

**COMPARISON BETWEEN BLADE ELEMENT MOMENTUM THEORY
AND COMPUTATIONAL FLUID DYNAMICS METHODS
FOR PERFORMANCE PREDICTION OF MARINE PROPELLERS**



M.Sc. THESIS

Kayhan ÜLGEN

Department of Naval Architecture and Marine Engineering

Naval Architecture and Marine Engineering Programme

FEBRUARY 2017

**COMPARISON BETWEEN BLADE ELEMENT MOMENTUM THEORY
AND COMPUTATIONAL FLUID DYNAMICS METHODS
FOR PERFORMANCE PREDICTION OF MARINE PROPELLERS**



M.Sc. THESIS

Kayhan ÜLGEN
(508131011)

Department of Naval Architecture and Marine Engineering

Naval Architecture and Marine Engineering Programme

Thesis Advisor: Prof. Dr. Şakir BAL

FEBRUARY 2017

**GEMİ PERVANELERİNİN PERFORMANSININ BELİRLENMESİNDE
KANAT ELEMANI MOMENTUM TEORİSİ VE HESAPLAMALI
AKIŞKANLAR DİNAMİĞİ YÖNTEMLERİNİN KARŞILAŞTIRILMASI**

YÜKSEK LİSANS TEZİ

**Kayhan ÜLGEN
(508131011)**

Gemi İnşaatı ve Gemi Makineleri Mühendisliği Anabilim Dalı

Gemi İnşaatı ve Gemi Makineleri Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Şakir BAL

ŞUBAT 2017

Kayhan ÜLGEN, a M.Sc. student of ITU Graduate School of Science Engineering and Technology 508131011 successfully defended the thesis entitled “COMPARISON BETWEEN BLADE ELEMENT MOMENTUM THEORY AND COMPUTATIONAL FLUID DYNAMICS METHODS FOR PERFORMANCE PREDICTION OF MARINE PROPELLERS”, which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Prof. Dr. Şakir BAL**
Istanbul Technical University

Jury Members : **Prof. Dr. Şakir BAL**
Istanbul Technical University

Y. Doç Dr. Uğur Oral ÜNAL
Istanbul Technical University

Prof. Dr. Fahri ÇELİK
Yıldız Technical University

Date of Submission : **02 December 2016**

Date of Defense : **04 January 2017**





To my parents,



FOREWORD

I would like to thank my supervisor Prof. Dr. Şakir BAL for his guidance, helpful comments, suggestions, improvements to my thesis. I would like to thank my mentors Fatih ERTINAZ and Metin ŞAYLAN for their guidances, supports and advices. The last but not least, I would like to express my appreciation to my parents for their endless support.

February 2017

Kayhan ÜLGÜN
Naval Architecture and Marine Engineer



TABLE OF CONTENTS

	<u>Page</u>
FOREWORD.....	ix
TABLE OF CONTENTS.....	xi
ABBREVIATIONS	xiii
SYMBOLS.....	xv
LIST OF TABLES	xvii
LIST OF FIGURES	xix
SUMMARY	xxiii
ÖZET	xxv
1. INTRODUCTION	1
2. BLADE ELEMENT MOMENTUM THEORY.....	3
2.1 Momentum Theory	3
2.2 Blade Element Theory	6
2.3 Benini' s Approximation	9
2.4 McCormick's Approximations	11
2.4.1 Linear, small angle solution algorithm	11
2.5 Goldstein Tachmindji Circulation Correlation Factors	13
3. COMPUTATIONAL FLUID DYNAMICS METHOD	15
3.1 Governing Equations For Fluid Dynamics	15
3.1.1 Continuity equation	15
3.1.2 Momentum equations	16
3.2 Turbulence	17
3.2.1 Turbulence modeling	18
3.2.2 Reynolds averaged Navier-Stokes equations.....	18
3.2.2.1 Zero equation models	20
3.2.2.2 One equation models	20
3.2.2.3 Two equation models	21
3.2.3 Menter's shear stress transport model	21
3.2.4 Near-wall treatment	23
3.3 Numerical Methods	25
3.3.1 The finite volume method.....	25
3.3.2 Finite volume discretization	26
3.3.3 Interpolation	27
3.3.4 Pressure-velocity coupling methods.....	28
3.3.5 Linear solvers	31
3.3.6 Meshing	31
3.4 Rotational Motion.....	34
3.4.1 Multiple Frames of Reference Theory	34

4. NUMERICAL RESULTS	39
4.1 Open Water Computations with Blade Element Momentum Theory.....	41
4.1.1 Numerical algorithm.....	41
4.1.2 Computation case setup.....	41
4.1.3 Computations and analysis.....	41
4.2 Open Water Computations with OpenFOAM	44
4.2.1 Mesh generation	44
4.2.2 Computation case setup.....	54
4.2.3 Computations and analysis.....	55
4.2.3.1 Open water computations with coarse grid generation.....	56
4.2.3.2 Open water computations with medium grid generation.....	67
4.2.3.3 Open water computations with fine grid generation.....	79
5. CONCLUSIONS AND RECOMMENDATIONS.....	95
REFERENCES.....	97
APPENDICES.....	101
APPENDIX A.1	103
APPENDIX B.1.....	110
APPENDIX B.2.....	112
APPENDIX B.3.....	113
APPENDIX B.4.....	114
APPENDIX B.5.....	114
APPENDIX B.6.....	115
APPENDIX B.7.....	115
APPENDIX B.8.....	115

ABBREVIATIONS

AR	: Aspect Ratio
App	: Appendix
BEMT	: Blade Element Momentum Theory
CFD	: Computational Fluid Dynamics
DTMB	: David Taylor Model Basin
ITTC	: International Towing Tank Conference
MRF	: Multiple Rotating Reference Frame
NACA	: National Advisory Committee for Aeronautics
OpenFOAM	: Open Field Operation and Manipulation
PISO	: Pressure Implicit with Splitting of Operators
PVL	: Propeller Vortex Lattice
RANS	: Reynolds Averaged Navier Stokes
RSM	: Reynolds Stress Model
SIMPLE	: Semi Implicit Pressure Linked Equations



SYMBOLS

C_D	: Drag Coefficient
C_L	: Lift Coefficient
$C_{L\alpha}$	: Linear lift curve slope
C_P	: Breaking Power Coefficient
C_{M_Z}	: Rotational Torque Coefficient
C_T	: Thrust Coefficient
c	: Chord length
c_{xr}	: Local blade chord length
dD	: Propeller section drag
dD	: Propeller section lift
dM_Z	: Incremental propeller torque generated by blade element
dP	: Incremental breaking power generated by blade element
dr	: Blade radius element
dT	: Incremental thrust on blade element
$d\tau$	: Incremental torquing force on blade element
J	: Advance Coefficient
J_i	: Local induced advance coefficient on blade element
$\dot{m}$	: Mass flow through actuator disk
N	: Number of blades
P	: Propeller breaking power
p	: Pressure value before actuator disk
p_∞	: Freestream pressure
R_p	: Blade-tip radius
r	: Local radial section along blade
T	: Propeller thrust
V_e	: Effective stream tube exit velocity
V_i	: Induced velocity
V_T	: Total kinematic velocity of blade section
V_∞	: Freestream velocity
x_r	: Nondimensional radial location
α	: Angle of attack
α_i	: Local induced angle of attack
β	: Blade pitch angle
β_i	: Local blade pitch angle
Δp	: Pressure difference across actuator disk
λ	: Pitch length
ρ	: Fluid density
Φ	: Pitch angle
Ω	: Angular velocity



LIST OF TABLES

	<u>Page</u>
Table 3.1 : The constants of SST model.....	23
Table 4.1 : DTMB 4119 propeller geometry.	39
Table 4.2 : Open water operating conditions.....	42
Table 4.3 : The comparison of blade element momentum theory and experimental results.	43
Table 4.4 : The comparison of relative numerical percentage errors between blade element momentum theory and experimental results.....	44
Table 4.5 : Mesh information.	54
Table 4.6 : Water properties under experimental conditions.....	54
Table 4.7 : The boundary conditions in open water condition.	55
Table 4.8 : The comparison of numerical and experimental results.....	66
Table 4.9 : The comparison of relative numerical percentage errors between computational fluid dynamics method and experimental results.	67
Table 4.10 : The comparison of numerical and experimental results.....	78
Table 4.11 : The comparison of relative numerical percentage errors between computational fluid dynamics method and experimental results.	78
Table 4.12 : The comparison of numerical and experimental results.....	89
Table 4.13 : The comparison of relative numerical percentage errors between computational fluid dynamics method and experimental results.	90
Table 4.14 : The comparison of numerical and experimental results.....	93



LIST OF FIGURES

	<u>Page</u>
Figure 2.1 : Actuator disk theory (Whitmore and Merrill, 2012).	3
Figure 2.2 : Blade element theory (Whitmore and Merrill, 2012).	6
Figure 3.1 : Law of the wall.	24
Figure 3.2 : Parameters in finite volume discretisation [53].	25
Figure 3.3 : Mesh orthogonality.	32
Figure 3.4 : Mesh skewness.	32
Figure 3.5 : Expansion rate.	33
Figure 3.6 : Mesh aspect ratio.	33
Figure 3.7 : Representation of two reference frames in Cartesian coordinate system.	34
Figure 4.1 : Perspective view of DTMB 4119 propeller.	40
Figure 4.2 : Radial sections of DTMB 4119 propeller blade.	40
Figure 4.3 : C_T and C_P values using BEMT solution.	42
Figure 4.4 : Comparison of K_T , $10K_Q$ and η_O values using BEMT solution with experimental data.	43
Figure 4.5 : Computational domain.	44
Figure 4.6 : Computational static domain.	45
Figure 4.7 : Computational rotating domain.	45
Figure 4.8 : Computational coarse mesh in Numeca Hexpress/Hybrid.	46
Figure 4.9 : Computational coarse mesh around propeller blades, hub and shaft in Numeca Hexpress/Hybrid.	47
Figure 4.10 : Computational coarse mesh around propeller blades in Numeca Hexpress/Hybrid.	47
Figure 4.11 : Viscous layers at leading edge of the propeller.	48
Figure 4.12 : Viscous layers at trailing edge of the propeller.	48
Figure 4.13 : Computational medium mesh in Numeca Hexpress/Hybrid.	49
Figure 4.14 : Computational medium mesh around propeller blades, hub and shaft in Numeca Hexpress/Hybrid.	49
Figure 4.15 : Computational medium mesh around propeller blades in Numeca Hexpress/Hybrid.	50
Figure 4.16 : Viscous layers at leading edge of the propeller.	50
Figure 4.17 : Viscous layers at trailing edge of the propeller.	51
Figure 4.18 : Computational fine mesh in Numeca Hexpress/Hybrid.	51
Figure 4.19 : Computational fine mesh around propeller blades, hub and shaft in Numeca Hexpress/Hybrid.	52
Figure 4.20 : Computational fine mesh around propeller blades in Numeca Hexpress/Hybrid.	52
Figure 4.21 : Viscous layers at leading edge of the propeller.	53

Figure 4.22: Viscous layers at trailing edge of the propeller.....	53
Figure 4.23: Convergence graphs for $J = 0.1$	56
Figure 4.24: Convergence graphs for $J = 0.2$	56
Figure 4.25: Convergence graphs for $J = 0.3$	56
Figure 4.26: Convergence graphs for $J = 0.4$	57
Figure 4.27: Convergence graphs for $J = 0.5$	57
Figure 4.28: Convergence graphs for $J = 0.6$	57
Figure 4.29: Convergence graphs for $J = 0.7$	58
Figure 4.30: Convergence graphs for $J = 0.8$	58
Figure 4.31: Convergence graphs for $J = 0.833$	58
Figure 4.32: Convergence graphs for $J = 0.9$	59
Figure 4.33: Convergence graphs for $J = 1.0$	59
Figure 4.34: Convergence graphs for $J = 1.1$	59
Figure 4.35: Distribution of pressure coefficient on the propeller for $J = 0.1$	60
Figure 4.36: Distribution of pressure coefficient on the propeller for $J = 0.2$	60
Figure 4.37: Distribution of pressure coefficient on the propeller for $J = 0.3$	61
Figure 4.38: Distribution of pressure coefficient on the propeller for $J = 0.4$	61
Figure 4.39: Distribution of pressure coefficient on the propeller for $J = 0.5$	62
Figure 4.40: Distribution of pressure coefficient on the propeller for $J = 0.6$	62
Figure 4.41: Distribution of pressure coefficient on the propeller for $J = 0.7$	63
Figure 4.42: Distribution of pressure coefficient on the propeller for $J = 0.8$	63
Figure 4.43: Distribution of pressure coefficient on the propeller for $J = 0.833$...	64
Figure 4.44: Distribution of pressure coefficient on the propeller for $J = 0.9$	64
Figure 4.45: Distribution of pressure coefficient on the propeller for $J = 1.0$	65
Figure 4.46: Distribution of pressure coefficient on the propeller for $J = 1.1$	65
Figure 4.47: Comparison of K_T , $10K_Q$ and η_O values using CFD solution for coarse grid with experimental data.	66
Figure 4.48: Convergence graphs for $J = 0.1$	67
Figure 4.49: Convergence graphs for $J = 0.2$	67
Figure 4.50: Convergence graphs for $J = 0.3$	68
Figure 4.51: Convergence graphs for $J = 0.4$	68
Figure 4.52: Convergence graphs for $J = 0.5$	68
Figure 4.53: Convergence graphs for $J = 0.6$	69
Figure 4.54: Convergence graphs for $J = 0.7$	69
Figure 4.55: Convergence graphs for $J = 0.8$	69
Figure 4.56: Convergence graphs for $J = 0.833$	70
Figure 4.57: Convergence graphs for $J = 0.9$	70
Figure 4.58: Convergence graphs for $J = 1.0$	70
Figure 4.59: Convergence graphs for $J = 1.1$	71
Figure 4.60: Distribution of pressure coefficient on the propeller for $J = 0.1$	71
Figure 4.61: Distribution of pressure coefficient on the propeller for $J = 0.2$	72
Figure 4.62: Distribution of pressure coefficient on the propeller for $J = 0.3$	72
Figure 4.63: Distribution of pressure coefficient on the propeller for $J = 0.4$	73
Figure 4.64: Distribution of pressure coefficient on the propeller for $J = 0.5$	73
Figure 4.65: Distribution of pressure coefficient on the propeller for $J = 0.6$	74
Figure 4.66: Distribution of pressure coefficient on the propeller for $J = 0.7$	74

Figure 4.67: Distribution of pressure coefficient on the propeller for $J = 0.8$	75
Figure 4.68: Distribution of pressure coefficient on the propeller for $J = 0.833$...	75
Figure 4.69: Distribution of pressure coefficient on the propeller for $J = 0.9$	76
Figure 4.70: Distribution of pressure coefficient on the propeller for $J = 1.0$	76
Figure 4.71: Distribution of pressure coefficient on the propeller for $J = 1.1$	77
Figure 4.72: Comparison of K_T , $10K_Q$ and η_O values using CFD solution for medium grid with experimental data.	77
Figure 4.73: Convergence graphs for $J = 0.1$	79
Figure 4.74: Convergence graphs for $J = 0.2$	79
Figure 4.75: Convergence graphs for $J = 0.3$	79
Figure 4.76: Convergence graphs for $J = 0.4$	80
Figure 4.77: Convergence graphs for $J = 0.5$	80
Figure 4.78: Convergence graphs for $J = 0.6$	80
Figure 4.79: Convergence graphs for $J = 0.7$	81
Figure 4.80: Convergence graphs for $J = 0.8$	81
Figure 4.81: Convergence graphs for $J = 0.833$	81
Figure 4.82: Convergence graphs for $J = 0.9$	82
Figure 4.83: Convergence graphs for $J = 1.0$	82
Figure 4.84: Convergence graphs for $J = 1.1$	82
Figure 4.85: Distribution of pressure coefficient on the propeller for $J = 0.1$	83
Figure 4.86: Distribution of pressure coefficient on the propeller for $J = 0.2$	83
Figure 4.87: Distribution of pressure coefficient on the propeller for $J = 0.3$	84
Figure 4.88: Distribution of pressure coefficient on the propeller for $J = 0.4$	84
Figure 4.89: Distribution of pressure coefficient on the propeller for $J = 0.5$	85
Figure 4.90: Distribution of pressure coefficient on the propeller for $J = 0.6$	85
Figure 4.91: Distribution of pressure coefficient on the propeller for $J = 0.7$	86
Figure 4.92: Distribution of pressure coefficient on the propeller for $J = 0.8$	86
Figure 4.93: Distribution of pressure coefficient on the propeller for $J = 0.833$...	87
Figure 4.94: Distribution of pressure coefficient on the propeller for $J = 0.9$	87
Figure 4.95: Distribution of pressure coefficient on the propeller for $J = 1.0$	88
Figure 4.96: Distribution of pressure coefficient on the propeller for $J = 1.1$	88
Figure 4.97: Comparison of K_T , $10K_Q$ and η_O values using CFD solution for fine grid with experimental data.	89
Figure 4.98: Distribution of pressure coefficient on the propeller for $J = 0.6$ using PVL code (Bal, 2011a).....	90
Figure 4.99: Distribution of pressure coefficient on the propeller for $J = 0.6$ using OpenFOAM.....	91
Figure 4.100 Distribution of pressure coefficient on the propeller for $J = 0.8$ using PVL code (Bal, 2011b).....	91
Figure 4.101 Distribution of pressure coefficient on the propeller for $J = 0.8$ using OpenFOAM.....	92
Figure 4.102 Comparison of K_T , $10K_Q$ and η_O values using BEMT, CFD, PVL solutions with experimental data.	92



COMPARISON BETWEEN BLADE ELEMENT MOMENTUM THEORY AND COMPUTATIONAL FLUID DYNAMICS METHODS FOR PERFORMANCE PREDICTION OF MARINE PROPELLERS

SUMMARY

This thesis illuminates the comparison of blade element momentum theory and computational fluid dynamics method on marine propellers. For this purpose, on the first part of the thesis, blade element momentum theory algorithm is implemented using MATLAB and XFOIL softwares. In order to fulfill verification and validation of blade element momentum theory algorithm, the practical analysis technique is applied to DTMB 4119 propeller, for which hydrodynamic characteristics and open water results are given in the literature.

First, propeller blade geometry is subdivided into finite blade elements using piecewise cubic Hermitian interpolation method. According to Benini's studies on blade element momentum theory, a tentative value for axial inflow factor is specified for the first iteration based on innermost or outermost section. Second, angular inflow factor (swirl factor) can be computed using ideal blade element efficiency. Slipstream and axial velocity components are computed using these factors. Pitch angles are obtained using pitch distribution along the blade. Angle of attack values are calculated using the difference between geometric and hydrodynamic pitch angles. Lift and drag coefficients per blade element could be computed using angle of attack, local chord distribution and local Reynolds number values. Third, thrust and torque effects per blade element are defined and corrected by Goldstein-Tachmindji correlation factors. Then, new axial inflow factor value is computed and this loop is kept on the difference between old and new axial inflow factors are reach up to better tolerance. Finally, obtained thrust and torque values for each radial strip are radially integrated over the blade.

According to McCormick's linear, small angle approximation, a tentative value for induced angle of attack is specified for the first iteration. After the normalized induced velocity is computed, the new value of local induced angle of attack is calculated. This algorithm is iterated until the changes in the computed induced angle of attack become sufficiently small. Finally, thrust and power coefficients are computed when the convergence is reached.

On the second part of the thesis, incompressible and viscous fluid flow around DTMB 4119 propeller is discretised using computational fluid dynamics method. On this study, open water simulations are implemented using OpenFOAM which is an open source, object oriented library for numerical computations in continuum mechanics, written in the C++ programming language.

In order to perform open water computations using multiple reference frame technique, physical domain is constructed by rotating and static subdomains according to ITTC recommendations. Both subdomains are imported into Numeca Hexpress/Hybrid software and grid generation is done separately. First, an isotropic Cartesian block

grid is generated around the geometries. Second, the grid is refined in refinement regions by splitting the initial cell volumes. To avoid numerical errors arising from interpolation between non-conforming arbitrary mesh interfaces, the refinement cell sizes on interfaces are obtained approximately equal. Then, the volumic grid is snapped onto the model. Finally, so as to capture viscous effects, viscous layers are inserted according to the foreseen y^+ value.

SST $k-\omega$ turbulence model is chosen for near-wall treatment and effects of various turbulence models are neglected. Grid dependency is taken into account in order to satisfy open-source CFD code algorithm. Coarse, medium and fine mesh levels are specified according to the refinement levels by terms of propeller diameter.

Owing to the fact that steady time discretization scheme is chosen, SIMPLE (Semi-Implicit Pressure Linked Equation) formulation with consistent algorithm is used to improve convergence, accuracy and solution time.

Finally, pressure coefficient distributions on specified sections from blade element momentum theory and OpenFOAM are compared with experimental results. Besides, numerically computed open water performance values are compared with PVL and experimental results. However, in this study wake adaption, nonlinear blade element momentum theory solutions and cavitation effects are neglected.

GEMİ PERVANELERİNİN PERFORMANSININ BELİRLENMESİNDE KANAT ELEMANI MOMENTUM TEORİSİ VE HESAPLAMALI AKIŞKANLAR DİNAMİĞİ YÖNTEMLERİNİN KARŞILAŞTIRILMASI

ÖZET

Bu çalışmada kanat elemanı momentum teorisi ve hesaplamalı akışkanlar dinamiği yöntemleri kullanılarak, gemi pervanelerinin açık su performanslarının belirlenmesi üzerine çalışmalar gerçekleştirilmiştir.

Kanat elemanı momentum teorisi, momentum ve kanat elemanı teorilerinin birleşiminden oluşmaktadır. Momentum teorisi, akım tüpü içerisindeki sonsuz sayıda kanatlı bir pervane olarak ele alınan pervane diski eksenindeki ve akım yönündeki, hız ve basınç değerlerinin, enerjinin ve eksenel momentumun korunumu yasalarına bağlı olarak hesaplanabilmesini sağlamaktadır. Bu teoriye göre akım, viskoz ve rotasyonel olmayan, sıkıştırılmaz olarak incelenmektedir. Bu sebeple enerji kayıpları ihmal edilmektedir. Bunun yanı sıra momentum teorisi, burulma etkilerini göz önünde bulundurmamakta ve dolayısıyla teğetsel etkileri ihmal etmektedir. Bunun bir sonucu olarak, itme etkilerinin pervane diski üzerinde düzgün dağıldığı görülmektedir. Ele alınan tüm bu ideal koşullara rağmen, momentum teorisinin en önemli sonucu, pervane diski üzerindeki indüklenmiş hızların, sonsuz uzunluktaki akım tüpü çıkışındaki hızların yarısı kadar olacağıdır.

Kanat elemanı teorisinin temel amacı, pervane kanatlarının radyal kesit performans değerlerinden faydalanarak, pervane itme, tork ve ilerleme katsayısına bağlı olarak açık su verim ifadelerinin hesaplanmasıdır. Bu teoride radyal kanat kesitleri üzerine etkiyen bileşke hız değerlerinden faydalanılır. Bu sebeple, geometrik piç açısı, hidrodinamik piç açısı ve hücum açısı değerlerinin doğru tespit edilmesi gereklidir. Kanat elemanı teorisinin ön önemli özelliği açisal momentumun korunumu ilkesine bağlı kalmasıdır. Ancak bu teoride, pervane kanatlarının birbirleri üzerine olan etkisi, kanatlar etrafında oluşan girdapların hızları ve giriş açısında olan etkileri ihmal edildiğinden, pervane açık su performansının eldesinde tek başına yetersiz kaldığı görülmektedir.

İhmal edilen fiziksel koşulların yanı sıra, bu fiziksel koşulların matematiksel modellenmesi sırasında ihmal edilen etkilerin de göz önünde bulundurulması sonucu ortaya atılan kanat elemanı momentum teorisi, momentum ve kanat elemanı teorilerinin eksik yanlarını kısmen de olsa kapatmaktadır. İhmal edilen kayıplar Goldstein sirkülasyon düzeltme katsayıları ile hesaplanabilmektedir. Bununla birlikte pervane kanat kesitleri üzerindeki indüklenen hızlar, hücum açıları ve lokal viskoz etkiler göz önünde bulundurulabilir.

Bu nedenlerle, çalışmanın ilk bölümünde kanat elemanı momentum teorisi, MATLAB ve XFOIL yazılımları kullanılarak geliştirilmiştir. Önce, verilen pervane kanadı, parçalı kübik Hermite interpolasyonu yöntemi ile sonlu kanat elemanlarına bölünmüştür. En içteki veya en dıştaki kanat elemanı için eksenel akım faktörüne yaklaşık olarak bir değer atanmıştır. Daha sonra teğetsel akım faktörü ideal kanat

elemanı verimi ifade kullanılarak belirlenmiştir. Teğetsel ve eksenel hız bileşenleri bu faktörler kullanılarak hesaplanmıştır. Hatve açıları, pervane geometrisi baz alınarak hesaplanmıştır. Böylece geometrik ve hidrodinamik hatve açıları arasındaki farktan her bir kanat kesidine gelen akımın hücum açılarına ulaşılmıştır. Radyal kanat kesidi için kaldırma ve sürüklenme katsayıları XFOIL yazılımı kullanılarak hesaplanmış, bu değerler Goldstein faktörleri ile düzeltilip radyal integrasyon ile pervanenin itme ve tork değerleri hesaplanmıştır. Sonrasında, yeni bir eksenel indüklenme katsayısı hesaplanmış ve iteratif yöntem yeterli tolerans değerine ulaşınca dek sürdürülmüştür. Bu iteratif yöntem ile her bir radyal kesitteki itme ve tork değerleri kanat üzerinde integrasyon işlemi gerçekleştirilerek elde edilmiştir.

Gemi pervanelerinin açık su performansının belirlenmesinde kullanılan yöntemlerden biri de hesaplamalı akışkanlar dinamiği yöntemleridir. Hesaplamalı akışkanlar dinamiği yöntemleri, temel olarak, akışkanlar dinamiğini tanımlayan kısmi diferansiyel denklemlerin, doğrusal cebirsel denklemlere dönüştürülmesi, bu denklemlerin matris sistemleri halinde yapılandırılması ve bu sistemlerin iteratif olarak çözümlenmesine dayanmaktadır. Bu yapılandırma esnasında sonlu farklar, sonlu elemanlar, sonlu hacimler, sınır elemanları, spektral elemanlar, süresiz Galerkin yöntemi vb. yöntemler kullanılmaktadır. Burada, sınır elemanları yöntemi dışında kullanılan tüm yöntemlerde katsayılar matrisinin seyrek matris, diagonal olarak baskın ve iyi konumlanmış olması gerekmektedir.

Günümüzde hesaplamalı akışkanlar dinamiği yöntemleri oldukça gelişmiş olup, yüksek performanslı ve kapasiteli bilgisayarlar sayesinde akışkan hareketini tanımlayan kısmi diferansiyel denklemlerin ortalama olarak veya doğrudan çözümüne olanak sağlamaktadır. Bu çözümler, fiziksel ortam koşullarının matematiksel olarak ifade edilmesi teorisine dayanır. Bu sebeple akışkan denklemlerinin çözülebilmesi için doğal ve asıl sınır şartlarının doğru bir şekilde probleme dayatılması gerekmektedir. Aksi takdirde ele alınan problemin matematiksel ve fiziksel olarak doğruluk payı yüksek bir şekilde hesaplanması mümkün değildir.

Hesaplamalı akışkanlar dinamiği yöntemleri ile fiziksel olayların matematiksel modellenebilmesi için ticari veya açık kaynak kodlu yazılımlar kullanılmaktadır. Açık kaynak kodlu hesaplamalı akışkanlar dinamiği kütüphanelerinden biri olan OpenFOAM, C++ dilinde yazılmış, sürekli ortamlar mekaniği alanındaki nümerik uygulamalar için oluşturulmuş bir kütüphanedir. Bu kütüphanenin temeli, hücre temelli sonlu hacimler yöntemine dayanmaktadır. Açık kaynak kodlu olması, lisans ücreti gerektirmemesi, mümkün olan en yüksek işlemci sayısına kadar paralel bilgisayarlarda nümerik hesaplamalar yapılmasına olanak sağlayan OpenFOAM kütüphanesi, kod geliştirmeye izin veren ancak GPL lisanslı olması bakımından geliştirilen kodun tüm kullanıcılarla paylaşımını zorunlu kılmaktadır. Bu kütüphanenin tek dezavantajı, kısa zamanda profesyonel olarak kullanılabilmesi için literatürde yeterli derecede kaynak bulunamamasıdır.

Gemi pervanelerinin açık su performans değerlerinin eldesi için üç farklı yöntem kullanılmaktadır. Bunlar; tekli dönen eksen takımı yöntemi, çoklu dönen eksen takımı yöntemi ve kayar çözüm ağı yöntemleridir. Tekli dönen eksen takımı yöntemine göre akışkan hacmi içerisindeki tüm hücrelere ivme verilerek, basınç-hız ve türbülans değişkenleri hesaplanır. Bu yöntemde ele alınan hızlar, mutlak referans noktasına göre göreceli hızlardır. Çoklu dönen eksen takımı yönteminde ise, akışkan hacmi Coriolis etkilerinin hesaba katılacağı bir iç hacim ve dış hacimden meydana gelmekte olup,

yalnızca bu iç ve dış hacimler arasındaki ara yüzeyler arasında lineer interpolasyon yapılarak çözüm sağlanır. Kayar çözüm ağı yönteminde ise, dinamik çözüm ağı kullanılarak, pervanenin zamana bağlı olarak hareket ettirilmesi ile akışkan hacmi içerisindeki hız ve türbülans değerlerinin gerçek değerleri fiziksel olarak doğruluk payı yüksek bir şekilde hesaplanmış olur.

Bunun yanı sıra, düzenli veya düzensiz, daimi veya daimi olmayan zaman ayırıştırma yöntemlerine bağlı olarak geliştirilen basınç-hız bağlaşım yöntemleri mevcuttur. Bunlar temelde SIMPLE, PISO ve PIMPLE algoritmalarıdır. SIMPLE algoritması, akışkan denklemlerinin daimi olarak çözülmesine olanak sağlar. Bu sebeple akışkan hacmi içerisinde meydana gelen değişimleri incelemekte yetersiz kalmasına karşın, nümerik olarak hızlı ve yakın bir sonuç sunar. PISO algoritması ise daimi olmayan akımların modellenmesi amacı ile kullanılan, ancak nümerik difüzyonun yüksek olması durumunda istenilen sonucu vermekte yetersiz kalan bir algoritmadır. Bu sebeple PISO algoritması kullanılırken Courant-Friedrichs-Lewy şartına dikkat edilmesi gereklidir. Bunların dışında kullanılan PIMPLE algoritması ise, akışkan hareketini modellerken yüksek zaman adımı kullanmakta ve bu nümerik difüzyondan meydana gelebilecek hataları önlemek amacıyla iterasyon sayısına bağlı olarak hesaplamaktadır. Böylece akışkan daimi olarak modellenmemesine karşın, nümerik çözümün daimi akıma yaklaştığı, stabilitenin ve doğruluk payının yüksek ölçüde sağlandığı görülmüş olur.

Akışkanlar, daimi ve düzenli olduğunda laminar, düzensiz ve daimi olmadığında ise türbülanslı olarak kabul edilirler. Akışkanın laminar veya türbülanslı olması ise Reynolds sayısı ile göreceli olarak belirlenebilir. Bu sebeple akışkan içerisinde meydana gelen türbülans değişimlerinin incelenebilmesi için bir çok model geliştirilmiştir. Geliştirilen bu modeller, gerçekte fiziksel olayları ayırklaştırmada yetersiz kalmalarına karşın, doğruluk payı oldukça yüksek çözümler sunmaktadırlar. Temelde bu modeller; doğrudan nümerik simülasyon (DNS), büyük girdap simülasyonu (LES), Reynolds ortalama Navier-Stokes (RANS) ve Reynolds gerilme modeli (RSM) olmak üzere dörde ayrılmaktadır. Doğrudan nümerik simülasyon yöntemi ile Navier-Stokes denklemleri doğrudan çözülür ve türbülans modeli kullanılmaz. Büyük girdap simülasyonlarında ise, büyük ölçekteki girdaplar çözümlenirken, küçük ölçekteki girdaplar modellenir. Reynolds ortalama Navier-Stokes denklemleri ile akışkan hareketi zaman ortalama terimler baz alınarak hesaplanır. Reynolds gerilme yöntemi ile de, Reynolds gerilme tensörlerinin doğru bir şekilde çözümlenmesi sağlanmış olur.

Bahsedilen bilgiler doğrultusunda çalışmanın ikinci bölümünde ise DTMB 4119 pervanesinin etrafındaki akış alanı hesaplamalı akışkanlar dinamiği yöntemi ile ayırklaştırılmıştır. Bu hesaplamalar esnasında, dönen eksen yöntemi tekniği kullanılmıştır. Yakın duvar yaklaşımı için SST $k-\omega$ türbülans modeli seçilmiş ve diğer türbülans modellerinin etkileri ihmal edilmiştir. Çözüm ağının yeterince sık olmamasından dolayı meydana gelebilecek etkiler de ihmal edilmiş olup, pervane etrafındaki çözüm ağı yeterince sıklaştırılmış böylece duvar fonksiyonu kullanılarak kayma gerilmelerinin hesaplanması sağlanmaya çalışılmıştır. Bunun yanı sıra çözüm ağı bağımsızlık analizleri de gerçekleştirilmiştir. Daimi dönen eksen tekniğinin kullanıldığı akış analizleri esnasında, denklem sisteminin stabilitesinin sağlanabilmesi için SIMPLE-Consistent algoritması kullanılmıştır. Pervane etrafındaki basınç katsayısı dağılımı ve performans katsayıları hesaplanmış olup deney sonuçları ile karşılaştırmalı olarak sunulmuştur. Ayrıca itme ve tork katsayıları da hesaplanmıştır.



1. INTRODUCTION

Prediction of performance characteristics of marine propellers is one of the most complex analytical problem in fluid mechanics. Owing to the advances of execution speed, memory allocation and capacity of modern super computers day after day, computational fluid dynamics method is generally preferred. Despite of the fact that numerical techniques in computational fluid dynamics have been growing rapidly day by day, selection of boundary conditions, turbulence models and numerical schemes has significant influence on accuracy and stability. Therefore grid dependency is one of the most important problem in computational fluid dynamics, grid resolution has to be fine enough to specify wall shear stress terms and consequently, this increases computational cost. Due to these reasons, numerical methods which are developed for propeller analysis are applied on conceptual design stage.

Blade element momentum theory is one of these theories which is developed for propeller design and analysis. This theory is combination of actuator disk and blade element theories. While actuator disk theory is based on conservation of axial and angular momentum, blade element theory is hinge upon performance characteristics of local foil sections. Therefore, blade element momentum theory neglects three dimensional effects. In spite of the fact that three dimensional effects are neglected, circulation distribution around blade sections are corrected using Goldstein's approximation. Thus, total thrust and torque coefficients which are predicted from a radial section could be corrected and finally integrated radially over the blade. It shows that, this method is quite applicable during design procedure.

Computational fluid dynamics method is one of the commonly used approximation during analysis procedure. According to this method, flow domain around the propeller is subdivided into grids and governing equations for fluid dynamics are solved iteratively with finite volume discretisation over cell centers using numerical schemes. However, in order to discretise fluid flow motion accurately, turbulence modelling must be predicted definitely. Thus, Navier-Stokes equations are solved

coupled with turbulence equations. Reynolds-averaged Navier-Stokes equations are widely used in practise. This approach is based on decomposition of an instantaneous quantity into time-averaged and fluctuating parts. Thereby, eddies are discretised accurate enough.

As it is seen, blade element momentum approximation is less expensive way to find out propeller performance. Due to these reasons, blade element momentum and computational fluid dynamics approaches are both applied on DTMB 4119 propeller comparatively. Obtained performance results and pressure distribution along specified blade sections are compared with experimental results. Consequently, computational results are approximated accurate enough.



2. BLADE ELEMENT MOMENTUM THEORY

2.1 Momentum Theory

Momentum theory was first introduced by Rankine at 1865. Rankine's theory was based on axial motion of the water passing through the propeller disc. There are some assumptions had been considered, as follows Carlton (2007):

1. The propeller works in an ideal fluid and, therefore, does not experience energy losses due to frictional drag.
2. The propeller can be replaced by an actuator disc, and this is equivalent to saying that the propeller has an infinite number of blades.
3. The propeller can produce twist without causing rotation in the slipstream.

In brief, the flow is considered as inviscid, irrotational and incompressible.

Third assumption was removed by Froude's studies and actuator disk is allowed to produce rotational velocity to the slipstream (Carlton, 2007).

The most important result of this theory is that induced velocity at the propeller disk equals to the half of the induced velocity at the far downstream of the propeller.

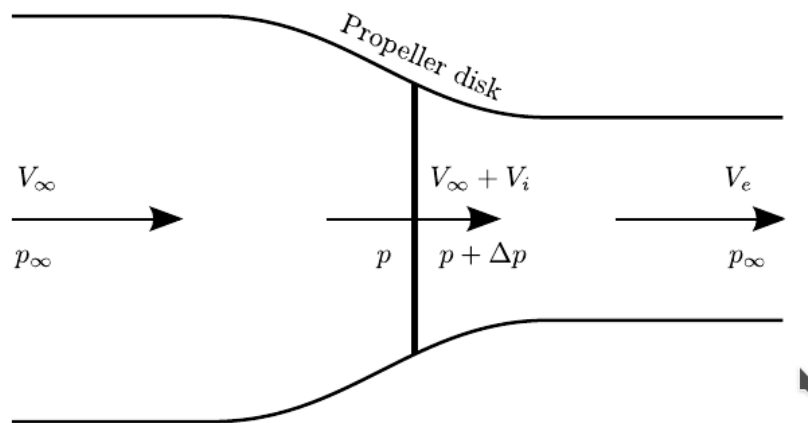


Figure 2.1 : Actuator disk theory (Whitmore and Merrill, 2012).

Let us write Bernoulli's equation between a point far upstream and another point far downstream respectively,

$$p_{\infty} + \frac{1}{2}\rho V_{\infty}^2 = p + \frac{1}{2}\rho (V_{\infty}^2 + \delta V_i^2) \quad (2.1)$$

$$p + \Delta p + \frac{1}{2}\rho (V_{\infty}^2 + \delta V_i^2) = p_{\infty} + \frac{1}{2}\rho (V_e^2) \quad (2.2)$$

where the differential pressure is expressed by,

$$\Delta p = \frac{1}{2}\rho (V_e^2 - V_{\infty}^2) \quad (2.3)$$

The preceding equation shows that differential pressure is independent of radius. The local differential thrust on the actuator disk could be expressed as,

$$dT = 2\pi \frac{1}{2} (V_e^2(r) - V_{\infty}^2) r dr = \pi \rho (V_e^2(r) - V_{\infty}^2) r dr \quad (2.4)$$

Let us perform the momentum balance across the actuator disk respectively,

$$dT = \dot{m}_{(r)} (V_e(r) - V_{\infty}) = \rho (V_{\infty} + \delta V_i) 2\pi r dr (V_e(r) - V_{\infty}) \quad (2.5)$$

Let us equalize equations 2.4 and 2.5, the outlet velocity could be expressed in terms of induced inflow velocity below,

$$V_e(r) = V_{\infty} + 2\delta V_i \quad (2.6)$$

Substituting equation 2.6 into equation 2.4 solving the thrust increment below,

$$\begin{aligned} dT &= \pi \rho \left((V_{\infty} + 2\delta V_i)^2 - V_{\infty}^2 \right) r dr \\ &= \pi \rho (V_{\infty}^2 + 4V_{\infty}\delta V_i + 4\delta V_i^2 - V_{\infty}^2) r dr \\ &= 4\pi \rho (V_{\infty}\delta V_i + \delta V_i^2) r dr \end{aligned} \quad (2.7)$$

Let us collect δV_i terms in equation 2.7:

$$\delta V_i^2 + V_\infty \delta V_i - \frac{dT}{4\pi\rho r dr} = 0 \quad (2.8)$$

and solving equation 2.8 for the local induced velocity gives:

$$\delta V_i = -\frac{V_\infty}{2} \pm \sqrt{\left(\frac{V_\infty}{2}\right)^2 + \frac{dT}{4\pi\rho r dr}} \quad (2.9)$$

So that the induced velocity must be positive, only the positive term is considered. Let us non-dimensionalize the induced velocity:

$$\begin{aligned} J_i(r) &\equiv \frac{\delta V_i}{[(\omega/\pi)R_p]} = -\frac{V_\infty/2}{[(\omega/\pi)R_p]} \\ &+ \sqrt{\left[\frac{V_\infty/2}{[(\omega/\pi)R_p]}\right]^2 + \frac{dT}{4\pi\rho r dr (\omega/2\pi)^2 (2R_p)^2}} \\ &= -\frac{J}{2} + \sqrt{\left(\frac{J}{2}\right)^2 + \frac{1}{4\pi(r/2R_p)(dr/2R_p)\rho(\omega/2\pi)^2(2R_p)^4} \frac{dT}{dr}} \\ &= -\frac{J}{2} + \sqrt{\left(\frac{J}{2}\right)^2 + \frac{1}{\pi x_r} \cdot \frac{dC_T}{dx_r}} \end{aligned} \quad (2.10)$$

Consequently, the induced angle of attack could be given by:

$$\begin{aligned} \alpha_i(r) &= \tan^{-1} \left(\frac{\delta V_i(r)}{\sqrt{V_\infty^2 + (\omega r)^2}} \right) \\ &= \tan^{-1} \left(\frac{\delta V_i(r)/\omega R_p}{\sqrt{(V_\infty/\omega R_p)^2 + (\omega r/\omega R_p)^2}} \right) \\ &= \tan^{-1} \left(\frac{J_i(r)/\pi}{\sqrt{(J/\pi)^2 + x_r^2}} \right) \\ &= \tan^{-1} \left(\frac{J_i(r)}{\sqrt{J^2 + (\pi x_r)^2}} \right) \end{aligned} \quad (2.11)$$

2.2 Blade Element Theory

According to the blade element theory, propeller blade is subdivided into a number of radial sections along the radius and at each radial section, a force balance is applied.

Since the propeller blade will be set at a given geometric pitch angle β , the local velocity vector will create a flow angle of attack on the section. Lift and drag of the section can be calculated using standard two dimensional airfoil properties.

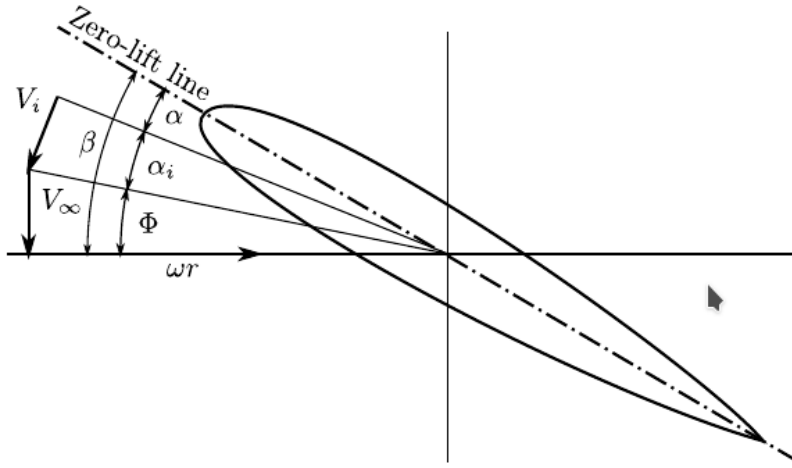


Figure 2.2 : Blade element theory (Whitmore and Merrill, 2012).

The difference in angle between thrust and lift directions is defined as:

$$\Phi + \alpha_i = \beta - \alpha \quad (2.12)$$

The elemental thrust and torque forces of this blade element could be written as:

$$dT = dL \cos(\Phi + \alpha_i) - dD \sin(\Phi + \alpha_i) \quad (2.13)$$

$$d\tau = dD \cos(\Phi + \alpha_i) + dL \sin(\Phi + \alpha_i) \quad (2.14)$$

The rotational torque at each radial section along the blade is given by:

$$dM_z = d\tau \cdot r \quad (2.15)$$

The ideal propeller shaft power required to overcome hydrodynamic effect acting on the radial section is given by:

$$dP = d\tau \cdot r \cdot \omega = r \cdot \omega \cdot [dL \sin(\Phi + \alpha_i) + dD \cos(\Phi + \alpha_i)] \quad (2.16)$$

The local lift and drag effecting on the two dimensional blade section is given by:

$$\frac{dL}{dr} = \left(\frac{1}{2} \rho V_T^2 \right) C_{Lc(r)} = \frac{1}{2} \rho \left(V_\infty^2 + (\omega r)^2 \right) C_{Lc(r)} \quad (2.17)$$

$$\frac{dD}{dr} = \left(\frac{1}{2} \rho V_T^2 \right) C_{Dc(r)} = \frac{1}{2} \rho \left(V_\infty^2 + (\omega r)^2 \right) C_{Dc(r)} \quad (2.18)$$

where ρ is the fluid density and $c_{(r)}$ is the radial section chord.

According to blade element theory, it is difficult to define axial and rotational velocities at the propeller disk. The axial velocity roughly equal to the advance velocity of the propeller but at the propeller disk, advance velocity increases due to propeller's own induced velocity into a slipstream. Rotational velocity at the propeller disk is roughly equal to the radial sections' rotational velocity but it is reduced due to the swirling effect. To compute velocity components at the propeller disk, conservation of angular momentum has to be satisfied.

Let us substitute equation 2.17 and 2.18 into equation 2.13 and 2.14:

$$\frac{dT}{dr} = \frac{1}{2} \rho \left(V_\infty^2 + (\omega r)^2 \right) c_{(r)} [C_L \cos(\Phi + \alpha_i) - C_D \sin(\Phi + \alpha_i)] \quad (2.19)$$

$$\frac{d\tau}{dr} = \frac{1}{2} \rho \left(V_\infty^2 + (\omega r)^2 \right) c_{(r)} [C_L \sin(\Phi + \alpha_i) + C_D \cos(\Phi + \alpha_i)] \quad (2.20)$$

The corresponding incremental braking power could be calculated by,

$$\frac{dP}{dr} = r \omega \frac{1}{2} \rho \left(V_\infty^2 + (\omega r)^2 \right) c_{(r)} [C_L \sin(\Phi + \alpha_i) + C_D \cos(\Phi + \alpha_i)] \quad (2.21)$$

Let us non-dimensionalize the incremental thrust and braking power and integrate along the length of the blade,

$$C_T = \frac{N \cdot \int_{r_{hub}}^{R_p} dT}{\rho \cdot (\omega/2\pi)^2 \cdot (2R_p)^4} \quad (2.22)$$

$$C_P = \frac{N \cdot \int_{r_{hub}}^{R_p} dP}{\rho \cdot (\omega/2\pi)^3 \cdot (2R_p)^5} \quad (2.23)$$

Let us collect the terms given in equations 2.22 and equation 2.23 and simplify:

$$C_T = \frac{\pi^3}{8} \int_{r_{hub}}^{R_p} \left(\left(\frac{V_\infty}{R_p \omega} \right)^2 + \left(\frac{r}{R_p} \right)^2 \right) \cdot [C_L \cos(\Phi + \alpha_i) - C_D \sin(\Phi + \alpha_i)] \left(\frac{Nc(r)}{\pi R_p} \right) \left(\frac{dr}{R_p} \right) \quad (2.24)$$

$$C_P = \frac{\pi^4}{8} \int_{r_{hub}}^{R_p} \left(\frac{r}{R_p} \right) \left(\left(\frac{V_\infty}{R_p \omega} \right)^2 + \left(\frac{r}{R_p} \right)^2 \right) \cdot [C_L \sin(\Phi + \alpha_i) + C_D \cos(\Phi + \alpha_i)] \left(\frac{Nc(r)}{\pi R_p} \right) \left(\frac{dr}{R_p} \right) \quad (2.25)$$

where N denotes number of propeller blades. Let us define the nondimensional quantities below:

$$x_r = \frac{r}{R_p} \quad (2.26)$$

$$\frac{J}{\pi} = \frac{V_\infty}{\omega r} \quad (2.27)$$

and

$$\sigma_{(x_r)} = \frac{Nc_r}{\pi R_p} \quad (2.28)$$

Although McCormick's small angle approach is as similar as Benini's study on combined blade element momentum theory (Benini, 2003), in spite of the fact that Benini recommends axial induction factor as a function of induced angle of attack and section advance coefficient and computes until the solution converges using Goldstein factor while McCormick suggests giving a tentative value for angle of attack and compute the induced angle of attack iteratively. Though, either Benini's study or

McCormick's small angle approach approximately the same result, McCormick have not used Goldstein Tachmindji circulation correlation factors.

2.3 Benini' s Approximation

The propeller blade could be divided into a number of strips along the radius, where a force balance is applied within the strip, involving two dimensional lift and drag with the produced thrust and torque. Besides, in due course of this computation, the balance of axial and angular momentum is applied as a function of axial and tangential induction factors. This means a number of nonlinear equations that could be solved iteratively for each radial strip. The final values of thrust and torque are radially integrated along the radius to compute the overall propeller performance (Benini, 2003).

During this process, in order to compute two dimensional section lift and drag coefficients, two dimensional integral boundary layer method is applied using XFOIL which is a coupled panel/viscous code developed at MIT (Dreala, 1989). XFOIL is an agglomeration of codes for foil design and analysis for incompressible/compressible viscous flows over an arbitrary foil.

According to Benini' s study, blade element momentum theory follows Goldstein' s vortex theory, Betz' s, Milan' s and Tachmindji' s studies. More particularly, the system of nonlinear equations to be solved at each radial strip along the blade is following:

$$\frac{dT}{dr} = \frac{1}{2}\rho \left(V_{\infty}^2 (1+a)^2 + (\omega r)^2 (1-a')^2 \right) N_{C(r)} [C_L \cos(\Phi) - C_D \sin(\Phi)] \quad (2.29)$$

$$V_T = \sqrt{V_{\infty}^2 (1+a)^2 + \omega^2 r^2 (1-a')^2} \quad (2.30)$$

$$\tan \Phi_i = \frac{(1+a)}{(1-a)} \frac{V_{\infty}}{\omega r} = \frac{(1+a)}{(1-a)} \tan \Phi = \frac{\tan \Phi + \tan \Phi_{\epsilon}}{2} \quad (2.31)$$

$$\alpha = \beta - \alpha_i - \Phi \quad (2.32)$$

$$\frac{dT}{dr} = 4\pi\rho r\kappa_\epsilon V_\infty^2 a(1 + \kappa a) \quad (2.33)$$

$$\kappa = \kappa(\Phi, r/R, \Phi_x) \quad (2.34)$$

$$V_\infty^2 a(1 + a) = (\omega r)^2 a'(1 + a') \quad (2.35)$$

After the local axial and tangential thrust have been calculated by solving the system of nonlinear equations given above for each radial strip, the total thrust and torque could be expressed by radial integration of the components along the propeller radius.

$$T = \int_{r_{hub}}^{R_p} dT \quad (2.36)$$

$$Q = \int_{r_{hub}}^{R_p} dQ = \int_{r_{hub}}^R \frac{1}{2} \rho V_T^2 N c(r) (C_L \sin \Phi + C_D \cos \Phi) r dr \quad (2.37)$$

The performance coefficients for marine propellers are calculated as usual:

$$K_T = \frac{T}{\rho n^2 D^4} \quad (2.38)$$

$$K_Q = \frac{Q}{\rho n^2 D^5} \quad (2.39)$$

and

$$\eta_o = \frac{J}{2\pi} \cdot \frac{K_T}{K_Q} \quad (2.40)$$

The blade element momentum theory which was developed by Benini is based on axial induction factor, as stated previously. First a tentative value for axial induction factor a is presupposed; a' is calculated using the equation 2.35. Second, Φ_i , α and V_T values are computed in sequence from equations 2.31, 2.30 and 2.32 respectively. The lift and drag coefficients of two dimensional airfoil section are calculated using XFOIL. The incremental thrust for blade section is numerically calculated with equation 2.29 and then determined new value of $a^\star$ is compared with the old value of a , using 2.33.

As stated above, the thrust values are corrected with Goldstein-Tachmindji circulation correlation factors (κ). This iteration is continued until the absolute tolerance between $a^\star$ and a is converged enough.

2.4 McCormick's Approximations

2.4.1 Linear, small angle solution algorithm

To solve blade element momentum equations iteratively, McCormick has developed an algorithm which gives better solution from the analytical solution (McCormick, 1967). According to the McCormick's approximation, due to the fact that difference between angle of attack and local pitch angle of the radial section is too large, drag force has a negligible effect on the induced angle of attack. For this approximation, the local incremental thrust could be rewritten in terms of,

$$dC_T = \frac{\pi^3}{8} \left(\left(\frac{J}{\pi} \right)^2 + x_r^2 \right) \cdot [C_L \cos(\Phi) - C_D \sin(\Phi)] \sigma_{x_r} dx_r \quad (2.41)$$

and it is seen that the drag term could be neglected and lift coefficient approximates using linearity.

$$C_L = C_{L\alpha} \cdot \alpha = C_{L\alpha} [\beta - (\alpha_i + \Phi)] \quad (2.42)$$

where $C_{L\alpha}$ denotes the linear lift curve slope. Let us substitute equation 2.42 to equation 2.41,

$$dC_T = \left(\frac{\pi}{8} \right) (C_{L\alpha} (\beta - \Phi - \alpha_i) \cos \Phi) \cdot [J^2 + (\pi x_r)^2] \sigma_{x_r} dx_r \quad (2.43)$$

As it is seen above, when the difference between angle of attack and local pitch angle of the radial section is too high, the induced angle of attack could also be neglected.

We can easily express from momentum theory,

$$\frac{\delta V_i}{\left(\frac{\omega}{\pi} R_p \right)} = -\frac{J}{2} + \sqrt{\left(\frac{J}{2} \right)^2 + \frac{1}{\pi x_r} \cdot \frac{dC_T}{dx_r}} \quad (2.44)$$

and let us make these substitutions,

$$\sin \Phi = \frac{V_{\infty}}{\sqrt{V_{\infty}^2 + (\omega r)^2}} \quad (2.45)$$

$$\cos \Phi = \frac{\omega r}{\sqrt{V_{\infty}^2 + (\omega r)^2}} \quad (2.46)$$

Let us resolve dC_T from the momentum theory gives,

$$dC_T = \pi^3 x_r dx_r \left(\frac{\delta V_i}{\sqrt{V_{\infty}^2 + (\omega r)^2}} \cdot \frac{x_r}{\cos \Phi} \right) \cdot \left(\tan \Phi \cdot x_r + \frac{\delta V_i}{\sqrt{V_{\infty}^2 + (\omega r)^2}} \cdot \frac{x_r}{\cos \Phi} \right) \quad (2.47)$$

Based on McCormick's approximation,

$$\delta V_i \approx \alpha_{i(r)} \sqrt{V_{\infty}^2 + (\omega r)^2} \cos \Phi \quad (2.48)$$

and equation 2.48 could be rewritten in terms of induced angle of attack as,

$$dC_T = (\pi x_r)^3 \alpha_{i(r)} dx_r \quad (2.49)$$

Let us equalise equations 2.43 and 2.49 gives,

$$(\pi x_r^3) \cdot \alpha_{i(r)} \cdot (\tan \Phi + \alpha_{i(r)}) dx_r = \left(\frac{\pi}{8} \right) (C_{L\alpha} (\beta - \Phi - \alpha_i) \cos \Phi) [J^2 + (\pi x_r^2)] \sigma_{(x_r)} dx_r \quad (2.50)$$

Then, let us make the substitutions below,

$$\tan \Phi = \frac{V_{\infty}}{\omega r} = \frac{V_{\infty}}{\omega R_p} \cdot \frac{R_p}{r} = \frac{J}{\pi x_r} \quad (2.51)$$

$$\cos \Phi = \frac{1}{\sqrt{\left(\frac{J}{\pi x_r} \right)^2 + 1}} = \frac{\pi x_r}{\sqrt{J^2 + (\pi x_r)^2}} \quad (2.52)$$

Then, let us collect α_i terms and write a quadratic equation,

$$\alpha_{i(r)}^2 + \left[J + \left(\frac{\sigma_{x_r}}{8x_r} \right) \cdot \sqrt{J^2 + (\pi x_r)^2} \right] \alpha_{i_r} - \left(\frac{\pi \sigma_{x_r}}{8x} \right) C_{L\alpha} (\beta - \Phi) \sqrt{J^2 + (\pi x_r)^2} = 0 \quad (2.53)$$

and the equivalent expression is derived as,

$$\begin{aligned} \alpha_{i(r)}^2 + \left[\frac{V_\infty}{\omega r} + \left(\sqrt{\left(\frac{V_\infty}{\omega r} \right)^2 + 1} \right) \cdot \left(\frac{C_{L\alpha} N c_r}{8\pi r} \right) \right] \alpha_{i(r)} \\ = \left((\beta - \Phi) \cdot \left(\sqrt{\left(\frac{V_\infty}{\omega r} \right)^2 + 1} \right) \right) \left(\frac{C_{L\alpha} N c_r}{8\pi r} \right) \end{aligned} \quad (2.54)$$

Solution of equation 2.54 for the induced angle of attack gives,

$$\alpha_i = -\frac{1}{2} \left(J + \frac{C_{L\alpha} N c}{8x_r} \sqrt{1 + J^2} \right) + \frac{1}{2} \sqrt{\left(J + \frac{C_{L\alpha} N c}{8x_r} \sqrt{1 + J^2} \right)^2 + \frac{C_{L\alpha} N c}{8x_r} \sqrt{1 + J^2} (\beta - \Phi)} \quad (2.55)$$

Equations 2.55, 2.24 and 2.25 allows us to numerically compute local thrust and power coefficients and integrate radially for the blade element momentum algorithm.

2.5 Goldstein Tachmindji Circulation Correlation Factors

Goldstein analysed the flow induced by a system of constant pitch helical surfaces of infinite length and produced a method of computing average momentum flux, as compared with infinite blades, in terms of the fluid velocities on the surface of the sheets in way of the blades [41].

Goldstein factors could be expressed in a closed form, as follows,

$$K = \frac{2}{\pi} \cos^{-1} \left(\frac{\cosh(xF)}{\cosh(F)} \right) \quad (2.56)$$

where,

$$F = \frac{N}{2x \tan \beta} - \frac{1}{2} \text{ for } F \leq 85, \text{ otherwise } K = 1.$$



3. COMPUTATIONAL FLUID DYNAMICS METHOD

Computational fluid dynamics method is used in a wide range of applications such as naval and aeronautical practices. Generally, this method is used to obtain distributions of pressure and velocity components on hull, propeller, rudder and other appendages. Thus, resistance, thrust and torque values are computed easily. Moreover, these applications could be modelled as steady or unsteady. Under computational fluid dynamics method, discretized governing equations for fluids could be computed using modeling algorithms such as steady-unsteady, viscous-inviscid or laminar-turbulent. Regarding this, improved turbulence models are determined according to the problem in hand.

3.1 Governing Equations For Fluid Dynamics

3.1.1 Continuity equation

The continuity equation identifies the conservation of mass in a control volume, i.e., mass can neither be created nor be disappeared. In differential form, continuity equation can be described as (Fergizer and Peric, 1997):

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0 \quad (3.1)$$

where $\mathbf{U}$ represents velocity, t is time and ρ is fluid density. Continuity equation for a steady-state flow is defined as,

$$\nabla \cdot (\rho \mathbf{U}) = 0 \quad (3.2)$$

For Newtonian and incompressible flows, the continuity equation is described as,

$$\nabla \cdot \mathbf{U} = 0 \quad (3.3)$$

3.1.2 Momentum equations

The objective of computational fluid dynamics method is computation of governing equations for fluid flow numerically. These governing equations are consist of continuity and momentum equations. Navier-Stokes equations which are consist of continuity and momentum equations are systems of ten partial differential equations with ten unknowns. This system could be simplified by taking into account incompressibility and isothermally.

In order to ensure incompressibility condition, when the fluid density is postulated as time-invariant, momentum equations could be discussed as a system of three partial differential equations.

$$\frac{D\mathbf{U}}{Dt} = \frac{\partial \mathbf{U}}{\partial t} + \mathbf{U} \cdot \nabla \mathbf{U} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{U} + \frac{1}{\rho} \mathbf{F} \quad (3.4)$$

This equation represents momentum equations at cartesian coordinates for incompressible and Newtonian fluids which are derived from Newton's Second Law. In this equation, $\mathbf{U}$ represents velocity, t is time, ρ is fluid density, p is pressure and ν is kinematic viscosity of the fluid.

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{U} \cdot \nabla \mathbf{U} = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{U} \quad (3.5)$$

Nondimensionalized form of this equation could be represented by (3.5). In this equation, Re represents Reynolds number. When the fluid density is considered as time-invariant, system of continuity and momentum equations could be called as Navier-Stokes equations for incompressible fluids. In order to solve this system, initial and boundary conditions must be defined. One of the kinematic boundary conditions on impermeable solid surface is $\mathbf{U} \cdot \hat{n} = 0$. This boundary condition illustrates impermeability of solid surface and continuity of the flow (no-flux). In addition, other kinematic boundary condition is $\mathbf{U} \cdot \hat{\tau} = 0$. This boundary condition represents finite shear stresses on the solid surface (no-slip). Another boundary condition that is defined on solid surface is dynamic boundary condition and this represents sustainability of shear stresses on the interface.

3.2 Turbulence

Whereas laminar flow qualified as steady and uniform, turbulent flow is considered as unsteady and non-uniform flow in both spatial and temporal dimensions. When the Reynolds number reaches up to a critical value, laminar flow will become unstable and transition to turbulent flow occurs.

The Reynolds number is defined as the ratio of inertial forces to viscous ones, and is defined as

$$Re = \frac{\rho UL}{\mu} = \frac{UL}{\nu} \quad (3.6)$$

where U is velocity, L is characteristic length, μ is dynamic viscosity and ν is kinematic viscosity. The value of critical Reynolds number varies according to the problem in hand. The difficulty of turbulence prediction accurately is the reason of the fact that high Reynolds numbers which are involves different scales, lies in the wide range. Richardson's studies show that eddies are regarded as energy steps which describes the eddy motions of turbulence. According to this approach, while large eddies are unstable and disappeared by time, the energy inside the large eddies are transferred to small scale eddies. This process continues and enery is transferred to small scale eddies until eddies reach to sufficient volume and the flow become stable and dissipate (Versteeg and Malalasakera, 2007). On the other hand, Kolmogorov's studies show that small scale eddies have generic form uniquely determined by dissipation. Besides, propagation of eddies is described by Kolmogorov length scale and is defined as (Fergizer and Peric, 2007):

$$\eta = \left(\frac{\nu^3}{\varepsilon} \right) \quad (3.7)$$

here ε denotes dissipation of turbulent kinetic energy per unit mass. Assuming that the velocity field is decomposed into two parts, the averaged velocity $\bar{U}$ and the turbulent fluctuation U' . Thus, dissipation could be described as

$$\varepsilon = \frac{AU'^3}{L_e} \quad (3.8)$$

where A is a constant of the order of unity and L_e is an average size of the energy which contains eddies. When equations 3.7 and 3.8 are considered and assuming $l_e \propto L_r$ ve $u' \propto U$,

$$\frac{L_r}{\eta} \propto Re^{\frac{3}{4}} \quad (3.9)$$

This equation shows, in order to discretize turbulence in all scales accurately, the number of cells in all directions must be of the same order as the ratio of $\frac{L_r}{\eta}$.

3.2.1 Turbulence modeling

In order to investigate turbulent flow, different methods have been improved to model it. The most widely used methods for turbulence modelling as follows,

1. Direct Numerical Simulation (DNS)
2. Large Eddy Simulation (LES)
3. Reynolds-averaged Navier-Stokes (RANS)
4. Reynolds Stress Model (RSM)

To sum up, no turbulence model is used when direct numerical simulation method is considered. The governing equations are discretized at all scales. When large eddy simulation method is taken into account, large scales of the flow are resolved and the smaller scales are modelled. Reynolds-averaged Navier-Stokes equations are time-averaged equations used for fluid motion. Time-average method is useful with regard to the computation, however it requires Reynolds stress terms are being modelled. Reynolds stress model (RSM) is one of the higher-level turbulence models, which is not to require modelling of stress terms (McDonough, 2007a).

3.2.2 Reynolds averaged Navier-Stokes equations

Reynolds-averaged Navier-Stokes equations are one of the turbulence models which are widely used in engineering applications. This approximation presents computationally feasible approach, due to the fact that it resolves averaged magnitudes of fluid flow, thus turbulent fluctuations are neglected (Versteeg and Malalasakera,

2007). Though RANS approximation is widely used, areas of usage are substantially limited. Discretization of eddies which involves high energy inside is impossible when RANS approximation is considered. Furthermore, RANS approach just computes time-averaged flow field, but it could not predicate phenomena such as separation accurately.

To avoid modelling of small scale eddies, $\mathbf{Q}$ is any value in steady flow, $\mathbf{Q}'$ is fluctuation term and $\overline{\mathbf{Q}}$ represents time-averaged term [53].

$$\mathbf{Q} = \mathbf{Q}' + \overline{\mathbf{Q}} \quad (3.10)$$

Time-averaged term could be stated as follows,

$$\overline{\mathbf{Q}} = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{t_0}^{t_0+T} \mathbf{Q} dt \quad (3.11)$$

where T represents integration interval which must be larger when compared to the time scale of fluctuations.

RANS equations are obtained, in consequence of pressure and velocity terms in Navier-Stokes equations are decomposed using equation 3.10 and the taking the time-average. Fluid density is supposed to be constant, time-averaged and fluctuation terms are verified continuity equation separately. Though, this assumption could be represented for non-linear Navier-Stokes equations as follows,

$$\rho \frac{D\overline{\mathbf{U}}}{Dt} + \rho \frac{\partial (\overline{u'_i u'_j})}{\partial x_j} = -\nabla \overline{p} + \mu \nabla^2 \overline{\mathbf{U}} + \mathbf{F} \quad (3.12)$$

where $\overline{u'_i u'_j}$ term denotes Reynolds stress tensor. Due to the fact that the presence of Reynolds stress terms yields to unknowns exceed the number of equations. $\overline{u'_i u'_j}$ term is a bounded variable which depends on physical characteristics of the fluid and flow conditions. To sum up, six new unknowns added to these equations and these terms depend on unknown turbulence variables. Therefore, in order to resolve RANS equations, turbulence models are developed. Turbulence models could be collected in under the same title by definition as follows.

3.2.2.1 Zero equation models

Zero equation models could be considered as the simplest type of turbulence models. Through Boussinesq eddy-viscosity approximation, Reynolds stress tensor could be represented by the product of an eddy viscosity and mean strain-rate tensor. Thus, Reynolds stress tensor could be represented by,

$$\left(\overline{u'_i u'_j}\right) = \mu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) - \frac{2}{3} \rho \delta_{ij} k = \tau_{ij} \quad (3.13)$$

where μ_t denotes eddy viscosity, δ_{ij} Kronecker delta and k turbulent kinetic energy. Turbulent kinetic energy could be defined as,

$$k = \frac{1}{2} \left(\overline{u'_i u'_i} \right) \quad (3.14)$$

When the terms which are given by i and j indexes in equation 3.13 are equal, the equation is satisfied. Eddy viscosity is not a physical property and it varies by flow conditions. In its simplest form, eddy viscosity could be defined in terms of two flow parameters, namely the turbulence velocity $q = \sqrt{2k}$ and length scale L , giving the following expression,

$$\mu_t = C_\mu \rho q L \quad (3.15)$$

where C_μ is a dimensionless constant. Essentially, C_μ is not a constant and it is defined by the ratio between the turbulent quantity to a mean flow quantity. Thus, Reynolds stress term is replaced with eddy viscosity term through Boussinesq approximation. This is quite effective simplification, however this method is easy to implement and provides advisably accurate results.

3.2.2.2 One equation models

In addition to Boussinesq approximation, these turbulence models are based on a different kind of turbulent kinetic energy equation. Turbulent kinetic energy equation is defined as (McDonough, 2007a),

$$\rho \frac{\partial k}{\partial t} + \rho U_j \frac{\partial k}{\partial x_j} = \tau_{ij} \frac{\partial U_j}{\partial x_i} - C_D \rho \frac{k^{\frac{3}{2}}}{l} + \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_t}{\sigma_K} \right) \frac{\partial k}{\partial x_j} \right] \quad (3.16)$$

where C_D and σ_K terms are closure coefficients. Whereas there many one equation turbulence models which are based on turbulent kinetic energy, equation 3.16 is felt into abeyance. Especially, complex flow problems with abrupt changes from wall-bounded to free-shear flows, or separation could not be discretised accurately. Nowadays, modern one equation models are used which are based on some other variable instead of k .

3.2.2.3 Two equation models

Two equation models are based on Boussinesq approximation and turbulent kinetic energy equation which is given by equation 3.16. These models are depend on specific turbulent dissipation rate ω and turbulent dissipation rate ε instead of k .

3.2.3 Menter's shear stress transport model

Shear-Stress Transport model (SST) is an eddy viscosity turbulence model of two equation models, which is represented by Menter (Menter, 1994). The starting point of this model is being able to discretise aeronautics flows with adverse pressure gradients and separation accurately. $k - \omega$ and $k - \varepsilon$ turbulence models could not predict accurately these problems themselves, thus SST's principal idea is enhance the capabilities of $k - \omega$ and $k - \varepsilon$ turbulence models. $k - \varepsilon$ turbulence model is robust, but it remains weak in the modelling of the boundary layer. However, $k - \omega$ turbulence model is robust in near wall treatments and adverse pressure gradients, but it behaves highly sensitive in free-stream flows.

In order to develop a more accurate model, Menter proposed a combination of $k - \omega$ and $k - \varepsilon$ turbulence models. In SST model formulation, $k - \omega$ turbulence model is used to discretise boundary-layer, whereafter a switch to the $k - \varepsilon$ turbulence model is made for the free-stream flow. So that, the major problems associated with $k - \omega$ and $k - \varepsilon$ turbulence models are both solved. This zonal approach depends on a blending function which ensures an appropriate selection of models.

SST $k - \omega$ turbulence model was given by Menter consists of the following formulations,

$$\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = P_k - \beta^* k \omega + \frac{\partial}{\partial x_j} \left[(v + \sigma_k v_t) \frac{\partial k}{\partial x_j} \right] \quad (3.17)$$

$$\frac{\partial \omega}{\partial t} + U_j \frac{\partial \omega}{\partial x_j} = \alpha S^2 - \beta \omega^2 + \frac{\partial}{\partial x_j} \left[(v + \sigma_v v_t) \frac{\partial \omega}{\partial x_j} \right] + 2(1 - F_1) \sigma_{\omega 2} \frac{1}{\omega} \frac{\partial k}{\partial x_i} \frac{\partial \omega}{\partial x_i} \quad (3.18)$$

where k denotes turbulent kinetic energy, v_t turbulent eddy viscosity, P_k production of turbulent kinetic energy, ω specific dissipation rate and σ_k , β , β^* , $\sigma_{\omega 2}$ are closure coefficients. The blending function F_1 is defined as,

$$F_1 = \tanh \left\{ \left\{ \min \left(\min \left[\max \left(\frac{\sqrt{k}}{\beta^* \omega y}, \frac{500v}{y^2 \omega} \right), \frac{4\sigma_{\omega 2} k}{CD_{k\omega}^+ y^2} \right], 10 \right) \right\}^4 \right\} \quad (3.19)$$

where

$$CD_{k\omega} = 2\sigma_{\omega 2} \frac{1}{\omega} \frac{\partial k}{\partial x_i} \frac{\partial \omega}{\partial x_i} \quad (3.20)$$

and y is the distance to the nearest wall. The turbulent eddy viscosity is defined as,

$$v_t = \frac{a_1 k}{\max(a_1 \omega, SF_2)} \quad (3.21)$$

where $S^2 = \left| \frac{1}{2} (\partial_j u_i + \partial_i u_j) \right|^2$ and $S = \left| \frac{1}{2} (\partial_j u_i + \partial_i u_j) \right|$ represents the invariant measure of strain. F_2 is the second blending function defined as,

$$F_2 = \tanh \left[\left[\min \left(\max \left(\frac{2\sqrt{k}}{\beta^* \omega y}, \frac{500v}{y^2 \omega} \right), 100 \right) \right]^2 \right] \quad (3.22)$$

To prevent the generation of turbulence in regions of stagnation, a production limiter is used as,

$$G = v_t \frac{\partial U_i}{\partial x_j} \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) \rightarrow P_k = \min(G, 10\beta^* k \omega) \quad (3.23)$$

From equation 3.13 it is distinct that blending function F_1 becomes close to zero away from the surface and the $k - \varepsilon$ model will be used, while the $k - \omega$ model will be used

Table 3.1 : The constants of SST model.

σ_{k1}	$\sigma_{\omega1}$	β_1	σ_{k2}	$\sigma_{\omega3}$	β_2	β^*	a_1
0.85	0.65	0.075	1.00	0.856	0.0828	0.09	0.31

at the boundary layer where F_1 becomes close to one. The constants for the SST model are presented in Table 3.1.

3.2.4 Near-wall treatment

Boundary layer is generally stated in logarithmic scale with in terms of dimensionless wall distance y^+ and dimensionless velocity u^+ , defined as (Wilcox, 1998);

$$y^+ = \frac{u_\tau y}{\nu} \quad (3.24)$$

$$u^+ = \frac{u}{u_\tau} \quad (3.25)$$

where, $u_\tau = \sqrt{\tau_\omega / \rho}$ is the friction velocity and τ_ω represents wall-shear stress. Velocity profile near the wall could be discussed in three separated region. In viscous sublayer, velocity profile becomes linear and turbulence doesn't occur. This layer is instead of dominated by viscous shear. In buffer layer, velocity profile is smooth merge of linear and logarithmic profiles. In log-law layer, the velocity profile behaves logarithmic and could be expressed by,

$$u^+ = \frac{1}{\kappa} \ln y^+ + C^+ \quad (3.26)$$

where, $\kappa = 0.41$ denotes von Karman constant, and $C^+ = 5.0$.

There are two different phenomenons about near-wall treatment when using CFD methods. First, two equation models could not be integrated through the viscous sublayer accurately, and second is the integration through the viscous sublayer is computationally expensive due to the fact that dimensionless wall distance has to be $y^+ = 1$ and the cells under viscous sublayer have to be high-quality. To achieve these phenomenons, wall-functions have to be used. Thus, instead the velocity profile is modeled, the flow is not required to be discretised with no-slip boundary condition ($U = 0$) completely which describes the finite wall-shear stresses.

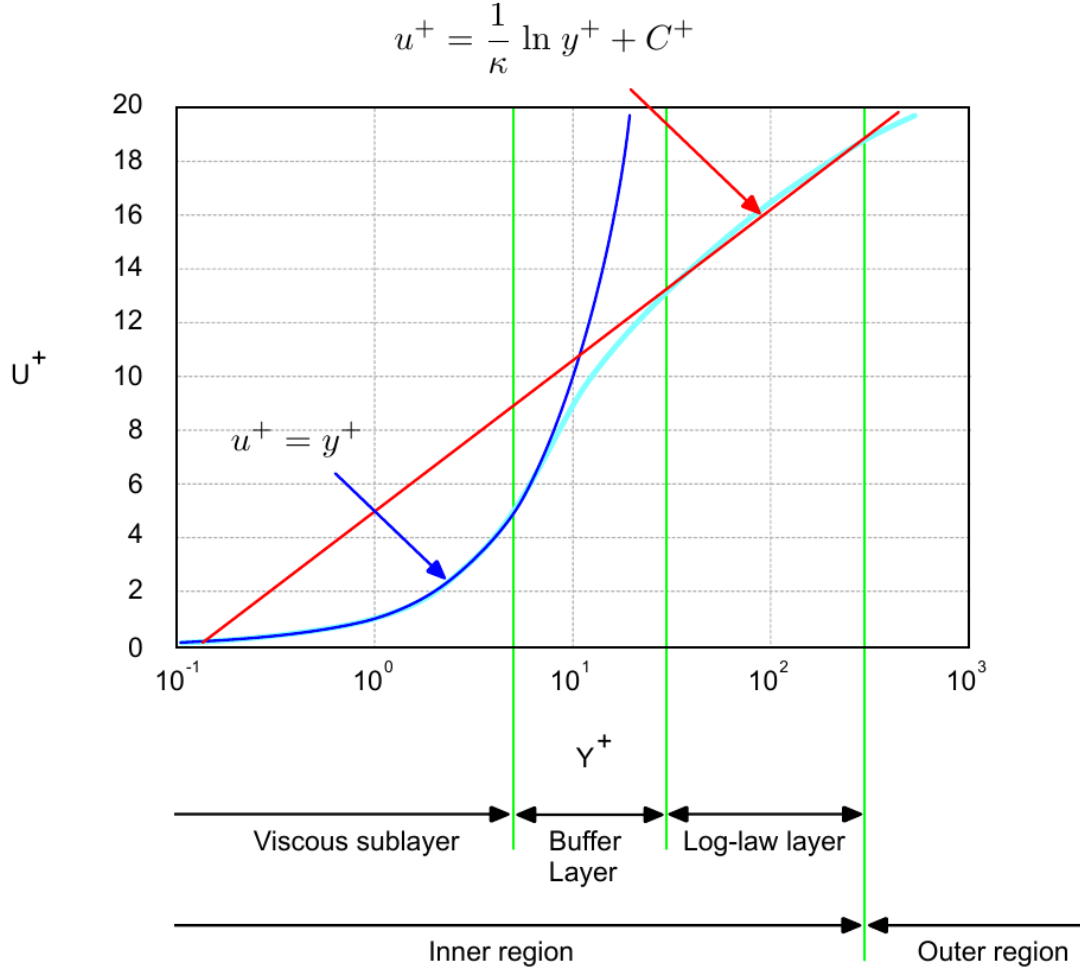


Figure 3.1 : Law of the wall.

Under the circumstances, the similar behaviour of different boundary layers, wall functions use the law of the wall as the fundamental relation between velocity and shear stresses.

Consequently, first layer thickness could be $y^+ \approx 30$ and coarser grid could be used.

Application of wall functions reduce the accuracy of the computation due to their nature. Due to the fact that velocity could not be computed accurately in viscous sublayer, physical conditions could not be satisfied. Especially detached flows cannot be modelled using wall functions, this phenomenon becomes crucial. In addition, another important effect is the cell height requirements. To satisfy these requirements, viscous layer must be generated iteratively.

3.3 Numerical Methods

Analytical solutions of governing equations for fluid flow are just available for some basic problems. Therefore, these nonlinear equations are constructed as a matrix system of linear equations. In order that, continuous equations are solved approximately with discrete system of linear equations, governing equations have to be formulated for discrete points in both space and time. In spite of the fact that, there are many approaches, finite volume, finite difference and finite element approaches are used extensively in computational fluid dynamics methods.

3.3.1 The finite volume method

The finite volume approach, is one of the solution methods for partial differential equations. This method is based on subdividing the solution domain into a finite number of smaller control volumes. These control volumes are adjacent, don't overlap with each other and fill the solution domain completely. The governing equations are integrated over each cell.

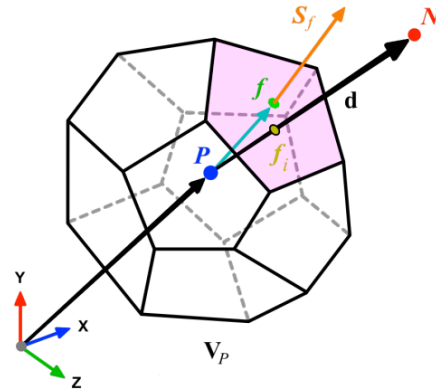


Figure 3.2 : Parameters in finite volume discretisation [53].

Variables are stored at the cell centroids P and N . Each cell is bounded by faces f , and each face is owned by one adjacent cell while the other cell is called the neighboring cell. Each face has an area of $|\mathbf{S}_f|$ and a unit normal vector $\mathbf{n}$ pointing towards the neighbour. A surface area vector can thus be defined as $\mathbf{S}_f = |\mathbf{S}_f|\mathbf{n}$. The vector $\mathbf{d}$,

points from the centre of the owner cell towards the centre of the neighbor cell $\mathbf{d} = \overline{PN}$. Finally, the volume of the cell is denoted as V_P .

3.3.2 Finite volume discretization

Navier-Stokes equations can be represented in an integral form. A general form of conservation equation could be expressed by,

$$\overbrace{\int_V \frac{\partial \rho \phi}{\partial t} dV}^{\text{time derivative}} + \overbrace{\int_V \nabla \cdot (\rho \mathbf{U} \phi) dV}^{\text{convection}} - \overbrace{\int_V \nabla \cdot (\rho \Gamma_\phi \nabla \phi) dV}^{\text{diffusion}} = \overbrace{\int_V S_\phi(\phi) dV}^{\text{source}} \quad (3.27)$$

where time derivative and source terms could be integrated over control volume. Time derivative term could be computed approximately using various methods. Convection and diffusion terms could not be integrated over the control volume and thus these volume integrals are converted to integrals over the cell surface $\mathbf{S}$ bounding the volume, using Gauss-Divergence theorem, defined as,

$$\int_V \nabla \cdot \phi dV = \oint_S \phi \cdot d\mathbf{S} \quad (3.28)$$

where, ϕ is any tensor field, $\mathbf{S}$ is surface vector and $\cdot$ represents either the gradient $\nabla \phi$, divergence $\nabla \cdot \phi$ or curl $\nabla \times \phi$. Let us apply the Gauss-Divergence theorem to the convective term, the volume integral will transform into a surface integral,

$$\int_V \nabla \cdot (\mathbf{U} \phi) dV = \oint_S \phi \mathbf{U} \cdot d\mathbf{S} \quad (3.29)$$

Then, let us approximate the surface integral through a sum over discrete surfaces, the convection term could be rewritten as

$$\int_V \nabla \cdot (\mathbf{U} \phi) dV = \oint_S \phi \mathbf{U} \cdot d\mathbf{S} \approx \sum_f \phi \mathbf{U} \cdot \mathbf{S}_f \quad (3.30)$$

In a similar way, the diffusion term could be rewritten through Gauss Divergence theorem and then approximated through a sum over discrete surfaces as follows,

$$\int_V \nabla \cdot (\mu \nabla \phi) dV = \oint_S (\mu \nabla \phi) \cdot d\mathbf{S} \approx \sum_f (\mu \nabla \phi) \cdot \mathbf{S}_f \quad (3.31)$$

The discrete transport equation could be rewritten as follows,

$$V_P \frac{\partial \phi}{\partial t} + \sum_f \phi \mathbf{U} \cdot \mathbf{S}_f = \sum_f (\mu \nabla \phi) \cdot \mathbf{S}_f + V_P S_\phi(\phi) \quad (3.32)$$

The variables $\mathbf{U}$, ϕ and $\nabla \phi$ are generally defined at cell centres but subject to the preceding equation, these variables must be defined on the cell faces f in the discrete formulation [54].

3.3.3 Interpolation

In central differencing, the value of ϕ at face center is interpolated by weight function which is defined as the ratio of the neighboring cell center distance to the face $\overline{fN}$ and the owner cell center $\overline{PN}$ [55].

$$\phi_f \approx f_x \phi_P + (1 - f_x) \phi_N \quad (3.33)$$

$$f_x = \frac{\overline{fN}}{\overline{PN}} = \frac{|\mathbf{n}_f \cdot \mathbf{d}_{fN}|}{|\mathbf{n}_f \cdot \mathbf{d}|} \quad (3.34)$$

By using the Taylor series expansion, the central differencing scheme can be shown to be of second-order accuracy, but it could be an oscillatory scheme due to its unboundedness.

Another straightforward method is to define values at a face-center is the upwind differencing scheme. Depending on the flow direction, either a backward or forward differencing methods could be used. The variable ϕ is approximated as,

$$\phi_f = \begin{cases} \phi_P & , \quad \mathbf{U}_f \cdot \mathbf{S}_f \geq 0 \\ \phi_N & , \quad \mathbf{U}_f \cdot \mathbf{S}_f < 0 \end{cases} \quad (3.35)$$

However this scheme is first-order accurate, this approximation unconditionally satisfies the boundedness of the scheme. Hence truncation error comprises first order term, oscillations will occur.

Besides, blending differencing scheme combines upwind and central differencing schemes in an attempt to conserve boundedness with reasonable accuracy,

$$\phi_f = (1 - \gamma) (\phi_f)_{UD} + \gamma (\phi_f)_{CD} \quad (3.36)$$

The truncation error of blending differencing scheme consists of those of the central and upwind differencing schemes. Hence, it comprises a first order term that is extended according to the value of γ which makes the solution more diffusive, on the other hand improves stability.

3.3.4 Pressure-velocity coupling methods

Pressure-velocity coupling is one of the phenomenons of CFD methods. Pressure does not have its own equation in incompressible flow, as it could be seen from the governing equations for fluid flow. Though the gradient of pressure term is a part of source term in the momentum equation, there isn't any certain way to obtain pressure. To handle this problem, the continuity equation could be used to correct the pressure values iteratively until both the continuity and momentum equations are satisfied correctly. Patankar developed SIMPLE (Semi-Implicit methods pressure-linked equations) algorithm which is widely used for modelling steady-state flows (Prakash and Patankar, 1985).

In order to obtain an equation for pressure term, momentum equations must be rewritten in discretised form. The convective term could be expressed by,

$$\sum_f \mathbf{U} \cdot \mathbf{S}_f \mathbf{U} = a_p \mathbf{U}_P + \sum_N a_n \mathbf{U}_N \quad (3.37)$$

where a_p and a_n are functions of $\mathbf{U}$. In case of the mass flux satisfies continuity equation, continuity and momentum equations have to be solved together and it means there will be a system of nonlinear equations. Thus equation 3.37 must be linearized. Therefore, in order to obtain solutions of a_p and a_n functions, velocity field is used which satisfies the continuity equation. Although this approach is quite good approximation for unsteady flows, a steady-state calculation is not affected by the linearization operation.

In order to reproduce pressure equation, momentum equations must be rewritten in semi-discretised form (Prakash and Patankar, 1985).

$$a_p \mathbf{U}_P = \mathbf{H}(\mathbf{U}) - \nabla p \quad (3.38)$$

where the $\mathbf{H}(\mathbf{U})$ term is composed of two parts. First part represents the matrix coefficients of the neighbouring cells multiplied by their velocity and the second part represents the unsteady term and all the sources except the pressure gradient.

$$\mathbf{H}(\mathbf{U}) = \sum_N a_N \mathbf{U}_N + \frac{U^\circ}{\Delta t} \quad (3.39)$$

The continuity equation is discretized as:

$$\nabla \cdot \vec{U} = \sum_f S \cdot \vec{U}_f \quad (3.40)$$

where $\vec{S}$ is outward-pointing face area vector and $\vec{U}_f$ the velocity on the face. The velocity on the face is obtained by interpolating the semi-discretized form of the momentum equation as follows,

$$\mathbf{U}_f = \left(\frac{\mathbf{H}(\mathbf{U})}{a_p} \right)_f - \left(\frac{1}{a_p} \right)_f (\nabla p)_f \quad (3.41)$$

By substituting this equation into the discretized continuity equation stated above, we obtain the pressure equation [55],

$$\nabla \cdot \left(\frac{1}{a_p} \nabla p \right) = \nabla \cdot \left(\frac{\mathbf{H}(\mathbf{U})}{a_p} \right) = \sum_f \mathbf{S} \cdot \left(\frac{\mathbf{H}(\mathbf{U})}{a_p} \right)_f \quad (3.42)$$

SIMPLE algorithm is developed for steady-state problems. The algorithm can be summed up as follows [55]:

1. Define the boundary conditions.
2. Solve the momentum equations to find the velocity components.
3. Compute the mass fluxes at cell faces.
4. Solve the pressure-correction equation at the cell faces.
5. Correct the pressure and velocities on the basis of intermediate values of the new pressure field and velocity components.

6. Increase the time step and repeat from step one until the solution converges.

For unsteady flows, PISO (Pressure Implicit with Splitting of Operators) algorithm was proposed by Issa in 1986 (Issa, 1985). The difference between SIMPLE and PISO algorithms is the momentum corrector step is performed more than once. The algorithm can be summed up as follows [56]:

1. Define the boundary conditions.
2. Solve the momentum equations to find the velocity components.
3. Compute the mass fluxes at cell faces.
4. Solve the first pressure-correction equation at the cell faces.
5. Correct the pressure and velocities on the basis of intermediate values of the new pressure field and velocity components.
6. Solve the second pressure-correction equation at the cell faces.
7. Correct the pressure and velocities on the basis of intermediate values of the new pressure field and velocity components.
8. Repeat from step three.
9. Increase the time step and repeat from step one until the solution converges.

For unsteady flows, PIMPLE algorithm is also developed which combines the loop structures of PISO and SIMPLE approaches. This algorithm can use adjustable time step and allows large time steps, thus it converges to the steady state solution quickly.

3.3.5 Linear solvers

The goal of CFD is reformation of partial differential equations for fluids as a system of algebraic equations. These algebraic equations are rewritten as a matrix system below,

$$A\phi = Q \quad (3.43)$$

where A is called coefficient matrix which must be sparse, diagonally-dominant and well-conditioned. Due to the fact that the matrices are sparse, they are solved iteratively using different algorithms. In this study, geometric-algebraic multi-grid method (GAMG) is used for the pressure equation, while smoothSolver is used with Gauss-Seidel smoother for velocity and turbulence quantities.

The idea of geometric-algebraic multi-grid method is providing a quick solution on a coarser mesh; mapping this solution onto a finer mesh; using it as an initial guess to achieve an accurate solution on the fine mesh. The finer mesh is coarsened stage by stage and refined while the matrix equations are solved for each step. GAMG is more faster than Krylov subspace solvers on cells which have low aspect ratio, when the increase in speed by solving first on coarser meshes outweighs the additional costs of mesh refinement and mapping of field data. In the course of mapping, prolongation (interpolation) or restriction methods are used. But before mapping any solution from one stage to another, the solution should be smoothed by using an iterative method that accomplishes smooth solutions.

Velocity and turbulent quantities are solved using smoothSolver. smoothSolver is an iterative solver for symmetric and asymmetric matrices which uses a run-time selected smoother to converge the solution to the required tolerance. To improve efficiency of the solution, the residual is evaluated after every smoothing iterations.

3.3.6 Meshing

In the previous chapter, we described finite volume discretisation method. In order to investigate characteristics of fluid flow, the physical computational domain is split into little subdomains which are called as cells, the Reynolds Averaged Navier Stokes equations are solved for each cell iteratively. This operation is called as grid generation

and it is very important to generate an appropriate grid structure for each case. Generally grids are subdivided into two categories: structured and unstructured grids. Although structured grids are basically made of hexahedral cells, unstructured grids are mainly made of tetrahedral, pyramids, hexahedral and prisms (hybrid meshes).

The quality of the mesh specifies the accuracy, stability, and the efficiency of the computation. The most common mesh quality metrics are introduced briefly, below:

Orthogonality is the angular deviation of the the vector S which is located at the face center f from the vector $\mathbf{d}$ that is connecting the two cell centers $\mathbf{P}$ and $\mathbf{N}$. Due to, it affects the gradient of the face center f , it adds diffusion to the solution and mainly effects the diffusive terms.

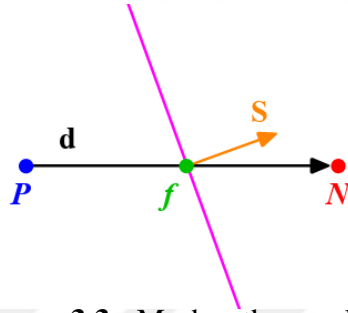


Figure 3.3 : Mesh orthogonality.

Skewness is the deviation of the vector $\mathbf{d}$ that connects the two cells $\mathbf{P}$ and $\mathbf{N}$, from the face center f . The deviation vector is represented with ∇ and f_i is the point where the vector $\mathbf{d}$ intersects the face f . Owing to, it affects the interpolation of the cell centered quantities to the face center f , it adds diffusion to the solution and affects convective terms.

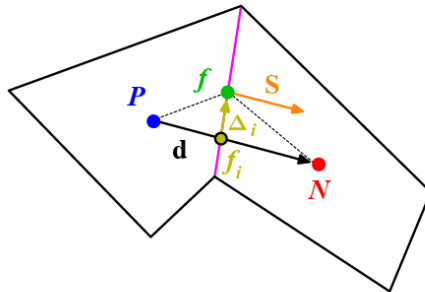


Figure 3.4 : Mesh skewness.

Expansion rate defines the transition in size between contiguous cells. Large transition ratios between cells add the diffusion to the solution. Ideally, the maximum change in mesh spacing should be less than %20.

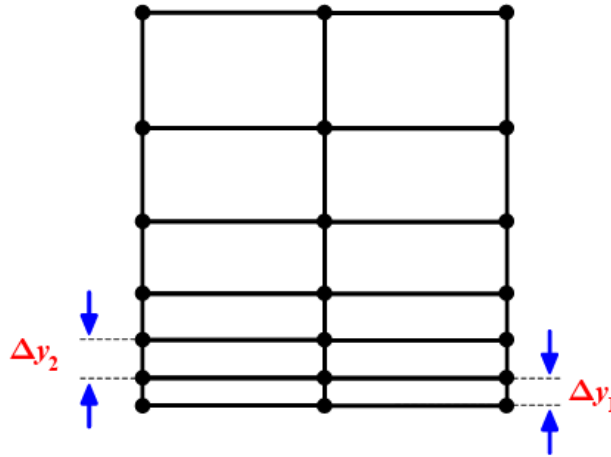


Figure 3.5 : Expansion rate.

Aspect ratio is the ratio between the longest side Δx and the shortest side Δy . Cells which have large aspect ratio could be acceptable, if the gradients in the largest direction are small, in spite of the fact that high aspect ratio smear gradients.

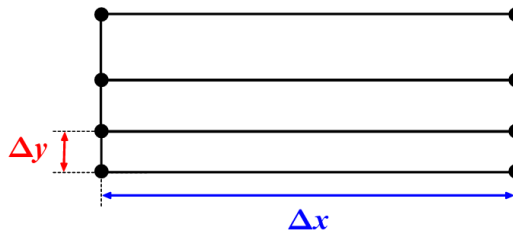


Figure 3.6 : Mesh aspect ratio.

In order to maintain second order accuracy and to avoid unboundedness, non-orthogonality and skewness errors are need to be corrected.

3.4 Rotational Motion

3.4.1 Multiple Frames of Reference Theory

Navier-Stokes equations in the inertial reference frame could not be applied to the reference frame precisely rotating with the propeller. Hence, a relation between two reference frames must be authorized and some synchronously transformations must be applied to the Navier-Stokes equations. Figure 3.7 shows a schematic representation of inertial and rotating reference frames.

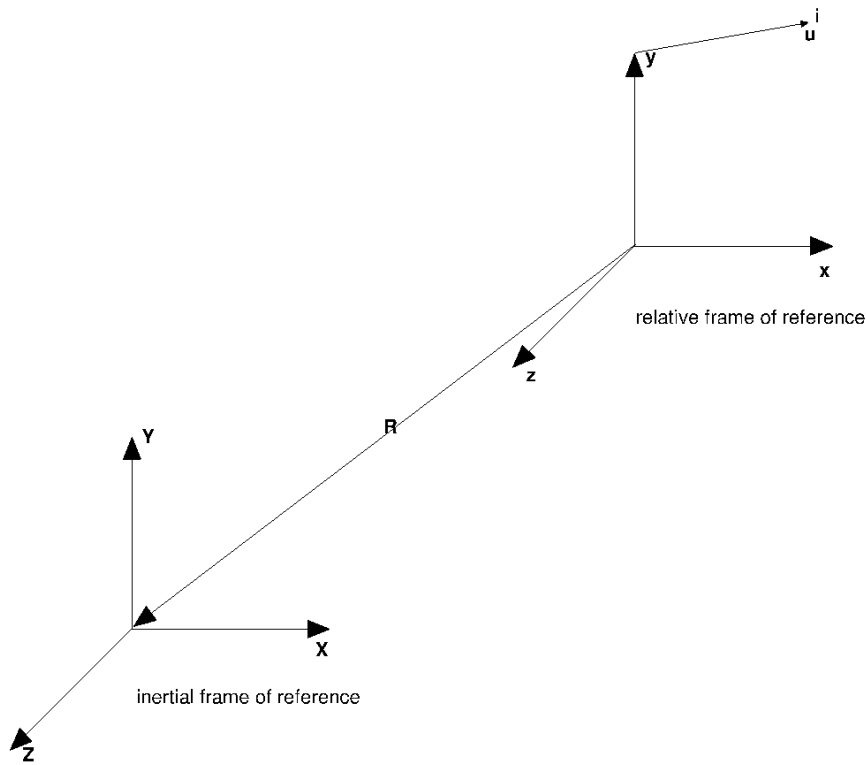


Figure 3.7 : Representation of two reference frames in Cartesian coordinate system.

As seen above, the inertial frame is denoted by (X, Y, Z) while the rotating frame is denoted by (x, y, z) Cartesian coordinate systems. The rotating axes is placed at a radius $R(t)$ from the inertial frame of reference and rotates with an angular velocity $\Omega(t)$ which are functions of time partially.

To begin, we will express the acceleration term for a rotating of frame by term of $\vec{\Omega}$. The position vector $\vec{r}$ can be stated as;

$$\left[\frac{d\vec{r}}{dt} \right]_I = \left[\frac{d\vec{r}}{dt} \right]_R + \vec{\Omega} \times \vec{r} \quad (3.44)$$

$$\vec{u}_I = \vec{u}_R + \vec{\Omega} \times \vec{r} \quad (3.45)$$

The acceleration can be denoted as,

$$\left[\frac{d\vec{u}_I}{dt} \right]_I = \left[\frac{d\vec{u}_I}{dt} \right]_R + \vec{\Omega} \times \vec{u}_I \quad (3.46)$$

$$\left[\frac{d\vec{u}_I}{dt} \right]_I = \left[\frac{d[\vec{u}_R + \vec{\Omega} \times \vec{r}]}{dt} \right]_R + \vec{\Omega} \times [\vec{u}_R + \vec{\Omega} \times \vec{r}] \quad (3.47)$$

$$\left[\frac{d\vec{u}_I}{dt} \right]_I = \left[\frac{d\vec{u}_R}{dt} \right]_R + \frac{d\vec{\Omega}}{dt} \times \vec{r} + \vec{\Omega} \times \left[\frac{d\vec{r}}{dt} \right]_R + \vec{\Omega} \times \vec{u}_R + \vec{\Omega} \times \vec{\Omega} \times \vec{r} \quad (3.48)$$

$$\left[\frac{d\vec{u}_I}{dt} \right]_I = \left[\frac{d\vec{u}_R}{dt} \right]_R + \frac{d\vec{\Omega}}{dt} \times \vec{r} + 2\vec{\Omega} \times \vec{u}_R + \vec{\Omega} \times \vec{\Omega} \times \vec{r} \quad (3.49)$$

The incompressible Navier-Stokes equations in the inertial frame for Newtonian fluids are given below.

$$\nabla \cdot \vec{u}_I = 0 \quad (3.50)$$

$$\frac{D\vec{u}_I}{Dt} = -\frac{1}{\rho} \nabla p + \nu \nabla \cdot \nabla (\vec{u}_I) \quad (3.51)$$

$$\frac{\partial \vec{u}_I}{\partial t} + \vec{u}_I \cdot \nabla (\vec{u}_I) = -\frac{1}{\rho} \nabla p + \nu \nabla \cdot \nabla (\vec{u}_I) \quad (3.52)$$

$$\frac{\partial \vec{u}_I}{\partial t} + \nabla \cdot (\vec{u}_I \otimes \vec{u}_I) - (\nabla \cdot \vec{u}_I) \vec{u}_I = -\frac{1}{\rho} \nabla p + \nu \nabla \cdot \nabla (\vec{u}_I) \quad (3.53)$$

$$\frac{\partial \vec{u}_I}{\partial t} + \nabla \cdot (\vec{u}_I \otimes \vec{u}_I) = -\frac{1}{\rho} \nabla p + \nu \nabla \cdot \nabla (\vec{u}_I) \quad (3.54)$$

Let us look at the left-hand side of the equation 3.51 , by taking into consideration equation 3.49 for the acceleration term;

$$\frac{D\vec{u}_I}{Dt} = \frac{D\vec{u}_R}{Dt} + \frac{d\vec{\Omega}}{dt} \times \vec{r} + 2\vec{\Omega} \times \vec{u}_R + \vec{\Omega} \times \vec{\Omega} \times \vec{r} \quad (3.55)$$

$$\frac{D\vec{u}_I}{Dt} = \frac{\partial \vec{u}_R}{\partial t} + \vec{u}_R \cdot \nabla \vec{u}_R + \frac{d\vec{\Omega}}{dt} \times \vec{r} + 2\vec{\Omega} \times \vec{u}_R + \vec{\Omega} \times \vec{\Omega} \times \vec{r} \quad (3.56)$$

$$\frac{D\vec{u}_I}{Dt} = \frac{\partial \vec{u}_R}{\partial t} + \nabla \cdot (\vec{u}_R \otimes \vec{u}_R) + \frac{d\vec{\Omega}}{dt} \times \vec{r} + 2\vec{\Omega} \times \vec{u}_R + \vec{\Omega} \times \vec{\Omega} \times \vec{r} \quad (3.57)$$

since

$$\nabla \vec{u}_I = \nabla \vec{u}_R \quad (3.58)$$

As it could be seen below,

$$\begin{aligned} \nabla \vec{u}_I &= \nabla \cdot [\vec{u}_R + \vec{\Omega} \times \vec{r}] = 0 \\ &= \nabla \cdot \vec{u}_R + \nabla \cdot [\vec{\Omega} \times \vec{r}] = 0 \\ &= \nabla \cdot \vec{u}_R = 0 \end{aligned} \quad (3.59)$$

Besides, it can be noted that,

$$\begin{aligned} \nabla \cdot \nabla \vec{u}_I &= \nabla \cdot \nabla [\vec{u}_R + \vec{\Omega} \times \vec{r}] \\ &= \nabla \cdot \nabla \vec{u}_R + \nabla \cdot \nabla [\vec{\Omega} \times \vec{r}] \\ &= \nabla \cdot \nabla \vec{u}_R \end{aligned} \quad (3.60)$$

The incompressible Navier-Stokes equations in terms of rotating velocities in the rotating frame can be represented by,

$$\nabla \cdot \vec{u}_R = 0 \quad (3.61)$$

$$\frac{\partial \vec{u}_R}{\partial t} + \frac{d\vec{\Omega}}{dt} \times \vec{r} + \nabla \cdot (\vec{u}_R \otimes \vec{u}_R) + 2\vec{\Omega} \times \vec{u}_R + \vec{\Omega} \times \vec{\Omega} \times \vec{r} = -\frac{1}{\rho} \nabla p + \nu \nabla \cdot \nabla (\vec{u}_R) \quad (3.62)$$

Due to the fact that the convected velocity is the velocity in the inertial frame, the term $\nabla \cdot (\vec{u}_R \otimes \vec{u}_R)$ in equation 3.62 can be further developed as,

$$\begin{aligned} \nabla \cdot (\vec{u}_R \otimes \vec{u}_R) &= \nabla \cdot \left(\vec{u}_R \otimes [\vec{u}_I - \vec{\Omega} \times \vec{r}] \right) \\ &= \nabla \cdot (\vec{u}_R \otimes \vec{u}_I) - \nabla \cdot \vec{u}_R (\vec{\Omega} \times \vec{r}) - \vec{u}_R \cdot \nabla (\vec{\Omega} \times \vec{r}) \\ &= \nabla \cdot (\vec{u}_R \otimes \vec{u}_I) - \vec{\Omega} \times \vec{u}_R \end{aligned} \quad (3.63)$$

Then, the steady term in left-hand side of equation 3.62 can be rewritten as,

$$\begin{aligned} \nabla \cdot (\vec{u}_R \otimes \vec{u}_R) + 2\vec{\Omega} \times \vec{u}_R + \vec{\Omega} \times \vec{\Omega} \times \vec{r} &= \nabla \cdot (\vec{u}_R \otimes \vec{u}_I) + \vec{\Omega} \times \vec{u}_R + \vec{\Omega} \times \vec{\Omega} \times \vec{r} \\ &= \nabla \cdot (\vec{u}_R \otimes \vec{u}_I) + \vec{\Omega} \times (\vec{u}_R + \vec{\Omega} \times \vec{r}) \\ &= \nabla \cdot (\vec{u}_R \otimes \vec{u}_I) - \vec{\Omega} \times \vec{u}_R \end{aligned} \quad (3.64)$$

The incompressible Navier-Stokes equations in the relative frame, in terms of the absolute velocity can be written as,

$$\nabla \cdot \vec{u}_I = 0 \quad (3.65)$$

$$\frac{\partial \vec{u}_R}{\partial t} + \frac{d\vec{\Omega}}{dt} \times \vec{r} + \nabla \cdot (\vec{u}_R \otimes \vec{u}_I) + \vec{\Omega} \times \vec{u}_I = -\frac{1}{\rho} \nabla p + \nu \nabla \cdot \nabla (\vec{u}_I) \quad (3.66)$$

Consequently, according to above mentioned formulations, multiple frames of reference theory is based on the computation of absolute Cartesian velocity components using the flux relative to the rotation of the local frame of reference.



4. NUMERICAL RESULTS

Blade element momentum theory was first applied to DTMB 4119 propeller. The DTMB 4119 propeller has the following geometric features and working conditions (Brizzolara, 2008):

1. The propeller inflow is uniform
2. The propeller has three blades.
3. The hub-to-diameter ratio is 0.2.
4. The blade geometries in terms of radial distribution of chord length c , camber f , thickness t and pitch P are given at Table 4.1.
5. The propeller has no skew and rake.

Table 4.1 : DTMB 4119 propeller geometry.

r/R	c/D	P/D	t_{max}/c	f_{max}/c
0.20	0.3200	1.1050	0.2055	0.0143
0.30	0.3635	1.1022	0.1553	0.0232
0.40	0.4048	1.0983	0.1180	0.0230
0.50	0.4392	1.0932	0.0902	0.0218
0.60	0.4610	1.0879	0.0696	0.0207
0.70	0.4622	1.0839	0.0542	0.0200
0.80	0.4347	1.0811	0.0421	0.0197
0.90	0.3613	1.0785	0.0332	0.0182
0.95	0.2775	1.0770	0.0323	0.0163
0.98	0.2045	1.0761	0.0321	0.0145
1.00	0.0800	1.0750	0.0316	0.0118

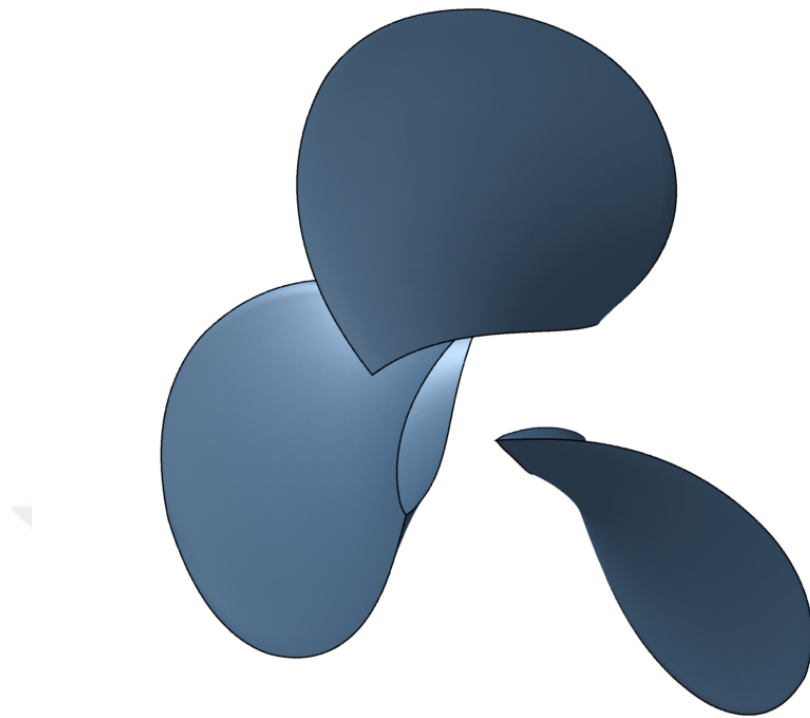


Figure 4.1 : Perspective view of DTMB 4119 propeller.



Figure 4.2 : Radial sections of DTMB 4119 propeller blade.

4.1 Open Water Computations with Blade Element Momentum Theory

4.1.1 Numerical algorithm

Developed code for blade element momentum theory was run for DTMB 4119 propeller under the given conditions. In order to obtain open water performance of the DTMB 4119 propeller, an extensive series of open water tests are carried out. For each radial blade section, a loop is performed to obtain the values of unknown variables of the system of the equations. First, a tentative value for $\alpha_{i(r)}$ is postulated for the first iteration and $\Phi_{(r)}$, x_r , J , $\sigma_{(x_r)}$, C_T , C_P , $J_{i(r)}$ and $\alpha_{i(r)}$ are calculated in sequence from equations 2.26, 2.27, 2.28, 2.24, 2.25, 2.10 and 2.11 respectively. Section lift and drag coefficients are estimated as a function of angle of attack, thickness ratio, camber ratio and Reynolds number using XFOIL. The equations are iterated until the requested and found $\alpha_{i(r)}$ were similar.

4.1.2 Computation case setup

The rotational velocity for open water case is considered constant as 10 rps. whereas the inflow speed of the propeller varies to generate different advance coefficients. There were 20 panels along the chordwise direction and 50 blade elements used along the radius of the blade. Open water computations are performed under the conditions given in the following Table 4.2.

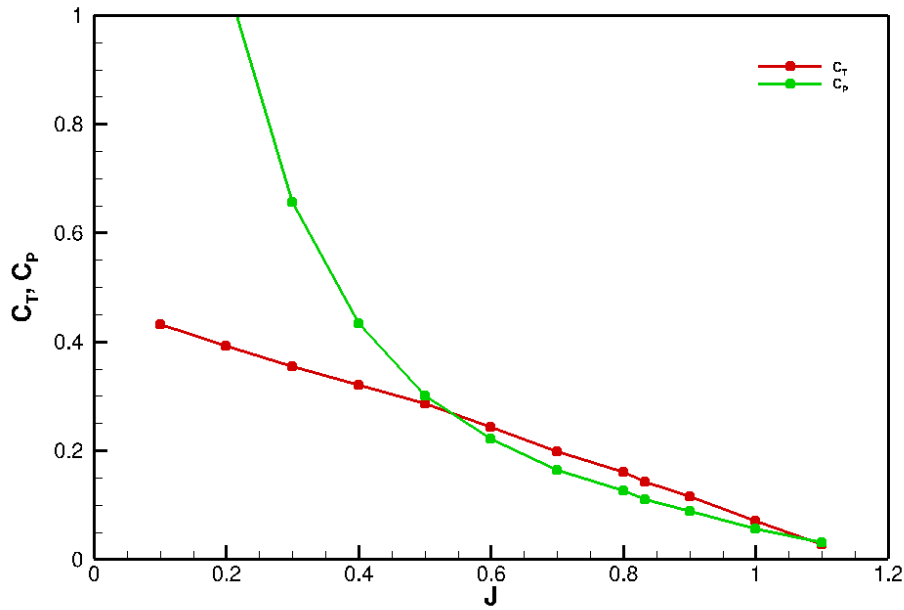
4.1.3 Computations and analysis

Due to the fact that linear airfoil theory is used for propeller section lift and drag coefficient calculations, the thrust coefficient is over predicted while the torque coefficient is under predicted for low advance coefficients. At low advance coefficients, angle of attack becomes sufficiently large, especially near the blade root. Figure 4.4 represents the thrust and torque coefficients K_T and K_Q and open water efficiency of the propeller versus advance coefficients (J) computed from blade element momentum theory and compared with experimental results which was introduced by Brizzolara et al (2008).

Table 4.2 : Open water operating conditions.

Advance Coefficient	Velocity Magnitude	Rotational Speed
$J[-]$	$V[m/s]$	$rps[1/s]$
0.100	0.305	10.000
0.200	0.610	10.000
0.300	0.914	10.000
0.400	1.219	10.000
0.500	1.524	10.000
0.600	1.829	10.000
0.700	2.134	10.000
0.800	2.438	10.000
0.833	2.539	10.000
0.900	2.743	10.000
1.000	3.048	10.000
1.100	3.353	10.000

Thrust and power coefficients which are computed for each advance coefficient can be seen at Figure 4.3.

**Figure 4.3** : C_T and C_P values using BEMT solution.

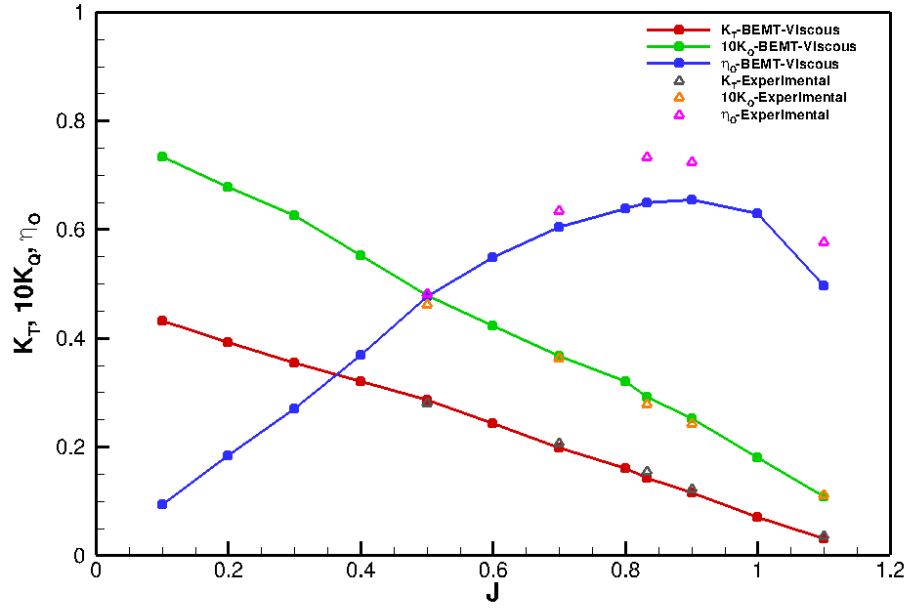


Figure 4.4 : Comparison of K_T , $10K_Q$ and η_o values using BEMT solution with experimental data.

The agreement between the results of blade element momentum theory algorithm and experimental data is satisfactory. Consequently, DTMB 4119 propeller is validated using blade element momentum theory algorithm. Table 4.3 shows the comparison of numerical computation results between blade element momentum theory and experimental results.

Table 4.3 : The comparison of blade element momentum theory and experimental results.

J	K_T (BEMT)	K_T (Exp.)	$10K_Q$ (BEMT)	$10K_Q$ (Exp.)	η_o (BEMT)	η_o (Exp.)
0.100	0.432	-	0.733	-	0.094	-
0.200	0.393	-	0.679	-	0.184	-
0.300	0.355	-	0.626	-	0.270	-
0.400	0.320	-	0.552	-	0.369	-
0.500	0.286	0.285	0.478	0.477	0.476	0.489
0.600	0.243	-	0.423	-	0.548	-
0.700	0.199	0.200	0.368	0.360	0.604	0.632
0.800	0.161	-	0.321	-	0.639	-
0.833	0.143	0.146	0.292	0.280	0.650	0.692
0.900	0.115	0.120	0.253	0.239	0.655	0.725
1.000	0.071	-	0.180	-	0.630	-
1.100	0.031	0.034	0.108	0.106	0.496	0.575

Table 4.4 shows the error propagation between blade element momentum theory and experimental results. The increase in the coefficient of advance causes an increase in the error, as expected. Since the viscous stresses are taken into account, open water efficiency values appear to be lower when compared to experimental results. Consequently, it could be said that blade element momentum theory results are quite satisfactory and it is as fast as potential codes, however it takes into account viscous effects.

Table 4.4 : The comparison of relative numerical percentage errors between blade element momentum theory and experimental results.

J	% ΔK_T	% $\Delta 10K_Q$	% $\Delta \eta_o$
0.500	-0.350877	-0.209644	2.658487
0.700	0.500000	-2.222222	4.430380
0.833	2.054795	-4.285714	6.069364
0.900	4.166667	-5.857741	9.655172
1.100	8.823520	-1.886792	13.739130

4.2 Open Water Computations with OpenFOAM

4.2.1 Mesh generation

Open water computations have been performed as a steady-state problem under specified boundary conditions. Due to provide these boundary conditions, the computational domain is designed based upon propeller diameter which is shown in Figure 4.5.

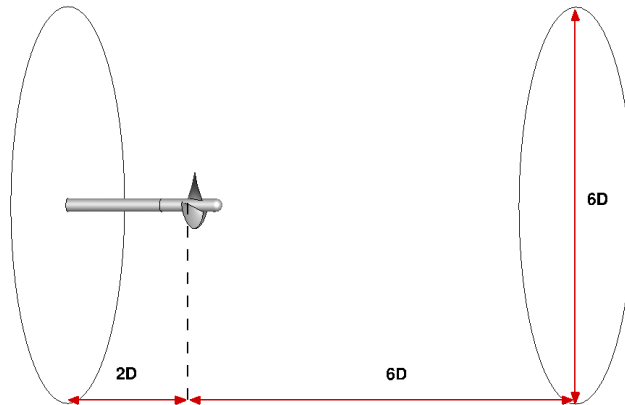


Figure 4.5 : Computational domain.

In order to perform mesh generation, we used Numeca Hexpress/Hybrid software which is quite easy to generate high-quality hybrid mesh structure for complex geometries. For our open water simulations, we divided computational domain into subdomains which consists of two volumes: rotating and static zones as shown in Figure 4.6 and 4.7 and they are connected with arbitrary mesh interfaces (AMI).

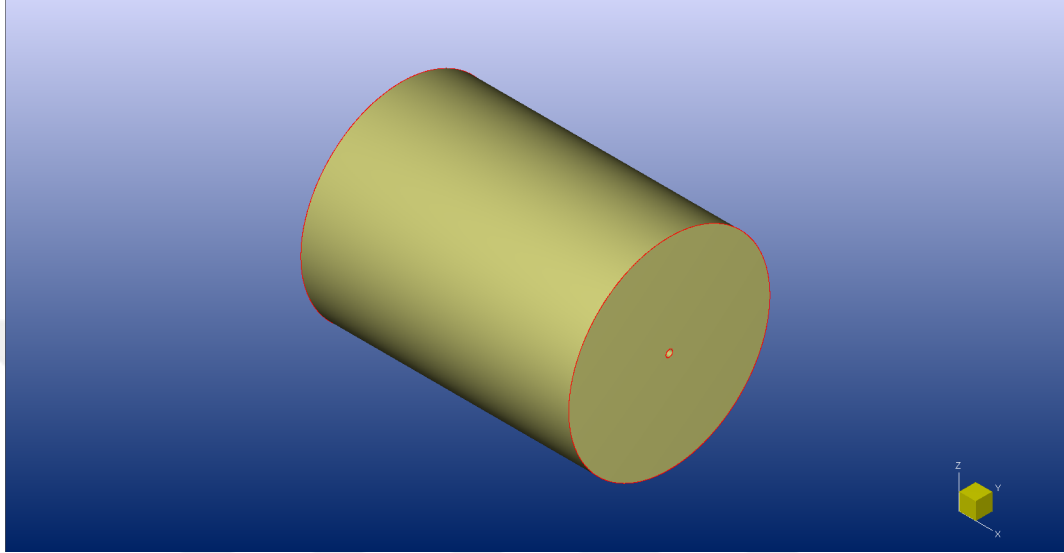


Figure 4.6 : Computational static domain.

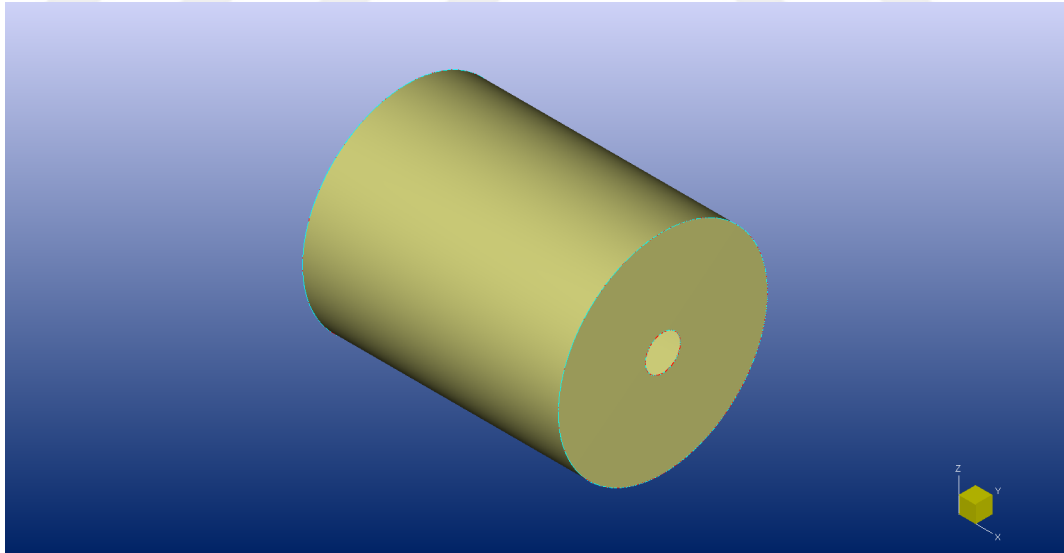


Figure 4.7 : Computational rotating domain.

Generation of mesh structure for open water simulations in Numeca Hexpress/Hybrid contains five different steps. First, the flow domains are divided into hexahedral block cells separately. Then a refinement step is used to spatially discretise the region around the propeller and in its wake and this performs mesh splitting according to the wake

region. After that, snapping procedure performs snapping cells to surfaces. Finally, generated mesh structure is optimized and viscous layer is added where possible according to the approximate y^+ specification.

In order to satisfy our open-source CFD code, mesh dependency studies are fulfilled. Therefore, three different mesh structures are generated; coarse, medium and fine. To generate coarse mesh structure, the computational static domain is divided into $48 \times 36 \times 36$ block cells in longitudinal, horizontal and vertical directions respectively. A cylindrical refinement volume is generated to spatially discretize the flow region around the propeller and its wake. This volume's diameter is two times bigger than the propeller's, and it covers the wake on 4 propeller diameters. Isotropic cells are used with a cell size corresponding to 24 points per diameter. The number of refinements on the solid patches are based on the patch names. Pressure and suction sides of the propeller blades are refined with a refinement level of 4, while 6 refinement levels are specified for leading and trailing edges. Besides, both propeller hub and shaft geometries are refined with a refinement level of 4. For the viscous layers, the initial spacing is set at $2.449E - 04$ [m] for $J = 0.833$. The stretching ratio is set at 1.2. An inflation technique is used to find the optimum number of layers, ensuring a smooth transition from viscous inner to outer Eulerian mesh.

The generated coarse mesh structure for the open water computations is shown in Figures 4.8, 4.9 and 4.10.

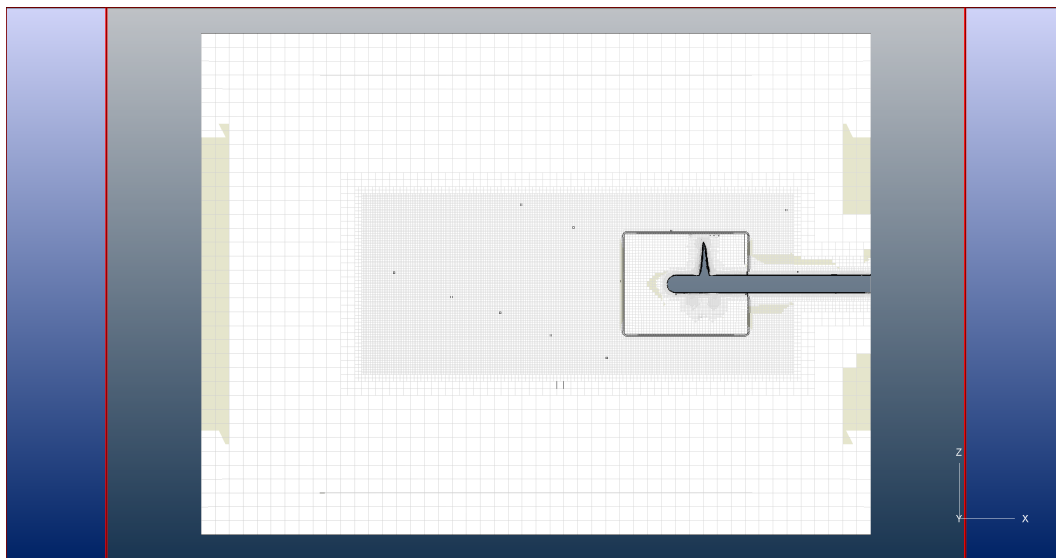


Figure 4.8 : Computational coarse mesh in Numeca Hexpress/Hybrid.

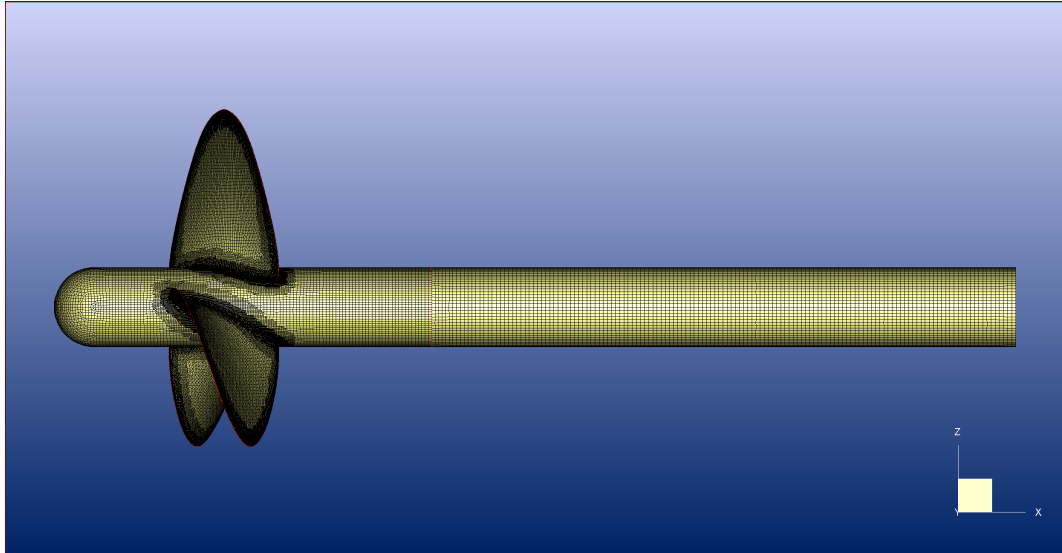


Figure 4.9 : Computational coarse mesh around propeller blades, hub and shaft in Numeca Hexpress/Hybrid.

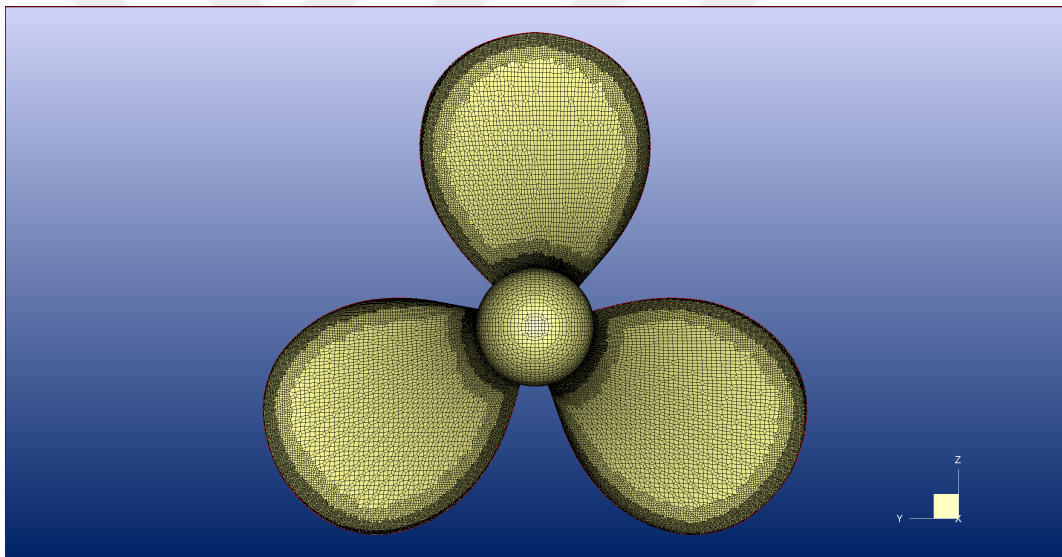


Figure 4.10 : Computational coarse mesh around propeller blades in Numeca Hexpress/Hybrid.

Since therefore shear stress terms must be computed accurately, boundary layer thickness has to be defined properly according to the law of the wall. In order to use extended wall functions at high advance ratios, the viscous layer thickness is approximately calculated, based on chord length at radial section $r = 0.7R$ and design point $J = 0.833$. The generated viscous layers for coarse grid, near the propeller are shown in Figure 4.11 and 4.12.

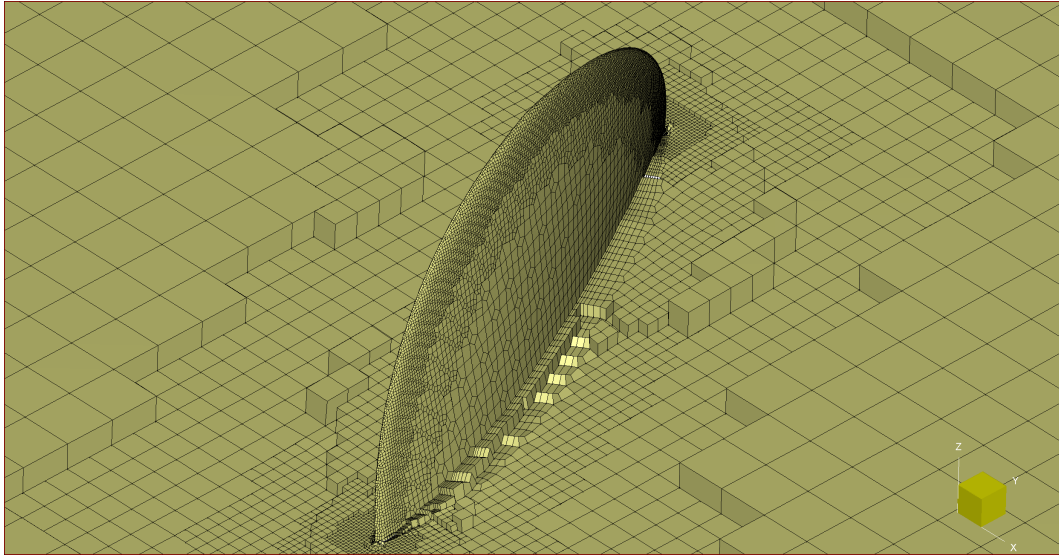


Figure 4.11 : Viscous layers at leading edge of the propeller.

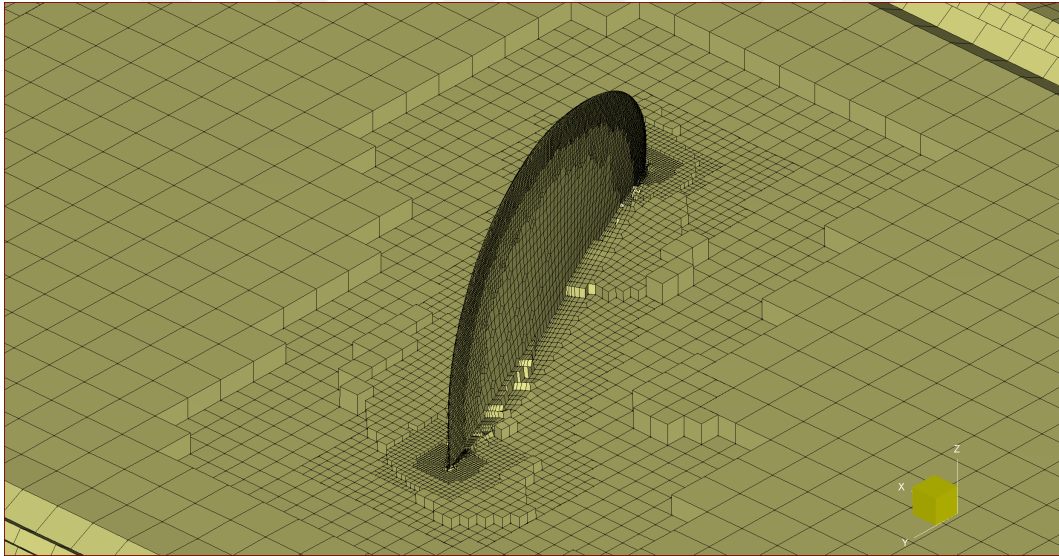


Figure 4.12 : Viscous layers at trailing edge of the propeller.

To generate medium mesh structure for open water computations, the isotropic refinement volume in static cell zone is used with a cell size corresponding to 24 points per diameter, like coarse mesh. Besides, pressure and suction sides of the propeller blades are refined with a refinement level of 6, while 8 refinement levels are specified for leading and trailing edges. Moreover, both propeller hub and shaft geometries are refined with a refinement level of 6.

The generated medium mesh structure for the open water computations is shown in Figures 4.13, 4.14 and 4.15.

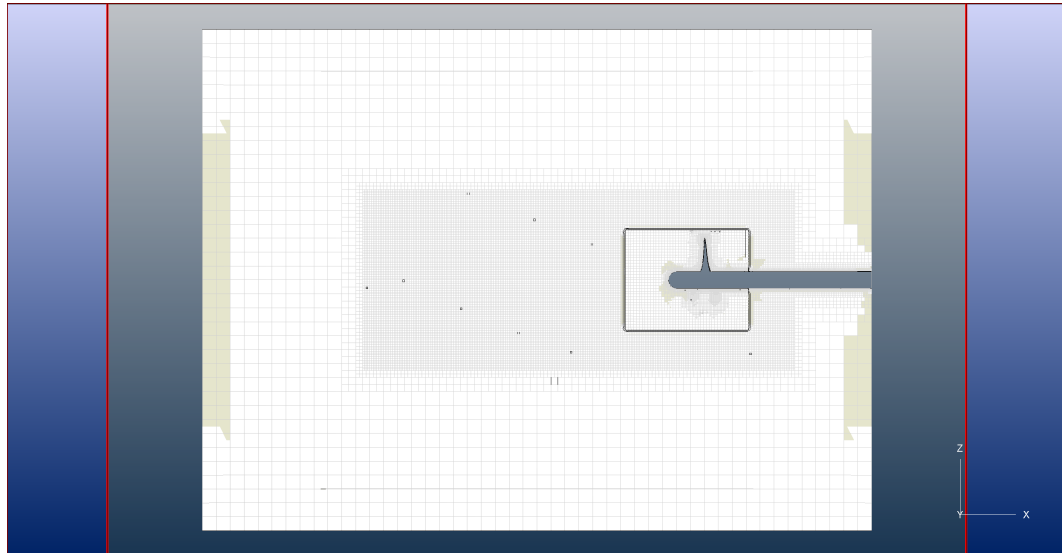


Figure 4.13 : Computational medium mesh in Numeca Hexpress/Hybrid.

So as to capture wake resolution at downstream of the propeller, maximum cell size is determined according to the propeller diameter. Although this provides the smooth transition between the mesh on the blade surfaces and the mesh on the interfaces, number of cells increases rapidly. To avoid this problem, the aspect ratio of the cells can be increased or refinement diffusion between cells can be decreased, but it should not be forgotten that local and cumulative continuity errors can increase, as expected.

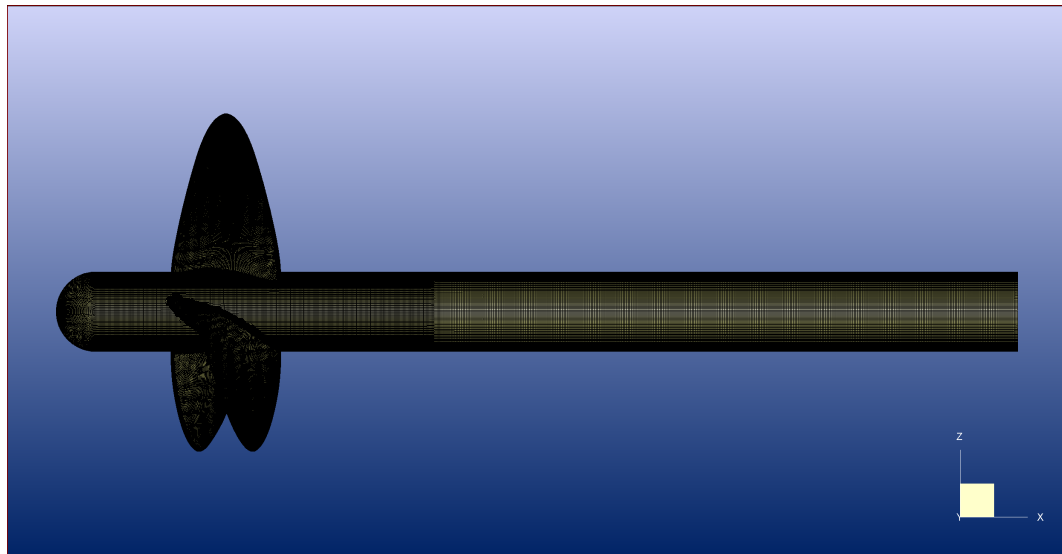


Figure 4.14 : Computational medium mesh around propeller blades, hub and shaft in Numeca Hexpress/Hybrid.

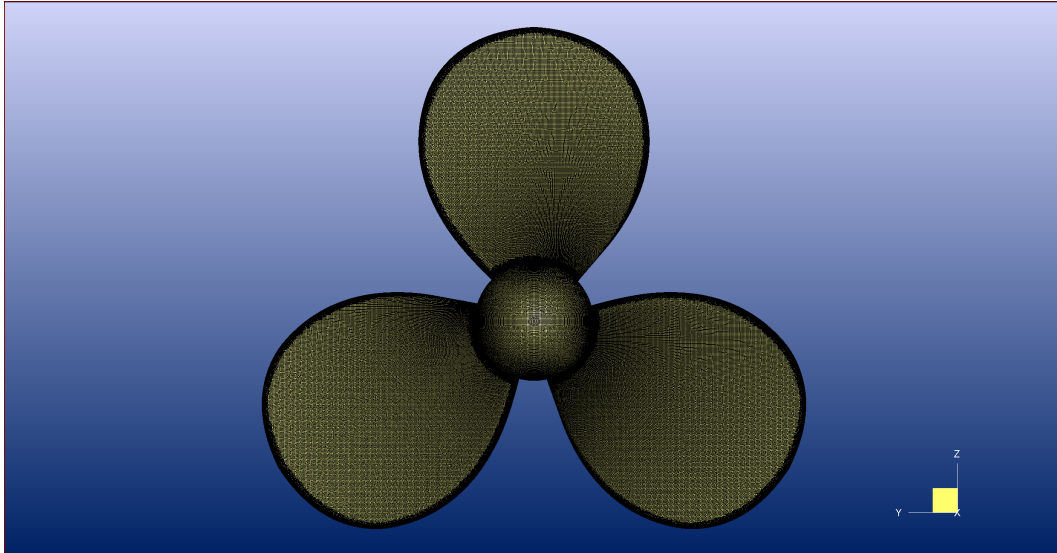


Figure 4.15 : Computational medium mesh around propeller blades in Numeca Hexpress/Hybrid.

As seen above, cell size on the propeller blade is too small and this causes more computational time according to the numerical simulations using coarse mesh structure. This means, despite of the fact that medium mesh quality is better than coarse mesh and results are more accurate than first computations, it is much more expensive. The increase in number of cells results the increase in number of processors which are used during computations, therefore no efficient calculation is executed.

The generated viscous layers for medium grid, near the propeller are shown in Figure 4.16 and 4.17.

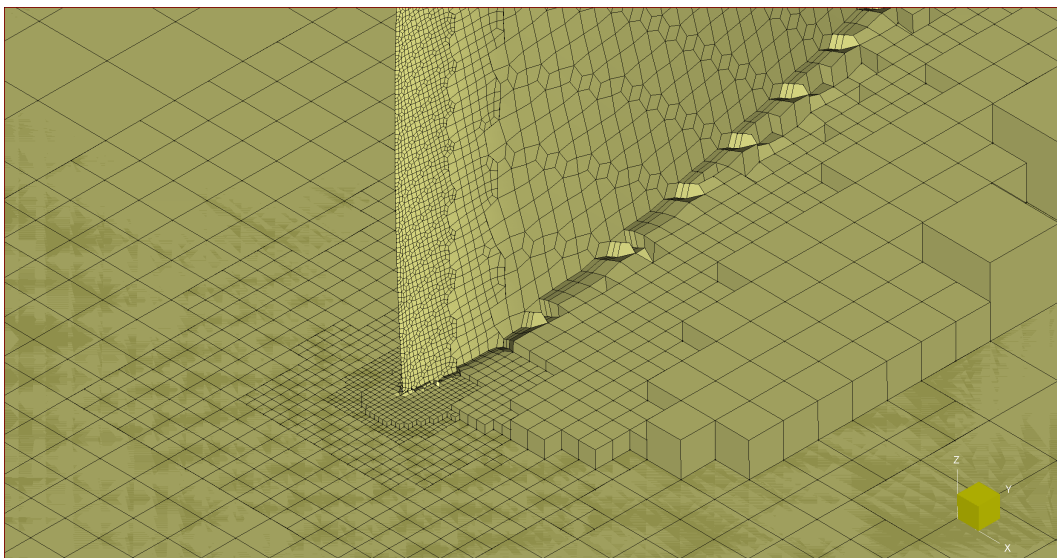


Figure 4.16 : Viscous layers at leading edge of the propeller.

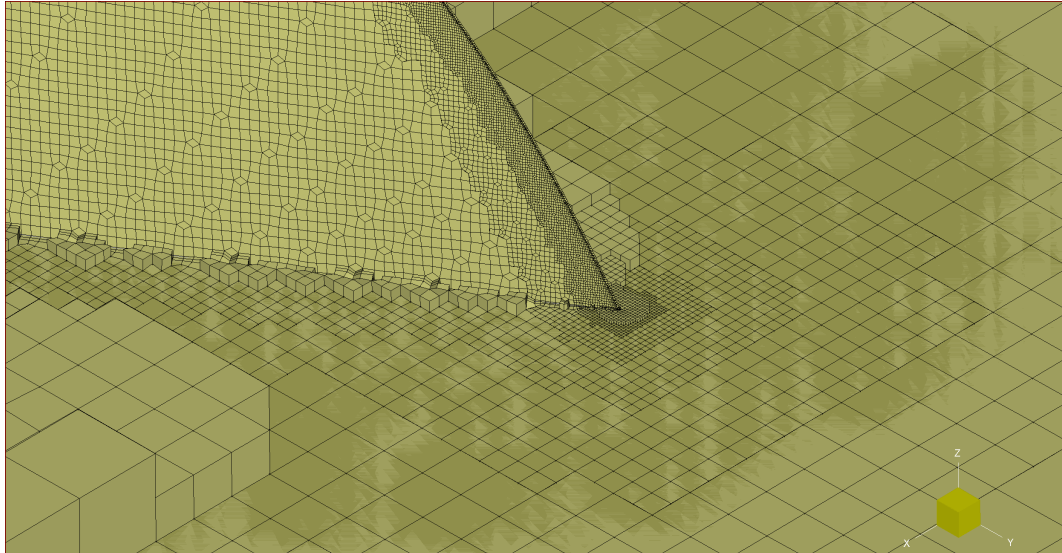


Figure 4.17 : Viscous layers at trailing edge of the propeller.

So as to generate fine mesh structure around propeller, the refinement volume in the wake region is refined with a cell size corresponding to 48 points per diameter, unlike medium mesh. However, pressure and suction sides of the propeller blades are refined with a refinement level of 6, while 8 refinement levels are specified for leading and trailing edges, propeller hub and shaft geometries are refined with a refinement level of 6.

The generated fine mesh structure for the open water computations is shown in Figures 4.16, 4.17 and 4.18.

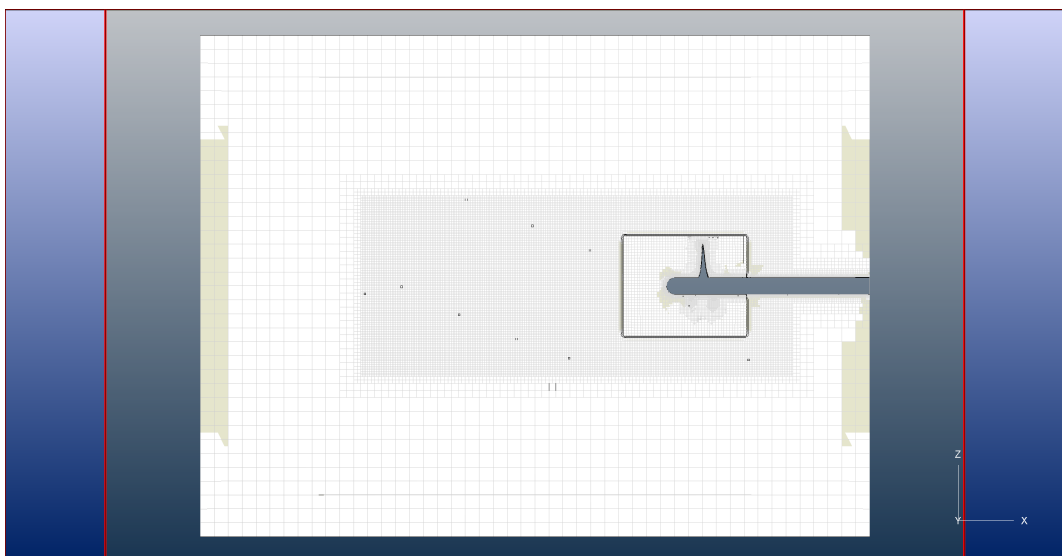


Figure 4.18 : Computational fine mesh in Numeca Hexpress/Hybrid.

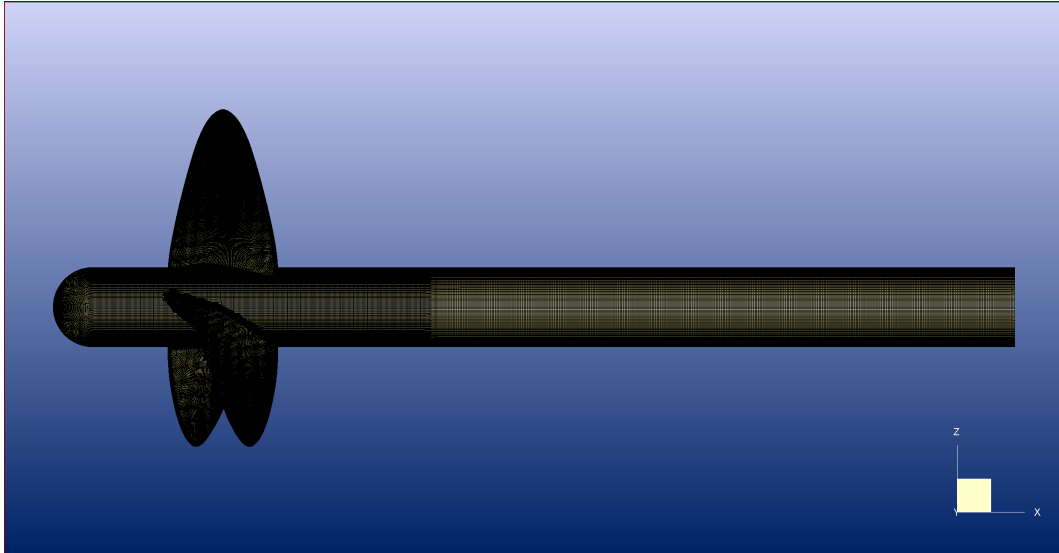


Figure 4.19 : Computational fine mesh around propeller blades, hub and shaft in Numeca Hexpress/Hybrid.

Due to the fact that the refinement cells at the propeller edges are too small, the number of cells to be produced in the viscous layer is reduced, although total thickness of boundary layer is similar. Since the cells in the refinement region and MRF zone are very small and the transition through the static zone is smooth, continuity can be computed with high accuracy. However, for these reasons, the number of processors to be used will increase further. In consequence, it is computationally much more expensive.

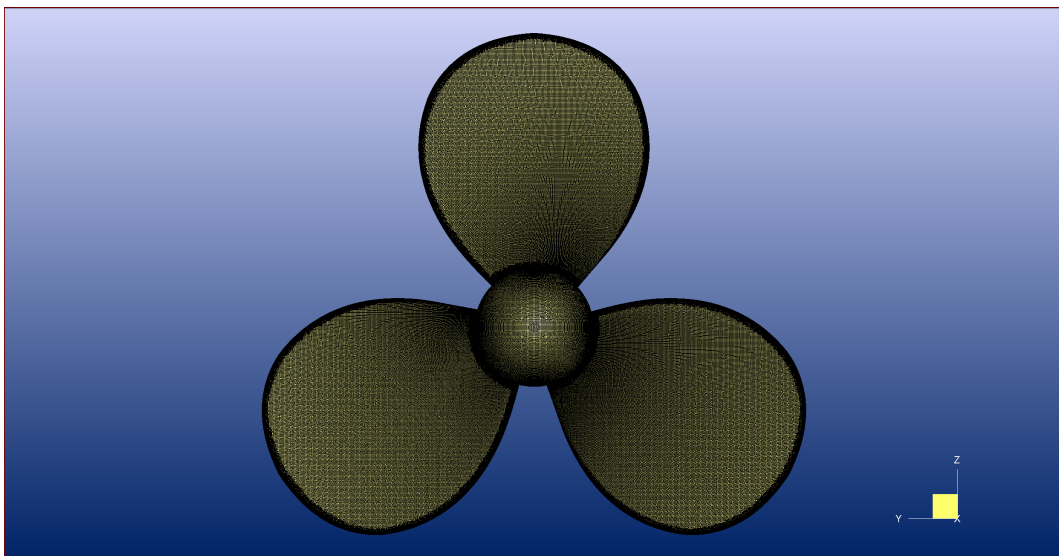


Figure 4.20 : Computational fine mesh around propeller blades in Numeca Hexpress/Hybrid.

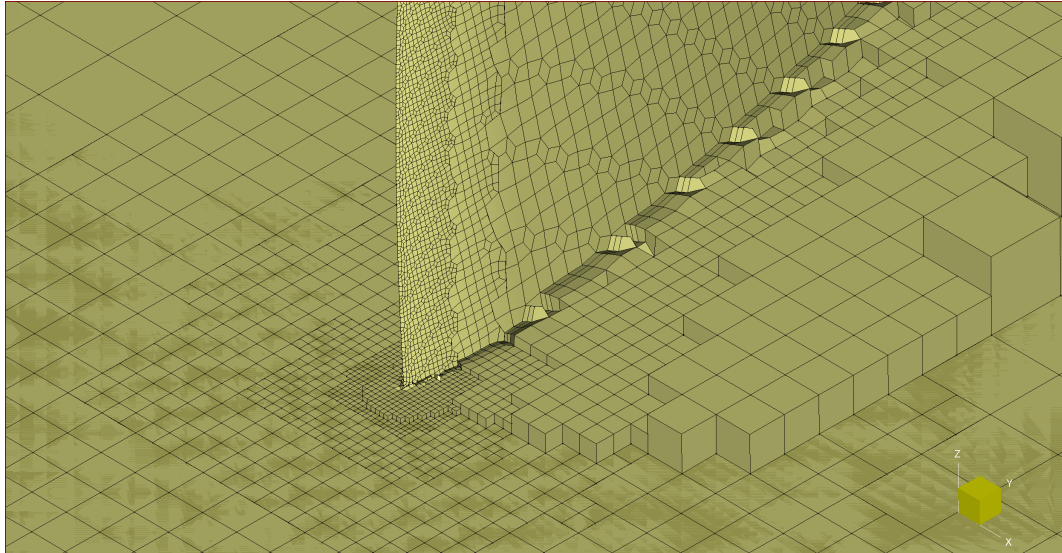


Figure 4.21 : Viscous layers at leading edge of the propeller.

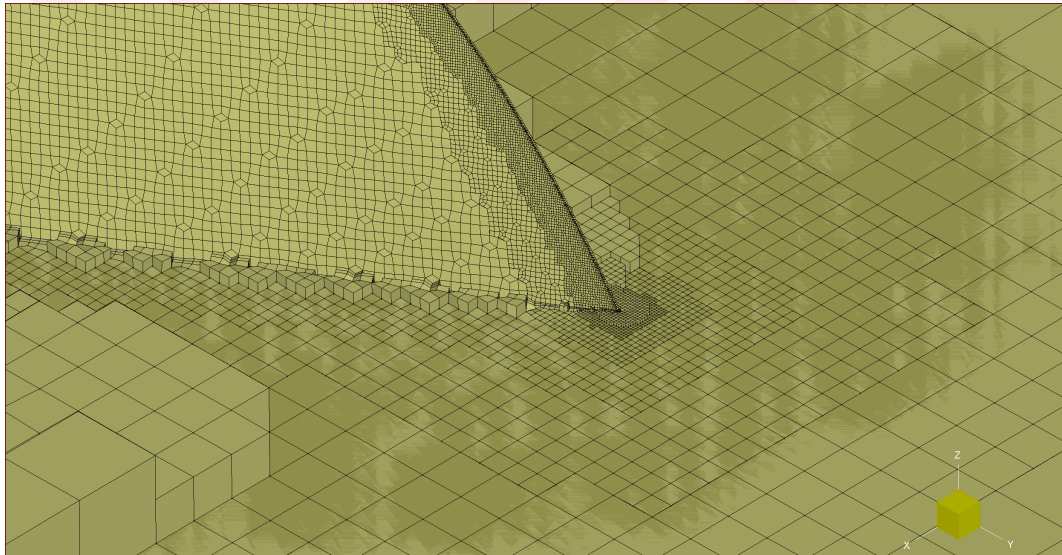


Figure 4.22 : Viscous layers at trailing edge of the propeller.

In order to prepare open water case for OpenFOAM, static and rotating parts have been converted from Numeca Hexpress/Hybrid to OpenFOAM separately. The interfaces between static and rotating parts must be coupled to be computed.

As it is mentioned before, mesh quality metrics have importance on accuracy, stability and efficiency of the numerical computations. In order to provide there criterias, neither approximate mesh structure is generated nor it has non-orthogonal, skewed and negative cells are generated must be checked. Table 4.5 shows mesh quality of the generated mesh in Numeca/Hexpress.

Table 4.5 : Mesh information.

Mesh	Mesh Criteria		Mesh Status
Coarse	Number of points	-	3225132
	Number of faces	-	9083384
	Number of cells	-	2930677
	Mesh non-orthogonality	< 80	76.6219
	Mesh skewness	< 4	3.17324
	Mesh aspect ratio	< 1000	19.9399
Medium	Number of points	-	5843092
	Number of faces	-	16336742
	Number of cells	-	5249919
	Mesh non-orthogonality	< 80	75.2437
	Mesh skewness	< 4	3.98637
	Mesh aspect ratio	< 1000	10.2697
Fine	Number of points	-	11776624
	Number of faces	-	32573897
	Number of cells	-	10404791
	Mesh non-orthogonality	< 80	75.4715
	Mesh skewness	< 4	3.90518
	Mesh aspect ratio	< 1000	9.24449

4.2.2 Computation case setup

In order to obtain hydrodynamic performance of the DTMB 4119 propeller, an extensive series of open water tests are carried out. Therefore, open water performance of the model is specified under experiment procedures. The rotational velocity for open water case is considered constant as 10 rps. whereas the inflow velocity of the propeller varies to generate different advance coefficients. Open water computations are performed under the operating conditions given in Table 4.2.

Physical conditions for open water computations are given below in Table 4.6.

Table 4.6 : Water properties under experimental conditions.

Water Properties		Value
Mass density	$\rho, [kg/m^3]$	1000
Kinematic viscosity	$\nu, [m^2/s]$	8.899×10^{-7}

Open water simulations are conducted using SIMPLE-Consistent algorithm with MRF technique. The aim of the MRF technique is to simulate rotation of the propeller by adding Coriolis acceleration term in the RANS equations. It follows from this that the mesh is static, not dynamic. So as to calculate thrust and torque values which are acting on propeller blades, *forces* function is used. In order to calculate wall shear

stresses, SST $k - \omega$ turbulence model is used with extended wall-functions. Numerical schemes and solution algorithms which are used, available in Appendix. Due to the fact that SIMPLE-Consistent algorithm is used, pressure field is under-relaxed with a factor of 0.9 while momentum and turbulence equations are used with an under-relaxation factor of 0.7.

To state our simulation as a well-posed problem, primary and secondary boundary conditions must be defined correctly. For this purpose at inlet, fixedValue is used for velocity as Dirichlet boundary condition, although zeroGradient is used for pressure as Neumann boundary condition. However at outlet inletOutlet derived boundary condition is used for velocity which prevents problem being ill-posed in event of reverse flow, where pressure is defined as fixedValue. Since the domain boundaries as far as an unaffected manner from rotational flow, they could be expressed as no-slip boundaries. Propeller blades, hub and shaft are modeled as walls mathematically, therefore fixedValue is used for velocity although zeroGradient is used for pressure on these patches. To sum up, boundary conditions are given below in Table 4.7.

Table 4.7 : The boundary conditions in open water condition.

The boundary	p	U	k	omega
inlet	zeroGradient	fixedValue	fixedValue	fixedValue
outlet	fixedValue	inletOutlet	inletOutlet	inletOutlet
outerWall	zeroGradient	noSlip	kqRWallFunction	omegaWallFunction
propellerBlade	zeroGradient	fixedValue	kqRWallFunction	omegaWallFunction
propellerHub	zeroGradient	fixedValue	kqRWallFunction	omegaWallFunction
propellerShaft	zeroGradient	fixedValue	kqRWallFunction	omegaWallFunction
AMI0/AMI1	cyclicAMI	cyclicAMI	cyclicAMI	cyclicAMI

4.2.3 Computations and analysis

For open water computations rotational speed is constant and inlet velocity of the flow domain is increased respectively to generate the desired advance coefficient values. In this current work, SST $k - \omega$ turbulence model is used to get better wake distribution. Extended wall functions with small values are used to compute wall shear stresses as seen from Table 4.7.

Generally, there is no strict criteria for residuals about convergence criteria, therefore mathematically, some assumptions could be considered for steady flows. In this point of view, in our cases we considered the residual value of equations less than 10^{-4} and

this convergence criteria is satisfied for residuals after less than 500 iterations. Total number of iterations are fixed to 2500 for each case, so as to stabilize the residuals, thrust and torque convergence. Computed force and moment values change by less than %1 which could be considered acceptable.

4.2.3.1 Open water computations with coarse grid generation

The convergence graphs for thrust and torque values are given below.

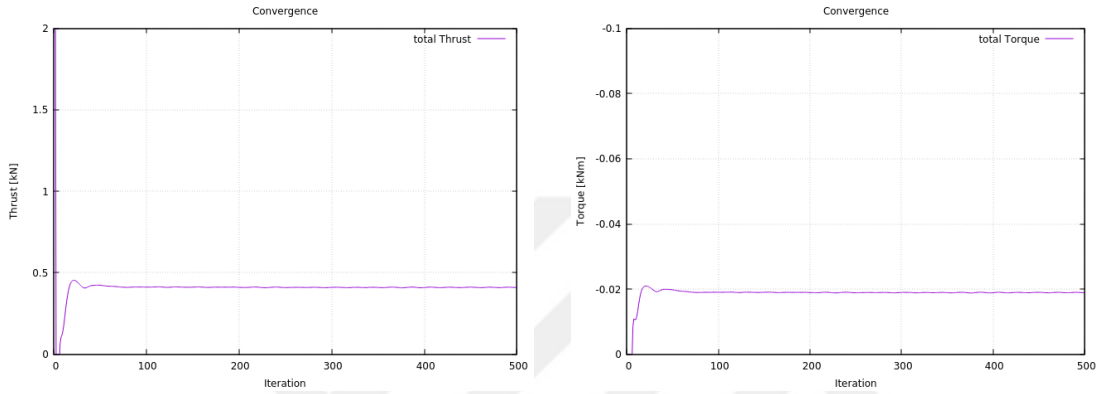


Figure 4.23 : Convergence graphs for $J = 0.1$.

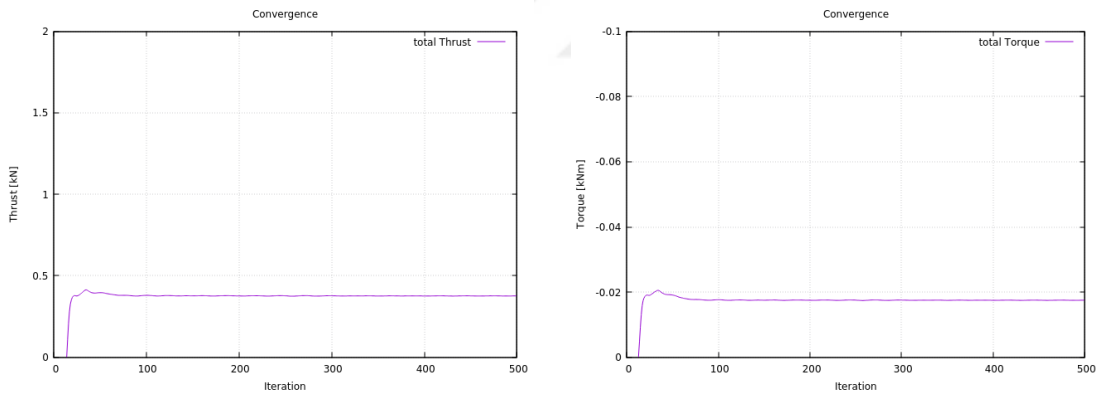


Figure 4.24 : Convergence graphs for $J = 0.2$.

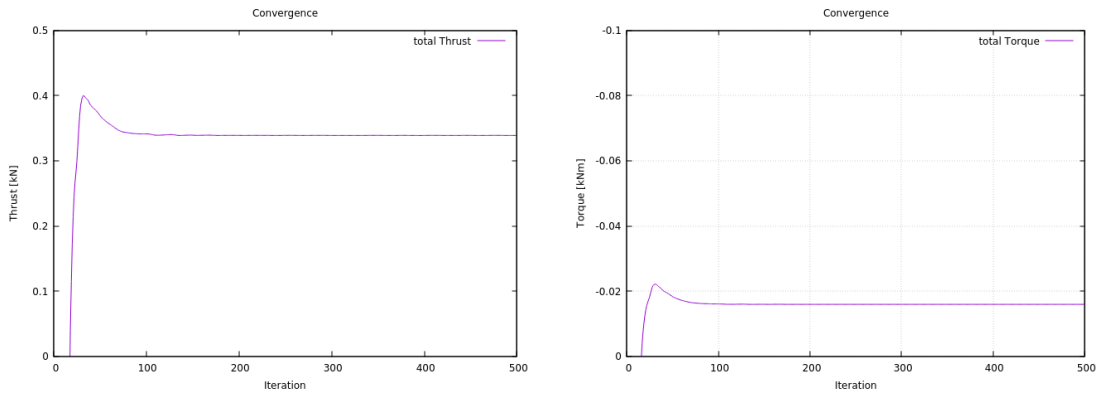


Figure 4.25 : Convergence graphs for $J = 0.3$.

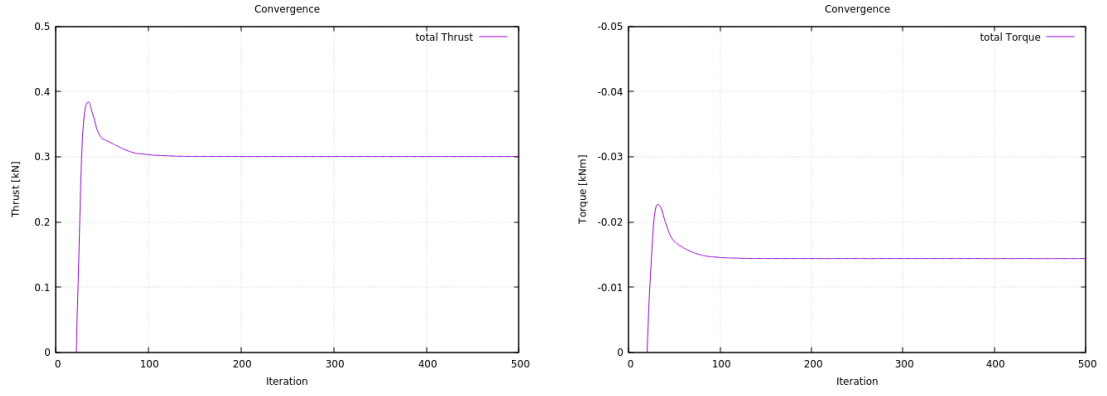


Figure 4.26 : Convergence graphs for $J = 0.4$.

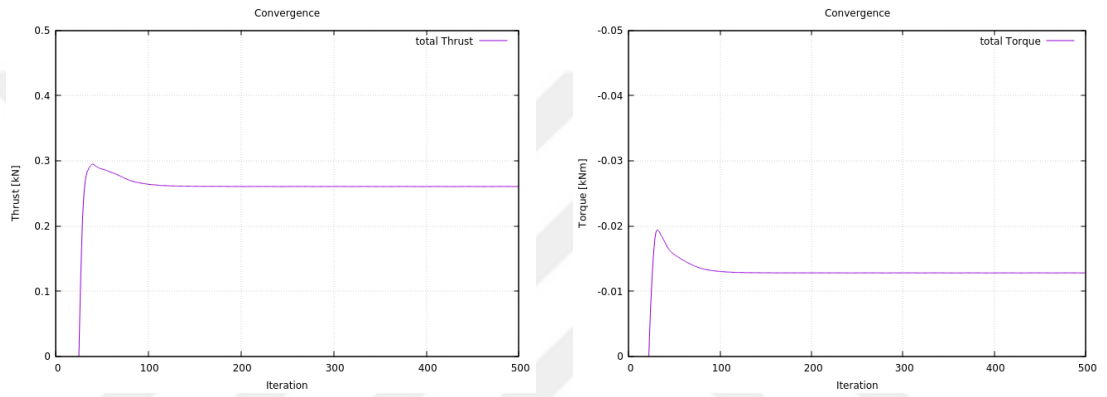


Figure 4.27 : Convergence graphs for $J = 0.5$.

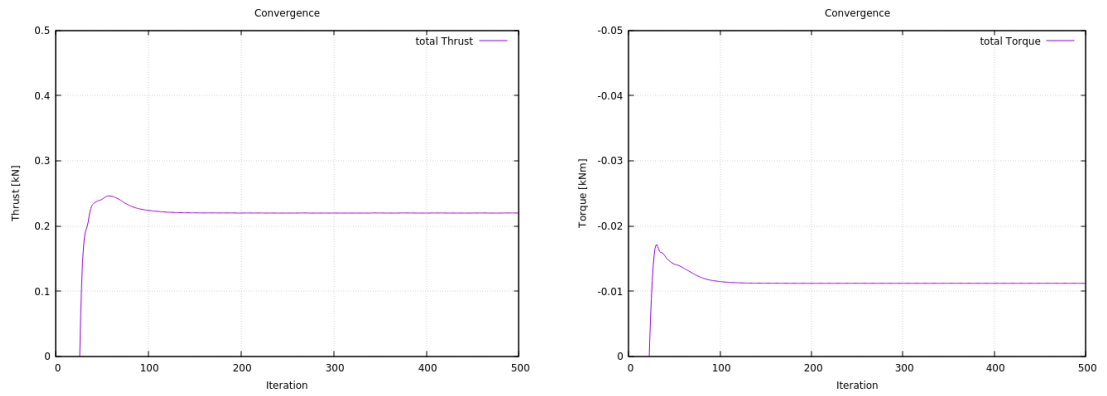


Figure 4.28 : Convergence graphs for $J = 0.6$.

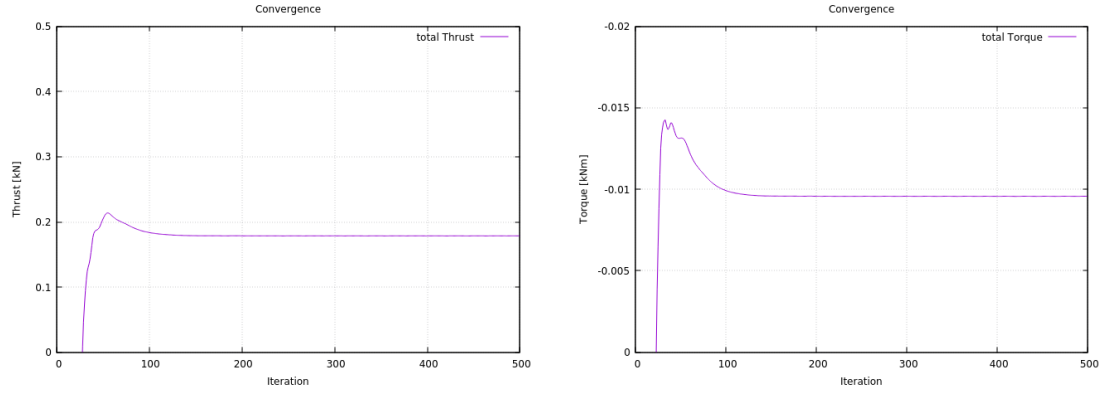


Figure 4.29 : Convergence graphs for $J = 0.7$.

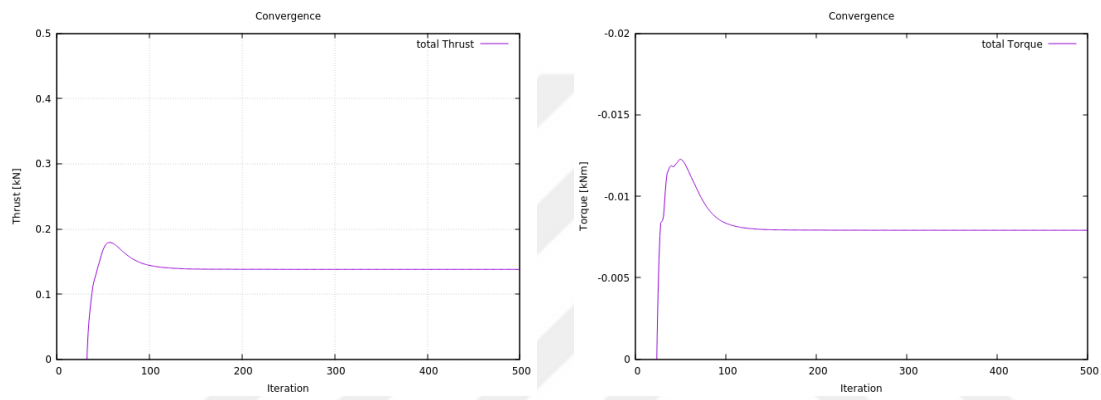


Figure 4.30 : Convergence graphs for $J = 0.8$.

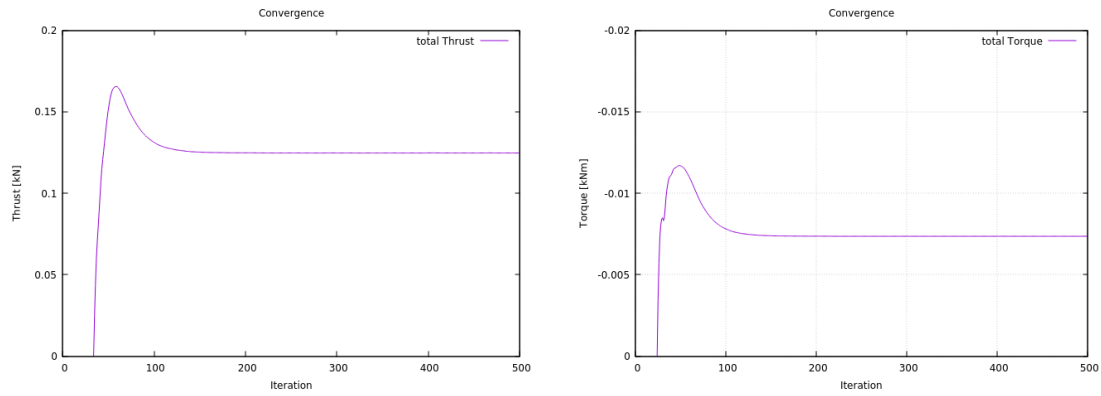


Figure 4.31 : Convergence graphs for $J = 0.833$.

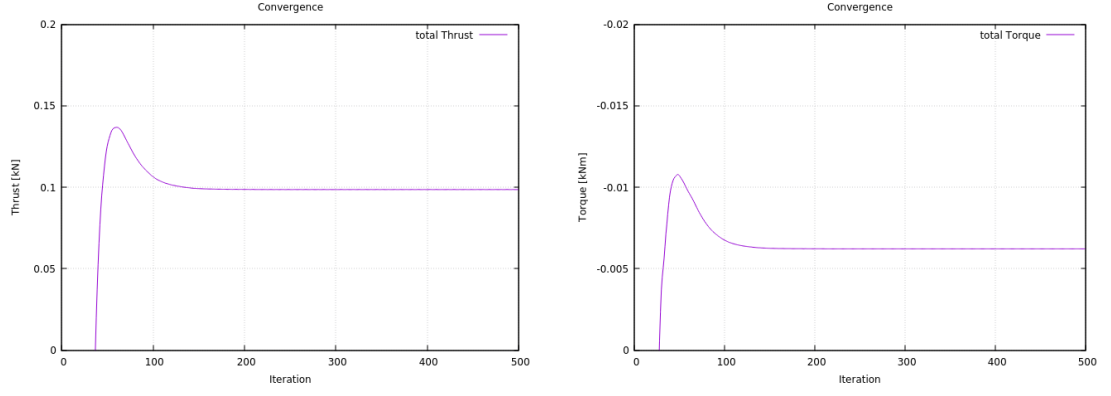


Figure 4.32 : Convergence graphs for $J = 0.9$.

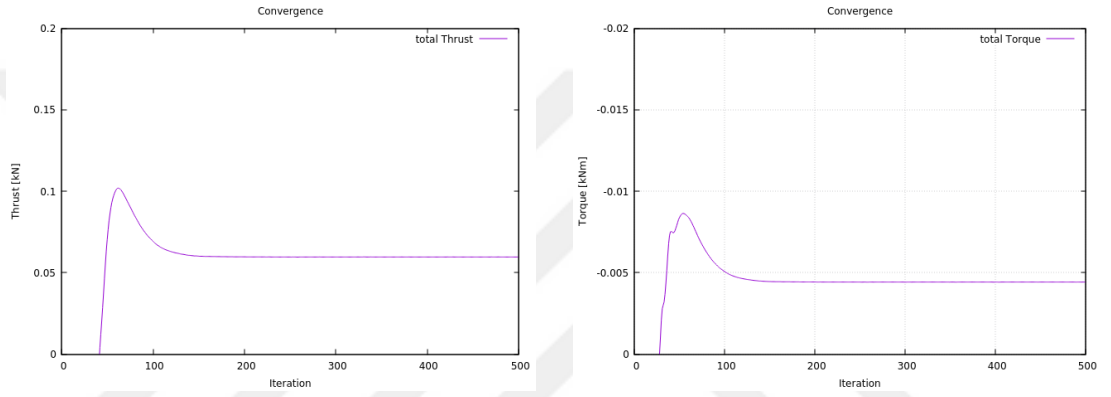


Figure 4.33 : Convergence graphs for $J = 1.0$.

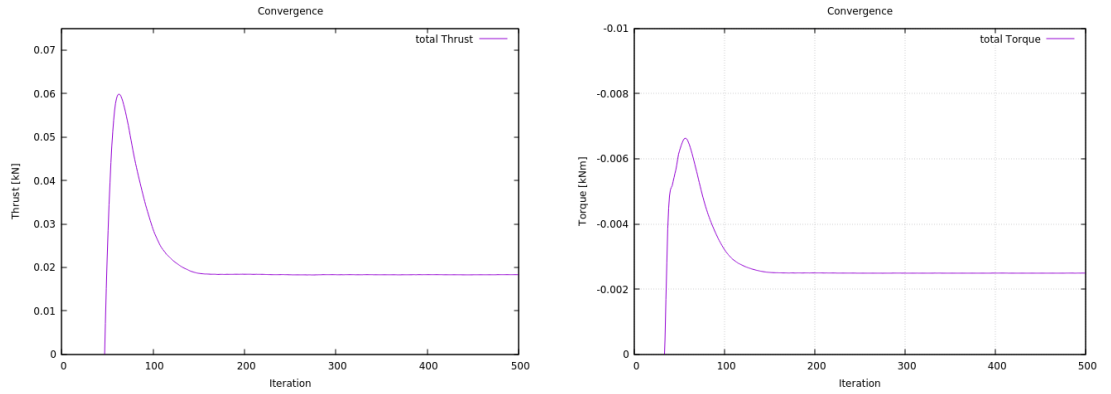


Figure 4.34 : Convergence graphs for $J = 1.1$.

During the modelling open water case with AMI, the connection of interfaces have effects on convergence of the simulation. This means poor interpolation between static and rotating zones along the interface could induces numerical errors. In our cases all convergence requirements are satisfied. For verification of the numerical computation, pressure coefficient contours on the propeller are given below.

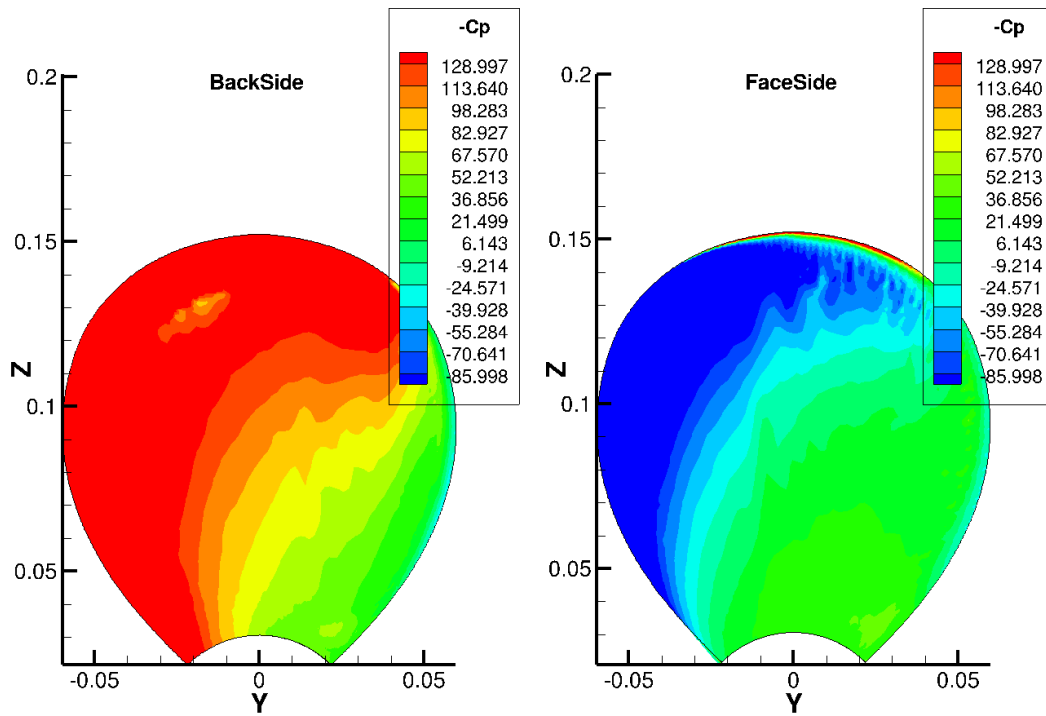


Figure 4.35 : Distribution of pressure coefficient on the propeller for $J = 0.1$.

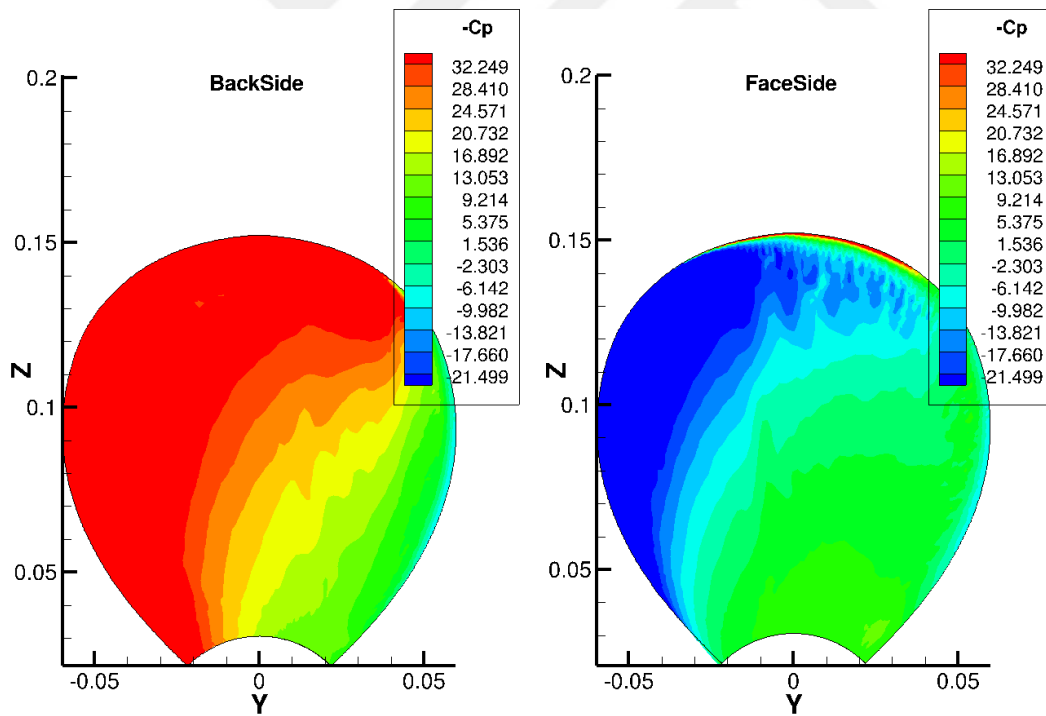


Figure 4.36 : Distribution of pressure coefficient on the propeller for $J = 0.2$.

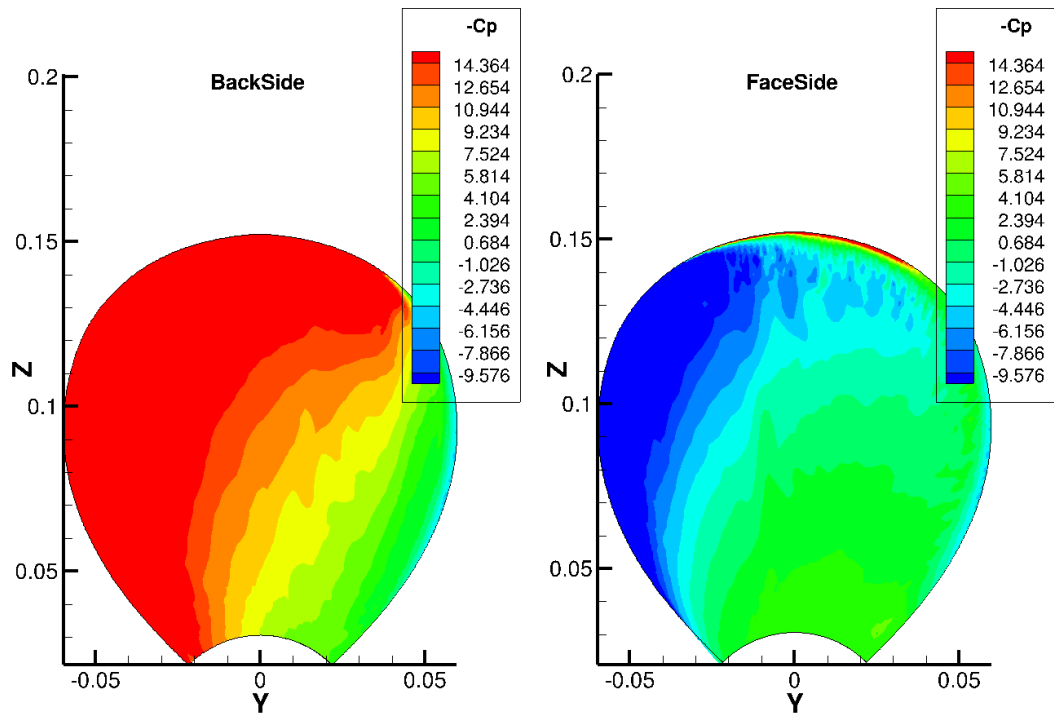


Figure 4.37 : Distribution of pressure coefficient on the propeller for $J = 0.3$.

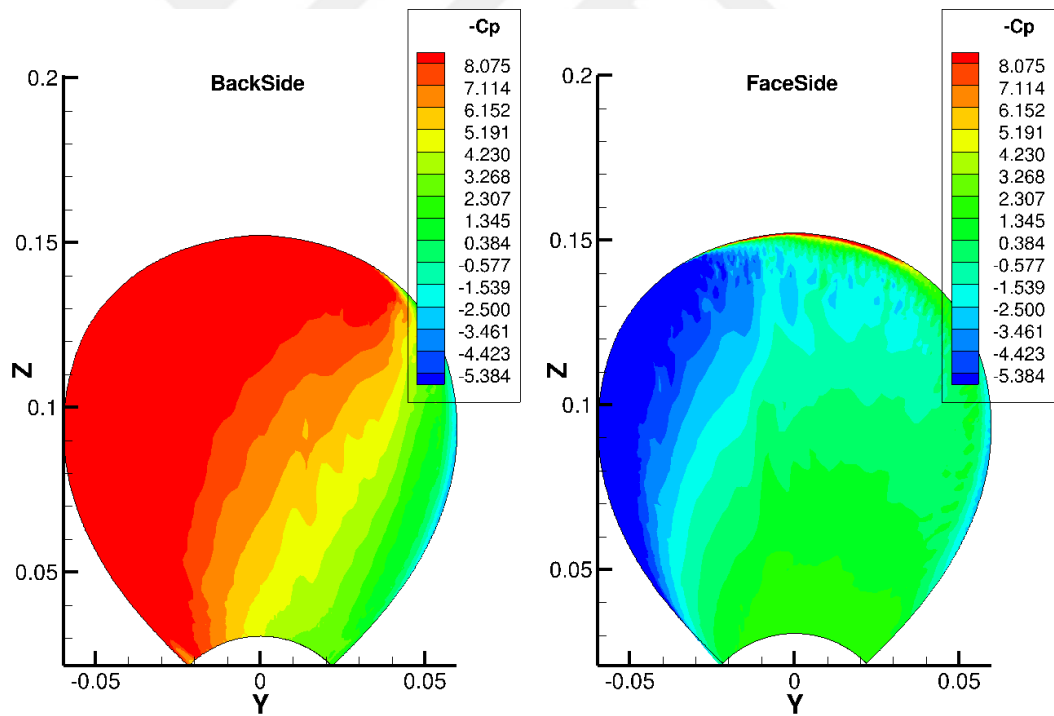


Figure 4.38 : Distribution of pressure coefficient on the propeller for $J = 0.4$.

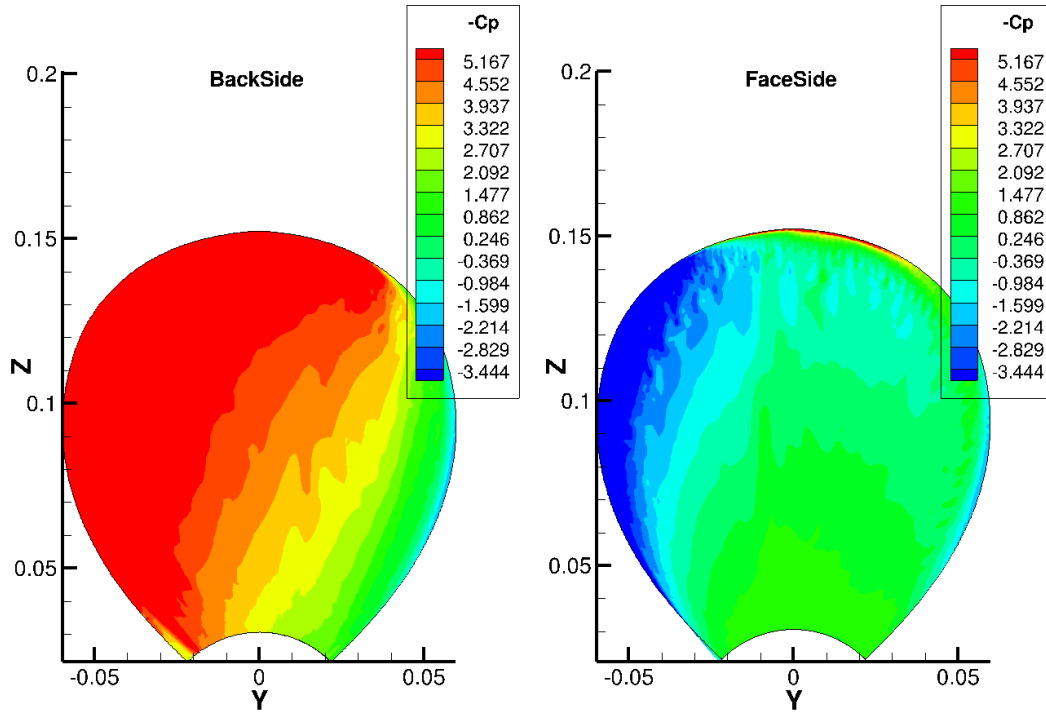


Figure 4.39 : Distribution of pressure coefficient on the propeller for $J = 0.5$.

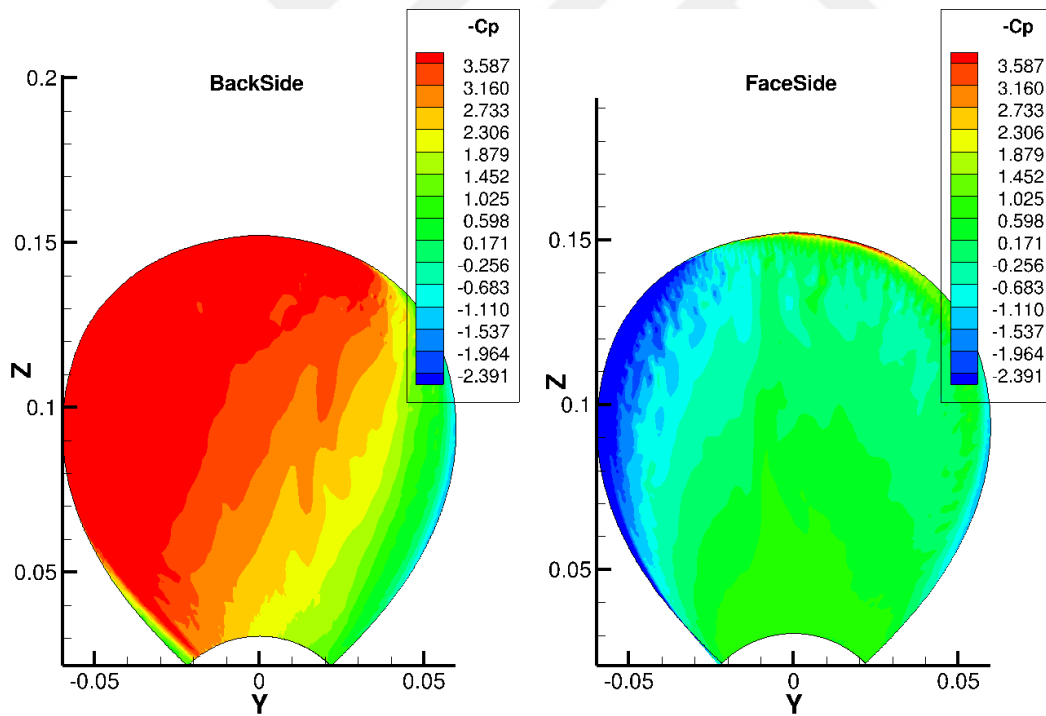


Figure 4.40 : Distribution of pressure coefficient on the propeller for $J = 0.6$.

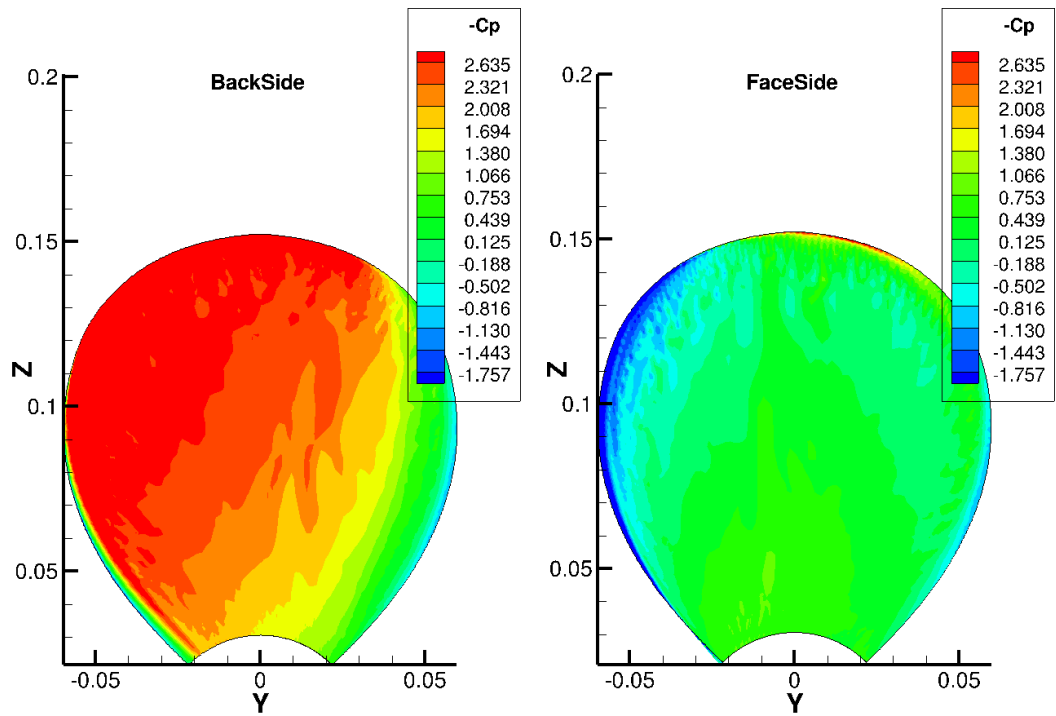


Figure 4.41 : Distribution of pressure coefficient on the propeller for $J = 0.7$.

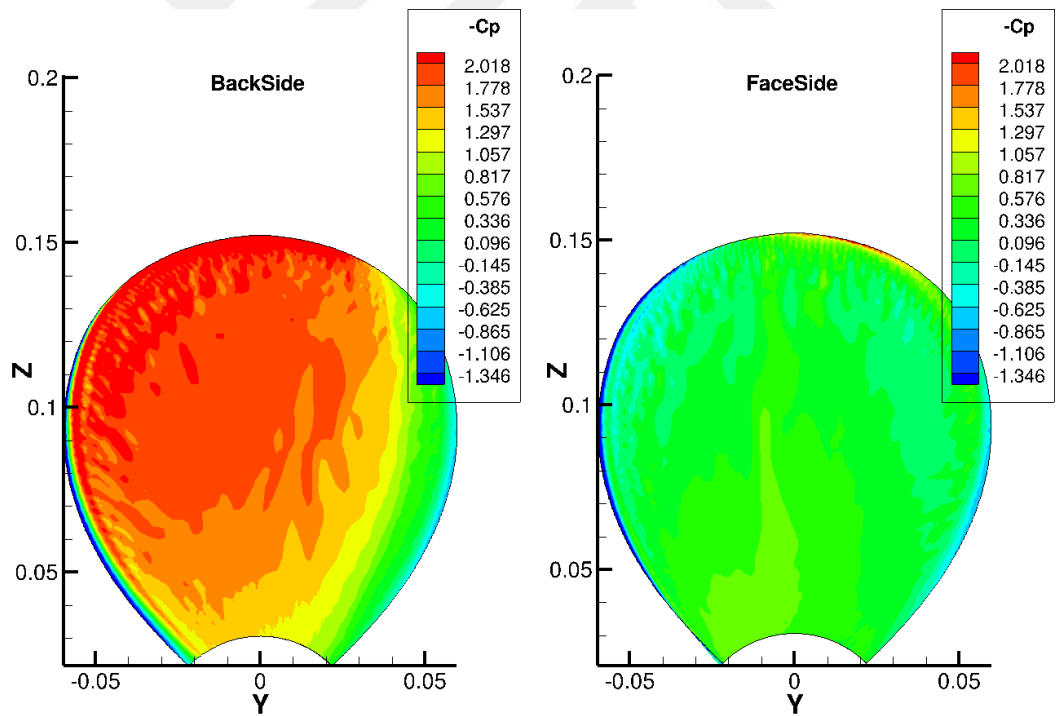


Figure 4.42 : Distribution of pressure coefficient on the propeller for $J = 0.8$.

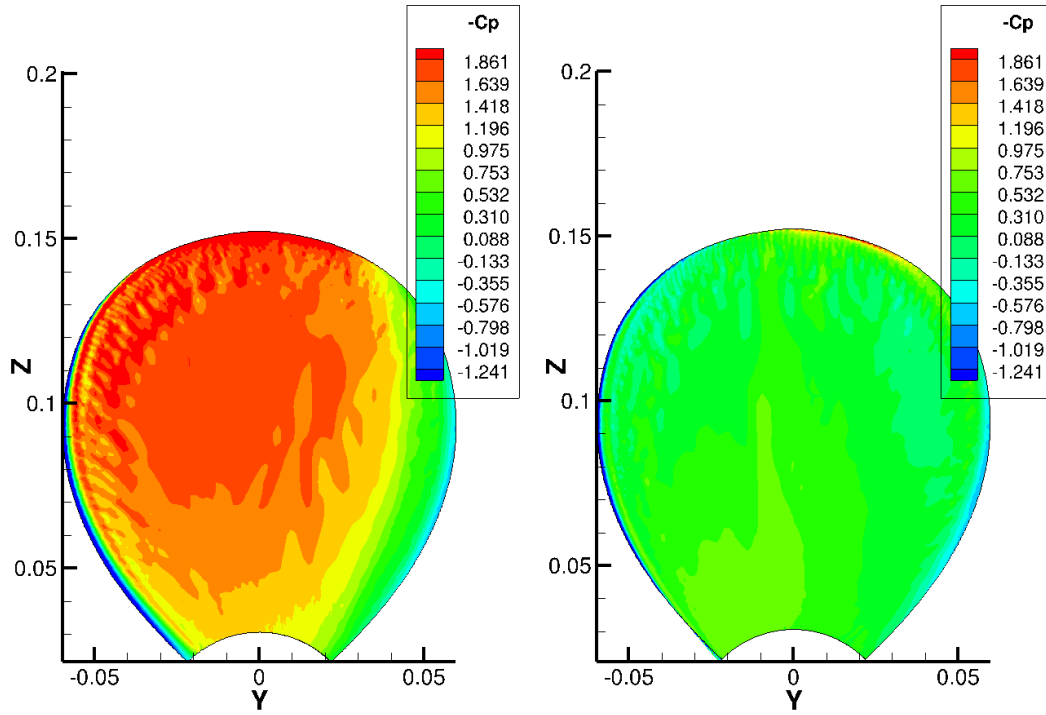


Figure 4.43 : Distribution of pressure coefficient on the propeller for $J = 0.833$.

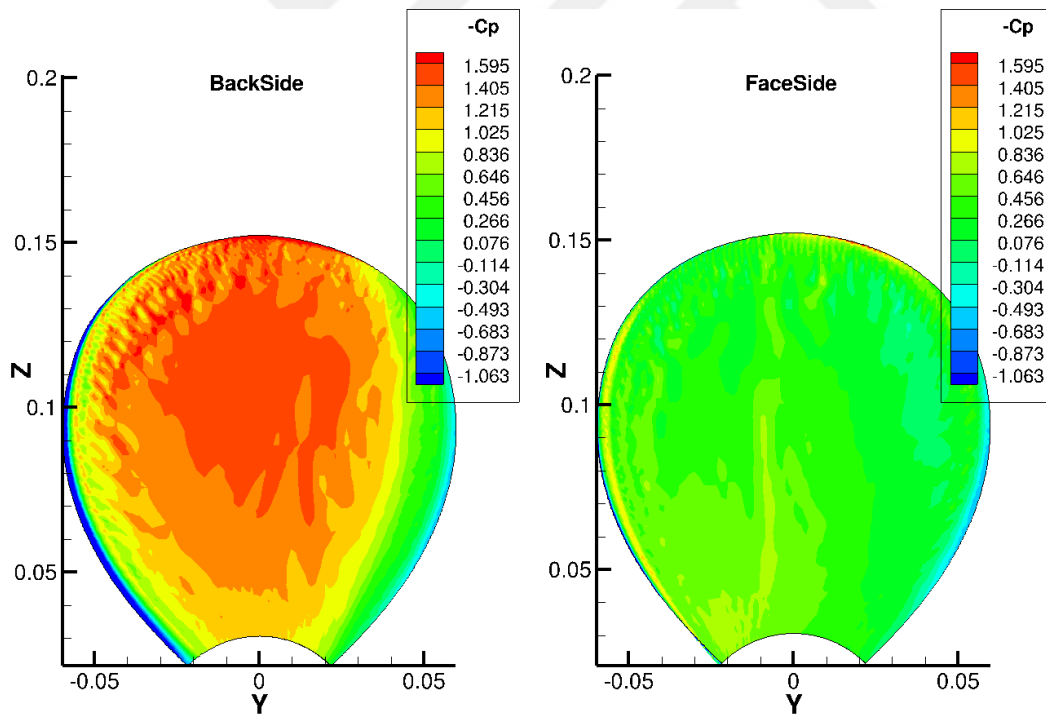


Figure 4.44 : Distribution of pressure coefficient on the propeller for $J = 0.9$.

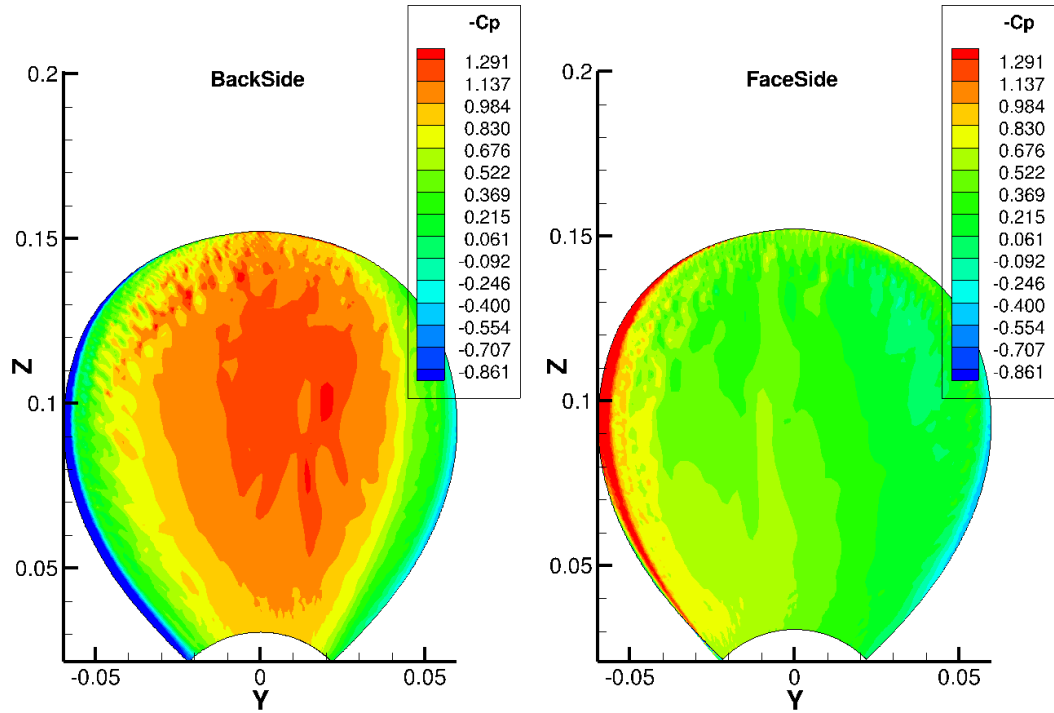


Figure 4.45 : Distribution of pressure coefficient on the propeller for $J = 1.0$.

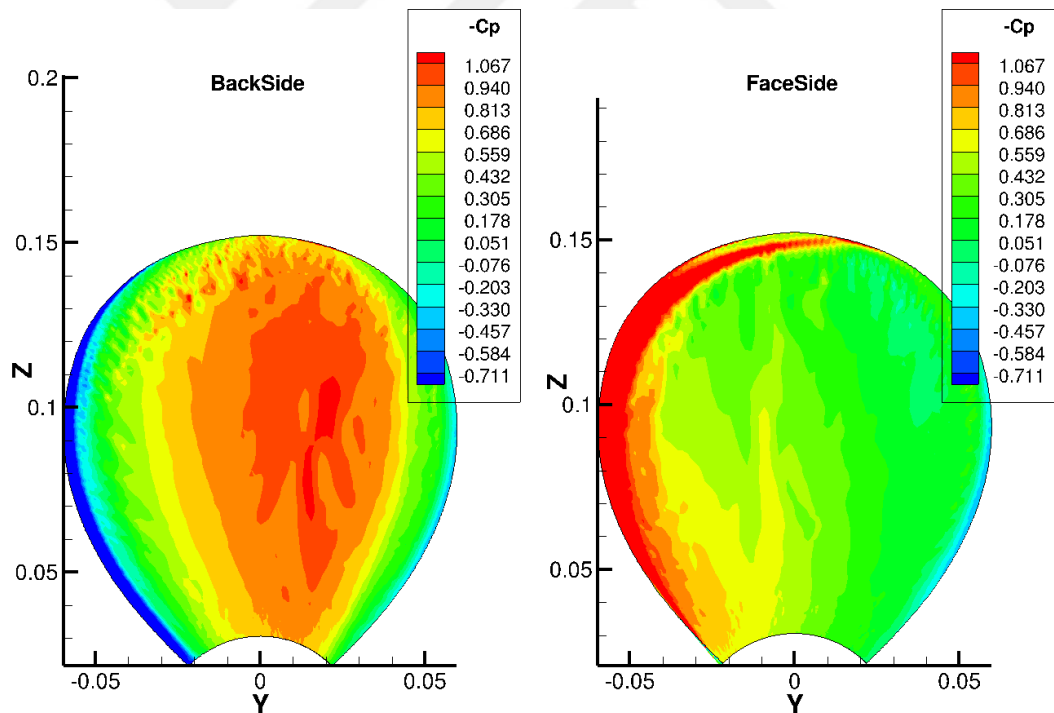


Figure 4.46 : Distribution of pressure coefficient on the propeller for $J = 1.1$.

Numerical computation results for different advance coefficients of DTMB 4119 propeller are compared with experimental results in Figure 4.40.

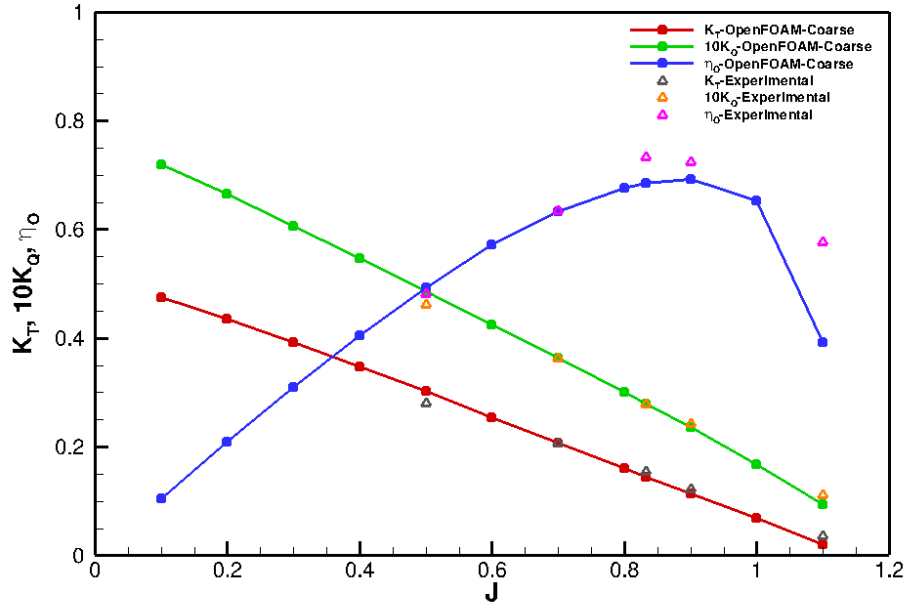


Figure 4.47 : Comparison of K_T , $10K_Q$ and η_O values using CFD solution for coarse grid with experimental data.

Table 4.8 shows the comparison of numerical computation and experimental results.

Table 4.8 : The comparison of numerical and experimental results.

J	K_T (CFD)	K_T (Exp.)	$10K_Q$ (CFD)	$10K_Q$ (Exp.)	η_O (CFD)	η_O (Exp.)
0.100	0.475336	-	0.719629	-	0.105126	-
0.200	0.436106	-	0.665071	-	0.208725	-
0.300	0.392496	-	0.605466	-	0.309518	-
0.400	0.347973	-	0.546581	-	0.405295	-
0.500	0.301734	0.285	0.486161	0.477	0.493894	0.489
0.600	0.254565	-	0.425377	-	0.571475	-
0.700	0.206873	0.200	0.363813	0.360	0.633497	0.632
0.800	0.159777	-	0.300987	-	0.675889	-
0.833	0.144538	0.146	0.279873	0.280	0.684678	0.692
0.900	0.114263	0.120	0.236510	0.239	0.692022	0.725
1.000	0.069148	-	0.168395	-	0.653542	-
1.100	0.021203	0.034	0.094595	0.106	0.392418	0.575

Table 4.9 shows the comparison of numerical errors between computational fluid dynamics method and experimental results. As it is seen below, the numerical predictions for K_T and $10K_Q$ values are quite acceptable, however the larger J gives the larger error. While modelling of open water simulation with AMI, it is noted that non-conformal faces would have an effect on the computation. This signifies interpolation errors along the interfaces between rotating and static regions might cause

sudden numerical predictions. But if absolute errors are discussed, it can be seen that percentage errors are under %2.00.

Table 4.9 : The comparison of relative numerical percentage errors between computational fluid dynamics method and experimental results.

J	% ΔK_T	% $\Delta 10K_Q$	% $\Delta \eta_O$
0.500	-4.491049	-0.484460	-1.108565
0.700	-3.184116	-0.533382	-0.515062
0.833	0.361331	-0.087583	0.550735
0.900	3.688999	0.536050	3.945398
1.100	31.753707	8.578914	27.096108

4.2.3.2 Open water computations with medium grid generation

The convergence graphs for thrust and torque values are given below.

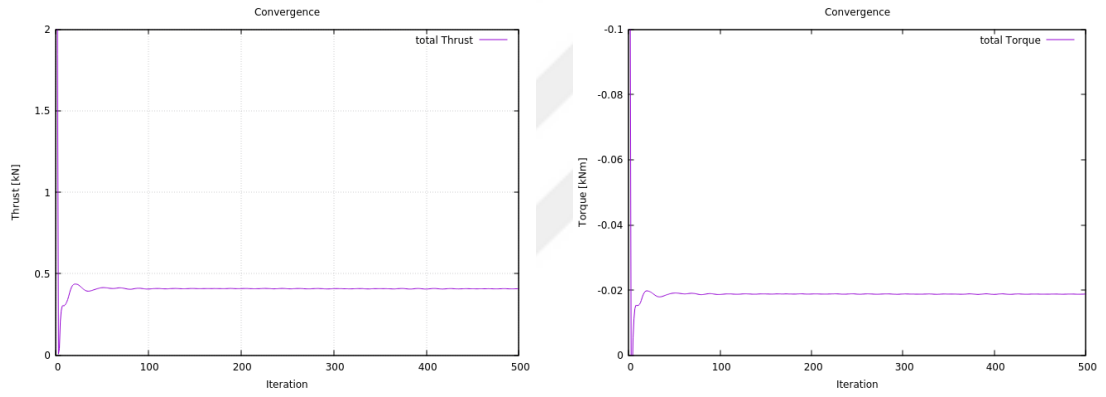


Figure 4.48 : Convergence graphs for $J = 0.1$.

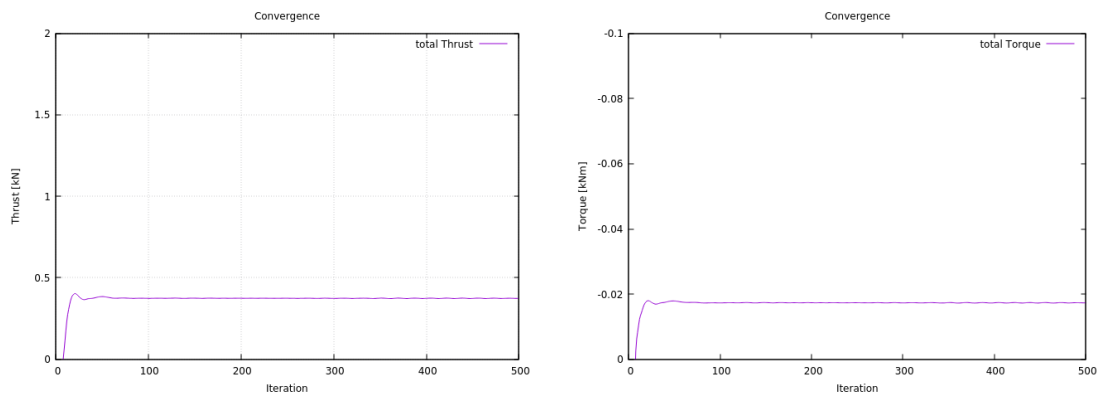


Figure 4.49 : Convergence graphs for $J = 0.2$.

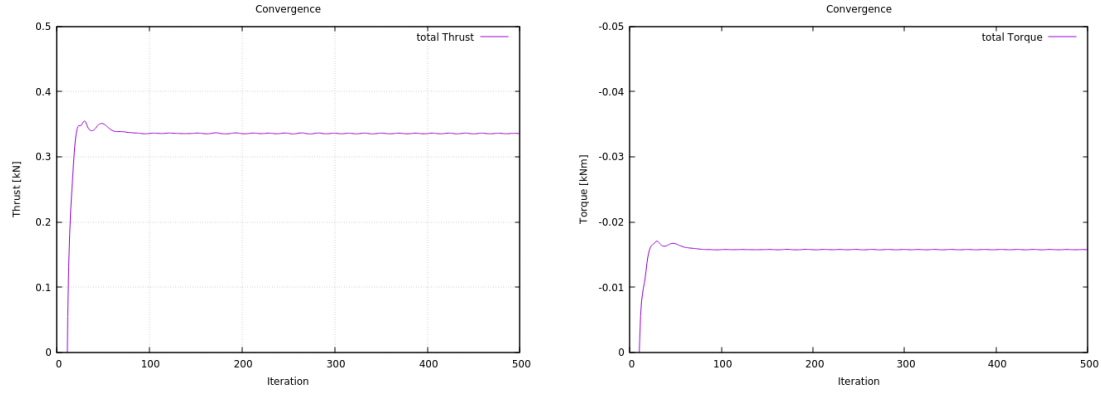


Figure 4.50 : Convergence graphs for $J = 0.3$.

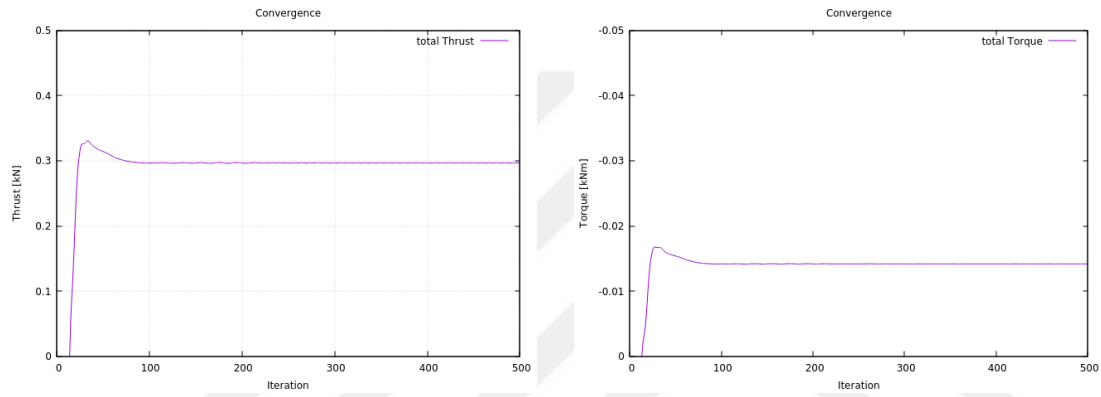


Figure 4.51 : Convergence graphs for $J = 0.4$.

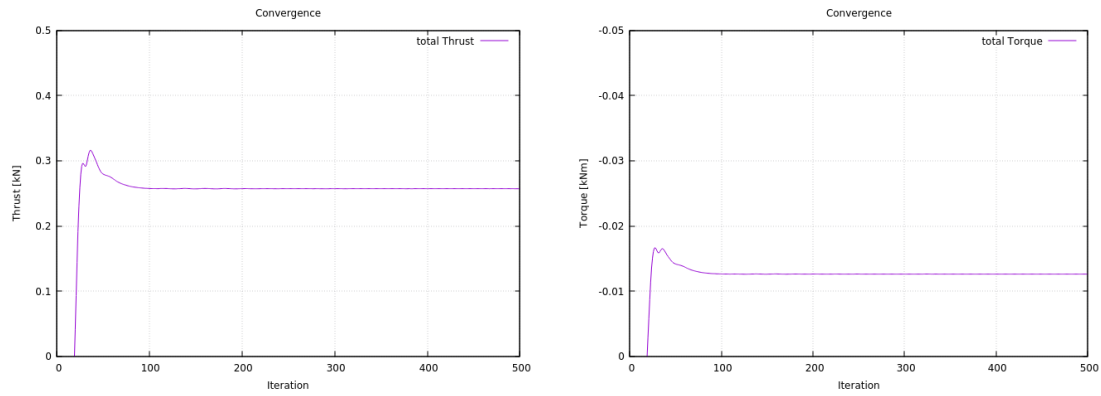


Figure 4.52 : Convergence graphs for $J = 0.5$.

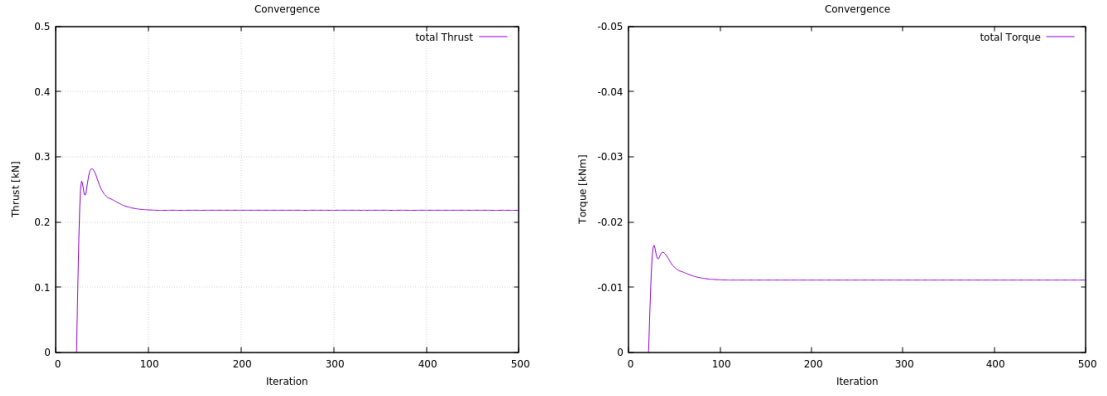


Figure 4.53 : Convergence graphs for $J = 0.6$.

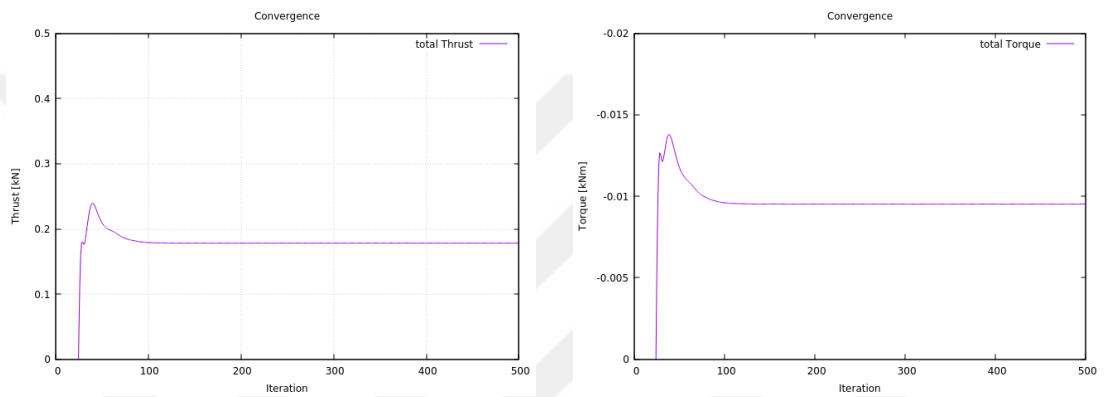


Figure 4.54 : Convergence graphs for $J = 0.7$.

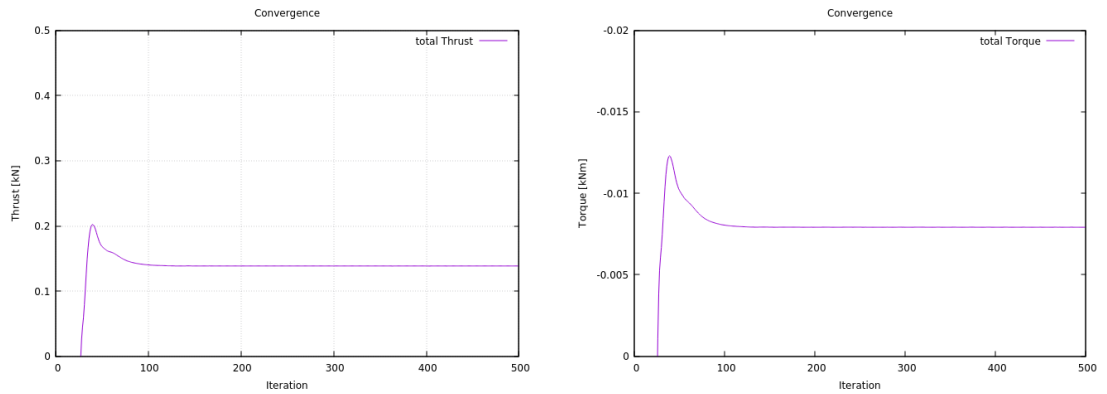


Figure 4.55 : Convergence graphs for $J = 0.8$.

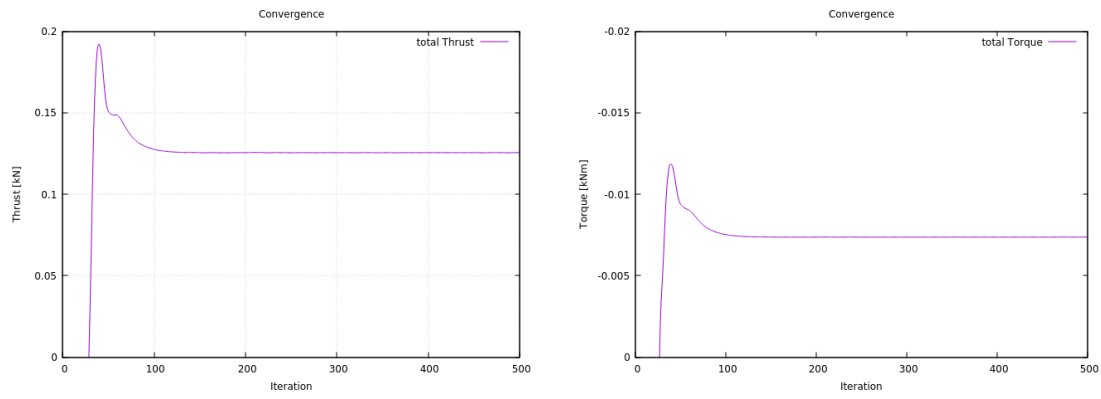


Figure 4.56 : Convergence graphs for $J = 0.833$.

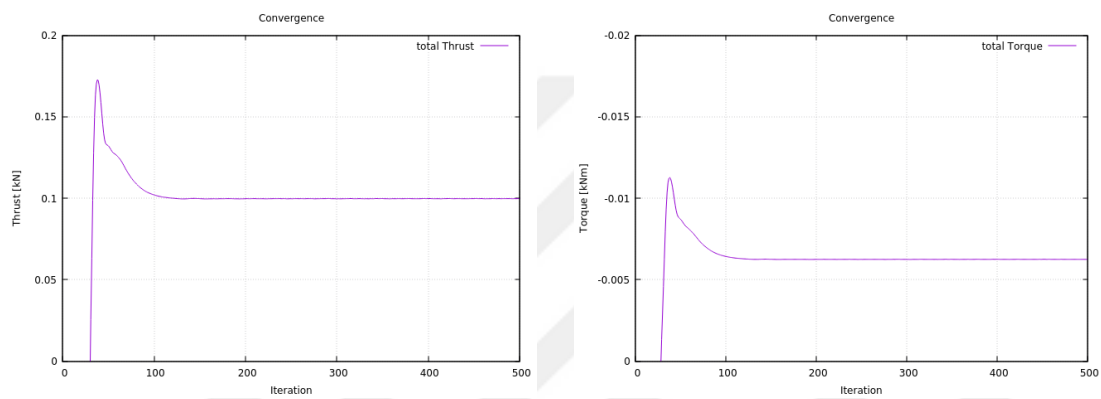


Figure 4.57 : Convergence graphs for $J = 0.9$.

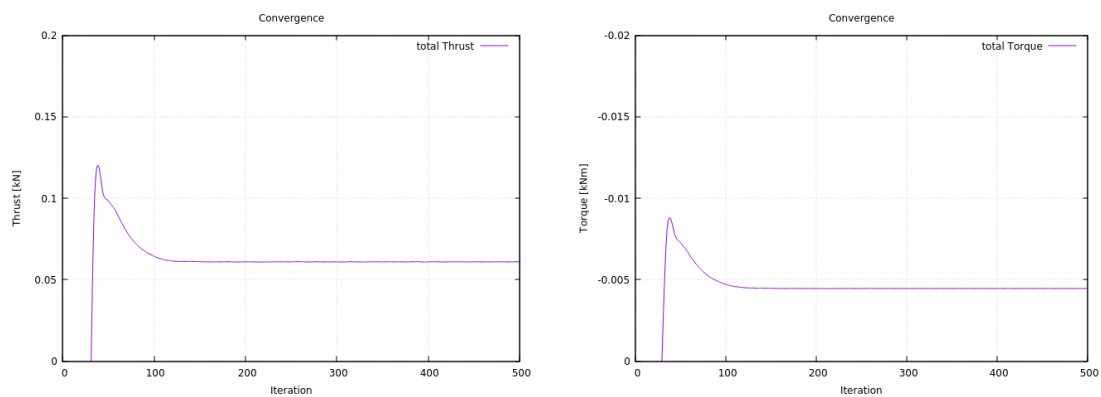


Figure 4.58 : Convergence graphs for $J = 1.0$.

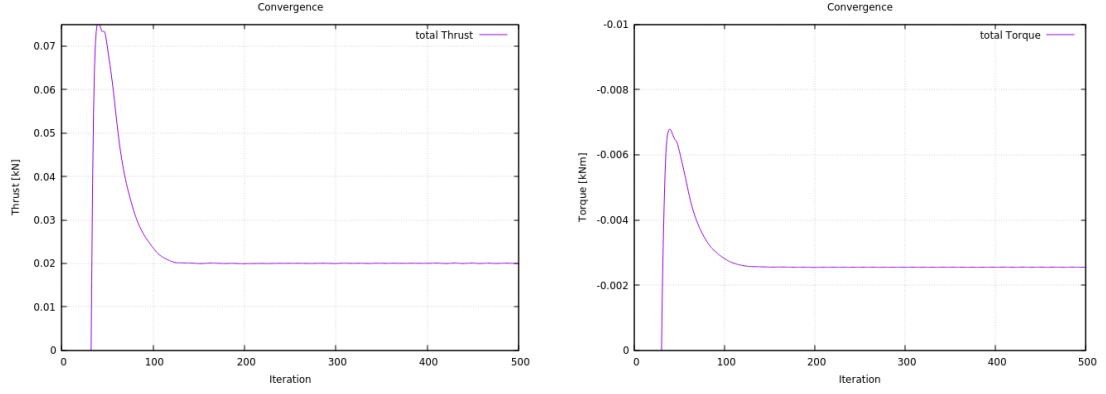


Figure 4.59 : Convergence graphs for $J = 1.1$.

In order to prevent from interpolation errors between refinement regions, the cell size on the interfaces is reduced. So as to decrease cell size, refinement diffusion of the cells around propeller blades is decreased. Though, value of *nNonOrthogonalCorrectors* factor is reduced, continuity errors caused by rough transition between refinement regions. For validation of the numerical computation, pressure coefficient contours on the propeller are given below.

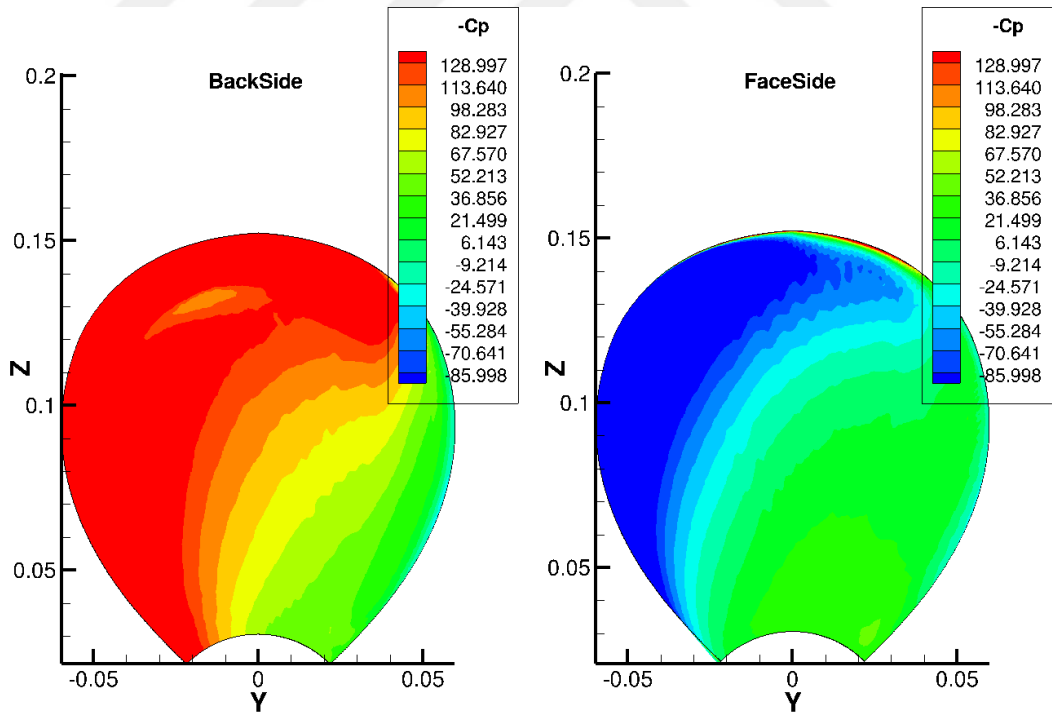


Figure 4.60 : Distribution of pressure coefficient on the propeller for $J = 0.1$.

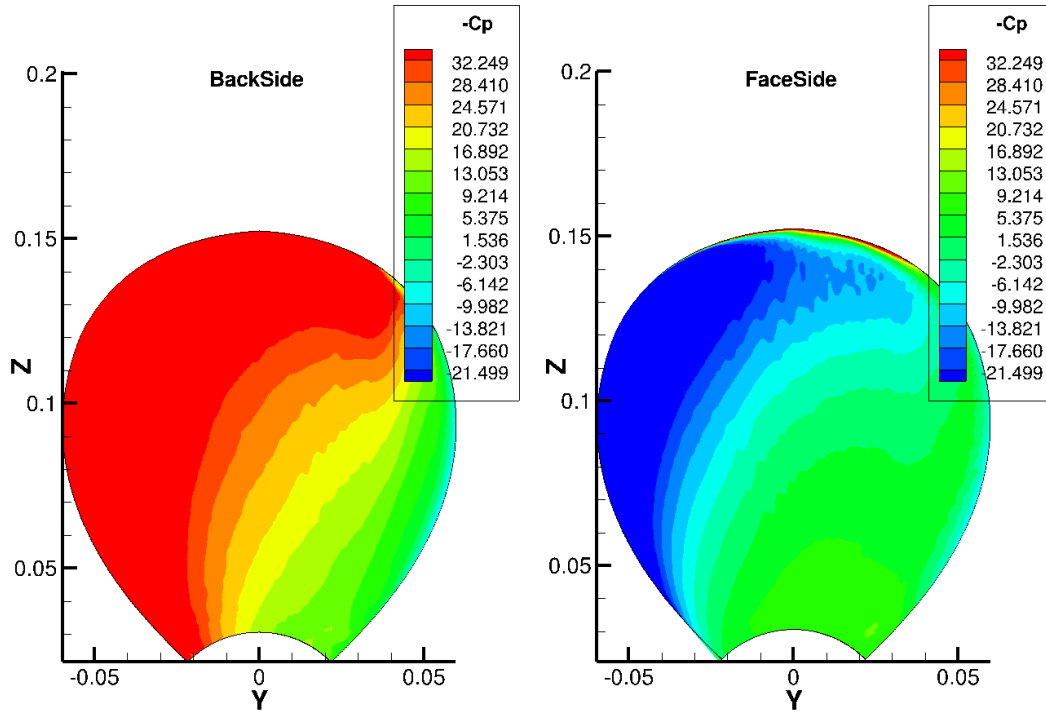


Figure 4.61 : Distribution of pressure coefficient on the propeller for $J = 0.2$.

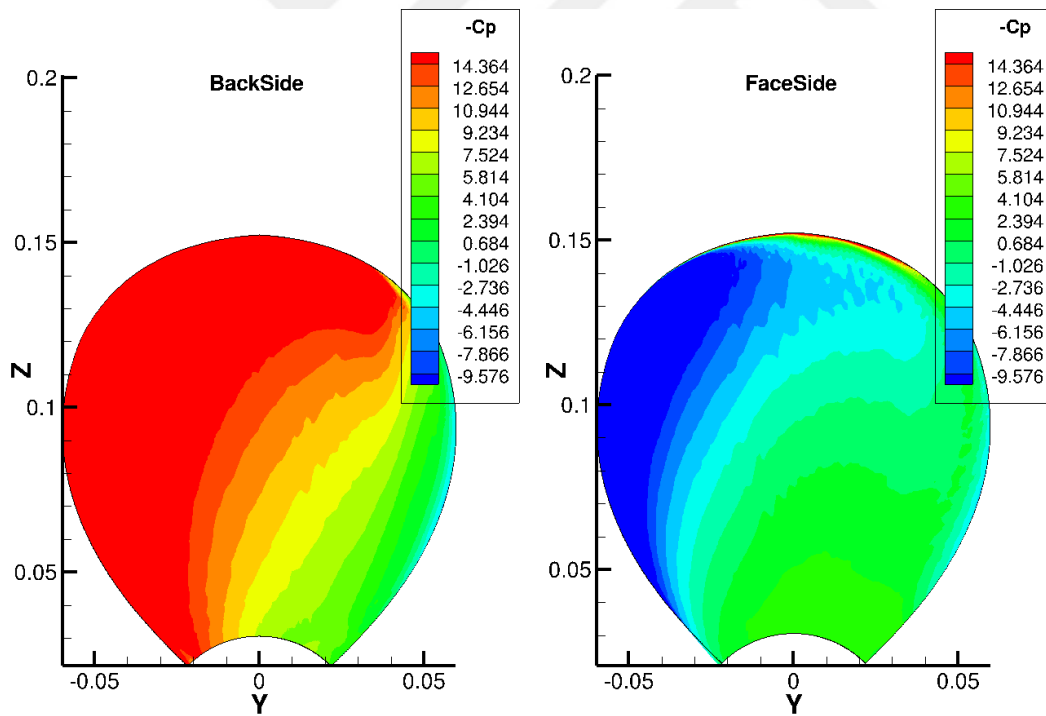


Figure 4.62 : Distribution of pressure coefficient on the propeller for $J = 0.3$.

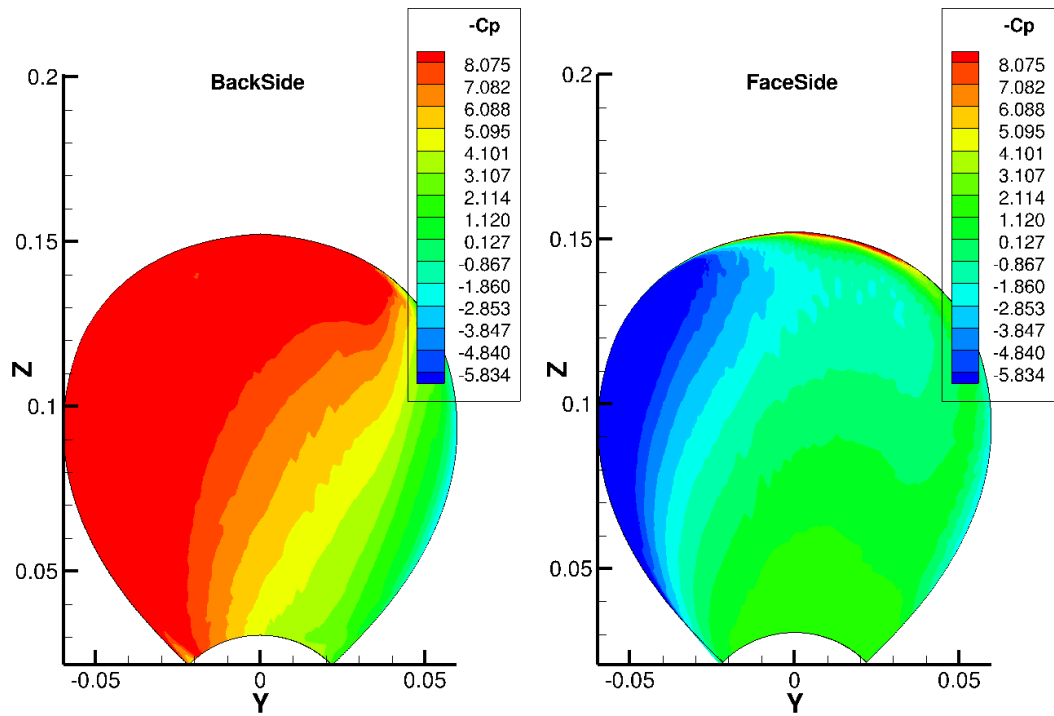


Figure 4.63 : Distribution of pressure coefficient on the propeller for $J = 0.4$.

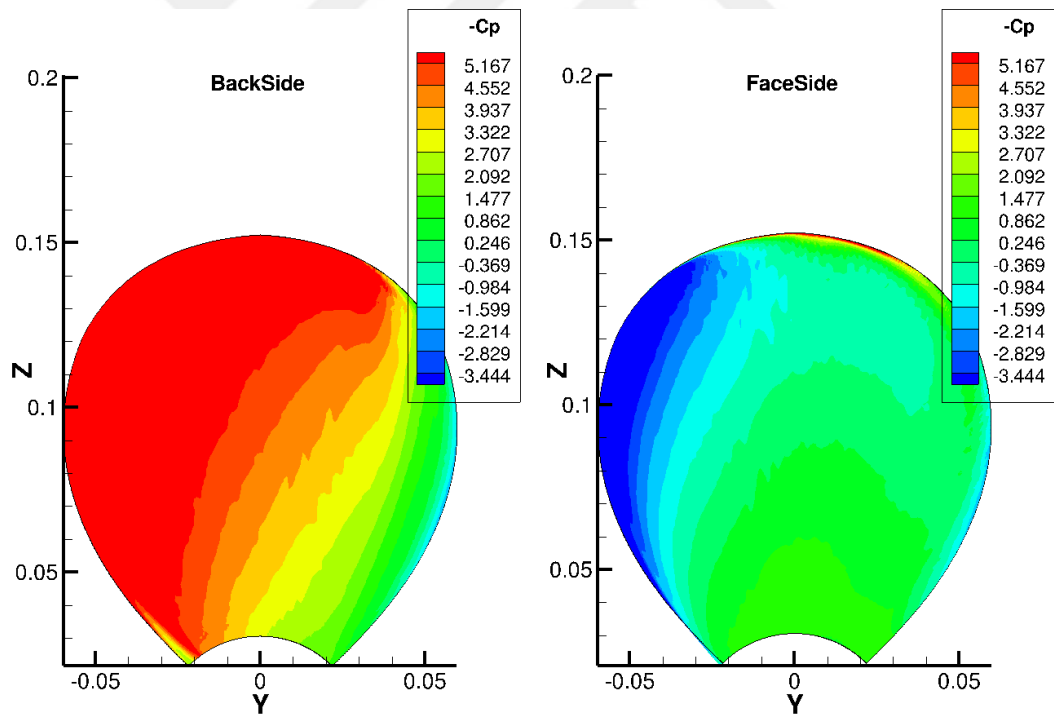


Figure 4.64 : Distribution of pressure coefficient on the propeller for $J = 0.5$.

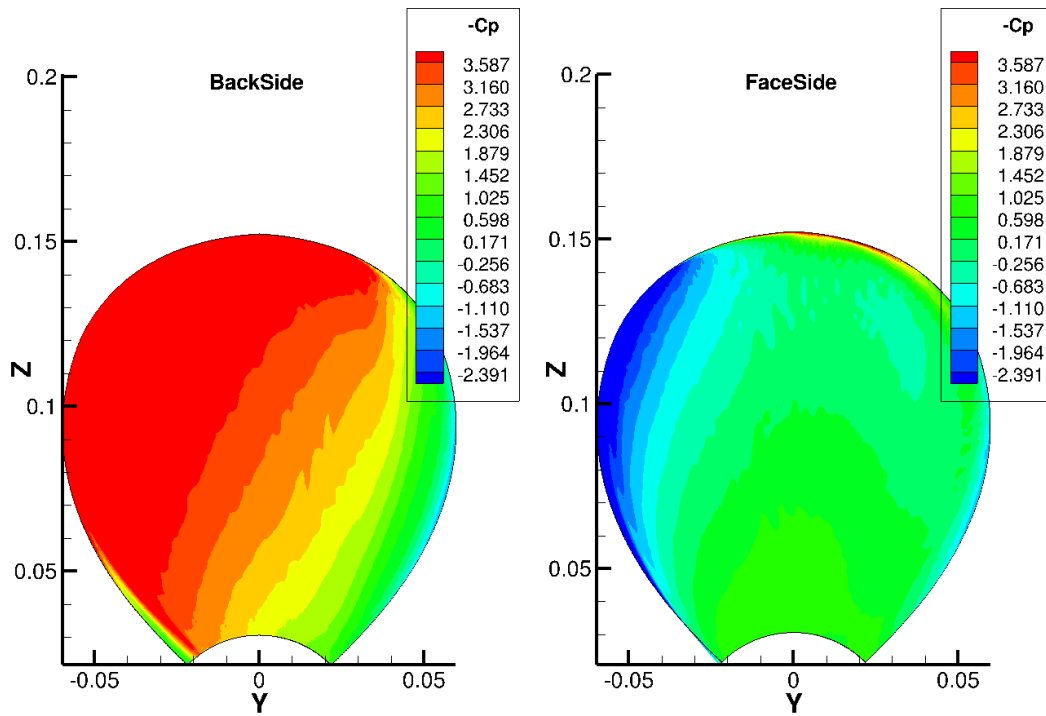


Figure 4.65 : Distribution of pressure coefficient on the propeller for $J = 0.6$.

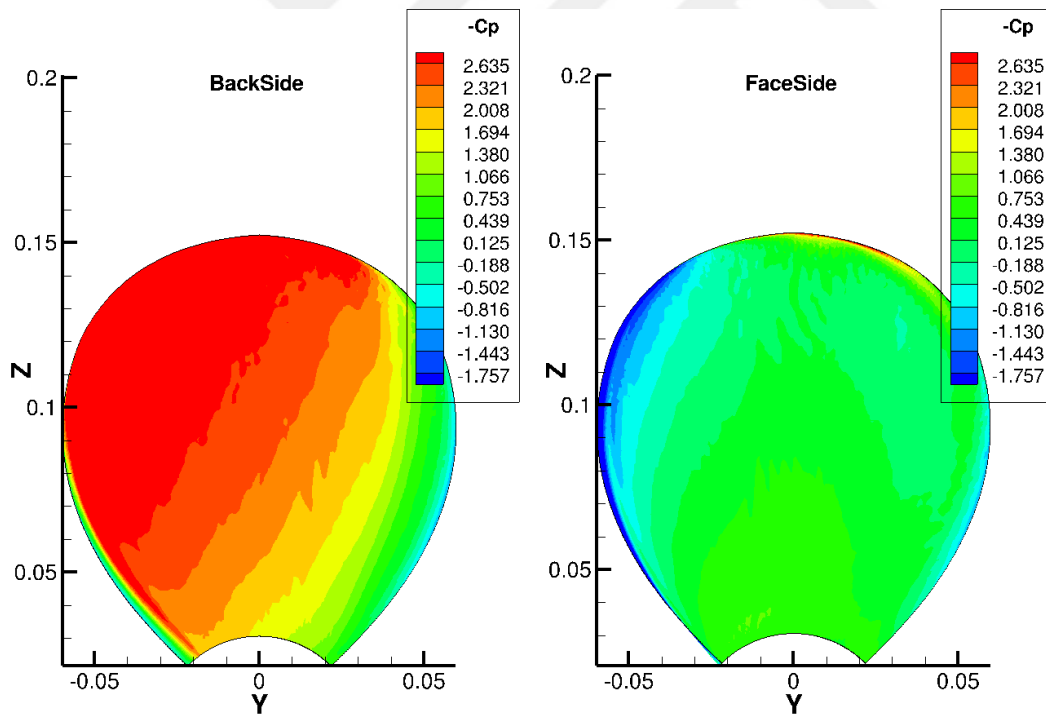


Figure 4.66 : Distribution of pressure coefficient on the propeller for $J = 0.7$.

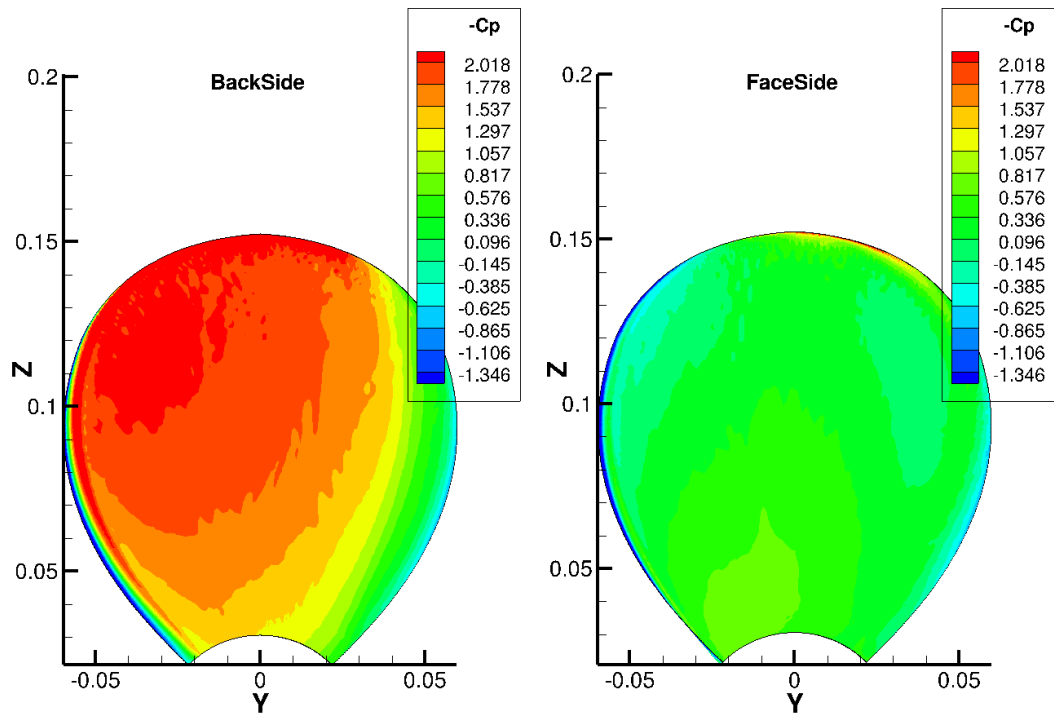


Figure 4.67 : Distribution of pressure coefficient on the propeller for $J = 0.8$.

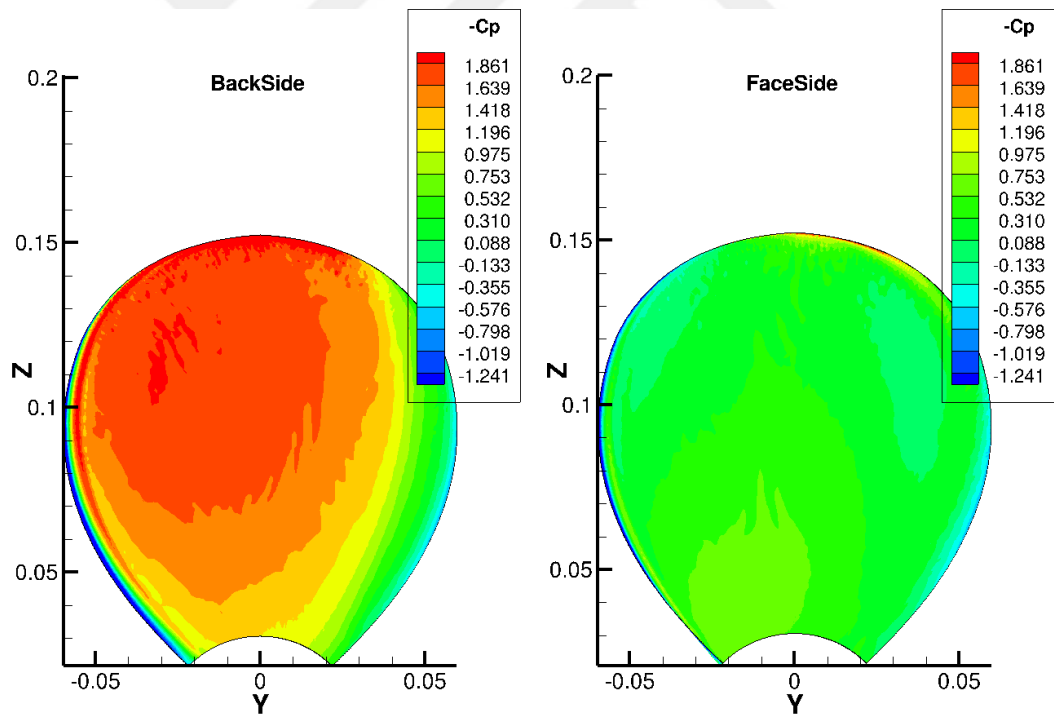


Figure 4.68 : Distribution of pressure coefficient on the propeller for $J = 0.833$.

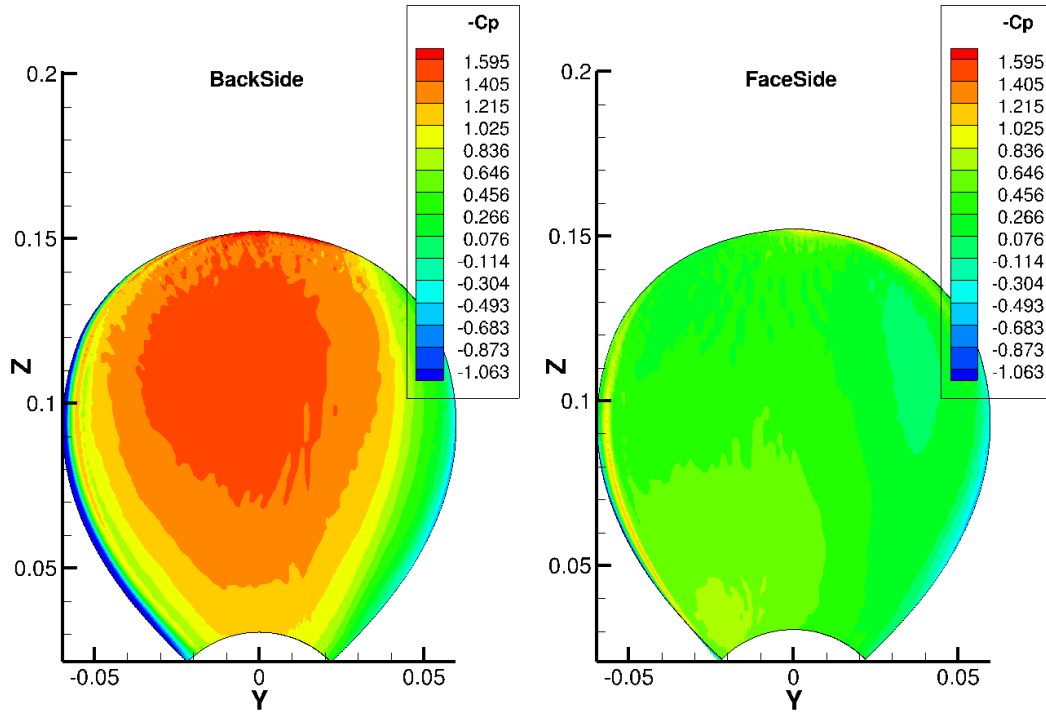


Figure 4.69 : Distribution of pressure coefficient on the propeller for $J = 0.9$.

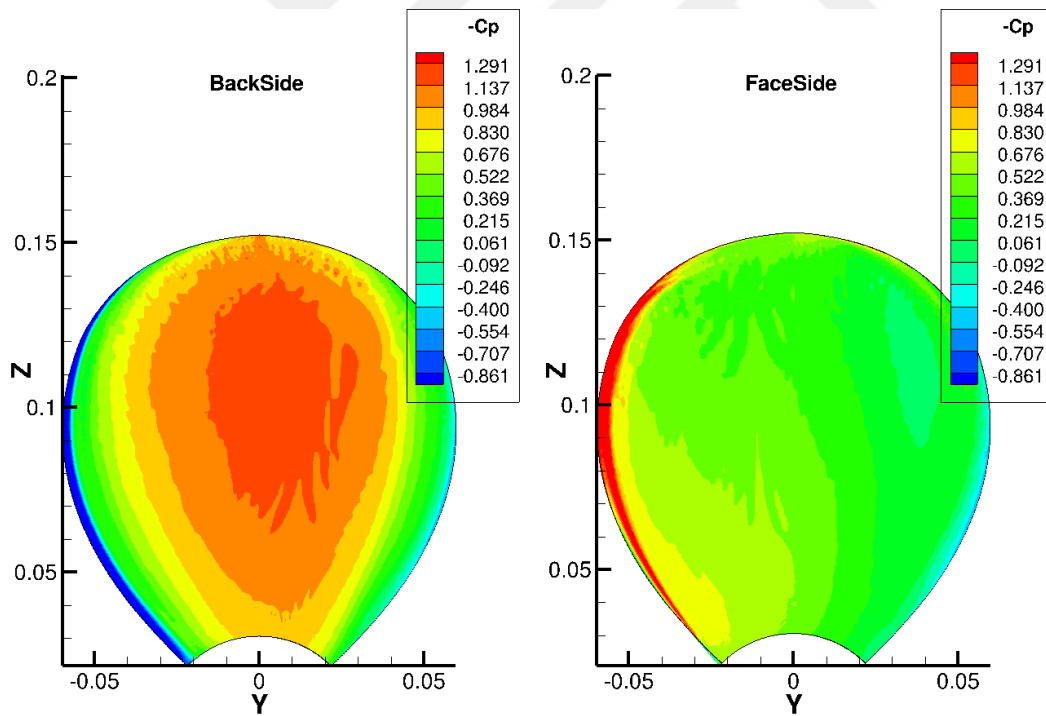


Figure 4.70 : Distribution of pressure coefficient on the propeller for $J = 1.0$.

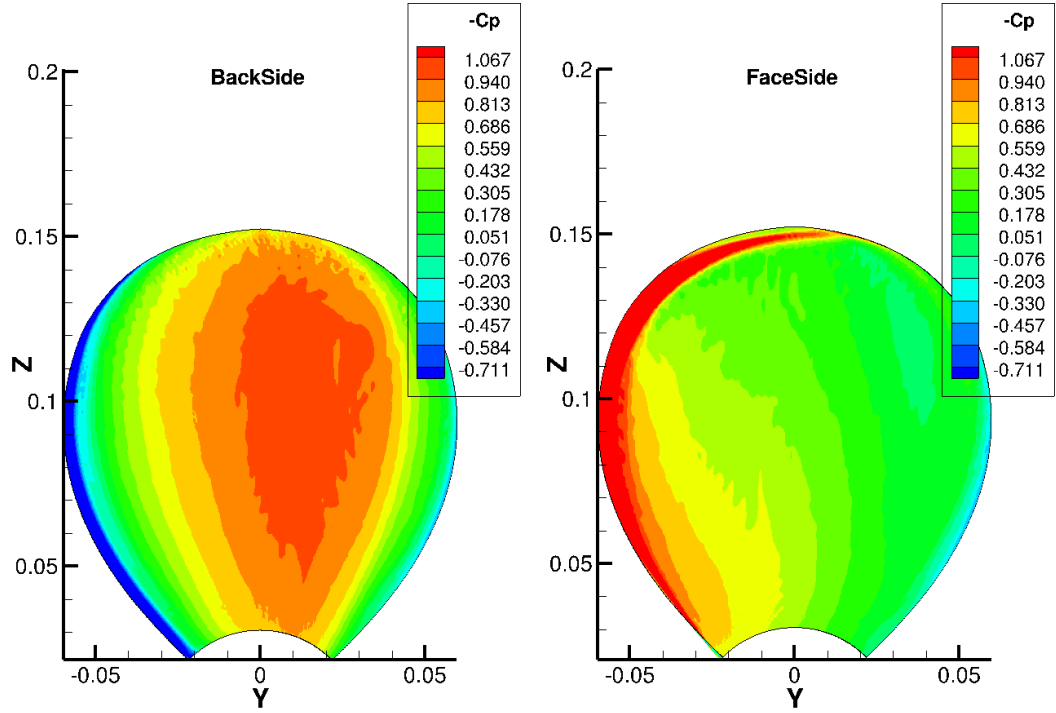


Figure 4.71 : Distribution of pressure coefficient on the propeller for $J = 1.1$.

Numerical computation results for different advance coefficients of DTMB 4119 propeller are compared with experimental results in Figure 4.72.

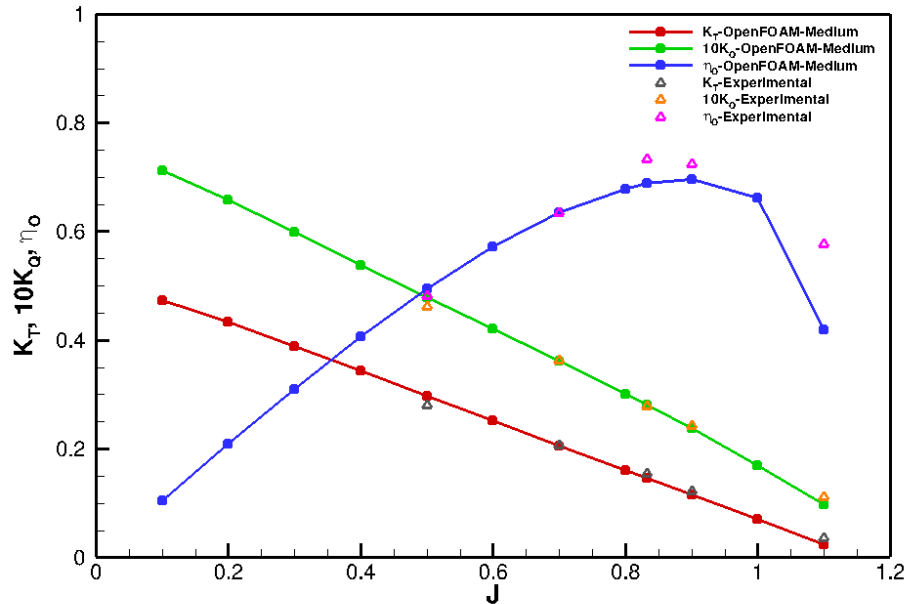


Figure 4.72 : Comparison of K_T , $10K_Q$ and η_O values using CFD solution for medium grid with experimental data.

Table 4.10 shows the comparison of numerical computation and experimental results.

Table 4.10 : The comparison of numerical and experimental results.

J	K_T (CFD)	K_T (Exp.)	$10K_Q$ (CFD)	$10K_Q$ (Exp.)	η_o (CFD)	η_o (Exp.)
0.100	0.472727	-	0.713008	-	0.105520	-
0.200	0.433176	-	0.658583	-	0.209365	-
0.300	0.388816	-	0.598661	-	0.310102	-
0.400	0.343507	-	0.538671	-	0.405968	-
0.500	0.297799	0.285	0.479311	0.477	0.494421	0.489
0.600	0.252385	-	0.421046	-	0.572406	-
0.700	0.206368	0.200	0.361920	0.360	0.635255	0.632
0.800	0.160499	-	0.301031	-	0.678847	-
0.833	0.145472	0.146	0.280245	0.280	0.688189	0.692
0.900	0.115573	0.120	0.237719	0.239	0.696396	0.725
1.000	0.070651	-	0.169745	-	0.662436	-
1.100	0.023204	0.034	0.096906	0.106	0.419197	0.575

Table 4.11 shows the comparison of numerical errors between computational fluid dynamics method and experimental results.

Table 4.11 : The comparison of relative numerical percentage errors between computational fluid dynamics method and experimental results.

J	% ΔK_T	% $\Delta 10K_Q$	% $\Delta \eta_o$
0.500	-4.516845	0.726398	-2.367070
0.700	-3.083293	1.032983	-2.006158
0.833	-0.176534	1.557150	-1.656625
0.900	3.310712	3.027857	1.090194
1.100	27.001208	14.053642	17.051943

As it can be seen from the above table, the maximum relative error propagation occurs at $J = 1.100$. This means, near wall treatment with extended wall functions is inadequate at high advance coefficients. In order to capture physical fluctuations accurately, wall functions could not be used at high advance ratios or mesh cell size must be reduced and refinement diffusion has to be increased.

4.2.3.3 Open water computations with fine grid generation

The convergence graphs for thrust and torque values are given below.

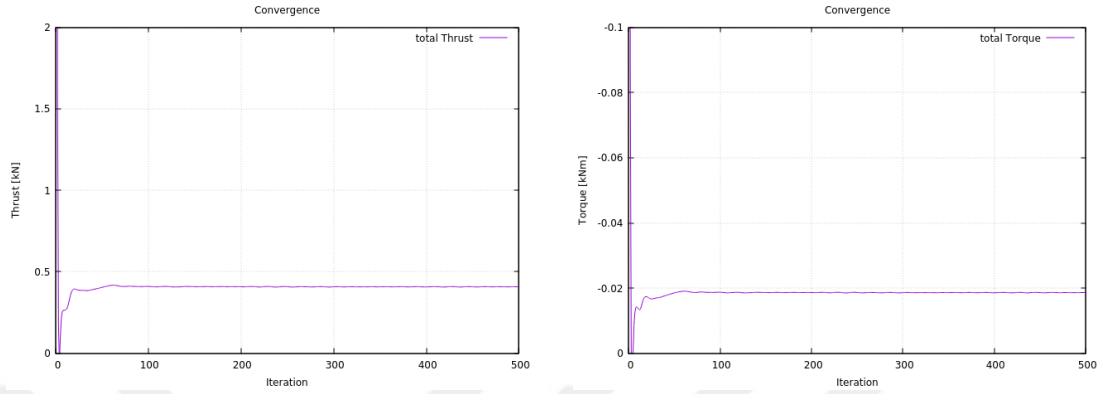


Figure 4.73 : Convergence graphs for $J = 0.1$.

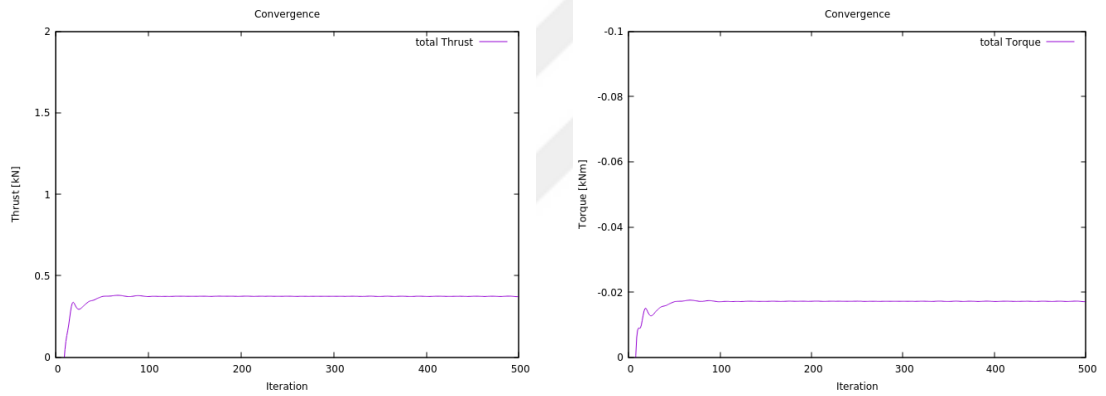


Figure 4.74 : Convergence graphs for $J = 0.2$.

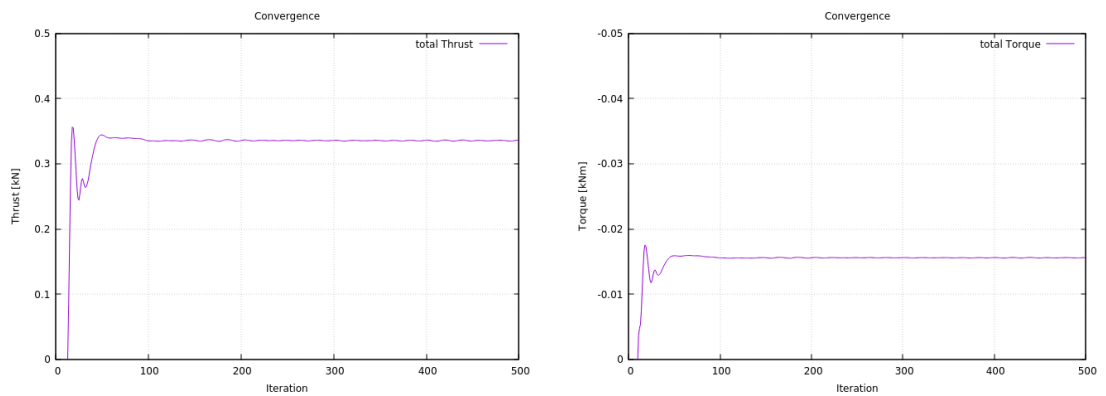


Figure 4.75 : Convergence graphs for $J = 0.3$.

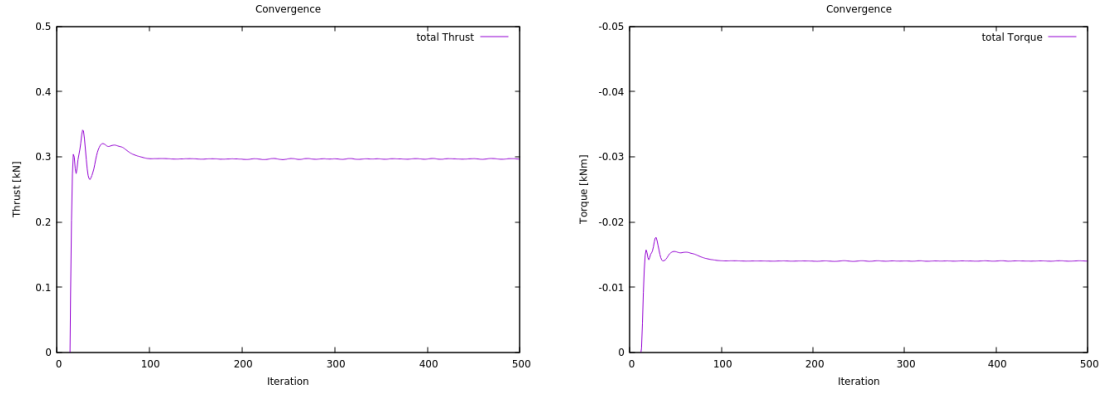


Figure 4.76 : Convergence graphs for $J = 0.4$.

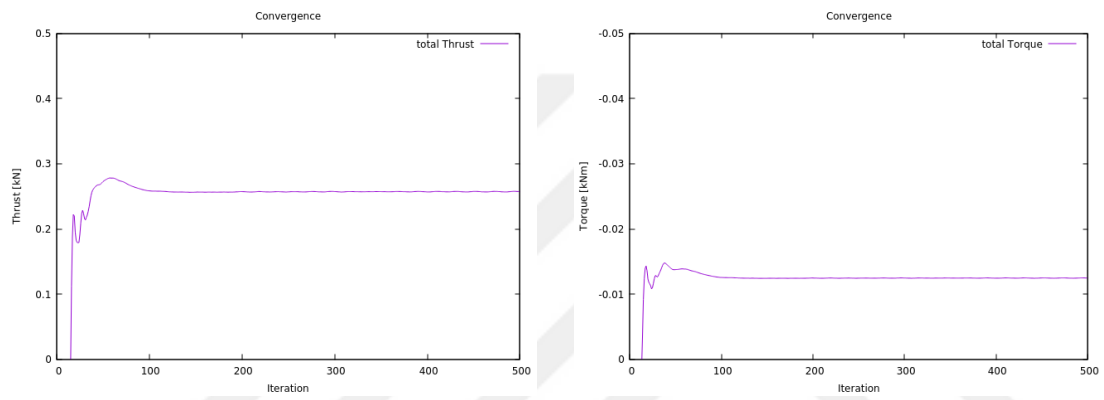


Figure 4.77 : Convergence graphs for $J = 0.5$.

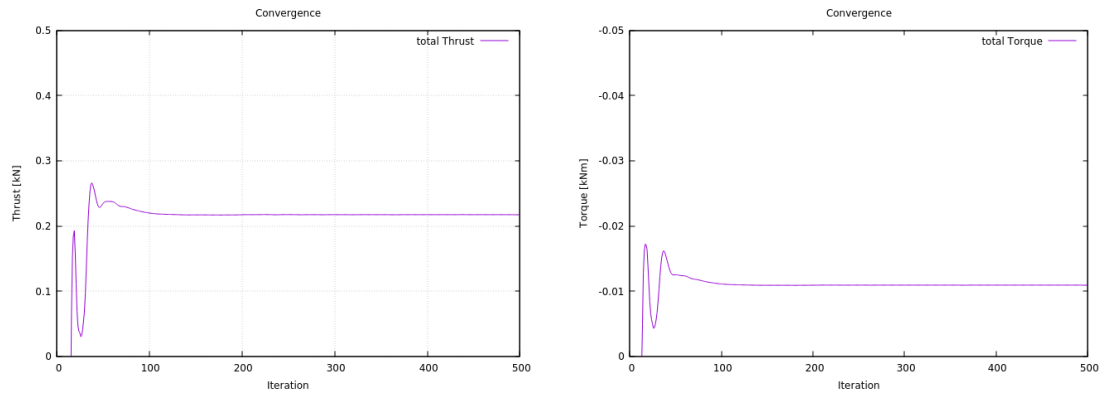


Figure 4.78 : Convergence graphs for $J = 0.6$.

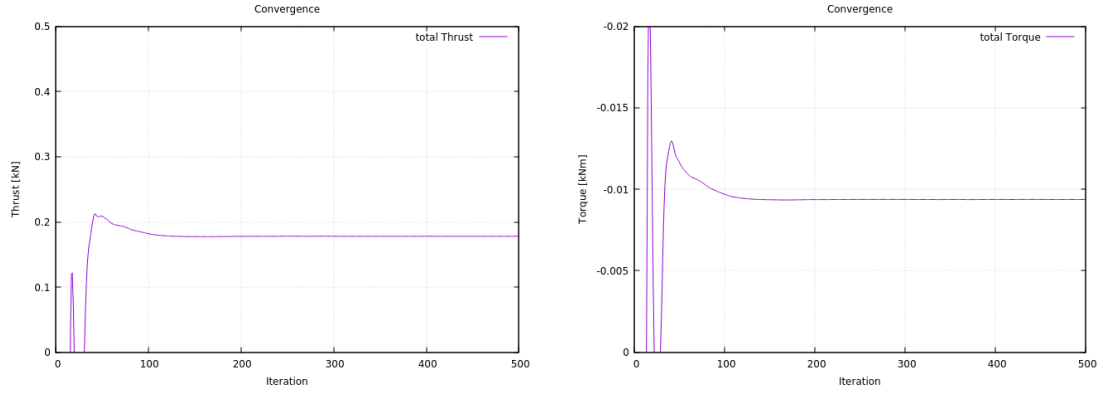


Figure 4.79 : Convergence graphs for $J = 0.7$.

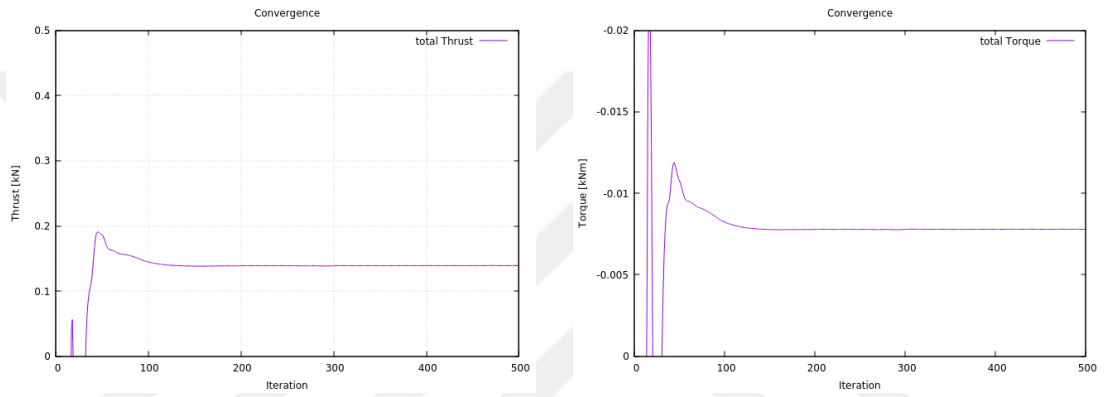


Figure 4.80 : Convergence graphs for $J = 0.8$.

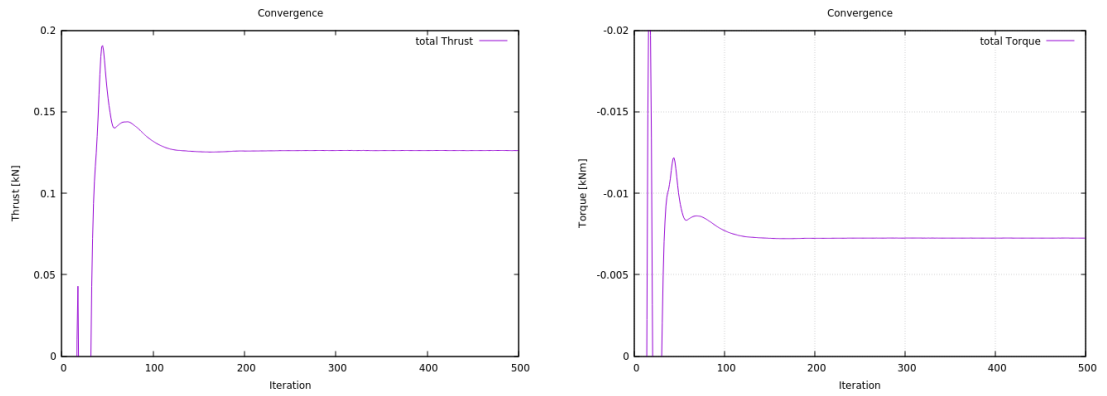


Figure 4.81 : Convergence graphs for $J = 0.833$.

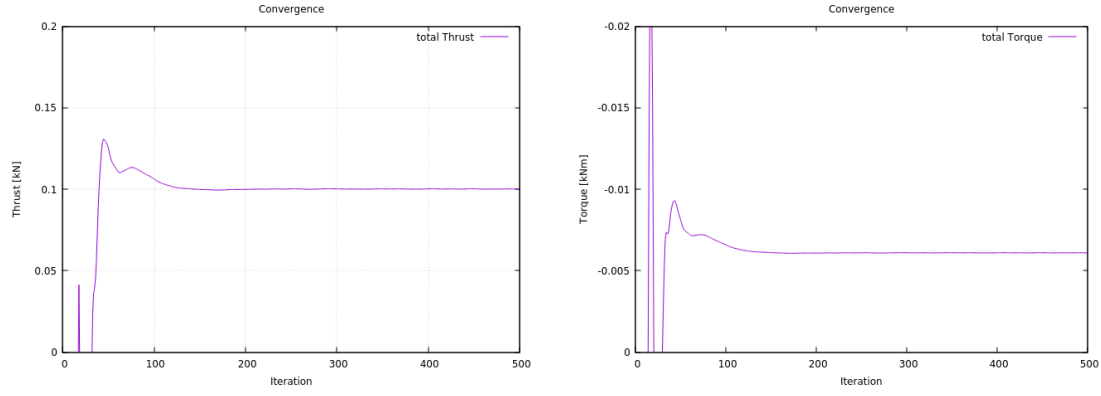


Figure 4.82 : Convergence graphs for $J = 0.9$.

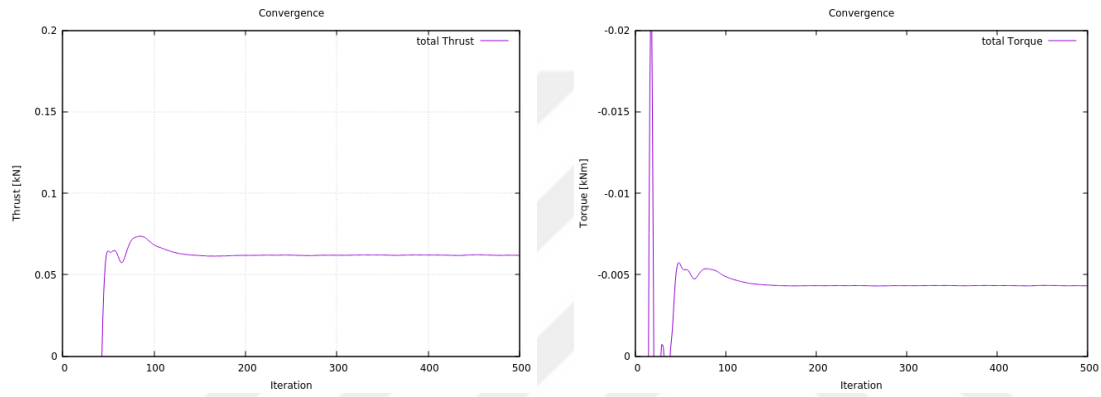


Figure 4.83 : Convergence graphs for $J = 1.0$.

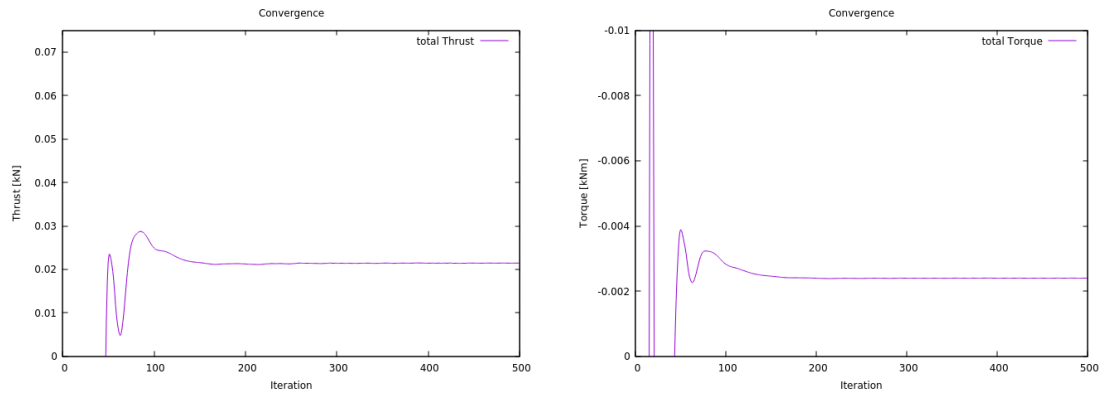


Figure 4.84 : Convergence graphs for $J = 1.1$.

For verification of the numerical computation, pressure coefficient contours on the propeller are given below.

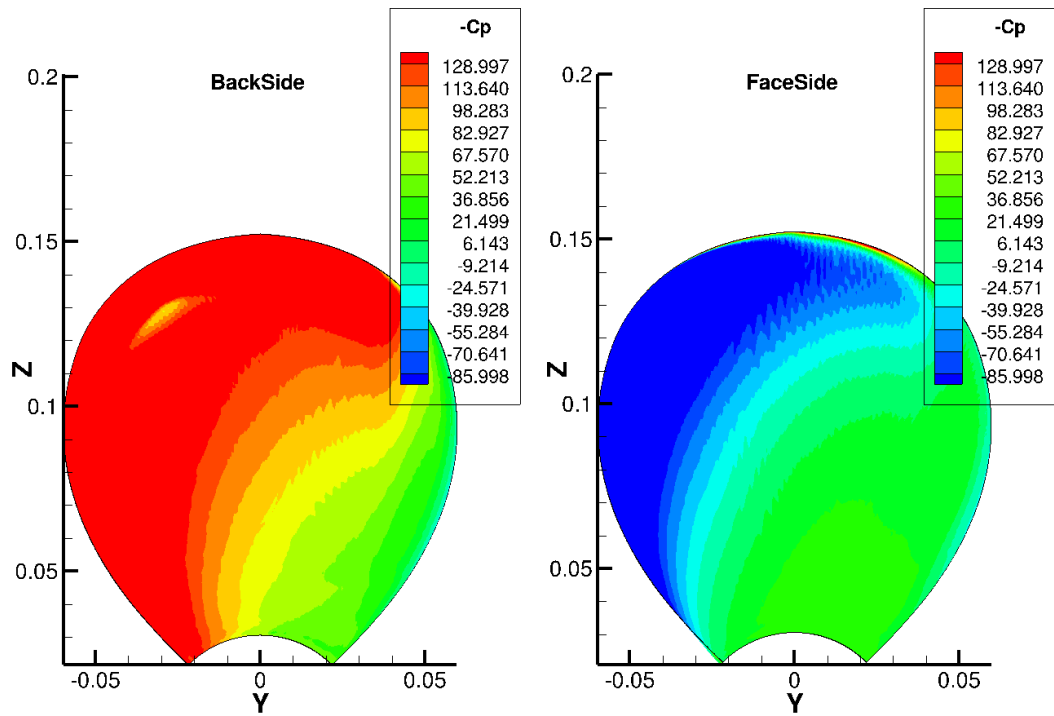


Figure 4.85 : Distribution of pressure coefficient on the propeller for $J = 0.1$.

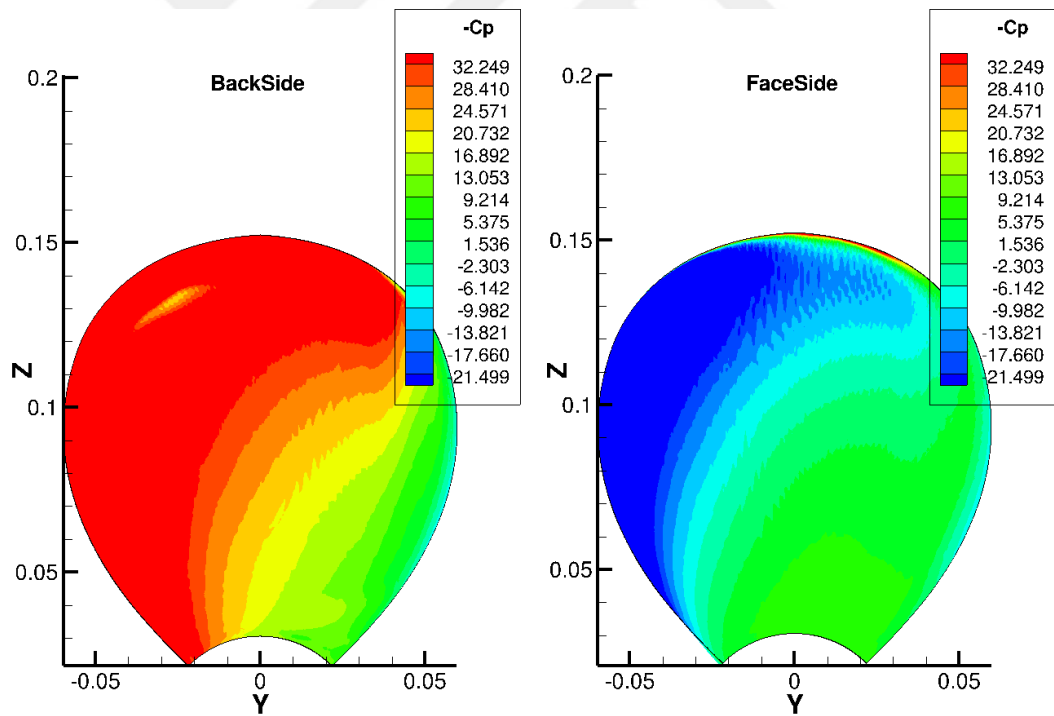


Figure 4.86 : Distribution of pressure coefficient on the propeller for $J = 0.2$.

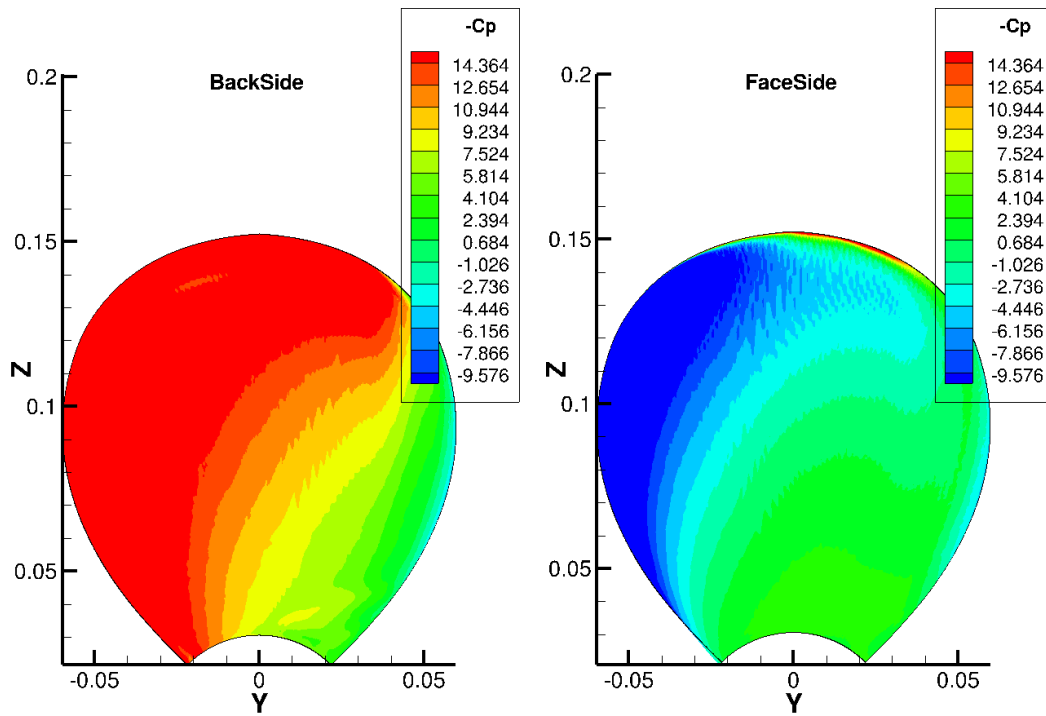


Figure 4.87 : Distribution of pressure coefficient on the propeller for $J = 0.3$.

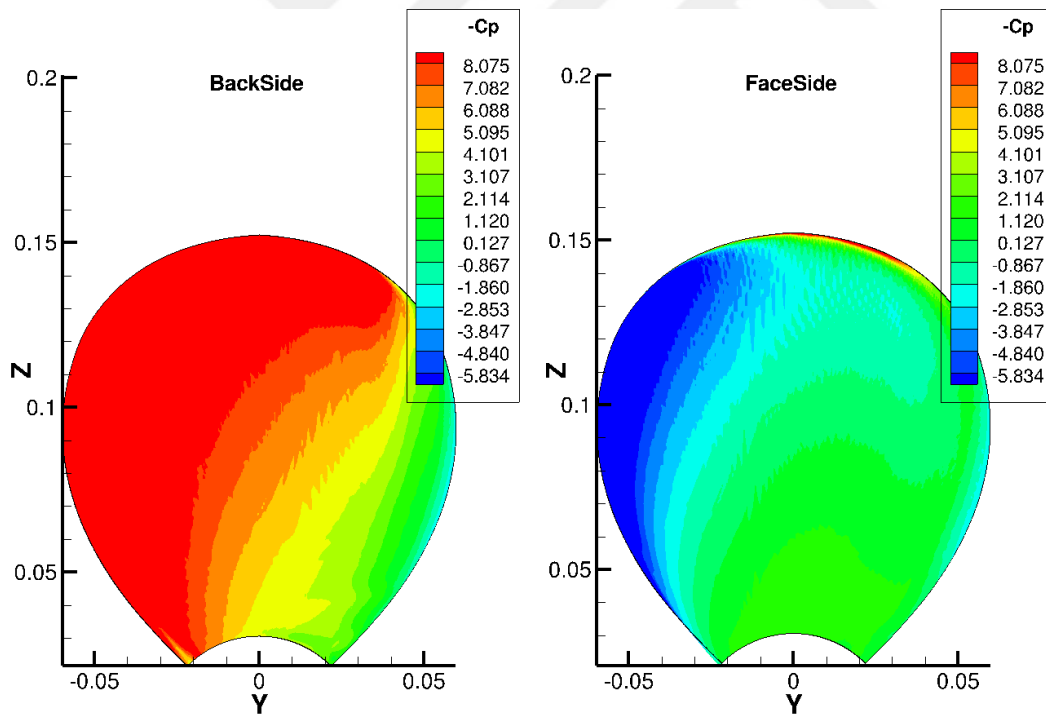


Figure 4.88 : Distribution of pressure coefficient on the propeller for $J = 0.4$.

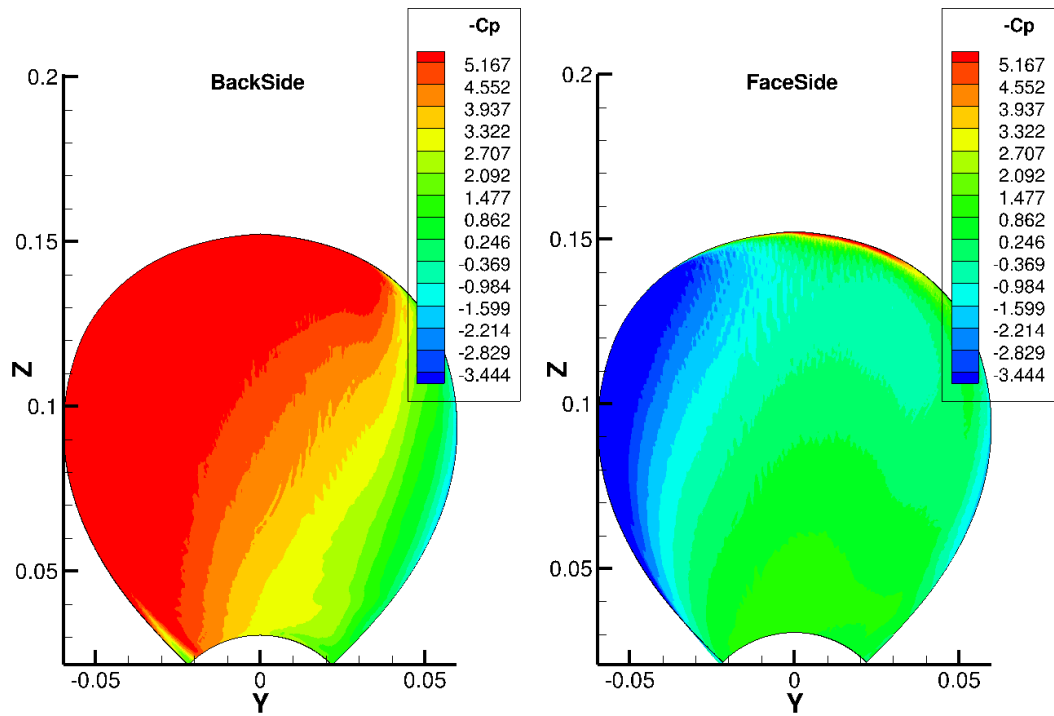


Figure 4.89 : Distribution of pressure coefficient on the propeller for $J = 0.5$.

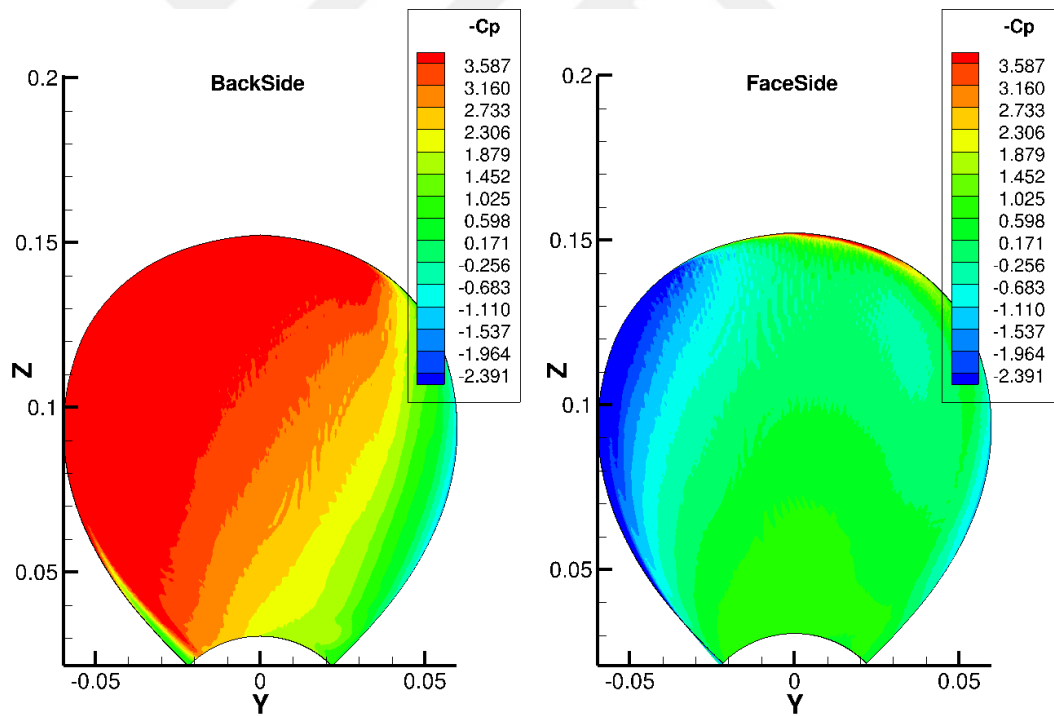


Figure 4.90 : Distribution of pressure coefficient on the propeller for $J = 0.6$.

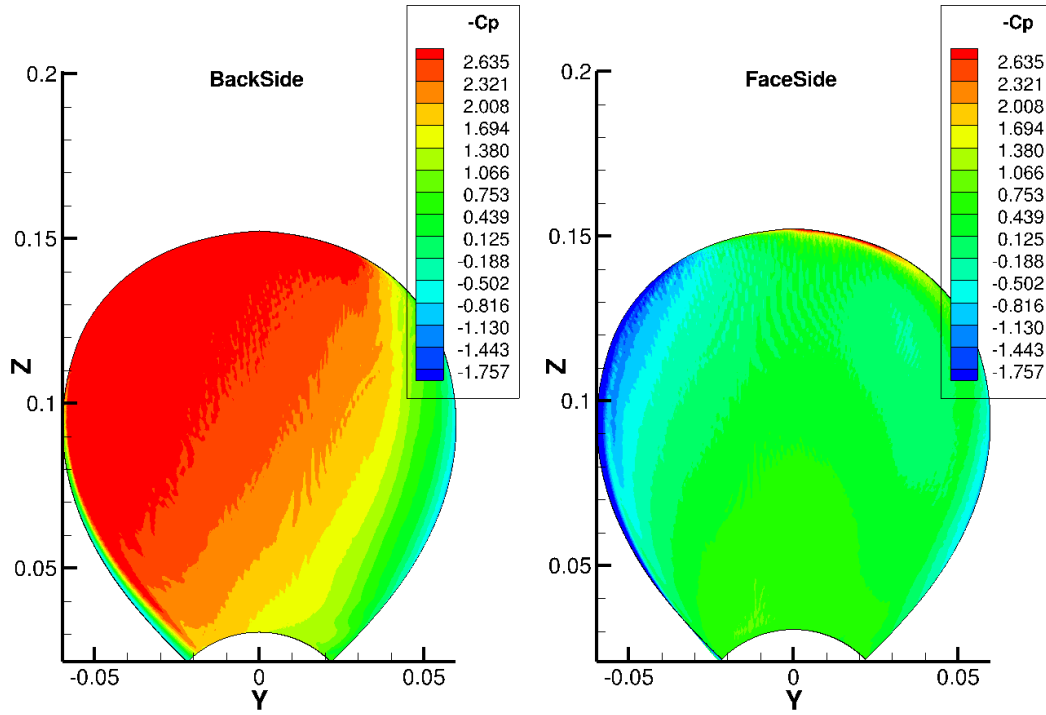


Figure 4.91 : Distribution of pressure coefficient on the propeller for $J = 0.7$.

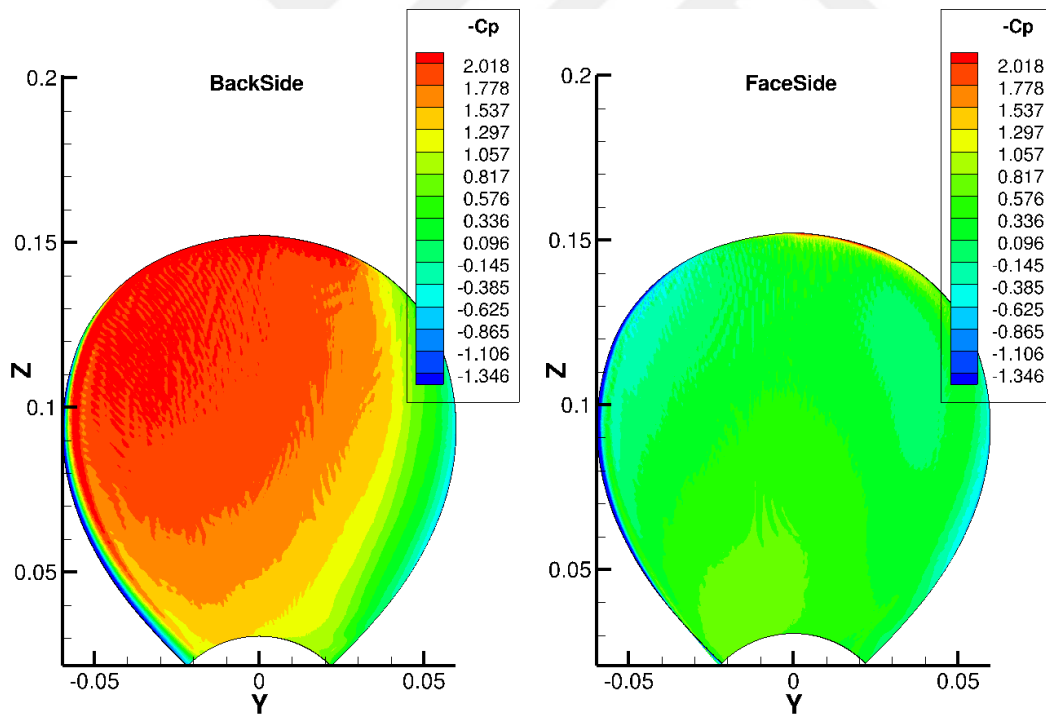


Figure 4.92 : Distribution of pressure coefficient on the propeller for $J = 0.8$.

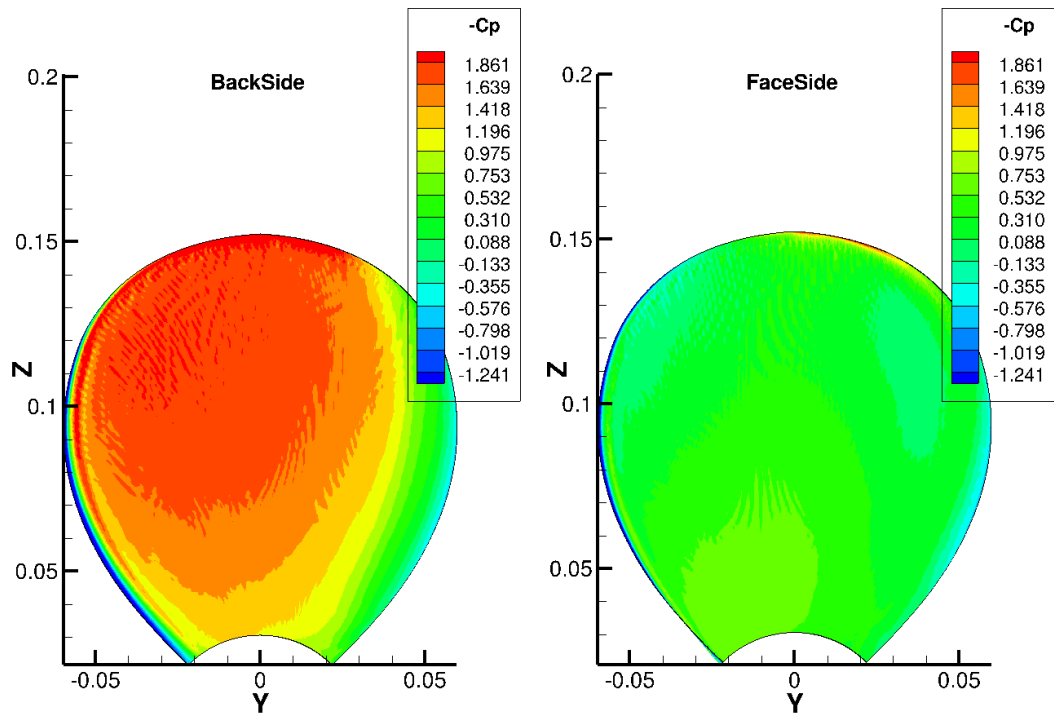


Figure 4.93 : Distribution of pressure coefficient on the propeller for $J = 0.833$.

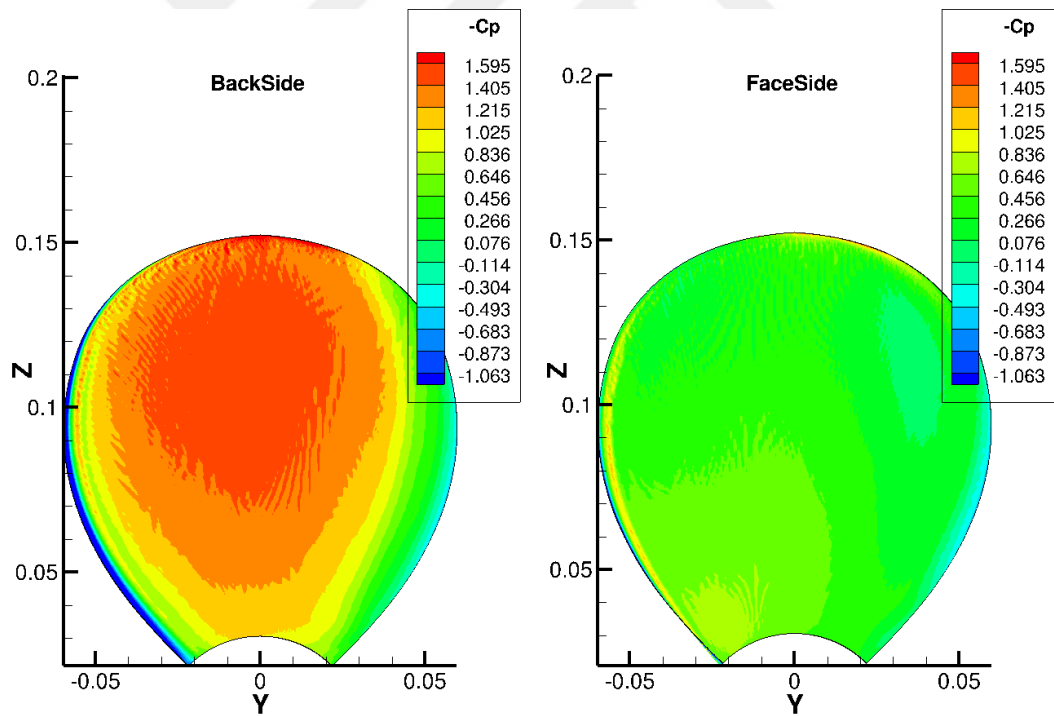


Figure 4.94 : Distribution of pressure coefficient on the propeller for $J = 0.9$.

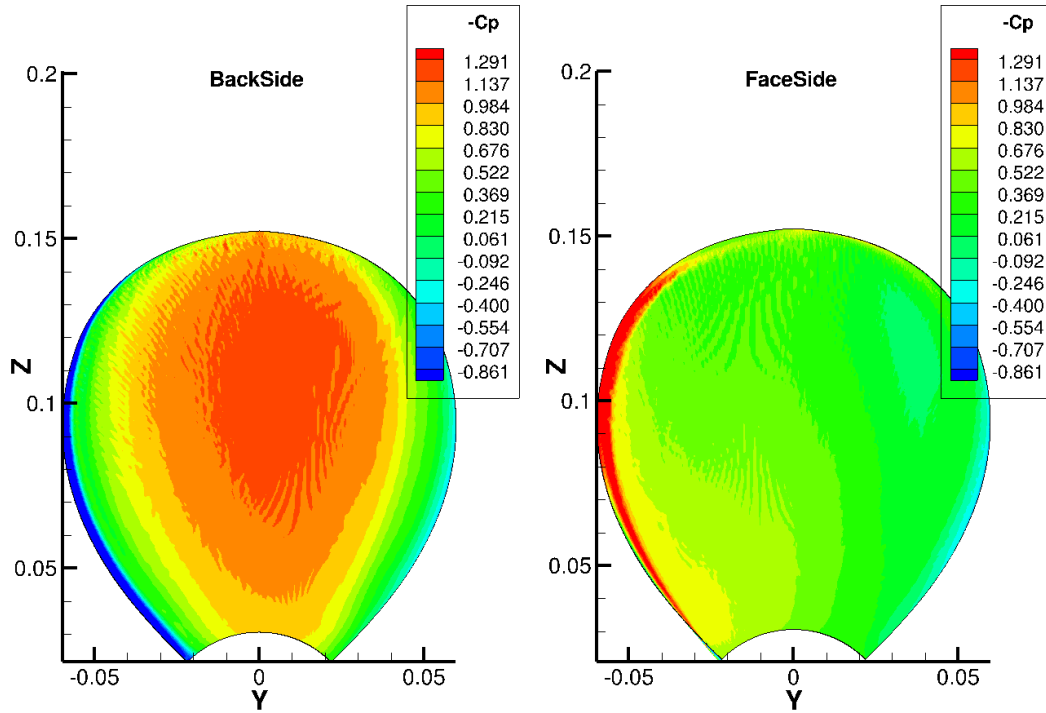


Figure 4.95 : Distribution of pressure coefficient on the propeller for $J = 1.0$.

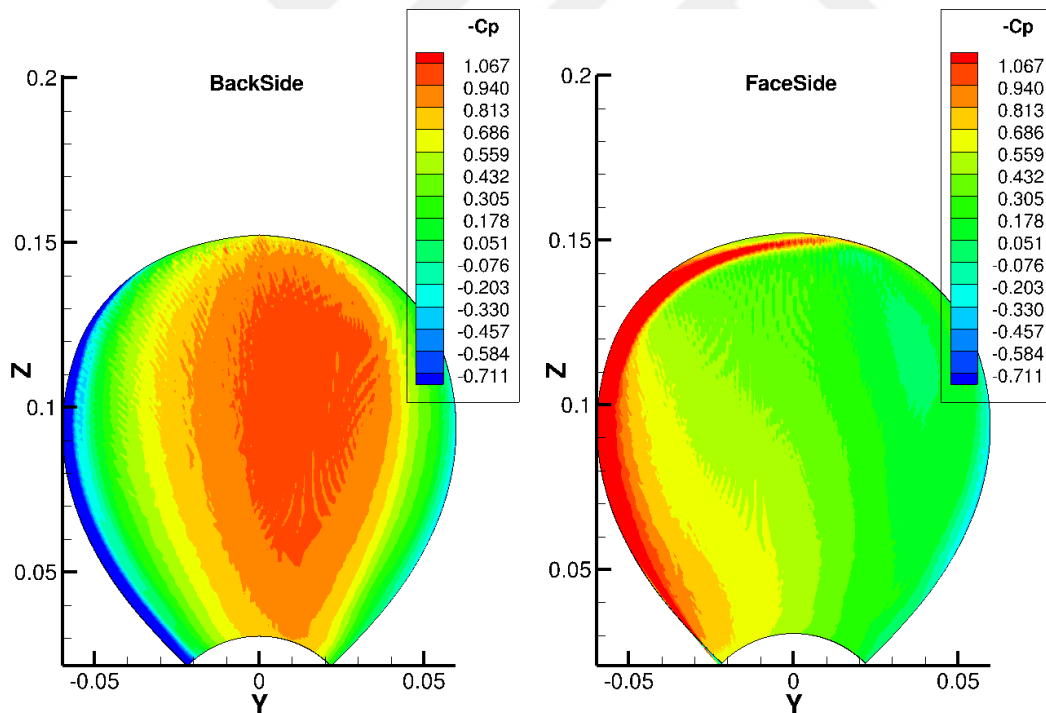


Figure 4.96 : Distribution of pressure coefficient on the propeller for $J = 1.1$.

Numerical computation results for different advance coefficients of DTMB 4119 propeller are compared with experimental results in Figure 4.97.

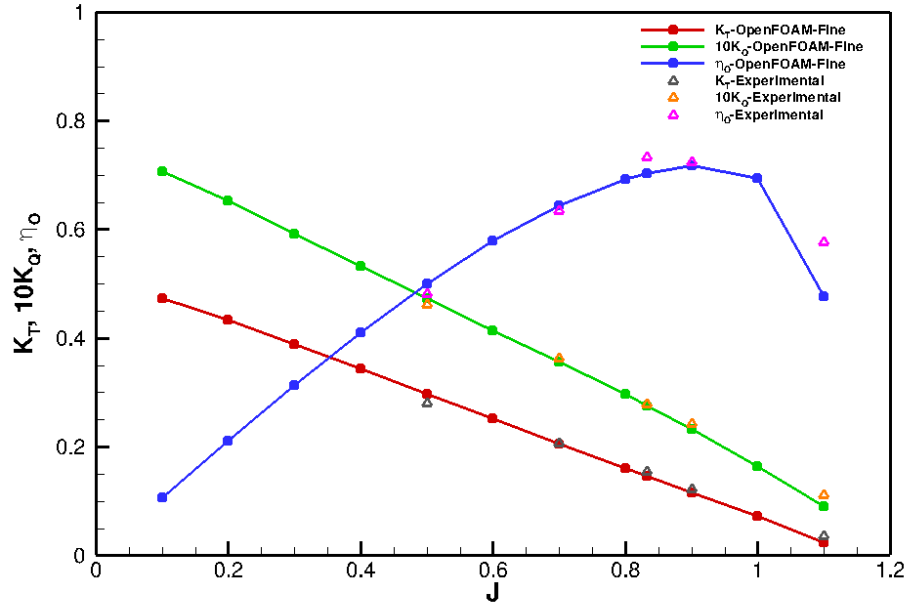


Figure 4.97 : Comparison of K_T , $10K_Q$ and η_O values using CFD solution for fine grid with experimental data.

Table 4.12 shows the comparison of numerical computation and experimental results. As it is seen below, the numerical predictions for K_T and $10K_Q$ values are quite acceptable, however the larger J gives the larger error, as expected. But it is seen that the absolute value of relative errors between coarse and medium mesh structures are decreased. However, it is computationally expensive. A hundred thousand cells per processor at maximum is taken into consideration. This approximation is quite enough to prevent from large standby time during information exchange between processors.

Table 4.12 : The comparison of numerical and experimental results.

J	K_T (CFD)	K_T (Exp.)	$10K_Q$ (CFD)	$10K_Q$ (Exp.)	η_O (CFD)	η_O (Exp.)
0.100	0.472714	-	0.707419	-	0.106351	-
0.200	0.433061	-	0.652361	-	0.211306	-
0.300	0.388792	-	0.591962	-	0.313592	-
0.400	0.343647	-	0.532135	-	0.411122	-
0.500	0.297873	0.285	0.473535	0.477	0.500575	0.489
0.600	0.251649	-	0.414724	-	0.579439	-
0.700	0.206167	0.200	0.356281	0.360	0.644679	0.632
0.800	0.161048	-	0.296286	-	0.692078	-
0.833	0.146258	0.146	0.275640	0.280	0.703464	0.692
0.900	0.116027	0.120	0.231763	0.239	0.717096	0.725
1.000	0.071895	-	0.164661	-	0.694908	-
1.100	0.024820	0.034	0.091103	0.106	0.476951	0.575

Table 4.13 shows the comparison of numerical errors between computational fluid dynamics method and experimental results.

Table 4.13 : The comparison of relative numerical percentage errors between computational fluid dynamics method and experimental results.

J	$\% \Delta K_T$	$\% \Delta 10K_Q$	$\% \Delta \eta_O$
0.500	-0.631	-1.530	3.680
0.700	0.650	-2.777	5.316
0.833	4.246	-3.464	7.529
0.900	8.000	-3.389	11.724
1.100	41.470	-2.641	42.260

OpenFOAM solutions are compared with PVL solutions which is given by Bal's studies. Pressure coefficient distribution on key blade for $J = 0.6$ and $J = 0.8$ open water test cases as seen below;

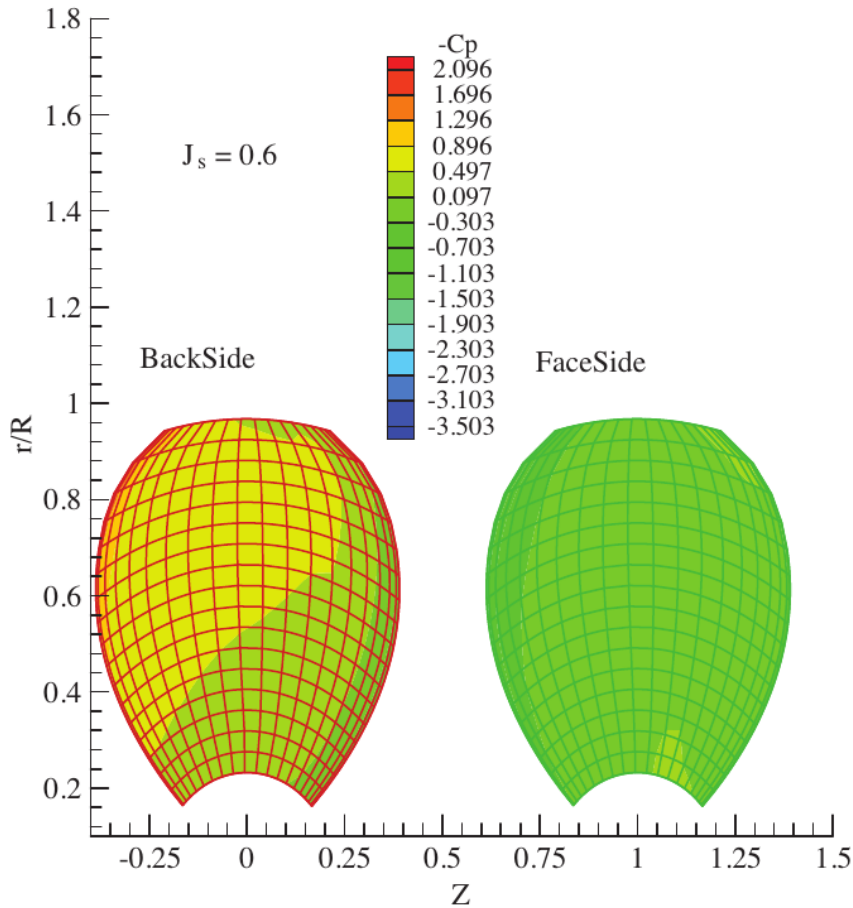


Figure 4.98 : Distribution of pressure coefficient on the propeller for $J = 0.6$ using PVL code (Bal, 2011a).

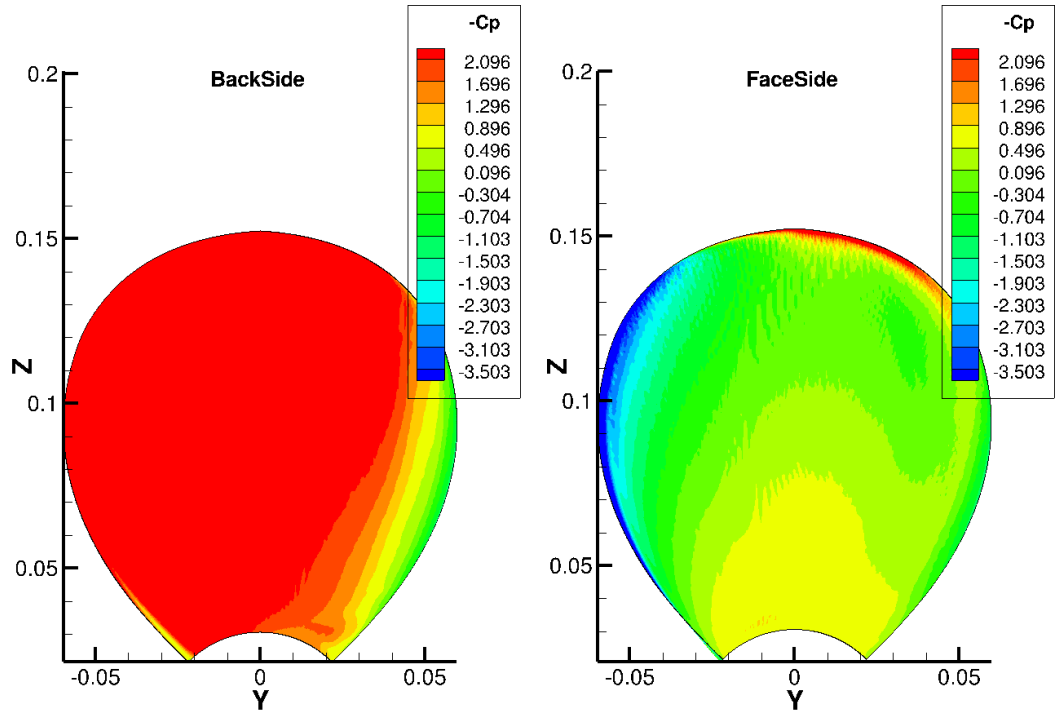


Figure 4.99 : Distribution of pressure coefficient on the propeller for $J = 0.6$ using OpenFOAM.

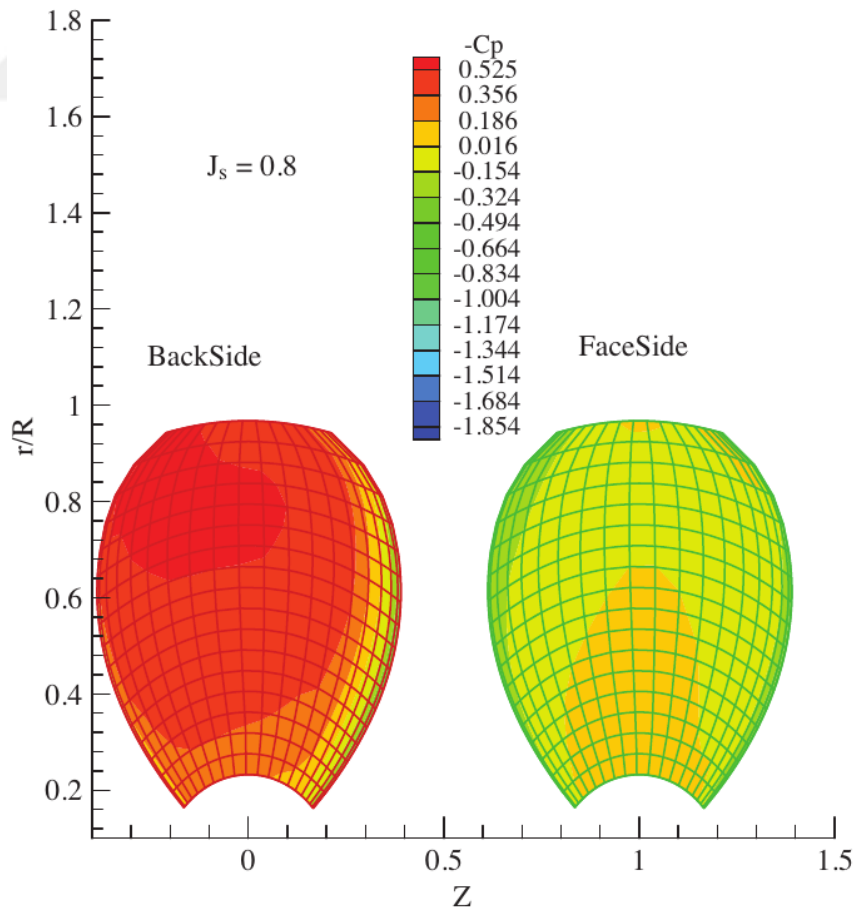


Figure 4.100 : Distribution of pressure coefficient on the propeller for $J = 0.8$ using PVL code (Bal, 2011b).

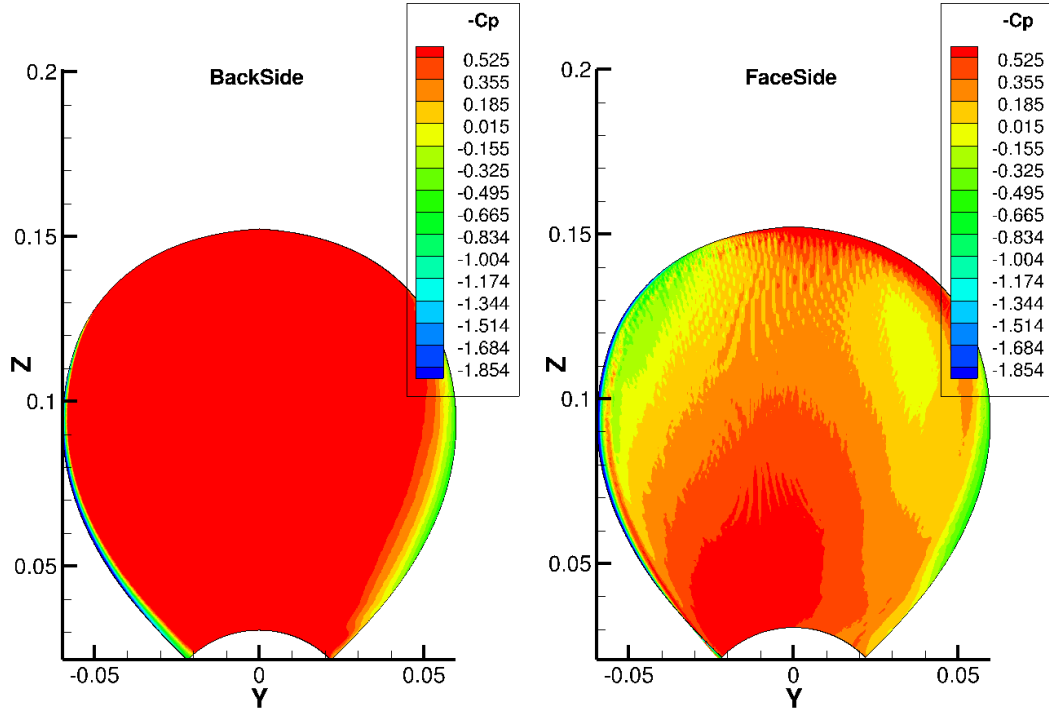


Figure 4.101 : Distribution of pressure coefficient on the propeller for $J = 0.8$ using OpenFOAM.

In consequence, blade element momentum theory solutions are compared with OpenFOAM, PVL and experimental results. Table 4.14 shows the comparison of experimental data between numerically computed K_T , $10K_Q$ and η_O values using BEMT, CFD and PVL computations.

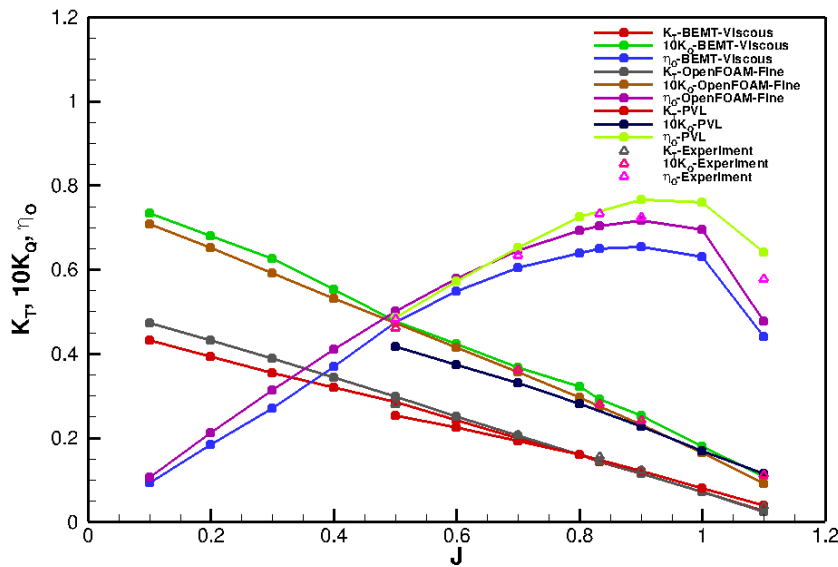


Figure 4.102 : Comparison of K_T , $10K_Q$ and η_O values using BEMT, CFD, PVL solutions with experimental data.

Table 4.14 : The comparison of numerical and experimental results.

J	BEMT			CFD			PVL			Exp.	
	K_T	$10K_Q$	η_O	K_T	$10K_Q$	η_O	K_T	$10K_Q$	η_O	K_T	$10K_Q$
0.100	0.432	0.733	0.094	0.472714	0.707419	0.106351	-	-	-	-	-
0.200	0.393	0.679	0.184	0.433061	0.652361	0.211306	-	-	-	-	-
0.300	0.355	0.626	0.270	0.388792	0.591962	0.313592	-	-	-	-	-
0.400	0.320	0.552	0.369	0.343647	0.532135	0.411122	-	-	-	-	-
0.500	0.286	0.478	0.476	0.297873	0.473535	0.500575	0.254	0.416	0.485	0.281	0.463
0.600	0.243	0.423	0.548	0.251649	0.414724	0.579439	0.224	0.374	0.572	-	-
0.700	0.199	0.368	0.604	0.206167	0.356281	0.644679	0.193	0.330	0.652	0.207	0.363
0.800	0.161	0.321	0.639	0.161048	0.296286	0.692078	0.160	0.281	0.725	-	-
0.833	0.143	0.292	0.650	0.146258	0.275640	0.703464	-	-	-	0.155	0.280
0.900	0.115	0.253	0.655	0.116027	0.231763	0.717096	0.122	0.228	0.767	0.123	0.243
1.000	0.071	0.180	0.630	0.071895	0.164661	0.694908	0.081	0.169	0.760	-	-
1.100	0.031	0.108	0.496	0.024820	0.091103	0.476951	0.039	0.116	0.642	0.037	0.112
											0.578



5. CONCLUSIONS AND RECOMMENDATIONS

In the present study, it is obtained that application of blade element momentum theory on marine propellers gives more accurate results than computational fluid dynamics method using OpenFOAM, although nonlinear effects are neglected. However, DTMB 4119 propeller has no rake and no skew. Thus, in this study rake and skew effects are neglected. In addition cavitation effects are neglected.

First, thrust and power coefficients are calculated using blade element momentum theory according to McCormick's small angle solution algorithm without hub effects. C_L and C_D coefficients are computed iteratively using XFOIL where viscous effects are included during the solution. Then, these computations are repeated using CFD method. An appropriate domain is prepared using Numeca Hexpress/Hybrid software and open water test cases are performed using OpenFOAM. Numerical results of blade element momentum theory and CFD method are compared with experimental results and also panel method solutions. Owing to the fact that, the relative error between numerical results and experimental data increases when advance coefficient grows up too. Pressure coefficient distribution on propeller blades for $J = 0.6$ and $J = 0.8$ are compared between results of CFD and panel methods. The comparison is quite satisfactory.

As a future work, it is planned to take into consideration the McCormick's nonlinear large angle solution, inclined shaft angle effects, wake adaption and cavitation for blade element momentum theory algorithm. Besides, so as to discretize flow fluctuations physically, sliding grid method must be executed for all open water cases.



REFERENCES

- [1] **Abbott, I.H., and von Doenhoff, A.E.** (1959). The Theory of Wing Sections, Dover Inc., New York, NY, USA.
- [2] **Bal, Ş.,** (2011a). A Method for Optimum Cavitating Ship Propellers, *Proceedings of the Institution of Mechanical Engineers, Part M, Journal of Engineering for the Maritime Environment*, 125(4), 375–386.
- [3] **Bal, Ş.,** (2011b). A Practical Technique for Improvement of Open Water Propeller Performance, *Turkish Journal of Engineering and Environmental Sciences*, 10(3906).
- [4] **Bal, Ş.,** (2013). Boundary Element Methods, *Lecture Notes*.
- [5] **Bal, Ş.,** (2014). Hydrodynamics of Ship Propellers, *Lecture Notes*.
- [6] **Belhenniche, S.E., Aounallah, M., Imine, O. and Çelik, F.,** (2012). Application of CFD Analysis of Steady and Unsteady Turbulent Flow Past a Marine Propeller, *7th International Symposium on Turbulence, Heat and Mass Transfer*, Italy.
- [7] **Benini, E.,** (2003). Significance of Blade Element Theory in Performance Prediction of Marine Propellers, *Ocean Engineering*, 31(2004), 957–974.
- [8] **Breslin, J. P. and Andersen, P.,** (1994). Hydrodynamics of Ship Propellers, Cambridge University Press, New York, USA
- [9] **Brizzolara, S., Villa, D. and Gaggero, S.,** (2008). A Systematic Comparison Between RANS and Panel Methods for Propeller Analysis, *Proceedings of 8th International Conference on Hydrodynamics*, Nantes, France.
- [10] **Carlton J. S.,** (2007). Marine Propellers and Propulsion, Butterworth-Heinemann, Burlington, USA, 2. edition.
- [11] **Cebeci, T. and Bradshaw, P.,** (1977). Momentum Transfer in Boundary Layers, McGraw-Hill Book Comp., USA.
- [12] **Colley, E.,** (2012). Analysis of Flow Around a Ship Propeller Using OpenFOAM , (*Master's thesis*), Retrieved from <http://cmst.curtin.edu.au>.
- [13] **Çelik, F.,** (2011). Ship Resistance and Propulsion, *Lecture Notes*.
- [14] **Çelik, F., Arıkan, Y. and Bal, Ş.,** (2014). Numerical Simulation of The Flow Around Two-Dimensional Partially Cavitating Hydrofoils, *Journal of Marine Science and Application*, 13(3), 245–254.

- [15] **Davidson, L.**, (2017). Fluid Mechanics, Turbulent Flow and Turbulence Modeling, *Lecture Notes*.
- [16] **Drela, M.**, (1989). Integral Boundary Layer Formulation for Blunt Trailing Edges, *AIAA Journal*, 89(2200).
- [17] **Drela, M.**, (1989). XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils, *Conference on Low Reynolds Number Airfoil Aerodynamics*, University of Notre Dame.
- [18] **Drela, M. and Giles, M. B.**, (1987). Viscous-Inviscid Analysis of Transonic and Low Reynolds Number Airfoils, *AIAA Journal*, 25(10), 1347–1355
- [19] **Durand, W.F.**, (1943). Aerodynamic Theory: A General Review of Progress, Wiley, Pasadena, 4. edition.
- [20] **Ekinci, S., Çelik, F. and Güner, M.**, (2006). Numerical Analysis of Marine Propellers for Steady and Unsteady Cases, *An International Conference on Small Craft Related Sciences and Engineering*, Turkey.
- [21] **Fergizer, J. H., and Peric, M.**, (1997). Computational Methods for Fluid Dynamics, Springer-Verlag, Berlin, 2. edition.
- [22] **Goldstein, S.**, (1929). On the Vortex Theory of Screw Propellers, *Proceeding of the Royal Society of London*, The Royal Society, 123, 440–465
- [23] **Hirsch, C.**, (2007). Numerical Computation of Internal and External flows, volume1, Butterworth-Heinemann, Burlington, USA, 2. edition.
- [24] **Hirsch, C.**, (2007). Numerical Computation of Internal and External flows, volume2, Butterworth-Heinemann, Burlington, USA, 2. edition.
- [25] **Issa, R.I.**, (1985). Solution of the Implicitly Discretized Fluid Flow Equations by Operator-Splitting, *Journal of Computational Physics*.
- [26] **Jasak, H.**, (1996). Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows, (*Ph.D. thesis*), Retrieved from <http://powerlab.fsb.hr/ped/kturbo/OpenFOAM/docs>
- [27] **Juretic, F.**, (2004). Error Analysis in Finite Volume CFD, (*Ph.D. thesis*), Retrieved from <http://powerlab.fsb.hr/ped/kturbo/OpenFOAM/docs>
- [28] **Kerwin, J. E., and Jacques, B. H.**, (2010). Principles of Naval Architecture Series: *Propulsion*, SNAME, USA.
- [29] **Kundu P., Cohen I. and Dowling D.** (2015). Fluid Mechanics, Academic Press, USA, 6. edition.
- [30] **Lamb, H.**, (1963). Hydrodynamics, Cambridge University Press, New York, NY, USA.
- [31] **LeVeque, R. J.**, (2002). Finite Volume Methods for Hyperbolic Problems, *Cambridge Texts in Applied Mathematics*, Cambridge University Press, New York, NY, USA, 1. edition.

- [32] **McCormick, B. W.**, (1967). Aerodynamics of V/STOL Flight, Academic Press, Orlando, FL, 79–82
- [33] **McDonough, J. M.**, (2007a). Introductory Lectures on Turbulence: *Physics, Mathematics and Modeling, Lecture Notes*.
- [34] **McDonough, J. M.**, (2007b). Lectures in Computational Fluid Dynamics of Incompressible Flow: *Mathematics, Algorithms and Implementations, Lecture Notes*.
- [35] **Mehdipor, R.**, (2013). Simulating Propeller and Propeller-Hull Interaction in OpenFOAM, (*Master's thesis*), Retrieved from <http://www.diva-portal.se/smash/get/diva2:784016/FULLTEXT01.pdf>
- [36] **Menter, F. R.** (1994). Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications, *AIAA Journal*, 32(8), 1598–1605
- [37] **Sahin, M.**, (2014). Computational Fluid Dynamics, *Lecture Notes*.
- [38] **Sahin, M.**, (2014). Advanced Computational Fluid Dynamics, *Lecture Notes*.
- [39] **Milne-Thompson, L. M.**, (1955). Theoretical Hydrodynamics, McMillan Co, London, 3. edition.
- [40] **Molland A. F. and Turnock S. R.**, (2007). Marine Rudders and Control Surfaces, Butterworth-Heinemann, Burlington, USA, 1. edition.
- [41] **Molland A.F., Turnock S. R. and Hudson D. A.** (2011). Ship Resistance and Propulsion, Cambridge University Press, New York, USA, 1. edition.
- [42] **Newman, J.N.** ,(1977). Marine Hydrodynamics, MIT Press, USA.
- [43] **Prakash, C. and Patankar, S. V.**, (1985). A Control Volume-Based Finite-Element Method for Solving the Navier-Stokes Equations Using Equal-Order Velocity-Pressure Interpolation, *Numerical Heat Transfer*, 8.
- [44] **Pletcher R. H., Tannehill J. C. and Anderson D. A.**, (2013). Computational Fluid Mechanics and Heat Transfer, Taylor & Francis, Boca Raton, 3. edition.
- [45] **Pope, S. P.**, (2009). Turbulent Flows, Cambridge University Press, New York, USA , 2. edition.
- [46] **Rankine, W. J.**, (1865). On The Mechanical Principles of The Action of Propellers, *Trans RINA* , 6.
- [47] **Turunen, T., Siikonen, T., Lundberg, J. and Bensow, R.**, (2014). Open-Water Computations of A Marine Propeller Using OpenFOAM, *6th European Conference on Computational Fluid Dynamics*, Barcelona, Spain, 1123–1134.
- [48] **Turunen, T.**, (2014). Analysis of Multi-Propeller Marine Applications by Means of Computational Fluid Dynamics, *Master's thesis*, Retrieved from <https://aaltodoc.aalto.fi/handle/123456789/14078>

- [49] **Tibery, C. L. and Wrench, J. W. Jr.**, (1964). Tables of Goldstein Factor, *Applied Mathematics Lab. Report, 1534*, Washington D.C.
- [50] **Versteeg, H. K. and Malalasekera, W.**, (2007). An Introduction to Computational Fluid Dynamics : The Finite Volume Method, Harlow: Pearson/Prentice Hall, 2. edition.
- [51] **Whitmore, S.A. and Merrill, R. S.**, (2012). Nonlinear Large Angle Solutions of the Blade Element Momentum Theory Propeller Equations, *Journal of Aircraft*, 49(4).
- [52] **Wilcox, C. D.**, (1998). Turbulence Modeling for CFD, DCW Industries, 2. edition.
- [53] **Url-1**, <http://www.openfoam.com>, accessed at: 09.06.2016.
- [54] **Url-2**, <http://www.openfoam.org>, accessed at: 11.09.2016.
- [55] **Url-3**, https://openfoamwiki.net/index.php/OpenFOAM_guide/The_SIMPLE_algorithm_in_OpenFOAM, accessed at: 10.03.2016.
- [56] **Url-4**, http://openfoamwiki.net/index.php/OpenFOAM_guide/The_PISO_algorithm_in_OpenFOAM, accessed at: 10.03.2016.

APPENDICES

APPENDIX A.1 : MATLAB Files

APPENDIX B.1 : OpenFOAM boundary File

APPENDIX B.2 : OpenFOAM controlDict File

APPENDIX B.3 : OpenFOAM fvSolution File

APPENDIX B.4 : OpenFOAM fvSchemes File

APPENDIX B.5 : OpenFOAM forces File

APPENDIX B.6 : OpenFOAM MRFProperties File

APPENDIX B.7 : OpenFOAM transportProperties File

APPENDIX B.8 : OpenFOAM turbulenceProperties File



APPENDIX A.1

```
%
%-----%
% THIS PROGRAM CALCULATES PERFORMANCE OF A MARINE PROPELLER USING
% BLADE ELEMENT MOMENTUM THEORY
% Written By : Kayhan LGEN
% Date : 30 March 2016
%-----%

clear, close, clc,
%-----%
%-----Propeller Geometry Inputs-----%
Z = 3; % number of blades
D = 0.3048; % propeller diameter [m]
Dhub = 0.2 * D; % hub diameter [m]
rho = 1026.021; % density
nu = 1.189e-06; % kinematic viscosity
Meanline = 'NACA a=0.8'; % Meanline type
Thickness = 'NACA66 (DTRC modified)'; % Thickness form

XR = [0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 0.9500 1.0000];
R = (D/2) .* XR;
XCoD = [0.3200 0.3625 0.4048 0.4392 0.4610 0.4622 0.4347 0.3613 0.2775 0.0020];
c0 = XCoD .* D;
XCD = [0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080];
XVA = [1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000];
XVT = [0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000];
f0oc0 = [0.01429 0.02318 0.02303 0.02182 0.02072 0.02003 0.01967 0.01817 0.01631 0.01175];
f0 = f0oc0 .* c0;
XPoD = [1.105 1.102 1.098 1.093 1.088 1.084 1.081 1.079 1.077 1.075];
P = XPoD .* D;
t0oc0 = [0.2055 0.1553 0.1180 0.09016 0.06960 0.05418 0.04206 0.03321 0.03228 0.03160];
t0 = t0oc0 .* c0;
skew0 = zeros(size(XR)); % skew
rake0 = zeros(size(XR)); % rake
Xt0oD = t0oc0 .* XCoD; % thickness / diameter

Mp = 20; % number of radial 2D cross-sections
Np = 50; % number of points in each 2D section

XRi = linspace(min(XR),max(XR),Mp+1);
Ri = pchip(XR,R,XRi);
XCoDi = pchip(XR,XCoD,XRi);
c0i = pchip(XR,c0,XRi);
XCDi = pchip(XR,XCD,XRi);
XVAi = pchip(XR,XVA,XRi);
XVTi = pchip(XR,XVT,XRi);
f0oc0i = pchip(XR,f0oc0,XRi);
f0i = pchip(XR,f0,XRi);
XPoDi = pchip(XR,XPoD,XRi);
Pi = pchip(XR,P,XRi);
t0oc0i = pchip(XR,t0oc0,XRi);
t0i = pchip(XR,t0,XRi);
skew0i = pchip(XR,skew0,XRi);
rake0i = pchip(XR,rake0,XRi);
Xt0oDi = pchip(XR,Xt0oD,XRi);

%-----%
%-----Propeller Performance Inputs-----%
VA = 2.54; % axial inflow velocity
Omega = 10; % revolutions per second [rps]
J = VA/(Omega*D); % advance coefficient

%-----%
%-----Analyze-----%
[x2D,y2D] = BladeSection2D(Meanline,Thickness,Mp,Np,XRi,c0i,f0oc0i,t0i);

Thrust = 0.0;
Torque = 0.0;
Eta = 0.0;

%-----%
for j = 1:size(XRi,2);
    a = 0.1;
    aPrime = max(roots([(Omega*Ri(j)).^2 -(Omega*Ri(j)).^2 -(XVAi(j)*VA).^2*a*(1+a)]));
    Beta = atand(J/(pi()*Ri(j)));
    Betai = atand(((1+a)/(1-aPrime))*tand(Beta));
    Phi = atand(XPoDi(j)./(pi()*Ri(j)));
    finished = 0;
    sum=1;
    while (finished==0),
        V1 = XVAi(j)*VA*(1+a);
        V2 = Omega*Ri(j)*(1-aPrime);
        W = sqrt(V1.^2+V2.^2);
        Alpha = Phi-Betai;
        ReL = (XVAi(j)*VA.*2*Ri(j))/nu;
        airfoilCoordinate = [x2D(j,:); y2D(j,:)];
        fileID = fopen('foilGeom.dat','w');
        fprintf(fileID,'%12.6f %12.6f\r\n',airfoilCoordinate);
        fclose(fileID);
        [pol, foil] = xfoil('foilGeom.dat',1,100000,0.0,'oper iter 50');
        Theta = atand(2*tand(Beta)-tand(Betai));
        DtDr=0.5*rho.*W(j).*W(j).*2.0.*c0i(j).*(pol.CL*cos(Theta(j))-pol.CD*sin(Theta(j)));
        DqDr=0.5*rho.*W(j).*W(j).*2.0.*c0i(j).*(pol.CL*sin(Theta(j))-pol.CD*cos(Theta(j)));
        Lambda1 = 1/(XRi(j).*tand(Betai(j)));
        Lambda2 = 1/(XRi(j).*tand(Beta(j)));
    end
end
```

```

[ K1 ] = GoldsteinFactors(Z,XRi(j),Lambda1(j));
[ K2 ] = GoldsteinFactors(Z,XRi(j),Lambda2(j));
aStar = max(roots([4*pi()*rho*Ri(j)*K2(j)*(XVAi(j)*VA)^2*K1(j) 4*pi()*rho*Ri(j)*K2(j)*...
...(XVAi(j)*VA)^2 -0.5*rho.*W(j).*W(j)*2.0.*c0i(j).*(pol.CL*cos(Theta(j))-pol.CD.*sin(Theta(j))))]);
if (abs(aStar-a)<1.0e-6),
    finished=1;
end;
a=aStar;
sum=sum+1;
if (sum>2000),
    finished=1;
end;
end;
Thrust=Thrust+DtDr*Mp;
Torque=Torque+DqDr*Mp;
end
disp(thrust);
disp(torque);

function [x2D,y2D] = BladeSection2D(Meanline,Thickness,Mp,Np,XRi,c0i,f0c0i,t0i)
RG = XRi + (1-XRi).*(sin((0:Mp)*pi/(2*Mp)));
x0 = zeros(1,Np);
x1 = zeros(Mp+1,Np);
for j = 1:Np
    x0(j) = 0.5*(1-cos(pi*(j-1)/(Np-1))); % Cosine spacing along the chord % [0 : 1]
end
for i = 1:Mp+1
    x1(i,:) = c0i(i)*x0; % for each radial section along the span
    % lifting line at mid-chord
end

[junk1, junk2, junk3, fof0_temp, df0f0dxoc_temp, tot0_temp] = GeometryFoil2D(Meanline,Thickness,x0);
fof0 = zeros(Mp+1,Np); % eventually given versus [RG(1:Mp+1),x/c(1:Np)]
df0f0dxoc = zeros(Mp+1,Np); % eventually given versus [RG(1:Mp+1),x/c(1:Np)]
tot0 = zeros(Mp+1,Np); % eventually given versus [RG(1:Mp+1),x/c(1:Np)]

for i = 1:Mp+1
    fof0(i,:) = fof0_temp;
    df0f0dxoc(i,:) = df0f0dxoc_temp;
    tot0(i,:) = tot0_temp;
end

t = zeros(Mp+1,Np);
f = zeros(Mp+1,Np);
dfdx = zeros(Mp+1,Np);

for i = 1:Mp+1
    % for each radial section along the span
    f(i,:) = fof0(i,:)*f0c0i(i)*c0i(i);
    dfdx(i,:) = df0f0dxoc(i,:)*f0c0i(i);
    t(i,:) = tot0(i,:) * t0i(i);
end

% ----- Find 2D unroated section profiles
% x2D [m], x position in 2D space on upper (x2D_u) and lower (x2D_l) foil surfaces
% y2D [m], y position in 2D space on upper (x2D_u) and lower (x2D_l) foil surfaces
x2D_u = zeros(Mp+1,Np); x2D_l = zeros(Mp+1,Np);
y2D_u = zeros(Mp+1,Np); y2D_l = zeros(Mp+1,Np);

for i = 1:Mp+1
    % for each section along the span
    for j = 1:Np
        % for each point along the chord
        x2D_u(i,j) = x1(i,j) + (t(i,j)/2)*sin(atan(dfdx(i,j))); % 2D upper surface x
        x2D_l(i,j) = x1(i,j) - (t(i,j)/2)*sin(atan(dfdx(i,j))); % 2D lower surface x
        y2D_u(i,j) = f(i,j) + (t(i,j)/2)*cos(atan(dfdx(i,j))); % 2D upper surface y
        y2D_l(i,j) = f(i,j) - (t(i,j)/2)*cos(atan(dfdx(i,j))); % 2D lower surface y
    end
end

x2D = zeros(Mp+1,2*Np);
y2D = zeros(Mp+1,2*Np);
x2D(:,1:Np) = x2D_u(:,Np:-1:1); % The first Np values are the upper surface (suction side).
x2D(:,Np+1:Np+Np) = x2D_l(:,1:Np); % and the second Np values are the lower surface (pressure side).
y2D(:,1:Np) = y2D_u(:,Np:-1:1);
y2D(:,Np+1:Np+Np) = y2D_l(:,1:Np);
end

function [f0ocilde, CLItilde, alphaItilde, fof0, df0f0dxoc, tot0, As0] = GeometryFoil2D(Meanline,Thickness,x0)

if nargin == 0
    % Print a list of available meanlines and thickness profiles:
    disp(' ')
    disp(' Available meanline and thickness profiles:')
    disp(' ')
    disp(' Available meanline profiles:')
    disp(' NACA a=0.8')
    disp(' NACA a=0.8 (modified)')
    disp(' parabolic')
    disp(' flat')
    disp(' ')
    disp(' Available thickness profiles:')
    disp(' NACA 65A010')
    disp(' NACA 65A010 (Epps modified)')
    disp(' NACA 66 (DTRC modified)')
    disp(' NACA 00xx')
    disp(' elliptic')
    disp(' parabolic')
    return
elseif nargin == 2
    x0 = [0:0.1:1];
end

% -----

```

```

% Error checking for GUI:
% -----
Print_list = 0;

if Meanline==0 | strcmp(Meanline,'NACA a=0.8 (modified)') %—Use NACA a=0.8 (modified) meanline
    Meanline = 'NACA a=0.8 (modified)';
elseif Meanline==1 | strcmp(Meanline,'NACA a=0.8') %—Use NACA a=0.8 meanline
    Meanline = 'NACA a=0.8';
elseif Meanline==2 | strcmp(Meanline,'parabolic') | strcmp(Meanline,'Parabolic') %—Use parabolic meanline
    Meanline = 'parabolic';
elseif strcmp(Meanline,'flat')
    else
        Print_list = 1;
end

if Thickness==1 | strcmp(Thickness,'NACA 65A010') %—Use NACA 65A010 thickness form
    Thickness = 'NACA 65A010';
elseif Thickness==2 | strcmp(Thickness,'elliptic') | strcmp(Thickness,'elliptical') %—Use elliptical thickness form
    Thickness = 'elliptic';
elseif Thickness==3 | strcmp(Thickness,'parabolic') %—Use parabolic thickness form
    Thickness = 'parabolic';
elseif Thickness==4 | strcmp(Thickness,'NACA 65A010 (Epps modified)') | strcmp(Thickness,'NACA 65A010 (modified)')
    Thickness = 'NACA 65A010 (Epps modified)';
elseif Thickness==5 | strcmp(Thickness,'NACA 66 (DTRC modified)') | strcmp(Thickness,'NACA66 (DTRC modified)')
    Thickness = 'NACA 66 (DTRC modified)';
elseif Thickness==6 | strcmp(Thickness,'NACA 00xx')
    Thickness = 'NACA 00xx';
else
    Print_list = 1;
end

if Print_list == 1
    % Print a list of available meanlines and thickness profiles:
    disp(' ')
    disp('Available meanline and thickness profiles:')
    disp(' ')
    disp('Available meanline profiles:')
    disp(' NACA a=0.8')
    disp(' NACA a=0.8 (modified)')
    disp(' parabolic')
    disp(' flat')
    disp(' ')
    disp('Available thickness profiles:')
    disp(' NACA 65A010')
    disp(' NACA 65A010 (Epps modified)')
    disp(' NACA 66 (DTRC modified)')
    disp(' elliptic')
    disp(' parabolic')
    return
end
% -----

% -----
% MEANLINE PROFILE
% -----

% ----- NACA a=0.8 (modified) meanline
if strcmp(Meanline,'NACA a=0.8 (modified)')

    xoc = [0 .5 .75 1.25 2.5 5 7.5 10 15 20 25 30 35 40 45 50 ...
            55 60 65 70 75 80 85 90 95 100]./100;

    foc = [0 0.281 0.396 0.603 1.055 1.803 2.432 2.981 3.903 4.651 5.257 ...
            5.742 6.120 6.394 6.571 6.651 6.631 6.508 6.274 5.913 5.401 ...
            4.673 3.607 2.452 1.226 0 ]./100;

    % dfdxoc(1) == interp1(xoc(2:end),dfdxoc(2:end),0,'pchip','extrap')
    dfdxoc = [ 0.53699 0.47539 0.44004 0.39531 0.33404 0.27149 0.23378 0.20618 0.16546 ...
                0.13452 0.10873 0.08595 0.06498 0.04507 0.02559 0.00607 ...
                -0.01404 -0.03537 -0.05887 -0.08610 -0.12058 -0.18034 -0.23430 ...
                -0.24521 -0.24521 -0.24521];

    alphaItilde = 1.40; % [deg] NACA data ideal angle of attack
    CLItilde = 1.00; % [ ] NACA data ideal lift coefficient
    f0oocilde = max(foc); % f0/c NACA data for CLI == CLItilde

    % -----
    fof0 = foc/f0oocilde; % f/f0 at xoc = x/c positions
    df0f0dxoc = dfdxoc/f0oocilde; % d(f/f0)/d(x/c) at xoc = x/c positions

    fof0 = pchip(xoc, fof0, x0); % f/f0 at x0 = x/c positions
    df0f0dxoc = pchip(xoc, df0f0dxoc, x0); % d(f/f0)/d(x/c) at x0 = x/c positions

% ----- NACA a=0.8 meanline
elseif strcmp(Meanline,'NACA a=0.8')

    % -----
    % NACA a=0.8 meanline
    % TABLExN = [0.0000 0.0125 0.0250 0.0500 0.0750 0.1000 0.1500 0.2000 0.3000 ...
    % 0.4000 0.4500 0.5000 0.6000 0.7000 0.8000 0.9000 0.9500 1.0000];
    % TABLEfof0 = [0.0000 0.0907 0.1586 0.2712 0.3657 0.4482 0.5869 0.6993 0.8635 ...
    % 0.9615 0.9881 1.0000 0.9786 0.8892 0.7027 0.3586 0.1713 0.0000];
    % TABLEdfdx = interp1(0.5*(TABLExN(1:end-1)+TABLExN(2:end)),diff(TABLEfof0)./diff(TABLExN),TABLExN,'pchip','extrap');
    % % yields [8.3847 6.1752 5.0305 4.0985 3.5190 3.1063 2.4930 2.0291 1.2979 ...
    % 0.6792 0.3852 0.0901 -0.5337 -1.3293 -2.7246 -3.7019 -3.6522 -3.1505];

    % Camber distribution and slope of meanline — excluding ideal angle of attack
    % xcn = TABLExN;

```

```

% xcn = 0.4:0.0000001:0.6;
xcn = x0; % NOTE: indexed leading edge to trailing edge

a = 0.8; % For NACA a=0.8
g = -1/(1-a) * (a^2*(log(a)/2 - 1/4)+1/4);
h = 1/(1-a) * ((1-a)^2*log(1-a)/2 - (1-a)^2/4) + g;

C1 = max(1 - xcn,1e-6);
CA = a - xcn;

N = length(xcn);

for i = 1:N
    if abs(CA(i))<1e-6
        CA(i) = CA(i)+1e-5;
    end
end

P = 1/2*CA.^2.*log(abs(CA))-1/2*C1.^2.*log(C1)+1/4*(C1.^2-CA.^2);

% Camber distribution — excluding ideal angle of attack
F = (P/(1-a)-xcn.*log(xcn)+g-h*xcn)/(2*pi*(a+1));

% Slope of meanline — excluding ideal angle of attack
B = ((-CA.*log(abs(CA))+C1.*log(C1))/(1-a) - log(xcn) -1-h)/(2*pi*(a+1));

% max(F) == 0.067943423879
% xcn( find(F==max(F)) ) == 0.5153285
maxF = 0.067943423879;

% maxF = 0.067895802340206; % == F at x/c = 0.50, which is assumed to be the max f/c in the table above!

fof0 = F / maxF; % camber of meanline / max camber ( f/fo )
dfdx = B / maxF; % slope of meanline / max camber ( d(f/fo)/d(x/c) )

if x0(1) == 0
% fig(10001); plot(x0,fof0,'.-r')
% fig(10002); plot(x0,dfdx,'.-r')

    fof0(1) = 0;
    dfdx(1) = spline(x0(2:end),dfdx(2:end),x0(1));%,'spline','extrap');

% fig(10001); plot(x0,fof0,'.-b')
% fig(10002); plot(x0,dfdx,'.-b')
end

% [xv,xcn,fof0,dfdx]
%
% xcn = TABLExN ... dfdx = B./maxF, fof0 = F./maxF ... yields:
% TABLEfof0 = [NaN 0.0906 0.1585 0.2709 0.3654 0.4478 0.5866 0.6988 0.8629 0.9609 ...
% 0.9874 0.9993 0.9779 0.8885 0.7022 0.3584 0.1711 0.0000];
% TABLEdfdx = [Inf 5.9401 5.0195 4.0795 3.5128 3.0982 2.4861 2.0214 1.2915 0.6771 ...
% 0.3846 0.0913 -0.5315 -1.2937 -2.7092 -3.7653 -3.6654 -3.0004];
%
% fof0_table = interp1(TABLExN(2:end),TABLEfof0(2:end),(0.5-xv),'spline','extrap');
% dfdx_table = interp1(TABLExN(2:end),TABLEdfdx(2:end),(0.5-xv),'spline','extrap');
% [fof0, fof0_table]
% [dfdx, dfdx_table]
%
%
% xcn = TABLExN ... dfdx = B./max(F), fof0 = F./max(F) ... yields:
% TABLEfof0 = [NaN 0.0907 0.1586 0.2711 0.3657 0.4481 0.5870 0.6993 0.8635 0.9615 ...
% 0.9881 1.0000 0.9786 0.8892 0.7027 0.3586 0.1712 0.0000];
% TABLEdfdx = [Inf 5.9442 5.0230 4.0824 3.5153 3.1004 2.4878 2.0228 1.2924 0.6776 ...
% 0.3849 0.0914 -0.5319 -1.2946 -2.7111 -3.7679 -3.6680 -3.0025];
%
% whereas: TABLEdfdx = interp1(0.5*(TABLExN(1:end-1)+TABLExN(2:end)),diff(TABLEfof0)./diff(TABLExN),TABLExN,'pchip','extrap');
% yels: TABLEdfdx = [8.3847 6.1752 5.0305 4.0985 3.5190 3.1063 2.4930 2.0291 ...
% 1.2979 0.6792 0.3852 0.0901 -0.5337 -1.3293 -2.7246 -3.7019 -3.6522 -3.1505];
%
% figure, plot(TABLExN,TABLEfof0,'r',xcn,fof0,'.b')
% figure, plot(TABLExN,TABLEdfdx,'r',xcn,dfdx,'.b')
%
%
% alphaItilde = 1.54; % [deg] NACA data ideal angle of attack
% CLItilde = 1.00; % [ ] NACA data ideal lift coefficient
% f0oocilde = maxF; % f0/c NACA data for CLI == CLItilde

dfdx0dxoc = dfdx;

% ----- NACA a=0.8 meanline
elseif strcmp(Meanline,'NACA a=0.8 (v1)')

xoc = [0 0.5 0.75 1.25 2.5 5 7.5 10 15 20 25 30 35 40 45 50 55 60 65 ...
70 75 80 85 90 95 100]/100;
foc = [0 0.287 0.404 0.616 1.077 1.841 2.483 3.043 3.985 4.748 5.367 5.863 6.248 ...
6.528 6.709 6.790 6.770 6.644 6.405 6.037 5.514 4.771 3.683 2.435 1.163 0]/100;

% dfdxoc(1) == interp1(xoc(2:end),dfdxoc(2:end),0,'pchip','extrap')
dfdxoc = [0.54823 .48535 .44925 .40359 .34104 .27718 .23868 .21050 .16892 .13734 ...
.11101 .08775 .06634 .04601 .02613 .00620 -.01433 -.03611 -.06010 -.08790 -.12311 ...
-.18412 -.23921 -.25583 -.24904 -.20385];

alphaItilde = 1.54; % [deg] NACA data ideal angle of attack
% CLItilde = 1.00; % [ ] NACA data ideal lift coefficient
f0oocilde = max(foc); % f0/c NACA data for CLI == CLItilde

```

```
%
fof0      = foc/f0ociltde;           % f/f0      at xoc = x/c positions
dfof0dxoc = dfdxoc/f0ociltde;       % d(f/f0)/d(x/c) at xoc = x/c positions

fof0      = pchip(xoc, fof0, x0);   % f/f0      at x0 = x/c positions
dfof0dxoc = pchip(xoc, dfof0dxoc, x0); % d(f/f0)/d(x/c) at x0 = x/c positions

% ----- NACA a=0.8 meanline
elseif strcmp(Meanline, 'NACA a=0.8 (v2)')

    alphaItilde = 1.54;              % [deg] NACA data ideal angle of attack
    CLItilde    = 1.00;              % [ ] NACA data ideal lift coefficient
    f0ociltde   = 0.0679;           % f0/c NACA data for CLI == CLItilde

    xoc         = [0.0000 0.0125 0.0250 0.0500 0.0750 0.1000 0.1500 0.2000 0.3000 ...
0.4000 0.4500 0.5000 0.6000 0.7000 0.8000 0.9000 0.9500 1.0000];
    fof0        = [0.0000 0.0907 0.1586 0.2712 0.3657 0.4482 0.5869 0.6993 0.8635 ...
0.9615 0.9881 1.0000 0.9786 0.8892 0.7027 0.3586 0.1713 0.0000];
    dfof0dxoc   = pchipslopes(xoc, fof0); % == d(f/f0)/d(x/c)

    fof0        = pchip(xoc, fof0, x0); % f/f0      at x0 = x/c positions
    dfof0dxoc   = pchip(xoc, dfof0dxoc, x0); % d(f/f0)/d(x/c) at x0 = x/c positions

% ----- Parabolic meanline
elseif strcmp(Meanline, 'parabolic')

    % For parabolic meanline: alphaItilde == 0, CLItilde == 4*pi*f0oc
    % Therefore, set CLItilde and f0ociltde such that f0oc == f0ociltde * CL / CLItilde
    alphaItilde = 0;
    CLItilde    = 1;
    f0ociltde   = 1 / (4*pi);

    fof0        = (1-(2*(x0-0.5)).^2);
    dfof0dxoc   = -8 * (x0-0.5);

% ----- Flat plate meanline
elseif strcmp(Meanline, 'flat')

    % For flat plate meanline: alphaItilde == f0ociltde == CLItilde == 0
    % However, set CLItilde == 1 such that f0oc == f0ociltde * CL / CLItilde does not crash
    alphaItilde = 0;
    CLItilde    = 1;
    f0ociltde   = 0;

    fof0        = 0*x0;
    dfof0dxoc   = 0*x0;

% -----
else
    disp('ERROR: Meanline type unknown')
    disp(' ')

    % Print a list of available meanlines profiles:
    disp('Available meanline profiles:')
    disp(' NACA a=0.8')
    disp(' NACA a=0.8 (modified)')
    disp(' parabolic')
    disp(' ')
end

% -----
% THICKNESS PROFILE
% -----

% ----- NACA 65A010 thickness form
if strcmp(Thickness, 'NACA 65A010')

    xoc = [0 .5 .75 1.25 2.5 5 7.5 10 15 20 25 30 35 40 45 50 ...
55 60 65 70 75 80 85 90 95 100]./100;

    toc = [0 .765 .928 1.183 1.623 2.182 2.65 3.04 3.658 4.127 ...
4.483 4.742 4.912 4.995 4.983 4.863 4.632 4.304 ...
3.899 3.432 2.912 2.352 1.771 1.188 .604 .021]./100;

    tot0 = toc / max(toc);           % t/t0      at xoc = x/c positions

    % -----
    tot0 = pchip(xoc, tot0, x0);     % t/t0      at x0 = x/c positions

    As0 = 0.6771; % == trapz(x0, tot0), with x0 = 0:0.00001:1; cross-sectional area for c == t0 == 1

% ----- NACA 65A010 (modified) thickness form
elseif strcmp(Thickness, 'NACA 65A010 (Epps modified)') | strcmp(Thickness, 'NACA 65A010 (modified)')

    xoc = [0 0.005000000000000 0.007500000000000 0.012500000000000 ...
0.025000000000000 0.050000000000000 0.075000000000000 0.100000000000000 ...
0.150000000000000 0.200000000000000 0.250000000000000 0.300000000000000 ...
0.350000000000000 0.400000000000000 0.471204188481675 0.523560209424084 ...
0.575916230366492 0.628272251308901 0.680628272251309 0.732984293193717 ...
0.785340314136126 0.837696335078534 0.890052356020942 0.942408376963351 ...
0.968586387434555 0.981675392670157 0.989528795811518 0.994764397905759 ...
0.997382198952880 1.000000000000000];
    toc = [0 0.007650000000000 0.009280000000000 0.011830000000000 ...
0.016230000000000 0.021820000000000 0.026500000000000 0.030400000000000 ...
0.036580000000000 0.041270000000000 0.044830000000000 0.047420000000000 ...
0.049120000000000 0.049950000000000 0.048630000000000 0.048300000000000 ...
0.048000000000000 0.047700000000000 0.047400000000000 0.047100000000000 ...
0.046800000000000 0.046500000000000 0.046200000000000 0.045900000000000 ...
0.045600000000000 0.045300000000000 0.045000000000000 0.044700000000000 ...
0.044400000000000 0.044100000000000 0.043800000000000 0.043500000000000 ...
0.043200000000000 0.042900000000000 0.042600000000000 0.042300000000000 ...
0.042000000000000 0.041700000000000 0.041400000000000 0.041100000000000 ...
0.040800000000000 0.040500000000000 0.040200000000000 0.039900000000000 ...
0.039600000000000 0.039300000000000 0.039000000000000 0.038700000000000 ...
0.038400000000000 0.038100000000000 0.037800000000000 0.037500000000000 ...
0.037200000000000 0.036900000000000 0.036600000000000 0.036300000000000 ...
0.036000000000000 0.035700000000000 0.035400000000000 0.035100000000000 ...
0.034800000000000 0.034500000000000 0.034200000000000 0.033900000000000 ...
0.033600000000000 0.033300000000000 0.033000000000000 0.032700000000000 ...
0.032400000000000 0.032100000000000 0.031800000000000 0.031500000000000 ...
0.031200000000000 0.030900000000000 0.030600000000000 0.030300000000000 ...
0.030000000000000 0.029700000000000 0.029400000000000 0.029100000000000 ...
0.028800000000000 0.028500000000000 0.028200000000000 0.027900000000000 ...
0.027600000000000 0.027300000000000 0.027000000000000 0.026700000000000 ...
0.026400000000000 0.026100000000000 0.025800000000000 0.025500000000000 ...
0.025200000000000 0.024900000000000 0.024600000000000 0.024300000000000 ...
0.024000000000000 0.023700000000000 0.023400000000000 0.023100000000000 ...
0.022800000000000 0.022500000000000 0.022200000000000 0.021900000000000 ...
0.021600000000000 0.021300000000000 0.021000000000000 0.020700000000000 ...
0.020400000000000 0.020100000000000 0.019800000000000 0.019500000000000 ...
0.019200000000000 0.018900000000000 0.018600000000000 0.018300000000000 ...
0.01800000
```

```

0.0463200000000000    0.0430400000000000    0.0389900000000000    0.0343200000000000 ...
0.0291200000000000    0.0235200000000000    0.0177100000000000    0.0118800000000000 ...
0.0089600000000000    0.007499530848329    0.006623639691517    0.0060400000000000 ...
0.004049015364794    0.0002100000000000];

tot0 = toc / max(toc); % t/t0 at xoc = x/c positions

% -----
tot0 = pchip(xoc,tot0,x0); % t/t0 at x0 = x/c positions

As0 = 0.7107; % == trapz(x0,tot0), with x0 = 0:0.00001:1; cross-sectional area for c == t0 == 1

% ----- NACA 66 (DTRC Modified) thickness form
elseif strcmp(Thickness,'NACA 66 (DTRC modified)') | strcmp(Thickness,'NACA66 (DTRC Modified)')

xoc = [0.0000 0.0125 0.0250 0.0500 0.0750 0.1000 0.1500 0.2000 0.3000 0.4000 0.4500 ...
0.5000 0.6000 0.7000 0.8000 0.9000 0.9500 1.0000];
tot0 = [0.0000 0.2088 0.2932 0.4132 0.5050 0.5814 0.7042 0.8000 0.9274 0.9904 1.0000 ...
0.9924 0.9306 0.8070 0.6220 0.3754 0.2286 0.0000];

% -----
tot0 = spline(xoc,tot0,x0); % t/t0 at x0 = x/c positions

As0 = 0.7207; % == trapz(x0,tot0), with x0 = 0:0.00001:1; cross-sectional area for c == t0 == 1

% ----- NACA 00xx (four-digit airfoil)
elseif strcmp(Thickness,'NACA 00xx') | strcmp(Thickness,'NACA00xx')

yot0 = ( 0.29690*sqrt(x0) ... % This NACA 00xx formula gives the y ordinate
- 0.12600*x0 ... % of the upper and lower surface (i.e. the half-thickness).
- 0.35160*x0.^2 ... % Here, we assume t0/c = 1 and thus ignore it in the formula,
+ 0.28430*x0.^3 ... % so this is in fact: y/t0 at the x0 = x/c positions.
- 0.10150*x0.^4) / 0.20; % The thickness of the section is twice the y values: (t/t0) = 2*(y/t0)

tot0 = 2 * yot0; % t/t0 at x0 = x/c positions

As0 = 0.3425; % == trapz(x0,tot0), with x0 = 0:0.00001:1; cross-sectional area for c == t0 == 1

% ----- Elliptical thickness form
elseif strcmp(Thickness,'elliptic')

tot0 = sqrt(1-(2*(x0-0.5)).^2);
As0 = pi/4;

% ----- Parabolic thickness form
elseif strcmp(Thickness,'parabolic')

tot0 = (1-(2*(x0-0.5)).^2);
As0 = 2/3;

% -----
else
disp('ERROR: Thickness type unknown')
disp(' ')

% Print a list of available thickness profiles:
disp(' Available thickness profiles:')
disp(' NACA 65A010')
disp(' NACA 65A010 (Epps modified)')
disp(' NACA 66 (DTRC modified)')
disp(' elliptic')
disp(' parabolic')
end

function [ K ] = GoldsteinFactors( Z,XRi,Lambda )
r_R = [0.200, 0.300, 0.400, 0.500, 0.600, 0.700, 0.800, 0.850, 0.900, 0.925, 0.950, 0.975];
beta_ii = [0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 2.00, 2.25, 2.50, 2.75, 3.00, 3.25, 3.50, 3.75, ...
4.00, 4.25, 4.50, 4.75, 5.00, 5.25, 5.50, 5.75, 6.00, 7.00, 8.00, 9.00, 10.00, 11.00, 12.00];
if Z == 3;
Xi = [[0.16344e+01, 0.11819e+01, 0.90911e+00, 0.71839e+00, 0.57137e+00, 0.44848e-00, 0.33662e-00, ...
0.28065e-00, 0.22101e-00, 0.18801e-00, 0.15066e-00, 0.10400e-00];... %0.25
[0.16034e+01, 0.11+56e+01, 0.90367e+00, 0.72150e+00, 0.58102e+00, 0.46262e-00, 0.35280e-00, ...
0.29666e-00, 0.23573e-00, 0.20148e-00, 0.16228e-00, 0.11275e-00];... %0.50
[0.15600e+01, 0.11440e+01, 0.89833e+00, 0.72874e+00, 0.59767e+00, 0.48543e-00, 0.37801e-00, ...
0.32126e-00, 0.25803e-00, 0.22175e-00, 0.17962e-00, 0.12562e-00];... %0.75
[0.15101e+01, 0.11208e+01, 0.89501e+00, 0.74055e+00, 0.62049e+00, 0.51510e+00, 0.40980e-00, ...
0.35193e-00, 0.28557e-00, 0.24666e-00, 0.20084e-00, 0.14129e-00];... %1.00
[0.14576e+01, 0.10982e+01, 0.89424e+00, 0.75617e+00, 0.64770e+00, 0.54917e+00, 0.44552e-00, ...
0.38608e-00, 0.31600e-00, 0.27410e-00, 0.22412e-00, 0.15843e-00];... %1.25
[0.14054e+01, 0.10772e+01, 0.89585e+00, 0.77441e+00, 0.67741e+00, 0.58540e+00, 0.48293e-00, ...
0.42165e-00, 0.34754e-00, 0.30247e-00, 0.24816e-00, 0.17607e-00];... %1.50
[0.13552e+01, 0.10583e+01, 0.89936e+00, 0.79404e+00, 0.70795e+00, 0.62198e+00, 0.52035e+00, ...
0.45715e-00, 0.37896e-00, 0.30072e-00, 0.27206e-00, 0.19361e-00];... %1.75
[0.13081e+01, 0.10418e+01, 0.90423e+00, 0.81402e+00, 0.73805e+00, 0.65761e+00, 0.55669e+00, ...
0.49161e-00, 0.40947e-00, 0.35816e-00, 0.29530e-00, 0.21065e-00];... %2.00
[0.12649e+01, 0.10277e+01, 0.90995e+00, 0.83357e+00, 0.76679e+00, 0.69145e+00, 0.59123e+00, ...
0.52444e+00, 0.43863e+00, 0.38442e-00, 0.31757e-00, 0.22701e-00];... %2.25
[0.12258e+01, 0.10157e+01, 0.91612e+00, 0.85215e+00, 0.79363e+00, 0.72300e+00, 0.62362e+00, ...
0.55535e+00, 0.46620e-00, 0.40932e-00, 0.33874e-00, 0.24258e-00];... %2.50
[0.11909e+01, 0.10058e+01, 0.92242e+00, 0.86941e+00, 0.81826e+00, 0.75201e+00, 0.65369e+00, ...
0.58421e+00, 0.49212e-00, 0.43279e-00, 0.35875e-00, 0.25736e-00];... %2.75
[0.11599e+01, 0.99764e+00, 0.92864e+00, 0.88519e+00, 0.84056e+00, 0.77844e+00, 0.68142e+00, ...
0.61105e+00, 0.51640e+00, 0.45487e-00, 0.37765e-00, 0.27136e-00];... %3.00
[0.11326e+01, 0.99104e+00, 0.93462e+00, 0.89942e+00, 0.86055e+00, 0.80233e+00, 0.70690e+00, ...
0.63592e+00, 0.53912e+00, 0.47562e-00, 0.39549e-00, 0.28463e-00];... %3.25
[0.11088e+01, 0.98578e+00, 0.94027e+00, 0.91213e+00, 0.87833e+00, 0.82382e+00, 0.73025e+00, ...
0.65896e+00, 0.56038e+00, 0.49514e-00, 0.41235e-00, 0.29723e-00];... %3.50
[0.10881e+01, 0.98168e+00, 0.94555e+00, 0.92339e+00, 0.89404e+00, 0.84307e+00, 0.75162e+00, ...
0.68029e+00, 0.58029e+00, 0.51352e+00, 0.42831e-00, 0.30921e-00];... %3.75

```

```

[0.10701e+01, 0.97855e+00, 0.95043e+00, 0.93331e+00, 0.90786e+00, 0.86028e+00, 0.77117e+00, ...
0.70005e+00, 0.59897e+00, 0.53086e+00, 0.44345e-00, 0.32064e-00];... %4.00
[0.10547e+01, 0.97624e+00, 0.95492e+00, 0.94200e+00, 0.91997e+00, 0.87561e+00, 0.78905e+00, ...
0.71838e+00, 0.61652e+00, 0.54725e+00, 0.45785e-00, 0.33157e-00];... %4.25
[0.10414e+01, 0.97462e+00, 0.95902e+00, 0.94960e+00, 0.93055e+00, 0.88927e+00, 0.80543e+00, ...
0.73540e+00, 0.63304e+00, 0.56279e+00, 0.47158e-00, 0.34204e-00];... %4.50
[0.10300e+01, 0.97356e+00, 0.96276e+00, 0.95621e+00, 0.93977e+00, 0.90143e+00, 0.82043e+00, ...
0.75124e+00, 0.64863e+00, 0.57754e+00, 0.48469e-00, 0.35210e-00];... %4.75
[0.10203e+01, 0.97297e+00, 0.96615e+00, 0.96195e+00, 0.94778e+00, 0.91224e+00, 0.83419e+00, ...
0.76598e+00, 0.66336e+00, 0.59157e+00, 0.49725e-00, 0.36179e-00];... %5.00
[0.10120e+01, 0.97276e+00, 0.96924e+00, 0.96693e+00, 0.95475e+00, 0.92185e+00, 0.84682e+00, ...
0.77974e+00, 0.67731e+00, 0.60495e+00, 0.50929e+00, 0.37118e-00];... %5.25
[0.10050e+01, 0.97286e+00, 0.97203e+00, 0.97125e+00, 0.96079e+00, 0.93039e+00, 0.85842e+00, ...
0.79258e+00, 0.69054e+00, 0.61773e+00, 0.52087e+00, 0.38017e-00];... %5.50
[0.99915e+00, 0.97320e+00, 0.97456e+00, 0.97498e+00, 0.96602e+00, 0.93798e+00, 0.86909e+00, ...
0.80459e+00, 0.70310e+00, 0.62995e+00, 0.53202e+00, 0.38892e-00];... %5.75
[0.99422e+00, 0.97373e+00, 0.97686e+00, 0.97821e+00, 0.97055e+00, 0.94473e+00, 0.87891e+00, ...
0.81584e+00, 0.71505e+00, 0.64166e+00, 0.54276e+00, 0.39741e+00];... %6.00
[0.98170e+00, 0.97700e+00, 0.98406e+00, 0.98732e+00, 0.98335e+00, 0.96506e+00, 0.91109e+00, ...
0.85428e+00, 0.75750e+00, 0.68400e+00, 0.58227e+00, 0.42907e-00];... %7.00
[0.97668e+00, 0.98091e+00, 0.98892e+00, 0.99244e+00, 0.99050e+00, 0.97783e+00, 0.93449e+00, ...
0.88431e+00, 0.79293e+00, 0.72039e+00, 0.61716e+00, 0.45770e-00];... %8.00
[0.97668e+00, 0.98462e+00, 0.99222e+00, 0.99535e+00, 0.99452e+00, 0.98588e+00, 0.95163e+00, ...
0.90796e+00, 0.82277e+00, 0.75201e+00, 0.64832e+00, 0.48392e-00];... %9.00
[0.97675e+00, 0.98781e+00, 0.99447e+00, 0.99704e+00, 0.99678e+00, 0.99098e+00, 0.96424e+00, ...
0.92668e+00, 0.84808e+00, 0.77968e+00, 0.67638e+00, 0.50812e+00];... %10.00
[0.97878e+00, 0.99042e+00, 0.99602e+00, 0.99805e+00, 0.99807e+00, 0.99422e+00, 0.97354e+00, ...
0.94155e+00, 0.86964e+00, 0.80401e+00, 0.70179e+00, 0.53060e+00];... %11.00
[0.98115e+00, 0.99250e+00, 0.99710e+00, 0.99867e+00, 0.99881e+00, 0.99629e+00, 0.98041e+00, ...
0.95338e+00, 0.88805e+00, 0.82549e+00, 0.72490e+00, 0.55158e+00];... %12.00
elseif Z==4;
Xi = [[0.15743e+01, 0.12486e+01, 0.10173e+01, 0.83596e+00, 0.68374e+00, 0.54793e+00, 0.41783e-00, ...
0.35068e-00, 0.27787e-00, 0.23711e-00, 0.19067e-00, 0.13232e-00];...
[0.15453e+01, 0.12300e+01, 0.10092e+01, 0.83744e+00, 0.69334e+00, 0.56356e+00, 0.43664e-00, ...
0.36961e-00, 0.29549e-00, 0.25332e-00, 0.20470e-00, 0.14293e-00];...
[0.15035e+01, 0.12043e+01, 0.99924e+00, 0.84179e+00, 0.70954e+00, 0.58828e+00, 0.46548e-00, ...
0.39827e-00, 0.32188e-00, 0.27746e-00, 0.22548e-00, 0.15841e-00];...
[0.14547e+01, 0.11755e+01, 0.98977e+00, 0.84952e+00, 0.73121e+00, 0.61965e+00, 0.50109e+00, ...
0.43333e-00, 0.35388e-00, 0.30662e-00, 0.26047e-00, 0.17696e-00];...
[0.14033e+01, 0.11464e+01, 0.98189e+00, 0.86007e+00, 0.75639e+00, 0.65474e+00, 0.54017e+00, ...
0.47154e-00, 0.38857e-00, 0.33815e-00, 0.27743e-00, 0.19691e-00];...
[0.13523e+01, 0.11189e+01, 0.97594e+00, 0.87245e+00, 0.78306e+00, 0.69095e+00, 0.58006e+00, ...
0.51043e+00, 0.42380e-00, 0.37015e-00, 0.30477e-00, 0.21713e-00];...
[0.13040e+01, 0.10940e+01, 0.97183e+00, 0.88563e+00, 0.80957e+00, 0.72635e+00, 0.61893e+00, ...
0.54834e+00, 0.45819e-00, 0.40141e-00, 0.33150e-00, 0.23691e-00];...
[0.12594e+01, 0.10721e+01, 0.96930e+00, 0.89880e+00, 0.83476e+00, 0.75969e+00, 0.65865e+00, ...
0.58429e+00, 0.49095e-00, 0.43126e-00, 0.35709e-00, 0.25589e-00];...
[0.12193e+01, 0.10533e+01, 0.96805e+00, 0.91138e+00, 0.85793e+00, 0.79027e+00, 0.68964e+00, ...
0.61779e+00, 0.52170e+00, 0.45939e-00, 0.38129e-00, 0.27390e-00];...
[0.11837e+01, 0.10375e+01, 0.96779e+00, 0.92300e+00, 0.87872e+00, 0.81779e+00, 0.72068e+00, ...
0.64868e+00, 0.55033e+00, 0.48571e-00, 0.40404e-00, 0.29090e-00];...
[0.11525e+01, 0.10245e+01, 0.96825e+00, 0.93350e+00, 0.89705e+00, 0.84223e+00, 0.74879e+00, ...
0.67698e+00, 0.57689e+00, 0.51025e+00, 0.42538e-00, 0.30694e-00];...
[0.11254e+01, 0.10139e+01, 0.96922e+00, 0.94282e+00, 0.91299e+00, 0.86371e+00, 0.77410e+00, ...
0.70282e+00, 0.60148e+00, 0.53314e+00, 0.44541e-00, 0.32208e-00];...
[0.11021e+01, 0.10055e+01, 0.97054e+00, 0.95097e+00, 0.92670e+00, 0.88247e+00, 0.79682e+00, ...
0.72640e+00, 0.62425e+00, 0.55450e+00, 0.46423e-00, 0.33640e-00];...
[0.10820e+01, 0.99881e+00, 0.97209e+00, 0.95805e+00, 0.93841e+00, 0.89875e+00, 0.81719e+00, ...
0.74790e+00, 0.64539e+00, 0.57448e+00, 0.48197e-00, 0.34999e-00];...
[0.10650e+01, 0.99365e+00, 0.97376e+00, 0.96414e+00, 0.94834e+00, 0.91285e+00, 0.83544e+00, ...
0.76752e+00, 0.66503e+00, 0.59320e+00, 0.49872e-00, 0.36292e-00];...
[0.10505e+01, 0.98975e+00, 0.97549e+00, 0.96935e+00, 0.95672e+00, 0.92502e+00, 0.85179e+00, ...
0.78546e+00, 0.68332e+00, 0.61079e+00, 0.51459e+00, 0.37526e-00];...
[0.10382e+01, 0.98688e+00, 0.97723e+00, 0.97380e+00, 0.96377e+00, 0.93550e+00, 0.86645e+00, ...
0.80188e+00, 0.70040e+00, 0.62736e+00, 0.52967e+00, 0.38708e-00];...
[0.10278e+01, 0.98484e+00, 0.97893e+00, 0.97759e+00, 0.96969e+00, 0.94452e+00, 0.87959e+00, ...
0.81692e+00, 0.71637e+00, 0.64300e+00, 0.54402e+00, 0.39841e-00];...
[0.10191e+01, 0.98347e+00, 0.98057e+00, 0.98080e+00, 0.97464e+00, 0.95228e+00, 0.89139e+00, ...
0.83073e+00, 0.73134e+00, 0.65781e+00, 0.55772e+00, 0.40931e-00];...
[0.10119e+01, 0.98264e+00, 0.98213e+00, 0.98353e+00, 0.97877e+00, 0.95894e+00, 0.90200e+00, ...
0.84343e+00, 0.74539e+00, 0.67183e+00, 0.57083e+00, 0.41982e-00];...
[0.10058e+01, 0.98223e+00, 0.98361e+00, 0.98585e+00, 0.98223e+00, 0.96467e+00, 0.91153e+00, ...
0.85511e+00, 0.75861e+00, 0.68516e+00, 0.58338e+00, 0.42997e-00];...
[0.10008e+01, 0.98215e+00, 0.98499e+00, 0.98781e+00, 0.98510e+00, 0.96959e+00, 0.92011e+00, ...
0.86587e+00, 0.77105e+00, 0.69782e+00, 0.59542e+00, 0.43978e-00];...
[0.99664e+00, 0.98232e+00, 0.98628e+00, 0.98948e+00, 0.98750e+00, 0.97382e+00, 0.92784e+00, ...
0.87580e+00, 0.78278e+00, 0.70983e+00, 0.60699e+00, 0.44928e-00];...
[0.99326e+00, 0.98268e+00, 0.98747e+00, 0.99089e+00, 0.98950e+00, 0.97745e+00, 0.93480e+00, ...
0.88496e+00, 0.79385e+00, 0.72138e+00, 0.61812e+00, 0.45850e-00];...
[0.98525e+00, 0.98518e+00, 0.99136e+00, 0.99475e+00, 0.99468e+00, 0.98744e+00, 0.95646e+00, ...
0.91515e+00, 0.83238e+00, 0.76243e+00, 0.65880e+00, 0.49288e-00];...
[0.98272e+00, 0.98812e+00, 0.99405e+00, 0.99683e+00, 0.99721e+00, 0.99308e+00, 0.97086e+00, ...
0.93729e+00, 0.86338e+00, 0.79687e+00, 0.69425e+00, 0.52386e+00];...
[0.98290e+00, 0.99078e+00, 0.99588e+00, 0.99800e+00, 0.99846e+00, 0.99613e+00, 0.98048e+00, ...
0.95360e+00, 0.88847e+00, 0.82600e+00, 0.72545e+00, 0.55208e+00];...
[0.98432e+00, 0.99294e+00, 0.99711e+00, 0.99868e+00, 0.99911e+00, 0.99782e+00, 0.98691e+00, ...
0.96565e+00, 0.90886e+00, 0.85075e+00, 0.75308e+00, 0.57798e+00];...
[0.98620e+00, 0.99462e+00, 0.99794e+00, 0.99910e+00, 0.99945e+00, 0.99875e+00, 0.99122e+00, ...
0.97456e+00, 0.92547e+00, 0.87186e+00, 0.77765e+00, 0.60186e+00];...
[0.98814e+00, 0.99590e+00, 0.99851e+00, 0.99936e+00, 0.99964e+00, 0.99928e+00, 0.99410e+00, ...
0.98115e+00, 0.93904e+00, 0.88992e+00, 0.79959e+00, 0.62400e+00];];
end
K= interp2(r_R, beta_ii, Xi, XRI, Lambda, 'linear ');
end

```

APPENDIX B.1

```

/*----- C++ -----*/
|=====|
| \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / O p e r a t i o n | Version: 4.1 |
| \ \ / A n d | Web: www.OpenFOAM.org |
| \ \ / M a n i p u l a t i o n |
/*-----*/

FoamFile
{
    version      2.0;
    format       ascii;
    class        polyBoundaryMesh;
    location     "constant/polyMesh";
    object       boundary;
}

// *****

17
(
    ami_r3
    {
        type            cyclicAMI;
        inGroups         1(cyclicAMI);
        nFaces           1072;
        startFace        31507623;
        matchTolerance   0.0001;
        transform         noOrdering;
        neighbourPatch    ami_s3;
    }
    ami_r1
    {
        type            cyclicAMI;
        inGroups         1(cyclicAMI);
        nFaces           5672;
        startFace        31508695;
        matchTolerance   0.0001;
        transform         noOrdering;
        neighbourPatch    ami_s1;
    }
    propellerBlade1
    {
        type            wall;
        inGroups         1(wall);
        nFaces           120311;
        startFace        31514367;
    }
    propellerBlade2
    {
        type            wall;
        inGroups         1(wall);
        nFaces           119010;
        startFace        31634678;
    }
    propellerBlade3
    {
        type            wall;
        inGroups         1(wall);
        nFaces           120915;
        startFace        31753688;
    }
    propellerBlade4
    {
        type            wall;
        inGroups         1(wall);
        nFaces           118963;
        startFace        31874603;
    }
    propellerBlade5
    {
        type            wall;
        inGroups         1(wall);
        nFaces           121431;
        startFace        31993566;
    }
    propellerBlade6
    {
        type            wall;
        inGroups         1(wall);
        nFaces           118894;
        startFace        32114997;
    }
    ami_r2
    {
        type            cyclicAMI;
        inGroups         1(cyclicAMI);
        nFaces           9174;
        startFace        32233891;
        matchTolerance   0.0001;
        transform         noOrdering;
        neighbourPatch    ami_s2;
    }
    propellerHub
    {
        type            wall;
        inGroups         1(wall);
    }

```

```

        nFaces      253056;
        startFace   32243065;
    }
    outlet
    {
        type          patch;
        nFaces        3388;
        startFace     32496121;
    }
    inlet
    {
        type          patch;
        nFaces        2424;
        startFace     32499509;
    }
    ami_s1
    {
        type          cyclicAMI;
        inGroups      1(cyclicAMI);
        nFaces        4664;
        startFace     32501933;
        matchTolerance 0.0001;
        transform      noOrdering;
        neighbourPatch ami_r1;
    }
    ami_s3
    {
        type          cyclicAMI;
        inGroups      1(cyclicAMI);
        nFaces        2900;
        startFace     32506597;
        matchTolerance 0.0001;
        transform      noOrdering;
        neighbourPatch ami_r3;
    }
    domain
    {
        type          symmetry;
        inGroups      1(symmetry);
        nFaces        1536;
        startFace     32509497;
    }
    propellerShaft
    {
        type          wall;
        inGroups      1(wall);
        nFaces        45600;
        startFace     32511033;
    }
    ami_s2
    {
        type          cyclicAMI;
        inGroups      1(cyclicAMI);
        nFaces        17264;
        startFace     32556633;
        matchTolerance 0.0001;
        transform      noOrdering;
        neighbourPatch ami_r2;
    }
}

// ***** //

```

APPENDIX B.2

```
/*----- C++ -----*\
|=====|
| \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / O p e r a t i o n | Version: dev |
| \ \ / A n d | Web: www.OpenFOAM.org |
| \ \ / M a n i p u l a t i o n |
\*-----*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "system";
    object       controlDict;
}
// *****

application      simpleFoam;

startFrom        startTime;

startTime        0;

stopAt           endTime;

endTime          2500;

deltaT           1;

writeControl      timeStep;

writeInterval     100;

purgeWrite       0;

writeFormat       ascii;

writePrecision    6;

writeCompression off;

timeFormat        general;

timePrecision     6;

runTimeModifiable true;

functions
{
    #include "forces"
    #includeFunc residuals
}

// *****
```

APPENDIX B.3

```

/*----- C++ -----*/
|=====|
| \\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \\ / O p e r a t i o n | Version: dev |
| \\ / A n d | Web: www.OpenFOAM.org |
| \\ / M a n i p u l a t i o n |
/*-----*/

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       fvSolution;
}

// *****

solvers
{
    p
    {
        solver      GAMG;
        tolerance    1e-06;
        relTol       0.001;
        smoother     GaussSeidel;
        nCellsInCoarsestLevel 20;
    }

    U
    {
        solver      smoothSolver;
        smoother     GaussSeidel;
        nSweeps      2;
        tolerance    1e-08;
        relTol       0.0;
    }

    k
    {
        solver      smoothSolver;
        smoother     GaussSeidel;
        nSweeps      2;
        tolerance    1e-08;
        relTol       0.0;
    }

    epsilon
    {
        solver      smoothSolver;
        smoother     GaussSeidel;
        nSweeps      2;
        tolerance    1e-08;
        relTol       0.0;
    }

    omega
    {
        solver      smoothSolver;
        smoother     GaussSeidel;
        nSweeps      2;
        tolerance    1e-08;
        relTol       0.0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 3;
    pRefCell 0;
    pRefValue 0;
    consistent yes;
}

relaxationFactors
{
    equations
    {
        U      0.9;
        k      0.7;
        omega  0.7;
        epsilon 0.7;
    }
}

cache
{
    grad(U);
}

// *****

```

APPENDIX B.4

```

/*----- C++ -----*\
|=====|
| \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / O p e r a t i o n | Version: dev |
| \ \ / A n d | Web: www.OpenFOAM.org |
| \ \ / M a n i p u l a t i o n | |
\*-----*/

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "system";
    object       fvSchemes;
}

// ***** //

ddtSchemes
{
    default      steadyState;
}

gradSchemes
{
    default      cellLimited leastSquares 1;
    grad(U)      cellLimited leastSquares 1;
}

divSchemes
{
    default      none;
    div(phi,U)   bounded Gauss linearUpwind grad(U);
    div(phi,k)   bounded Gauss upwind;
    div(phi,epsilon) bounded Gauss upwind;
    div(phi,omega) bounded Gauss upwind;
    div((nuEff*dev2(T(grad(U)))) Gauss linear;
}

laplacianSchemes
{
    default      Gauss linear limited corrected 0.5;
}

interpolationSchemes
{
    default      linear;
}

snGradSchemes
{
    default      limited corrected 0.5;
}

wallDist
{
    method meshWave;
}

// ***** //

```

APPENDIX B.5

```

/*----- C++ -----*\
|=====|
| \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / O p e r a t i o n | Version: dev |
| \ \ / A n d | Web: www.OpenFOAM.org |
| \ \ / M a n i p u l a t i o n | |
\*-----*/

forces
{
    type         forces;

    libs         ("libforces.so");

    writeControl timeStep;
    timeInterval 1;

    log          yes;

    patches      ("propellerBlade.*");
    rho          rhoInf; // Indicates incompressible
    log          true;
    rhoInf       1; // Redundant for incompressible

    CofR         (0 0 0); // Rotation around centre line of propeller
    pitchAxis    (1 0 0);
}

// ***** //

```

APPENDIX B.6

```
/*----- C++ -----*\
|=====|
| \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / O p e r a t i o n | Version: dev |
| \ \ / A n d | Web: www.OpenFOAM.org |
| \ \ / M a n i p u l a t i o n |
\*-----*/

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "constant";
    object       MRFProperties;
}
// ***** //

MRFI
{
    cellZone     region0;
    active       yes;

    // Fixed patches (by default they 'move' with the MRF zone)
    nonRotatingPatches (ami_s1 ami_r1 ami_s2 ami_r2 ami_s3 ami_r3);

    origin       (0 0 0);
    axis         (1 0 0);
    omega        62.831853072;
}

// ***** //
```

APPENDIX B.7

```
/*----- C++ -----*\
|=====|
| \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / O p e r a t i o n | Version: dev |
| \ \ / A n d | Web: www.OpenFOAM.org |
| \ \ / M a n i p u l a t i o n |
\*-----*/

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "constant";
    object       transportProperties;
}
// ***** //

transportModel  Newtonian;

nu              [0 2 -1 0 0 0 0] 8.889e-7;

// ***** //
```

APPENDIX B.8

```
/*----- C++ -----*\
|=====|
| \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / O p e r a t i o n | Version: dev |
| \ \ / A n d | Web: www.OpenFOAM.org |
| \ \ / M a n i p u l a t i o n |
\*-----*/

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "constant";
    object       turbulenceProperties;
}
// ***** //

simulationType  RAS;

RAS
{
    RASModel      kOmegaSST;

    turbulence     on;

    printCoeffs    on;
}

// ***** //
```



PHOTO

CURRICULUM VITAE

Name Surname: Kayhan Ülgen

Place and Date of Birth: Istanbul, 1990

E-Mail: ulgenk@itu.edu.tr

EDUCATION:

- **B.Sc.:** 2013, Yıldız Technical University, Faculty of Naval Architecture and Marine Engineering, Department of Naval Architecture and Marine Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2013-2016 CFD Analysis Engineer at Milper Propeller Technologies Inc.