ISRN: LUTMDN/TMHP-19/5436-SE
ISSN: 0282-1990

Meanline method for design and off-design turbine performance predictions

Joel Sjödin

Thesis for the degree of Master of Science in Engineering
Division of Thermal Power Engineering
Department of Energy Sciences
Faculty of Engineering | Lund University

Meanline method for design and off-design turbine performance predictions

Joel Sjödin

June 2019, Lund

This degree project for the degree of Master of Science in Engineering has been conducted at the Division of Thermal Power Engineering, Department of Energy Sciences, Faculty of Engineering, Lund University, and at GKN Aerospace.

Supervisor at the Division of Thermal Power Engineering was Professor Magnus Genrup.
Supervisor at GKN Aerospace was Dr. Pieter Groth.
Examiner at Lund University was Associate Professor Marcus Thern.

The project was carried out in cooperation with GKN Aerospace.

Thesis for the Degree of Master of Science in Engineering

ISRN: LUTMDN/TMHP-19/5436-SE

ISSN: 0282-1990

Division of Thermal Power Engineering

Department of Energy Sciences

Faculty of Engineering, Lund University

Box 118, 221 00 Lund

Sweden

www.energy.lth.se

Abstract

Early designs are often based on rough estimations and assumptions. Such is also the case when designing space turbines that power rocket engine fuel pumps which generally have obscure performance and size requirements. Often times, this leads to unique, single-use designs. This complicates and restricts the amount of possible testing. To aid in this early design process and to help predict turbine performance in various scenario, a one-dimensional meanline method program is developed in Python. It is able to perform calculations for various geometries and operating conditions by varying inlet conditions such as temperature, pressure, rotational speed and pressure ratios. The program itself makes use of general turbine knowledge with relevant literature in aerodynamic loss modeling, outlet angle calculations and pressure convergence. The accuracy of the program's output has been continuously validated against different types of turbines and shown adequate correlations to real-world performance data, down to half a percentage in relative error for certain cases. Case-specific tailoring to fit certain operating conditions and fast run times of approximately 5 seconds will provide the user with the possibility of performing turbine performance characteristics with ease.

Keywords: Meanline program, space turbine, loss model, turbine predictions, early stage design

Sammanfattning

Design i ett tidigt skede baseras ofta på förenklingar och antaganden. Denna princip gäller även design av rymdturbiner som driver raketmotorernas bränslepump, som ofta har väldigt hårda krav gällande storlek och prestanda. Detta leder till unika designer, som inte nödvändigtvis används mer än en gång. Detta näst intill omöjliggör och förhindrar tester av turbinerna. Syftandes till att underlätta denna designprocess och förutspå prestanda i diverse scenarion har en endimensionell mittlinjekod tagits fram i språket Python. Programmet kan utföra beräkningar för olika typer av turbingeometrier och driftpunkter genom att variera indata som temperatur, tryck, varvtal och tryckförhållande över en turbin. Detta görs genom att kombinera generell turbinteori med relevant forskning inom aerodynamisk förlustmodellering, vinkelberäkningar och tryckkonvergens. Programmets tillförlitlighet har utvärderats kontinuerligt under utvecklandets gång och påvisat god anknytning till verklig testdata, ner till en halv procents fel i optimala fall. Driftpunktsspecifik skräddning av programmet och snabba lösningstider möjliggör det för användaren att köra storskaliga studier av turbinkarakteristik.

Nyckelord: Mittlinjekod, rymdturbin, förlustmodellering, turbinprestanda, förutsägelse, tidig design

Acknowledgements

This section is dedicated to the people and institutions that have made the attempt and finalization of this project possible.

I would like to start by thanking my former professor and current project supervisor, Magnus Genrup. For realizing me as a potential candidate for this thesis, assisting me in my choices of a master thesis and for your invaluable guidance during the execution of this work. Without your support, this thesis would not be half of what it is today.

Thank you to Pieter Groth of GKN for having me during my one month stay in Trollhättan as part of the intensive validation phase of the thesis. Your energy and guidance made my experience a lot better than I ever imagined it would be.

I would also like to thank my family - my mother for always having faith in me and believing I was destined for an academic life, and my father for always encouraging me. Last but not least, my lovely girlfriend Love, who has supported me since day one of this project, during the highs and during the lows.

Nomenclature

Following the format of: unit - description - SI unit.

Roman letters

A	Area	m^2
a	Speed of sound	[m s^{-1}]
C	Absolute velocity	[m s^{-1}]
c_p	Specific heat capacity	[kJ kg^{-1} K^{-1}]
CFM	Supersonic drag rise coefficient	[K]
clr	Blade tip clearance	[m]
h	Enthalpy (static)	[J kg^{-1}]
I	Rothalpy	[J kg^{-1}]
M	Mach number	[-]
$\dot{m}$	Mass flow	m^2
N	RPM	[1 min^{-1}]
P	Pressure (static)	[kg m s^{-2}]
R	Gas constant	[kJ kg^{-1} K^{-1}]
r	Radius	[m]
s	Entropy	[J K^{-1}]
T	Temperature (static)	[K]
U	Blade speed	[m s^{-1}]
W	Relative velocity	[m s^{-1}]
Y	Pressure loss coefficient	[-]
b	Blade axial chord	[m]
c	Blade chord	[m]

Greek letters

α	Absolute flow angle, axial/tangential	[°]
β	Relative flow angle, axial/tangential	[°]
η	Efficiency	[–]

γ	Heat capacity ratio	[−]
λ	Laval number	[−]
ω	Rotational speed	[rad s^{-1}]
Φ	Loss component	[−]
ϕ	Streamline flow angle, meriodional/axial	[°]
ρ	Density	[kg m^{-3}]

Abbreviations

AM	Ainley & Mathieson	[−]
AR	Aspect ratio	[−]
DC	Dunham & Came	[−]
GT	Gas turbine	[−]
KO	Kacker & Okapuu	[−]
LE	Leading edge	[−]
MK	Moustapha & Kacker	[−]
PR	Pressure ratio	[−]
RLX	Relaxation factor	[−]
TE	Trailing edge	[−]

Subscripts

0	Total or stagnation property	[−]
θ	Tangential flow component	[−]
a	Axial flow component	[−]
J	TE position (Denton's model)	[−]
R	Rotor	[−]
r	Radial flow component	[−]
S	Stator	[−]
1	Stator inlet, inlet	[−]
2	Stator outlet before mixing, outlet	[−]
2gap	Rotor inlet	[−]

2mix	Stator outlet after mixing	[–]
3	Rotor outlet before mixing, outlet	[–]
3gap	Next stage stator inlet	[–]
3mix	Rotor outlet after mixing, outlet	[–]
E	Euler radius position	[–]
hub	Hub radius position	[–]
rel	Relative property	[–]
th	Throat	[–]
tip	Tip radius position	[–]

Contents

Abstract i
Sammanfattning ii
Acknowledgements iii
Nomenclature iv

1 Introduction **1**
1.1 Background 1
1.2 Aims and objectives 1
1.3 Literature study 2
1.4 Constraints and limitations 3
1.4.1 Availability of literature and intellectual property 3
1.5 Requirements 3

2 Theory **4**
2.1 The gas turbine 4
2.1.1 Turbine blade rows 5
2.1.2 Stage-wise calculations 5
2.1.3 Row-wise calculations 6
2.1.4 Gas turbine operation 6
2.1.5 Turbine coordinate system 6
2.2 Turbine thermodynamics 8
2.2.1 Enthalpy, temperature, pressure and entropy 8
2.2.2 Constant enthalpy and rothalpy 11
2.2.3 Thermodynamic loss correlations 11
2.3 Fluid calculations 12
2.3.1 CoolProp 12
2.3.2 Miscellaneous fluid relations 12
2.3.3 Radial equilbrium 12
2.4 Geometry 13
2.4.1 Program meanline radius 13
2.4.2 Meriodional slope 13
2.4.3 Turbine blade nomenclature and calculations 14
2.5 Blade angle modeling 15
2.5.1 Mamaev's angle out model 15
2.5.2 Traupel's angle correction 18
2.5.3 Mixing and gap calculations 19
2.6 Flow loss modeling 21
2.6.1 Profile losses 21
2.6.2 Secondary losses 22
2.6.3 Tip clearance losses 23
2.6.4 Modified loss model 24
2.7 Pressure-finding algorithms 24
2.7.1 Initial pressure guess 25

2.7.2 Denton's target pressure method . . . 25
2.7.3 Came's brute-force method . . . 26
2.8 Iterative solvers . . . 26
2.8.1 Newton Raphson's method . . . 26

3 Methodology **27**
3.1 Structure . . . 28
3.2 Geometry data . . . 29
3.2.1 Data parsing . . . 29
3.2.2 Subsequent geometry-related calculations . . . 29
3.3 Angle calculations . . . 30
3.3.1 Blade outlet angle . . . 30
3.3.2 Velocity vectors module . . . 31
3.3.3 Meriodional flow angle ϕ . . . 31
3.4 Loss model - AMDCKO et. al. . . . 31
3.5 Turbine stage calculations . . . 32
3.5.1 Initialization . . . 32
3.5.2 Station nomenclature . . . 32
3.5.3 General methodology . . . 33
3.5.4 Turbine inlet . . . 34
3.5.5 Entropy iteration . . . 35
3.5.6 Mix calculations . . . 37
3.5.7 Gap calculations . . . 38
3.6 Main routine . . . 39
3.6.1 Running a single case . . . 39
3.6.2 Pressure solver implementation - Denton's method . . . 40
3.6.3 Came's brute force-style solution . . . 41
3.6.4 Convergence . . . 42
3.7 Mass testing . . . 42
3.8 Post-processing . . . 42
3.8.1 Relative errors . . . 43
3.8.2 Plotting . . . 43
3.9 Other routines . . . 43
3.9.1 Pressure distribution . . . 43
3.9.2 Velocity triangle drawer . . . 43

4 Results and discussion **44**
4.1 Turbine test data . . . 44
4.2 Phase 1 - Subroutine validation . . . 44
4.2.1 Blade outlet angle model validation . . . 44
4.2.2 Aerodynamic losses validation . . . 45
4.3 Phase 1 - Complete program validation . . . 47
4.3.1 Turbine A validation . . . 47
4.3.2 Target pressure validation . . . 48
4.4 Phase 2 - Large scale testing . . . 50

4.5 Phase 2 - Turbine B testing 50
4.5.1 Turbine B - Air flow, untweaked 51
4.5.2 Turbine B - Parahydrogen flow, untweaked 53
4.5.3 Turbine B - Parahydrogen flow, Traupel correction . . . 54
4.5.4 Turbine B - Parahydrogen flow, gauge modification . . . 55
4.5.5 Turbine B - Parahydrogen flow, Traupel and gauge modification 56
4.5.6 Turbine B - Closing discussion 57
4.6 Phase 2 - Turbine C testing 58
4.6.1 Turbine C - Mass flow anomaly 59
4.6.2 Turbine C - Parahydrogen flow, untweaked 60
4.6.3 Turbine C - Parahydrogen flow, Traupel correction . . . 61
4.6.4 Turbine C - Parahydrogen flow, gauge modification . . . 62
4.6.5 Turbine C - Parahydrogen flow, Traupel and gauge modification 62
4.6.6 Turbine C - Closing discussion 64
4.7 Sources of errors 65
4.7.1 Meanline methodology 65
4.7.2 Input data 65
4.7.3 Sensor and measurement accuracy 65
4.7.4 Model accuracy 65
4.7.5 Total-to-total pressure ratio 66
4.7.6 Iteration criteria 66
4.7.7 Axial force 66
4.7.8 Programming mishap 66
4.8 Future work possibilities 67
4.8.1 Continued validation 67
4.8.2 Characteristics 67
4.8.3 Varying gauge modification and usage of Traupel 67
4.8.4 Code modularization and speed ups 67

5 Conclusion **69**

Appendices **72**

A Geometry indata **72**

1 Introduction

The idea of harvesting energy and using it through various means has driven the development of the human race since the dawn of modern time. A remarkable way of doing this has been the construction of machinery known as turbomachines; allowing a process of converting the kinetic energy of a fluid to and from a rotary device resulting in useful work. The colossal field of application for these kinds of mechanical systems has kept the technology relevant, particularly in the power production industry and aerospace industry, the latter in which GKN Aerospace is a prominent figure. This thesis is conducted in collaboration with GKN Aerospace.

1.1 Background

GKN Aerospace designs and manufactures components related to the aviation and space industry. As of today, they supply approximately 90 % of the world's aircraft and engine manufacturers [1]. The designs of these components are very complex and depend on many parameters including, but not limited to, material stability in terms of withstanding high temperatures, pressures and vibrations, meeting exhaust regulations, maintaining size requirements and interacting properly with other system components. This is of particular interest at early stages of designs or when seeking to identify system *characteristics*. To continue understanding how these criteria interact, one may build virtual or computational models of these systems.

In simple terms, a rocket engine produces thrust by ignition and expulsion of a propellant, usually liquid hydrogen, oxygen or kerosene [2]. This propellant, or fuel, is fed into the combustion chamber by a pump, which in turn is driven by a gas turbine (GT). In other words, the gas turbine is an enormously important system. In reality, the mechanics behind how this system works is a lot more complex for the reasons stated earlier and as such there has emerged a demand for computational programs to be used for simulation purposes. In addition, projects involving the designing, manufacturing, building and testing of flying engines are extremely costly and time consuming, further raising the need for these simulative tools.

1.2 Aims and objectives

The thesis project has a main purpose, namely to build such a tool to allow for thermodynamic computations to be performed. The main focus of this thesis and program will be the gas turbine expansion and its inherent processes. The intent is to deliver a program able to

1. Perform calculations to allow for performance predictions for given scenarios in the earlier stages of a design process, and

2. Deliver system characteristics and map operating points.

The last point pertains to issues related to off-design calculations to gain more insight into how varying external parameters such as temperature, rotational speed and pressure may affect the performance. It is also intended to be able to be modified and worked upon further if needed. The program should deliver these qualities with modest accuracy and reliability and will be tested against real-world data for validative purposes.

This will be done by constructing a program in the programming language Python. Assistance in the form of relevant literature studies (on turbine flow mechanics), supervisors at LTH and GKN as well as previous works will also be utilized to gain more insight into how coding a computational turbine program works. Naturally, knowledge from previous engineering courses both on turbomachinery, thermodynamics, computational fluid dynamics and other related subjects will come in handy as part of the engineering toolbox.

The contents of the thesis assumes that the reader has a decent understanding of thermodynamic systems and how they work with some prior experience within turbomachinery. As such, the most basic concepts within these fields will not be fully explained and left to the reader to realize. However, the program does feature a great amount of theory which will be described in depth later on.

1.3 Literature study

The aforementioned theory lays the very foundation of the program. Thus, the results obtained will be heavily dependent on included theory and, if present, modifications of said theory. Rudimentary concepts by Saravanamuttoo et. al. [3] describes general turbine workings and the theory found therein is used. The thesis will make use of a very well studied meanline loss model originally formulated by Ainley and Mathieson [4] in 1951. Improvements were published by Dunham and Came in 1970 followed by Kacker and Okapuu in 1982 and by Moustapha et. al. [5] in 1990. The combination of the authors' names is also the identifying name in this report, namely AMDCKOMK.

A solution method for obtaining flow angles by Mamaev [6] was also studied. Denton's [7] work on turbine convergence models is also highly relevant for the subject of this master thesis.

Equation of state calculations are primarily handled by the Python package CoolProp [8] using HEOS state tables. As supplement to the above referenced turbomachinery authors, the thesis will include extracts from other sources. However, since the intended use for this meanline code is very niche and classified space turbines, some sources can not be referenced due to confidentiality.

1.4 Constraints and limitations

Single-handedly building a complex program that is able to account for 3D (three-dimensional) analyses with many interacting systems is undoubtedly impossible given the time frame for a Master's thesis. However, a quasi-2D program governing one of these systems, in this case the turbine in a gas turbine, with quick run-times to provide a general overview of how varied conditions may change the results is entirely within the scope of this thesis.

In order to achieve sufficiently fast run times during this thesis period, limitations must be consciously imposed. While it would be possible to implement models that could supersede these limitations, it is important to realize that they might not even exist, be available or simply be far too complicated for implementation in a 1-D program - which brings us to the main limitation of quasi-2-dimensionality.

The program will principally perform calculations pertaining to flows in the axial direction at a single radius. Granted, radial and tangential changes (stemming from the turbine geometry due to its conical shape and due to rotation) will affect the flow considerably. For this reason, various models are implemented to account for these. Previous programs that have been studied has had a limitation on choked flows but the introduction of an extensive angle model has taken care of this. In addition the program uses a so-called *meanline* calculation philosophy and thus "ignores" what happens at the hub and tip. Once again, models are used in place of this to try to account for what happens at these locations without performing throughflow calculations at any locations other than the meanline radius.

1.4.1 Availability of literature and intellectual property

The field of turbine studies is an incredibly wide area with historical interest from the academic, industrial and military world. The reason for this is its enormous applicability range from power generation, to aviation, to military defence systems. Taking into account the complexity of these systems and time taken to develop said systems tells the tale of an engineering territory where the many of the recent discoveries might not be academically published. These thoughts are also shared by Moustapha et. al. [5]. With that said, there is definitely a great amount of work that is available and is used in this thesis.

1.5 Requirements

The meanline code is written in Python 3 which requires a compiler able to handle this. Spyder [9] from the Anaconda environment is a suitable choice. Installation of the module CoolProp is also expected. The CoolProp module also has a built functionality for integration of REFPROP [10] which is another state program.

2 Theory

In this section, the underlying theory of the program's contents will be outlined. Firstly, the general workings of a gas turbine will be described including thermodynamics, followed by descriptions of ways to characterize one dimensional flows. For instance how to recognize turbine geometry and perform necessary calculations, which flow loss models may be used and what kind of blade row angle calculations are needed to obtain results. In essence, the program will work by obtaining information in the form of geometry data and boundary conditions about a particular axial gas turbine and its operating state, and then iteratively calculate various properties such as pressures, enthaplies, entropies continuously in the turbine until convergence criteria is met. The actual methodology for these calculations are introduced in the methodology section.

2.1 The gas turbine

An axial gas turbine is normally comprised of two or three major sets of pieces - a compressor whose purpose is to compress any entering gas to a much higher pressure followed by a turbine with the main purpose of extracting work by expansion of said fluid. Between these, a combusting unit is placed to assist in increasing the temperature prior to work extraction in the expansion turbine if needed [11]. This extraction of work can be either torque production and thus, feeding into generators or driving other auxilliary systems such as pumps. It may also produce thrust; allowing for rapid acceleration and flight.

The two subsystems (compressor and turbine) may seem different from a functional perspective but they share many details, such as the usage of stator rows (commonly called *vanes* or *nozzles*) and rotor rows. As the naming implies, the stator rows are stationary blades whereas the rotor rows consist of blades rotating around its axis at very high revolution speeds. This introduces relative flow components which will be examined later. Figure 1 shows a detailed overview of an axial gas turbine.

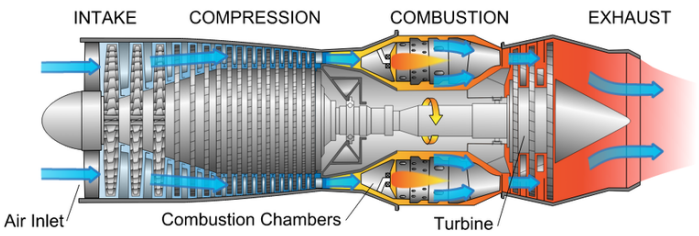


Figure 1: Axial gas turbine schematics. Courtesy of Wikimedia Commons [Online].

2.1.1 Turbine blade rows

The expanding turbine part of the gas turbine, henceforth called just *turbine*, consists of one or more row pairs of stators and rotors (one of which composes a *stage*) continually expanding the fluid that is flowing through, resulting in pressure and temperature drops as the turbine extracts energy. The stator vanes act to appropriately direct the fluid for optimal angle entry into the rotor blades - which is where the actual work transfer is happening from fluid to metal. This is also the fundamental difference to the compressor, where the rotor blades are transferring energy **to** the fluid and thus compressing it instead of absorbing the energy.

The turbine rows consist of evenly positioned blades with a specific shape, along the circumference of an axial gas turbine's axis. A blade has an entry point called the leading edge (LE) and an exit point denoted trailing edge (TE). The blade shape is not constant along the height of the blade in the radial direction due to the blades experiencing increasing tip speeds even though the rotational speed is constant. This can be seen from equation (2.1):

$$U = \omega r = \frac{2\pi \cdot N}{60} r \tag{2.1}$$

Where U is the blade speed $[\frac{m}{s}]$, ω measures radians per second, blade radius $r[m]$ and rotational speed N in rounds per minute. An increase in radius yields a higher blade speed as N is constant. As this is a meanline program, information about a blade will be gathered at a chosen radius as will be outlined later on.

For a general turbine, the stator-rotor pattern is quickly realized and one can start working on ways to systematically schematize the flow as it makes its way axially through the blade rows. In this report, two major methods will be introduced and the benefits and drawbacks of each of them will be discussed. The two methods that allow for easy and continuous calculations and mapping of a turbine are *stage-wise* and *row-wise*. Following the naming above, the stage-wise division handles a stator and rotor pair at the same time whereas the latter works on a blade-to-blade basis.

2.1.2 Stage-wise calculations

The benefits of having a stage-based structure will be further outlined later in this report but generally allows for a better overview of a stage. If more than one stage is present, this division increases readability and allows for straightforward comparisons between stages.

2.1.3 Row-wise calculations

Although the previous method is possibly better for readability, row-wise calculations allow for simple data introduction. Positions that are interesting in this thesis are at entry (LE) and outlet (TE) of any blade or *row*. The row structure may be used for the initial calculations that are related to geometry and then converted to stage-wise when stage calculations are being performed and a more generalized stage-oriented model is needed. Ultimately the choice of row structure has no impact on the results since the two methods are nothing but different ways of expressing the same information. The thesis will make use of both methods, each where appropriate.

2.1.4 Gas turbine operation

A gas turbine normally operates with a set design rotational speed for optimal efficiency. The pressure ratio describing the proportion of turbine inlet pressure to outlet pressure is another system-defining characteristic. Theoretically, this expansion ratio could be close to 1 for more niche uses but could also exceed 100 for large industrial machines, should the materials be able to handle it.

Since a turbine expands its working medium, a fluid, and extracts work, the thermodynamic equivalent of this is a loss of pressure and temperature until it reaches its outlet value. With good design, the axial pressure distribution loads each blade row evenly. Uneven loading for standard operating conditions would require stronger blades and could render a project economically unfeasible. In other cases or for off-design cases, the last blade of the turbine will experience a substantial pressure differential. It is also important to know that a pressure difference drives flows due to its innate diffusive properties.

2.1.5 Turbine coordinate system

All turbine calculations are based on a spherical coordinate system consisting of the three base axes **axial** with subscript a, **radial** with subscript r and **tangential** with subscript θ. Additional layers of calculations are needed for *relative* components as a gas turbine is spinning around its own axis while alternating static rows with rotating ones. If the flow hits a rotating blade as opposed to a stator, there will be a perceived difference in speed and equally important, difference in direction. In figure 2 this is shown by the blade colors - the red shows the flow from a stationary frame whereas the blue shows the flow from a moving reference point.

The image also introduces new nomenclature: a way of distinguishing the relative velocities and angles from the stationary ones. The letter W is henceforth used to describe the relative velocity of the flow and C is used for the

absolute equivalent. Their corresponding angles may also be defined by

$$\beta = \arctan \frac{W_\theta}{W_a} \tag{2.2}$$

$$\alpha = \arctan \frac{C_\theta}{C_a} \tag{2.3}$$

Generally the axial velocity may be assumed equal between the two viewpoints and the difference in tangential velocity is given by the prescribed blade speed:

$$W_\theta = C_\theta - U \tag{2.4}$$

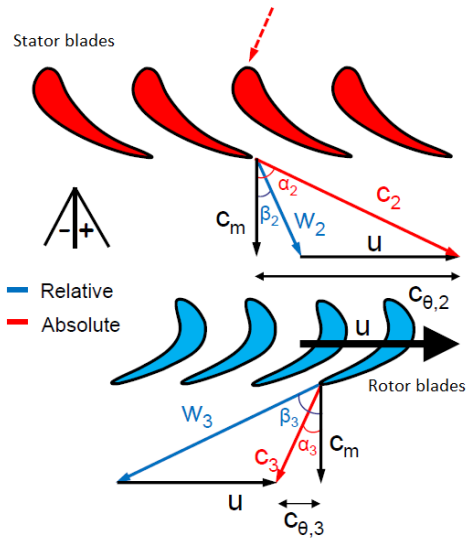


Figure 2: A single stage showing absolute and relative flow components in the axial-tangential plane.

The program also needs a way to downscale from complex 3D flows. From the geometrical considerations of figure 3, it is seen that the following relations can be used to express a 3D velocity in each of its three base dimensions θ, a and r. In these sets of equations, α_{2D} is the two dimensional absolute flow angle.

$$C_\theta = \frac{C_{3D} \cos\phi}{\sqrt{\cos^2\phi \sin^2\alpha_{2D} + \cos^2\alpha_{2D}}} \cdot \sin\alpha_{2D} \tag{2.5}$$

$$C_a = \frac{C_{3D} \cos\phi}{\sqrt{\cos^2\phi \sin^2\alpha_{2D} + \cos^2\alpha_{2D}}} \cdot \cos\alpha_{2D} \tag{2.6}$$

$$C_r = C_a \tan\phi \tag{2.7}$$

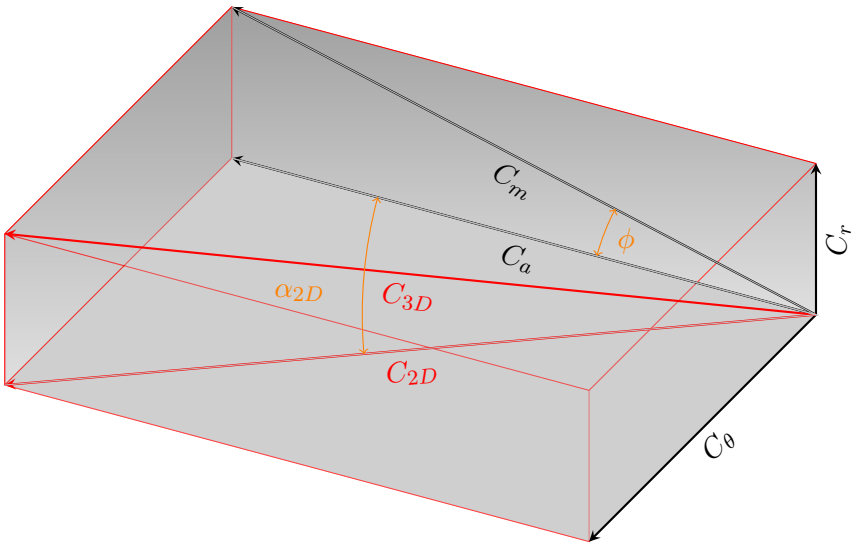


Figure 3: 3D flow vector projected on the tangential-axial plane as C_{2D}.

When a fully defined equation system of angles and velocities is known, vector sum algebra may be used to find the value of the 3D flow angle:

$$C = \sqrt{C_a^2 + C_r^2 + C_\theta^2} \tag{2.8}$$

The same can be done for the relative counterparts.

2.2 Turbine thermodynamics

The very foundation of turbomachinery is rooted deeply within the field of thermodynamics. The thesis relies on and works with calculations relating to the interactions between the **enthalpy** h $[\frac{J}{kg}]$, **entropy s** $[\frac{J}{kgK}]$, **temperature T** $[K]$ and **pressure p** $[Pa = \frac{kg \cdot m}{s^2}]$ of a system which are fundamental properties of thermodynamics.

For an ideal Brayton cycle, the first law of thermodynamics which states that the total energy of a system is conserved, gives the work transferred during an ideal expansion through a turbine (assuming constant pressure, no losses and no change of velocity)

$$W_{turbine} = (h_{in} - h_{out}) = c_p(T_{in} - T_{out}) \tag{2.9}$$

as work transfer per unit mass flow. In reality, flows are not ideal (i.e. not without losses) and therefore one must understand the influence and synergy of the above mentioned properties.

2.2.1 Enthalpy, temperature, pressure and entropy

Pressure and temperature are two physical properties that describe the state of a fluid. They both share the same diffusive properties in that they both

seek to reach equilibrium innately by following a gradient from a place of high concentration to low. However, this is not enough to describe turbine systems and their properties, therefore two additional main concepts need to be introduced: entropy and enthalpy.
Enthalpy is a state variable of a gas. It is defined [12] as the internal energy of a system plus pressure times volume. Divided by the mass, this gives *specific* enthalpy:

$$h = e + pv \tag{2.10}$$

Equation (2.10) shows that a variation of pressure will lead to a change in enthalpy. Equation (2.9) also showed that enthalpy (and consequently pressure) is directly related to temperature.
The introduction of the entropy of a system becomes relevant when analyzing losses and non-ideal processes. Consider once again a Brayton cycle expansion depicted by the enthalpy-entropy (h-s) diagram in figure 4. A vertical process represents an *isentropic process*, a process without losses or process with constant entropy. The tilted dotted line next to it shows the non-ideal equivalent of the same process.

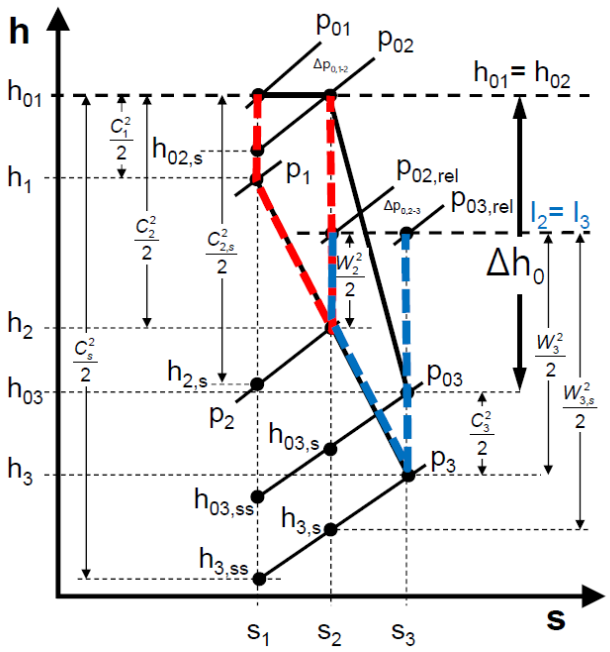


Figure 4: Enthalpy-entropy (h-s) diagram showing single stage turbine expansion (inlet to outlet): From state 1>2>3. Image by Magnus Genrup.

Change of specific entropy is defined as the change of heat flow divided by the temperature:

$$\delta s = \frac{\delta q}{T} \tag{2.11}$$

The following relation is always true for all thermodynamic processes:

$$s_{final} \geq s_{start} \tag{2.12}$$

The two above are only ever equal if a process is said to be completely ideal. In turbines, these processes are not and those cases will be analyzed. With the addition of entropy and figure 4, the connections between the 4 properties are once again shown.
Another important concept is that of a variable's stagnation and static property. Stagnation enthalpy is the enthalpy a flow system would have at a given instance if the velocity of the system was halted and the fluid brought to rest. In equation-form, it is expressed in specific form through

$$h_0 = h_s + \tfrac{1}{2}C^2 \tag{2.13}$$

with h_0 being the *stagnation* or *total* enthalpy, h_s or simply h the static counterpart and C the absolute velocity. These would be represented at different parts in a T-s diagram but along the same vertical axis - meaning that *both* h_0 and h have the same entropy value. Similarly a relative enthalpy may be formulated, also sharing the uniform entropy quality:

$$h_{0,rel} = h + \tfrac{1}{2}W^2 \tag{2.14}$$

The same can be shown for the other properties. This is a central notion in the program and will be used thoroughly. If two state variables are known, the others can be found, for example the entropy value at turbine inlet as a function of the specified inlet total temperature and total pressure: $s_{in} = f(T0_{in}, P0_{in})$. Similarly, using either total or relative pressure and entropy a value for the total or relative enthalpy can be found:

$$h_0 = f(p_0, s) \qquad \text{Total enthalpy from total pressure} \tag{2.15}$$

$$h_s = f(p_s, s) \qquad \text{Static enthalpy from static pressure} \tag{2.16}$$

In the event of unknown values, i.e. when only the pressure is known after blade passage the program must work with a guess or estimation of the increase in entropy $h_2 = f(p_2, s_1 + ds)$. Conceptually this is synonymous with assuming losses exist in the system.
The following relations for isentropic expansions are also useful when static properties or velocities are unknown:

$$\frac{T_{01}}{T_1} = (1 + \frac{\gamma - 1}{2}M^2) \tag{2.17}$$

$$\frac{P_{01}}{P_1} = \frac{T_{01}}{T_1}^{(\gamma-1)/\gamma} \tag{2.18}$$

2.2.2 Constant enthalpy and rothalpy

The information above is not enough to solve for a turbine flow. Two new concepts are introduced: constant enthalpy and rothalpy.

The stator's main objective is to redirect fluid. As it does not move, it does not allow for energy transfer. The stator is as a result said to have the same total enthalpy both at outlet and prior to inlet. This is a useful definition to include in the calculations.

$$h_{0,stator,in} = h_{0,stator,out} \tag{2.19}$$

The rotor equivalent of this is the concept of constant rothalpy I defined below:

$$I = h + \tfrac{1}{2}\Big(W^2 - U^2\Big) \tag{2.20}$$

The value of the rothalpy is constant during rotor passage.

2.2.3 Thermodynamic loss correlations

Suppose a stage has positions 1, 2 and 3 denoting stator inlet, stator outlet and rotor outlet respectively. From the work of Saravanamuttoo et. al. [3], a pressure loss coefficient is defined for stator (suffix N) and rotor (suffix R) passage:

$$Y_N = \frac{P_{01} - P_{02}}{P_{02} - P_2} \tag{2.21}$$

$$Y_R = \frac{P_{02,rel} - P_{03,rel}}{P_{03,rel} - P_3} \tag{2.22}$$

where P_{01} and P_2 denote total pressure at stator inlet and static pressure at stator outlet, respectively. In a similar fashion, $P_{02,rel}$ and P_3 denote the relative total pressure at rotor inlet and static pressure at rotor outlet.
Using the relations above and introducing the Gibbs free energy concept:

$$dU = TdS - pdV \tag{2.23}$$

Through algebraic manipulations a correlation for the change of entropy may be formulated as, with subscript 1 as inlet and 2 for outlet:

$$s_2 - s_1 = ds = -R \cdot \log \frac{P_2}{P_1} \tag{2.24}$$

Using equations (2.21) and its rotor equivalent (2.22) gives

$$ds_S = -R \cdot \log\Big(1 - Y_N \cdot \frac{P_{02} - P_2}{P_{01}}\Big) \tag{2.25}$$

$$ds_R = -R \cdot \log\Big(1 - Y_R \cdot \frac{P_{03,rel} - P_3}{P_{02,rel}}\Big) \tag{2.26}$$

2.3 Fluid calculations

A turbine with a liquid hydrogen flow will behave differently when compared to the same turbine with a flow consisting of air. This is due to differing fluid properties. As such, there emerges a need for a module to handle such fluid calculations since they are heavily dependant on changes of and interactions between temperature, pressure, enthalpy and entropy. CoolProp can return information if two state variables are input, such as in equations (2.15) and (2.16).

2.3.1 CoolProp

CoolProp is an open source library that has been developed to perform such fluid property calculations [8]. The CoolProp library and its database is fully accessible within the program and contains a large amount of substances. The module also governs each fluid's equations of state, facilitating calculations.

2.3.2 Miscellaneous fluid relations

Through CoolProp, information may be gathered about a particular fluid. For example, specific heat capacities c_P for constant pressure conditions and c_V for constant volume may be accessed. Through these, two fluid relations may be derived, namely the gas constant R and heat capacity ratio γ:

$$R = c_P - c_V \tag{2.27}$$

$$\gamma = \frac{c_P}{c_V} \tag{2.28}$$

The Reynolds number is also a useful parameter to characterize flow patterns.

$$Re = \frac{CL}{\nu} \tag{2.29}$$

L is the characteristic length which in turbines translates to diameter, C the velocity and ν the kinematic viscosity - another parameter which is accessible from CoolProp.

2.3.3 Radial equilbrium

Even though this is a primarily one dimensional analysis tool, it is of great interest to be able to find out information about hub and tip positions in order to establish performance and stress endurance parameters for the entire turbine. Dejc and Trojanovskij [13] derived a model that can be used for this purpose. The velocity C at a radius r can be modelled through the following equation:

$$C(r) = \left[\left(\frac{r_E}{r} \right)^{2Q_1(1-\zeta)\cos\alpha - \dfrac{2.5Q_2 \sin(2\alpha) r_E \sin\phi}{b_x}} \cdot C_E^2 \right]^{\frac{1}{2}} \tag{2.30}$$

In the equation above, $Q_1 = 0.94$ and $Q_2 = 0.97$ are empirically derived values and all other values are known from meanline calculations. The velocity may then be used to find the static enthalpy at either tip or hub locations when assuming a constant total enthalpy. Following this, the static pressure can be found through CoolProp.

2.4 Geometry

In order to accurately calculate results, it is of critical importance to have a correct setup. In this context this includes geometric parameters, fluid parameters, any assumptions made and initialization values (also known as *boundary conditions*). It is equally important to realize how variations of these setups may change the results obtained and the validity of these new results. More information about this in the sensitivity analysis section.

2.4.1 Program meanline radius

This program will utilize meanline calculations meaning that one single streamline through the turbine is chosen to represent the entire flow. Therefore, there is a need to accurately present any position along this theoretical streamline based on geometric parameters. A way of doing this is using the so-called *Euler radius,* r_E which splits an annulus area into two equally sized annuli. This is defined as follows:

$$\begin{aligned} A &= \pi(r_o^2 - r_E^2) = \pi(r_E^2 - r_i^2) \Rightarrow \\ r_E &= \sqrt{\frac{r_o^2 + r_i^2}{2}} \end{aligned} \tag{2.31}$$

Where r_o and r_i are the outer and inner radius of any given annulus, respectively. The point of this is to try to balance out impacts from both the hub and tip along the center and achieve a meanline methodology. Unless stated otherwise, subsequent calculations use the euler radius.

2.4.2 Meriodional slope

Generally turbines have increasing radii in the axial direction to aid expansion. This introduces a radial flow angle ϕ. Bitterlich [14] suggests a way of estimating this as

$$\tan\phi_2 \approx \frac{r_{1_{i+1}} - r_1}{b_{rotor_{i+1}} + b_{stator}} \cdot 0.8 \tag{2.32}$$

for a station 2 between a stator and rotor. Similar expressions exist for other positions.

2.4.3 Turbine blade nomenclature and calculations

The shape of a turbine blade plays a big role in deciding the flow characteristics of a turbine. Normally stator and rotor blade designs vary due to having different duties. Figure 5 shows a detailed image with the nomenclature of a turbine blade.

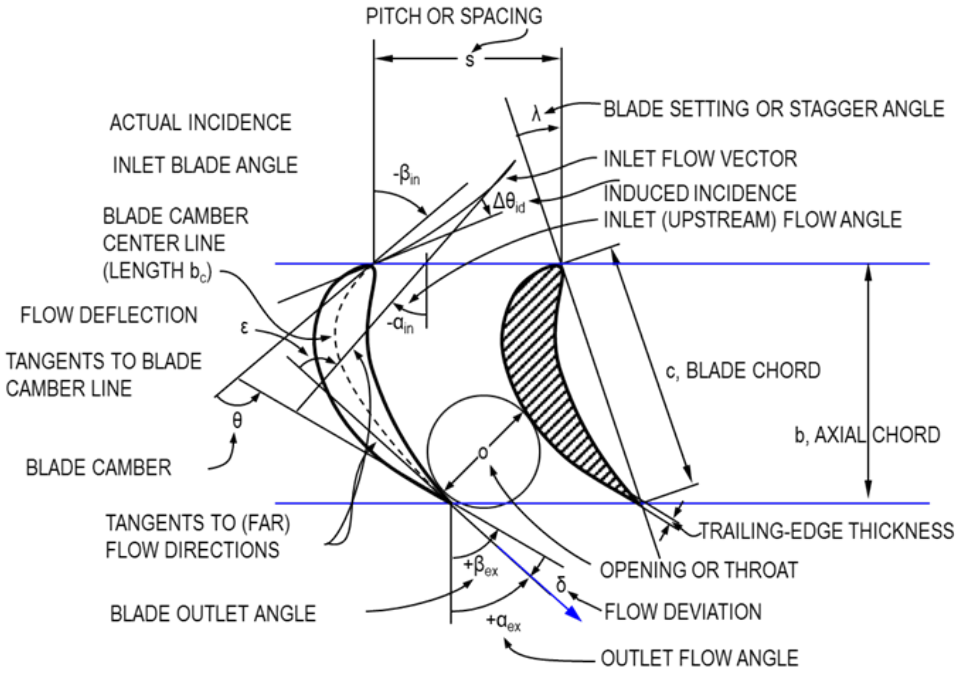


Figure 5: Blade nomenclature. Image by Marcus Thern.

By introducing radii at various positions along with other parameters such as angles and chord information one may fully define the geometry of the turbine. The axial chord b_x of a blade is known either from supplied axial positions or the actual chord value c and the stagger angle.

$$b_x = \cos(stagger) \cdot c \tag{2.33}$$

An average blade height may be calculated after knowing hub and tip radii:

$$h = \frac{r_{tip,LE} - r_{hub,LE} + r_{tip,TE} - r_{hub,TE}}{2} \tag{2.34}$$

Using the same data the channel areas can be produced

$$A_{LE} = \pi(r_{tip,LE}^2 - r_{hub,LE}^2) \tag{2.35}$$

$$A_{TE} = \pi(r_{tip,TE}^2 - r_{hub,TE}^2) \tag{2.36}$$

By estimation of Ainley and Mathieson [4] the limiting flow channel area or throat area can be expressed as

$$A_{TH} = \frac{O}{s} \cdot \frac{5 \cdot A_{TE} + A_{LE}}{6} \tag{2.37}$$

In equation (2.37) above, O is the throat distance and s the spacing or pitch, calculated as:

$$s = \frac{2\pi r_E}{\text{amount of blades}} \tag{2.38}$$

Another interesting blade parameter is the aspect ratio found from the proportion of blade height to chord

$$AR = \frac{h}{c} \tag{2.39}$$

2.5 Blade angle modeling

The beginning of this report mentioned the usage of *mathematical models* to deal with various flow phenomena. Two of the more sophisticated ones that are integrated into this program are:

1. A comprehensive blade outlet angle model (Mamaev), and
2. An extensive loss module (AMDCKOMK)

Within each of these two, deeper layers of models are found based on either entirely on derivations or empirical data. Countless other smaller models and relations to aid calculations are also used, most of which will be detailed.

2.5.1 Mamaev's angle out model

Even though the program is structured to calculate mostly axially in the mean-line, it is realized that multi-dimensional effects may and will affect the flow and its many important qualities. In this case, the variation of blade outlet angle. As such, various models will be used to account for these effects. The equations below are published by B.I. Mamaev [6] and will handle calculations pertaining to the flow and the angle out of which the flow leaves each blade row, in this section henceforth called β_2. These theoretically derivated relations are implemented into the program. The calculation process for obtaining the magnitude of this value is a rather complex iterative process which depends on a lot of factors as will be shown below.

The program works on the basis that a reduced isentropic exit velocity in leveled flow λ_{2T} is known. Henceforth, subscript 1 will denote inlet, 2 will denote outlet, and n or th denotes the throat section of any *blade row*.
The work done by Mamaev allows for calculations to be made whether the flow

exceeds sonic velocities or not by using different formulas. The component λ_{2T} is defined as:

$$\lambda = \frac{v_{local}}{v_{critical}} \tag{2.40}$$

where the velocities v are normal to the area through which the flow progresses.

To find the relationships described below, Mamaev starts off with a function y to represent a general mass flow. Functions π and q are also needed, all of which are dependant on the C_p-ratio γ and λ number.

$$q(\lambda,\gamma) = \lambda \cdot \Big(\frac{\gamma+1}{2}\Big)^{\frac{1}{\gamma-1}} \Big(1 - \frac{\gamma-1}{\gamma+1}\cdot\lambda^2\Big)^{\frac{1}{\gamma-1}} \tag{2.41}$$

$$\pi(\lambda,\gamma) = \Big(1 - \frac{\gamma-1}{\gamma+1}\cdot\lambda^2\Big)^{\frac{\gamma}{\gamma-1}} \tag{2.42}$$

$$y(\lambda,\gamma) = \frac{q(\lambda,\gamma)}{\pi(\lambda,\gamma)} = \frac{y(\lambda)}{\pi(\lambda)} \tag{2.43}$$

To decide if the flow is sonic or not, the author introduces a limiting velocity λ_{2T}^{lim}:

$$\begin{aligned} 0 = &\frac{y(\Psi_n)\pi(1)\sqrt{\bar{T}}\sin\beta_{2eff}\cos\chi}{\sqrt{y^2(\lambda_{2T}^{lim}\Psi)\pi^2(\lambda_{2T}^{lim})\Big(1+\frac{\tan^2\beta_n}{\cos^2\nu_n}\Big)\cos^2\beta_n\cos^2\nu_2 - \mathbf{B}}} \\ &- \sqrt{\frac{\lambda_{2T}^{lim}\cdot\Psi^2}{\Psi_n K}\,\bar{r}^2\bar{T}\Big(1+\frac{\tan^2\beta_n}{\cos^2\nu_n}\Big) - 1} \\ &\text{where } \mathbf{B} = \bar{T}y^2(\Psi_n)\pi^2(1)\sin^2\beta_{2eff}\cos^2\chi \end{aligned} \tag{2.44}$$

A lot of parameters are introduced in equation (2.44) which adds to the complexity in using Mamaev's method. They will be described more thoroughly continuously in this section.
Ψ is a term related to losses via the following:

$$\Psi_i = \sqrt{1-\zeta_i} \tag{2.45}$$

where i can be any station of a blade and ζ is found from the following relation:

$$Y = \frac{\Big[1 - \frac{\gamma-1}{2}M^2\big(\frac{1}{1-\zeta}-1\big)\Big]^{\frac{-\gamma}{\gamma-1}} - 1}{1 - \Big(1+\frac{\gamma-1}{2}M^2\Big)^{\frac{-\gamma}{\gamma-1}}} \tag{2.46}$$

In equation (2.46) above, Y is the pressure loss coefficient defined earlier in equation (2.21) and M is the mach number, $M = \dfrac{V}{a}$ where a is the speed of

sound in the local medium. To convert from Mach to Mamaev's usage of the laval number λ, the following relation is used:

$$M = \sqrt{\frac{2\lambda^2}{(\gamma+1)-(\gamma-1)\lambda^2}} \Rightarrow \lambda = \sqrt{\frac{M^2(\gamma+1)}{2+(\gamma-1)M^2}} \tag{2.47}$$

Continuing with definitions from equation (2.44):

$$\bar{r} = \frac{r_2}{r_n} \approx 1 + 0.2 \cdot (\frac{r_1}{r_2} - 1) \tag{2.48}$$

$$\bar{T} = \frac{T_{02}}{T_{0n}} \approx \left(1 + U_2^2(\frac{1}{\bar{r}^2} - 1)/(2H_2 + W_1)\right)^{-1} \tag{2.49}$$

$$\sin\beta_{2eff} = \frac{A_n}{A_{ax_2}} \tag{2.50}$$

$$\chi = \frac{\delta - \omega_2}{2} \tag{2.51}$$

Equation (2.51) has a term ω_2 which can be found from the geometrical considerations of figure 6.

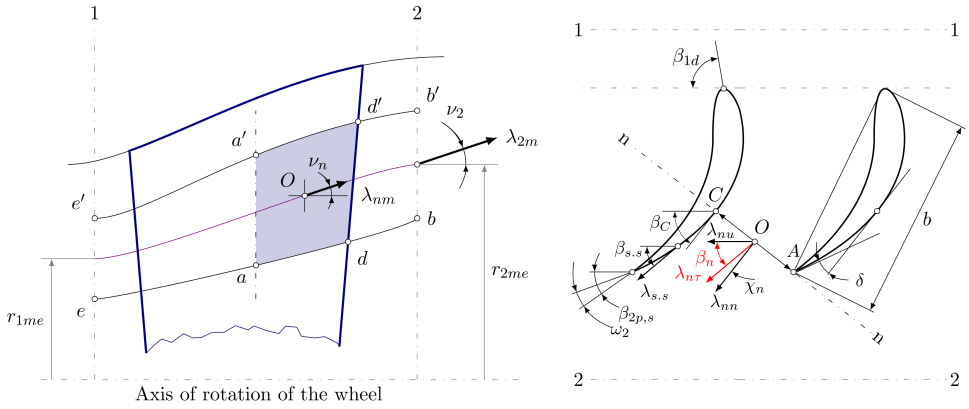


Figure 6: Blade row throat geometry. Image by Marcus Thern.

$$\cos\omega_n = \frac{\cos\nu_n}{\cos\beta_n\sqrt{\cos^2\nu_n + \tan^2\beta_n}} \tag{2.52}$$

Moreover, the angles β_n, ν_n and ν_2 are defined by:

$$\beta_n = \beta_{metal} + \delta/2 \tag{2.53}$$

$$\nu_n = \arctan\left(\frac{r_{E,TE,i} - r_{E,LE,i}}{b_i}\right) \tag{2.54}$$

Subscript *metal* indicates the physical metal angle out of the blade and $b_i = X_{TE,i} - X_{LE,i}$ unless defined by stagger as per equation (2.33) is the axial

chord of each blade i. LE and TE denote leading edge and trailing edge of a blade, where the leading edge is the part of the blade that is struck by the fluid first, and the trailing edge where this fluid exits the blade. Following discussions with the supervisor, it was determined that v_2 should be assumed to be equal to v_n as it has a very minimal impact on the results.

With these definitions in place, equations (2.47) and (2.44) may be used to determine if the flow is sonic or not. This is especially important due to how shockwaves and choked flows affect the passage. The first case without an onset of sonic flow uses the following relation to find the outlet angle β_2, i.e if $\lambda_{2T} < \lambda_{2T}^{lim}$:

$$\begin{aligned}\beta_2 = \arctan \Bigg[& \frac{\sin \beta_{2eff} \cos \chi}{\cos \beta_n} \cdot y(\lambda_n \Psi) \cdot y^{-1}(\lambda_{2T} \Psi) \\ & \times \pi(\frac{\lambda_n \Psi}{\Psi_n}) \cdot \pi^{-1}(\lambda_{2T}) \cdot \sqrt{\frac{1 + \tan^2 \beta_2 / \cos^2 \nu_2}{1 + \tan^2 \beta_n / \cos^2 \nu_n} \bar{T}} \Bigg]\end{aligned} \tag{2.55}$$

This equation introduces a parameter for the λ-number at the throat which might be interesting to analyze separately as it provides information about the conditions of the throat passage. It is defined as per Mamaev from the following relation:

$$\begin{aligned}\lambda_2 r_2 \sqrt{\bar{T}} / \sqrt{1 + \tan^2 \beta_2 / \cos^2 \nu_2} &= \lambda_n r_n K \cos \nu_n / \sqrt{\cos^2 \nu_n + \tan^2 \beta_n} \\ \Rightarrow \lambda_n &= \frac{\lambda_{2T} \bar{r}}{K} \sqrt{\frac{1 + \tan^2 \beta_n / \cos^2 \nu_n}{1 + \tan^2 \beta_2 / \cos^2 \nu_2} \bar{T}}\end{aligned} \tag{2.56}$$

The other case is when the flow is sonic and possibly choked. In other words, if the λ_{2T} number is at or exceeds the limit, $\lambda_{2T} \geq \lambda_{2T}^{lim}$:

$$\beta_2 = \arctan \frac{y(\Psi_n)\pi(1)\sqrt{\bar{T}} \sin \beta_{2eff} \cos \chi \cos \nu_2}{\sqrt{y^2(\lambda_{2T}\Psi)\pi^2(\lambda_{2T}) \left(1 + \frac{\tan^2 \beta_n}{\cos^2 \nu_n}\right) \cos^2 \beta_n \cos^2 \nu_2 - \mathbf{C}}} \tag{2.57}$$

where $\mathbf{C} = y^2(\Psi_n)\pi^2(1)\bar{T} \sin^2 \beta_{2eff} \cos^2 \chi$

Stemming from the complexity of this model, values for each component in these equations are more often than not unknown which leads to iterative solution methods, estimations of values and previous knowledge of similar geometries when defining the system. Regardless, usage of Mamaev's model will provide results independent of if the flow is sonic or not. This in combination with the fact that it is theoretically based is one of the model's main strengths.

2.5.2 Traupel's angle correction

The model above does not actively account for geometries with a relatively large clearance to height ratio. As supplement, Traupel [15] introduces a

correction to the blade outlet angle in these cases. The correction $\beta_{corr.}$ can be written as

$$\beta_{corr.} = \beta_2 + 70.5\frac{\text{clearance}}{\text{height}}\frac{r_{tip}}{r_E} - 0.714\frac{\text{spacing}}{\text{height}} - 0.5^\circ \tag{2.58}$$

2.5.3 Mixing and gap calculations

In the middle of any blade pair exists a void referred to as gap. This gap is interesting as it allows for injection of cool air to avoid overheating. On top of this it may also feature increased gas expansion if radii are not constant. The program includes the possibility for mixing at two locations at every gap location: trailing edge mixing and gap mixing.

Mixing at the trailing edge is rather simple and uses energy, momentum and impulse balances. If cooling is present, it is added to the main flow at the trailing edge of a blade. Subscript mix refers to positions just at trailing outlet at the moment of mix injection and are considered to be a separate position from standard blade outlet.

$$\dot{m}_{mix} = \dot{m} + \dot{m}_{cool} \tag{2.59}$$

By knowing properties (usually pressure and temperature) of the mixing fluid, an energy balance for the enthalpy gives the mixing enthalpy:

$$h_{0mix} = \frac{\dot{m} \cdot h_0 + \dot{m}_{cool} \cdot h_{0cool}}{\dot{m}_{mix}} \tag{2.60}$$

A tangential momentum balance is set up to find the velocity component $C_{\theta,mix}$:

$$C_{\theta mix} = \frac{\dot{m} \cdot c_\theta + \dot{m}_{cool} \cdot C_{\theta cool}}{\dot{m}_{mix}} \tag{2.61}$$

Similarly for the axial velocity, the axial impulse is used with the addition of pressure differential influence at the trailing edge area A:

$$C_{a,mix} = \frac{(P - P_{mix}) \cdot A + \dot{m} \cdot C_a + \dot{m}_{cool} \cdot C_{a,cool}}{\dot{m}_{mix}} \tag{2.62}$$

Gap calculations differ from their mix counterpart. The gap mixing is assumed to occur between two rows of blades, after the mixing process. The balance equations are slightly changed to include changes in radius and thus area, if present, from blade outlet to next row's blade inlet. Increased areas normally result in a reduced mass flow and therefore reduced axial velocity. The gap mass flow is expressed as

$$\dot{m}_{gap} = \dot{m}_{mix} + \dot{m}_{20} + \dot{m}_{21} \tag{2.63}$$

The flows 20 and 21 denote cooling and leak flows respectively. Their tangential flow component is assumed to be 0. $dR_\theta = dR_a = 0$ is the viscous wall component loss for the flows. The gap version of the tangential velocity affected by radius change is expressed as

$$C_{\theta,gap} = \Big(r_{E,TE} \cdot \dot{m}_{mix} C_{\theta mix} + \dot{m}_{20} C_{\theta 20} r_{mid} + \dot{m}_{21} C_{\theta 21} r_{mid} - dR_\theta\Big) \cdot \frac{1}{\dot{m}_{gap} \cdot r_{E,LE}} \tag{2.64}$$

The axial impulse of the flows, similar to above, are assumed to be 0. Fielding [16] establishes an equation for the flow force on an element through expansion. This gives the expression for axial velocity:

$$\begin{aligned} C_{a,gap} = \Big[&(P_{mix} A_{mix} - P_{gap} A_{gap}) + (P_{mix} + P_{gap})/2 \cdot (A_{gap} - A_{mix}) \\ &+ \dot{m}_{mix} \cdot C_{a,mix} + \dot{m}_{20} \cdot C_{a,20} \\ &+ \dot{m}_{21} \cdot C_{a,21} - dR_a\Big] \cdot \frac{1}{\dot{m}_{gap}} \end{aligned} \tag{2.65}$$

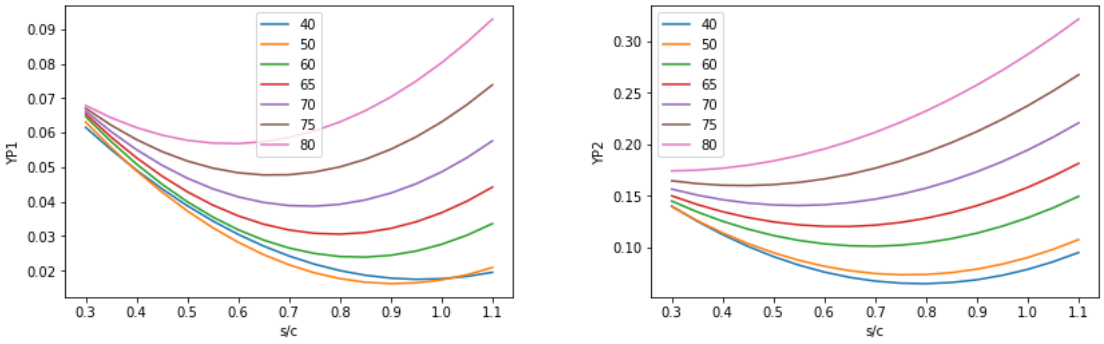
2.6 Flow loss modeling

This thesis uses work originally conducted by Ainley Mathieson et. al. [4] [5] that has been implemented into MATLAB by Lund University in a program called LUAX-T [17] which has been proven to work for conventional gas turbines. This code however is also meant to be able to correctly estimate performance for space turbines. These turbines have special designs and thus the loss model must be modified to account for this. Generally, this is done by adjustment of coefficients or introduction of factors if certain conditions are met. The very performance of these models depend heavily on the minutiae of the modifications.
The total pressure loss coefficient is defined as the sum of the profile, secondary, trailing edge and clearance losses:

$$Y_{TOT} = Y_P + Y_S + Y_{TE} + Y_{CLR} \tag{2.66}$$

2.6.1 Profile losses

The losses stemming from the shape of a blade can be found by studying two different types of profile loss coefficients. The first type of coefficient represents nozzles and the second impulse blades and is plotted below in figure 7 against the pitch-chord ratio s/c.



(a) Profile loss for nozzle type blading

(b) Profile loss for impulse type blading

Figure 7: Profile loss coefficients plotted versus s/c with varying exit flow angles from 40 to 80°.

Generally, turbine blades are a combination of nozzle and impulse blades. Interpolation is therefore needed and this is expressed as:

$$Y_P^* = \left(Y_{P1} + \left| \frac{\alpha_{1,b}}{\alpha_2} \right| \cdot \frac{\alpha_{1,b}}{\alpha_2} \cdot (Y_{P2} - Y_{P1}) \right) \left(\frac{t/c}{0.2} \right)^{\frac{\alpha_{1,b}}{\alpha_2}} \tag{2.67}$$

An extra profile loss shock component is given by

$$\tilde{K}_{SH} = \left(\frac{\Delta p_0}{q_1}\right)_{SH} \left(\frac{p_1}{p_2}\right) \frac{1 - \left(1 + \frac{\gamma-1}{2} M_1^2\right)^{\frac{\gamma}{\gamma-1}}}{1 - \left(1 + \frac{\gamma-1}{2} M_2^2\right)^{\frac{\gamma}{\gamma-1}}} \tag{2.68}$$

where

$$\left(\frac{\Delta p_0}{q_1}\right)_{SH} = \left(\frac{r_{hub}}{t_{ttp}}\right) \left(\frac{\Delta p_0}{q_1}\right)_{hub} \tag{2.69}$$

Equation (2.69) is also modeled to have an upper asymptotic limit of unity to avoid faulty calculations for certain Mach numbers, given by

$$K_{SH} = \sqrt{\frac{\tilde{K}_{SH}^2}{\tilde{K}_{SH}^2 + 1}} \tag{2.70}$$

A fix for channel acceleration is also used, as per

$$\begin{aligned} K_1 &= \begin{cases} 1.0 & \text{for } M_2 \leq 0.2 \\ 1 - 1.25\,(M_2 - 0.2) & \text{for } M_2 > 0.2 \end{cases} \\ K_2 &= \left(\frac{M_1}{M_2}\right)^2 \end{aligned} \tag{2.71}$$

These equations result in the acceleration factor K_{accel}:

$$K_{accel} = 1 - K_2(1 - K_1) \tag{2.72}$$

A term adhering to supersonic drag rise is introduced:

$$CFM = 1 + 49.5(M_2 - 1)^3 + 3.3(M_2 - 1)^2 \text{ for } M_2 > 1 \tag{2.73}$$

Finally a simple Reynolds number correction factor is used though its influence is only relevant for very high or low Reynolds numbers.
This results in the concluding expression for profile loss:

$$Y_P = 0.914(\tfrac{2}{3} Y_P^* K_{accel} + K_{SH}) \cdot CFM \cdot f_{Re} \tag{2.74}$$

2.6.2 Secondary losses

The secondary flow losses are modeled through a combination of aspect ratio logic, acceleration and Ainley's loading parameter:

$$Y_S^* = \frac{0.0334 f_{AS}}{f\,(\delta'/c)} f_{AS} \left(\frac{\cos\alpha_2}{\cos\alpha_{1,b}}\right) \left(\frac{C_L}{s/c}\right) \frac{\cos^2\alpha_2}{\cos^3\alpha_m} \tag{2.75}$$

In equation (2.75) above, the aspect ratio adjustment f_{AS} is found from

$$f_{AS} = \begin{cases} \frac{1}{2}\frac{1}{\left(\frac{h/c}{2}\right)^{0.7}} & \text{for } h/c \leq 2 \\ \frac{1}{h/c} & \text{for } h/c > 2 \end{cases} \tag{2.76}$$

Acceleration is simply denoted as

$$\frac{\cos\alpha_2}{\cos\alpha_{1,b}} \tag{2.77}$$

And the loading parameter is:

$$\frac{Clr}{s/c}\frac{\cos^2\alpha_2}{\cos^3\alpha_m} \tag{2.78}$$

It was claimed by Aungier [18] that an asymptotic limit should be imposed, shown below:

$$Y_S^* = \sqrt{\frac{Y_S^{*2}}{1 + 7.5Y_S^{*2}}} \tag{2.79}$$

Similar to the case in profile loss modeling, another factor K_3 is introduced for the axial aspect ratio:

$$K_3 = \frac{1}{\left(\frac{h}{b_x}\right)^2} \tag{2.80}$$

A compressibility correction is also used, defined as:

$$K_{cs} = 1 - K_3\left(1 - K_{accel}\right) \tag{2.81}$$

Resulting in the final expression for Y_S:

$$Y_S = 1.2 \cdot Y_S^* \cdot K_{cs} \cdot K_{denton} \tag{2.82}$$

where K_{denton} is a factor that varies based on if it is a stator or rotor blade.

2.6.3 Tip clearance losses

For unshrouded blades, a stage efficiency of 90 % is assumed as well as $\Delta Y = 2\Delta\eta$. The clearance loss may then be defined as:

$$Y_{CLR} = 2 \cdot 0.93 \cdot 0.9\frac{Clr/h}{\cos(\alpha_2)}\left(\frac{r_{tip}}{r_E}\right) \tag{2.83}$$

2.6.4 Modified loss model

The fundamental technique used to model for these losses are heavily dependant on validation against real-world test data as Moustapha et. al. [5] reiterates in their documentation. It is also pointed out that in times of lacking or missing meaningful data, common sense and more importantly experience should be used to model. This thesis' supervisor fits well under this category and it is in at their discretion that the ensuing modifications have been made. Specifically for space turbines where aspect ratios typically could fall below very low values or where excessive values for row acceleration occur. For sufficiently high row accelerations, adjustments are made for a parameter in the secondary loss model by adjusting equation (2.75):

$$Y^*_{S,new} = Y^*_{S,old} \cdot \frac{1}{AVR} \tag{2.84}$$

where AVR is the axial velocity ratio. This should help predict losses with regards to model versus reality acceleration discrepancies.
Additionally, the limiting value for equation (2.79) is changed to 3 from 7.5:

$$Y^*_S = \sqrt{\frac{Y^{*2}_S}{1 + \mathbf{3}Y^{*2}_S}} \tag{2.85}$$

Equation (2.81) runs the risk of attaining negative values at high values of equation (2.80) which has no physical validity. Therefore, it is limited asymptotically to prevent this, as per:

$$K_3 = \sqrt{\frac{K_3^2}{1 + K_3^2}} \tag{2.86}$$

The modifications also feature additions to the clearance loss as blade heights drop to low levels which in turn increases the relative influence of blade tip clearance.

2.7 Pressure-finding algorithms

With a working flow calculator in place, the axial pressure distribution must be found. The outlet angle model is able to handle transonic flows. This is an important quality since one of the pressure finding algorithms of the program is using Denton's [7] method which as mentioned in the article strongly depends on being fed correct data for accurate results, data out of which the outlet angle plays a role - especially if sonic flows are present. The program will also feature a more straightforward brute force-style method developed by Came [19]. However, these require boundary conditions to be set as the system would otherwise be under-defined.

2.7.1 Initial pressure guess

A gas turbine's performance and behavior is bound by its pressure ratio, specifically the way that this ratio translates into inlet and outlet pressure. For all intents and purposes, the pressure gradient will be positive from inlet to outlet and the mass flow through a turbine must be continuous. Modeling after this continuous reduction in pressure allows the system to become defined. However, since the mass flow is a function of pressure change (among many, many other things), correctly establishing an axial pressure distribution is a very difficult task. For this reason, an initial guess is required to initiate calculations. For a given pressure ratio and outlet pressure, the axial pressure distribution may be estimated linearly.

2.7.2 Denton's target pressure method

Denton's method works by continuously adjusting the pressure based on a derivative of the change in mass flow with regards to pressure at the trailing edge of any blade in the turbine. Through a series of equations Denton reveals that the mass flow at a trialing edge J is a function of the change of static pressure one trailing edge upstream, $J-1$, from the relation below:

$$\frac{\delta \dot{m}_J}{\delta P_{J-1}} = \frac{\dot{m}_J}{P_{J-1}} \cdot \frac{1}{\gamma M_J^2} \cdot \left(\frac{\Omega r \sin \beta}{W} \right)_{J-1} \tag{2.87}$$

The change of mass flow as a result of a pressure variation of the same station is also needed:

$$\frac{\delta \dot{m}_J}{\delta P_J} = \frac{\dot{m}_J}{\rho_J C_J^2} \cdot (M_J^2 - 1) \tag{2.88}$$

Each iteration with a specific pressure distribution will yield a certain mass flow out of the first stator. This is the sought mass flow - $\dot{m}_{target}$. For each trailing edge, the error in mass flow is created

$$d\dot{m}_{TE} = \dot{m}_{target} - \dot{m}_{TE} \tag{2.89}$$

For the first stator outlet, this will be equal to 0. Each new iteration *i+1* will also update the target flow using the previous iteration *i's* value as per

$$\dot{m}_{target,new} = \dot{m}_{target,old} + RLX \cdot (\dot{m}_{stator,out} - \dot{m}_{target,old}) \tag{2.90}$$

with a relaxation factor $RLX \leq 1$ for stability. The corresponding required change in pressure for trailing edge J may be expressed as

$$dP_J = dm_{TE,J+1} - \frac{\frac{\delta \dot{m}_J}{\delta P_J} \cdot dP_{J+1}}{\frac{\delta \dot{m}_J}{\delta P_{J-1}}} \tag{2.91}$$

This will then yield a new pressure distribution, which after convergence will satisfy mass flow continuity at all stations.

2.7.3 Came's brute-force method

If for whatever reason the first pressure solver would fail, a second method should be tried. Came [19] developed a significantly more simple and straightforward method that corrects each trailing edge pressure $P_{TE,J}$ based on a ratio of newly computed target mass flow by mass flow of a trailing edge downstream:

$$P_{TE,J,new} = P_{TE,J,old} \cdot \frac{\dot{m}_{target}}{\dot{m}_{TE,J+1}} \tag{2.92}$$

In equation (2.92) above, the target mass flow is corrected using the previous iteration's value of first stator $\dot{m}_2$ outlet and a relaxation factor RLX of less than unity:

$$\dot{m}_{target,new} = \dot{m}_{target,old} + RLX \cdot (\dot{m}_2 - \dot{m}_{target,old}) \tag{2.93}$$

As the method reaches convergence, the newly calculated inlet pressure reaches the stator outlet flow value.

2.8 Iterative solvers

The program will also use other types of iterative solvers for many of the mathematical problems encountered. Denton's special method described above is one of them but only works for its intended use. The mix and gap calculations mentioned earlier use the so-called Newton Raphson method.

2.8.1 Newton Raphson's method

This solution procedure essentially works by establishing the rate of change of the sought parameter and changing it in favor of minimizing this derivative. This is done by using the first two terms of a Taylor series:

$$f(a_{n+1}) = f(a_n) + (a_{n+1} - a_n) \cdot f'(a_n) \tag{2.94}$$

Solving for the value results in

$$a_{n+1} = a_n - RLX \cdot \frac{error}{f'(a_n)} \tag{2.95}$$

The right hand side has a RLX or relaxation factor to avoid overshooting. This methodology is used in mixing calculations, gap calculations and in Mamaev calculations, for example. The method works whether the derivative is known or unknown. For the latter, a numerical derivative may be set up.

3 Methodology

In this section of the thesis the overall structure of the code will be outlined to show how different files, functions and subroutines interact and work together. The terms function, routine and subroutine may be used interchangeably in this section but all refer to Python calculation procedures. Moreover, the general method for each file is described.
The program contains a large number of subroutines and calculation processes, some of which are of an iterative nature. For a single test case, the main overseeing calculation structure can be seen in the flow chart below:

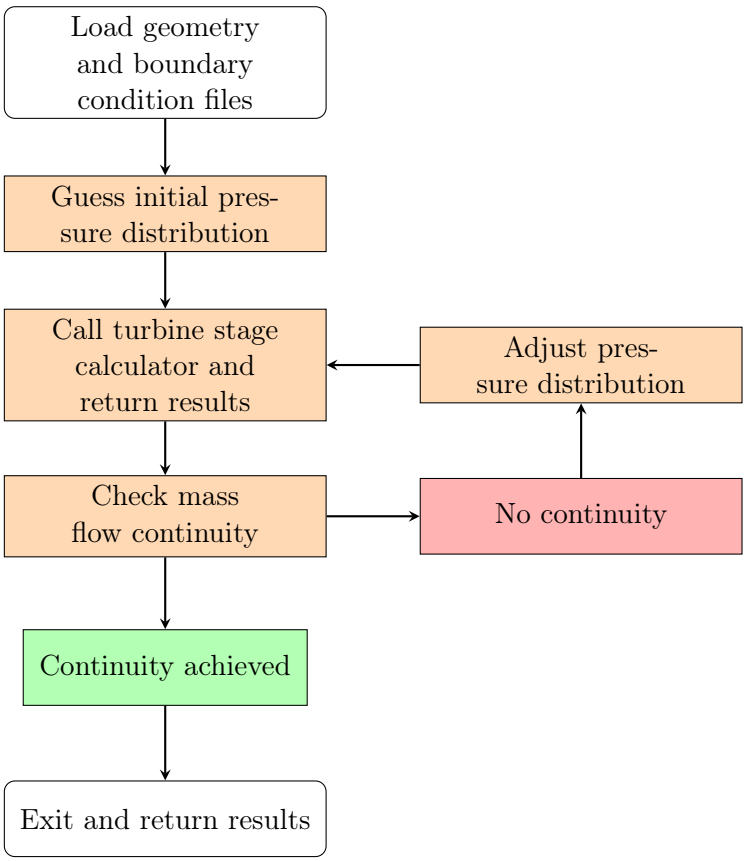


Figure 8: General program overview.

The program loads geometry data and other needed parameters such as rotational speed, number of stages, pressure ratio and temperatures from the supplied files. Using this data, the system is defined and calculations may begin. Roughly described, the loop above solves iteratively based on pressure changes and mass flows as a result of the pressure. Therefore an initial guess of how the pressure is distributed in the turbine is needed - in this program it is linearly based. With an initial pressure guess, the program then enters

the turbine stage calculator and passes through all stages, row by row. After each blade row passage, the program predicts an entropy increase which is then corrected until entropy convergence is met. After the calculation is finished, the program checks if mass flow continuity is achieved. If not, the program reverts and adjusts the pressure based on the last iteration's results and re-enters the turbine stage calculator until pressure convergence is met, at which point the program may return full scale data from the calculations for post-processing.

The calculations are performed on a row-wise basis beginning in the turbine inlet. As such, properties downstream are strictly dependant on the known properties upstream which in reality means that small changes can influence the flow behavior greatly - hence the need for iteration solving. In grossly oversimplified terms, to reach mass flow continuity relies on finding the correct pressure change as well as correct entropy change.

3.1 Structure

The actual structure of the program follows the logic presented in the flow chart in figure 8. However, the program itself is divided into several files for readability and ease of use. This allows for modularization if components need to be updated or changed, without changing the entire structure of the code and thereby compromising the reliability. The following Python files are required to run the program:

`aero_losses.py`	AMDCKOMK modified loss model
`main.py`	Main method containing pressure solver
`mamaev_angleout.py`	Outlet angle model
`turbine_stage_calc.py`	Turbine stage calculator
`read_geometry.py`	Geometry reading and calculations
`velocityvectors.py`	Flow conversion subroutine
`radial_eq.py`	Radial pressure distribution

The files above are the bare minimum for single-case testing and obtaining results. For large scale validating and post-processing purposes, these files may be used:

```
run_mass_test.py
post_processing.py
miscellaneous_turb_calc.py
draw_velocity_triangles.py
```

Additionally, it is required that the user have `CoolProp` installed for Python.

3.2 Geometry data

The `read geometry` file handles all geometry related definitions. The basic geometry object is defined and subsequent initialization calculations are made. The data that is to be input into the program must be entered on a row basis to fully define the geometry object and fit the program's architecture. The full list of required parameters are listed in Appendix A along with a brief description. The unit convention is easily changed but should be consistent for any new geometry object being read. It is highly important that these parameters are introduced correctly with identical naming. This enables the program to access the relevant values and perform correct computations.

3.2.1 Data parsing

The entire program bases its calculations on the values provided by the user from a .csv file. The values should be separated via spaces. This is done using the Python library Pandas [20]. The program reads and defines parameters from two files:

`geometry_file.csv`	Geomtery data (see Appendix A)
`init_vals.csv`	Operating point/boundary condition/initialization data

Pandas offers the choice of also reading excel files. This is easily implemented should future users choose to do so. The initialization values contain the following five boundary conditions:

`RPM`	Rotational speed [RPM]
`T0_init`	Total inlet temperature [K]
`P0_init`	Total inlet pressure [Pa]
`PR`	Pressure ratio (Total-to-static or total-to-total)
`P_out`	Static outlet pressure [Pa]

The last two parameters are essentially the same expressed differently if the pressure ratio is given as total-to-static. However, if the total-to-total ratio is given the program must have a supplied value of the outlet static pressure as this is a key component in the iterative solution method of the target pressure method.

3.2.2 Subsequent geometry-related calculations

The geometry routine reads these values and performs the geometry calculations for each row, based on the number of row values present in the geometry file. For example, r_{Euler} values are calculated for each leading and trailing edge of each row, along with areas, blade speeds or average blade height. These values along with the results of the rest of the calculation are then used to find the geometrical parameters that are required for `aero losses` to function.

The geometry subroutines then return these values when called upon. As the program is a function of the geometry, a majority of the files in the program will need to receive geometry data at various times.

3.3 Angle calculations

The program continuously performs angle calculations. Most notably when conditions at a trailing edge are needed, which is where the outlet angle model is used. There is also a need to convert 3D flow vectors into a 2D plane which in turn may be converted to the program's desired 1D representation.

3.3.1 Blade outlet angle

The Mamaev angle calculation method is implemented in `mamaev angleout` and features sophisticated equations outlined previously in this report which require solvers that operate iteratively. This is due to insufficient information about flow and that the equation for solving for the outlet angle β_2 is a function of itself. This requires an initial guess to start the iteration. The flow chart below describes the inner structure of the function.

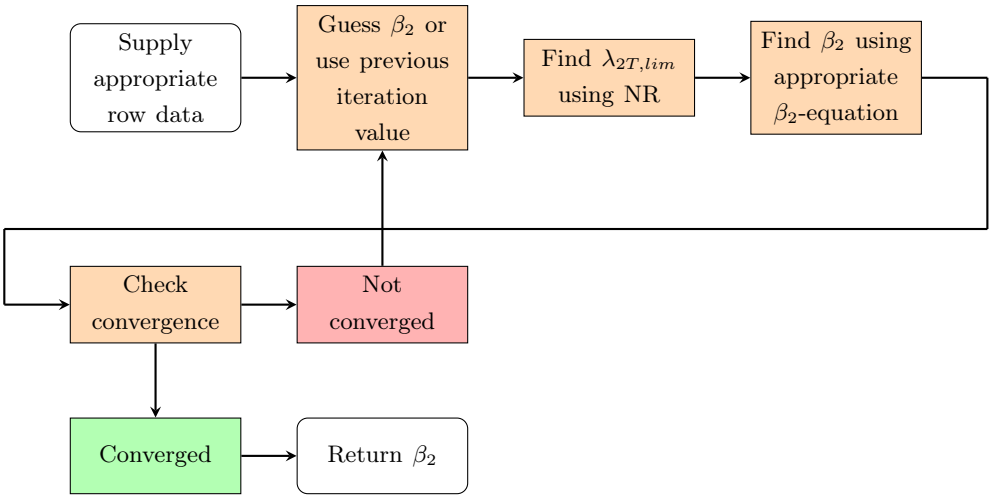


Figure 9: General methodology for finding the outlet angle.

From figure 9 it is seen that the blade outlet angle function is initialized by supplying data about the current stator or rotor row. It takes information that is known at the inlet of a blade row and is accompanied by properties at the outlet that are known from an estimation of the entropy increase. With this information, additional parameters such as λ_{2T}^{lim} may be calculated. This parameter which stems from equation (2.44) may behave in a problematic manner due to containing a negative root for certain values of λ which renders it undefined and useless.
This equation can be solved by Newton-Raphson (NR) solution described in

equation (2.95) using a numerical derivative or by using of one of Python's numerical solvers. The method below uses NR and creates a vector with two different input values of λ_{2T}^{lim} which in turn give two different values of equation (2.44) in order to find the rate of change. The goal is to bring this to 0 without compromising the negative root. Suppose the equation whose root to find is called y and find the root using the methodology outlined in algorithm 1:

Algorithm 1 Obtaining λ_{2T}^{lim} using NR

Set error = 1 and $\lambda_{2T,1}^{lim} = 1.05$
while |error|> conv limit **do** ▷ Checks error value
  Create temporary help parameter: $\lambda_{2T,2}^{lim} = \lambda_{2T,1}^{lim} \cdot 1.001$
  Find the two values for $y(\lambda)$
  Create numerical derivative: $\delta k = dy/d\lambda$
  Adjust the value: $\lambda_{2T,1,new}^{lim} = \lambda_{2T,1}^{lim} - RLX \cdot y/\delta k$
end while
Return $\lambda_{2T,1}^{lim}$

The limiting velocity is now known which lets the program continue with the angle calculations. A guess value for β_2 is initialized by setting $\beta_2 = \beta_{2eff}$ from equation (2.50). The program then checks which β_2 equation is relevant, see equations (2.55) and (2.57). Following this, the guess value is compared to the newly calculated value and the process repeats until an acceptable value of the error is achieved.

3.3.2 Velocity vectors module

This file is called `velocityvectors` and is used to convert the flow from a 3D value with components in the tangential, axial and radial plane to its corresponding base component values. This is done using the equations defined in (2.5)-(2.7) usually after obtaining a result from the outlet angle model and when converting to or from a stationary reference frame.

3.3.3 Meriodional flow angle ϕ

The radial-axial flow angle is the angle between the meriodional vector and axial vector. It is calculated using turbine geometry in the `read geometry` file and is part of the velocity vectors module. If the turbine has constant blade radii across its axis, the flow angle is per definition 0°.

3.4 Loss model - AMDCKO et. al.

The program features the modified version of the so-called *AMDCKO et. al.* model. These losses are obtained by importing the `aero losses` file. As outlined in the theory section, it is heavily dependent on geometry and coefficients to provide reliable answers. The file itself is rather linear and contains

programming logic to handle different scenarios that has been input. Modifications are easily made in the start of the file for certain coefficients that are used.

3.5 Turbine stage calculations

The `turbine stage calc` is where the majority of the program's calculations occur. It starts at the turbine inlet with supplied geometry and fluid data and works its way through the turbine on a stage-by-stage basis with additional station categorization.

3.5.1 Initialization

In order to start the program, the turbine system's boundary conditions must be supplied in addition to geometry and fluid information. These are obtained from the geometry and initialization data. Currently, the program uses the following arguments to start computations:

1. Axial pressure distribution (from the pressure ratio)
2. Total pressure at inlet
3. Total temperature at inlet
4. RPM
5. Main iteration number*
6. Inlet mass flow*
7. Entropy increase (optional)*

This combination of initialization data will return a resulting mass flow at each blade. Usually, this does not imply convergence as the initial pressure distribution guesses are made linearly. For this reason, the calculations are of an iterative nature. The last three parameters pertain to this iterative process and will be described more in-depth in the main section.

3.5.2 Station nomenclature

Figure 10 below displays a turbine stage accompanied by the station nomenclature that is used in the program. In the image, the blue blade represents a stator and the yellow represents a rotor. Since it is a meanline program, all locations below refer to the euler radius defined in the theory section, located approximately in line with the black arrows.

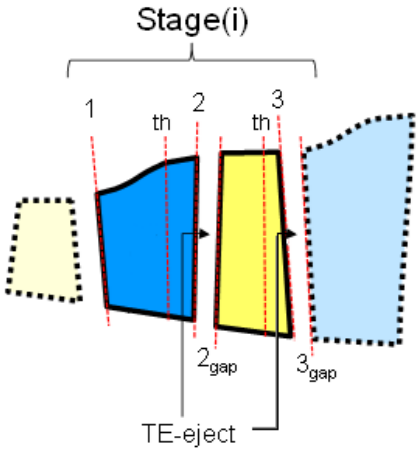


Figure 10: Station nomenclature for a stage i.

The stations above are described in Table 1.

Station	**Description**	**Source/usage**
1	Leading edge	Indata or from 3_{gap} of previous stage
2	Trailing edge	Stator outlet
2_{th}	Vane throat	Mamaev angle model
2_{mix}	TE ejection cooling	Mixing model - stator outlet
2_{gap}	Gap mixing, leading edge	Mixing model - rotor inlet
3	Trailing edge	Rotor outlet
3_{th}	Rotor throat	Mamaev angle model
3_{mix}	TE ejection cooling	Mixing model - rotor outlet
3_{gap}	Gap mixing, leading edge	Mixing model - next stage's stator inlet

Table 1: Descriptions of stage station nomenclature.

3.5.3 General methodology

Figure 11 offers insight on the turbine stage calculator's procedure. For any trailing edge, the supplied pressure is assigned and calculations begin. At the start of the calculation (for the stator of stage 1), the system requires initializing boundary conditions. For subsequent stages (stage 2 or higher), the stator inlet also known as station 1 inherits values that have been calculated by the previous stage at the rotor gap location. Following this, the program enters station 2 and assumes an increase of the entropy as a result of the stator passage and iterates on this until convergence is met. If stator TE mixing and gap mixing are present, the program iterates on mass flow continuity until convergence is met. The program then sweeps through the rotor blade and enters station 3, performing similar calculations to that of the stator. Gap positions of station 2 and 3 are identical to inlet positions of following blade (i.e. station 2_{gap} is rotor inlet and station 3_{gap} is next stage's stator inlet when

applicable).

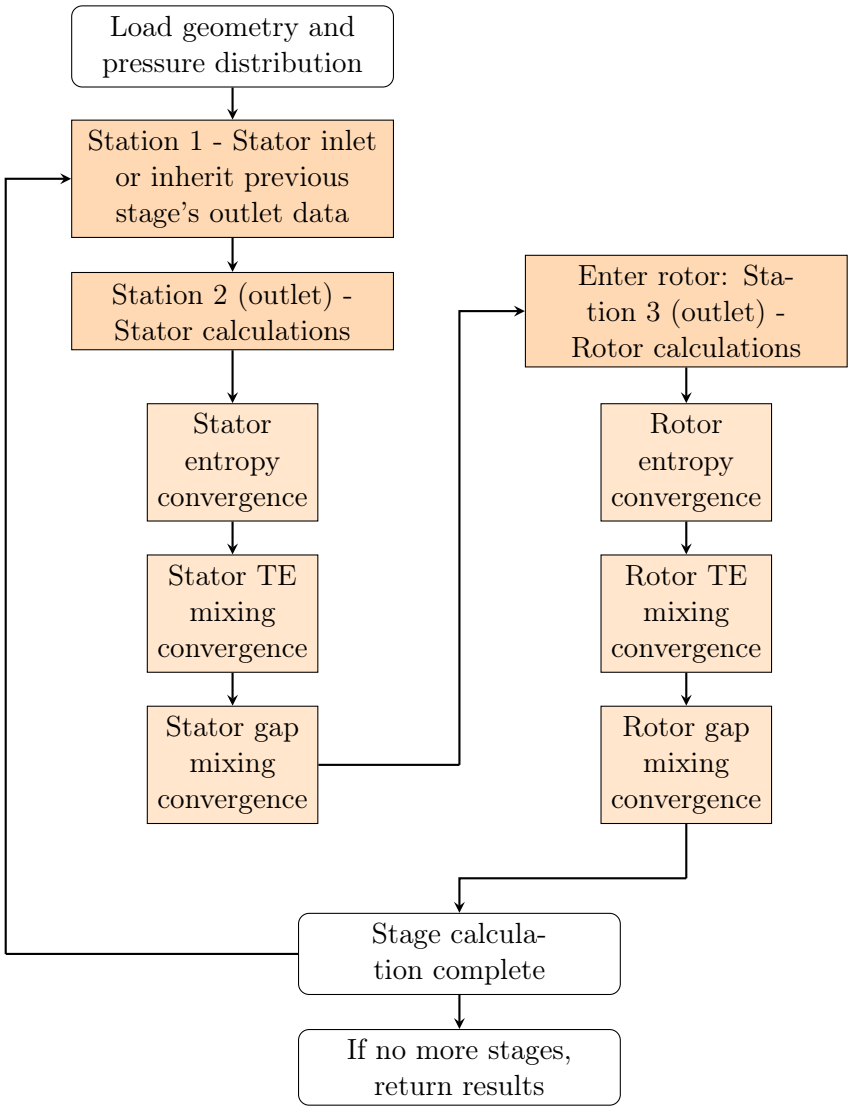


Figure 11: Flow chart showing the general procedure of the turbine stage calculator.

3.5.4 Turbine inlet

The inlet of the turbine is a special case as it requires an initial guess or initialization value due to the turbine system being under-defined with regards to mass flow information. For the very first iteration when *main iteration = 0* there is no information about the mass flow and as such these conditions are adequately approximated using a guess of the inlet mach number in equations (2.17) and (2.18). With these, CoolProp may be used to find the speed of sound a and the resulting inlet absolute velocity C_1, which can be used to find

the static enthalpy.

For *main iteration > 0*, the mass flow from the previous iteration is supplied and the program will iterate towards the flow by guessing a velocity C using the following algorithm:

Algorithm 2 Inlet velocity with known mass flow

Set error = 1 and $C_{1,temp} = 100$;
while |error|> conv limit **do** ▷ Checks error value
  Find $h_1 = h_{01} - C_{1,temp}^2$
  Find density $\rho = f(h_1, S_1)$
  Calculate $C_1 = \dot{m}/(A \cdot \rho)$
  Calculate error = $1 - C_{1,temp}/C_1$
  Set $C_{1,temp} = C_1$ and re-do until convergence
end while
Find static h, P, c_p, c_v, a

Note that this is only relevant for the first stage of a turbine - subsequent stator inlets will inherit the mass flow from rotor gap outlet as described.

3.5.5 Entropy iteration

Flows are not loss-less and thus experience an increase in entropy as it progresses and the same applies for turbines. In order to calculate this entropy increase, one must guess or estimate the very same value due to how enthalpy and pressure changes work. Only then can the necessary calculations be made to find the loss component Y_{tot}, which can be converted to a change in entropy through equation (2.21). Figure 12 shows the steps needed to find a correct value of the entropy change ΔS. The procedure varies slightly for a stator and rotor.
The convergence criteria is achieved when the entropy result that has been returned from the AMDCKO et. al. matches the used entropy value.

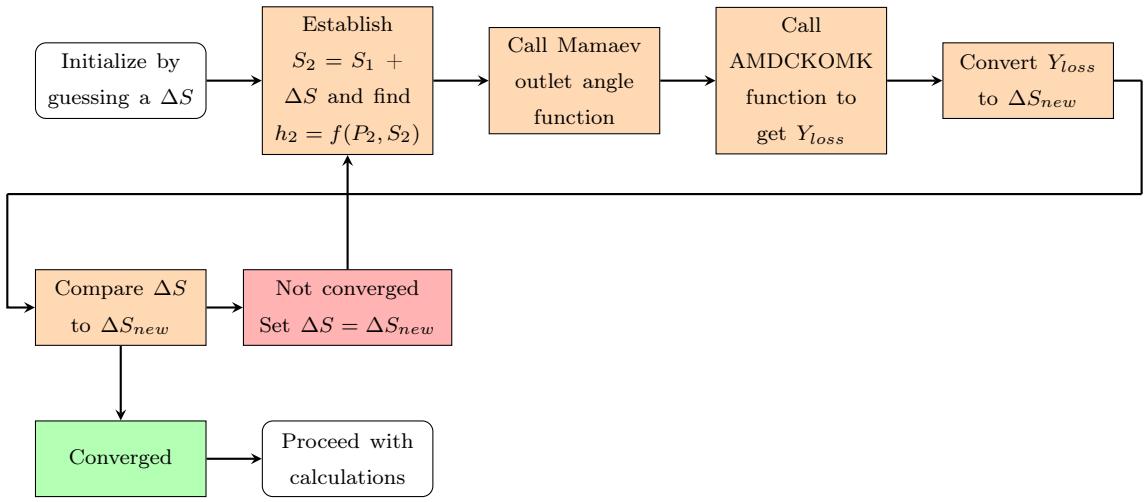


Figure 12: Genralized entropy loop iteration in `turbine_stage_calcs.py` file for a stator and rotor.

Stator loop

The program can either work with a supplied entropy increase from a previous iteration to reduce processing time or start off from scratch by guessing an arbitrary ds. With blade outlet entropy known, the static enthalpy can be found from CoolProp by combining it with the static pressure from the axial distribution guess. Using the concept of constant enthalpy $h_{02} = h_{01}$ as described in (2.19), the velocity C_2 can be found through

$$C_2 = \sqrt{2 \cdot (H_{02} - H_2)} \tag{3.1}$$

CoolProp is then used to find more static properties needed to call the angle model. With an outlet angle and absolute flow value, the velocity vectors module may be used. At this point the program has all values necessary to calculate the input arguments for the aero loss module which then returns a Y_{tot} before being converted to an increase in entropy per (2.21). This value is then checked and used as a new value for the entropy increase until convergence is achieved.

Algorithm 3 Stator entropy loop

Set error = 1 and assign Δs guess
while |error|> conv limit **do** ▷ Checks error value
  Find $S_2 = S_1 + \Delta s$
  Find C_2 from (3.1)
  Find static components from $f(P_2, S_2)$
  Calculate relevant parameters for outlet angle model; get angle
  Calculate relevant parameters for loss model; get Δs_{AMDC}
  Compare losses
  Set $\Delta s_{new} = \Delta s_{AMDC}$
end while

Rotor entropy loop
The main difference to the stator entropy loop is the change in reference frame to relative and using the constant rothalpy assumption from equation (2.20) resulting in a relative velocity W_3 through:

$$W_3 = \sqrt{2 \cdot (h_{2gap} - h_3 + \tfrac{1}{2}W_{2_{gap}}^2 + \tfrac{1}{2}(U_3^2 - U_{2gap}^2))} \tag{3.2}$$

For some iterations this value might dip below 0 which yields a complex, unusable answer. Therefore a fix was implemented to account for this by assigning a maximum value H_{3max} using the same equation with $W_3 = 0$.

3.5.6 Mix calculations

After entropy convergence is achieved for a blade, the program moves on to the mix station. The mix calculations featured in the program use the NR solution method and the theory from equations (2.59) to (2.61). Assuming unchanged radial velocity, the resulting flow C_{mix} is easily found from equation (2.8). This enables the program to calculate the static mixing enthalpy. The methodology is not as straight forward though, since the pressure P_{mix} is unknown. If no mixing is present, then P_{mix} will inherit the value of P but otherwise, an initializing guess is needed as well as iterative solving using NR, equation (2.95):

$$P_{mix,new} = P_{mix,old} - \frac{\dot{m}_{mix,error}}{\frac{d\dot{m}}{dP}} \tag{3.3}$$

The error is calculated from a standard flow equation using the density ρ at current mixing conditions. This gives

$$\dot{m} = \rho A C_a \tag{3.4}$$

which is compared to the actual mixing sum from (2.59). The derivative in the denominator is given as

$$\frac{d\dot{m}}{dP} = \frac{\dot{m}}{\rho C_a^2} \cdot \left(M_a^2 - 1\right) \tag{3.5}$$

When the mass flows from (3.4) and the flow from definition (2.59) are within acceptable range of each other, the mixing is considered finished and the program moves on to the gap calculations.

3.5.7 Gap calculations

After trailing edge mixing and before next blade row inlet is the gap. The gap calculations follow the same methodology with slight modifications. The axial impulse and tangential momentum balance are governed by equations (2.65) and (2.64). Other than that, the same solution methodology is employed for finding the correct pressure after the mix zone.
It is important to realize that the described methods assume stator outlet. Note that the same method is used after rotor exit only with relative components.

3.6 Main routine

The `main` file can be seen as the brain of the code. Through this file, the user runs the program for single operating points. Structurally speaking, the `main` file is the top of the pyramid and calls other files when the program is being run, with the exception of being fed initialization data from the `geometry` module as seen in figure 13.
The `main` file is also the home of the pressure solution methods and allows the user to change what algorithm is used for the correct pressure seeking. In the program, Denton's method is used by default. A more straight-forward brute force-style algorithm by Came is also implemented. As a supplement to user-written solvers, native Python solvers found in the `scipy.optimize` [21] environment can also be implemented. They were tested during the earlier days of the program but are not updated to account for the new input arguments of the turbine stage calculator. These may be used to validate the results from the other methods or in the rare cases when the other models do not converge. Generally, they are much slower due to the fact that they solve numerically and with much stricter convergence criteria.

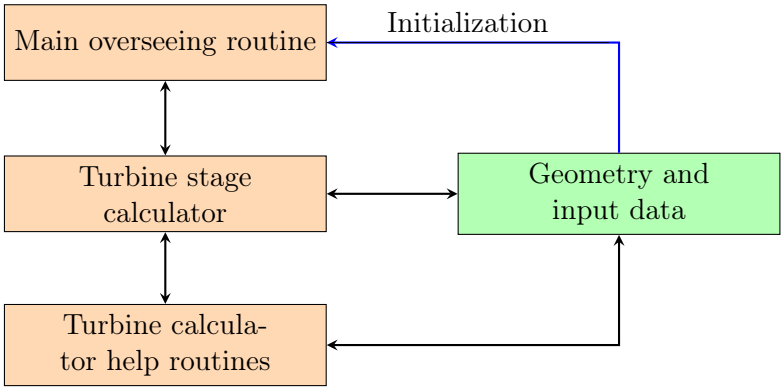


Figure 13: Data flow paths between the program files for a single operating point.

3.6.1 Running a single case

In order to run the program for single operating points, the function `run solver` is used. The function formally requires four arguments to define and start calculations, by integer and text-strings. These are:

1. geometry file: Location and name of geometry data file (`str`)
2. boundary file: Location and name of initialization data file (`str`)
3. solver type: Pressure method, presently 1 for Denton and 2 for Came (`int`)
4. substance: Turbine fluid: normally air or parahydrogen (`str`)

With this information, the program loads and unpacks data to a pandas DataFrame from the input file locations. An initial linear pressure guess is then made based on the amount of stages in the turbine. The program then has all necessary parameters to call the pressure solver function.

3.6.2 Pressure solver implementation - Denton's method

The actual algorithm for pressure solving is located in the sub-function `Denton`. This function requires the following arguments:

1. geo data: Unpacked geometry (`DataFrame`)
2. init data: Unpacked initialization (`DataFrame`)
3. solver: Solver type number from input (`int`)
4. $P_{TE,init}$: Initial axial pressure distribution guess (`float vector`)
5. substance: Turbine fluid from input (`str`)

The program then initializes `main iteration` $= 0$, `convergence` $= False$ and passes data onto the turbine stage calculator to get placeholder results for the first iteration. The placeholder results are used to define a target flow from the first stage's stator outlet and to get a better approximation of the flow losses Δs, both of which are used as input data for subsequent iterations in the actual loop.

The iterative loop uses Denton's [7] equations depicted in (2.87) to (2.91). Figure 14 visualizes the method in a flow chart.

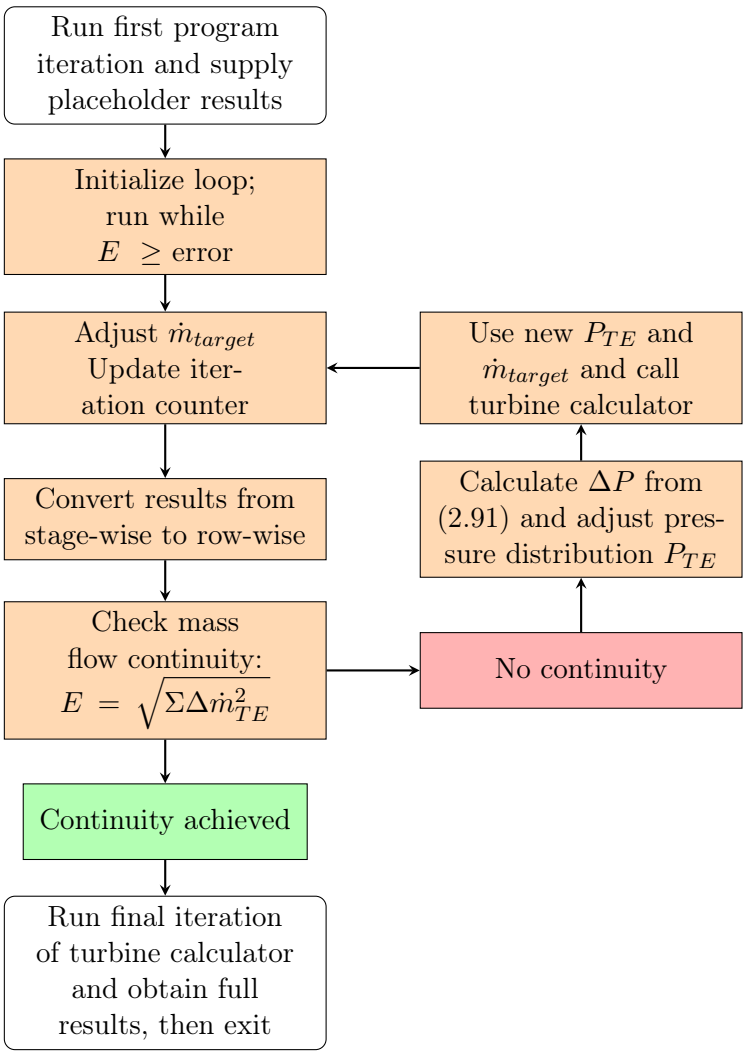


Figure 14: Detailed re-visualization of the iterative target pressure method.

The results need to be converted to row-wise due to Denton's algorithm working on a per-trailing edge basis and since the program returns results stage-wise.

3.6.3 Came's brute force-style solution

This method uses the theory by Came [19] to correct the pressure with a mass flow fraction that is based on the target flow, $\dot{m}_{target}$, of the previous iteration's results. This ratio of mass flows and its influence on the pressure change may be changed using a relaxation factor, as shown in equation (2.92). Methodologically, the algorithm is very similar to the Denton variant. The main difference is the lack of derivatives resulting in more sudden adjustments of the junction pressures, based solely on the flow ratio.

3.6.4 Convergence

When the program achieves convergence, a final iteration of `turbine stage calc` is ran, only this time with a changed input value to `Convergence = True`. This tells the program to save all results in a pandas DataFrame (matrix with named index and named columns) for easier post-processing and data handling. The resulting DataFrame object does at the time of writing contain approximately 50 different parameters from the turbine stage calculator, scaled up by the number of stages. Addition or removal of results is made easy by adjusting the DataFrame object in the turbine calculator.

3.7 Mass testing

While single case studying provides detailed information about an operating point, it is more interesting to study cases where the initialization data consists of varied parameters. This is useful for so-called *turbine characteristics.* For example, it might be interesting to vary the rotational speed but keep the other parameters constant in order to study how the program reacts.

This is made possible in the `run mass test` file. The file can load a matrix of operating points (i.e. varied PRs, inlet pressure, inlet temperature, rotational speed) and uses the pressure solver method from the `main` file until convergence has been achieved. The process is repeated for each case using a `for-loop` and the results (if convergence has been achieved) are stored in a DataFrame for post-processing. The pandas DataFrame functionality is very useful for these tests, as it allows for each case to retain a unique identification index. In addition to the indexing, the initialization data for each operating point is inserted as the first couple of columns in the results DataFrame.

The program also includes an option to store the resulting DataFrame in an excel file.

3.8 Post-processing

The `post processing` file is used to calculate relevant parameters and relate the program output to test data. The program imports values from an excel file and stores them in a DataFrame object. If test data exists for a large number of cases, the program results are best visualized using the plotting capabilities of this file using relative errors. Currently the function has calculation capabilities for deviations pertaining to mass flow, power, pressure, axial forces and efficiency. The pandas package can also be used to pinpoint "bad" results or results with relative errors exceeding a set threshold.

3.8.1 Relative errors

A very important calculation procedure is the relative error which is defined in the program as follows:

$$\text{Relative error} = \frac{\text{Program output} - \text{Test data result}}{\text{Test data result}} \tag{3.6}$$

This related the program output to actual test data and shows how far off the program is from the sought value.

3.8.2 Plotting

The program utilizes plotting made possible by the `matplotlib` package. Relative errors can be calculated for each operating point and plotted versus interesting axes (e.g. pressure ratios or test case numbering).

3.9 Other routines

3.9.1 Pressure distribution

The `radial eq` file uses a radial equilibrium relation by Dejc and Trojanovskij [13] to calculate properties at the hub and tip radius. This is used to try and evaluate the axial loading at rotor inlet which is where its effects are most substantial.

3.9.2 Velocity triangle drawer

A velocity triangle is a 2D visualization of the flow in the tangential and axial plane. The representation includes angles for the relative and absolute velocity field. This code could be used in conjunction with the above pressure distribution to evaluate variation between the hub, mean-line and tip positions of the blades. This file is not part of the general program methodology and is only used for demonstrative purposes.

4 Results and discussion

Obtaining trustworthy results from a mean-line code is a demanding task. The difficulties arise in trying to validate the output and insuring its reliability. Part of this is also to establish when these results are not viable and thus not applicable.

4.1 Turbine test data

The complete program validation consisted of two main phases - 1 and 2 - shown below. This was done using existing data from three different physical turbines.

1. Turbine A (LTH) - air flow
2. Turbine B (GKN) - air and parahydrogen flow
   Turbine C (GKN) - parahydrogen flow

Phase 1 was further subdivided into subroutine validation plus complete program validation using mostly single turbine operating points. Phase 2 validation on the other hand was conducted with large amounts of data with a wide range of operating points to try and capture behavior that was not specifically modeled for initially.

4.2 Phase 1 - Subroutine validation

During construction, the program's subroutines were continuously validated using existing test data to ensure expected results with certain parameters. In the case of code translation, the results of the subroutines have also been compared to those of the previous language's. Continuous validation also played a big role in ensuring future stability in the program since many bug fixes were implemented.

4.2.1 Blade outlet angle model validation

The program's extensive outlet angle model was tested against existing results of Turbine A as well as variation of indata parameters to ensure expected behavior. By varying the Mach number of a 2-staged turbine, the plot in figure 15 is made. The resulting lines show continuity and the expected decrease in outlet flow angle as the outlet Mach number exceeds unity.

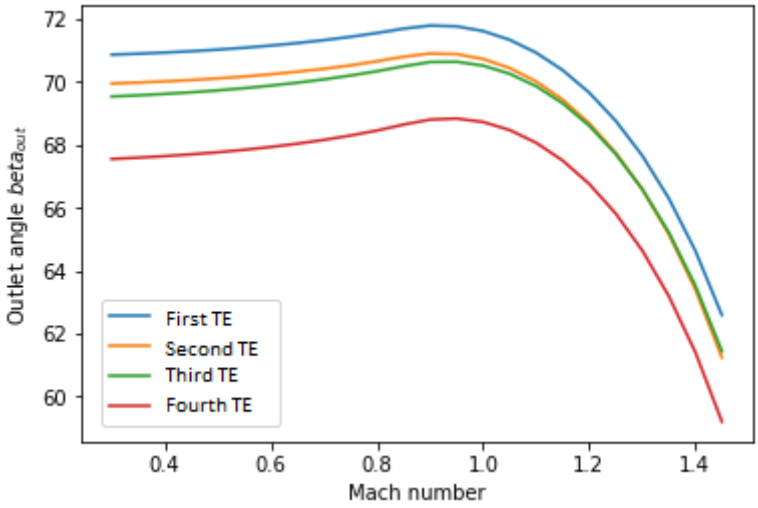


Figure 15: Influence of varied outlet Mach number on outlet angle for a 2-staged turbine.

4.2.2 Aerodynamic losses validation

The aerodynamic loss model based on AMDCKO. et. al. was translated from its MATLAB counterpart. The results from the MATLAB model had been tested against CFX simulations for special design (low AR) tweaking purposes and it is against these results that the Python model has been validated. Figure 16 shows the relative difference for the total loss coefficient and 3 of its loss components: profile, secondary and clearance, using 16 different test cases with varying geometries.

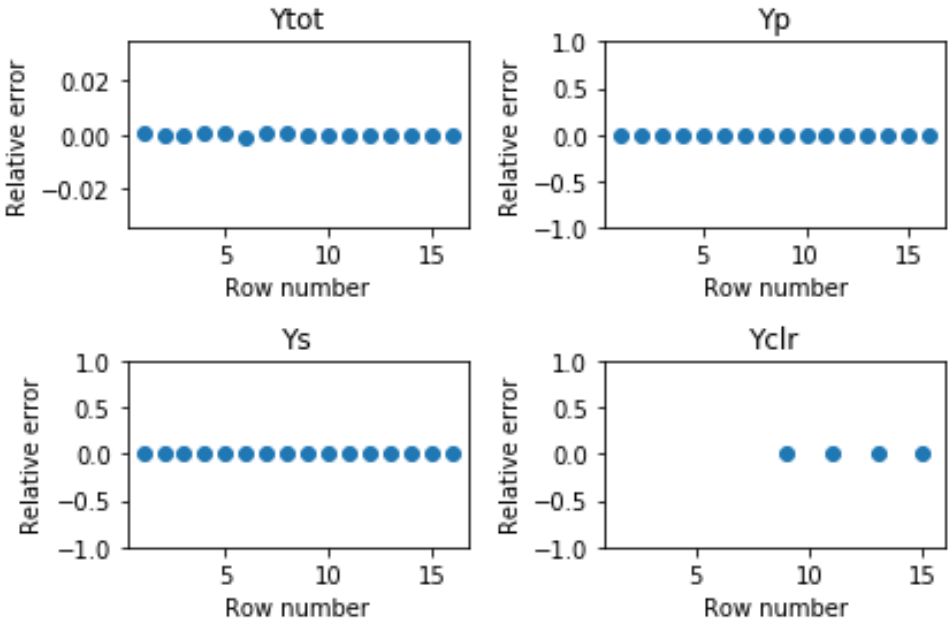


Figure 16: Python results related to MATLAB results.

The results show next to identical predictions between the two programs. The actual numerical difference between the two program outputs are on the order of `1e-5` which could be explained by rounding when comparing values.

4.3 Phase 1 - Complete program validation

The majority of the early program validation consisted of achieving similar flow values and qualities to already existing data. Turbine A was used for this sake and featured a two-stage configuration with pure air flow due to lacking access to classified GKN turbines. The first task at hand was to ensure that the input of geometry was valid by forcing the program to work with constant losses and a "correct" axial pressure distribution. With these boundary conditions the program should realistically predict very similar mass flows through each station of the turbine purely through physics. Adequate flow qualities such as angles and velocity components should naturally follow this.

4.3.1 Turbine A validation

Table 2 shows a comparison between various velocities, angles and loss components for two different cases. The reference value column shows test data at a normalized value of 1. This data has an accompanying axial pressure distribution, which was tried in the program. The *Fake output/ref* column shows how the program performs with this "fake" pressure distribution and the results are mostly within one percent error for the velocities and angles. The other column shows the program's output with the, from Denton's method, correct pressure distribution. The true program output has a continuous mass flow through the turbine whereas the fake output varies. This is due to the program being force fed pressures, as opposed to finding them on its own through mass flow continuity.
The Y loss components also varies significantly - this is likely due to using differing loss models. The loss model in this program is tailored to fit low AR turbines.

	Reference value	Fake output/ref		True output/ref	
Stage	-	1	2	1	2
C_1	1	1.002	1.007	1.010	1.021
C_2	1	1.010	1.005	0.997	0.990
C_3	1	1.008	1.001	1.022	1.013
α_2	1	0.999	0.999	0.999	0.998
β_3	1	0.999	1.001	1.001	1.002
$C_{\theta 2gap}$	1	1.037	1.003	0.996	0.988
W_3	1	0.999	0.998	1.012	1.009
Y_{stator}	1	0.769	0.785	0.592	1.503
Y_{rotor}	1	1.133	1.122	1.127	1.117
$\dot{m}$	1	1.01	0.999	1.009	1.009

Table 2: Comparison of program output to old test data, with two different axial pressure distributions.

4.3.2 Target pressure validation

Table 2 showed that the program predicts decently well for a 2-staged "normal" turbine despite being tweaked to handle more niche variants. To achieve those results, the Denton method was used and its convergence plot typically looks like figures 17 and 18.

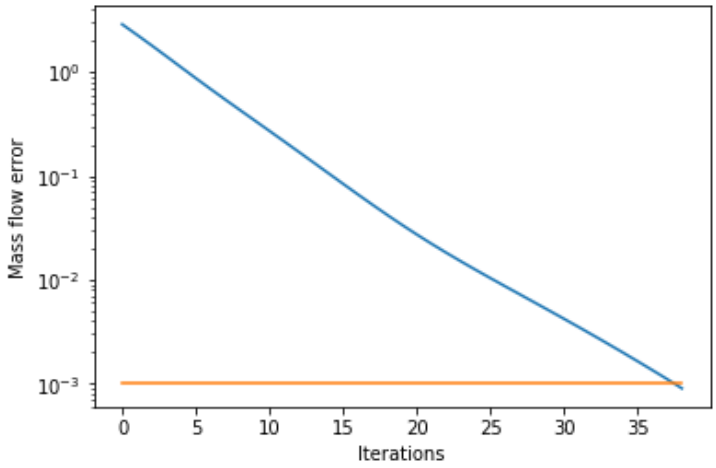


Figure 17: Denton's method: convergence plot with a convergence limit of $1e-4, t \approx 8s$.

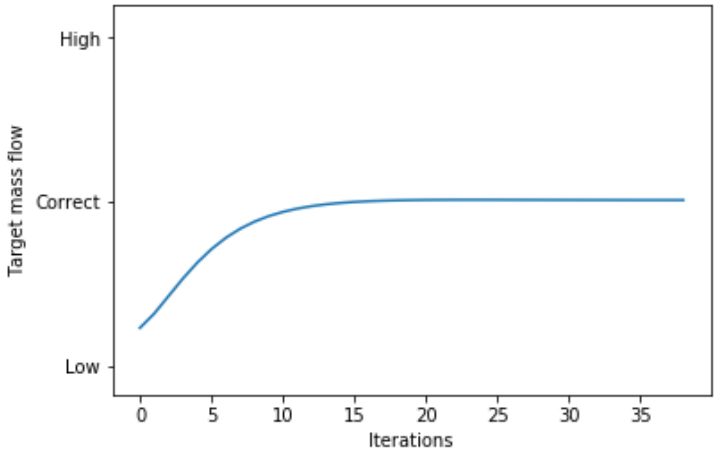


Figure 18: Denton's method: mass flow variation with a convergence limit of $1e-4, t \approx 8s$.

The same case is studied with Came's method, shown in figures 19 and 20.

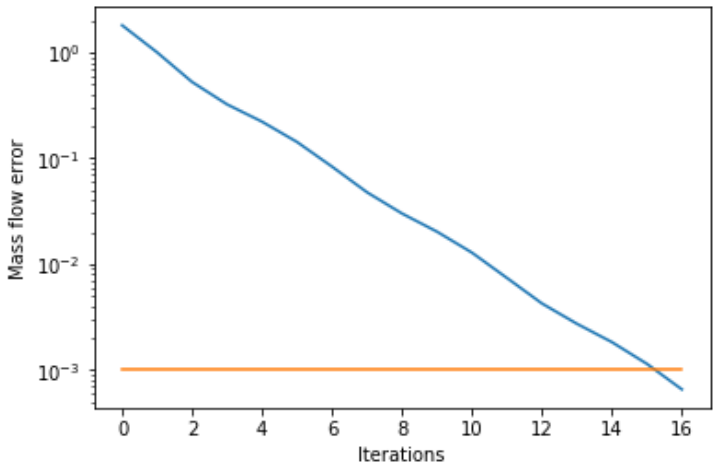


Figure 19: Came's method: convergence plot with a convergence limit of $1e-4, t \approx 4s$.

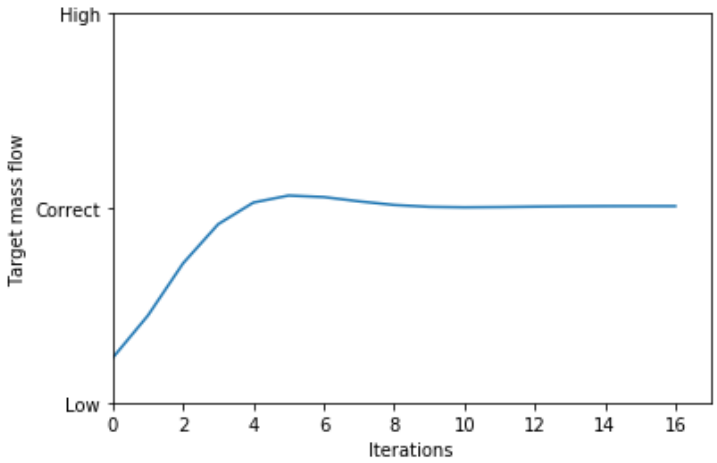


Figure 20: Came's method: mass flow variation with a convergence limit of $1e-4, t \approx 4s$.

The two solutions provide an axial pressure distribution within $10Pa$ of each other at every station with a convergence limit of $1e-4$. This translated to a roughly $500ppm$ deviation at most. Came's method also appears to be faster; spending half the time to reach convergence. By comparison of the target flow plots it can also be seen that it is more aggressive in its adjustments whereas Denton's method displays a smoother converging line.

4.4 Phase 2 - Large scale testing

After sufficient reliability was achieved with the available LTH turbine, it was time to move on to the space turbines and implement functionality for mass testing. The two tested turbines B and C both feature a one-stage configuration. The pressure results from the program is compared to a sensor located at the blade tip and since the program performs calculations at the mean-line, the output of the program has been adjusted to account for this using radial pressure distribution according to equation (2.30). This is done for all cases as most of the sensor data is taken from tip positions. Initially, the program did not achieve meaningful results or lacking convergence for some cases which prompted tweaks outlined in detail in the methodology section. The added tweaks were

1. Traupel's high-clearance blade outlet flow angle correction
2. Flow angle β_{gauge} adjustments

As will be shown in the results, the two tweaks were used independently and together to try to match test data better. Unless otherwise stated, errors show the relative difference from program output to existing test data using equation (3.6) and not actual test data values. In the event of concrete values, these have been scaled arbitrarily for the sake of confidentiality. In cases of varying x-axes, the variation of PR is not connected to test case number.

4.5 Phase 2 - Turbine B testing

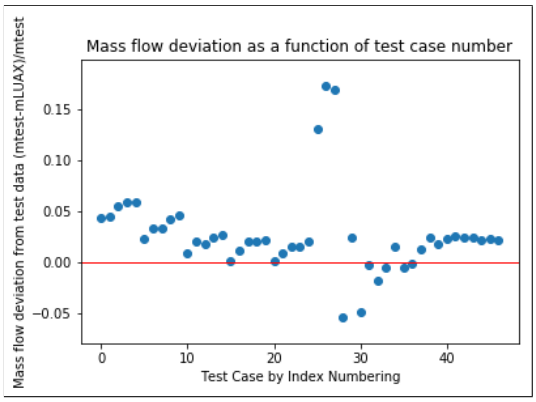
Results from simulations with turbine B was tested against data with two adaptations. The first analysis features turbine B with a flow of air. The second edition of tests used parahydrogen, the original working fluid.

Data on inlet mass flow was available and pressure sensors were in place at tip locations which allowed for initial comparisons for the air turbine, totaling around 50 test cases. For the other case with parahydrogen flow, additional test data was available and thus more comparisons could be made. In total, approximately 90 performance points were used for this purpose. In all of the following plots, each point represents a test case.

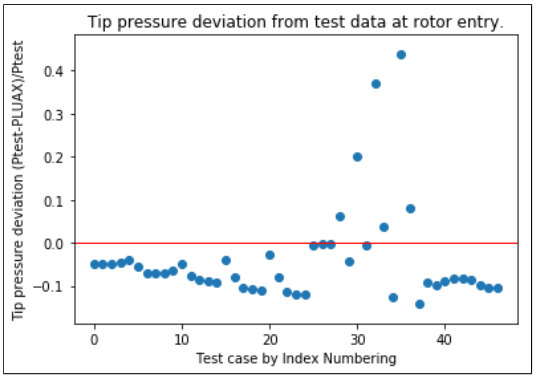
Each test case has its unique setup of *total inlet temperature, total inlet pressure, pressure ratio, rotational speed and outlet pressure*. Each of these data sets corresponds to a certain axial pressure distribution, mass flow, power and more - most of which had existing data to be compared against. The results are from different revisions of the program with varying degrees of tweaks.

4.5.1 Turbine B - Air flow, untweaked

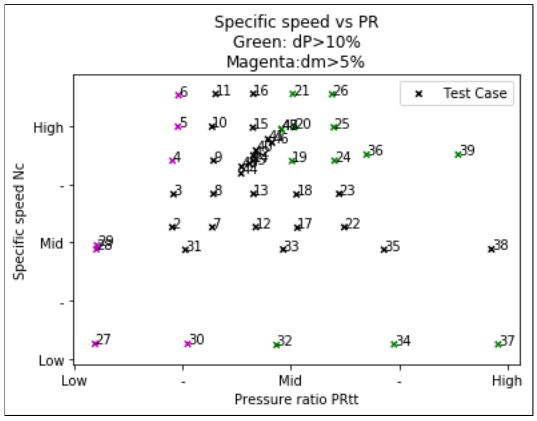
This adaptation of turbine B shares the same fluid flow as turbine A, air, and was the first natural result to study, displayed in figure 21. The mass flow shows a decent correlation, mostly staying within 5 percent deviation but slightly shifted up suggesting overprediction. The pressure prediction is of an opposite nature; slightly weak in the sense that it underpredicts. Both the mass and pressure plots however show a similar decreasing trend and stabilizing somewhat at the end. (c) shows an operating point map of specific speed vs PR. A few points deviate which is also shown in subplot (c) where pressure deviations of over 10 % and flow deviations of over 5 % are plotted in green and magenta, respectively, along with their test case number as shown by the x-axis in (a) and (b). Some of these highly deviating points correlate to low values of the specific speed and extremer values of the pressure ratio outside of the program's standard operating points.



(a) Relative inlet mass flow deviation vs test case



(b) Pressure deviation at rotor entry vs test case

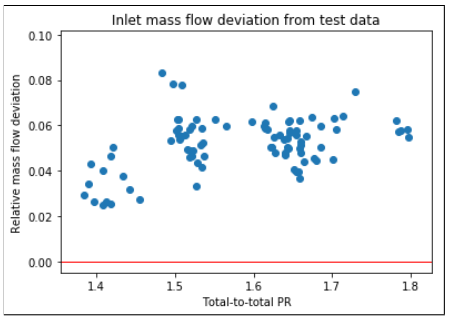


(c) Specific speed vs PR with test cases

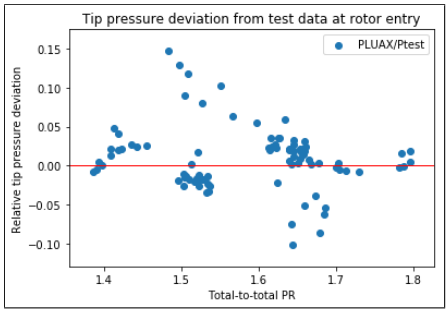
Figure 21: Results from turbine B with air flow showing deviations in predicted mass flow, pressure and a operating point map of specific speed vs PR.

4.5.2 Turbine B - Parahydrogen flow, untweaked

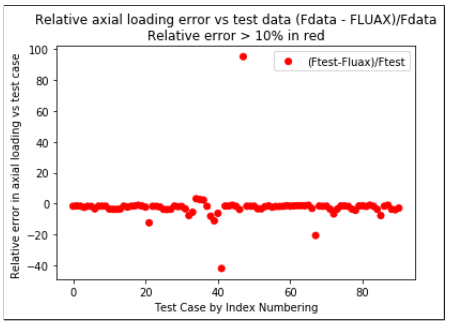
The graphs below show parahydrogen flow without Traupel or gauge modifications. From (a) of figure 22 it can be seen that the program slightly overshoots mass flows in the 2 to 8 % range. The pressure calculations seem to be more stable along $y = 0$ however with some worrisome behavior for certain test cases with relative errors exceeding $|10|\%$. The total span of pressure error seems to be about $|dP_{max} - dP_{min}| \approx 25$ percentage units. It is desirable to even out the error in inlet mass flow with regards to its position to $y = 0$. It is also seen from (c) that the axial force model is entirely unreliable as no value is predicted within an accuracy of 10 %. The efficiency on (d) however shows decent correlation for some pressure ratios with a trend towards higher values. Essentially most of these results are inconclusive which prompts new configurations with the mentioned tweaks.



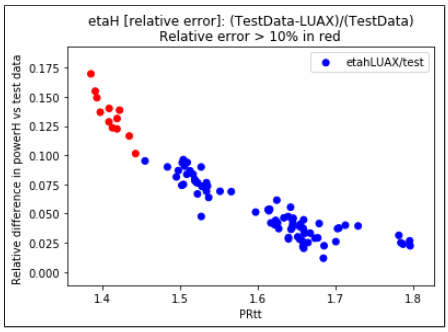
(a) Relative inlet mass flow deviation vs PR



(b) Relative pressure deviation at rotor entry vs PR



(c) Relative axial force deviation vs test case

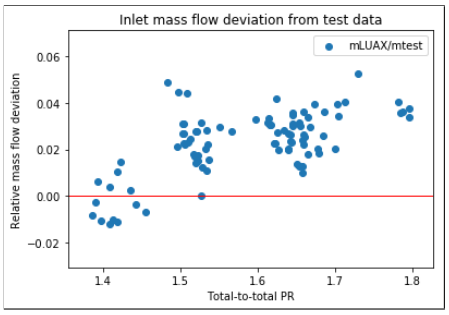


(d) Relative efficiency deviation vs PR

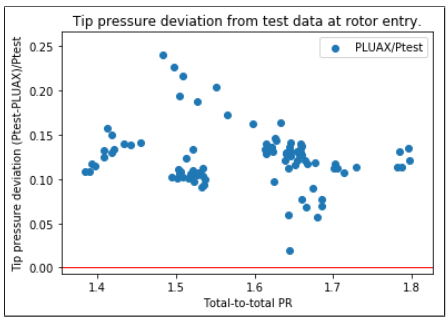
Figure 22: Relative errors in mass flow and tip pressure at rotor entry and renditions of axial force and efficiency plotted against PR and test case number.

4.5.3 Turbine B - Parahydrogen flow, Traupel correction

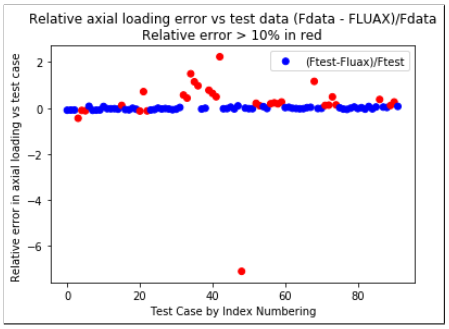
The addition of the Traupel correction at blade outlet is presented in figure 23 by plotting relative mass flow error and rotor entry tip pressure error. The addition of the Traupel correction shows slightly better correspondence to inlet mass flow but with a substantial shift of the pressure compared to without it. The program predicts a higher pressure at rotor entry by about 10 percentage units compared to figure 22 (b). However the program does seem to lower the overall pressure error spread from approximately 2% to 25%, effectively resulting in a 23 percentage unit span. The higher pressure deviation seems to correlate very well with the current calculation model for the axial force as a majority of the data points are within the displayed chosen 10 % threshold. The axial force is calculated using blade hub pressure among other things and the previous data that is being compared to could have been calculated using a different radial pressure distribution.



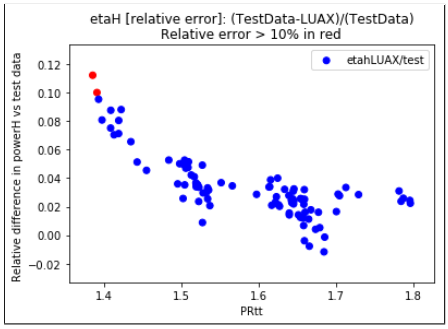
(a) Relative inlet mass flow deviation vs PR



(b) Relative pressure deviation at rotor entry vs PR



(c) Relative axial force deviation vs test case

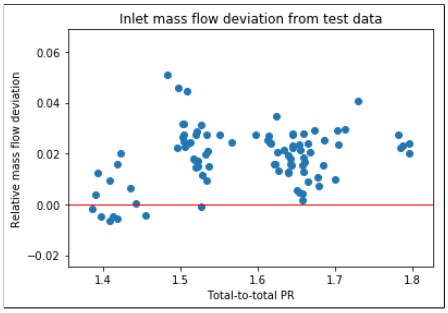


(d) Relative efficiency deviation vs PR

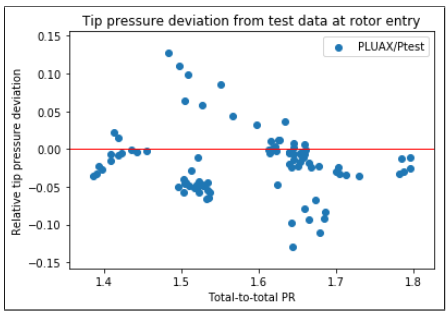
Figure 23: Relative errors in mass flow, tip pressure at rotor entry and renditions of axial force and efficiency plotted against PR and test case number with addition of Traupel angle correction.

4.5.4 Turbine B - Parahydrogen flow, gauge modification

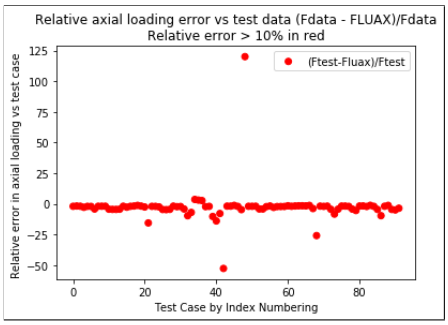
Instead of using the Traupel correction, the gauge angle may be manually modified by a small amount to adjust the flow channel and achieve better mass flow predictions. A modification of $+0.62°$ (chosen from single-case study) shows a much better correlation of inlet mass flow without compromising and shifting the overall spread of pressure deviation, seen in figure 24. For turbine B it would therefore seem that modifications using the gauge angle allows for easy mass flow manipulation. The pressure span seems to have increased to approximately $|dP_{max} - dP_{min}| \approx 28$ percentage units, slightly worse than the two previous results but significantly better than the Traupel tweak due to the pressure being more centered along $y = 0$. Additionally, the relative axial force and efficiency deviation are largely unchanged from the untweaked counterpart. This is expected since the pressure is mostly unchanged and the calculation procedure of axial force and efficiency stems from the pressure.



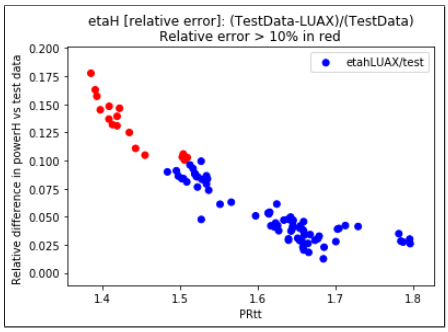
(a) Relative inlet mass flow deviation vs PR



(b) Relative pressure deviation at rotor entry vs PR



(c) Relative axial force deviation vs test case

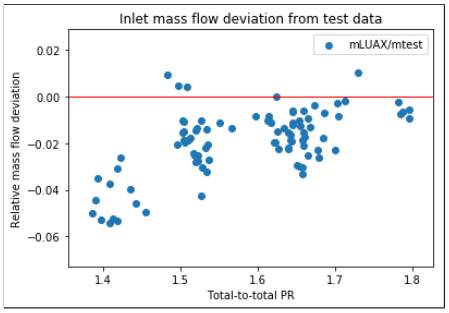


(d) Relative efficiency deviation vs PR

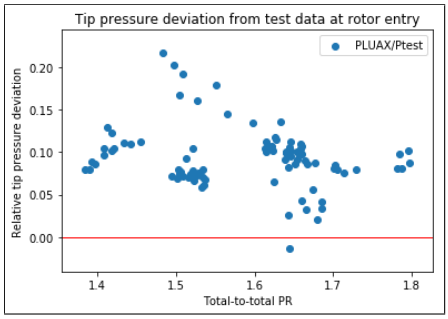
Figure 24: Relative errors in mass flow, tip pressure at rotor entry and renditions of axial force and efficiency plotted against PR and test case number with modified gauge angle.

4.5.5 Turbine B - Parahydrogen flow, Traupel and gauge modification

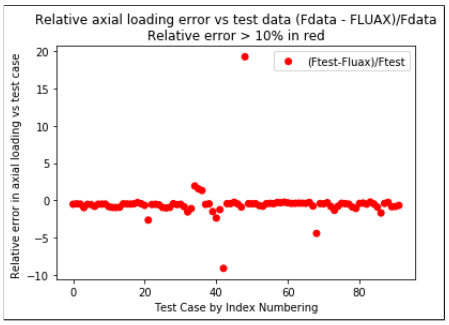
Using both the Traupel and gauge modification of 0.62° yields interesting results for the axial force calculation, shown in figure 25. Previous results would suggest that a combination of Traupel and gauge modification might lessen the deviation in mass flow while establishing pressures that gives decent values for the resulting axial force from (c). These results only seem to inherit reasonable accuracy for mass flow and efficiency shown in (d), almost identical to that of figure 23(d).



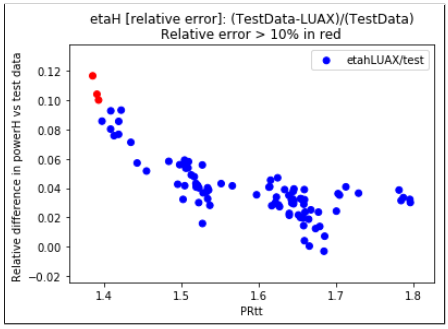
(a) Relative inlet mass flow deviation vs PR



(b) Relative pressure deviation at rotor entry vs PR



(c) Relative axial force deviation vs test case



(d) Relative efficiency deviation vs PR

Figure 25: Relative errors in mass flow, tip pressure at rotor entry and renditions of axial force and efficiency plotted against PR and test case number with modified gauge angle and Traupel correction.

4.5.6 Turbine B - Closing discussion

The addition of gauge and Traupel led to some improvements when studied separately. However the results indicate a trade-off of more accurate mass flow, axial force and efficiency predictions with elevated pressure with Traupel's angle correction.

With the gauge modification only, the program keeps a more $y = 0$-centered pressure prediction like the untweaked case with some improvements in the mass flow deviation. As for the axial force and efficiency, the former behaves inaccurately to test data and the latter shows large deviations for low pressure ratios trending downwards as said ratio increases.

The combination of the two provides no real insight other than that it seems to have broken the axial force calculation, or at the very least failed to improve it like the Traupel-only edition of the program seems to have done.

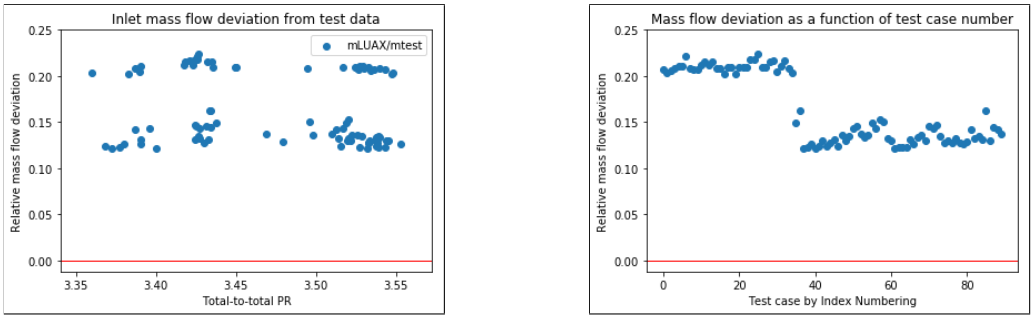
4.6 Phase 2 - Turbine C testing

Turbine C features test data from only one fluid, namely its native parahydrogen. Naturally, this turbine has different operating points and a differing set of geometry. Apart from this, conditions were similar. Approximately 90 test cases were used for validative purposes with the same type of modifications (Traupel correction and gauge modifications).

Early tests showed discrepancies for mass flows; these were quickly identified and are explained by a change of calculation procedure in the middle of GKN's test data gathering campaign. In other words, the first half of the 90 used test cases had its mass flow calculated in a now obsolete way, resulting in an offset cluster of mass flow data points. As will be shown, it was decided to keep these results as the most of the other results did not show this kind of behavior suggesting that subsequent calculations was not a result of this error.

4.6.1 Turbine C - Mass flow anomaly

Figure 26 reveals the relationship between measured test data mass flow and test case number as it drops about 10 percentage units in relative error from circa test 40 onwards. As this is the case for all following LOX results, relative mass flow errors will be plotted against test case number as opposed to the previously used PR.

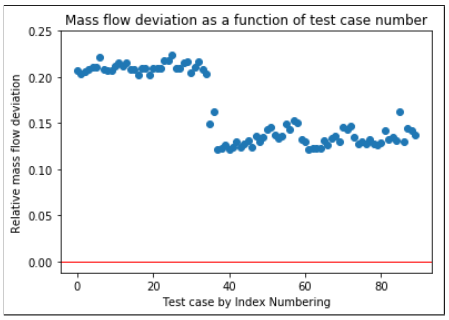


(a) Relative inlet mass flow deviation vs PR (b) Relative inlet mass flow deviation vs test case

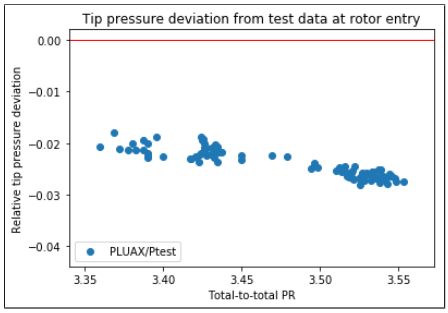
Figure 26: Relative mass flow deviation plotted against pressure ratio and test cases highlighting the aforementioned calculation procedure change.

4.6.2 Turbine C - Parahydrogen flow, untweaked

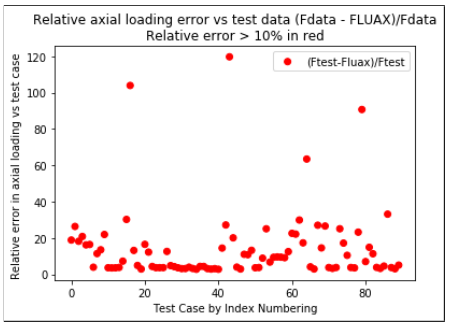
Figure 27 shows a more complete result of the untweaked version of turbine C. It is clear that the pressure predictions fit surprisingly well, with less than 3 % deviation. However, as can be seen by both the mass flow and axial force deviation, there is room for improvement or in the case of axial force, a complete re-assessment. The program predicts decently well for the efficiency but with somewhat of a large spread.



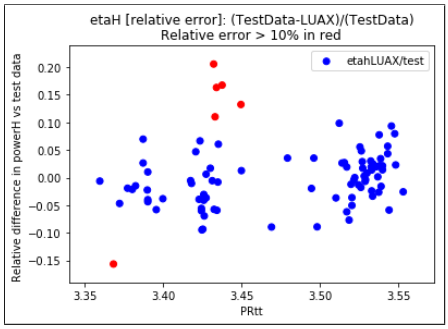
(a) Relative flow deviation vs test case



(b) Relative pressure deviation at rotor entry vs PR



(c) Relative axial force deviation vs test case

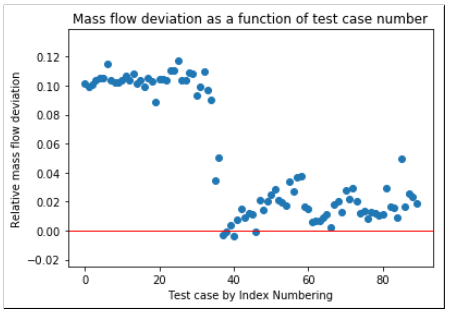


(d) Relative efficiency deviation vs PR

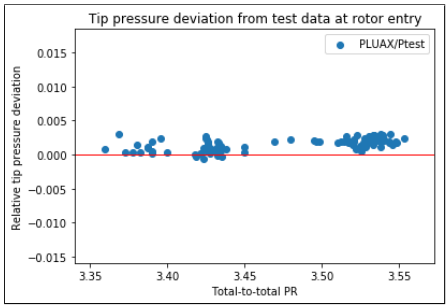
Figure 27: Relative errors in mass flow and tip pressure at rotor entry and renditions of axial force and efficiency plotted against PR and test case number.

4.6.3 Turbine C - Parahydrogen flow, Traupel correction

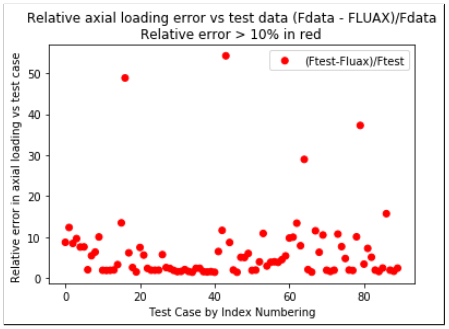
Introduction of the Traupel correction leads to significant improvements of mass flow (disregarding the anomaly of the first 35 test cases) and an even better prediction of the tip pressure as seen in figure 28(a) and (b). Efficiency seems largely unchanged as expected due to enthalpies being based on axial pressure distribution. Once again, the results for the axial force are unsatisfactory but a slight drop in total error is noted.



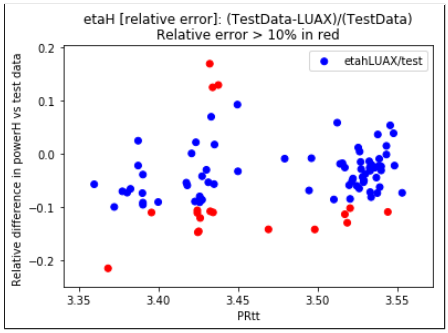
(a) Relative flow deviation vs test case



(b) Relative pressure deviation at rotor entry vs PR



(c) Relative axial force deviation vs test case

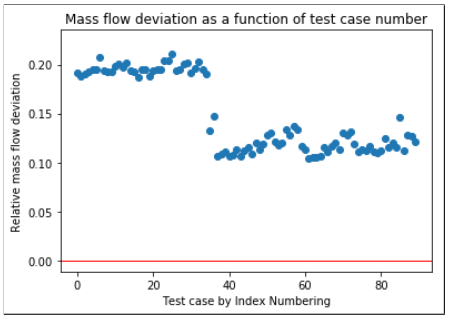


(d) Relative efficiency deviation vs PR

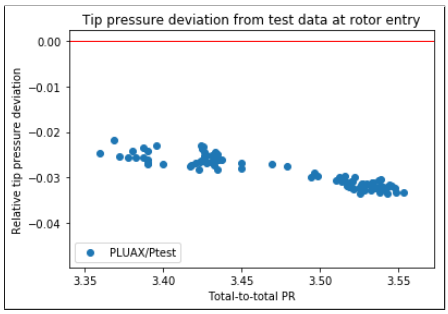
Figure 28: Relative errors in mass flow and tip pressure at rotor entry and renditions of axial force and efficiency plotted against PR and test case number with addition of Traupel correction.

4.6.4 Turbine C - Parahydrogen flow, gauge modification

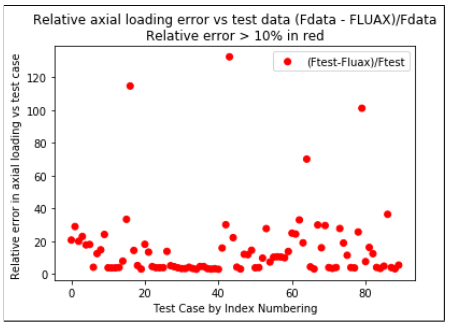
Removing the Traupel correction and only working with a similar modification of 0.62° of the gauge results in principially identical results to the untweaked version with small variations in mass flow prediction in figure 29(a) in the form of a few percentage units shift towards $y = 0$ and a less pronounced down-shift for the pressure prediction (b). Efficiency similar to untweaked case. Axial force deviation leaves much to be desired.



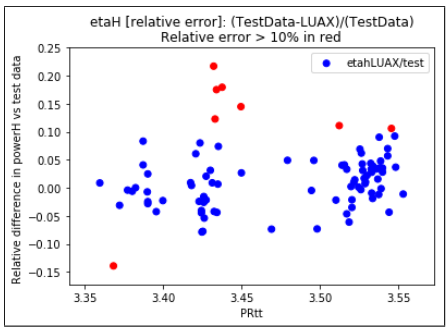
(a) Relative flow deviation vs test case



(b) Relative pressure deviation at rotor entry vs PR



(c) Relative axial force deviation vs test case



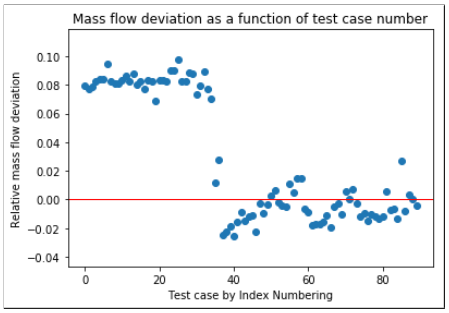
(d) Relative efficiency deviation vs PR

Figure 29: Relative errors in mass flow, tip pressure at rotor entry and renditions of axial force and efficiency plotted against PR and test case number with modified gauge angle.

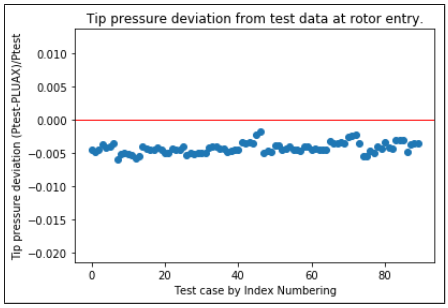
4.6.5 Turbine C - Parahydrogen flow, Traupel and gauge modification

For analysis' sake both tweaks were tested for turbine C show in figure 30. Due to the gauge angle modification's inefficiency in affecting the results, all 4 graphs are very similar to those of figure 28 which only had the Traupel correction. Disregarding the anomaly of (a) shows a very appealing correlation and also for (b), staying mostly within half a percentage relative deviation. Nevertheless, the axial force calculations remain unconvincing and is most

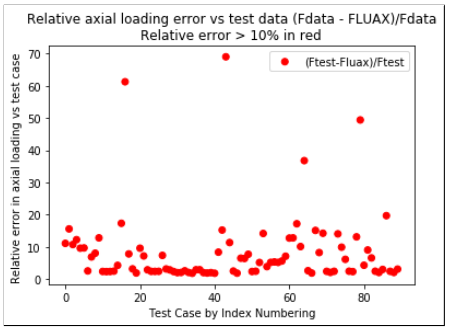
likely in need of an overhaul.



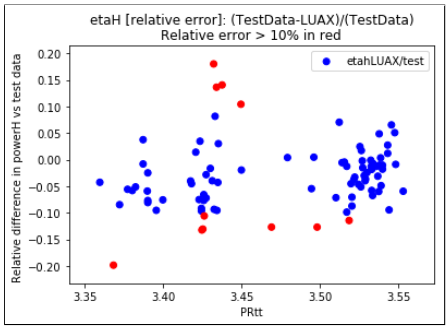
(a) Relative inlet mass flow deviation vs test



(b) Relative pressure deviation vs test case



(c) Relative axial force deviation vs test case



(d) Relative efficiency deviation vs PR

Figure 30: Relative errors in mass flow, tip pressure at rotor entry and renditions of axial force and efficiency plotted against PR and test case number with modified gauge angle and Traupel correction.

4.6.6 Turbine C - Closing discussion

First and foremost, the predictions for turbine C shows very stable pressure correlations to test data and decent correlations for mass flows, especially for the runs where the Traupel correlation is used. Neither of the axial force calculations showed viable results which might be indicative of

(a) faulty radial pressure distribution, or

(b) faulty calculation model

This should be addressed in upcoming studies of the program to ensure continued reliability.

4.7 Sources of errors

The results of this validation study showed some promising data but also lacked in other departments, particularly axial force calculations. Possible sources of errors will be listed and described below.

4.7.1 Meanline methodology

The most obvious source of erroneous behavior is the downscaling to meanline from three dimensions. To fully capture and account for complex 3D fluid- and thermodynamics is unrealistic and also not the intent of this program, as a matter of fact the opposite. But it should go without saying that this is likely the single most substantial reason for hardships in predictions.

4.7.2 Input data

The program bases all of its computations on data input by the user. If some boundary conditions or geometrical data is input incorrectly or wrong assumptions are made, then the results will likely vary greatly. It is therefore very important to define the system correctly.

4.7.3 Sensor and measurement accuracy

Another key component, especially when validating, is accuracy of sensors and other measurement techniques. Hypothetically speaking, a sensor could have a $\pm 1 - 3$ % accuracy which theoretically could mean that if the program calculates "correctly" for two points while these two points have been measured as $+3\%$ and -3% respectively, this would effectively result in a 6 % error span. This does not only apply to validation though, since the program requires boundary condition information on static properties at rotor outlet when using total-to-total pressure ratios as opposed to a total-to-static pressure ratio. This would mean that the uncertainty propagates right from the start.

4.7.4 Model accuracy

The thoughts and reasonings from the previous segment continues with model accuracy. Although scientists and researchers have done an outstanding job developing models for reality, the fact of the matter is that that is what they remain - *models.* Even if the models predict well within their intended area, inconsistencies may present themselves at extremer conditions. Since this program is intended to be used for off-design analyses and characteristics, it is very important to understand the limits and reliability of the models used. After all, a chain is only as strong as its weakest link and the same concept can be applied to this program.

4.7.5 Total-to-total pressure ratio

The usage of total-to-total pressure ratios raises some concern - in the absence of valid data on the static properties of rotor outlet, the program would need to extend its calculations to include an iterative check at the end of calculations to verify that the actual total pressure at outlet matches that of the boundary condition specified value. This is on account of GKN turbines having specially designed outlets that have not been modeled for.

4.7.6 Iteration criteria

The program relies on iterative calculations. These iterations halt when a convergence limit is reached which means that too loose of a limit could provide results that are not entirely converged. This is somewhat analogous to the weakest link in the chain description used above.

4.7.7 Axial force

The axial force calculations are based on pressure differences before and after the rotor disc. Specifically hub pressures which are derived using Dejc and Trojankovskij's [13] model on radial equilibrium. It is possible that this estimation differs too much from previous calculations or sensor values and therefore yields very deviating results.

4.7.8 Programming mishap

Possible however rather unlikely due to continuous validation is the prospect of mistakes in the code.

4.8 Future work possibilities

The program is by no means finished - quite the opposite, its promising results for some cases has been shown that this is just its inception. In other words, there is a plethora of future work possible.

4.8.1 Continued validation

Figure 26(a) showed the calculation anomaly as as a function of PR. In future testing, it could be of interest to study only the cases with the updated calculation procedure as some of the results are based on mass flows. Other interesting cases for both turbine B and C are those deviating significantly from other values. Gaining more insight as to why these values don't converge as well as the others could be beneficial for when performing turbine characteristics. A suggestion is sensitivity analyses.
Part of the continued validation is also making sure that post processing is done correctly with regards to equations used and methodology.

4.8.2 Characteristics

Additions of tweaks have shown that the results can be manipulated to better fit test data. Evidently, most notably for turbine C, the carefully tuned tweaks resulted in superb correlations to mass flows and pressures from test data. This increases confidence in the program's ability to predict performance and lays an acceptable foundation for turbine characteristics with fictional operating points. The other points (axial forces and efficiency) should be addressed however, specifically the former to find an explanation for the anomalous behavior.
It is recommended to change to a total-to-static pressure ratio when defining the system for characteristics as this eludes the need for an iterative pressure ratio loop which currently does not exist in the program. The total-to-total pressure ratio may be calculated after convergence by applying constant losses to the total pressure after outlet.

4.8.3 Varying gauge modification and usage of Traupel

Continuing from the previous point - further testing of the influence of Traupel and gauge angle modification is needed to evaluate its impact and more importantly its applicability in different scenarios. It could also be of interest to see (if) any dependencies on specific variations of input parameters.

4.8.4 Code modularization and speed ups

The current coding state of the turbine stage calculator is at times rather linear and straight forward. For increased readability, it could be modularized by moving calculations into smaller subfunctions (for example: entropy, mix and gap iterative loops). The concept of object oriented programming is also not

prominent and the program could theoretically be rewritten to accomodate for it.
Possible speed ups could be addressed when using CoolProp; the program could be studied to try and use T and ρ as much as possible as these are the variables that are used in the equations of state according to CoolProp documentation [8]. T and P is 3-10 times slower than previously mentioned. Other combinations without T or ρ are much slower, potentially slowing down the program due to its iterative nature.

5 Conclusion

In the end, the program managed to establish some valid correlations to existing test data. The continuous validation technique also proved that sub routines were working as expected independently and together. It was also shown that by introducing regulating models such as Traupel's angle correction for high-clearance blades and angle modifications, results could be tailored to better fit said test data. The results also imply that program output for turbine C had a much better overall precision for mass flow and pressure predictions than turbine B. Fixes were implemented to mitigate crashing issues which has resulted in a decently reliable and stable tool for early stage turbine design.

With that in mind, it is deemed that continued testing is essential in further validation of the code. The most perplexing results are those related to the axial force calculations. It could prove to be wise to challenge the current axial force calculation procedure, especially for turbine C where the pressure prediction (on which axial force is based) most certainly is as good as it realistically could be.

Implementation of turbine characteristics based on theoretical values such as total-to-static pressure ratio is also well within the scope of the program's capabilities. Additions of more models or own modifications is possible due to the program's structure.

Given the time frame and resources available I would like to claim that this has been a successful thesis and hope that the program continues to be evaluated, further developed and pushed to its absolute limit.

References

[1] GKN Aerospace. *GKN*. URL: https://www.gkn.com/aerospace/ (visited on 04/07/2019).

[2] NASA. *Liquid Hydrogen–the Fuel of Choice for Space Exploration*. URL: https://www.nasa.gov/content/liquid-hydrogen-the-fuel-of-choice-for-space-exploration/ (visited on 04/09/2019).

[3] H. Cohen P. Straznicky H. Saravanamuttoo G. Rogers and A. Nix. *Gas Turbine Theory*. Seventh Edition: Pearson Education Ltd, 2017.

[4] D. G. Ainley and G. C. R. Mathieson. "A Method of Performance Estimation for Axial-Flow Turbines". In: *Aeronautical Research Council* 2974.December (1957).

[5] N. Baines H. Moustapha M. Zelesky and D. Japikse. *Axial and Radial Turbines*. Concepts ETI, 2003.

[6] B. I. Mamaev. "A Method for Calculating Exit Angles of Flow through a Turbine Blade Row". In: *Thermal Engineering* 47.2 (2015), pp. 121–127. URL: https://www.researchgate.net/publication/291925252_A_method_for_calculating_exit_angles_of_flow_through_a_turbine_blade_row.

[7] J. D. Denton. "Throughflow Calculations for Transonic Axial Flow Turbines". In: *Journal of Engineering for Power* 100.April (2015), pp. 212–218.

[8] CoolProp. *Welcome to CoolroProp - CoolProp 6.2.1 documentation*. URL: http://www.coolprop.org/ (visited on 05/16/2019).

[9] Spyder. *Spyder Website*. URL: https://www.spyder-ide.org/ (visited on 05/24/2019).

[10] E. W. Lemmon et al. *NIST Standard Reference Database 23: Reference Fluid Thermodynamic and Transport Properties-REFPROP, Version 10.0, National Institute of Standards and Technology*. 2018. DOI: https://dx.doi.org/10.18434/T4JS3C. URL: https://www.nist.gov/srd/refprop.

[11] University of Calgary. *Gas turbine - Energy Education*. URL: https://energyeducation.ca/encyclopedia/Gas_turbine (visited on 05/05/2019).

[12] National Aeronautics and Space Administration. *Enthalpy*. URL: https://www.grc.nasa.gov/www/k-12/airplane/enthalpy.html (visited on 05/09/2019).

[13] B.M. Trojanovskij M.E. Dejc. *Untersuchung und Berechnung axialer Turbinenstufen*. VEB Verlag Technik, Berlin, 1973.

[14] S. Ausmeier W. Bitterlich and U. Lohmann. *Gasturbinen und Gasturbinenanlagen*. Teubner Verlag, 2012.

[15] Walter Traupel. *Thermische Turbomaschinen*. Springer-Verlag Berlin Heidelberg, 2001.

[16] Leslie Fielding. *Turbine Design - The Effect on Axial Flow Turbine Performance of Parameter Variation*. New York: ASME Press, 2000.

[17] D. Olsson. "LUAX-T". In: *Lund* (2008).

[18] R. Aungier. *Turbine Aerodynamics - Axial-Flow and Radial-Inflow Turbine Design and Analysis*. New York: ASME Press, 2006.

[19] P. M. Came. "Streamline Curvature Throughflow Analysis of Axial-Flow Turbines". In: *VDI Berichte* 1185 (1995), pp. 291–308.

[20] Wes McKinney. *Data Structures for Statistical Computing in Python*. 2010. URL: `https://pandas.pydata.org/` (visited on 06/01/2019).

[21] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. Online. 2001. URL: `http://www.scipy.org/` (visited on 06/02/2019).

Appendices

A Geometry indata

`geometry_filename.csv`

var	row0	row1	row2	row3	Description, unit
r_hub_LE	0.00	0.00	0.0	0.0	Leading edge hub radius [m]
r_hub_TE	0.0	0.0	0.0	0.0	Trailing edge hub radius [m]
r_tip_LE	0.0	0.0	0.0	0.0	Leading edge tip radius [m]
r_tip_TE	0.0	0.0	0.0	0.0	Trailing edge tip radius [m]
nb	1	2	3	4	Number of blades
chord	0	0	0	0	Blade chord [m]
TEt	0	0	0	0	Trailing edge thickness [m]
OP	0	0	0	0	Throat opening [m]
RPM	0	0	0	0	Rotational speed [RPM]
metal_angle_out	0	0	0	0	Blade metal outlet angle [deg]
metal_angle_in	0	0	0	0	Blade metal inlet angle [deg]
wedge_out	0	0	0	0	Wedge outlet angle [deg]
thickness	0	0	0	0	Blade thickness/chord
X_LE	0	0	0	0	Leading edge x-coordinate [mm]
X_TE	0	0	0	0	Trailing edge x-coordinate [mm]
clr	0	0	0	0	Blade clearance [mm]