International Forum on Aeroelasticity and Structural Dynamics
IFASD 2024
17-21 June 2024, The Hague, The Netherlands

NONLINEAR AEROELASTIC MODELING WORKFLOW USING SIMCENTER NASTRAN

Emre Öngüt[1], Maria Francesca Illiano[2]

[1] Siemens Digital Industries Software
[2] Leonardo Helicopters

Keywords: Nonlinear aeroelastic analysis, highly flexible wing, Nastran flutter analysis

Abstract: An aeroelastic analysis workflow for modelling and simulation of very flexible wings using Simcenter Nastran is demonstrated. The workflow establishes file-based data exchange between Nastran nonlinear structural and the aeroelastic solvers for static and flutter analysis. For the static analysis, first, wing twist information is extracted from nonlinear structural solution and included in the Nastran aeroelastic analyses with DMI W2GJ cards. Subsequently, aerodynamic loads are applied to the nonlinear model as follower forces using Nastran FORCE2 cards. For the flutter analysis workflow, static results are linearized using pre-stressed normal modes and used in usual flutter analysis routines. The workflow is automated with the help of an NX Open script, minimizing the user intervention. The workflow is applied to PAZY wing model of Technion and validated against computational and experimental data for static aeroelastic and flutter cases.

1 INTRODUCTION

Civil aviation has a significant ecological footprint with notable contribution to the emission of greenhouse gases like carbon dioxide (CO2) and nitrogen oxide (NOx). New propulsion technologies show definite potential of reducing greenhouse gas emissions, but propulsion technologies are just one line of action. The aerodynamic design and the structural weight are other drivers for reducing CO2 and NOx emissions. Looking at the aerodynamic design, drag shall be reduced while maintaining the lift force required to fulfil a certain transport task (passenger capacity, range, maximum take-off weight, etc.). One of the ways to achieve this target is with high aspect ratio wings. These are known to have less induced drag due to smaller vortices at the wing tips. Several examples for future aircraft configurations (Sugar Volt Concept by Boeing and the Zero Emission Aircraft concept by Airbus, Figure 1) show a trend towards highly flexible high aspect ratio wings. In addition to the reduction of induced drag by high aspect ratio wings, natural laminar flow has been identified to further contribute to drag reduction. In order to enable natural laminar flow, the sweep angle for the leading edges of the wings will be reduced significantly in order to avoid crossflows. High aspect ratio wings will not only feature increased wing span but also decreased chord length and consequently decreased wing thickness. These effects are driven by aerodynamic design in pursue of drag reduction. Structural solutions are required to use such aerodynamic concepts. Therefore, the wings of future aircraft will have less wing sweep, larger wingspan, smaller wing chord length and smaller wing thickness. Consequently, the wings will feature significantly more flexibility, especially if lightweight composite materials are used.

Figure 1 Boeing Sugar Volt Concept: Strut-braced high aspect ratio wing (left); Airbus zero emission aircraft

The increased wing flexibility necessitates that the geometrical nonlinearities due to large deformations must be taken into consideration in the design process. However, available industrial aeroelastic tools, such as Nastran, employ linear methods, which are not suitable for understanding the behavior of geometrically nonlinear structures. The main factors that render Nastran aeroelastic analysis ineffective in capturing nonlinear aeroelastic behavior can be broken down as such:

- Excessive structural deformation is not possible to capture as the structural solution in Nastran SOL14x is based on linearity assumption.
- Deformation of the structure changes the aero loads and its distribution (especially due to wing twist). If the structural solver cannot solve the structural deformation accurately (previous point), resulting aero loads would also be wrong.
- Nastran SOL14x calculates the loads in vertical direction and does not align to the structure. However, the aerodynamic loads are aligned to the structure in real case (i.e. follower force) and they contribute to further deformation of the structure.

The work presented here summarizes a workflow to address the challenge of nonlinear aeroelastic analysis. As a first step the process is restricted to static aeroelastic analysis and analysis of flutter boundary. The implementation of the workflow is explained in Section 2 and results are discussed in Section 3.

2 IMPLEMENTATION

2.1 Nonlinear Static Aeroelastic Analysis Workflow

Previously, it is investigated to approach the nonlinear aeroelastic analysis using multi-body and FE solvers, where aeroelastic loads are imported into the respective solver and static/dynamic aeroelastic behavior is captured [1]. The process applied in the present study aims to couple nonlinear structural and standard aeroelastic solvers, which is investigated previously by others such as Zhao et. al. [2] for SUGAR truss-braced wing and Xie et. al. [3]. Underyling assumption of this workflow is that the main source of nonlinearities is the geometric nonlinearity of the highly flexible wing. The process can be illustrated in Figure 2 for the nonliner static aeroelastic analysis.

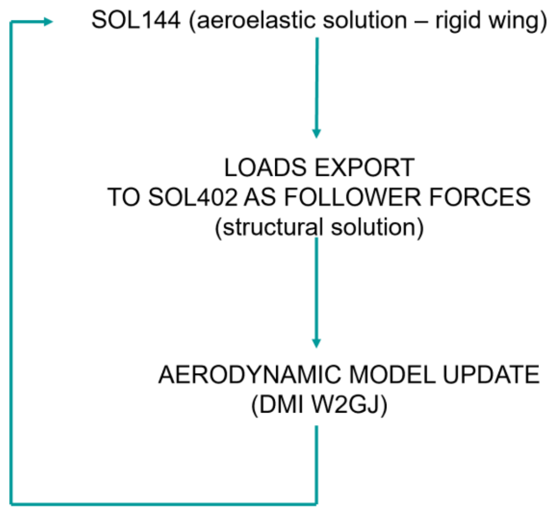

Figure 2 Nonlinear static aeroelastic analysis workflow

Two analysis models of the wing, aeroelastic and nonlinear structural, are needed for the process to be applied. Both models (aeroelastic and nonlinear structural) are prepared with Simcenter 3D pre-processor for Simcenter Nastran aeroelastic solver, SOL144, and nonlinear structural solver, SOL402.

The aeroelastic model is treated rigidly to determine the loads on the undeformed wing structure. These loads are then transferred to the nonlinear structural model as follower forces (Nastran FORCE2 cards) and nonlinear structural analysis is performed using the necessary options to capture the geometric nonlinearities. The resulting wing twist is extracted from the deformed wing structure and applied to the aeroelastic model as user defined downwash entries using Nastran DMI W2GJ entries. This step ensures that the wing can stay in an undeformed shape, and the rigid aerodynamic loads can be captured from standard aeroelastic analysis. The process is repeated until the converged solution is obtained for given operating conditions.

2.2 Nonlinear Flutter Analysis Workflow

A similar approach is followed for the nonlinear flutter analysis of the wing. The process starts with an initial estimation of the flutter speed, and the static nonlinear analysis process, which is explained previously, is executed for this operating condition. After the nonlinear static analysis process is completed, the pre-stressed modes are extracted from the results of nonlinear structural analysis and imported to SOL145. The reason for this step is that the loading on the structure effects the dynamic behavior due to geometric stiffening effect. The pre-stressed modes are easily imported in SOL145 with Simcenter 3D Pre/Post interface and no DMAP instructions was needed. The initial estimation of the flutter speed, which is used for the static analysis, is varied until the estimate and the flutter speed from SOL145 matches. An overview of the workflow is illustrated in Figure 3.

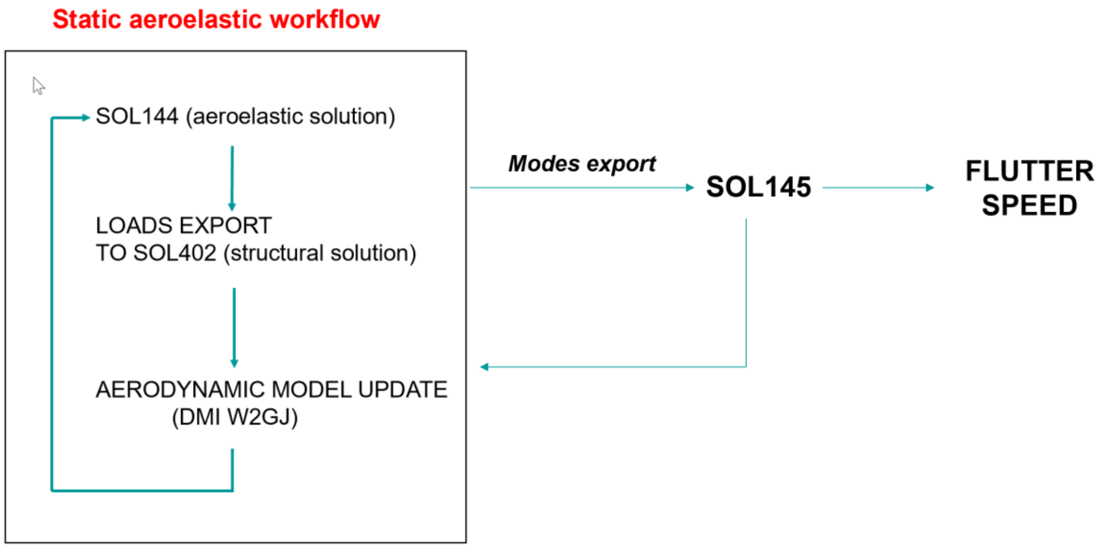

Figure 3 Nonlinear flutter analysis workflow

2.3 Workflow Automation

The static aeroelastic analysis workflow is automated using an NX Open script in Python. This script allowed an easy management of the workflow and minimized the user intervention. The code listing for the script used to run static nonlinear analysis is given in Appendix to facilitate further application of similar process by others.

3 RESULTS

For the validation of the workflow, Pazy wing model [4] of Israeli Institute of Technology, Technion is used. This model is developed with the intention of being used in aeroelastic research for highly flexible wings. Due to its flexible design, it exhibits very large elastic deformations in the order of 50% span at highest dynamic pressures and angles of attack in the wind tunnel. In this study, the so-called Pre-Pazy wing model is used as a reference. It is a preliminary flexible wind-tunnel experimental model for nonlinear aeroelastic tests in low-speed flow. It was developed in preparation for the Pazy wing experimental campaign. It was designed to achieve tip vertical displacements up to 60% semispan in wind-tunnel tests. This large range of deflections will enable to extensively validate numerical models of very flexible wings and to further investigate geometrically non-linear aeroelastic phenomena.

The Pre-Pazy wing is made of an Aluminum 7075 spar of 60 mm of width, and a 3D-printed PA2200 (Nylon) frame covered with a Oralight Polyester skin. The wing has a semispan of 550 mm and no sweep, dihedral, or twist. The cross section is shaped as a NACA 0018 airfoil with a uniform chord of 100 mm. An overview of the geometric properties can be seen in Table 1 and a photo of the wing can be seen in Figure 4.

Table 1 Properties of the Pre-Pazy Wing

Property	Value
Span	550 mm
Chord	100 mm
Area	0.055
Main spar	550x60x2.5 mm
Aspect ratio	5.5
Airfoil	NACA 0018
Mass	0.321 kg

Figure 4 Pre-pazy wing model test setup

3.1 Modal Analysis

Experimental and numerical modal analyses are performed by the original developers of Pazy wing and the finite element model was calibrated to match the experimental results. This was achieved using a wingtip mass of 10 gr. For the present study, the modal analysis is applied to the provided finite element model to verify the model import and boundary conditions.

The analysis is run with Nastran SOL103 with the provided boundary conditions. As reported in Table 2, the modal analysis yielded expected results, indicating the dynamic properties of the structural finite element model are in line with the experimental model. The first 4 mode shapes of the wing are also illustrated in Figure 5.

Table 2 Pre-Pazy wing without skin

	GVT	SOL 103
Mode	Frequency (Hz)	Frequency (Hz)
1	4.20	4.42
2	28.10	28.98
3	40.30	40.31
4	-	82.41

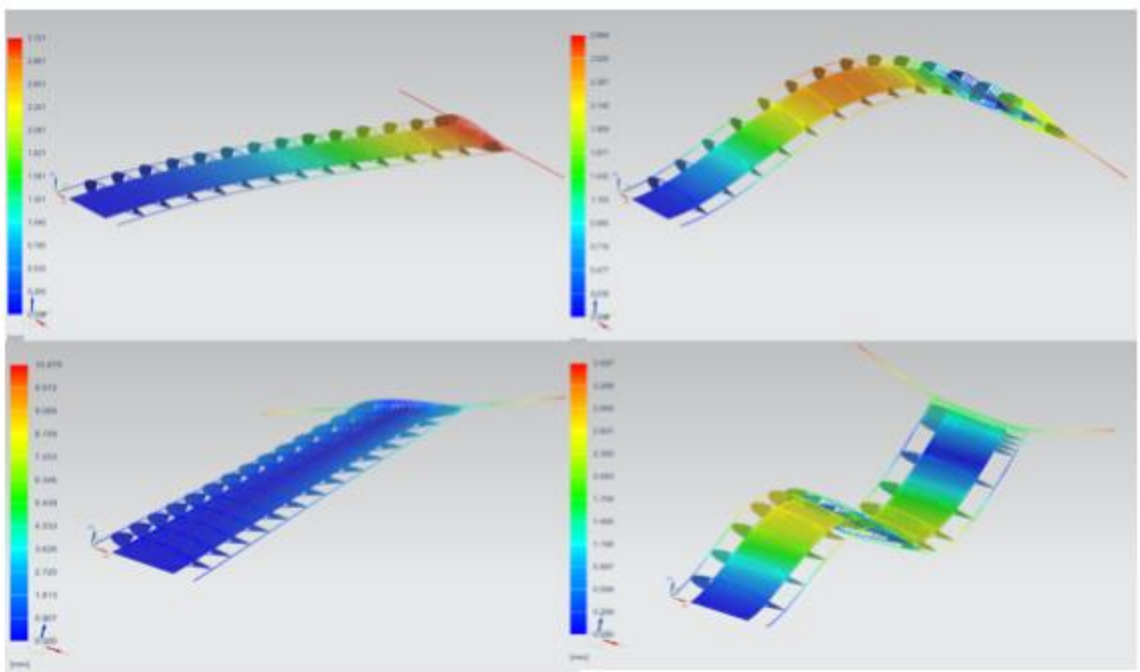

Figure 5 First 4 mode shapes - Wing without the skin model

3.2 Static structural analysis

Pre-Pazy wing is subjected to various static loads in the lab by the original designers for the calibration of the structural models. For the present study, the cases where 0-3 kg load is applied to half-cord location for bending and 0.08 m fore of wing leading edge for torsion, are compared to the experimental results. The results with the static linear and nonlinear structural solver are compared against experimental results to ensure that the model is imported, constrained, and loaded accurately. In Figure 6, maximum wing displacement for bending (left) and torsion (right) cases can be seen with respect to static loading. 5 and 8 % error is observed between the experimental results and nonlinear structural model for bending and torsion for 3 kg loading case, respectively.

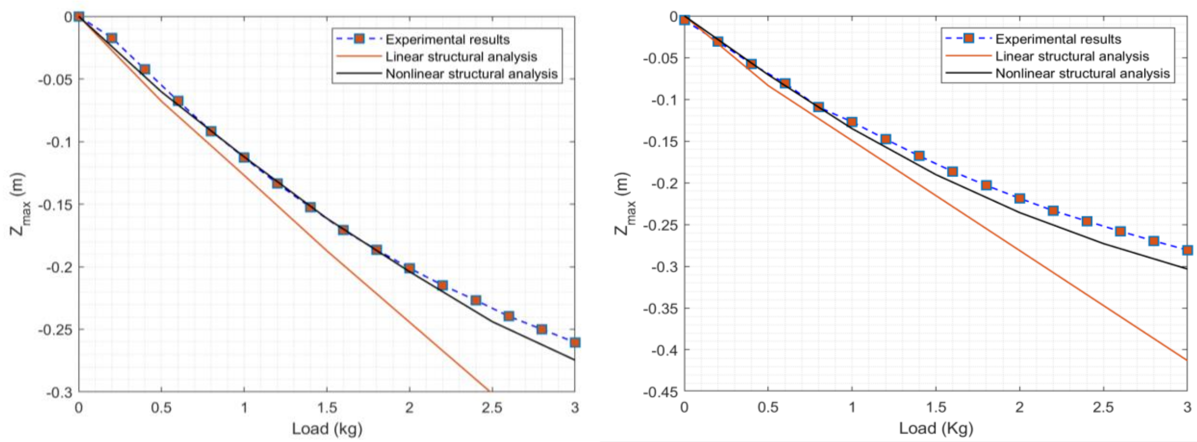

Figure 6 Maximum wing displacement in z direction for static loading for bending (left) and torsion (right)

3.3 Nonlinear static aeroelastic analysis

For the validation of the nonlinear static aeroelastic workflow, the version of the Pre-Pazy wing without the wing skin is used. The main reason is that the wing skin showed a highly nonlinear buckling behavior through the majority of upper surface, which lead to either very long, inaccurate, or non-converging nonlinear structural results. However, the experimental results, obviously, are produced with a model with the skin. Therefore, for the validation of nonlinear aeroelastic workflow, the results are compared to computational results by Riso and Cesnik [5]. In Figure 7, the comparison of vertical displacements values can be seen for 5 (left) and 7 (right) degrees angle of attack between two computational approaches with respect to the freestream velocity.

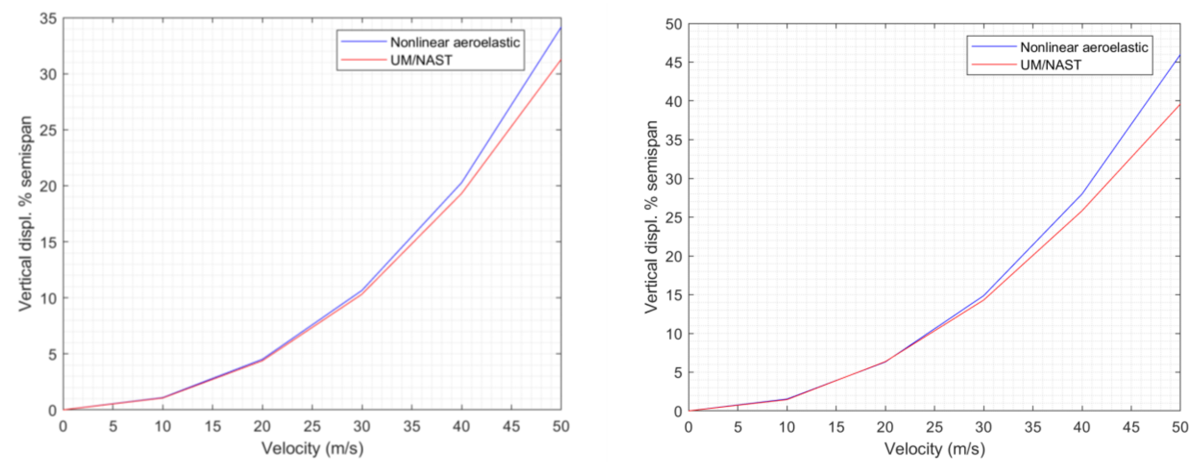

Figure 7 Vertical wing displacement (normalized to % semispan) with respect to air velocity for 5 degree (left) and 7 degree (right) angle of attack.

For this comparison, please note that the UM/NAST structural formulation by [5] is based on a geometrically exact, strain-based beam formulation while the present results are based on full nonlinear structural model. Therefore, it is expected to observe differences due to different treatment in structural model.

3.4 Nonlinear Flutter Analysis

As in the static nonlinear aeroelastic analysis, Pre-Pazy wing model without the skin is used for the validation of the nonlinear flutter analysis workflow. The workflow is applied for different angle of attack values in the range of 0-5 degrees and flutter onset velocities are determined. The results are compared to reference computational results from Hilger and Ritter (DLR) [6] and Goizueta et. al. (Imperial College) [7] and are presented in Figure 8. Results of the present study are shown with blue squares for each angle of attack value from 0 to 5 in the figure. For the angle of attack value of zero, the flutter onset velocity and normalized tip displacement values coincided with SHARPy tool of Imperial College, while further deviation is observed for higher angles of attack. On the other hand, the results for first flutter onset point lied consistently below the results reported by DLR.

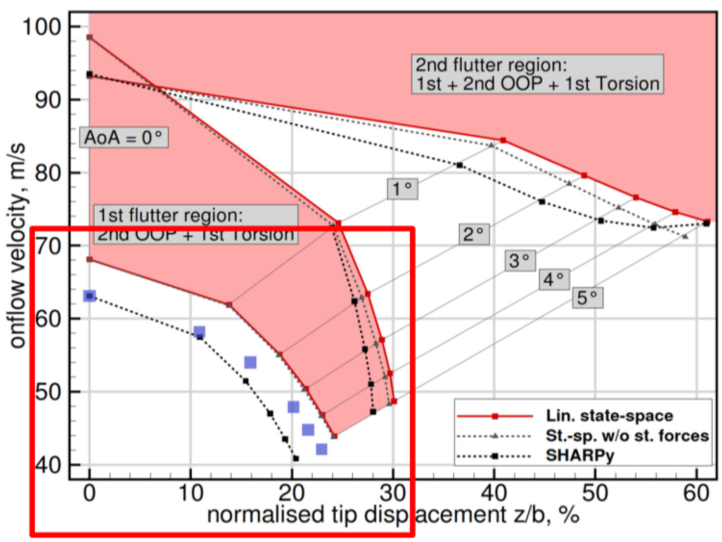

Figure 8 Flutter onflow velocity vs normalized tip displacement [6]. Present results are shown with blue squares.

Note that in the reference results, the first flutter offset, and second flutter onset velocities are reported as well, which are not investigated in the current study.

4 CONCLUSIONS

An aeroelastic analysis workflow for highly flexible wings is developed using Simcenter Nastran multistep nonlinear structural solver (SOL402) and Simcenter Nastran linear aeroelastic solver (SOL144/5). The accuracy of the workflow is shown for nonlinear static aeroelastic and nonlinear flutter cases using the Pre-Pazy wing results (experimental and computational) as reference.

For the nonlinear static aeroelastic analysis workflow, it is assumed that the main source of nonlinearity is the high flexibility of the structure. To capture the effect of large deformations, an iterative process between the nonlinear structural solver and linear aeroelastic solver is established.

After the first structural analysis, the structural wing twist information is extracted from the nonlinear solver results and applied to aeroelastic solver as user defined downwash entries using Nastran DMI W2GJ cards. The aeroelastic model is treated rigidly as the structural calculation is performed in the coupled nonlinear solver. Aero loads are then transferred to nonlinear solver and applied to the structure as follower forces using Nastran FORCE2 cards. The workflow is automated with an NX Open script, which handles the data transfer between solvers and minimizes the user intervention.

For the nonlinear flutter analysis workflow, a new workflow is established, where the structure is linearized under different operating conditions. This is achieved by generating pre-stressed normal modes and employing them in the usual linear flutter analysis. The workflow starts with the converged static nonlinear analysis. The pre-stressed normal modes are then calculated and imported to Nastran flutter analysis to determine the flutter onset velocity. This iteration is repeated until the static operating conditions match the flutter speed. This workflow, similar to the nonlinear static analysis, assumes that the main source of nonlinearities is the large deformation of the structure.

In both workflows it is observed that the skin of the Pre-Pazy wing model causes numerical instabilities; therefore, the model is used without the skin. The results are thus compared against the results of other researchers where the skin is also removed due to similar numerical challenges. Both for the nonlinear static and flutter analysis, a good correspondence to results reported in literature is observed. The proposed workflow is built using industrial off-the-shelf design tools, which makes it suitable for industrial applications for the design of aircraft with very flexible wings.

REFERENCES

[1] M. Castellani, J. E. Cooper and Y. Lemmens, "Nonlinear static aeroelasticity of high-aspect-ratio-wing aircraft by finite element and multibody methods," *Journal of Aircraft* , vol. 54, no. 2, pp. 548-560, 2017.

[2] W. Zhao, K. K. Rakesh, J. A. Schetz and J. M. Coggin, "Nonlinear aeroelastic analysis of SUGAR truss-braced wing (TBW) wind-tunnel model (WTM) under in-plane loads," in *In 56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2015.

[3] C. Xie, Y. Liu, C. Yang and J. E. Cooper, "Geometrically nonlinear aeroelastic stability analysis and wind tunnel test validation of a very flexible wing," *Shock and Vibration*, 2016.

[4] O. Avin, A. Drachinsky, Y. Ben-Shmuel and D. E. Raveh, "Design of an experimental benchmark of a highly flexible wing," in *In 60th Israel Annual Conference on Aerospace Sciences*, Haifa, Israel, 2020.

[5] C. Riso and C. E. Cesnik, "Correlations Between UM/NAST Nonlinear Aeroelastic Simulations and the Pre-Pazy Wing Experiment," in *In AIAA Scitech 2021 Forum*, 2021.

[6] J. Hilger and M. R. Ritter, "Nonlinear Aeroelastic Simulations and Stability Analysis of the Pazy Wing Aeroelastic Benchmark," *Aerospace*, vol. 8, p. 308, 2021.

[7] N. Goizueta, A. Wynn, R. Palacios, A. Drachinsky and D. E. Raveh, "Flutter prediction for a very flexible wing wind tunnel test," in *AIAA Scitech Forum*, Virtual Event, 2021.

APPENDIX

```
def main():

    workdir = "D: \\Models\\HARW\\PAZY WING\\PazyWing_Technion_1_10_2020\\" #UPDATE
    #NOTE: All file names should be lower case.
    #NOTE: Both sim files and journal file should be in the same folder.
    simNameAeroel  = "pazi_noskin_s"                        # UPDATE
    solNameAeroel  = "sol144"                               # UPDATE
    DMI_W2GJ_File  = workdir + "W2GJ TWIST.dat"             # UPDATE
    SOL144_f06File = workdir + simNameAeroel + "-" + solNameAeroel + ".f06"

    simNameNonlin  = "pazi_noskin_s"                        # UPDATE
    solNameNonlin  = "sol402aeroel"                         # UPDATE
    SOL402_twist   = workdir + "ResultsExtractionTwist.csv" # UPDATE
    SOL402_datFile = workdir + simNameNonlin + "-" + solNameNonlin + ".dat"

    twistNodesSpaced = "2781 2801 2959 3136 3152 3308 3330 3349 3365 3381 3539 3712 3728 3884"
    twistNodes = twistNodesSpaced.split()

    iter = 0
    maxTwistOld = 0.0
    percentError = 100.0
    errorHist = []
    while percentError > 1:

        # Initialize DMI W2GJ file with zeros
        if iter == 0:
            twistZero = [0.0] * 14                          # Update for n twist points
            twistUpdate(iter, twistZero, DMI_W2GJ_File)

        # Run SOL144: Export input file and then solve it
        exportInpFileAndSolve(simNameAeroel, solNameAeroel)

        # Extract loads from SOL145 output file (f06)
        # This can be replaced with NX Open routines to extract data from Simcenter directly
        ID, loads = get_loads(SOL144_f06File)

        # Export input file first from SOL402
        if iter == 0:
            exportInpFile(simNameNonlin, solNameNonlin)

        # And update the FORCE2 cards in the input file
        update_loads(iter, ID, loads, SOL402_datFile)

        # Run SOL402 with the existing input file (Which has the updated follower forces)
        solveExistingInpFile(simNameNonlin, solNameNonlin)

        # Here we extract the twist angle and update DMI WF2G matrix
        twistExtractCsv(simNameNonlin, solNameNonlin, SOL402_twist, twistNodes)
        twist = twistReadCsv(SOL402_twist)
        twistUpdate(iter, twist, DMI_W2GJ_File)

        # Update error
        maxTwistNew = max(twist)
        percentError = abs(maxTwistNew - maxTwistOld) / abs(maxTwistNew) * 100.0
        errorHist.append(str(percentError))
        #displayMessage("Percent error: " + str(percentError) )
        maxTwistOld = maxTwistNew

        iter = iter + 1

    displayMessage("Process completed in " + str(iter) + " iterations.")
    displayMessage("Error history: " + ' '.join(errorHist))
```

COPYRIGHT STATEMENT