

12-2005

An Automated Optimal Design of a Fan Blade Using an Integrated CFD/MDO Computer Environment

Uyi O. Idahosa

Embry-Riddle Aeronautical University - Daytona Beach

Follow this and additional works at: <https://commons.erau.edu/db-theses>



Part of the [Aerospace Engineering Commons](#)

Scholarly Commons Citation

Idahosa, Uyi O., "An Automated Optimal Design of a Fan Blade Using an Integrated CFD/MDO Computer Environment" (2005). *Theses - Daytona Beach*. 310.

<https://commons.erau.edu/db-theses/310>

This thesis is brought to you for free and open access by Embry-Riddle Aeronautical University – Daytona Beach at ERAU Scholarly Commons. It has been accepted for inclusion in the Theses - Daytona Beach collection by an authorized administrator of ERAU Scholarly Commons. For more information, please contact commons@erau.edu.

**AN AUTOMATED OPTIMAL DESIGN OF A FAN BLADE
USING AN INTEGRATED CFD/MDO COMPUTER ENVIRONMENT**

By

Uyi O. Idahosa

**Thesis Submitted to the
Department of Aerospace Engineering
In Partial Fulfillment of the Requirements for the Degree of
Master of Science**

**Embry Riddle Aeronautical University
Daytona Beach, Florida
December 2005**

UMI Number: EP32044

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform EP32044
Copyright 2011 by ProQuest LLC
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

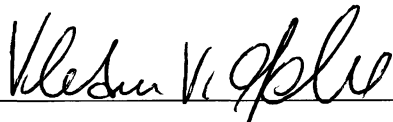
ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

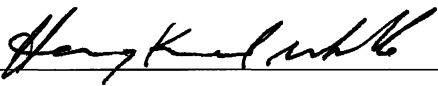
AN AUTOMATED OPTIMAL DESIGN OF A FAN BLADE
USING AN INTEGRATED CFD/MDO COMPUTER ENVIRONMENT


By
Uyi O. Idahosa


This thesis was prepared under the direction of the thesis committee chairman, Dr. Vladimir Golubev, Department of Aerospace Engineering and has been approved by the members of the thesis committee. It was submitted to the Aerospace Engineering Department and was accepted in partial fulfillment of the requirements for the degree of Master of Science in Aerospace Engineering.

THESIS COMMITTEE:


Dr. Vladimir Golubev
Chairman


Dr. Hany Nakhla
Member


Associate Provost


Dr. Eric Perrell
Member


Department Chair, Aerospace Engineering

2/24/06
Date

ACKNOWLEDGEMENTS

I would like to thank the member of the various research teams who contributed to the initial phases of the projects, particularly Siddarth David, Sean Kotka, Syed Abedi, Joe Burkhart and James Moss

I would like to thank Dr. Golubev for all his help in getting the research project completed while giving me the latitude to apply my initiative in establishing what direction the research would eventually take.

I am also indebted to my parents Gabriel and Bola Idahosa, who have made significant financial sacrifices to make my graduate studies possible as well as steadily encouraging me to work and achieve the best I can.

ABSTRACT

Author: Uyi O. Idahosa

Title: An Automated Optimal Design of a Fan Blade Using an Integrated CFD/MDO Computer Environment

Institution: Embry-Riddle Aeronautical University, Daytona Beach, FL

Year: 2005

The objective of the investigation is the development of more efficient design methodologies based on the applications of established design tools including Computational Fluid Dynamics (CFD) and non-linear Multidisciplinary Design Optimization (MDO) algorithms. Well known evolutionary type optimization algorithms include the Particle Swarm Optimization (PSO), Response Surface Optimization (RSO) and Genetic (GA) Algorithms. The benchmark case study is the optimal design of a low speed fan for an industrial air-conditioning application using the Response Surface Optimization (RSO) algorithm.

The optimization algorithm controls the variations of parameters that describe the three-dimensional geometry of the blade while applying performance and geometrical constraints on blade shapes that are investigated. The optimal design is defined as the blade geometry which produces the maximum total efficiency subject to specified constraints on the volume flow rate (CFM) and rotational rate (RPM) of the fan.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	iii
ABSTRACT.....	iv
TABLE OF CONTENTS.....	v
TABLE OF FIGURES.....	viii
NOMENCLATURE.....	xiii
1 INTRODUCTION.....	1
1.1 Background.....	1
1.2 Problem Statement.....	2
1.3 Approach.....	3
1.4 Expected Results.....	8
2 METHODOLOGY.....	9
2.1 Blade Geometry Generation.....	9
2.1.1 CFX-Bladegen Coordinate System.....	12
2.1.2 Blade Geometry File Format.....	17
2.2 Design Evaluation Using Computational Fluid Dynamics.....	21
2.2.1 CFX-Bladegen(Plus) Batch Utilities.....	24
2.2.1.a CFD Pre-Processing:.....	24
2.2.1.b Mesh Generation.....	25
2.2.1.c Flow Field Solution.....	26
2.2.1.d Post-Processing.....	27
2.3 The CFX-Bladegen(Plus) CFD Solver.....	29
2.3.1 Governing Navier-Stokes Equations.....	29
2.3.2 The Rotating Reference Frame.....	32
2.3.3 Reynolds-Averaged Navier Stokes (RANS) Equations.....	35
2.3.4 Reynolds Stresses and the Zero Equation Turbulence Model.....	37
2.3.5 Solver Numerical Discretization and Solution Scheme.....	39
2.3.6 Solver Residual Computation Scheme.....	42

2.3.7	Solver Timestep and Target Residual Selection	43
2.4	Multidisciplinary Optimization (MDO) Algorithms	44
2.4.1	Response Surface Approximate Optimization (RSO)	48
2.4.2	Particle Swarm Optimization (PSO).....	54
2.4.3	Genetic Optimization Algorithm (GA).....	58
2.4.4	Selecting the MDO Algorithm for the Benchmark Study	61
3	MDO DESIGN IMPLEMENTATION.....	62
3.1	3-Step Multidisciplinary Design Optimization Process.....	62
3.2	Blade Geometry Parameterization	65
3.2.1	Blade Airfoil/Twist Parameterization Using Bezier Curves.....	70
3.2.1.a	Bezier Control Point Indexing Scheme.....	73
3.2.1.b	Meridional ($M'_{R S}$) Bezier Polygon MDO Design Variables.....	74
3.2.1.c	Angular ($\beta_{R S}$) Bezier Polygon MDO Design Variables	77
3.2.2	Blade Sweep Parameterization - Spanwise LE Angular Coordinate	78
3.2.3	Dynamic Constraints on Blade Geometry Parameterization	80
3.2.3.a	Dynamic Constraints on Tangential ($\beta_{r s}$) Design Variables	81
3.2.3.b	Dynamic Constraints on LE Circumferential (θ_r) Design Variables	82
3.3	MDO Environment Implementation (VisualDoc)	83
3.3.1	Static Constraints for Design Variables and Responses	91
3.4	MDO Response Analysis Implementation (VisualScript).....	96
3.4.1	VisualScript Element Definition – Build_Bezier	98
3.4.2	VisualScript Element Definition – Bladetest.....	100
3.4.3	VisualScript Element Definition – Extract_Bezier.....	102
3.4.4	VisualScript Element Definition – CFX_Update_Blade	104
3.4.5	VisualScript Element Definition – CFX_CFD_Pre-Process	107
3.4.6	VisualScript Element Definition – GridTest.....	110
3.4.7	VisualScript Element Definition – CFX_CFD_Solver.....	112
3.4.7	VisualScript Element Definition – SolutionTest	116
3.4.8	VisualScript Element Definition – Extract_Solution.....	118

3.4.9	VisualScript Element Definition – Post_Process.....	119
4	RESULTS	125
4.1	Base (Starting) Fan Design Models	125
4.2	Results – Response Surface Optimization (RSO) Algorithm	129
4.2.1	RSO Results – Model1.....	130
4.2.2	RSO Results – Model2.....	148
4.2.3	RSO Results – Model3.....	165
4.3	Similarity (Uniqueness) of Optimal RSO Designs	183
4.4	Robustness of Automated MDO Environment	189
5	CONCLUSIONS & RECOMMENDATIONS.....	192
	REFERENCES	196
	APPENDIX A: SAMPLE INPUT FILES	200
	APPENDIX B: MATLAB SCRIPTS.....	216

TABLE OF FIGURES

Figure 1: VisualDoc Graphical User Interface	4
Figure 2: Interaction Flowchart for MDO Design Components	5
Figure 3: MDO Design Base Model1	7
Figure 4: MDO Design Base Model2	7
Figure 5: MDO Design Base Model3	7
Figure 6: CFX-Bladegen Graphical Interface.....	9
Figure 7: Bladegen GUI Meridional and Auxiliary Views.....	10
Figure 8: Pressure/Suction Mode Blade Layer View	11
Figure 9: Angle/Thickness Mode Blade Layer View	11
Figure 10: Bladegen Coordinate System	12
Figure 11: Chord-Meridional Length Relationship	14
Figure 12: β -M' and θ -M' Sample Distributions.....	16
Figure 13: Bladegen Batch File Nomenclature.....	18
Figure 14: Batch Input File Section Samples	20
Figure 15: CFX-Bladegen(Plus) Graphical User Interface.....	21
Figure 16: Blade Passage for Fluid Flow Analyses	22
Figure 17: Blade Passage Unstructured Grid.....	23
Figure 18: Bladegen(Plus) GUI Functions – <i>BgBatch</i> Utility	24
Figure 19: Bladegen(Plus) GUI Function – <i>BgGrid</i> Utility.....	25
Figure 20: Unstructured Grid Detailing Inflation Layers	26
Figure 21: Bladegen(Plus) GUI Function – <i>BgSolve</i> Utility	26
Figure 22: Bladegen(Plus) GUI Function – <i>BgExtract</i> Utility	27
Figure 23: Sample Plot of P _{tr} at 50% Span	28
Figure 24: Sample Finite Volume [42]	40
Figure 25: VisualDoc Graphical User Interface (GUI).....	44
Figure 26: VisualScript Graphical User Interface (GUI).....	45
Figure 27: Response Surface Approximation of Objective Function.....	48

Figure 28: Data points and Response Surface Approximation (1-Variable Problem).....	51
Figure 29: Number of Response Analyses Calls Relative to RSO Model.....	53
Figure 30: MDO Component Interaction.....	63
Figure 31: CFX-Bladegen Coordinate System	65
Figure 32: Meridional View (LHS) of Fan with Span Layers Visible	66
Figure 33: β vs. M' For Airfoil Camber Line Generation at Span 0.00R (Hub)	66
Figure 34: Sample Camberline of Blade Profile at Span Layer.....	67
Figure 35: Camberline with Imposed Thickness Distribution.....	69
Figure 36: Camberline Generation using a Bezier Control Polygon	70
Figure 37: Generalized Blade Geometry CP Grid (Composed of Bezier CPs)	71
Figure 38: Generalized Blade CP Grid Coordinate Components	72
Figure 39: MDO Design Variables for Meridional Blade CP Grid	74
Figure 40: Blade Profile Generation Using Bezier Polygon.....	75
Figure 41: Meridional Design Variables on Blade CP Grid w/“Displacement Factor” ...	76
Figure 42: MDO Design Variables for Angular (β) Blade CP Grid.....	77
Figure 43: Circumferential Angle (θ) Nomenclature.....	78
Figure 44: Invalid Blade Geometry Obtained from Unbounded Parameterization	80
Figure 45: MDO Environment Component Interaction.....	83
Figure 46: VisualDoc Design Variables (Input) Specification (Part I).....	85
Figure 47: VisualDoc Design Variables (Input) Specification (Part II)	86
Figure 48: VisualDoc Response Specification	88
Figure 49: Decomposed MDO Component Interaction Flowchart.....	94
Figure 50: VisualScript MDO Response Analysis (CFD) Implementation.....	96
Figure 51: Conditional (<i>if</i>) VisualScript Element.....	97
Figure 52: VisualScript <i>if</i> Element Definition.....	97
Figure 53: VisualScript Element Definition – Build_Bezier	99
Figure 54: VisualScript Element Definition – Bladetest	100
Figure 55: Conditional (IF) Element Definition – Bladetest	101
Figure 56: VisualScript Element Definition – Extract_Bezier	102

Figure 57: VisualScript Element Definition – CFX_Update_Blade (3D Blade File)	104
Figure 58: VisualScript Element Definition – CFX_Update_Blade (Parameter File) ...	105
Figure 59: VisualScript Element Definition – CFX_CFD_Pre-Process (BladeBatch) ..	107
Figure 60: VisualScript Element Definition – CFX_CFD_Pre-Process (BgBatch)	108
Figure 61: VisualScript Element Definition – CFX_CFD_Pre-Process (BgGrid)	109
Figure 62: VisualScript Element Definition – GridTest	110
Figure 63: Conditional (IF) Element Definition – GridTest	111
Figure 64: VisualScript Element Definition – CFX_CFD_Solver (BgSolve)	112
Figure 65: VisualScript Element Definition – CFX_CFD_Solver (BgExtract)	115
Figure 66: VisualScript Element Definition – SolutionTest	116
Figure 67: Conditional (IF) Statement Definition (SolutionTest)	117
Figure 68: VisualScript Element Definition – Extract_Solution	118
Figure 69: VisualScript Element Definition – Post_Process (Responses)	119
Figure 70: VisualScript Element Definition – Post_Process	120
Figure 71: VisualScript Response Transfer Order Specification	121
Figure 72: VisualScript Design Variables Transfer Order Specification	122
Figure 73: Detailed Components of MDO Environment	123
Figure 74: MDO Benchmark Fan Design Study – Base Model1	125
Figure 75: MDO Benchmark Fan Design Study – Base Model2	126
Figure 76: MDO Benchmark Fan Design Study – Base Model3	127
Figure 77: Model1 RSO Results – Target Function (Total Efficiency)	130
Figure 78: Model1 RSO Results – Bezier CP Tangential ($\beta_{r s}$) Coordinates	132
Figure 79: Model1 RSO Results – CP2 Merid Coord ($M'_{r s}$) and Displ. Factor (δ_r)	133
Figure 80: Model1 RSO Results – Blade Profiles/Passages	134
Figure 81: Model1 RSO Results – Circumferential LE Sweep (θ_r) Distribution	135
Figure 82: Model1 RSO Results – Blade Passage (with Throat Surface)	136
Figure 83: Model1 RSO Results – RPM Optimization History	137
Figure 84: Model1 RSO Results – # Blades Optimization History	138
Figure 85: Model1 RSO Results – Static Pressure, P_S (Mass Avg.)	139

Figure 86: Model1 RSO Results – Blade Loading (By Span Layer).....	140
Figure 87: Model1 RSO Results – Static Pressure (P_S) Contours	141
Figure 88: Model1 RSO Results – TE Deviation Angle (delta) Distribution.....	142
Figure 89: Model1 RSO Results – Relative Velocity (W) Vector Plots.....	143
Figure 90: Model1 RSO Results – LE Incidence Angle (i) Distribution.....	144
Figure 91: Model1 RSO Results – Relative Total Pressure (P_{Trel}) Contours	145
Figure 92: Model1 RSO Results – Fan Design CFD Analysis Summary	147
Figure 93: Model2 RSO Results – Target Function (Total Efficiency).....	148
Figure 94: Model2 RSO Results – Bezier CP Tangential ($\beta_{r s}$) Coordinates.....	150
Figure 95: Model2 RSO Results – CP2 Merid Coord ($M'_{r s}$) and Displ Factor (δ_r).....	151
Figure 96: Model2 RSO Results – Blade Profiles/Passages.....	152
Figure 97: Model2 RSO Results – Blade Passage (with Throat Surface)	153
Figure 98: Model2 RSO Results – LE Sweep (θ_r) Distribution	154
Figure 99: Model2 RSO Results – RPM Optimization History	155
Figure 100: Model2 RSO Results – # Blades Optimization History	156
Figure 101: Model2 RSO Results – Static Pressure, P_S (Mass Avg.)	157
Figure 102: Model2 RSO Results – Blade Loading (by Span Layer)	158
Figure 103: Model2 RSO Results – Static Pressure (P_S) Contours	159
Figure 104: Model2 RSO Results – Relative Velocity (W) Vector Plots	160
Figure 105: Model2 RSO Results – TE Deviation (delta) Angle Distribution.....	161
Figure 106: Model2 RSO Results – Relative Total Pressure (P_{Trel}) Contours	163
Figure 107: Model2 RSO Results – Fan Design CFD Analysis Summary	164
Figure 108: Model3 RSO Results – Target Function (Total Efficiency).....	165
Figure 109: Model3 RSO Results – Bezier CP Angular ($\beta_{r s}$) Coordinates.....	167
Figure 110: Model3 RSO Results – CP2 Merid Coord ($M'_{r s}$) and Displ Factor (δ_r).....	168
Figure 111: Model3 RSO Results – Displ Factor (δ_r) Perturbation History	169
Figure 112: Model3 RSO Results – Blade Profiles/Passages.....	170
Figure 113: Model3 RSO Results – LE Sweep (θ_r) Distribution	171
Figure 114: Model3 RSO Results – Blade Passages (with Throat Surface).....	172

Figure 115: Model3 RSO Results – RPM Optimization History	172
Figure 116: Model3 RSO Results - # Blades Optimization History.....	173
Figure 117: Model3 RSO Results – Static Pressure, P_S (Mass Avg.).....	174
Figure 118: Model3 RSO Results – Blade Loading (by Span Layer)	175
Figure 119: Model3 RSO Results – Static Pressure (P_S) Contours	177
Figure 120: Model3 RSO Results – Blade Rel. Velocity (W) Vector Plots.....	178
Figure 121: Model3 RSO Results – TE Deviation (δ) Angle Distribution.....	179
Figure 122: Model3 RSO Results – Relative Total Pressure (P_{Trel}) Contours	180
Figure 123: Model3 RSO Results – Fan Design CFD Analysis Summary	181
Figure 124: RSO Optimized Fan Design – CFD Based Performance Comparison	183
Figure 125: RSO Optimal Designs – Chord Length (c) Comparison.....	185
Figure 126: RSO Optimal Designs – Stagger Angle (ζ) Comparison	186
Figure 127: RSO Optimal Designs – Pitch-Chord Ratio (s/c) Comparison	186
Figure 128: RSO Optimal Designs – Solidity (c/s) Comparison.....	187
Figure 129: Fan Design with Highly Skewed Blade Geometry	190

NOMENCLATURE

- a_p - CFD solution residual normalization parameter
- B_n - Response surface model regression coefficient
- \bar{B} - Vector of response surface model regression coefficients
- c - Airfoil chord length
- c_1 - PSO particle self confidence parameter
- c_2 - PSO group/swarm confidence parameter
- c_p - Constant pressure specific heat
- C - True camber length/distance
- CP - Bezier control point
- e - Response surface model approximation error
- \bar{e} - Vector of response surface approximation errors
- f_μ - Proportionality constant (solver default = 0.01)
- h - Static enthalpy
- h_{ref} - Solver reference enthalpy state (0 J/kg)
- h_T - Total Enthalpy
- h_{Trel} - Relative total enthalpy
- $h_{T,rot}$ - Rotating frame total enthalpy (Rothalphy)
- h_{Tabs} - Total enthalpy in stationary frame
- $\hat{i}, \hat{j}, \hat{k}$ - Cartesian coordinate system unit vectors
- i - Rank of individual for reproductive fitness in genetic algorithm
- l_t - Zero equation turbulence model length scale
- k - Turbulent kinetic energy
- m - Genetic algorithm population size (number of designs)
- M - Meridional (axial) coordinate/length

M'	Normalized meridional coordinate ($M' = M/R$)
\underline{M}'	Normalized meridional coordinate in percentage (%)
N	Number of true response values (real analysis results) for RSO algorithm
p	Thermodynamic (static) pressure
p	Number of regression coefficients in response surface approximation model
\vec{p}	Best position vector of particle in PSO swarm
Pr_t	Turbulent Prandtl Number
PSO	Particle swarm optimization algorithm
$[\tilde{r}_\phi]$	Normalized residual of flow variable ϕ
$[r_\phi]$	Raw residual of flow variable ϕ
$r_{0,1}$	Particle swarm random parameters ($0 \leq r_{0,1} \leq 1$)
R_0	Universal gas constant
R	Fan radius
\vec{R}	Local radius vector
RSO	Response surface optimization algorithm
\vec{S}_E	Energy source term
\vec{S}_M	Momentum source term
S	Fractional distance along curve in CFX-Bladegen coordinate system ($0 \leq S \leq 1$)
t	Time
t	Blade airfoil thickness
T	Thermodynamic (static) temperature
T_{ref}	Solver reference state temperature (0 K)
u	Turbulent fluctuating flow property
U	Magnitude of velocity vector (speed)
\bar{U}	Reynolds-averaged flow property
\vec{U}	Velocity vector such that: $\vec{U} = U_x \hat{i} + U_y \hat{j} + U_z \hat{k}$

- \vec{U}_r - Velocity in rotating frame of reference
- \vec{U}_s - Velocity in stationary frame of reference
- U_t - Zero equation turbulence model velocity scale (max velocity in fluid domain)
- $U_{x,y,z}$ - Velocity components in x, y, z directions
- \vec{v} - PSO particle velocity vector
- w - PSO particle inertia parameter
- w - Molecular weight of the fluid
- V_D - Volume of fluid domain in zero equation turbulence model
- x - Response surface model design variable
- \vec{x} - PSO particle position vector
- \vec{X} - Response surface approximation model matrix
- y - True response analysis value in RSO algorithm
- \vec{y} - Vector of true response analysis values in RSO algorithm
- \hat{y} - Vector of approximated response analysis values
- \otimes - The tensor product of two vectors
- $(\nabla\vec{U})'$ - Transpose of resultant $(\nabla\vec{U})$ vector

Greek Symbols

- $\hat{\beta}$ - Advection blend/discretization order ($0 \leq \hat{\beta} \leq 1$)
- β - Bezier control point tangential angular coordinate
- $\hat{\delta}$ - The Kronecker Delta function (identity matrix) such that:
- δ - Bezier control point meridional coordinate displacement factor
- ∇ - Gradient vector/operator
- $\Delta\vec{r}$ - Vector from upwind node to nodal location n
- $\Delta\phi$ - Change in flow variable ϕ from previous iteration (time-step)
- Δt - Time step

ε	- Efficiency
Γ_t	Turbulent (eddy) diffusivity
ϕ_n	- Conserved property at nodal location n
ϕ_{up}	- Value of conserved property at upwind node
λ	- Thermal conductivity
μ	- Dynamic (molecular) viscosity
μ_t	- Turbulent (eddy) viscosity
θ	Blade leading edge circumferential angular coordinate
ρ	Fluid density
ω	Angular velocity (RPM)
ξ	- Airfoil stagger angle
ζ	Bulk viscosity

Subscripts

abs	absolute frame of reference
h	hub location
LE	- Leading edge
r	- Bezier control point span layer index ($1 \leq r \leq 5$)
rel	- relative frame of reference
s	Bezier control point meridional (axial) index ($1 \leq s \leq 4$)
S	- Static
t	- Turbulent
T	- Total
TE	- Trailing edge

1 INTRODUCTION

1.1 Background

Recent industry trends have seen the field of Computational Fluid Dynamics (CFD) extensively and successfully utilized in the design of various aerospace applications including aircraft wings, rotors, fuselage shapes, as well as turbomachinery to include compressor blades, propellers and turbine blades.

The intent of any engineering design endeavor is to develop the optimal solution to the specified problem. However, this objective is often subject to a multitude of competing and sometimes contradictory constraints and design requirements. As a result, an inordinate number of design cycles need to be repeatedly performed before the optimal design can be identified.

The overall objective of this investigation is to develop an alternative design methodology based on the integrated application of available design tools in a completely automated environment. A proper application of the design approach will ensure an efficient convergence to the optimal solution without significant input from the designer.

In establishing the requirements of a completely automated design methodology, a benchmark problem has to be selected as the basis on which the feasibility of the approach can be evaluated. This benchmark problem has to be representative of the level of design complexity faced by designers in the aerospace field, in order to sufficiently validate the applicability of a completely automated design approach. The benchmark problem chosen for this research project is the design of an optimal fan blade subject to several geometric and performance constraints.

1.2 Problem Statement

The objective is to create a completely automated design methodology that will be used to maximize the total efficiency of a given fan blade through the optimization of its blade geometry (shape). For the purpose of the particular benchmark case study, the design strategy is to account for given operational and performance constraints provided as follows:

- Volume Flow Rate: 10,000 ± 1000 [CFM]
- Static Pressure Rise: 0.5 [in. H2O], 0.125e⁻³ [MPa]
- Fan Diameter: 30.0 [in]

In this study, a set of “base” fan designs were generated using traditional turbomachinery design techniques. The focus will be on evolving from such base designs to more efficient blade geometries using a completely automated methodology employing non-traditional techniques. The optimization objective, the total efficiency (ϵ_T) is defined by:

$$\epsilon_T|_{LE-TE} = \left[\frac{\partial P_{T_{abs}}}{\partial P_{T_{abs}} + \partial P_{T_{rel}}} \right]_{mass_{avg}} \quad (1.1)$$

Where: $\partial P_{T_{abs}}$ - Change in the total pressure in the absolute reference frame
 $\partial P_{T_{rel}}$ - Change in the total pressure in the relative reference frame

The total efficiency is essentially a measure of the change in the total pressure in the relative (rotating) frame of reference. In an ideal machine with 100% total efficiency, the total pressure is conserved in the relative frame of reference, yielding a total change of zero from the leading edge to the trailing edge of the blade. In practice, losses are always present but the hallmark of a highly efficient design methodology is to minimize these losses. It is to this end that the automated MDO based design methodology is being employed.

1.3 Approach

The problem statement requires that “base” designs suggested using traditional design techniques are modified to obtain an optimal solution to the design problem. This thesis focuses on developing a methodology involving the proper parameterization of the blade geometry and the application of a completely automated iterative process in achieving the optimal design.

It becomes necessary to first identify the components necessary to build a design environment that is not only completely automated, but also robust enough to tolerate any potential discrepancies between mathematically optimal solutions and practically feasible solutions. The essential elements (functions) of the automated design environment specific to this problem can be identified as:

1. Blade Geometry Modeling
2. Design Evaluation
3. Optimization based on Results of Design

For the modeling of the Blade Geometry, the *CFX-Bladegen* utility from *ANSYS Inc.* is selected due to its ability to model various classifications of turbomachinery including fans, axial and radial compressors, turbines, etc.

The application of a Computational Fluid Dynamics (CFD) solver becomes essential when various blade designs need to be evaluated for performance e.g., in terms of the total efficiency (as calculated in equation 1.1). Although CFD has not yet evolved to the stage where it may be considered unquestionably accurate, it is a useful tool in estimating the performance of various blade designs and can often be trusted to realistically reflect an improvement in fan efficiency, obtained for a corresponding optimal modification of the blade geometry.

The CFD solver selection is critical to this particular benchmark problem, as it must be able to automatically and rapidly handle all the tasks critical to performing a complete CFD analyses. These include pre-processing, grid generation and post-processing of the flow field solution. The CFD solver chosen for the benchmark problem is the *CFX-Bladegen(Plus)* flow solver, which is available as an add-on module to the CFX-Bladegen turbomachinery modeling utility. The selection of the flow solver is based on its ability to perform relatively rapid analyses of blade designs.

The final necessary component for the automated design methodology is the optimization of the blade geometry based on reported results of the CFD analyses. This component modifies the various design parameters as dictated by the optimization algorithm, subject to the design constraints imposed on the problem. For the benchmark case, the selected optimization tool is the *VisualDoc* Multidisciplinary Optimization (MDO) utility from *Vanderplaats Research and Development, Inc., Colorado*. This utility allows for the efficient integration of all employed design and analysis tools in a completely automated manner using a graphics based programming interface, as illustrated in Figure 1.

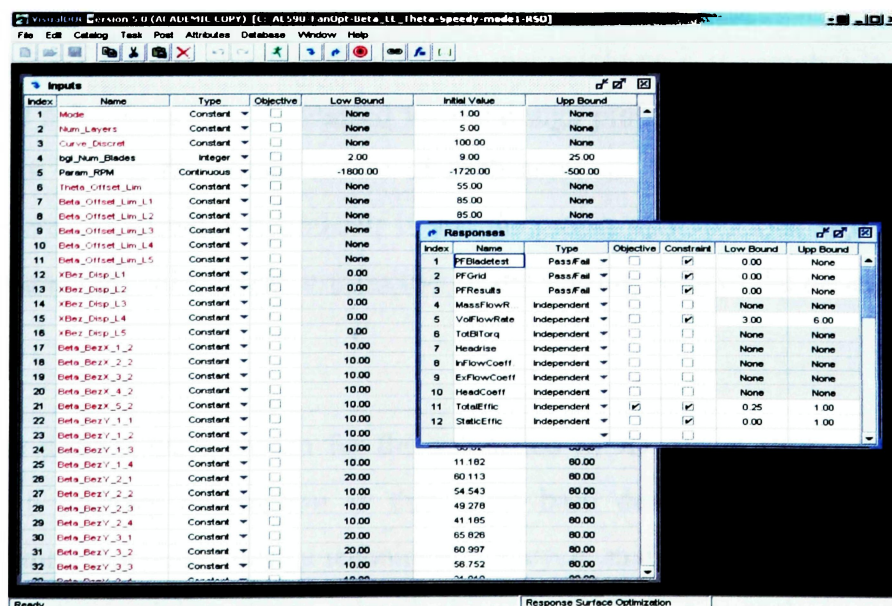


Figure 1: VisualDoc Graphical User Interface

A generalized model to illustrate the interaction between the various components of the automated design approach is presented in Figure 2. Essentially, these components will iteratively generate and evaluate several blade designs until the optimization algorithm identifies the MDO analysis as having converged to the optimal (best) design possible.

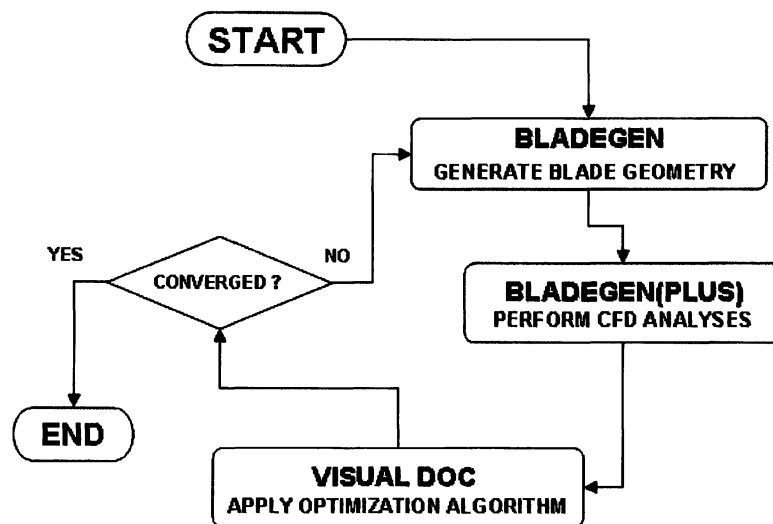


Figure 2: Interaction Flowchart for MDO Design Components

In developing a completely automated design methodology, it is essential to verify that the optimal solution is indeed obtained for the design problem. Hence, the investigation will attempt to test the *global* nature of such solutions by using four different base designs. Here, the essential hypothesis is that the proper optimizations algorithm should yield the same optimal solution irrespective of the starting point in the parametric design space.

Three base designs were chosen for the automated MDO design study. As previously indicated, a primary assumption is that these base designs were developed using traditional turbomachinery design techniques, i.e. cycle analyses, blade angle calculations and blade vortex modeling. Table 1 details the various parameters of the base fan designs for the MDO based fan design optimization study.

The fan models for the base designs developed in CFX-Bladegen are shown in Figure 3. In creating the base designs, the position of the model inlet and outlet planes is located by offsetting the plane at an equivalent distance from the central plane of the fan as shown in Figure 3. An extrapolation of the blade shape through five spanwise constant-radius layers is subsequently used to generate the complete three-dimensional geometry. A comprehensive description of the blade geometry generation process, the Bladegen coordinate system and the various blade views is presented in section 2.1.

Table 1: Base Fan Designs for MDO Optimization

		Model1	Model2	Model3
N (RPM)		1140	1720	1140
Diameter, D	(in.)	30	24.3	30
	(mm)	762	617.33	762
# Blades		9	9	9
R_h/R_{shr}		0.4	0.4	0.45
c_{shr}/c_h		2	2	1.88
Blade Vortex Model		-1	-1	0.75
t_{max}/c		0	0	0
t_{LE}/t_{TE}		4/1	4/1	4/1
h_{LE} Incidence Angle (°)		2.5	2.5	2.5
shr_{LE} Incidence Angle (°)		-2.5	2.5	2.5
h_{TE} Deviation Angle (°)		5	5	5
shr_{TE} Deviation Angle (°)		5	10	10
Camber Load		Aft Tip and Mid Load	Aft Tip Load	Aft Hub Load

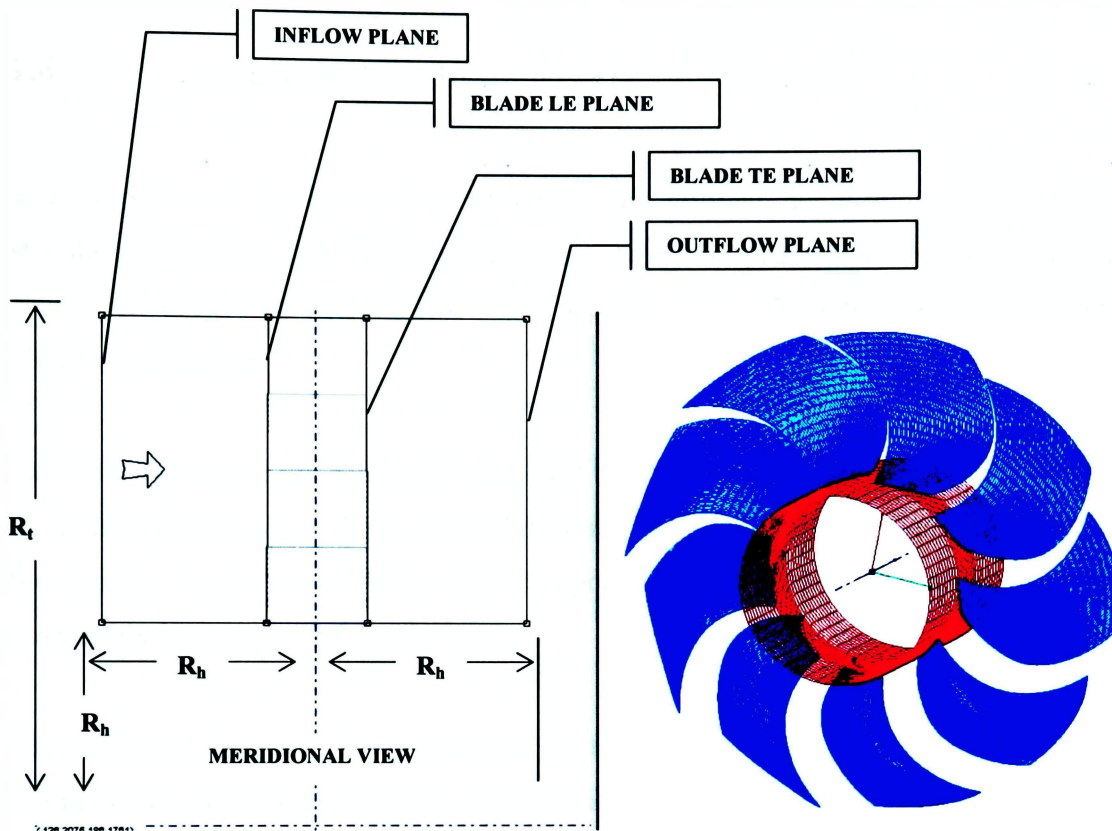


Figure 3: MDO Design Base Model1

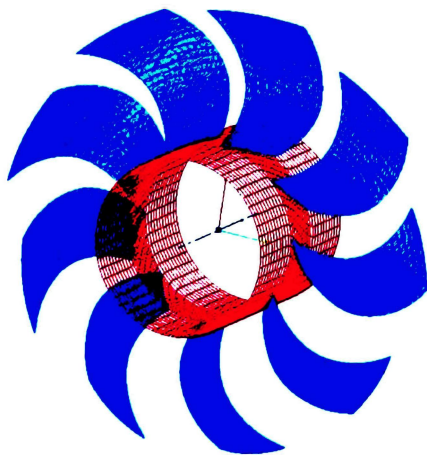


Figure 4: MDO Design Base Model2

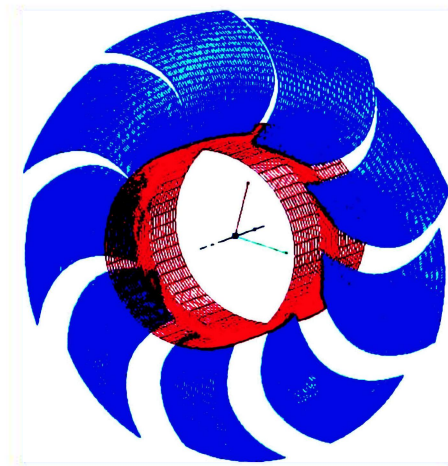


Figure 5: MDO Design Base Model3

1.4 Expected Results

The objective of the research is to obtain the optimal fan design for the problem statement of section 1.2. The approach to solving the problem involves developing a completely automated environment which will utilize a mathematical *search* algorithm to obtain this solution.

The effectiveness of the MDO approach will be validated through the use of three “base” or starting point designs. Truly optimal designs should be similar in nature irrespective of the starting point, provided the search algorithm is proficient and robust enough to thoroughly investigate the design space.

The optimal design is defined in term of the total efficiency of the fan. This is calculated as previously mentioned, using the relationship in equation 1.2 below:

$$\mathcal{E}_T|_{LE-TE} = \left[\frac{\partial P_{T_{abs}}}{\partial P_{T_{abs}} + \partial P_{T_{rel}}} \right]_{mass_{avg}} \quad (1.2)$$

Equation 1.2 is essentially a measure of the change in the total pressure in the relative or (rotational) plane, $\delta P_{T_{rel}}$. For a theoretically perfect design, there should be no change in the total pressure in the relative frame of reference i.e., $\delta P_{T_{rel}} = 0$. Although it is not expected that the MDO approach will yield a fan with 100% efficiency, it is expected that there will be significant improvement in the efficiency of the design subject to the given problem constraints.

2 METHODOLOGY

2.1 Blade Geometry Generation

The selected blade geometry design utility has been previously identified as the CFX-Bladegen turbomachinery modeling utility from ANSYS Inc. This particular utility is selected due to its versatility in modeling a wide range of turbomachinery classifications including axial and radial turbines, compressors, stators, swirl vanes and diffusers. In this section, a detailed description of several Bladegen features is given. The features discussed are those directly applicable to the development of the MDO environment for the benchmark study.

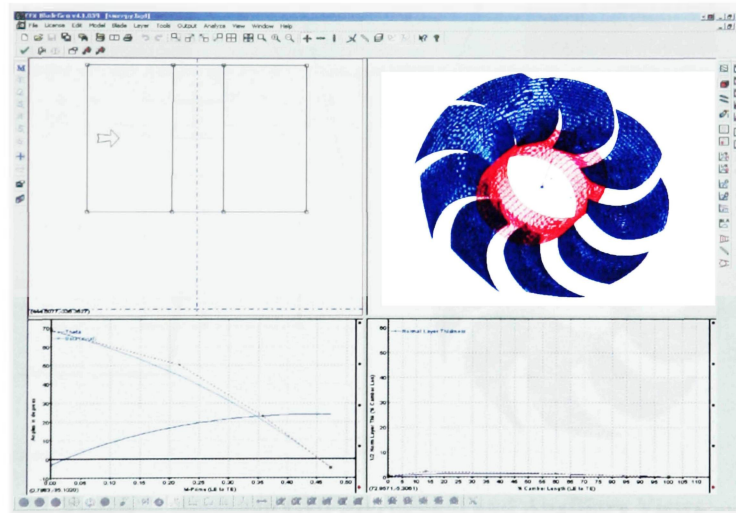


Figure 6: CFX-Bladegen Graphical Interface

CFX-Bladegen provides the designer with the option to operate in two distinct modes namely the *Angle/Thickness (Ang/Thk)* mode and the *Pressure/Suction (Prs/Sct)* mode. Typically, radial turbomachinery components are designed in the *Ang/Thk* mode while axial components are designed in the *Prs/Sct* mode. However it is possible to switch the modes in order to facilitate easier manipulation and analyses of designs. The primary mode used in the benchmark MDO study is the *Ang/Thk* mode. The primary difference between the design modes is the parameters used to describe

the blade geometry. In the *Prs/Sct* mode, the designer directly modifies the airfoil shapes at various constant-radius layers along the blade. In the *Ang/Thk* mode, the airfoil shapes are generated from blade angle distributions at several constant-radius planes, referred to as span layers.

The *Ang/Thk* and *Prs/Sct* modes use a common set of views in the graphical user interface. These views as detailed in Figure 7, are a meridional view and an auxiliary view of the blade design and they constitute the upper half of the standard Bladegen GUI of Figure 6. The meridional view defines the blade design in radial versus axial space while the auxiliary view provides a variety of views to include 3D, meridional contour and blade-to-blade views as well as plots of several blade parameters.

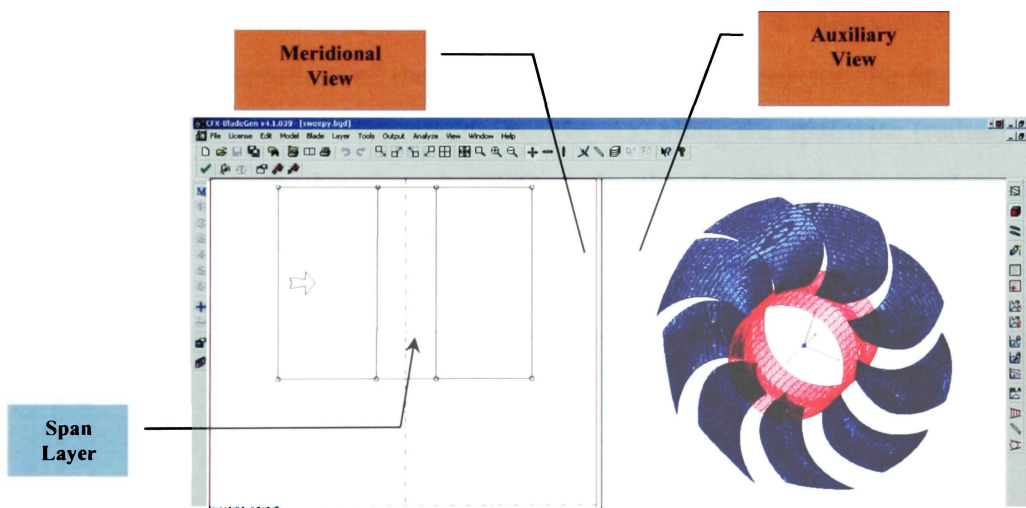


Figure 7: Bladegen GUI Meridional and Auxiliary Views

The *Prs/Sct* mode provides a Section View of the blade profile at several span layers spanning the blade from hub to shroud. A sample *Prs/Sct* view is shown in Figure 8 and is usually located in the bottom half of the Bladegen GUI of Figure 6. The *Prs/Sct* view provides the designer with the ability to *directly* manipulate the pressure and suction sides of the blade profile to achieve the desired shape.

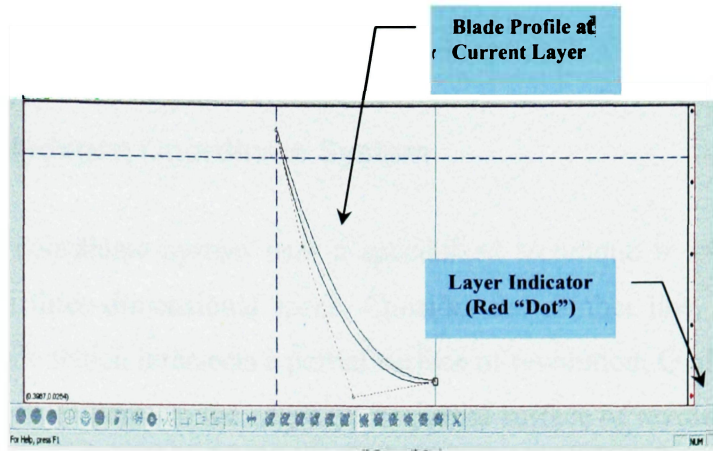


Figure 8: Pressure/Suction Mode Blade Layer View

In the *Ang/Thk* mode, the two views in Figure 9 are added in the bottom half of the window and with the common views of Figure 7, they form the complete Bladegen GUI of Figure 6. The additional views in the *Ang/Thk* mode include an *Angle View* detailing the distributions of various blade angles and a *Thickness View* detailing the thickness distribution of the blade profile. Bladegen provide the various views in all modes at several discrete streamlines (layers) along the spanwise (radial) direction of the blade. For the benchmark MDO study, the *Ang/Thk* design mode is used in the generation of the blade geometry.

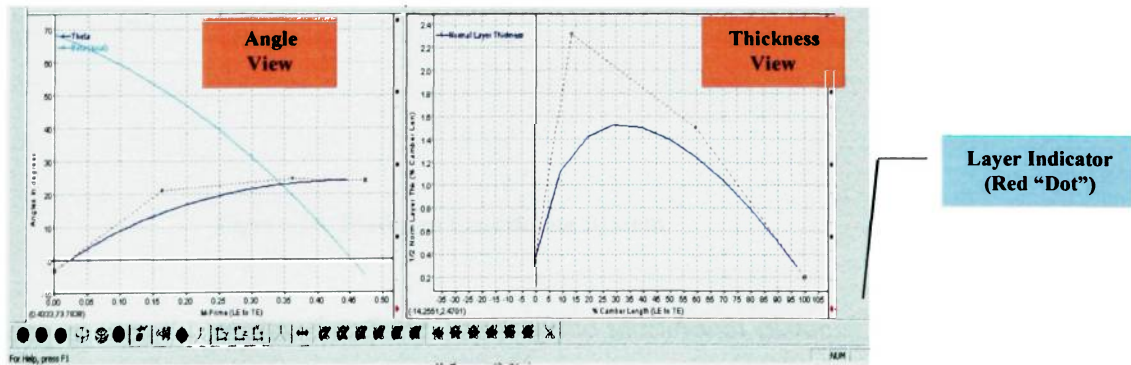


Figure 9: Angle/Thickness Mode Blade Layer View

In order to clearly describe the blade geometry design process, as well as understand the information contained in the Graphical User Interface views of Figure 7 through Figure 9, it is essential that the details of the Bladegen coordinate system be discussed.

2.1.1 CFX-Bladegen Coordinate System

The Bladegen coordinate system uses a specialized technique in order to locate a design point in three-dimensional space. Consider the camber line, P of the airfoil section of a blade which intersects a partial surface of revolution, Q of constant radius R. The camber line intersects the edges of the partial surface of revolution at points A and B:

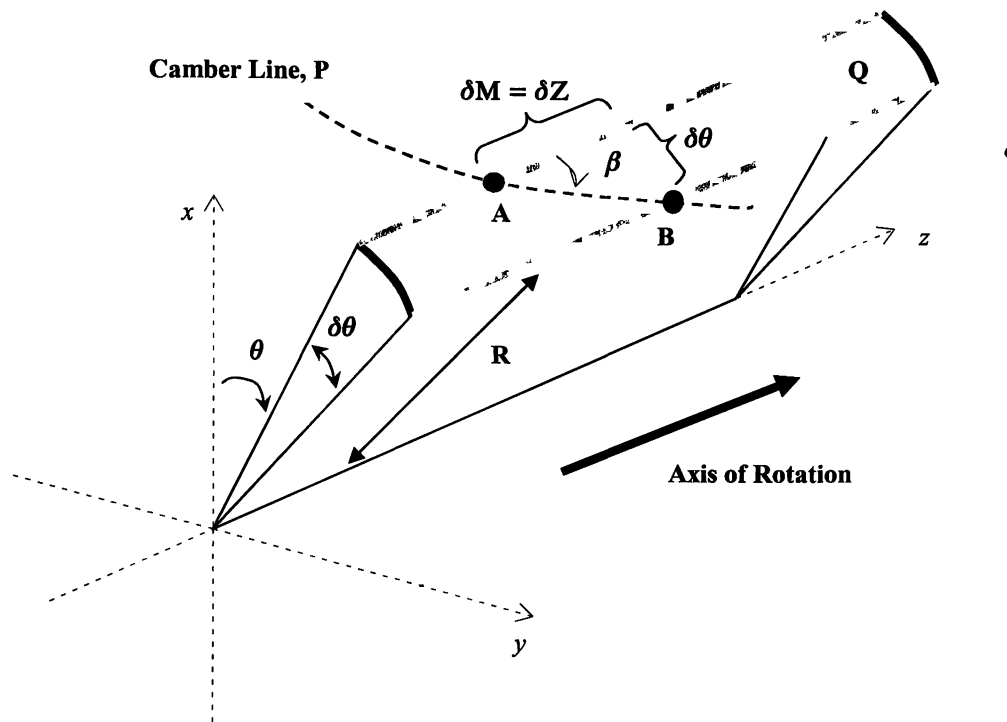


Figure 10: Bladegen Coordinate System

CFX-Bladegen uses the angles β and θ as well as the meridional distance, M along the curve to locate the points, A and B. The meridional distance along the curve is determined by integrating the differential meridional distance over the length of the curve (equation 2.2):

$$\partial M = \sqrt{\partial R \cdot \partial R + \partial Z \cdot \partial Z} \quad (2.1)$$

Where: ∂M - Differential meridional displacement
 ∂R - Differential radial displacement
 ∂Z - Differential axial displacement

The true meridional coordinate of any point (such as A and B) can be computed as the integral of the differential meridional distance:

$$M = \int_0^s \partial M \cdot dS \quad (2.2)$$

Where: S - Fractional distance along the curve $0 \leq S \leq 1$

The meridional coordinate essentially ignores the circumferential displacement between succeeding points as can be seen from equation 2.1. As a result, when the blade geometry is viewed on meridional plane, the appearance of the chordwise curvature (camber) of the blade is skewed and generally appears flat. An example of this can be seen in Figure 7 where the blade model in the auxiliary 3D view possesses considerable curvature but appears as a rectangle when viewed in the Meridional View window.

From a closer observation of equation 2.1, it follows that for a constant-radius surface of revolution (span layer), incrementing the meridional coordinate is equivalent to incrementing the axial or Z-coordinate, since there is no change in the radial coordinate along a span layer. Thus, for any two points on the same “span layer” (surface of constant radius), $\delta R = 0$.

$$\begin{aligned} \partial M &= \sqrt{\partial R \cdot \partial R + \partial Z \cdot \partial Z} \\ \partial M &= \sqrt{\partial Z \cdot \partial Z} = \partial Z \end{aligned} \quad (2.3)$$

The meridional length of the airfoil, M is related to the chord length of the airfoil through the stagger angle (ξ). Using the stagger angle which is a traditional turbomachinery design parameter, the chord length (c) of the airfoil is related to its total meridional length (M) as follows:

$$c = \frac{M}{\cos(\xi)} \quad (2.4)$$

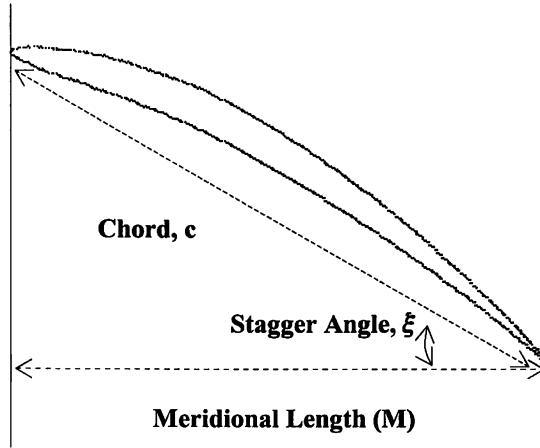


Figure 11: Chord-Meridional Length Relationship

The camberline, P of Figure 10 is also referred to as the meanline in Bladegen coordinate system terminology. The true length of the camber is calculated by integrating the differential change in distance along the meanline using the relationship:

$$\partial C = \sqrt{\partial X \cdot \partial X + \partial Y \cdot \partial Y + \partial Z \cdot \partial Z}$$

$$C = \int_0^s \partial C \cdot \partial S \quad (2.5)$$

Where:

C True camber length

The circumferential sweep angle, θ in Figure 10 is defined as the rotation around the z-axis increasing in the direction originating from the x-axis towards the y-axis using the right hand rule. The θ coordinate is used to account for the circumferential displacement between succeeding points on the meanline; a quantity ignored in the meridional coordinate relationship (equation 2.1).

Using a small angle approximation, the angular change between two succeeding points in the circumferential plane, is the same as the linear displacement between them if they lie on the same surface of revolution (ref. Figure 10). This observation, when combined with equation 2.3 results in the introduction of the second angular coordinate (β) which is shown in Figure 10. The tangential angular coordinate, β is defined by the relationship:

$$\beta = \tan^{-1} \left(\frac{\partial \theta}{\partial M'} \right) \quad (2.6)$$

Where: $\delta M'$ is the meridional coordinate (δM) normalized by the radial coordinate, R . Thus:

$$\delta M' = \frac{\delta M}{R} \quad (2.7)$$

The β coordinate is consequently the *local relative* angular offset from one point to the next along the meanline of the blade profile (airfoil). As a result, there are four essential elements used to describe the location of a point in 3D space using the Bladegen coordinate system. They are the θ and β angular coordinates of the point, the radial location (R) and the axial location which is described by the meridional coordinate (M) or its non-dimensionalized form (M'). The two blade angles β and θ are related by equation 2.6. Providing one of the angular coordinates is sufficient to deduce the other if the axial location (meridional coordinate) is known.

In generating the blade geometry, these four coordinates are used to locate a series of points in three-dimensional space that constitute the camberline at specific planes of constant radius (span layers). These layers are spread out along the span of the blade from hub to shroud as in Figure 7. The airfoil shape is then completed by imposing a thickness distribution along the camberline. The final three-dimensional blade is generated by interpolating between the blade profiles at all the span layers. For the benchmark case, a general radial interpolation is used to develop the 3D blade geometry from the blade profiles.

The plots displayed in the bottom half of the *Ang/Thk* views of Figure 9 can now be described as simply the β - M' distribution (left hand side) and the blade profile thickness distribution (right hand side) at the specific span layer being designed. A sample β - M' plot is also shown in Figure 12. The normalized meridional M' is specified as a percentage with 0% corresponding to the LE position and 100% corresponding to the TE position. The percentage form of the normalized meridional is used in the benchmark study for convenience and the \underline{M}' nomenclature is used to reflect this change in the form of the meridional coordinate being used.

The blue curve in the plot specifies the leading to trailing edge β - \underline{M}' distribution while the red line specifies the θ - \underline{M}' distribution. These coordinate distributions are used, as earlier discussed, to generate the camberline of the blade profile (airfoil) shape at the respective spanwise layers. The current span layer is indicated by the “red dot” on the right hand side of the plot window as also seen in Figure 8 and Figure 9. The curves in Figure 12 show the angle distribution curves for the 0% span layer (hub).

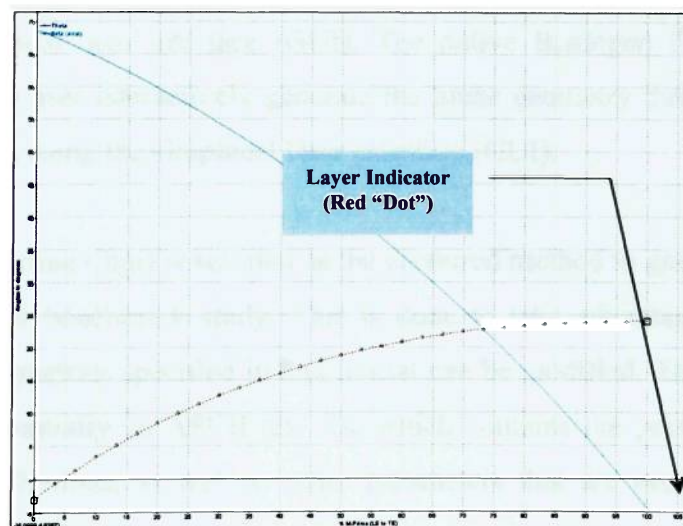


Figure 12: β - M' and θ - M' Sample Distributions

In the MDO approach, it is necessary to describe the 3D geometry of the blade in terms of design parameters or variables. The preceding discussion of how the blade geometry is generated is especially important, as it forms the foundation from which we begin to consider how the blade geometry may be efficiently parameterized for proper implementation in a completely automated MDO environment. Subsequent sections will describe how the details of the Bladegen coordinate system are harnessed in developing a proper parameterization scheme for the 3D blade geometry.

2.1.2 Blade Geometry File Format

CFX-Bladegen provides the user with a multitude of file formats for providing the blade geometry coordinates discussed in the previous section. This creates the ability to describe almost any blade geometry as a function of the design mode being implemented.

The primary CFX input file formats pertinent to the benchmark case are the Batch File Format (.bgi) and the Bladegen native geometry file format (.bgd). The screen capture of Figure 6 shows a native Bladegen file format in the context of the CFX-Bladegen graphical user interface (GUI). The native Bladegen file format (.bgd) requires that the user interactively generate the blade geometry through a system of menu selections, using the Graphical User Interface (GUI).

The batch file format (.bgi) is selected as the preferred method in generating the blade geometry for the benchmark study. This is done to take advantage of the ease in which blade geometries specified in this format can be modified. The CFX-Bladegen batch file is essentially an ASCII text file which contains the previously discussed coordinate distributions, as well as other parameters that are used to generate the three-dimensional blade geometry.

The ability to create and modify the batch file using a simple text editor is essential in selecting the batch file format as the standard blade geometry design format to be utilized in the automated MDO design methodology. The batch file blade design file format is primarily composed of sections which are specified in a predetermined order and used to describe various properties of the blade model. Table 2 provides a brief description of the various sections of the Bladegen batch file format. Figure 13 details the nomenclature used in specifying the attributes of the model in the meridional view.

Figure 14 provides screenshots of several sections of a sample blade geometry batch (.bgi) file.

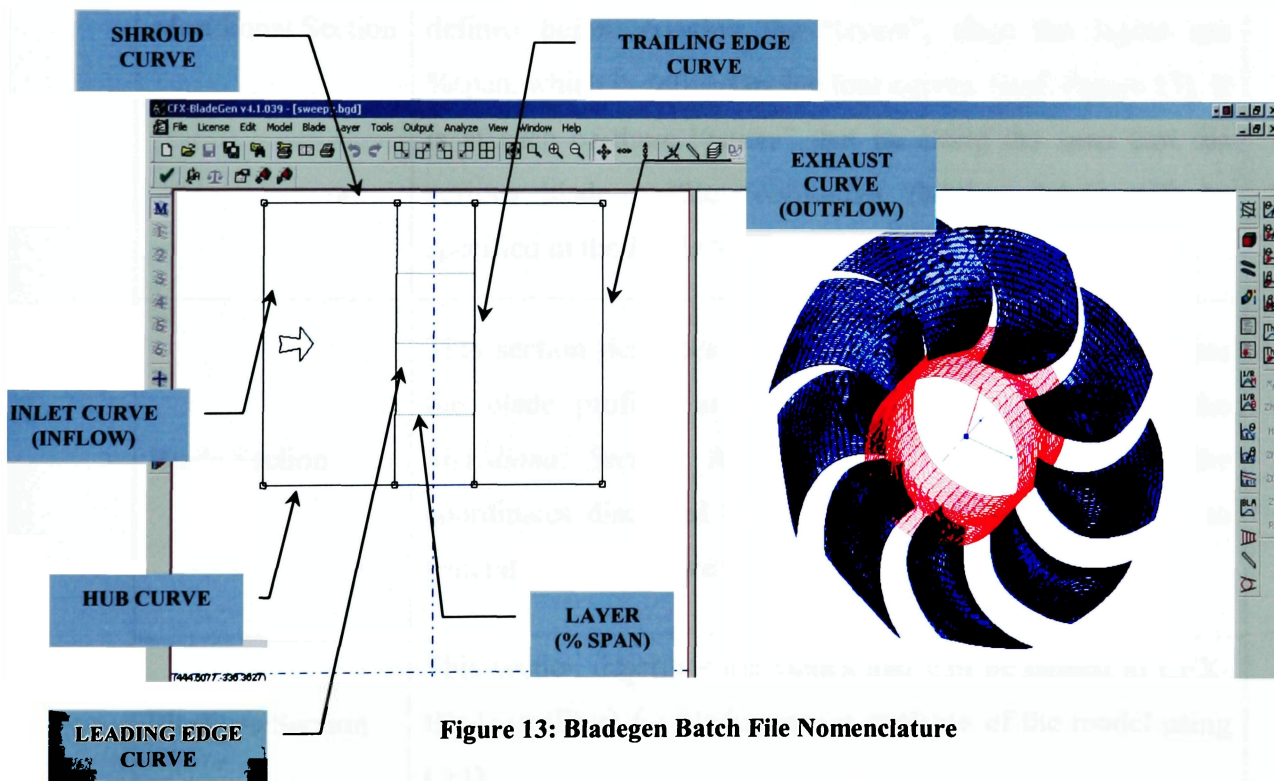


Figure 13: Bladegen Batch File Nomenclature

Table 2: Bladegen Batch File Sections

Section	Function
Model Section	This section describes the parameters that apply to the model as a whole. For example, the number of blades, number of points for blade coordinate distributions
Meridional Section	This section describes the meridional definition of the model. The hub and shroud curves must be defined before the leading, and trailing edge curves since these two curves reference the prior two curves for their end points. All of the curves must be defined before creating the “layers”, since the layers use %span, which is defined by the four curves. (Ref. Figure 13). It is at each of these “layers” that lie along the span that the various blade profile coordinates (section 2.1.1) will be specified in the <i>Blade Section</i> below.
Blade Section	This section describes the coordinates necessary to generate the blade profiles at each of the layers defined in the <i>Meridional Section</i> . At each layer along the span all the coordinates discussed in section 2.1.1 must be provided to generate the respective blade profiles.
PlusData Section	This section describes the values that will be passed to CFX-Bladegen(Plus) for blade passage analyses of the model using CFD.

```

Begin Model
NumMainBlades=3
Mode=Angle/Thickness
RightHandedCoordSystem

BladeOutputPointClustering=Linear
NumBladePoints=54
NumLeadingEdgePoints=12
NumTrailingEdgePoints=12

CurveDisplayMaximumError=0.02133670000

DataFromLeToTe
BetaAxialDef
ThicknessIsDimension

MeridionalSpanCurveRuledShape=RuledElement

Designer="idahosau"
Company=""
Comment=""
DeviceType=Fan
ConfigurationType=Axial
RotationType=Negative
GeometryUnits=MM

MODEL SECTION
End Model

```

```

MERIDIONAL SECTION
Begin Meridional
MeridionalControlCurveMode=Minimal
SpanByGeom
Begin HubCurve
New Segment
CurveType=Bezier
UpstreamControl=Free
Begin Data
( -152.8000000,152.0000000 )
( -35.0000000,152.0000000 )
End Data
DownstreamControl=Free
End Segment
New Segment
CurveType=Bezier
UpstreamControl=Free
Begin Data
( -35.0000000,152.0000000 )
( 36.9157000,152.0000000 )
End Data
DownstreamControl=Free
End Segment
New Segment
CurveType=Bezier
UpstreamControl=Free
Begin Data
( 36.9157000,152.0000000 )
( 152.0000000,152.0000000 )
End Data
DownstreamControl=Free
End Segment
End HubCurve

```

```

New Blade
PitchFraction=0.000000000
LeadingEdgeEndType=Ellipse
HubLE_EllipseRatio=4.000000000
ShrLE_EllipseRatio=4.000000000
TrailingEdgeEndType=Ellipse
HubTE_EllipseRatio=1.000000000
ShrTE_EllipseRatio=1.000000000
EllipticArcMean=T

Begin AngleDefinition
AngleLocation=MeanLine
SpanwiseDistribution=General

New AngleCurve
Layer="Layer1"
DefinitionType=BetaCurve
HorizDim=PercentMeridionalPrime
VertDim=Degree

LE_Theta=-1.000000000
New Segment
CurveType=Bezier
UpstreamControl=Free
Begin Data
( 0.000000000,88.74830307 )
( 45.67735006,50.35778979 )
( 78.72794937,22.31020639 )
( 100.0000000,-4.549213767 )
End Data
DownstreamControl=Free
End Segment
End AngleCurve

```

```

Begin PlusData
Begin Case
Comments = ".bgi file configured for optimization RPM = -1140 "
Machine Type = fan
Component Type = rotor
Units = MM
End Case
End PlusData

PLUSDATA SECTION

```

Figure 14: Batch Input File Section Samples

2.2 Design Evaluation Using Computational Fluid Dynamics

CFX-Bladegen includes a tool for rapidly evaluating the performance of the blade model. This capability is provided in the *Plus* module which is integrated as part of the Bladegen package. The *Plus* module is generally run under the graphical user interface, and is composed of “panels” (Figure 15) which define the various stages needed to carry out CFD based performance analyses of the blade model. Command line utilities are also provided that allow the designer to run a completely automated CFD analysis in batch mode without any user interaction. This particular capability is highly significant in selecting CFX as the design and analysis tool utilized in the automated design environment for the benchmark case.

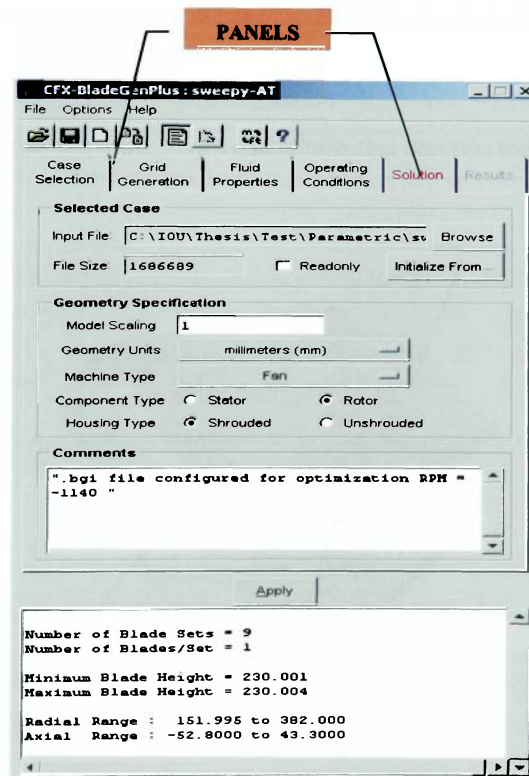


Figure 15: CFX-Bladegen(Plus) Graphical User Interface

The subsequent sections briefly discuss each panel and its functions in setting up the CFD blade passage analyses for the performance evaluation of the model. It is important to note that although Bladegen provides for the specification of the number of blades in the complete model, the CFD analysis is typically performed only on a single blade passage as depicted in Figure 16. It is the size and geometry of the blade passage that is determined from the number of blades. The rotational speed of the model is used to determine the behavior of the periodic boundaries that simulate the rotation of the complete turbomachinery model.

Table 3: Bladegen(Plus) GUI Panels – Functions

PANEL	FUNCTION
Case Selection	Sets the geometry type and units for proper conversion
Grid Generation	Automatically creates unstructured computational mesh over the blade passage
Fluid Properties	Defines the properties of the fluid
Operating Conditions	Sets the specific flow conditions
Solution	Generates the flow field solution
Results	Generates plots, tables and reports that describe the results

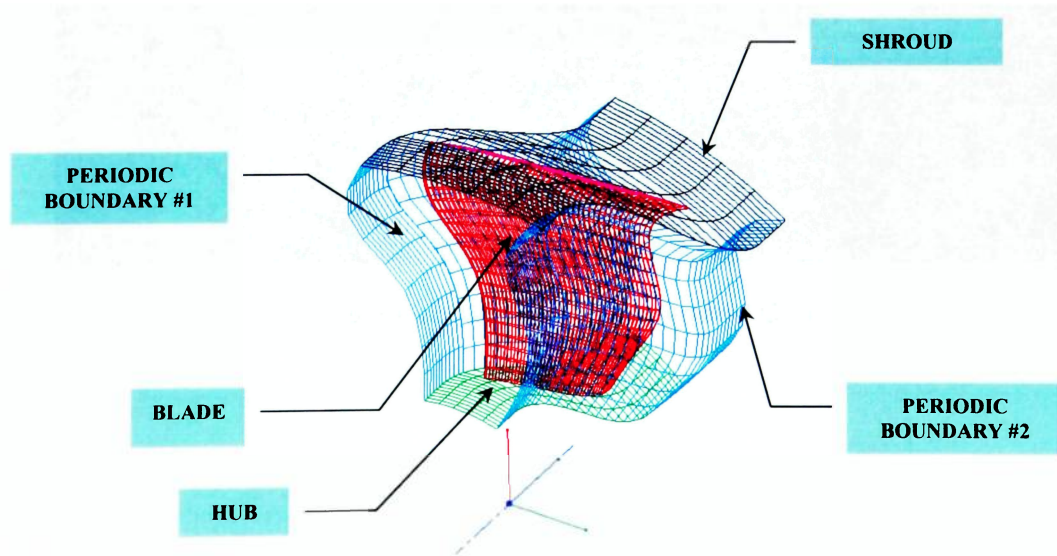


Figure 16: Blade Passage for Fluid Flow Analyses

The functions of the different panels of the Bladegen(Plus) graphical user interface are discussed in Table 3. The ease with which the CFD analyses can be set up allows for the rapid evaluation of the performance of the blade model. Furthermore, all the functionality embodied in the CFX-Bladegen(Plus) GUI is also available using batch utilities. This allows the user to carry out a complete CFD analyses on a model using only command line instructions. This capability is especially useful in the automated design and analysis methodology for the benchmark case.

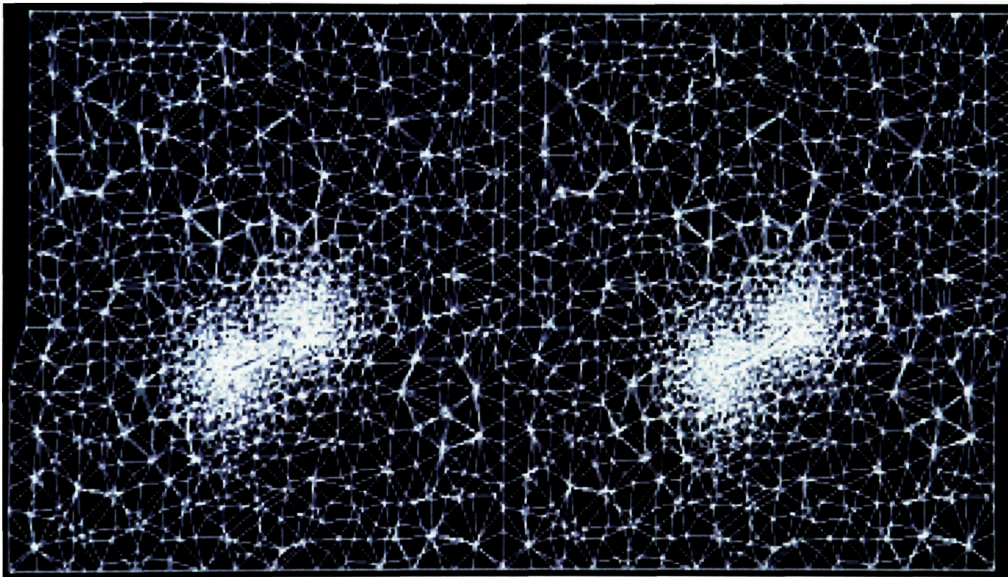


Figure 17: Blade Passage Unstructured Grid

2.2.1 CFX-Bladegen(Plus) Batch Utilities

It is necessary to briefly discuss the utilities that allow for the command line execution of all the steps required to run a CFD analyses using the CFX-Bladegen *Plus* module. These command line tools are provided with the CFX-Bladegen package and are matched with their respective graphical interface panels for comparison. They are presented in the order in which they are employed to carry out the pre-processing, solution and post-processing stages of a complete CFD analysis of the blade model.

2.2.1.a CFD Pre-Processing:

Bladebatch: This utility is used to convert the blade model format from the batch input file format (.bgi) to the CFX-Bladegen native blade geometry file format (.bgd).

BgBatch: Merges the model geometry output from the *Bladebatch* utility with the operating conditions of the model as well as freestream condition parameters supplied in a *parameter file*. All the information is consolidated and used to create a fluid flow analyses problem file (.bg+).

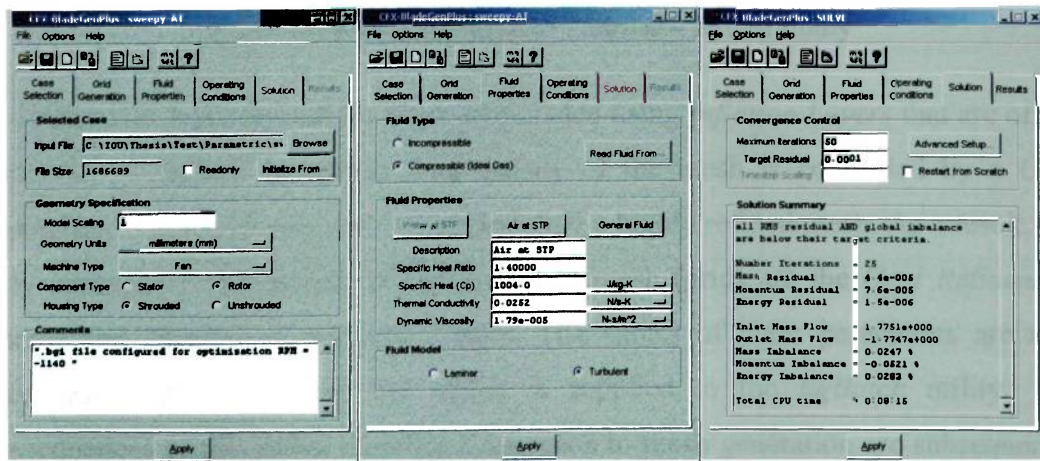


Figure 18: Bladegen(Plus) GUI Functions – *BgBatch* Utility

The *BgBatch* utility combines the functionality contained in the *Case Selection*, *Fluid Properties* and *Solution* panels of the Bladegen(Plus) GUI shown in Figure 18. The primary advantage of this utility is that trade studies may be carried out by running the blade geometry through a multitude of operating and freestream configurations simply by changing the specifications in the *parameter file*.

2.2.1.b Mesh Generation

BgGrid: This utility generates an unstructured grid for the single blade passage of Figure 16. This utility represents the function of the Grid Generation panel of the Bladegen(Plus) GUI (Figure 19).

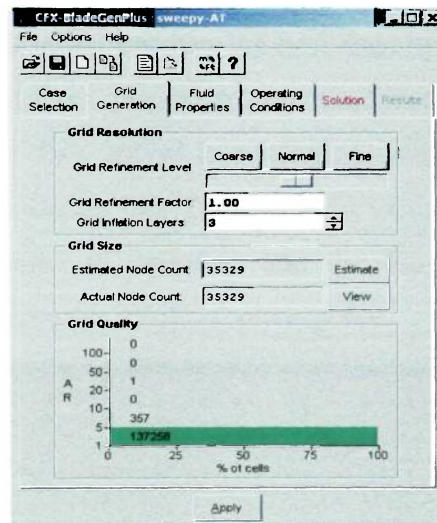


Figure 19: Bladegen(Plus) GUI Function – BgGrid Utility

The boundary layer around the blade is modeled using *Inflation Layers* that are offset from the surface of the blade. The density of the mesh is controlled by a *Grid Refinement Factor* (0-10), with 10 representing a highly refined mesh. As a result, the only controls on the quality of the generated mesh (grid) are the *Grid Refinement Factor* and *Number of Inflation Layers*. The values of these parameters are also specified in the *parameter file* which is supplied to the *BgBatch* utility. The consequences of this rather simplified approach to mesh generation are subsequently discussed.

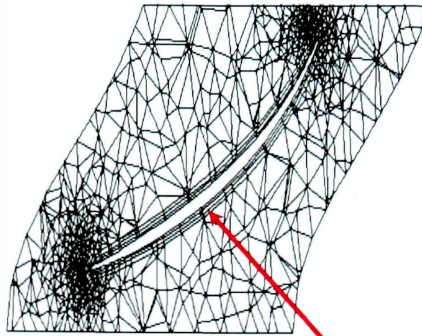


Figure 20: Unstructured Grid Detailing Inflation INFLATION LAYERS

2.2.1.c Flow Field Solution

BgSolve:

This utility resolves the flow field by solving the Navier-Stokes Equations over the unstructured mesh from the *BgGrid* Utility. The flow solver is controlled using variables defined in the parameters file. These control variables include the *Maximum number of iterations* and the *Target Residual* value which determines solution convergence.

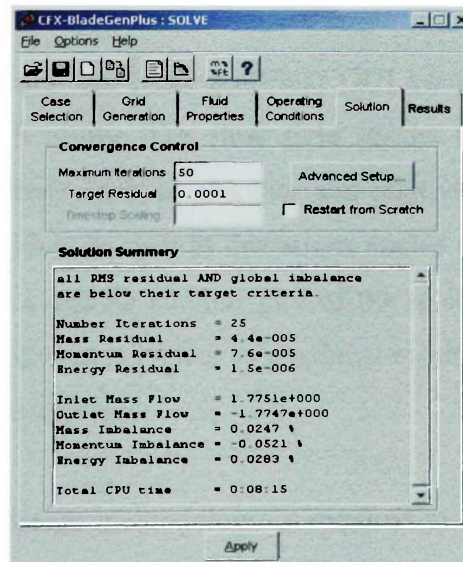


Figure 21: Bladegen(Plus) GUI Function – *BgSolve* Utility

2.2.1.d Post-Processing

BgExtract: This utility is used to extract results computed from the CFD analysis of the fan blade passage. The computed results include total and static efficiency, head coefficient, pressure rise as well as volumetric flow rate.

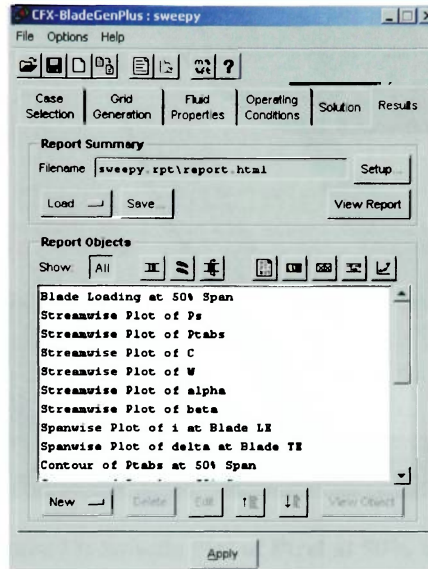


Figure 22: Bladegen(Plus) GUI Function – *BgExtract* Utility

CFD results obtained using the Bladegen(Plus) GUI contain more data than those obtained from the batch utility. The flow field results typically include plots of various performance criteria such as absolute and relative total pressure (P_{Tabs} and P_{Trel}) contours, blade loading as well as deviations angles at the trailing edges of the blade. Figure 23 shows a sample contour plot of the relative total pressure at 50% span location. The contour plot is typical of graphical results obtained using the CFX-Bladegen(Plus) graphical interface.

Since the MDO based design methodology is required to be completely automated, the results that need to be produced from the CFD solver must require relatively little input/manipulation on the part of the user. It is this requirement that the BgExtract utility fulfills. It only extracts results in data format and leaves the graphical post

processing to the user. The main advantage of this feature is that fan performance data can be transferred to subsequent MDO components with minimal input from the user.

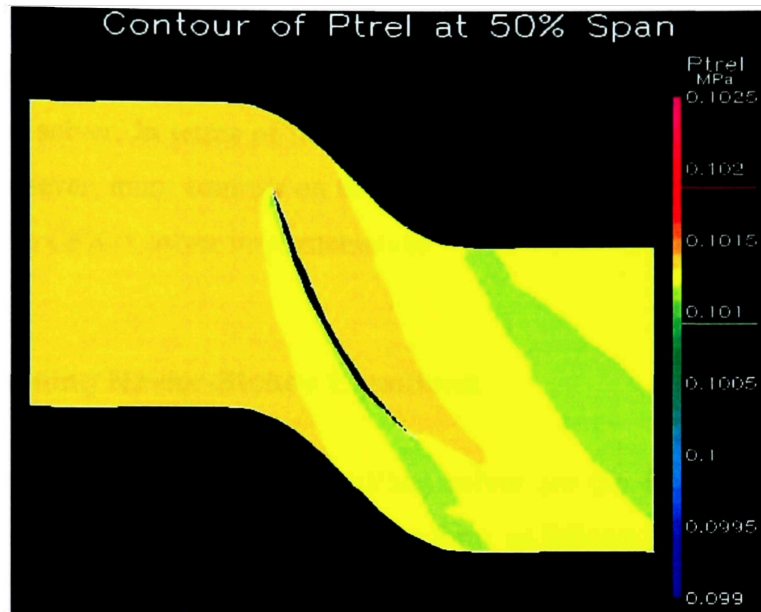


Figure 23: Sample Plot of Ptrel at 50% Span

For the benchmark case, the batch utilities provided as part of the CFX-Bladegen(Plus) software suite provide capabilities that particularly lend themselves to automatically interfacing with other MDO environment components. Particularly significant is the ability to completely describe a blade model using the batch input, text-based file format and perform a complete CFD analysis of the model with no interactive input from the user. This provides enhanced control and flexibility in changing parameters that define the blade model. A detailed theoretical discussion of the numerical schemes utilized by the CFX-Bladegen(Plus) N-S solver is presented in the following section.

2.3 The CFX-Bladegen(Plus) CFD Solver

The CFX-Bladegen(Plus) computational fluid dynamics solver is the same solver that is employed in the CFX-5 general purpose commercial CFD package. More turbulence models and solutions control parameters are available in the CFX-5 version of the solver. In terms of the numerical scheme used, both CFD solvers are the same. However, more controls on the behavior of the CFX numerical scheme are available in the CFX-5 solver implementation.

2.3.1 Governing Navier-Stokes Equations

The equations solved by the Bladegen(Plus) solver are the full, unsteady, viscous Navier-Stokes equations in their conservative form as follows [47]:

The Continuity Equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{U}) = 0 \quad (2.8a)$$

The Momentum Equations:

$$\frac{\partial \rho \vec{U}}{\partial t} + \nabla \cdot (\rho \vec{U} \otimes \vec{U}) = \nabla \cdot (-p \hat{\delta} + \mu (\nabla \vec{U} + (\nabla \vec{U})')) + \vec{S}_M \quad (2.8b)$$

The Energy Equation:

$$\frac{\partial \rho h_T}{\partial t} - \frac{\partial p}{\partial t} + \nabla \cdot (\rho \vec{U} h_T) = \nabla \cdot (\lambda \nabla T) + S_E \quad (2.8c)$$

Where:

- ρ - Density
- \vec{U} - Velocity vector such that: $\vec{U} = U_x \hat{i} + U_y \hat{j} + U_z \hat{k}$
- $U_{x,y,z}$ - Velocity components in x, y, z directions
- U - Magnitude of velocity vector (speed)
- $\hat{i}, \hat{j}, \hat{k}$ - Cartesian coordinate system unit vectors
- p - Thermodynamic (static) pressure

T - Thermodynamic (static) temperature

λ - Thermal conductivity

h_T Total Enthalpy such that:

$$h_T = h + \frac{1}{2}U^2 \quad (2.9)$$

h - Static enthalpy

μ - Dynamic (molecular) viscosity

\vec{S}_E Energy source term

\vec{S}_M - Momentum source term

$(\nabla \vec{U})'$ Transpose of resultant $(\nabla \vec{U})$ vector

$\hat{\delta}$ - The Kronecker Delta function (identity matrix) such that:

$$\hat{\delta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.10)$$

∇ - Gradient vector such that:

$$\nabla = \frac{\partial}{\partial x} \hat{i} + \frac{\partial}{\partial y} \hat{j} + \frac{\partial}{\partial z} \hat{k} \quad (2.11)$$

\otimes - The tensor product of two vectors such that:

$$\vec{U} \otimes \vec{V} = \begin{bmatrix} U_x V_x & U_x V_y & U_x V_z \\ U_y V_x & U_y V_y & U_y V_z \\ U_z V_x & U_z V_y & U_z V_z \end{bmatrix} \quad (2.12)$$

Thus using tensor notation, the product in the energy equation:

$$\nabla \cdot (\rho \vec{U} \otimes \vec{U}) = \begin{bmatrix} \frac{\partial}{\partial x} (\rho U_x U_x) & \frac{\partial}{\partial y} (\rho U_y U_x) & \frac{\partial}{\partial z} (\rho U_z U_x) \\ \frac{\partial}{\partial x} (\rho U_x U_y) & \frac{\partial}{\partial y} (\rho U_y U_y) & \frac{\partial}{\partial z} (\rho U_z U_y) \\ \frac{\partial}{\partial x} (\rho U_x U_z) & \frac{\partial}{\partial y} (\rho U_y U_z) & \frac{\partial}{\partial z} (\rho U_z U_z) \end{bmatrix} \quad (2.13)$$

For fluid problems when viscous work is significant, an additional term is included in the energy equation. This additional term is used to account for the effect of viscous shear in the energy transport equation as follows:

$$\frac{\partial \rho h_T}{\partial t} - \frac{\partial p}{\partial t} + \nabla \cdot (\rho \vec{U} h_T) = \nabla \cdot (\lambda \nabla T) + S_E + \nabla \cdot (\mu \nabla \vec{U} + \nabla \vec{U}) - \frac{2}{3} \nabla \cdot \vec{U} \hat{\delta} \vec{U} \quad (2.14)$$

The set of five N-S equations contain seven unknowns ($U_x, U_y, U_z, p, T, \rho, h$), hence two additional relationships are required to close the set of equations. In the CFX-Bladegen solver, The *Equation of State* and the *Constitutive Equation* are used to close the set of N-S equations.

The *Equation of State* is used by the flow solver to deduce density from the pressure and static enthalpy. The *Constitutive Equation* is used to deduce the temperature from the calculated pressure and enthalpy values. The solver allows the use of the ideal gas law (IGL) to model ideal gases and also allows for the fluid properties to be specified using tables for non-Newtonian fluids. The ideal gas law is used for the benchmark MDO study as follows with air as the working fluid:

$$\rho = \frac{w(p)}{R_0 T} \quad (2.15)$$

Where:

- w - Molecular weight of the fluid
- R_0 - Universal gas constant

The *Constitutive Equation* to close the set of governing equation is formed from the relationship describing the enthalpy change for an ideal gas whose specific heat, c_p is constant or a function of temperature:

$$c_p = c_p(T) \quad (2.16)$$

The CFX-Bladegen(Plus) solver calculates the static enthalpy (h) either directly or from the total enthalpy (h_T) using the relationship in equation 2.9. The thermodynamic (static) temperature is then determined from the enthalpy using the following relationship:

$$h - h_{ref} = \int_{T_{ref}}^T c_p(T) \delta T \quad (2.17)$$

Where:

T_{ref} - Solver reference state temperature (0 K)

h_{ref} - Solver reference enthalpy state (0 J/kg)

In a simplified case such as the benchmark study where the density and specific heat are constant due to the incompressible (low speed) nature of the flow, the integral equation of 2.17 reduces to the following simplified algebraic form:

$$h - h_{ref} = c_p (T - T_{ref}) \quad (2.18)$$

The specific heat-enthalpy relationship closes the set of equations that are solved to obtain the seven unknown properties ($U_x, U_y, U_z, p, T, \rho, h$) in the set of N-S equations.

2.3.2 The Rotating Reference Frame

Some modifications are necessary to the original set of N-S equations in order to account for the swirling nature of flows associated with turbomachinery. This is particularly the case in the benchmark case where the problem to be solved involves the rotating reference frame of the fan.

The first modification to be made is to the velocity of the fluid which must take into account a rotational component induced by the rotation of the fan. The velocity in the rotating frame of reference (\vec{U}_r) is defined as follows:

$$\vec{U}_r = \vec{U}_s - \omega \times \vec{R} \quad (2.19)$$

Where:

- \vec{U}_r - Velocity in rotating frame of reference
- \vec{U}_s - Velocity in stationary frame of reference
- ω - Angular velocity (RPM)
- \vec{R} Local radius vector

The thermodynamic (static) pressure, p related to the total pressure in the relative frame (P_{Trel}) using the following relationship for incompressible flow:

$$p_{Trel} = p + \frac{1}{2} \rho (\vec{U}_r \cdot \vec{U}_r - ((\omega \times \vec{R}) \cdot (\omega \times \vec{R}))) \quad (2.20)$$

The total pressure in the stationary frame of reference is also known as the total pressure in the absolute frame of reference, P_{Tabs} . It is related to the thermodynamic (static) pressure as follows:

$$p_{Tabs} = p + \frac{1}{2} \rho (\vec{U}_s \cdot \vec{U}_s) \quad (2.21)$$

The temperature calculation is dependent on the proper determination of the static enthalpy which is in turn usually deduced from the total enthalpy. In the rotating reference frame, three distinct total enthalpies are introduced. They are computed as follows:

$$h_{Trel} = h + \frac{1}{2} (\vec{U}_r \cdot \vec{U}_r) \quad (2.22)$$

$$(Rothalpy), h_{T,rot} = h + \frac{1}{2} (\vec{U}_r \cdot \vec{U}_r - (\omega \times \vec{R}) \cdot (\omega \times \vec{R})) \quad (2.23)$$

$$h_{Tabs} = h + \frac{1}{2} (\vec{U}_s \cdot \vec{U}_s) \quad (2.24)$$

Where:

- h_{Trel} - Relative total enthalpy
- $h_{T,rot}$ - Rotating frame total enthalpy (Rothalphy)
- h_{Tabs} Total enthalpy in stationary frame

In a rotating frame of reference, the CFX solver always solves for the relative total enthalpy (h_{Trel}). In an ideal gas, with a specific heat that only varies relative to temperature, the Relative Total Temperature (T_{Trel}), Rotating Frame Total Temperature ($T_{T,rot}$) and the Stationary Frame Total Temperature (h_{Tabs}) are computed as follows:

$$h_{Trel} - h_{ref} = \int_{T_{ref}}^{T_{Trel}} c_p(T) \delta T \quad (2.25)$$

$$h_{T,rot} - h_{ref} = \int_{T_{ref}}^{T_{T,rot}} c_p(T) \delta T \quad (2.26)$$

$$h_{Tabs} - h_{ref} = \int_{T_{ref}}^{T_{Tabs}} c_p(T) \delta T \quad (2.27)$$

For a fluid with constant specific heat, the thermodynamics (static) temperature (T) is deduced using any of the following total temperature relationships:

$$T_{Trel} = T + \frac{\vec{U}_r \cdot \vec{U}_r}{2c_p} \quad (2.28)$$

$$T_{T,rot} = T + \frac{(\vec{U}_r \cdot \vec{U}_r - (\omega \times \vec{R}) \cdot (\omega \times \vec{R}))}{2c_p} \quad (2.29)$$

$$T_{Tabs} = T + \frac{(\vec{U}_s \cdot \vec{U}_s)}{2c_p} \quad (2.30)$$

2.3.3 Reynolds-Averaged Navier Stokes (RANS) Equations

CFX-Bladegen(Plus) solves the Reynolds Averaged Navier-Stokes (RANS) equations which are the N-S equations modified to include fluctuating and time-averaged quantities. The approach uses the time-averaged quantities to represent the mean flow properties. The modeling of turbulent fluctuations in the flow is achieved using statistical turbulence models, which attempt to model all the length scales involved in the fluctuations in the mean flow properties. As an example, the velocity U may be represented as the combination of an average component, \bar{U} and a time-dependent component, u such that:

$$U = \bar{U} + u \quad (2.31)$$

The Reynolds averaged component is determined as:

$$\bar{U} = \frac{1}{\Delta t} \int_t^{t+\Delta t} U \partial t \quad (2.32)$$

Where:

Δt represents a time scale that is large relative to the turbulent fluctuations but small relative to the time scale of the problem.

The introduction of the time averaged quantities into the N-S equations (2.8) yields the Reynolds-averaged form of the N-S equations as follows. Note that the “bar” is used for products of fluctuating quantities:

The Continuity Equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \bar{U}) = 0 \quad (2.33a)$$

The Momentum Equations:

$$\frac{\partial \rho \bar{U}}{\partial t} + \nabla \cdot (\rho \bar{U} \otimes \bar{U}) = \nabla \cdot (\tau - \overline{\rho u \otimes u}) + \bar{S}_M \quad (2.33b)$$

The Energy Equation:

$$\frac{\partial \rho h_T}{\partial t} - \frac{\partial p}{\partial t} + \nabla \cdot (\rho \vec{U} h_T + \overline{\rho u h}) - \lambda \nabla T = S_E \quad (2.33c)$$

The use of the Reynolds-averaged flow properties leads to the alteration of the momentum and energy equations. These equations now contain turbulent flux ($\overline{\rho \vec{u} \otimes \vec{u}}$) and molecular diffusive ($\overline{\rho u h}$) terms which arise from the non-linear convective terms in the original N-S equations. These terms are also known as the *Reynolds Stress* and *Reynolds Flux* terms, respectively. In the Reynolds-averaged energy equation (2.33c), the total enthalpy (h_T) is determined by:

$$h_T = h + \frac{1}{2} \vec{U}^2 + \frac{1}{2} \overline{u}^2 \quad (2.34)$$

The third term in the total enthalpy expression is referred to as the *turbulent kinetic energy* (k). Thus:

$$k = \frac{1}{2} \overline{u}^2 \quad (2.35)$$

The introduction of the Reynolds stresses as additional unknowns implies that they must be determined to close the set of RANS equations. The use of the turbulence models is to serve this need of providing analytic closure to the set of RANS equations.

The number of available turbulence models is one of the most significant distinguishing factors between the CFX-Bladegen(Plus) solver and its implementation in the more general CFX-5 package. In CFX-Bladegen(Plus) the *Zero-Equation* turbulence model is the default and **only** eddy viscosity turbulence model available. This turbulence model is not as widely used as the more common *k-ε Two-Equation* turbulence models but is satisfactory for use in preliminary CFD analysis of the fluid flow problem [47].

2.3.4 Reynolds Stresses and the Zero Equation Turbulence Model

The Zero Equation turbulence model is the default and only turbulence model available in CFX-Bladegen(Plus). It is one of the eddy viscosity models used to provide analytic closure to the set of RANS equation discussed in the previous section. The primary function of the turbulence model is to estimate the Reynolds stresses introduced as additional unknowns from the Reynolds-averaging of the N-S equations.

Eddy viscosity turbulence models assume that the Reynolds stresses can be related to the mean velocity gradients in the flow. Additionally, the Reynolds stresses are assumed to also be related to the turbulent (eddy) viscosity of the flow. These relationships are assumed to be similar to the stress-strain tensor relationship in laminar Newtonian flow [47]. The Reynolds stress and flux terms can then be expressed as follows:

$$\overline{\rho \vec{u} \otimes \vec{u}} = -\frac{2}{3} \rho k \hat{\delta} - \frac{2}{3} \mu_t \nabla \cdot \vec{U} \hat{\delta} + \mu_t (\nabla \vec{U} + (\nabla \vec{U})') \quad (2.36)$$

Where:

μ_t Turbulent (eddy) viscosity

Similar to the eddy viscosity approach, the eddy diffusivity is assumed to be linearly related to the mean scalar gradient:

$$\overline{\rho u h} = \Gamma_t \nabla h \quad (2.37)$$

Where:

Γ_t - Eddy diffusivity such that:

$$\Gamma_t = \frac{\mu_t}{Pr_t} \quad (2.38)$$

Pr_t - Turbulent Prandtl Number

The eddy viscosity is directly prescribed while the eddy diffusivity is commonly deduced from a prescribed turbulent Prandtl number.

The eddy viscosity models essentially describe the Reynolds stress terms and their associated turbulent fluctuation as functions of the turbulent (eddy) viscosity μ_t . Substituting the eddy viscosity and diffusivity models in the RANS equations yields a set of momentum and energy RANS equations directly dependent on the eddy viscosity and diffusivity properties. Note that the RANS continuity equation remains unchanged:

The Continuity Equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{U}) = 0 \quad (2.39a)$$

The Momentum Equations:

$$\frac{\partial \rho \vec{U}}{\partial t} + \nabla \cdot (\rho \vec{U} \otimes \vec{U}) = \vec{B} + \nabla p'' + \nabla \cdot (\mu_{eff} (\nabla \vec{U} + (\nabla \vec{U})')) \quad (2.39b)$$

The Energy Equation:

$$\frac{\partial \rho h_T}{\partial t} - \frac{\partial p}{\partial t} + \nabla \cdot (\rho \vec{U} h_T) = \nabla \cdot \left(\lambda \nabla T + \frac{\mu_t}{Pr_t} \nabla h \right) + S_E \quad (2.39c)$$

Where:

μ_{eff} Effective viscosity such that:

$$\mu_{eff} = \mu + \mu_t \quad (2.40)$$

p'' - Modified pressure defined by:

$$p'' = p + \frac{2}{3} \rho k + \nabla \cdot \vec{U} \left(\frac{2}{3} \mu_{eff} - \zeta \right) \quad (2.41)$$

ζ - Bulk viscosity

The *Zero Equation* turbulence model is the default model used in the CFX-Bladegen(Plus) solver. It is a simple eddy viscosity model that computes a global value for μ_t from the mean velocity and length scales using an empirical formula. The mean velocity and length scales are computed from the physical geometry of the fluid flow problem. No additional transport equations are solved hence the *Zero Equation* nomenclature.

In order to estimate the turbulent fluctuations in the mean flow, the solver computes a single turbulent eddy viscosity for the entire fluid flow domain. The relationship used to predict the eddy viscosity is based on turbulent velocity and length scales as proposed by Prandtl and Kolmogorov [47]:

$$\mu_t = \rho f_\mu U_t l_t \quad (2.42)$$

Where:

f_μ - Proportionality constant (Solver default = 0.01)

U_t - Velocity scale (Max velocity in fluid domain)

l_t Length scale calculated by:

$$l_t = \frac{\left(\frac{1}{V_D^3} \right)}{7} \quad (2.43)$$

V_D Volume of fluid domain

2.3.5 Solver Numerical Discretization and Solution Scheme

In solving the RANS equations for a fluid problem over a specified, domain, it is usually not possible to obtain a closed form or analytic solution to the set of N-S equations which govern real fluid flow. This means that in order to obtain solutions for real fluid flow problems, a numerical approach must be adopted whereby the set of N-S or RANS equations are replaced by algebraic approximations which can be resolved using a numerical scheme over a discretized flow domain.

As previously mentioned, the CFX-Bladegen(Plus) solver solves the N-S equations over an unstructured mesh which discretizes the domain into finite control volumes. The discretization is such that the fluid properties such as momentum, energy and mass are conserved discretely for each control volume.

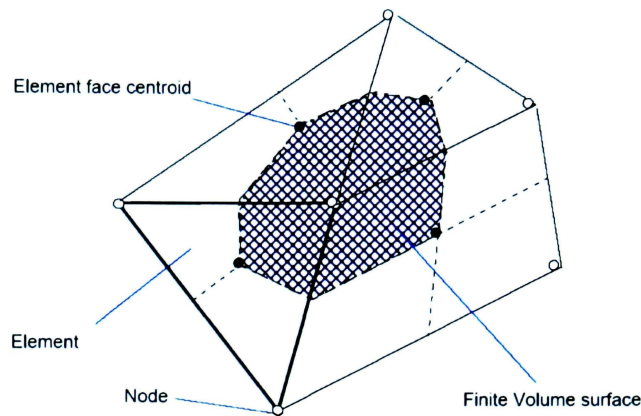


Figure 24: Sample Finite Volume [42]

Figure 24 shows a typical finite volume for an unstructured mesh where each node is surrounded by several surfaces which comprise the finite volume. All solution variables to the set of the RANS are stored at the finite volume nodes. The differential RANS equations are converted to integral form and the Gauss' divergence theorem is used to convert volume integrals to surface integrals. The integral forms of the RANS equations are in turn linearized (discretized) into a set of algebraic equation for each constituent node in the unstructured mesh. Reference [47] presents the RANS equation linearization procedure used in the CFX-Bladegen(Plus) solver.

At each node, each conserved property, ϕ in the RANS equations is determined from neighboring nodal values using the following discretization (advection) scheme:

$$\phi_n = \phi_{up} + \hat{\beta} \nabla \phi \cdot \Delta \vec{r} \quad (2.44)$$

Where:

ϕ_n - Conserved property at nodal location n

- ϕ_{up} - Value of conserved property at upwind node
- $\hat{\beta}$ - Advection blend/discretization order ($0 \leq \hat{\beta} \leq 1$)
- $\Delta\vec{r}$ - Vector from upwind node to nodal location n

The most important variable in the discretization scheme of equation 2.44 is the advection blend factor ($\hat{\beta}$). The selection of a value between 0 and 1 for this parameter determines the order of accuracy of the discretization scheme used by the CFD solver.

An advection blend ($\hat{\beta}$) value of 0 corresponds to the first order *Upwind Differencing Scheme* (UDS) which is robust (numerically stable) [47]. It is also guaranteed not to introduce non-physical over or undershoots in the estimation of flow variables. However, in fluid flow problems where the flow direction is not always normal to the finite volume surfaces, the UDS scheme may produce inaccurate results. UDS inaccuracies are usually caused by *numerical diffusion* and it often occurs in regions of recirculation where the fluid from one finite volume flows into more than one element downstream.

An advection blend value of 1 corresponds to a second order accurate discretization scheme. The quantity ($\hat{\beta}\nabla\phi \cdot \Delta\vec{r}$) in equation 2.44 is known as the *Numerical Advection Correction* and can be considered an anti-diffusive flux to the diffusion susceptible UDS scheme [47]. Advection blend values close to 1 generally yield more accurate discretization schemes, but are also less robust (more prone to instability) than the UDS scheme. This may cause CFX-Bladegen(Plus) simulations based on advection blends closer to 1 to display non-physical over/undershoots in the computed flow variables. This phenomenon is known as *numerical dispersion* and usually occurs in numerical schemes that are even-order accurate. Numerical dispersion is exhibited as oscillations or wiggles in the flow field solution, particularly where steep gradients occur [47].

The linearized sets of equations that arise using the CFX finite volume method for all elements in the flow domain are discrete conservation equations. These equations for all finite volume nodes constitute the complete coupled linear system of equations. The CFX-Bladegen(plus) solver is known as a coupled solver because the equations governing the flow variables (U_x, U_y, U_z, p) are solved as a single system. At no point are any of the equations for different properties (such as mass or momentum) solved separately as is done in a non-coupled or segregated approach [47]. The advantages of the coupled solution approach used in the CFX solver include robustness, generality, simplicity and efficiency. The principal drawback is the high storage needed for all the coefficients of the constituent linearized RANS equations.

2.3.6 Solver Residual Computation Scheme

Residuals are used to track the success of the CFD simulation in converging to an optimal solution. The CFX-Bladegen(Plus) solver computes a normalized residual for each flow variable ϕ as follows:

$$[\tilde{r}_\phi] = \frac{[r_\phi]}{a_p \Delta\phi} \quad (2.45)$$

Where:

- $[\tilde{r}_\phi]$ - Normalized residual of flow variable ϕ
- $[r_\phi]$ - Raw residual of flow variable ϕ
- $\Delta\phi$ - Change in flow variable ϕ from previous iteration (time-step)
- a_p - Normalization parameter (ref [47])

In calculating the normalized residuals for each flow variable, the following criteria are always maintained:

1. The normalized residuals are independent of time step choice
2. The normalized residuals are independent of the initial guess
3. Multiphase flows integrate the volume fraction in the calculation of the normalized residuals.

2.3.7 Solver Timestep and Target Residual Selection

It is suggested in [47, pg. 349] that between 50 and 100 timesteps are required for to achieve reasonable convergence for most steady-state problems. For the benchmark MDO study, the maximum number of timesteps for the CFD analysis module is selected as 150 iterations. It is expected that the steady-state CFD analysis should achieve convergence by 150 iterations in accordance with the recommendation of [47]. The actually calculation of the size of each timestep is automatically done by the solver based on the velocity and length scales of the problem using the *Auto Timestep* option.

For CFD analysis in the benchmark MDO study, a target residual of $1e-5$ is selected even though this value exceeds the normalized residual of $1e-4$ suggested as sufficient in [47]. Residual values of $1e-6$ or lower are considered close to the machine round off on 32-bit machines. These target residual values are consequently discouraged in [47] as they will too closely approach the computational limit of convergence on single precision machines, consequently requiring the use of double precision CFD simulations.

The overview on the details of the CFX-Bladegen(Plus) CFD solver is complete. Reference [47] contains a more extensive discussion than that presented in this section and can be consulted for further details on the numerical solution scheme used in the CFD solver. In the following sections, the non-linear optimization algorithms available in the selected MDO package (VisualDoc) are discussed. These are the MDO algorithms of interest for potential application in the benchmark automated fan design optimization study.

2.4 Multidisciplinary Optimization (MDO) Algorithms

The objective of the research is to develop a design approach that automatically achieves the optimal solution to the benchmark problem, by using mathematical *search* algorithms to manipulate the design parameters to obtain an optimal configuration. It is first necessary to discuss the philosophy behind the optimization algorithms that will direct the design process.

The optimization utility selected is the VisualDoc general purpose multidisciplinary optimization tool from Vanderplaats Research and Development Inc., Colorado Springs, Colorado. VisualDoc allows for the efficient application of various optimization algorithms to almost any design problem.

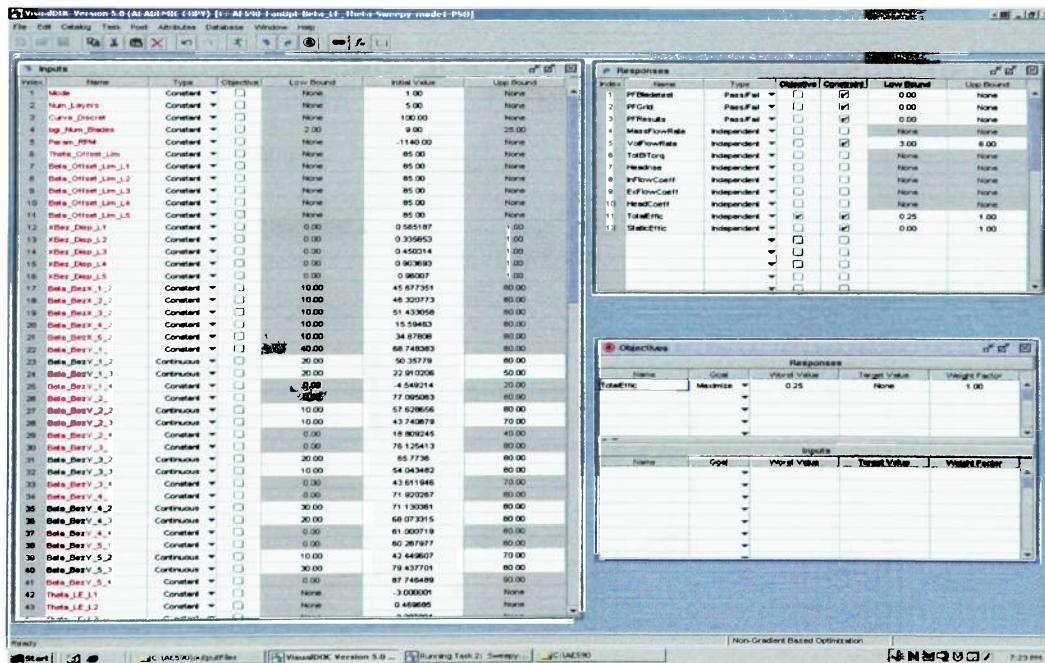


Figure 25: VisualDoc Graphical User Interface (GUI)

VisualDoc allows the easy integration of user specified engineering analysis tools with several available MDO algorithms in order to solve a specific problem. It achieves this through the use of its scripting tool called VisualScript.

For the benchmark case there are several steps which need to be accomplished between the design of the blade model, the performance evaluation of the blade model and the redesign phase by VisualDoc. As will be seen when the MDO implementation is discussed, all the batch utilities discussed in section 2.2 need to be executed in a predetermined sequence for a single successful design iteration. For the MDO based design of the fan, a design iteration includes the generation of the blade geometry and the subsequent evaluation of its performance using CFD.

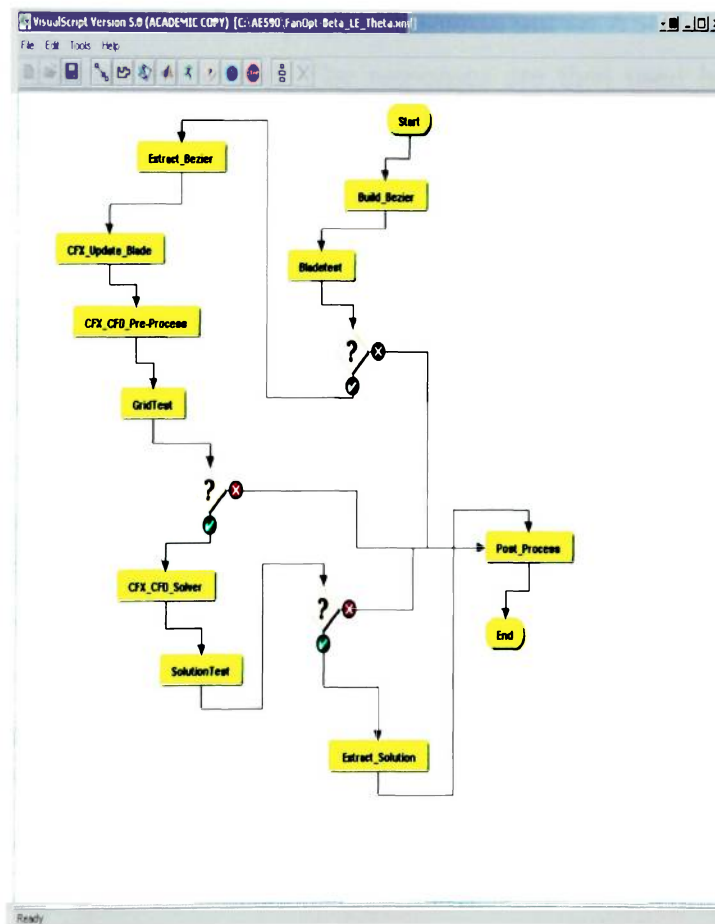


Figure 26: VisualScript Graphical User Interface (GUI)

VisualScript allows several design analyses tools (programs) to be coupled together using a graphical programming interface as shown in Figure 26. It represents each analysis program as an element in the visual program sequence. There are also tools which allow certain programs to be executed conditionally depending on values of various design parameters. All the analyses programs detailed in the VisualScript are then combined to form a “program” that is used as the *response analysis* tool in a VisualDoc optimization study.

The primary means of communication between various programs that constitute the response analyses tool is a simple text file format [1]. VisualDoc writes out an ASCII text file with the design variables (inputs) and the response analysis program reads this file to create the new design for the current optimization iteration. VisualDoc then expects the response analysis program to write out an ASCII text file with the results of the analyses (responses). The responses are then used by VisualDoc to determine the next design based on the optimization algorithm being used. Balabanov et al [1, 6], include more details on the implementation of the VisualDoc/VisualScript MDO package.

The focus of the investigation is applying MDO algorithms to the benchmark case in order to validate the automated MDO based design methodology. Of primary importance is a detailed understanding of the algorithms to be employed. In essence we are seeking to understand what guides the design process towards the best possible combination of modeling parameters. The answer to this question lies in the implementation of optimization algorithms whose typical functions include deriving the optimum value for a given dependent function subject to perturbations in independent variables that control the behavior of the function.

The MDO algorithms of particular interest in the benchmark case are the non-linear and evolutionary type optimization algorithms. The other primary group of optimization algorithms is the linear gradient based optimization techniques. These include the Sequential Quadratic Programming (SQP), Sequential Linear

Programming (SLP), Fletcher Reeves and the Modified Method of Feasible Directions (MMFD).

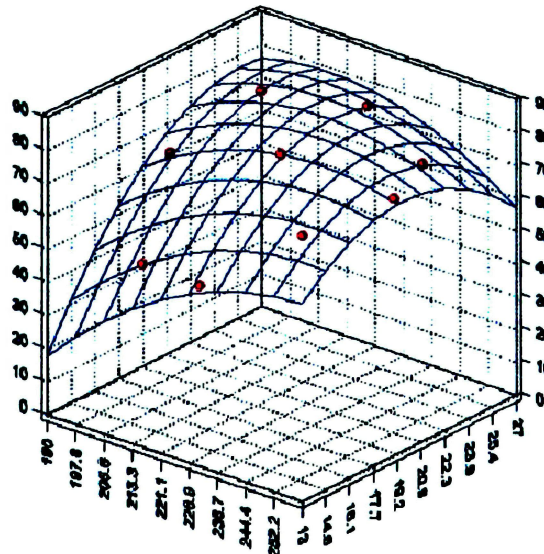
Generally, gradient based optimization algorithms are able to identify optimal designs in the design space close to the starting point of the optimization analyses and tend to locate local, rather than global optimal solutions. The primary focus of this effort is in exploring the non-linear, non-gradient based optimization algorithms. The non-gradient based algorithms are known to be computationally more expensive than their gradient based counterparts. However, they are significantly better at locating global optima and are more robust at tolerating discontinuities in the design space.

The three non-gradient based algorithms available in the VisualDoc package are of interest in the benchmark MDO study. They include the Particle Swarm Optimization (PSO), the Genetic (GA) and the Response Surface Optimization (RSO) algorithms. The RSO algorithm is not considered an evolutionary type but is particularly useful when no explicit relationships exist between design variables and responses, or when such relationships are highly complicated [1]. Examples of engineering applications with highly complicated variable-response relationships include FEA and CFD design applications. The following sections discuss the non-gradient based, non-linear MDO algorithms available in the VisualDoc MDO package.

2.4.1 Response Surface Approximate Optimization (RSO)

The Response Surface algorithm attempts to optimize a function by first creating an explicit approximation of the objective function relative to the design constraints. A surface is mapped to a series of design points evaluated throughout the design space using a least squares regression analyses technique. The approximation of the explicit function is usually a combination of low-order polynomials. Higher order polynomials (4th order and higher) are rarely employed in response surface algorithms due to the highly non-linear increase in computational expense.

The mapped function developed using the regression analysis is assumed to be representative of the behavior of the objective function in the design space and is used as a replacement for the often computationally expensive analyses. The best design point is obtained by optimizing the function mapping in the design space. Figure 27 shows a sample response surface approximation map created using analyses results at nine design points (red dots).



The response surface approximation is essentially an attempt to fit a surface to the distribution of valid points in the objective function design space after constraints are imposed. Thus:

$$\vec{y} = \hat{\vec{y}} + \vec{e} = \vec{X}\vec{B} + \vec{e} \quad (2.46)$$

Where:

- \vec{y} - Vector of true response function values
- $\hat{\vec{y}}$ - Corresponding vector of approximated responses
- \vec{X} - Model Matrix (Dependent on the order of approximation model)
- \vec{B} - Vector of regression coefficients
- \vec{e} - Vector of approximation errors

The vector of approximation errors is used to capture sources of variability in the approximated responses. These sources of variability include random numerical noise and modeling inaccuracies resulting from insufficient response approximation models (e.g. low order polynomial model). For statistical purposes, it is assumed that approximation errors are normally distributed about a mean of zero.

The response approximation linear equation (2.46) is written as a product between the model matrix \vec{X} and the vector regression coefficients, \vec{B} . N will be used to represent the number of actual response analyses results and p will represent the number of regression coefficients specified by the polynomial model. The model matrix \vec{X} will possess a dimension of $[N, p]$. The vector of regression coefficients (\vec{B}) and the vector of approximation errors (\vec{e}) will each possess dimensions of $[p, 1]$. Each row in the model matrix contains values of functions evaluated at a data point in the design space. In general, more data points are available than required to determine the number of regression coefficients.

Consider an example of a quadratic (2nd-order polynomial) response surface model for a problem with only one design variable. Response analyses values are provided at four data points (red data points in Figure 28). The response surface model (equation 2.46) for a single response analysis point becomes:

$$y = B_0 + B_1 x + B_2 x^2 + e \quad (2.47)$$

Writing equation (2.47) for each of the four response analyses points yields a system of equations:

$$\begin{Bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{Bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \end{bmatrix} \begin{Bmatrix} B_0 \\ B_1 \\ B_2 \end{Bmatrix} + \begin{Bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{Bmatrix} \quad (2.48)$$

The system of equations in 2.48 is analogous to the vector form of the general response surface approximation equation (2.46). The system of equations defines a curve which passes close to the four response analyses points as shown in Figure 28. The objective of the response analyses approximation is to find the regression coefficients B_0 , B_1 and B_2 which minimize the sum of squares of the elements in the vector of approximation errors (e_1, e_2, e_3, e_4).

Rewriting equation 2.48 with the vector of approximation errors on the LHS yields:

$$\vec{e} = \vec{y} - \vec{X}\vec{B} \quad (2.49)$$

The sum of squares of the terms in the approximate error is given by the relation:

$$SS_{\vec{e}} = \sum_{i=1}^N e_i^2 = \vec{e} \cdot \vec{e} \quad (2.50)$$

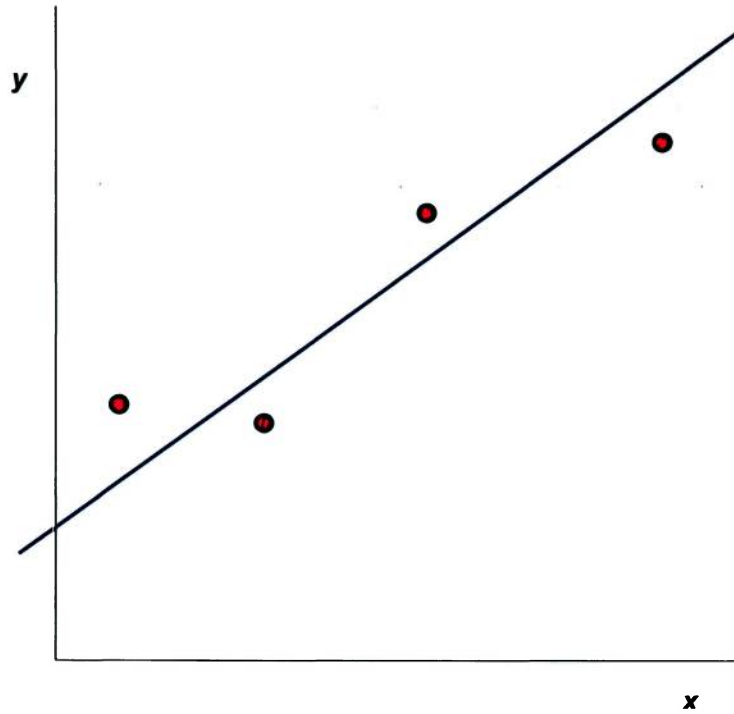


Figure 28: Data points and Response Surface Approximation (1-Variable Problem)

Combining equations 2.49 and 2.50 yields a vector relationship for the sum of squares of the vector of approximation errors in terms of the model matrix, \bar{X} and the vector of regression coefficients, \bar{B} :

$$\sum_{i=1}^N e_i^2 = \bar{e} \cdot \bar{e} = (\bar{y} - \bar{X}\bar{B})' \cdot (\bar{y} - \bar{X}\bar{B}) \quad (2.51)$$

Expanding the RHS of 2.51 using vector algebra yields:

$$\sum_{i=1}^N e_i^2 = (\bar{y} - \bar{X}\bar{B})' \cdot (\bar{y} - \bar{X}\bar{B}) = \bar{y}'\bar{y} - 2\bar{B}'\bar{X}'\bar{y} + \bar{B}'\bar{X}'\bar{X}\bar{B} \quad (2.52)$$

The objective of the response surface approximation is to reduce the sum of squares of the errors term by selection the appropriate vector of regression coefficients, $\bar{\mathbf{B}}$. This requires that we perform the derivative of equation 2.52 with respect to the regression coefficient vector and set the value of the derivative equal to zero. Thus, from eq. 2.52:

$$\frac{\partial \left(\sum_{i=1}^N e_i^2 \right)}{\partial \bar{\mathbf{B}}} = -2\bar{\mathbf{X}}' \bar{\mathbf{y}} + 2\bar{\mathbf{X}}' \bar{\mathbf{X}} \bar{\mathbf{B}} \quad (2.53)$$

Setting the RHS of equation 2.53 to zero yields the estimate of the vector of regression coefficients, $\bar{\mathbf{B}}$ that will provide the minimum sum of the squares of the approximation errors:

$$\bar{\mathbf{B}} = (\bar{\mathbf{X}}' \bar{\mathbf{X}})^{-1} \bar{\mathbf{X}}' \bar{\mathbf{y}} \quad (2.54)$$

The vector from equation 2.54, when substituted into equation 2.48 will cause the solid line of Figure 28 to pass closest to the red points in the design space which are actual response analyses. It should also be noted that the derivative of equation 2.54 is positive. This ensures that the vector of regression coefficients will yield a minimum sum of squares of the error terms and not a maximum. The response surface solution is generally applicable to any problem whose number of response analyses points, N is greater than the number of terms (p), in the response surface model (i.e. $N \geq p$).

In considering the response surface model to be employed in any optimization problem, the dimensionality of the problem must take into account the number of functions calls required to determine the coefficients of the vector of regression coefficients. For example, a problem with y number of design variables requires that $y+1$ analyses call be made in order to use a linear response surface model (the additional function call for the constant term B_0 in equation 2.47. Increasing the order of the response surface model to a quadratic approximation requires an additional

$y(y+1)/2$ function calls. A total of $(y+1)(y+2)/2$ response analyses calls are required to determine all the components of the regression coefficient vector (\vec{B}) for a fully quadratic, 2nd order response surface model.

It subsequently become rather computationally expensive to use response surface polynomial models of high orders for problem with large numbers of design variables, as increasing the order response surface model corresponds to a highly non-linear increase (growth) in the number of analyses calls required. This response surface approximation phenomena is referred to as the *curse of dimensionality*.

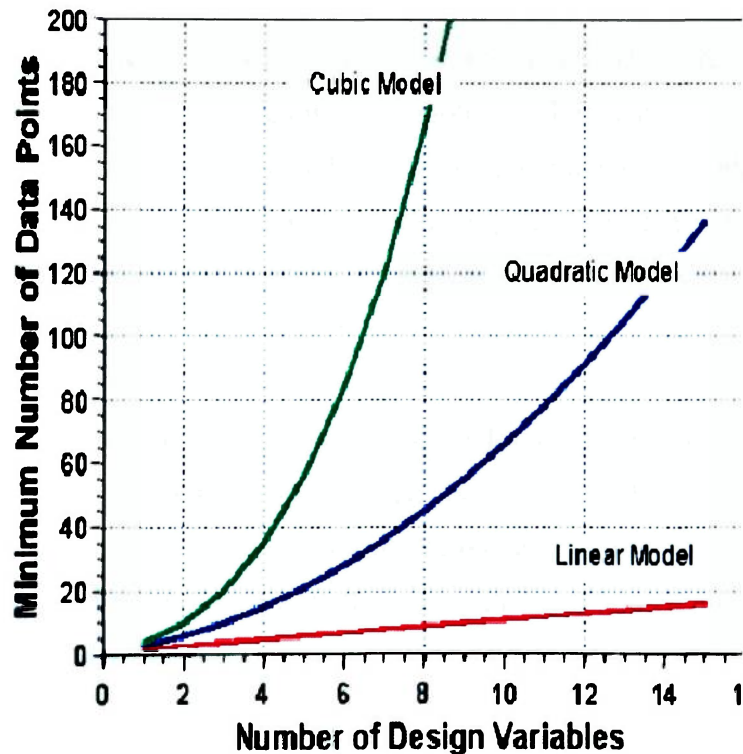


Figure 29: Number of Response Analyses Calls Relative to RSO Model

(Courtesy Vanderplaats Research and Development, Inc.,)

Ideally one would wish to employ a response surface model using a high order polynomial, say of the 4th, 5th or 6th order. However, this would require an inordinate number of analyses calls in order to calculate the elements of the vector of regression coefficients as can be seen from Figure 29. The benchmark case employs a quadratic response surface model for all RSO analyses. This ensures that the response surface model maintains a reasonable degree of accuracy without being computationally prohibitive.

2.4.2 Particle Swarm Optimization (PSO)

The Particle Swarm Optimization algorithm belongs to a class of probabilistic search algorithms commonly termed evolutionary algorithms (EO). They are often based on simplified models that attempt to duplicate sociological behavior observed in natural phenomena. PSO is based on the social theory that a population of individuals adapt to their environment by returning to promising regions that were previously discovered [16]. These optimal locations must also be appropriately communicated to all members in the swarm.

The PSO search algorithm can be considered analogous to the sociological model exhibited by a swarm of ants locating a rich food source. The information on the location of the food source is communicated to the swarm by a combination of the individual memory of the bee that initially locates the food source and the collective memory of the swarm of bees.

The PSO algorithm is a particularly robust algorithm and has been proven as especially suitable for problems dealing with functions containing discontinuities as well as numerical noise in the design space [16]. Unlike most gradient based optimization algorithms, which require gradient information (or at least the existence of a continuous function whose gradient can be evaluated), PSO is often employed when dealing with optimization problems that involve discrete design parameters.

In determining the optimal solution to a specified problem, the PSO algorithm is significantly dependent on the efficient communication between members of the swarm (particles) spread throughout the design space. The algorithm seeks to update the position of each particle in the swarm over time (design iteration) such that most of the particles in the swarm converge on the optimum location in the design space.

The update of the position of a particle in the swarm is dependent on the memory of the various optimal locations that individual particles have passed through (local memory), as well as the best position in the design space *any* particle in the swarm has encountered up to the current iteration (global memory). The update to the position of a particle is done using a “velocity” vector with the following as the general PSO strategy:

1. Create an initial population of particles randomly spread throughout the design space with random initial velocities.
2. Estimate a velocity vector for each particle using its individual memory and the group memory of the swarm.
3. Update the location of each particle in the design space using the estimated velocity vector and its previous position.
4. Return to step 2 and iterate until convergence is achieved.

If \bar{x}_k^i represents the position of particle x at iteration k , the update to the position of the particle is computed by:

$$\bar{x}_{k+1}^i = \bar{x}_k + \bar{v}_{k+1}^i \Delta t \quad (2.55)$$

Where: \bar{x}_{k+1}^i - position of particle i at iteration $k+1$
 \bar{v}_{k+1}^i - velocity of particle i at iteration $k+1$
 Δt - step size

The step size in equation 2.55 is often used to control the rate at which the particle moves in the direction of the optimal solution. The velocity vector, \bar{v}_{k+1}^i is estimated using the relationship [17]:

$$\bar{v}_{k+1}^i = w\bar{v}_k^i + c_1 r_1 \frac{(\bar{p}^i - \bar{x}_k^i)}{\Delta t} + c_2 r_2 \frac{(\bar{p}_k^g - \bar{x}_k^i)}{\Delta t} \quad (2.56)$$

Where: r_1, r_2 - random numbers between 0 and 1
 \bar{p}^i - best location found by particle i up to current iteration
 \bar{p}_k^g - best position in the swarm population at iteration k
 c_1 - self confidence parameter
 c_2 - group/swarm confidence parameter
 w - particle inertia

The inertia parameter, w controls the exploratory properties of the particle in question. Larger inertia values promote a more global search behavior and smaller values causing a more local search behavior. Since the object of the particle is to advance its position based the estimated velocity vector, the two confidence parameters, c_1 and c_2 are particularly important in the velocity vector calculation.

The c_1 parameter is a measure of how much trust or confidence the particle should have in its own memory, while the c_2 parameter indicates how much confidence the particle should have in the collective memory of the swarm. There are several variations of the strategy used to estimate the velocity vector. One approach suggests

that the best position in the swarm to date, \bar{p}^g be used instead of the best position in the swarm at the current iteration, \bar{p}_k^g as detailed in equation 2.56. However studies indicate that the approach of equation 2.18 performs better for certain applications [17].

Another influence in the application of the particle swarm algorithm is the generation of the initial swarm of particles. It is important that the initial swarm is random enough to adequately cover the design space. A common approach is to make use of a pseudo-random swarm function to generate the initial velocity and location of the particles throughout the design space as follows:

$$\bar{x}_0^i = \bar{x}_{\min} + r_1 (\bar{x}_{\max} - \bar{x}_{\min}) \quad (2.57)$$

$$\bar{v}_0^i = \frac{\bar{x}_{\min} + r_2 (\bar{x}_{\max} - \bar{x}_{\min})}{\Delta t} \quad (2.58)$$

where: \bar{x}_{\min} - vector of lower bounds of design variables
 \bar{x}_{\max} - vector of upper bounds of design variables

The ability of the PSO algorithm to handle functions with numerical noise and spurious fluctuations in the design space makes the algorithm of particular interest in engineering design applications which are highly non-linear and discontinuous.

2.4.3 Genetic Optimization Algorithm (GA)

The genetic algorithm is another optimization methodology in the previously identified class of evolutionary optimization algorithms (EOs). The genetic algorithm (GA) is based on the numerical simulation of the evolutionary principle of the survival of the fittest. The genetic algorithm seeks optimal solutions through a modeling Darwin principle of natural selection using three operators: *selection*, *reproduction through crossover* and *elitist strategy* [19].

Genetic algorithms are as robust as the particle swarm optimization algorithm previously discussed. However, unlike the PSO algorithm which gives the optimal value in the design space, the GA search methodology will often provide the designer with a series of near-optimal designs rather than a single design [19]. As is the case for the PSO algorithm, they are also ideally suited for dealing with problems that are highly non-linear, exhibit discontinuities and/or require the manipulation of discrete design variables.

The initial population consists of a number of designs randomly distributed throughout the design space. Each initial point is generated using a random combination of design variables. The design points are described using a number of *genes* which essentially identify the unique combination of variables that constitute the design point. The following describes the general methodology of the subsequent genetic search:

1. Generate initial population from random combination of design variables.
2. Rank each individual (design point) based on fitness to reproduce.
3. Copy best design to the next generation (elitist strategy)
4. Select fit designs for reproduction

5. Generate child designs for subsequent generation.
6. Apply mutation operator to prevent genetic uniformity.
7. Return to step 2 and repeat subsequent steps until evolution of individuals is no longer possible (convergence)

Once the initial designs are evaluated using response analysis calls (objective function calls), all the designs are ranked according to their fitness to reproduce, with the best design being assigned the maximum fitness. Following the fitness evaluations, an *elitist strategy* is implemented by copying over the best design (highest fitness value) into the next generation without any modification its design variables. This ensures that the best design is always propagated to subsequent generations (iterations) once it is initially identified.

The fitness rankings of the design points are used to select designs that are fit for reproduction. The probability of selection (between 0 and 1) for reproduction of a particular design is determined using the relationship:

$$\frac{2[m + 1 - i]}{m^2 + m} \quad (2.59)$$

where: m – total number of designs
 i – rank of current design being considered

A pair of designs is selected as parent designs and they are used to generate a new design referred to as the child design. The process of reproduction used to generate the child design is referred to as a *two point crossover*. Table 4 demonstrates the concept of the two point cross over technique used to simulate reproduction between species (designs) in the genetic algorithm.

Table 4: Genetic Algorithm Reproduction Using Two Point Crossover

Variable	a	b	c	d	e	f	g
Parent Design	44.52	52.15	25.14	45.17	255.00	25.00	65.10
Parent Design	64.15	15.54	36.00	47.00	365.00	55.00	49.12
Child Design	44.52	52.15	36.00	47.00	255.00	25.00	65.10

The two parent designs on Table 4 are made up of a combination of values for the design variables *a* through *g*. Upon selection for reproduction, two break points in the parent designs are selected at random, consequently dividing the parent designs into three substrings. Reproduction is simulated by generating a child design by combining constituent substrings of the parent designs. In Table 4, the child design is composed of three substrings, the first and last substring being taken from the first parent and the second substring, coming from the second parent design.

One of the most important phenomena in biological reproduction and evolution is the concept of mutation. The purpose of mutation is to prevent the genetic pool of the offspring generation from becoming exceedingly uniform. The same philosophy applies in the case of the genetic algorithm [19].

The diversity in the offspring designs is numerically simulated by introducing a *mutation operator* whose function is to induce some measure of randomness in the design points obtained by the two-point crossover. Essentially, the mutation randomly selects a gene (variable) in the offspring design and modifies its value. Each variable in the offspring design has a small probability of having mutation performed on it and this probability serves as the basis on which the selection for mutation is made. Thus after mutation, the child design from Table 4 is altered by changing the value of the *f* variable as shown in Table 5.

Table 5: Effect of Mutation on the Offspring Design

Variable	a	b	c	d	e	f	g
Parent Design	44.52	52.15	25.14	45.17	255.00	25.00	65.10
Parent Design	64.15	15.54	36.00	47.00	365.00	55.00	49.12
Child Design	44.52	52.15	36.00	47.00	255.00	11.15	65.10

The reproduction process completes one design iteration or *generation* of the genetic algorithm. The search strategy is repeated until convergence or alternatively, for a set number of generations specified by the designer. The genetic algorithm does not always yield the optimal design but rather, a number of near optimal designs [18]. In order to obtain better results, it is important that the optimization be repeated a number of times, each time starting from different positions in the design space.

2.4.4 Selecting the MDO Algorithm for the Benchmark Study

An important requirement of the genetic algorithm implementation in the commercial VisualDoc optimization package is that it requires all the design variables to be discrete-type variables. Since most of the design variables for the benchmark case are continuous, the effort to convert the problem into a discrete one is considerable. The GA is consequently in its application to the current MDO study but still remains a viable algorithm to be considered for future optimization studies.

Of the PSO and RSO algorithms, the RSO is selected as the optimization algorithm for the current benchmark study. The selection is based on the lower computational effort required for the RSO algorithm in comparison with the particle swarm optimization strategy.

3 MDO DESIGN IMPLEMENTATION

3.1 3-Step Multidisciplinary Design Optimization Process

The automated design environment for the benchmark optimization of a fan blade is primarily composed of the computational fluid dynamics (CFD) element as well as the multidisciplinary optimization (MDO) component, all of which have been previously discussed. The various components are integrated to automatically solve the fan design benchmark optimization problem by iteratively using the following 3-step methodology:

1. Blade Geometry Modeling
2. Design Evaluation
3. Optimization based on Results of Design

The integration of the various components of the MDO based design methodology is detailed in Figure 30. As discussed in section 2, the CFX-Bladegen software suite is selected to accomplish the first step of modeling of the blade geometry. The design evaluation of step 2 is achieved using the CFX-Bladegen(Plus) utility, which is a Navier-Stokes equation solver that rapidly solves the flow through the fan by using a single blade passage analysis for the flow field computation.

The performance data generated from the flow field solution is used to determine the modifications in the variables that will yield the optimal fan blade design. The non-gradient algorithms available in the VisualDoc commercial MDO package have been studied and the RSO algorithm has been selected for the benchmark case. Figure 30 graphically illustrates the interaction between the various components of the MDO design environment, which iteratively generates new designs dictated by the selected algorithm until convergence is detected.

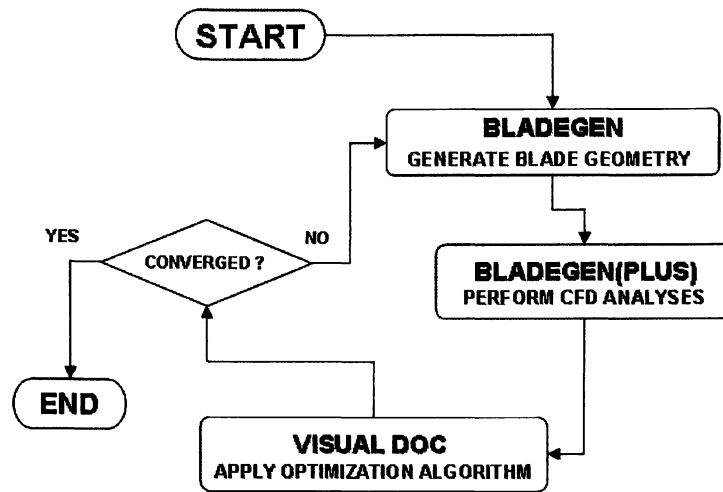


Figure 30: MDO Component Interaction

Although the concept of an automated design environment is being implemented here on a specific benchmark case, the methodology can be applied to almost any engineering application. As an example, a structural engineering application would require a Finite Element Analyses (FEA) tool in place of the CFD component of the current application. In terms of the MDO components for the benchmark case, the 3-step design process previously enumerated may be rewritten to include the various commercial engineering design utilities as follows:

1. Blade Geometry Modeling Using CFX-Bladegen
2. Design Performance Evaluation Using CFX-Bladegen(Plus)
3. Optimization of Design based on Performance Data Using VisualDoc

The objective of a completely automated design methodology creates the necessity for a transparent, seamless and efficient design and performance data transfer between the various components of the MDO environment. This necessary data transfer is accomplished using simple ASCII text files including the CFX-Bladegen *.bgi* blade geometry batch file, a text based CFD solution file and a VisualDoc Input/Output design variables and response file. It is important to note that the precise mechanisms required to facilitate communication (data transfer and modification)

between the MDO components is actually accomplished using the VisualScript tool described in section 2.3 and a discussion on of how the process is implemented in VisualScript is included in [22].

The first step in the MDO methodology for the benchmark case is the generation of the blade geometry as previously specified. The approach taken to model the blade geometry must have the capability to be modified as part of the automated MDO approach. It has already been indicated that the CFX-Bladegen *.bgi* blade geometry batch file format will be employed in the automated blade generation process.

Since the optimization process is dependent on modifying design variables based on the optimization algorithm, of primary importance is how the three-dimensional geometry of the fan blade is described in terms of design parameters. It is critical to the optimization process to develop a scheme for an efficient parameterization of the blade geometry using design variables, such that a modification of these variables is equivalent to designing different blade geometries.

The following sections detail the various sub-elements that comprise the various components of the 3-step the MDO optimization process. First we discuss the details of the fan blade geometry parameterization scheme for the benchmark problem and its implementation in the MDO design methodology.

3.2 Blade Geometry Parameterization

In discussing the methodology behind the parameterization of the blade geometry we begin from the discussion of the CFX-Bladegen coordinate of section 2.1.1. The Bladegen coordinate system from Figure 10 is shown.

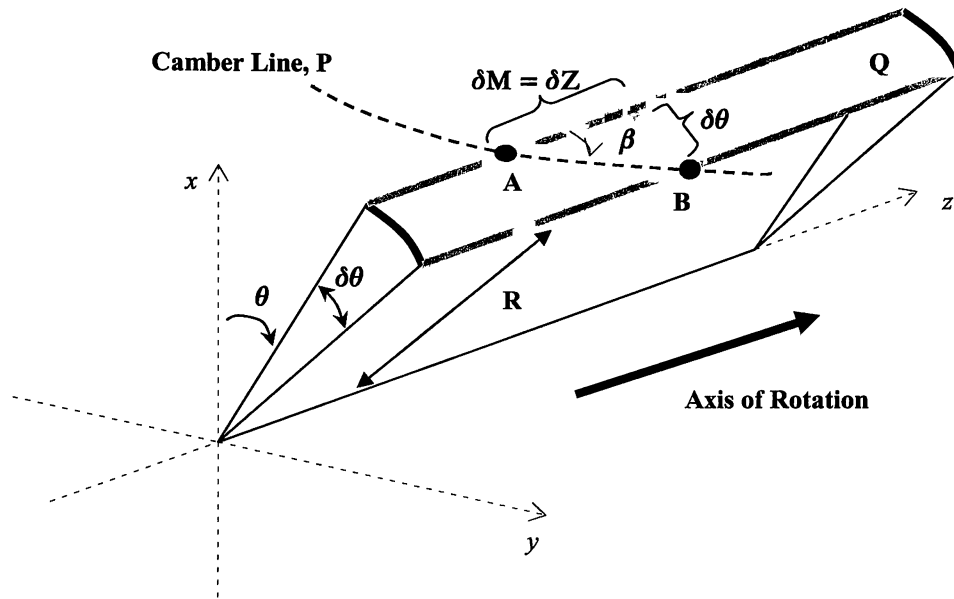


Figure 31: CFX-Bladegen Coordinate System

As previously discussed, the blade geometry in Bladegen is generated by providing the blade profile (airfoil) at several constant-radius planes called *span layers*. The constant radius layers are shown in the meridional view of Figure 32 and are spread out at intervals over the span of the blade from the hub (0%) to the shroud (100%). For the benchmark case a total of 5 span layers were used, even spread over the blade from hub to shroud at 25% increments. Thus the span layer locations were at 0.00R (Hub), 0.25R, 0.50R, 0.75R and 1.00R (Shroud), with R representing the span of the blade. Several span layers are shown in Figure 32

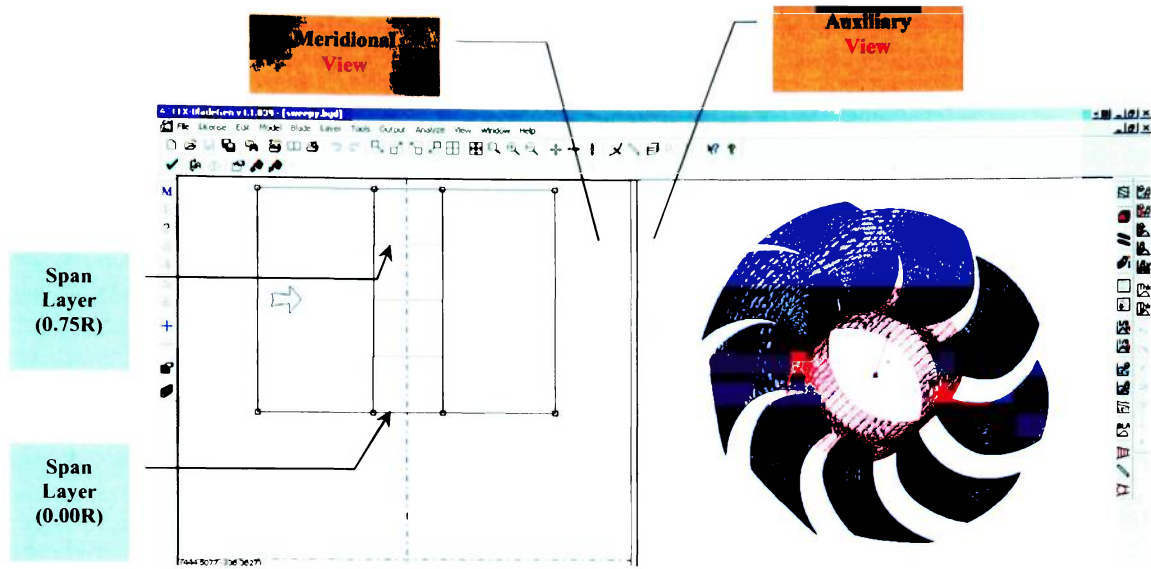


Figure 32: Meridional View (LHS) of Fan with Span Layers Visible

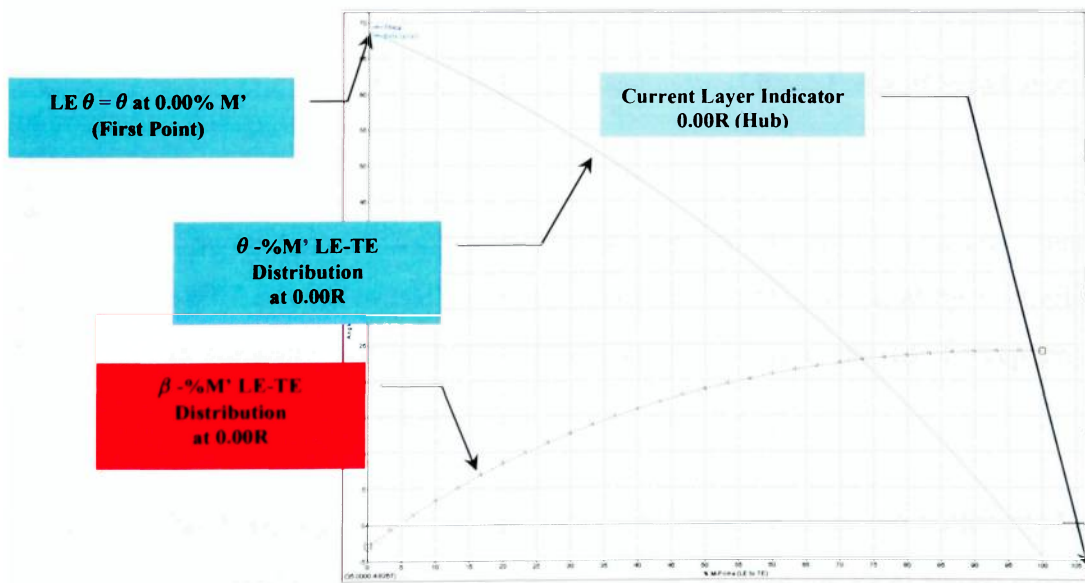


Figure 33: β vs. M' For Airfoil Camber Line Generation at Span 0.00R (Hub)

At each of the span layers of Figure 32, the blade profile shape (airfoil) is created by using the Bladegen coordinate system of Figure 31 to specify a set of points in three dimensional space that represent the camberline of the blade profile such as in Figure 34.

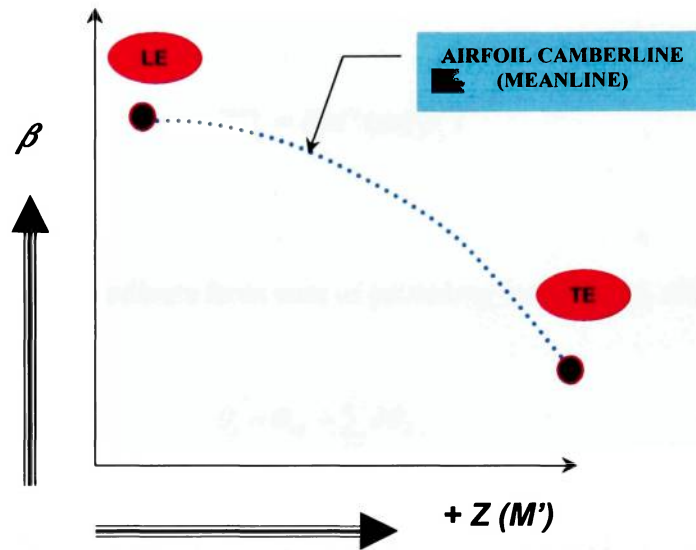


Figure 34: Sample Camberline of Blade Profile at Span Layer

The coordinates needed to generate the camber line at a specific span layer are the radial coordinate of the span layer e.g. 0.00R, the leading to trailing edge β - M' distribution such as is shown in Figure 33, and the LE Sweep angle, θ_{LE} at the respective span layers.

Note the only the LE sweep (θ_{LE}) coordinate (also the circumferential angle) needs to be specified at each span layer., This is because the local sweep angle offset ($\partial\theta$) of any point i , along the airfoil camber line relative to its preceding point can be calculated from its local relative blade angle (β) using equation 2.6. The local sweep coordinate (θ) at each point along the camberline, aft of the leading edge, is determined by adding the LE sweep to the sum of all the local sweep angle offsets

($\partial\theta_i$ s) up the point in question as in equation 3.2 below. Thus starting from equation 2.6:

$$\beta_i = a \tan\left(\frac{\partial\theta_i}{\partial M'}\right) \quad (2.6)$$

Solving for $\partial\theta$ at point i :

$$\partial\theta_i = \partial M' \tan(\beta_i) \quad (3.1)$$

Determine actual sweep coordinate from sum of preceding local sweep offsets:

$$\theta_i = \theta_{LE} + \sum_{n=1}^i \partial\theta_n \quad (3.2)$$

As a result of the β - θ relationship of equation 2.6 only the β - M' distribution (red curve) of Figure 33 needs to be specified. The blue (θ) curve is calculated by Bladegen and used to generate the camberline of the blade profile (Figure 34) at the specified span. The consequence of the β - θ relationship of 2.6 is that only one set of angular coordinates needs to be controlled in order to generate the blade geometry.

The process of generating airfoil camberlines is repeated at all the five span layers used for generating the blade geometry. The definition of the airfoil shape is completed by imposing a normal thickness distribution on to the camberline of Figure 34. The airfoil distribution is generally specified by selecting a standard NACA airfoil series thickness distribution. The thickness distribution selected is not restricted to the NACA series and may even be included as design variables in the optimization process. For the benchmark case, the thickness distribution of the NACA0006 airfoil is used at all span layers, with $\left(\frac{x}{c}\right)_{\max} = 0.30c$. This thickness distribution is held fixed and not included as an optimization design variable.

The camberline of Figure 34 combined with the thickness distribution specified, yields the airfoil shape shown in Figure 35. It is important to note that the airfoil shape of Figure 35 requires that the camberline be smooth and continuous in order to generate a valid airfoil shape. Thus a large emphasis is placed on developing blade geometry parameterization schemes that will consistently yield smooth as well as continuous camberlines. The next section discusses the blade parameterization scheme used to develop the blade profiles at the radial span layers.

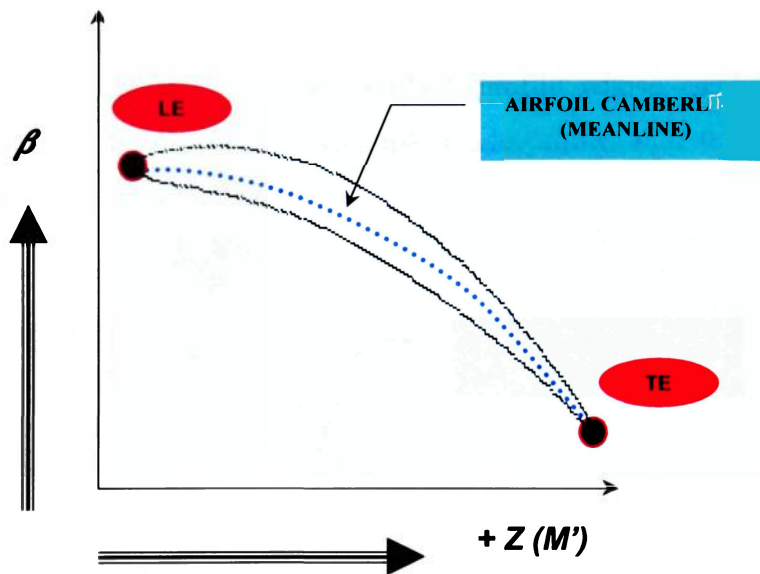


Figure 35: Camberline with Imposed Thickness Distribution

3.2.1 Blade Airfoil/Twist Parameterization Using Bezier Curves

The generation of a smooth airfoil shape as in Figure 35 is highly critical to the development of a valid three dimensional blade geometry. Since the thickness distribution is constant, the airfoil shape is essentially a product of the shape of the camberline. Hence, the objective of the parameterization scheme is to consistently generate smooth and continuous airfoil camberlines.

The parameterization scheme devised is based on generating the airfoil camberlines at each span layer using bezier control polygons. Imposing the bezier control polygons on the blade profile of Figure 35 yields a blade profile whose camberline is consistently smooth and continuous. An example blade profile with the imposed bezier control polygon is shown in below

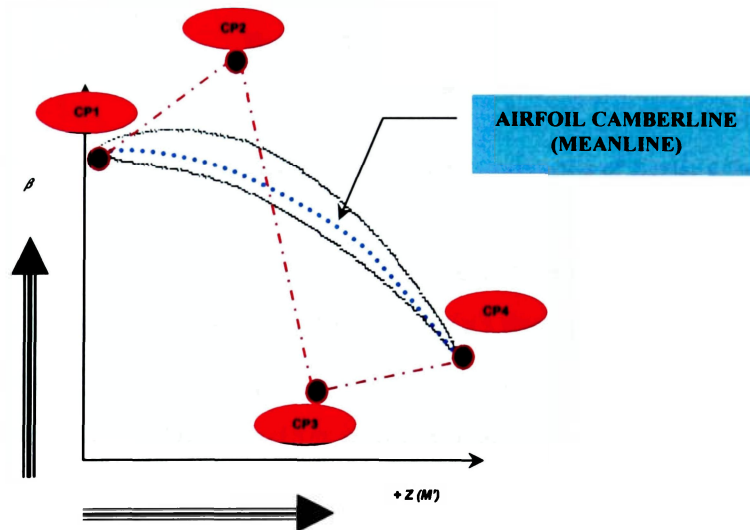


Figure 36: Camberline Generation using a Bezier Control Polygon

The use of the bezier control polygon in Figure 36 is restricted to generating only the camberline of the blade profile. The actual airfoil shape is still generated by imposing a standard NACA0006 thickness distribution on the meanline curve generated using the bezier control points. By using the Bezier curve to generate blade profiles at all span layers, a parameterization scheme can then be developed based on using the Bezier control points (CPs) to generate different blade geometries. The various

geometries can then be passed to the CFD analyses module which will evaluate the performance of the blade design based on the total efficiency.

The blade geometry controls at each span layer are the 4 control points of the bezier control polygon as shown in Figure 36. Repeating the blade profile generation scheme all five span layers in the radial direction yields a total of 20 bezier control points. We may then consider a blade geometry “generalized grid” made up of the four bezier control points at each span layer of the blade.

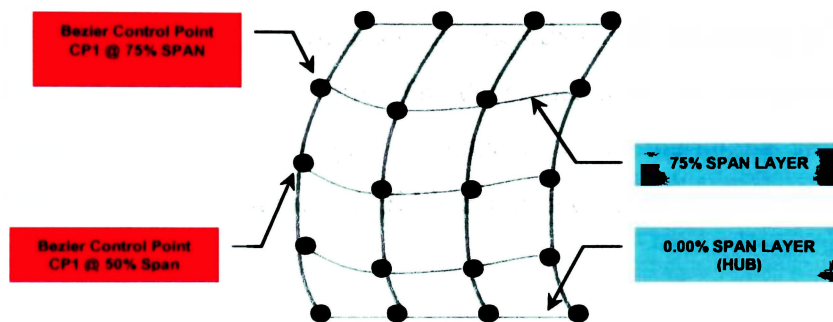


Figure 37: Generalized Blade Geometry CP Grid (Composed of Bezier CPs)

The blade geometry CP grid in Figure 37 represents the general blade shape parameterization strategy to be employed in the MDO design methodology. At each span layer, the Bezier CPs are modified in order to generate various airfoil shapes at their respective span layers. The variation in the 3D geometry of the blade is then a consequence of the radial interpolation between the modified blade profiles.

In the development of the geometry CP grid of Figure 37, we can now consider how this parameterization scheme is translated into design variables that can be manipulated by the selected MDO algorithm. The primary criteria in selecting design parameters is that all effort should be made to minimize the number of design variables, thereby increasing the effectiveness the MDO algorithm in locating globally optimal solutions. This is especially the case for the selected RSO algorithm where the “curse of dimensionality” requires that fewer design variables be used with the more accurate, high order response surface models.

In the blade geometry CP grid of Figure 37, each bezier CP is located using a spatial coordinate pair. The first element in the coordinate pair is the “x” coordinate which lies along the axial/meridional axis (\underline{M}') of the machine. The second element of the CP coordinate pair is the “y” coordinate (β). This coordinate element is the angular location of the bezier CP in the tangential direction, on the plane of constant radius (span layer). Thus, in order to locate the four bezier CPs at each span layer in meridional space, the (\underline{M}', β) coordinate pair for each CP must be provided.

The blade geometry CP grid of Figure 37 is decomposed into two component grids: once consisting of the \underline{M}' -coordinates and the other grid consisting of the β angular coordinates of each CP in the grid. The decomposition into component coordinate grids is shown below. The blue grid is composed of the meridional/axial coordinates (\underline{M}') of each Bezier CP and the red grid is composed of the angular coordinates (β) of each CP.

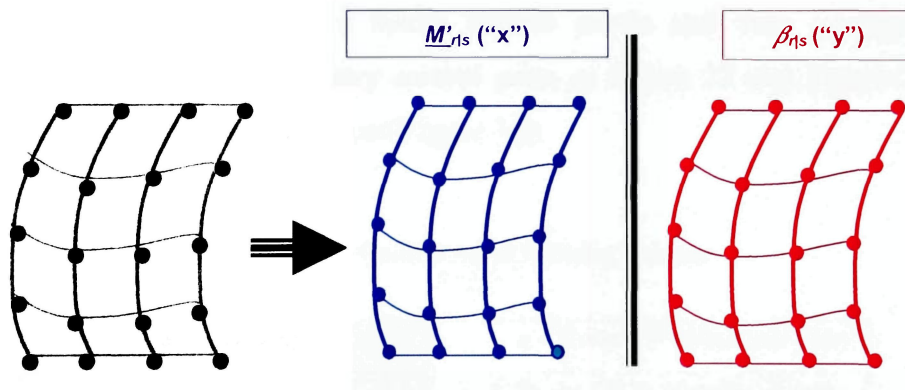


Figure 38: Generalized Blade CP Grid Coordinate Components

The decomposition of the CP grid into its coordinate components allows various assumptions to be made to reduce the total number of design variables. In designating the meridional coordinates (blue grid) of the Bezier control points, the actual value of the meridional coordinate is non-dimensionalized by the meridional coordinate of the trailing edge (TE) at the respective span layers. In addition, the meridional coordinate at all span layers is assumed to start at the origin. We also introduce a standard

indexing convention in order to locate the position of the Bezier Control Points on the generalized CP parameterization grid thus:

$$\underline{M}'_{r|s} = \left(\frac{M'_{r|s}}{M'_{r|TE}} \right) * 100 \quad (3.3)$$

Where:

- r - Bezier control point span layer index
- s - Bezier control point meridional Index
- TE – Trailing edge
- $M'_{r|s}$ - Meridional coordinate of CP $_{r|s}$
- $M'_{r|TE}$ - Meridional coordinate of TE at span layer r

3.2.1.a Bezier Control Point Indexing Scheme

In order to locate the respective bezier control points and their corresponding coordinates on the various geometry control grids in Figure 37 and Figure 38, the following indexing scheme is used (ref Figure 36):

Table 6: Bezier Control Point Indexing Scheme

<i>r</i> – Span Layer Radial Index:	<i>s</i> – Bezier CP Meridional Index
$r = 1$: 0% Span Layer (Hub)	$s = 1$: Bezier CP #1 (Leading Edge)
$r = 2$: 25% Span Layer	$s = 2$: Bezier CP #2
$r = 3$: 50% Span Layer	$s = 3$: Bezier CP #3
$r = 4$: 75% Span Layer	$s = 4$: Bezier CP #4 (Trailing Edge)
$r = 5$: 100% Span Layer (Shroud)	

These subscripts are used in order to identify the location of CPs within the blade geometry. In equation 3.3, the meridional coordinate of each bezier CP ($\underline{M}'_{r|s}$) is non-dimensionalized by the meridional coordinate of the trailing edge (TE) at the corresponding span layer ($\underline{M}'_{r|4}$ or $\underline{M}'_{r|TE}$).

3.2.1.b Meridional ($\underline{M}'_{R|S}$) Bezier Polygon MDO Design Variables

The non-dimensionalization of the meridional ($\underline{M}'_{r|s}$) coordinates of the Bezier control points in equation 3.3 implies that the axial coordinate of the trailing edge control point will always be assigned a constant value of 100% at all span layers, i.e. $\underline{M}'_{r|4} = 100$. The assumption that the LE meridional coordinate always lies at the origin also allows a constant value of 0% to be assigned as the meridional coordinate for the LE Bezier CPs at all span layers (i.e. $\underline{M}'_{r|1} = 0$).

The constant LE and TE meridional coordinates consequently reduce the number of true design variables available in the meridional (blue) geometry CP grid. The meridional design variables become the non-constant coordinates of the interior Bezier CPs. These design variables can be manipulated by the MDO search algorithm in the search for the optimal design. Figure 38 shows the constant LE and TE meridional coordinates deactivated in the blue geometry control grid leaving only the interior Bezier CPs active as design variables.

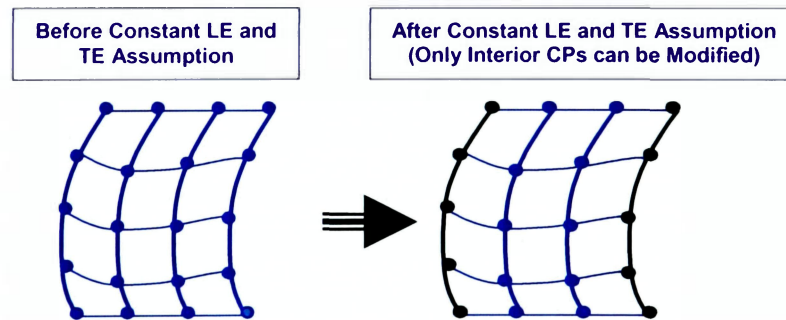


Figure 39: MDO Design Variables for Meridional Blade CP Grid

The consequence of the constant LE and TE meridional coordinates is that the number of design variables is reduced from 20 to 10 as illustrated in Figure 39. Since the meridional coordinates of the leading and trailing edges are held constant, these modifiable coordinates are used to control the interior meridional shape of the blade geometry.

A further modification must be made to the final control grid of Figure 39 which contains the modifiable meridional coordinates. The need for this modification arises from a close observation of the CPs of the Bezier polygons used to generate the blade profiles at the various span layers.

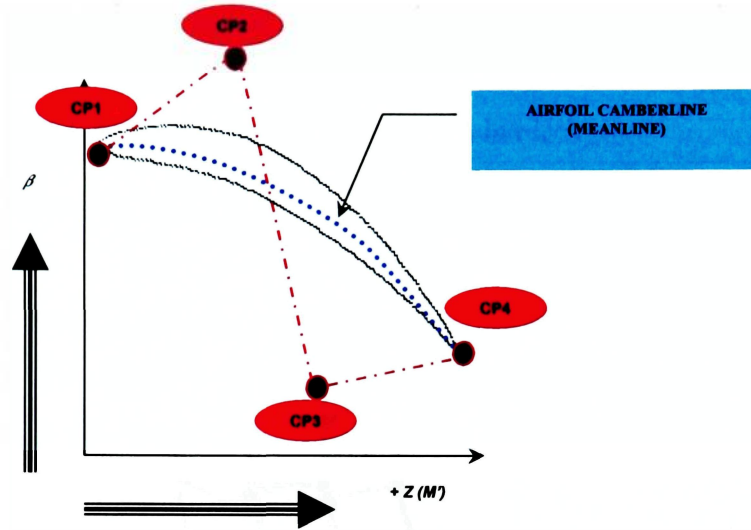


Figure 40: Blade Profile Generation Using Bezier Polygon

In generating a valid bezier polygon (dotted red line in Figure 40), CFX-Bladegen requires that the third control point (CP3) lies to the right side (aft) of the second bezier control point (CP2). This implies that meridional coordinate of CP3 must be greater than that of CP2. It becomes necessary to extend the parameterization scheme to accommodate this requirement. This is accomplished by employing a *displacement factor* to generate the meridional location of CP3 based on the location of CP2 as follows:

$$\underline{M}'_{r3} = \underline{M}'_{r2} + (90 - \underline{M}'_{r2})\delta_r + 5 \quad (3.4)$$

$$\text{Limits:} \quad 10 \leq \underline{M}'_{r3} \leq 80$$

$$0 \leq \delta_r \leq 1$$

Where: δ_r = “Displacement Factor” at Span Layer r

The relationship of equation 3.4 essentially assures that the meridional coordinate of CP3 in Figure 40 will always occur to the right of that of CP2 as required by the CFX-Bladegen. Consequently, the meridional design variables for the 3D blade geometry are composed of the meridional coordinates of CP2 ($\underline{M}'_{r|2}$) as well as the displacement factor (δ_r) used to determine the location of CP3 at each span layer. The meridional CP grid of Figure 39 is updated in Figure 41 and shows the variable CPs and *displacement factors* in blue and the constant CPs (leading and trailing edges) in black.

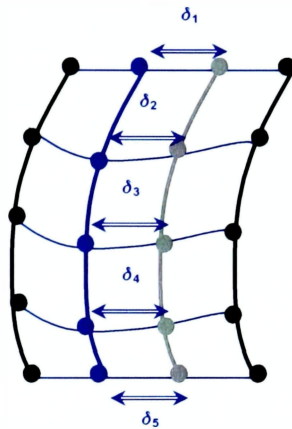


Figure 41: Meridional Design Variables on Blade CP Grid w/“Displacement Factor”

Note that in the updated blade control grid of Figure 41, all the CPs along the radial line that runs through the third Bezier Control point at all span layers is “grayed” out to indicate that CP3 at all span layers remains a design parameter. However, the actual meridional location of CP3 is determined from equation 3.4 using the *displacement factor* and CP2 coordinates at corresponding span layers. The meridional coordinate parameterization scheme yields a total of 10 design variables ($\underline{M}'_{r|s}$ and δ_r ; $1 \leq r \leq 5$; $s = 2$) for the 3D blade geometry.

3.2.1.c Angular ($\beta_{r|s}$) Bezier Polygon MDO Design Variables

The generalized blade geometry CP grid shown in Figure 38 is also composed of the tangential (red) coordinate components. These correspond to the angular (β) coordinate of the CPs in the sample bezier polygon of Figure 40.

Due to the coordinate system used in the generation of the blade geometry, the angular (β) coordinates of the LE and TE control points correspond to the local angle of the camberline of the blade profile. These angles are the relative blade angles usually calculated from the radial equilibrium and vortex blading techniques employed in the generation of the initial blade design. As a result of this observation, the β coordinate of the LE bezier control points at all span layers are held fixed (constant).

The β coordinate components of the generalized blade CP grid are shown below, before and after the LE coordinates are deactivated. The consequence of the assumption of a constant LE angular location is to reduce the number of design variables from 20 in the original blade CP grid to 15 in the final CP grid of Figure 42. Variable angular coordinates are shown in red and constant/fixed coordinates are shown in black. The parameterization scheme yields a total of 15 tangential CP coordinates ($\beta_{r|s}$: $1 \leq r \leq 5$; $2 \leq s \leq 4$) as additional MDO blade design variables.

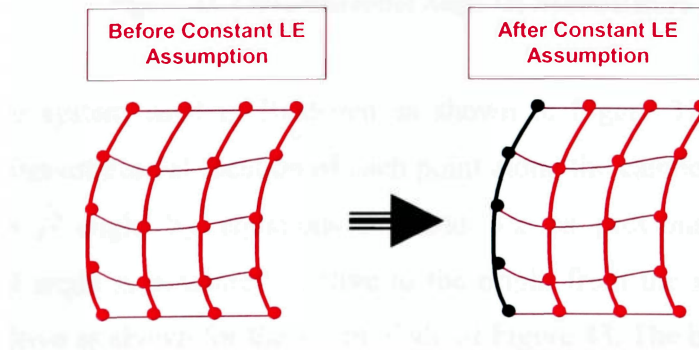


Figure 42: MDO Design Variables for Angular (β) Blade CP Grid

3.2.2 Blade Sweep Parameterization - Spanwise LE Angular Coordinate

The parameterization scheme discussed in section 3.2.1 is primarily limited to the airfoil camber variation at various span layers. The resulting effect is 3D blade geometries composed of spanwise varying camber/airfoil shapes as well as variations in the radial twist distribution of the blade geometry.

In addition to the twist and airfoil variation of the blade geometry, the parameterization of the blade geometry is extended to allow for variation in the radial sweep distribution of the blade geometry. This is achieved by taking advantage of the native coordinate system used in the Bladegen turbomachinery design tool.

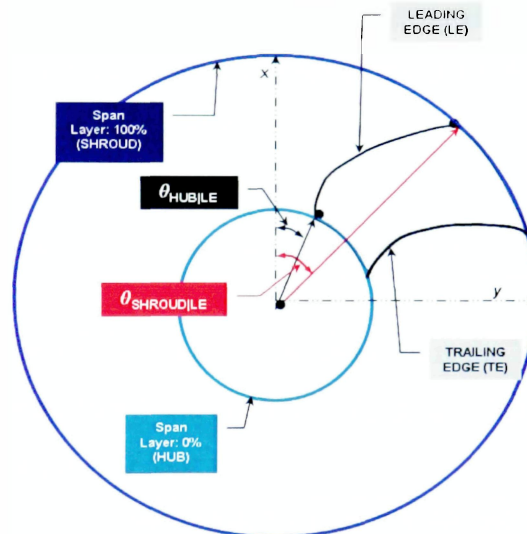


Figure 43: Circumferential Angle (θ) Nomenclature

The coordinate system used in Bladegen as shown in Figure 31 uses the angle θ to establish the circumferential location of each point along the camberline P. This angle is related to the β angle by equations 3.1 and 3.2 as previously discussed. The circumferential angle is measured relative to the origin from the x-axis towards y-axis defined as positive as shown for the swept blade of Figure 43. The backward swept blade of Figure 43 shows the location on the circumferential plane of the LE at the hub (0% span) and the shroud (100% span). A close examination of Figure 43 reveals that a

variation in the LE circumferential coordinate at several span layers running from hub to shroud is equivalent to imposing a LE sweep variation on the blade. To generate the blade geometry for the benchmark MDO design, five span layers are used. The five span layers are evenly spaced in 25% increments from hub to shroud. Hence, the leading edge sweep of the blade is parameterized by specifying the circumferential location (θ_r) of the leading edge at each span layer. The LE sweep parameterization scheme yields a total of 5 MDO design variables (θ_r : $1 \leq r \leq 5$).

All the design variables necessary to geometrically define any 3D blade geometry for the MDO study have been identified. Combining the 10 CP meridional ($\underline{M}'_{r|s}$) variables with the 15 tangential ($\beta_{r|s}$) coordinates, in addition to the 5 LE circumferential (θ_r) variables, yields a total of 30 design variables for the complete 3D shape parameterization of the blade geometry. Table 7 provides a summary of all the geometric design variables in the developed blade parameterization scheme.

Table 7: Summary of Blade Geometry Parameterization Design Variables

Variable	Description	Quantity
$\underline{M}'_{r s}$	Meridional location of the CPs used to generate the blade profiles at each of the 5 span layers. Only $\underline{M}'_{r s}$ values for CP2 are varied. ($\underline{M}'_{r s}$: $1 \leq r \leq 5$; $s = 2$)	5
δ_r	<i>Displacement Factor</i> used to deduce the meridional location of the 3 rd bezier CP (equation 3.4) used to generate the blade profiles at each span layer. (δ_r : $1 \leq r \leq 5$)	5
$\beta_{r s}$	Tangential angular locations of the bezier CPs on their respective span layers. The LE β location on all span layers ($\beta_{r 1}$) assumed constant. ($\beta_{r s}$: $1 \leq r \leq 5$; $2 \leq s \leq 4$)	15
θ_r	Blade profile LE circumferential (sweep) location at each span layer. (θ_r : $1 \leq r \leq 5$)	5
	Total Blade Geometry Design Variables	30

3.2.3 Dynamic Constraints on Blade Geometry Parameterization

The parameterization scheme described in the previous sections enable the 3D blade geometry to be described in terms of variables that can be modified by the selected MDO algorithm to achieve the best fan design. However, in addition to the basic design variables obtained from the parameterization scheme, it is essential to impose boundaries or constraints on the design variables. This is necessary in order to prevent the optimizer from exploring invalid fan blade designs such as the excessively swept and cambered blade in Figure 44.

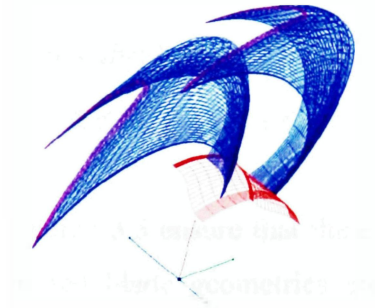


Figure 44: Invalid Blade Geometry Obtained from Unbounded Parameterization

The nature of the constraints must be dynamic as they do not set a specific range on the variables themselves. Rather, they control how much one design variable may vary relative to other design variables, hence the use of the “dynamic constraint” nomenclature. The dynamic constraints on the parameterization scheme are introduced on each group of design variables obtained from the blade parameterization scheme. The limits on the design variables can be classified into three general groups namely:

1. Constraints between index-similar bezier CPs on adjacent span layers.
2. Constraints between adjacent bezier CPs on the same span layer.
3. Constraints between LE circumferential coordinates on adjacent span layers.

The following sections discuss each constraint category in detail.

3.2.3.a Dynamic Constraints on Tangential ($\beta_{r|s}$) Design Variables

The angular design variables for the Bezier CPs span the blade geometry from the second CP index ($s = 2$) to the trailing edge ($s = 4$) and also from the Hub ($r = 1$) to the Shroud ($r = 5$) using the parameterization scheme of Figure 37. Constraints are imposed on adjacent pairs of tangential ($\beta_{r|s}$) bezier CP coordinates in both the radial and axial directions. In the radial direction, the dynamic constraints are imposed by limiting the maximum difference in the tangential coordinates of bezier control points with the same index ($r|s$) pair that lie on adjacent span layers:

$$0 \leq \text{abs}(\beta_{r|s} - \beta_{r-1|s}) \leq 30 \quad (3.5)$$
$$2 \leq r \leq 5, 2 \leq s \leq 4$$

These limits imposed using equation 3.5 ensure that the excessive layer-to-layer angular offsets that create highly distorted blade geometries are avoided. This layer-to-layer offset limit is insufficient as within each span layer, constraints must be imposed on the maximum offset between adjacent bezier CPs that lie on the same span layer. The second set of dynamic constraints ensures that blade shapes with excessively skewed camberlines are avoided. For this case, the following limits are imposed on adjacent CPs located on the same span layer:

$$0 \leq \text{abs}(\beta_{r|s} - \beta_{r|s-1}) \leq 30 \quad (3.6)$$
$$1 \leq r \leq 5, 2 \leq s \leq 4$$

With the specification of the limits in equations 3.5 and 3.6, the parameterization of the angular coordinates of the Bezier control points is complete. Note that the meridional coordinates do not require span-specific constraints such as those in equation 3.5 and 3.6. This is as a result of the already specified limits of the *displacement factor* parameterization approach of equation 3.4. Layer-to-layer constraints are also ignored for the meridional coordinates of the bezier control points.

3.2.3.b Dynamic Constraints on LE Circumferential (θ_r) Design Variables

The blade leading edge sweep parameterization scheme as previously discussed requires that the LE shape of the blade be specified using 5 angular coordinates. Each coordinate corresponds to the circumferential location of the blade leading edge at each span layer.

Similar to the dynamic constraints imposed on the layer-to-layer offset of the tangential ($\beta_{r|s}$) coordinates of the bezier CPs, limits are created to constrain the relative offset between the LE circumferential coordinates at adjacent span layers as follows:

$$0 \leq \text{abs}(\theta_r - \theta_{r-1}) \leq 30 \quad (3.7)$$

The boundaries of equation 3.7 complete the set of constraints necessary to transform the parameterization scheme into a considerably robust scheme capable of generating physically reasonable blade designs, while maintaining enough variety in the design space such that enough blade designs are available to be evaluated by the optimization algorithm.

At this stage the development of a robust blade geometry parameterization scheme is complete and the next phase in the development of the MDO design methodology may be considered. Of primary importance are the singular components that are integrated to create the MDO environment and how effective intercommunication between these components is achieved. In the following sections, the details of the developed MDO design environment are discussed, including the implementation of the dynamics variable constraints in VisualDoc.

3.3 MDO Environment Implementation (VisualDoc)

The development of a parameterization scheme for the 3D blade geometry allows for the details involved in the implementation of the MDO design scheme to be developed. As previously discussed, the selected optimization package is the VisualDoc Multidisciplinary Optimization Software package from Vanderplaats Research and Development Inc., Colorado Springs, Colorado.

VisualDoc is a GUI based multidisciplinary optimization tool which allows the user to interface with third-party analyses software provided the output from the analyses can be specified using simple ASCII text data files. Intercommunication between the various components (blocks) of the MDO design scheme is primarily accomplished using these ASCII text files.

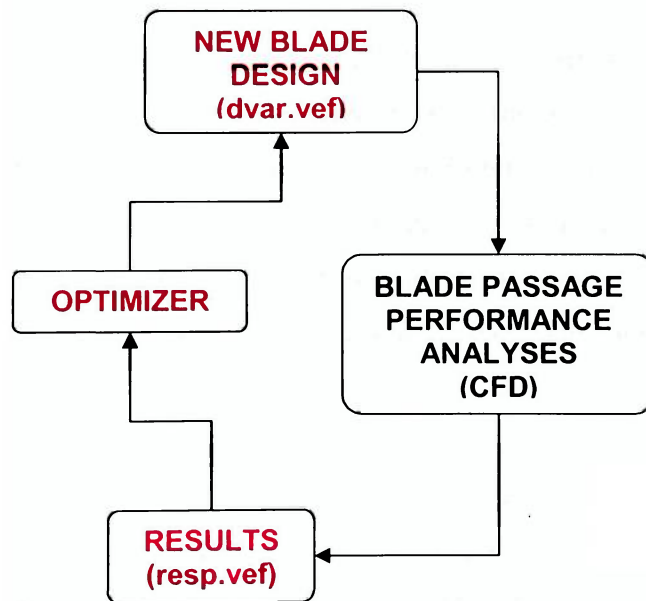


Figure 45: MDO Environment Component Interaction

The design variables are specified in the main VisualDoc graphical user interface as shown in Figure 46 and Figure 47. In addition to the 30 design variables already identified in Table 7, the rotational rate (RPM) and the number of blades in the fan cascade (Num_Blades) are added as design variables for a **total of 32 design variables**. Several variable types are available in VisualDoc including constant, integer and continuous design variables. Table 8 includes several “non-traditional” design variables types also available in VisualDoc:

Table 8: Non-traditional Variable Types in VisualDoc

Variable Type	Description
Pass/Fail	Used to form Boolean (T/F) type constraints. Useful in indicating to the optimizer if an analysis completed successfully or not.
Synthetic	Used when one “variables” can be defined as an explicit linear/non-linear function of other design variables.

Figure 46 and Figure 47 show a listing of all the design variables as they are entered in the VisualDoc GUI. The designation of the type of the design variables is done in the second column, the lower bound, initial value and upper bound of each variable is also explicitly specified in their respective columns. The nomenclature for Figure 46 and Figure 47 needs to be discussed in a bit more detail so as to clarify how each variable corresponds to the design variables identified in the parameterization scheme of the previous section.

Table 9 summarizes the correlation between the design variables of the parameterization scheme and the parameters specified in Figure 46 and Figure 47. Parameters that appear in the listing but that are not design variables are identified with the NDV designation in Table 9. The variable *type* listings of Table 9 take precedence over the designations in Figure 46 and Figure 47. Discrepancies in the *type* listings between Table 9 and Figure 46/Figure 47 are a result of the subsequently discussed optimization strategy utilized in the benchmark MDO study.

Index	Name	Type	Objective	Low Bound	Initial Value	Upp Bound
1	Mode	Constant	<input type="checkbox"/>	None	1.00	None
2	Num_Layers	Constant	<input type="checkbox"/>	None	5.00	None
3	Curve_Discret	Constant	<input type="checkbox"/>	None	100.00	None
4	Num_Blades	Discrete	<input type="checkbox"/>	2.00	9.00	25.00
5	RPM	Continuous	<input type="checkbox"/>	-1500.00	-1140.00	-500.00
6	Theta_Offset_Lim	Constant	<input type="checkbox"/>	None	55.00	None
7	Beta_Offset_Lim_L1	Constant	<input type="checkbox"/>	None	85.00	None
8	Beta_Offset_Lim_L2	Constant	<input type="checkbox"/>	None	85.00	None
9	Beta_Offset_Lim_L3	Constant	<input type="checkbox"/>	None	85.00	None
10	Beta_Offset_Lim_L4	Constant	<input type="checkbox"/>	None	85.00	None
11	Beta_Offset_Lim_L5	Constant	<input type="checkbox"/>	None	85.00	None
12	XBez_Displ_L1	Constant	<input type="checkbox"/>	0.00	0.56519	1.00
13	XBez_Displ_L2	Constant	<input type="checkbox"/>	0.00	0.33585	1.00
14	XBez_Displ_L3	Constant	<input type="checkbox"/>	0.00	0.45031	1.00
15	XBez_Displ_L4	Constant	<input type="checkbox"/>	0.00	0.90369	1.00
16	XBez_Displ_L5	Constant	<input type="checkbox"/>	0.00	0.96007	1.00
17	Beta_BezX_1_2	Constant	<input type="checkbox"/>	10.00	45.677	80.00
18	Beta_BezX_2_2	Constant	<input type="checkbox"/>	10.00	46.321	80.00
19	Beta_BezX_3_2	Constant	<input type="checkbox"/>	10.00	51.433	80.00
20	Beta_BezX_4_2	Constant	<input type="checkbox"/>	10.00	15.595	80.00
21	Beta_BezX_5_2	Constant	<input type="checkbox"/>	10.00	34.878	80.00
22	Beta_BezY_1_1	Constant	<input type="checkbox"/>	20.00	68.748	80.00
23	Beta_BezY_1_2	Constant	<input type="checkbox"/>	10.00	66.00	80.00
24	Beta_BezY_1_3	Constant	<input type="checkbox"/>	10.00	30.00	60.00
25	Beta_BezY_1_4	Constant	<input type="checkbox"/>	-10.00	-4.5492	30.00
26	Beta_BezY_2_1	Constant	<input type="checkbox"/>	30.00	77.095	80.00
27	Beta_BezY_2_2	Constant	<input type="checkbox"/>	10.00	47.394	80.00
28	Beta_BezY_2_3	Constant	<input type="checkbox"/>	10.00	18.453	70.00
29	Beta_BezY_2_4	Constant	<input type="checkbox"/>	10.00	18.809	50.00
30	Beta_BezY_3_1	Constant	<input type="checkbox"/>	30.00	76.125	80.00
31	Beta_BezY_3_2	Constant	<input type="checkbox"/>	20.00	54.467	80.00
32	Beta_BezY_3_3	Constant	<input type="checkbox"/>	10.00	42.128	80.00
33	Beta_BezY_3_4	Continuous	<input type="checkbox"/>	10.00	43.612	80.00
34	Beta_BezY_4_1	Constant	<input type="checkbox"/>	30.00	71.92	80.00
35	Beta_BezY_4_2	Constant	<input type="checkbox"/>	30.00	74.772	80.00
36	Beta_BezY_4_3	Constant	<input type="checkbox"/>	20.00	65.192	80.00
37	Beta_BezY_4_4	Continuous	<input type="checkbox"/>	20.00	39.00	80.00
38	Beta_BezY_5_1	Constant	<input type="checkbox"/>	20.00	60.288	80.00
39	Beta_BezY_5_2	Constant	<input type="checkbox"/>	10.00	80.00	80.00
40	Beta_BezY_5_3	Constant	<input type="checkbox"/>	30.00	79.322	80.00
41	Beta_BezY_5_4	Constant	<input type="checkbox"/>	30.00	67.00	90.00
42	Theta_LE_L1	Constant	<input type="checkbox"/>	-30.00	-3.00	30.00
43	Theta_LE_L2	Constant	<input type="checkbox"/>	-30.00	0.46969	30.00

Figure 46: VisualDoc Design Variables (Input) Specification (Part I)

Index	Name	Type	Objective	Low Bound	Initial Value	Upp Bound
40	Beta_BezY_5_3	Constant	<input type="checkbox"/>	30.00	79.322	80.00
41	Beta_BezY_5_4	Constant	<input type="checkbox"/>	30.00	67.00	90.00
42	Theta_LE_L1	Constant	<input type="checkbox"/>	-30.00	-3.00	30.00
43	Theta_LE_L2	Constant	<input type="checkbox"/>	-30.00	0.46969	30.00
44	Theta_LE_L3	Constant	<input type="checkbox"/>	-30.00	-2.3079	30.00
45	Theta_LE_L4	Constant	<input type="checkbox"/>	-30.00	-9.4676	30.00
46	Theta_LE_L5	Constant	<input type="checkbox"/>	-30.00	-20.00	30.00
47	SL112	Synthetic	<input type="checkbox"/>	0.00	2.748	40.00
48	SL123	Synthetic	<input type="checkbox"/>	0.00	36.00	40.00
49	SL134	Synthetic	<input type="checkbox"/>	0.00	34.549	40.00
50	SL212	Synthetic	<input type="checkbox"/>	0.00	29.701	40.00
51	SL223	Synthetic	<input type="checkbox"/>	0.00	28.941	40.00
52	SL234	Synthetic	<input type="checkbox"/>	0.00	0.356	40.00
53	SL312	Synthetic	<input type="checkbox"/>	0.00	21.658	40.00
54	SL323	Synthetic	<input type="checkbox"/>	0.00	12.339	40.00
55	SL334	Synthetic	<input type="checkbox"/>	0.00	1.484	40.00
56	SL412	Synthetic	<input type="checkbox"/>	0.00	2.852	40.00
57	SL423	Synthetic	<input type="checkbox"/>	0.00	9.58	40.00
58	SL434	Synthetic	<input type="checkbox"/>	0.00	26.192	40.00
59	SL512	Synthetic	<input type="checkbox"/>	0.00	19.712	40.00
60	SL523	Synthetic	<input type="checkbox"/>	0.00	0.678	40.00
61	SL534	Synthetic	<input type="checkbox"/>	0.00	12.322	40.00
62	RL112	Synthetic	<input type="checkbox"/>	0.00	8.347	30.00
63	RL123	Synthetic	<input type="checkbox"/>	0.00	0.97	30.00
64	RL134	Synthetic	<input type="checkbox"/>	0.00	4.205	30.00
65	RL145	Synthetic	<input type="checkbox"/>	0.00	11.632	30.00
66	RL212	Synthetic	<input type="checkbox"/>	0.00	18.606	30.00
67	RL223	Synthetic	<input type="checkbox"/>	0.00	7.073	30.00
68	RL234	Synthetic	<input type="checkbox"/>	0.00	20.305	30.00
69	RL245	Synthetic	<input type="checkbox"/>	0.00	5.228	30.00
70	RL312	Synthetic	<input type="checkbox"/>	0.00	11.547	30.00
71	RL323	Synthetic	<input type="checkbox"/>	0.00	23.675	30.00
72	RL334	Synthetic	<input type="checkbox"/>	0.00	23.064	30.00
73	RL345	Synthetic	<input type="checkbox"/>	0.00	14.13	30.00
74	RL412	Synthetic	<input type="checkbox"/>	0.00	23.358	30.00
75	RL423	Synthetic	<input type="checkbox"/>	0.00	24.803	30.00
76	RL434	Synthetic	<input type="checkbox"/>	0.00	4.612	30.00
77	RL445	Synthetic	<input type="checkbox"/>	0.00	28.00	30.00
78	Rtheta12	Synthetic	<input type="checkbox"/>	0.00	3.4697	30.00
79	Rtheta23	Synthetic	<input type="checkbox"/>	0.00	2.7776	30.00
80	Rtheta34	Synthetic	<input type="checkbox"/>	0.00	7.1597	30.00
81	Rtheta45	Synthetic	<input type="checkbox"/>	0.00	10.532	30.00

Figure 47: VisualDoc Design Variables (Input) Specification (Part II)

Table 9: VisualDoc Design Variable Nomenclature

PARAMETER	DESIGN VARIABLE	TYPE	DESCRIPTION
Mode	NDV	Constant	Created for future work to allow alternate parameterization schemes. For current (default) scheme, mode =1 (constant)
Num_Layers	NDV	Constant	Number of Spanwise Layers (5), constant. Used to deduce the number of Bezier CPs used in the blade parameterization scheme.
Curve_Discret	NDV	Constant	Used to generate the underlying camberline of blade profiles from Bezier CPs. Default (constant) = 100
Num_Blades	# of Blades	Integer	Number of Blades in complete fan cascade
RPM	RPM	Continuous	Rotational rate of fan in RPM (negative from right-hand rule)
Theta_Offset_Lim	NDV	Constant	Not implemented. For future use. NOTE: this is NOT equivalent to the sweep variable (θ) bounds of section 3.2.3.b
Beta_Offset_Lim_Ln (n = 1..5)	NDV	Constant	Not Implemented. For Future use. NOTE: this is NOT equivalent to the Bezier CP angular (β) coordinate bounds of section 3.2.3.a
xBez_Displ_Ln (n = 1..5)	δ_r ($1 \leq r \leq 5$)	Continuous	The “Displacement Factor” used to determine the meridional coordinate of the 3 rd Bezier Polygon CP (M'_{r3}) using equation 3.4, at each span layer. [ref. Figure 40, Figure 41]
BetaX_n_2 (n = 1..5)	M'_{r2}	Continuous	The meridional coordinate of the 2 nd Bezier Polygon CP (M'_{r2}) at each span layer. [ref. Figure 40, Figure 41]
BetaY_n_m (n = 1..5) (m=1..4)	β_{rs} ($1 \leq r \leq 5$) ($2 \leq s \leq 4$)	Continuous	The tangential coordinate of the bezier CPs (β_{rs}). A continuous type except the angular location of the camberline LE (β_{r1}), which is constant [ref. Figure 40, Figure 42]
Theta_LE_Ln (n = 1..5)	θ_r ($1 \leq r \leq 5$)	Continuous	The circumferential location of the LE of the blade profile camberline at each span layer. (ref Figure 43)
SL[xyz]	Dynamic Constraint	Synthetic	Tangential coordinate constraints for adjacent bezier CPs that reside on the same span layer i.e. $\text{abs}(\beta_{xiz} - \beta_{xjy})$ is verified to be within limits specified by equation 36. [ref section 3.2.3a]
RL[ijk]	Dynamic Constraint	Synthetic	Tangential coordinate constraints for bezier CPs with the same index pair that lie on adjacent span layers, i.e. $\text{abs}(\beta_{kii} - \beta_{jii})$ is computed at every iteration and verified to be within limits set by equation 35. [ref. section 3.2.3.a]
Rtheta[uv]	Dynamic Constraint	Synthetic	LE sweep constraints (max offset) between circumferential angle (θ_r) at adjacent span layers i.e. $\text{abs}(\theta_v - \theta_u)$ is verified to be within limits set by equation 3.7. [ref. section 3.2.3.b]

As previously discussed, it is necessary to implement the constraints specified by equations 3.5 through 3.7, to ensure that only feasible blade designs are investigated by optimizer. The only caveat in implementing these constraints on the design variables is that they are not fixed bounds, i.e. they only specify what the maximum difference between the two design parameters should be. This creates the need to dynamically monitor the values of the design variables are chosen and ensure they conform to the specified limits. These dynamic bounds are implemented in the MDO environment using special variable types in VisualDoc known as synthetic variables.

Synthetic variables are considered “fake” variables as they are typically expressed as explicit functions of “true” design variables. An advantage of synthetic variables is that they are computationally efficient as they are handled internally to VisualDoc, but most importantly they do not affect the dimension of the problem in terms of the number of **independent** design variables.

The synthetic variables ensure that the relative offsets between variables are within the ranges specified by equations 3.5 through 3.7. Highly distorted blade shapes such as the example in Figure 44 are consequently eliminated from being investigated. In essence, implementing the dynamic constraints using the synthetic variables eliminates infeasible regions of the design space that contain distorted, impractical blade geometries.

Index	Name	Type	Objective	Constraint	Low Bound	Upp Bound
1	PFBladetest	Pass/Fail	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0.00	None
2	PFGGrid	Pass/Fail	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0.00	None
3	PFRResults	Pass/Fail	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0.00	None
4	MassFlowRate	Independent	<input type="checkbox"/>	<input type="checkbox"/>	None	None
5	VolFlowRate	Independent	<input type="checkbox"/>	<input checked="" type="checkbox"/>	3.00	6.00
6	TotBlTorq	Independent	<input type="checkbox"/>	<input type="checkbox"/>	None	None
7	Headrise	Independent	<input type="checkbox"/>	<input type="checkbox"/>	None	None
8	InFlowCoeff	Independent	<input type="checkbox"/>	<input type="checkbox"/>	None	None
9	ExFlowCoeff	Independent	<input type="checkbox"/>	<input type="checkbox"/>	None	None
10	HeadCoeff	Independent	<input type="checkbox"/>	<input type="checkbox"/>	None	None
11	TotalEffic	Independent	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0.25	1.00
12	StaticEffic	Independent	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0.00	1.00

Figure 48: VisualDoc Response Specification

Table 10: Pass/Fail Response Parameters

VARIABLE	DESCRIPTION
PFBladetest	Currently not implemented. Provided for manual verification of the blade geometry such as in future blade parameterization schemes. Default to 1 (pass) [1 = pass, -1 = fail]
PFGrid	Results of test which checks to ensure a valid grid file exists as indicative of a successful grid generation. Implemented in MATLAB® m-file, <i>gridtest.m</i> [1 = pass, -1 = fail]
PFResults	Results of test block to evaluate validity of CFD results. Implemented in MATLAB® m-file, <i>restest.m</i> [1 = pass, -1 = fail]

The responses to the optimization module (VisualDoc) are specified as shown in Figure 48. For the benchmark case, the responses include several *Pass/Fail* parameters used to indicate when the CFD analysis of a design has failed. Three *Pass/Fail* parameters are used to detect not only a failure of the CFD response analyses but also the underlying reason behind the failure as shown in Table 10.

The CFD response analysis of each fan design is strongly dependent on the successful execution of several steps in sequence (pre-processing, grid generation, etc). The use of the three *Pass/Fail* responses improves the robustness of the MDO design environment by enhancing tolerance to failures in the response analyses. The optimizer is consequently able to thoroughly explore the design space, further eliminating and invalid portions of the design space from being explored by the selected MDO algorithm.

In the response specification of Figure 48, several physical parameters are also specified as response values from the CFD analysis. These include the volume flow rate (CFM), headrise, static efficiency as well as the target function (total efficiency). Each iteration, the optimizer selects a combination of design variables from the available parameters listed in Figure 46 and Figure 47. The design is then passed to the CFD analyses block to determine the flow field solution, from which the total efficiency is computed.

The optimizer expects to receive a list of responses in the same order as the parameters listed in the response specification of Figure 48. The transfer of the CFD analysis results and other responses is done using the response ASCII text file, *resp.vef*. This file is written by the CFD response analyses block and read by the optimizer. The transfer of design variables to the CFD response analyses block is achieved using the default design variables ASCII text file, *dvar.vef*. Each response value or design variable is written to a separate line in the corresponding “.vef” file.

The interaction between the various components in Figure 45 can be seen as a basic layout of the constituent elements of the MDO design methodology for the benchmark case. As previously indicated, VisualDoc allows for the selection of different optimization search algorithms. The non-linear, non-gradient based algorithms available in VisualDoc include the Response Surface Optimization (RSO), Particle Swarm Optimization (PSO) and the Genetic Algorithm (GA).

The Genetic Algorithm requires that the optimization variables be discrete in nature i.e. they may be made of only integers or discrete sets of values. However, the design variables of the parameterization scheme are continuous in nature. Although it is possible to study the benchmark problem by creating discrete sets for each design variable, this task is considerably tedious given the number of design variables involved. Thus the Genetic Algorithm is excluded as a feasible in the benchmark application of the MDO methodology. Of the PSO and RSO algorithms, the response surface optimization (RSO) selected for the benchmark study due to computational considerations.

3.3.1 Static Constraints for Design Variables and Responses

Apart from the dynamics constraints earlier imposed on the design variables, limits on the ranges of the variables are required to further define the feasible region of exploration in the MDO study. These ranges are static in nature and are implemented using lower and upper bounds for each design variable and response value. The upper and lower bounds of the design variables were determined as shown in Table 11:

Table 11: Upper/Lower Bounds on MDO Design Variables

DESIGN VARIABLE	LOWER BOUND	UPPER BOUNDS
# of Blades	2	25
RPM	-1500	-500
δ_r ($r = 1..5$)	0	1
$\underline{M}_{r 2}$	10	80
β_{rs} ($r=1..5$) ($s = 2..4$)	SP* - 40 (10) ⁺	SP* + 40 (80) ⁺
θ_r ($r = 1..5$)	-30	30
SL[xyz]	0	40
RL[ijk]	0	30
Rtheta[uv]	0	30

* SP – Starting point of bezier coordinate (from base design)

+ Min/Max value (Used if computed bound exceeds this value)

The limits are necessary in order to appropriately define the design space within which the optimization search algorithm operates. In specifying the limits on the design variables as shown in Table 11, the tangential coordinates (β_{rs}) of the Bezier CPs are allowed to vary $\pm 40^\circ$ from their initial values. However, the upper and lower bounds are truncated to 80° and 10° respectively. For example, each CP β coordinate, if the calculated upper limit based on its initial value (base design) exceeds 80° , the upper limits is truncated back to 80° . The lower limit truncation is set at 10° , with an exception made for initial values that are negative.

Although the limits on the design variables appropriately define the boundaries within which the optimization algorithms may operate, further constraints are necessary on the responses in order to limit the optimizer from considering physically infeasible designs in the optimization analysis for the benchmark problem. These limits on the responses serve as a further refinement of the feasible design space earlier defined by the upper/lower bounds on the design variables. The specified response constraints are shown in Figure 48 and are also listed in Table 12.

Table 12: Constraints on MDO Design Analyses Responses

RESPONSE	LOWER BOUND	UPPER BOUND
PFBladetest	0	None
PFGrid	0	None
PFResults	0	None
VolFlowRate (m ³ /s)	3	6
Total Efficiency	0.25	1.00
Static Efficiency	0.00	1.00

The response constraints complete the necessary bounds that completely define what regions of the global design space for the benchmark problem are valid for exploration by the RSO algorithm. The lower bound of the static efficiency is set to ensure positive efficiency values from the CFD analysis of the fan designs. As will be seen in the discussion of the results, a better approach for a more accurate problem definition may have been to use the static efficiency of the base design as the lower bound, in order to further restrict the design space.

The last step is to identify the target function (objective) of the MDO analysis in VisualDoc. Typically the target function is a response or a function of several responses from the analyses program. In the benchmark case the total efficiency is calculated as a response from the CFD analyses using the mass averaged absolute and relative pressure values (P_{Tabs} and P_{Trel}) at the leading and trailing edges.

The total efficiency is the target function for the benchmark fan design MDO study. It is determined in the CFX-Bladegen(Plus) CFD analyses as a function of the mass averaged relative total pressure drop from the leading edge (LE) to the trailing edge (TE) of the blade (P_{Trel}), relative to the increase in the total pressure in the absolute frame (P_{Tabs}).

$$\varepsilon_T|_{LE-TE} = \left[\frac{\partial P_{T_{abs}}}{\partial P_{T_{abs}} + \partial P_{T_{rel}}} \right]_{mass\ avg} \quad (3.8)$$

The static efficiency is computed in the CFX-Bladegen(Plus) flow field analyses as a function of the mass averaged static pressure rise across the blade relative to the change in the total pressure in the absolute frame.

$$\varepsilon_S|_{LE-TE} = \left[\frac{\partial P_S}{\partial P_{T_{abs}} + \partial P_{T_{rel}}} \right]_{mass\ avg} \quad (3.9)$$

The various mass-averaging relationships as well as the formulas for computing the other responses such as the headrise and blade torque are included in [21]. In additions to the constraints on the total and static efficiency, the design constraint on the required volume flow rate is imposed as shown in Figure 48. The other constraints are the *Pass/Fail* responses which come from the response analyses program. The lower bound of zero is the only necessary parameter for the *Pass/Fail* variables. Fan designs with *Pass/Fail* response values above the lower bound are considered valid designs. A value of 1 for all three *Pass/Fail* parameters indicates to the optimizer that a valid point in the design space has been successfully analyzed.

As discussed in Table 10, the different *Pass/Fail* parameters are used to detect the reason for a failed CFD analyses. Typical reasons as will be seen when the MDO results are discussed, include a failure to generate a valid mesh for the blade design and erroneous values for the target function calculated by the CFD solver.

All the elements necessary to properly define the optimizer block of the MDO design layout of Figure 45 have been developed. A blade parameterization scheme has been formulated, yielding design variables that adequately describe the 3D blade shape while simultaneously limiting the number of parameters needed.

The target function (total efficiency) has been specified in the VisualDoc module and appropriate bounds and limits imposed on the design variables. Constraints have also been imposed on the responses from the CFD flow field analyses, ensuring that the target function is only investigated within the design space defined by the combination of the specified upper/lower bounds and dynamic variable constraints.

At this stage, it is necessary to further decompose the CFD analyses block in Figure 45. The primary subcomponents are the module that generates the blade model from design variables based on the formulated blade geometry parameterization scheme. A secondary module is then needed to perform the actual CFD analyses on the blade model. The interaction between the decomposed CFD analyses block and the other components of the MDO environment is shown in Figure 49.

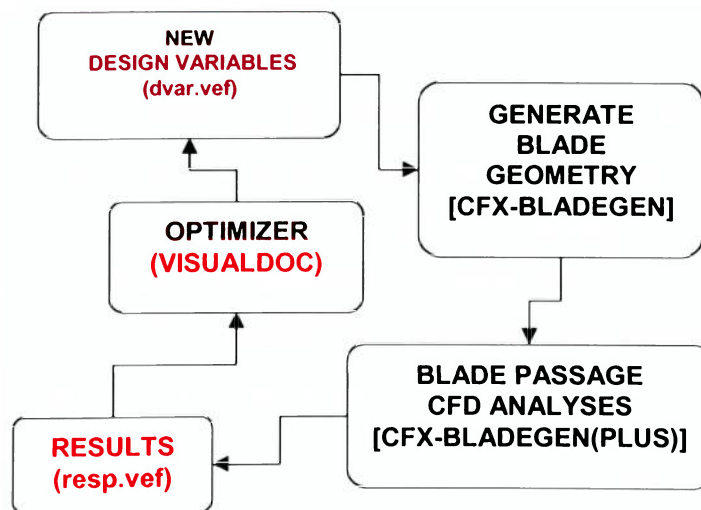


Figure 49: Decomposed MDO Component Interaction Flowchart

A critical issue still remains and it is the exact implementation of the MDO blocks (components) that comprise the response analyses starting from the blade model generation to the CFD analyses. In section 2, various CFX-Bladegen(Plus) scripts that perform a multitude of CFD and turbomachinery modeling related tasks are identified. The resulting issue is how to appropriately call all of these scripts in the correct order. It is also necessary to ensure that the information from the optimizer is properly transferred to the respective MDO components, such that a complete and accurate CFD analysis of every fan design can be carried out.

The development of an integrated blade modeling and CFD response analyses is accomplished using the native VisualDoc scripting tool called VisualScript. This utility is essentially a graphical programming interface that allows several third-party analyses tools (CFD related programs for the benchmark case), to be integrated into a single “program.” This *program* then serves as a response analyses tool which is called by the optimizer for analysis of a design being investigated. The VisualScript *program* is actually composed of several calls to third-party tools/utilities. In the benchmark case, these utilities are the various scripts supplied as part of the CFX-Bladegen(Plus) Turbomachinery design/analyses package, as well several in-house scripts developed using MATLAB.

The following section details the implementation of the response analyses program in VisualScript to include the various tools employed to accomplish each step of the blade model generation and CFD analyses. The details of a complete MDO design iteration is discussed beginning from the generation of design variables by the optimizer. The subsequent blade model generation, CFD analyses and transfer of the responses (including the computed target function) back to the optimizer are also presented.

3.4 MDO Response Analysis Implementation (VisualScript)

The response analyses implementation involves using the VisualScript graphical programming interface provided as part of the VisualDoc MDO package. VisualScript uses an “element” to represent each component of the response analyses program. For the benchmark case, the response analyses flowchart showing the interaction between element components is shown below:

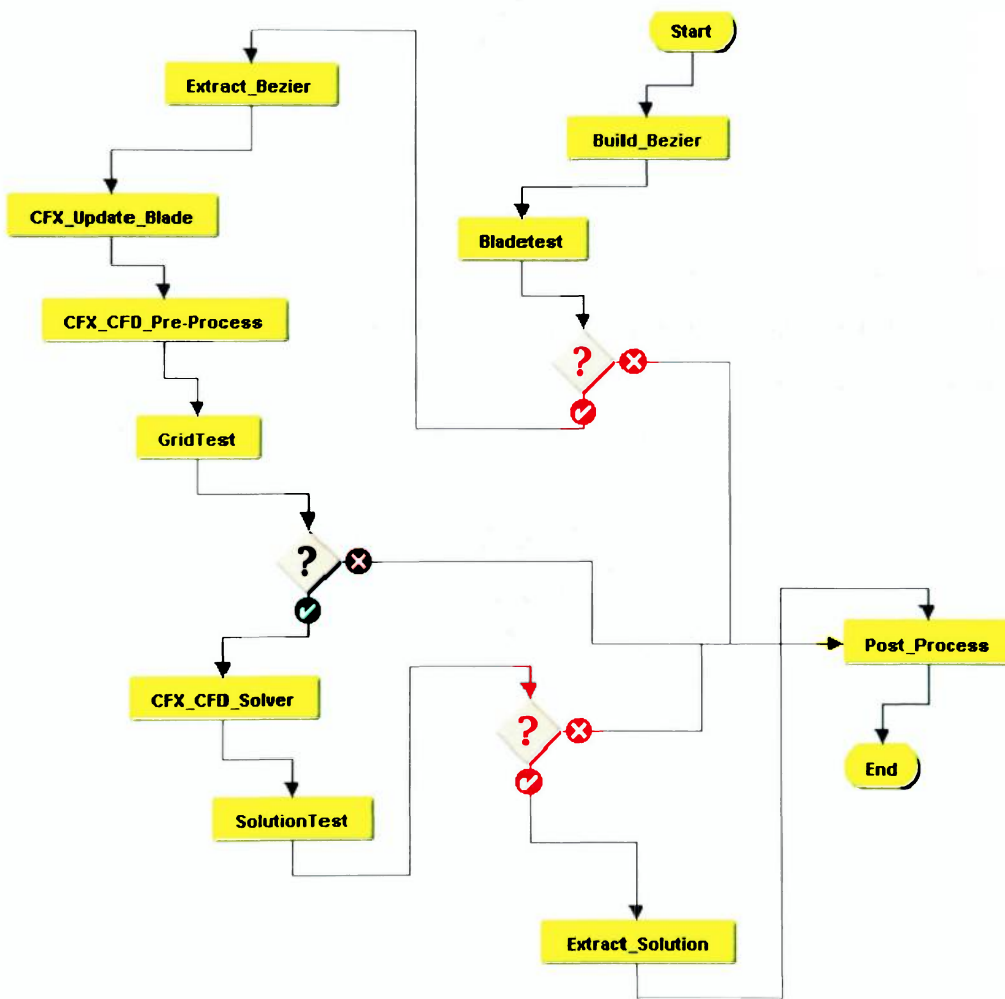


Figure 50: VisualScript MDO Response Analysis (CFD) Implementation

The elements of the visualscript are the yellow “blocks” that are interconnected in Figure 50 to make up the complete response analysis. There are also special elements called *if* elements that allow for the conditional execution of certain blocks depending on the values of a variable. The specifics of an *if* block are shown in Figure 51. Other elements exist in VisualDoc which allows for the direct coupling of VisualScript to MATLAB scripts as well MS-Excel files, however they are not directly utilized in MDO response analyses script of Figure 50.

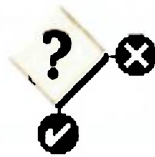


Figure 51: Conditional (*if*) VisualScript Element

All elements in the visualscript require an element definition. It is in the element definition that the exact 3rd-party programs to be called by the respective elements are defined. The conditional (*if*) element of Figure 51 executes the block attached to the green circle if the test specified in the element definition is passed and it executes the block attached to the red circle if the test specified in the element definition is failed.

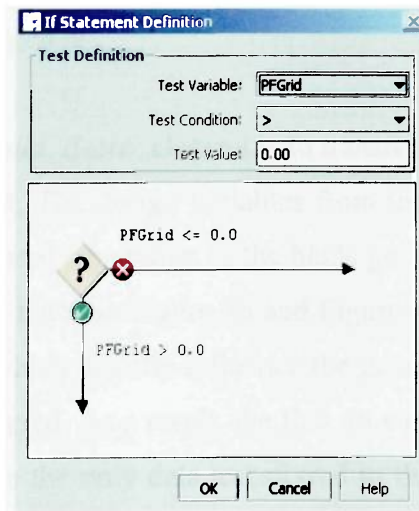


Figure 52: VisualScript *if* Element Definition

A sample *if* element definition window is shown Figure 52 that a *test value* of zero is used to identify the proper value for a Pass/Fail result. This is the reason why the lower bound of the *PFGrid* response in Figure 48 and Table 12 is set to zero. VisualScript looks at the value of the *PFGrid* response as the analyses executes and chooses to execute the true or false block. The decision to execute either the true or false block depends on whether the value of the *PFGrid* variable is less than or greater than the *test value*. The *test value* must correspond to the lower bound (zero) set for that variable in the VisualDoc response specification (as done in Figure 48).

The VisualScript element definitions are created by double-clicking on the appropriate element in the visualscript. Each yellow block represents a combination of tasks which need to be accomplished. In the following sections, the tasks that are accomplished in each of the blocks in Figure 50 are discussed. These include appropriate importing, exporting and transfer of design variables to and from the optimizer using ASCII based text files.

3.4.1 VisualScript Element Definition – Build_Bezier

Data Files: *input.txt, coordinates.txt*

Scripts/Utilities: *curvegen.m*

The main function of the *Build_Bezier* element is to transfer the design variables from the optimizer to the visualscript. The design variables from the optimizer are written to the text file *input.txt*, which is used generation of the blade geometry in subsequent modules. Not all the design variables listed in Figure 46 and Figure 47 are transferred to *input.txt*. Only the design variables which directly influence the generation of the spanwise Bezier control polygons are transferred. As a result, the first 46 variables in the specifications of Figure 46 and Figure 47 are the only data transferred to the *input.txt* file. The synthetic variables (47 – 81) that implement various bounds on the design variables are not transferred. The order in which the variables are listed in *input.txt* corresponds to the order in which they are listed in the variable specifications. Appendix A contains a sample *input.txt* file with a legend detailing the order in design variables.

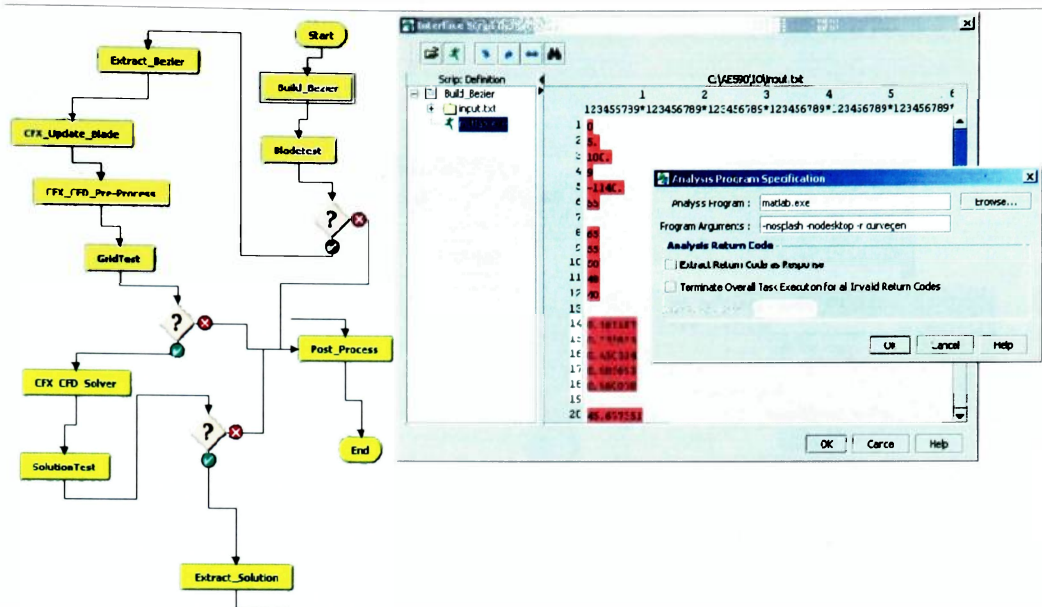


Figure 53: VisualScript Element Definition – Build_Bezier

The next step is to use the design variables to define the bezier control polygons that will generate the airfoil camberlines at each span layer. This is accomplished using a MATLAB script called *curvegen.m*. This script simply reads the design variables from *input.txt* and uses the data to generate the coordinate pairs for all the bezier CPs necessary to define the blade profiles at all 5 span layers. The command line arguments (*-nosplash*, *-nodesktop*, *-r curvegen*) specified in the *Analyses Program Specification* window of Figure 53 simply instruct MATLAB to run the *curvegen.m* script (also known as an m-file) in batch mode.

The coordinates of the Bezier CPs are generated in *curvegen.m* using the equations detailed in the blade parameterization scheme of section 3.2 and written to a file called *coordinates.txt*. This file serves as the input to the next block which attempts to test the bezier control polygons to ensure the specified constraint criteria are satisfied by the blade geometry.

3.4.2 VisualScript Element Definition – Bladetest

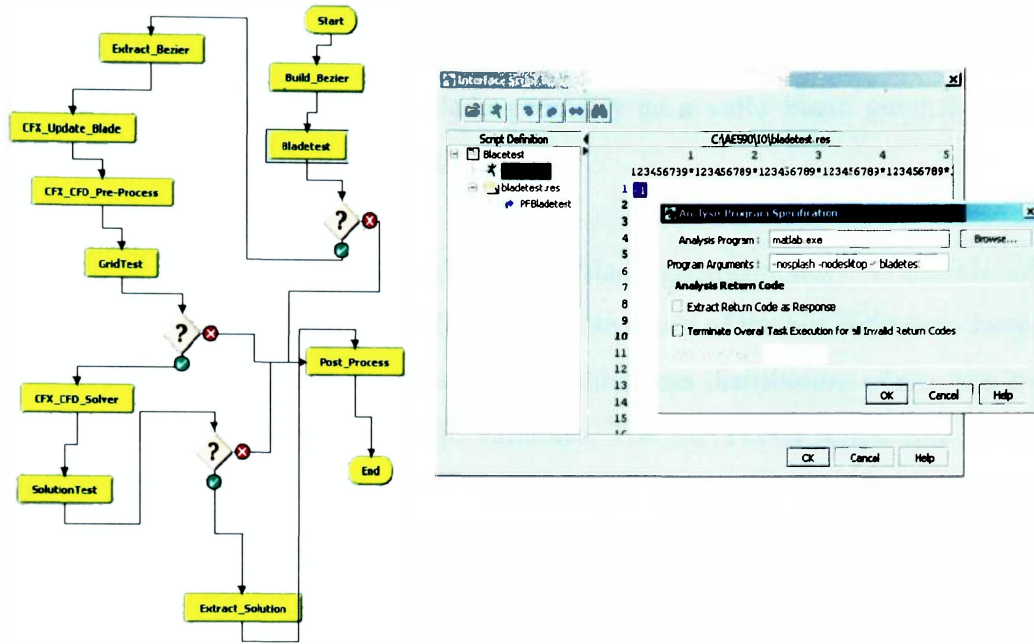


Figure 54: VisualScript Element Definition – Bladetest

Data Files: *input.txt, coordinates.txt, bladetest.res*

Scripts/Utilities: *bladetest.m*

The *Bladetest* element is developed to enable tests on the bezier control polygons to be carried out to ensure that the blade designs generated are valid. The blade geometry validation tests are usually based on constraints such as those developed in equations 3.5 through 3.7. However, the availability of the synthetic variables as part of the VisualDoc package eliminates the need for this feature in the VisualScript. The file *bladetest.res* is written by the MATLAB script *bladetest.m* and contains the result of the test on the bezier control polygons. The result is specified as a single non-zero integer with a value of 1 corresponding to a valid geometry (pass) and -1 for an invalid geometry (fail).

Bladetest reads the Bezier polygon CP coordinates from *coordinates.txt* and also reads other geometric and performance conditions from *input.txt*. As previously mentioned, synthetic variables are used in VisualDoc to integrate the constraint verification of the bezier CPs into the optimization loop. As a result, the checks in the MATLAB script are modified to always report the blade geometry as a valid blade geometry (i.e. the file *bladetest.res* always contains a value of 1).

Advanced testing schemes to validate the blade geometry may be introduced into the *bladetest.m* script if desired and the appropriate result of the test written to the output file. For example, parameters that describe the thickness distribution of the blade geometry can be introduced as future design variables. The MATLAB script may be modified to ensure these variables are within certain ranges and report the blade design as invalid depending on the results of the test. As a result, this element is retained in the event complicated geometry checking algorithms beyond the capabilities of the VisualDoc synthetic variables need to be integrated into the MDO cycle. Although the *Bladetest* element has been included as a part of the MDO environment, it may be considered inactive as it always reports that the blade geometry is valid by writing a value of 1 (pass) to the *bladetest.res* file.

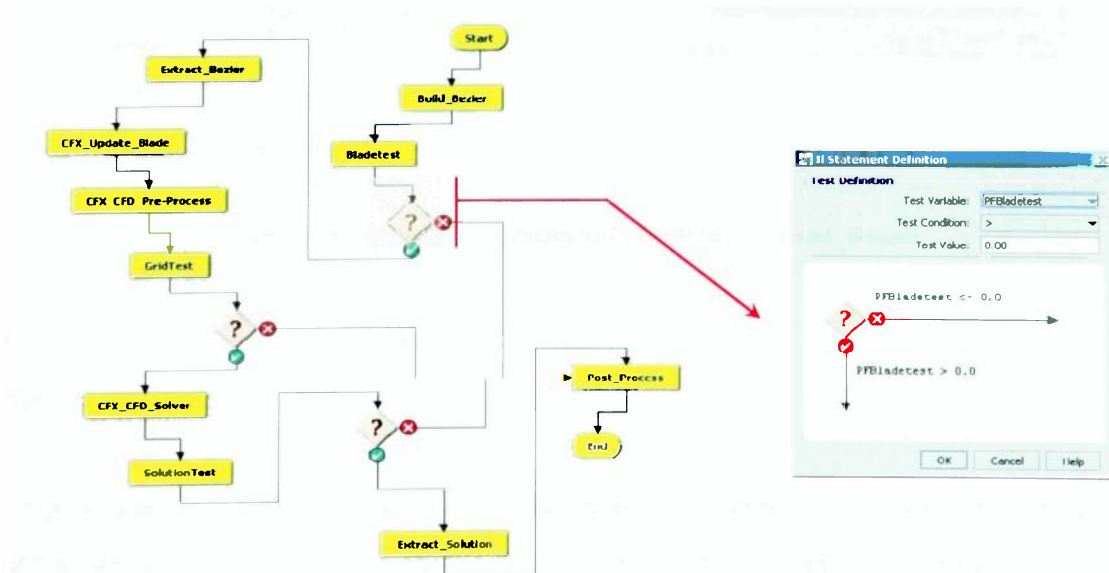


Figure 55: Conditional (IF) Element Definition – Bladetest

The next element (Figure 55) is the first conditional element of the visualscript. This element is used to select the blocks to be executed depending on the result of the *bladetest.m* script, contained in the *bladetest.res* file. The element examines the result after extracting it from the *bladetest.res* file. If the blade geometry is invalid, the post-processing block (*Post_Process*) is executed skipping the CFD analyses of the blade geometry. If the blade geometry is valid (*bladetest.res* containing a value of 1), the execution proceeds to the transfer the validated bezier control point coordinates to the blade geometry definition file.

3.4.3 VisualScript Element Definition – Extract_Bezier

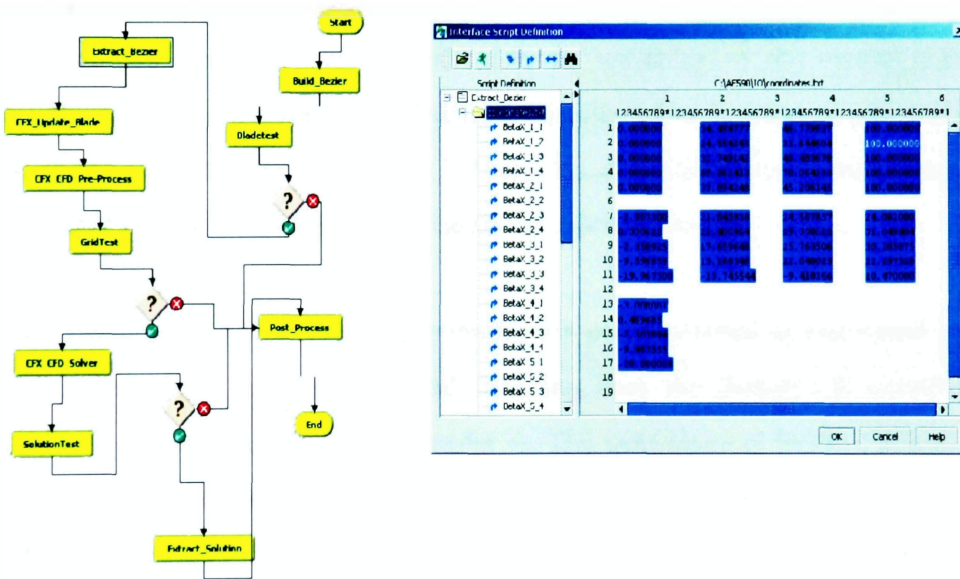


Figure 56: VisualScript Element Definition – Extract_Bezier

Data Files: *coordinates.txt*

Scripts/Utilities: None

At this stage of the MDO analyses, the coordinates that locate the bezier control points in 3D space are defined. These bezier control points are subsequently used to generate airfoil shapes at each of the 5 span layers along the blade from hub to shroud. It should be noted that through the use of the synthetic variables, the bezier control points have been

automatically validated as satisfying the dynamics constraints of equations 3.5 through 3.7. The constraint verification automatically done even before the design variables are exported by VisualDoc.

The next stage is to transfer the Bezier CPs into the blade definition batch (.bgi) file and the first step in this process is the extraction of the bezier CP coordinates from the *coordinates.txt* output file. It should be noted that the *coordinates.txt* file also contains the LE circumferential angular (θ_r) coordinates of the 5 span layers of the blade. The extraction of the Bezier CP coordinates is accomplished in the *Extract_Bezier* visualscript element.

Figure 56 illustrates the extracting the bezier CPs from the *coordinates.txt* file. The extraction process stores the values internally as variables in the visualscript. These internal values can be subsequently modified, transferred or written to other files in the script. It is the internally stored bezier CP and LE circumferential coordinates that are used to modify the CFX-Bladegen blade geometry definition file.

The exact mechanics of how the data extraction is accomplished is discussed in [22]. A sample *coordinates.txt* file and a legend detailing how the Bezier CP coordinates are specified in the file are included in Appendix A. The next element in the visualscript uses the internally stored Bezier coordinates and LE sweep values to modify the CFX-Bladegen blade geometry definition file. This element does not run any executables and hence no third party programs are specified in the *Analyses Program Specification* window as is done in the visualscript element definitions discussed so far.

3.4.4 VisualScript Element Definition – CFX_Update_Blade

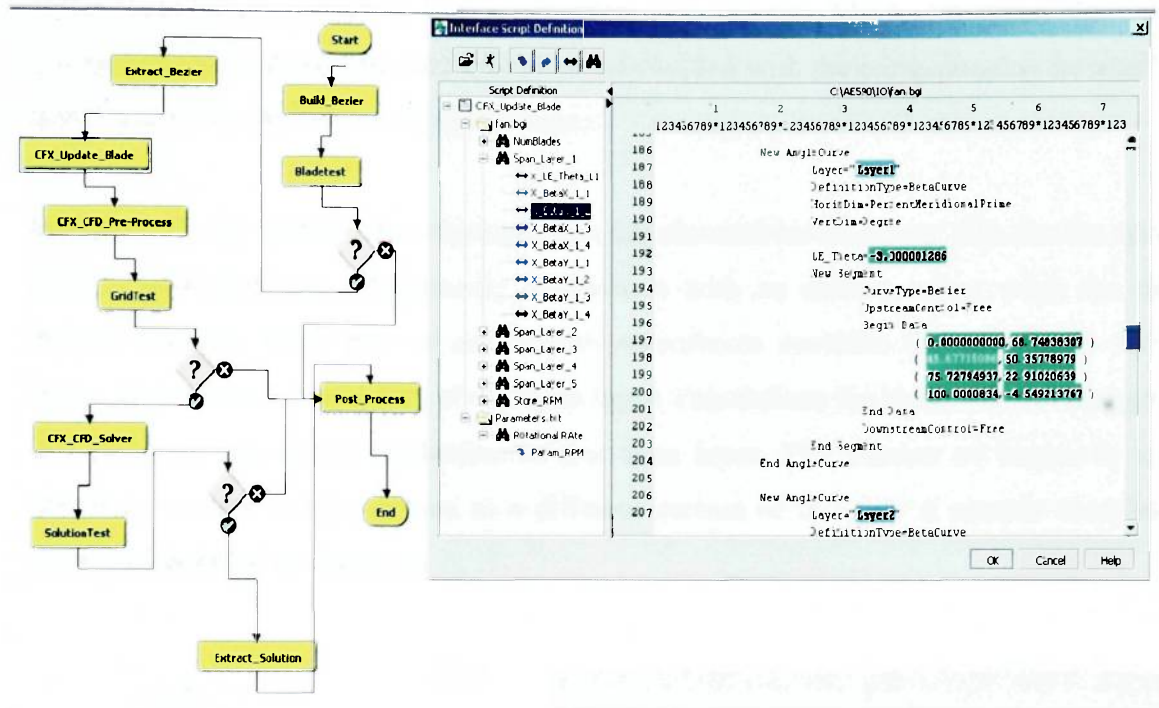


Figure 57: VisualScript Element Definition – CFX_Update_Blade (3D Blade File)

Data Files: *fan.bgi, Parameters.txt*

Scripts/Utilities: None

The primary function of this element is to transfer the bezier CP coordinates internally stored in the visualscript (from the execution of the previous block), to the CFX-Bladegen input file that describes the turbomachinery model. As discussed in section 2.1.2, the batch file (.bgi) format is selected as the file format to be used. This format is essentially an ASCII text-based description of the 3D geometrical make-up of the blade model. It included the number of span layers, rotational rate (RPM), the number of fan blades, bezier CP and LE circumferential coordinate at each span layer along the blade.

The 3D geometry of the blade is defined in the *fan.bgi* file. It is to this file that the internally stored CP and LE sweep coordinates are written. Figure 57 illustrates a sample

section of a blade geometry definition file where the layer is identified and the various design variables are inserted into the file. The text in blue highlight identifies the section of the blade (span layer #1 or hub) to which the data belongs. The text in green highlight is a specification of the data that should be overwritten with the corresponding internally stored values for the current design iteration.

In Figure 57, the number that represents the LE circumferential sweep (θ_r), for the span layer is identified and subsequently overwritten with its corresponding value for the current iteration. Each pair of numbers in parentheses specifies the coordinate pairs ($M'_{r|s}, \beta_{r|s}$) for the 4 bezier CPs at that span layer. This defines the bezier control polygon that produces the airfoil camberline at that span layer. The number of blades in the complete cascade is also written to a different section of this file. A sample complete *fan.bgi* is included in Appendix A.

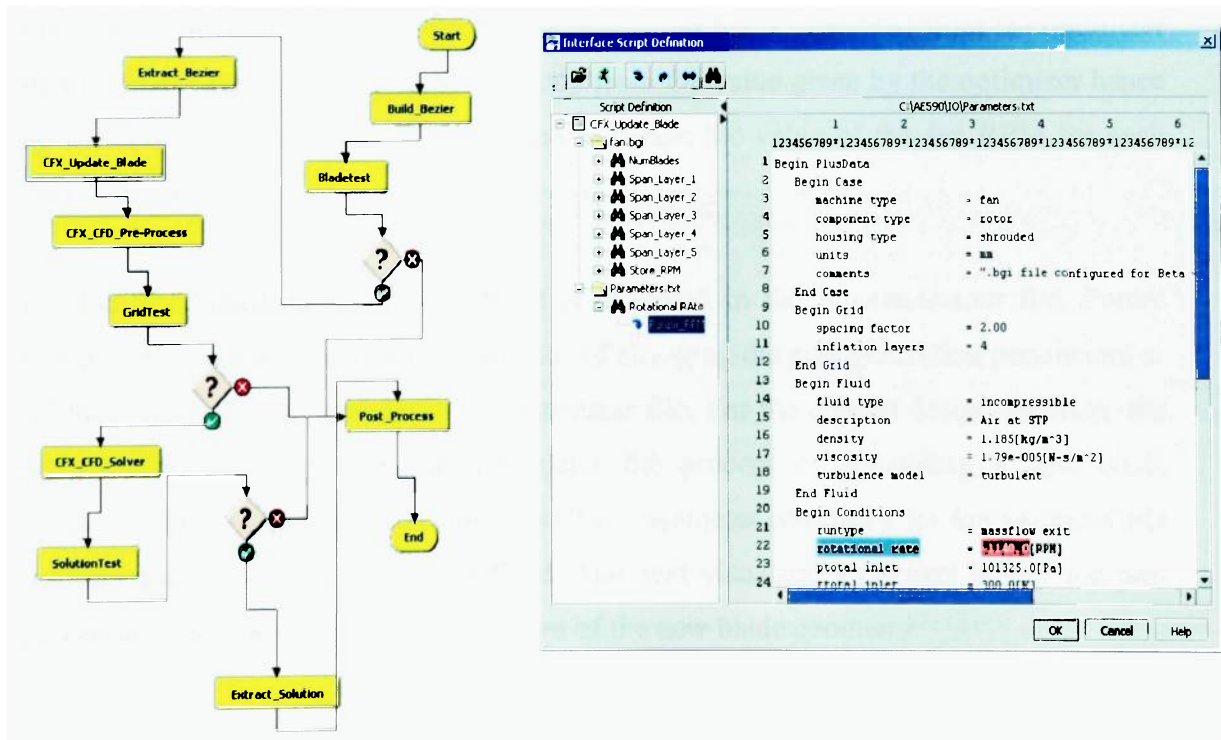


Figure 58: VisualScript Element Definition – CFX_Update_Blade (Parameter File)

Apart from the specification of the 3D geometry of the blade, the operating conditions such as fluid properties, freestream temperature, and the rotational rate (RPM) of the fan need to be specified and this is accomplished in a separate file known as the *parameter file*. A sample *parameter file* is provided in Appendix A. Appropriately named *Parameters.txt*; this file contains a listing of data necessary to fully define a fluid flow problem that can be analyzed using CFD. A section of sample parameter file is illustrated in the *Interface Script Definition* window shown in Figure 58. The *parameter file* includes the spacing factor and number of inflation layer values for the CFX-Bladegen(Plus) unstructured mesh generator. It also contains CFD solution control parameters including freestream conditions, turbulence and discretization scheme (advection blend) settings necessary for the CFX-Bladegen(Plus) N-S flow solver.

The rotational rate (RPM) of the fan is also specified in *Parameters.txt* and is indicated by the data in red highlight in Figure 58. Unlike the Bezier CPs which have to pass through the three previous visualscript elements after being generated from the optimizer in VisualDoc, the RPM is modified directly from the value given by the optimizer hence no extraction from any data files is done to obtain the value of the fan RPM for each design iteration.

For the benchmark case only the RPM is modified in the *Parameters.txt* file. Future research efforts may consider the influence of changing the grid generation parameters or solution discretization settings in the parameter file. For the current design iteration, the modification of the *fan.bgi* file completes the process of generating a new blade geometry. The operating conditions as well as parameter selections for the various CFD solution subcomponents are also defined. The next visualscript element begins the pre-processing necessary for the CFD analyses of the new blade geometry.

3.4.5 VisualScript Element Definition – CFX_CFD_Pre-Process

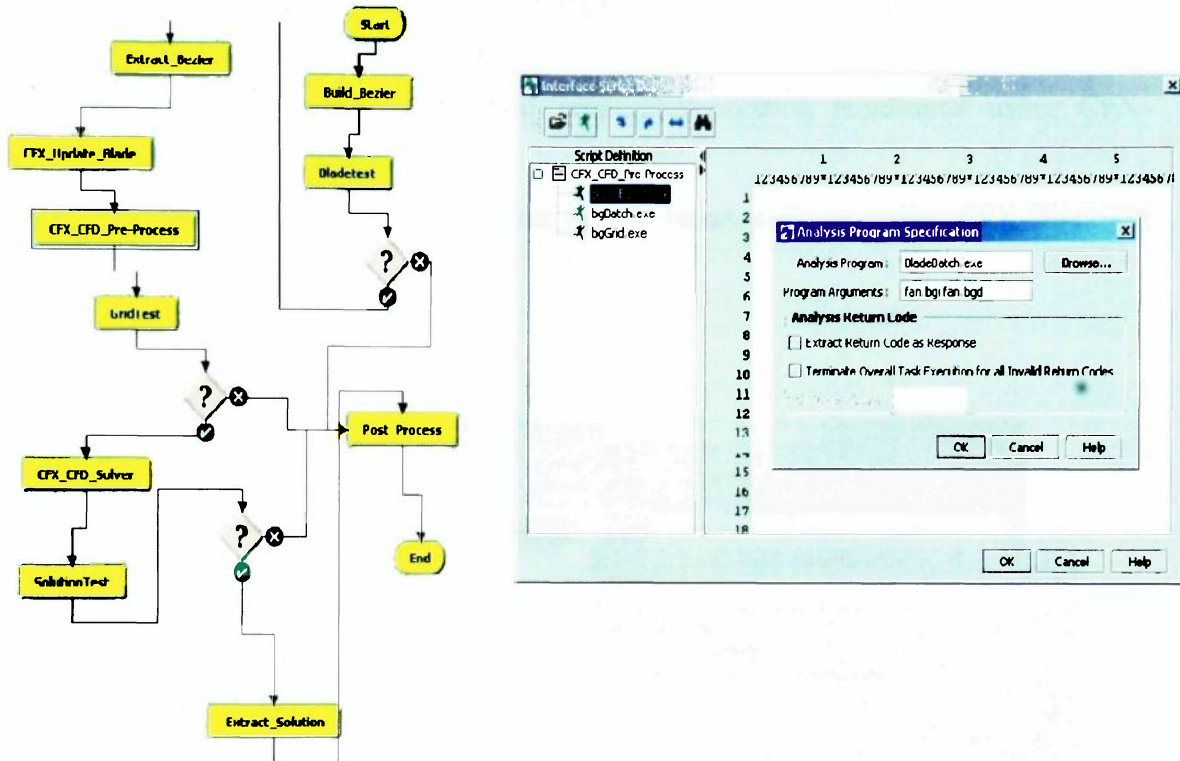


Figure 59: VisualScript Element Definition – CFX_CFD_Pre-Process (BladeBatch)

Data Files: *fan.bgi, fan.bgd, batch.bg+, grid.bg+,
parameters.txt, template.bg+*

Scripts/Utilities: *BladeBatch, BgBatch, BgGrid*

The function *CFX_CFD_Preprocess* block is to combine the 3D fan model with the operating conditions specified in the *parameter file* and create a CFX-Bladegen(Plus) CFD solution file (.bg+). The flow field solution file is essentially a synthesis of the *fan.bgi* and the *Parameters.txt* files modified in previous visualscript elements. The *CFX_CFD_Preprocess* block essentially carries out all the pre-processing necessary to properly define a CFD problem that is solved using the CFX-Bladegen(Plus) solver.

The pre-processing for the flow solver is composed of three steps:

1. Convert blade model from batch format (.bgi) to native CFX format (.bgd) using CFX-Bladegen *Bladebatch* Utility.
2. Combine fan model with fluid properties/operating conditions using CFX-Bladegen(Plus) *BgBatch* Utility.
3. Generate unstructured mesh over single blade passage using CFX-Bladegen *BgGrid* Utility.

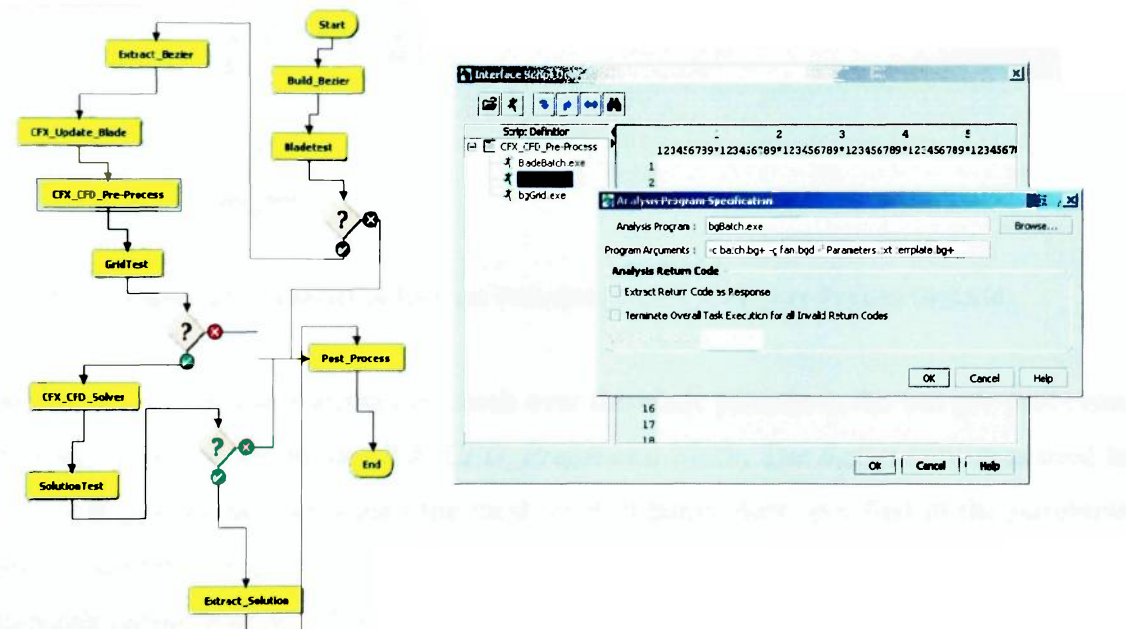


Figure 60: VisualScript Element Definition – CFX_CFD_Pre-Process (BgBatch)

The conversion of the blade model from the batch format to the CFX native format is necessary because the CFX flow solver requires the blade geometry to be provided in the CFX native (.bgd) format. This is accomplished using the CFX-Bladegen *Bladebatch* utility as illustrated in Figure 59. The next step is the integration of the blade model with the operating conditions and freestream fluid properties in order to define the CFD problem to be solved. This second pre-processing stage is accomplished using the *BgBatch* utility as illustrated in Figure 60. Further details on the functions of programs arguments specified in Figure 60 can be found in [21].

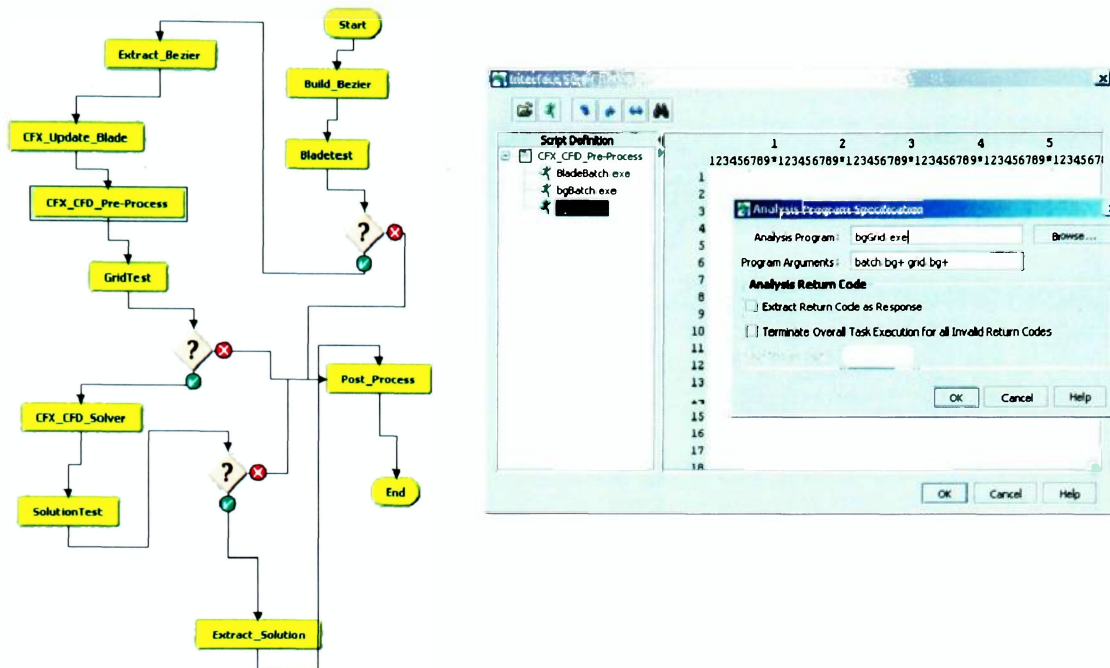


Figure 61: VisualScript Element Definition – CFX_CFD_Pre-Process (BgGrid)

The generation of the unstructured mesh over the blade passage is the last pre-processing step that is performed in the *CFX_CFD_Preprocess* block. The *BgGrid* utility is used for the mesh generation and it uses the mesh control parameters specified in the *parameter file* to determine the quality of the grid to be generated. For the benchmark study, the **spacing factor** is set to 2 (fine mesh) and the number of **inflation layers** selected to be 4. These setting generally yield mesh sizes in the order of 190,000 cells and 48,000 nodes for the fan designs being studied in the MDO benchmark case.

3.4.6 VisualScript Element Definition – GridTest

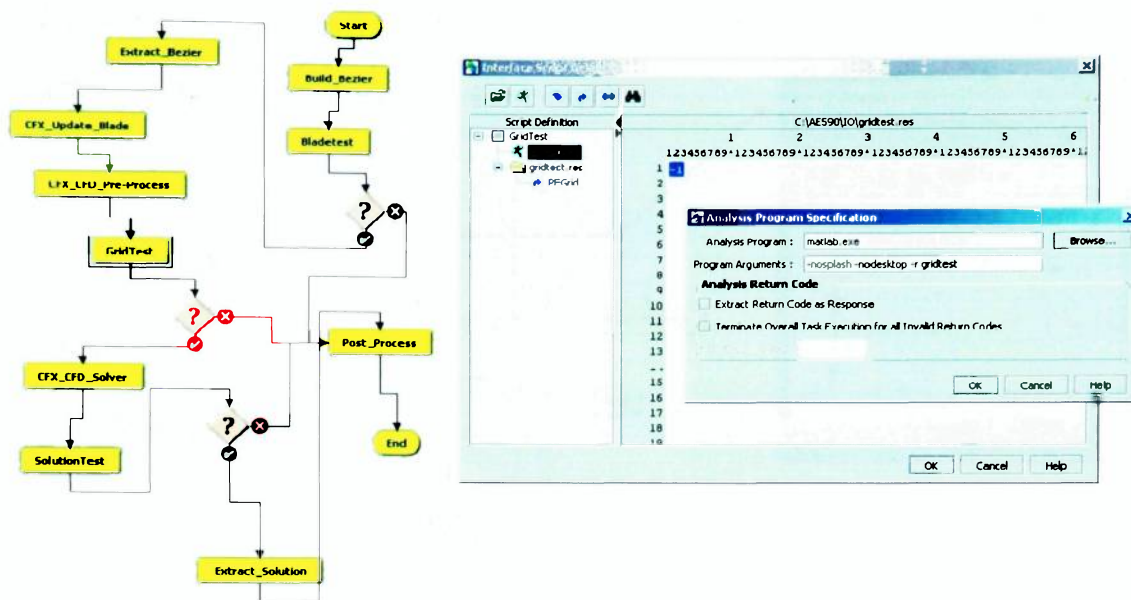


Figure 62: VisualScript Element Definition – GridTest (BgGrid)

Data Files: None

Scripts/Utilities: *gridtest.m*

The capability for robustness in the MDO environment is critical in designing the visualscript for the benchmark case. It is probable that despite all specified bounds and constraints, the optimizer will produce blade designs for which valid meshes cannot be generated. As such, an element must be included to determine if a valid mesh is generated for the blade design and communicate the result back to the optimizer.

The *GridTest* block is executed after the CFD pre-processing block and it runs a MATLAB script (*gridtest.m*), which checks to see if a valid mesh is generated for the new blade geometry. This is accomplished by searching the working directory for a *grid.bg+* file that is written by the preprocessing block **only** if a mesh was successfully generated for the new blade geometry.

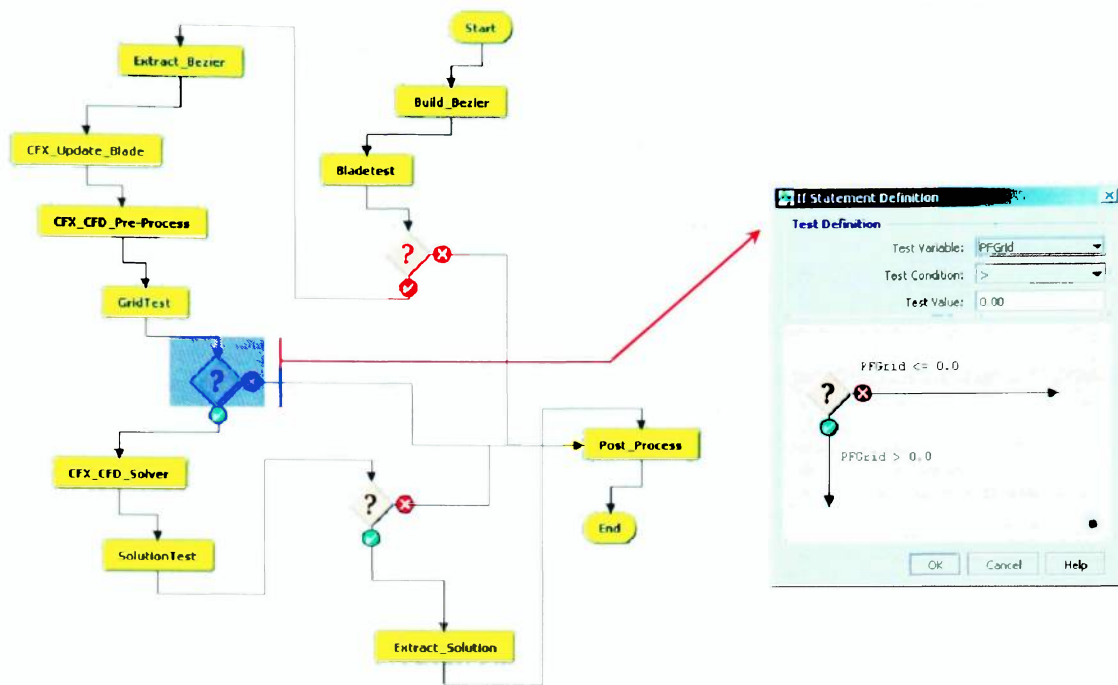


Figure 63: Conditional (IF) Element Definition – GridTest

If a valid mesh file is found in the working directory, then a pass value of 1 is written to the *GridTest* output file, *gridtest.res*. A failed mesh generation in the preprocessing block prevents a *grid.bg+* file from being written to the working directory. The *GridTest* block will be unable to locate the *grid.bg+* file and will a fail value of -1 to the output file. The result from the output file is then extracted by VisualScript, stored internally as a visualscript variable called *PFGrid*. This internal VisualScript variable is returned to the VisualDoc optimizer as a *Pass/Fail* parameter.

The extracted *GridTest* result is used by the succeeding conditional element as illustrated in Figure 63. The conditional element proceeds immediately to the post-processing block for a failed mesh generation or to the CFD solution block in the event a valid mesh has been generated for the new blade geometry (*gridtest.res* contains a value of 1).

3.4.7 VisualScript Element Definition – CFX_CFD_Solver

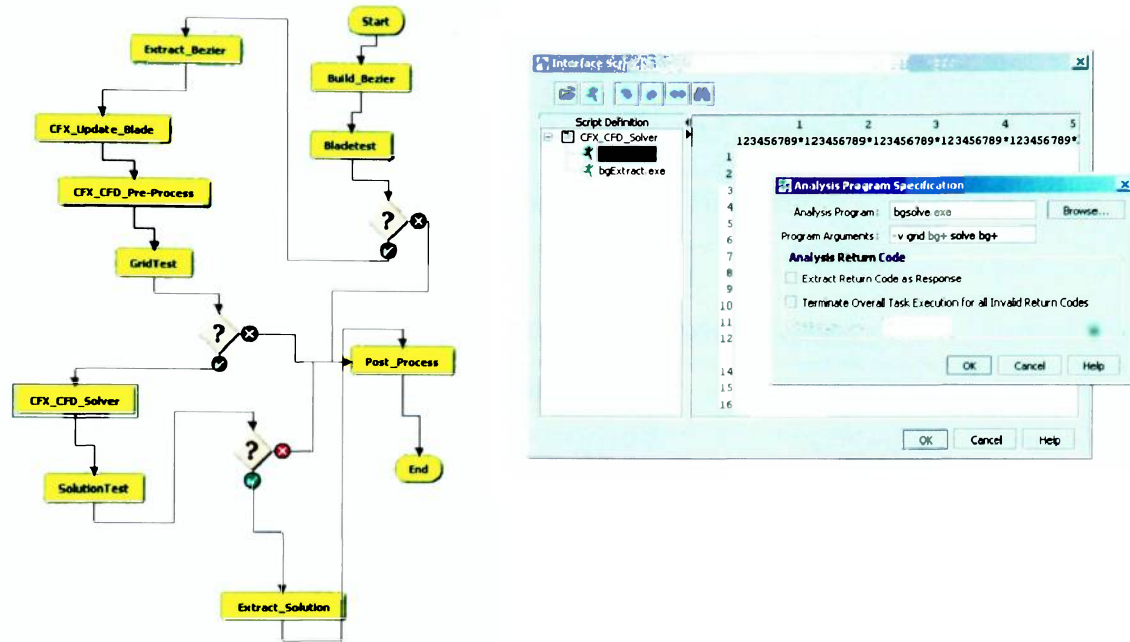


Figure 64: VisualScript Element Definition – CFX_CFD_Solver (BgSolve)

Data Files: *grid.bg+*, *solve.bg+*, *Results.txt*

Scripts/Utilities: *BgSolve*, *BgExtract*

The *CFX_CFD_Solver* block computes the CFD solutions for the fan blade geometry using the freestream and operating conditions specified in the *parameters file*. The solver takes as input the CFX-Bladegen(Plus) file *grid.bg+*, containing the mesh generated in the preprocessing block. The CFD solution is carried out using the *BgSolve* utility using settings from the *parameter file* to determine solutions control parameters such as the max number of iterations and discretization order for the numerical scheme. The following tables contain the specifications used in the benchmark case for the freestream and operating conditions as well as the solutions control parameters used by the CFX-Bladegen(Plus) CFD solver.

Table 13: MDO/CFD Study Fluid Properties

Fluid	Air @ STP
Fluid Type	Incompressible
Density (ρ)	1.185 [kg/m ³]
Dynamic Viscosity (μ)	1.79e ⁻⁵ [Ns/m ²]
Fluid Model	Turbulent

Table 14: MDO/CFD Study Operating Conditions

Inlet Boundary Conditions (BC)	P _{TOTAL}
Outlet Boundary Conditions (BC)	Mass Flow Rate (MFR)
Rotational Speed (RPM)	1140, 1720
Inlet Swirl Angle	0 [rad]
Inlet P _{TOTAL}	101325 [Pa]
Inlet T _{TOTAL}	300 [K]
Total MFR (Complete Machine)	4.286 [kg/s]
Wall Roughness Model	Smooth

In Table 14 multiple rotational rates (RPM) are specified in order to determine the effect of starting from different base design on the ability of the RSO algorithms to converge on a uniform solution. Further discussion on the multiple starting designs is presented in subsequent sections.

Table 15: CFX-Bladegen(Plus) Flow Solver – Solution Control Parameters

Advection Blend (Discretization Order)	0.88
Solution Time Step	Autocompute
Target Residual	1e-5
Max # of Iterations	150
Reynolds Number (Re)	2.5e ⁵

The problem specifications shown above are used by the CFX-Bladegen(Plus) solver for the CFD analyses of the fan design. The advection blend specification is used to control the order of the discretization scheme used for the solution to the N-S equations (0 – 1st Order, 1 – 2nd Order). The selection of an advection blend of 0.88 is to allow for the robustness typically associated with lower order discretization schemes to be combined with the accuracy that is characteristic of CFD solutions based on higher order discretization schemes. Although robustness is a critical factor in the MDO CFD analyses, excessive tradeoffs in the accuracy of the CFD solution must be avoided, hence a target residual value of 1e⁻⁵ is used to ensure the proper convergence of the CFD analyses.

The output from the CFD solver is a solution file (*solve.bg+*) that contains the flow field solution for the fan design as well as computed performance metrics including static efficiency, headrise and the target function (total efficiency). These computed performance metrics are subsequently extracted from the CFD solution file using the *BgExtract* utility as illustrated in Figure 65.

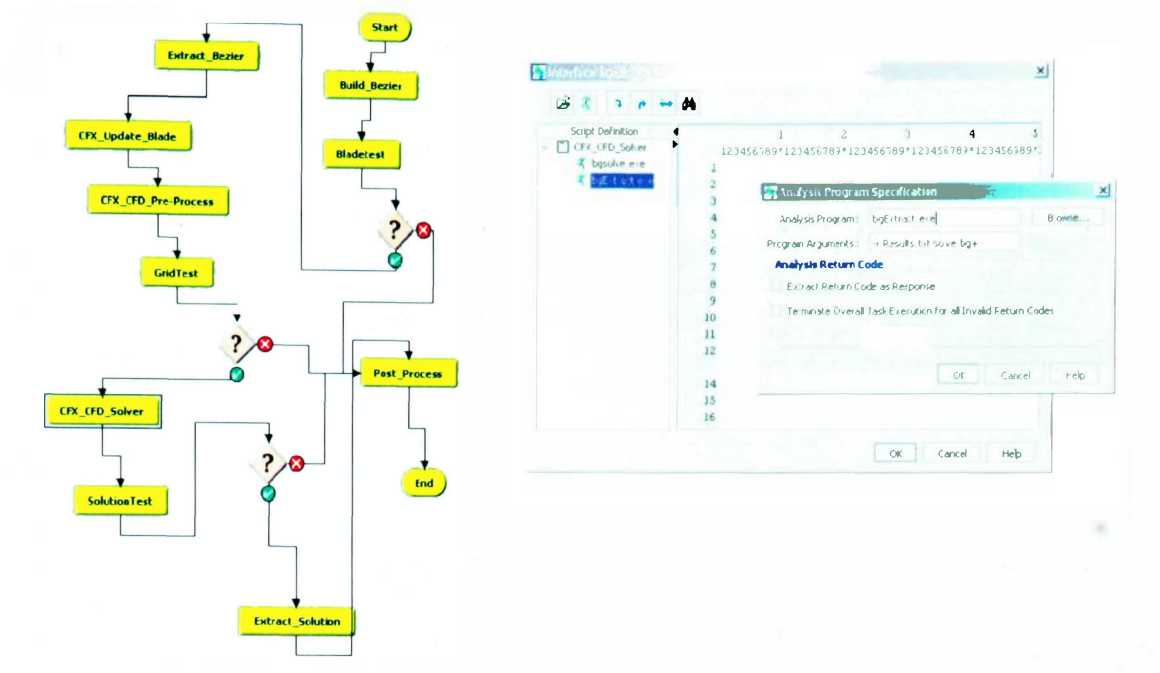


Figure 65: VisualScript Element Definition – CFX_CFD_Solver (BgExtract)

The output from the *BgExtract* utility is an ASCII text file (*Results.txt*), containing the results of computed performance metrics for the fan design. The calculated values contained in the results output file are detailed below. These values are used by the subsequent blocks to validate the CFD solution.

- Mass Flow Rate, MFR [kg/s]
- Volume Flow Rate, VFR [m³/s]
- Total Blade Torque [nm]
- Headrise [m]
- Inlet Flow Coefficient
- Exit Flow Coefficient
- Head Coefficient
- Total Efficiency, ϵ_T
- Static Efficiency, ϵ_S

3.4.7 VisualScript Element Definition – SolutionTest

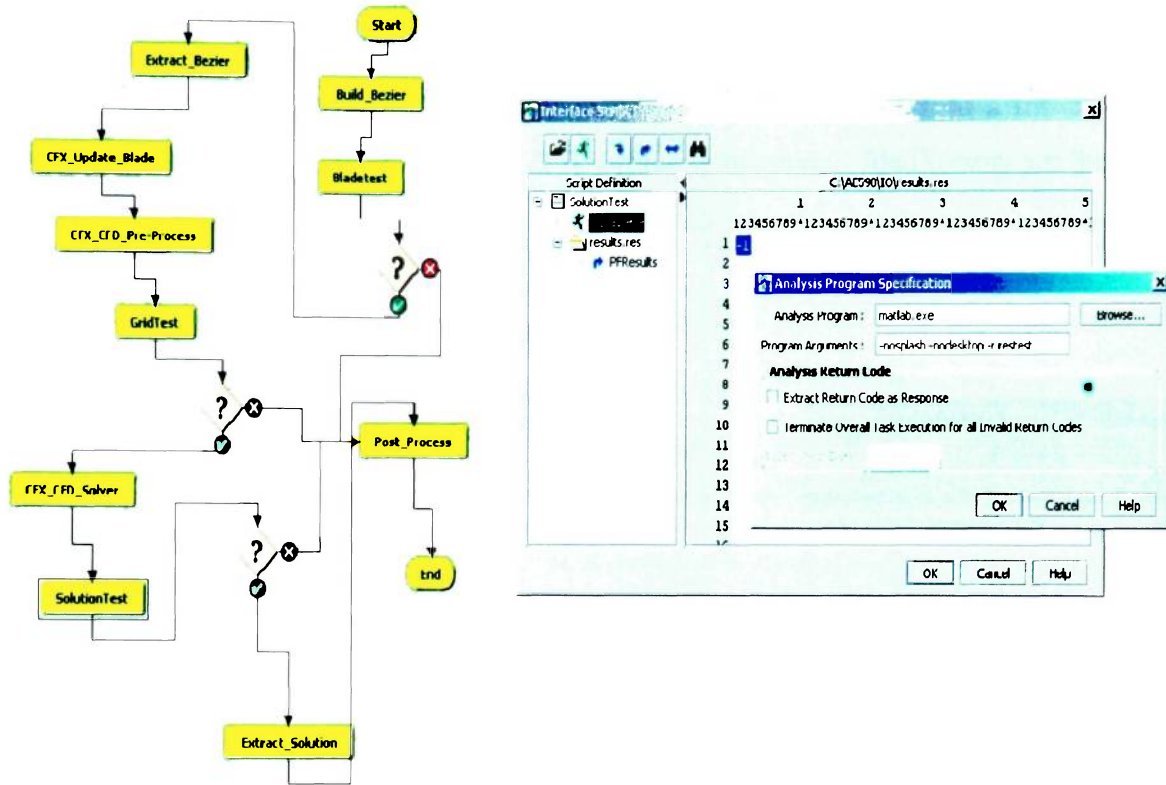


Figure 66: VisualScript Element Definition – SolutionTest

Data Files: *Results.txt, retest.res*

Scripts/Utilities: *retest.m*

The *SolutionTest* block is created in order to further enhance the robustness of the MDO visualscript by increasing its tolerance to failed/invalid analyses of the generated fan designs. It was determined during several CFD calculations that there are instances in which the CFD calculation apparently proceeds correctly, but yield invalid results in the *Results.txt* file created by the *CFX_CFD_Solver* block. This creates the need for a visualscript element that checks to ensure that the calculated performance metrics in the *Results.txt* file are valid.

The *SolutionTest* element executes a MATLAB script (*retest.m*), which scans the *Results.txt* for invalid data such as computed efficiency values less than zero or negative mass flow rate values. The results file is also scanned for invalid data including erroneous character values written in place of computed numeric values. If no errors are found in the CFD results file, the *retest.m* script then writes a pass value of 1 to a file called *retest.res*. A corresponding fail value of -1 is written to the output file if errors are found in the CFD results file.

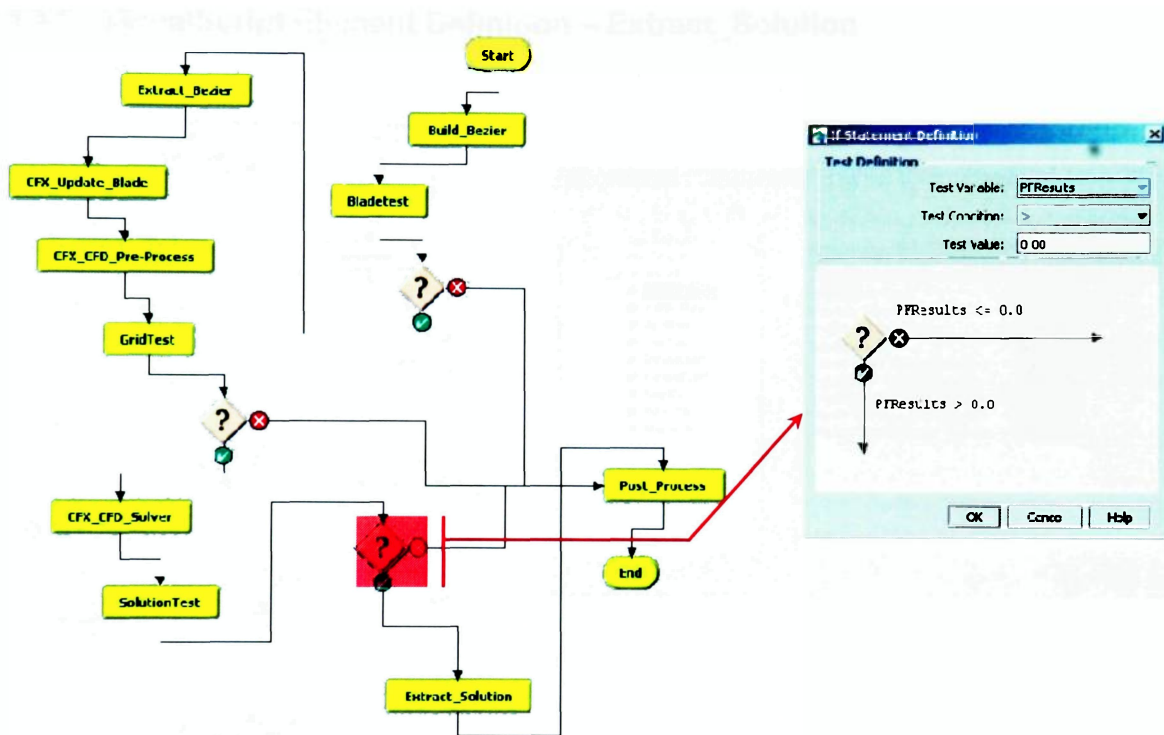


Figure 67: Conditional (IF) Statement Definition (*SolutionTest*)

Once the validity of the computed CFD results is verified, the *SolutionTest* block extracts the value in the *retest.res* file and stores it internally as the visualscript variable, *PFResults*. This variable is subsequently transferred to the VisualDoc optimizer as the third *Pass/Fail* variable in the MDO design methodology for the benchmark case. This value is used to identify the regions in the design space where designs that will generate bad CFD results are located.

The internal script variable extracted from the *SolutionTest* results file is used by the third conditional statement of the visualscript as shown in Figure 67. This value determines the next element to be executed based on the validity of the fan design performance metrics contained in the *Results.txt* file. Valid CFD results cause the script execution to proceed to the *Extract_Solution* block while invalid CFD results (PFResults = -1) will cause the script to proceed immediately to the *Post_Process* block as shown in Figure 67.

3.4.8 VisualScript Element Definition – Extract_Solution

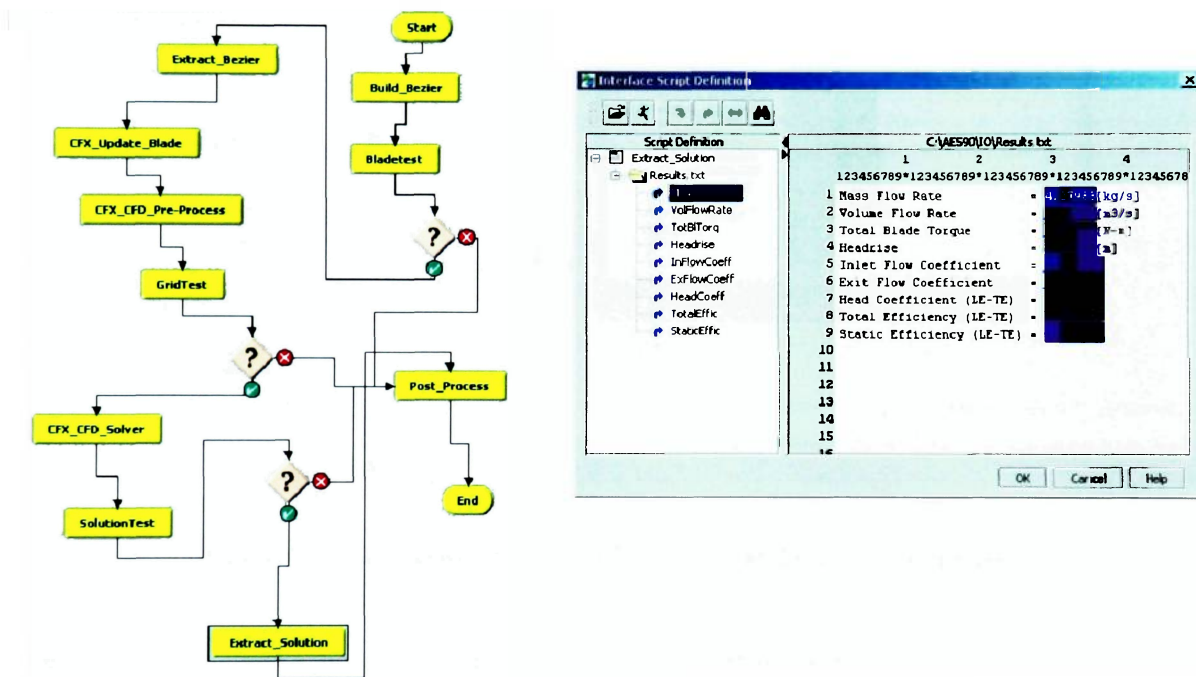


Figure 68: VisualScript Element Definition – Extract_Solution

Data Files: *Results.txt*

Scripts/Utilities *None*

The purpose of the *Extract_Solution* visualscript block as the name implies is to extract the fan design performance metrics (efficiency, MFR etc.,) computed from the CFD analyses. This block is only executed after the performance metrics are validated by the *SolutionTest* block previously discussed. A sample CFD results file containing computed

performance metrics is included in appendix A. the values in this file are extracted to internal, appropriately named visualscript variables. These internal visualscript variables are used to transfer the computed results (fan performance metrics) to the visualdoc optimizer as responses.

3.4.9 VisualScript Element Definition – Post_Process

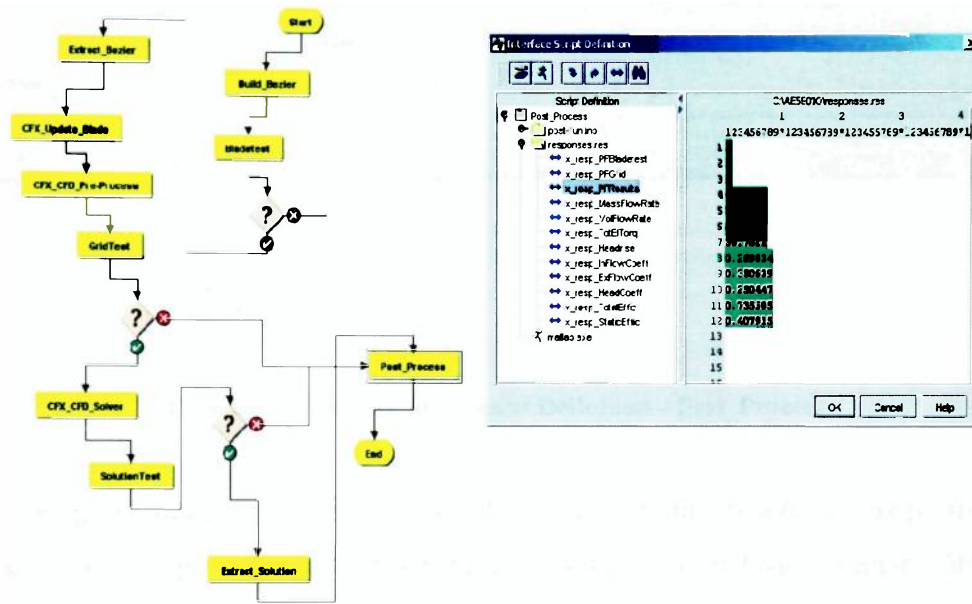


Figure 69: VisualScript Element Definition – Post_Process (Responses)

Data Files: *post_run.inp*, *responses.res*, *summary.res*

Scripts/Utilities: *postrun.m*

At this stage in the visualscript, the CFD analysis of the fan design is complete and the performance of the new fan design is known. In order to examine the history of the MDO analysis, the *Post_Process* block is developed to store all designs explored by the RSO algorithms. The *Post_Process* element writes the various internally stored test results as well as the computed performance metrics to two files: *post_run.inp* and *responses.res*. These files are read by the MATLAB script *postrun.m*, which performs the actual post processing of the data for the current design iteration

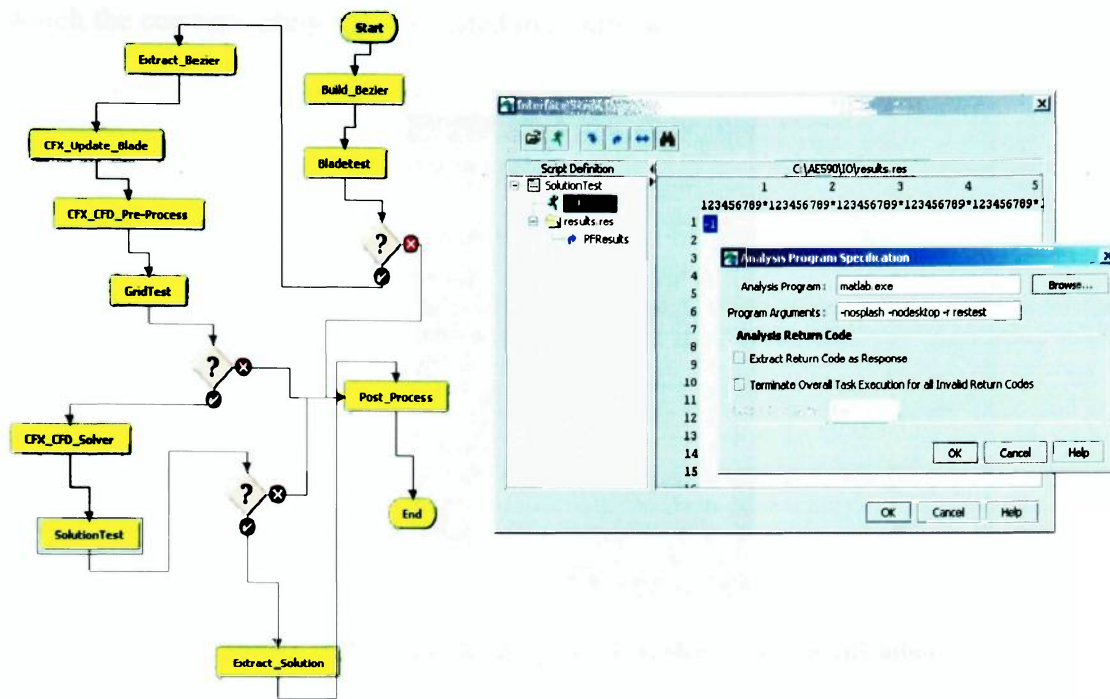


Figure 70: VisualScript Element Definition – Post_Process

All the design variables and responses which describe the fan design and its performance are read from the optimizer and written to an optimization analysis summary file called *summary.res*. A sample optimization analyses summary file as well as sample *Post_Process* input files are included in appendix A.

The completion of the *Post_Process* block essentially terminates the execution of the visualscript. All the previously presented elements of the visualscript combine to form a single analysis *program* that is used by the optimizer to evaluate the each fan design generated by the RSO algorithm.

The communication between the visualscript and the optimizer is done by writing all the responses specified in VisualDoc to the response transfer file (*resp.vef* by default). The specification of the transfer order is done by assigning a line number that correlates the internally stored visualscript variable to appropriate response in VisualDoc. In other

words, the line number for the visualscript variables in Figure 71 must match the order in which the corresponding value is listed in Figure 48.

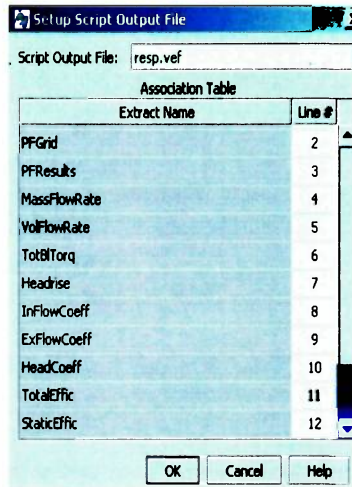


Figure 71: VisualScript Response Transfer Order Specification

A similar listing is used to setup the order in which the design variables will be transferred from the optimizer. They are written from VisualDoc to a visualscript input file (*dvar.vef* by default). The order of the design variables in the input file must match the VisualDoc variable specification (Figure 46 and Figure 47). The order specification is also done in visualscript by specifying a line number in which the design variable will be found in the input file coming from VisualDoc. In other words, the line number specified for each variable in Figure 72 must correspond to the correlating variable in Figure 46 or Figure 47.

The discussions presented in section 3.4.1 through 3.4.9 are intended to illustrate the details involved in the response analyses each fan design investigated during the MDO analyses for the benchmark study. Each elements discussed is executed during each design iteration by the visualdoc optimizer. Each element is generally dependent on the successful execution of the preceding elements. As a result, a failure to incorporate some robustness into the MDO design analyses process as is done using the *Pass/Fail* variables, generally results in failed or erroneous optimization analyses.

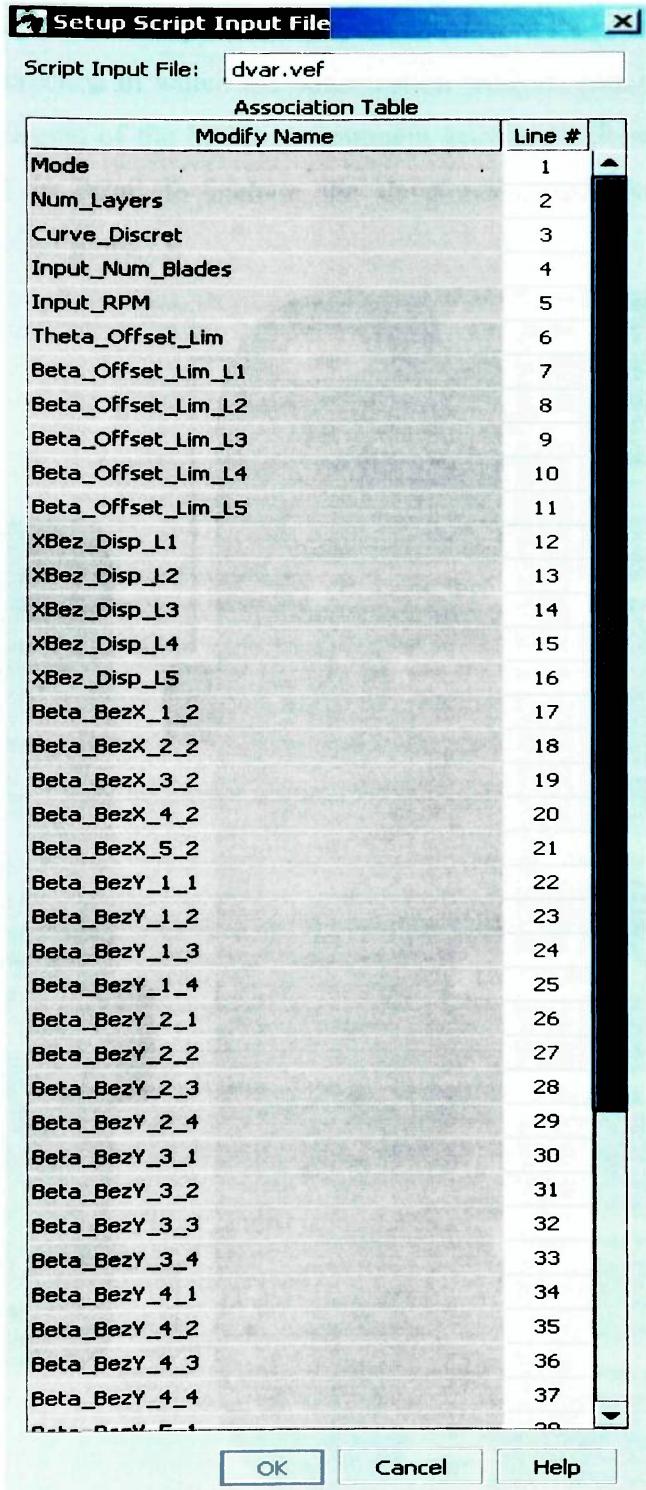


Figure 72: VisualScript Design Variables Transfer Order Specification

The visualscript is an important component of the MDO environment for the benchmark study. However, the visualdoc optimizer may be considered the most critical component as it controls the direction in which the optimization analyses proceeds. The interaction between the components of the MDO environment previously illustrated Figure 30 can now be expanded in detail to include the developed VisualDoc and VisualScript elements, as shown in Figure 73.

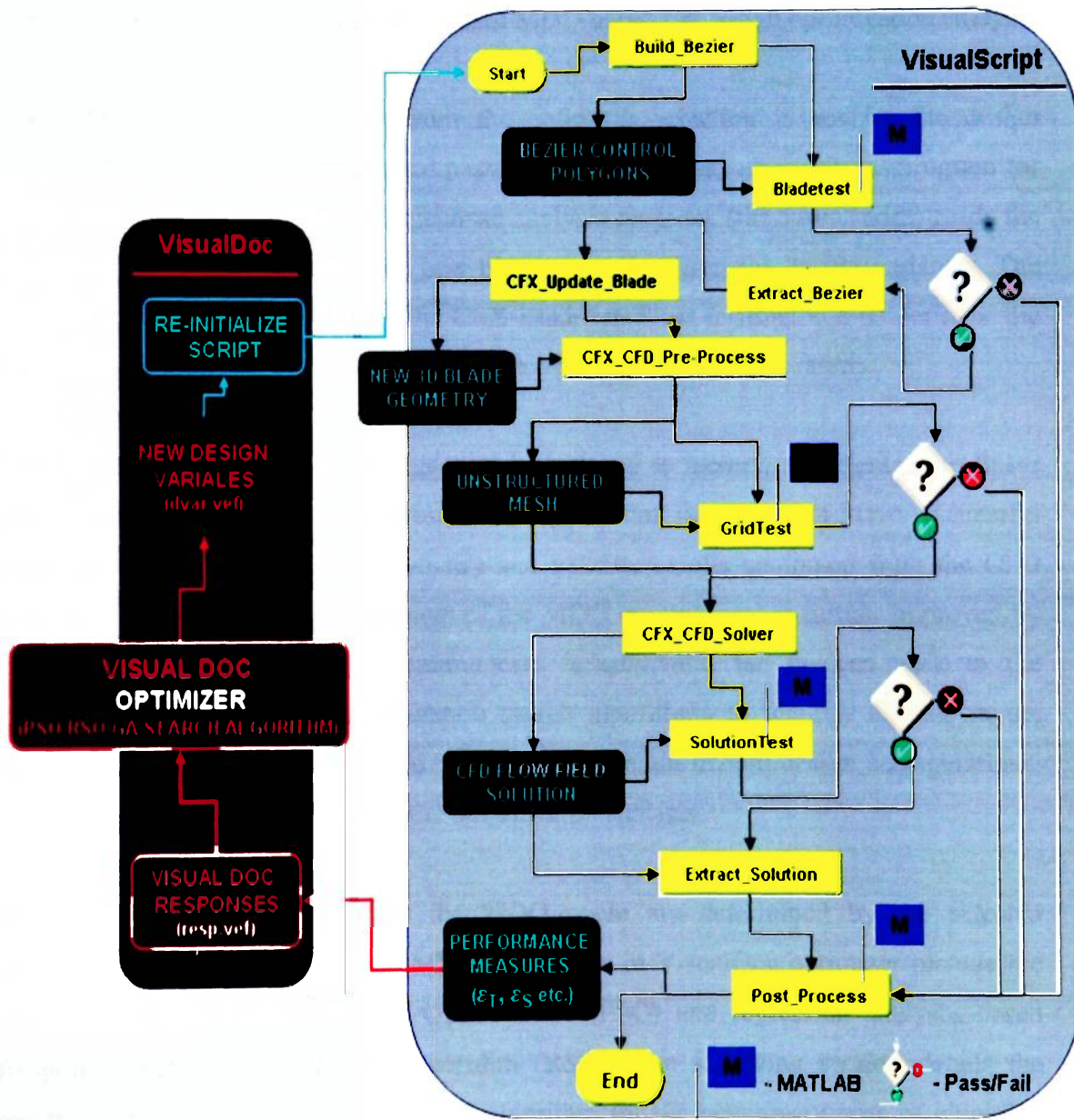


Figure 73: Detailed Components of MDO Environment

The creation of the automated MDO environment for the benchmark fan design case is now complete. The blade geometry is parameterized using bezier control polygons that define the camberline of the blade airfoil shape at 5 constant radius layers that span the blade from hub to shroud. The coordinates of the 4 control points for each of the 5 bezier polygons serve as the design variables for the optimization of the blade geometry. Including the number of blades and the rotational rate (RPM) of the fan, a total of 32 design variables are used for the benchmark MDO based fan design optimization study.

The MDO design cycle begins from the optimizer selecting values for the design variables for the current iteration and passing them to the response analyses program for evaluation of the fan design. The response analysis *program* (the visualscript) reads the design variables and generates the new blade geometry from the design variables. The CFD analyses is performed using the CFX-Bladegen(Plus) unstructured solver after the successful completion of all preprocessing steps including grid generation.

The MDO analyses program (visualscript) is designed to incorporate tolerance to failed grid generation and invalid CFD results through testing schemes that verify successful mesh generation for the blade geometry and validate results computed from the CFD analyses of the design. The robustness of the MDO environment is further improved by using Pass/Fail parameters to communicate failed/invalid fan designs back to the optimizer. This allows the optimization search algorithms to identify regions in the design space that should be ignored because they contain invalid design configurations, further increasing the efficiency of the optimization analysis.

The fan designs investigated by the MDO cycle are determined by the selected optimization algorithm. Available MDO algorithms in VisualDoc optimizer include the evolutionary type Particle Swarm Optimization (PSO) and regression analyses based Response Surface Optimization Algorithm (RSO). The following section details the results obtained using the selected RSO algorithm for the benchmark case of obtaining the maximum total efficiency for a parameterized fan blade design.

4 RESULTS

4.1 Base (Starting) Fan Design Models

The optimization analysis is performed using three different starting (base) designs. The objective in starting from 3 different base designs is in order to investigate the ability of the optimization search algorithm to converge to the same optimal design. The three base design and their properties are detailed in Figure 74 through Figure 76.

The primary difference between model1 and model2 as detailed in Table 1 is the decrease in the diameter of the fan. The deviation angle at the trailing edge (TE) of the shroud (tip) is also increased in model2. It should be noted that the diameter of the fan is not a design variable for the MDO optimization analysis for the benchmark case. Hence the variation in the diameter of the fans in model1 and model2 will allow for a measurement of the effect/influence of the diameter on the optimization analysis. The degree of variation in the optimal designs obtained starting from model1 and model2 will demonstrate the effect of the diameter on convergence to a single optimal solution.


		Model1
N (RPM)		1140
Diameter, D	(in.)	30
	(mm)	762
# Blades, B		9
R_h/R_t		0.4
C_t/C_h		2
c, t/m		-20/-5
Blade Vortex Model		-1
t_{max}/C		0
t_{LE}/t_{TE}		4/1
h_{LE} incidence Angle ($^{\circ}$)		2.5
s_{LE} incidence Angle ($^{\circ}$)		-2.5
h_{TE} deviation Angle ($^{\circ}$)		5
s_{TE} deviation Angle ($^{\circ}$)		5
Camber Load		Aft Tip and Mid Load

Figure 74: MDO Benchmark Fan Design Study – Base Model1

The primary difference between model1 and model3 is the slightly larger Hub-tip ratio in model3. The blade vortex model is also increased from -1 to 0.75 to study the effect of different vortex models on the optimization cycle.

For the benchmark case, additional constraints are imposed for the problem in order to yield fan designs that are physically reasonable and consistently perform better than the original design. The following are the additionally imposed performance constraints:

- Min. Static Pressure:
0.5in. H2O [$0.125e^{-3}$ MPa]
- Vol. Flow Rate Range:
 ± 1000 cfm [± 0.47 m³/s] from base design

The static pressure rise is not returned as a response to the VisualDoc optimizer. Thus, constraints cannot be directly imposed on this property. The consequences of this inability to directly impose constraints on the static pressure rise are subsequently discussed. The constraint on the volume flow rate is applied in the VisualDoc optimizer response specification as shown in Figure 48.

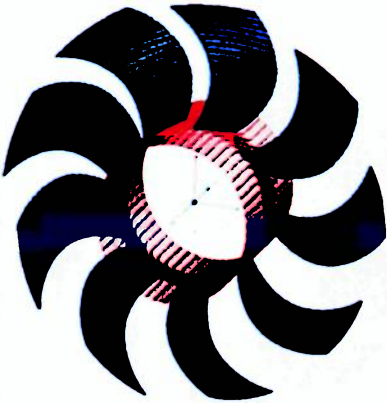
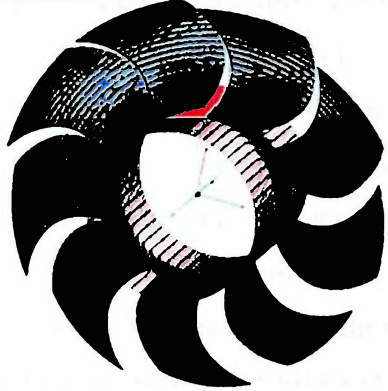
		Model2
N (RPM)		1720
Diameter, D	(in.)	24.3
	(mm)	617
# Blades, B		9
R_h/R_t		0.4
C_v/C_h		2
c, t/m		-10/4
Blade Vortex Model		-1
t_{max}/C		0
t_{LE}/t_{TE}		4/1
h_{LE} incidence Angle (°)		2.5
s_{LE} incidence Angle (°)		2.5
h_{TE} deviation Angle (°)		5
s_{TE} deviation Angle (°)		10
Camber Load		Aft Tip Load

Figure 75: MDO Benchmark Fan Design Study – Base Model2

We consider the results of the optimization analyses for the benchmark fan design case using the Response Surface Approximate Optimization (RSO) algorithm. The results of each starting design are individually discussed and a comparison between the optimized designs obtained in each case.

As previously noted in the discussion on the details of the RSO algorithm (section 2.3.1), it is highly impractical to use more than 10 design variables when the more accurate quadratic response model is used, due to the high computational effort required. For the benchmark fan design case, a total of 32 design variables are available including 30 design variable for the parameterization of the 3D blade geometry and two additional design variables (RPM and Num. of blades).

In the design variable specifications of VisualDoc Figure 46 and Figure 47, any of the design variables can be designated as constant by selecting the appropriate (constant) option in the *type* field for the design variable. The availability of this feature ensures that the RSO algorithm can be used for the benchmark fan design study with a reasonable amount of required computational effort.

		Model3	
		N (RPM)	1140
Diameter, D	(in.)	30	
	(mm)	762	
		# Blades, B	9
		R_h/R_t	0.45
		C_t/C_h	1.88
		c, t/m	-10/4
		Blade Vortex Model	0.75
		t_{max}/C	0
		t_{LE}/t_{TE}	4/1
		h_{LE} incidence Angle (°)	2.5
		s_{LE} incidence Angle (°)	2.5
		h_{TE} deviation Angle (°)	5
		s_{TE} deviation Angle (°)	10
		Camber Load	Aft Hub Load

**Figure 76: MDO Benchmark Fan Design Study
– Base Model3**

Since a maximum of 10 design variables can be varied during an optimization cycle, an approach for varying the 32 design variables in the fan design study has to be devised. The adopted strategy can be considered a *global optimization through the superposition of locally optimal solutions*. Essentially, the fan is optimized relative to an initial subset of design parameters with all other design variables kept constant. The optimized subset of design variables are then kept constant while the formerly constant variables are changed to varying (or continuous) design variables and the optimization analysis repeated. This allows for a number of variables less than the limit of 10 to be varied simultaneously.

The following table presents the order in which the variables are selected for investigation with the response surface optimization algorithms utilizing the strategy of optimization through superposition of optimal solutions.

Table 16: Variation Order of Design Variables (RSO Benchmark Study)

Order	Variables (Unless Otherwise Specified $r = 1..5$)	Total # Variables
1	Tangential Coord. of Interior Bezier CPs ($\beta_{r 2}, \beta_{r 3}$)	10
2	Circumferential LE sweep location (θ_r)	5
3	Meridional coordinate of CP2 ($\underline{M}'_{r 2}$)	5
4	<i>Displacement factor</i> for CP3 meridional coordinate (δ_r)	5
5	Tangential coordinate of TE bezier control points ($\beta_{r 4}$)	5
6	RPM & Num of Blades.	2

4.2 Results – Response Surface Optimization (RSO) Algorithm

The results obtained for the benchmark MDO design study using the Response Surface Optimization (RSO) Algorithm are presented in this section. The table below details the control parameters used for the optimization analysis particularly the design variables and objective function convergence criteria. [23] describes in more detail, the application of each parameter to the RSO implementation in VisualDoc.

Table 17: RSO Optimization and Convergence Parameters for Benchmark Fan Design MDO Study

Min Number of Design Points	4
Max Number of Design Points	106
Num. of User Supplied Design Points	1
Order of Approximations	Full Quadratic
Generate Initial Points	Simplex Design
Consecutive Iterations for Convergence	5
Initial Quadratic Relative Move Limit	0.2
Quadratic Absolute Move Limit	0.02
Relative Objective Convergence Tolerance	0.001
Absolute Objective Convergence Tolerance	0.0001
Relative Design Variable Convergence Tolerance	0.001
Absolute Design Variable Convergence Tolerance	0.0001
Constraint Tolerance	-0.03
Violated Constraint Tolerance	0.003
Objective	Maximize

4.2.1 RSO Results – Model1

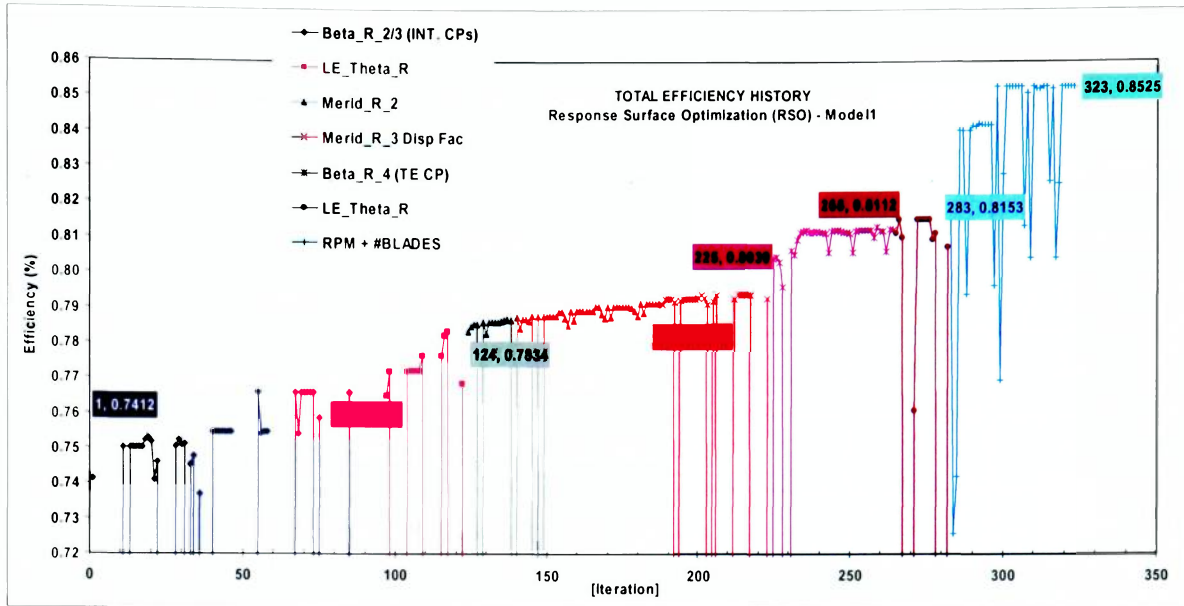


Figure 77: Model1 RSO Results – Target Function (Total Efficiency)

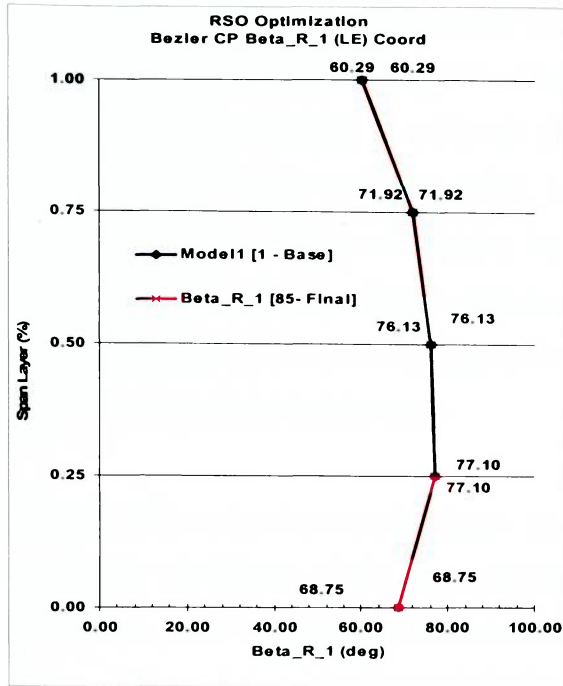
The target function plot for the first base model obtained using the Response Surface Optimization algorithm is shown in Figure 77 above. The target function plot shows a cumulative efficiency gain of 11% from the base design to the optimized design. The data labels in the target function plot show the value of the efficiency at the beginning of the each subset of design parameters as listed in Table 16. For example the first data label (1, 0.7412) shows the iteration number and efficiency before the variation of the angular coordinates ($\beta_{r|s}$) of the interior Bezier control points.

The next data label (97, 0.7650) shows the iteration number and total efficiency just after the interiors Bezier CPs have been deactivated as design variables (i.e. just before the optimization of the total efficiency relative to the circumferential LE sweep (θ) distribution of the blade. A total efficiency gain of approximately 2.5% is obtained by varying the angular coordinates of the interior Bezier control points. A summary of the influence of each set of design variables is shown in Table 18.

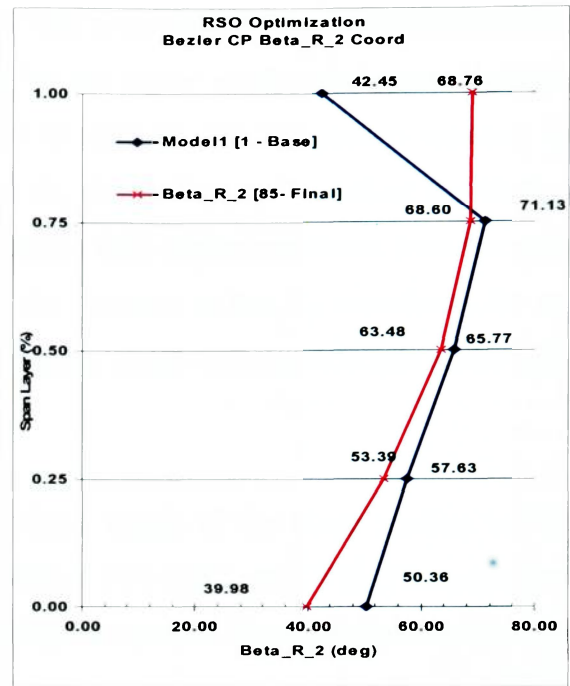
Table 18: Model1 RSO Results - Influence of Design Variables on Target Function

Order	Variables (Unless Otherwise Specified $r = 1..5$)	Total # Vars.	Gain in Total Effic. (%)
1	Tangential Coord. of Interior Bezier CPs ($\beta_{r 2}, \beta_{r 3}$)	10	2.38
2	Circumferential LE sweep location (θ_r)	5	1.84
3	Meridional coordinate of CP2 ($\underline{M}'_{r 2}$)	5	0.77
4	<i>Displacement factor</i> for CP3 meridional coordinate (δ_r)	5	1.28
5	Tangential coordinate of TE bezier control points ($\beta_{r 4}$)	5	0.73
6	Circumferential LE sweep location (θ_r) (Repeated)	5	0.41
7	RPM & Num of Blades	2	3.72
Model1 Total Efficiency Gain (%)			11.13

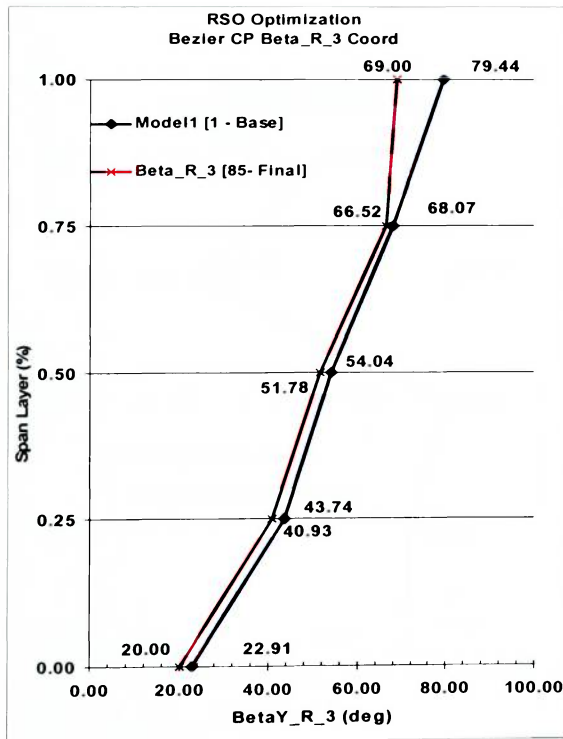
As can be seen from the above table, the maximum gain in total efficiency is from the variation of the rotational rate and the total number of blades in the fan cascade. The next most significant gain in efficiency is obtained from varying the angular coordinates of the interior Bezier control points. It is worthwhile to note that the primary consequence of varying the angular coordinates of the Bezier control points is the modification of the camber of the blade profile at the respective span locations. The following plots show the original and optimized curves for the tangential ($\beta_{r|s}$) coordinates for the bezier CPs at the various span layers for model1.



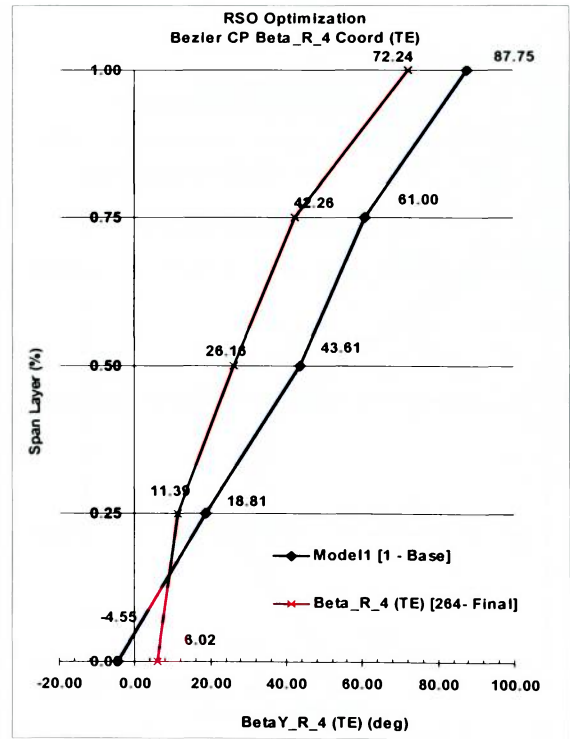
(a)



(b)



(c)



(d)

Figure 78: Model1 RSO Results – Bezier CP Tangential (β_{rs}) Coordinates

In Figure 78, the optimized distribution from hub to shroud of the tangential coordinates of the Bezier CPs is shown. The [85 -Final] in the legend implies that convergence for the plotted design variable is achieved after 85 iteration (i.e. the plotted variables are “turned off”). As seen in Figure 78(a), the $\beta_{r_{ls}}$ distribution at the LE of the blade is maintained, for the purpose of retaining the design inlet angles/conditions from the radial equilibrium and velocity triangle analyses. An increase in the $\beta_{r_{ls}}$ coordinate for any bezier control point causes a change in the camber shape and chord length of the blade profile at that particular span layer.

Further influencing the camber shape and chord length of the blade profiles are the meridional coordinates of the two interiors Bezier CPs (CP2 and CP3). The meridional coordinates of CP3 is deduced from substituting the value of the respective *displacement factors* (δ) equation 3.4. Plots of the base and optimized values for the meridional coordinate of CP2 and displacement factor for CP3 at each span layer are shown below:

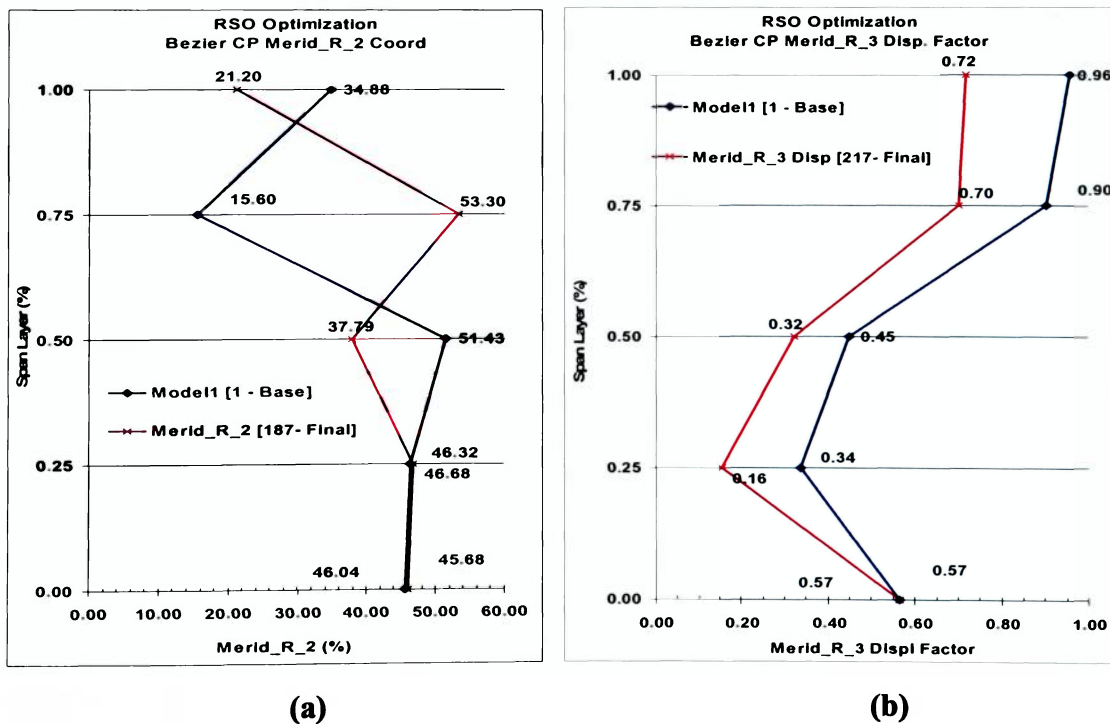


Figure 79: Model1 RSO Results – CP2 Merid Coord ($M'_{r_{ls}}$) and Displ. Factor (δ)

It is somewhat difficult to visualize how the various optimized Bezier CPs coordinates change the camber shape of the blade profile at each span layer. Hence plot of the blade profile camber shape in the original and optimized blade passages are shown below. All subsequent spanwise layer plots are provide only for the hub (0.0%), midspan (50%) and shroud (100%) span locations (SL):

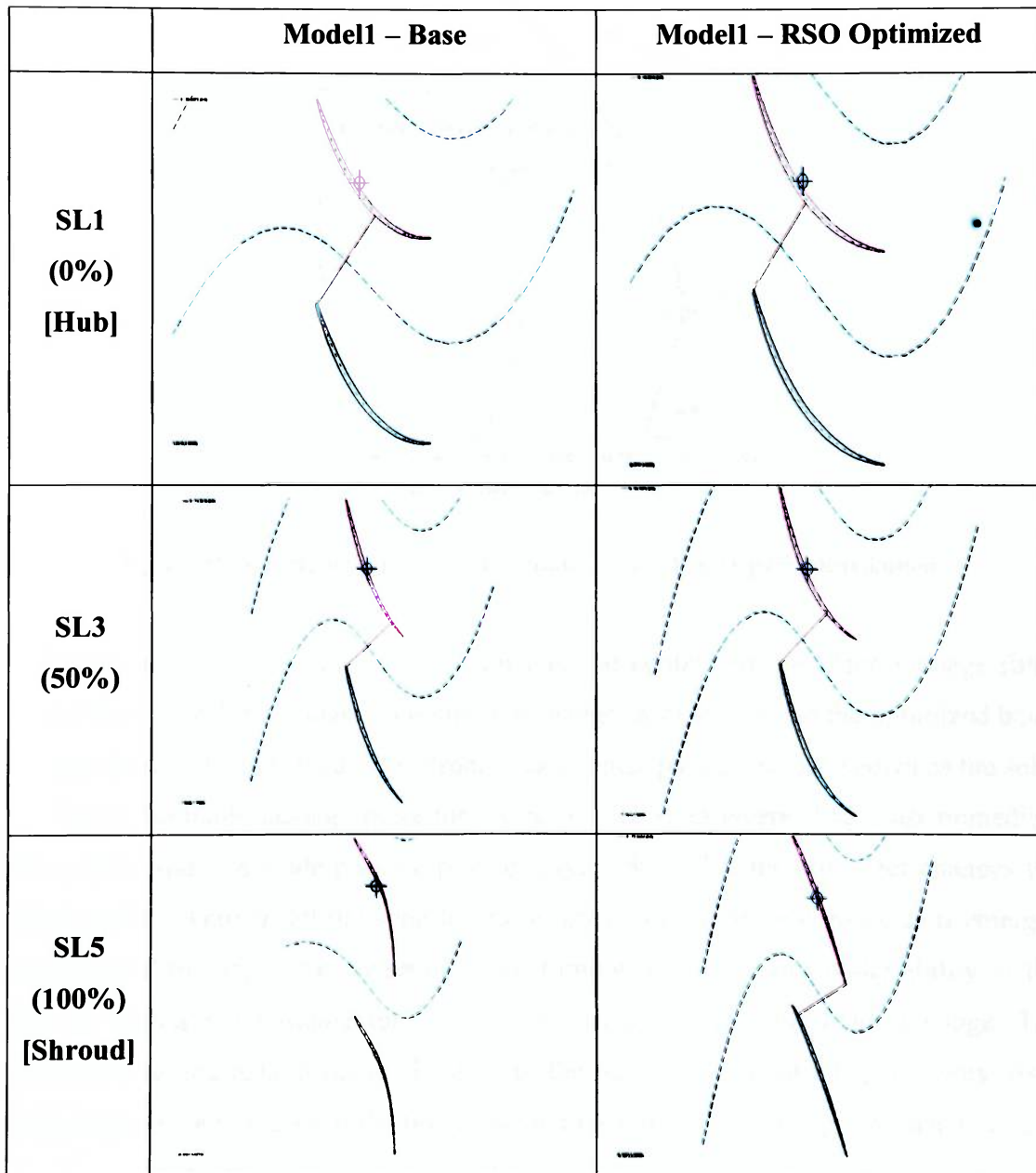


Figure 80: Model1 RSO Results – Blade Profiles/Passages

The last major geometric design variable is the leading edge sweep (θ_{LE}) of the blade. The base and optimized LE sweep distribution of the blade is shown below:

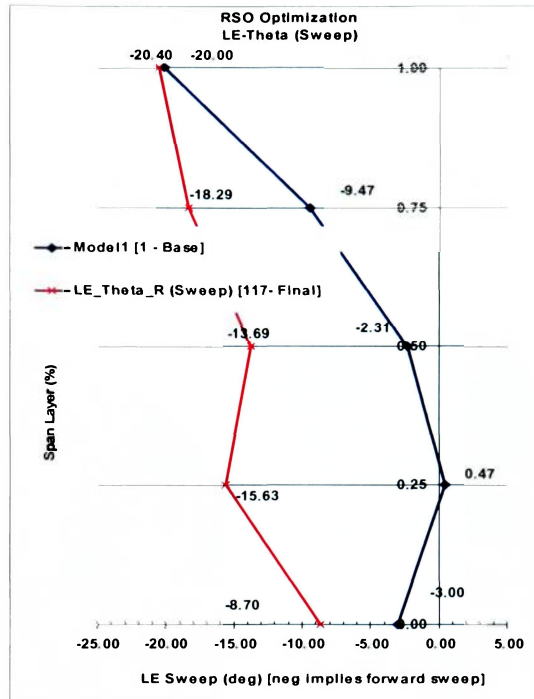


Figure 81: Model1 RSO Results – Circumferential LE Sweep (θ) Distribution

In Figure 80, the blade profiles are shown in the context of the blade passage (blue broken line) at each span layer, allowing the change from the base to the optimized blade passage shape to be observed. The throat in each blade passage is also shown as the solid red line in the blade passage plots for the respective span layers. The most immediate observation from the blade passage plots of Figure 80 is that the optimizer changes the airfoil camber shape at all the span layers in order to turn the flow more as it emerges from the trailing edge. The consequence of turning the flow more is the ability of the blade to impart more momentum to the flow as it traverses the blade passage. The optimizer also attempts to move slightly aft, the throat location at all span layers. As a result, a larger surface area of the blade is able to act on the flow (larger wetted area) and a larger pressure rise is obtained due to the resulting increase in the cross-sectional area normal to the flow as it exits the blade passage.

The combined effect of the optimized blade profile camber shapes as well as the new LE sweep distribution can be seen in Figure 82. The continuous polynomial-like LE sweep of the base design is changed to the *S-shaped* LE sweep in the RSO optimized design. As will be subsequently seen, the S-shaped leading edge sweep is characteristic of all optimized fan designs obtained using the RSO algorithm. Subsequent sections briefly discuss clarifying the nature of this LE shape, whether it is a consequence of actual physical fluid flow phenomena or a numerical effect of the blade geometry parameterization scheme.

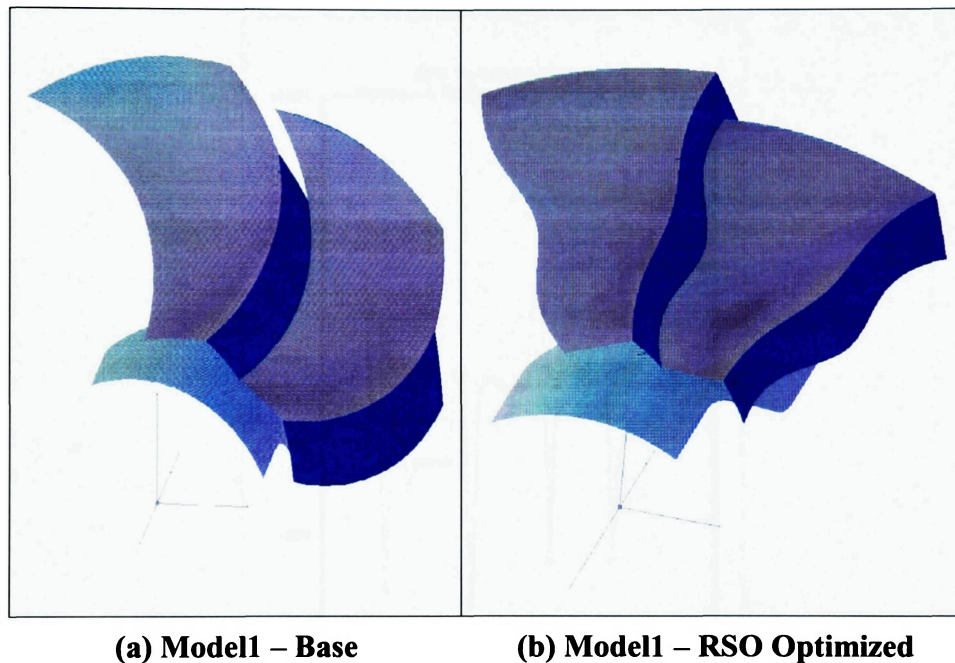


Figure 82: Model1 RSO Results – Blade Passage (with Throat Surface)

The optimization of the throat surface can also be more clearly seen in Figure 82. The throat surface (shown in blue) does not extend all the way to the shroud in the base design, which is essentially an indication that only a portion of the blade actually imparts useful work to the fluid as it passed through the fan. In the optimized blade geometry, the throat surface completely covers the inlet of the blade passage from hub to shroud. As a result, more of the blade surface acts on the flow, which is expected to result in a higher amount of momentum being imparted to the flow. This increased momentum is physically manifested as a greater pressure rise in the optimized blade geometry.

The Bezier CP coordinates and the LE sweep distribution constitute the geometric design variables which define the 3D shape of the fan blade. The remaining design variables including the rotational rate (RPM) and number of blades simultaneously affect the geometry of the blade as well as the operating conditions of the fan. The variable history for these design parameters are presented below. Note that the plots do not begin from the first design iteration where they are designated as constants. Rather the plots begin from the design iteration when these variables are “turned on,” which is iteration 283 for model1.

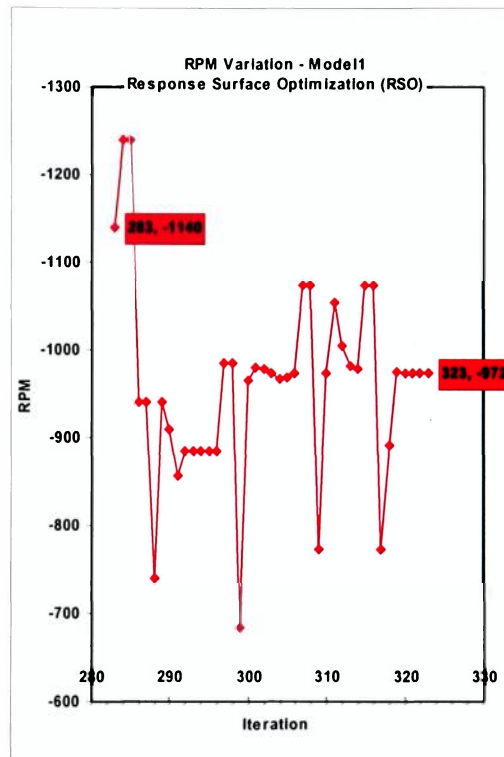


Figure 83: Model1 RSO Results – RPM Optimization History

The rotational rate of the fan is decreased from the initial rotational rate of 1140 rpm to the final value of 972 rpm. The reasons for the decrease in the rotational rate may stem from the increased efficiency of the optimized blade geometry in imparting work to the flow. Since efficiency is a measure of the ratio of the work output to the work input, the optimizer attempts to lower the power input by reducing the rotational rate of the fan. The reduction in the rotational rate of the fan is also accompanied by an increase in the

number of blades in the complete blade cascade from 8 in the base model to the 12 in the final design as shown in Figure 84.

Another effect of the increase in number of blades is a reduction in the amount of aerodynamics loading per blade. A decreased blade passage size implies less mass of fluid on which work has to be performed. Highly loaded blades are known to be sources of losses in turbomachinery, as they often cannot achieve the turning required to impart sufficient momentum to the flow. This often leads to an onset of flow separation in the blade passage. The traditional remedy to this has typically been to increase the solidity of the cascade through an increase in the number of blades similar to the increase implemented by the optimizer.

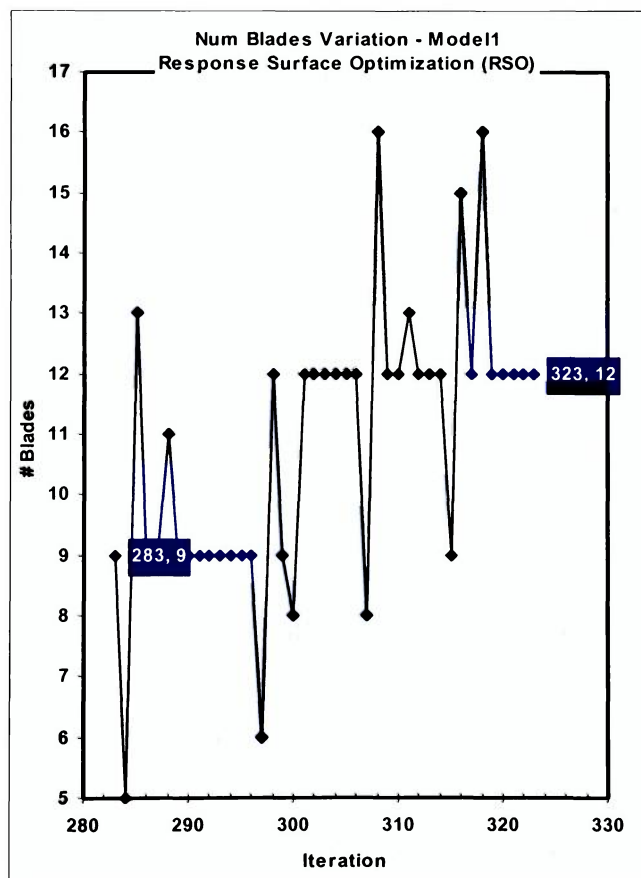


Figure 84: Model1 RSO Results – # Blades Optimization History

The improvement in the performance (total efficiency) of the fan design can be observed in a detailed comparison between CFD results for the base model and the optimized design. As expected, Figure 85 shows the increase in the streamwise, mass-averaged static pressure rise obtained for the optimized blade geometry. Note that because the geometric blade parameters at the LE are kept constant (ref Figure 78), the inlet conditions between the base and optimized geometry are the same. The increase in static pressure rise is obtained purely as a result of the optimization of the blade geometry and not from a modification of the inlet conditions.

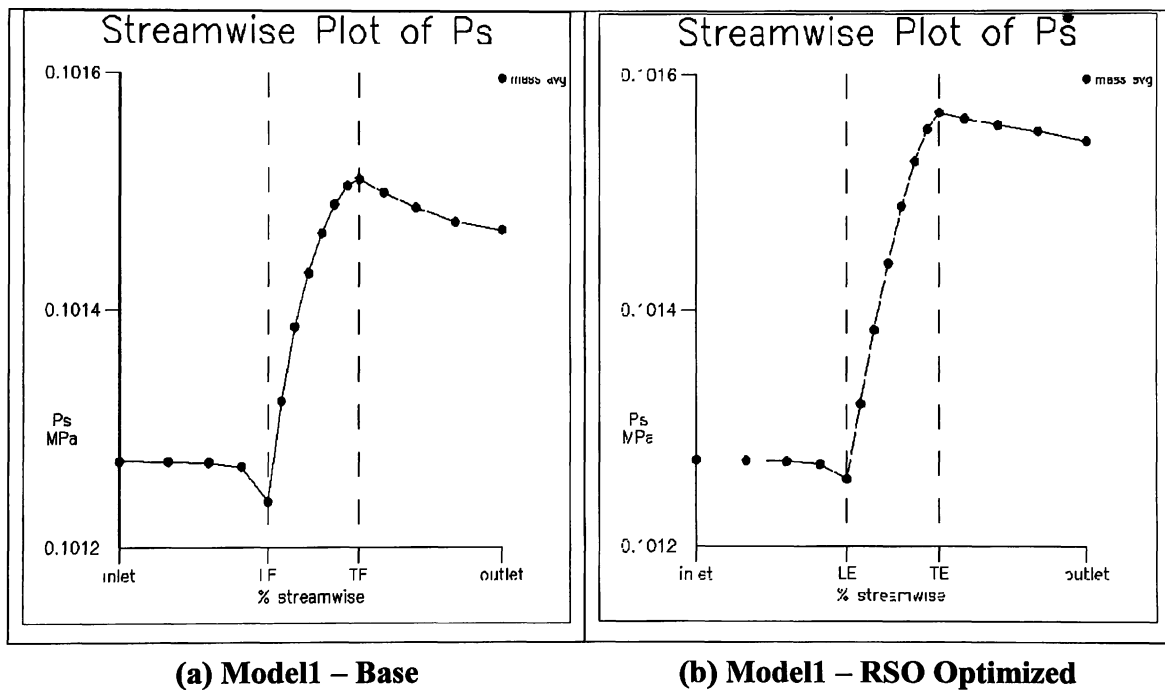


Figure 85: Model1 RSO Results – Static Pressure, P_s (Mass Avg.)

The reasons for the significant increase in the static pressure rise are further observed when the blade loadings at the 5 span layers are compared between the base model and the optimized designs. Generally the blade loading is improved as a result of the optimization of the blade passage shape. The changes in the camber shape of the blade profiles result in a modification of the throat surface and the geometry of the blade passage.

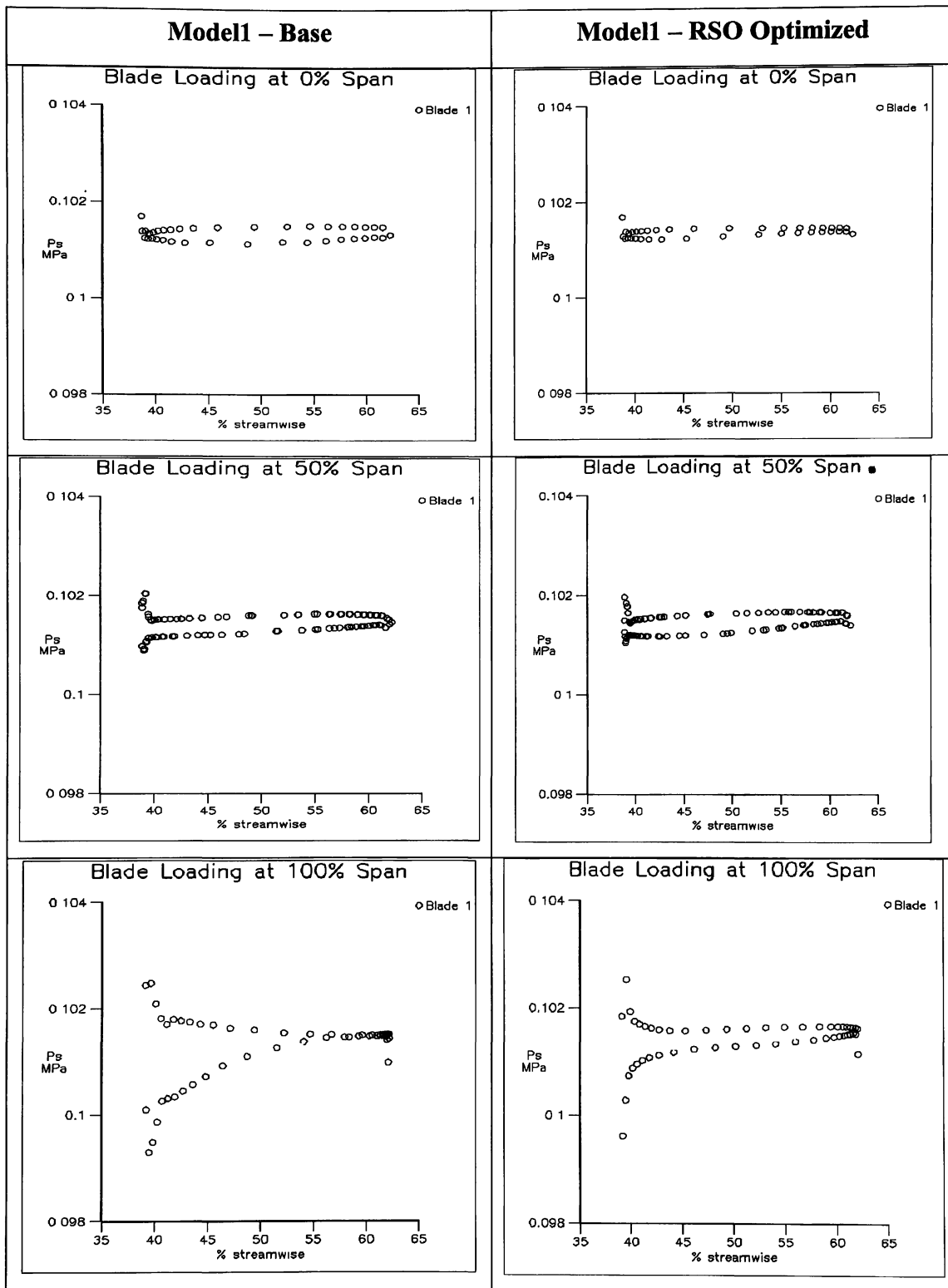


Figure 86: Model1 RSO Results – Blade Loading (By Span Layer)

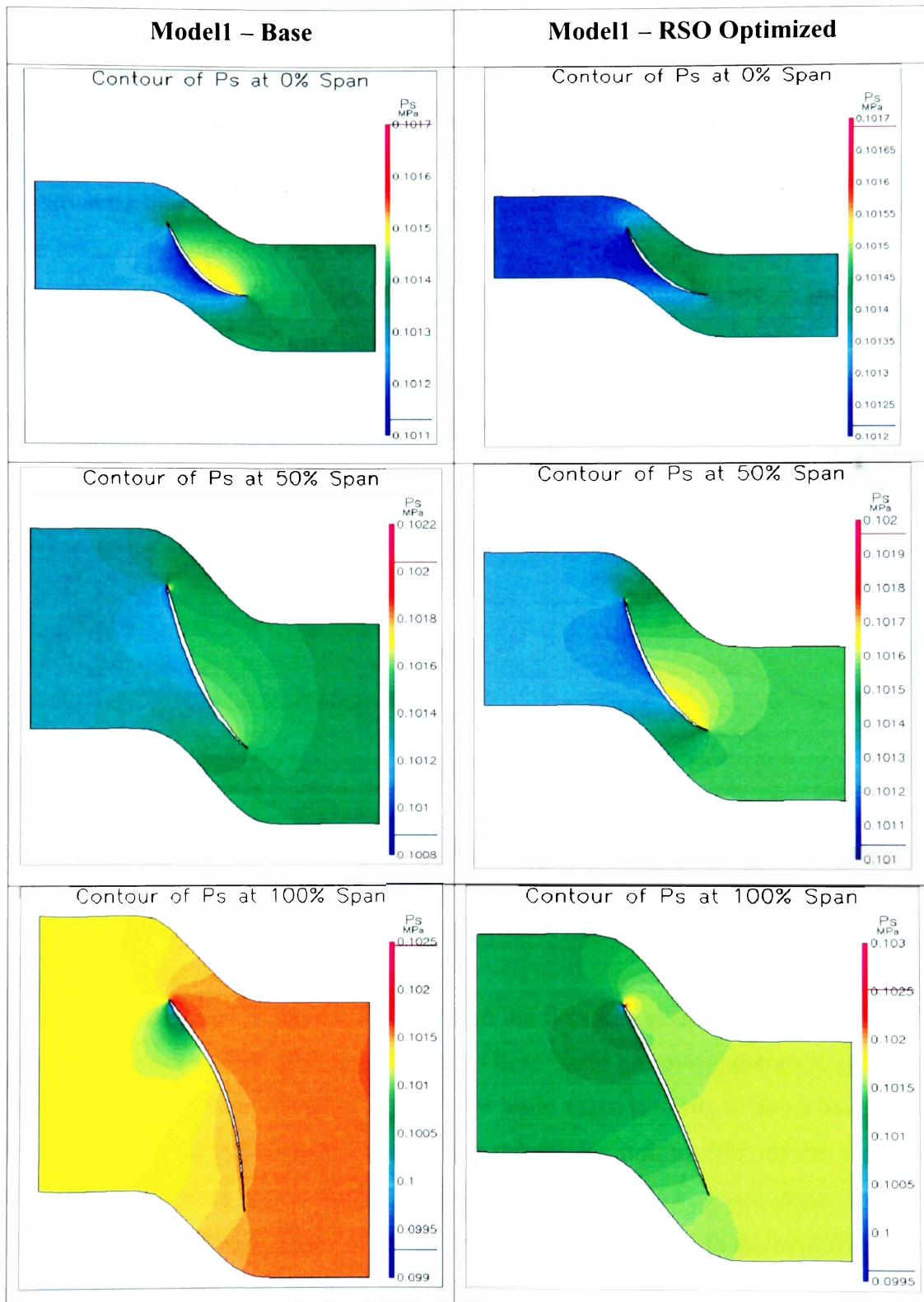


Figure 87: Modell RSO Results – Static Pressure (P_s) Contours

The various span layer plots in Figure 86 and Figure 87 show the improvements made from the base design to the optimal design particularly at the uppermost span layers of the blade. In Figure 86 and Figure 87, the effect of the awkward shape of the blade profile at the tip can be seen in the pressure contour plots. In the base model design, at the shroud trailing edge, the pressure and suction surfaces are almost at the same pressure value essentially nullifying any work input to the fluid. In the optimized blade, the new shape of the blade profile at that location clearly delineates the pressure and suction surfaces allowing for more work input from this region of the blade geometry.

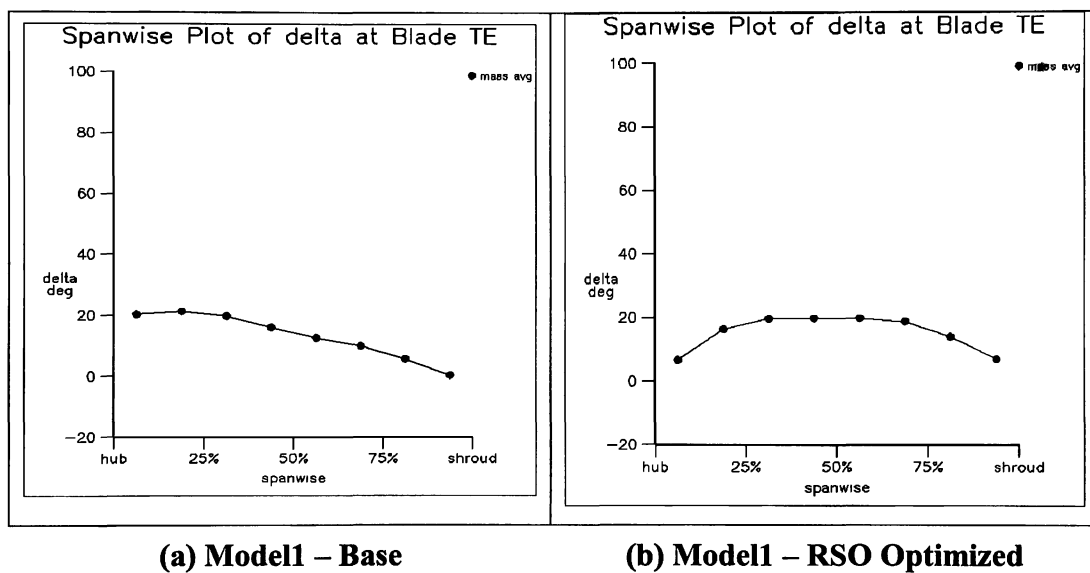


Figure 88: Model1 RSO Results – TE Deviation Angle (delta) Distribution

The ability of the flow to more efficiently turn the flow can also be seen in the relative velocity (W) vector plots of Figure 89. In the base blade geometry, there are generally more deviations from the direction in which the blade shape is trying to direct the flow as it leaves the trailing edge. In the case of the shroud, the trailing edge of the awkward blade shape is actually almost normal to the direction of the flow. These issues combine to adversely affect the efficiency of the blade as they induce losses in the performance of the fan. These losses are reduced in the optimized blade shape where the flow still deviates from the trailing edge in most span locations but the magnitude of the deviation is generally less than is the case in the original design.

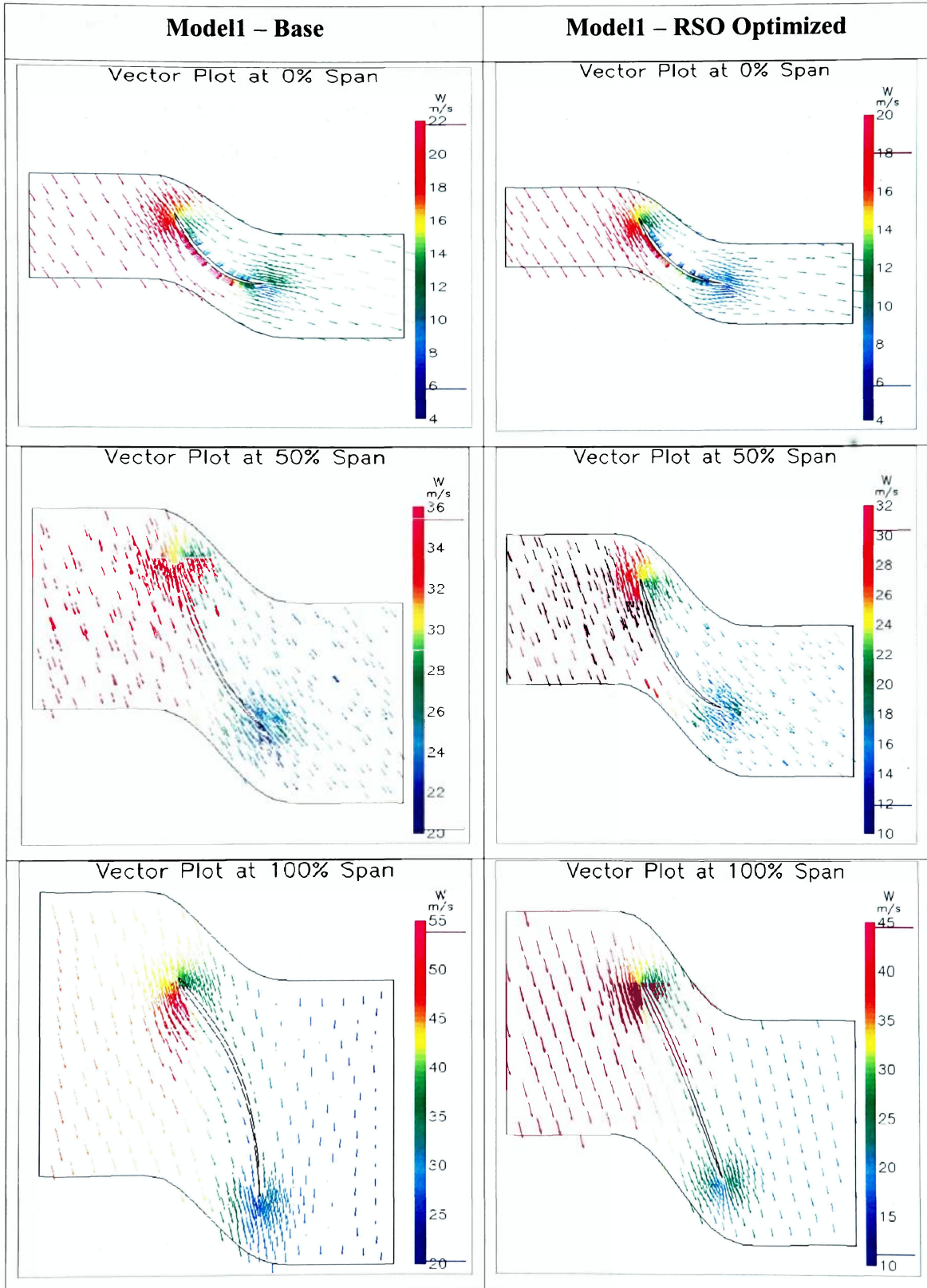
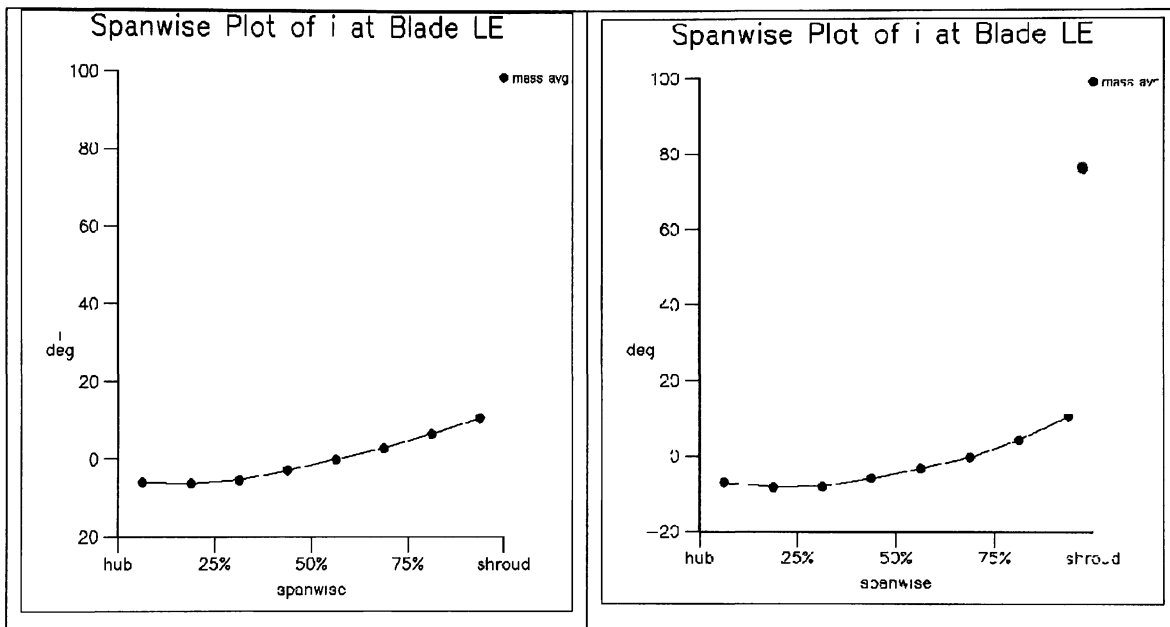


Figure 89: Modell RSO Results – Relative Velocity (W) Vector Plots

The reduction of the deviation at the trailing edge of the optimized blade geometry is captured in the mass-averaged trailing edge deviation angle (δ) of Figure 88. The reduction is particularly significant close to the hub where the deviation is reduced from 20° in the original design to about 5° in the optimized fan blade. Another potential measure of losses in the blade passage flow is the LE incidence angle distribution (mass-averaged) which is shown below:



(a) Model1 – Base

(b) Model1 – RSO Optimized

Figure 90: Model1 RSO Results – LE Incidence Angle (i) Distribution

There was no significant difference in the blade leading edge (LE) incidence angle. A potential reason for this result is because the LE geometrical design variables are kept constant to maintain inlet conditions from the initial radial equilibrium analyses for the blade design. That an 11% increase in total efficiency is obtained is an indication of the efficacy of the MDO design optimization methodology. A considerable gain in efficiency is achieved from a modification of the 3D geometry.

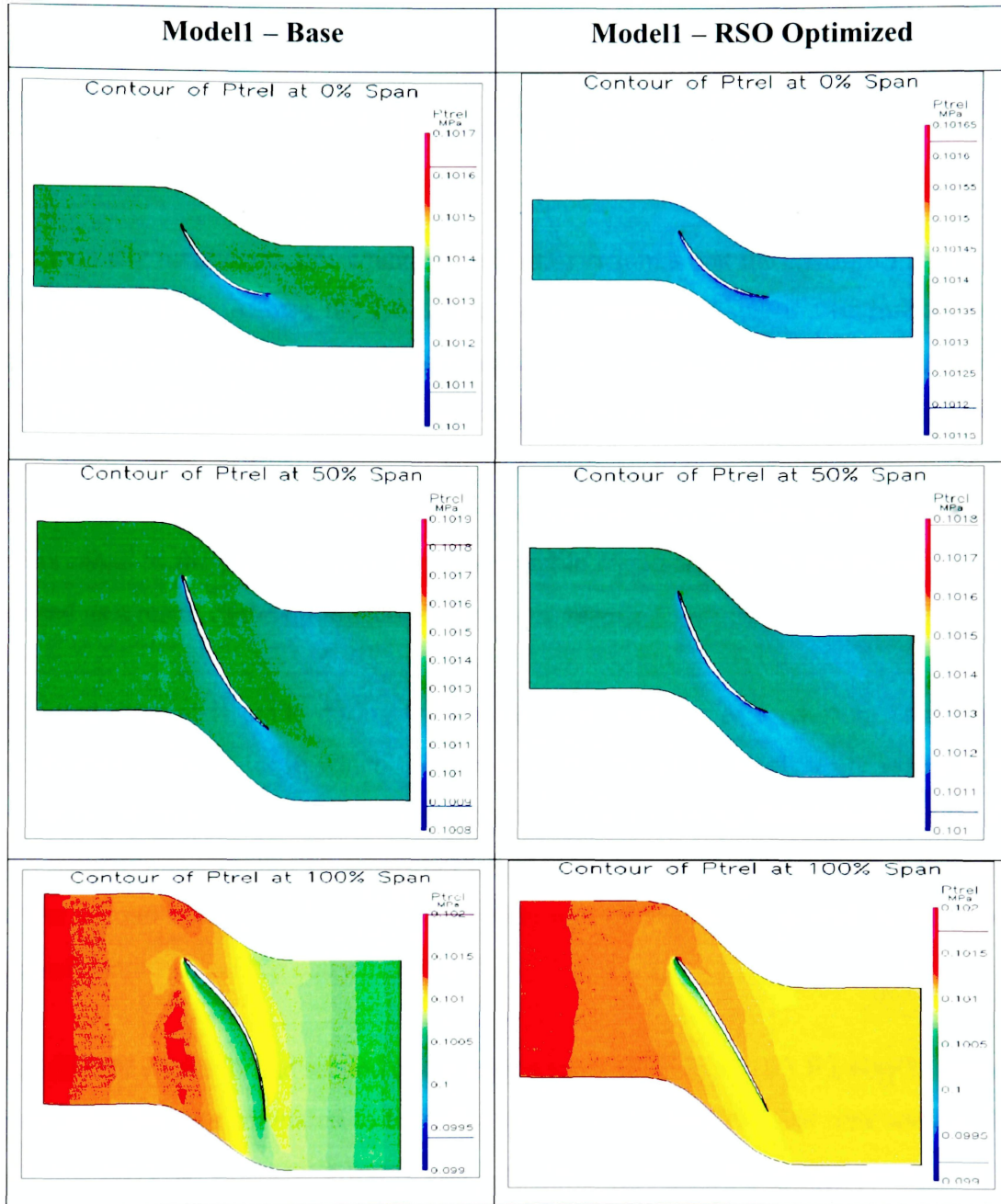


Figure 91: Modell RSO Results – Relative Total Pressure (P_{Trel}) Contours

An overall picture of the total efficiency increase is captured by the total pressure in the relative frame of the blade (P_{Trel}). In an ideal design with 100% efficiency, there should be no change in the relative total pressure from the LE to the TE of the blade. The analytic reason for this can be observed when the relationship (equation 3.8) for the computation of the total efficiency is considered:

$$\varepsilon_T|_{LE-TE} = \left[\frac{\partial P_{T_{abs}}}{\partial P_{T_{abs}} + \partial P_{T_{rel}}}_{avg}^{mass} \right] \quad (3.8)$$

As previously mentioned, this equation essentially requires that the change in the relative total pressure (δP_{Trel}) from the LE to the TE be minimal. Thus, the optimized design should display a generally more uniform relative total pressure distribution. The relative total pressure contour plots for the base and optimized design are shown in Figure 91 using a scale local to each span layer so as to more accurately capture the relative total pressure variation. The most significant improvement from the base to the final design occurs close to the shroud where the considerable P_{Trel} variation in the initial design is reduced as a result of the optimized blade profile shape.

A limited attempt is made during the optimization analyses of model1 to consider the significance of the order in which the groups of the design parameters are varied during the optimization study. To do this, the circumferential LE sweep distribution is *turned on* for a second time after the TE tangential CP coordinates are optimized. This is the reason for the second θ_r curve in the target function plot of Figure 77 as well as the second circumferential LE sweep entry in Table 18.

The efficiency gain for the second optimization study relative to the LE sweep is found to be considerable less compared to the efficiency gain obtained the first time around. This does not serve as exhaustive proof that the order in which the design parameters are varied is not significant when using an optimization through superposition approach. However, it does lend some credence to the utilization of the approach in the current MDO based fan design optimization study. As a result, no attempt is made to repeat an optimization study relative to a set of design variables. It is assumed that once the optimization is converged relative to a subset of design parameters, the computational expense outweighs the gain in trying to re-optimize relative to the same subset of design parameters. The data suggests that in the current MDO based fan design optimization

study, the potential gain in the target function may not be significant. A summary of the results for the base and optimized model designs obtained is presented below. For the presented results, the 2nd order discretization scheme for the CFD solver, so as to obtain the most accurate prediction of the performance of the base and optimized designs.


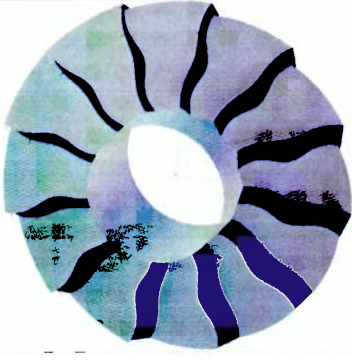
Algorithm: Response Surface Optimization (RSO)		
		
	Model1 – Base	Model1 - Optimized
RPM	1140	972
# Blades	9	12
VFR (m³/s)	3.617	3.616
Torque (Nm)	14.31	17.85
Head Rise (m)	31.14	37.66
Static Effic. (ϵ_{ST})	0.526	0.598
Total Effic (ϵ_T)	0.765	0.871
δP_{Tabs} (MPa)	$0.362e^{-3}$	$0.438e^{-3}$
dP_{Trel}	$-0.111e^{-3}$	$-0.064e^{-3}$
dP_{st}	$0.248e^{-3}$	$0.301e^{-3}$

Figure 92: Model1 RSO Results – Fan Design CFD Analysis Summary

4.2.2 RSO Results – Model2

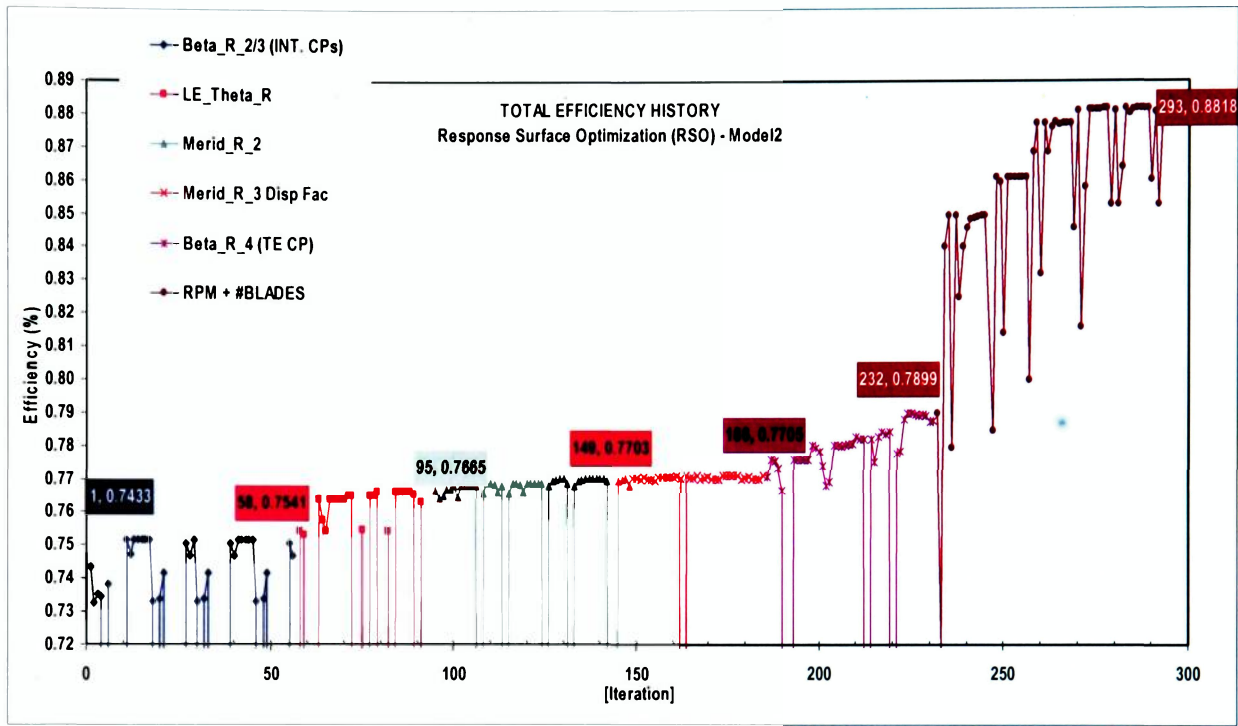


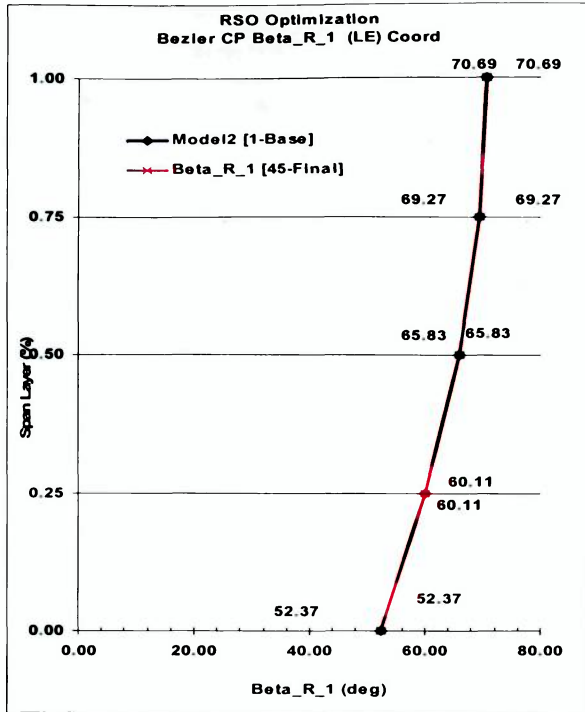
Figure 93: Model2 RSO Results – Target Function (Total Efficiency)

The target function plot for model2 above shows a gain of 14% in total efficiency. The data labels in the target functions plot show the design variable dependence of the gains in the total efficiency. As is the case for model1, the largest increase in total efficiency is obtained when the rotational rate (RPM) and number of blades in the cascade are varied. Table 19 shows the gain in efficiency obtained from the sets of design parameters. In the MDO design optimization for model2, the largest gains in total efficiency are obtained from the modification of the RPM and number of fan blades. The efficiency gain from modifying the rotational rate and number of blades is much more significant in magnitude compared to the increase obtained for model1. The total efficiency gain of 13.9% is also comparable to that obtained for model1 (11%). As is the case in the previous base model results, significant efficiency gains are obtained from the optimization of the interior Bezier CP angular coordinates as well as the LE sweep distribution.

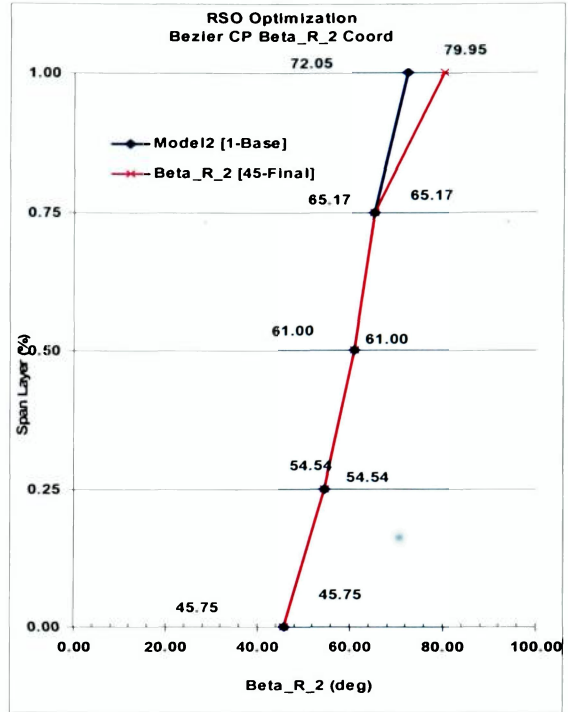
Table 19: Model2 RSO Results - Influence of Design Variables on Target Function

Order	Variables (Unless Otherwise Specified $r = 1..5$)	Total # Vars.	Gain in Total Effic. (%)
1	Tangential Coord. of Interior Bezier CPs ($\beta_{r 2}, \beta_{r 3}$)	10	1.08
2	Circumferential LE sweep location (θ_r)	5	1.24
3	Meridional coordinate of CP2 ($M'_{r 2}$)	5	0.38
4	<i>Displacement factor</i> for CP3 meridional coordinate (d_r)	5	0.02
5	Tangential coordinate of TE bezier control points ($\beta_{r 4}$)	5	1.94
7	RPM & Num of Blades	2	9.19
Model2 Total Efficiency Gain (%)			13.85

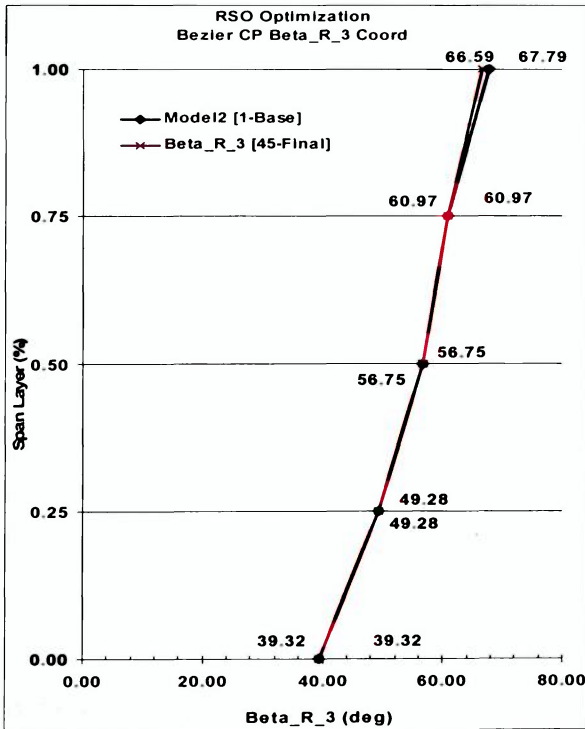
Items 3 and 4 in Table 19 yield the least in terms of increase in the total efficiency. This is similar to the results obtained in the previous base model (model1). The significant gain obtained from the angular coordinates of the Bezier CPs suggests that the optimized design is obtained by decreased losses in the blade passage from changes in the geometry of the blade trailing edge. The considerable modification in the trailing edge CP angular coordinates from the base to the optimized design can be seen in Figure 94(d). The angular coordinates for the other three Bezier CPs at all span layers are only slightly modified from the base to the optimized design. The only major modification for the interior Bezier CPs (Figure 94b and Figure 94c) occurs at the shroud. The LE Bezier CPs are also constant between the base and optimized designs as is the case in model1.



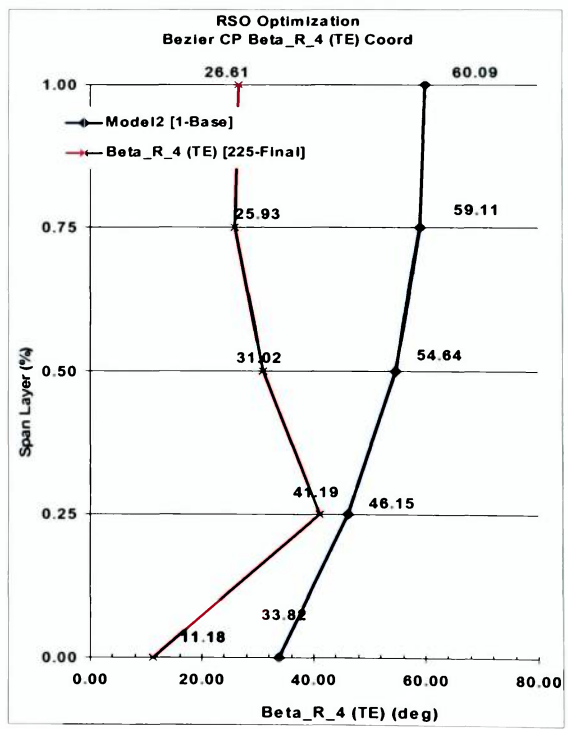
(a)



(b)



(c)



(d)

Figure 94: Model2 RSO Results – Bezier CP Tangential ($\beta_{r/s}$) Coordinates

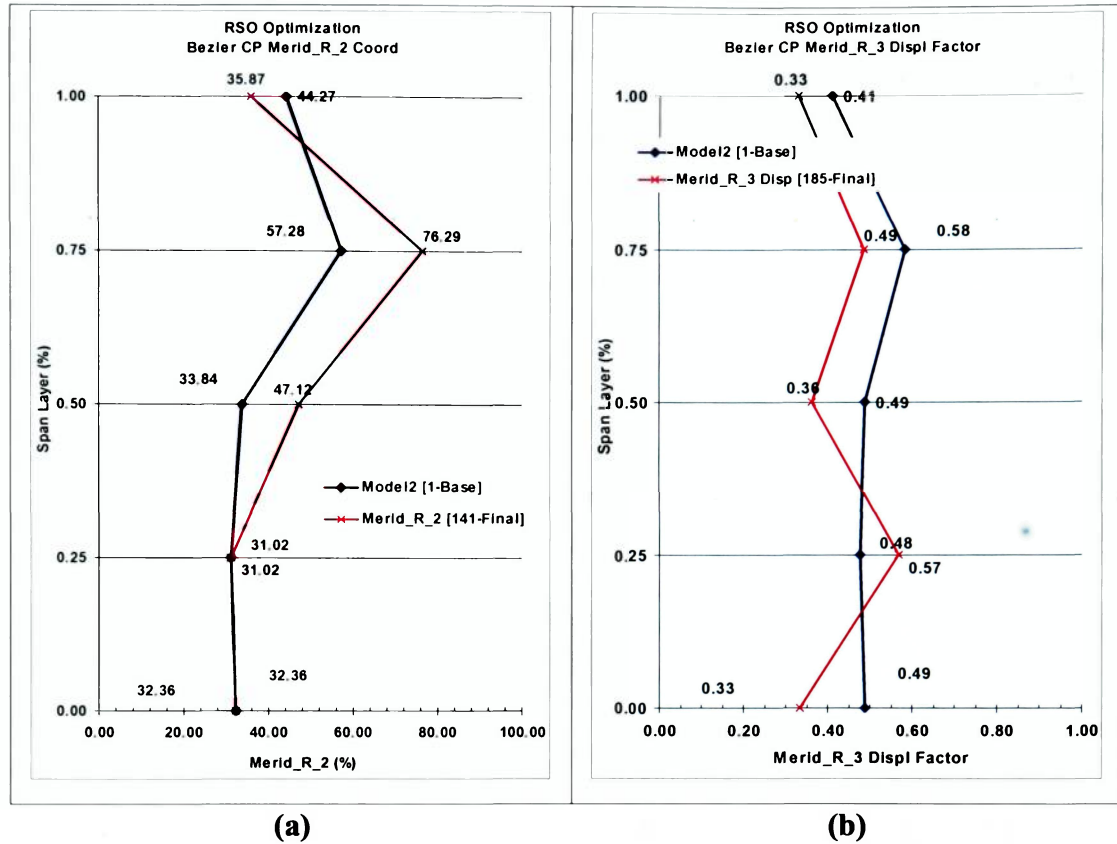


Figure 95: Model2 RSO Results – CP2 Merid Coord (M'_{rs}) and Displ Factor (δ_r)

The design variables defining the meridional coordinates for the interior bezier CPs (items 3 and 4 on Table 19) do not yield significant gains in the total efficiency. The variables plots in Figure 95 demonstrate the exploration of the design variables by the RSO algorithm. The relatively minute improvements in efficiency due to the meridional design variables may be numerical phenomena rather than actual physical consequences. It appears that the sensitivity of the target function to the meridional design variables is relatively minuscule. Consequently, changes in the meridional coordinates do not yield improvements in the target function for the optimized design.

Significant increases in the target function for the model2 optimization study are obtained from the variation of the TE bezier control points. As is the case for the optimization of base model1, changes in the blade passage shape are obtained. These changes in the blade profiles and blade passages are visible in Figure 96.

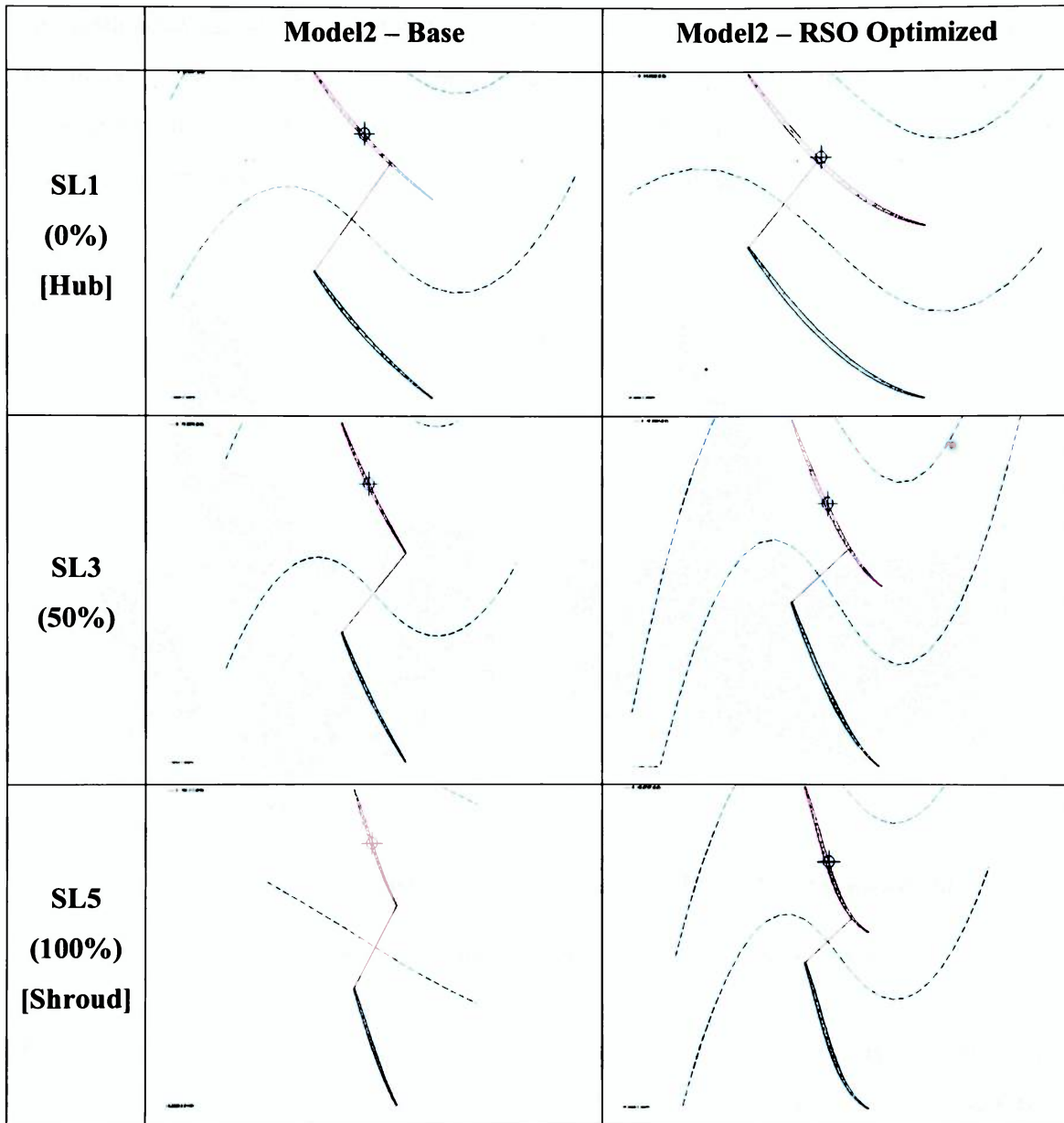


Figure 96: Model2 RSO Results – Blade Profiles/Passages

The plots for the blade profiles at the difference span layers show the improvements made in terms of the camber of the blade profiles. In the base model, the blade profiles are generally flat, showing little or no camber in the airfoil shape. The RSO optimized blade profiles show the improved camber shape that help achieve the 2% increase to total efficiency due to the optimization of the TE shape. The increased camber in the blade

profiles results in the aft translation of the blade passage throat surface. This is similar to the result obtained in the optimized design for model1. This increases the surface area of the blade in contact with the fluid, consequently augmenting the amount of work imparted to the flow. An increase in the LE-TE absolute total pressure rise (δP_{Tabs}) is an indication of the higher work input to the flow by the fan blades.

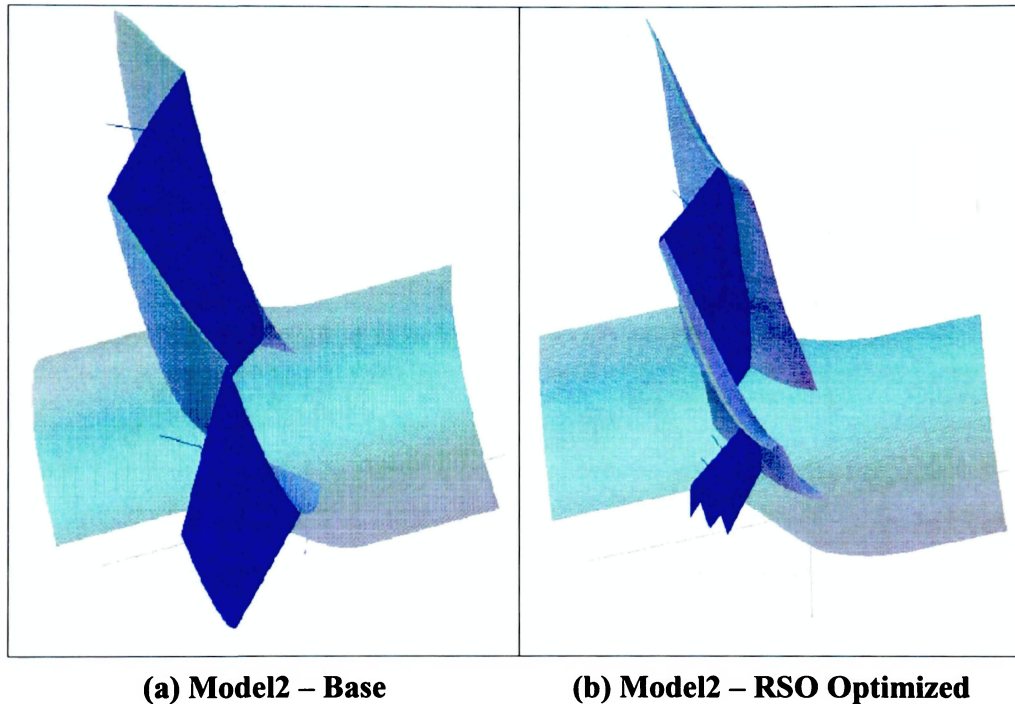


Figure 97: Model2 RSO Results – Blade Passage (with Throat Surface)

The changes in the model2 blade passage are clearly visible in Figure 97. Here, the improved camber in the optimized blade geometry causes the aft translation of the throat surface (shown in blue). The throat surface change is essentially a spanwise interpolation of the blade profiles and throats at the different span layers. Figure 97(b) is a 3D interpolation of the optimization blade profiles observed in Figure 96.

The shape of the leading edge sweep is controlled by the hub-to-shroud blade LE circumferential angle (θ) distribution. The comparison between the model2 base and optimized distributions is shown in Figure 98. The optimized LE sweep distribution displays a *S-shape* similar to the result obtained for in the model1 optimized design.

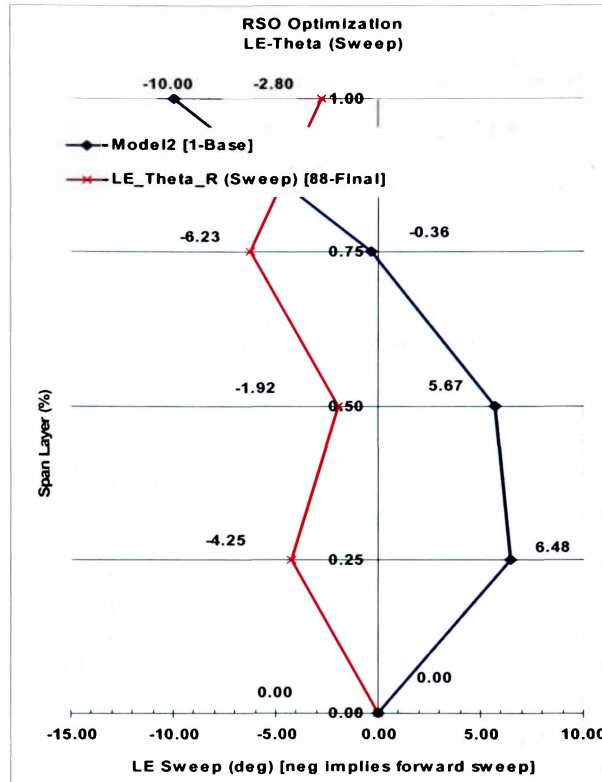


Figure 98: Model2 RSO Results – LE Sweep (θ) Distribution

The optimization of the rotational rate (RPM) and number of blades for model2 yields a considerable increase in the total efficiency (more than two-thirds of the total increase). As in model1, the final rotational rate is lower than that of the base design. The decrease in the rotational rate of the fan is evidence of the optimizer trying to reduce the aerodynamic loading on the fan blades. A reduction in aerodynamic loading may cause a decrease in the amount of momentum imparted to the flow (manifested as less pressure rise). However, the optimization of the blade geometry allows for a significant reduction on the RPM while maintaining significant work input to the flow.

From the history plot of the rotational rate (Figure 99), the optimized RPM value of 1287 is significantly greater than the final/optimized value (972) obtained for base model1. This discrepancy is resolved when the difference in the fan diameter of the two models is considered. Model2 possesses a smaller fan diameter (0.617m) in contrast to model1,

which has a fan diameter of 0.762m. The target function plot for model2 (Figure 93) also shows an efficiency of approximately 88% for the optimal design in contrast to the 85% efficiency obtained for model1. This higher performance achieved for model2 relative to model1 causes the increase in the final RPM value. The smaller model2 must spin slightly faster to achieve better performance than the bigger model1.

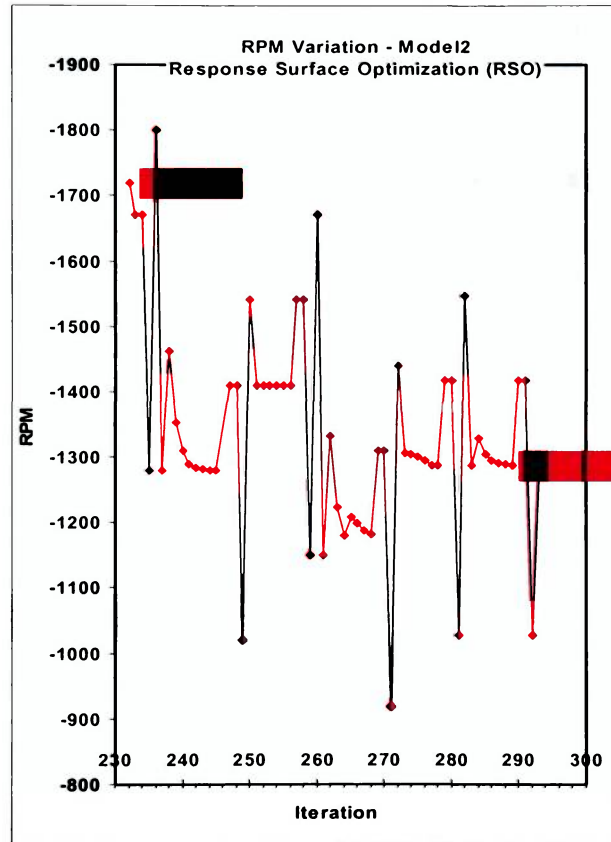


Figure 99: Model2 RSO Results – RPM Optimization History

The effort of the optimizer to further reduce the aerodynamic loading per blade is also seen in considering the optimization of the number of blades in the fan. Figure 100 shows the RSO optimizer almost doubling the number of blades in the cascade from 8 to 15 in the optimized design. There are two primary consequences for this and they both help to increase the efficiency of the fan design. The first is that the size of the blade passage is reduced and this aids the fan blades in turning the flow more effectively. The second

effect of increasing the number of fan blades is also to decrease the aerodynamic loading per blade, essentially the amount of work the blade has to expend in turning the flow through the blade passage.

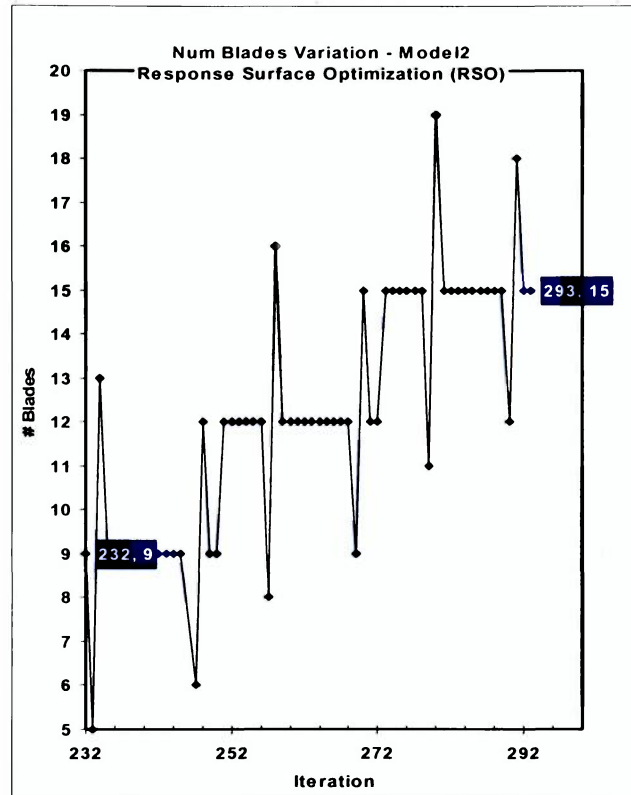


Figure 100: Model2 RSO Results – # Blades Optimization History

The cumulative result of the modifications to the base model2 geometry is examined from a study of CFD results comparing the base and final designs. Of primary importance is if the CFD results show the expected increase in the total pressure in the absolute frame as a result of the optimized blade throat geometry and the resulting efficient turning of the flow. Another expected result is a larger static pressure rise in the optimized blade geometry also due to the change in the blade passage geometry.

As is the case for model1, CFD results are computed for the base and optimized designs using a higher discretization (advection blend =1) scheme in the CFX-Bladegen(Plus) solver. This takes advantage of the higher accuracy usually inherent in higher order

numerical schemes. It is noteworthy to recall that within the context of the MDO optimization analyses, the order of the CFX-Bladegen(Plus) solver discretization scheme is slightly lower (advection blend = 0.88) for stability and robustness of the MDO analyses CFD calculation component.

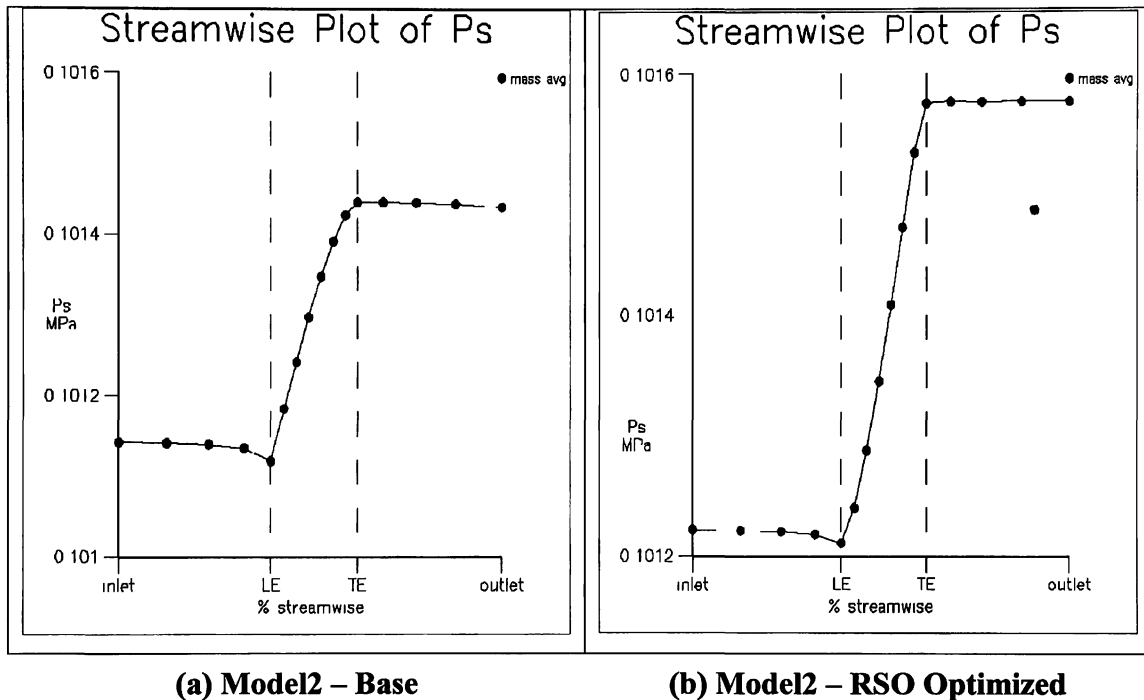


Figure 101: Model2 RSO Results – Static Pressure, P_s (Mass Avg.)

The plots of the CFD computed mass averaged static pressure distribution shown in Figure 101 show the considerable increase in static pressure rise obtained in the optimized fan blade design. The spanwise blade loading plots show the increases in the difference in static pressure between the pressure and suction surfaces, particularly at the region of the blade close to the tip (shroud) of the blade.

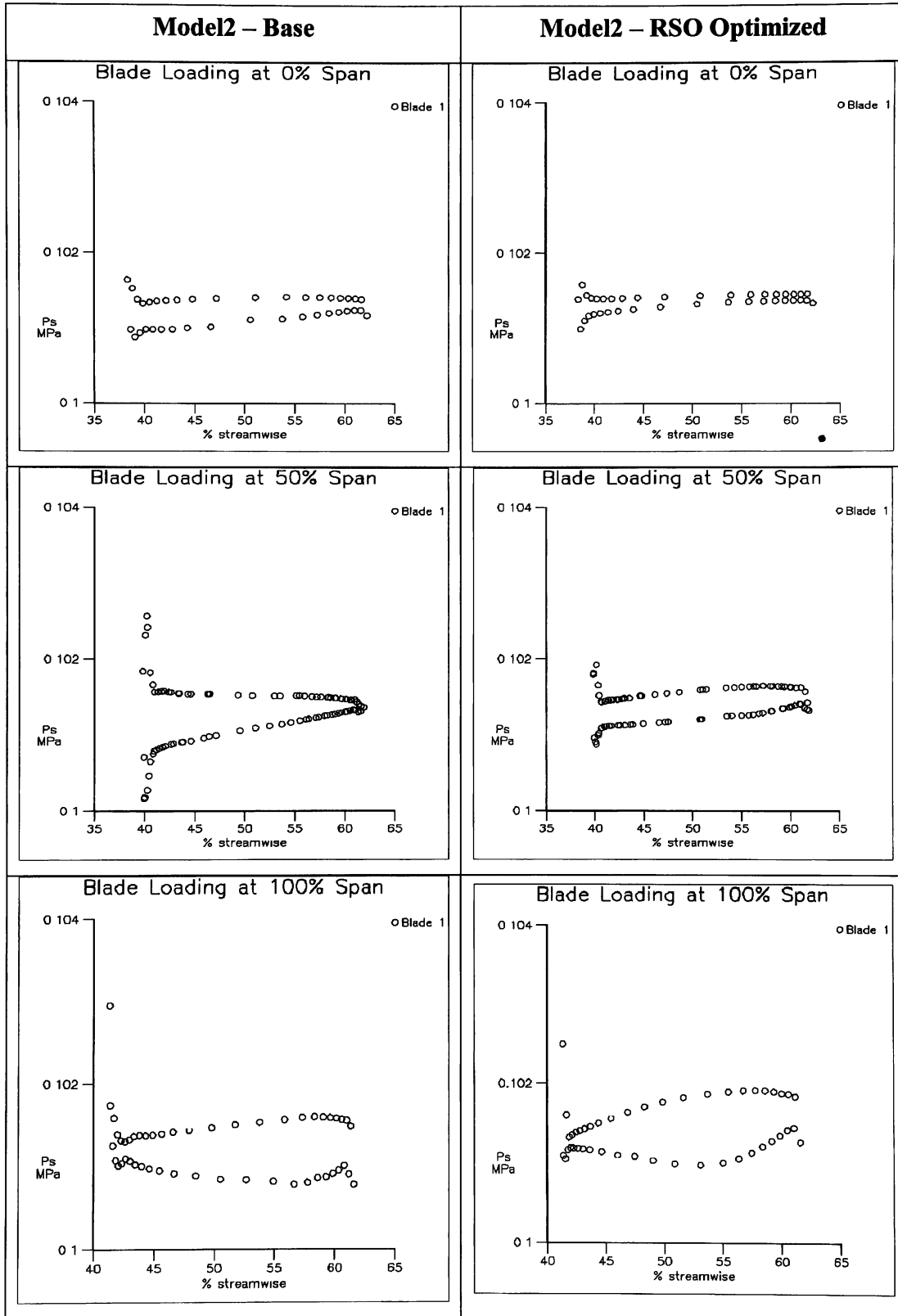


Figure 102: Model2 RSO Results – Blade Loading (by Span Layer)

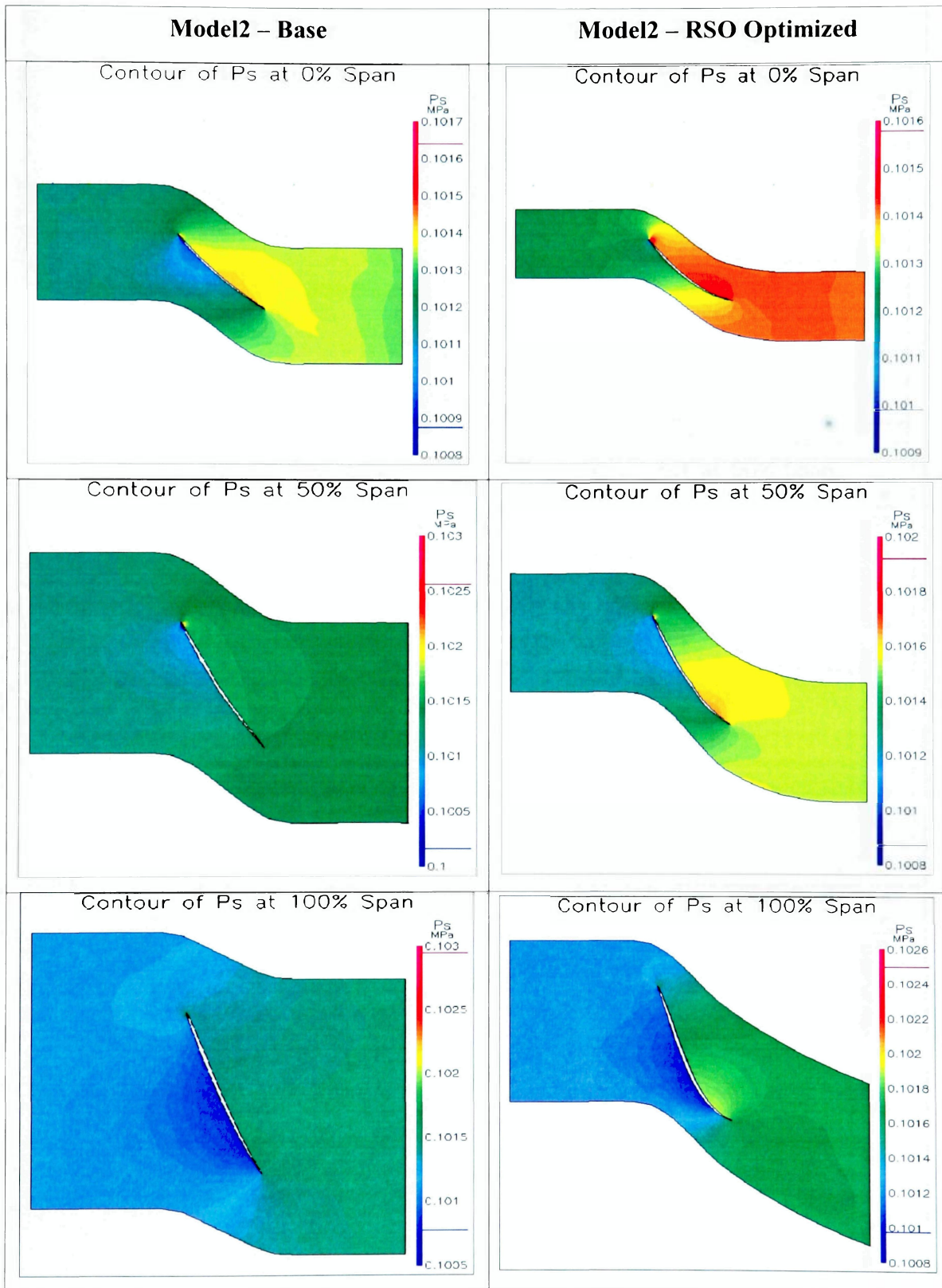


Figure 103: Model2 RSO Results – Static Pressure (P_s) Contours

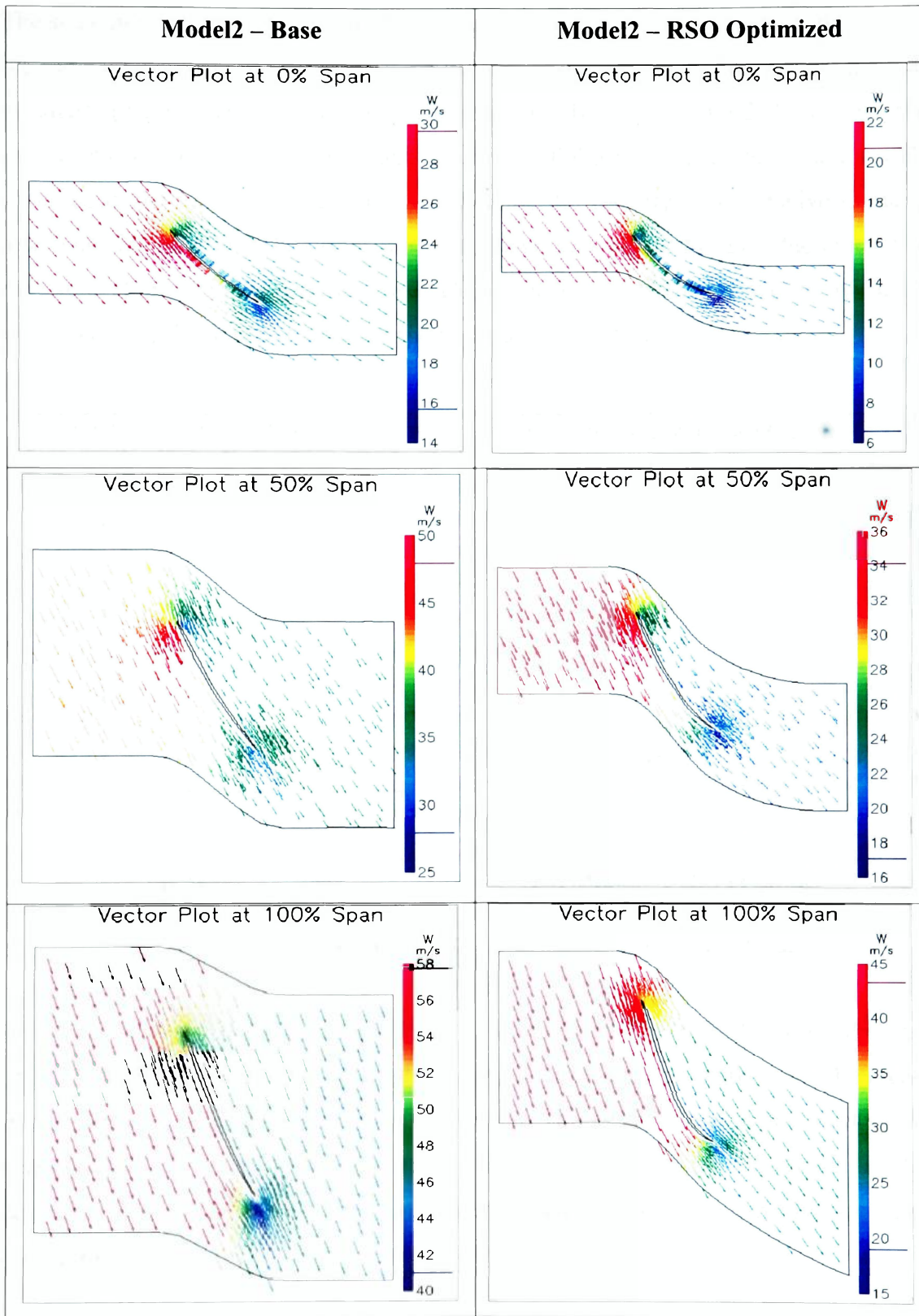


Figure 104: Model2 RSO Results – Relative Velocity (W) Vector Plots

The static pressure contour plots of the base and optimized fan design shows the gains in the static pressure rise in the base model2 and optimized fan design. Most of the increase in static pressure rise is obtained close to the hub and shroud locations. The improvements in the hub and shroud locations of the blade geometries are readily observed in the span layer vector plots colored by the velocity in the relative frame of reference (Figure 104). The vector plots also serve as a means to judge the amount of deviation at the trailing edge of the blade, an indicator of the magnitude of losses experienced by the flow as it passes through the blade passage.

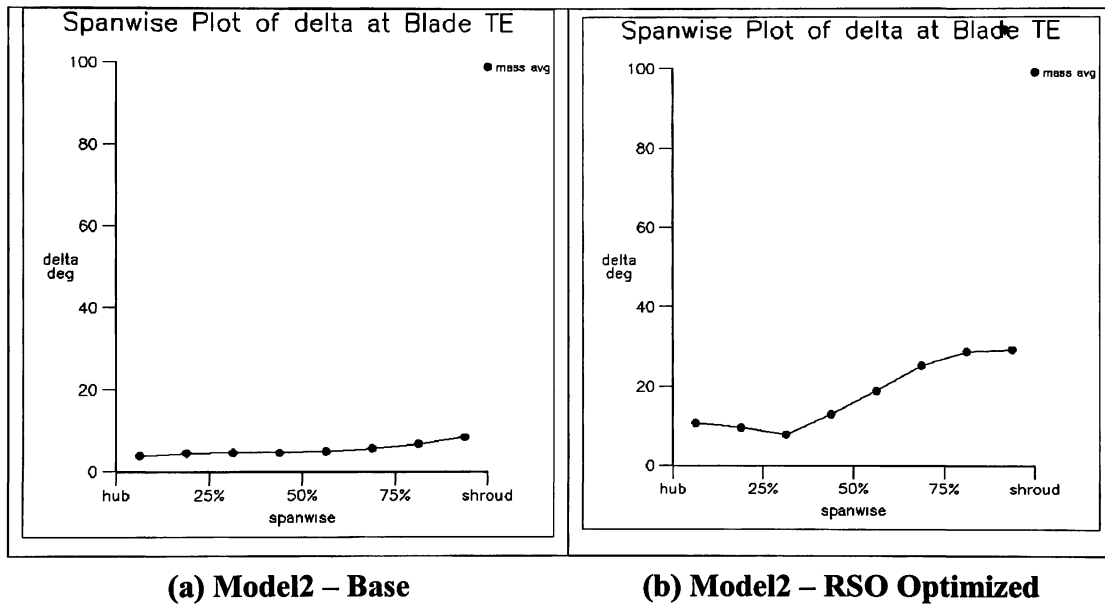


Figure 105: Model2 RSO Results – TE Deviation (delta) Angle Distribution

The mass averaged deviation angle (delta) at the trailing edge of the blade for the base and optimized model2 designs are plotted below. They show a generally marked increase in the deviation angles in the optimized design particularly close to the shroud. An examination of the shape of the geometry of blade passages in Figure 96 reveals that the optimizer attempts to increase the camber in the blade profiles to achieve the increased efficiency.

However, the manner in which the camber shape is modified is somewhat unconventional as the camber shape is excessively modified by the optimizer closer to the TE edge of the blade profile. This leaves the rest of the blade profile with relatively unaltered geometries. This effect is most significant toward the tip of the blade and is clearly visible in the optimized blade profile at the tip span layer in Figure 96.

As the flow travels over the blade towards the TE, the inertia of the flow cannot be dampened enough to allow it to completely stick with the blade through the *sharp turn* close to the TE of the blade. This sharp turn is caused by the awkward shape of the TE of the optimized blade geometry. The consequence is the marked increase in the flow deviation from the trailing edge reflected in Figure 105. It is interesting to note that even with the significant TE deviation in the optimized blade geometry, the static pressure rise and efficiency of the final design still exceed that of the base design.

The reasons for the increases in the static pressure rise and efficiency of the final model2 design can be seen in the relative velocity vector plots of Figure 104. In comparing the vector plots of the optimized design with those of the base model2 design, the success of the optimized blade geometry is turning the flow more than the base model is evident in the direction of the velocity vectors. The effect of turning the flow is that more work is imparted to the flow and that leads to a rise in the total pressure in the absolute frame. The optimized blade is able to exceed the base model2 efficiency, partially because it is able to input more work into the flow even though there are more losses as a result of the awkward TE shape.

As previously discussed, the other important factor influencing the efficiency of the fan blade is the change in the total pressure in the relative frame of the blade (∂P_{Trel}). Ideally, an optimal design should display a minimal relative total pressure change, coupled with a high absolute total pressure change (∂P_{Tabs}). These two conditions ensure a minimal total efficiency as defined by equation 3.8. Comparison plots for the base and optimized model2 designs are shown in Figure 106.

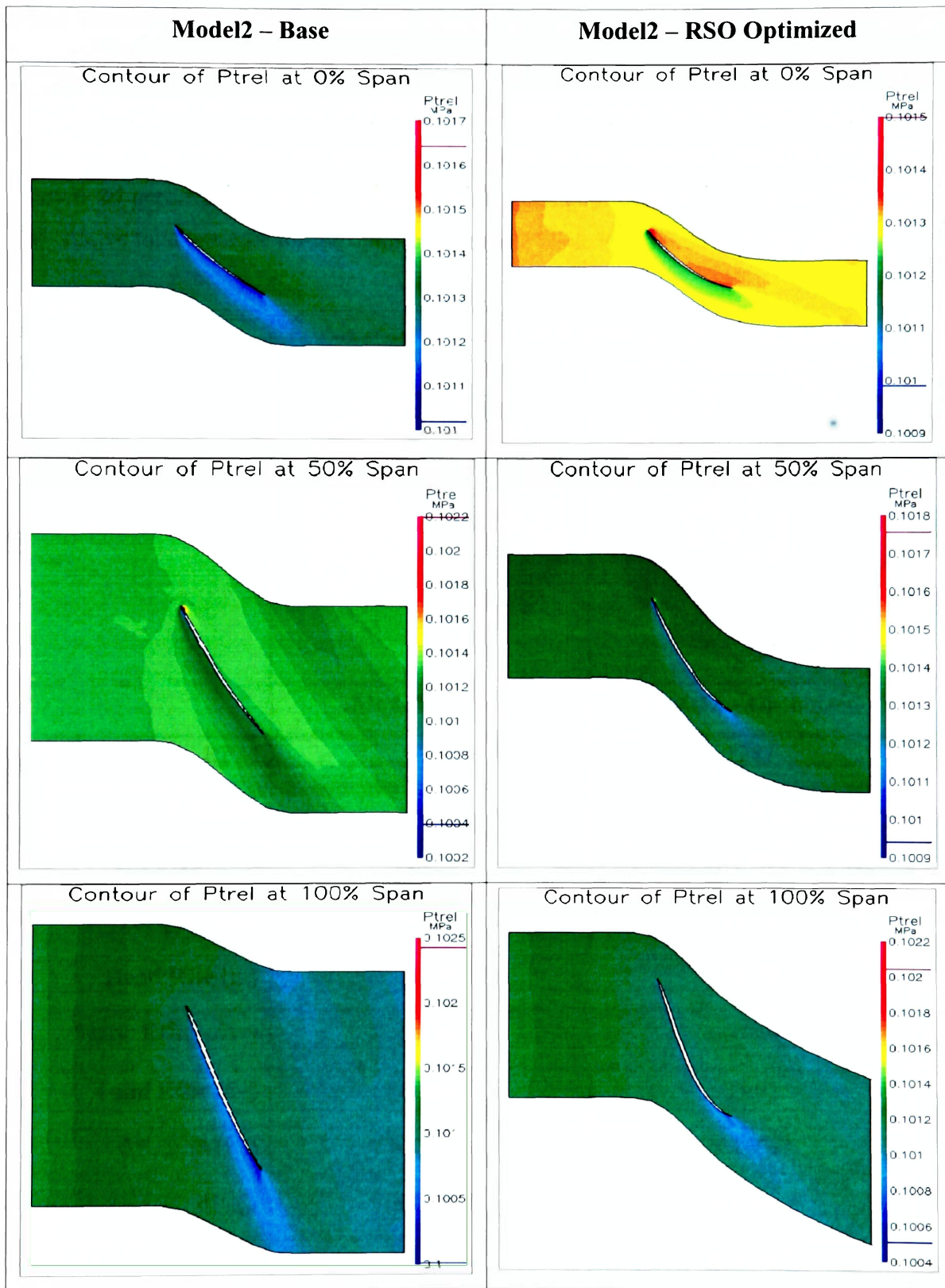


Figure 106: Model2 RSO Results – Relative Total Pressure (P_{Trel}) Contours

The relative total pressure contours shows the improvements from the base to the optimized model2 fan designs. In Figure 106 the contours are plotted using local scales so as to capture the range of relative total pressure variation in each span layer. These ranges of P_{Trel} variation are generally decreased in the optimized model2 blade. The improved performance of the optimized blade geometry including the decreased change in relative total pressure for the optimized model2 blade is captured in Figure 107.

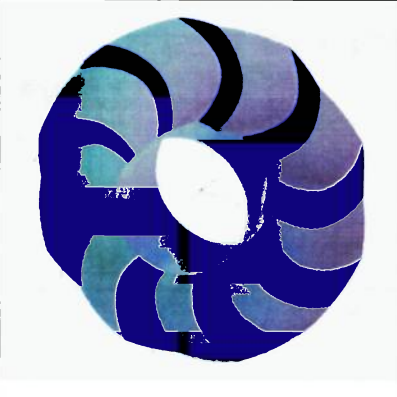
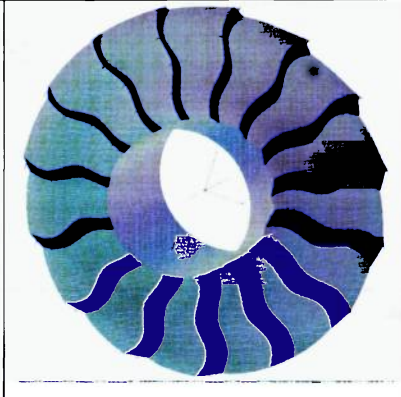
Algorithm: Response Surface Optimization (RSO)		
		
	Model2 – Base	Model2 – Optimized
RPM	1720	1287
# Blades	9	15
VFR (m³/s)	4.400	3.30
Torque (Nm)	10.49	13.41
Head Rise (m)	30.47	42.59
Static Effic. (ϵ_{ST})	0.726	0.660
Total Effic. (ϵ_T)	0.824	0.902
δP_{Tabs} (MPa)	$0.353e^{-3}$	$0.495e^{-3}$
dP_{Trel}	$-0.076e^{-3}$	$-0.054e^{-3}$
dP_{st}	$0.312e^{-3}$	$0.362e^{-3}$

Figure 107: Model2 RSO Results – Fan Design CFD Analysis Summary

4.2.3 RSO Results – Model3

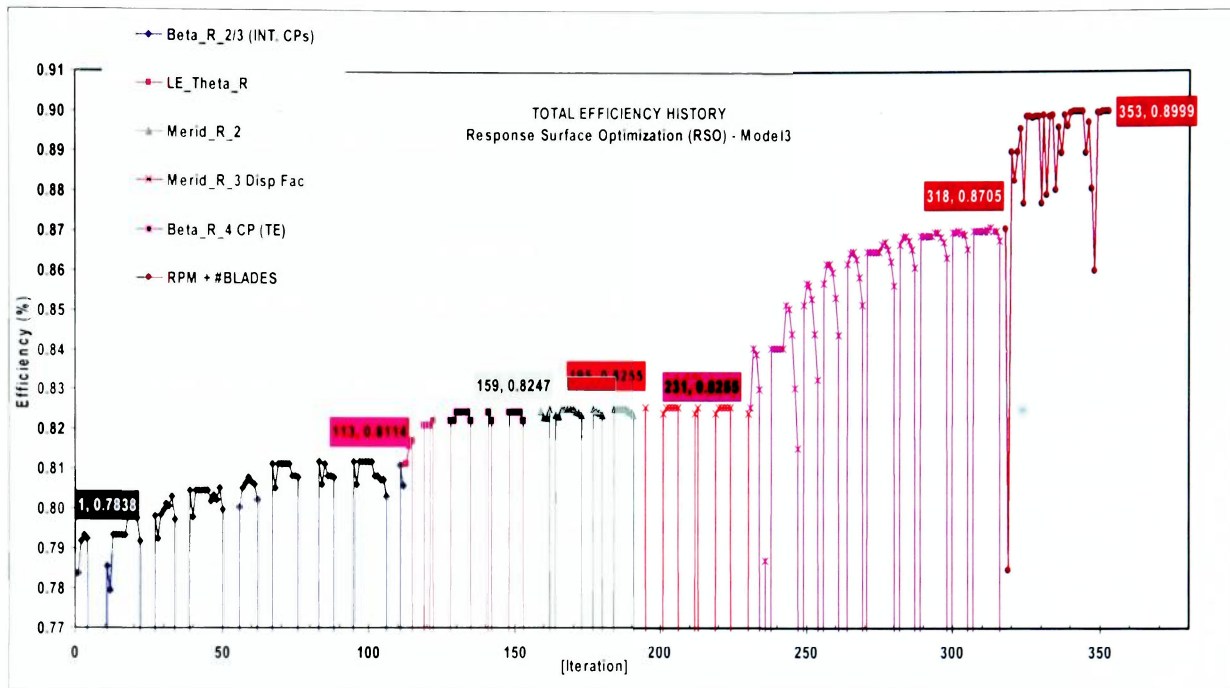


Figure 108: Model3 RSO Results – Target Function (Total Efficiency)

The target function plot for model3 above shows a gain of 11% in total efficiency. The largest increase in total efficiency is obtained from the variation of the angular coordinate of the TE bezier CPs (β_{r4}). The total efficiency gain from the TE bezier CP design variables represents two-thirds (66%) of the total gain in efficiency in the optimized model3 design. As is the case in the previous two base models, significant gains in efficiency are also obtained from the optimization of the interior bezier CPs, circumferential LE Sweep, as well as the number of blades and RPM of the fan.

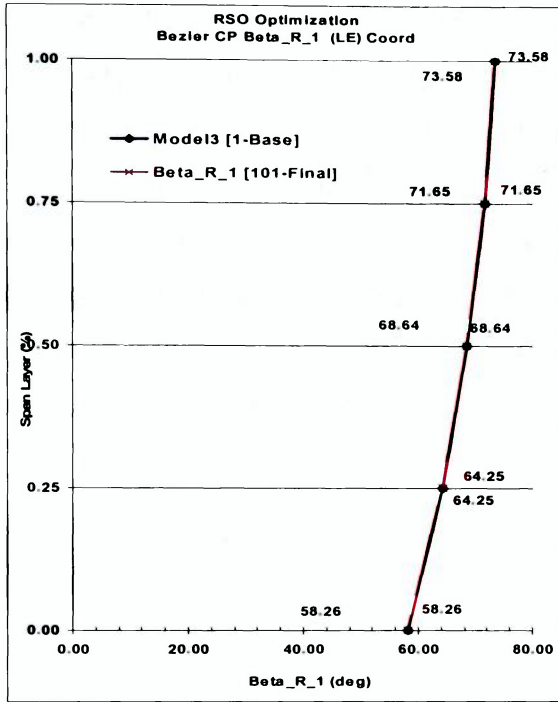
The meridional design variables ($M'_{r|s}$) yield almost no improvement in the final optimized design. This lends further support to the hypothesis that the blade geometry is generally insensitive to these design variables. As is the case for the previous optimized models, the contribution of these meridional design variables to the increase in total efficiency is minimal at best. Such minute improvements indicate that future optimization analyses can ignore these design variables or make them constant. This will save on the

computational cost of running an optimization analyses relative to the meridional design variables. The insensitivity of the design to these parameters may also be indicative inherent weaknesses in the adopted parameterization scheme that require further investigation in future MDO studies.

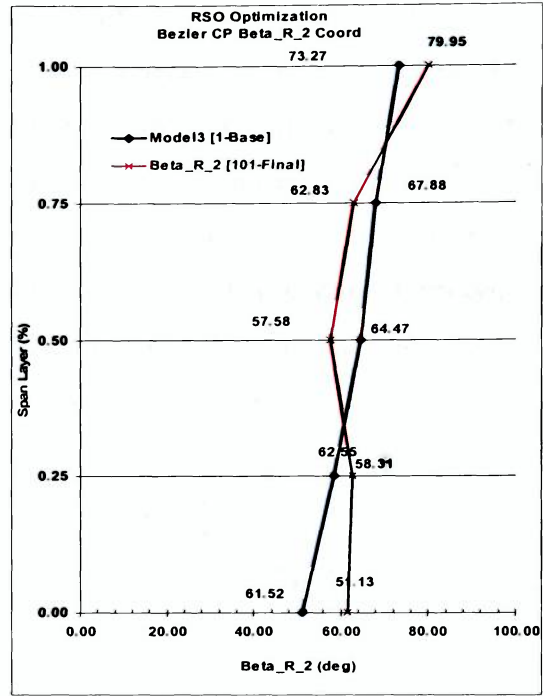
Table 20: Model3 RSO Results - Influence of Design Variables on Target Function

Order	Variables (Unless Otherwise Specified R = 1..5)	Total # Vars.	Gain in Total Effic. (%)
1	Tangential Coord. of Interior Bezier CPs ($\beta_{r 2}, \beta_{r 3}$)	10	2.76
2	Circumferential LE sweep location (θ_r)	5	1.33
3	Meridional coordinate of CP2 ($M'_{r 2}$)	5	0.08
4	<i>Displacement factor</i> for CP3 meridional coordinate (δ_r)	5	0.00
5	Tangential coordinate of TE bezier control points ($\beta_{r 4}$)	5	4.50
7	RPM & Num of Blades	2	2.94
Model3 Total Efficiency Gain (%)			11.61

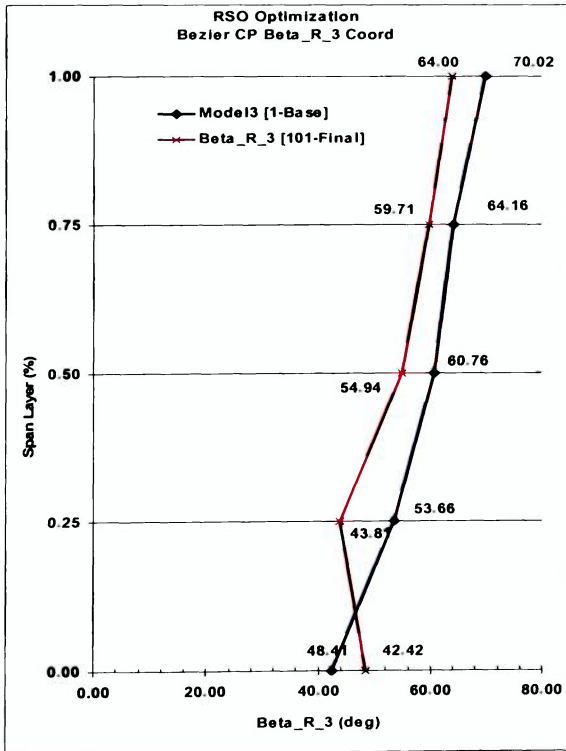
The modifications to the bezier CP tangential coordinates in the optimized design are shown in the spanwise plots of Figure 109. In keeping the LE bezier CPs constant for previously discussed reasons (maintaining velocity and radial equilibrium conditions at the blade LE), the base and optimized plots of Figure 109(a) remain the same. The other bezier CPs are modified by the RSO optimizer to obtain optimal configurations, with the largest variations occurring at the blade TE bezier CPs. The effect of the large modification to the blade TE on the total efficiency is immediately obvious from the target function plot of Figure 108 and the summarized results in the table above. As seen in the previous results, the geometric effect of changing the angular coordinates of the bezier Cps is a modification in the camber shape of the blade profiles at each span layer.



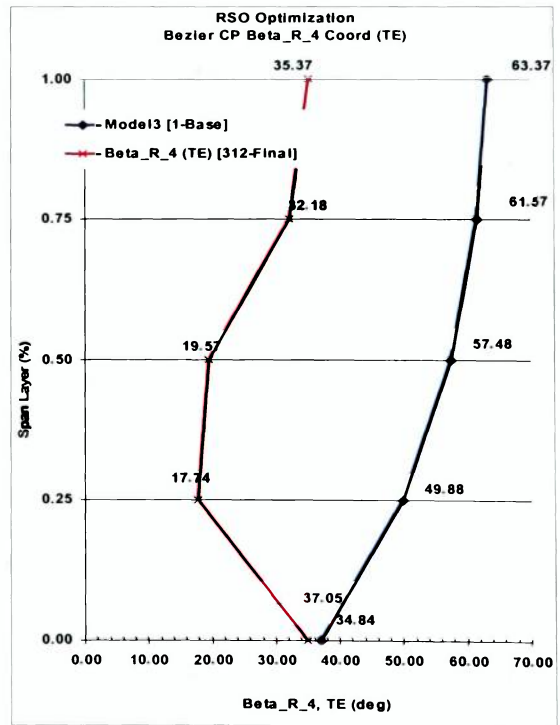
(a)



(b)



(c)



(d)

Figure 109: Model3 RSO Results – Bezier CP Angular (β_{rs}) Coordinates

The modification of the tangential coordinates of the bezier CPs accounts for a 7.3% increase in the total efficiency of the fan design (a 2.76% for the interiors bezier CPs and a 4.5% efficiency increase from the optimization of the TE bezier CPs). It is worthwhile to mention that the increase in efficiency from modifying the tangential coordinates of the bezier CPs exceeds the total gains in efficiency of all the other design variables combined (ref. Table 20). This is evidence of the ability of the developed parameterization scheme to properly describe the blade geometry that is to be optimized. Its is also a measure of the sensitivity of the target function to the tangential bezier Cp coordinates as similar result are obtained for the previous base designs.

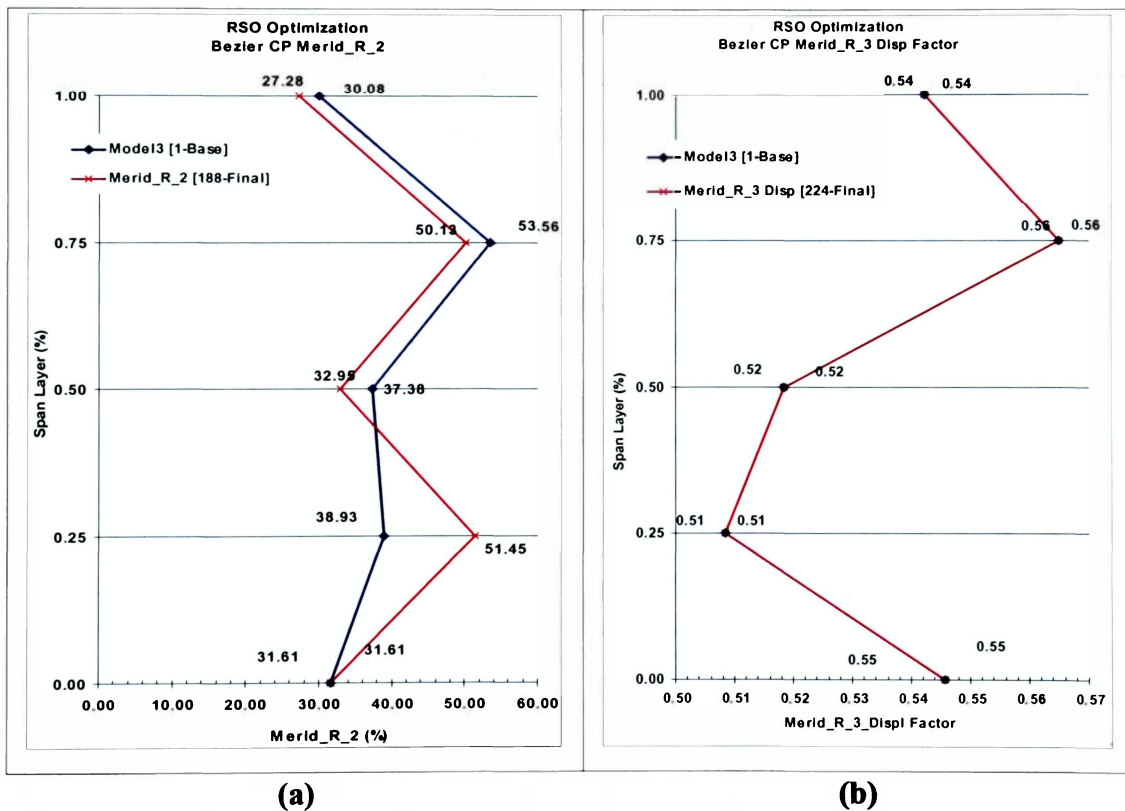


Figure 110: Model3 RSO Results – CP2 Merid Coord (M'_{r2}) and Displ Factor (d)

The changes in the meridional coordinate of the second Bezier CP (M'_{r2}) are shown in Figure 110(a). Only slight modifications in the design variables are made by the RSO optimization algorithm and once again they result in relatively minute gains in total efficiency (less than 1% from Table 20). Figure 110(b) show the base and optimized

values of the *displacement factor* used in equation 3.4 compute the meridional coordinate of the third bezier control point ($M'_{r|3}$). The optimized values at the five span layers are the same as those of the base design. This does not imply that the optimizer fails to investigate the displacement factor design variables. In fact, an analysis of the data shows the perturbation of the spanwise displacement factors (δ_r) at all span layers by the RSO algorithm. However, at all 5 span layers the optimizer returns to the original values for each design variable. This behavior is typical when the target function is relatively insensitive to the design variable in question. We can assume that the spanwise displacement factors in the base model3 design are sufficiently close to optimal values and hence could not be further exploited for the purpose of increasing the target function. However, in the two previous base models, minor gains in efficiency are also obtained relative to the meridional design variables. Thus, we can conclude that the target function is highly insensitive to the meridional coordinates of the bezier CPs.

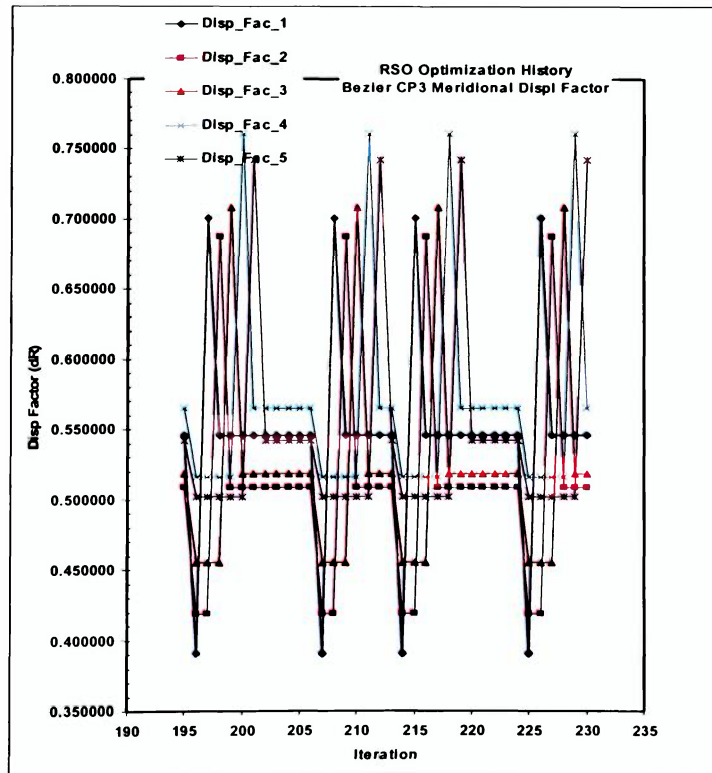


Figure 111: Model3 RSO Results – Displ Factor (δ_r) Perturbation History

The principal effect of varying the meridional and angular coordinates of the bezier CPs is the modification for the blade passage shapes at each span layer. Examining the shape of the blade profiles enables a more complete visualization of the geometric changes from the base to the optimized model3 design. The blade profiles for the hub, shroud and mid-span are presented below for the base and optimized model3 fan designs.

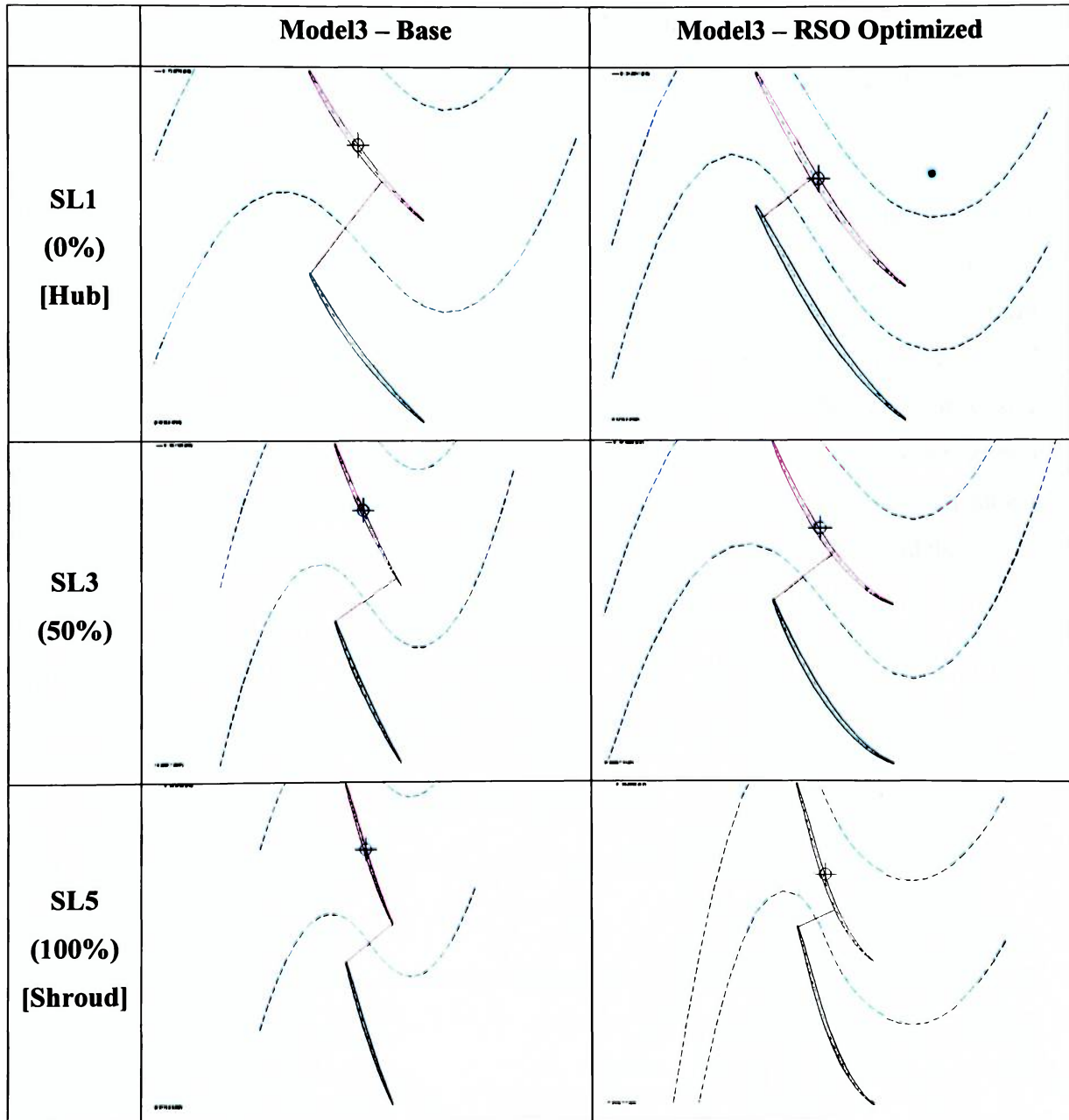


Figure 112: Model3 RSO Results – Blade Profiles/Passages

The reason for the significant increase in total efficiency for the optimized model3 fan is evident when the changes in the blade passage shapes of Figure 112 are examined. The same phenomena seen in the previous fan models are repeated for the model3 optimized design. The change in the location of the throat as it moves aft into the blade passage at each span layer is clearly visible between the original and optimized designs. The improve camber in the optimized model3 design is also readily observed in Figure 112. The blade passage in the optimized design is also narrower than in the base model3 design. This is an indication of an increased number of blades in the complete cascade of the final design.

In the spanwise direction, comparisons of the hub to shroud circumferential LE sweep distributions are shown below. Once again, the optimized LE sweep distribution is an *S-shaped* curve as is the case in the previous base fan designs. The *S-shape* of the LE in the optimized design and changes in the geometry of the blade passage are visible in the 3D blade passage plots of Figure 114. Here, the three dimensional effects of the design variable modifications, particularly to the geometry and location of the throat surface (shown in blue), as immediately visible. The translation of the throat surface aft allows more work to be imparted to the flow as is done in the previous optimized models.

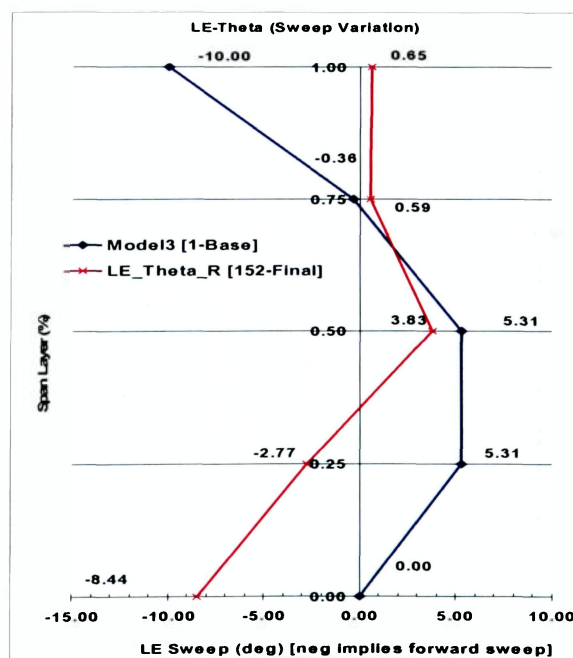
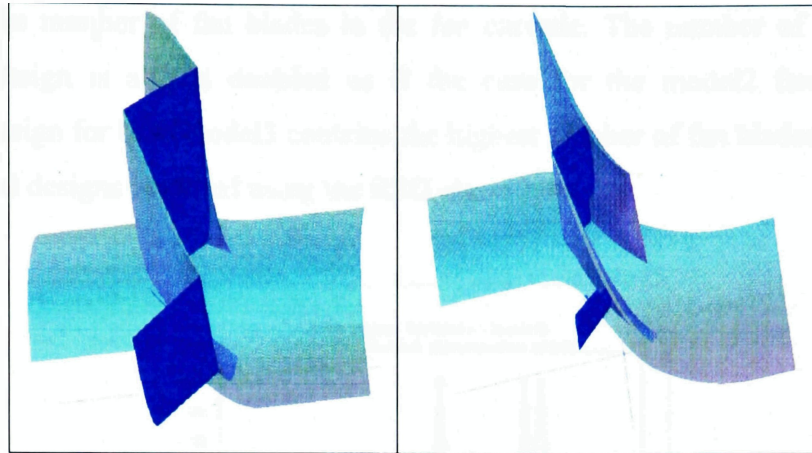


Figure 113: Model3 RSO Results – LE Sweep (θ) Distribution



(a) Model3 – Base

(b) Model3 – RSO Optimized

Figure 114: Model3 RSO Results – Blade Passages (with Throat Surface)

The rotational rate (RPM) optimization history plot for the model3 base design shows the final RPM value to be slightly larger than that of the base design. Note that the negative RPM values are only necessary to indicate the direction of rotation of the fan to the CFX-Bladegen(Plus) flow solver and do not imply magnitude.

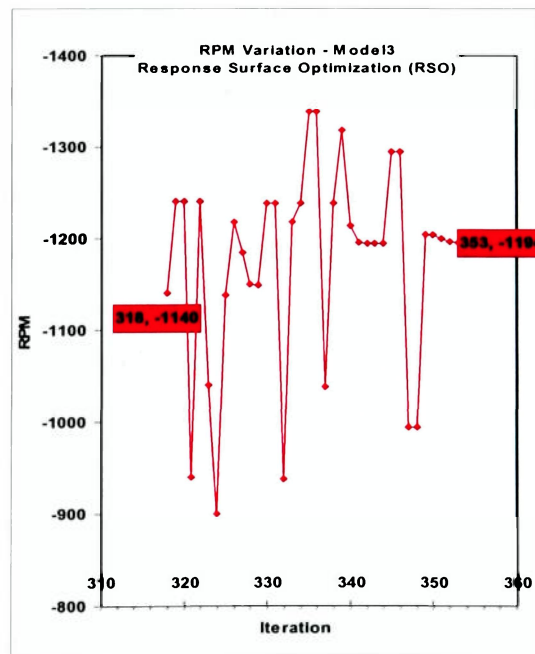


Figure 115: Model3 RSO Results – RPM Optimization History

The optimization history plot of the number of fan blades shows the RSO algorithm increasing the number of fan blades in the fan cascade. The number of blades in the optimized design is almost doubled as is the case for the model2 fan design. The optimized design for base model3 contains the highest number of fan blades (17) in all of the optimized designs obtained using the RSO algorithm.

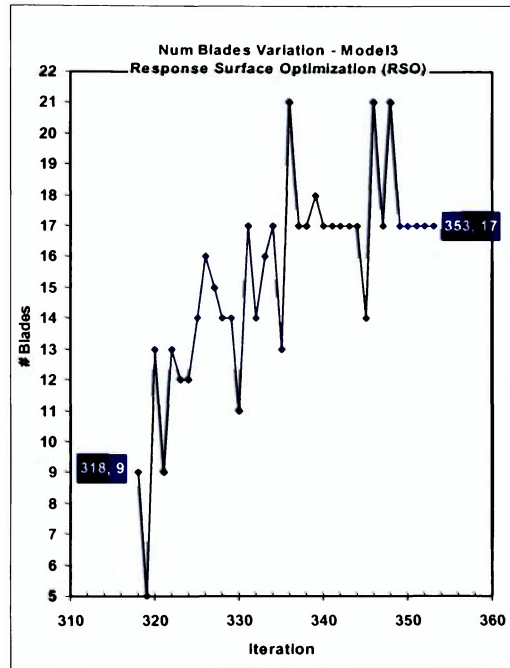


Figure 116: Model3 RSO Results - # Blades Optimization History

The optimizer also increases the solidity of the fan design by increasing the number of blades. The main advantage of the increase solidity being the improved ability of the blade to turn the air mass as it flows through the blade passage. This is accompanied by an increase in the amount of work imparted to the airflow. The increased work input to the flow is expected to yield a higher static pressure rise in the flow as it exits the blade passage. The optimized fan design is achieved by a combination of three factors: increased solidity through an increased number of blades, improving the camber shape of the blade profile and decreasing the aerodynamics loading per blade also through increasing the number of blades in the fan cascade.

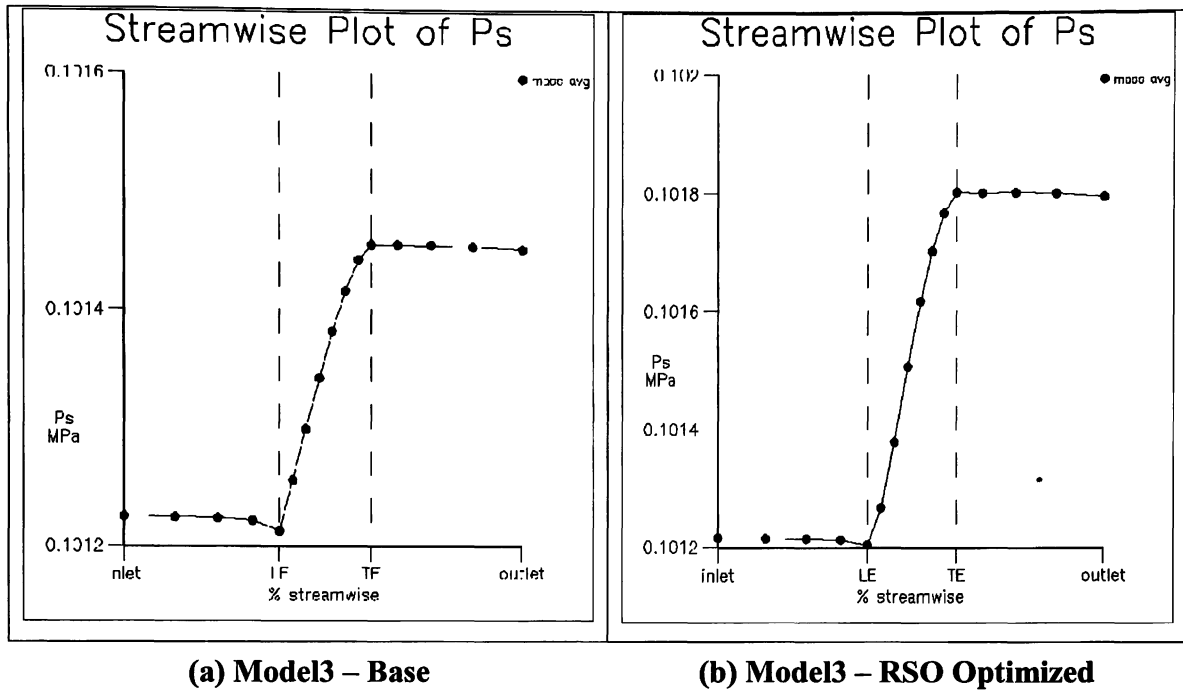


Figure 117: Model3 RSO Results – Static Pressure, P_s (Mass Avg.)

The significant increase in the mass-average static pressure rise through the optimized blade passages can be seen in Figure 117. The mass averaged static pressure is a consolidated look at the improvements made to the fan design by the RSO optimizer. For a more detailed comparison between the base and optimized model3 designs, blade loading plots at three spanwise layers are shown in Figure 118.

The optimized fan design shows a much improved static pressure distribution in the blade loading plots. The static pressure difference between the pressure and suction surfaces of the blade is a measure of how much work is transferred to the fluid mass as it passes through the blade passage. In Figure 118, all the blade loading plots show a higher difference between the pressure and suction surfaces in the optimized fan design. The improvements in the blade loading become more significant in the region of the blade between the mid-span and the shroud, evidenced by the increases in the mid-span (50% span) and shroud (100% span) blade loading plots in Figure 118.

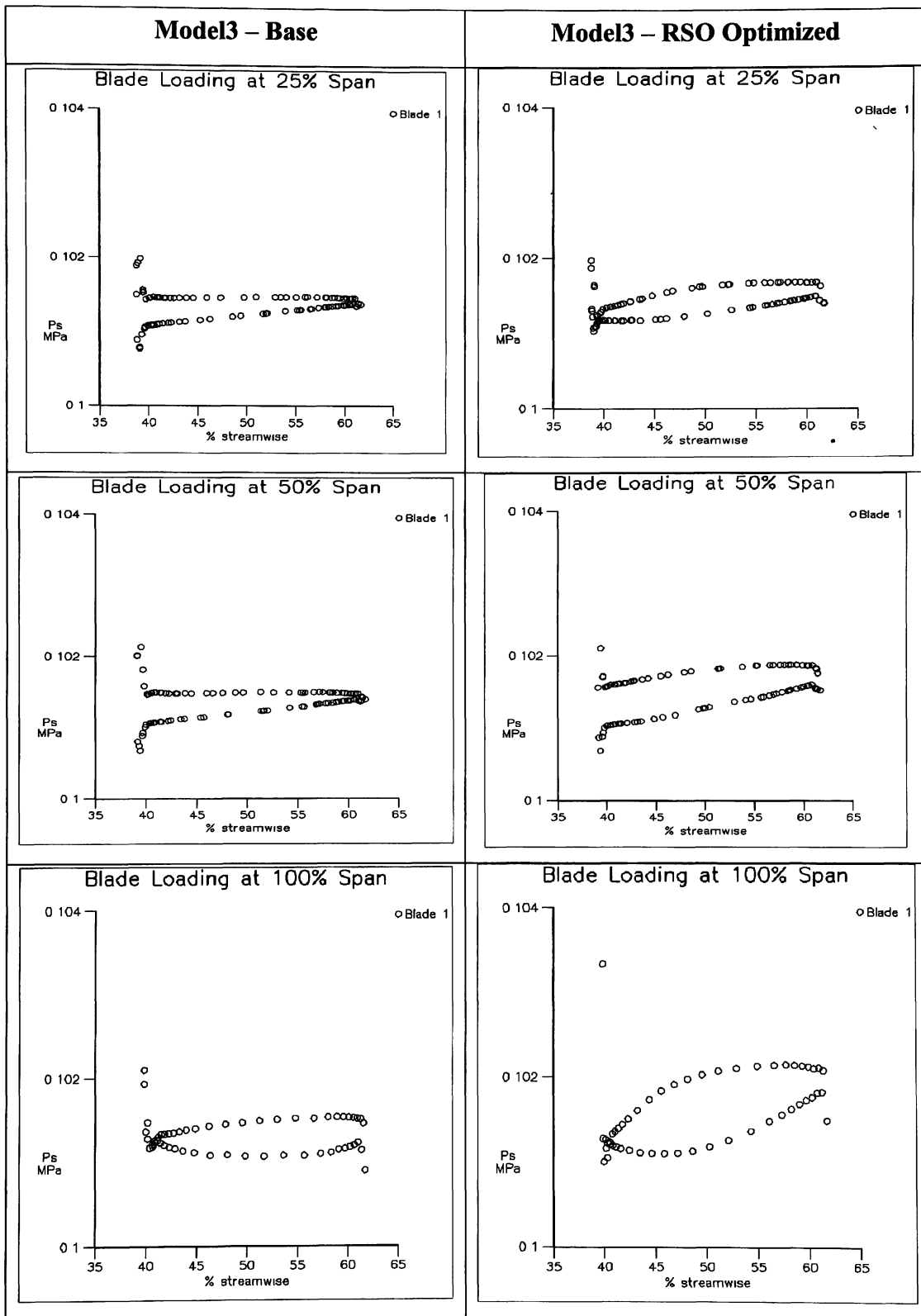


Figure 118: Model3 RSO Results – Blade Loading (by Span Layer)

The contours of the static pressure distribution along the span of the blade allows for a more complete comparison between blade passages in the base and optimized model3 fan designs. These contours are shown in Figure 119 clearly showing the improvements in the fluid flow through the blade passage. The increased static pressure rise in the midspan and shroud static pressure contours is also clearly illustrated.

The higher solidity of the optimized model3 design, as a result of the increased number of fan blades, is the reason the blade passages in the optimized design are generally narrower than those of in the base model. It is important to also point out the considerable change to the geometry of the blade passage resulting from the optimization of the blade geometry design parameters. The shape of the blade passage in the CFX-Bladegen modeler is controlled by the geometry of the blade. As a result, an optimization of blade geometry inherently yields a corresponding optimization of the blade passage shape.

The effect of the improved camber in the optimized design is seen in the vector plots of the fluid velocity in the relative frame shown in Figure 120. The vectors plots are colored by the magnitude of the relative velocity (W), and they show the increased amount of deceleration of the flow that occurs in the optimized fan design. For example, at the mid-span (50%) span layer in Figure 120, the flow is decelerated from 36m/s at the inlet of the fan to approximately 20m/s as it exits the fan, in comparison with a deceleration from 35m/s to 27 m/s in the base model2 fan design, also at the mid-span. The direct consequence of improved deceleration in the flow is an increase in the static pressure from the leading edge to the trailing edge of the blade.

The direction of the flow is also improved from the base to the RSO optimized fan design. In the base design, the vector plots show the flow with very minimal turning at the shroud of the original fan design, signifying low work input at the shroud. The optimized design turns the flow considerably, hence the large jump in the pressure distribution at the shroud location seen in the blade loading plots of Figure 118. As expected, there is an associated drawback to the increased turning of the flow.

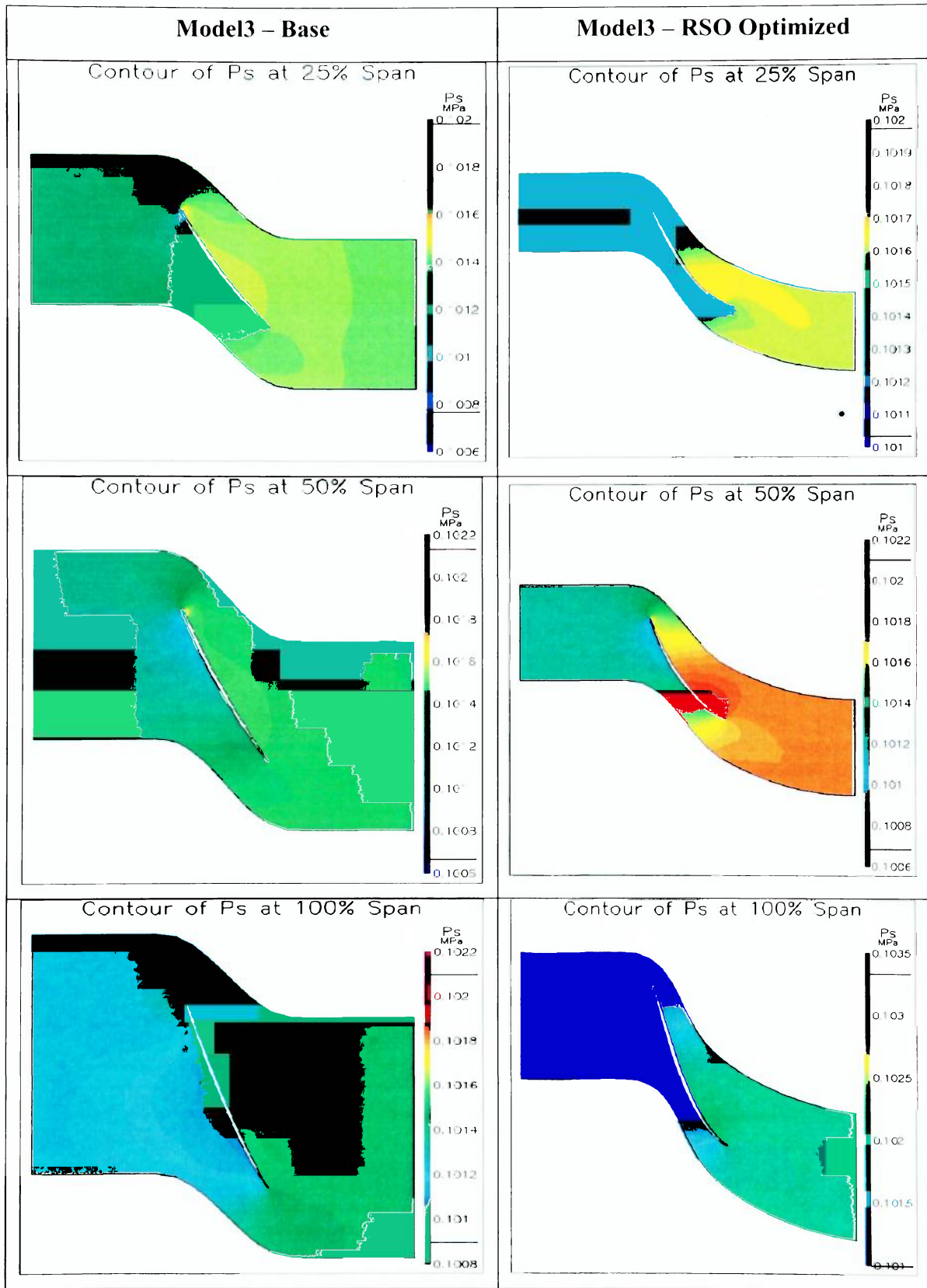


Figure 119: Model3 RSO Results – Static Pressure (P_s) Contours

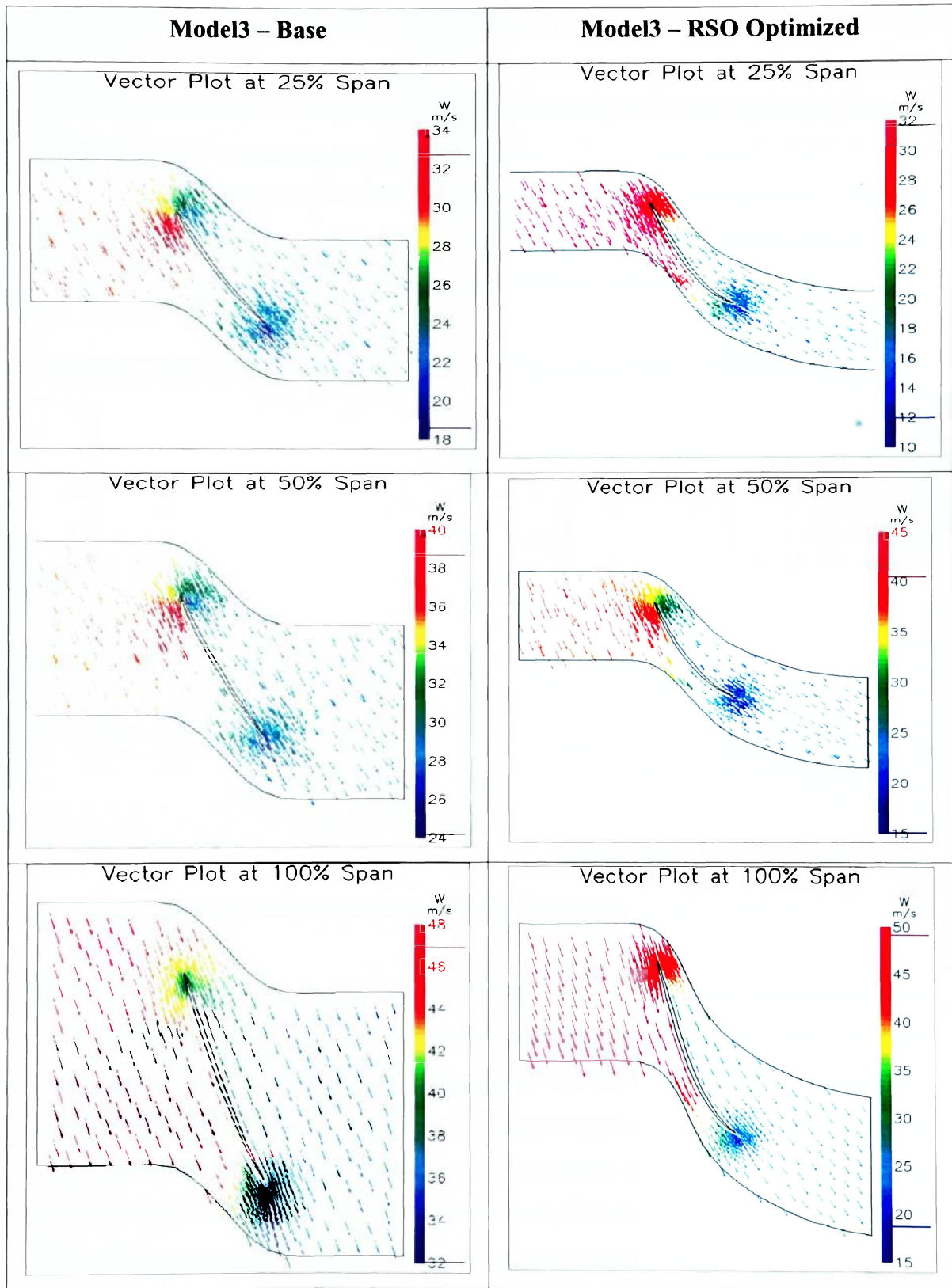


Figure 120: Model3 RSO Results – Blade Rel. Velocity (W) Vector Plots

The increased camber shape in the optimized blade shape causes increased separation of the flow from the blade as it traverses the blade passage. Typically, the inertia resulting from the momentum of the fluid mass causes the difficulty in the flow remaining attached to the blade. This phenomenon is typically observed in compressors where the area normal to the flow increases from the entrance to the exit of the blade passage. Hence, in comparison with the base model3 design, the optimized fan design yields an increase in the mass averaged trailing edge deviation angle (delta) as plotted in Figure 121.

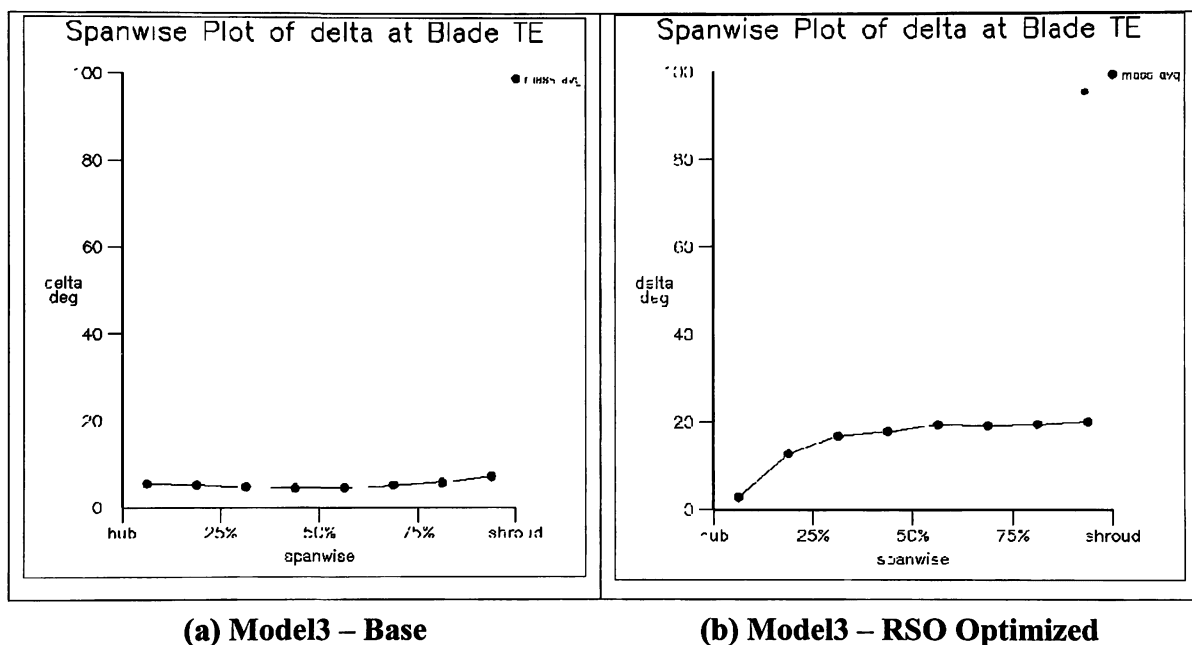


Figure 121: Model3 RSO Results – TE Deviation (delta) Angle Distribution

Ordinarily, the increase in the trailing edge deviation implies a loss in the performance of the blade design. However, the efficiency is calculated relative to the mass averaged change in the relative total pressure (δP_{Trel}) from the LE to the TE of the blade. As a result, the variation in P_{Trel} is considered a critical measure of the performance of the fan design. The change in the relative total pressure is ideally zero but practically, this value is very difficult to achieve. In the benchmark case, the optimized design is expected to display either a minimal δP_{Trel} , or a lower variation relative to that of the base design.

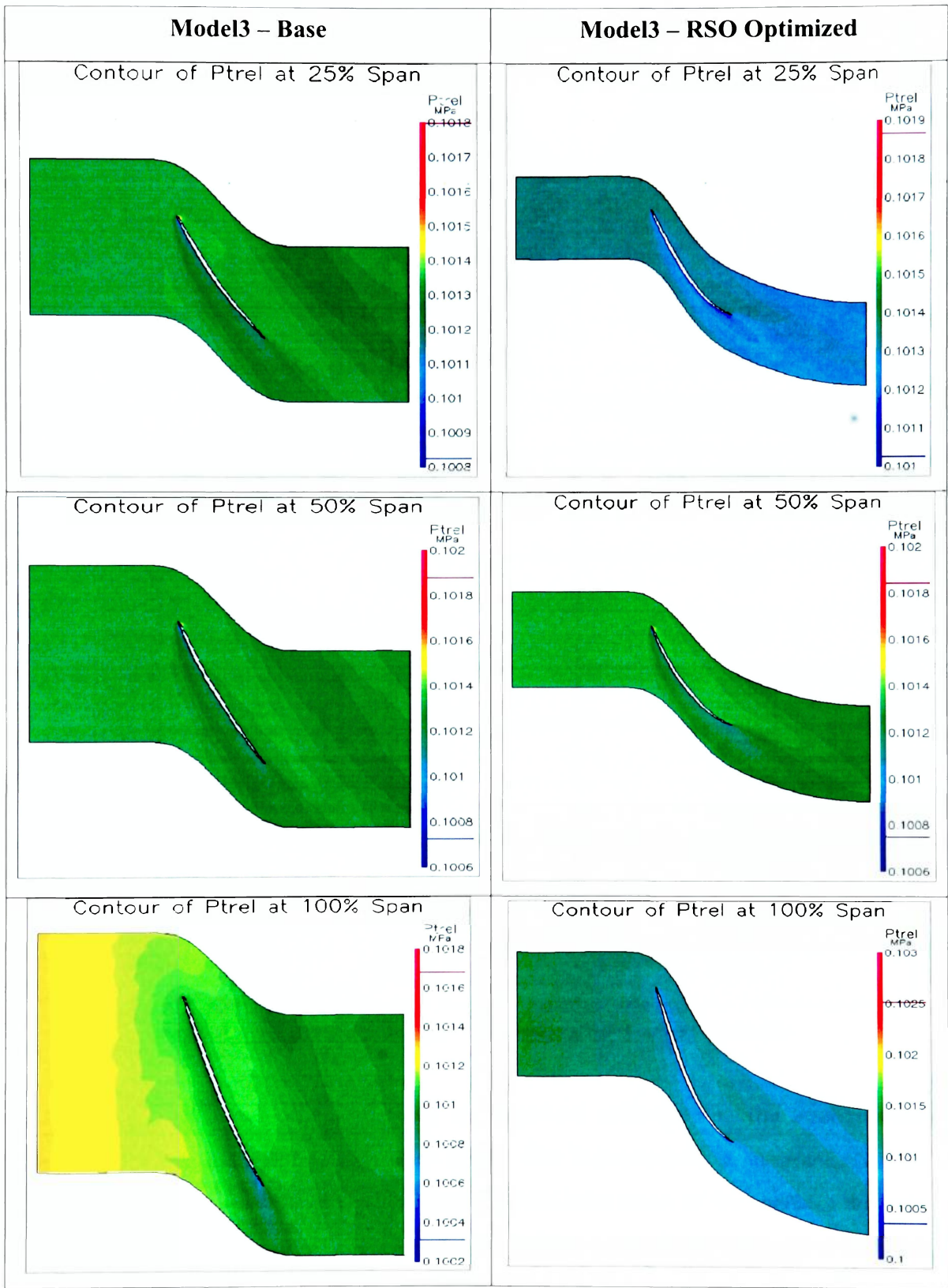


Figure 122: Model3 RSO Results – Relative Total Pressure (P_{Trel}) Contours

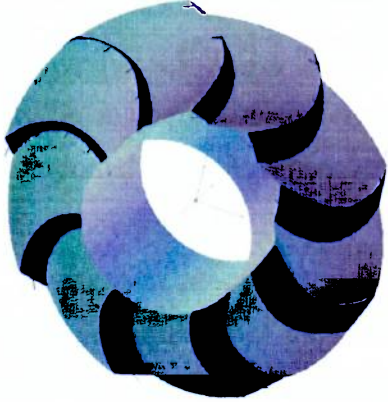

Algorithm: Response Surface Optimization (RSO)		
		
	Model3 – Base	Model3 – Optimized
RPM	1140	1193
# Blades	9	17
VFR (m³/s)	4.74	4.96
Torque (Nm)	13.53	38.61
Head Rise (m)	23.97	76.78
Static Effic. (ϵ_{ST})	0.697	0.608
Total Effic. (ϵ_T)	0.817	0.917
$\delta P_{T_{abs}}$ (MPa)	$0.279e^{-3}$	$0.892e^{-3}$
$dP_{T_{rel}}$	$-0.062e^{-3}$	$-0.080e^{-3}$
dP_{st}	$0.238e^{-3}$	$0.592e^{-3}$

Figure 123: Model3 RSO Results – Fan Design CFD Analysis Summary

The P_{Trel} contours in Figure 122 show an improved uniformity in the relative total pressure distribution at the various span layers. The improvement in the total pressure distribution is most considerable at the shroud (100% span layer) of the blade. In the base model3 design, the relative total pressure contours show a significant variation at the inlet of the fan close to the shroud, which is typical of losses that negatively affect efficiency. In the optimized design, the total pressure contours are more uniform at the shroud, with

less variation in the P_{Trel} distribution at the inlet than was observed in the base model3 design. The P_{Trel} contours at the 25% span layer show some deterioration in the relative total pressure plots for the optimized design. This is primarily the result of the increased camber of the blade profile in the optimized design. The increased camber not only causes more work to be imparted to the flow (positive effect), but it also causes increased deviation from the blade surface which results in relative total pressure losses.

A comparison between the performance of the base and optimized designs as obtained from the CFD analyses of the two models is shown in Figure 123. The effect of the increased number of blades in the optimized design is reflected in the significant increase in the torque required to turn the fan. This is particularly the case considering that the rotational rate of the optimized design is almost equivalent to that of the base model3 design.

In Figure 123, the consequence of the increased amount of work extracted by the fluid mass in the optimized model3 design is seen in the considerable jump in the change in the absolute total pressure (∂P_{Tabs}). This increase in absolute total pressure is primarily responsible for the 10% increase in the efficiency obtained in the final design. The static pressure rise associated with this massive jump in P_{Tabs} is less than required to reflect an increase in the static efficiency of the optimized fan design. In other words, the static pressure rise should be much higher considering the large increase in the absolute total pressure.

4.3 Similarity (Uniqueness) of Optimal RSO Designs



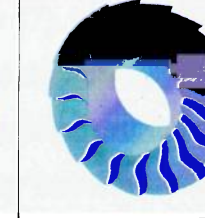
			
	Model1	Model2	Model3
RPM	972	1287	1193
# Blades	12	15	17
VFR (m³/s)	3.616	3.30	4.96
Torque (Nm)	17.85	13.41	38.61
Head Rise (m)	37.66	42.59	76.78
Static Effic. (\mathcal{E}_{ST})	0.598	0.660	0.608
Total Effic. (\mathcal{E}_T)	0.871	0.902	0.917
Total Effic. Gain ($\delta\mathcal{E}_T$)	+ 10%	+7.8%	+10%
δP_{Tabs} (MPa)	$0.438e^{-3}$	$0.495e^{-3}$	$0.892e^{-3}$
dP_{Trel}	$-0.064e^{-3}$	$-0.054e^{-3}$	$-0.080e^{-3}$
dP_{st}	$0.301e^{-3}$	$0.362e^{-3}$	$0.592e^{-3}$

Figure 124: RSO Optimized Fan Design – CFD Based Performance Comparison

A critical characteristic of an effective optimization algorithm is the location of a truly optimal solution within the problem design space. The starting location in the design space should be insignificant and a proper search of the parameterized design space by the MDO algorithm should yield the same optimal result. For the benchmark study, the starting designs are selected to reflect three very different starting locations in the design space. The geometrical properties of the base designs are shown in Figure 74, through Figure 76 and the optimized designs obtained using the RSO algorithm are shown in

Figure 124. It is important to consider that the primary properties which differentiate the three base models are not included as design variables. For example the diameter of the fan is not included as a design variable (ref. Figure 46 and Figure 47), but the diameter of the base model2 fan is smaller (0.617m) than the other two base designs (0.762m). The Hub-Tip Ratio (HTR) of the fan design was also not included as a design variable, but the 0.45 HTR of the base model3 fan is higher than that of the other two fan designs. It is therefore reasonable that the optimization analysis does not automatically converge on the same final design.

The MDO based design optimization methodology is based on three initial designs with different properties. As a result, it is possible to observe the impact of these geometric properties on the convergence of the MDO analysis to an optimal design. Four geometric properties are studied for the purpose of investigating the impact of the geometric differences in the initial designs on the convergence of the benchmark MDO study. This analysis is particularly important given that properties which differentiate the base designs such as the fan diameter and HTR are not included as design variables. The selected properties used in the comparison of the optimal designs are the chord length (c), stagger angle (ξ), pitch-to-chord ratio (s/c) and the solidity (c/s).

Figure 125 shows the distribution of the chord length (c) at each span layer for the base (blue) and optimal designs (red). The smaller diameter of the model2 design causes the smaller chord length across the span of the blade in comparison with the other two base designs. In the optimal designs obtained using the RSO algorithms, the chord length distribution of the model2 design is once again significantly different from that of the other two optimal designs. There is some difference between the model1 and model3 designs at the hub (0%) span, but this is a result of the increased hub diameter in the base model3 design (higher HTR). The convergence of the chord length distribution of the fan blade can be said to be significantly affected by the dissimilarities in the diameter of the fan.

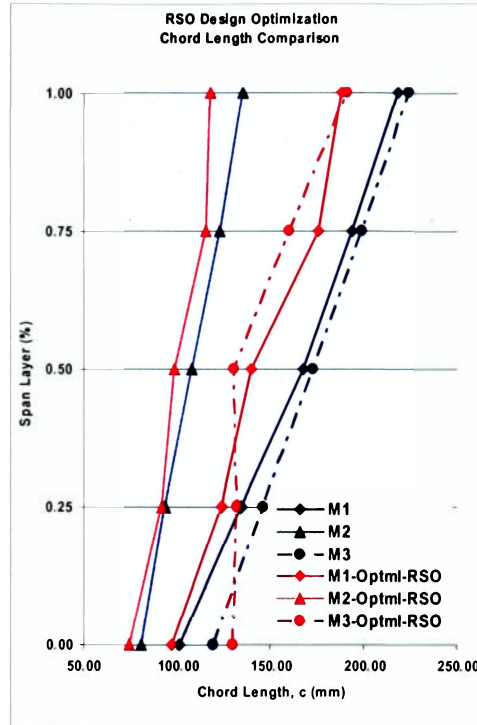


Figure 125: RSO Optimal Designs – Chord Length (c) Comparison

The spanwise stagger angle (ξ) distributions of the base and optimized fan designs are shown in Figure 126. As is the case in the chord length distribution of Figure 125, the model1 and model3 have similar spanwise variations in stagger angle, with a larger difference at the hub (0% span) due to the larger HTR of model3. The stagger angle is related to the angle in which the airfoil encounters the oncoming flow. In the benchmark study, it is measured relative to the **tangential** direction. It is expected to increase towards the shroud due to the increase in relative speed in the radial direction.

The RSO optimized fans show more similarity between the model1 and model2 models in the spanwise variation of the stagger angle. The optimized model3 design shows a significant variation from the other two models but maintains the radially increasing stagger angle variation expected in a feasible fan design. The increased HTR in the base model3 design is responsible for the significant difference between its stagger angle distribution and that of the other two fan designs obtained using the RSO optimization algorithm.

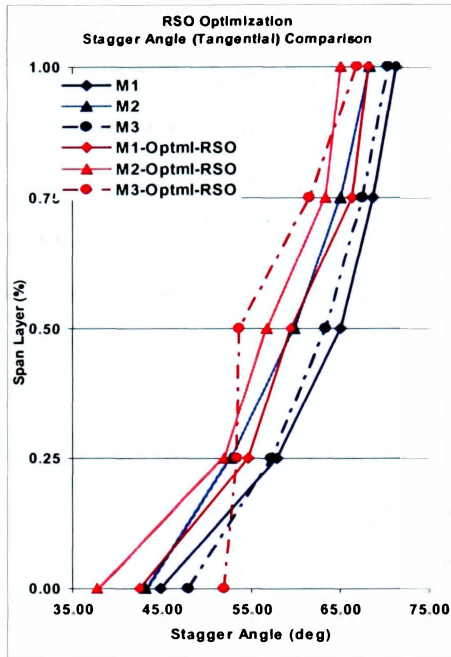


Figure 126: RSO Optimal Designs – Stagger Angle (β) Comparison

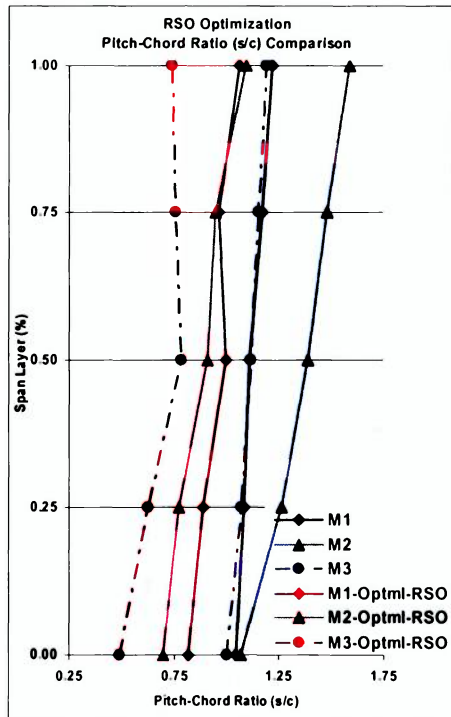


Figure 127: RSO Optimal Designs – Pitch-Chord Ratio (s/c) Comparison

The spanwise variation of the pitch-chord ratio (s/c) is the third geometric property selected to study the convergence of the benchmark MDO study. A comparison of the pitch-chord ratio distributions of the base (blue) and optimal (red) designs are shown in Figure 127. As seen in the previously discussed geometric properties, the spanwise distributions of the model1 and model3 base designs are similar. Slight variations in the base model1 and model3 designs are present close to the hub (0% span layer) due to the difference in the HTR between the two models.

Although the pitch-chord ratio (PCR) distributions for the base model1 and base model3 designs are alike, it is the PCR distributions for the optimized model1 and model2 that are similar. The agreement in Pitch-Chord ratio distributions between the model1 and model2 designs starts to decrease towards the hub (0% span); a characteristic that can also be attributed to the higher HTR of the base model3 design. Note the similarity in PCR between model1 and model2 is achieved using a different number of blades. A total of 12 blades are used for the optimized model1 and 15 blades for the optimized model2 design, compensating for the smaller chord lengths in the optimized model2 design (see Figure 125)

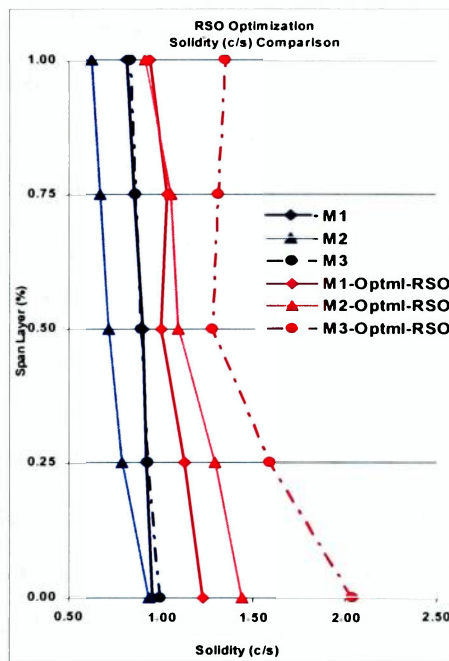


Figure 128: RSO Optimal Designs – Solidity (c/s) Comparison

The solidity is the fourth property used to compare the similarity of the optimal designs obtained using the MDO design methodology. It is calculated as the inverse of the pitch-chord ratio (c/s). Hence it is essentially another way quantifying the size of the fan blades as is done using the pitch-chord ratio. Figure 128 show the radial distribution of the base and optimized fan designs for the benchmark study. The similarities in the base model1 and base model3 designs (same outer diameter) are illustrated. Since the solidity is an inverse of the Pitch-Chord Ratio, the solidity distributions are also similar for the optimized model1 and model2 designs as expected, with increased variation towards the hub (0% span).

The lack of a unique optimal solution for the MDO benchmark fan design study is primarily a consequence of two factors. The first issue is that the RSO algorithm is typically less effective in performing a truly global search over the design space when compared to the evolutionary type optimization algorithms such as the PSO. The second factor is that the properties which distinguish the various base designs are not included as design optimization variables which can be changed by the optimizer.

However, the results obtained allow for certain conclusions on how the various geometric properties influence the optimization analysis for the benchmark case. An immediate conclusion is that the diameter of the fan is a highly influential factor on the convergence of the MDO based design methodology to a unique optimal solution. Variations in the diameters of starting bases designs negatively affect the convergence of design variables that control the chord length (c), Pitch-Chord ratio (s/c) and the Solidity (c/s) of the optimal designs.

The Hub-Tip ratio also negatively affects the convergence of the design optimization analyses. However, the effect is of a lesser extent compared to the diameter of the fan. It would consequently be acceptable to use a series of base designs with slightly varying Hub-Tip ratios, but similar diameters for the benchmark MDO application. This would serve to alleviate the computational cost of including the HTR as a design variable in the design optimization analysis.

For the completed benchmark MDO study, the optimized model2 design offers the best combination of high efficiency with lowest power requirements (related to the torque required to turn the fan). The high static and total efficiency combined with the high head rise also make the optimized model2 design the best possible for the benchmark study. It is important to note that the model2 fan design may not represent the globally optimal design. This is particularly the case as RSO algorithms in general are not as efficient as the evolutionary type algorithms such as the Particle Swarm and the Genetic Algorithm. The use of a series of base models which incorporates some of the observation of the current benchmark study, will allow for the identification of a unique optimal design for the current application of the MDO based design methodology.

4.4 Robustness of Automated MDO Environment

A critical issue in the benchmark MDO based fan design study is the ability to tolerate invalid fan designs that cause iterations of the optimization analyses to fail. These failed designs are caused when the optimizer attempts to investigate regions in the problem design space that do not violate the problem constraints but contain designs which may be considered too difficult to analyze using the automated meshing and CFD analyses tools. Two particular areas were identified as sources for failed design iterations and they are:

1. Failed grid generation for highly skewed or swept fan designs (ref Figure 129)
2. Invalid results from CFX-Bladegen(Plus) CFD analysis

Of the two primary sources of failed optimization iterations, the failure in the grid generation step of the optimization cycle is the most prevalent. The immediate result of failed optimization iterations is that the optimizer then assumes that the region immediately containing that particular design point is invalid and should be ignored in the optimization of the target function. This decreases the effectiveness of the optimization algorithms in locating a globally optimal solution to the design problem.

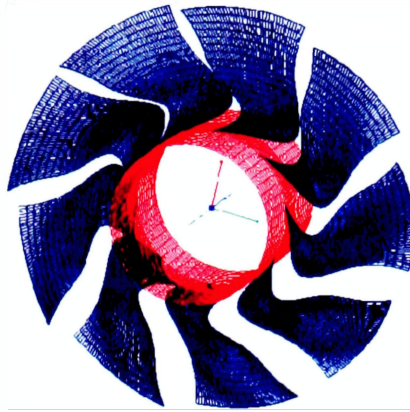


Figure 129: Fan Design with Highly Skewed Blade Geometry

For the benchmark MDO study, Table 21 summarizes the observed occurrence rates of failed function calls (equivalent to optimization iterations). A function call represents a complete design cycle from the generation of a unique blade geometry by the optimizer, to the CFD analyses of the blade design using the CFX-Bladegen(Plus) solver. The failed functions calls in the optimization analyses are represented by the “spikes” or infinitely long lines seen in the target function plots of Figure 77, Figure 93 and Figure 108. The target function plot for model1 (Figure 77) is duplicated below for convenience.

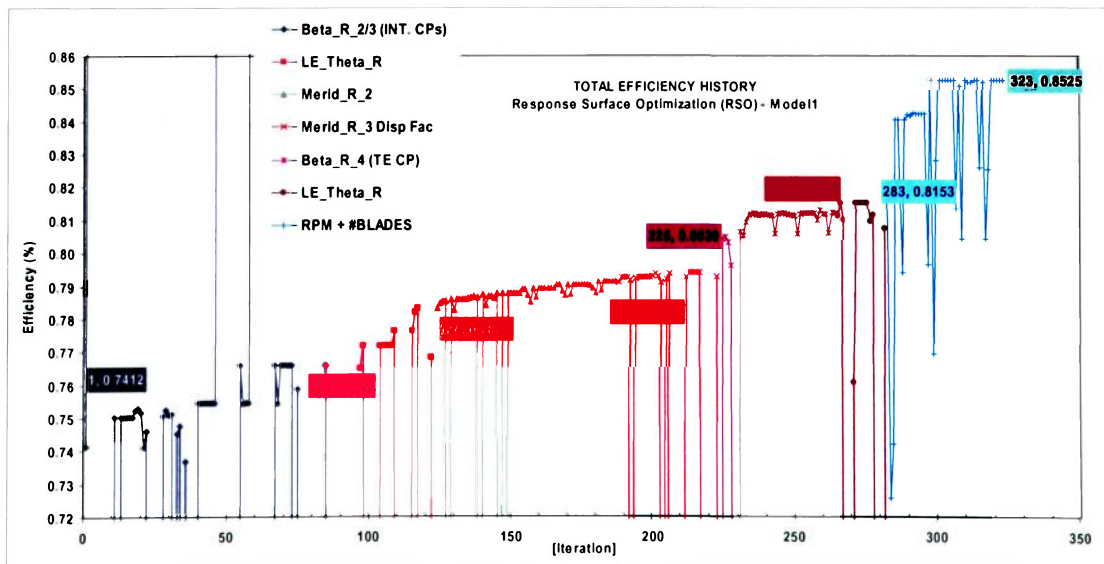


Table 21: Robustness of MDO Based Fan Design Optimization (RSO Algorithm)

	# Fcn. Calls	Failed Mesh	Failed CFD Results	% Mesh	% CFD Results
Model1	323	84	1	26.0	0.5
Model2	293	47	0	16.0	0.0
Model3	353	102	0	28.9	0.0
Average				23.6	0.17

In the target function plots, each spike represents an invalid design point where the analyses of the design failed. The tabulated data clearly shows that most of the functions call failures occur because the CFX-Bladegen(Plus) grid generator cannot successfully generate a mesh for the blade geometry specified by the optimizer. Function call failures due to invalid CFD results are relatively rare (less than 1%), only occurring once during optimization of the base model1 design. The occurrence of a large number of failed function calls is partly responsible for the inability of the optimization analyses to converge on a single optimal solution. It is likely that regions of the design span which contain better fan design are ignored by the optimizer since the failed designs are communicated back to the optimizer using the Pass/Fail design parameters.

In all cases, significant gains are made in the target function (total efficiency) even though a unique, globally optimal solution is not obtained. The use of base design with distinct geometric properties that are not included as design variables also negatively affected the convergence of the benchmark MDO analyses to a globally optimal solution using the RSO algorithm. However, the use of geometrically distinct base design models facilitates the identification of the influence of the fan diameter and Hub-Tip ratio on the convergence of the benchmark MDO fan design optimization study.

5 CONCLUSIONS & RECOMMENDATIONS

The primary aim in the benchmark MDO based fan design optimization is to increase the total efficiency of the fan design by modifying the blade geometry. Three base designs were used in order to investigate the ability of the search algorithms to converge on a unique optimal solution. The MDO methodology was implemented using the VisualDoc software package as the design optimization tool and the CFX-Bladegen(Plus) tool as the turbomachinery modeling and analyses tool. The CFD component of the CFX-Bladegen(Plus) package was used to iteratively analyze blade designs and return the performance results to the optimizer.

The investigated fan designs were generated based on mathematical search algorithms which are typically used to optimize discontinuous functions in the parametric design space. For the benchmark MDO study, the blade geometry was parameterized by dividing the blade geometry into 5 spanwise layers. At each layer, the shape of the airfoil was modified using 4 bezier control points. The circumferential LE sweep distribution of the blade was controlled using a set of LE angular coordinates. A total of 32 design variables including the rotational rate and number of blades in the cascade were used as design variables. In addition to the design variables, constraints were dynamically imposed on the relative magnitudes of the design variables to further limit the investigation to feasible fan designs, eliminating blade geometries with excessive camber or sweep.

The MDO environment for the benchmark study including the communication between the component modules, was implemented using the VisualScript tool, a component of the VisualDoc software package from Vanderplaats Research and Development, Colorado Springs, Colorado. The response surface optimization (RSO) algorithm was used in the benchmark fan design optimization study. The RSO algorithms attempts to approximate the behavior of the objective function by using a surface to map the design points obtained using analysis function calls. The optimum of the response surface is used to obtain the optimal design configuration. Evolutionary type search algorithms were not studied as part of the current MDO benchmark case.

The results showed considerable improvements in the target function with no violated constraints. The RSO optimized fan designs showed increases in total efficiency between 7% and 10% over the base fan designs. Considerable gains in the static pressure and headrise across the blades accompanied the increases in total efficiency. In two of three cases, the optimized fan designs showed a lower mass-averaged change in the relative total pressure (δP_{Trel}), an important measure of the losses in the flow as it traverses the blade surface. A remarkable *S-shaped* leading edge was obtained in the three optimized blade geometries.

The optimization analysis did not converge to a single unique solution, and this could be attributed to several factors. The first issue is that the RSO algorithm is known to be generally less efficient compared to the evolutionary type search algorithms in obtaining global optima of highly discontinuous functions. Hence, in the benchmark DMO study, three distinct optimized designs were obtained for the three base design used. The use of three distinct base designs with differing geometric characteristics that are not included as design variables inherently precludes the optimization analyses from converging to a single, global unique solution.

The differing geometric parameters not included as design variables included the fan outer diameter and the Hub-Tip ratio. The effect of these parameters on the convergence of the MDO analysis is readily observed. The results of the benchmark case suggest that the convergence of the MDO analysis is more tolerant of small variations in Hub-Tip ratio and highly intolerant to variations in the outer diameter of the fan. In other words, for the specific benchmark application, the RSO algorithm would yield more similar optimized designs for a starting set of designs with the same outer diameter but slightly differing Hub-Tip ratio, with the reverse situation yielding very different optimized designs.

CFD analyses for a large number of designs generated by the optimizer failed on numerous occasions. The primary reason for failure was the inability of the CFX-Bladegen(Plus) mesher to generate grids for approximately 25% of the blade designs.

This created an inherently smaller sample of blade designs to be used in the response surface approximation, further degrading the ability of the optimizer to locate more globally optimal solutions in the benchmark study. The results obtained from the benchmark MDO fan design study lead to the following suggestions for future fan design optimization studies:

1. Clearly, the more efficient evolutionary type search algorithms must be investigated in subsequent studies for the benchmark case. The highly non-linear increase in computational effort that these algorithms require precluded their inclusion in this optimization study. Of particular interest is the Particle Swarm Optimization which does not require that all design variables are discrete-type design parameters; a critical requirement in the VisualDoc genetic optimization (GA) algorithm implementation.
2. An exhaustive study of the effect of varying the design variables in different sequences is required in future studies. For example, does varying RPM and number of blades first change the optimal LE shape or is the concept of optimization through linear superposition a valid assumption?
3. The current effort varied the design variables in the same order in all three cases. A brief effort was made in the first base design to determine if the order in which design parameters are varied is significant. The results suggest that for this specific benchmark case, once the target function is optimized relative to a set of design variables, no appreciable gain was obtained for any subsequent optimization relative to the same set of design variables. A more thorough study to specifically address this issue is required.
4. The results showed incomplete CFD analyses for about 25% of the fan designs generated by the optimizer. It is important to mention that the fan designs generated by the optimizer were verified as not violating any of the design constraints, but still could not be analyzed because of a failure of the grid generation step of the CFD

analysis. The numerical consequence of this is an artificial narrowing of the design space investigated by the optimizer. A more robust grid generation tool is required in order to ensure all designs generated are analyzed using CFD. A study to identify optimal grid generation control parameters may also serve as an alternative solution to this problem. The use of different CFD analyses tools with more robust grid generation utilities may also be investigated.

5. The unusual wavy leading edge shape observed requires further investigation to determine if it is a consequence of physical or numerical phenomena. Variations in the number of span layers used and application of different MDO algorithms will further clarify the significance of the wavy LE shape in the optimized designs.
6. An investigation into the effect of the various parameters that control the behavior response surface algorithms is needed. This is necessary in order to determine their effect on the ability of the algorithms to locate more globally optimal (unique) solution to the benchmark MDO fan design problem. Similar studies will also be needed to in future optimization analyses that utilize evolutionary type optimization algorithms.

REFERENCES

1. Balabanov, V.O., Charpentier, C., Ghosh, D., Quinn, G., Vanderplaats, G.N. and Venter, G., "VisualDoc: A Software System for General-Purpose Integration and Design Optimization," AIAA Paper 2002-5513.
2. Collis, S.S., Ghayour, K. and Heinkenschloss, M., "Optimal control of Aeroacoustic Flows: Transpiration Boundary Control," AIAA Paper 2002-2757.
3. Jameson, A., Martinelli, L. and Haimes, B., "Aerodynamics Shape Optimization of Complete Aircraft Configurations Using Unstructured Grids," AIAA Paper 2004-533.
4. Rao, C.S., Ray, T. and Tsai, H.M., "Aircraft Configuration Design Using a Multidisciplinary Optimization Approach," AIAA Paper 2004-536.
5. Sasaki, D., Yang, G. and Obayashi, S., "Automated Aerodynamic Optimization System for SST Wing-Body Configuration," AIAA Paper 2002-5549.
6. Ghosh, D.K., Garcelon, J.H., Balabanov, V.O., and Vanderplaats, G.N., "Development of a Flexible Design Optimization Study Tool," AIAA Paper 98-4726.
7. Painchaud-Ouellet, S., Tribes, C., Trepanier, J.Y. and Pelletier, D., "Airfoil Shape Optimization Using NURBS Representation Under Thickness Constraint," AIAA Paper 2004-1095.
8. Collis, S.S., Ghayour, K. and Heinkenschloss, M., "Towards Adjoint-Based Methods for Aeroacoustic Control," AIAA Paper 2001-0821.
9. Collis, S.S., Ghayour, K. and Heinkenschloss, M., "Optimal Transpiration Boundary Control for Aeroacoustics," AIAA Journal, Vol. 41, No. 7, July 2003.
10. Baysal, O. and Ghayour, K., "Limit-Cycle Shape Optimization Using Time-Dependent Transonic Equation," AIAA Paper 99-3375.
11. Balabanov, V.O. and Venter, G., "Multi-Fidelity Optimization with High-Fidelity Analysis and Low-Fidelity Gradients," AIAA Paper 2004-4459.
12. Stefko, G.L. and Jeracki, R.J., "Wind Tunnel Results of Advanced High Speed Propellers in Takeoff, Climb and Landing Operating Regimes," AIAA Paper 85-1259.

13. Idahosa, U. and Golubev, V.V., "On Noise Control in Turbomachinery Using an Automated Multidisciplinary Optimization System," IMECE2005-21789, ASME International Mechanical Engineering Congress and Exposition, November 5-11, 2005.
14. Croce, G., Marini, M. and Noera, F., "Applications of a Genetic Algorithm to Axial Compressor Redesign."
15. Idahosa, U., Golubev, V.V. and Balabanov, V.O., "Application of Distributed Automated MDO Environment to Aero/Acoustic Shape Optimization of a Fan Blade," AIAA Paper 2005-2907.
16. Sobieszczanski-Sobieki, J. and Venter, G., "Multidisciplinary Optimization of a Transport Aircraft Wing Using Particle Swarm Optimization," AIAA Paper 2002-5644.
17. Sobieszczanski-Sobieki, J. and Venter, G., "Particle Swarm Optimization," AIAA Paper 2002-1235.
18. Venter, G. and Haftka, R.T, "Minimum-Bias Experimental Design for Constructing Response Surfaces in Structural Optimization," AIAA Paper 1997-1053.
19. Venter, G. and Haftka, R.T, "A Two Species Genetic Algorithm For Designing Composite Laminates Subject to Uncertainty," AIAA Paper 1996-1535.
20. Carlisle, A. and Dozier, G., "Adapting Particle Swarm Optimization to Dynamic Environment."
21. "CFX-Bladegen & CFX-Bladegen(Plus) v4.1 User's Guides," AEA Technology, Engineering Software Inc., El Dorado Hills, CA, 1997-2002.
22. "VisualScript, Design Integration Toolkit, Version 3.0, Getting Started Manual," Vanderplaats R&D Inc., Colorado Springs, CO, 2002.
23. "VisualDoc, Design Optimization Software, Version 3.0, Getting Started Manual," Vanderplaats R&D Inc., Colorado Springs, CO, 2002.
24. Benini, E., "Three-Dimensional Multi-Objective Design Optimization of a Transonic Compressor Rotor," AIAA Paper 2003-4090.
25. Roberts, W.B., "A Study of Axial Compressor Blade Optimization for Operation in the Low Reynolds Number Regime," AIAA Paper 78-245.

26. Brawley, S.C. and Hobson, G.V., "Turbine Blade Design Using Parallel Processors," AIAA Paper 96-0452.
27. Jha, R., Chattopadhyay, A. and Rajadas, J.N., "Optimization of Turbomachinery Airfoil Shape for Improved Performance," AIAA Paper 98-1917.
28. Medic, G., Mohammadi, B., Moreau, S. and Stanciu, M., "Optimal Airfoil and Blade Design in Compressible and Incompressible Flows," AIAA Paper 98-2898.
29. Chamis, C.C., Abumeri, G.H. and Patanik, S.N., "Multi-Scale Multi-Level Multidisciplinary Analysis and Design for Engine Structures," AIAA Paper 98-4862.
30. Talya, S.S., Jha, R., Chattopadhyay, A. and Rajadas, J.N., "Development of Multidisciplinary Optimization Procedure for Gas Turbine Blade Design," AIAA Paper 99-16261.
31. Giguere, P. and Selig, M.S., "Blade Geometry Optimization for the Design of Wind Turbine Rotors," AIAA Paper 2000-0045.
32. Li, G. and Grandhi, R., "Accuracy and Efficiency Improvement of Response Surface Methodology for Multidisciplinary Design Optimization," AIAA Paper 2000-4715.
33. Pierret, S. Demeulenaere, A., Gouverneur, B., Hirsch, C., and Van den Braembussche, R., "Designing Turbomachinery Blades with the Function Approximation concept and the Navier Stokes Equations," AIAA Paper 2000-4879.
34. Burman, J., Gebart, B. and Martensson, H., "Development of a Blade Geometry Definition with Implicit Design Variables," AIAA Paper 2000-0671.
35. Eyi, S. and Lee, K.D., "Turbomachinery Blade Design Via Optimization," AIAA Paper 2000-0740.
36. Dornberger, R., Stoll, P., Buche, D. and Neu, A., "Multidisciplinary Turbomachinery Blade Design Optimization," AIAA Paper 2000-0838.
37. Papila, N., Shyy, W., Griffin, L.W. and Dorney, D.J., "Shape Optimization of Supersonic Turbines Using Response Surface and Neural Network Methods," AIAA Paper 2001-1065.

38. Tappeta, R.V., Kolonay, R.M. and Burton, S.A., "Application of Approximate Optimization to Turbine Blade Design in a Network-Centric Environment," AIAA Paper 2002-1588.
39. Rajadas, J. and Chattopadhyay A., "Application of Design Optimization to Turbomachinery Design," AIAA Paper 2002-5662.
40. Iepan, H., Guibault, F. and Vallet, M.G., "CGNS-Based Data Model for Turbine Blade Optimization," AIAA Paper 2005-334.
41. Oyama, A., Liou, M.S. and Obayashi, S., "High-Fidelity Swept and Leaned Rotor Blade Design Optimization Using Evolutionary Algorithm," AIAA Paper 2003-4091.
42. Ferlauto, M., Iollo, A. and Zannetti, L., "Set of Boundary Conditions for Aerodynamic Design," AIAA Journal, Vol. 42, No. 8, August 2004.
43. Kodiyalam, S. Parthasarathy, V.N., Hartle, M.S. and McKnight, R.L., "Ply Layup Optimization and Micromechanics Tailoring of Composite Aircraft Engine Structures," AIAA Journal of Propulsion and Power, Vol. 10, No. 6, Nov-Dec 1994.
44. Martin, T. and Dulikravich, G.S., "Analysis and Multidisciplinary Optimization of Internal Coolant Networks in Turbine Blades," AIAA Journal of Propulsion and Power, Vol. 18, No. 4, July-August 2002.
45. Wright, T., "Fluid Machinery: Performance, Analysis and Design," CRC Press LLC, Boca Raton, FL, 1999.
46. Bleier, F.P., "Fan Handbook: Selection, Application and Design," McGraw-Hill, New York, NY, 1998.
47. "CFX-5 Users Manual, Solver Theory," CFX ANSYS Inc., Canonsburg, PA, 1996-2004.

APPENDIX A: SAMPLE INPUT FILES

CONTENTS

App. [A] File 1: fan.bgi – CFX-Bladegen Turbomachinery Definition Batch File.....	201
App. [A] File 2: Parameters.txt – CFX-Bladegen(Plus) CFD Analyses Input File.....	208
App. [A] File 3: dvar.vef – VisualScript Design Variables Input File	209
App. [A] File 4: resp.vef – VisualScript Response Output File	211
App. [A] File 5: input.txt – VisualScript Element Input File	211
App. [A] File 6: coordinates.txt – MATLAB Script (curvegen.m) Output File.....	213
App. [A] File 7: bladetest.res – VisualScript/MATLAB (bladetest.m) I/O File	213
App. [A] File 8: gridtest.res – VisualScript/MATLAB (gridtest.m) I/O File.....	213
App. [A] File 9: restest.res – VisualScript/MATLAB (restest.m) I/O File	214
App. [A] File 10: Results.txt – VisualScript/CFD Analyses Results File	214
App. [A] File 11: post-run.inp – VisualScript/MATLAB (postrun.m) Input File.....	214
App. [A] File 12: responses.res – VisualScript/MATLAB (postrun.m) Input File	214
App. [A] File 13: summary.res – MATLAB Script (postrun.m)/MDO Summary File..	215

App. [A] File 1: fan.bgi – CFX-Bladegen Turbomachinery Definition Batch File

```
Begin Defaults
End Defaults

Begin Equations
End Equations

Begin Model
  NumMainBlades=9
  Mode=Angle/Thickness
  RightHandedCoordSystem

  BladeOutputPointClustering=BothEnds
  NumBladePoints=60
  NumLeadingEdgePoints=9
  NumTrailingEdgePoints=9

  CurveDisplayMaximumError=0.2920620000

  DataFromLeToTe
  BetaAxialDef
  ThicknessIsPercentCamber

  MeridionalSpanCurveRuledShape=AnyBladeType
  OldSpanwiseInterpolationScheme

  Designer="idahosau"
  Company="Embry Riddle Aeronautical Univeristy"
  Comment=".bgi file configured for optimization "
  DeviceType=Fan
  ConfigurationType=Axial
  RotationType=Negative
  GeometryUnits=KM
End Model

Begin Meridional
  MeridionalControlCurveMode=Minimal
  SpanByGeom
  Begin HubCurve
    New Segment
      CurveType=Bezier
      UpstreamControl=Free
      Begin Data
        ( -152.8000000,152.0000000 )
        ( -35.00000000,152.0000000 )
      End Data
      DownstreamControl=Free
    End Segment
    New Segment
      CurveType=Bezier
      UpstreamControl=Free
      Begin Data
        ( -35.00000000,152.0000000 )
        ( 36.91570000,152.0000000 )
      End Data
      DownstreamControl=Free
```

```

End Segment
New Segment
  CurveType=Bezier
  UpstreamControl=Free
  Begin Data
    ( 36.91570000,152.000000 )
    ( 152.000000,152.000000 )
  End Data
  DownstreamControl=Free
End Segment
End HubCurve
Begin ShroudCurve
  New Segment
    CurveType=Bezier
    UpstreamControl=Free
    Begin Data
      ( -152.800000,382.000000 )
      ( -33.72580000,382.000000 )
    End Data
    DownstreamControl=Free
  End Segment
  New Segment
    CurveType=Bezier
    UpstreamControl=Free
    Begin Data
      ( -33.72580000,382.000000 )
      ( 36.48250000,382.000000 )
    End Data
    DownstreamControl=Free
  End Segment
  New Segment
    CurveType=Bezier
    UpstreamControl=Free
    Begin Data
      ( 36.48250000,382.000000 )
      ( 152.000000,382.000000 )
    End Data
    DownstreamControl=Free
  End Segment
End ShroudCurve
Begin InletCurve
  New Segment
    CurveType=Bezier
    UpstreamControl=Free
    Begin Data
      ( -152.800000,152.000000 )
      ( -152.800000,382.000000 )
    End Data
    DownstreamControl=Free
  End Segment
End InletCurve
Begin ExhaustCurve
  New Segment
    CurveType=Bezier
    UpstreamControl=Free
    Begin Data
      ( 152.000000,152.000000 )

```

```

                ( 152.0000000,382.0000000 )
                End Data
                DownstreamControl=Free
            End Segment
        End ExhaustCurve
    Begin LeadingEdgeCurve
        New Segment
            CurveType=Bezier
            UpstreamControl=Free
            Begin Data
                ( -35.0000000,152.0000000 )
                ( -33.72580000,382.0000000 )
            End Data
            DownstreamControl=Free
        End Segment
    End LeadingEdgeCurve
    Begin TrailingEdgeCurve
        New Segment
            CurveType=Bezier
            UpstreamControl=Free
            Begin Data
                ( 36.91570000,152.0000000 )
                ( 36.48250000,382.0000000 )
            End Data
            DownstreamControl=Free
        End Segment
    End TrailingEdgeCurve

    New SpanLayer
        Name=Layer1
        OutputLayer=T
        SpanFraction=0.0000000000
    End SpanLayer

    New SpanLayer
        Name=Layer2
        OutputLayer=T
        SpanFraction=0.2500000000
    End SpanLayer

    New SpanLayer
        Name=Layer3
        OutputLayer=T
        SpanFraction=0.5000000000
    End SpanLayer

    New SpanLayer
        Name=Layer4
        OutputLayer=T
        SpanFraction=0.7500000000
    End SpanLayer

    New SpanLayer
        Name=Layer5
        OutputLayer=T
        SpanFraction=1.0000000000
    End SpanLayer

```

End Meridional

New Blade

PitchFraction=0.0000000000
LeadingEdgeEndType=Ellipse
HubLE_EllipseRatio=4.000000000
ShrLE_EllipseRatio=4.000000000
TrailingEdgeEndType=Ellipse
HubTE_EllipseRatio=1.000000000
ShrTE_EllipseRatio=1.000000000
EllipseAtMean=T

Begin AngleDefinition

AngleLocation=MeanLine
SpanwiseDistribution=General

New AngleCurve

Layer="Layer1"
DefinitionType=BetaCurve
HorizDim=PercentMeridionalPrime
VertDim=Degree

LE_Theta=-3.000001286

New Segment

CurveType=Bezier
UpstreamControl=Free
Begin Data

(0.0000000000,68.74838307)
(45.67735086,50.35778979)
(75.72794937,22.91020639)
(100.0000834,-4.549213767)

End Data

DownstreamControl=Free

End Segment

End AngleCurve

New AngleCurve

Layer="Layer2"
DefinitionType=BetaCurve
HorizDim=PercentMeridionalPrime
VertDim=Degree

LE_Theta=0.4696850173

New Segment

CurveType=Bezier
UpstreamControl=Free
Begin Data

(0.01796375222,77.09506259)
(46.32077282,57.62865642)
(65.99057750,43.74067857)
(100.0178741,18.80924492)

End Data

DownstreamControl=Free

End Segment

End AngleCurve

New AngleCurve

```

Layer="Layer3"
DefinitionType=BetaCurve
HorizDim=PercentMeridionalPrime
VertDim=Degree

LE_Theta=-2.307894063
New Segment
  CurveType=Bezier
  UpstreamControl=Free
  Begin Data
    ( 0.01843622179,76.12541339 )
    ( 51.43305827,65.77359969 )
    ( 73.80030341,54.04348231 )
    ( 100.0180939,43.61194563 )
  End Data
  DownstreamControl=Free
End Segment
End AngleCurve

New AngleCurve
  Layer="Layer4"
  DefinitionType=BetaCurve
  HorizDim=PercentMeridionalPrime
  VertDim=Degree

  LE_Theta=-9.467554607
  New Segment
    CurveType=Bezier
    UpstreamControl=Free
    Begin Data
      ( 0.01128194190,71.92026747 )
      ( 15.59462955,71.13036083 )
      ( 87.83423468,68.07331507 )
      ( 100.0109035,61.00071858 )
    End Data
    DownstreamControl=Free
  End Segment
End AngleCurve

New AngleCurve
  Layer="Layer5"
  DefinitionType=BetaCurve
  HorizDim=PercentMeridionalPrime
  VertDim=Degree

  LE_Theta=-20.00000857
  New Segment
    CurveType=Bezier
    UpstreamControl=Free
    Begin Data
      ( 0.0000000000,60.28797650 )
      ( 34.87808036,42.44960694 )
      ( 92.79899874,79.43770111 )
      ( 99.99980344,87.74648883 )
    End Data
    DownstreamControl=Free
  End Segment

```

```

End AngleCurve
End AngleDefinition

Begin ThicknessDefinition
ThicknessType=Normal To Camber Line
SpanwiseDistribution=General

New ThicknessCurve
  Layer="Layer1"
  HorizDim=PercentCamberByPercentCamber

  New Segment
    CurveType=Bezier
    UpstreamControl=Free
    Begin Data
      ( 0.0000000000,0.3377100000 )
      ( 13.49236676,2.314710000 )
      ( 59.81837068,1.499160000 )
      ( 100.00000000,0.2025200000 )
    End Data
    DownstreamControl=Free
  End Segment
End ThicknessCurve

New ThicknessCurve
  Layer="Layer2"
  HorizDim=PercentCamberByPercentCamber

  New Segment
    CurveType=Bezier
    UpstreamControl=Free
    Begin Data
      ( 0.0000000000,0.3124710000 )
      ( 14.91090367,2.235520000 )
      ( 65.06228908,1.214610000 )
      ( 100.00000000,0.1857600000 )
    End Data
    DownstreamControl=Free
  End Segment
End ThicknessCurve

New ThicknessCurve
  Layer="Layer3"
  HorizDim=PercentCamberByPercentCamber

  New Segment
    CurveType=Bezier
    UpstreamControl=Free
    Begin Data
      ( 0.0000000000,0.2841400000 )
      ( 13.91452017,1.915000000 )
      ( 59.40517411,1.255590000 )
      ( 100.00000000,0.1688710000 )
    End Data
    DownstreamControl=Free
  End Segment
End ThicknessCurve

```



```

New ThicknessCurve
  Layer="Layer4"
  HorizDim=PercentCamberByPercentCamber

  New Segment
    CurveType=Bezier
    UpstreamControl=Free
    Begin Data
      ( 0.0000000000,0.2547730000 )
      ( 13.88794966,1.721080000 )
      ( 59.18748779,1.135350000 )
      ( 100.0000000,0.151950000 )
    End Data
    DownstreamControl=Free
  End Segment
End ThicknessCurve

New ThicknessCurve
  Layer="Layer5"
  HorizDim=PercentCamberByPercentCamber

  New Segment
    CurveType=Bezier
    UpstreamControl=Free
    Begin Data
      ( 0.0000000000,0.2251400000 )
      ( 15.29092714,1.716780000 )
      ( 71.35495376,0.7512370000 )
      ( 100.0000000,0.1350140000 )
    End Data
    DownstreamControl=Free
  End Segment
End ThicknessCurve
End ThicknessDefinition
End Blade

Begin PlusData
Begin Case
Comments = ".bgi file configured for optimization RPM = -1140 "
Machine Type = fan
Component Type = rotor
Units = MM
End Case
End PlusData

```

App. [A] File 2: Parameters.txt – CFX-Bladegen(Plus) CFD Analyses Input File

```
Begin PlusData
  Begin Case
    machine type      = fan
    component type    = rotor
    housing type      = shrouded
    units             = mm
    comments          = ".bgi file configured for Beta + LE Theta
optimization RPM = -1140 "
  End Case
  Begin Grid
    spacing factor    = 2.00
    inflation layers  = 4
  End Grid
  Begin Fluid
    fluid type        = incompressible
    description       = Air at STP
    density           = 1.185 [kg/m^3]
    viscosity         = 1.79e-005 [N-s/m^2]
    turbulence model  = turbulent
  End Fluid
  Begin Conditions
    runtime           = massflow exit
    rotational rate   = -1140. [RPM]
    ptotal inlet     = 101325.0 [Pa]
    ttotal inlet     = 300.0 [K]
    massflow rate     = 4.285 [kg/s]
    pstatic exit     = 0.0 [Pa]
    swirl angle       = 0.0 [rad]
    wall roughness    = 0.0 [m]
  End Conditions
  Begin Solution
    maximum iterations = 150
    target residual    = 0.00001
    advection blend    = 0.88
    restart option     = current
    physical timestep  = autocompute
    timestep control   = false
    vinlet             = 28.268999
    pinlet             = 101273.000000
    tinlet             = 300.000000
    voutlet            = 12.507400
    poutlet            = 101567.000000
    toutlet           = 300.000000
  End Solution
End PlusData
```

App. [A] File 3: dvar.vef – VisualScript Design Variables Input File

Sample Data	Legend
1.0000000000000000e+000	Mode
5.0000000000000000e+000	Num_Layers
1.0000000000000000e+002	Curve_Discret
9.0000000000000000e+000	Num_Blades
-1.1400000000000000e+003	Param_RPM
5.5000000000000000e+001	Theta_Offset_Lim
8.5000000000000000e+001	Beta_Offset_Lim_L1
8.5000000000000000e+001	Beta_Offset_Lim_L2
8.5000000000000000e+001	Beta_Offset_Lim_L3
8.5000000000000000e+001	Beta_Offset_Lim_L4
8.5000000000000000e+001	Beta_Offset_Lim_L5
5.6518999999999997e-001	XBez_Disp_L1
3.3584999999999998e-001	XBez_Disp_L2
4.5030999999999999e-001	XBez_Disp_L3
9.0368999999999999e-001	XBez_Disp_L4
9.6006999999999998e-001	XBez_Disp_L5
4.5677000000000000e+001	Beta_BezX_1_2
4.6320999999999998e+001	Beta_BezX_2_2
5.1433000000000000e+001	Beta_BezX_3_2
1.5595000000000001e+001	Beta_BezX_4_2
3.4878000000000000e+001	Beta_BezX_5_2
6.8748000000000005e+001	Beta_BezY_1_1
6.6000000000000000e+001	Beta_BezY_1_2
3.0000000000000000e+001	Beta_BezY_1_3
-1.0000000000000000e+001	Beta_BezY_1_4
7.7094999999999999e+001	Beta_BezY_2_1
4.7393999999999998e+001	Beta_BezY_2_2
1.8452999999999999e+001	Beta_BezY_2_3
1.5644000000000000e+001	Beta_BezY_2_4
7.6125000000000000e+001	Beta_BezY_3_1
5.4466999999999999e+001	Beta_BezY_3_2
4.2128000000000000e+001	Beta_BezY_3_3
2.2382000000000001e+001	Beta_BezY_3_4
7.1920000000000002e+001	Beta_BezY_4_1
7.4772000000000006e+001	Beta_BezY_4_2
6.5191999999999993e+001	Beta_BezY_4_3
3.2000000000000000e+001	Beta_BezY_4_4
6.0287999999999997e+001	Beta_BezY_5_1
8.0000000000000000e+001	Beta_BezY_5_2
7.9322000000000003e+001	Beta_BezY_5_3
6.1000000000000000e+001	Beta_BezY_5_4
-1.4000000000000000e+001	Theta_LE_L1
0.0000000000000000e+000	Theta_LE_L2
-2.3365999999999998e+000	Theta_LE_L3
-4.5366999999999997e+000	Theta_LE_L4
-1.0000000000000000e+001	Theta_LE_L5
2.7480000000000047e+000	SL112
3.6000000000000000e+001	SL123
4.0000000000000000e+001	SL134
2.9701000000000001e+001	SL212
2.8940999999999999e+001	SL223

2.8089999999999993e+000	SL234
2.1658000000000001e+001	SL312
1.2338999999999999e+001	SL323
1.9745999999999999e+001	SL334
2.8520000000000039e+000	SL412
9.5800000000000125e+000	SL423
3.3191999999999993e+001	SL434
1.9712000000000003e+001	SL512
6.779999999999727e-001	SL523
1.8322000000000003e+001	SL534
8.346999999999942e+000	RL112
9.699999999999886e-001	RL123
4.20499999999983e+000	RL134
1.1632000000000005e+001	RL145
1.8606000000000002e+001	RL212
7.0730000000000004e+000	RL223
2.0305000000000007e+001	RL234
5.227999999999944e+000	RL245
1.1547000000000001e+001	RL312
2.3675000000000001e+001	RL323
2.3063999999999993e+001	RL334
1.4130000000000010e+001	RL345
2.564399999999998e+001	RL412
6.7380000000000013e+000	RL423
9.617999999999986e+000	RL434
2.9000000000000000e+001	RL445
1.4000000000000000e+001	Rtheta12
2.336599999999998e+000	Rtheta23
2.200099999999999e+000	Rtheta34
5.4633000000000003e+000	Rtheta45

App. [A] File 4: resp.vef – VisualScript Response Output File

Sample Data	Legend
1.0000000000000000e+000	PFBladeTest
1.0000000000000000e+000	PFgrid
1.0000000000000000e+000	PFResults
4.2848600000000001e+000	MassFlowRate_kgs
3.6159100000000000e+000	VolFlowRate_m3s
2.2956600000000002e+001	TotalBladeTorque_Nm
5.4108300000000000e+001	Headrise_m
2.7043400000000001e-001	InletFlowCoeff
2.9360700000000001e-001	ExitFlowCoeff
4.4024099999999999e-001	HeadCoeff_LE_TE
8.2905600000000002e-001	TotalEffic
5.0586500000000001e-001	StaticEffic

App. [A] File 5: input.txt – VisualScript Element Input File

Sample Data	Legend
1	Flag Discretization Type (Mode)
5.	# Span Layers - 5
100.	Curve Discretization Factor
9	# of Blades
-1140.	RPM
55	Theta_Offset_Limit
65	Beta_Offset_Limit_Layer_1 (Unit: deg)
55	Beta_Offset_Limit_Layer_2 (Unit: deg)
50	Beta_Offset_Limit_Layer_3 (Unit: deg)
48	Beta_Offset_Limit_Layer_4 (Unit: deg)
40	Beta_Offset_Limit_Layer_5 (Unit: deg)
0.565187	Distance_Offset_Layer_1 (Range: 0-1)
0.335853	Distance_Offset_Layer_2 (Range: 0-1)
0.450314	Distance_Offset_Layer_3 (Range: 0-1)
0.903693	Distance_Offset_Layer_4 (Range: 0-1)
0.960070	Distance_Offset_Layer_5 (Range: 0-1)
45.677351	BetaX_1-2 (Range: 10 80)
46.320773	BetaX_2-2 (Range: 10 - 80)
51.433058	BetaX_3-2 (Range: 10 - 80)
15.594630	BetaX_4-2 (Range: 10 - 80)
34.878080	BetaX_5-2 (Range: 10 - 80)

68.748383	BetaY_1-1 (Unit: deg)
50.357790	BetaY_1-2 (Unit: deg)
22.910206	BetaY_1-3 (Unit: deg)
-4.549214	BetaY_1-4 (Unit: deg)
77.095063	BetaY_2-1 (Unit: deg)
57.628656	BetaY_2-2 (Unit: deg)
43.740679	BetaY_2-3 (Unit: deg)
18.809245	BetaY_2-4 (Unit: deg)
76.125413	BetaY_3-1 (Unit: deg)
65.773600	BetaY_3-2 (Unit: deg)
54.043482	BetaY_3-3 (Unit: deg)
43.611946	BetaY_3-4 (Unit: deg)
71.920267	BetaY_4-1 (Unit: deg)
71.130361	BetaY_4-2 (Unit: deg)
68.073315	BetaY_4-3 (Unit: deg)
61.000719	BetaY_4-4 (Unit: deg)
60.287977	BetaY_5-1 (Unit: deg)
42.449607	BetaY_5-2 (Unit: deg)
79.437701	BetaY_5-3 (Unit: deg)
87.746489	BetaY_5-4 (Unit: deg)
-3.000001	LE_Theta_Span Layer1 (Unit: deg)
0.469685	LE_Theta_Span Layer2 (Unit: deg)
-2.307894	LE_Theta_Span Layer3 (Unit: deg)
-9.467555	LE_Theta_Span Layer4 (Unit: deg)
-20.000009	LE_Theta_Span Layer5 (Unit: deg)

App. [A] File 6: coordinates.txt – MATLAB Script (curvegen.m) Output File

Sample Data	0.000000	45.677351	75.727936	100.000000
	0.000000	46.320773	65.990572	100.000000
	0.000000	51.433058	73.800292	100.000000
	0.000000	15.594630	87.834242	100.000000
	0.000000	34.878080	92.798982	100.000000
	68.748383	50.357790	22.910206	-4.549214
	77.095063	57.628656	43.740679	18.809245
	76.125413	65.773600	54.043482	43.611946
	71.920267	71.130361	68.073315	61.000719
	60.287977	42.449607	79.437701	87.746489
	-3.000001			
	0.469685			
	-2.307894			
	-9.467555			
	-20.000009			
Legend	BetaX1_1	BetaX1_2	BetaX1_3	BetaX1_4
	BetaX2_1	BetaX2_2	BetaX2_3	BetaX2_4
	BetaX3_1	BetaX3_2	BetaX3_3	BetaX3_4
	BetaX4_1	BetaX4_2	BetaX4_3	BetaX4_4
	BetaX5_1	BetaX5_2	BetaX5_3	BetaX5_4
	BetaY1_1	BetaY1_2	BetaY1_3	BetaY1_4
	BetaY2_1	BetaY2_2	BetaY2_3	BetaY2_4
	BetaY3_1	BetaY3_2	BetaY3_3	BetaY3_4
	BetaY4_1	BetaY4_2	BetaY4_3	BetaY4_4
	BetaY5_1	BetaY5_2	BetaY5_3	BetaY5_4
	THeta_LE_L1			
	Theta_LE_L2			
	Theta_LE_L3			
	Theta_LE_L4			
	Theta_LE_L5			

App. [A] File 7: bladetest.res – VisualScript/MATLAB (bladetest.m) I/O File

Sample Data	Legend
0	PFBladeTest

App. [A] File 8: gridtest.res – VisualScript/MATLAB (gridtest.m) I/O File

Sample Data	Legend
0	PFgrid

App. [A] File 9: retest.res – VisualScript/MATLAB (retest.m) I/O File

Sample Data	Legend
0	PFResults

App. [A] File 10: Results.txt – VisualScript/CFD Analyses Results File

Sample Data	
Mass Flow Rate	= 4.28026 [kg/s]
Volume Flow Rate	= 3.61203 [m3/s]
Total Blade Torque	= 14.7036 [N-m]
Headrise	= 30.7576 [m]
Inlet Flow Coefficient	= 0.269866
Exit Flow Coefficient	= 0.35143
Head Coefficient (LE-TE)	= 0.250253
Total Efficiency (LE-TE)	= 0.735001
Static Efficiency (LE-TE)	= 0.407289

App. [A] File 11: post-run.inp – VisualScript/MATLAB (postrun.m) Input File

Sample Data	Legend
1.	PFBladeTest
1.	PFgrid
1.	PFResults

App. [A] File 12: responses.res – VisualScript/MATLAB (postrun.m) Input File

Sample Data	Legend
1	PFBladeTest
1	PFgrid
1	PFResults
4.27983	MassFlowRate_kgs
3.61167	VolFlowRate_m3s
14.7034	TotalBladeTorque_Nm
30.7811	Headrise_m
0.269834	InletFlowCoeff
0.350639	ExitFlowCoeff
0.250447	HeadCoeff_LE_TE
0.735505	TotalEffic
0.407915	StaticEffic

APPENDIX B: MATLAB SCRIPTS

CONTENT

App. [B] Script 1: MATLAB – genpostfile.m.....	217
App. [B] Script 2: MATLAB – curvegen.m.....	218
App. [B] Script 3: MATLAB – bladetest.m.....	223
App. [B] Script 4: MATLAB – gridtest.m.....	226
App. [B] Script 5: MATLAB – restest.m.....	227
App. [B] Script 6: MATLAB – postrun.m.....	228

App. [B] Script 1: MATLAB – genpostfile.m

```

function [header] = genpostfile(mode)
% GENPOSTFILE creates the template for the results summary file used
for
% the the optimization runs in VisualDoc. It requires an input file
% variables-XX.txt which contains the list of the variables and results
as
% listed in the visualdoc .vdb file

% XX refers to the variaous modes:
% AT - Angle thickness mode - mode = 1
% PS - Pressure Suction Mode     mode = 2

% Obtain Title of Optimization Run
% =====
fprintf(1, '\n\nBLADE DESIGN OPTIMIZATION SUMMARY FILE GENERATION,
TOOL');
fprintf(1, '\n=====');
optrun = input('\n\n--> Enter Optimization Task Name:  ', 's');
lnth = length(optrun);

% Read Variable List File to Get Headers
% =====
ext='.txt';
filename='variables-mode';
file=[filename,num2str(mode),ext]

header = textread([file], '%s', 'delimiter', '\n', 'whitespace', ' ');

% Printf List of Variables to Output File
% =====

fid = fopen('\outputFiles\summary.res', 'wt');
fprintf(fid, '%s      \n', optrun);
for i=1:lnth
    fprintf(fid, '=');
end
fprintf(fid, ' \n\n');

fprintf(fid, 'Geometry_File      ');
list = length(header);

for i = 1:list
    fprintf(fid, '%s      ', char(header(i)));
end

fprintf(fid, ' \n');
fclose(fid);

% Create New Optimization Iterations TRacking File
% =====
opttrk=1;
fid = fopen('optiter.trk', 'w');
fprintf(fid, '%i', opttrk);           % Refresh Optimization Iterations
Counter

```

```

fclose(fid);

fprintf(1,'\n\n--> SUCCESS:      Post-Optimization Summary File
Generation Complete...');
fprintf(1,'\n\n-->              You may begin your optimization
task.');
```

App. [B] Script 2: MATLAB – curvegen.m

```

function [] = curvegen()

% This function generates the bezier coordinates for the blade geometry.
% Two modes are supported:
% Mode = 1: Read Beta Bezier Curve Coordinates
% Mode = 2: Read Beta Bezier Spline Curve Equation (4th-Order)
Coefficients
% Note: This Function Implies that 3-rd Order Bezier Poly. are used

%
=====

% READ RELEVANT DATA FROM INPUT FILE

%
=====

% Open Blade Parameterization Input file to read data
fid = fopen('input.txt','rt');

% Read Mode for Parameterization Type Being Used
mode = fscanf(fid,'%f',1);
fprintf(1,'\nBlade Geometry Parameterization Mode = %i\n',mode);

% Read Number of Spanwise Blade Layers
nspan = fscanf(fid,'%f',1);

% Read Curve Discretization Number (Number of Points)
curve_discret = fscanf(fid,'%f',1);

% Read Number of Blades
nblades = fscanf(fid,'%f',1);

% Read Rotational Speed (RPM)
rpm = fscanf(fid,'%f',1);

% Read LE Edge Sweep (Theta-angle) Layer-to-Layer Max Offset
theta_offset = fscanf(fid,'%f',1);

%
=====

% Read Spanwise Blade "Camber" Angle (Beta) Layer-to-Layer Max. Offset
beta_offset=zeros(nspan,1);
for i = 1:nspan
    beta_offset(i) = fscanf(fid,'%f',1);
```

```

end

%beta_offset

%
=====
% Read Spanwise Interior Bezier X2-Coord "Distance-Offset" Design
Parameter
dist_offset=zeros(nspan,1);
for i = 1:nspan
    dist_offset(i) = fscanf(fid,'%f',1);
end

%dist_offset

%
=====
% Mode = 1: Read Beta Bezier Curve X2-Coordinates
% Mode = 2: Read Beta Beiaer X2 Spline Curve Eq. (4th-Order)
Coefficients

bez_beta_x = zeros(nspan,4);
bez_coeff_x = zeros(nspan,1);
fprintf(1,'\n');

if (mode <= 2) % Direct Parameterization   Bezier Coordinates Being Used

    for i = 1:nspan
        bez_beta_x(i,1) = 0;
        bez_beta_x(i,2) = fscanf(fid,'%f',1);
        bez_beta_x(i,3) = bez_beta_x(i,2) + (dist_offset(i)*(90 -
bez_beta_x(i,2))) + 5;
        bez_beta_x(i,4) = 100;
        fprintf(1,'\n-->      Computing Beta Bezier X Coords. for Layer
%i:      ...Done',i);
    end

elseif (mode > 2)

    % Read Coefficients
    for i = 1:nspan
        bez_coeff_x(i) = fscanf(fid,'%f',1);
    end

    % Calculate Coordinates

    for i = 1:nspan
        s=((i-1)/(nspan-1));           % Determine Current Span Layer
        sum = 0;                       % Reset Value of Quantity

        % Use polynomial of Order(nspan-1) to determine Quantity
        for j = 1:nspan
            sum = sum + bez_coeff_x(j) * s^(j-1);
        end

        bez_beta_x(i,1) = 0;
        bez_beta_x(i,2) = sum;
    end

```

```

        bez_beta_x(i,3) = bez_beta_x(i,2) + (dist_offset(i)*(90 -
bez_beta_x(i,2))) + 5;
        bez_beta_x(i,4) = 100;
        fprintf(1,'\n-->          Computing Beta Bezier X Coords. for Layer
%i:      ...Done',i);
    end

% Add Other Modes as Necessary   Future Expansion (Mized Parameterization
Modes)
end

%
=====
% Mode = 1: Read Beta Bezier Curve Y-Coordinates
% Mode = 2: Read Beta Bezier Y Spline Curves   Eq. (4th-Order)
Coefficients

bez_beta_y = zeros(nspan,4);
bez_coeff_y = zeros(nspan,4);
fprintf(1,'\n');

if (mode == 1) % Direct Parameterization   Bezier Coordinates Being Used

    for i = 1:nspan
        for j=1:4
            bez_beta_y(i,j) = fscanf(fid,'%f',1);
        end
        fprintf(1,'\n-->          Computing Beta Bezier Y Coords. for Layer
%i:      ...Done',i);
    end

elseif (mode >= 2)

    for i = 1:nspan
        for j=1:4
            bez_coeff_y(i,j) = fscanf(fid,'%f',1);
        end
    end

    % Calculate Coordinates

    for i = 1:nspan
        for j = 1:4

            s=((i-1)/(nspan-1));          % Determine Current Span Layer
            sum = 0;                      % Reset Value of Quantity

            % Use polynomial of Order(nspan-1) to determine Quantity
            for k = 1:nspan
                sum = sum + bez_coeff_y(k,j) * s^(k-1);
            end

            bez_beta_y(i,j) = sum;
        end
        fprintf(1,'\n-->          Computing Beta Bezier Y Coords. for Layer
%i:      ...Done',i);
    end

```

```

end

% Add Other Modes as Necessary Future Expansion (Mized Parameterization
Modes)
end

%
=====
% Mode = 1: Read LE Sweep (Theta)
% Mode = 2: Read LE Sweep (theta) Spline Curve Eq. (4th-Order)
Coefficients

le_theta = zeros(nspan,1);
le_theta_coeff = zeros(nspan,1);
fprintf(1,'\n');

if (mode == 1) % Direct Parameterization Bezier Coordinates Being Used

    for i = 1:nspan
        le_theta(i) = fscanf(fid,'%f',1);
        fprintf(1,'\n--> Computing LE Theta for Layer %i:
...Done',i);
    end

elseif (mode >= 2)

    for i = 1:nspan
        le_theta_coeff(i) = fscanf(fid,'%f',1);
    end

    % Calculate Coordinates

    for i = 1:nspan
        s = ((i-1)/(nspan-1)); % Determine Current Span Layer
        sum = 0; % Reset Value of Quantity

        % Use polynomial of Order(nspan-1) to determine Quantity
        for j = 1:nspan
            sum = sum + le_theta_coeff(j) * s^(j-1);
        end

        le_theta(i) = sum;
        fprintf(1,'\n--> Computing LE Theta for Layer %i:
...Done',i);
    end

% Add Other Modes as Necessary Future Expansion (Mized Parameterization
Modes)
end

fclose(fid);

fprintf(1,'\n\n--> SUCCESS: Blade Geometry Bezier Generation Complete...
Writing Output File\n\n');

%bez_coeff_x

```

```

%bez_beta_x

%bez_coeff_y
%bez_beta_y

%le_theta_coeff
%le_theta

% =====
% Write Beta Bezier X Coordinates To File
fid=fopen('coordinates.txt','wt');
for i=1:nspan
    for j=1:4
        fprintf(fid,'% -12.6f    ',bez_beta_x(i,j));
    end
    fprintf(fid,'\n');
end

fprintf(fid,'\n');

% =====
% Write Beta Bezier Y Coordinates To File
for i=1:nspan
    for j=1:4
        fprintf(fid,'% -12.6f    ',bez_beta_y(i,j));
    end
    fprintf(fid,'\n');
end

fprintf(fid,'\n');

% =====
% Write LE Theta Values To File
for i=1:nspan
    fprintf(fid,'% -12.6f\n',le_theta(i));
end

fclose(fid);
quit force;

```


App. [B] Script 3: MATLAB – bladetest.m

```

function [] = bladetest()
% This function evaluates the various blade designs generated by the VisuakDoc
% Optimizer and Determines if the Following Criteria are met:
%   -> The Maximum Difference Between LE Sweep From one Layer to the next
%       does not exceed the "theta_offset" limit
%   -> The Beta Bezier Y-Coordinates are within the layer-specific limits
%       imposed by the Beta_Offset_Lim Constraint
% The test issues an output file with the "grade" according to the
% following:
%   -> 1   :   Pass (Optimization can proceed on this blade design)
%   -> -1  :   Fail (CFD calculation is skipped for this geometry)

% =====

% READ RELEVANT DATA FROM INPUT FILE

% =====

% Open Blade Parameterization Input file to read data
fid = fopen('input.txt','rt');

% Read Mode for Parameterization Type Being Used
mode = fscanf(fid,'%f',1);
fprintf(1,'\nBlade Geometry Parameterization Mode = %i\n',mode);

% Read Number of Spanwise Blade Layers
nspan = fscanf(fid,'%f',1);

% Read Curve Discretization Number (Number of Points)
curve_discret = fscanf(fid,'%f',1);

% Read Number of Blades
nblades = fscanf(fid,'%f',1);

% Read Rotational Speed (RPM)
rpm = fscanf(fid,'%f',1);

% Read LE Edge Sweep (Theta-angle) Layer-to Layer Max Offset
max_theta_offset = fscanf(fid,'%f',1);

% =====
% Read Spanwise Blade "Camber" Angle (Beta) Layer-to-Layer Max. Offset
max_beta_offset=zeros(nspan,1);
for i = 1:nspan
    max_beta_offset(i) = fscanf(fid,'%f',1);
end

%max_theta_offset
%max_beta_offset
fclose(fid);

% =====
% Open Blade Bezier coordinates file to read data

```

```

%=====
fid = fopen('coordinates.txt','rt');

% =====
% Read Theta Bezier Coordinates

xbezbeta = zeros(nspan,4);
ybezbeta = zeros(nspan,4);
le_theta = zeros(nspan,1);

% Read in Beta Bezier Curve X-Coordinates
for i = 1:nspan
    for j=1:4
        xbezbeta(i,j) = fscanf(fid,'%f',1);
    end
end

% Read in Beta Bezier Curve Y-Coordinates
for i = 1:nspan
    for j=1:4
        ybezbeta(i,j) = fscanf(fid,'%f',1);
    end
end

% Read in LE_Theta (Sweep) Values
for i = 1:nspan
    le_theta(i) = fscanf(fid,'%f',1);
end

fclose(fid);

%xbezbeta;
%ybezbeta;
%le_theta;

% Default is to pass
PFBladetest=1;

% =====
% TEST FAN BLADE GEOMETRY BASED ON LIMITS SET IN CONSTRAINTS FILE
% =====

fprintf(1,'\n\n-> TESTING BLADE GEOMETRY...');

% =====
% TEST #1: LE Sweep Data
% Evaluate Layer-to-Layer Sweep and ensure within limits
% =====
fail_le_theta = 0;
for i=1:nspan-1
    if (abs(le_theta(i+1) - le_theta(i)) > max_theta_offset)
        PFBladetest = -1;
        fprintf(1,'\n\n-> LE Sweep Test: FAILED!');
        fprintf(1,'\n-> Max Sweep Offset = %10.4f',max_theta_offset);
    end
end

```

```

        fprintf(1, '\n--> Sweep Limit Exceeded between Following Layer:');
        fprintf(1, '\n--> Layer #i LE Sweep = %10.4f deg', i, le_theta(i));
        fprintf(1, '\n--> Layer #i LE Sweep = %10.4f deg', i+1, le_theta(i+1));
        fail_le_theta = i;
        break;
    elseif ((i == (nspan-1)) && (PFBladetest == 1))
        fprintf(1, '\n\n--> LE Sweep Test: PASSED');
    end
end

% =====

% TEST #2: Bezier Y-Coordinate Offset Test
% This test checks successive Beta Bezier Curve Y-Coordinates at each layer
% and ensures they conform to the following relation:
%     y(i+1,m) <= y(i,m) + max_beta_offset_Lm
% And y(i+1,m) >= -y(i,m) - max_beta_offset_Lm
% Where:
%     i - Bezier Point Index (1 -> LE, 3 -> TE)
%     m - Layer (Plane of Constant Radius)
% =====

span_fail_bez_beta = 0;
index_fail_bez_beta = 0;
if PFBladetest == 1
    for i=1:nspan
        for j=1:3
            if ((abs(ybezbeta(i,j+1)) > (abs(ybezbeta(i,j)) + max_beta_offset(i))
|| (-abs(ybezbeta(i,j+1)) < (-abs(ybezbeta(i,j)) - max_beta_offset(i))))
                PFBladetest = -1;
                fprintf(1, '\n\n--> Beta Bezier Curve Y-Coordinate Test:
FAILED!');
                fprintf(1, '\n--> Max Bezier Y-Coordinate Offset =
%10.4f', max_beta_offset(i));
                fprintf(1, '\n--> Bezier Y-Coordinate Limit Exceeded on Layer
#i: ', i);
                fprintf(1, '\n--> 1st Index #i BetaY =
%10.4f', j, ybezbeta(i,j));
                fprintf(1, '\n--> 2nd Index #i BetaY =
%10.4f', j+1, ybezbeta(i,j+1));
                span_fail_bez_beta = i;
                index_fail_bez_beta = j;
                break;
            end
        end
    end
    if (PFBladetest == -1)
        break;
    elseif ((i == nspan) && (j==3) && (PFBladetest == 1))
        fprintf(1, '\n\n--> Beta Bezier Curve Y-Coordinate Test: PASSED!');
    end
end
end

% =====
%
% Write Output to Data File "pf.dat" and to Screen. Data at First Failure
% Point is captured also

```

```

%
% =====
fid = fopen('bladetest.res','wt');
fprintf(fid,'%i',PFBladetest);
fprintf('\n');

if((PFBladetest == -1) && (fail_le_theta ~=0))

    fprintf(fid,'\n\n--> LE Sweep Test:    FAILED!');
    fprintf(fid,'\n-->    Max Sweep Offset = %10.4f',max_theta_offset);
    fprintf(fid,'\n-->    Sweep Limit Exceeded between Following Layer:');
    fprintf(fid,'\n-->    Layer #i    LE Sweep = %10.4f
deg',fail_le_theta,le_theta(fail_le_theta));
    fprintf(fid,'\n-->    Layer #i    LE Sweep = %10.4f
deg',fail_le_theta+1,le_theta(fail_le_theta+1));

elseif((PFBladetest == -1) && (span_fail_bez_beta ~=0))

    fprintf(fid,'\n\n--> Beta Bezier Curve Y-Coordinate Test:    FAILED!');
    fprintf(fid,'\n-->    Max Bezier Y-Coordinate Offset =
%10.4f',max_beta_offset(span_fail_bez_beta));
    fprintf(fid,'\n-->    Bezier Y-Coordinate Limit Exceeded on Layer #i:
',span_fail_bez_beta);
    fprintf(fid,'\n-->    1st Index #i    BetaY =
%10.4f',index_fail_bez_beta,ybezbeta(span_fail_bez_beta,index_fail_bez_beta));
    fprintf(fid,'\n-->    2nd Index #i    BetaY =
%10.4f',index_fail_bez_beta+1,ybezbeta(span_fail_bez_beta,index_fail_bez_beta+1));
end
%PFBladetest;
fclose(fid);

quit force;

```

App. [B] Script 4: MATLAB – gridtest.m

```

function [] = gridtest()
% GRIDTEST Checks for the existence of a grid file from the grid
generation
% process. Bbefore Allowing optimization loop to continue

gridexist = exist('grid.bg+','file');

if gridexist ~= 0
    PassFailGrid = 1;
else
    PassFailGrid = -1;
end

fid = fopen('gridtest.res','wt');
fprintf(fid,'%d',PassFailGrid);
fclose(fid);
quit force;

```

App. [B] Script 5: MATLAB – retest.m

```
function [] = retest()
% This code determines if the results file generated by bgSolve is
valid
% by looking for text strings "Error" and "divide by 0". String
searches
% are case insensitive

fid = fopen('Results.txt', 'rt');
fprintf(1, '\n\n--> Examining CFD Analysis Results File...\n\n')
i=0;
while (feof(fid) == 0 && (i<9))
    i=i+1;
    Line = fgetl(fid);
    fprintf(1, '--> Line #%d: %s\n\n', i, Line)
    match1 = regexp(Line, 'Error');
    match2 = regexp(Line, 'divide by 0');

    if (isempty(match1) && isempty(match2))
        PRes = 1;
    else
        PRes = -1;
        break;
    end
end

end

fclose(fid);

% Create Results Test Response File
fid = fopen('results.res', 'wt');
fprintf(fid, '%d', PRes);
fclose(fid);

if PRes == 1
    fprintf(1, '\n\n--> SUCCESS: Valid CFD Analysis Results
File\n\n')
else
    fprintf(1, '\n\n--> WARNING: Invalid CFD Analysis Results
File... Overwriting\n\n')
    system('del Results.txt', '-echo');
    system('copy Reference\Results.txt Results.txt', '-echo');
end
quit force;
```

App. [B] Script 6: MATLAB – postrun.m

```
function [] = postrun()
% POSTRUN executes the tasks required for the "post run" module in the
% visualscript optimization script. It requires an input file
% 'postrun.inp' that it uses to figure out results of the grid and CFD
% results tests to determine validity of if the dataset and geometry file

% PRIMARY POST RUN TASKS:
% =====

% 1.      It eliminates the current grid file if it exists for a
%         successful run.

% 2.      It Reads the variables and Results for the current optimization
run
%         and appends them to the "summary.txt" file located in the
%         "outputFiles" subdirectory

% 3.      It copies the blade geometry file and the parameter file to the
%         'outputFiles' subdirectory for storage purposes

% NOTE: It is important that the order of the test results in post-
run.inp
%       be in Geometry, Grid and Results' test order. It may necessary to
modify
%       the .xml VisualScript to ensure the correct values are going in
the right
%       places.

% CODE
% =====

% Read Various Test Results from post-run.inp file
% =====

fid = fopen('post-run.inp','r');

PFGeom = fscanf(fid,'%f',1);           % Read Results of Geometry Test

PFGrid = fscanf(fid,'%f',1);           % Read Results of Grid Test

PFResults = fscanf(fid,'%f',1);        % Read Results of CFD Results'
Test

fclose(fid);

% Read Optimization Iteration Count
% =====
fid = fopen('optiter.trk','r');
opttrk = fscanf(fid,'%f',1);           % Read Current Count of
Optimization Iterations
fclose(fid);

% Update Optimization Iteration Count
% =====
```

```

fid = fopen('optiter.trk','w');
fprintf(fid,'%i',opttrk+1);           % Update Count of Optimization
Iterations
fclose(fid);

% Create .bgd File and duplicate Blade Geometry File
% =====

dat = datestr(now,1);                 % Get Date
day = datestr(now,8);                 % Get Day
[Y,M,D,H,MI,S] = datevec(datenum(fix(clock))); % Get Time

folder = './outputFiles\';          % Store
Folder Name
extg = '.bgd';                       % Store
Geometry File Extension
extp = '.txt';                       % Store
Parameter File Extension

% Copy Geometry File
% gfilem = [folder,day,'-',date,'-',num2str(H),'-',num2str(MI),'-',
% num2str(S),extg]; % Generate Filename
gfilem = [folder,'fan_',num2str(opttrk),extg];
% Generate Filename
copyfile('fan.bgd',[gfilem])
% Copy File

% Copy Parameter File
% pfilem = [folder,day,'-',date,'-',num2str(H),'-',num2str(MI),'-',
% num2str(S),extp]; % Generate Filename
pfilem = [folder,'param_',num2str(opttrk),extp];
% Generate Filename
copyfile('Parameters.txt',[pfilem])
% Copy File

% Read Variables file (dvar.vef) and responses file (resp.vef) and store
% values
% =====

% Variables File 'dvar.vef'
dvars = textread('dvar.vef','%s','delimiter','\n','whitespace',' ');
dvarlnth = length(dvars);
% The two prior lines determine the # of variables
fid = fopen('dvar.vef','rt');
% This is to prevent "hardwiring"
dvar = zeros(dvarlnth,1);
for i=1:dvarlnth
    dvar(i) = fscanf(fid,'%f',1);
end
fclose(fid);

% Responses File 'resp.vef'

%resps = textread('resp.vef','%s','delimiter','\n','whitespace',' ');
resps = textread('responses.res','%s','delimiter','\n','whitespace',' ');

```

```

resplnth = length(resps);
% The prior two lines determine the # of responses

%fid = fopen('resp.vef','rt');
% This is also to prevent "hardwiring"
fid = fopen('responses.res','rt');
resp = zeros(resplnth,1);
for i=1:resplnth
    resp(i) = fscanf(fid,'%f',1);
end
fclose(fid);

% Open Output File for Storing Variables and Results
% =====
fid = fopen('outputFiles\summary.res','at');

%Print File Name
lfilem = ['fan_',num2str(opttrk),extg];
%lfilem = [day,'-',date,'-',num2str(H),'-',num2str(MI),'-',
',num2str(S),extg]; % This file name excludes the "folder" thus the
fprintf(fid,'\n%s',lfilem);
% distinction btw lfilem and gfilem

% Print Variables
for i=1:dvarlnth
    fprintf(fid,'          %10.4e          ',dvar(i));
end

% Print Responses
% =====
for i=1:resplnth
%   if((PFGrid == -1) || (PFResults == -1) || (PFGeom == -1))
% This Implies Bad Data Set. Only Printf Test Results
%
%       if((resp(i) == 1) || (resp(i) == -1))
%           fprintf(fid,'          %d          ',resp(i));
%       else
%           fprintf(fid,'          --          ');
%       end
%
%   elseif((PFGrid ~= -1) && (PFResults ~= -1) && (PFGeom ~= -1))
% This implies Good Data Set. Printf Everything
%
%       fprintf(fid,'          %d          ',resp(i));
%       end
end

fclose(fid);

% Check Existence of Grid File And Eliminate This is critical to ensure
% the System check for a valid blade mesh for each design iteration
% functions properly
%
% =====
gridexist = exist('grid.bg+', 'file');
if gridexist ~= 0

```



```
fprintf(1,'\n\n-->  Upgating Blade Mesh File... Complete');
system('del grid.bg+', '-echo');
end

quit force;

% Completed Jan 19, 2004
% =====

% Entended to Beta Angle Implementation: SEpt 9, 2005
% =====
```