VALIDATION AND EXTENSION OF AN MDO FRAMEWORK INCLUDING DYNAMIC AEROELASTIC ANALYSIS

Author: Lucas Bernácer Soriano
Supervisor: Rauno Cavallaro
Co-supervisor: Francesco Torrigiani
1. INTRODUCTION

THE AIRCRAFT DESIGN PROCESS

The classic vs modern approach

In the **classic approach**, the number of modeling details increases with time, being **reduced** the number of parameters which can be **modified**. [1]
1. INTRODUCTION

THE AIRCRAFT DESIGN PROCESS

The classic vs modern approach

Modern approach challenges

- Automatization
- Robustness

Modern aircraft design approaches try to increase the level of details and fidelity in the early phase of the design process. [2]

Reducing the design process time and cost
1. INTRODUCTION

THE PURPOSE OF THE THESIS

Developed tools

- **NOI** - Nastran Optimization Interface
- **FAEDO** - Framework for Aeroelastic Design and Optimization

Optimization framework

- Able to deal with flutter analysis in the conceptual phase.
- Automatized and streamlined the use of Nastran.
- Evaluation of different test cases, increasing the flexibility and robustness of the tools.

Optimization platform

**OpenMDAO**

Capable of supporting gradient-based optimization
Optimization

The aim of optimization is to achieve the "best" design relative to a set of prioritized criteria or constraints. [3]

Minimize $f(x, y)$ with respect to $x$ and $y$
Subject to
- Behavioral constraints on $y$
- Design constraints on $x$
- Equilibrium constraints

Integration in the aircraft design process
The aim of optimization is to achieve the "best" design relative to a set of prioritized criteria or constraints. [3]

Objective function \( (f) \): Denotes the function that points out the goodness of the design. Usually \( f \) indicates the weight, stress or displacement in the structure.

Design variables \( (x) \): The design variables can be a function or a vector that represents the design. They are usually related to a geometry feature or even to a choice of material.

State variables \( (y) \): Conform the responses of the problem for a given design condition. In structural optimization state variables are mainly related to stresses, strains, or forces.
The aim of optimization is to achieve the "best" design relative to a set of prioritized criteria or constraints. [3]

Minimize $f(x, y)$ with respect to $x$ and $y$

Subject to
- Behavioral constraints on $y$
- Design constraints on $x$
- Equilibrium constraints

Sensitivity analysis

- Provides the variation of the responses with respect to the design variables.
- Allows identifying how a certain change in a design variable may impact the product.
- Serves as an input to guide the optimizer in the proper direction.

$$\lambda_{ij} = \frac{\partial g_i}{\partial x_j} \bigg|_{x^0}$$
2. NOI

Nastran Optimization Interface, also known as NOI, is a self-developed Python tool that provides the capability of writing Nastran optimization, modal and flutter input files given the finite element or design model description.
2. NOI

NOI’s modules

BDFWriter.py

XBDF

Class developed to exploit the creation of different types of BDF files incorporates modal, static, flutter and optimization solution sequences.

OP2Reader.py

XOP2

Extension of the OP2 class in order to provide support for the modal, static and sensitivity analysis results stored in an OP2 file.

F06Reader.py

XF06

Module that expands the F06 results handling capabilities. Includes the possibility to represent V-g, V-f and root locus plots of flutter results.
2. NOI

**BDFOpt**

- Focus on the *optimization input file* creation.
- Possibility to define *size design variables*.
- Accounts for *modal and static analysis*.
- Requires a set of *user-defined dictionaries*.

**Definition of objective function, responses and design variables**

**CREATION OF NASTRAN INPUT FILES**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input BDF</td>
<td>Classes developed to exploit the creation of different types of BDF files incorporates modal, static, flutter and optimization solution sequences.</td>
</tr>
</tbody>
</table>
| BDFOpt | - set_new_properties (property_block_dict)  
- set_property_dv (property_block_dict, property_dict)  
- set_mass_dv (masses_dict)  
- set_responses (responses_dict)  
- set_case_control_deck (analysis_dict)  
- write_opt_bdf (property_block_dict, property_dict, masses_dict, responses_dict, user_test_cases, bdf_filename)  |
| Optimization BDF | |
| XBDF | |
| BDFModal | - write_modal_bdf (final_filename, n_modes, norm)  |
| Modal BDF | |
| BDFFlutter | - write_flutter_bdf (final_filename, n_modes, density_range, mach_range, velocity_range, velocity_ref, l_ref, rho_ref, sym_xz, sym_xy, k_range)  |
| Flutter BDF | |
2. NOI

CREATION OF NASTRAN INPUT FILES

BDFModal
- Able to create modal input files.
- Possibility to define the number of modes and normalization method of eigenvectors.

BDFFlutter
- Outputs flutter input files.
- Allows specifying the flutter analysis parameters such as speed range and reference magnitudes.

Input BDF

XBDF
Class developed to exploit the creation of different types of BDF files incorporates modal, static, flutter and optimization solution sequences.

BDFOpt

set_new_properties (property_block_dict)
set_property_dev (property_block_dict, property_dict)
set_mass_dev (masses_dict)
set_responses (responses_dict)
set_case_control_deck (analysis_dict)
write_opt_bdf (property_block_dict, property_dict, masses_dict, responses_dict, user_test_cases, bdf_filename)

Optimization BDF

BDFModal
write_modal_bdf (final_filename, n_modes, norm)

Modal BDF

BDFFlutter
write_flutter_bdf (final_filename, n_modes, density_range, mach_range, velocity_range, velocity_ref, l_ref, rho_ref, sym_xz, sym_xy, k_range)

Flutter BDF
2. NOI

**OP2Static**

- **Stores the stress results** when a static analysis has been performed.
- Accounts for the **global sensitivity matrix**.

**OP2Modal**

- Outputs the **eigenvalues and eigenvectors**.
- Stores the **sorted version of the sensitivity matrix**.
- Post-processing of the **structural matrices**.

---

**POST-PROCESSING OF RESULTS**

**OP2Static**

**OP2Modal**

**OP2Static**

**OP2Modal**

**XOP2**

Extension of the OP2 class in order to provide support for the modal, static and sensitivity analysis results stored in an OP2 file.

- `get_model_weight()`
- `get_model_eg()`
- `get_weight_derivatives(sens)`

---

**OP2Static**

**OP2Modal**

**XOP2**

- `stress_output()`
- `sensitivity_global()`

---

**OP2Static**

**OP2Modal**

**XOP2**

- `eigenvalues_output()`
- `sensitivity_custom (responses_dict)`
- `get_structural_shapes_dict()`
- `get_structural_shapes_derivatives_dict(sens)`
- `get_generalized_structural_matrices`
- `get_generalized_structural_matrices_derivatives(sens)`
2. NOI

POST-PROCESSING OF FLUTTER RESULTS

XF06

- Eases the task of post-processing the *Nastran* flutter analysis results.
- Capable of representing the root locus.
- Possibility to display the V-g and V-f plots.

Goland wing V-g V-f plots obtained with the XF06 class [4]
3. FAEDO

Framework for Aeroelastic Design and Optimization

What is FAEDO?

Framework for Aeroelastic Design and Optimization, also known as FAEDO, is a Python tool that provides the capability of carrying out an optimization process considering an aeroelastic constraint.

- Structural analysis
- Aero-structural mapping
- Aerodynamic analysis
- Stability analysis

...
Integration of NOI in order to solve for the modal and sensitivity analysis of the FE model and to post-process the structural matrices.
This section of FAEDO performs a mapping of the structural modal displacements, and sensitivities, to the aerodynamic grid of the geometry.

- Requires the aerodynamic and structural grids.
- The eigenvectors computed in the structural analysis block must be provided.
- Uses a Infinite Plate Spline method. [5]
The aerodynamic analysis has been performed by means of an unsteady panel method that formerly goes under the name of GAF analysis in the framework. [6]

The stability analysis computes the finite state approximation for the aerodynamic results and provides the flutter speed and its derivative with a simple root locus method. [7]
3. FAEDO

Overview of the framework

- Eases the task of evaluating aeroelastic responses in optimization problems.
- Capable of computing the flutter speed and sensitivity.
- Integration of NOI in the structural analysis section.
- Requires the structural and aerodynamic grid.
- Solves the aerodynamic analysis based on the GAF method.
- Possibility to be coupled with an optimization platform.

OpenMDAO [8]
4. CASES OF STUDY

Objectives

- Verifying NOI’s integration in FAEDO.
- Check the proper flow of information inside FAEDO.
- Assessment of FAEDO when computing the aeroelastic constraint and its sensitivity.
- Coupling of FAEDO and OpenMDAO.

DOE for the flutter speed derivatives
Test case focused on verifying that the flutter speed derivatives computed with respect the design variables are reliable and accurate enough.

Optimization of the Goland+ model
Optimization process taking the mass as objective function and including a constraint on the flutter speed. The selected design variables represent the set of lumped masses that compose the front and rear spars.

Optimization of the Goland beam model
Optimization of the classic Goland wing mass modifying its cross-sectional area and restricting the flutter speed within a given bounds.
4. CASES OF STUDY

FINITE ELEMENT MODELS

GOLAND+ MODEL [9]

- Variation of the classic Goland wing model.
- Wing modeled with CQUAD elements.
- Includes lumped masses to get a certain aeroelastic response.
- Used to assess the lumped masses as design variables.
- Flutter speed of 119.09 m/s.

<table>
<thead>
<tr>
<th>FE model identifier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000, 1001, 1020, 1021</td>
<td>14.338 kg</td>
</tr>
<tr>
<td>1002 ... 1019</td>
<td>28.677 kg</td>
</tr>
<tr>
<td>1100, 1101, 1120, 1121</td>
<td>28.780 kg</td>
</tr>
<tr>
<td>1102 ... 1119</td>
<td>57.561 kg</td>
</tr>
<tr>
<td>1200, 1201, 1220, 1221</td>
<td>38.964 kg</td>
</tr>
<tr>
<td>1202 ... 1219</td>
<td>77.928 kg</td>
</tr>
</tbody>
</table>

GOLAND+ lumped masses values
4. CASES OF STUDY

FINITE ELEMENT MODELS

GOLAND BEAM MODEL [4]

- Replicates the classic Goland wing behavior.
- Wing modeled with CBEAM elements.
- Allows defining the distance between the elastic and the inertial axis.
- Used to assess the element properties as design variables.
- Flutter speed of 154.8 m/s.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.01323 m²</td>
</tr>
<tr>
<td>I₁</td>
<td>0.00262 m⁴</td>
</tr>
<tr>
<td>I₂</td>
<td>0.0001396 m⁴</td>
</tr>
<tr>
<td>J</td>
<td>3.809 × 10⁻⁵ m⁴</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Young Modulus</td>
<td>71.02 GPa</td>
</tr>
<tr>
<td>Shear Modulus</td>
<td>25.9 GPa</td>
</tr>
<tr>
<td>Poisson ratio</td>
<td>0.35</td>
</tr>
<tr>
<td>Density</td>
<td>2700 kg/m³</td>
</tr>
</tbody>
</table>

Goland beam model properties
4. CASES OF STUDY

OBJECTIVES

Verifying that the flutter speed derivatives computed by the optimization framework are reliable and accurate enough.

Methodology

Comparison of the analytical flutter speed derivatives provided by FAEDO with the ones obtained with finite differences.

Set-up

Definition of a toy optimization problem with a set of the lumped masses of the Goland+ model as design variables.

<table>
<thead>
<tr>
<th>Design variable</th>
<th>Set of masses identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSM</td>
<td>1000 ... 1021</td>
</tr>
<tr>
<td>RSM</td>
<td>1200 ... 1221</td>
</tr>
</tbody>
</table>

DOE design variables definition
4. CASES OF STUDY

DESIGN OF EXPERIMENTS

Mapping of the flutter speed

Analytical derivatives provided by FAEDO
Comparison between the analytical and FD derivatives
4. CASES OF STUDY

VERIFICATIONS

Verifying that FAEDO coupled with OpenMDAO is able of properly carrying out an optimization with aeroelastic constraint.

Optimization of the Goland+ model
Optimization process taking the mass as objective function and including a constraint on the flutter speed. The selected design variables represent the set of lumped masses that compose the front and rear spars.

Objective function: Mass of the Goland+ model.
Design variables: Lumped masses that compose the front and rear spar.
Aeroelastic constraint: Flutter speed between 100 and 140 m/s.

Optimization of the Goland beam model
Optimization of the classic Goland wing mass modifying its cross-sectional area and restricting the flutter speed within a given bounds.

Objective function: Mass of the Goland beam model.
Design variables: Cross-sectional area of the beam.
Aeroelastic constraint: Flutter speed between 100 and 140 m/s.
OPTIMIZATION OF THE GOLAND+ MODEL

4. CASES OF STUDY

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Initial value</th>
<th>Final value</th>
<th>% change</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>3283.33 kg</td>
<td>1812.359 kg</td>
<td>-44%</td>
</tr>
<tr>
<td>FSM</td>
<td>28.677 kg</td>
<td>2.787 kg</td>
<td>-90%</td>
</tr>
<tr>
<td>RSM</td>
<td>77.928 kg</td>
<td>22.097 kg</td>
<td>-77%</td>
</tr>
<tr>
<td>$V_f$</td>
<td>114.508 m/s</td>
<td>140 m/s</td>
<td>+22%</td>
</tr>
</tbody>
</table>

Results

- Constraint violation

![Graphs showing optimization results](image-url)
4. CASES OF STUDY

OPTIMIZATION OF THE GOLAND BEAM MODEL

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Initial value</th>
<th>Final value</th>
<th>% change</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>217.75 kg</td>
<td>85.29 kg</td>
<td>-61%</td>
</tr>
<tr>
<td>A</td>
<td>0.0132 m²</td>
<td>0.0052 m²</td>
<td>-61%</td>
</tr>
<tr>
<td>Vf</td>
<td>155.44 m/s</td>
<td>100 m/s</td>
<td>-36%</td>
</tr>
</tbody>
</table>
5. CONCLUSIONS

NOI
- Tool able to carry out structural and sensitivity analysis.
- Possibility to write optimization, modal and flutter input files.
- Able to post-process Nastran results stored in a f06 or OP2 files.

FAEDO
- Framework capable of solving optimization problems with a flutter speed constraint.
- Possibility to link FAEDO with any optimization platform.

Development of new methods capable of dealing with automatization, and which provide a high degree of robustness.

Integration in the aircraft design process.


THANK YOU FOR YOUR ATTENTION

Author: Lucas Bernácer Soriano
Contact: lucasberso@hotmail.com / 100396861@alumnos.uc3m.es
Date: 23/07/2020