Computer-Aided Design and Applications

http://www.cadanda.com

A CAD Independent Approach to Automate Laminate Composite Design and Analysis

Ammon I. Hepworth[1], C. Greg Jensen[2] and James T. Roach[3]

[1]Brigham Young University, ammon.hepworth@gmail.com
[2]Brigham Young University, cjensen@byu.edu
[3]Pratt & Whitney, james.roach@pw.utc.com

ABSTRACT

The design of non-planar laminate composite parts is a difficult and time consuming process. To streamline the process, a composite design automation program was written that uses CAD independent methods available in GSNLib. This program utilizes the speed and portability of CAD independent algorithms while incorporating the benefits of visualization, simplicity and data storage of a CAD centric approach. The program takes the outer composite surface definition, translates it into its NURBS surface representation, automatically creates the ply lay-up and translates it into the CAD system for visualization. In addition, it automatically pre-processes a finite element model of the composite lay-up for engineering analysis.

Keywords: laminate composite design, automation, CAD independent methods.
DOI: 10.3722/cadaps.2009.147-156

1. INTRODUCTION

For thousands of years people have been combining two or more materials together to form a heterogeneous material that has improved properties over the original material. One example is combining mud and straw to form an adobe brick. This was an early form of a composite material. Today composites have become so widespread that one can find them in almost any industry, from bathtubs to fighter jets. Composite materials have also become quite sophisticated. Generally, these advanced composite material systems involve using a high-modulus fiber imbedded into a much lower modulus matrix material. Several architectural choices may be made, including chopped fiber, braided, three-dimensionally woven and two-dimensional laminates. Specifically, the two-dimensional laminate architecture is formed by stacking several layers of fiber dense sheets or laminates together to build up the 3-dimensional part. While this process allows designers control over the material properties of the part, the design and manufacturing of complex parts using this lay-up process can often be difficult and time consuming [7].

One method to automate composite design was developed by Vasey-Glandon et al. They patented a computer implemented system for generating an optimized 3-D ply definition for a laminate composite part called the Parametric Composite Knowledge System (PACKS). The system provides means for viewing the 3-D ply definition and the tabular data corresponding to each ply. The system optimizes plies using FEA analysis software and predefined rules for laminate design and manufacturing [8]. This system has been successful in reducing the cost of many aircraft composite development programs by over 60% [3].

Another method, developed by Menayo et al., creates a preliminary ply CAD model of a composite part. A stacking sequence for the lay-up comes from a stress analysis program called ARPA. Their method generates a ply stacking table and grid table to organize the plies into zones. The stacking table is then optimized to reduce the number of total plies in each zone. These plies are automatically staggered following design rules to order the plies in a given zone. The final step is the output of a ply table containing the necessary information to create a ply model in a CAD system [4].

Application programming interfaces (API) are a way to programmatically interface with commercial software applications. Most commercial CAD systems have APIs and they are often used in industry to automate design processes. A method to automate laminate composite design within a commercial CAD system provides several advantages:

- Provides a simple way to visualize geometry
- Simple to adopt for companies with existing CAD system in place
- Simplifies data storage and transfer

Despite the advantages, there are two major weaknesses to the CAD-centric approach. First, many CAD APIs exist a level or two above the geometry kernel and are therefore are relatively slow in performing geometrical computation when compared with CAD independent methods. This makes them impractical to use when large amounts of computation are necessary. Second, using a CAD API limits the automation tool to be utilized within a single CAD system. Since data exchange between CAD systems is difficult, limiting any proprietary method or design tool to a specific CAD system can be a major problem because many large corporations today, on a division by division basis, use more than one commercial CAD system. Therefore this keeps the divisions within the same company using different CAD systems from using the design tool unless it is rewritten using the specific API for their CAD system.

In 2003, Astle developed a way to create custom geometry applications that are independent of any one CAD system. He successfully applied CAD independent algorithms in flank milling automation and integrated it into NX using NX's API. He concluded that it was trivial to connect this application to another CAD system, but did not successfully integrate it with another CAD program. The advantage of CAD independence was that it allowed the application to be portable and quickly integrated with any CAD system possessing an API [1].

The method described in this paper is similar to the method Astle applied to flank milling. It applies this CAD independent approach to the automation of laminate composite part design and analysis while still utilizing the benefits of a CAD centric approach. The advantage to doing this is that it utilizes the speed and portability of the CAD independent algorithms while incorporating the benefits of visualization, simplicity and data storage of a CAD centric approach.

2. METHOD

The method takes geometry from a user familiar source like a commercial CAD system and translates it into the mathematical representation of that surface. It is then stored in CAD independent data structures. CAD independent algorithms are applied to create new geometry and analysis data. Finally, the new mathematical surfaces are translated back into user familiar data for visualization and analysis (See Figure 1).

A non uniform rational B-spline (NURBS) is the mathematical surface representation used as the CAD independent data storage in this method. NURBS surface representation was chosen for this method for several reasons. NURBS's are a common parametric curve and surface representation used within many of the leading commercial CAD systems. Complex surfaces can be represented with relatively small amounts of data using NURBS. Also, since NURBS is parametric in nature, it has the advantage of being able to quickly compute the coordinates of any point on a surface, making the parsing of a surface quick and easy [6]. In addition, many NURBS based, CAD independent algorithms have been published, making it simple to implement CAD independent algorithms that replace those based within

the CAD system. For more information regarding the basic mathematics of NURBS curves and surfaces see [5] and [6].

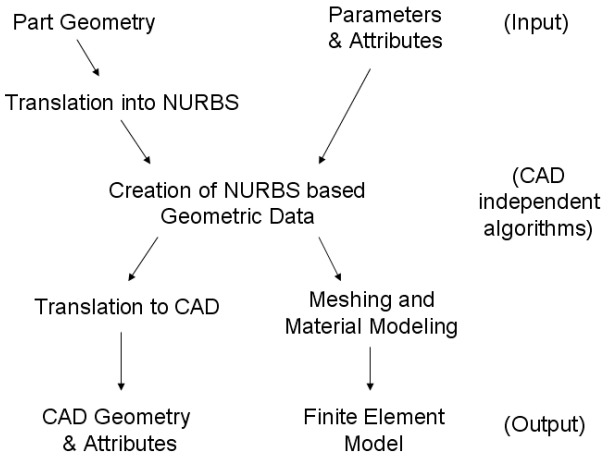

Fig. 1: Method flow chart.

2.1 Input Data from a user Familiar Source

Starting with a CAD model, a CAD API function is used to prompt a user to select l number of surfaces from within the model. As the user selects each surface, they are stored as CAD based data structures. Another API function is used to extract the NURBS surface data from the CAD based surface. This data is then transferred to NURBS based, CAD independent data structures. Mathematically this can be shown as follows:

$$\sum_{h=0}^{l}\sum_{i=0}^{n}\sum_{j=0}^{m} w_{h,i,j}P_{h,i,j}\Big|_{CADIndep} = \sum_{h=0}^{l}\sum_{i=0}^{n}\sum_{j=0}^{m} w_{h,i,j}P_{h,i,j}\Big|_{CAD} \tag{2.1}$$

$$\sum_{h=0}^{l} U_h\Big|_{CADIndep} = \sum_{h=0}^{l} U_h\Big|_{CAD} \tag{2.2}$$

$$\sum_{h=0}^{l} V_h\Big|_{CADIndep} = \sum_{h=0}^{l} V_h\Big|_{CAD} \tag{2.3}$$

Attributes and parameters that define the new surfaces to be created are predefined by the user in a spreadsheet program such as Microsoft Excel. These parameters are programmatically queried and stored within the CAD independent data structures to be used by the program for geometry creation.

2.2 CAD Independent Algorithms

New NURBS surfaces are created based on the geometry that came originally from the CAD system and the parameters that were read in from the spreadsheet. These are created using CAD independent algorithms. One of the most common of these algorithms is that of offsetting a surface.

An offset surface is created mathematically as follows. The original NURBS surface $\{S(u,v)\}$ is queried for a grid of points $\{G_{i,j}\}$ with a given tolerance $\{t\}$ in *u* and *v*, which lie on that surface.

$$G_{i,j} = \sum_{i=0}^{n}\sum_{j=0}^{n} S(i/n, j/n) \tag{2.4}$$

Note that $n = 1/t - 1$ and *t* is defined such that its inverse must be an integer value. The equation of the unitized surface normal vector $\{\hat{n}(u,v)\}$ at a given *u* and *v* can be found using the traditional formula:

$$\hat{n}(u,v) = \frac{\dfrac{\partial S}{\partial u} \times \dfrac{\partial S}{\partial v}}{\left\| \dfrac{\partial S}{\partial u} \times \dfrac{\partial S}{\partial v} \right\|} \tag{2.5}$$

Where "$\times$" denotes the cross product [2].

The normal vectors $\{H_{i,j}\}$ are found at each parameter value along u and v with a same specified tolerance that was used in finding the points:

$$H_{i,j} = \sum_{i=0}^{n} \sum_{j=0}^{n} \hat{n}(i/n, j/n) \tag{2.6}$$

The grid of points $\{K_{i,j}\}$ that lie on the offset surface that is offset by distance $\{d\}$ are found using equation 2.7 [9]:

$$K_{i,j} = \sum_{i=0}^{n} \sum_{j=0}^{n} G_{i,j} + H_{i,j} \cdot d \tag{2.7}$$

Once a grid of points on the offset surface is found, the offset surface is created utilizing a interpolating function to find the control points and knot vector for the new surface.

Another algorithm that should be noted here is that of surface trimming. Due to the complexity of the trimming algorithm, it will not be described in this paper. This algorithm is used to trim the offset surfaces so that they do not overlap any of the other offset surfaces that are created.

After performing CAD independent algorithms to create offset surfaces and trim them to the appropriate size, they are then stored in CAD independent data structures. Next, a mesh is generated to be used for finite element analysis. The meshing algorithm written makes 2D quad elements specified by a number of nodes along the u and v directions on a mid-surface of the composite part. This algorithm takes advantage of the ease of parsing NURBS surfaces. The nodes that make up the mesh are found by querying a NURBS surface $\{S(u,v)\}$ for a grid of node points on the mid-surface $\{D_{i,j}\}$ with a specified number of nodes in u $\{a\}$ and a specified number of nodes in v $\{b\}$ which lie on that surface:

$$D_{i,j} = \sum_{i=0}^{a} \sum_{j=0}^{b} S(i/(a-1), j/(b-1)) \tag{2.8}$$

Quadrilateral shell elements are created by specifying which four nodes are associated with each element. Once they are found they are stored in a 4-D row vector $\{E_{i,j}\}$:

$$E_k = \sum_{k=0}^{(a-1)(b-1)} \begin{bmatrix} e_{1k} \\ e_{2k} \\ e_{3k} \\ e_{4k} \end{bmatrix} = \sum_{i=0}^{a-1} \sum_{j=0}^{b-1} \begin{bmatrix} D_{i,j} \\ D_{i+1,j} \\ D_{i+1,j+1} \\ D_{i,j+1} \end{bmatrix} \tag{2.9}$$

2.3 Output Data into a Useable Source

The NURBS surface data is translated back into CAD centric data for the user to visualize. This is done by looping through all of the surfaces that were created and stored in the CAD independent data structures and querying their control points, weights and knot vectors. This data is taken and put into a CAD API function that takes NURBS surface parameters as input and outputs a surface body to the CAD system. Mathematically this can be shown as follows:

$$\sum_{h=0}^{l} \sum_{i=0}^{n} \sum_{j=0}^{m} w_{h,i,j} P_{h,i,j}{}_{CAD} = \sum_{h=0}^{l} \sum_{i=0}^{n} \sum_{j=0}^{m} w_{h,i,j} P_{h,i,j}{}_{CADIndep} \tag{2.10}$$

$$\sum_{h=0}^{l} U_h \Big|_{CAD} = \sum_{h=0}^{l} U_h \Big|_{CADIndep} \tag{2.11}$$

$$\sum_{h=0}^{l} V_h \Big|_{CAD} = \sum_{h=0}^{l} V_h \Big|_{CADIndep} \tag{2.12}$$

Non geometric attributes are translated to a CAD specific format by using a CAD API function that associates the attributes with the CAD bodies created. Finite element input files are written containing the mesh that was created using the CAD independent algorithms. The material properties as well as other element specific properties (such as thickness) are organized into the finite element input file.

3. IMPLEMENTATION

The methodology discussed above was implemented in the automation of laminate composite design. A computer program was written that used this methodology to automatically create a ply definition within a CAD program given an outer surface definition of a composite part and an excel spreadsheet that defines the parameters of the ply definition. The methodology is also utilized to create a finite element model containing the necessary information to be used for engineering analysis of the composite part.

The commercial CAD systems used for this implementation were Siemens – NX and Dassault Systemes – CATIA. The CAD independent software used is the General Surface NURBS Library (GSNLib). This is a software toolkit distributed by Solid Modeling Solutions Inc. that provides methods for creation, storage and manipulation of NURBS surfaces [8]. This toolkit lends itself to be a good option for implementing this method with NX and CATIA because GSNLib and CATIA and NX's APIs are all based in the C/C++ programming language. This allowed the application to be integrated along with both CAD systems all within a single programming language. The API for the NX programming is called NX Open, a C based programming library that has several functions that allow for the creation and manipulation of NX geometry. The API used for the CATIA programming is called CAA RADE, a C++ based programming library that allows programmatic control of all interactive functions in the CAD system.

The program is written within the framework of the CAD API and run from within the interactive CAD system. Two separate pieces of code for NX and CATIA need to be written for the translation to and from the specified CAD system. However, both NX and CATIA share the same base CAD independent code which is written in C/C++. The code written utilizes many GSNLib function calls to create the NURBS surfaces that represent the composite ply definition and to parse through NURBS surfaces in order to define the mesh for the finite element model. The only prerequisite to running the program within NX or CATIA is that a user has an outer surface definition of the composite part defined.

Figure 2 shows a graphical representation of the approach of this program. It shows that the program is built upon the foundation of GSNLib and uses the CAD API framework for an interface to go between the interactive CAD system and the GSNLib algorithms. The commercial CAD system is used for user interaction, the API framework is used to translate data to and from the CAD independent algorithms and the GSNLib functions are used to create geometric and analytical data.

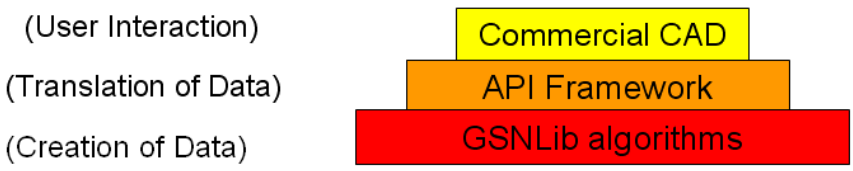

Fig. 2: A representation of the approach of the automation program.

3.1 Input Data from User Familiar Data

After running the program from within the CAD system, an API function is called in the program that prompts the user to select the surfaces that make up the outer shell of the composite part. This surface definition can be defined in the CAD system or as a set of surface points that make up the surface. Using another CAD API function call, the selected surfaces are queried for their NURBS surface control points, weights and knot vectors. These values are passed into a function that constructs an IwBSplineSurface object that is predefined in GSNLib. This object contains all the data to store a NURBS surface in CAD independent form.

Another call to a CAD API function prompts the user to select an Excel spreadsheet containing the ply parameters for the composite part. These parameters are ply thickness, fiber angle and ply material. Next, the user is prompted to select a spreadsheet that contains material properties for the materials used in the first spreadsheet. After selecting these documents they are opened and parsed programmatically to get out the data and store it in data structures within the program. If no such document exists for ply and material properties the user has the option to enter only the ply thickness into a GUI prompt to be stored. However, if this option is chosen, additional pre-processing will be required. After being prompted for the ply and material properties, the user is also prompted to input the number of elements in the u and v directions to be used to create a mesh for finite element analysis of the composite part.

3.2 CAD Independent Algorithms

The ply definition is created using the CAD independent functions provided in GSNLib. Using the parameters that were obtained from the Excel spreadsheet, surfaces are created by offsetting from the outer surface definition of the composite part. These surfaces are offset using the CAD independent methods discussed in the previous section describing the method. After the new surfaces are created, they are stored in IwBSplineSurface objects to await translation back into the CAD system. The parameters and attributes that were read in from the Excel spreadsheet are stored in a data structure vector that contains all the IwBSplineSurface objects and the attributes that are associated with them.

To create the information necessary to create a finite element analysis input file, a mesh is generated on the mid-surface, which is the surface that lies halfway between the outer two surfaces that make up the part definition. Meshing on the mid-surface allows the surface definition of the composite part to be represented as a shell model in the finite element analysis. The mid-surface is meshed using the algorithm discussed in the method section that creates nodes and elements. This mesh element size is based on the number of elements in u and v specified earlier by the user. The data that is associated with each element, such as the ply lay-up, ply thickness, fiber angle and material, is organized and stored in preparation to be output to a finite element input file.

3.3 Output Data into Useable Data

The surfaces that were stored in the IwBSplineSurface objects are translated back into CAD surfaces for visualization. In NX this is simply done using an NX Open function that takes as input the NURBS control points, weights and knot vectors and outputs a surface directly into NX. In CATIA, however, there are a couple more steps to the process. In CAA RADE there is a function that takes the same inputs as the NX Open function and outputs a geometrical NURBS object. This object can only be viewed in CATIA after it is converted into a skin and then into a datum feature. Last it must be added to the procedural view for actual visualization in CATIA.

The attributes that are associated with each IwBSplineSurface are translated into NX using a function that takes as input a string of text and numerical values and outputs an NX attribute that is associated with the NX surface. Since CATIA has no such “attribute” feature available to associate general attributes with surfaces, this step was not implemented in CATIA.

The finite element analysis input file is created programmatically from the nodes and elements as well as the ply materials and properties. Based on what the user selects as the analysis package to be output to, the data is organized in a way that conforms to that software’s input file format. Essentially

the program automatically makes a composite finite element model from a surface definition and ply lay-up specified in an Excel spreadsheet.

4. RESULTS

Laminate visualizations have been successfully created in NX and CATIA using the composite automation program described above. Figure 3 shows a simple test case of a ply definition successfully created in NX. The simplicity of this model comes from the fact that the outer surface definition only varies in two dimensions, being constant in the third.

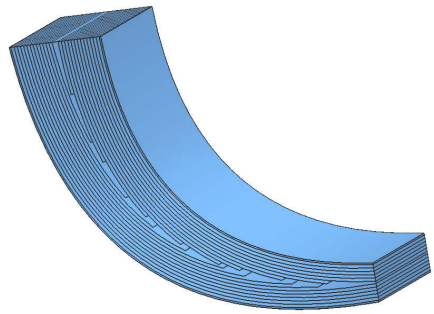

Fig. 3: Simple test case to showing ply definition created in NX.

The program was used to successfully create a more advanced ply definition for a composite vertical stabilizer for a model airplane given an outer surface definition in NX. Figure 4 shows the surface definition of the vertical stabilizer loaded in NX and Figure 5 shows the ply definition. This definition is much more difficult to create due to the fact that outer surface definition varies in all three dimensions. Table 1 shows the ply table created in Excel that defines the ply properties to create this ply definition.

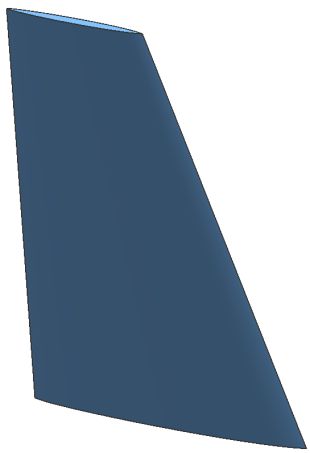

Fig. 4: Vertical stabilizer loaded in NX.

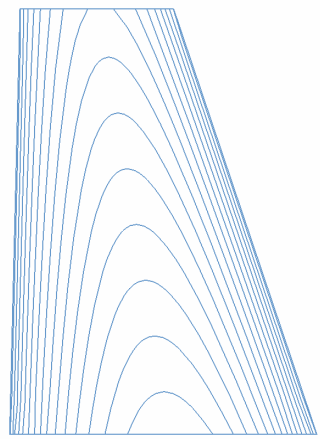

Fig. 5: Vertical stabilizer ply definition.

Ply Properties						
			(mm)	(degrees)		
Ply	Material ID	Pre-form	Thickness	Angle	Material	Weave
1	1	B	0.5	0	graphite	Plain
2	1	B	0.5	45	graphite	Plain
3	1	B	0.5	-45	graphite	Plain

4	1	B	0.5	90	graphite	Plain
5	1	B	0.5	0	graphite	Plain
6	1	B	0.5	45	graphite	Plain
7	1	B	0.5	-45	graphite	Plain
8	1	B	0.5	90	graphite	Plain
9	1	B	0.5	0	graphite	Plain
10	1	B	0.5	45	graphite	Plain
11	1	B	0.5	-45	graphite	Plain
12	1	B	0.5	90	graphite	Plain
13	1	B	0.5	0	graphite	Plain
14	1	B	0.5	45	graphite	Plain
15	1	B	0.5	-45	graphite	Plain
16	1	B	0.5	90	graphite	Plain

Tab. 1: The ply property table from MS Excel.

The program described above automatically creates a 2D quad mesh on the mid-surface of a composite part. It then associates each element with the plies that correspond to it, thus creating a finite element shell model of the composite lay-up. Input files containing a finite element model were successfully created for LSTC – LS-DYNA, ANSYS and MSC – NASTRAN to be used for finite element analysis. Figure 6 shows the mesh on the mid-surface of the simple ply definition that was automatically generated with 10 nodes in each direction. Figure 7 shows the mesh created for the vertical stabilizer with 30 nodes in each direction. Both Figures 6 and 7 are images of the mesh brought into Altair's finite element software Hyper Mesh for visualization.

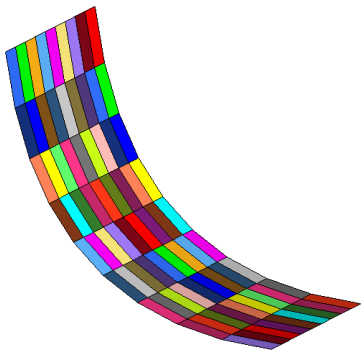

Fig. 6: Mesh for the simple ply definition.

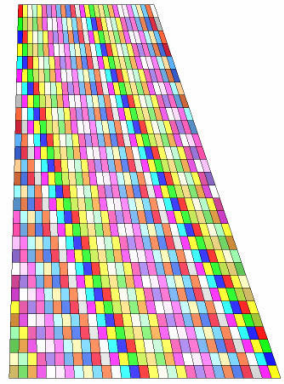

Fig. 7: Mesh for vertical stabilizer.

To compare the speed of a composite design automation program using the CAD independent method vs. using a CAD API, a function that is used frequently for the implementation of this method is tested for speed in NX Open, CAA RADE and GSNLIB. The function tested is one of the primary functions used in the automated laminate composite design program and is used hundreds of thousands of times throughout. Although this function may not represent the actual computational speed of the entire program, it does represent a large portion of the program, and therefore it is a good measure of the speed of the entire program. This function takes in u and v surface parameters as input and outputs the Cartesian point on the surface associated with it. The speed test performed calls the function a total of 1,000,000 times on each surface tested. The test calls the function to parse the surface making 1/1000 unit steps in the u direction and 1/1000 steps in the v direction. Mathematically this can be shown as parsing through the surface $\{S(u,v)\}$ to get a grid of points $\{G_{i,j}\}$:

$$G_{i,j} = \sum_{i=1}^{1000}\sum_{j=1}^{1000} S(i/1000, j/1000) \tag{3.1}$$

The speed is clocked for each surface test in each of the programming libraries. Table 2 shows the results of times recorded in NX Open, GSNLib and CAA RADE vs. GSNLib.

Surface	Time in NX Open	Time in GSNLIB	Time in CAA RADE
Surface 1	46.766 sec.	1.516 sec.	.375 sec.
Surface 2	47.953 sec.	1.516 sec.	.375 sec.
Surface 3	47.187 sec.	1.578 sec.	.390 sec.
Surface 4	47.719 sec.	1.516 sec.	.375 sec.

Tab. 2: Test results from speed comparison of NX, GSNLIB and CAA RADE.

When comparing NX Open to GSNLib, the test in GSNLib ran about 30 times faster than NX Open. This is dramatic increase in computational time using GSNLib over NX Open. The performance of GSNLib and CAA RADE, however, are on the same order of magnitude. This is because CAA RADE allows for direct access to CATIA's geometry kernel, where NX Open operates at a level or two above the geometry kernel.

The ply definition and analysis files created automatically using the automation program written takes a designer less than one minute to run. In contrast, it takes the designer an estimated 40 hours to create the same ply definition if done interactively in a CAD system and FEA package. Therefore, when using this automation program, companies that design laminate composite parts will save a significant amount of time and money. In addition, this program allows designers to evaluate several design configurations in the same time it would take to create a single design. This allows for superior designs to be created in less time.

Although this automation process could be programmed using NX Open, without utilizing the CAD independent functions in GSNLib, there are several reasons why using these functions in an automation program makes the program superior. First, the advantages of the CAD centric approach including simplified viewing and data storage/transfer are held intact. Second, the application is portable enough to "plug" into any commercial CAD system with an existing API. Third, using GSNLib functions allow for a 30 times speed increase over the NX Open API. This significant speed increase makes the process of optimizing the ply definition much more practical. Since an optimization routine would need to create hundreds of ply definitions to find an optimum, the NX Open process is too slow to be done in a reasonable amount of time. Using functions available in GSNLib increase the processing speed enough to make the optimization process feasible.

5. CONCLUSIONS

Applying a CAD independent approach to the automation of laminate composite part design has allowed for the application to be faster than a similar program written solely in the NX Open API. This can be proven by the fact that GSNLib ran a frequently used function approximately 30 times faster than the same function in NX Open. The reason for this is the fact that the NX Open API functions operate at a level or two above the geometry kernel and are therefore slower in performing geometrical computation when compared with GSNLib and CAA RADE which interface directly with the geometry definition. The CAD independent approach has also allowed the application to be portable enough to work within multiple CAD systems. This is proven by the fact that the same CAD independent code was used in both NX and CATIA for geometry creation, utilizing their individual CAD APIs merely for translational purposes. The benefits of visualization, simplicity and data storage of a CAD centric approach are kept intact with the CAD independent approach because the user runs the entire program from within the CAD system just as if the program were made specifically for it.

6. REFERENCES

[1] Astle, T. L.: System Architecture and Development of CAD Independent Algorithms for Integration with Commercial CAD Software, M.S. Thesis, Brigham Young University, Provo, UT, 2003.

[2] Farin, G.: Curves and Surfaces for Computer Aided Geometric Design, Academic Press, Inc., San Diego, CA, 1988.
[3] Hale, R; Schueler K.: Knowledge-Based Software Systems for Composite Design, Analysis and Manufacturing, Society of Automotive Engineers, 2001.
[4] Menayo, G.; Quero, A.: Computer-aided method of obtaining a ply model of a composite component, U.S. Patent App. 731,794, 2007.
[5] Piegl, L.; Tiller, W.: The NURBS Book, Springer-Verlag, Berlin, Heidelberg, New York, 1997.
[6] Rogers, D. F.: An Introduction to NURBS: with Historical Perspective, Academic Press, San Francisco, 2001.
[7] Sederberg, T.: BYU NURBS, http://cagd.cs.byu.edu/~557/text/ch1.pdf, 2007.
[8] Solid Modeling Solutions, http://www.smlib.com/gsnlib.html, 2009.
[9] Strong, A. B.: Fundamentals of Composites Manufacturing Materials, Methods, and Applications, Society of Manufacturing Engineers, USA, 2008.
[10] Vasey-Glandon, V. M.; Kunkee, D.: Knowledge driven composite design optimization process and system therefore, U.S. Patent No. 7,010,472, 2006.
[11] Zeid, I.: Mastering CAD/CAM, McGraw-Hill, New York, NY, 2005.