Prediction of Gust Aeroelastic performance of HALE using Graph Neural Networks

Michele Colombo, Joseph Morlier, Michaël Bauerheim

Prediction of Gust Aeroelastic performance of HALE using Graph Neural Networks

Michele Colombo*
AIRBUS OPERATIONS SAS, 316 Route de Bayonne, 31060 Toulouse, France

Michael Bauerheim†
ISAE-SUPAERO, 10 Avenue Edouard Belin, 31055 Toulouse

Joseph Morlier‡
ICA, Université de Toulouse, ISAE-SUPAERO, MINES ALBI, UPS, INSA, CNRS 3 Rue Caroline Aigle, 31400 Toulouse, France.

Graph Neural Networks have been applied to learn the flight and structural dynamics of an High Altitude Long Endurance aircraft in discrete gust. The graph network methodology allows building a model for structural displacements, loads and aircraft flight dynamics leveraging on the inductive bias given by the physical connections. The results show promising capabilities in model approximation and potential for symbolic identification of aerodynamics and structural forces.

I. Nomenclature

G	=	Graph
u	=	global attribute of a graph
V	=	node or vertex attribute of a graph
E	=	Edge attribute of a of a graph
ρ	=	aggregation function in a graph
ϕ	=	update function in a graph
$\overline{\mathbf{e}}, \overline{\mathbf{v}}$	=	aggregated edges or vertex
$E^{'}, V^{'}, u^{'}, G^{'}$	=	updated attributes or entire Graph
$\mathrm{d}t$	=	time step
X	=	generic term for displacement
K	=	generic term for stiffness
M	=	generic term for mass
U_z	=	Gust vertical speed
u_{de}	=	Design gust speed
S	=	Design gust length

II. Introduction

Artificial intelligence (AI) in the last years has seen a rapid development in many different fields and it has been applied with success to tackle problems in many disciplines. Hybrid methodologies to introduce physical knowledge in AI are more and more studied, but its usage in modelling physical dynamical systems is still marginal. One of the main culprit in research in AI applied to dynamical systems is the capability to introduce inductive biases that drive AI to learn a system coherently with its physical laws. Graph Networks (GNs) offer a framework that is aimed at representing and identifying dynamical physical systems composed by entities in relation with each other such as those naturally emerging when studying vehicles such as an aircraft and its subsystems.

*Phd Candidate and Engineer in Loads and Aeroelastics Department.
†Professor, Department of Aerodynamics, Energetics, Propulsion
‡Professor, AIAA Member, Department of Mechanics, Structures and Materials

In this work GNs are used to learn the fundamental dynamics of a simple aeroelastic model aircraft subject to longitudinal discrete gusts of different amplitude and strength. Key structural characteristics, structural displacements and their derivatives and aircraft dynamics parameters are embedded as attributes of a GN and their time evolution is learned. A simple example on masses-spring chains is also presented to compare with existing literature on GNs applied to dynamics of mechanical systems.

GNs offer a wide flexibility in terms of how the graph that represents the dynamical system is built. GNs also allow the AI practitioner to choose different approximation functions for its internal update functions. In particular Fully Connected Neural Networks (FCNNs) are chosen in first instance due to their capability to be universal approximators [1] that can be easily differentiated, resulting in efficient training with large datasets. However, this work discusses also the limits of FCNNs and the benefits in replacing them with other type of functions in a second step leveraging on engineering knowledge. Other techniques of identification with particular regard to AI driven methods are reviewed and compared.

III. Related Work

Identification of dynamical systems can be summarised to two macro categories. The first is when the governing equations of a physical system are considered to be adequately known and the task summarises in discovering the parameters of these equations from the available data. Within this task we may also consider Reduced Order Models (ROMs) where a system is identified with a purposely reduced number of parameters or governing equations. In particular LTI (Linear Time Invariant Systems) and their derivations are often the fundamental bricks to characterise aeroelastic systems in aeronautics. A wide literature exist on the subject thanks to decades of scientific development in control theory and its applications.

The second macro category, in which this work is positioned, is when governing equations of a system are considered to be partially or totally unknown. This macro category encompasses a variety of data-driven methods whose primary tasks are either to build a black-box or grey-box model of a dynamical system able to accurately predict the behaviour of a system given a different set of inputs than those used to build the model and/or to identify the underlying governing equations, with a particular regard to non-linear systems. [2] approaches the identification of governing equations. The core of the Sparse Identification of Nonlinear Dynamics (SINDy) method is identifying an operator built upon a library of functions under the hypothesis of sparsity. This method is successful in identifying non linear systems such as the Lorentz attractor, the van der pol oscillator and flow fields characterised by periodicity, and the longitudinal dynamics of a fighter aircraft [3] with the constraint of choosing a suitable library of functions and features.

Graph Networks has been applied in [4], [5] to model the dynamics of physical systems, in particular mass spring chains subject to gravity or multibody systems. In these works physical systems are represented as graphs allowing Deep Learning algorithms to work on restricted sets of features to discover the mechanisms of interaction between interconnected bodies. This methodology has been recently developed by [6] combining Graph Networks, Deep learning on Fully Connected Neural Networks and Symbolic Identification to extract the laws of orbital mechanics and re-identifying some parameters such as masses. [7] uses instead GNs as a form of Reduced Order Modelling to encode the information of a flow field issued by Computational Fluid Dynamics (CFD) simulations into a latent space. Dynamics is modelled with a temporal attention network and GNs are also used to decode back its prediction to the initial space.

To summarise these works, if the purpose of the system identification is predicting the dynamics of a differently parameterised system, latent-space formulations offer flexibility and ease of implementation. However, when the ultimate task is being able to identify a mathematical relationship between the graph features or their evolution in time attention has to be paid in respecting used dimensions to retrieve the interaction among physical features.

Data-driven methods use statistical validation on purposely kept-aside data sets using various error metrics to assess how an identified system performance under an unseen set of parameters. This method is commonly used to validate models built upon FCNNs or other forms of neural networks made by large numbers of parameters. While being mathematically being proven to be universal approximators [1], most NNs based on commonly used activation functions struggle in approximating simple mathematical operations such as multiplication or division and require many terms. Research is currently active in developing units able to approximate better identity extrapolation, multiplication, division [8], [9].

Traditional knowledge in aerospace engineering is expressed via mathematical formulas synthesizing interactions between structural characteristics, the surrounding environment, control inputs etc. This approach has 80 years track record in leading to successful designs and capability to generalize to new designs. An extended literature exist on

building Reduced Order Models on aeroelastic aircrafts. In particular on High Altitude Long Endurance (HALE) [10] focused on some data-driven methods to build precise and quick models to be used for Model Predictive Control (MPCs) applications. It is not in the ambition of this work to improve on these formulations. It is however worth to point out some theoretical points that set this work apart and where there are similarities.

Data-driven ROMs used to build models fed to MPC tasks use simulation data that is not necessarily available in flight to simplify part of the mathematical equations normally solved to simulate the dynamics of the full system, for example the aerodynamic calculations by Unsteady Vortex Lattice Method (UVLM). These ROMs are built with attention to the underlying hypothesis necessary to tackle a specific shortcoming of previously used methods and integrated within the equations to describe the flight mechanics of an aircraft body motion.

This work instead we focused on learning the dynamic of a aeroelastic system subject to gust as a whole exclusively from measures selected for being commonly available on a standard flight test aircraft, using minimum amount of extra information on structural properties.

The ambition of this work is laying some foundations in bridging emerging AI methods to traditional engineering in the field of identification of dynamics of aeronautical systems.

IV. Graph Networks

Neural networks operating on graphs have been developed and explored extensively for more than a decade under the name of "graph neural networks" . In section IV.A we present the basic algorithmic block adopted for this study and how they can be composed.

A. GN Block

The main unit of computation in the GN framework is the GN block, a "graph-to-graph" module which takes a graph as input, performs computations over the structure, and returns a graph as output. A graph is defined as a structure having some nodes connected by some edges. These edges may have a direction, so we distinguish between a sender node and a receiver node. The graph may also have some global properties. Nodes, edges, and global properties are represented by attributes. In this work attributes are numerical vectors.

1. Graph Definition

A graph is defined as a 3-tuple $G = (\mathbf{u}, V, E)$. The u are global attributes of the entire system. For a generic mechanical system, u may contain all data representing the system behaviour as a whole. As as an example aircraft states such as heave, pitch angle, incidence angle etc as well as atmospheric characteristics such as a gust speed.

The $V = \{v_i\}_{i=1:N_v}$ is the set of nodes attributes. For a multi-body mechanical system, it is trivial to attribute to a node each single body and its states. But it is possible to adopt this representation also for discrete continuous systems: for a loaded beam we could consider the position ,the velocities, the stresses and other attributes at some measuring station alongside it.

Finally the $E = \{e_k, r_k, s_k\}_{k=1:N}$ is the set of edges. This contains the eventual attributes for its edge e_k and the connectivity information given by the set of sender nodes s_k and receiver nodes r_k. In this set we may represent all information necessary to explain how nodes interact with each other. Again, for a multi-body or continuous mechanical system the geometrical connections between bodies would naturally translate in this set. The content of the edge attributes would be all additive information to describe what relates node to each other : for example the torsional and rotational stiffness of the beam portion connecting two nodes. Edges attributes may also contain information on the connectivity status (example: if a joint is blocked or not).

2. Graph Network update

A Graph network is built by a chain of functions transforming the original Graph to another.

$$G = (\mathbf{u}, V, E) \rightarrow G^{'} \tag{1}$$

In the first step for all edges the edge , neighbouring nodes and global attributes are collected and updated via a function, called edge function to new edge attributes. The dimensions can change as well.

$$\mathbf{E}_k^{'} \leftarrow \phi^e \left(\mathbf{e}_k, \mathbf{v}_{rk}, \mathbf{v}_{sk}, \mathbf{u}\right), k \in \{1...N^e\} \tag{2}$$

Subsequently for each node, edges connected to it are collected and their attributes are pooled by a aggregation function. This function has to be variadic, as different nodes may have a different number of neighbours. It has also to be permutation invariant.

$$\bar{\mathbf{e}}_i^{'} \leftarrow \rho^{e\rightarrow v}\left(\mathbf{E}_i^{'}\right), i \in \{1...N^n\} \tag{3}$$

In the following step for all nodes the node and global attributes are collected together with the edge aggregation and updated via a function, called node function to new node attributes. Again, node attribute dimensions can change in this step.

$$\mathbf{n}_i^{'} \leftarrow \phi^n\left(\bar{\mathbf{e}}_i^{'}, \mathbf{v}_i, \mathbf{u}\right) i \in \{1...N^n\} \tag{4}$$

Subsequently aggregation is computed over all nodes and edges to feed the global. The aggregation functions used for this step do need also to be variadic and permutation invariant to allow the GN to train over graphs having different architectures.

$$\bar{\mathbf{e}}_i^{'} \leftarrow \rho^{e\rightarrow u}\left(\mathbf{E}_i^{'}\right), \bar{\mathbf{v}}_i^{'} \leftarrow \rho^{v\rightarrow u}\left(\mathbf{V}_i^{'}\right) \tag{5}$$

The last step consists in updating the global attributes using aggregated edge and node information via a function, called global function.

$$\mathbf{u}_i^{'} \leftarrow \phi^u\left(\bar{\mathbf{e}}_i^{'}, \bar{\mathbf{v}}_i^{'}, \mathbf{u}\right) \tag{6}$$

B. GN blocks architectures

This work uses a modified GN block in which the aggregation function is replaced by a linear function on collected inputs. This allows for more flexibility in feeding node and edges attributes to global function of the network at the cost of not being able to train the GN on different graphs, which is the case for our problem. Actually, aggregation function could be adapted without the constraints of being variadic and permutation invariant also if different graph structures would be considered to represent the chosen system, provided they share a similar structure.

Fig 1 exemplifies how this block will be used to learn the mapping from edge attributes E to a transformed set $E^{'}$ at time t and node and global attributes from time t to time $t+1$.

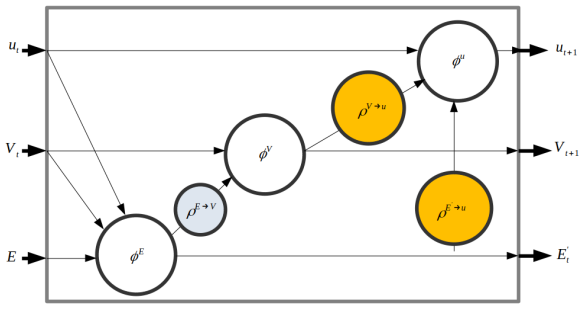

Fig. 1 Modified GN block

[4] uses a Encode-Process-Decode architecture in which a GN block encodes all input attributes to a latent space having arbitrary dimensions. The Process block is a GN block with a recurrent structure with a fixed number of message passing. The Decode block decodes from latent space to output space. This architecture is conceived to generalize the same learning problem over graphs having a different number of edges or nodes. The limits of aggregation functions, currently limited to sums, max, mean and other reducers are in approximation superseded by the latent space extra dimensions. Part of this architecture has been considered to build a GN network architecture with a encoder and decoder for the global attributes specific for this work displayed in Fig 2.

In section V the dynamics of a flexible aircraft subject to gust is introduced and these choices in GN architecture explained in detail.

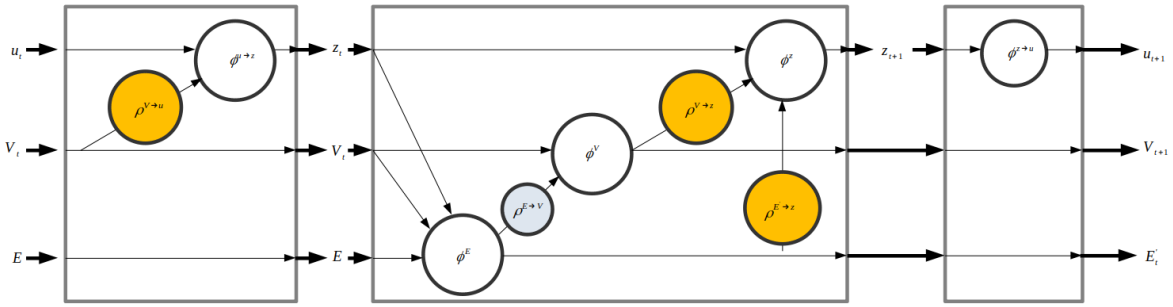

Fig. 2 Custom Encode Process Decode GN network

V. Configuration and Numerical Setup

A. HALE aircraft and Spring System

In this work we studied two dynamical systems: a mass-spring model ass a reference and an aeroelastic aircraft.

1. Spring System

The mechanical system tested in [4] has been considered to test the methodology. This system is composed by a number of masses connected in a chain by springs and subject to gravity. The initial and final mass of the chain are fixed. Fig **??** shows an example of these systems for 4 masses.

The learning problem consists in learning node velocities at $t+1$ given $G = (\mathbf{u}, V, E)$. The training dataset is composed by 256 graphs having a number of masses ranging from 4 to 8 masses; masses are all equal to 1 Kg. Spring stiffness is constant and equal to $50N/m$, spring rest length is a random number between 0.2 and 1.0 m. Masses at t=0 are disposed in a line perpendicular to the gravity vector equally spaced by the spring rest lengths. The validation dataset is composed by 100 graphs having 9 masses. Spring stiffness is unchanged and spring rest lengths are randomized in the same manner.

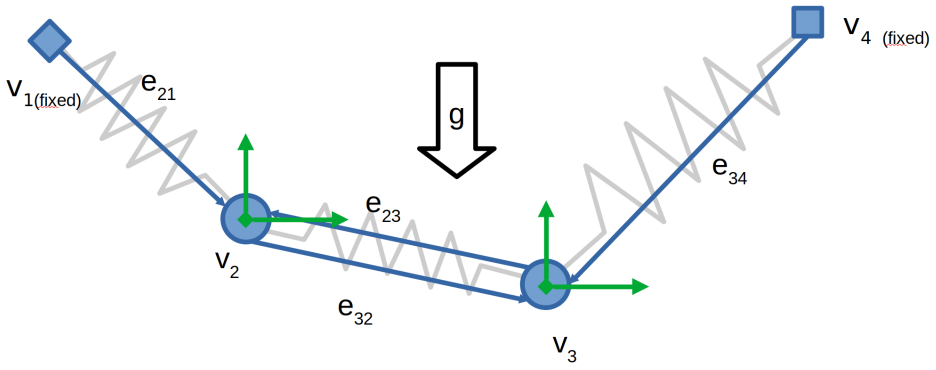

Fig. 3 Example of a mass spring chain system

2. HALE aircraft

The aeroelastic aircraft case uses simulations of discrete gust dynamics of a High Altitude Long Endurance (HALE) concept aircraft. This aircraft has a high wing aspect ratio and a T-tailed mounted Tail plane. The overall aircraft weighs 28.25 kg with a easy to customised payload carried at the wing intersection with the fuselage, that can be used to cause

a relevant deflection of the wing in his steady 1-g static flight. The choice for this Aircraft model lies in the fact that a medium fidelity model is publicly available within the SHARPy (Simulation of High Aspect Ratio aeroplanes and wind turbines in Python) tool. SHARPy [11] is a nonlinear aeroelastic simulation toolbox and available under open-source licence . At its core, SHARPy couples a nonlinear beam solver with a full-order nonlinear Unsteady Vortex Lattice Method (UVLM). Fig 4, 5 summarise geometrical and structural characteristics

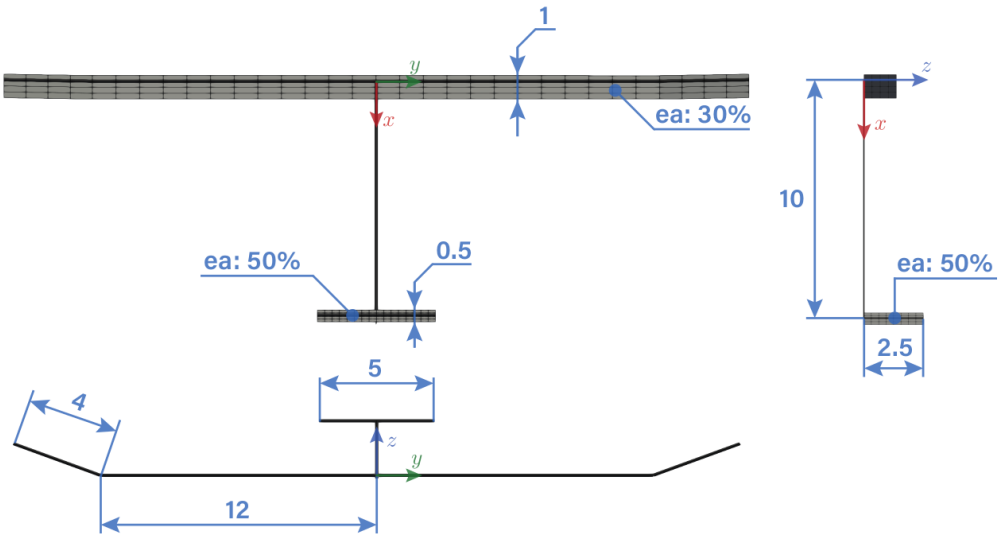

Fig. 4 HALE Aircraft Geometry

Component	EA [N m^{-1}]	GJ [N m^2]	EI_y [N m^2]	EI_z [N m^2]	$\bar{m}$ [kg m^{-1}]	$\bar{J}$ [kg m]
			Flexible			
Wing	1.5×10^7	1.5×10^4	3.0×10^4	6.0×10^5	0.75	0.075
Fuselage	1.5×10^8	1.5×10^5	3.0×10^5	6.0×10^6	0.2	0.08
Fin	1.5×10^8	1.5×10^5	3.0×10^5	6.0×10^6	0.3	0.08
Tail	1.5×10^8	1.5×10^5	3.0×10^5	6.0×10^6	0.3	0.08

Fig. 5 HALE Aircraft Properties

The discrete gust is modelled with the one minus cosine formula described by CS25 specifications for aircraft design [12] and reported below.

$$U_z = \frac{u_{de}}{2}\left[1 - \cos\left(\frac{2\pi x}{S}\right)\right] \tag{7}$$

A total of 28 simulations with gust intensity u_{de} ranging from 0.02 to 0.08 m/s and gust length S ranging from 0.25m to 1.75 m has been computed and used for training and validation; These values are just meant to excite the system with a moderate gust intensity. 4 simulations are kept as validation dataset and 24 for training. A payload of 10 Kg has been chosen for all simulations. The dataset for nodal displacements and loads has been restricted to the wing, as the aeroelastic behaviour is displayed essentially by this component. A light gust has been chosen to avoid for time being non linear effects due to strong deformations on this very flexible wing. Investigating more complex and realistic cases is envisaged for the next step.

B. Graph Network Setup

The GN block architecture has been chosen to guarantee a link of coherence among all used measurements and to guarantee a physical interpretability of the GN block functions.

The spring system is represented as a graph with spring stiffness and rest length as edge attributes , position , velocity, mass and a boolean equal to 1 to if the mass is fixed as node attributes and gravity acceleration as the only global attribute. A new, simpler architecture based on modified GN block is assessed against the original Encode-Process-Decode architecture. Neural Network dimensions are chosen to keep an equivalent number of parameters between the two compared architectures. In particular reference EPD is using 16 neurons , 2 layers FCNN with Rectified Linear activation functions (RELU) while in our simplified GN block we adopt 40 neuron, 2 layers RELU units.

SHARPy simulations output distributed mass and stiffness data for each beam element while structural displacements and their derivatives are given for tip and central node for each beam element. Lumped masses are given at the nodes. The aeroelastic case is represented with two different graphs as input/output couples for training. In our simplified GN block we adopt 70 neuron per layer, 2 layers RELU units int his case.

Structural node positions and velocities in aircraft reference frame and lumped nodal masses at tip nodes for each beam element are encoded as node attributes for both graphs. Global attributes get CG positions and velocities, including angular velocities compared to a fixed ground reference frame (for). Vertical Wind velocity from the discrete gust is a relevant attribute only for the input graph. Edge attributes for input graph are beam structural properties. Edge attributes for output graph are the computed beam loads.

$$G_{in} = (\mathbf{u}_{in}, V_{in}, E_{in}) : \begin{cases} X_{for,t}, \dot{X_{for,t}}, U_{z,t} \rightarrow \mathbf{u}_{in} \\ X_{node,t}, \dot{X_{node,t}}, M_{node} \rightarrow V_{in} \\ M_{beam}, K_{beam}, \rightarrow E_{in} \end{cases} \tag{8}$$

$$G_{out} = (\mathbf{u}_{out}, V_{out}, E_{out}) : \begin{cases} X_{for,t+1}, \dot{X_{for,t+1}} \rightarrow \mathbf{u}_{out} \\ X_{node,t+1}, \dot{X_{node,t+1}} \rightarrow V_{out} \\ L_{beam,t} \rightarrow E_{out} \end{cases} \tag{9}$$

In particular the Encode step of the custom Encode Process Decode architecture has the ambition to encode global attributes (flight mechanics parameters such as pitch angle, vertical wind speed etc) to a latent space information drawn from node attributes, namely position or velocities from points on the structure. In a dual view, latent variables may represent projections of the entire geometry merged with flight mechanic information at time t.

The edge function has the task to learn the beam Loads at time t from neighbouring node attributes, latent space global attributes and initial information stored in edges (stiffness and mass information). Beam Loads at time t are the interaction forces responsible for nodal attribute update in time. Updated node attribute and beam loads contribute to global attribute update in time The decoder step decode the global latest space attributes at time $t + 1$ to the flight mechanics variable space u.

VI. Results

A. Mass and Spring System

The performance is assessed with the Mean Squared Error (MSE) along a 50 time samples roll-out from an initial condition. Derivatives of nodal positions are computed through the learned GN and integrated using a single step of Forward Euler integration scheme to compute the next step node positions. A 2.7x improvement is observed in chosen error metrics.

The observed improvement is due to the better performance obtained in approximating derivatives. As a way to improve further performance in roll-out we included a regularization term in the loss based on short term prediction along a 4 time samples roll-out in a similar manner to [13]. The purpose of this term is to help minimizing drift errors. We observed a further improvement of the results (up to 6x) but we point out that this methodology is relevant only if the purpose is to use the learned model as a surrogate model to predict the dynamics of a system. If the task is identifying the interactions between the different elements of the graph it doesn't necessarily add precision and what matters most is the capability to predict the next step.

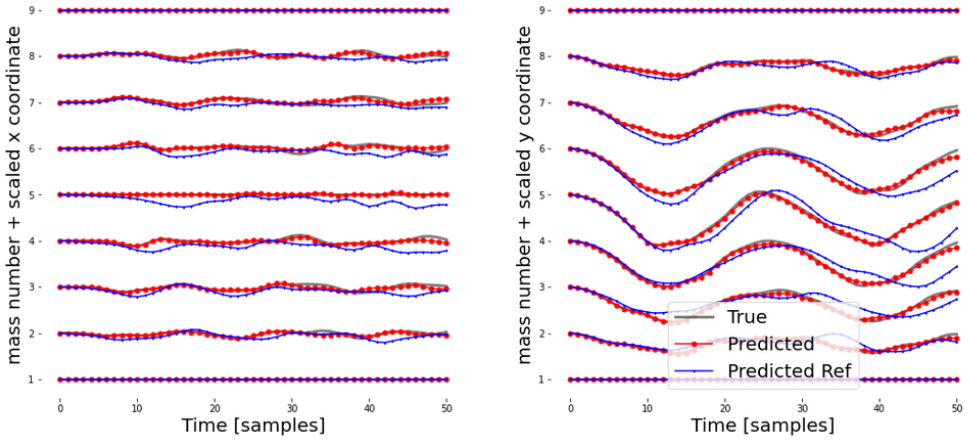

Fig. 6 Example of Spring System Roll-out

B. HALE aircraft results

In this case performance is assessed by the capability to approximate the next time step given the input graph. Since no reference result is available, this assessment is of qualitative nature. Error measurements are nevertheless computed: Fig 9, 10 summarise the Mean Squared Error (MSE) and the Maximum Absolute Error (MAE) per attribute, after normalization. A good capability to model next step attributes is visible, in particular derivatives seem to be well captured. Fig 7 shows a selection of global attributes : vertical position, velocity and angular pitch rate for the 4 gusts of the validation dataset. Fig 8 shows the nodal velocities for all nodes on the wing.

A auto-regressive roll-out of the model from initial conditions has been performed and it shows a reasonable convergence for 20 samples before drifting. Run time was 0.11s, compared to 75s for the original simulation for the same time span , resulting in a speed-up of 680x. This shows the potential benefit in using the GN identified models as a surrogate models for tasks requiring computing a large number of cases.

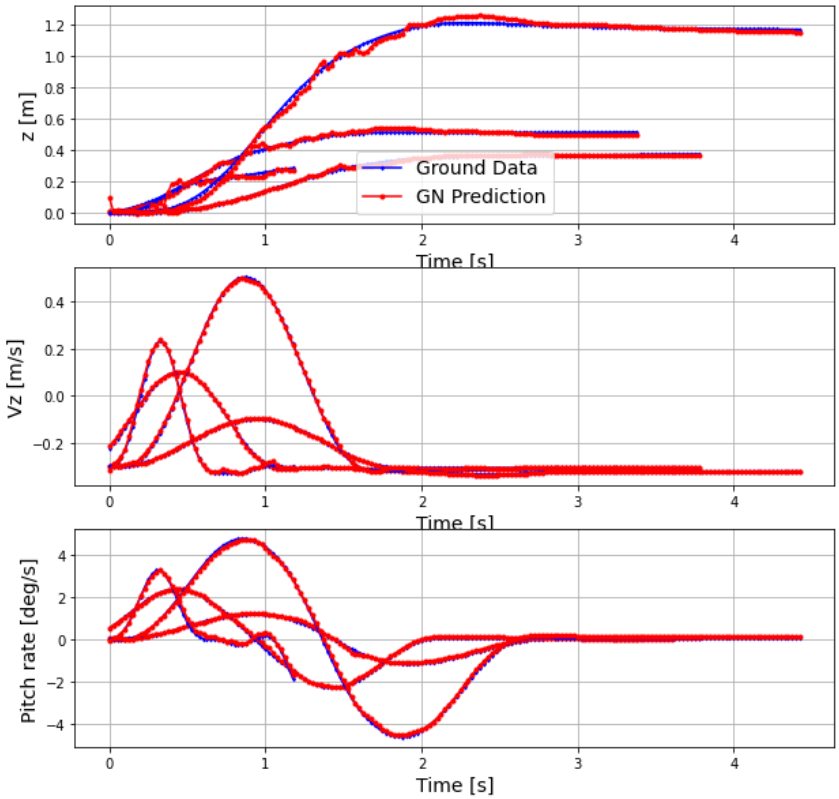

Fig. 7 Prediction on Next time step - Global attributes

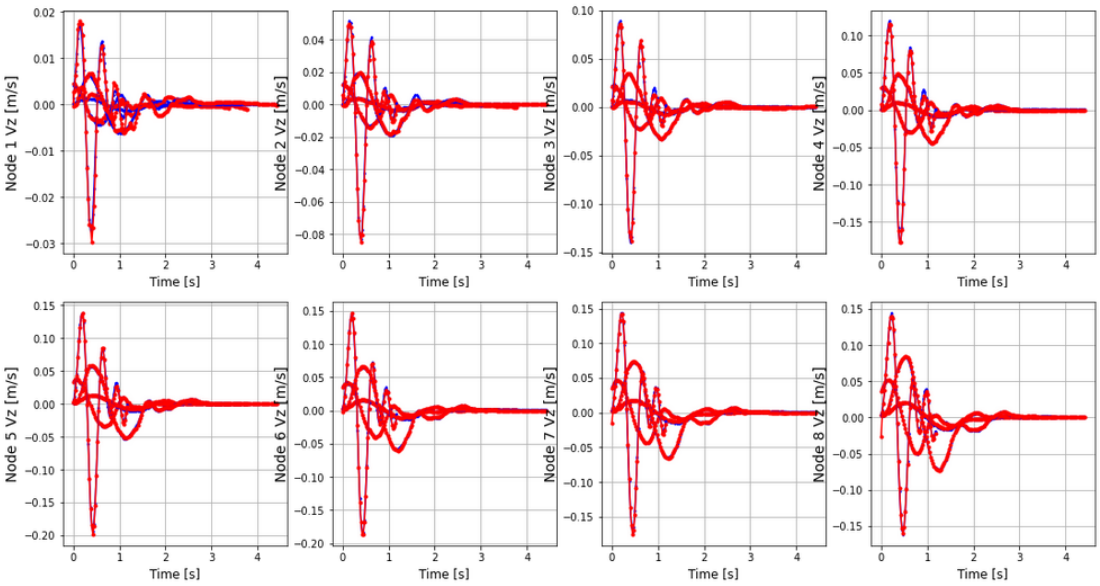

Fig. 8 Prediction on Next time step - Nodal vertical velocities

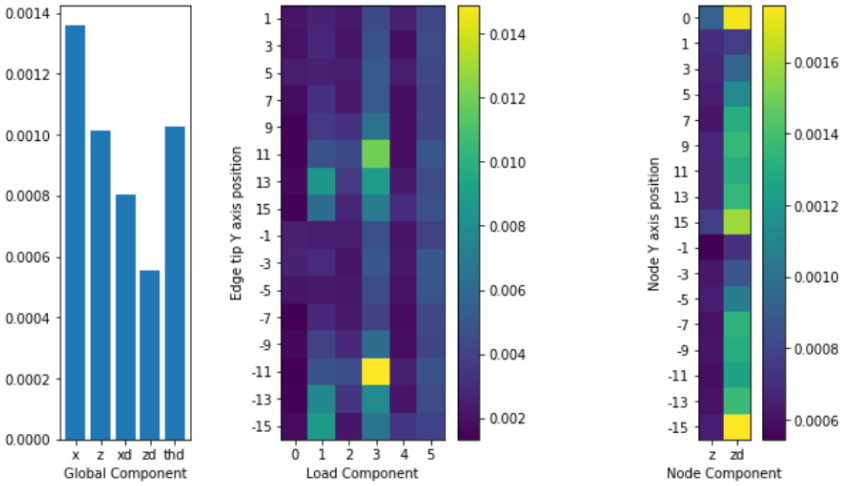

Fig. 9 Mean Squared Error - MSE

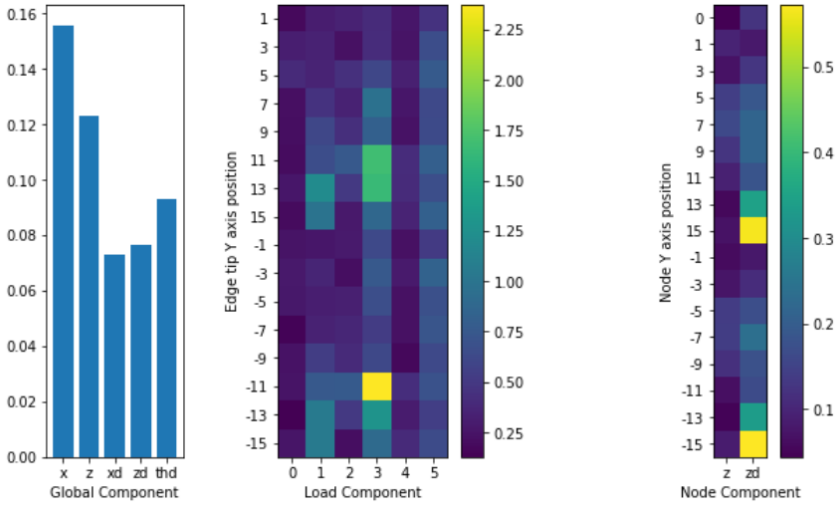

Fig. 10 Maximum Absolute Error - MAE

VII. Conclusion

A Graph Network Model has been trained to learn the dynamics of an aeroelastic aircraft. The GN model succeeds in approximating the behaviour of the entire model, including nodal displacements and structural loads for a next-time step prediction.

Using the model as a surrogate model to replace the calculation of aircraft dynamics showed predictable issues with drift errors but also extremely quick calculation times. The Authors believe that overcoming these issues may lead to use the identified GN as a surrogate model for applications such as MPC or Design optimizations.

The quality of this approximation could be improved by replacing the used FCNN used as update functions by identified symbolic mathematics or other forms of neural Networks. Using symbolic mathematics identification could also yield valuable engineering knowledge on the nature of the interactions that the GN identifies. Further directions to improve results include building larger datasets, and condition the learning process by including a short term roll-out in the loss of the GN.

The main contribution given by this work is applying a novel framework that is designed to merge information from different connected sources. The GN methodology exploits the aeronautical engineering knowledge on connections between entities and some very general assumptions on the nature of their interactions. In particular, although simulations were used to build the data, the choice of attributes has been taken with regard to engineering experience in flight

testing. To our knowledge most work in identifying the dynamics of an aircraft using flight dynamics data and load data normally is based on more restrictive sets of hypothesis.

Acknowledgments

The Authors gratefully acknowledges the support provided by Airbus Operations SAS.

References

[1] Hornik, K., Stinchcombe, M., and White, H., "Multilayer Feedforward Networks Are Universal Approximators," *Neural Networks*, Vol. 2, No. 5, 1989, pp. 359–366. https://doi.org/10.1016/0893-6080(89)90020-8.

[2] Brunton, S. L., Proctor, J. L., and Kutz, J. N., "Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems," *Proceedings of the National Academy of Sciences*, Vol. 113, No. 15, 2016, pp. 3932–3937. https://doi.org/10.1073/pnas.1517384113.

[3] Kaiser, E., Kutz, J. N., and Brunton, S. L., "Sparse Identification of Nonlinear Dynamics for Model Predictive Control in the Low-Data Limit," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 474, No. 2219, 2018, p. 20180335. https://doi.org/10.1098/rspa.2018.0335.

[4] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R., "Relational Inductive Biases, Deep Learning, and Graph Networks," , 2018. URL http://arxiv.org/abs/1806.01261.

[5] Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P., "Graph Networks as Learnable Physics Engines for Inference and Control," , 2018. URL http://arxiv.org/abs/1806.01242.

[6] Lemos, P., Jeffrey, N., Cranmer, M., Ho, S., and Battaglia, P., "Rediscovering Orbital Mechanics with Machine Learning," , 2022. URL http://arxiv.org/abs/2202.02306.

[7] Han, X., Gao, H., Pffaf, T., Wang, J.-X., and Liu, L.-P., "Predicting Physics in Mesh-reduced Space with Temporal Attention," , 2022. URL http://arxiv.org/abs/2201.09113.

[8] Madsen, A., and Johansen, A. R., "Neural Arithmetic Units," , 2020. URL http://arxiv.org/abs/2001.05016.

[9] Trask, A., Hill, F., Reed, S., Rae, J., Dyer, C., and Blunsom, P., "Neural Arithmetic Logic Units," , 2018-08-01. URL http://arxiv.org/abs/1808.00508.

[10] Wynn, A., Artola, M., and Palacios, R., "Nonlinear Optimal Control for Gust Load Alleviation with a Physics-Constrained Data-Driven Internal Model," American Institute of Aeronautics and Astronautics, 2022. https://doi.org/10.2514/6.2022-0442.

[11] del Carre, A., Muñoz-Simón, A., Goizueta, N., and Palacios, R., "SHARPy: A dynamic aeroelastic simulation toolbox for very flexible aircraft and wind turbines," *Journal of Open Source Software*, Vol. 4, No. 44, 2019, p. 1885. https://doi.org/10.21105/joss.01885, URL https://doi.org/10.21105/joss.01885.

[12] "Certification Specifications for Large Aeroplanes (CS-25)," ????, p. 617. URL https://www.easa.europa.eu/sites/default/files/dfu/CS-25_Amdt%203_19.09.07_Consolidated%20version.pdf.

[13] Cheng, L., Bauerheim, M., Illarramendi, E. A., and Cuenot, B., "PlasmaNet: A Framework to Study and Solve Elliptic Differential Equations Using Neural Networks in Plasma Fluid Simulations," ????, p. 6. URL https://ml4physicalsciences.github.io/2021/files/NeurIPS_ML4PS_2021_104.pdf.