2017

Automated Construction of Layer-by Layer Finite Element Sub-Models of Damaged Composites Based on NDE Data

Stephen D. Holland
Iowa State University, sdh4@iastate.edu

Adarsh Krishnamurthy
Iowa State University, adarsh@iastate.edu

Onur Bingol
Iowa State University, orbingol@iastate.edu

Robert Grandin
Iowa State University, rgrandin@iastate.edu

Automated Construction of Layer-by Layer Finite Element Sub-Models of Damaged Composites Based on NDE Data

Abstract

Composite laminate structures are usually modeled as a shell in finite element analysis tools for strength and stiffness determination. However, modeling for fatigue or degradation analysis often needs to be performed with layer-by-layer solid models, but building these models for nontrivial geometries can be extremely difficult, especially when trying to represent realistic defects. This paper discusses how the process of generating layer-by-layer solid finite element models, including insertion of defects, can be automated. We have developed a tool, Delamo, to automate the construction of such models. The tool provides an interface to a commercial solid modeling kernel (ACIS) and a commercial finite element analysis package (ABAQUS). It allows the solid model and finite element model to be built in parallel, layer by layer, starting with a mold, following the same assembly steps as the physical laminate. The bonding step determines the boundary conditions to be applied in the finite element model. Delaminations, determined from nondestructive evaluation (NDE) data can be inserted between layers as needed and are represented as unbonded regions. Potential delamination growth regions can be modeled with a cohesive layer or cohesive boundary condition. Fiber breakage in a layer will be represented by an internal boundary. Based on a mold and a sequence of layer construction and bonding instructions, the tool generates both a solid model and a Python script for ABAQUS that will generate a complete finite element model based on that solid model.

Disciplines

Materials Science and Engineering | Structures and Materials

Automated Construction of Layer-by Layer Finite Element Sub-Models of Damaged Composites Based on NDE Data

STEPHEN D. HOLLAND, ADARSH KRISHNAMURTHY, ONUR BINGOL and ROBERT GRANDIN

ABSTRACT

Composite laminate structures are usually modeled as a shell in finite element analysis tools for strength and stiffness determination. However, modeling for fatigue or degradation analysis often needs to be performed with layer-by-layer solid models, but building these models for nontrivial geometries can be extremely difficult, especially when trying to represent realistic defects. This paper discusses how the process of generating layer-by-layer solid finite element models, including insertion of defects, can be automated. We have developed a tool, Delamo, to automate the construction of such models. The tool provides an interface to a commercial solid modeling kernel (ACIS) and a commercial finite element analysis package (ABAQUS). It allows the solid model and finite element model to be built in parallel, layer by layer, starting with a mold, following the same assembly steps as the physical laminate. The bonding step determines the boundary conditions to be applied in the finite element model. Delaminations, determined from nondestructive evaluation (NDE) data can be inserted between layers as needed and are represented as unbonded regions. Potential delamination growth regions can be modeled with a cohesive layer or cohesive boundary condition. Fiber breakage in a layer will be represented by an internal boundary. Based on a mold and a sequence of layer construction and bonding instructions, the tool generates both a solid model and a Python script for ABAQUS that will generate a complete finite element model based on that solid model.

INTRODUCTION

Composite laminates can easily end up with damage, whether from manufacturing flaws, in-service impacts, or long-term degradation. Nondestructive evaluation

Holland et al.
Iowa State University Center for NDE
1915 Scholl Road
Ames, IA 50011 USA

(NDE) can be used to assess the presence of such damage, but in many cases it is difficult to determine whether a repair is warranted. Composite repairs are expensive and do not necessarily achieve the full original strength. If the damage does not affect structural integrity or fatigue resistance, repair might not be necessary. In this paper we discuss a process for automated construction of a finite element model of a damaged region based in part on input from NDE data.

Finite element analysis is a primary tool used for evaluating the strength and stiffness of composite structures, substructures, joints, and even simple laminates. Finite element models of composite structures or major substructures are usually based on stiffened shells. When damage or internal defects are a concern, especially when considering a fatigue scenario as opposed to ultimate strength, it is necessary to model the defect growth process in detail. The structural performance of a damaged region in terms of failure strength and delamination growth may depend on the orientations of individual plies adjacent to the damage, so a ply-by-ply model is needed in order to evaluate the effect of a defect on structural integrity.

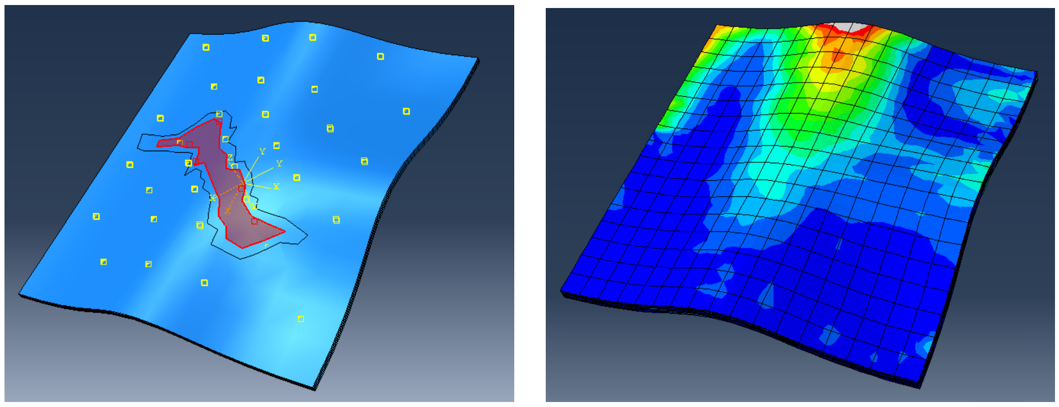
Assembling such a ply-by-ply model is particularly complicated when the structure is not simple and flat, such as with curved surfaces and/or stiffening elements such as hat stiffeners or core materials. Creating such a model is an intricate and error prone task, involving sophisticated computer aided design (CAD) operations, such as "offsetting", to form the contour of a new layer from the contour of the previous layer. After creating the CAD model, it must be imported into a finite element package, and the layers tied together with the proper boundary conditions. The additional complexity of a defect, for example a delamination region represented as a contact zone surrounded by a cohesive zone, makes manual creation of such a model even less practical.

This project automates the construction of joint CAD and finite-element models of composite materials with defects by providing a new tool called Delamo. Constructing a finite element model with Delamo is accomplished by creating a series of instructions for building the model. The instructions follow the composite manufacturing process: First a mold shape (perhaps by analysis of a pre-existing CAD model) is defined. Then, individual lamina are placed onto that mold and bonded into place. If a delamination is present at the layer, then only a sub-region will be bonded. Stiffeners can be added as needed.

The solid model is constructed using a CAD kernel by executing the series of instructions to form the layers from the mold contour. The same process creates a second series of instructions to be executed later by the finite element package (ABAQUS [1], in this case). This second series of instructions includes steps to load in the solid model; apply tie, cohesive, and contact boundary conditions between the layers as appropriate; and perform an analysis. The net result, illustrated for example in figure 1, is a combination solid model and finite element model of the defect region, suitable for direct evaluation or insertion into a structural model.

Related work

DESICOS [2] is a European Union project to improve the accuracy of predictions of compressive buckling strength of composite rocket components, so that a



(a) (b)

Figure 1. Example finite element model generated via the above process and containing an internal delamination: Cutaway showing delaminated surface (a) and stress field under cantilever bending (b). The highlighted area shown in the cutaway is given a contact boundary condition. The thin surrounding strip, which also represents part of the delamination, has no boundary condition (to assist with convergence). The remainder of the bond is represented with a cohesive boundary condition.

less conservative knockdown factor can be used in design. As part of the project, the DESICOS members have developed and published ABAQUS Python scripts to generate shells with certain types of imperfections. The types of defects modeled are a wide range of mostly manufacturing defects that could affect buckling: Dimples, incorrect fiber orientation, cutouts, nonuniform thickness, perturbation loads, etc. They are modeled on a shell basis, as opposed to the ply-by-ply models of arbitrary delaminations or fiber breakage models generated by Delamo.

TexGen [3] is a tool for generating microstructural models of textiles, including woven fabrics and 3D weaves used in advanced composites. It can assemble a model from individual yarns, predict and represent the individual yarn paths, and create a meshed or voxelized representation of the yarns for export to various possible analysis tools, including ABAQUS.

PREPROCESSING FOR FINITE ELEMENT MODELING

A finite element kernel operates on explicit spatial elements, and solves a system of equations representing elasticity, plasticity, boundary conditions, and geometry to determine internal stresses and deformed shapes. In the early days of finite element modeling, shapes and boundary conditions needed to be defined manually, with some automation of the meshing process. The more complicated the shape to be modeled, the more difficult this process became. Starting in the 1990's, pre-processors were developed that could load geometry from CAD, create a mesh, and apply boundary conditions, ultimately generating the input file needed by the finite element kernel. The problem with input processors is that assembly of the model – especially the task of applying boundary conditions – remains a mostly manual (and therefore time consuming and error-prone) process.

When creating explicit ply-by-ply finite element models of nontrivial geometries, this problem becomes even worse. CAD models do not usually include ply-level detail, so a new CAD model needs to be manually created and imported. Then the layers must be identified, and individual faces must be picked to have continuity or cohesive boundary conditions to tie the layers together. If the model is to include a defect, then the process is even worse; the layers need to be subdivided so that the defect region can be identified and picked separately. Any error in picking and applying boundary conditions could cause the finite element analysis process to fail or, worse to give incorrect results.

The latest preprocessors help to address these difficulties by being scriptable. The ABAQUS/CAE and ANSYS Workbench preprocessors both support scripting using the Python language [4, 5]. Nevertheless, scripting only partially solves the problems of creating a ply level finite element model of a composite laminate. Generation of the CAD model needs to be scripted, too. CAD functionality within preprocessors is not generally adequate to define complex shapes, so that means a separate script needs to be created inside the CAD package. Correct identification of boundary condition faces, especially when it is desirable to to be robust to even small changes in geometry, is very difficult. Faces are usually identified by number, and that numbering might change in a new version of the CAD packages. Faces can be identified by the coordinates of a surface point, but even a small geometric change will invalidate all of those coordinates, which would then need to be manually re-evaluated.

Just as the pre-processor automates the use of a finite element kernel within a CAD abstraction, our Delamo tool is a pre-pre-processor that automates the use of a finite element kernel within the abstraction of a lamination process. It allows users to define a laminate in terms the of the mold shape and the layer creation and bonding steps. Based on this definition it creates a solid model plus a Python script for ABAQUS/CAE that loads the model, defines boundary conditions, and prepares an analysis.

DELAMO SCRIPTING AND MODEL ASSEMBLY

The Delamo script is written in Python. An example is shown in Fig. 2. It consists of some front matter, including referencing initialization code for ABAQUS that defines material properties, sections, interaction models, etc. After the front matter, it loads in a NURBS surface [6] to provide a mold shape. Then, it creates a layer of a specified thickness at 0 degree orientation. After the layer is created it is meshed, with parameters specified using the usual ABAQUS Python constants. The second layer is then created, this time at a -45 degree orientation, and that layer too is meshed. The two layers are bonded together with a delamination, referencing contact and cohesion interaction models (defined in the ABAQUS initialization code) and the delamination outline stored in an external file. Additional layers may be added and bonded if desired. Once all layers have been created and bonded, the `Finalize()` call generates a CAD file and the ABAQUS/CAE Python script. The CAD file contains the geometry of the individual layers. The ABAQUS/CAE Python script will

```
# Front matter: Initialize the DeLaMo model
DLO_Model=DelamoModeler.Initialize(globals(),
                                   facepointtolerancefactor=3.0,
                                   normaltolerance=100e-4)
# Reference the ABAQUS initialization script
DLO_Model.abaqus_init_script("abqparams_CFRP.py",globals())

mold = delamo.load_nurbs("mold_shape.nurbs")

# Create 1st layer
layer1 = Layer.CreateFromMold(DLO_Model,mold,
                              delamo.CADwrap.OFFSET_DIRECTION,
                              thickness,"Layer_1",LaminaSection,0)
# Mesh 1st layer
layer1.fe_layer_meshing.MeshSimple(MeshElemTypes,meshsize,
                                   abqC.HEX_DOMINATED,
                                   abqC.SYSTEM_ASSIGN)

# Create 2nd layer
layer2 = Layer.CreateFromLayer(DLO_Model,layer1.gk_layer,
                               delamo.CADwrap.OFFSET_DIRECTION,
                               thickness,"Layer_2", LaminaSection,
                               -45)
# Mesh 2nd layer
layer2.fe_layer_meshing.MeshSimple(MeshElemTypes,meshsize,
                                   abqC.HEX_DOMINATED,
                                   abqC.SYSTEM_ASSIGN)

# Bond layers 1 and 2 with a delamination
bond_layers(DLO_Model,layer1, layer2,
            delamo.CADwrap.BC_DEFAULT_CONTACT,
            CohesiveInteraction, ContactInteraction,
            "data/Delamination1.csv")

# Generate CAD model and ABAQUS/CAE script
DLO_Model.Finalize(cad_file_name, script_to_generate)
```

Figure 2. Example Delamo script for generating a 2-layer laminate (some boilerplate code has been omitted for clarity). The script creates two layers and then bonds them with a delamination.

automatically perform the specified initialization, load the CAD file into ABAQUS as a series of parts, form the parts into an assembly, apply internal boundary conditions between layers, and generate the mesh. All that is left in this case is applying external boundary conditions and running the finite element model to evaluate the resulting deformation.

METHODS

Many of the instructions in the script affect both the CAD model and the finite element model. For example, creating a layer requires the CAD operations of creating an offset surface [7] (a shifted copy of the mold) and forming a body, as well as the finite element operation of loading the resulting bodies as parts from the CAD file and adding them to the assembly. Likewise, bonding two delaminated layers requires imprinting the delamination outline(s) on both layer surfaces (a CAD operation), then applying the appropriate boundary conditions (cohesive, free, or contact) to the correct pairs of faces.

The Delamo script runs outside the ABAQUS/CAE context, but it needs to be able to perform ABAQUS operations such as applying boundary conditions and intersperse those operations with the CAD operations such as imprinting. The Delamo script can even include direct calls to ABAQUS. How is it possible to specify these operations in parallel when ABAQUS is not even running?

The Delamo functions and methods call the ACIS [8] CAD kernel to construct a concrete solid model as the Delamo script executes. When the calls to ABAQUS – the meshing calls or the `bond_layers()` call – are attempted the CAD model is not yet complete, so ABAQUS could not possibly be called directly. Instead, the calls to ABAQUS along with their parameters are captured by proxy objects. The proxy objects are substitutes for ABAQUS objects that exist as variables in the context of the Delamo script. These proxy objects capture and store any operations performed on them, rather than performing the operations directly.

For example, the `MeshSimple()` calls in Fig. 2 are in fact method calls on such a proxy object. Hence when the Delamo script is executed, the `MeshSimple()` call is not in fact executed (it does not even exist in the Delamo script context). Instead, the call to it is recorded as an operation to be performed during the meshing phase of the ABAQUS script. Later, when `Finalize()` is called to generate the ABAQUS script, the `MeshSimple()` calls will be inserted along with all of the other meshing calls.

In this way, model construction calls are performed immediately during execution of the Delamo script, but ABAQUS calls are queued up by topic (initialization, assembly, boundary conditions, meshing, and running the compute job) and then reassembled into the ABAQUS script.

Bonding layers

The trickiest part of the finite element model generation is in identifying the correct faces for applying boundary conditions. The difficulty is compounded by the separation between CAD and finite element domains; the CAD domain is where the faces are created and where their identity is known, but face numbering can not be relied on to stay consistent across domains.

Each layer is created by starting from an initial shape (represented as a NURBS surface) and performing an "offset" operation to create a new surface that is shifted from the original at each location along the surface normal (in general it has a slightly different shape unless the original surface is flat). The two surfaces plus an outer edge are joined to form the boundary representation of a solid body (a “part” in ABAQUS parlance) representing a layer of the laminate. If fiber breakage is to be modeled then this body would be split in two.

When two such layers are to be bonded together using the `bond_layers()` `function`, the top faces of the lower layer and the bottom faces of the upper layer must be identified. Since lists of faces are kept as part of the layer data structure and a global face adjacency list is maintained, at this phase of construction identifying the faces to be bonded is not difficult. The faces are uniquely identified by coordinates of a non-edge point and the corresponding outward normal vector of the body. Since the identification is now just a few numbers, those numbers can be passed in a function call to define the boundary condition between that pair of faces. Per the previous discussion, the call actually goes to an ABAQUS proxy object that queues up the boundary condition assignment call for inclusion in the boundary condition assignment phase of the generated ABAQUS script.

If the bond between the layers has a delamination, then the process is slightly more complicated. Specifically, to get reliable finite element convergence, the delamination outline has to be imprinted on both the top surface of the lower layer and the bottom surface of the upper layer, breaking those surfaces into multiple pieces. The bonded area of a delamination will usually be modeled with a cohesive boundary condition, so that the delamination is capable of growing should it be overloaded. Most of the delaminated area is modeled with a contact interaction property. Leaving a narrow ring at the outer edge of the delaminated region with no boundary condition applied (i.e. free boundary condition) is also helpful for convergence. This ring is generated with an offset curve from the delamination boundary and also imprinted on the two surfaces. As face adjacency is tracked across these imprint operations, it is again relatively straightforward to iterate over the pairs of faces and apply appropriate boundary conditions.

CONCLUSIONS

Delamo is a tool for constructing finite element models of damaged composite laminates. It is intended to be useful in predicting the structural integrity of damaged composite structures. Because damage propagation in a composite material depends on the individual layers, the finite element model needs to resolve individual layers. Building such models for nontrivial geometries by hand would be be extremely difficult, especially when trying to represent realistic defects. Delamo can automate the the process of generating layer-by-layer solid finite element models, including insertion of defects based on NDE data.

The Delamo model is generated from a series of construction and assembly instructions such as definition of a mold, creation of a layer, meshing of the layer, and bonding of layers with and without defects. These instructions are represented as a Python script. Delamo then generates a CAD solid model and a Python script that will load that solid model into ABAQUS and apply appropriate boundary conditions such as contact and cohesive conditions around a delamination. Delamo automates a formerly manual process of building layered finite element structures, making it practical to create ply-by-ply finite element models of complicated layups in curved composite structures.

ACKNOWLEDGMENTS

This work is supported by National Aeronautics and Space Administration (NASA) under contract number NNL15AA12C through the Center for Non-Destructive Evaluation (CNDE) at Iowa State University and is a part of the NASA Advanced Composites Project. We would like to thank Dr. William P. Winfree and Dr. Cheryl A. Rose of NASA for their valuable input and support for this project. We would also like to thank Dr. Ronald Roberts from Iowa State University CNDE for ultrasonic scan data of actual composite delaminations and Ashraf Bastawros of Iowa State University for his assistance with ABAQUS cohesive modeling.

REFERENCES

1. Dassault Systemes, ABAQUS `http://www.3ds.com/products-services/simulia/products/abaqus/` Accessed June 16, 2017.

2. DESICOS: New Robust Design Guidline for Imperfection SensitiveComposite Launcher Structures `http://www.desicos.eu`, Last viewed June 16, 2017.

3. Lin, H., L. P. Brown, and A. C. Long. 2011. "Modeling and Simulating Textile Structures using TexGen," *Advanced Materials Research* 331:44-47.

4. Guido van Rossum. The Python Programming Language. `http://python.org` Accessed June 16, 2017.

5. Puri, G. 2011. *Python scripts for Abaqus: Learn by example*. `http://www.abaquspython.com/` Accessed June 16, 2017.

6. Piegl, L., and W. Tiller. 2012. *The NURBS book*. Springer Science & Business Media.

7. Pham, B. 1992. "Offset curves and surfaces: a brief survey," *Computer-Aided Design*, 24(4):223-229.

8. Dassault Systemes, 3D ACIS Modeler `https://www.spatial.com/products/3d-acis-modeling` Accessed June 16, 2017.