Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Kristoffer Lehre Fagerheim

Educational Analysis-Program for Stirling Engines

TMM4935

Master's thesis in Engineering and ICT
Supervisor: Bjørn Haugen
June 2022

NTNU
Kunnskap for en bedre verden

Kristoffer Lehre Fagerheim

Educational Analysis-Program for Stirling Engines

TMM4935

Master's thesis in Engineering and ICT
Supervisor: Bjørn Haugen
June 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU

Kunnskap for en bedre verden

Preface

This project was completed in the course "TMM4935 - Industrial ICT, Master's Thesis" at **NTNU - Norwegian University of Science and Technology** for the Department of Mechanical and Industrial Engineering.

I would like to thank my supervisor Bjørn Haugen for providing assistance with the project throughout the course, as well as answering questions regarding the project.

Abstract

The goal of this project was to create a modifiable and expandable analysis program for Stirling engines that could be used for educational purposes. To achieve this, the program lets the user enter the input-data used for Schmidt analysis and adiabatic analysis. The results are then visualized through plots with markers that are synchronized with an animation of the Stirling cycle. Lastly, the results and plots are stored as CSV- and PDF-files, respectively.

The report begins with an introduction, followed by establishing the theoretical background used during the development of the program. This includes information regarding Stirling engines, such as components, configurations, and benefits and limitations, as well as Schmidt analysis and adiabatic analysis.

Then, there is a thorough explanation of the program and its functionality. This is followed by an explanation of the development approach that includes a description of which measures were taken to achieve the project's goals.

The structure of the main component of the program is then explained with a modified class diagram before describing each window in the program's GUI and its functionality. Next, a numerical example is then used to show the results of the analysis methods and the corresponding plots.

The next section describes the technological-, theoretical-, and design decisions which were made during the development of the program, and the outcome of each decision is evaluated in relation to the goals of the project. Additionally, possible expansions to be added to the program in future are also discussed.

Finally, the main points of the report are summarized and it is established whether the program achieved the goals set in this project.

Contents

List of Figures **4**

List of Tables **4**

Nomenclature **5**

Glossary **6**

Acronyms **6**

1 Introduction **7**

2 Theory **8**
2.1 Stirling engines . . . 8
2.1.1 Components . . . 8
2.1.2 Configurations . . . 9
2.1.3 Benefits and limitations . . . 9
2.2 Schmidt-analysis . . . 10
2.2.1 Assumptions . . . 10
2.2.2 Calculation . . . 11
2.3 Adiabatic analysis . . . 13
2.3.1 Assumptions . . . 13
2.3.2 Calculation . . . 14

3 Method **15**
3.1 Program . . . 15
3.2 Development approach . . . 15

4 Results **16**
4.1 Windows in GUI . . . 17
4.1.1 Intro . . . 17
4.1.2 Manual input . . . 18
4.1.3 Visualization . . . 19
4.1.4 Results . . . 20
4.2 Numerical example . . . 21
4.2.1 Volume variation . . . 22
4.2.2 Pressure variation . . . 22
4.2.3 Mechanical work variation . . . 23
4.2.4 Force variation . . . 23

5 Discussion **24**
5.1 Technological decisions . . . 24
5.2 Theoretical decisions . . . 25
5.3 Design decisions . . . 26
5.4 Future implementations and expansions . . . 27

6 Conclusion **29**

7 References **30**

List of Figures

1 Pressure-volume diagram of a Stirling cycle [12] . . . 8
2 Alpha-configuration of a Stirling engine [27] . . . 8
3 Idealized animation created with VTK . . . 9
4 Simplified illustration of Stirling cycle using Schmidt analysis [28] . . . 10
5 Simplified illustration of Stirling cycle using adiabatic analysis [29] . . . 13
6 Modified class diagram for '*main.py*' . . . 16
7 'Intro'-window . . . 17
8 'Manual input'-window . . . 18
9 'State visualization'-window . . . 19
10 'Results'-window . . . 20
11 Volume variation . . . 22
12 Pressure variation . . . 22
13 Mechanical work variation . . . 23
14 Force variation . . . 23

List of Tables

1 Values required for performing a Schmidt analysis. . . . 11
2 Values required for performing an adiabatic analysis. . . . 14
3 Example data used for numerical example [2] . . . 21

Nomenclature

Subscript

Avg	Average
Com	Compression
Cyl	Cylinder
Exp	Expansion
i	For each component
O	Over
Reg	Regenerator
Rod	Piston rod
R	Difference
U	Under
C	Celsius
Deg	Degree
K	Kelvin
Rad	Radian

Values

β	Phase-angle	rad
A	Area	mm^2
m	Mass	kg
P	Pressure	$\mathrm{N\,mm}^{-2}$
R	Gas constant	$\mathrm{J\,kg}^{-1}\,\mathrm{K}^{-1}$
T	Temperature	K
V	Volume	mm^3
W	Mechanical work	J

Glossary

CamelCase Naming convention where each word is delimited using capital letters [3].

data point Discrete piece of data / information often used for analysis or graphs [5].

framework Set of tools and functions used to build software and systems [21].

object-oriented programming Programming language that divides the code into modules, which may share properties or behaviours [14].

overhead Excess or indirect computation time or resource usage required to attain a specific goal [15].

procedural programming Programming language that follows a set of commands in order [16].

program difficulty The difficulty to use a program and its functionality [25].

progressbar Widget that shows progress as a bar [17].

slider Widget to alter a value by dragging a marker [19].

spinbox Textbox that displays an integer with an up/down-button to increment or decrement the integer-value [18].

widget Addition to a program or application that provides additional functionality [30].

Acronyms

CSV Comma-Separated Values.

GUI Graphical User Interface.

ICE Internal Combustion Engine.

JSON JavaScript Object Notation.

PDF Portable Document Format.

VTK The Visualization Toolkit.

1 Introduction

This thesis is a continuation of the project assignment named "Developing an educational analysis program for Stirling engines" which had a goal of creating an educational analysis program for Stirling engines [7]. During the development of the program the focus was to create a modifiable program that could be used for educational purposes and have the option to be expanded through future implementations. To achieve this goal, the program seeks to provide a good user-experience for both experienced and inexperienced users, as well as future developers.

The program analyzes a Stirling engine by utilizing Schmidt analysis and adiabatic analysis, and then provides visualization of the results. Due to the focus on creating an educational analysis program, visualization of data and the option to view specific steps of the Stirling cycle in relation to the animation are the most integral parts of the program. The engine used for calculation in the program is a 4-cylinder alpha-configured Stirling engine.

This report describes the development of the analysis program and the resulting end product. Its focus is to evaluate the technological, theoretical, and design decisions which were made during the program's development, as well as presenting the background for each decision and how it has affected the final program. Therefore, this report can serve as a foundation for future development.

Firstly, this report provides a description of the theory used in the creation of this program. This consists of first providing a general overview of Stirling engines and relevant key concepts used throughout the report before continuing with describing the theory used for the analysis methods. The report then provides a detailed explanation of the program's functionality and usage before describing the developmental approach to creating the program. Then a class-diagram and a detailed description are shown to explain the structure of the program. The report then presents each window of the program's GUI with a description of its usage and functionality and reviews a use-case using example-data. A discussion and evaluation of the technological, theoretical, and design decisions are then presented in the next chapter. Finally, the conclusion summarizes the main points raised in this report before reviewing whether the project achieved its set goal.

The code to the program is included with this report as a compressed file. This includes the program and its GUI in 'main.py', in addition to the other files which contain additional functionality. Data required for the analyses are stored in the 'assets'-folder, while the results and corresponding plots are stored in the 'results'-folder. An executable of the program is located in the 'output'-folder.

2 Theory

2.1 Stirling engines

Stirling engines are based upon using temperature differences to generate a difference in pressure which moves the piston to one end of the cylinder. The crankshaft then moves the piston back to its original position. It is classified as an external combustion engine, which means it relies on an external power source to provide the temperature difference [13]. Therefore, it can run on any power-source capable to provide the required heat difference.

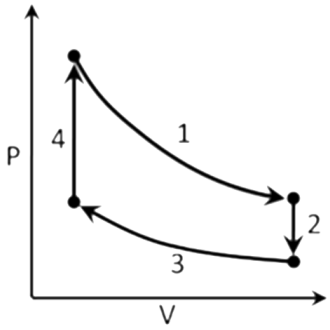


Figure 1: Pressure-volume diagram of a Stirling cycle [12]

The movement of the piston is cyclical and it generates mechanical work by transforming the piston-movement to rotational movement through a crankshaft. Figure 1 depicts a pressure-volume curve for the Stirling cycle which consists of expanding the volume (1), decreasing the pressure (2), decreasing the volume (3), and increasing the pressure (4). The pressure changes depending on the temperature, while a displacer is used to move the transfer medium which increases or decreases the volume. A flywheel is often added to keep the crankshaft's movement continuous.

2.1.1 Components

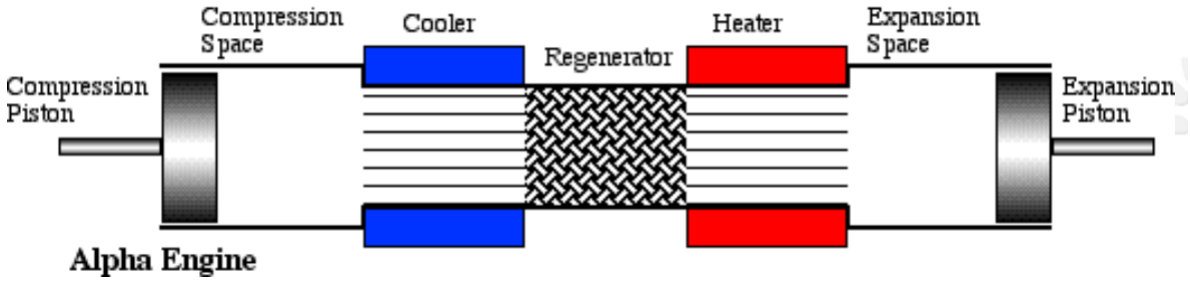


Figure 2: Alpha-configuration of a Stirling engine [27]

A Stirling engine consists of cylinders, pistons, displacers, and a crankshaft. In addition, heat exchangers are used to increase the speed of the heat transfer. A heat exchanger consists of several metal rods enveloped in a transfer medium. When heat is transferred either to or from the heat exchanger, the relatively large surface area of the metal rods increases their ability to transfer heat between the transfer medium [31]. A Stirling engine utilizes two heat exchangers; one which transfers heat from the expansion chamber to the transfer medium, and one which transfers heat from the transfer medium to a heat sink [2]. The price of heat exchangers may increase in price when increasing in size, due to the inherent complexity during assembly.

Another integral part of a Stirling engine is the regenerator, which holds the heat received from heat exchangers until the transfer medium is returned to the hot side of the cylinder [26]. It is situated between the heat exchangers and holds heat which would otherwise be lost as spill heat. While the heat exchangers transfer heat rapidly by using metal rods parallel to the flow of the transfer medium, the regenerator transfers heat slowly by arranging the metal rods perpendicular to the flow instead [2]. This is also illustrated in Figure 2.

2.1.2 Configurations

Stirling engines are generally classified into three main categories according to their configuration; alpha (α), beta (β), and gamma (γ) [6]. Each configuration is defined by their structure, which results in different benefits and limitations. For this program, an adaptation of the alpha-configuration was chosen. Stirling engines using the alpha-configuration have two cylinders coupled in series with a regenerator, and two heat exchangers (heater and cooler) [27], as is illustrated in Figure 2. This is often referred to as the 5-volume approach [24]. It was mainly chosen due to its simple structure which simplifies the visualization and calculations.

Figure 3 shows the program's implementation of an alpha-configured Stirling engine. By comparing Figure 2 and Figure 3, some differences can be observed. While both figures use the 5-volume approach for calculations, the VTK-animation uses an idealized version of the principle where the volume of both the heater and cooler is set to *0* [4]. This can be seen in Figure 3 due to the animation only consisting of a regenerator and two cylinders. The idealization was used to simplify both the calculations and the animation.

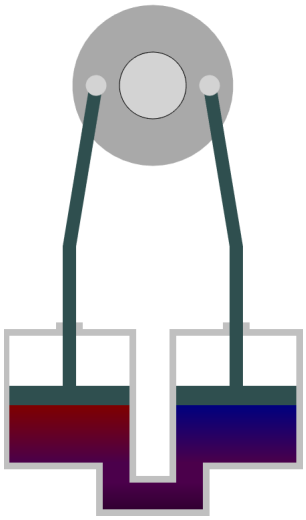


Figure 3: Idealized animation created with VTK

2.1.3 Benefits and limitations

While Stirling engines share several similarities with Internal Combustion Engine (ICE), there are differences that yield different benefits and limitations.

Firstly, Stirling engines are comparatively more efficient when comparing thermal efficiency. Otto- and diesel engines have thermal efficiencies of 25% and 35%, respectively, while Stirling engines reach an efficiency of 40% [1]. This provides a substantial benefit for Stirling engines in relation to resource utilization and optimization of larger systems.

Stirling engines are also able to work in reverse. Normally a temperature difference is added to the Stirling engine which results in mechanical work. Instead, a Stirling engine can receive mechanical work and yield a temperature difference, which enables it to be used as a heat pump or cryocooler [31]. Stirling engines are therefore well-suited for industrial usage where constant temperatures are necessary due to their efficiency. Additionally, Stirling engines can also reverse the direction of the rotation in the crankshaft by changing which side of the cylinder has the highest temperature.

Another benefit with Stirling engines is their ability to run on any fuel that can provide the necessary temperature difference. This makes them highly suitable for applications like utilizing spill-heat or running on fuel sources with variable output, such as solar energy.

Stirling engines are also considered to be relatively quiet, because they require either a temperature difference or mechanical work as input to run. The main source of sound is therefore the movement of the pistons.

It is also a relatively simple construction, due to its few components and working principles. In addition, it also has few movable parts. This results in several benefits, such as improved reliability, prolonged component lifetime, and easier maintenance. Few movable parts also results in less vibrations in the engine.

Due to its working principles, Stirling engines are not able to rapidly change their output, which make them inapplicable for several usages. Inversely, they are well-suited for applications where constant output is desired [1]. Due to its inability to rapidly change its output, it also needs to "warm up" when first starting the engine.

2.2 Schmidt-analysis

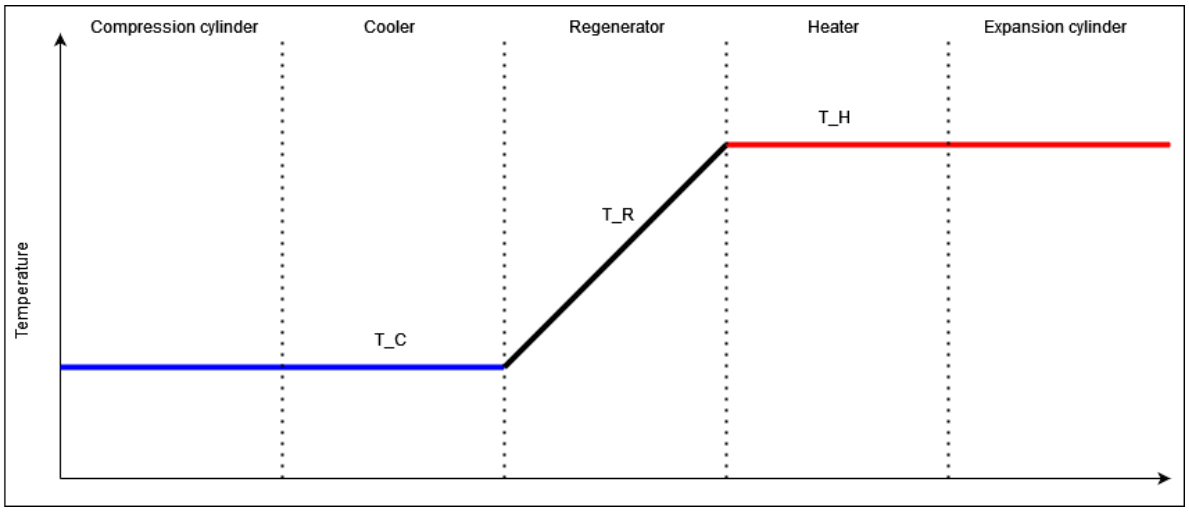


Figure 4: Simplified illustration of Stirling cycle using Schmidt analysis [28]

Schmidt analysis is a method for estimating the output of an idealized Stirling engine by examining the Stirling cycle [28]. While the method only analyzes the thermodynamic process itself and not the rest of the engine, it provides relevant output-values, such as mechanical work, pressure, and piston forces. These calculations can then be visualized and used for further analysis.

There were several reasons why Schmidt analysis was implemented as opposed to other analysis methods. It provided relevant output-values, was relatively simple to implement, and much documentation was available online. The main reason however was that it serves as an initial step for further analysis methods, such as adiabatic analysis and simple analysis [24]. While Schmidt analysis itself does not take several relevant factors into consideration through idealization, these factors are considered in the subsequent analysis methods. Because the goal of this project was to create a modifiable and expandable analysis program, Schmidt analysis provided a solid foundation for expanding the program with additional analysis methods.

2.2.1 Assumptions

Despite yielding relevant output-values, the Schmidt analysis makes several assumptions and does not take multiple factors into consideration. Firstly, the calculations assume that both the heat exchangers and regenerator are idealized and therefore lose no heat during operation. This substantiates the alternative name of the Schmidt analysis, which is "isothermal analysis", and is illustrated in Figure 4. In addition, neither fluid friction nor pumping losses are taken into consideration during calculation [24].

While these assumptions result in less realistic output-values, they are necessary for performing the calculations because they simplify the analysis, and provides a more concise software implementation [9]. The assumptions which were made during implementation are described below.

1. No heat is lost from the system and the process is isothermal.
2. Both the heat exchangers and the regenerator are idealized and do not lose heat.
3. There are no internal differences in pressure.
4. The expansion dead space maintains the temperature of the expansion gas, while the compression dead space maintains the temperature of the compression gas during the thermodynamic cycle.
5. The piston movement is harmonic and follows a sine curve.

2.2.2 Calculation

The implementation of the Schmidt-analysis in the analysis program was inspired by an existing implementation [2]. The values required to perform the analysis are described in Table 1, and in the list of nomenclature.

Symbol	Description
R	Gas constant
m	Mass
θ	Number of degrees in radians
T_{Exp}, T_{Com}, T_{Reg}	Temperatures in kelvin
V_{Com}, V_{Exp}, V_{Reg}	Volume of the components
V_{Cyl}	Swept volume of the cylinder
V_{Avg}	Average volume of the cylinder
A_{Rod}	Area of the piston rod
A_{Cyl}	Area of the cylinder
β	Phase-angle of pistons

Table 1: Values required for performing a Schmidt analysis.

In the program one Stirling cycle is analyzed, which is the movement of the piston from 0° to 360° assuming the piston's movement follows a sine curve. Calculations are performed for every tenth degree.

This program's implementation uses a 4-cylinder Stirling engine. Therefore, the engine has a total of four alpha-configured cylinders working in pair, where each circuit pair is 180° offset from each other, and the pair themselves are 90° offset. This manifests in the plots displayed in Section 4.2.

Before performing the analysis, each data point in the Stirling cycle must be converted from degrees to radians. This is done by using Equation 1.

$$\theta_{Rad} = \theta_{Deg} \cdot \frac{2\pi}{360} \tag{1}$$

Additionally, the temperatures were converted from celsius to kelvin using:

$$T_{\mathrm{K}} = T_{\mathrm{C}} + 273.15^{\circ} \tag{2}$$

The first step of the analysis is to calculate the compressed and expanded cylinder volume as a function of the piston's movement. Equation 3 calculates the compression volume, while the phase-angle β is added to Equation 4 to calculate the expansion volume.

$$V_{Com,i} = V_{Avg} + \sin\theta \cdot \frac{V_{Cyl}}{2} \tag{3}$$

$$V_{Exp,i} = V_{Avg} + \sin(\theta + \beta) \cdot \frac{V_{Cyl}}{2} \tag{4}$$

The ideal gas law is then used to calculate the pressure at each step. Equation 5 is the ideal gas law, and it is rearranged to calculate the pressure in Equation 6.

$$PV = mRT \tag{5}$$

$$P = mR \cdot \sum_i \frac{T_i}{V_i} \tag{6}$$

Next, the sum of the components' temperatures divided by their corresponding volume was calculated using Equation 7.

$$\sum_i \frac{T_i}{V_i} = \frac{V_{Com}}{T_{Com}} + \frac{V_{Exp}}{T_{Exp}} + \frac{V_{Reg}}{T_{Reg}} \tag{7}$$

After having calculated the resulting pressure, the total mechanical work was then calculated by using Equation 8. The mechanical work for each step was calculated by using Equation 9.

$$W = \int P\Delta V \tag{8}$$

$$\Delta W = P \cdot \Delta V \tag{9}$$

The piston forces were then calculated by using Equation 10.

$$P = \frac{F}{A} \iff F = P \cdot A \tag{10}$$

2.3 Adiabatic analysis

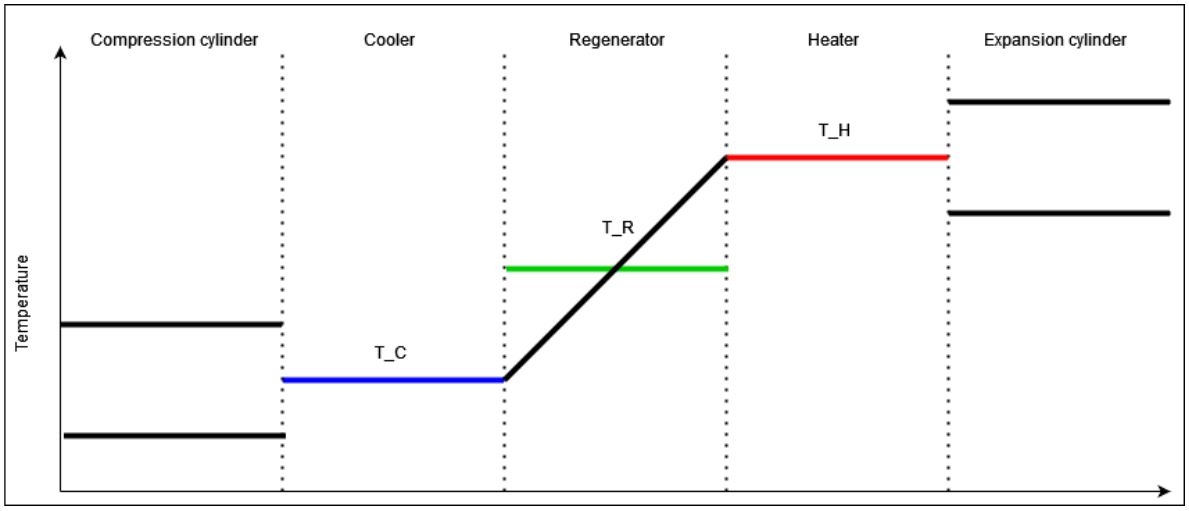


Figure 5: Simplified illustration of Stirling cycle using adiabatic analysis [29]

Adiabatic analysis is a direct continuation of the Schmidt analysis which assumes the Stirling cycle to be adiabatic rather than isothermal [24]. This means that the net heat transfer during a Stirling cycle is provided by the heat exchangers, as opposed to the compression- and expansion cylinders [29]. The difference between the Stirling cycle in Schmidt- and adiabatic analysis can be seen by comparing Figure 4 and Figure 5.

Adiabatic analysis was implemented because it is a continuation of the Schmidt analysis which yields more realistic results. This results in the analysis program becoming better suited for more thorough analysis, which may increase its usability for experienced users and for professional usage. It can also be further expanded through the simple analysis which takes several new factors into consideration, such as pumping losses and fluid friction by subtracting them from the results of the adiabatic analysis [24].

2.3.1 Assumptions

While adiabatic analysis account for more factors than the Schmidt analysis, it still relies on some assumptions.

Firstly, it is assumed that the volume of both heat exchangers is 0 following the 5-volume approach. This is an idealization made to estimate an optimal configuration and to simplify the implementation of the analysis method [4]. It also reduces the number of required input-values which improves the usability for inexperienced users.

Additionally, the mean effective temperature in the regenerator (T_R) is set to be log mean temperature difference, as shown in Equation 11.

2.3.2 Calculation

The implementation of the adiabatic analysis follows an existing set of equations which utilizes the volume variation used in the Schmidt analysis, but recalculates other values, such as pressure [29]. Table 2 shows the values required to perform an adiabatic analysis in addition to the volumes from the Schmidt analysis.

Symbol	Description
R	Gas constant
m	Mass
θ	Number of degrees in radians
T_{Exp}, T_{Com}, T_{Reg}	Temperatures in kelvin
V_{Com}, V_{Exp}, V_{Reg}	Volume of the components
V_{Cyl}	Swept volume of the cylinder
V_{Avg}	Average volume of the cylinder

Table 2: Values required for performing an adiabatic analysis.

First, the mean effective temperature in the regenerator is calculated using Equation 11.

$$T_{Reg} = \frac{T_{Exp} - T_{Com}}{\ln\left(T_{Exp} - T_{Com}\right)} \tag{11}$$

By using a reformulated version of the ideal gas law seen in Equation 5, the pressure was calculated using Equation 12.

Because the volume of each heat exchanger is 0, as stated in Section 2.3.1, the volumes of the compression- and expansion cylinder are used instead.

$$P = \frac{mR}{\frac{V_{Com}}{T_{Com}} + \frac{V_{Reg}}{T_{Reg}} + \frac{V_{Exp}}{T_{Exp}}} \tag{12}$$

Then, the pressure for each step is calculated by using Equation 13.

$$\Delta P = P \cdot \frac{\frac{\Delta V_{Com}}{T_{Com}} + \frac{\Delta V_{Exp}}{T_{Exp}}}{\frac{V_{Com}}{T_{Com}} + \frac{V_{Reg}}{T_{Reg}} + \frac{V_{Exp}}{T_{Exp}}} \tag{13}$$

The mass for each component is then calculated by using Equation 14, which is a variation of the ideal gas law.

$$m_i = \frac{P \cdot V_i}{R \cdot T_i} \tag{14}$$

Next, each of the components' mass for each step is calculated using equation 15.

$$\Delta m_i = \frac{P\Delta V_i + V_i \Delta P}{R \cdot T_i} \tag{15}$$

The change in temperature for each step is calculated by using Equation 16.

$$\Delta T_i = T_i \cdot \left(\frac{\Delta P}{P} + \frac{\Delta V_i}{V_i} - \frac{\Delta m_i}{m_i}\right) \tag{16}$$

The change in temperature can then be added to the respective temperature, as seen in Equation 17.

$$T_i = T_i + \Delta T_i \tag{17}$$

Finally, the mechanical work for each step is calculated by using Equation 9.

3 Method

3.1 Program

The program first presents the user with an introduction-window that lets the user select what values to use during the analysis and visualization. Three options are available to the user; default values, using a custom JSON-file, and entering the input values manually. These values are then used to perform a Schmidt analysis and an adiabatic analysis, where the results are visualized as graphs along with an animation. The results and plots are then saved as CSV- and PDF-files, respectively, to enable the user to review them without starting the program.

While reviewing the results of the analyses, the user can also see an animation displaying an idealized alpha-configured Stirling engine. This animation is synchronized with markers in each graph, that shows the output of the engine for each step of the Stirling cycle. The user can change the speed of the animation, move step-by-step, and set the desired number of degrees manually. Due to simplicity, only graphs from the Schmidt analysis is shown within the program.

After the user has reviewed the results generated by the analysis methods, the results are saved in CSV-files. They are divided by their respective analysis method, and can be used for further analysis, to review the results, or to generate custom visualizations. The plots from the Schmidt- and adiabatic analysis are saved as PDF-files so the user can view them without restarting the program. The user can view the plots for each analysis method individually or view the combined plots for both analysis methods.

3.2 Development approach

The main focus during the development of the program was to make it modifiable, expandable, and provide a good user-experience for both experienced and inexperienced users.

To make the program modifiable and expandable through future implementations, several programming principles were utilized. Firstly, the code is well-documented by having each method contain several explanatory comments regarding its purpose, input-values, and output-values. Therefore, it becomes simpler for a developer unfamiliar with the code to gain a good understanding of it. The code also strives to have high cohesion and low coupling. Cohesion is the degree of focus in a class's functionality, while coupling is the degree of which classes rely on each other [10]. Additionally, the code aims to avoid hard-coded variables, and employing explanatory variable names using the CamelCase naming convention.

The program also aimed to provide a good user-experience for both experienced and inexperienced users, and therefore has several relevant features. This applies to end-users, as well as the program's developers. Firstly, the Stirling cycle is visualized by a VTK-animation that is synchronized with markers in plots showing the results of the Schmidt analysis. The user can then see the output-values generated by the engine for each degree of the Stirling cycle. While this feature is useful for experienced users as well, they will also benefit from being able to view the results and corresponding plots in separate files after running the program. This is also useful for further analysis or validating the accuracy of the results.

Because the program is used for educational purposes it is currently primarily aimed towards inexperienced users, but the program's modifiability and expandability allows more functionality to be added. This could include features that cater more to experienced users or professional usage.

4 Results

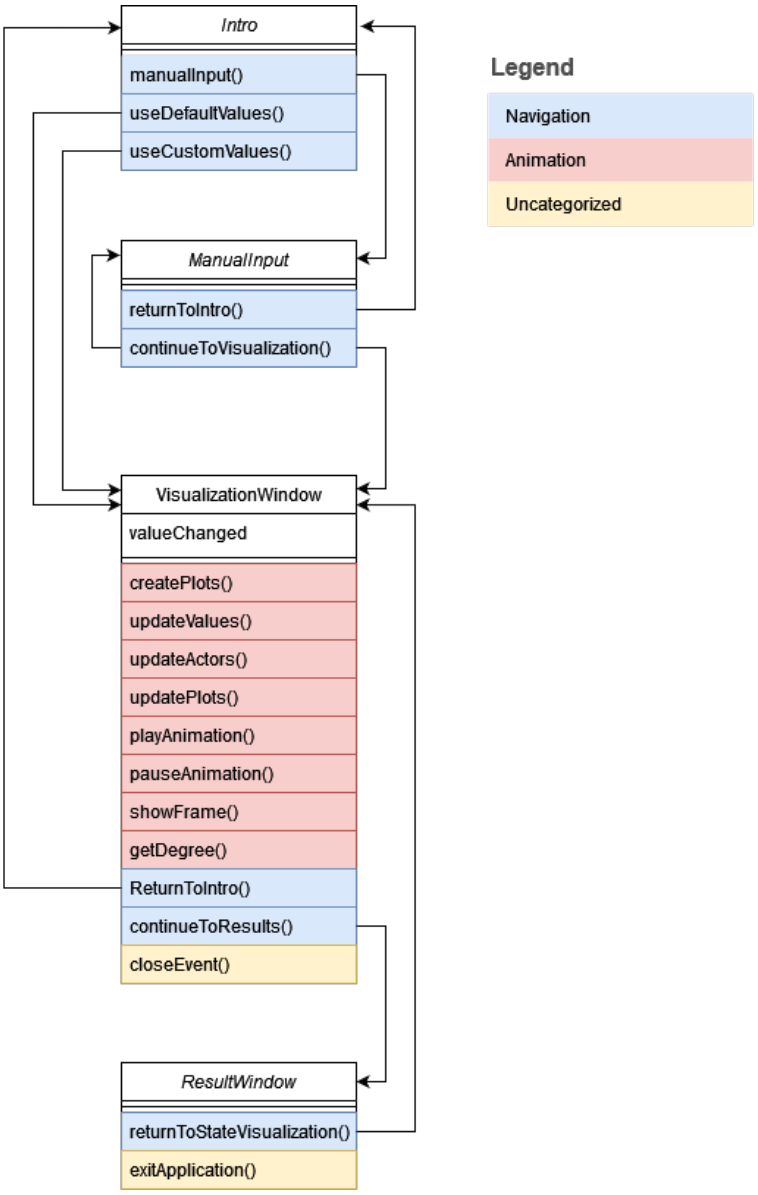


Figure 6: Modified class diagram for '*main.py*'

Figure 6 depicts a modified class diagram for the main component of the program found in the file '*main.py*'. The functions in the diagram are color-coded to specify their usage.

The main component is responsible for creating the program itself, as well as its GUI. Functionality such as file management, animation, and analysis is located in external **Python**-files to increase the code's cohesion and reduce coupling [10]. Additionally, the program also has separate folders for the assets, which include JSON-files for specifying the input-data, and results, where the results of the analyses are stored as CSV-files and the corresponding plots are stored as PDF-files.

The resulting program is an educational analysis program for Stirling engines, that uses a 4-cylinder alpha-configured engine for analysis and visualization. Input-data from the user is analyzed using Schmidt analysis and adiabatic analysis, and the results are presented as plots with markers that are synchronized with a VTK-animation. The results and plots are also stored as CSV- and PDF-files, respectively.

4.1 Windows in GUI

The program's GUI consists of a total of four separate windows. Each window provides different functionality, such as manual input of data, or visualization of results. This subsection gives a thorough description of each window and their functionality.

4.1.1 Intro

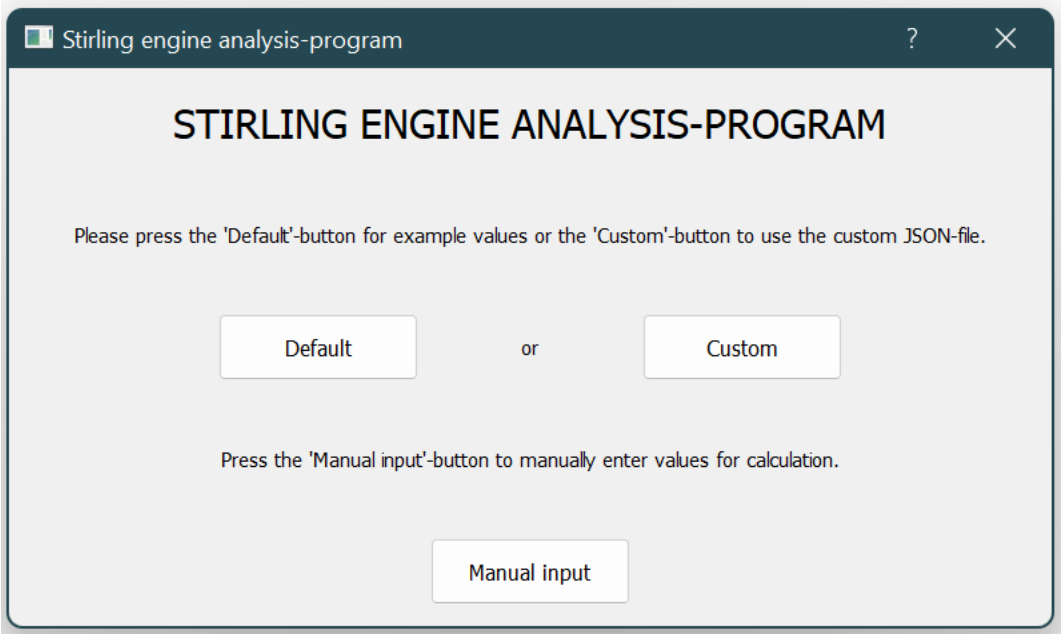


Figure 7: 'Intro'-window

When starting the program, the user is first presented with an introductory window, as seen in Figure 7. Here, the user can select between three options for providing the input-data required to perform the analyses. The options are; using default-values, using a customized JSON-file, or entering the values manually. By pressing an option, the user is navigated to the next window.

4.1.2 Manual input

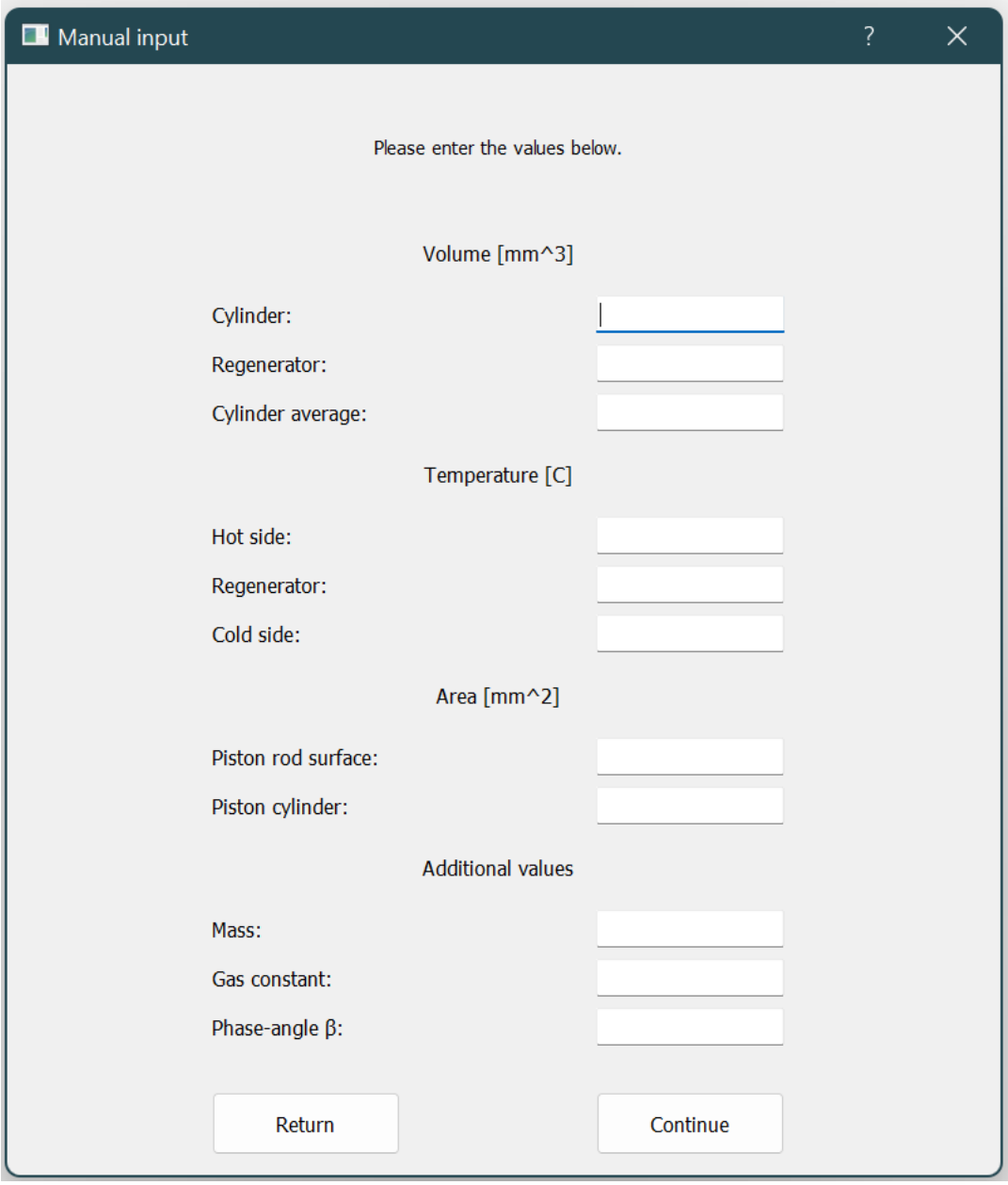


Figure 8: 'Manual input'-window

Next, the user is prompted to enter the values required to perform the analyses. Each value must be a float and more than 0, which is a limitation that is further explained in Section 5. When the user has entered the values, they can press the 'Continue'-button to commence the analyses. If the entered values are deemed invalid, the user is navigated to an empty 'Manual Input'-window, which can be seen in Figure 8. The user can also return to the previous window by pressing the 'Return'-button.

4.1.3 Visualization

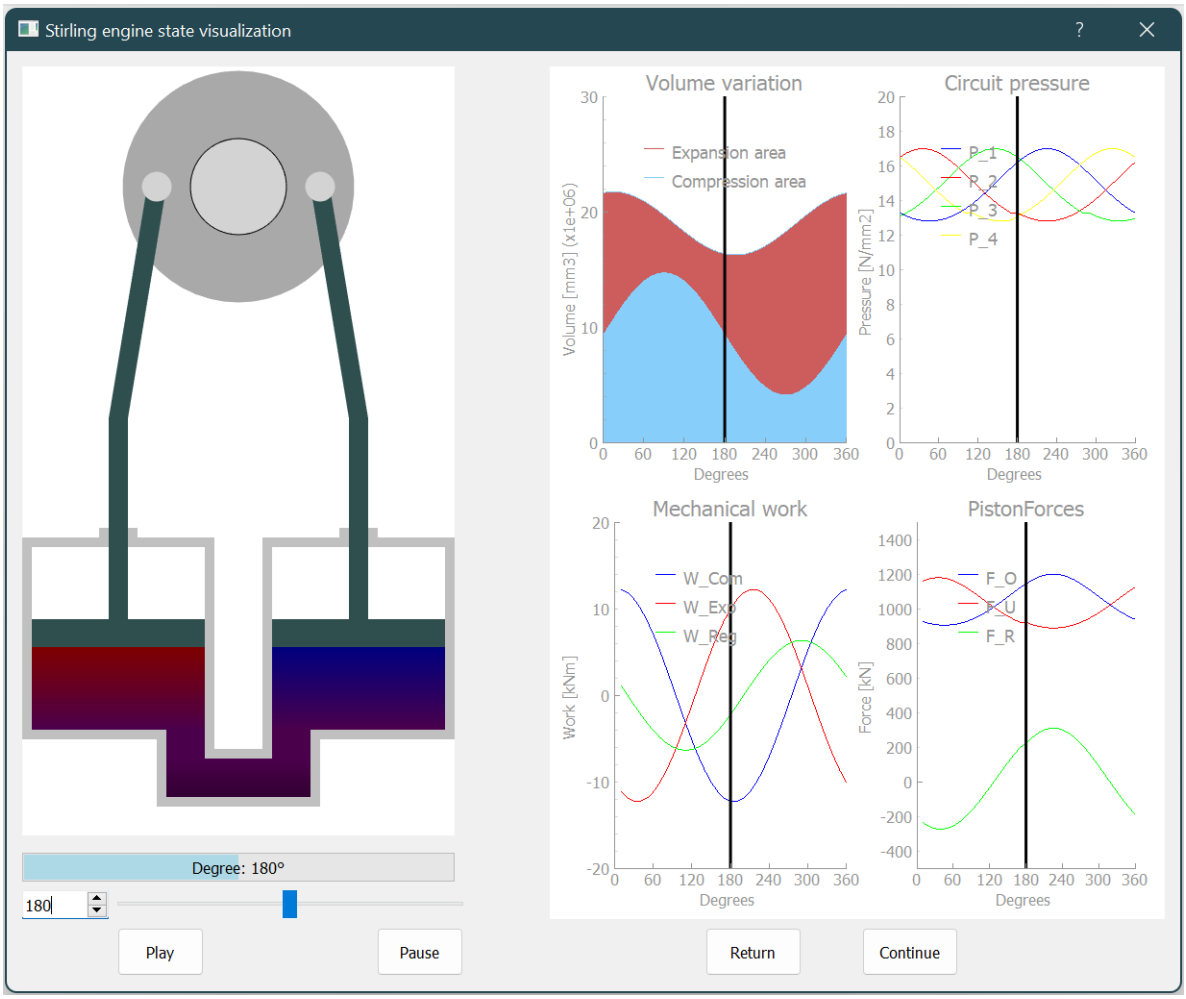


Figure 9: 'State visualization'-window

In this window, the user is presented with the results of the analyses. The results are displayed as plots showing the volume variation, circuit pressure, mechanical work, and piston force. This can be seen in Figure 9. There is also a VTK-animation that illustrates the Stirling cycle. The animation is synchronized with markers in each plot that shows the results for each data point. Several features are added to improve the navigation between the number of degrees. The user can use the spinbox to specify the number of degrees, the slider to alter the animation's speed, and the progressbar to see the progress of the animation. The 'Return'- and 'Continue'-button are used to navigate to other windows.

4.1.4 Results

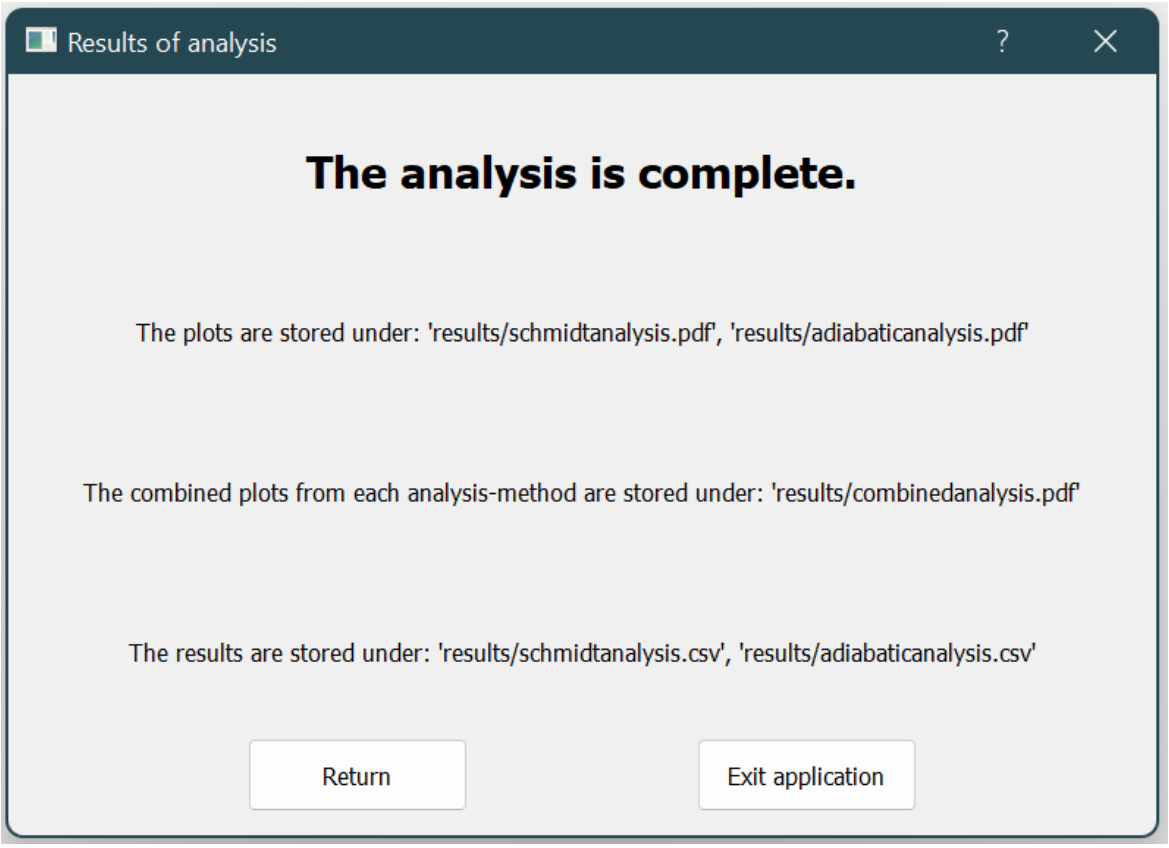


Figure 10: 'Results'-window

The final window signalizes to the user that the analyses have been completed, and that the results and corresponding plots are saved locally. Results are saved as CSV-files, while the plots are saved as PDF-files, and their storage destinations are shown in this window. This is shown in Figure 10. The user can navigate to the previous window or exit the application by using the navigation buttons.

4.2 Numerical example

To display the plots generated from the results of the analyses, a numerical example using values from a report written by Rikard Åsheim [2] is utilized. This is because they share similarities in their implementation due to this program's implementation being inspired of that in the report, as stated in Section 2.2.2. Additionally, these values also coincide with the default-values in the program. The values are listed in Table 3. By using the same values for the analyses, the generated plots become comparable to the plots in the report, and it therefore becomes easier to assess the validity of the results.

Symbol	Value
$R\ [\frac{J}{kg} \cdot K]$	2077
m [kg]	0.502
T_{Exp} [C]	100
T_{Reg} [C]	50
T_{Com} [C]	0
$V_{Cyl}[mm^3]$	10600000
$V_{Reg}[mm^3]$	3619115
$V_{Cyl,avg}[mm^3]$	9500000
$A_{Rod}[mm^2]$	1256.637
$A_{Cyl}[mm^2]$	70685.83
$\beta[^\circ]$	150

Table 3: Example data used for numerical example [2]

These values are entered into the program using the default-option, and are then used in a Schmidt analysis and an adiabatic analysis. The results of the analyses are then used to generate the plots shown in the following subsections.

While the plots in Åsheim's report only contain the results of the Schmidt analysis [2], the plots showing the combined results of both analysis methods are displayed in this report. Each plot uses the number of degrees in the stirling cycle, 0° to 360°, as their x-axis.

By comparison to the plots given in Åsheim's report [2], the plots generated by this program produced similar results. This substantiates the notion that the implemented analysis methods yield accurate results.

4.2.1 Volume variation

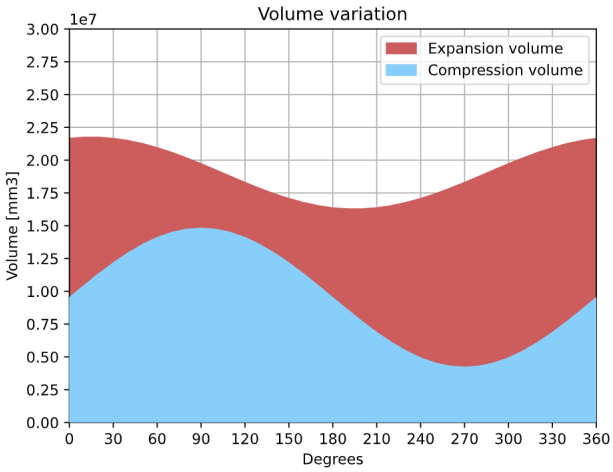


Figure 11: Volume variation

The first plot shows the volume variation in the expansion- and compression-cylinder during the Stirling cycle, where the volume is shown along the y-axis. As seen in the legend in Figure 11, the blue color represents the compression volume, while the red color represents the expansion volume. The total volume seen as the area covered by both volumes can be calculated by adding V_{Exp} and V_{Com}. This data is especially relevant when determining the optimal phase-angle β, because of the importance of the ratio between the expansion-volume and the total volume [2].

4.2.2 Pressure variation

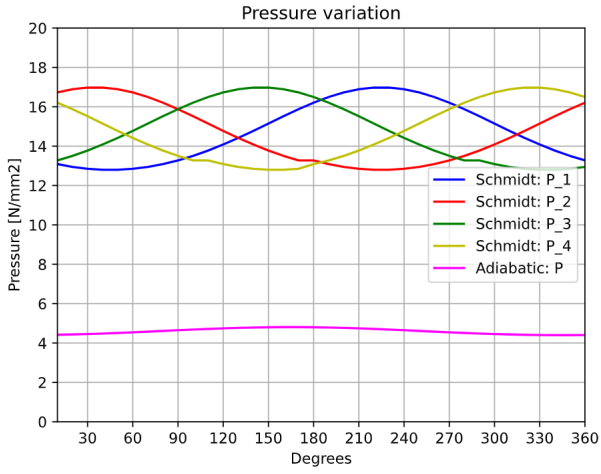


Figure 12: Pressure variation

As stated in Section 3, this implementation of the Stirling engine has four circuits, which results in the pressure for each circuit being calculated using Schmidt analysis. This can be seen in Figure 12. The pressure is only calculated for one circuit in the adiabatic analysis, which is due to its implementation [29].

4.2.3 Mechanical work variation

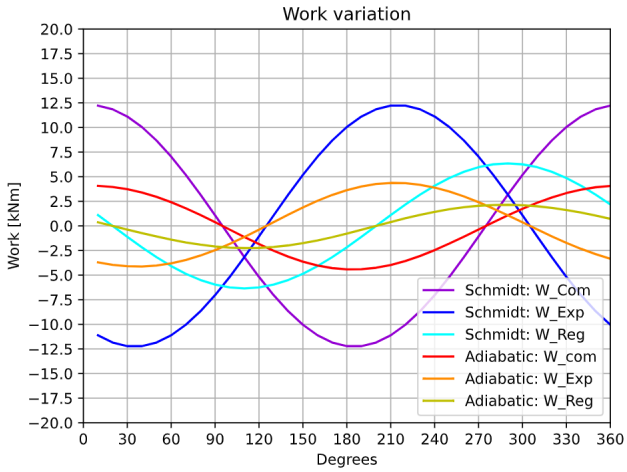


Figure 13: Mechanical work variation

Figure 13 shows the variation of mechanical work generated in each volume. The mechanical work in the regenerator V_{Reg} is defined as the sum of V_{Com} and V_{Exp}.

4.2.4 Force variation

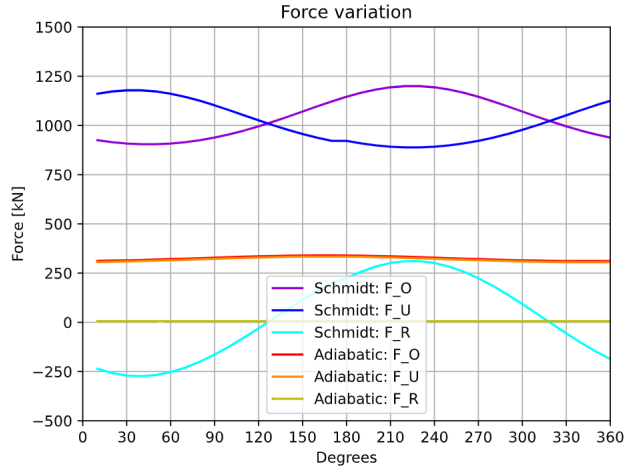


Figure 14: Force variation

The last plot shows the variation of the piston force for each step in the Stirling cycle. Notation from Åsheim's report was used here, and F_O is the force *over* the piston, F_U is the force *under* the piston, and F_R is the difference between F_O and F_U [2]

5 Discussion

This chapter provides the explanation and background for each technological, theoretical, and design decision which were made during the development of the program, in addition to describing what challenges were encountered and how they were resolved.

5.1 Technological decisions

The first decision was to decide which programming language should be used. Several different programming languages could have been used to create this program, but the final choice fell upon **Python**. Firstly, it was chosen due to previous experience with the programming language as well as a number of relevant libraries. **Python** also has a wide selection of useful libraries which proved vital during development, such as *Numpy*, *PyQt*, and *Matplotlib*. It is also widely used and therefore it is easier to troubleshoot occurring issues and fix them. In addition, **Python** is an object-oriented programming which implements several aspects of procedural programming which is well-suited for implementing algorithms, formulas, and equations [11].

Then it was necessary to choose which framework to be used to create the program's Graphical User Interface (GUI). The three main candidates were *PySide*, *PyQT*, and *tkinter* due to their popularity and therefore support and examples on the internet. Firstly, *tkinter* provided several benefits, such as already being integrated in **Python** which would have resulted in the program requiring fewer dependencies, being available on several platforms, and having little overhead which is beneficial for performance [22]. Due to already being integrated in **Python** it is also simple to implement. *PySide* and *PyQT* on the other side are external frameworks with a larger overhead which are more difficult to implement than *tkinter*. While there are some drawbacks, they are also more flexible in terms of appearance and functionality, in addition to having more support for being used in conjunction with other external libraries, such as *Matplotlib PyQtGraph*, and *VTK*. This is the reason why *tkinter* was then removed from consideration, despite providing several benefits. There are many similarities between *PyQt* and *PySide* due to sharing the same foundation, but there are some differences [8]. Between the two, *PyQt* was chosen.

As stated in Chapter 1, this project is a continuation of a project assignment. There *PySide6* was used because it was the newest, but during development of the program several challenges were encountered due to the choice of framework. Firstly, the program used video-animation for providing an animation of the Stirling cycle. This caused an issue when using *PySide6* because it did not have the option to loop the video, but this was resolved when a new version of *PySide6* released during the program's development [20]. Another challenge which was encountered when using *PySide6*, was that it lacked support for plots made using *Matplotlib*. The program could therefore not display these plots, so they were instead stored in a PDF-file which could be accessed when the program was not running.

When starting the continuation of the project, one of the first changes made were to change from *PySide6* to *PySide2*. This was mainly due to the lack of support for *Matplotlib* in the GUI. Later, the GUI-framework was changed from *PySide2* to *PyQt5*, due to changes in the animation and the plots.

While the previous implementation of the program utilized video-animation to display the Stirling cycle, this project's implementation displays the Stirling cycle by using VTK-animation instead. The animation was created using *VTK*, which is a visualization-toolkit similar to e.g., *OpenGL* [23]. Here, the position of each vertex in the rendering is specified directly, which means that an animation can be created by manipulating the position of each vertex. *VTK* also specifies what vertices each engine part consists of, as well as its color. This method of implementing an animation has the benefit of being easy to manipulate precisely and create complex movements and structures. Additionally, the position of each point and the color of each vertex can be interpolated to create smoother animations and transitions. However, because each vertex needs to be specified directly it becomes difficult and time-consuming to create complex renderings or alter an existing rendering. Because the rendering can be animated by updating the position of a vertex using a single variable, it is also simple to integrate the animation with other types of widgets, such as progressbars or a slider to adjust the

animation-speed. *PyQt* was more suited for including VTK-animations than *PySide* due to having more functionality when interacting with *VTK*.

As mentioned previously, the GUI-framework was altered from *PySide6* to *PySide2* to be able to implement plots made using *Matplotlib*. Because the goal of this program is to be used for educational purposes, the plots were included in the program with a marker synchronized with the VTK-animation to show what the resulting data is for each step of the Stirling cycle. When the plots were included and markers synchronized with the VTK-animation were added, the program experienced a major drop in performance which was noticeable through sudden freezes in the program. The reason behind this significant performance-loss, was that *Matplotlib* was not optimized for full animations when being implemented in a program. To resolve this issue, *Matplotlib* was replaced with *PyQtGraph*.

While this project achieved its goal of changing from a premade video-animation to a modifiable VTK-animation, there were also some additional goals regarding animation which unfortunately were not implemented. For instance, it was also intended that the dimensions of the VTK-animation were depending on the input-variables. This was not implemented because the program would become more difficult to use for inexperienced users due to the number of required input-variables. Additionally, there were other features for enhancing user-experience which were not added. Firstly, the user should be able to select a custom JSON-file with input-variables using a file-explorer. There should also be a ReadMe-file or user-manual accessible in the program by using e.g. a button. Lastly, the user should be able to set the step size in number of degrees when calculating results from the analyses. These features were not added due to time-constraints during development.

In addition to the libraries included in **Python**; *Numpy*, *PyQt5*, *VTK*, *Matplotlib*, and *PyQtGraph* were used in the final program.

5.2 Theoretical decisions

To produce the results required for visualization, both Schmidt- and adiabatic analysis were utilized. Schmidt analysis was mainly chosen because it serves as a foundation for other analysis methods, such as adiabatic analysis [24]. It is also simple and the equations are suitable for programming. While it only analyzes the thermodynamic process, it yields relevant information regarding pressure, mechanical work, and piston forces, as stated in 2.2. This can be seen in plots generated in the numerical example in Section 4.2. However, it is idealized and therefore does not consider fluid friction or pumping losses [24]. In addition, the Schmidt analysis also assumes that neither the heat exchangers nor the regenerator lose any heat.

Adiabatic analysis is an expansion on the Schmidt analysis which assumes the Stirling cycle is adiabatic rather than isothermal [24]. This means that Schmidt analysis assumes that the temperatures are constant in the heater and cooler, which would result in no net heat transfer making them redundant. The adiabatic analysis on the other hand assumes that any net heat transfer is due to the heat exchangers [29]. Therefore is the adiabatic analysis considered a more realistic analysis method than Schmidt analysis, although the Schmidt analysis provides several relevant results.

Schmidt- and adiabatic analysis were implemented because they both yield relevant results and are relatively simple to implement. They also serve as a foundation for additional analysis methods, such as simple analysis [24]. Simple analysis, for instance, was not implemented due to the increasing number of input-variables which would make the program more complicated to use. In addition, there were time-constraints regarding implementation of other analysis methods.

During the development some assumptions were made when implementing the analysis methods. Firstly, the calculations were made using a 4-cylinder alpha-configured Stirling engine. Four cylinders were used due to the simplicity in calculations and the possibility to expand to a higher number of cylinders in the future. Of the four circuits used in calculations and seen in the plots in Section 4.2 two pairs are located on each side of the engine, and each pair is offset by 90 degrees. The alpha-configuration was chosen because it is a simple configuration often used for teaching, and it matches

the VTK-animation.

As stated in Section 2, the calculations are based on the 5-volume approach which consists of the expansion-, compression-, heater-, cooler-, and regenerator volume. To simplify the calculations and user-experience, an idealized version of the approach was used where the volume of the heater and cooler is set to 0, which is also reflected in the VTK-animation.

5.3 Design decisions

Several design decisions were made to achieve the goals set for this project.

Firstly, the user has different options regarding how to enter the input-variables. The user can select default-values which includes the same values used in Section 4.2, and therefore generates the same plots. This is to provide the user with a simple way of viewing the Stirling cycle with corresponding plots. There is also an option for the user to enter the values manually in the program or edit a custom JSON-file. To further the educational aspect of the program, the more advanced options such as heat capacity at constant pressure and volume are only editable when using the custom JSON-file. This is because experienced users are most likely to use the custom JSON-file rather than manually enter values, which makes the program easier to use for a first-time user. The entered input-values are stored in a separate JSON-file which is relevant when reviewing results or creating a custom JSON-file.

The results of the analyses are also extracted to corresponding CSV-files which can be used for further analysis outside the program or with different analysis methods in future expansions. This also makes it easier to review the values after calculations and compare them if necessary.

As stated in Section 5.1, both *Matplotlib* and *PyQtGraph* are used in the program to generate plots due to performance issues. Previously, the plots were only saved in a PDF-file, but now the plots are also included internally in the program. This was due to missing functionality in *PySide6* which later was changed to *PyQt5*. The feature of saving the plots as a PDF-file was kept because it may be desired by users, and it then becomes possible to review the results and corresponding plots without starting the program.

The internal plots only show the results of the Schmidt analysis. This is because they provide a concise visualization of the results, and the plots were deemed to be to crowded when utilizing combined plots from both analysis methods.

Because *PyQtGraph* is inherently more lightweight than *Matplotlib* and therefore contains less formatting and features, there is a difference between the plots in the PDF-file and the plots in the program. The plots created using *Matplotlib* are more detailed and better formatted, while the internal plots created in *PyQtGraph* are less refined. This is acceptable because the internal plots mostly serve as a visualization for different values in each step of the Stirling cycle.

While the plots in the program could be modified to include whichever plots are desired, they currently show the results from the Schmidt analysis. These plots are the same as shown in Section 4.2. They were chosen because they provide good visual representation of relevant values, and are therefore useful for educational purposes.

The main goal of this project was to create an educational program which could inter alia provide inexperienced users with insight of the Stirling cycle. Several steps were taken to achieve this, such as having a VTK-animation which is synchronized with markers for each plot. As a consequence of the animation's implementation it becomes simple to add additional features. In addition to the animation, both a progressbar and a slider for animation speed were added. It is also possible to specify the number of degrees directly and press a button to navigate one degree at a time. All these features results in the animation showing the correct frame and the markers in the plots showing the same number of degrees.

This program's GUI use the standard appearance of *PyQt5* with few alterations, which proved to be an adequate solution. The standard appearance is quite simplistic and professional, which is suitable for this program. Due to *PyQt5*'s style-sheet it is possible to simply alter the GUI's appearance.

5.4 Future implementations and expansions

Modifiability and expandability were in focus throughout the development of this program, and there are therefore several ways to expand it in the future.

Firstly, the program could be expanded by implementing simple analysis. Simple analysis is a natural continuation of adiabatic analysis by subtracting the effects of pumping losses and friction from the results of the analysis [24]. It was not implemented in this program due to the focus on educational usability and simple analysis would increase the number of input values, which would make the program become more difficult to use for inexperienced users. However, this could be desired if the program changes focus to a more professional usage. While simple analysis would be a suitable continuation, several other analysis methods could also be suitable. Due to the focus on modifiability during the development of the program, as described in Section 3.2, it should be relatively simple to implement additional analysis methods.

The Stirling engine used for calculations in the program currently has four cylinders and an alpha-configuration. While this yields relevant results and provide a good overview of the Stirling cycle, it could be beneficial to expand the number of options. E.g., the number of cylinders and a selection of other configurations, such as beta- and gamma-configuration. The user would then also have to specify which circuits were used in calculations. While these options would be useful, they would also increase the program difficulty significantly, which would contribute to the exclusion of inexperienced users. However, this could be more relevant if the usage of the program changed. This would be especially relevant if the program was modified to be used commercially or for the development of a new Stirling engine.

If the focus of the program changes from educational to professional usage, it would also be advantageous to modify the plots. This includes both the internal plots and the plots saved in a PDF-file. For instance, more details could be added, more plots of the results could be created, and the plots could include comparable values from each analysis method. Another option could be to make the formatting of the CSV-results containing the results modifiable, which would improve usability for analysis outside the program. These are suggestions for possible features which could be added, but other features could prove more ideal depending on the program's focus.

Another feature which was discussed, but not implemented, was having a dynamic VTK-animation based on the input-values. The dimensions and appearance of the VTK-animation would then rely on the given input-variables, rather than remain constant regardless of its input. This was not implemented because the user would then have to enter even more input-variables and the implementation would be time-consuming. It is also possible to reflect other input-variables in the animation, such as number of cylinders, configurations, geometries, and which circuits are used. This is currently not relevant as there is no variation in either of these values, but it could become desired in future expansions.

In addition, the user receives minimal feedback regarding the validity of the results. If an input-value is considered invalid, the user is presented with an empty 'Manual input'-window. This could be improved by giving the user detailed feedback regarding which value was invalid. To implement this, the checks of the input-data must also be extended to provide additional information about its validity.

Another beneficial addition would be to add a pressure-volume curve to the "State"-window, as seen in Figure 1. This curve provides insight into the Stirling-cycle and is therefore especially relevant when used for educational purposes. A marker synchronized with the VTK-animation could also be added to increase the learning outcome. The pressure-volume curve was not added to the program due to time-constraints.

Lastly, the appearance of the program's GUI could be improved. As stated previously, the GUI is currently using *PyQt5*'s standard appearance because it was deemed simple, but professional. While the appearance is currently adequate, this could change in the future. Especially if the usage of the program changes. This also includes changing the general layout of the program which has room for improvement. In particular the "State"-window does currently not support additional plots. This could be a requirement if more plots were added to the program or plots from other analysis methods were included.

6 Conclusion

The goal of this project was to create a modifiable and expandable analysis program for Stirling engines that could be used for educational purposes. It should also provide a good user experience for both experienced and inexperienced users. The final program is an analysis program for Stirling engines that lets the user enter input-data that is then used to perform several analyses. The results of the analyses are then presented as plots along with a VTK-animation to visualize the output. Finally, the results and the corresponding plots are stored locally in CSV- and PDF-files, respectively.

To ensure that the program was modifiable and expandable, multiple software principles were followed during the development of the program, such as high cohesion and low coupling, explanatory variable names, few hard-coded variables, and thorough documentation. The goals also influenced the outcome of the technical decisions that were encountered during the development.

Throughout the development, a goal was to create an analysis program that could be used for educational purposes. To achieve this, the final program has several features aimed towards users with a different amount of experience. Firstly, there are three options for entering the input-data. The results of the analyses are visualized using plots and each contain a marker synchronized with a VTK-animation, which lets the user see the engine's output for each step of the Stirling cycle. This is especially useful for users unfamiliar with the Stirling cycle. In addition, the number of interactions required by the user to perform the analyses is minimal. Finally, both the results and plots are stored locally. This makes them available for further analysis or review to confirm the results' validity.

After discussing the outcomes of the decisions made during the development of the program, it can be concluded that the final program achieved its goal. This goal was to create a modifiable, expandable, and educational analysis program for Stirling engines. Several measures were taken to ensure that the program achieved its goal, and the final program is modifiable and expandable through future implementations. It can also be used for educational purposes due to its visualization of the results of the analysis methods and low complexity which lets users navigate the program with ease. While additional functionality can be added in future expansions, the current program provides sufficient functionality to serve as a full and independent analysis program. In conclusion, the program achieved the goals set in this project.

7 References

[1] Mohammad Hossein Ahmadi, Mohammad Ali Ahmadi, and Mehdi Mehrpooya. "Investigation of the effect of design parameters on power output and thermal efficiency of a Stirling engine by thermodynamic analysis". In: *International Journal of Low-Carbon Technologies* 11.2 (May 2016), pp. 141–156. ISSN: 17481325. DOI: 10.1093/ijlct/ctu030.

[2] Rikard Åsheim. "Analysis and Dimensioning of a Stirling Engine". In: June (2011).

[3] *CamelCase*. 2020. URL: https://www.computerhope.com/jargon/c/camelcase.htm (visited on 06/09/2022).

[4] M Cavazzuti and G S Barozzi. "Stirling engine optimization based on Schmidt and adiabatic analyses". In: (), pp. 1–56. URL: https://www.esteco.com/cmis/browser?id=workspace://SpacesStore/f5c0f9f7-99e2-4c62-913f-86d797d87546.

[5] *Data point*. 2017. URL: https://www.computerhope.com/jargon/d/data-point.htm (visited on 06/08/2022).

[6] Jose Egas and Don M. Clucas. "Stirling engine configuration selection". In: *Energies* 11.3 (Feb. 2018). ISSN: 19961073. DOI: 10.3390/en11030584.

[7] Kristoffer Lehre Fagerheim. *Developing an educational analysis-program for Stirling engines*. Tech. rep. 2021, pp. 1–21.

[8] Martin Fitzpatrick. *PyQt6 vs PySide6*. Mar. 2021. URL: https://www.pythonguis.com/faq/pyqt6-vs-pyside6/ (visited on 12/19/2021).

[9] Koichi Hirata. *SCHMIDT THEORY FOR STIRLING ENGINES*. Tech. rep. National Maritime Research Institute. URL: http://www.nmri.go.jp/env/khirata/.

[10] JavaTpoint. *Coupling and Cohesion*. 2021. URL: https://www.javatpoint.com/software-engineering-coupling-and-cohesion (visited on 12/19/2021).

[11] Wie Kiang H. *Python: Procedural or Object-Oriented Programming?* 2020. URL: https://towardsdatascience.com/python-procedural-or-object-oriented-programming-42c66a008676 (visited on 06/08/2022).

[12] C C Kwasi-Effah, A I Obanor, and F A Aisien. "Stirling Engine Technology : A Technical Approach to Balance the Use of Renewable and Non-Renewable Energy Sources". In: *American Journal of Renewable and Sustainable Energy* 1.3 (2015), pp. 156–165. URL: http://www.aiscience.org/journal/ajrse.

[13] Rachid Maamri et al. "Development of External Combustion Engine". In: *American Journal of Vehicle Design* 1.2 (2013), pp. 25–29. DOI: 10.12691/ajvd-1-2-2.

[14] *OOP*. 2019. URL: https://www.computerhope.com/jargon/o/oop.htm (visited on 06/08/2022).

[15] *Overhead (computing)*. 2022. URL: https://en-academic.com/dic.nsf/enwiki/11569862 (visited on 06/08/2022).

[16] *Procedural language*. 2019. URL: https://www.computerhope.com/jargon/p/proclang.htm (visited on 06/08/2022).

[17] *Progressbar*. 2021. URL: https://pythonbasics.org/progressbar/ (visited on 06/08/2022).

[18] *PyQt - QSpinBox Widget*. 2022. URL: https://www.tutorialspoint.com/pyqt/pyqt_qspinbox_widget.htm (visited on 06/08/2022).

[19] *QSlider*. 2021. URL: https://pythonbasics.org/qslider/ (visited on 06/08/2022).

[20] Qt. *Qt for Python Development Notes*. Dec. 2021. URL: https://wiki.qt.io/Qt_for_Python_Development_Notes (visited on 12/19/2021).

[21] Ritesh Ranjan. *What is a Framework in Programming & Why You Should Use One*. 2021. URL: https://www.netsolutions.com/insights/what-is-a-framework-in-programming/ (visited on 06/08/2022).

[22] Real Python. *Python GUI Programming With Tkinter*. 2021. URL: https://realpython.com/python-gui-tkinter/ (visited on 12/19/2021).

[23] William J. Schroeder and Kenneth M. Martin. "The visualization toolkit". In: *Visualization Handbook* July (2005), pp. 593–614. DOI: 10.1016/B978-012387582-2/50032-0. URL: https://gitlab.kitware.com/vtk/textbook/raw/master/VTKBook/VTKTextBook.pdf.

[24] H. Snyman, T.M. Harms, and J.M. Strauss. "Design analysis methods for Stirling engines". In: *Journal of Energy in Southern Africa* 19.3 (2008), pp. 4–19. ISSN: 1021-447X. DOI: 10.17159/2413-3051/2008/v19i3a3329.

[25] *Software Engineering — Halstead's Software Metrics*. 2020. URL: https://www.geeksforgeeks.org/software-engineering-halsteads-software-metrics/ (visited on 06/08/2022).

[26] Israel Urieli. *Chapter 1 – Background and Introduction*. 2020. URL: https://www.ohio.edu/mechanical/stirling/intro.html (visited on 05/26/2022).

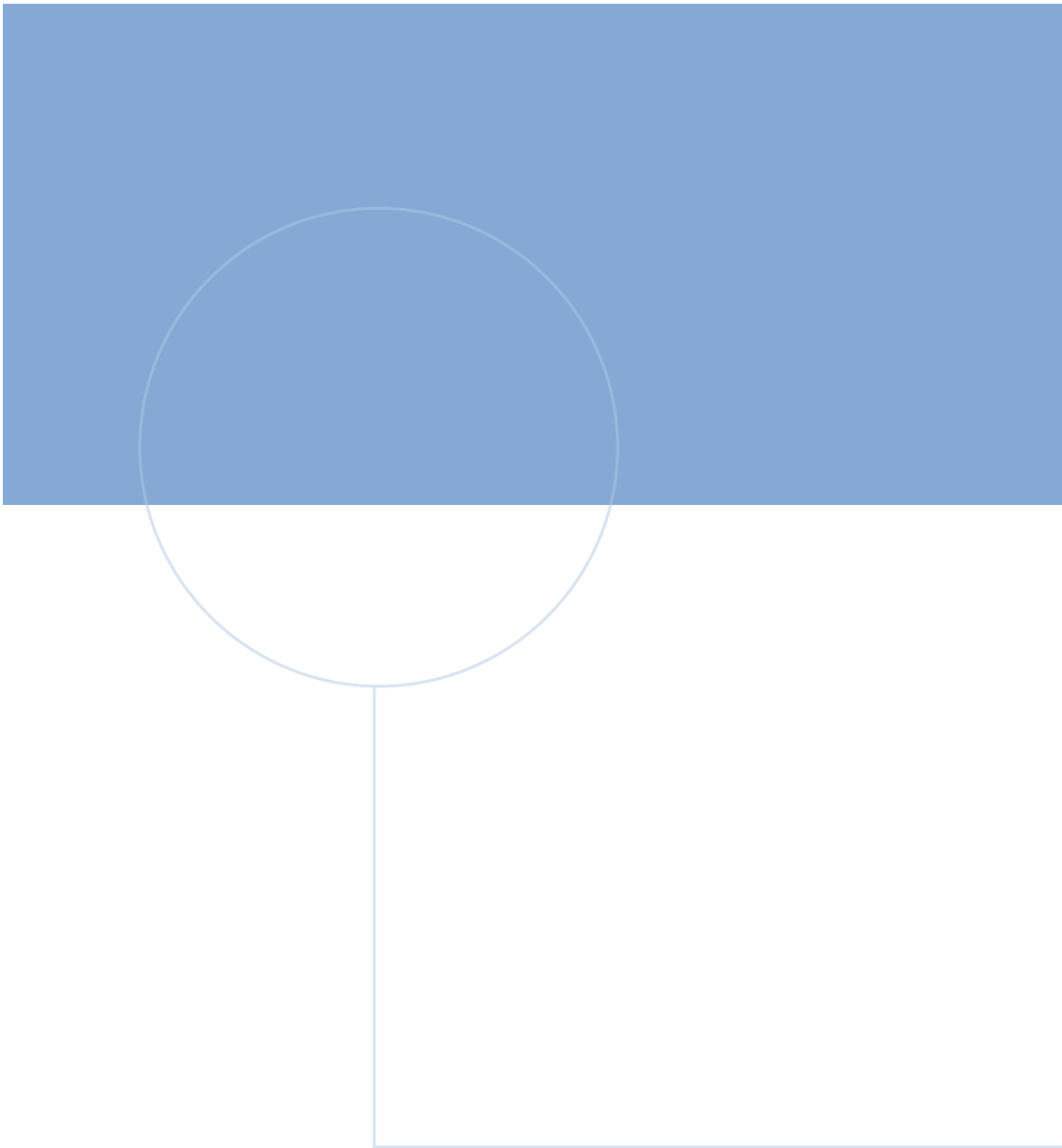
[27] Israel Urieli. *Chapter 2a – Alpha Stirling Engines*. 2020. URL: https://www.ohio.edu/mechanical/stirling/engines/engines.html (visited on 05/26/2022).

[28] Israel Urieli. *Chapter 3a - Ideal Isothermal Analysis*. 2020. URL: https://www.ohio.edu/mechanical/stirling/isothermal/isothermal.html (visited on 05/30/2022).

[29] Israel Urieli. *Chapter 4a - Ideal Adiabatic Analysis*. 2020. URL: https://www.ohio.edu/mechanical/stirling/adiabatic/adiabatic.html (visited on 05/17/2022).

[30] *Widget*. 2021. URL: https://www.computerhope.com/jargon/w/widget.htm (visited on 06/08/2022).

[31] Ron Zevenhoven et al. "Performance improvement of an industrial Stirling engine heat pump". In: *ECOS 2020 - Proceedings of the 33rd International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems* (2020), pp. 1042–1053.



NTNU

Kunnskap for en bedre verden