AN INTEGRATED DATA GENERATION PROCESS FOR FLIGHT DYNAMICS MODELING IN AIRCRAFT DESIGN

R. Kuchar, German Aerospace Center (DLR), Institute of System Dynamics and Control (SR), Oberpfaffenhofen, 82234 Wessling, Germany

Abstract

Flight mechanics analysis, loads analysis and control law synthesis share a common requirement – the availability of rigid and flexible flight dynamics models. These models usually derive input datasets from a multitude of engineering domains. In order to utilize these models early-on in the design process – specifically for the development of controlled simulation models at a preliminary design stage, methods have to be developed that allow sufficiently accurate input data decks from limited base information. In order to be capable of multi-configuration analysis, a sufficient degree of automation for each of the methods as well as for the overall generation process has to be achieved. This paper presents concepts and methods for this task – together with a specific insight regarding a derived method for the utilization of a Matlab/Catia link via the MS Windows COM interface together with respective applications.

1. INTRODUCTION

Currently preliminary aircraft design focusses mainly on design for performance. On the contrary, aspects of flight dynamics and control are under reduced supervision in standard design procedures – most likely due to the lack of sufficient models at this early stage. This situation potentially causes respective analysis to be performed significantly later in the overall design process – thus often causing major redesign activities, delays and exceeding costs.

In order to bypass these problems, current activities at the DLR Institute of System Dynamics and Control (DLR-SR) focus on the creation of a process chain capable of generating sufficient model data for rigid and flexible flight dynamics models (FDM) from reduced available pre-design data. The derived models can then be used in extensive analysis procedures – thus allowing in-depth investigations regarding flight-mechanics, handling-qualities and controllability analysis.

As most contemporary aircraft systems are equipped with flight control systems, it is of interest to analyze the flight mechanics models in open- as well as closed-loop configurations. For this purpose a Rapid Control Prototyping (RCP) process for the automatic generation of Flight Control laws (FCL) can be used – utilizing model based Non-Linear Dynamic Inversion (NDI) methods for the generation of respective – model dependent – inner-loop control structures.

In other words, the described process tries to shift the generation of valid flight dynamics models from the end of the design process to an early stage, where only limited input data is available. This opens the opportunity to include flight mechanics aspects in the aircraft multidisciplinary design optimization (MDO) process from the very beginning – together with a sufficiently better knowledge of the overall system at later stages.

This paper is basically divided in two parts: The first part tries to give an impression on the requirements and methods used for a model data process chain. The second part presents an in-depth discussion of methods used for direct linking of the numerical software Matlab with the CAD/CAE software Catia via the Microsoft Windows COM interface. The discovered capabilities and drawbacks of this method are discussed as it forms a flexible framework for various tasks and might be of general interest to a wider audience.

2. INTEGRATED DATA GENERATION IN FLIGHT DYNAMICS MODELING

This section is dedicated to a description of the respective input data generation process for the generation of FDM in preliminary design.

Basically the following input data elements are required for the generation of rigid flight dynamics models:

- Base Geometry: Required for determination of characteristic reference data (e.g. wing span, MAC, basic control layout etc.)
- Aerodynamics: Static and dynamic coefficients of the overall aircraft configuration – including control derivatives.
- Mass: Total mass and inertia tensor of the respective aircraft configuration – including fuel, payload and specific fuel flow.
- Propulsion: Thrust tables or respective input data for a dynamic engine model.
- Actuation: The dynamic behavior of the installed actuators together with angular rate limitations and mechanical end stops.
- Optional components: E.g. Landing Gear, Sensor models, etc.

Flexible flight dynamic models are an expansion of the respective rigid versions and consider the fluid-structure interaction during flight. Depending on the aircraft

configuration it is worth taking the occurring flexible effects into account. For example control reversal effects, as well as significantly higher flight loads can be determined in comparison with rigid analysis.

In addition to the above mentioned data structure for rigid models, flexible models require further model input data:

- Mass and inertia distribution: Sectioned fuselage and wing, including payload and fuel. Integrated in the Global FEM model.
- Distributed aerodynamics: Required for fluid-structure coupling.
- Structural stiffness distribution: Integrated in the Global FEM model.

2.1. Requirements for an integrated data generation process

Considering the given input data structure from the listings above, together with practical issues, preprocessing all relevant data samples in an integrated environment faces a number of challenges and requirements:

- The elements of the tool chain are predominantly multidisciplinary and share certain computational dependencies. Therefore the respective execution stack together with the results storage should be controlled by a centralized control unit.
- Three cases can be distinguished regarding the possible starting points in the process chain:
  - Configurations with only very limited information shall be analyzed. Therefore a specific processing branch should be available with tools that are capable of filling the existent data “gaps” with adapted empirical model data.
  - In case sufficient initial data is available (geometry, aircraft base parameters etc.) – but issued in non-conforming input formats. This case is most likely in preliminary design and shall be partially supported with respective utility tools.
  - In cases, where predominantly full databases are existent, the process chain shall be capable of detecting the completeness of the dataset.
- A centralized and well-structured data base should be used in order to allow all relevant tools to access data elements for reading and writing.
- The database shall be structured in a task oriented way, resembling the following elements:
  - Global aircraft parameters
  - Geometry database of all relevant components
  - Mass and Inertia database
  - Structure database
  - Propulsion data
  - Mission data
- The database shall allow the storage and processing of multiple versions of the base configuration with respective documentation.
- The database format should be extensible – as future needs cannot be foreseen at the current stage.
- The database shall allow the link with other data formats – especially formats designed for large data volume storage.
- The individual tool chain elements should be exchangeable in order to use different codes for similar purposes (e.g. different levels of fidelity and methods).
- Existent numerical methods should be integratable with little effort.
- The database content shall be transformable to other established data formats in the field of flight dynamics in order to allow interoperability of the generated datasets.

Based on these considerations, an approach for the implementation of a general data generation process for rigid and flexible flight dynamics models has been found:

- The execution control element can be adopted by a number of existent tools specializing in the control of distributed applications. Examples for software products in this field are DLR RCE/Chameleon [2], [4] or Phoenix Modelcenter [5].
- In order to fulfill the requirements for a centralized database together with the use of existent numerical methods, the use of an established database format is proposed. A feasible format for this purpose is the DLR developed format CPACS [1], [3] – an xml scheme-based format for the coordinated storage of various aircraft configurations. The use of this format would also allow the utilization of tools and datasets generated in other project environments (e.g. DLR projects TIVA II, VAMP, UCAV2010 etc.).
- In addition to this centralized format, the use of large data storage formats is proposed for voluminous datasets as usually generated by 3D aerodynamic panel codes or FEM solvers. An example for a widely used format in this field is the HDF5 binary format [21]. Together with this specific file format, a number of utility tools are available.

2.2. Elements and dataflow for FDM data generation

Based on the defined requirements and various ideas for the implementation a concept has been developed covering all relevant steps in the data generation process.

The process covers the various steps, starting with an evaluation of the available input dataset, proceeding with preprocessing of respective elements and finishing with the generation of data for rigid and flexible aircraft models. Tools of multiple domains are utilized in order to generate datasets as required for flight dynamics models.

As the process sketch had to be separated into two figures – see figure 1 for an impression of the process data flow and methods for rigid FDM generation and figure 2 for flexible FDM generation respectively. At this stage the existent DLR-SR DAMIP process is utilized for the further processing and integration of the generated data in order to derive flexible models – refer to [9] for more information in this field.

During the research for feasible methods in the discussed context focus has been given to the capabilities of the CAD/CAE Catia software. Catia seems to be flexible enough to cover all respective needs regarding geometry representation, mass and inertia measurements – but the question regarding a feasible interface from external software products remained open.

Over time a powerful yet undocumented method could be found – based on information found in [10] – allowing the direct access to Catia via the COM interface. This method is extensively presented in the following section.

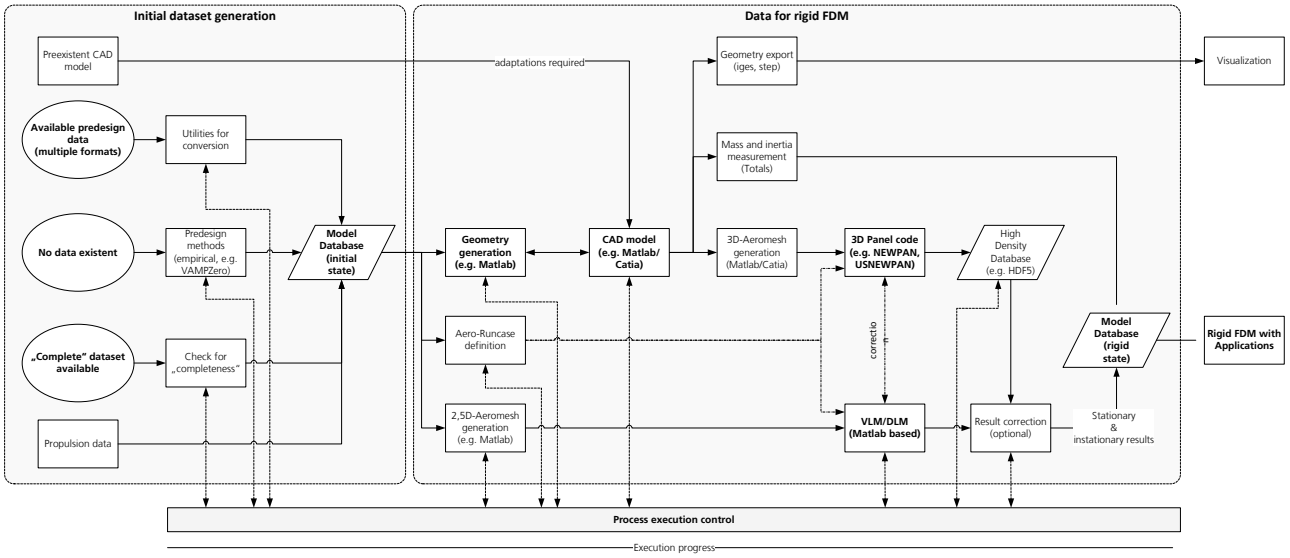

FIG. 1. Process chain for the generation of input data for rigid FDM data generation.

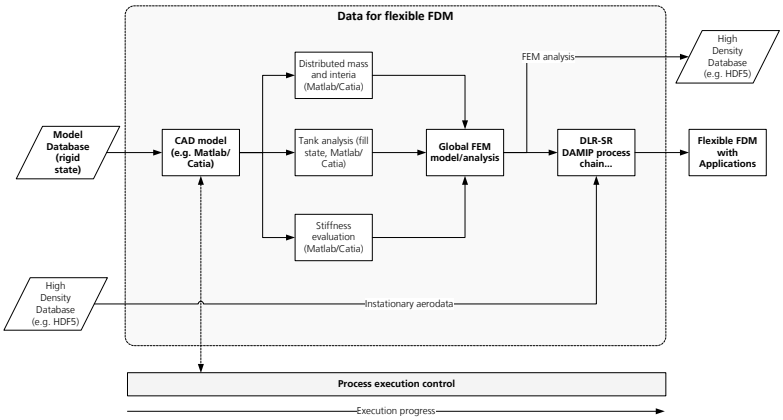

FIG. 2. Process chain for the generation of input data for flexible FDM data generation.

3. LINKING MATLAB AND CATIA

The Mathworks Matlab software is one of the primary tools for numerical computation in the field of engineering and Dassault Catia similarly is one of the standard tools in aerospace engineering for the creation of integrated CAD/CAE models. As of today, an easy-to-use and direct interface between these two software packages is not yet available – although the combination of these two powerful packages seems promising in the context of FDM data generation.

Current workflows involving access to preprocessed data in Catia are based on two technologies: Using fully-parameterized Catia models with attachment to external "construction tables" – Microsoft Excel or ASCII based files with a fixed set of parameters on the one hand side, or utilizing the Catia macro languages VBScript or CATScript for the creation of geometry based on externally imported ASCII files.

CAA Rade – a third – yet more powerful – method is not considered herein, as it requires a sufficient license extension and is primarily focused for the development of Catia 3rd party extension modules.

Nevertheless the CAA Rade API offers access to all native Catia functions through a C++ API. An impression of the various automation levels can be seen in figure 3 – in depth information regarding these capabilities can be found in [11].

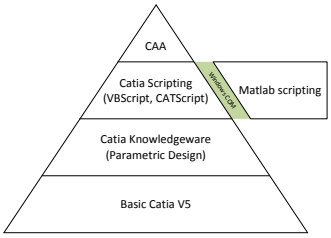

FIG. 3. Catia automation levels – with Matlab COM link categorized

The first two described methods involve major drawbacks at runtime: In order to interface between an external calculation method and Catia, an intermediate step has to

be performed – which causes extra effort for small changes in the scripted data configuration.

In order to overcome these restrictions a novel method has been tried out for its potential use with Matlab and Catia, previously utilized for the coupling of Python and Catia [10] via the Windows COM interface.

3.1. Environmental parameters for the use of the Matlab/CATIA link

In order to utilize the described Matlab/CATIA link through the Windows "Component-Object-Model" (COM) interface – a technology for interprocess communication and dynamical client/server creation, the following pre-requisites have to be fulfilled:

- Microsoft Windows XP, 7 with Service Packs
- MATLAB 2011a or later
- CATIA V5 R20 with standard modules
- ActivePython 2.7 or later – containing the required win32com package

The free ActivePython distribution has been selected, because it includes a COM accessible interpreter together with the required "win32com" package.

On the hardware side it is recommended to use a fully equipped graphics workstation with performance oriented processors and graphics cards.

Although it is possible to execute the entire interface framework between Matlab and Catia on different machines over the network using the DCOM interface, it is recommended to perform all operations on one local machine – due to performance and simplicity reasons.

Additional to these prerequisites the Matlab/Catia link requires further settings to be performed on the Catia side:

- Language settings of Catia have to conform to the scripting framework.
- Units have to be set due to Matlab script standards.
- The manual drawing update property has to be set in order to control this behavior from the executing script.
- Create a link to a material catalogue, when applying material to an item.
- Enable hybrid design in parts and bodies – this opens the opportunity to place surfaces below bodies in the hierarchy tree.

3.2. Basic command syntax for the Matlab COM interface

In order to allow Matlab to use COM services, the internal "Automation Server" service has to be enabled. This can be achieved using the following command (exemplary pseudo-code):

```
retVal = enableservice('AutomationServer', true);
```

As next step the actual COM interprocess communication can be started, using either a preexistent COM service:

```
appHandle = actxGetRunningServer(strAppID);
```

Alternatively a newly instantiated COM service can be established:

```
appHandle = actxserver(strAppID);
```

Where strAppID represents the respective application ID as stated in the MS Windows registry collection – for example:

- *'Matlab.Desktop.Application'*
- *'CATIA.Application'*
- *'Python.Interpreter'*
- *'Excel.Application'*

It has to be noted, that only applications supporting COM operations can be used in collaborative manner together with Matlab. The following statements are based on the use of Catia as COM server in collaboration with Matlab.

As soon as an operative COM server is established further commands can be issued from Matlab to the connected application. It has to be noted, that due to the absence of an explicit declaration of types and internal definitions in Catia, workarounds have been found in order to invoke the respective commands.

Three basic syntax patterns can be distinguished in the context of the Matlab/Catia link:

- *appHandle.set(strParam, valParam)* … set the named parameter with the given value.
  Example: *catia.set('Visible', 1);*
- *appHandle.get(strParam)* … get the value of the named parameter.
  Example: *hSF = CatPart.get('HybridShapeFactory');*
- *appHandle.invoke(strCommand, valParam1, valParam2, …)* … invoke the respective command, together with the individually required parameters.
  Example: *CatBody.invoke('InsertHybridShape', hSF);*

A complete documentation of the useable Matlab functions for COM can be found in the Mathworks product documentation [13].

Most command syntax patterns used in order to control Catia from Matlab can be derived using the standard Catia Macro Recorder. This tool records manually performed tasks in Catia in one of the two available macro languages. This respective code can be used for the transformation to Matlab-specific syntax.

The following example shall clarify the use of recorded CatVBScript code for the transformation into Matlab COM code for Catia:

- CatVBScript (recorded):

```
Set hSR = CatPart.CreateReferenceFromObject(hHSPExplicit)
```

vs.

- Matlab Script (transcription):

```
hSR = CatPart.invoke('CreateReferenceFromObject', hHSPExplicit);
```

Alternatively various resources – e.g. [12] – can be used to prepare a valid strategy for scripting Catia via Matlab, as not all automation strategies are well documented in the Catia references system.

3.2.1. Handling of non-scalar parameters via a Python interpreter gateway

As outlined before, Catia V5 obviously does not include a "TypeDefinition" file, which would enable Matlab to interpret the internal data types of Catia in a sufficient way. As Catia uses the Visual Basic specific type "VariantArray" for vector and array data, this causes problems in the handover between the two linked applications. Without any further information these circumstances allow only scalar values and parameters to be directly manipulated between Matlab and Catia.

Most interestingly there has been a workaround detected for this problem: The COM instance of a Python interpreter can be used as "Man-in-the-Middle" software, receiving data from one application and transforming/typecasting it appropriately to the other involved COM applications.

See the following sketch for a respective presentation of the problem:

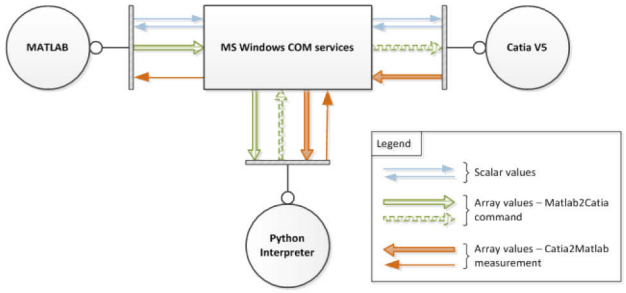

FIG. 4. Matlab to Catia link with intermediate Python array conversion

The following syntax examples show how to establish the Python interpreter link from Matlab (exemplary pseudo-code):

- Start a Python interpreter COM instance in Matlab:

  PyCOM = actxserver ('Start', 'Python.Interpreter');

- Load the "win32com" package in Python – utilizing the Python "exec" command for the execution of the added command string:

  PyCOM.invoke('exec', 'import win32com.client');

- Establish a "backlink" to the single Matlab desktop in order to access all data in the Matlab workspace from Python:

  PyCOM.invoke('exec', 'MatCOM = win32com.client.Dispatch("Matlab.Desktop.Application")');

- Create an empty array:

  PyCOM.invoke('exec', 'dCatArray = []');

- Loop through the Matlab array and rebuild the respective Matlab array in Python:

  PyCOM.invoke('exec', 'dTempPy = MatlabCOMServer.GetWorkspaceData("dTempMat", "caller")');

  PyCOM.invoke('exec', ' dCatArray.append(dTempPy)');

- Obtain the Catia COM object from the Matlab workspace:

  PyCOM.invoke('exec', 'CATCOMObjectPy = MatCOM.GetWorkspaceData("CATCOMObject", "caller")');

- After assembly of the entire command string – execute the respective command:

  PyCOM.invoke('exec', 'strCATCommandPy');

In opposite direction it is possible to determine data arrays – as generated by Catia measurements (mass and inertia) – in a similar way. This functionality uses the Catia "TechnologicalObject" method in order to derive the various features embedded therein:

PyCOM.invoke('exec', 'Inertia = catiaObject.GetTechnologicalObject("Inertia")');

Again the problem with data type conversion of internal Catia arrays appears, but can be resolved in a similar way, as described above.

- In principle it is possible to use preexistent Catia models, if their internal structure allows complete parsing of the content. It is not possible to use model representations, if the internal structure contains referenced parts and bodies.

Currently the framework is used for the generation of geometrical models, splitting operations, measurement of mass, inertia and coordinates as well as distances. Future uses can also include dynamical data exchange between Matlab/Simulink and Catia – e.g. the use of Simulink in-the-loop with Catia can be imagined for various applications.

3.3. Application of the Matlab/Catia link in the context of distributed mass measurement

One of the main applications of the developed Matlab/Catia link is the generation of distributed mass models. These models are required in the process of establishing a Global FEM input deck for flexible structural model generation as required in the process for flexible Flight Dynamics Models and in Loads analysis processes.

The following steps have to be performed in Catia – controlled by the respective Matlab scripts – in order to determine the distributed aircraft mass and inertia tensors:

3.3.1. Fuselage generation and measurement

The following procedure is performed for the distributed mass and inertia analysis of the fuselage:

- Generation of the complete fuselage loft based on the database.
- Thickening of the generated loft surface in order to apply material properties on the created voluminous body.
- Applying an estimated surface "density". This value can be determined by obtaining the precalculated fuselage empty mass from database and dividing it by the measured surface area of the respective element.

- In case the fuselage shall be analyzed with payload an optional generation procedure for payload (e.g. LD3 containers) and dummy passenger models at defined locations can be run. See figure 5 for an impression of the generated aircraft with payload.
- If the assembly of the fuselage is finished, the segmentation of the fuselage can be performed. The Catia “Split” function is used sequentially along the fuselage main axis. See figure 6 for an impression of the splitting procedure.
- Every segment is then measured for mass and inertia – the respective values are written to an intermediate mass database.

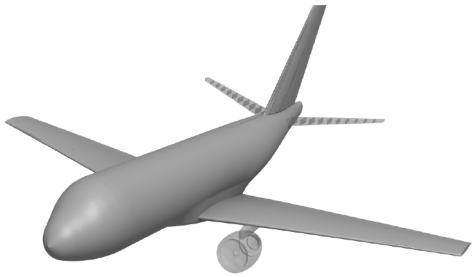

FIG. 5 Catia model of a single aisle aircraft model

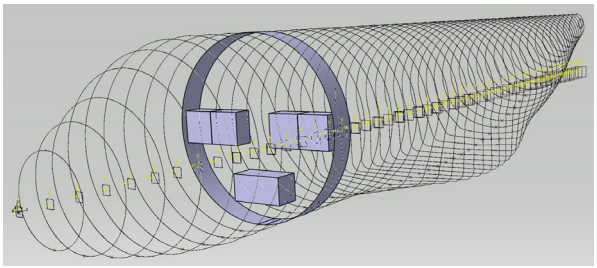

FIG. 6 Sectioned fuselage model (including pax and payload) for distributed mass and inertia analysis

3.3.2. Wing generation and measurement

The following procedure is performed for the distributed mass and inertia analysis of the wing components (main wing, horizontal tail, vertical tail, canards – if existent):

- Generation of the complete wing loft based on the CPACS datasets
- Thickening the generated loft surface in order to apply material properties on the created voluminous body.
- Applying an estimated surface “density”. This value can be determined by obtaining the precalculated wing empty mass (e.g. from database) and dividing it by the measured surface area of the respective element.
- Generation of fuel tanks (optional) – these require a front and rear spar – forming the actual chord-wise boundaries of the fuel tank system.
- Generation of fuel cells (optional): In defined span-wise locations (normally coinciding with rib distance), planes are used to form a defined fuel cell. These cells can be used separately to perform fuel mass and inertia tensor analysis based on variable fill levels. These can be set by two means:
    - Fill levels – defined by the linear distance between the lowest and topmost corner layer.
    - In order to perform analysis at various angles (static flight path) – the entire aircraft model can be rotated in order to generate the respective tilted fuel configuration – see figure 7 for an impression.
- Apart from fuel analysis – the overall wing elements can be analyzed for mass and inertia properties in a similar way as the related fuselage method – see figure 8.

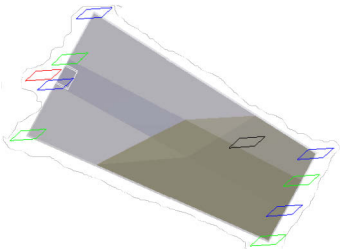

FIG. 7 Example of a partially filled tank as used for mass analysis of fuel tanks.

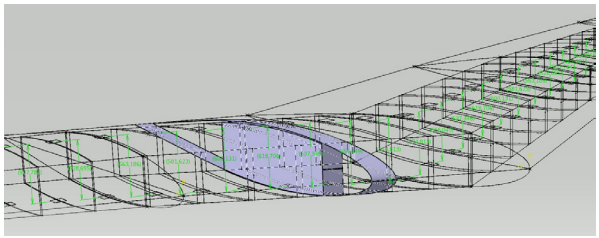

FIG. 8 Sectioned wing element model (including tank)

3.3.3. Mass and inertia integration in Global FEM

The derived distributed data of the mass database can then be used to set up a Global FEM model – covering the various fuel fill states and payload configurations. Stiffness and material data have to be predefined, as a sizing loop is not considered at the present state.

Mass and inertia data are at the moment stored in a defined Matlab-based format – for future use it is intended to integrate these elements into the overall database. Furthermore this data has to be transformed into MSC Nastran input decks in order to be used in the Global FEM evaluation.

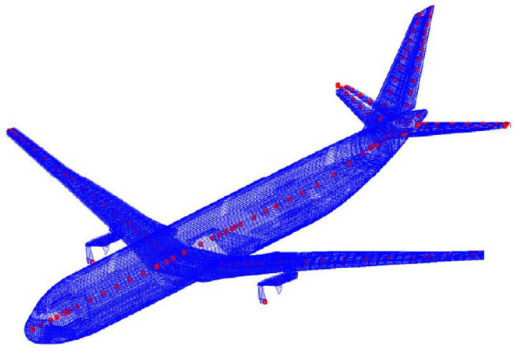

FIG. 9 Global FEM with integrated distributed mass elements.

In a next step the respective computational results can be reduced in order (Guyan reduction) and integrated into the Varloads framework [8] for further analysis – e.g. flight loads analysis in open- and closed loop simulations of the flexible aircraft. This step also requires the calculation of

aerodynamic properties – in the case of flexible structure models the respective unsteady aerodynamics calculation results are required.

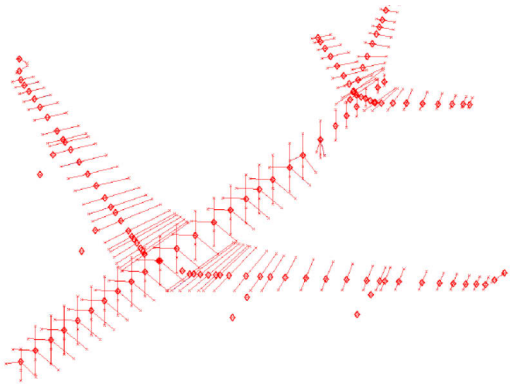

FIG. 10 Condensed (reduced order) FEM model after Guyan reduction

3.4. Aerodynamic data set generation

Aside the determination of mass properties, the computation of aerodynamic datasets is an important element in the data generation process for rigid and flexible Flight Dynamics models.

For the application at flexible flight dynamics models and loads analysis it is essential to calculate distributed aerodynamical datasets for stationary and instationary cases.

For performance reasons only data generated by low fidelity tools for steady and unsteady aerodynamics (e.g. the “vortex lattice method” VLM and the “doublet lattice method” DLM) are considered for employment in the actual flight dynamics models. Nevertheless, middle fidelity tools (e.g. 3D steady panel code NEWPAN and unsteady counterpart USNEWPAN) are considered for the generation of correction data in the context with lower fidelity tools.

The respective meshes can be generated by the use of the central database:

- Vortex lattice codes require thin double-sided quadrilateral panels. Therefore the chord plane of either fuselage or wing is used for the generation of the respective panel geometries.
- 3D panel codes on the contrary can be used with either flat double-sided or single-sided body panels. Therefore the generated geometry can be utilized to derive respective 3D panel configurations for this purpose.

Meshing for 3D panel codes can be established using similar methods of the scripted Matlab/Catia link as discussed for the generation of fuselage and wings. Basically the following procedure has to be followed:

- The intersection curves between fuselage and wing elements have to be computed.
- Then the single wing segments along the fuselage main axis can be determined – using the min/max functionality of Catia with respect to the generated intersection curves.
- The connection of min/max points of the various wing segments can be used in order to separate the fuselage into upper and lower shell segments.
- The created segments are then used in order to apply a defined discretization – along the main axis specified intersection curves are derived.
- Along the “half” intersections points are distributed with defined spacing.
- All point coordinates can now be measured using the “Offset” edition of the “GetTechnologicalObject” feature. The respective coordinates are stored in a Matlab structure.
- The stored points are then sorted and converted into a fitting input format for the 3D panel method.

Aside the pure generation of panel geometries, panel methods depend on the creation of wake surfaces for each lifting component. These have to be defined manually via IDE, as there is currently no method for the automatic generation available. This is due to the fact that “wake collisions” cannot be ruled out in an automatic process at the current state.

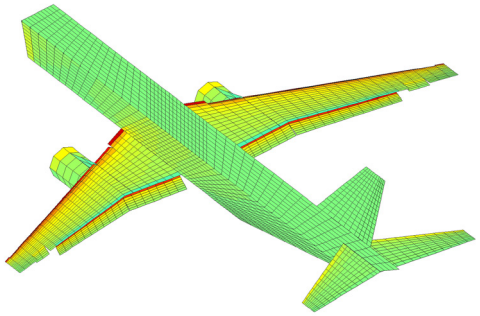

FIG. 11 VLM grid and pressure distribution of a single aisle aircraft

3.5. Further applications of the developed Matlab/Catia link

The established framework functions have also been in domains other than aircraft related modeling. It has been used as an integrated element in the Pareto based design process for an electrical generator. Therein it delivered geometrical models for a multitude of configurations together with respective mass and inertia data as can be seen in [14].

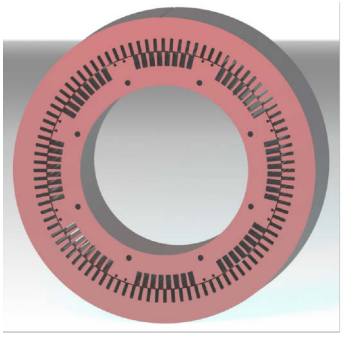

FIG. 12 Exemplary generator layout as generated with the Matlab/Catia link

In this context the generated model helped in the clarification of the involved parameter laws. All relevant rules are evaluated prior to the actual design process – thus allowing the knowledge of all calculated parameters also on the Matlab side.

The used method is based on a drawing of the respective cross-section in a Catia “Sketch”, then extruding the defined geometry and applying defined materials to the generated elements. At the end of that treatment, all material properties can be measured.

3.5.1. Advantages and drawbacks of the utilized method

One of the most valuable advantages of the described method is the added flexibility as there are no links to Catia construction tables, ASCII files etc. required. The described workaround for arrays is fully encapsulated and therefore not of practical relevance during runtime. In addition to that, an object oriented approach is used for the developed framework. This enables the re-use of once generated Catia content for different purposes. Consequent re-use also saves computational time for rebuilding the entire or parts of the model. Furthermore it is possible to create a new software layer – thus encapsulating the most used features and allowing an easy-to-use interface.

The actual performance is determined by a number of parameters:

- Complexity of the configuration
- Number of measurements – as this feature requires computational effort
- Visibility of the Catia GUI requires computational effort

During the implementation of the respective code a number of disadvantages and/or inconveniences where discovered:

- The overall performance is an issue that can only partly be resolved – e.g. by disabling the visibility of Catia and consequent object re-use.
- The handling of non-scalar data requires the involvement of additional COM components (Python).
- The developed methods are only available under the MS Windows platform.

Considering the performance of the applied Matlab/Catia link in the framework of the presented methods for the generation of fuselage and wing elements together with mass and inertia measurement, the following observations can be made:

- Fuselage generation: 51 sections with 100 control points each – requires 7s.
- Wing generation: 7 sections with 80 control points and 10 fuel tank boxes each – requires 6s.
- Splitting and measurement of the fuselage: 51 sections – requires 50s.
- Splitting and measurement of 140 wing-“half-tanks” – requires 160s.

All durations are based on a system with Intel Core i7, 2.0 GHz and 4GB RAM with the software setup as described in section 3.1.

4. FLIGHT DYNAMICS MODEL ASSEMBLY USING THE DLR-SR MODELICA FLIGHT DYNAMICS LIBRARY

If all relevant datasets have been successfully computed, the collected data can now be used for the creation of the actual Flight Dynamics model. Therefore all in- and output ports have to be assigned with standardized interfaces that allow a predominantly generic base model to be established for the majority of use-cases.

The actual model environment is based on the DLR-SR Modelica Flight Dynamics Library in collaboration with the Dymola IDE. Modelica itself is an object-oriented, equation based, multi-domain modeling language for the component oriented modeling of complex systems.

The Flight Dynamics library framework offers a multitude of advantages regarding model specifications for various applications: 3-DOF models for mission simulation, 6-DOF models for full featured flight simulation and finally automatic generation of inverted 6-DOF models for flight control law synthesis. Especially these inverted models (based on the Nonlinear Dynamic Inversion concept) can be used as a core element in the generation of a Rapid Control Prototyping process as proposed in the overall FDM concept.

Utility methods regarding the integration of external databases into the frame of the Flight Dynamics library are existent and are going to be extended.

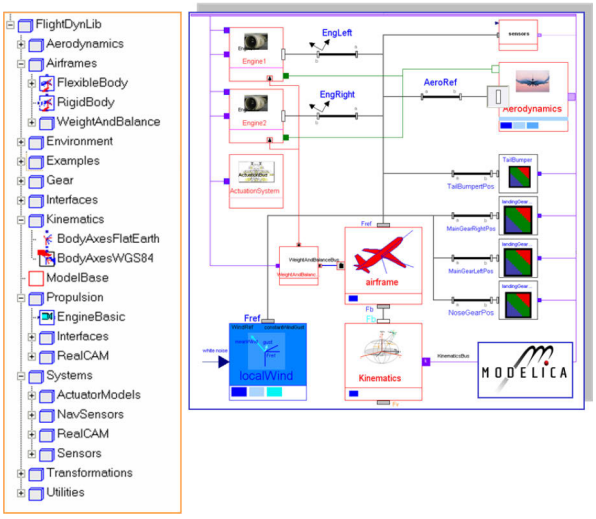

FIG. 13 The Modelica based DLR-SR Flight Dynamics Library

One of the core characteristics of the Modelica/Dymola approach is the automatic generation of optimized C-code from each model. This allows the direct integration in general-purpose simulation frameworks.

As all relevant methods in the data generation process are focused on performance and data reductions methods, the resulting rigid and flexible flight dynamics models are clearly real-time capable.

A complete discussion of the DLR-SR Flight Dynamics library and its applications is available in [17] and [18].

5. SUMMARY AND OUTLOOK

The presented methods for the generation of model data provide a flexible, yet powerful and extendable framework for the generation of FDM models for various applications in the aircraft pre- and preliminary design process. The process and its individual methods are under continuous extension – based on specific requirements.

A flexible method for the direct linking of Matlab and Catia has been discussed, involving the use of a Python interpreter for non-scalar data. Apart from the presented use cases a number of further interesting applications can be considered involving direct in-the-loop interaction between Simulink and Catia for simulation purposes for example.

It is planned to apply the resulting process chain in the framework of the Cassidian Open Innovation/Sagitta project. Respective utility methods for this use case have to be developed, easing the integration of existent design datasets into the respective FDM database format. Due to the specific configuration of this aircraft, a number of adaptations in the overall code are likely to be required.

Furthermore it is intended to use the resulting flight dynamic models in a number of interesting applications ranging from multi-configuration analysis of open-loop models, the application of Rapid Control Prototyping methods for closed loop analysis, loads analysis and determination of valid operational areas (e.g. envelope protection) to the use as simulation model for pilot training, software-in-the-loop and finally as “flying” platform in the context of integration testing.

6. ACKNOWLEGMENT

The author would like to thank G. Looye and T. Kier of DLR-SR for their continuous interest and valuable comments regarding the FDM generation chain and model integration.

The author also wishes to thank J. Seifert and C. Munzinger of Cassidian for their interest and in the ongoing activities in respect to the overall FDM process chain and Rapid Control Prototyping activities.

Thanks are due to C. Rink for his active support in the implementation and testing of the Matlab/Catia framework.

7. REFERENCES

[1] A. Rizzi, M. Zhang, B. Nagel, D. Böhnke, P. Saquet: Towards a Unified Framework using CPACS for Geometry Management in Aircraft Design, AIAA Aerospace Sciences Meeting, Nashville, USA, 2012.

[2] Bachmann, A., Kunde, M., Litz, M., Schreiber, A.: Advances in Generalization and Decoupling of Software Parts in a Scientific Simulation Workflow System, Fourth International Conference on Advanced Engineering Computing and Applications in Sciences, Florence, Italy, 2010.

[3] CPACS – A Common Language for Aircraft Design and related tools (OpenSource): http://code.google.com/p/cpacs/

[4] RCE/Chameleon – distributed, collaborative problem solving environment software (OpenSource): http://code.google.com/a/eclipselabs.org/p/rce/

[5] Bachmann, A., Kunde, M., Litz, M., Schreiber, A., Bertsch, L.: Automation of Aircraft Pre-design Using a Versatile Data Transfer and Storage Format in a Distributed Computing Environment, Third International Conference on Advanced Engineering Computing and Applications in Sciences, Sliema, Malta, 2009.

[6] VAMPzero - Conceptual Design for the Needs of MDO (OpenSource): http://code.google.com/p/vampzero/

[7] Guyan, R. J., Reduction of stiffness and mass matrices, American Institute of Aeronautics and Astronautics Journal 3(2), 380, 1965.

[8] Hofstee, J., Kier, T., Cerulli, C., and Looye, G.: A Variable, Fully Flexible Dynamic Response Tool For Special Investigations (VarLoads). Technical report Loads & Aeroelastics, Airbus Deutschland GmbH; DLR – German Aerospace Center, Institute of Robotics and Mechatronics; Delft University of Technology, Faculty of Aerospace Engineering, Delft, Netherlands, 2003.

[9] Kier, T. and Looye, G. and Scharpenberg, M. and Reijerkerk, M.: Process, methods and tools for flexible aircraft flight dynamics model integration. International Forum on Aeroelasticity and Structural Dynamics, Stockholm, Sweden, 2007.

[10] Brodd, H.: Vollautomatisiertes Erstellen eines parameterisierten Bauteils in CATIA V5, gesteuert über die COM-Schnittstelle mittels Scriptsprache Python, Studienarbeit, Bergische Universität Gesamthochschule Wuppertal, Wuppertal, 2006.

[11] Hansen, J.: Kochbuch CATIA V5 automatisieren., Carl Hanser Verlag München, 1st Edition, 2009.

[12] Ziethen, D. R.: CATIA V5 Makroprogrammierung mit Visual Basic Script., Carl Hanser Verlag München Wien, 2nd edition, 2006.

[13] The Mathworks Product Documentation (Matlab): http://www.mathworks.de/help/techdoc/

[14] Kuhn, M. R.: Advanced generator design using Pareto-optimization. Proceedings of 9th PEDS Conference, Singapore 2011.

[15] Moormann, D.: Automatisierte Modellbildung der Flugsystemdynamik. VDI Verlag, Reihe 8, Dissertation RWTH Aachen, Institut für Flugdynamik, Düsseldorf, 2002.

[16] Looye, G.: An Integrated Approach to Aircraft Modeling and Flight Control Law Design. Dissertation, Delft University of Technology, Delft, Netherlands, 2008.

[17] Looye, G.: The new DLR Flight Dynamics Library, Modelica Conference 2008, Germany.

[18] Looye, G.: Rapid prototyping using inversion based control and object-oriented modeling. Chapter 8, Lecture notes in Control and Information Sciences. Springer Verlag, Berlin, 2007.

[19] Otter, M., Blochwitz T., Elmqvist H., Junghanns A., Mauss J., Olsson H., Functional Mockup Interface – Overview, Modelisar, 2010. http://synchronics.inria.fr/lib/exe/fetch.php/modelica-fmi-elmqvist.pdf

[20] Murri, D. and Jackson, Bruce E.: Flight Simulation Model Exchange. NASA/TM-2011-217085, NASA, Langley Research Center, 2011.

[21] HDF5 – File format specification (OpenSource): http://www.hdfgroup.org/HDF5/doc/H5.format.html