MULTIROTOR DRONE SIZING & TRAJECTORY OPTIMIZATION WITHIN MODELON IMPACT

October 26th, 2022
AGENDA

• Case study
  • Use case
  • Drone architecture

• Drone model
  • Model choices
  • Propeller model (as an example)
  • Sizing scenarios

• Optimization
  • Problem statement
  • Solving
  • Implementation and results

• Conclusions and Perspectives
CASE STUDY
CASE STUDY PAYLOAD LIFTING

• Use case - lifting small payloads on top of buildings
  • Maximum 25 kg
  • 150 climbs of 10m height with 5secs hovering

• What we want to solve, for a fixed drone architecture – simultaneously:
  • Optimize the drone trajectory to minimize energy consumption
  • Size the drone main parts
1. Four fixed pitch propellers
2. Four out-runner brushless motors
3. Four electronic speed controllers (ESC) mainly made from MOSFET inverters
4. One battery based on Li-Ion cells
5. One mechanical structure (frame) consisting of four arms and one central body
DRONE MODEL
DRONE MODEL CHOICES

• Model purpose
  • 1-D trajectory optimization
  • Component sizing

All equations are, at least, C2-continuous (optimization algorithm based on gradients)
Design variables have max, min, nominal attributes for bounds and scaling
Initial (not optimized) trajectory provided

A-causality: use of Modelica language
  • Sizing requires inverse simulation
  • Performance flight simulation requires direct simulation

Scaling laws and meta models for sizing
Efficiency-based modeling (low fidelity)
**DRONE PROPELLER MODEL**

- **Model purpose**
  - Physics (performance)
  - Scaling laws (sizing)

\[
M_{\text{prop}} = M_{\text{ref}} \left( \frac{D_{\text{prop}}}{D_{\text{ref}}} \right)^2
\]

\[
l_{\text{prop}} = M_{\text{prop}} \left( \frac{D_{\text{prop}}}{2} \right)^3 / 3
\]

\[
\beta = \frac{\text{pitch}}{D}
\]

\[
J = \frac{V}{(nD)}
\]

\[
B = \frac{K}{(\rho n^2 D^2)}
\]

\[
C_T = f(\beta, J, B)
\]

\[
C_p = f(\beta, J, B)
\]

\[
\text{Thrust} = C_T \rho_{\text{air}} n^2 D^4
\]

\[
\text{Power} = C_p \rho_{\text{air}} n^3 D^5
\]

Polynomial fitting on maps

Buckingham π theorem

©2022 Modelon. All rights reserved.
DRONE SIZING MODEL

- Sizing scenarios
  - Hover – J=0 (as V=0)
  - Takeoff – maximum power, increasing J
  - Climb – constant J

\[
\begin{align*}
\beta &= \text{pitch to diameter ratio} \\
J &= \frac{V}{(nD)} \\
B &= \frac{K}{(\rho n^2 D^2)} \quad \text{air compressibility indicator}
\end{align*}
\]
OPTIMIZATION
We want to maximize the number of flights; hence our objective is to minimize the energy consumption per flight.

Drone trajectory is a degree of freedom with constraints.
Drones trajectory is a degree of freedom.

We want to maximize the number of flights; hence our objective is to minimize the energy consumption per flight.

Optimica model including:
- Objective
- Trajectory optimization
- Constraints

Modelica Drone model with:
- Pre-sizing
- Mass estimation
- Behavior
optimization SizingAndTrajectoryOptim {
    objective = M_total(startTime),
    finalTime(free=true, min=1, max=10, start=5)
} // Minimize the total drone mass and relax the final simulation time within bounds.

import Modelica.Units.SI.DimensionlessRatio;

extends Drone{
    x(start = 0, fixed=true),
    xp(start = 0, fixed=true),
    a(start = 0, fixed=true),
    beta(free=true, min=0.3, max=0.6, start=0.4),
    T_nom_mot(free=true, min=0, max=1),
    K_mot(free=true, min=0),
    M_bat(free=true, min=0, max=100),
    P_esc(free=true, min=0),
    k_D(free=true, min=0.01, max=1, start=0.05),
    D_out_arm(free=true, min=0.001, max=1));
} // Inherit the Modelica drone model, fix initial conditions and relax design parameters within bounds.

Modelica.Blocks.Interfaces.RealInput Traj_in;

// Add input to the trajectory to optimize
...
OPTIMIZATION RESULTS

Drone ~45kg
Incl. ~27kg battery
+ 25kg payload
CONCLUSIONS & PERSPECTIVES
In comparison with a FAST-OAD optimization, relying on a drone FMU

- Solving initialization problem
  - No need to adapt the code for solving an IVP
- A-causality
  - Torque trajectory optimization fed with an initial Position-controlled trajectory, simulated from the same model
- Normalization for convergence, based on nominal attributes and bounds
- Derivatives at hand of the optimizer
- "Simple, fast and robust"
  - Optimica language is similar to Modelica language (as it is an extension from it)
  - 30sec optim vs 2min for the FAST-OAD solution
- Everything is never all black or white: FAST-OAD allows integration of models from different fidelities (e.g. CFD) and well suited for large scale optimizations
CONCLUSION & PERSPECTIVES

- Simultaneous drone 1-D trajectory optimization and sizing, in Modelon Impact
- Scaling-law based sizing and efficiency-based performance models
- Results obtained with FAST-OAD and Modelon Impact were similar – though the Optimica implementation was of low threshold for a Modelica engineer (the model requires preparation)

- 6-DOF
- Regulatory constraints
THANK YOU