Proceedings of the ASME Turbo Expo 2010: Power for Land, Sea and Air
GT2010
June 14-18, 2010, Glasgow, Scotland

GT2010-22194

DEMONSTRATION OF AN AERODYNAMIC DESIGN PROCESS FOR TURBOMACHINES USING OPEN SOURCE SOFTWARE TOOLS

Oliver Borm, Balint Balassa, Sebastian Barthmes,
Julius Fellerhoff, Andreas Kührmann, Hans-Peter Kau

Institute for Flight Propulsion,
Technische Universität München,
Boltzmannstr. 15
D-85748 Garching, Germany
Email: oli.borm@web.de

ABSTRACT

This paper demonstrates an aerodynamic design process for turbomachines for compressible flows, using exclusively open source software tools. Some relevant software already existed and few additional components were required, which have been developed mainly by students and are available at `ftp.lfa.mw.tum.de`. *The geometry of turbomachine blades is described with a newly developed NURBS based blade designer. One-dimensional preliminary analysis is done with OpenOffice.org Calc and an extended mean line program, where loss models are already included. For two-dimensional through-flow computations a compressible streamline curvature method was implemented. Two-dimensional blade-to-blade and three-dimensional simulations are performed with the CFD toolbox OpenFOAM. The two- and three-dimensional results are visualized and analyzed using the open source postprocessing tool ParaView. The presented tools are regularly used in student projects. A generic one stage axial compressor was created with the workflow as a showcase in order to demonstrate the capabilities of the open source software tools.*

NOMENCLATURE

Latin symbols

m	meridional direction
s	chord length

Abbreviations

CGNS	CFD General Notation System
CAD	Computer Aided Design
GNU	GNU's Not Unix
GUI	Graphical User Interface
IGES	Initial Graphics Exchange Specification
NURBS	Non Uniform Rational Basis Splines
SCM	Streamline Curvature Method
VRML	Virtual Reality Modeling Language
VTK	Visualization Toolkit

INTRODUCTION

In the past years open source software has become more and more important in many areas. Modern GNU/Linux distributions are able to compete with proprietary operating systems, for server and even for desktop applications. Until now it was not possible to design turbomachines exclusively with open source software. In the presented work a new aerodynamic design process for turbomachines was created, using existing, or if missing, self implemented open source tools. This process covers only aerodynamical apsects and no structural mechanics. The general design procedure is demonstrated with a generic one stage axial compressor. The computational results of the used software are partially compared with established CFD tools. An extended validation would go beyond the scope of this paper.

Copyright © 2010 by ASME

The presented aerodynamic design process for turbomachines for compressible fluids is mainly developed and used by students. Especially for educational purposes, open source software seems to be an appropriate choice. Therefore, this approach is widely used in student projects at the Institute for Flight Propulsion. This paper describes the capabilities and the physical background of the software.

Utilizing open source software, one benefits from the know-how of others, which has already been implemented in a well documented way. The GNU General Public License declares the execution, manipulation, copying and redistribution of the software. These aspects have to be clarified in advance in a multiuser development, as every student has the copyright on his own code.

This paper is organized analogously to the presented design process as shown in Fig. 1. The demonstration of the design process is introduced by the creation of a geometry with the *BladeDesigner*. Subsequently the analysis of the blade geometry is described. The used solvers are OpenOffice, *Mittel* the mean line program, streamline curvature method (SCM) and OpenFOAM. In a final step the postprocessing is depicted via ParaView and *TurboVTK*. The application is demonstrated on a generic single stage axial compressor.

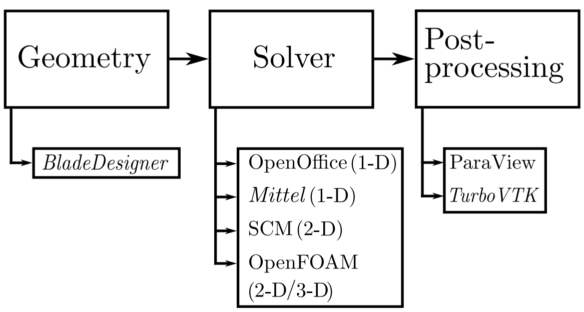

Figure 1. DEMONSTRATED DESIGN PROCESS

SOFTWARE LIBRARIES AND LANGUAGES

The tools which were developed are either implemented in C++ or python under a modern GNU/Linux operating system. Hence some additional software libraries are required.

For the purpose of handling non uniform rational basis spline (NURBS) curves and surfaces, the NURBS++ software library [1] is applied in C++. Python is a high level scripting language with a large number of additional libraries, which is very easy to learn for beginning programmers. There is no python library available for handling NURBS curves and surfaces with the functional range of the NURBS++ library. On account of this, python bindings for the NURBS++ library have been created by Blaim and Walter [2].

CFD General Notation System (CGNS) is a widley used file format to store results from CFD simulations. In order to read and write CGNS files with python an appropriate interface to the CGNS library is required. As the old pyCGNS python bindings are only working up to CGNS version 2.3, a set of new python bindings for CGNS input and output in python has also been implemented by Walter [3].

BLADE DESIGNER

For the design of the turbomachine blade geometry the parametric CAD tool *BladeDesigner* [4] has been developed by Balassa, Barthmes [5], Fellerhoff [6], Kührmann [7] and Walter. This tool generates a three-dimensional NURBS model from basic profile parameters and blade describing parameters. It supports multistage geometries and provides output in IGES, VRML, blockMeshDict and XML format as well as direct access to discrete points or NURBS curves and surfaces. The project is based on a preceding work of Mögele [8] who investigated the blade modelling possibilities with NURBS surfaces. Furthermore a graphical user interface (GUI) was implemented by Walter in PyQt4, which is shown in Fig. 2.

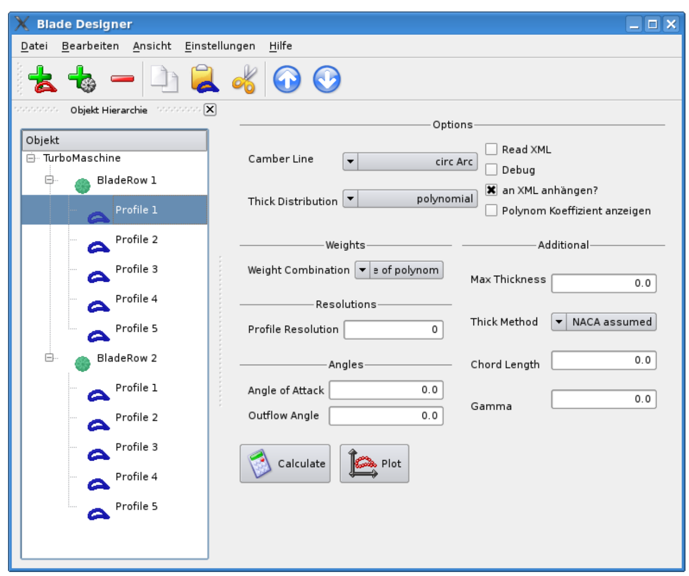

Figure 2. BLADEDESIGNER GUI

Copyright © 2010 by ASME

Two Dimensional Profile Modelling

A profile is defined parametrically and without dimensions by specifying mathematical functions for the camber line and the thickness distribution.

Each camber line function is primarily specified by defining an angle of attack and an outflow angle. The following alternative camber line functions are implemented:

- circular arc
- polynomial (degrees 2, 3, 4)
- NURBS curve
- NACA-65

A thickness distribution has to be chosen, which will be superposed over the camber line. For the purpose of assembling this thickness distribution perpendicularly to the camber line, the local gradient angle at each point of the camber line is calculated. Every implemented thickness distribution can be combined with any camber line. There are multiple methods available to create a thickness distribution:

- elliptic
- polynomial
- NURBS curve
- NACA-65

A special case is the double circular arc profile, which contains a circular arc camber line.

All camber line functions and thickness distributions are evaluated at discrete points. A finite difference method, in this case the iterative Tridiagonal Matrix Algorithm, is applied for the purpose of distributing the discrete points in a reasonable way. For this algorithm a weighting function is required, which distributes a higher number of points to areas, where the profile contour has a strong curvature. This weighting uses either directly the curvature of the thickness distribution, or is constituted by utilizing several simple mathematical functions like tangent or arcsin. All of these operations create a narrow allocation of points closely to the leading and trailing edges while arranging for a wider distribution in between.

For the axial compressor test case, the rotor mean line profile is shown in Fig. 3. A slightly modified NACA-65-(12)10 profile is used. It features a more realistic small trailing edge radius instead of the sharp edge of a NACA-65 airfoil. The profile is generated from a circular arc camber line and a polynomial thickness distribution, which resembles the original NACA-65 distribution very closely.

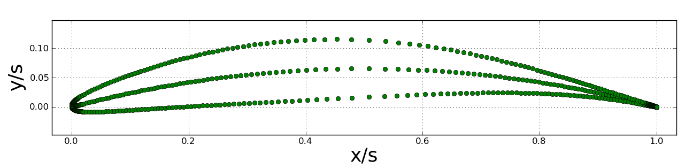

Figure 3. TWO-DIMENSIONAL PROFILE

Three Dimensional Blade Design

The three-dimensional blade modelling transforms the discrete points from the two-dimensional profiles to their proper position in three-dimensional space. Each blade row is defined by a set of two-dimensional profiles, the hub and shroud curves and the stagger angle of each profile. To position each profile correctly, either the three-dimensional location of the blade can be defined by the stacking line and chord length of each profile, or it can be defined by the leading and trailing edges of the blade. The first method is used mainly for axial machines, the second is useful for centrifugal turbomachines. The barycenter of the profile is used as stacking point for the stacking line. According to this definition each profile is sized, rotated and translated to its proper position in the two-dimensional m-y-space where m describes the meridional and y the pitchwise coordinate.

At this state the profile geometry describes the exact angles of the defined geometry. To keep these essential angles, a conformal mapping algorithm has been implemented by Barthmes [5] which is not only angle preserving but also keeps the dimensions of the profile with minimal deformation. The results of the computation are NURBS surfaces of the blade, the camber, the hub and the shroud. These NURBS surfaces can be used for further data processing in order to provide direct access to the blade geometry of each blade row. At the moment the Virtual Reality Modelling Language (VRML) or IGES output file formats are available.

Turbomachine Definition and Data Control

If more than one blade row is defined, the computation algorithm is repeated for each row. The *BladeDesigner* creates a project folder structure to organize different computations of the turbomachine.

The geometry of the generic axial compressor stage, which was generated with the *BladeDesigner* is shown in Fig. 4. The rotor has 23 blades, while the stator consists of 29 blades. Both blade rows are designed with the afore mentioned modified NACA-65-(12)10 profiles.

Block Mesh Output

A mesh definition file for the standard OpenFOAM mesh generator `blockMesh` can be generated from the turbomachine geometry. For each blade row a specific blockMeshDict file is created and consecutively processed by `blockMesh` in order to produce a single mesh for every blade row. For simulations with more than one blade row it is possible to merge these single grids.

Copyright © 2010 by ASME

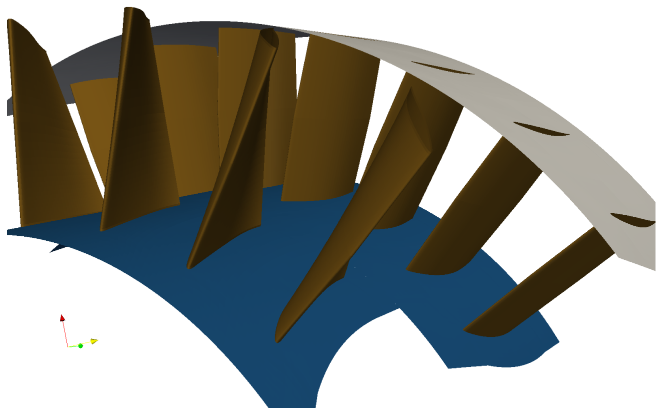

Figure 4. AXIAL COMPRESSOR STAGE GEOMETRY

A layer based architecture was implemented, where each mesh layer can either be flat or a surface of revolution. If only two layers are used, a two-dimensional mesh is generated. For more than two layers, three-dimensional grids are constructed. Based on the layer type either a linear cascade or a three-dimensional blade is created.

For each mesh layer a default O10H topology has been implemented, which is plotted in Fig. 5. The profile is surrounded by the O-grid which consists of six blocks. Around the O-grid ten H-blocks are positioned to complete the passage. In order to apply the *cyclic* boundary conditions for the one passage model in OpenFOAM, it is necessary that the vertex distribution at the upper and lower side of the mesh is exactly the same. If this vertex distribution is not exactly the same, the *cyclicGgi* boundary condition, with an interpolation between upper and lower side, has to be applied.

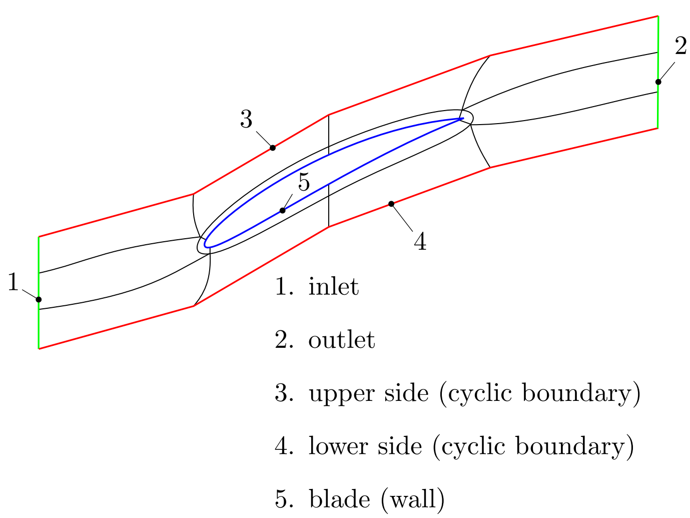

Figure 5. O10H TOPOLOGY

As the `blockMesh` tool does not support freeform surfaces, alternative open source mesh generators like gmsh [9] and calculix-cgx [10] are under active investigation.

Streamline Curvature Output

To transfer the geometric data from the *BladeDesigner* to the streamline curvature method, an appropriate interface is needed between the two. The *BladeDesigner* is implemented with such an interface, which rotates the provided three-dimensional NURBS surfaces of the turbomachine geometry into the meridian plane. For any blade row the projection is divided into three blocks. Each block edge is discretized with a user defined number of points. These points form a polyline with an optional grading in meridional and orthogonal direction. Based on these polylines a mesh is generated utilizing the OpenFOAM mesher `blockMesh` and converted back into a structured grid.

The SCM requires additional geometric properties at the vertices of this grid. These properties are the axial and radial coordinates of the vertex, the relative angles of the current camber surface and the blockage ratio of the blade thickness. If the camber surface is not available, it can be computed from the blade NURBS surface recursively. The grid properties are written into an output file, along with additional input information. The output file format is either plain text or CGNS, with partially user defined flow solution data.

ONE-DIMENSIONAL ANALYSIS

Two different duct flow methods are used for the preliminary one-dimensional analysis of turbomachines.

A basic mean line design of turbomachines is possible with OpenOffice.org Calc. Seven different analytical twist distributions, which are based on the simple radial equilibrium, were integrated for axial compressors. In Tab. 1 the global input parameters for the mean line design of the generic axial compressor are listed.

The standard fluid is air. Angles, velocities and thermodynamic state variables are computed at five different radial and three axial positions for each twist distribution. The final configuration uses the potential vortex swirl distribution for the simple radial equilibrium with a constant change of total enthalpy in radial direction.

For a quick preliminary analysis an extended one-dimensional mean line program for axial compressors was developed by Hüttl et al. [11]. Several loss models for different physical phenomena are included in the *Mittel* tool. At each axial station the flow can be evaluated at five radial positions. A radial total enthalpy distribution and twist distribution have to be

Copyright © 2010 by ASME

Table 1. INPUT PARAMETERS AXIAL COMPRESSOR.

Absolute Stagnation Temperature	293.15 K
Absolute Stagnation Pressure	101325 Pa
Absolute Inlet Machnumber	0.5 –
Total Pressure Ratio	1.12 –
Revolutions	8140 $\frac{1}{min}$
Mass Flow	12 $\frac{kg}{s}$
Hub-to-Tip Ratio	0.65 –
Polytropic Efficency	0.88 –
Degree of Reaction	0.5 –

chosen from five available distributions each as boundary condition. Also a modern GUI was implemented by Walter [12] using TkInter. Either a geometry or a work load and total pressure ratio can be specified as input. The output then quickly delivers the other data combined with additional radial distributions of angles, velocities, state variables, pressure losses and efficiencies. In Fig. 6 the mean line velocity triangles at the three axial stations of the generic axial compressor are plotted.

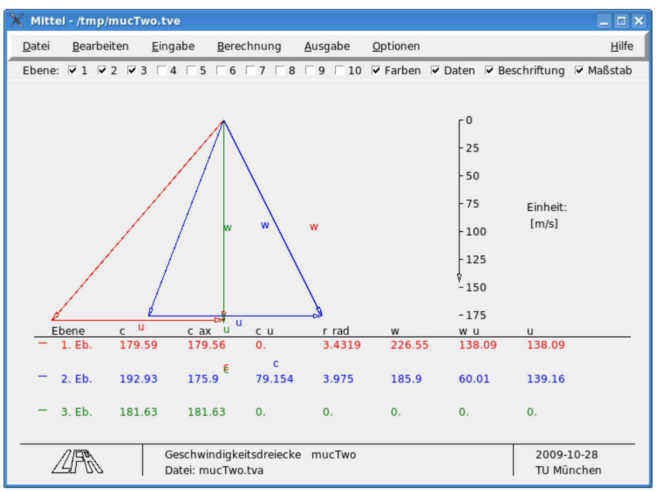

Figure 6. MEAN LINE VELOCITY TRIANGLES (MITTEL).

STREAMLINE CURVATURE METHOD

In order to get more detailed information about the flow distribution in the meridian plane of turbomachines, a compressible streamline curvature throughflow method [13] was implemented by Borm [14], Mayer [15] and Walter. The method is based on the theory of Lichtfuß [16] and Happel [17]. It was successfully used with multirow axial and centrifugal compressors.

The input files have to be provided in either plain text or standard CGNS format. Both file formats can be generated automatically with the *BladeDesigner*. The standard output file format is CGNS, which can be used with several post-processing tools.

The absolute stagnation temperature and pressure are needed for input. Also the overall mass flow and the angular velocity in each blade row need to be specified. There is an option to apply the static pressure at the hub and the circumferential component of the absolute velocity for each streamline at the inlet. Otherwise default initial static pressure at the hub and a swirl free inlet condition are used. An ordinary differential equation for the static pressure is solved at each quasi-orthogonal from an initial value at the hub to the shroud. This inner iteration is repeated until the overall mass flow has converged for the quasi-orthogonal. It is rerun for each quasi-orthogonal successively. There should be the same partial mass flow in each streamtube between two streamlines. Thus the streamlines are adopted to the new flow quantities at each quasi-orthogonal, except for the hub and shroud streamlines. These outer iterations are continued until the positions of the streamlines are not changing anymore. The method should converge in less than twenty outer iterations and one minute CPU clock time on modern processors from a standard initial solution.

The axial compressor stage was simulated with the SCM. The boundaries at the inlet were specified as mentioned in Tab. 1. The static temperature distribution of the stage is shown in Fig. 7. As the stage was designed with the potential vortex swirl distribution, there is a positive radial pressure gradient which results in a radial temperature gradient after the rotor row.

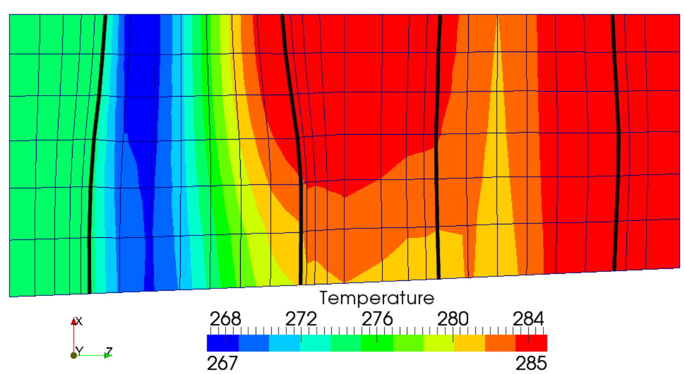

Figure 7. STATIC TEMPERATURE OF THE AXIAL COMPRESSOR STAGE (SCM).

Copyright © 2010 by ASME

OPENFOAM

For two- and three-dimensional flow simulations the CFD toolbox OpenFOAM [18] is used. OpenFOAM consists of several libraries to model steady, unsteady, compressible, incompressible, laminar and turbulent RANS as well as LES and other flow physics. The main advantage is that different libraries can be combined in the top-level solvers in order to simulate multi-physics flow.

Flat Cascade Profile

Flat two-dimensional cascade profiles are simulated with the standard `sonicFoam` solver of OpenFOAM-1.6. `SonicFoam` is a transient solver for sub- and supersonic, laminar or turbulent flows, including RANS and LES models. The rotor mean line profile of the axial compressor is computed with this solver and the k-Omega SST turbulence model is used. A constant stagnation temperature, stagnation pressure and the direction of the velocity are assumed as boundary conditions at the inlet. A constant static pressure is applied at the outlet. Periodic boundary conditions for the upper and lower sides, cf. Fig. 5, and special *empty* boundary conditions for two-dimensional problems are used at “hub” and “shroud”.

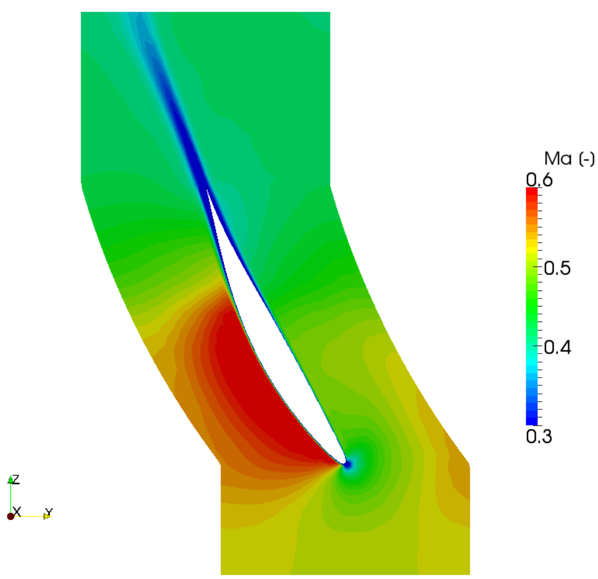

Figure 8. MACH CONTOUR OF THE TWO-DIMENSIONAL FLAT ROTOR PROFILE.

In Fig. 8 the Mach contour values of the plane NACA-65 rotor mean line profile are plotted. The local area with higher Mach numbers is formed on the suction side as anticipated. As expected the flow is adjacent to the geometry and fulfills periodicity. There is only a small separation in the rear part of the profile on the suction side.

Linear Cascade

The linear cascade is a straightforward three-dimensional model of the flat cascade profile. Thus the same solver, turbulence model and boundary conditions at inlet, outlet and upper as well as lower sides are used in order to simulate the linear cascade of the flat rotor mean line profile. As the simulation only covers half the blade span, a symmetry plane at the “shroud” and solid wall boundary conditions at the hub are used.

The simulated linear cascade passage was repeated once in y direction and is shown in Fig. 9. The static pressure distribution is visualized at all solid walls. Moreover four surfaces with constant meridional coordinates are plotted with the contour of the turbulent kinetic energy. In the mean surface of the cascade, which is in this simulation the top of the blade, the fluid flow is of almost two-dimensional character. Fluid of a higher turbulent kinetic energy is concentrated in the corner of the blade suction side and hub. This fluid is convected downstream the blade and mixed out. The vertical streamlines are separated slightly from the hub in the direction towards the mean surface at approximately 80% of the suction side, exactly where the fluid with the high turbulent kinetic energy accumulates.

S1 Surface

For the simulation of rotating blade rows some additional models were needed. The *SRFZones* library was implemented

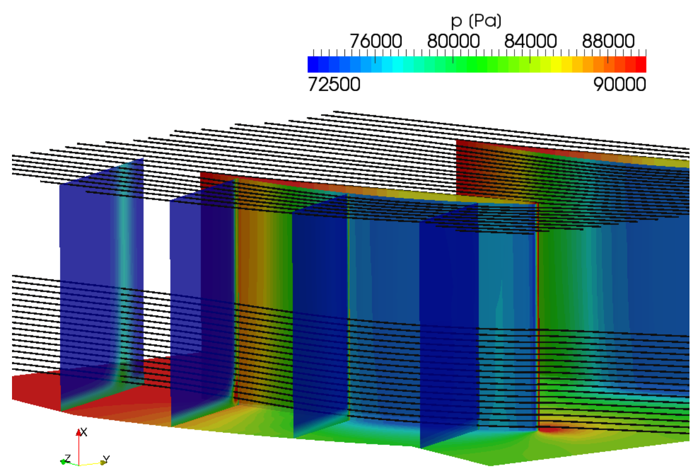

Figure 9. STATIC PRESSURE AT THE SOLID WALLS OF THE LINEAR CASCADE.

Copyright © 2010 by ASME

[19] in order to compute additional fictitious forces in the relative coordinate systems within multiple zones. The coriolis and centrifugal forces for the momentum equation and the centrifugal force for the energy equation are evaluated in each cell. This library is used in the `sonicSRFFoam` solver, which is basically derived from the standard `sonicFoam` solver in order to compute the relative velocity in relative coordinate systems.

The `sonicSRFFoam` solver is used for the computation of the mean line profile of the axial compressor rotor at the two-dimensional S1 surface. This type of simulation can be either used separately for blade-to-blade simulations or in conjunction with a streamline curvature method for the evaluation of the entropy rise. The absolute stagnation temperature, the absolute stagnation pressure and the direction of the absolute velocity in cylindrical coordinates are set as boundary conditions at the inlet. A constant static pressure is used at the outlet. A matching periodic boundary condition is applied at the pitchwise sides. The "hub" and "shroud" boundaries are of type symmetry plane.

For solving the relative velocity in rotating coordinate systems, some appropriate boundary conditions are necessary. As a result a boundary condition was implemented [19] for the purpose of computing the static temperature in a relative coordinate system from the specified absolute stagnation temperature and the relative and angular velocity at each specified cell. In addition a boundary condition was developed to convert the defined direction of the absolute velocity into the direction of the relative velocity in either cartesian or cylindrical coordinates.

As only one blade passage was simulated, it was repeated in pitchwise direction for a better visualization of the flow. The magnitude of the relative velocity distribution is plotted in Fig. 10. There is a local area in front of the the leading edge with a low magnitude of the relative velocity. The high velocity field at the suction side of the profile decreases towards the pressure side of the profile due to the periodic boundary conditions. The incidence at the leading edge seems to converge to zero.

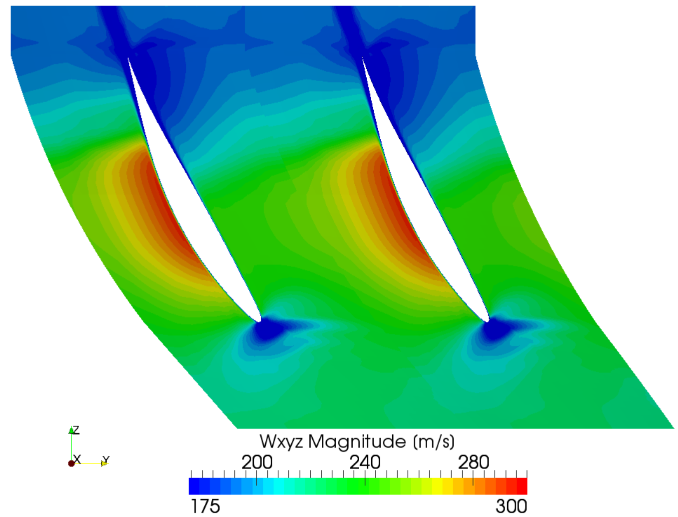

Figure 10. MAGNITUDE OF THE RELATIVE VELOCITY AT THE ROTOR MIDSPAN S1 SURFACE.

Single Rotor

A straightforward extension of the computation of S1 surface flows is the single rotor computation. All the afore mentioned methods can be used analogously. If the rotor stator interface is not coincident with geometrical gaps at a boundary wall, like at hub and shroud, this boundary patch has two different absolute velocities. Therefore, an additional boundary condition, in which two different angular velocities for a boundary patch are defined, was introduced [19]. In order to cut the boundary patch into two regions with the two different angular velocities, a bounding box in the meridian plane has to be specified. Thus all cell centers of the boundary patch that are inside the bounding box are associated with the second specified angular velocity, and all other cell centers outside the bounding box use the first angular velocity.

The rotor of the axial compressor was simulated with the `sonicSRFFoam` solver. The boundary conditions are identical to the ones used in the S1 surface computation, except for the ones at the hub and the shroud patches. For these patches the above mentioned boundary condition was used. The mesh has approximately 160k cells and was generated with the blockMesh Output of the *BladeDesigner*.

In Fig. 11 the temperature distribution is shown at the solid wall boundary patches. Moreover surface streamlines at the solid wall boundaries are visualized with the shear stress vector. The fluid flow seems to have an almost two-dimensional topology. No serious three-dimensional flow effects are visible.

For comparison reasons the geometry of the generic axial compressor rotor was also simulated with Numeca/Fine Turbo. The boundary conditions are identical to the ones of the OpenFOAM simulation and the number of cells are similar. Nevertheless, the mesh topologies differ and a comparison between these two solvers is strongly rough. This task should only serve as an indication to validate the OpenFOAM solutions.

Table 2. COMPARISON OF GLOBAL PARAMETERS.

Flow Quantity	OpenFOAM	Fine Turbo	Difference
θ_{tt}	1.0294 –	1.0314 –	-0.1980 %
Π_{tt}	1.1047 –	1.1077 –	-0.3055 %
$\dot{m}$	12.719 $\frac{kg}{s}$	12.699 $\frac{kg}{s}$	1.9469 %

Copyright © 2010 by ASME

In Tab. 2 global parameters are compared between the two simulations. The static temperature and the components of the absolute velocity are averaged with mass flow weighting at the outlet, whereas the static pressure is averaged with an area weighting. The absolute total temperature ratio θ_{tt} and the absolute total pressure ratio Π_{tt} are computed with these averaged variables. Furthermore, the mass flux is integrated at the outlet. The global parameters from both simulations are quite similar, underlining the capability of the open source software.

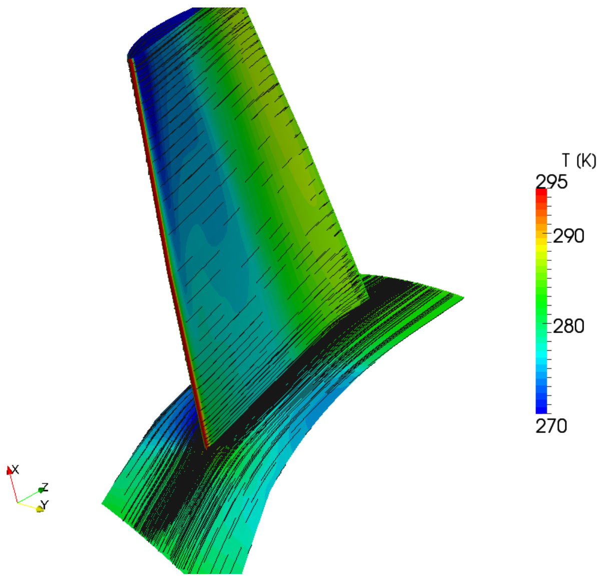

Figure 11. STATIC TEMPERATURE AT THE SOLID WALLS OF THE ROTOR WITH SURFACE STREAMLINES.

Some preliminary work at steady (*mixing plane*) and unsteady (*domain scaling*) rotor stator interfaces for incompressible flows has been done by Blaim [20]. Newly developed rotor stator interfaces are available in OpenFOAM. But due to time limitations these *mixing plane* and *domain scaling* interfaces have not yet been tested.

POSTPROCESSING

The post processing is carried out with the open source, scalable, multi-platform data analysis and visualization application ParaView [21]. ParaView is based on the Visualization Toolkit (VTK). It is designed for visualization of large data sets and has been used with up to 6 billion cells in structured grids. It is able to run in parallel mode on multiple-processor distributed-memory supercomputers. In order to analyze turbomachines some additional tools like *TurboVTK* [22] were implemented in python using VTK bindings.

Dimensionless Meridional and Spanwise Coordinates

For each blade row dimensionless meridional and spanwise coordinates are computed with a linear transfinite interpolation. Thus a background mesh is created in the meridian plane with four boundary curves (hub, inlet, shroud, outlet). The boundary curves have to be provided either from the geometry modelling system or can be extracted from the boundary patches inside ParaView. The values of the meridional and spanwise coordinates of the background mesh are interpolated onto the three-dimensional grid points. With these values it is possible to create iso surfaces of constant spanwise and meridional coordinates from the three-dimensional mesh.

The boundary curves of the axial rotor in the meridian plane are prepared with the *BladeDesigner*. The dimensionless spanwise and meridional coordinates are generated for the whole rotor mesh as described above. In Fig. 12 the distribution of the relative Mach number is shown at such a constant spanwise coordinate. For an axial turbomachine this geometric S1 surface is identical to a surface with a constant radius. The computation of the coordinates can be accomplished either within an external python script or inside the ParaView GUI with the *Programmable* filter.

Conformal Mapping of S1 Surfaces

To visualize and analyze the flow on S1 surfaces, a common approach is to flatten the three-dimensional surface into two dimensions. An angle-preserving algorithm for conformal mapping based on fundamental nets has been developed and imple-

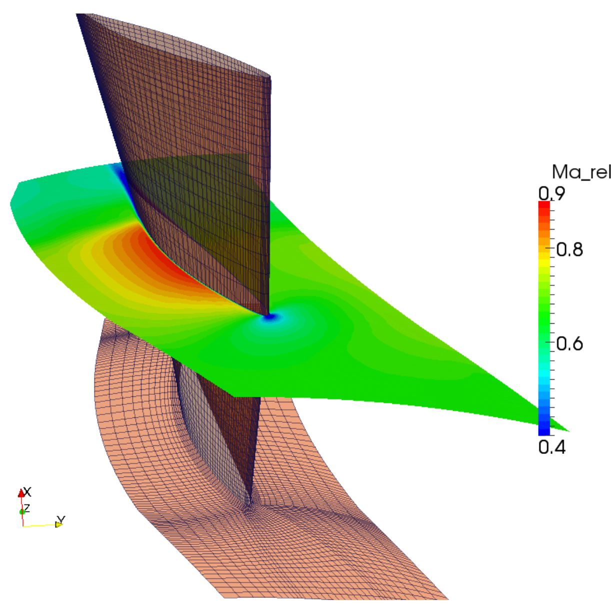

Figure 12. MAGNITUDE OF THE RELATIVE MACH NUMBER AT THE ROTOR MIDSPAN S1 SURFACE.

Copyright © 2010 by ASME

mented. This tool extracts an S1 surface at a specified spanwise coordinate from a three-dimensional mesh with given dimensionless meridional and spanwise coordinates. Each point of this surface is transformed conformally to the two-dimensional plane. This algorithm can be applied either directly in a python script or via the ParaView GUI within a *Programmable* filter.

An S1 surface at 50% dimensionless spanwidth coordinate is extracted from the rotor mesh of the axial compressor and mapped in two dimensions. This is done with the results of the Fine Turbo computation within *TurboVTK* and Numeca/CFView in order to compare both implementations. The results for the magnitude of the relative velocity distribution at this spanwise coordinate are quite similar, as shown in Fig. 13. Furthermore, this can be compared with the two-dimensional OpenFOAM simulation in Fig. 10.

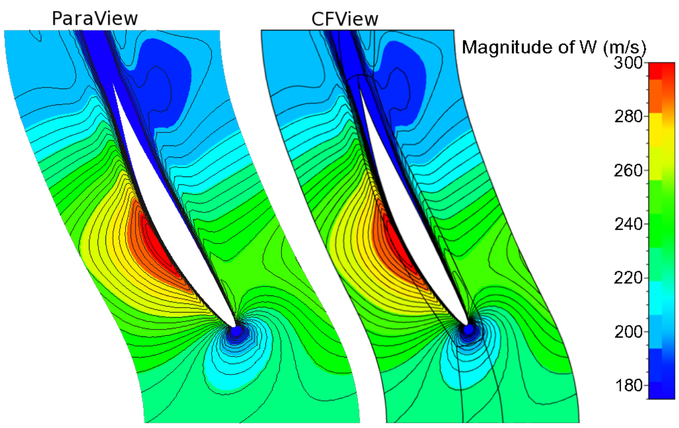

Figure 13. MAGNITUDE OF THE RELATIVE VELOCITY AT THE ROTOR MIDSPAN S1 SURFACE (CONFORMAL TWO-DIMENSIONAL VIEW).

Pitchwise Averaging

For the purpose of visualizing the flow distribution in the meridian plane from three-dimensional simulations, a pitchwise averaging is performed. Thus each specified flow quantity which is available at each grid point of the three-dimensional mesh can be averaged either area or mass flow weighted. For each dimensionless meridional coordinate from the background mesh a constant meridional surface inside the three-dimensional mesh is created. Moreover this constant meridional surface is divided for each grid point of the background mesh into a stripe in spanwise direction. For each stripe an averaged value for each desired flow quantity is computed. These values are stored at the associated grid point of the background mesh. This algorithm can be used inside the ParaView GUI in the *Programmable* filter or in an external python script.

The result of the single rotor computation from Fine Turbo was used in order to compare the pitchwise averaging between *TurboVTK* and CFView. In Fig. 14 the pitchwise mass flow weighted average static temperature is plotted. There are only minor differences, which may result from different background meshes. In addition the pitchwise averaged static temperature can be compared with the result from the streamline curvature method in Fig. 7.

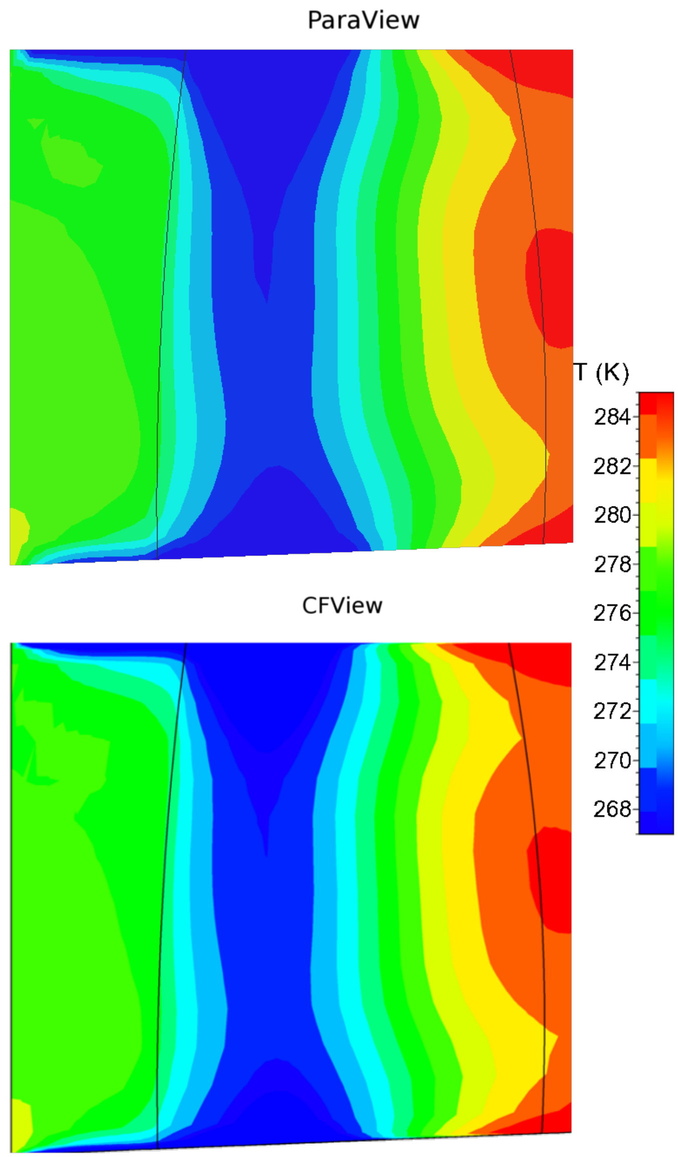

Figure 14. PITCHWISE MASSFLOW WEIGHTED AVERAGE STATIC TEMPERATURE OF THE ROTOR.

Copyright © 2010 by ASME

Due to the thickness of the rotor blade an acceleration of the axial velocity takes place, resulting in a decrease of the static temperature inside the rotor blade passage. The relative deceleration of the flow in the compressor rotor causes an overall static temperature rise at the outlet compared to the inlet.

Turbo Calculator

An array calculator to compute additional flow quantities in ParaView is available. Nevertheless, it is annoying and error-prone to type the formula for each flow quantity into the array calculator. On account of this a python based *Programmable* filter was written for the purpose of computing specified additional flow quantities like absolute and relative stagnation flow quantities, velocities in cylindrical coordinates, entropy and Mach number among others. This tool is not restricted to OpenFOAM cases and can therefore easily be extended to other flow solvers. Only the names of the input variables have to be adjusted accordingly.

Beyond that it is possible to calculate scalar, vector and tensor gradients with the *Compute Derivatives* filter in order to get the vorticity or strain of flow quantities for example.

CONCLUSION

A new aerodynamic design process for compressible turbomachines using open source software tools has been developed and described. The functionality of the whole design process was successfully demonstrated with a single stage axial compressor. A new blade designer was implemented in order to describe the blade geometry in a parametric way for multiple axial and radial blade rows. Furthermore, the geometric input for a streamline curvature method and OpenFOAM can be generated automatically. Tools for one-, two- and three-dimensional flow analysis were presented. The postprocessing of the two- and three-dimensional results is accomplished with ParaView. These software tools have already proven very useful for the education of students.

Future development will aim for the implementation of rotor stator interfaces in OpenFOAM. Moreover the software tools will be further validated and verified via well known measurements and established CFD codes. In addition to that a compressible multi physics solver for coupled Fluid-Structure-Interaction in turbomachines is planned. Additionally the design process could also be coupled with optimization techniques like Genetic Algorithms for instance Eva2 [23] and Neuronal Networks like SNNS [24].

ACKNOWLEDGMENT

Special thanks go to Herbert Mögele for the first version of the *BladeDesigner* [8] and Steve Walter for his work on several subprojects [2, 3, 4, 12, 13]. Furthermore, special thanks go to Franz Blaim for the preliminary work on rotor stator interfaces in OpenFOAM [20].

REFERENCES

[1] NURBS++, 2002. Version 3.0.11. `http://libnurbs.sourceforge.net`.

[2] PythonNURBS, 2009. Version 0.3. `http://pypi.python.org/pypi/PythonNURBS`.

[3] PythonCGNS, 2009. Version 2009.10.30. `http://pypi.python.org/pypi/CGNS`.

[4] BladeDesigner, 2009. Version 2009.09.22. `ftp://ftp.lfa.mw.tum.de/pub/Software/bladedesigner`.

[5] Barthmes, S., 2010. "Entwicklung eines parametrischen CAD-Systems für Turbomaschinen". Semester thesis, Institute for Flight Propulsion, Technische Universität München.

[6] Fellerhoff, J., 2010. "Entwicklung eines parametrischen Profilgenerators zur Beschreibung von 2-D Profilschnitten in Turbomaschinenbeschaufelungen". Semester thesis, Institute for Flight Propulsion, Technische Universität München.

[7] Kührmann, A., 2010. "Entwicklung eines Programms zur parametrischen Generierung von OpenFOAM Netzdefinitionsdateien für Turbomaschinenbeschaufelungen". Semester thesis, Institute for Flight Propulsion, Technische Universität München.

[8] Mögele, H., 2006. "Parametrische Geometriedefinition von Turbomaschinenbeschaufelungen". Semester thesis, Institute for Flight Propulsion, Technische Universität München.

[9] gmsh, 2009. Version 2.4.2. `http://www.geuz.org/gmsh`.

[10] calculix-cgx, 2009. Version 2.0. `http://www.dhondt.de`.

[11] Hüttl, T., 1994. "Zur EDV-gemäßen Darstellung von Axial-Turboverdichter-Auslegungen auf der Basis eines erweiterten Mittelschnittsverfahren". Diploma thesis, Institute for Flight Propulsion, Technische Universität München.

[12] Mittel, 2009. Version 2009.04.16. `ftp://ftp.lfa.mw.tum.de/pub/Software/mittel`.

[13] skv, 2010. Version 0.0.4. `ftp://ftp.lfa.mw.tum.de/pub/Software/skv`.

[14] Borm, O., 2006. "Entwicklung eines Stromlinienkrümmungsverfahrens". Semester thesis, Institute for Flight Propulsion, Technische Universität München.

[15] Mayer, F., 2008. "Entwicklung eines Stromlinienkrümmungsverfahrens für die Turboladervorentwicklung". Diploma thesis, Institute for Flight Propulsion, Technische Universität München.

Copyright © 2010 by ASME

[16] Lichtfuß, H.-J., WS 2005/06. *Anwendung strömungsmechanischer Berechnungsverfahren in Flugtriebwerken*. Technische Universität München. Lecture Notes.

[17] Happel, H.-W., WS 2007/08. *Rechenverfahren zur Turbomaschinenauslegung*. Universität der Bundeswehr München. Lecture Notes.

[18] OpenFOAM, 2009. Version 1.6. `http://www.openfoam.com`.

[19] Borm, O., 2009. Contributions to the OpenFOAM SIG TurboMachinery. Version 1.6. `http://openfoam-extend.svn.sourceforge.net/viewvc/openfoam-extend/trunk/Breeder_1.6/OSIG/TurboMachinery`.

[20] Blaim, F., 2008. "Implementation of Rotor Stator Interfaces for the Open Field Operation and Manipulation (OpenFOAM) CFD Toolbox". Diploma thesis, Institute for Flight Propulsion, Technische Universität München.

[21] ParaView, 2009. Version 3.6.1. `http://www.paraview.org`.

[22] TurboVTK, 2009. Version 2009.11.15. `ftp://ftp.lfa.mw.tum.de/pub/Software/turbovtk`.

[23] EvA2, 2010. Version 2.043. `http://www.ra.cs.uni-tuebingen.de/software/EvA2`.

[24] SNNS, 2008. Version 4.3. `http://www.ra.cs.uni-tuebingen.de/software/snns`.

Copyright © 2010 by ASME