Computational Methods and Modelling

Antonio Attili

antonio.attili@ed.ac.uk

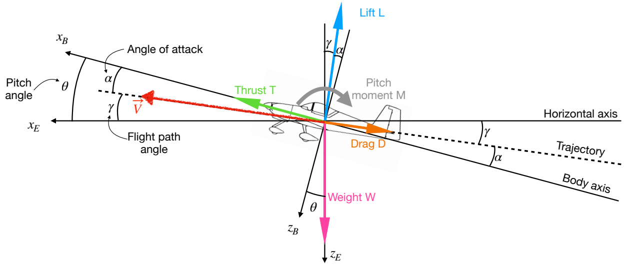
School of Engineering
University of Edinburgh
United Kingdom

Group Project
Longitudinal dynamics of a small aircraft

Project information

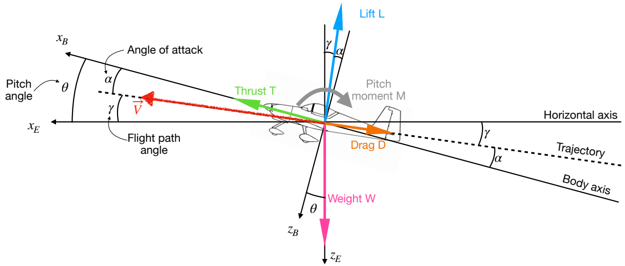
- **Goals of the Project**
  - Design and implement a simulation code for the longitudinal dynamics of a small aircraft
  - Use the developed code to perform simulations to address typical design tasks
- **Submission Instructions:**
  - Deadline: Tuesday 21 November, 4 PM
  - Submit a single ZIP file on Learn containing:
    - Overall Project Summary (Five page report).
    - Code and User Interface.
  - The design project accounts for 50% of the course grade
  - Marking Scheme for the Design Project (Rubrics):
    - 30% of the marks for the validity of the numerical models (code inspection).
    - 30% of the marks for the correctness of the solution (code testing).
    - 30% of the marks for the quality and clarity of the report
    - 10% of the marks for the efficiency and user-friendliness of the code (i.e., how concise and computationally efficient it is, easy to use, etc).
- **Important information**
  - In the header section of your codes you should give a brief set of statements to describe the codes capabilities and limitations.
  - Insert comments in all major subroutine lines explaining what is happening.
  - The code will be tested performing first a trim simulation and then a response to commands (more details in the following). A simple user interface is important for successful testing.
  - Include a short README file with your code to help the user to test your code.

Design Project: dynamics of a small airplane



- We consider the longitudinal motion of a small airplane with mass m (and weight W) and moment of inertia I_{yy}
- Only the movement in a vertical plane is considered. The plane moves forward in the x_E direction, it can move vertically in the z_E direction and can rotate around the y axis with angular velocity q.
- Two frames of reference are considered:
  - Earth axes x_E and z_E (with z_E pointing down)
  - Body axes x_B and z_B (with z_B pointing down)
- The state variable of the system are:
  - u_B, w_B: velocities in body axes
  - θ, q: pitch angle and angular velocity
  - x_E, z_E: position in the Earth axes ($z_E = 0$ on the Earth surface)
  - Altitude $h = -z_E$

Equations of Motion for an Airplane: 3 DoF model in body axes



Velocity equations

$$\frac{\mathrm{d}u_B}{\mathrm{d}t} = \frac{L}{m} sin\alpha - \frac{D}{m} cos\alpha - q w_B - \frac{W}{m} sin\theta + \frac{T}{m}$$

$$\frac{\mathrm{d}w_B}{\mathrm{d}t} = -\frac{L}{m} cos\alpha - \frac{D}{m} sin\alpha + q u_B + \frac{W}{m} cos\theta$$

Angular velocity and pitch angle equations

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{M}{I_{yy}}$$

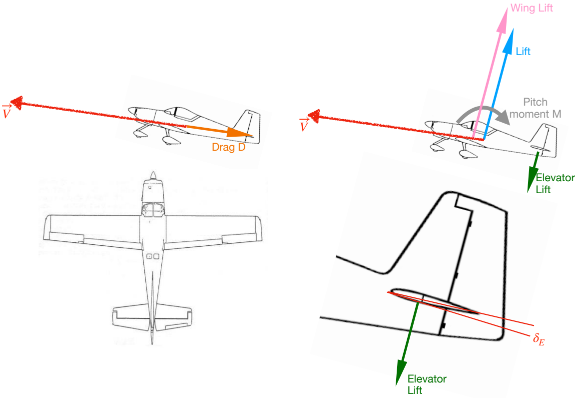
$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = q$$

Navigation equations

$$\frac{\mathrm{d}x_E}{\mathrm{d}t} = u_B cos\theta + w_B sin\theta$$

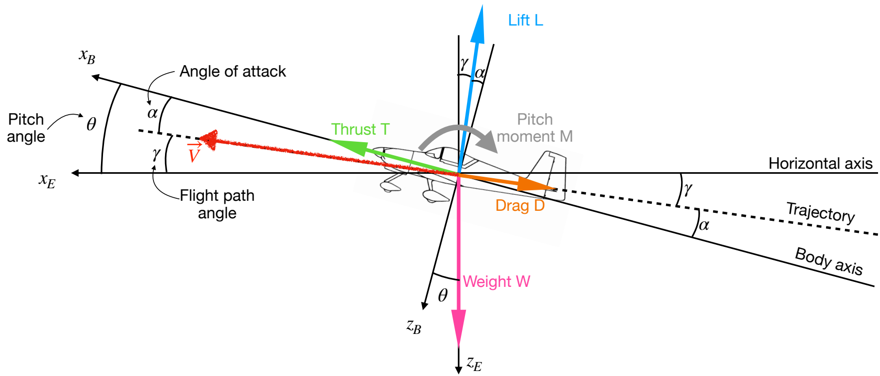
$$\frac{\mathrm{d}z_E}{\mathrm{d}t} = -u_B sin\theta + w_B cos\theta$$

Origin of forces and moments



- The elevator lift contributes to the total lift of the airplane L and to the pitch moment M
- Note that the sign of δ_E is negative when it contributes with a negative lift
- The **elevator angle** δ_E is one of the **two commands** we consider in this model (the other one is the **thrust** T)
- In general, the Lift L, Drag D, and Pitch Moment M are non-linear (and often unknown) functions of velocity V, angle of attack α and elevator angle δ_E: $L = L(V, \alpha, \delta_E)$, $M = M(V, \alpha, \delta_E)$, $M = M(V, \alpha, \delta_E)$. However, we will use simplified relations.

Reference frames for the aircraft dynamics and the direction of the forces



- In general, the Lift L and Drag D are NOT aligned with the earth axes x_E-z_E nor the body axes x_B-z_B.
- The Lift L is perpendicular and the Drag D is parallel to the velocity $\vec{V}$, which is the direction of the trajectory of the airplane motion.
- The Weight W always points towards the earth surface, so it is aligned with the earth axis z_E.
- The Thrust T is always aligned with the body axis x_B.

Definition of forces and moments

- Lift

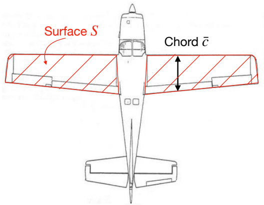
$$L = \frac{1}{2}\rho V^2 S C_L$$

- Drag

$$D = \frac{1}{2}\rho V^2 S C_D$$

- Pitch moment

$$M = \frac{1}{2}\rho V^2 S \bar{c} C_M$$



- where:
  - ρ is the air density and V is the velocity
  - S and $\bar{c}$ are the wing surface and the airfoil chord
  - C_L, C_D, and C_M are the aerodynamics coefficients
- The aerodynamics coefficients are often modelled using the simplified equations:

$$C_L = \underbrace{C_{L_0} + C_{L_\alpha}\alpha}_{C_L^{wing}} + \underbrace{C_{L_{\delta_E}}\delta_E}_{C_L^{el}}$$

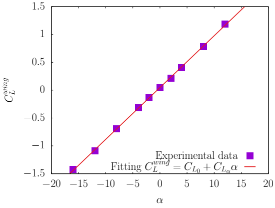
$$C_D = C_{D_0} + K C_L^2$$

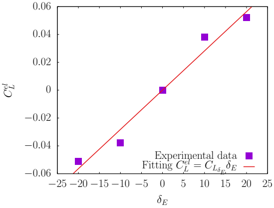
$$C_M = \underbrace{C_{M_0} + C_{M_\alpha}\alpha}_{C_M^{wing}} + \underbrace{C_{M_{\delta_E}}\delta_E}_{C_M^{el}}$$

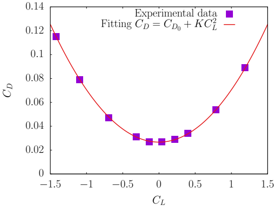
Aerodynamics coefficients from experimental data

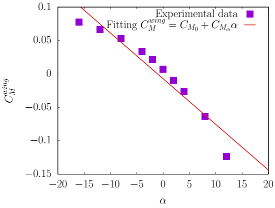
- The constants C_{L_0}, C_{L_α}, etc, can be obtained from experimental data using curve fitting

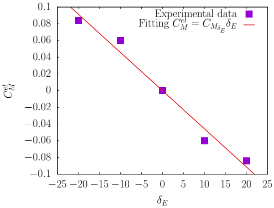
$$C_L = \underbrace{C_{L_0} + C_{L_\alpha}\alpha}_{C_L^{wing}} + \underbrace{C_{L_{\delta_E}}\delta_E}_{C_L^{el}} \qquad C_M = \underbrace{C_{M_0} + C_{M_\alpha}\alpha}_{C_M^{wing}} + \underbrace{C_{M_{\delta_E}}\delta_E}_{C_M^{el}} \qquad C_D = C_{D_0} + KC_L^2$$











Experimental data for aerodynamics coefficients

Experimental data in python format (file "aero_table.py")

```python
# python files with the aerodynamics coefficients
# at discrete values of the angle of attack alpha and
# elevator angle delta_el

# importing modules
import numpy as np
import math
# ------------------------------------
# Angle of attack alpha
alpha = np.array([-16,-12,-8,-4,-2,0,2,4,8,12])

CD_wing = np.array([
    0.115000000000000
  , 0.079000000000000
  , 0.047000000000000
  , 0.031000000000000
  , 0.027000000000000
  , 0.027000000000000
  , 0.029000000000000
  , 0.034000000000000
  , 0.054000000000000
  , 0.089000000000000
  ])

CL_wing = np.array([
   -1.421000000000000
  ,-1.092000000000000
  ,-0.695000000000000
  ,-0.312000000000000
  ,-0.132000000000000
  , 0.041000000000000
  , 0.218000000000000
  , 0.402000000000000
  , 0.786000000000000
  , 1.186000000000000
  ])

CM_wing = np.array([
    0.077500000000000
  , 0.066300000000000
  , 0.053000000000000
  , 0.033700000000000
  , 0.021700000000000
  , 0.007300000000000
  ,-0.009000000000000
  ,-0.026300000000000
  ,-0.063200000000000
  ,-0.123500000000000
  ])
# ------------------------------------
# Elevator angle delta_E
delta_el  = np.array([-20,-10,0,10,20])

CL_el = np.array([
   -0.051000000000000
  ,-0.038000000000000
  ,                 0
  , 0.038000000000000
  , 0.052000000000000
  ])

CM_el = np.array([
    0.084200000000000
  , 0.060100000000000
  ,-0.000100000000000
  ,-0.060100000000000
  ,-0.084300000000000
  ])
```

Trim: airplane equilibrium

- Time derivatives of velocities, angular velocity, and angular position equal to zero (equilibrium condition)
- Select a velocity V and a flight path angle γ
- Task: find angle of attach α and the value of the two commands T (thrust) and δ_E (elevator angle) that satisfy the equilibrium (trim) equations:

$$\frac{\mathrm{d}u_B}{\mathrm{d}t} = \frac{L}{m} sin\alpha - \frac{D}{m} cos\alpha - qw_B - \frac{W}{m} sin\theta + \frac{T}{m} = 0$$

$$\frac{\mathrm{d}w_B}{\mathrm{d}t} = -\frac{L}{m} cos\alpha - \frac{D}{m} sin\alpha + qu_B + \frac{W}{m} cos\theta = 0$$

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{M}{I_{yy}} = 0$$

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = q = 0$$

- From $M/I_{yy} = 0$ we have:

$$M = \frac{1}{2}\rho V^2 S\bar{c}C_M = 0 \quad \Rightarrow \quad C_M = C_{M_0} + C_{M_\alpha}\alpha + C_{M_{\delta_E}}\delta_E = 0$$

so

$$\delta_E = -\frac{C_{M_0} + C_{M_\alpha}\alpha}{C_{M_{\delta_E}}}$$

Trim: airplane equilibrium

- From:
$$\frac{\mathrm{d}w_B}{\mathrm{d}t} = -\frac{L}{m}cos\alpha - \frac{D}{m}sin\alpha + qu_B + \frac{W}{m}cos\theta = 0$$
since $q = 0$ and $\theta = \alpha + \gamma$, and simplifying m:
$$-Lcos\alpha - Dsin\alpha + Wcos(\alpha + \gamma) = 0$$
- From the previous definitions of Lift L and Drag D:
$$L = \frac{1}{2}\rho V^2 SC_L$$
$$D = \frac{1}{2}\rho V^2 SC_D$$
the models for the aerodynamics coefficients:
$$C_L = C_{L_0} + C_{L_\alpha}\alpha + C_{L_{\delta_E}}\delta_E$$
$$C_D = C_{D_0} + KC_L^2$$
and the relation between α and δ_E found in the previous slide by setting $M = 0$:
$$\delta_E = -\frac{C_{M_0} + C_{M_\alpha}\alpha}{C_{M_{\delta_E}}}$$
- It is found that the equation $-Lcos\alpha - Dsin\alpha + Wcos(\alpha + \gamma) = 0$ contains the single unknown α, so we can use it to find α

Trim: airplane equilibrium

- Once we have α, we compute δ_E from:

$$\delta_E = -\frac{C_{M_0} + C_{M_\alpha}\alpha}{C_{M_{\delta_E}}}$$

- and $\theta = \gamma + \alpha$
- Then the thrust T can be found from (taking into account that $q = 0$):

$$\frac{\mathrm{d}u_B}{\mathrm{d}t} = \frac{L}{m} sin\alpha - \frac{D}{m} cos\alpha - q w_B - \frac{W}{m} sin\theta + \frac{T}{m} = 0$$

- In summary:
  1. We selected a velocity V and a flight path angle γ.
  2. We computed the angle of attach α that ensures the equilibrium of forces, in the z_B direction, for the given velocity V and a flight path angle γ.
  3. Then, we computed the value of the command δ_E (elevator angle) required to impose the angle of attack α just computed.
  4. Finally, we computed the value of the command T (thrust) to assure the equilibrium of forces in the x_B direction.
- Note: Trim conditions do not depends on altitude. This is due to the simplified assumption of using a **constant air density**

Example of trim for code validation

Plane characteristics and environment (files "vehicle.py" and "env.py")

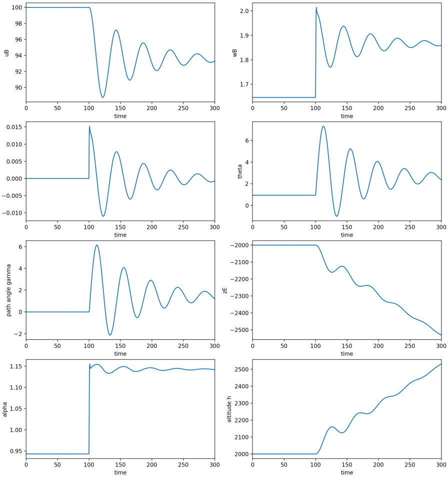
```
# Airplane Characteristics
# Wing surface m^2
Sref = 20.0
# airfoil chord m
cbar = 1.75
# Mass of the airplane Kg
acMass = 1300.0
# Moment of inertia kg/m^2
inertia_yy = 7000

# Environment
# Gravity acceleration m/s^2
gravity = 9.81
# Air density kg/m^3
air_density = 1.0065
```

- Consider a plane with the above characteristics and the aerodynamis described by simplified relation obtained fitting the experimental data provided in the file "aero_table.py"
- Trim for velocity $V = 100\ m/s$ and flight path angle $\gamma = 0.05$ radian
- The following solution is obtained:
  $\alpha = 0.0164$ rad
  $\delta_E = -0.0519$ rad
  $T = 3392.35$ N
  $q = 0$ rad/s
  $\theta = 0.0664$ rad
  $u_B = 99.986$ m/s
  $w_B = 1.641$ m/s

Response to commands

- We start from a trim condition with $V = 100\ m/s$ and flight path angle $\gamma = 0.0$ radian
  For this condition, the initial state is:
  $\alpha = 0.0164$ rad $\delta_E = -0.0520$ rad
  $T = 2755.17$ N $q = 0$ rad/s
  $\theta = 0.01646$ rad $u_B = 99.986$ m/s
  $w_B = 1.646$ m/s
- At $t = 100$ seconds, we change the elevator angle δ_E by 10% from -0.0520 to -0.0572.



*All angles are in degree in the graphs above

Part A. Develop python code with the following functionality

- Compute the coefficients for the simplified models of C_L, C_D, and C_M from a set of experimental data.
- Trim the airplane: For given values of the velocity V and flight path angle γ, compute the angle of attack α, the value of the commands T (thrust) and δ_E (elevator angle), and all the other state variables.
- Solve the 3 DoF equations of motion of the airplane. The simulation should start from a trim initial condition and then compute the response of the system to time-dependent commands such as a variation of the thrust T and elevator angle δ_E, or a combination of the two.

Part B1. Use the python code you developed to perform the following engineering design simulations:

- Trim the airplane for a range of values of the velocity $V_{min} < V < V_{max}$ and flight path angle $\gamma_{min} < \gamma < \gamma_{max}$ to find the value of the commands T (thrust) and δ_E (elevator angle) for several combinations of V and γ in the range above. Plot the T and δ_E vs V and γ.
  Pay attention at the min and max values of the ranges for V and γ. Limit the ranges such that physical constrains are not violated. For example, the thrust T should always be positive, α and δ_E should be in the ranges of the experimental data provided for the aerodynamics coefficients C_L, C_D, and C_M, etc.

Design project tasks

Part B2. Use your python code to analyse the climb from horizontal flight at an altitude $h_1 = 1000m$ to horizontal flight at another altitude $h_2 = 2000m$.

- First, consider the following 3 equilibrium conditions:
  - **Trim condition 1:** Consider a trim condition at constant altitude $h_1 = 1000m$ (flight path angle $\gamma = 0$) with a velocity $V = (100 + U)\ m/s$, where U is the day of birth (1-31) of the oldest member of your group.
    – In this conditions the commands are T_1 and δ_{E_1}.
  - **Trim condition 2:** Compute another trim condition with the same velocity and a flight path angle $\gamma = 2$ degrees.
    – In this conditions the commands are T_2 and δ_{E_2}.
  - **Trim condition 3:** Same as Trim condition 1, but at different altitude $h_2 = 2000m$ (in our model, altitude has no effect on trim)
    – In this conditions the commands are $T_3 = T_1$ and $\delta_{E_3} = \delta_{E_1}$.
- Then:
  - Simulate the system for 10 seconds starting from **Trim condition 1** (commands at T_1 and δ_{E_1})
  - Change commands to T_2 and δ_{E_2} and simulate the system for an additional interval of t_{climb} seconds.
  - Change commands to T_3 and δ_{E_3} and simulate the system for a time long enough that oscillations are dumped.
- Question: how long should the commands T_2 and δ_{E_2} be applied (in other words, find the appropriate t_{climb}), such that the altitude at the end of the simulation is approximately $h_2 = 2000m$?
- To this end, run steps 1,2,3 several times for different values of t_{climb} to identify the appropriate t_{climb}.

Part C. Develop a user interface that allows the user to do the following:

- Prescribe a desired flight trim condition (the velocity V and flight path angle γ) and see the resulting angle of attack α, the value of the commands T (thrust) and δ_E (elevator angle).
- Starting from the trim condition computed above, prescribe a step change of commands (T and δ_E) and a total solution time, to see the resulting time evolution. For example the software could output the plots of some variables vs time.

Code testing and user interface

- Your submitted code will be tested by trying to perform the two tasks in Part C described before.
- The user should be able to run the code easily by prescribing the desired inputs (velocity V and flight path angle γ for the trim and the subsequent step change in the commands.)
- A simple user interface could be a file that is imported in your python code. The user will insert the input in this file.

Example of user interface based on a simple file

```
# ---------------------------------------------
# Input for initial trim

# Trim velocity [m/s]
TrimVelocity = <filled by user> # e.g. 100 (m/s)

# Trim flight path angle [radian]
TrimGamma = <filled by user> # e.g. 0.05 (rad)
# ---------------------------------------------



# ---------------------------------------------
# Input for step change of commands
# Total simulation time (seconds)
TotalTimeSimulation =  <filled by user> # e.g. 200 (seconds)

# Increase elevator angle by
# a percentage PercentageChangeElevator
# at a certain time TimeChangeElevator
PercentageChangeElevator  <filled by user> # e.g. 10 (\%)
TimeChangeElevator = <filled by user> # e.g. 20 (seconds)

# Increase Thrust by
# a percentage PercentageChangeThrust
# at a certain time TimeChangeThrust
PercentageChangeThrust  <filled by user> # e.g. 10 (\%)
TimeChangeThrust = <filled by user> # e.g. 30 (seconds)
# ---------------------------------------------
```

- The clarity and simplicity of use is very important for a successful test.

Final remarks

- **Python files provided in Learn for the project:**
  - Files containing the airplane characteristics, its aerodynamics and the environment.
    - aero_table.py
    - env.py
    - vehicle.py
  - Files with python code (often to be completed by students) with useful functions for the development of the project code.
    - aero_analytical_build_ToBeCompleted.py
    - ...

Enjoy coding and flight mechanics