CAD-centric Creation and Optimization of a Gas Turbine Flowpath Module with Multiple Parameterizations

Damon Delap[1], Jeff Hogge[2] and C. Greg Jensen[3]

[1]NASA Glenn Research Center, damon.c.delap@nasa.gov
[2]Brigham Young University, jhogge@byu.edu
[3]Brigham Young University, cjensen@byu.edu

ABSTRACT

This paper presents CAD-centric approach to the integration of CAD, analysis and optimization. This is demonstrated through the development of a tool that allows a gas turbine flowpath to be created, analyzed and optimized by users who are not necessarily familiar with the CAD, analysis and/or optimization software. A key to being able to develop this type of design tool is the use of the CAD Application Program Interface (API). In the process of developing this tool, the following concepts are demonstrated: 1) how to overcome problems of data exchange between software packages and robust CAD models suitable for optimization, 2) how to dramatically reduce the learning curve associated with CAD, analysis and optimization software, and 3) how to employ more that one parameterization in the same model by utilizing the CAD API.

Keywords: Computer-Aided Design; parametric modeling; knowledge-based CAD; CAD-based optimization; gas turbine flowpath.

1. INTRODUCTION

In today's world passengers demand shorter travel times, quieter, vibration-free flights. Airlines demand more efficient planes with longer mean times between overhauls, and governments demand more pollution-free aircrafts. As a result, the commercial airframe and aeropropulsion manufacturers are being driven to be more aggressive in their design. For example, the propulsion manufacturers must achieve ever lower drag coefficients while improving lift/drag ratios, improve the power/weight ratios, increase engine horsepower, etc. all while making a greener, more cost efficient engine to operate.

A similar historical perspective could be given for all most any engineering/manufacturing company that has endured for this last century. Everyone agrees that intense global competition is causing today's companies to look for ways to increase productivity, improve quality and reduce costs. Many engineering tools have been developed to help achieve these objectives. Three tools that have had the greatest impact on engineering design include parametric/relational surface and solid modeling, parametric analyses and optimization. By integrating these tools into a single design application that is simple and easy to use, the advantages of each tool can be exploited.

The research presented in this paper takes a CAD-centric approach to the integration of CAD with analysis and optimization. The analysis and optimization are defined and initialized from within the CAD system. The focus of this integration was the development of a gas turbine design/analysis/optimization application that allows engineers with different areas of expertise and little experience with CAD and/or optimization software to create optimized engines and engine components.

Another problem with modern parametric CAD tools is their inability to support multiple parameterizations, i.e. what parameter scheme the designer chooses is generally not how the structures or flow analysts would parameterize the model. This paper also discusses how two parameterizations are embedded into a single CAD model, providing the first ever CAD tool to accommodate different modes of creating and manipulating the model.

While much of the focus of this paper is on the development of a turbine flowpath design tool the authors believe many of the integration, automation and optimization techniques can be generally applied. This paper has particular

value to those doing concept initiation and optimization work in the conceptual and preliminary design stages of a new product. Specifically, this paper discuss the integration of TURBAN (1-D flow analysis code developed at NASA's Glenn Research Center) and iSIGHT (a commercial optimization code from Engineous) to interoperate within the NX system from UGS. Design, analysis and automation engineers will benefit from the integration and automation principles presented in this paper.

2. BACKGROUND

The integration of CAD and optimization presents two main problems for designers. First, having two or more programs exchanging information in a useful manner is difficult and must be done reliably [2]. Second, in order for a CAD model to be used in an optimization loop, it must be very robust. Optimization routines tend to find and exploit weaknesses in a model, resulting in infeasible designs and/or failure of the entire process, thus wasting valuable computation time and human effort [6]. Rohl et al. sites this as one of the major stumbling blocks associated with using CAD and optimization [8]. Another roadblock encountered by designers is the learning curve associated with either CAD or optimization. An experienced CAD user can build complex parametric models that are very robust and accommodate minute details. However, this level of skill can take years of training and experience to acquire. The optimization of CAD models is also very complex and takes extensive training to execute correctly. Immense benefits occur if the learning curve that exists with producing and optimizing parametric models can be compressed or eliminated. This can be done most effectively when focused on specific design applications.

The newest method of representing a solid model is feature-based/parametric. Parametric CAD was introduced by Parametric Technology Corporation when they released Pro/Engineer in the late 1980's [7], and it became the scheme that all major CAD systems followed from then on [5]. Parametric representation has many advantages over previous representation schemes. The designer is not limited to a set of primitives in the solid modeling system, and features can be built with higher-level representations of curves and surfaces. These entities can be expanded into more accurate versions of actual parts, and in turn yield better analyses. In addition, the ability of the model to be modified by changing constraints allows parametric models to be used in automated optimization loops. The advantages of feature-based/parametric modeling can be used to produce a robust CAD model that can then be manipulated with optimization software.

3. METHOD

The method presented here for the development of a CAD-centric design/analysis/optimization application requires several steps, also see Fig. 1:

1. Development of an object-oriented data structure
2. Creation of a programmatic CAD model
3. Design of a user-friendly CAD interface
    a) Collect inputs, instantiate the model and check its validity
    b) Preprocess the model for analysis
    c) Initiate the solver
    d) Execute an optimization run
4. Modification and incorporation of the analysis codes to run from inside the CAD system
5. Creation an intuitive optimization process that runs within the CAD environment

These steps lead to successful CAD-centric applications that essentially remove the burden of modeling, analysis and optimization from the interactive designer's hands. It captures the knowledge, rules and behaviors of CAD, analysis and optimization experts. It is their combined expertise captured within the code that fuel the applications and provide the competitive edge over their rival companies.

The remainder of this article takes the reader through the process of applying these steps to the creation of a gas turbine flowpath design application (see Fig. 2).

3.1 Dynamic Object-oriented Data Structure

In the development of a CAD-centric design application one must first create a dynamic object-oriented data structure (DOODS). The main purpose of DOODS is to isolate the CAD application program interface (API) operations in order for the program to be highly modular between CAD systems. It also provides a means for dividing the application into

smaller classes or objects see Fig. 3. DOODS should also contain all of the mathematical mappings that handle the various parameterization schema. Ultimately, DOODS contains all key data that is common to all parameterizations.

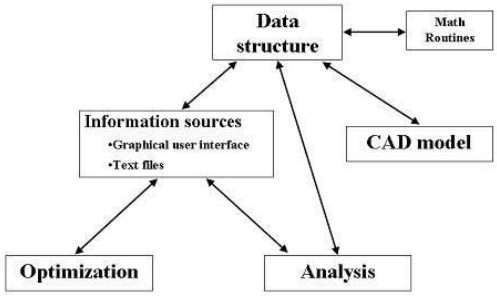

Fig. 1. Method Schematic.

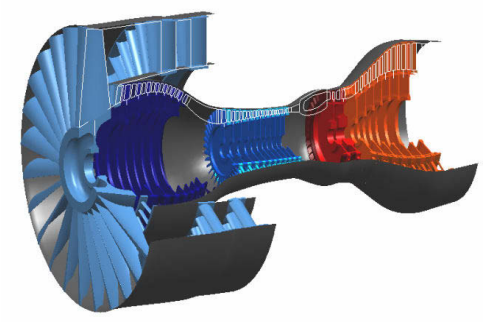

Fig. 2. Gas Turbine Engine Flowpath.

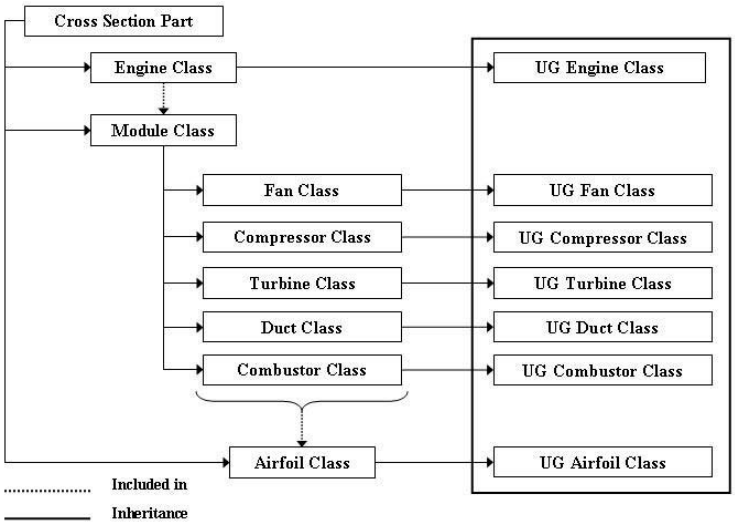

Fig. 3. DOODS Structure.

3.2 The Programmatic Model

A programmatic model that is fully parametric must be created. The parameterization that dictates how the model will be manipulated must be determined. Inputs must be determined and named based on each parameterization. Next additional routines that interface DOODS with the programmatic model are required. These routines should be built with the CAD system's API and perform the following functions: 1) Fill and update DOODS with user input, 2) Fill and update DOODS from all other possible sources of information, 3) Create, name, assemble and save parts, 4) Extract information from the assembly and update DOODS, 4) Extract information from DOODS and update the assembly.

3.3 Graphical User Interface for Model Creation

On CAD-centric applications an intuitive graphical user interface (GUI) must be created that will systematically take the user through the standard work process to create the optimal design. This interface not only collects the needed parameters for the model but also collects all of the inputs for the analysis and optimization. Fig. 4 shows just one of many dialog boxes and toolbars that were created for the turbine engine application. This dialog box was created in the UI/styler (a module within Unigraphics). Greater modularity could have been achieved by using a CAD independent GUI generator such as Visual Basic or QT.

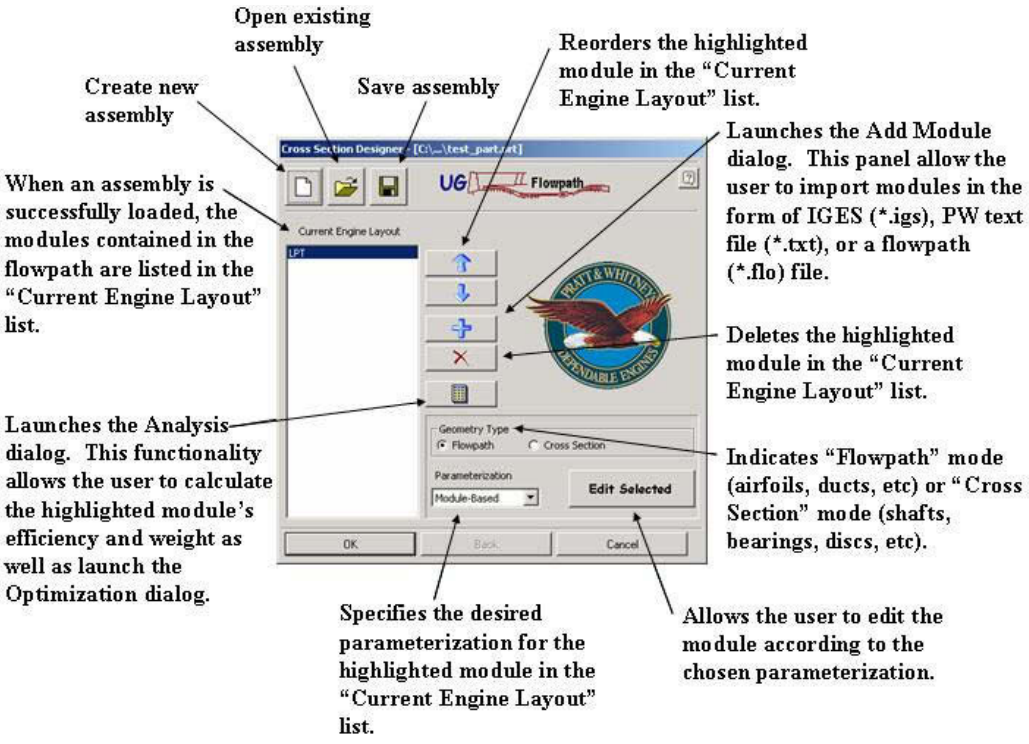

Fig. 4. Cross Section Designer GUI.

In the creation of a user friendly GUI it is important to follow a logical standardized sequencing of commands and options, only presenting the user with valid options. This will greatly aid in creating a robust reusable design/analysis/optimization application. It must be ensured that a good callback structure is in place to read, write, manage, etc. the parameter and topology being created. Help menus, standard work documents and sample inputs should be accessible from the GUI. Incorrect inputs should also be rejected with clear error messages. Once a complete set of inputs are entered the GUI must automatically begin the creation and checking of all topology to ensure a valid parametric model.

In general, the GUI should contains several paths that the user (depending who he/she is) may go down. In the case of the turbine flowpath GUI, its end product is a fully constrained and completely parametric gas turbine flowpath that has been created using one of four different parameterizations. The user still has the opportunity to go back and alter the flowpath through the GUI according to any one of the parameterizations that are available, or to analyze and/or optimize a module.

3.4 CAD-centric Analysis

Many CAD systems provide some level of CAD-centric analysis, NX provides the Scenario application, CATIA has ELFINI built in, and Pro/E has Pro/Mechanica. While these tightly integrated programs provide a single foray into the analysis space, their vendors do not provide companies with an easy interface to bring other commercial and proprietary CAE codes into a CAD-centric paradigm. This paper presents an example of how other CAE tools can be made CAD-centric compliant and can be run either in an interactive or batch format.

The GUI of the design application must be extended to collect all of the preprocessing input required to mesh, load, constrain and define all of the material properties for the FEA model. Just as the GUI for the geometry creation this interface must also be supported with good help and documentation files. Finally, the CAD GUI must allow the user to submit the analysis model to the solver and wait for results. When the results are returned, the user can manipulate, via the GUI, any and all (geometry and/or analysis) parameters and rerun the job. With a successful analysis complete, the user is prepared to move to the optimization GUI.

In the case of the turbine flowpath application the interactive and batch execution are slightly different, i.e iSIGHT requires the reading of parameters from ASCII text files, as shown in Fig. 5.

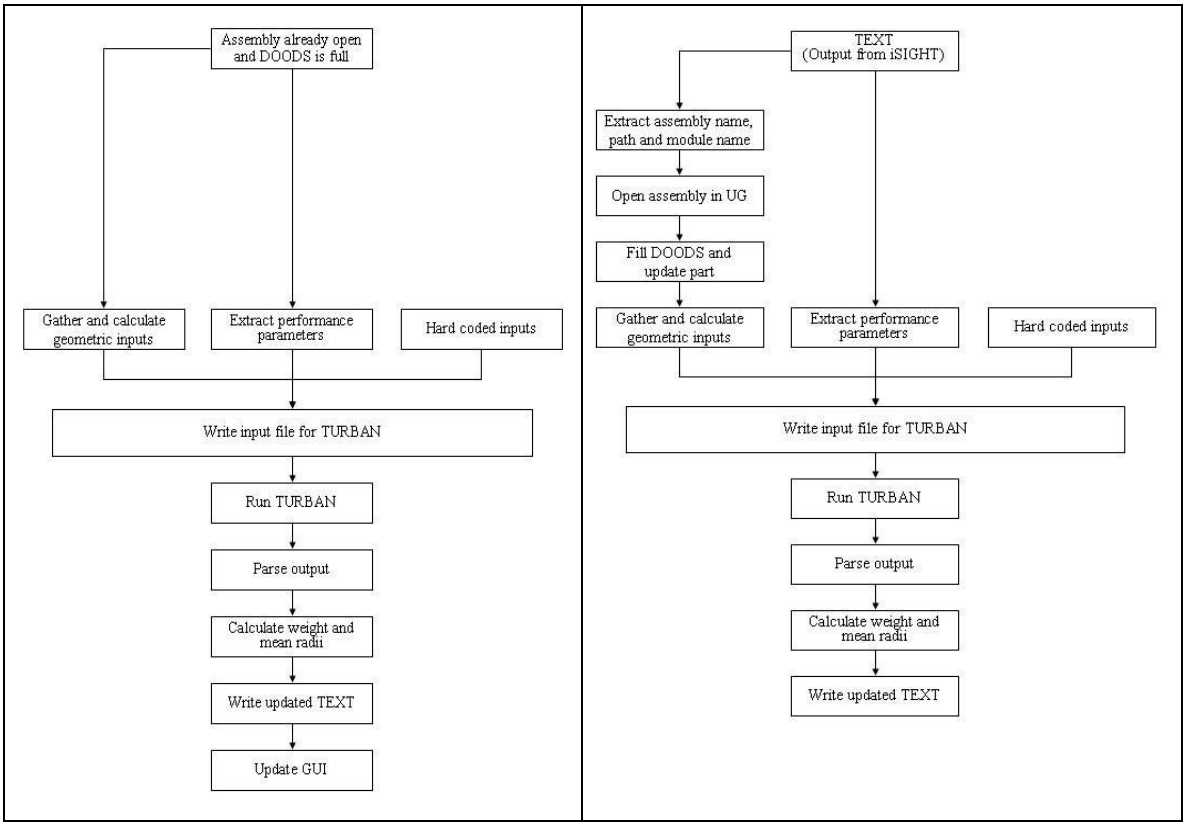

Fig. 5. Flowcharts for Interactive and Batch Analysis.

3.5 CAD-centric Optimization

Commercial CAD systems lack the ability to run an optimization loop that incorporates codes and applications that are not *made* or *linked* within their *.dll* libraries. Some CAD vendors have put simple optimization routines inside their knowledge or mass properties applications, but have not provided any toolbar or dialog interface for a more universal optimization engine.

Once a valid model and analysis has been created, the optimization dialog should become accessible. This dialog should prompt the user the type of optimization algorithm to run as well as gathering all of the optimization inputs, i.e., objective functions, number of iterations, population size, etc. By extending the modeling and analysis GUI to include optimization the user does not need to learn another application interface, such as iSIGHT's. The GUI simply collects all of the required optimization inputs and starts the loop running in batch mode with the first model and analysis being the optimization seed.

The turbine flowpath application is designed in such a way that iSIGHT is launched from inside NX and the designer or analyst are able to change the model parameters, modify the analysis model, resolve and start a second optimization loop without ever leaving the CAD application. This functionality and CAD-centric methodology in general provides all users with the ability to model, analysis and do optimization studies regardless of their knowledge of the underlying codes. A graphical representation of the turbine flowpath application is shown in Fig. 6.

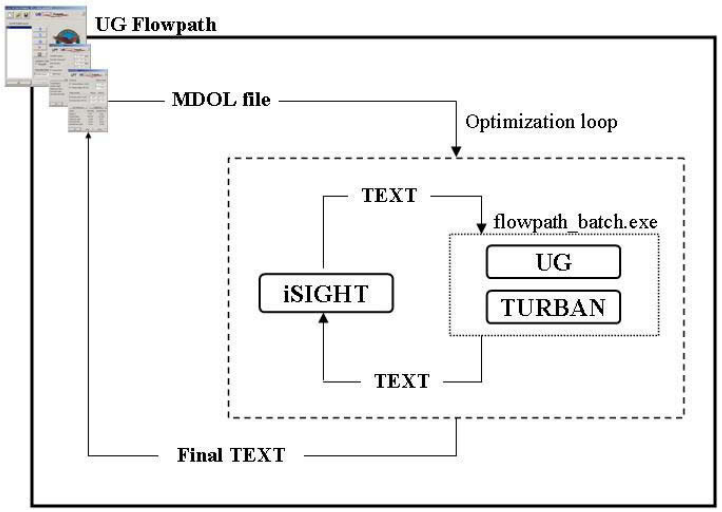

Fig. 6. Optimization Integration Process.

4. TURBINE FLOWPATH APPLICATION

The method described above was fully implemented for the low pressure turbine (LPT) and high pressure turbine (HPT) modules of a flowpath using the designed application, called UG Flowpath. This section discusses DOODS development and the basic functionality of UG Flowpath. It also describes how two parameterizations were included in the same model, the integration of the analysis, and the implementation of the optimization software for the turbine section of the flowpath.

4.1 Dynamic Object-oriented Data Structure Development

The development of DOODS will be described in this section because it is central to any operation performed on the CAD model. DOODS stores all the functions and variables for the entire engine, and the model cannot be modified without going through DOODS while UG Flowpath is running. DOODS is composed of classes that relate to the modules found in the engine. The classes can be independent or interdependent. If at any time another type of engine module needs to be added to DOODS, a new class can be defined and simply included in the structure.

DOODS contains a set of parallel classes, which can be visualized in two columns (see Fig. 3). The classes in the left column contain pure C++ function calls and are free from any UG API calls. The classes on the right contain all the UG API calls which interact with the UG model. This type of arrangement was used to isolate the CAD API calls from the rest of DOODS so that linking the program with other CAD packages in the future will be much easier. Theoretically, in order to use DOODS with another CAD package, the UG classes could be replaced with classes which contain API from any other CAD package. This separation took a little more work up front, but it significantly reduces the work that is needed to utilize additional CAD packages in the future.

In addition to the geometric classes mentioned above, there are other classes that are essentially a group of data manipulation functions. These classes provide the necessary parameterization mapping and translation capabilities. The two parameterizations we will discuss are mod-based and row-based. The parameterizations' variables and relationships are illustrated in Figs. 7 and 8.

The mod-based parameterization allows the user to manipulate the module on a global level by moving the control points of B-splines, which represent the flowpath walls. These curves will be referred to as the controlling B-splines. The airfoil corner points are constrained to be on the controlling B-splines, and the airfoil taper and chord are held constant during the modifications. The advantage of mod-based is that few parameters govern the shape of the module, namely the control points of controlling the B-splines. The user can quickly define radically different module shapes.

The row-based parameterization allows the user to modify the corner points of the blades and the airfoil properties of both the blades and the vanes. The position of the vane corner points are derived from the blade corner points by mathematical relationships. The corner points of the blades can be moved in the vertical direction, thus setting the radial location (defined as the perpendicular distance from the engine centerline). Changing the properties of the individual airfoils controls the axial location of the corner points (defined as the direction parallel to the engine centerline). The airfoil properties that may be modified are: axial chord, taper (b/a), cant angle, gap (immediately following the airfoil), and mid-gap ratio (the ratio of the gap that the black circles extends behind the airfoils in Fig. 8).

Translation between the two parameterizations is based on the common information that is available in both, i.e. the airfoil corner points. Based on the direction of the translation, different math routines and geometric relationships are used to establish the new desired behavior of the model.

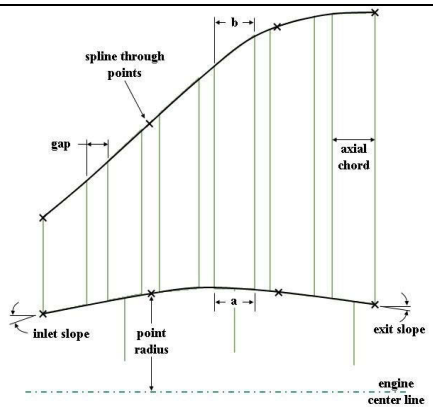

Fig. 7. Mod-based Parameterization.

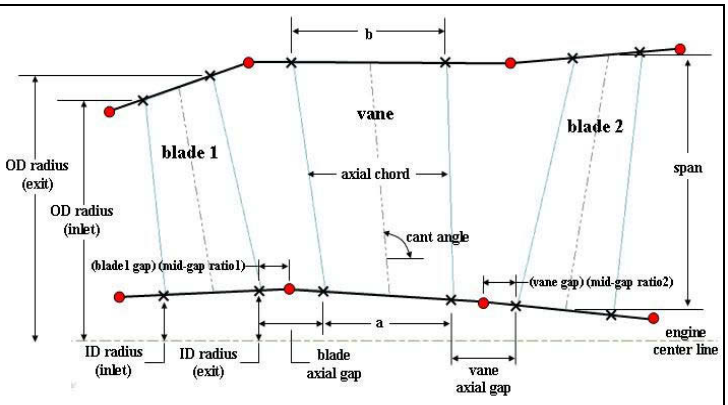

Fig. 8. Row-based Parameterization.

4.2 Analysis Integration

The integration of analysis into the UG Flowpath adds the ability to evaluate a flowpath module with TURBAN. TURBAN is an axial flow turbine analysis program developed by NASA's Glenn Research Center. It is described as, "a meanline design code, based on simplifying assumptions that limit the generality of the analysis but result in a very rapid calculation" [4]. For a complete discussion on TURBAN's method of analysis and capabilities, see [3]. TURBAN was chosen for this research because it is a fast, robust code that can calculate efficiency of a module based on changing geometry (inlet and exit radii, exit radius ratio). It also outputs the number of airfoils which, together with the radii of the module, is used to calculate the weight. This section describes the steps taken to integrate TURBAN into UG Flowpath.

Two different, yet very similar, versions of the analysis were developed. One version is called interactively by the user from within the GUI. The other version is called as part of the optimization loop, or in batch mode.

One of the main differences between the two versions is the source of input. The necessary inputs for TURBAN were broken down into the three categories: 1) inputs that are hard coded into the program, 2) inputs that can be gathered from DOODS, and 3) inputs that define the operating conditions of the module (performance parameters). The sources of the first two kinds of inputs remain the same regardless of whether the analysis is run from the GUI or in batch mode. The source of the third type of input is the GUI for interactive execution and TEXT for batch execution.

The inputs that are hard coded into the program are values that can be considered constant from one analysis to the next. The inputs that have only two values, one for LPTs and one for HPTs, are grouped in this category. The values for these inputs came from consulting the documentation for TURBAN.

The geometrical inputs are extracted from information contained in DOODS. If the interactive version of the analysis is being used, DOODS is already full and the information is readily available. If the batch mode of the analysis is being used, additional steps are needed in order to fill DOODS (see Fig. 5). Opening TEXT is the first operation that the batch version of the analysis does. TEXT includes, among other information, the name and path of the assembly which contains the module that is being analyzed. It also has the name of the target module and new geometric values which have been changed by the optimizer. Next, UG is opened (without its GUI), the assembly is loaded, and the routines which query the part's information and fill DOODS are called. The part is then updated so that the geometry reflects the changes from iSIGHT. At this point the analysis continues the same for both interactive and batch use, with the exception of the source of the performance parameters.

All the geometric inputs are extracted from information in DOODS. Note that these inputs could also be extracted from the CAD model, but in keeping with the notion of CAD isolation, the data contained in DOODS is used. In total, there are seven geometric-related inputs that had to be calculated. They are: mean radii of the inlet and exit, exit radius ratio, number of stages, meanline shape, stage number where the meanline changes slope (NMID), and the inlet flow angle (from the axial direction). Most of these inputs are fairly simple to calculate, but a geometry analysis algorithm had to be developed to calculate the meanline shape and to determine NMID. For a complete explanation of this algorithm, see [1].

The inputs from the third group, the performance parameters, can come from the GUI or TEXT. There are only five values required in this set: inlet pressure, inlet temperature, mass flow rate, RPM, and either the desired pressure ratio of the module or the shaft power. These values dictate the operating conditions of the module as well as the performance that is desired. If these inputs are to be gathered from the GUI, simple calls to the GUI are made that ask for the values of the input fields. If the inputs need to be gathered from TEXT, there is a slightly different path that needs to be taken. As described above, the first thing that the batch version does is to open TEXT. The five performance parameters are some of the other important information in TEXT. They are parsed out of the file and placed into the arrays of information that are formatted for TURBAN.

Upon gathering all the inputs from DOODS and either the GUI or TEXT, they are placed into arrays and passed to the routine that formats the information for TURBAN. This routine simply writes out another text file that serves as the input to TURBAN and puts the appropriate values next to key abbreviations that TURBAN recognizes. When the analysis finishes, TURBAN outputs a file of results. This file needs to be parsed in order to extract the desired results: efficiency, work factor, number of airfoils in each stage, exit radius ratio, and the hub and tip radii of the inlet and exit. The mean inlet and exit radii are calculated using the hub and tip radii as is the average mean radius for the module. The weight of the module is a rough calculation based on the number of stages, the total number of airfoils and the average mean radii of the inlet and exit. The work factor is used by the optimization as a constraint. At this point, all the results are calculated. For both interactive and batch use an updated TEXT is created, and for the interactive version only, the results are output to the GUI.

The process of integrating TURBAN into UG Flowpath resulted in two versions of the analysis portion of the research. The interactive version was compiled together with the GUI code to produce an internally executable program (with UG open). In order to make the batch version, the modifications stated in this section were made to the code, and an externally executable (with UG closed) program was compiled. This program is what is used by iSIGHT in the optimization loop.

4.3 Optimization Integration

The way in which the optimization was integrated into the design application allows a person who may not have experience with iSIGHT to perform an optimization on the flowpath portion of a turbine module. The goal was to link the optimization software to UG in a CAD-centric way. This allows the user to set up the optimization without having to go through the time and trouble of doing it in the iSIGHT GUI. During the optimization, the user can update the geometry with the information from the latest iteration. When the optimization is complete the UG model can be updated to reflect the optimized flowpath geometry.

The launching of iSIGHT and configuring of the optimization turned out to be surprisingly simple. iSIGHT can be launched with a command line. When iSIGHT reads in a text file, such as the files generated by DOOPT, the file is first

run through a "translator." This allows the use of an English based, command type file instead of a file containing cryptic machine code.

The optimization task plan in iSIGHT was performed in two steps. Originally, two gradient-based algorithms were used alone because these are the fastest and most robust algorithms. These algorithms were not very effective because of the characteristics of the design space. An optimization task plan with two steps was employed to overcome two possible deficiencies on the design space. The reader should note that additional data, such as plots of the design space, are needed to conclude which deficiency is causing the failure of the gradient-based algorithms. The first possible problem is the noise in the objective functions, where the algorithm terminates prematurely because this noise causes errors in the gradient calculations. The second possible problem is that the design space contains many local optima. This would cause the algorithm to terminate because it found an optimum; however, it may or may not be the global optimum.

The first step of the task plan used a genetic algorithm (GA) and the second step was a sequential quadratic programming (SQP) algorithm. The GA is substantially slower, but it is effective for noisy design spaces and spaces which may contain many local minima. The GA is used to survey the design space and find the region of the global optimum. Then SQP is used to quickly find the optimal design.

As part of the iteration of the optimization cycle, an output file is generated with changed values for the design variables. This is just an updated version of TEXT. At the end of the optimization, this file contains the information for the design to which the optimizer converged. This file is read into UG and DOODS and the geometry are updated. The user may now choose to modify the geometry by accessing the previous menus or change the performance parameters and begin another optimization. The implementation of the analysis and optimization produced an easy-to-use design analysis application.

5. RESULTS

The integration of object-oriented programming, analysis, and optimization was accomplished for the flowpath portion of turbine modules in a gas turbine engine. The application of the method developed in this research resulted in an intuitive flowpath design tool which designers with little or no CAD/optimization experience can use.

The final test for the analysis/optimization portion of the design application was to see whether the flowpath geometry could be manipulated by the optimizer in order to improve the objectives. This section presents the results of 60 optimization runs in which the behavior of the analysis/optimization was evaluated. These optimizations were performed on two test cases of an LPT and two test cases of an HPT. Within these test cases, the settings for the objective weights were changed and different starting points were evaluated for each design. The optimizations were run on a Dell Latitude computer with an Intel Pentium 4, 1.80 Ghz processor and 1Gb of RAM. Each single optimization run took approximately 20 to 25 minutes.

There were 20 separate designs (five for each test case) and three different starting points were executed for each design. These starting points were selected to be geometrically very different, i.e. represent three points in the design space that are distant from each other. A design is defined as a fixed set of performance parameters and a specified combination of objective weights. The reason for changing the objective weight values is to study the trade-off between the efficiency and weight of the module. The only things that change between starting points are the values of the design variables: mean inlet radius, mean exit radius and exit radius ratio. The purpose of changing the starting point of the design is to determine where the global maximum is. If all the starting points converge to the same answer, it can be concluded the answer is at or very near the global maximum for the design.

The weight settings for the objectives in DOOPT were varied. Each combination of settings was run for all the starting points. The optimizations on both test cases returned favorable results. There are two aspects of the results that should be noted.

The first aspect is demonstrated by the results determined from the different starting points for the efficiency and the module weight. The values were all in the same vicinity of the design space, and this means that the optimization converged on the same optimal design. In order to quantify how well the optimizations converged, the maximum separation of the final efficiency and weight were calculated for each set of three starting points then averaged over all the optimization runs. The maximum separation for the efficiency was 2.4% and for the weight it was 48.26 lbs.

The second aspect to note about the results can be seen by looking at the data obtained from the optimizations compared to the assigned objective weight. The final results of the optimizations followed the trend that would be expected by the objective weight values. For example, when the efficiency and weight were considered equal objectives, they both were able to improve. When the efficiency was considered half as important as the weight, both objectives improved, however the weight improved more at the expense of the efficiency.

6. CONCLUSIONS

A design tool for the creation, analysis and optimization of the flowpath section of the turbine modules in a gas turbine engine was developed. This application was able to show: 1) that data was reliably processed and passed between three different software packages, 2) with the use of API, a CAD model was created that is robust enough to be used for optimization, 3) through the development of custom design applications, the user is not required to have an in-depth knowledge of analysis or optimization in order to perform successful analyses and optimization runs on a CAD model, 4) more than one parameterization can exist in the same model using the CAD API .

These results were formally verified by using the design application to perform analyses and optimizations on several LPTs and HPTs. In order to verify that the analysis functioned properly, analyses were performed on three LPTs and three HPTs. The modules were created and modified through the row-based and mod-based GUIs and analyzed from the analysis GUI. The performance parameters as well as the geometric input in the analysis GUI were modified through a wide range of reasonable values for each module that was tested. A total of 60 optimization runs were performed on four test cases (2 LPTs and 2 HPTs), and three starting points were used for each design. These optimizations converged to the same result for each set of three starting points, with an average separation of 2.4% for efficiency and 48.26 lbs for weight.

7. REFERENCES

[1] Delap, D., *CAD-Based Creation and Optimization of a Gas Turbine Flowpath Module with Multiple Parameterizations*, M.S. Thesis, Brigham Young University, Provo, UT, 2003.

[2] Giesing, J. and Barthelemy J., A Summary of Industry MDO Applications and Needs, 7th *AIAA/USAF/NASA/ISSMO Symposium on Multi-disciplinary Analysis and Optimization,* St. Louis, MO, 1998.

[3] Glassman, A., Computer Code for Preliminary Sizing Analysis of Axial-flow Turbines, *NASA* CR-4430, 1992.

[4] Glassman, A., Design Geometry and Design/Off-Design Performance Computer Codes for Compressors and Turbines, *NASA* CR-198433, 1994.

[5] Hoffman, C.M. and Kim, K., Towards valid parametric CAD models, *Computer-Aided Design*, Vol. 33, No. 1, 2001, pp. 81-90.

[6] Hogge, D., *Integrating Commercial CAx Software to Perform Multidisciplinary Design Optimization*, M.S. Thesis, Brigham Young University, Provo, UT, 2002.

[7] LaCourse, D, *Handbook of Solid Modeling,* McGraw-Hill, Inc., New York, NY, 1995.

[8] Rohl, P., He, B. and Flannigan, P., A Collaborative Optimization Environment for Turbine Development, 7th *AIAA/USAF/NASA/ISSMO Symposium on Multi-disciplinary Analysis and Optimization*, St. Louis, MO, 1998, AIAA Paper 98-4734.