Advances in Engineering Software 68 (2014) 9–18

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Integrated aerodynamic design and analysis of turbine blades

Chengen Wang *

Northeastern University, Shenyang 110819, China

ARTICLE INFO

Article history:
Received 16 August 2013
Received in revised form 19 September 2013
Accepted 27 October 2013
Available online 21 November 2013

Keywords:
Aerodynamic blade design
Airfoil modeling
Design integration
Design collaboration
Engineering process management
Flow analysis

ABSTRACT

This paper presents an integrated approach for aerodynamic blade design in an MDO (multidisciplinary design optimization) environment. First, requisite software packages and data sources for flow computations and airfoil modeling are integrated into a single cybernetic environment, which significantly enhances their interoperability. Subsequently, the aerodynamic blade design is implemented in a quasi-3D way, supported by sophisticated means of project management, task decomposition and allotment, process definition and coordination. Major tasks of aerodynamic blade design include 1D meanline analysis, streamsurface computations, generation of 2D sections, approximation of 3D airfoils, and 3D flow analysis. After compendiously depicting all the major design/analysis tasks, this paper emphatically addresses techniques for blade geometric modeling and flow analysis in more detail, with exemplar application illustrations.

1. Introduction

Turbine blades are core components of an aeroengine, a typical sub-category of gas turbines, which transform aerodynamic and thermal energy carried by gas fluids into mechanical driving energy. Consequently, operations of turbine blades largely dominate the global engine performance, such as thrust, aerodynamic and thermal efficiency, fuel-efficiency, and reliability. Today's increasingly augmented requirements for high performance aeroengines subject turbine blades to fiercely intensive aerodynamic, thermal and mechanical loadings. These extreme physical loadings make turbine blades work in very harsh conditions that approach the limits of its material properties such as melting points, stress and damage-resistance thresholds.

Accordingly, turbine blade design has been regarded as one of the most difficult engineering problems, which has drawn intensive attention from both industrial and academic circles. Generally, design of turbine blades is a complex multidisciplinary process involving the integration of several disciplines such as aerodynamics, structures, dynamics, and heat transfer [1].

Disciplinarily, aerodynamics engineers have to control the very complex flow phenomena occurring in highly loaded stages, on the whole operating range of the engine [2]. And, thermodynamics engineers have to accurately comprehend the heat transfer processes between gas flows, coolants and solid structures, and keep temperatures within the blade structure well below the allowable limits. While, mechanics engineers are expected to precisely predict and simulate structure deformations and dynamic responses under the aerodynamic, thermal and mechanical loadings.

Conventionally, the multiple disciplinary computations were carried out separately by different groups of engineers, aided by a number of stand-alone problem-solvers. Conceivably, the isolated computing processes result in remarkable engineering inefficiency and ensnarement in local optimums. Besides, the isolated legacy problem-solvers leave out of account strong couplings among disciplinary computations for turbine blade engineering.

To effectively tackle the drawbacks of traditional sequential design approaches, companies are growingly employing integrated, collaborative, and multidisciplinary design technologies. Actually, since the advent of information technologies, researchers have been continuously looking for cost-effective integration approaches to build bridges among isolated "automation islands" [3,4].

In regard to aeroengine design, a variety of integrated engineering infrastructures have been implemented to improve interoperability among heterogeneous software packages and data sources. For instance, Talya et al. (2000) presented an integrated multidisciplinary design optimization procedure for design of both internally and externally cooled gas turbine blades [5]. Houstis et al. developed a multidisciplinary problem solving environment to support collaborative design of gas turbines, where the legacy FORTRAN and C codes were wrapped up and delegated by agents [6].

As a common practice, turbine blade design commences with aerodynamic shape design, of which the ultimate goal is to find the airfoil surfaces that optimize the blade's aerodynamic

---

* Tel.: +86 24 83683619.
E-mail address: wangc@mail.neu.edu.cn

0965-9978/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.advengsoft.2013.10.005

performance. Blade airfoil design essentially relies on numerical computations that describe and predicate various fluid flows within turbine stages, in compliance with governing equations of mass, momentum and energy conservations. The fluid governing equations are generally called Navier–Stokes equations in the literature, and the corresponding software packages are called 3D Navier–Stokes solvers.

Thanks to rapid enhancements of computational capabilities, software codes of finite difference methods (FDM), finite element methods (FEM) and finite volume methods (FVM) are able to solve the 3D governing equations in reasonable time periods. However, finding numerical solutions to the governing equations in 3D blade passages is so time-consuming that for a long time the equations have been solved in 2D or quasi 3D spaces by industries.

And, in engineering practice, many companies have not shifted to the full 3D computational paradigm; 2D and quasi-3D aerodynamics solvers still play a dominant role in blade engineering. Recently, Koini et al. (2009) developed a software tool for interactively constructing parametric 3D blade models on basis of 2D profile sections [7]. Qiu et al. (2010) presented an integrated blade design system that incorporates 1D meanline analysis, quasi-3D throughflow and blade-to-blade flow calculations, and 3D CFD [8].

In accompaniment to worldwide zeal for multidisciplinary design optimization (MDO), we have been developing an MDO environment [9] for a decade, which concordantly incorporate information and application integration techniques [10–13], optimization algorithms [14], response surface models [15], etc. The MDO environment is a platform of high maintainability, scalability and flexibility, which may accommodate existing legacy problem-solvers and emerging problem-solvers. With support from a number of manufacturing companies, the ad hoc MDO environment has been employed in several engineering analysis and design scenarios like turbine blade design, aircraft flight-load design, aeroengine heat transfer analysis, ship power system design, and developments of car molds.

The goal of this paper is to present how the MDO environment is implemented to aerodynamic turbine blade design, by performing "what-if" computations of flow velocity and pressure distributions, and blade geometries. Specifically, this work addresses an initial pilot phase of implementing MDO techniques to aeroengine design, which aimed at validating feasibility and merits of applying multidisciplinary design integration and optimization techniques.

To reduce implementation complexity and to rapidly gain technical benefits, the practitioner engineers decided that emphasis of the pilot phase was put on design integration and collaboration in the context of aerodynamic blade engineering. This meant that the pursuit of design optimization was not explicitly addresses in this stage, and perspectives of heat transfer analysis and structural analysis were not accounted.

The rest of this paper is structured as follows. Section 2 describes how software codes and data sources used for quasi-3D aerodynamic blade design are integrated. Section 3 first explains how aerodynamic design is defined and managed in terms of projects, tasks and workflows. Then, Section 3 depicts the major tasks of blade airfoil engineering. Section 4 depicts in more detail the geometric modeling techniques that are used to approximate blade airfoils. Section 5 presents how flow characteristics analysis, various losses and efficiencies are computed in the MDO environments. Section 6 finally concludes this paper.

2. Aerodynamic blade design integration

In practice, a turbofan engine may run with one, two or up to three spools, which respectively have one, two or three turbines for extracting energy from the exhaust fluid. In turn, a turbine comprises multiple stages, each of which has a rotor that extracts the fluid energy, and a stator that adjusts the flow velocity and direction. Compositionally, a rotor is a cascade of blades attached to a disc at the roots, in which the throughflows of exhausts mainly move forwards in parallel to the axis of rotation. And, a stator is a cascade of blades or vanes attached to the engine casing at the tips, in which the throughflow velocity and direction are adjusted. This work addresses integrated aerodynamic blade design through all turbine stages in the MDO environment.

As unable being computed by a monolithic software package, the quasi-3D blade aerodynamic design is collectively computed by over a dozen of disciplinary analysis and design codes that respectively predict 1D, 2D and 3D flow characteristics, and depict the corresponding blade geometries in 2D and 3D spaces. It is implicitly assumed that the aerodynamic computations are conducted without accounting for heat transfer (adiabatic conditions), and structural responses induced by the aerodynamic loads.

2.1. Integration of blade aerodynamic design codes

Noticeably, all the in-house developed 1D and 2D flow computing codes and blade profile modeling codes were written in procedural FORTRAN and C languages, which use plain text files for data retrieval and transfer. And, engineers had to manually prepare the input files and visually parse the output files of computation processes because the legacy software applications were default of Graphical User Interfaces (GUIs). Generally, these stand-alone problem-solvers did not support out-process communications and multiple concurrent user access.

Consequentially, integration techniques are implemented as an essential part of the MDO environment, which accommodate all the codes and data sources on a single cybernetic platform. This enables effective coordination and tight collaboration in design of turbine blade airfoils.

Application integration allows executions of disciplinary software packages to be effectively controlled and coordinated. In the MDO environment, aerodynamic engineers do not have to memorize a large number of command lines for launching the legacy problem-solvers. And, they are provided with means to interact with the computation processes, by using surrogate objects [10].

Software packages integrated into the MDO environment are no longer independent systems but parts of a whole. Therefore, the software packages need to be known of their existence and let their executions be controlled and coordinated. The MDO environment uses dynamic registration and configuration approaches to keep track of the integrated applications. As shown in Fig. 1, an application's registration information generally includes its name, discipline, command lines, versions, running environments, executable file with a suffix of exe or bat, numbers and names of the input/output files, its host machine and absolute path.

2.2. Interoperation of heterogeneous data sources

In conjunction with application integration, information integration carries out interoperations among heterogeneous data sources, by building mapping relations among data objects, semantically parsing unstructured input/output files, and transforming data representation schemas. For instances, data mapping methods are implemented to discern disciplinary parameters contained in a solver's input/output files, and to maintain dynamic linkages between the non-structured parameter files handled by a solver and the structured data objects (database records or XML files) handle by the integration platform [11].