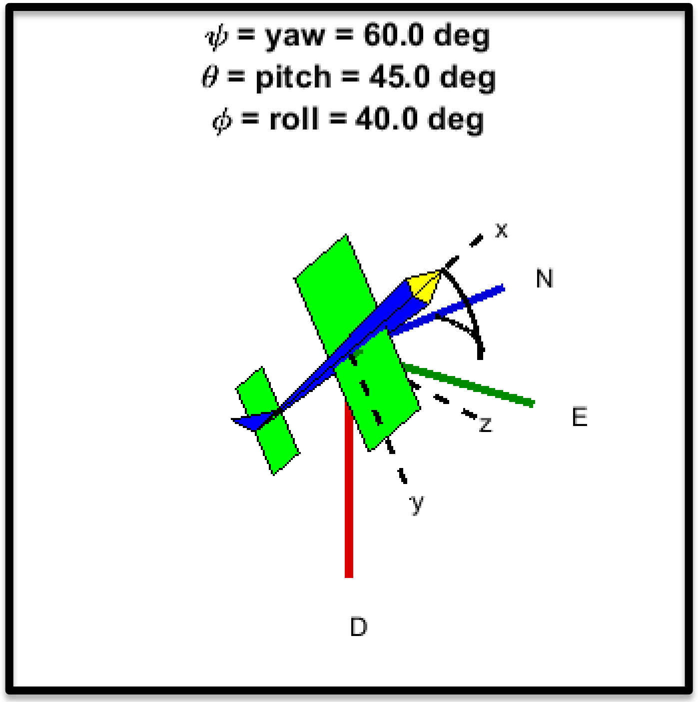
UAV Systems & Control, Module 1C

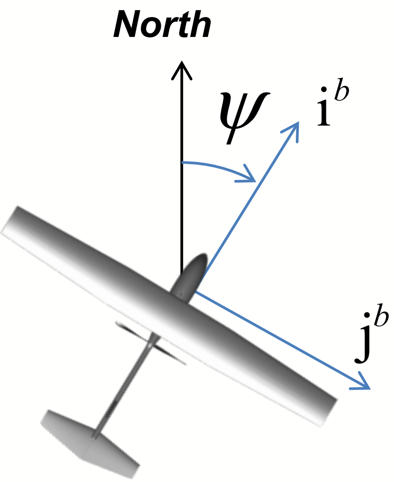
- Euler Angles
- Rotation Matrices
- Aerodynamic Angles
- Velocity Vector Angles

Euler Angles

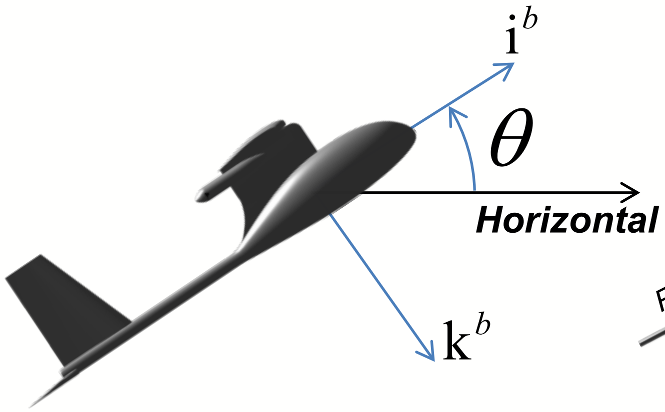
- Define body frame as: **i**^{b}, **j**^{b}, **k**^{b}
  - Forward, Right, Down unit vectors
- Need way to describe attitude of aircraft
- Common approach: Euler angles



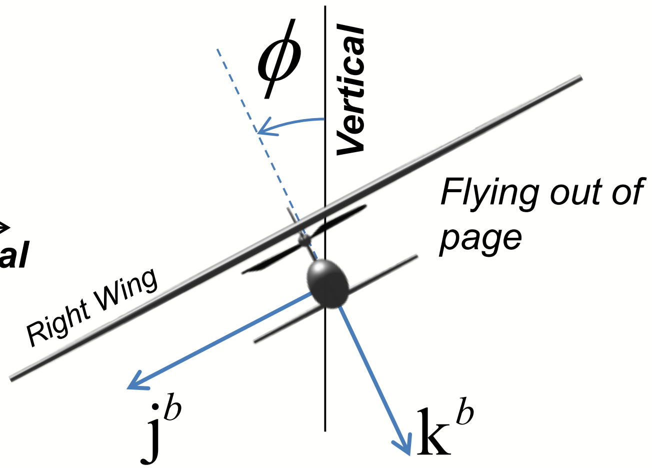




ψ : Yaw



θ : Pitch

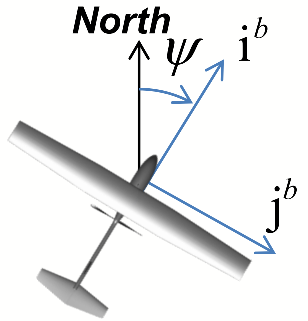


ϕ : Roll

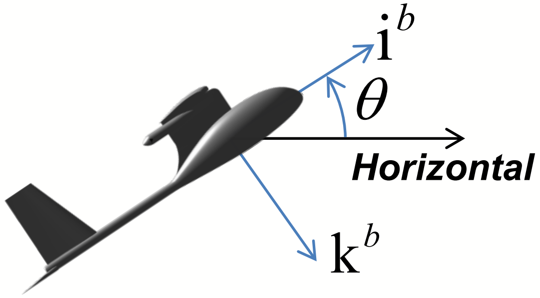
- Pro: Intuitive
- Con: Euler Angles are non-unique
  - E.g. ψ and ϕ are indistinguishable at $\theta = \pm 90°$
  - Results in a mathematical singularity at $\theta = \pm 90°$
  - Related to "Gimbal Lock"
- Quaternions are alternative for overcoming singularity

Rotation Matrix from Euler Angles

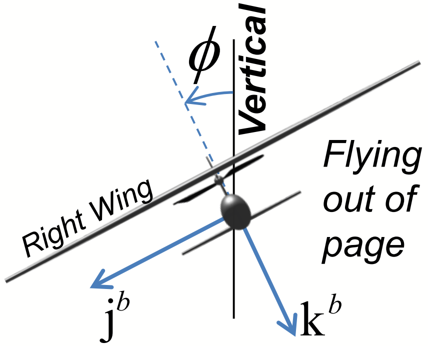
Any rotation between two 3D coordinate frames can be expressed through three Euler rotations. Consider the rotation: <u>NED frame to the vehicle body frame</u>:



1st Rotation:
ψ (yaw) about initial z



2nd Rotation:
θ (pitch) about resulting y

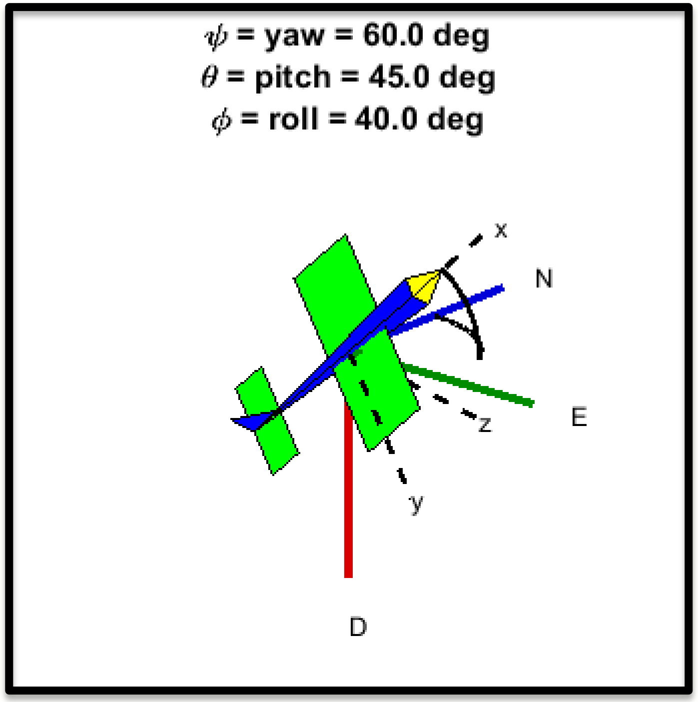


3rd Rotation:
ϕ (roll) about resulting x

$$
\begin{array}{cccc}
 & \textbf{3rd} & \textbf{2nd} & \textbf{1st} \\
R_{NED}^{b}=R_x(\phi)R_y(\theta)R_z(\psi)= & \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} & \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} & \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{array}
$$

← Rot. Matrices are multiplied <u>right</u>-to-<u>left</u>! ━●

$$
R_{NED}^{b}=R_x(\phi)R_y(\theta)R_z(\psi)=\begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi-\cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi+\cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi+\sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi-\sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix}
$$



Rotation Matrices

We can convert between coordinate frames using rotation matrices.

The order of rotation matters! (e.g. Z→Y →X, or X →Y →Z, or Y →X →Z, etc.).

We will generally use the Z →Y →X rotation order:

$$R = R_x(\phi)R_y(\theta)R_z(\psi) = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}}_{3^{rd}:\ \phi \text{ about } x} \underbrace{\begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}}_{2^{nd}:\ \theta \text{ about } y} \underbrace{\begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{1^{st}:\ \psi \text{ about } z} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi-\cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi+\cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi+\sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi-\sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix}$$

← Rotations are right to left

Rotation Matrix Properties:

R_A^B : Transformation from Frame A to Frame B

$R_B^A = \left(R_A^B\right)^{-1} = \left(R_A^B\right)^T$ (Note: $R^{-1} = R^T$ is a property of Rotation Matrices, not matrices in general)

$R_A^C = R_B^C R_A^B$ Rot. Matrices can be cascaded (right-to-left)

A vector $\underline{x}^A$ in Frame A coordinates can be expressed in Frame B: $\underline{x}^B = R_A^B \underline{x}^A$

Extract Euler Angles from Rot. Matrix:

$\lvert R_{13}\rvert \neq 1$	$\lvert R_{13}\rvert = 1 \left(\theta = \pm\frac{\pi}{2} = \pm 90^o\right)$
$\phi = \tan^{-1}\left(\frac{R_{23}}{R_{33}}\right)$	$\phi = 0$
$\theta = -\sin^{-1}\left(R_{13}\right)$	$\theta = -\sin^{-1}\left(R_{13}\right) = \pm\frac{\pi}{2}$
$\psi = \tan^{-1}\left(\frac{R_{12}}{R_{11}}\right)$	$\psi = -\tan^{-1}\left(\frac{R_{21}}{R_{22}}\right)$

Note: Lecture video shows incorrect equation here.

Remember to use atan2(num,den)

Frame Example: The Wind Triangle

Groundspeed vector can be expressed in NED or body coordinates:

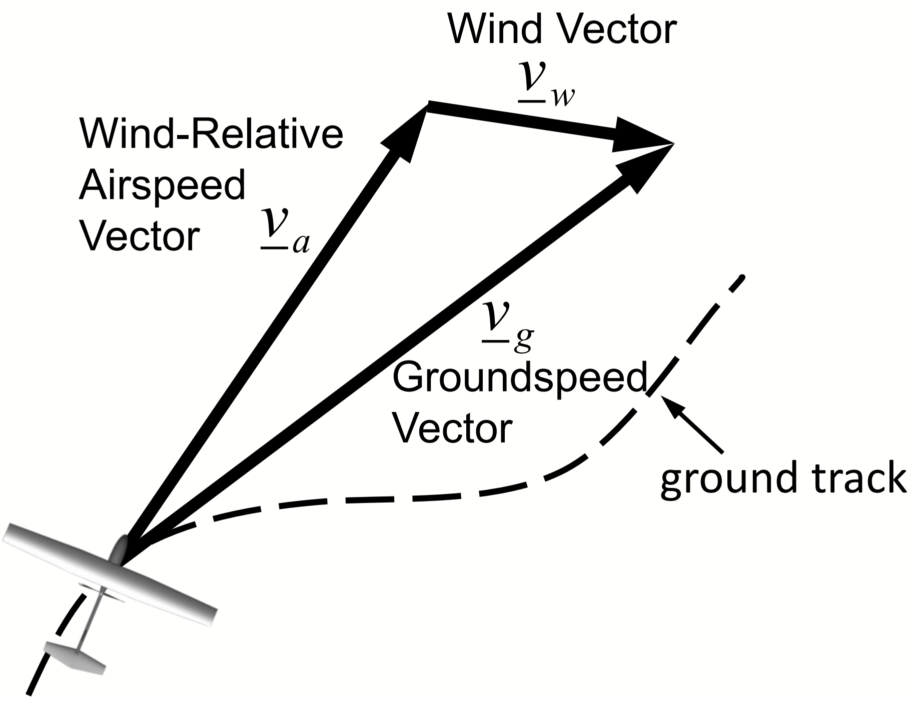
$$\underline{v}_g^{NED} = \begin{bmatrix} V_{north} & V_{east} & V_{down} \end{bmatrix}^T$$

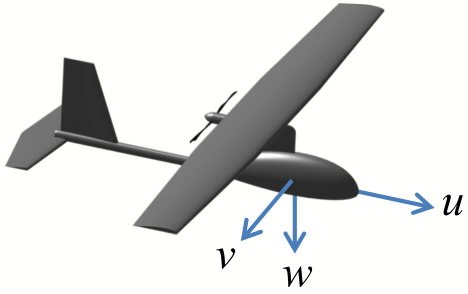
$$\underline{v}_g^b = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = R_{NED}^b \underline{v}_g^{NED}$$

Similarly, Wind vector can be expressed in NED or body coordinates:

$$\underline{v}_w^{NED} = \begin{bmatrix} w_n & w_e & w_d \end{bmatrix}^T$$

$$\underline{v}_w^b = \begin{bmatrix} u_w \\ v_w \\ w_w \end{bmatrix} = R_{NED}^b \underline{v}_w^{NED}$$



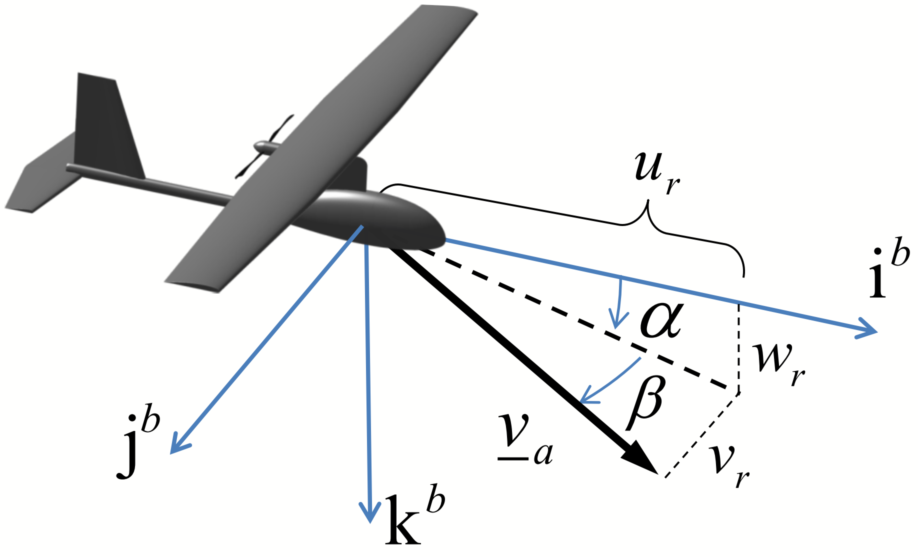


Wind-relative Airspeed vector expressed in body coordinates:

$$\underline{v}_a^b = \begin{bmatrix} u_r \\ v_r \\ w_r \end{bmatrix} = \underline{v}_g^b - \underline{v}_w^b = \begin{bmatrix} u - u_w \\ v - v_w \\ w - w_w \end{bmatrix}$$

Note: The letters u, v and w are used to express various velocity vectors in body frame

Aerodynamic Angles



We will find that aerodynamic forces are highly dependent on the aerodynamic angles (α and β) which describe the direction of the airspeed vector relative to body frame.

Angle-of-Attack:

$$\alpha = \tan^{-1}\left(\frac{w_r}{u_r}\right)$$

Sideslip:

$$\beta = \sin^{-1}\left(\frac{v_r}{\sqrt{u_r^2 + v_r^2 + w_r^2}}\right)$$

where:

$$\underline{v}_a^b = \begin{bmatrix} u_r \\ v_r \\ w_r \end{bmatrix} = V_a \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix}$$

Note:
Sideslip can alternatively be defined purely in the $\mathbf{i}^b$-$\mathbf{j}^b$ plane:

$$\beta_{i^b j^b} = \tan^{-1}\left(\frac{v_r}{u_r}\right)$$

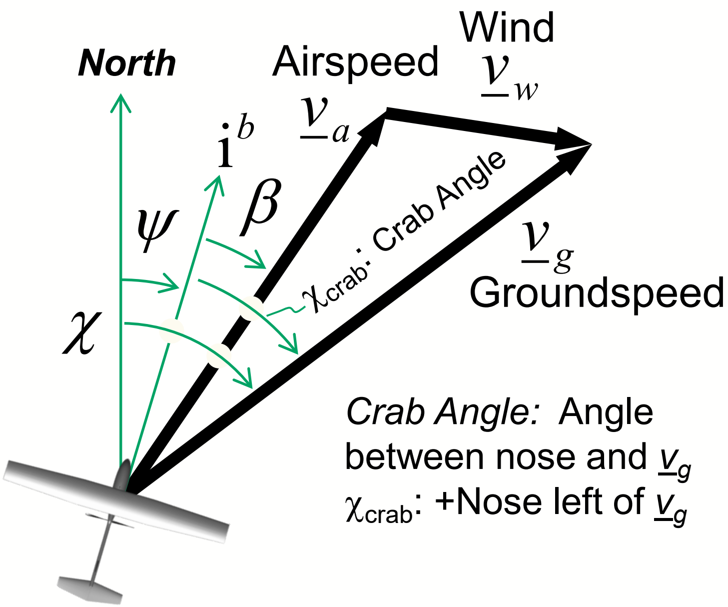
We will not use this alternative definition.

Wind-relative Airspeed vector expressed in body coordinates:

$$\underline{v}_a^b = \begin{bmatrix} u_r \\ v_r \\ w_r \end{bmatrix} = \underline{v}_g^b - \underline{v}_w^b$$

Airspeed: $$V_a = \left|\underline{v}_a^b\right| = \sqrt{u_r^2 + v_r^2 + w_r^2}$$

Vehicle Angles from Wind Triangle

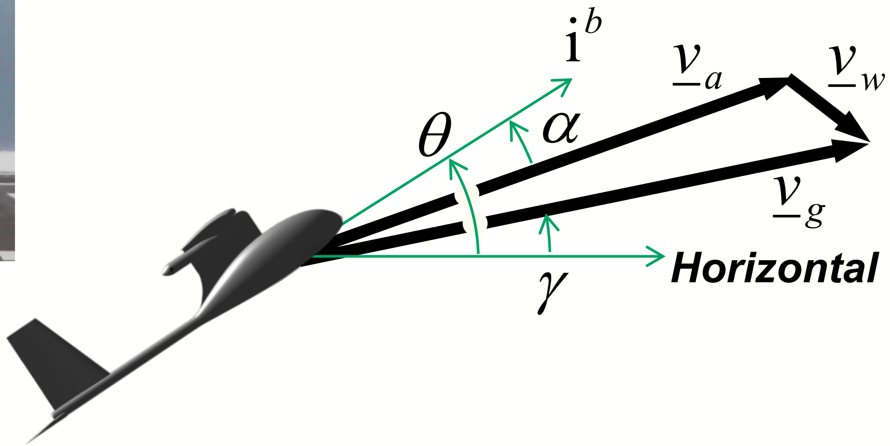


Crab Angle: Angle between nose and $\underline{v}_g$

χ_{crab}: +Nose left of $\underline{v}_g$

Crab Example

https://www.youtube.com/watch?v=Ia-hSjKP2TU



Euler Angles describing orientation relative to NED:

ψ : Yaw, +East-Of-North

θ : Pitch, +Nose-Up

ϕ : Roll, +Right-Wing-Down
(ϕ not shown)

$$R_{NED}^{b} = R_x(\phi) R_y(\theta) R_z(\psi)$$

Aerodynamic angles describing Airspeed Vector relative to body:

α : Angle-Of-Attack, +Nose above $\underline{v}_a$

β : Sideslip, +Nose left of $\underline{v}_a$

$$\underline{v}_a^b = \begin{bmatrix} u_r \\ v_r \\ w_r \end{bmatrix} = V_a \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix}$$

Angles describing Groundspeed Vector relative to NED:

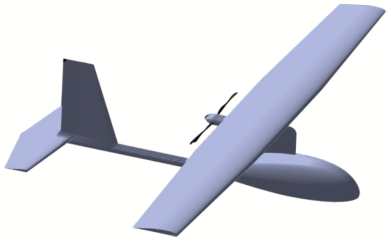
χ : Course (Horizontal), +East-Of-North

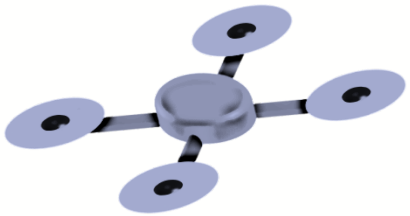
γ : Flight Path Angle (Vertical), +Up

$$\underline{v}_g^{NED} = \begin{bmatrix} V_{north} \\ V_{east} \\ V_{down} \end{bmatrix} = V_g \begin{bmatrix} \cos\chi\cos\gamma \\ \sin\chi\cos\gamma \\ -\sin\gamma \end{bmatrix}$$

Homework: V_g = ? γ = ? χ = ?

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING





UAV Systems & Control, Module 1D

- Differential Equations
- State Space

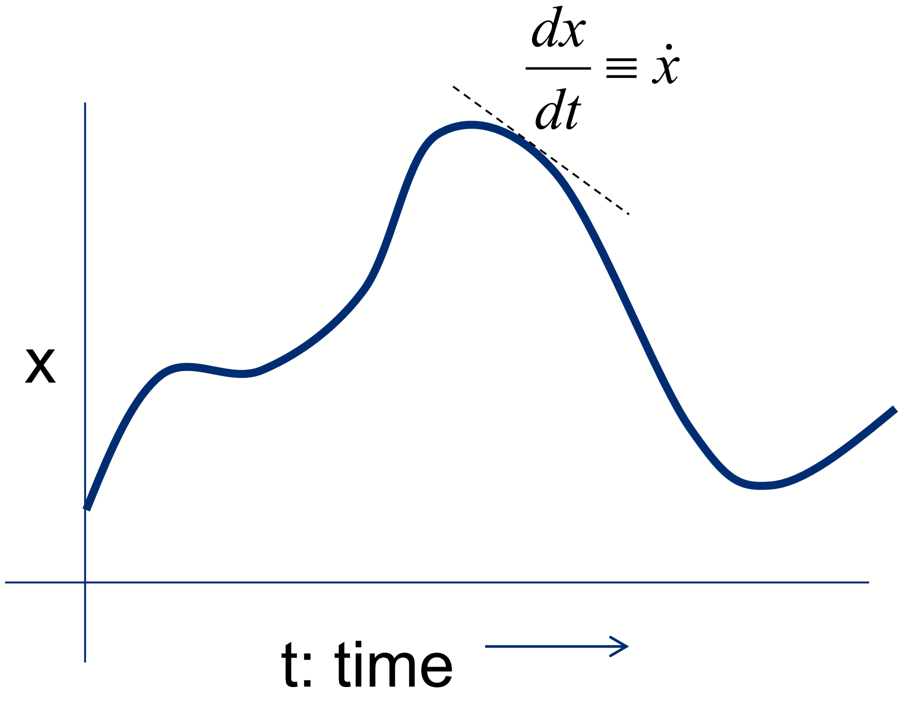
Differential Equations

- We will use differential equations to describe the motion of systems as a function of time

$$\frac{dx}{dt} = f(x, \text{other variables})$$

or

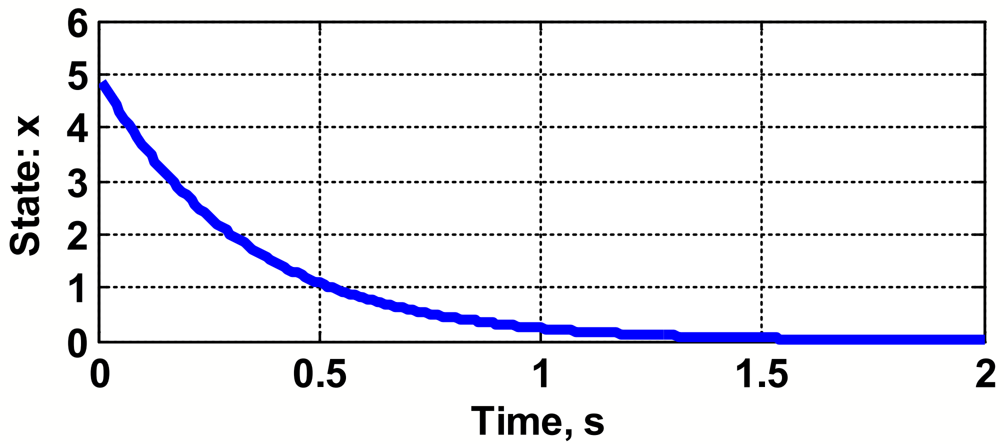
$$\dot{x} = f(x, \text{other variables})$$



Dynamic response example using Matlab

Simple script to emulate: $\dot{x} = -3x, \quad x(0) = 5$

```
% Initial state value
x = 5;

% Time starts at zero, increments by dt seconds
t = 0;
dt = 0.001;

% For retaining history
tHistory = [];
xHistory = [];

% Loop through time, stopping after 2 seconds
while t<2

    xdot = -3*x;              % Define state derivatives

    x = x + xdot*dt;          % Propagate state (simple Euler integration)

    t = t + dt;               % Increment time

    tHistory(end+1) = t;      % Retain history
    xHistory(end+1) = x;
end

% Plot
plot(tHistory, xHistory);
```



Coupled Differential Equations: Characteristics

The dynamical response of a system may be:

- Coupled between 2 or more variable "states"

$$\dot{x}_1 = f_1(x_1, x_2) \qquad \dot{x}_2 = f_2(x_1, x_2)$$

Example:

$$\dot{x}_1 = 4x_1 + 10x_2 \qquad \dot{x}_2 = -3x_1$$

- A function of one or more input stimuli:

$$\dot{x}_1 = f_1(x_1, x_2, u_1, u_2) \qquad \dot{x}_2 = f_2(x_1, x_2, u_1, u_2)$$

Example:

$$\dot{x}_1 = 4x_1 + 10x_2 + u_1 - 2u_2 \qquad \dot{x}_2 = -3x_1 + 5u_1$$

- Linear (i.e. Derivatives are a linear combination of variables & inputs)

Examples:

$$\dot{x}_1 = 4x_1 + 10x_2 + u_1 - 2u_2 \qquad \dot{x}_2 = -3x_1 + 5u_1$$

$$\dot{x}_1 = a_{11}x_1 + a_{12}x_2 \quad + \quad b_{11}u_1 + b_{12}u_2$$
$$\dot{x}_2 = a_{21}x_1 + a_{22}x_2 \quad + \quad b_{21}u_1 + b_{22}u_2$$

where a_{ij} and b_{ij} are constants

- Non-Linear

Examples:

$$\boxed{\begin{aligned}\dot{x}_1 &= 4x_1 + x_1x_2 \\ \dot{x}_2 &= -3/x_1\end{aligned}} \qquad \boxed{\begin{aligned}\dot{x}_1 &= 4x_1 + \tan x_2 \\ \dot{x}_2 &= -3x_1^2\end{aligned}} \qquad \boxed{\begin{aligned}\dot{x}_1 &= a_{11}x_1 + a_{12}\sqrt{x_2} \quad + \quad b_{11}u_1 + b_{12}u_2^3 \\ \dot{x}_2 &= a_{21}x_1 + a_{22}x_2 \quad + \quad b_{21}u_1u_2 + b_{22}x_1u_2\end{aligned}}$$

State Space Models

- Given a set of linear and coupled differential equations, we can build a linear mapping between the *states* and the *state derivatives*:

$$\dot{x}_1 = a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \quad + \quad b_{11}u_1 + b_{12}u_2$$
$$\dot{x}_2 = a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \quad + \quad b_{21}u_1 + b_{22}u_2$$
$$\dot{x}_3 = a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \quad + \quad b_{31}u_1 + b_{32}u_2$$

where:

x_j is the jth system state

a_{ij} and b_{ij} are constants

u_j is the jth input

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}}_{A} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \underbrace{\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}}_{B} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Inputs can be:
- zero
- constant
- A function of time

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = A \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + B \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \Rightarrow \underline{\dot{x}} = A\underline{x} + B\underline{u}$$

State Space Models: A, B, C, D

- In a state space model, the A and B matrices completely describe the dynamical motion of the states given:
  - An initial condition of the states
  - Time-varying inputs (stimuli)

$$\dot{\underline{x}} = A\underline{x} + B\underline{u} \qquad \underline{x}(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \end{bmatrix} \qquad \underline{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} f_{u1}(t) \\ f_{u2}(t) \end{bmatrix}$$

- A state space model can also have one or more outputs, which are a linear combinations of states and inputs

$$y_1 = c_{11}x_1 + c_{12}x_2 + c_{13}x_3 \quad + \quad d_{11}u_1 + d_{12}u_2$$

$$y_2 = c_{21}x_1 + c_{22}x_2 + c_{23}x_3 \quad + \quad d_{21}u_1 + d_{22}u_2$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\Rightarrow \quad \underline{y} = C\underline{x} + D\underline{u}$$

$\underline{y}$ is a combination of states and inputs

- $\underline{y}$ has no bearing on the dynamics of the states
- Think of $\underline{y}$ as an "observation" of some combination of states and inputs

D is often zero

- When output is purely a linear combination of states

State Space response example using Matlab

Simple script to emulate:

$$\underline{\dot{x}} = \begin{bmatrix} -1 & -2 \\ 1 & -1 \end{bmatrix}\underline{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\underline{u}$$

$$y = \begin{bmatrix} 2 & -1 \end{bmatrix}\underline{x} + 0\underline{u}$$

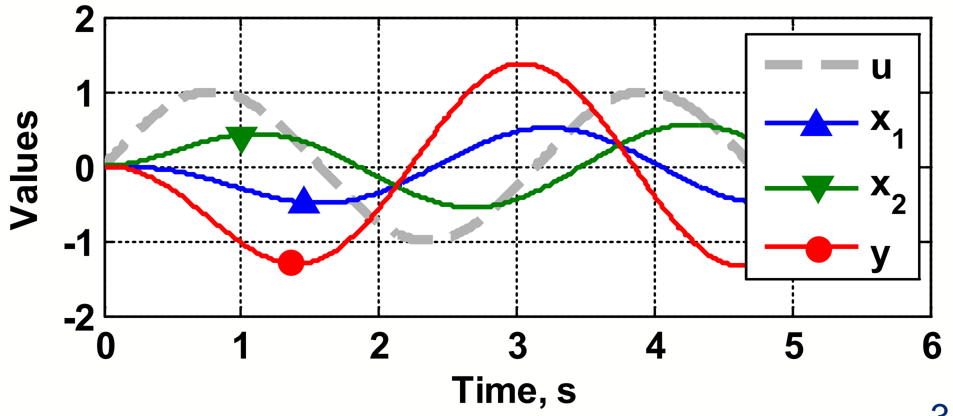
```
% System
A = [-1 -2; 1 -1];
B = [0; 1];
C = [2 -1];
D = 0;

% Initial state value
x = [0; 0];   % x is 2x1

% Time starts at zero,
% increments by dt sec
t = 0;
dt = 0.001;

% For retaining history
tHistory = [];
xHistory = [];
yHistory = [];
uHistory = [];
```

```
% Loop through time, stopping after 5 seconds
while t < 5
    u = sin(2*t);            % Input function

    xdot = A*x + B*u;        % Define state derivatives

    x = x + xdot*dt;         % Propagate state (simple Euler integration)

    y = C*x + D*u;           % Output is a combination of x and u

    t = t + dt;              % Increment time

    tHistory(end+1) = t;     % Retain history
    xHistory(end+1,:) = x';
    yHistory(end+1) = y;
    uHistory(end+1) = u;
end
```

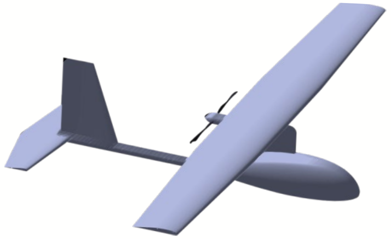
Note the colon:

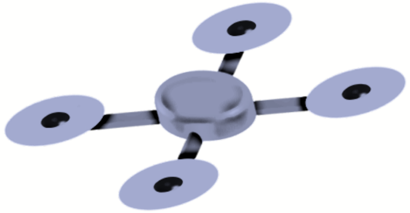
```
xHistory(end+1,:) = x';
```

The colon here indicates "all rows". So, if x is 2-by-1, xHistory will be an n-by-2 array.



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING



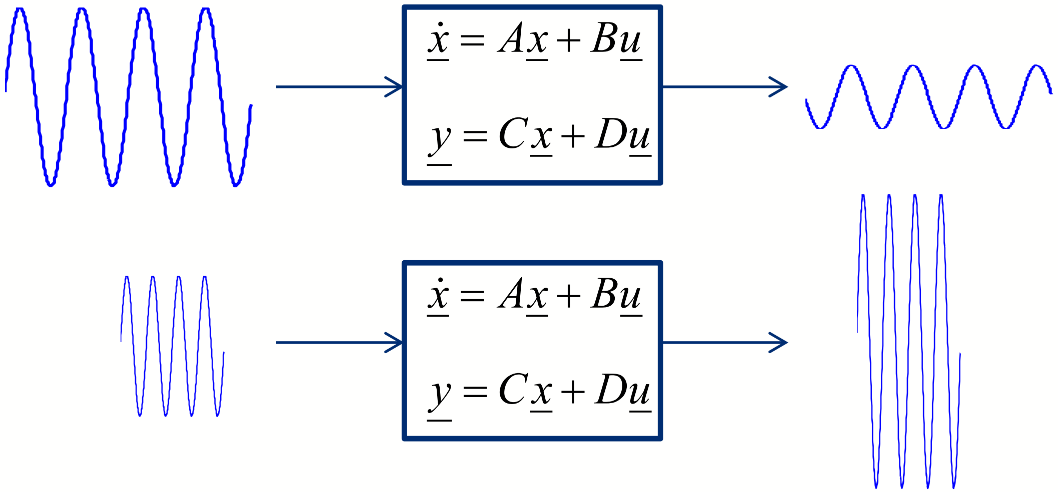


UAV Systems & Control, Module 1E

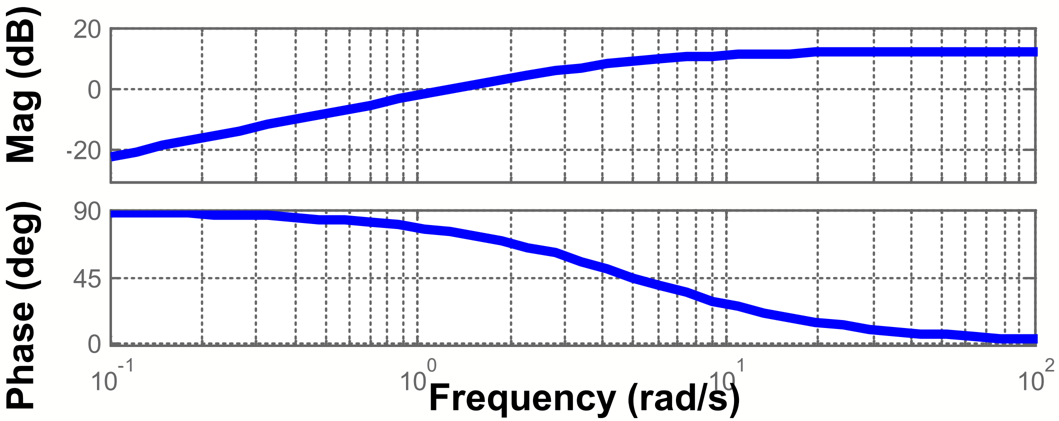
- Frequency Domain
- Laplace Transfer Functions
- Stability
- 1^{st} & 2^{nd} Order Systems

Frequency Domain

- An important property of linear, time-invariant systems is that if the input is sinusoidal, the output will be sinusoidal at the same frequency (with a generally different magnitude and phase)



- Thus, we can also view systems in a frequency domain:



Laplace Transform

- We use the Laplace Transform to convert a time-domain system into the frequency domain
  - The Laplace Transform of a time domain function is: $F(s) = \mathcal{L}\{f(t)\} = \int_0^\infty e^{-st} f(t)dt \longleftarrow$ *Just a definition. We don't actually do this.*
- In Laplace Domain "s" signifies frequency: $s = j\omega$, radians/sec
- The Laplace of the time derivative of a function is useful:

$$\mathcal{L}\{f(t)\} = F(s) \quad \Rightarrow \quad \mathcal{L}\{\dot{f}(t)\} = sF(s) - f(0)$$

Time Domain		*Laplace Domain*
$\dot{x}$	$\Leftrightarrow$	$sX(s)$

- As a result, we can express differential equations as algebraic equations:

$$\begin{aligned}\dot{x}_1 &= 4x_1 + 10x_2 + 2u \\ \dot{x}_2 &= -3x_1\end{aligned}$$

$$\begin{aligned}sX_1(s) &= 4X_1(s) + 10X_2(s) + 2U(s) \\ sX_2(s) &= -3X_1(s)\end{aligned}$$

$$sX_1(s) = 4X_1(s) + 10\left\{\frac{-3X_1(s)}{s}\right\} + 2U(s)$$

$$\left(s^2 - 4s - 30\right)X_1(s) = 2sU(s)$$

$$\frac{X_1(s)}{U(s)} = \frac{2s}{\left(s^2 - 4s - 30\right)}$$

$$\frac{X_2(s)}{U(s)} = \frac{-6}{\left(s^2 - 4s - 30\right)}$$

Transfer Functions

- In Laplace, we can conveniently view a SISO (Single-Input-Single-Output) system as a transfer function, defined as the ratio of output to input:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0}$$

- We can re-arrange the transfer function into zero-pole-gain form:

$$G(s) = \frac{Y(s)}{U(s)} = K \frac{(s - z_1)(s - z_2)\ldots(s - z_{m-1})(s - z_m)}{(s - p_1)(s - p_2)\ldots(s - p_{n-1})(s - p_n)}$$

  - "Zeros" are the roots of the numerator polynomial
  - "Poles" are the roots of the denominator polynomial (aka eigenvalues)
  - Both zeros and poles may be real or complex!

Note:
MIMO (Multi-Input-Multi-Output) systems will simply become an array of transfer functions from each input to each output,
e.g.:

$$\begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \\ G_{31}(s) & G_{32}(s) \end{bmatrix}$$

- A State-Space model (A,B,C,D) can be converted to a Transfer Function:

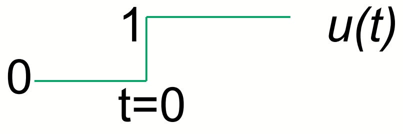
$$G(s) = \tfrac{Y(s)}{U(s)} = C(sI - A)^{-1} B + D$$

Can you show?

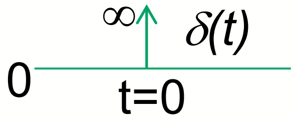
$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \Rightarrow \tfrac{Y(s)}{U(s)} = C(sI - A)^{-1} B + D$$

Laplace & Transfer Function Properties

- Linearity: $ax(t)+bv(t)\leftrightarrow aX(s)+bV(s)$
- Differentiation: $\dot{x}(t)\leftrightarrow sX(s)-x(0)$
- Integration: $\int_0^t x(\tau)d\tau \leftrightarrow \frac{1}{s}X(s)$
- Unit Step: $u(t)\leftrightarrow \frac{1}{s}$



- Impulse: $\delta(t)\leftrightarrow 1$



- Exciting a system: $Given: G(s)=\frac{Y(s)}{U(s)}\rightarrow Y(s)=G(s)U(s)\rightarrow y(t)=\mathcal{L}^{-1}\left(G(s)U(s)\right)$
- Commutativity: $H(s)G(s)=G(s)H(s)$
- Initial Value Theorem: $x(0)=\lim_{s\to\infty} sX(s)$
- Final Value Theorem: $\lim_{t\to\infty} x(t)=\lim_{s\to 0} sX(s)$

Provided that X(s) is rational and its poles are strictly negative

$s: Derivative$

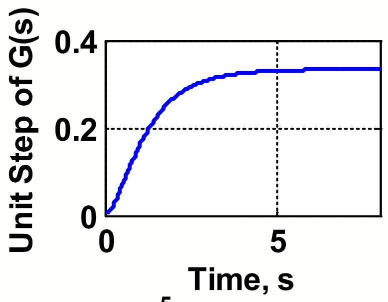
$\frac{1}{s}: Integration$

Stability

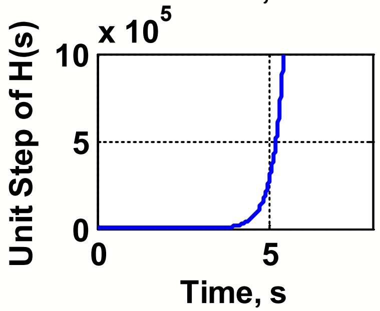
- A system is considered "stable" if a bounded (i.e. finite) input results in a bounded (i.e. finite) output
- For a Laplace Transfer Function, a system is stable if all the poles (roots of the denominator) are negative

$$G(s) = \frac{Y(s)}{U(s)} = K\frac{(s-z_1)(s-z_2)\ldots(s-z_{m-1})(s-z_m)}{(s-p_1)(s-p_2)\ldots(s-p_{n-1})(s-p_n)} \longleftarrow \text{Stable only if all: } p_1 < 0,\quad p_2 < 0,\quad \cdots \quad p_n < 0$$

- Stable example: $\begin{matrix} p_1 = -1 \\ p_2 = -3 \end{matrix} \rightarrow G(s) = \dfrac{1}{(s-\{-1\})(s-\{-3\})} = \dfrac{1}{(s+1)(s+3)}$



- Unstable example: $\begin{matrix} p_1 = -1 \\ p_2 = +3 \end{matrix} \rightarrow H(s) = \dfrac{1}{(s-\{-1\})(s-\{+3\})} = \dfrac{1}{(s+1)(s-3)}$



First Order Systems

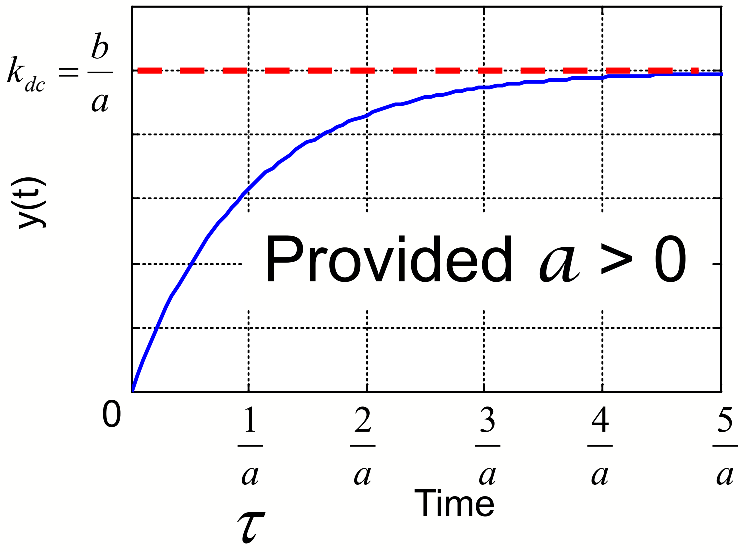
First Order Systems are the simplest dynamic systems:

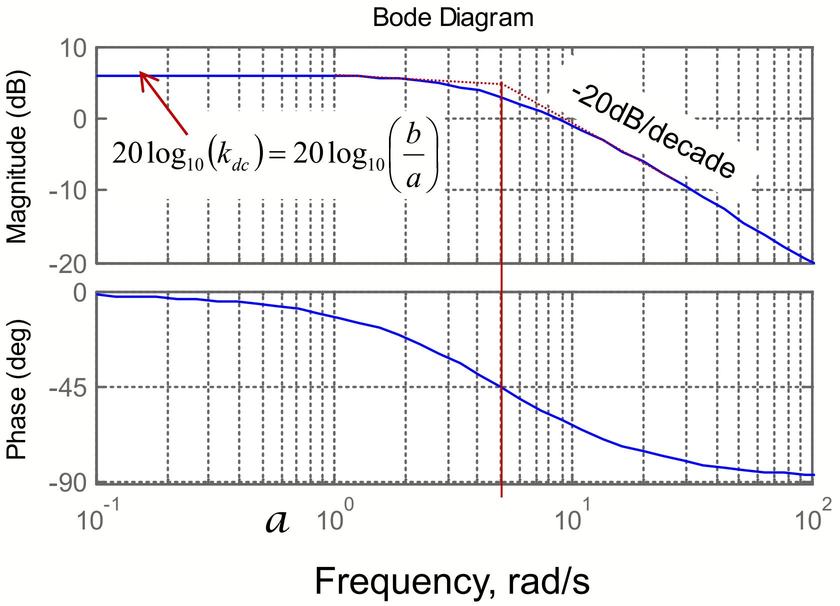
$$\dot{y} = -ay + bu \longrightarrow \boxed{\frac{Y(s)}{U(s)} = \frac{b}{s+a} = \frac{k_{dc}}{\tau s + 1}}$$

$k_{dc} = \frac{b}{a}$ Steady - State Gain

$\tau = \frac{1}{a}$ Time Constant (63% Rise Time)

Step Response: $y(t) = k_{dc}\left(1 - e^{-t/\tau}\right)$





Matlab:

```
s=tf('s'); % Make s
step(b/(s+a))
```

```
bode(b/(s+a))
```

Second Order Systems

Second Order Systems exhibit oscillations:

$$\ddot{y} + a_1 \dot{y} + a_0 y = bu \quad \longrightarrow \quad \frac{Y(s)}{U(s)} = \frac{b}{s^2 + a_1 s + a_0} = \frac{b}{(s - p_1)(s - p_2)}$$

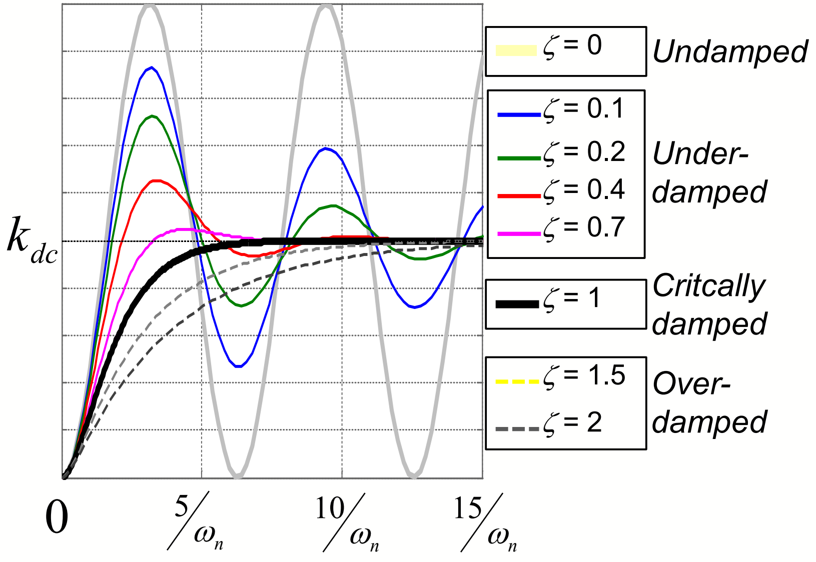
$$p_1 = \tfrac{1}{2}\left(-a_1 + \sqrt{a_1^2 - 4a_0}\right)$$
$$p_2 = \tfrac{1}{2}\left(-a_1 - \sqrt{a_1^2 - 4a_0}\right)$$

Conveniently re-written using natural frequency (ω_n) and damping (ζ)

$$\ddot{y} + 2\zeta\omega_n \dot{y} + \omega_n^2 y = k_{dc}\omega_n^2 u \quad \longrightarrow \quad \boxed{\frac{Y(s)}{U(s)} = \frac{k_{dc}\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}}$$

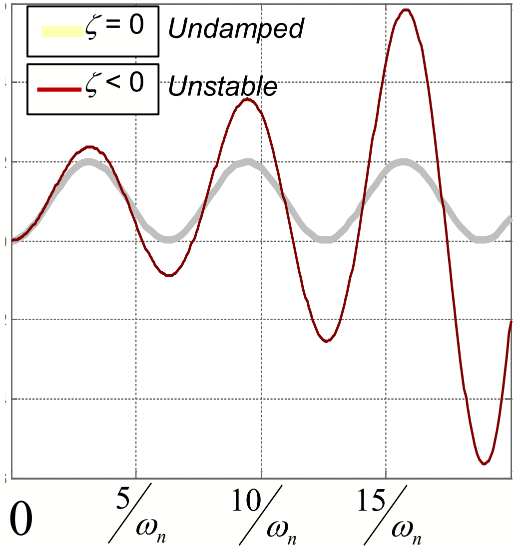
$$p_1 = -\zeta\omega_n + j\omega_n\sqrt{1-\zeta^2}$$
$$p_2 = -\zeta\omega_n - j\omega_n\sqrt{1-\zeta^2}$$

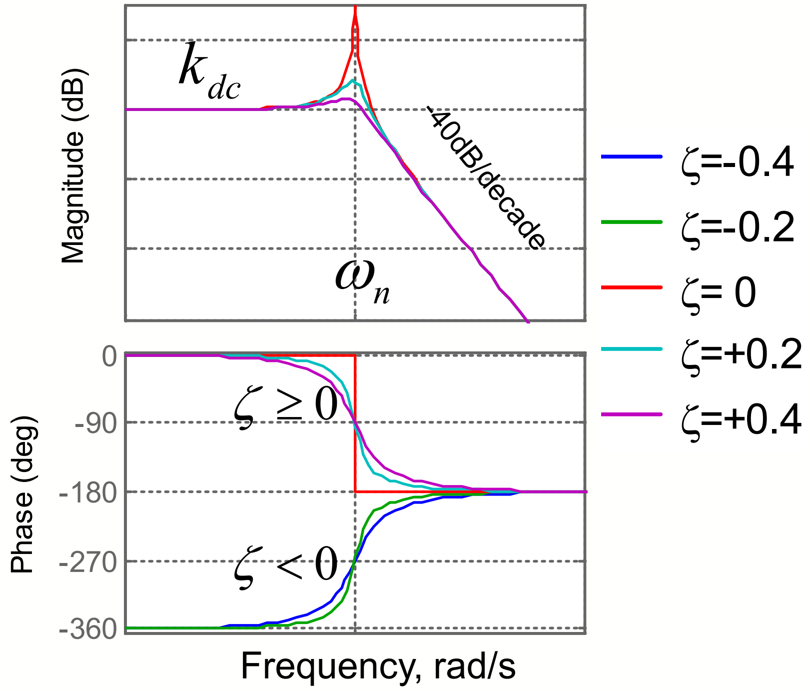
Stable Steps Responses




```
step(wn^2/(s^2+2*zeta*wn*s+wn^2))
```

Unstable Steps Response

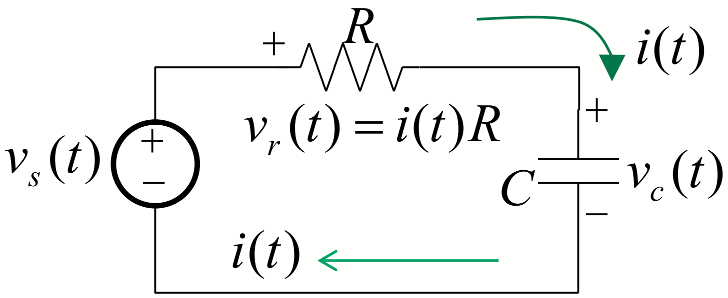





```
bode(wn^2/(s^2+2*zeta*wn*s+wn^2))
```

1st and 2nd Order Examples

First Order Example: RC Circuit



Given: R, C, and $v_s(t)$
Want: $v_c(t)$ (Voltage across cap.)

Capacitor: $i(t) = C\dfrac{dv_c(t)}{dt}$

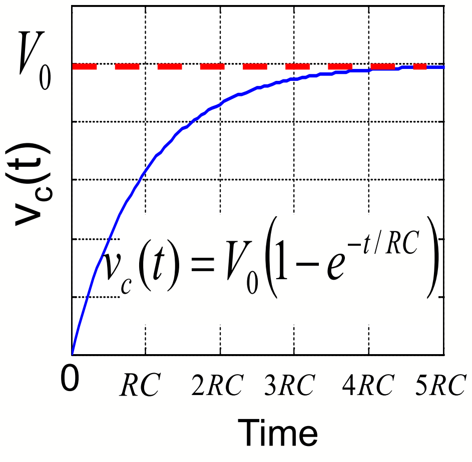
$$v_s(t) = v_r(t) + v_c(t)$$
$$v_s(t) = R \cdot i(t) + v_c(t)$$
$$v_s(t) = RC\tfrac{d}{dt}v_c(t) + v_c(t)$$

$$V_S(s) = RCsV_C(s) + V_C(s)$$
$$V_S(s) = (RCs+1)V_C(s)$$

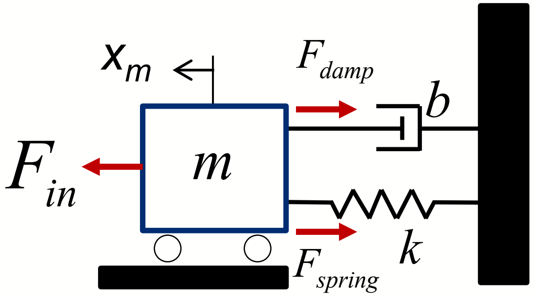
$$\Rightarrow \frac{V_C(s)}{V_S(s)} = \frac{1}{RCs+1}$$

$$\tau = RC \quad \text{seconds}$$

Capacitor voltage due to a step source:



Second Order Example: Spring Damper System



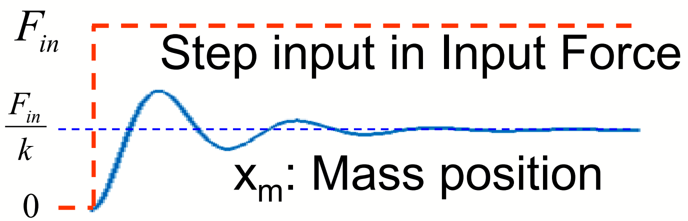
x_m : Mass Pos.
m : mass
$F_{damp} = b\dot{x}_m$
$F_{spring} = kx_m$
F_{in} = Input Force

Newton's Law: Sum of Forces = mass * accel

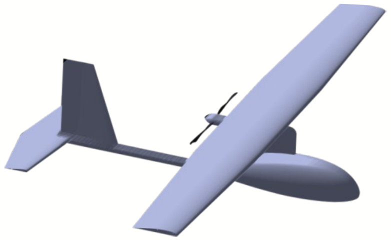
$$F_{in}(t) - F_{damp} - F_{spring} = m\ddot{x}_m$$

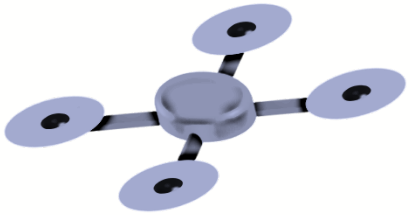
1) Plug in values for F_{damp} and F_{spring}
2) Collect x_m terms: $?\ddot{x}_m + ?\dot{x}_m + ?x_m = F_{in}(t)$
3) Convert to Laplace

$$\frac{X_m(s)}{F_{in}(s)} =$$ Make Transfer Function (Needed in HW)



JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING



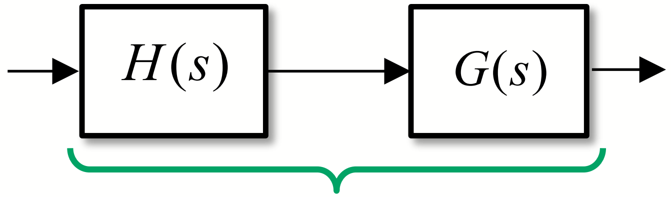


UAV Systems & Control, Module 1F

- Block Diagrams
- Useful Matlab Commands
- Module Summary

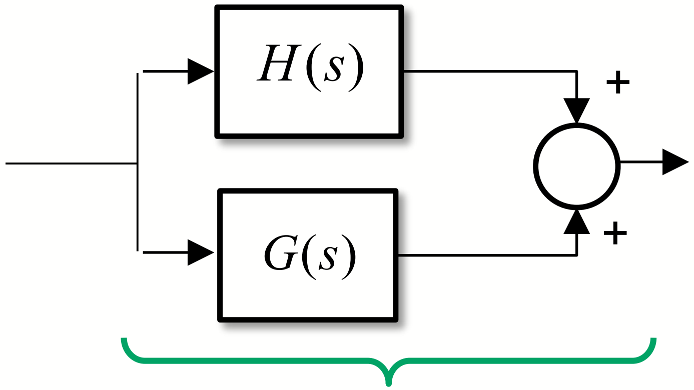
Block Diagrams

- Laplace Transfer Functions make block diagrams useful



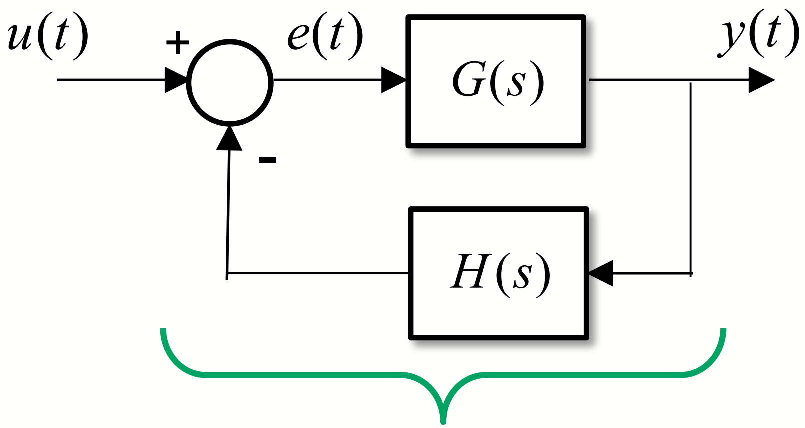
$H(s)G(s)$

For SISO systems: $H(s)G(s) = G(s)H(s)$



$H(s)+G(s)$

- Negative Feedback



$$\frac{G(s)}{1+G(s)H(s)} = \frac{\{FwdPath\}}{1+\{FwdPath\}\{FeedbackPath\}}$$

Proof:

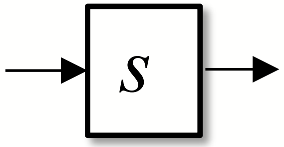
$$E(s) = U(s) - G(s)H(s)E(s)$$

$$\left(1+G(s)H(s)\right)E(s) = U(s)$$

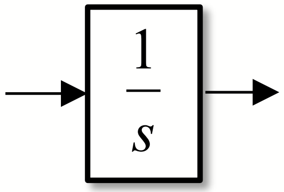
$$\frac{E(s)}{U(s)} = \frac{1}{1+G(s)H(s)}$$

$$\frac{Y(s)}{U(s)} = \frac{G(s)}{1+G(s)H(s)}$$

Derivative:

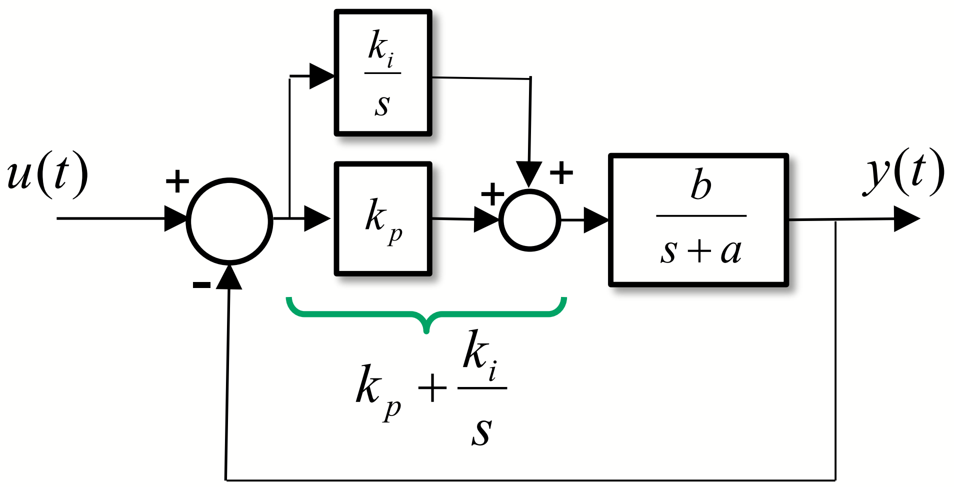


Integral:



Block Diagrams Feedback Example

- Example



$$\frac{Y(s)}{U(s)} = \frac{\{FwdPath\}}{1+\{FwdPath\}\{FeedbackPath\}} = \frac{\left(k_p + \frac{k_i}{s}\right)\left(\frac{b}{s+a}\right)}{1+\left(k_p + \frac{k_i}{s}\right)\left(\frac{b}{s+a}\right)} \longleftarrow \text{Multiply by } 1: \frac{s(s+a)}{s(s+a)}$$

$$\frac{Y(s)}{U(s)} = \frac{s(s+a)\left(k_p + \frac{k_i}{s}\right)\left(\frac{b}{s+a}\right)}{s(s+a)+s(s+a)\left(k_p + \frac{k_i}{s}\right)\left(\frac{b}{s+a}\right)} = \frac{(k_p s + k_i)(b)}{s(s+a)+(k_p s + k_i)(b)} = \frac{bk_p s + bk_i}{s^2 + (a + bk_p)s + bk_i}$$

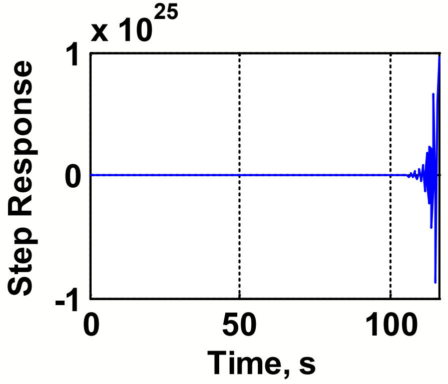
Useful Matlab Commands for Controls

```
% Create Transfer Functions and State Space Models

    s = tf('s');    % Make a Laplace s

    G = (2*s+5)/(4*s^2+7*s+2);    % Make a Transfer Function

    num = [2 5];
    den = [4 7 2];
    G = tf(num,den);      % Another way to make a Transfer Function

    G = ss(A,B,C,D);      % Create a State Space model from [A B C D]

% Convert between Transfer Functions, Zero-Pole-Gain form, and State Space

    G = tf(G);     % Convert to a Transfer Function
    G = ss(G);     % Convert to a State Space Model
    G = zpk(G);    % Convert to Zero-Pole-Gain form

    I=eye(size(A));
    G = C*inv(s*I-A)*B+D;  % Another conversion to a Transfer Function

    [z p k] = zpkdata(G,'v');  % Extract zeros, poles and gain ('v' to return vectors)

    [num den] = tfdata(G,'v'); % Extract numerator and denominator coefficients

    [A B C D] = ssdata(G);   % Extract State Space matrices

    G = minreal(G);       % Reduce to minimum realization transfer function
                          % (Performs pole/zero cancellations)
```

More Useful Matlab Commands for Controls

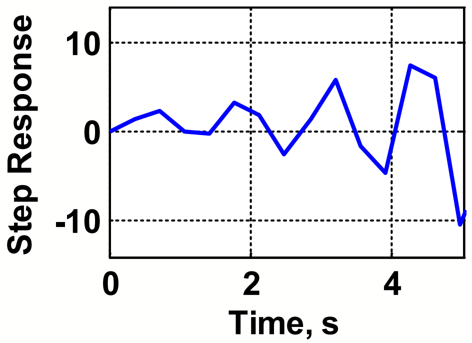
```
% Plotting and outputs

t=0:.1:10;
T=10;

step(G);          % Plot step response
step(G,H);        % Plot multiple step responses
step(G,T);        % Specify duration (first T seconds)
y=step(G,t);      % Output step response at times t

plot(t,step(G,t)); % Another way to plot a step response

impulse(G);        % Plot impulse response
impulse(G,H);      % Plot multiple impulse responses
impulse(G,T);      % Specify duration (first T seconds)
y=impulse(G,t);    % Output impulse response at times t

u = 4*sin(3*t);
lsim(G,u,t);      % Plot/Output response of G stimulated
y=lsim(G,u,t);    %     by input vector u at times t

bode(G)          % Plot Bode response
bode(G,H)        % Plot multiple Bode responses
```

```
s=tf('s');
wn=5; zeta = -0.1;
G = wn^2/(s^2+2*zeta*wn*s+wn^2);

step(G)   % Be Careful Trusting Time!
```

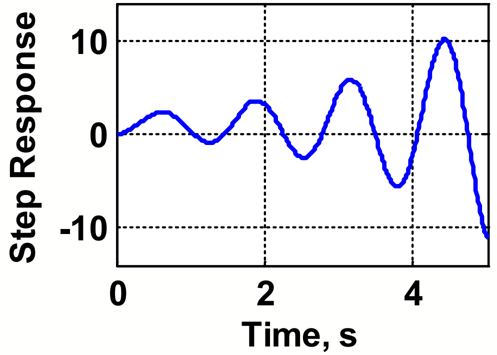


Zoomed in

```
xlim([0 5])
```



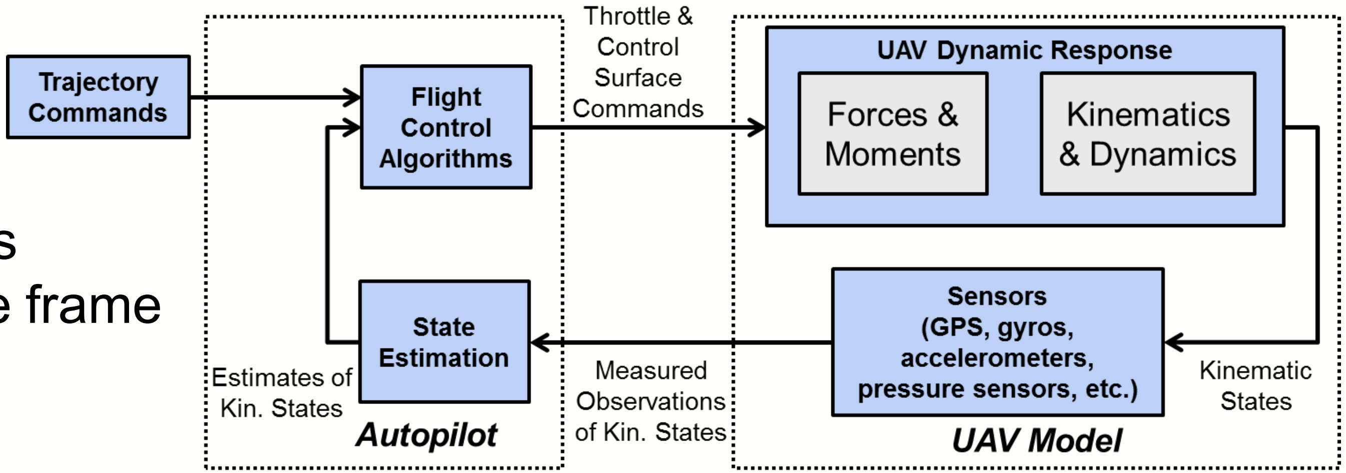

```
step(G,5)   % Better to specify time!
```



Module 1 Summary

- **In this module, we learned:**
  - Physical and system components in a UAV feedback control system
  - Basic vector geometry tools that are needed for 3D modeling of vehicle motion
    - *Vectors, Matrices, Coordinate Frames, Rotation Matrices, Euler Angles*
  - Aircraft velocity and angle definitions
    - *Wind Triangle relating airspeed, windspeed and groundspeed*
    - *Euler Angles representing aircraft body orientation wrt NED*
    - *Aerodynamic angles*
    - *Velocity vector angles*
  - Multiple ways to represent system dynamics
    - *Differential Equations, State Space, Laplace Domain, Block Diagrams*
    - *1st & 2nd order systems, and stability*

- **Next module:**
  - Refresher on Feedback Control Systems
  - Vector Geometry in a rotating coordinate frame



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING