Autonomous Flight and Stability Control

Capstone Presentation
By Kyle Gavin

Abstract Summary

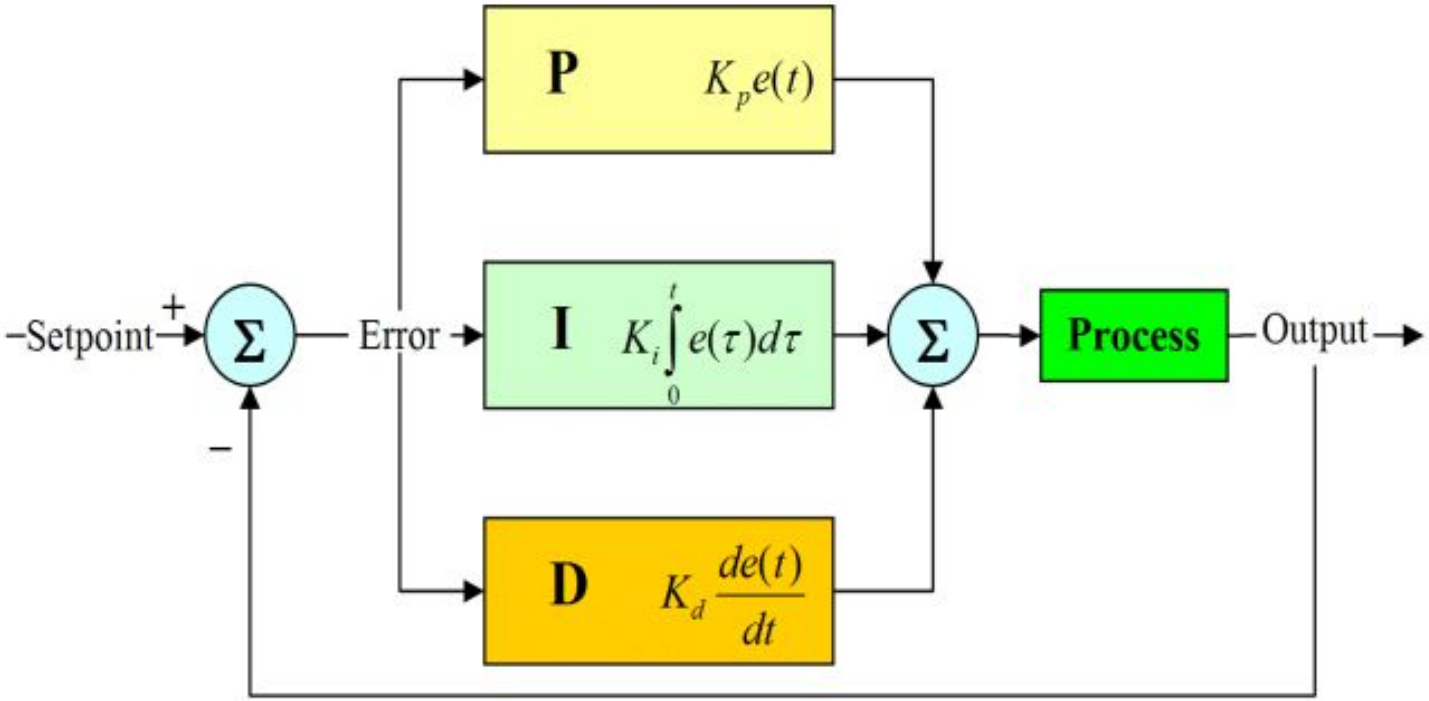
- Autonomous flight in aircraft vehicles is **a collection of computer-controlled states** coupled by an input/output loop to acquire desired control output to achieve state directives.
- The projected implements PID Controllers to achieve longitudinal (pitch), and lateral (roll) stability along with various modes or directives.
- Components are implemented as Python classes, allowing the reuse of control elements as objects in the main control loop.
- X-Plane 11 simulator provides environment
- UDP protocols designed to interface with X-Plane at the required speeds

Outline

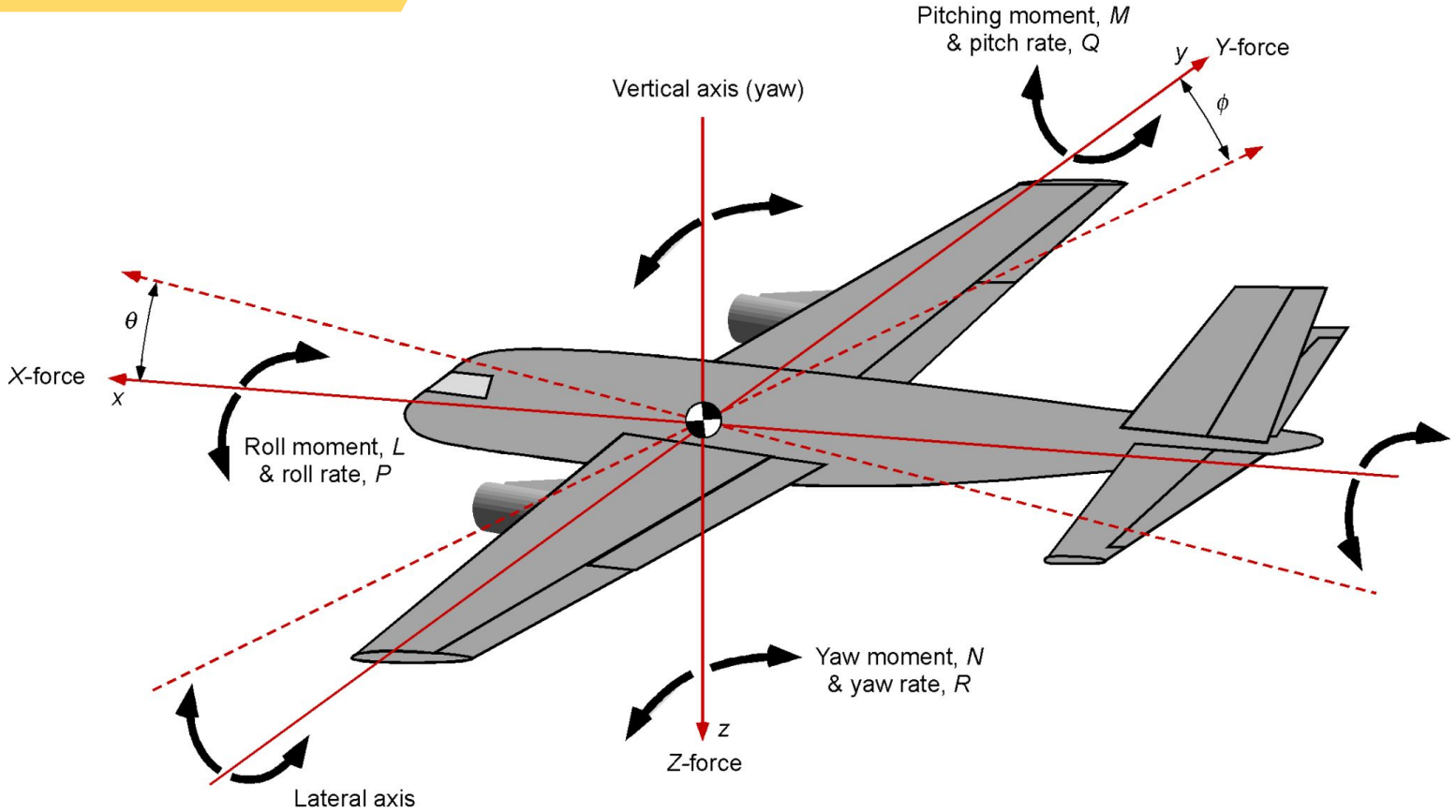
PID Controllers	• Relatable Example of a PID • System Diagram + Equation
Flight Dynamics	• Axes of movement + Free Body Diagram • Roll, Pitch, Raw + Thrust?
Interfacing w/ Sim	• What is UDP + Why UDP? • Datarefs + Data Types + Semantics
Roll + Pitch Stability	• Reiteration of PID • Graphical Analysis as calibration technique
Altitude + Heading dir	• Understanding degrees, turn direction • Intermediate functions
Navigation + Holding	• Haversine Equation • Advanced functions

Imagine This:

You're sitting in a car with cruise control on. The car goes up a hill. What happens?



$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$


Pitching moment, M
& pitch rate, Q
y
Y-force
ϕ
Vertical axis (yaw)
θ
X-force
x
Roll moment, L
& roll rate, P
Yaw moment, N
& yaw rate, R
z
Z-force
Lateral axis

Key Concepts of Flight Dynamics

Roll (lateral stab)

Imagine you're on your feet rocking left to right

- Ailerons
- Wings

Pitch (longitude stab)

Imagine you're leaning forwards and backwards

- Elevator
- Horizontal stabilizer

Yaw (directional stab)

Imagine you're pivoting left and right

- Rudder
- Vertical stabilizer

UDP? Why?

What is UDP?

UDP is a communication protocol for time sensitive information (our simulator params are very time sensitive!)

UDP Characteristics:

- Unordered Packets
- No Guarantee of Delivery
- Throwing Rocks

Why UDP?

The simulator has no API or interface for communication. UDP is the only gateway for system development not in the environment runtime!

Simulator Bottlenecks:

- No documentation
- Runtime is in C (no thanks!)

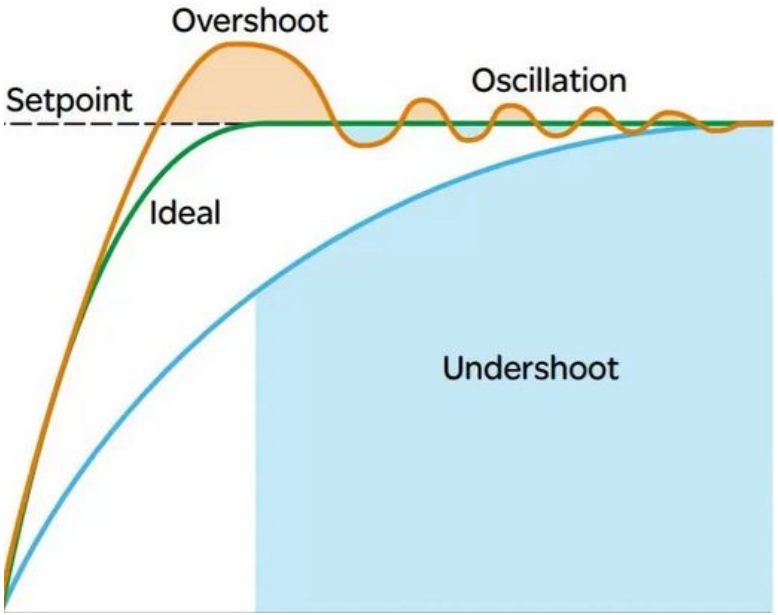
Datarefs!

- Over 1000 Parameters to interface with!
- Works like an API, data-points organized in a file like structure.
- Variables: Data Type? Writable? Unit Measure?

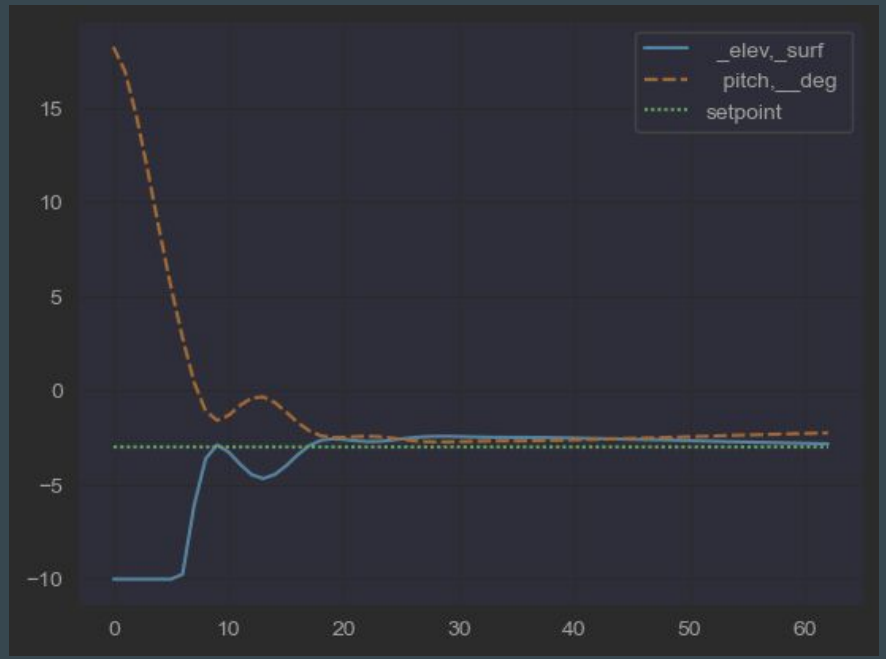
- `sim/flightmodel/position/y_agl` float, read-only, meters
- `sim/cockpit2/gauges/indicators/airspeed_kts_pilot` float, writable, knots
- `sim/cockpit2/gauges/indicators/heading_electric_deg_mag_pilot` float, writable, degrees-magnetic

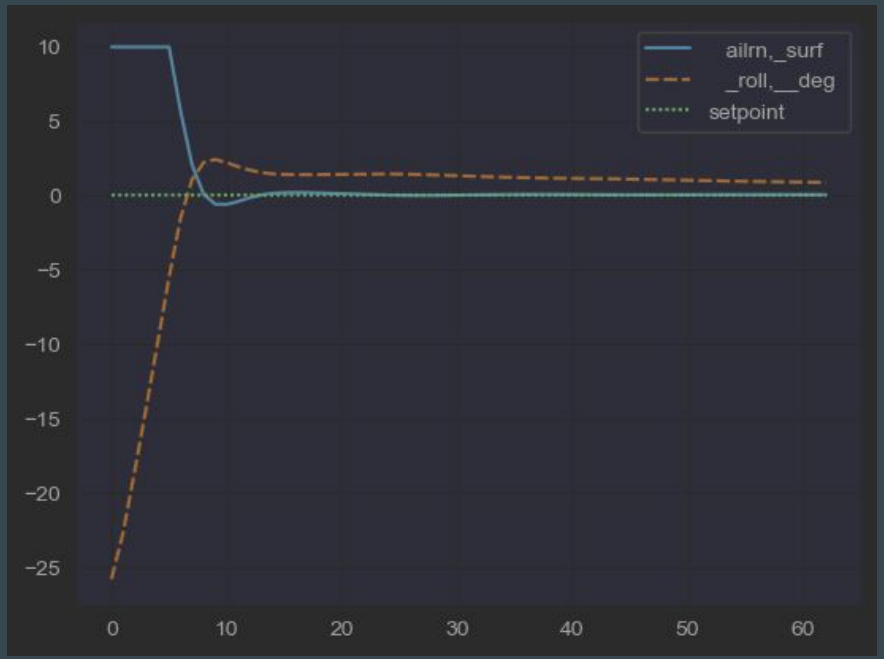
Stability Scenarios

- What is a set point?
- Overshoot: Over-reactive
- Undershoot: Not-reactive
- Oscillations? High/Low frequency
- How to achieve “Ideal scenario"?

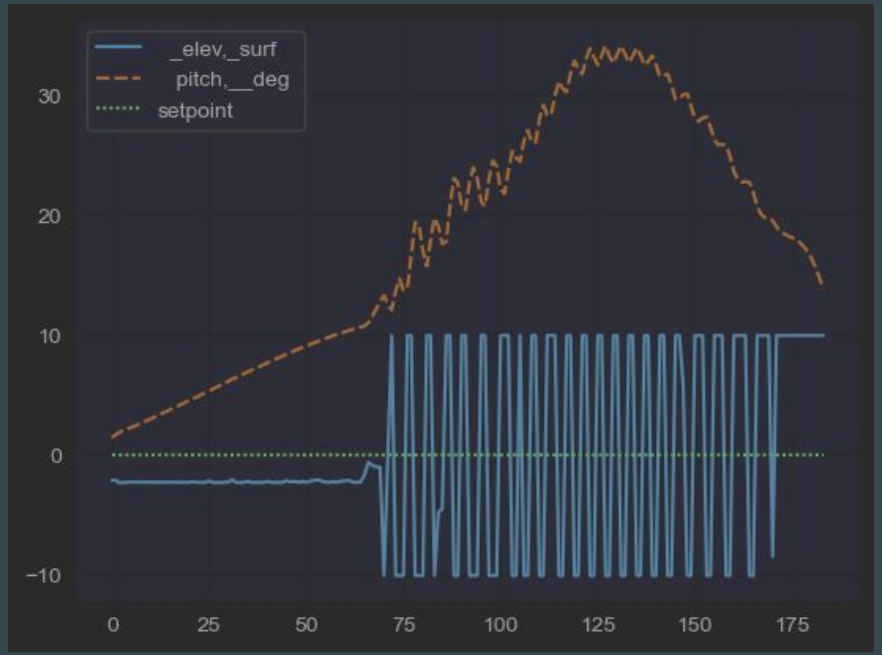


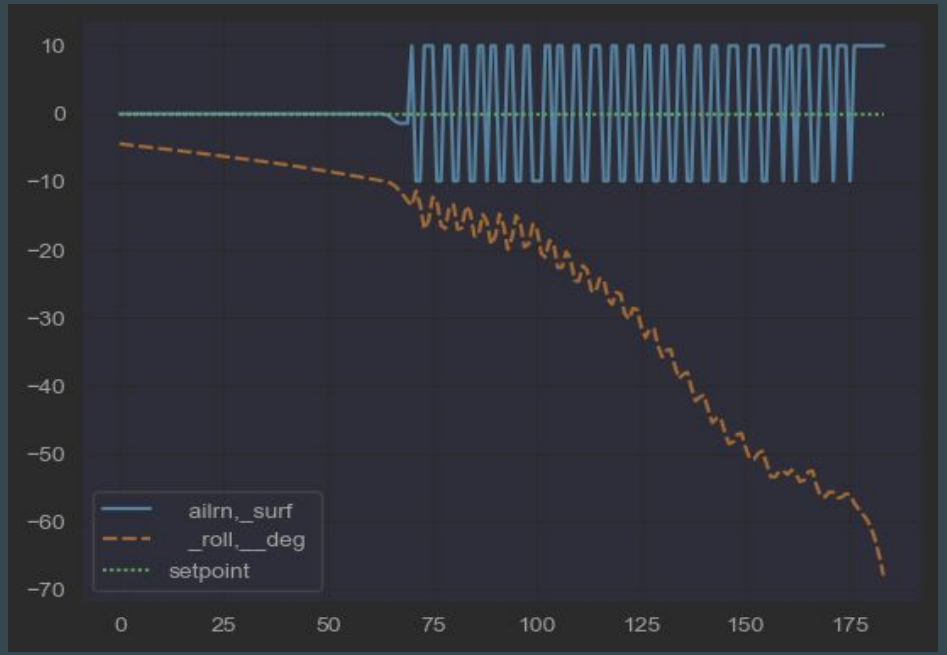
Graphical Analysis during Tuning - Relationships





Graphical Analysis during Tuning - Oscillations





Analysis during Tuning - Video of Roll Stab

Analysis during Tuning - Video of Pitch Stab

Analysis during Tuning - Oscillations

Intermediate Features

Altitude Capture/Hold

Manages the pitch controller to achieve desired longitudinal navigation (VNAV)

Characteristics:

- Normalization limits (-10/15 deg pitch)
- Coupled with Thrust/Airspeed
- Always seeking altitude

Heading Capture/Hold

Manages the roll controller to achieve desired lateral navigation (LNAV)

Characteristics:

- Normalization limits (+/-10 deg roll)
- Coupled with Rudder/Yaw axis
- Always seeking heading

Consider This:

How do you know when to turn Left or Right?

Determining Proper Heading

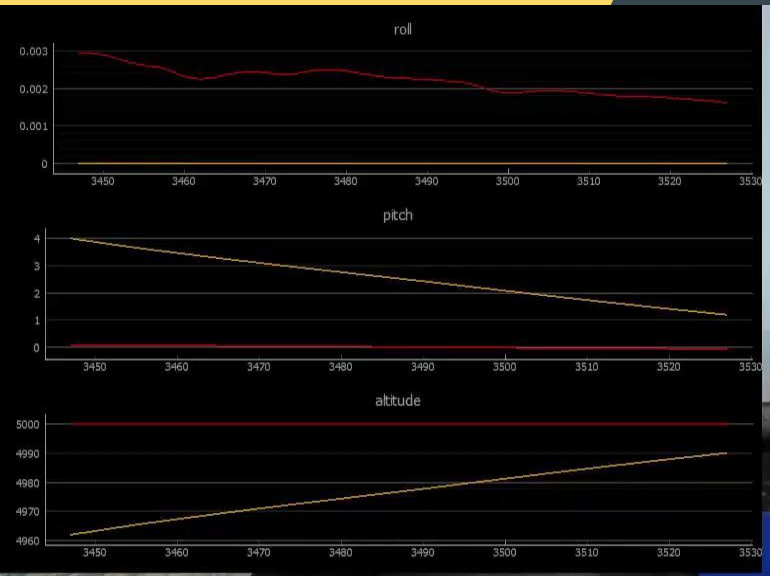
```
def find_turn_side(current, target) -> int:
    diff = target - current
    if diff < 0:
        diff += 360

    return -1 if diff > 180 else 1
```

Try with current=258 target = 030?

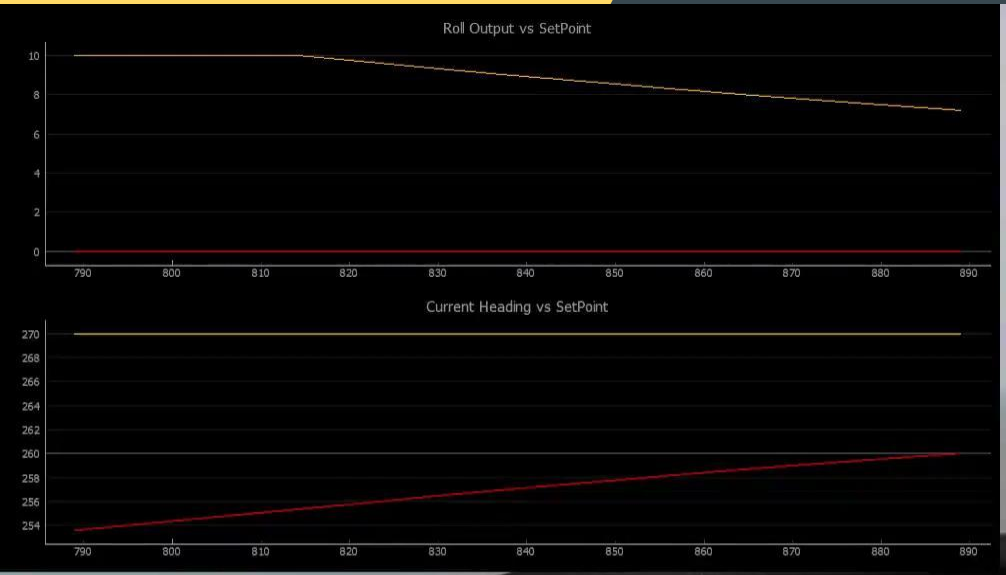
30 - 258 = -228
-228 < 0 = True
-228 + 360 = 132
132 > 180 = False
Right is Positive!

Level Off


roll
0.003
0.002
0.001
0
3450
3460
3470
3480
3490
3500
3510
3520
3530
pitch
4
3
2
1
0
3450
3460
3470
3480
3490
3500
3510
3520
3530
altitude
5000
4990
4980
4970
4960
3450
3460
3470
3480
3490
3500
3510
3520
3530

N172SP
HDG
ALT
COM
NAV
CRS
BARO
RANGE
PAN
FMS
X-PLANE 1000

Finish Turn


Roll Output vs SetPoint
10
8
6
4
2
0
790
800
810
820
830
840
850
860
870
880
890
Current Heading vs SetPoint
270
268
266
264
262
260
258
256
254
790
800
810
820
830
840
850
860
870
880
890

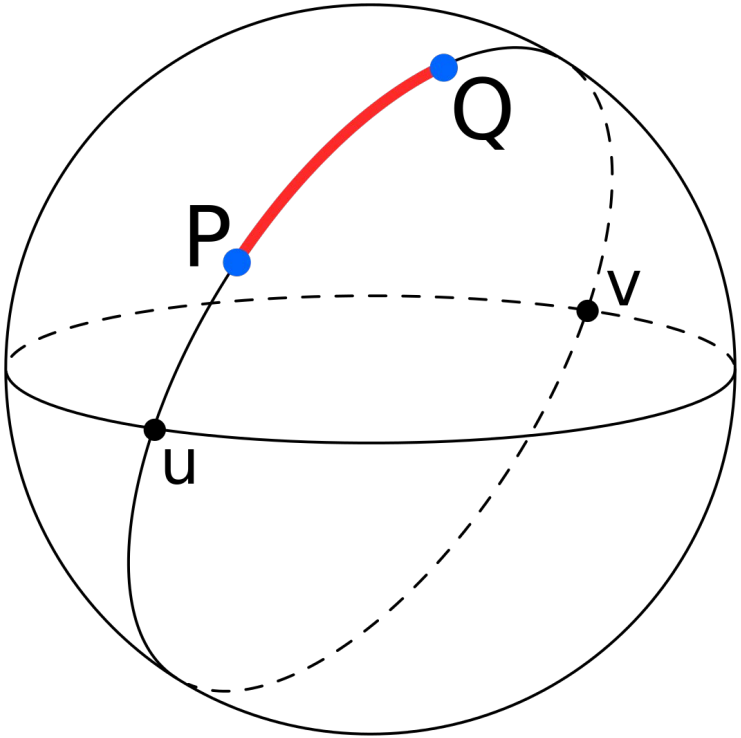

W
24


N172SP
NAV
HDG
COM
CRS
BARO
RANGE
PAN
X-PLANE 1000
250°

Determining Distance

```
dlon = lon2 - lon1
dlat = lat2 - lat1
a = sin(dlat / 2) ** 2 + cos(lat1) *
cos(lat2) * sin(dlon / 2) ** 2
c = 2 * asin(sqrt(a))
r = 6371  # km
return c * r
```

Not enough time for a demo here!



$$\Delta\hat{\sigma} = 2\arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_s\cos\phi_f\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

Determining Heading

- $\beta = \text{atan2}(X,Y)$
- $X = \cos \theta b * \sin \Delta L$
- $Y = \cos \theta a * \sin \theta b - \sin \theta a * \cos \theta b * \cos \Delta L$

Lets Try it with Dover(**39.161079, -75.525681**), DC (**38.889805, -77.009056**)

X= cos(39.161079) + sin(1.483375)

X = 0.80126053

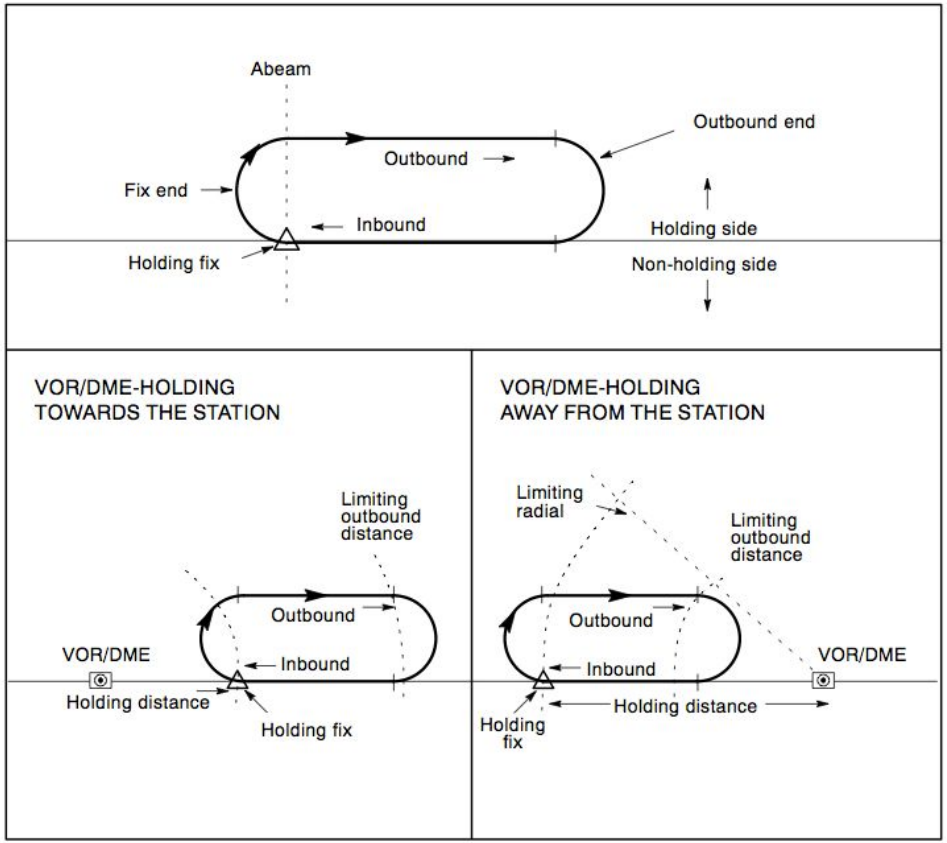
Y = cos(**38.889805**) * sin(**39.161079**) – sin(**38.889805**) * cos(**39.161079**) * cos(**1.483375**)

Y= 0.00489774

β = atan2(0.80126053 , 0.00489774) = 4.55 * 180/PI = Heading 260 From Dover!

Holding Attributes

- Holding Fix (lat/long)
- Altitude
- Leg Time/Dist
- Speed
- Time to Hold



Thanks for Listening!

•••

Questions?