A SIMULATION MODEL OF GAMMA CONFIGURATION STIRLING ENGINE WITH NON-SINUSOIDAL MOTION

ONG YON SIANG

A project report submitted in partial fulfilment of the requirements for the award of Master of Engineering (Mechanical)

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

April 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	[illegible]
Name	:	Ong Yon Siang
ID No.	:	22UEM01761
Date	:	15-April-2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“A SIMULATION MODEL OF GAMMA CONFIGURATION STIRLING ENGINE WITH NON-SINUSOIDAL MOTION”** was prepared by **ONG YON SIANG** has met the required standard for submission in partial fulfilment of the requirements for the award of Master of Engineering (Mechanical) at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	Wong
Supervisor	:	Dr. Wong Hong Mun
Date	:	15-April-2024

Signature	:	-
Co-Supervisor	:	-
Date	:	-

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my project supervisor, Dr. Wong Hong Mun for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement during the process of completing this project

ABSTRACT

Stirling engine is an external combustion engine that able to convert heat energy to work. The ideal Stirling engine works according to the Stirling cycle where the processes are isothermal gas compression, isochoric gas heating, isothermal gas expansion and finally isochoric gas cooling. These processes can be realised by 2 pistons that regulate the flow of gas between 2 different temperature chambers; and heat exchangers (heater, cooler and regenerator) that regulate the gas temperature in different parts of the engine. Each of the 2 pistons of the conventional Stirling engine moves in a sinusoidal/near sinusoidal pattern with a 90° phase different between them. Mechanisms that generate sinusoidal/near sinusoidal motions like slider-crank, scotch yoke, rhombic drive and etc. are robust, have smooth/continuous motion. However, using these sinusoidal/near sinusoidal motion sacrifice thermal efficiency or work output of the engine for smooth operation. In this work, a numerical model of gamma configuration Stirling engine with non-sinusoidal piston motion was built to study the changes in engine output and its thermal efficiency when compared with conventional sinusoidal motion. Form-closed cam with oscillating follower is used as a non-sinusoidal motion generator in this study to simulate the Stirling engine performance. Ideal isothermal model/ Schmidt model will be used in the simulation to predict the engine work output and thermal efficiency. Results showed that by substituting the sinusoidal motion mechanism with a non-sinusoidal form-closed cam, the indicated work output of the engine is increased by 1.0665W in the Schmidt model simulation or 0.9627W in Urieli and Berchowitz model simulation, while the thermal efficiency has no significant change. Modified Scotch yoke mechanism that can generate different non-sinusoidal piston motion was also simulated. It shows an increase of around 26% indicated power and a decrease in thermal efficiency by less than 1% when compared to the conventional crank-slider mechanism.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

a	acceleration, m/s^2
A	area, m^2
c_p	specific heat capacity at constant pressure, J/(kg·K)
c_v	specific heat capacity at constant volume, J/(kg·K)
$c_{gas\ speed}$	gas molecules average speed, m/s
comp	compression
d	diameter
exp	expansion
f_{RE}	Reynolds friction factor
f	frequency
h	heater
h_{stroke}	stroke, m or rad
$h_{k,h}$	heat transfer coefficient, W/(m^2·K)
j	jerk, m/s^3
$\mathrm{J_{gap}}$	gap, m
k	cooler
$\mathrm{k_{isentropic}}$	heat capaicty ratio
$\mathrm{k_{gas}}$	heat conductivity, W/(m·K)
L	length, m
M	mass, kg
$n_{polytropic}$	polytropic index
NTU	number of transfer unit
P	pressure, Pa
Pr	Prandtl number
Q	heat Transfer, J
r	radius, r
$R_{specific}$	specific gas constant, J/(K·kg)
Re	Reynolds number
St	Stanton number
S_n	Simpson’s numerical integration
T	temperature, K
T_{gas}	gas torque, Nm

u	velocity, m/s
v	velocity, m/s
V	volume, m^3
W	work done, J
x	displacement, m
Y_k	friction and wall properties
Z_k	wave form factor
θ	angle, rad
η	efficiency
ρ	density, kg/m^3
μ	viscosity, Pa·s
ω	angular velocity, rad/s
γ	heat capaicty ratio
ε	efficiency
$\emptyset$	pressure angle, rad

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Stirling engine is an external combustion heat engine invented by Robert Stirling, that converts heat energy to work. The heat energy input does not necessary to be sourced from combustion of fuel. The heat energy sources can be solar energy, biomass combustion or waste heat from process heating. One of the advantages of the Stirling engine is that it can operate if there are two different temperature sources available, and their temperature difference does not need to be very large if it is specially design for low temperature difference operation.

Ideal Stirling engine operates according to the Stirling cycle, where its processes are shown in Figure 1.1 (Cengel, Boles and Kanoğlu, 2011). Thermal efficiency of an ideal Stirling engine is similar to the Carnot engine where its formulation is shown below (Cengel, Boles and Kanoğlu 2011):

$$\eta_{th\ (ideal)} = 1 - \frac{T_K}{T_H} \tag{1.1}$$

where

$\eta_{th\ (ideal)}$ = thermal efficiency of an ideal Stirling engine

T_K = cooler temperature, K

T_H = heater temperature, K

Three common configurations of Stirling Engine are alpha, beta and gamma, which are shown in Figure 1.2 (Rahmati et al., 2020). Gamma configuration Stirling engine is chosen to be studied in this project due to its simple setup. As shown in Figure 1.3, the main components of gamma configuration Stirling engine are displacer piston, power piston, heater, cooler and regenerator (Martini, 1983). Heater and cooler try to keep two spaces at each side of the displacer piston at constant temperature. Regenerator helps to heat up or cool down the gasses that flows between the hot space and cool space by storing or releasing heat energy. Displacer piston moves one side of the gas to another side of the space for heating/cooling without changing the overall volume of the gas. On the other hand, power piston compresses the gas or allow gas expansion in the engine. The space in the power piston cylinder is considered a "cold" space as no heating process occurred and its position is normally closer to the cold side of the displacer.

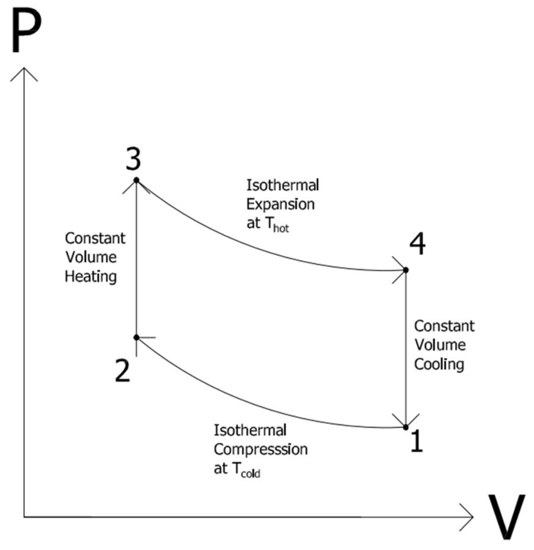


Figure 1.1: Ideal Stirling cycle (Cengel, Boles and Kanoğlu, 2011)

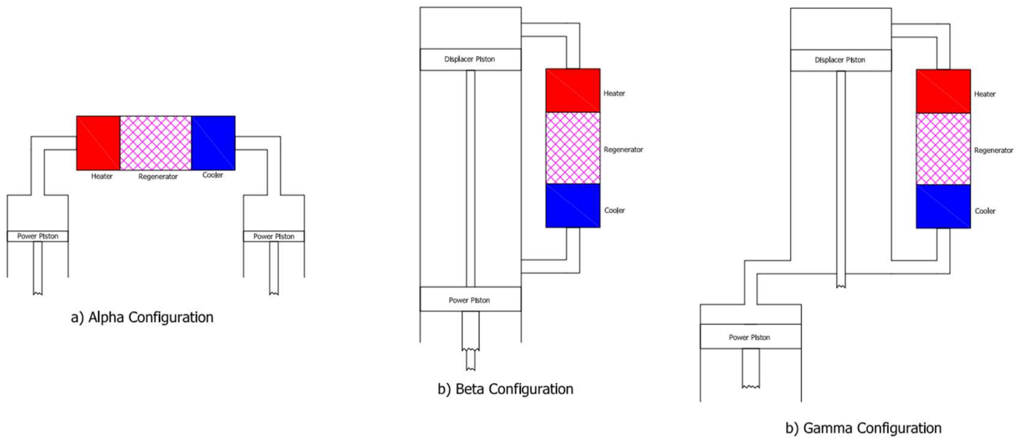


Figure 1.2: Three main configuration of Stirling engine (Rahmati et al., 2020)

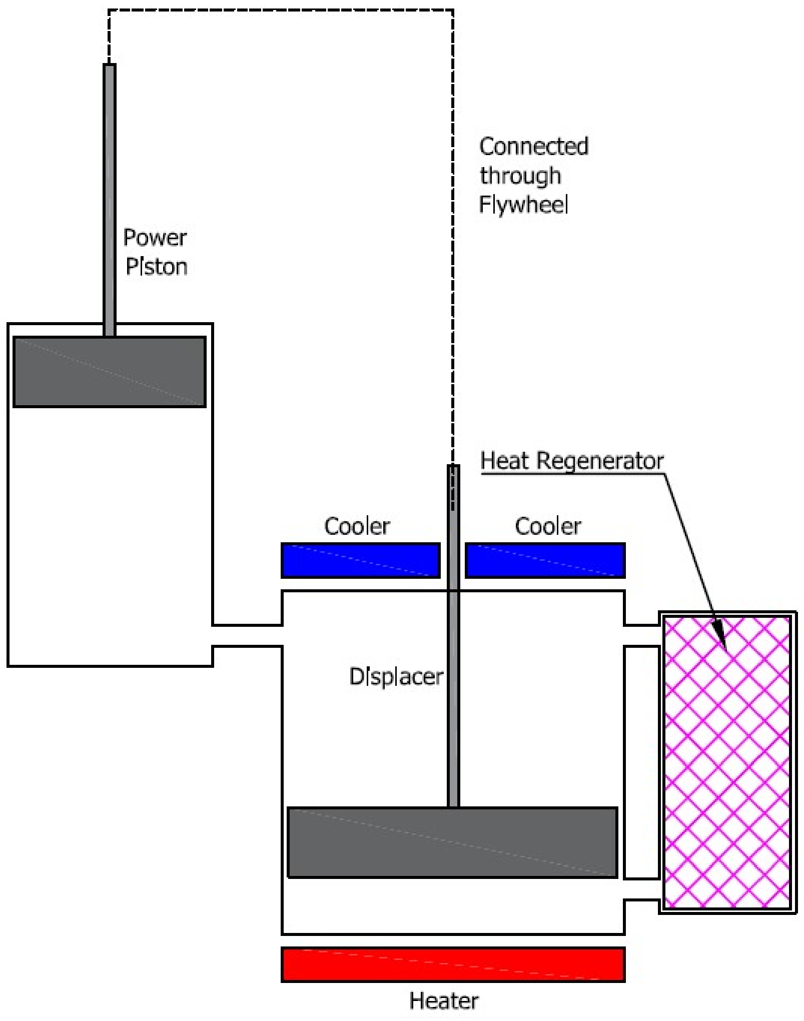


Figure 1.3: Gamma Stirling engine components (Kongtragool and Wongwises, 2003)

Ideal gamma Stirling engine has an ideal piston motion similar to Figure 1.4 (Kongtragool and Wongwises, 2003), where displacer piston motion is shown at the top, while power piston motion is shown at the bottom. Most conventional Stirling engines have their pistons move in sinusoidal form due to the robustness, smoothness and simple construction of the sinusoidal motion generator mechanism such as slider-crank mechanism. Figure 1.5 shows some common sinusoidal motion mechanisms in Stirling engine. However, some thermal efficiency of the engine is sacrificed in exchange for the benefits of the sinusoidal motion mechanism. Figure 1.6 shows the conceptual PV diagrams of Stirling engine with ideal piston motion and Stirling engine with sinusoidal piston motion (Nicol-Seto, 2021). It is observed that the work output of the engine (area within the curve) reduces with sinusoidal piston motion. In this project, a simulation model of gamma configuration Stirling engine with non-sinusoidal motion is built to study the improvement in thermal efficiency of the Stirling engine with non-sinusoidal motion mechanism.

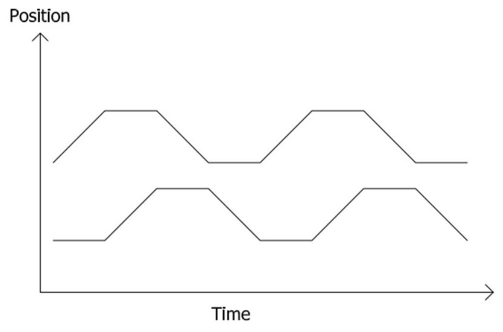


Figure 1.4: Ideal Piston motion of gamma Stirling engine (Kongtragool and Wongwises, 2003)

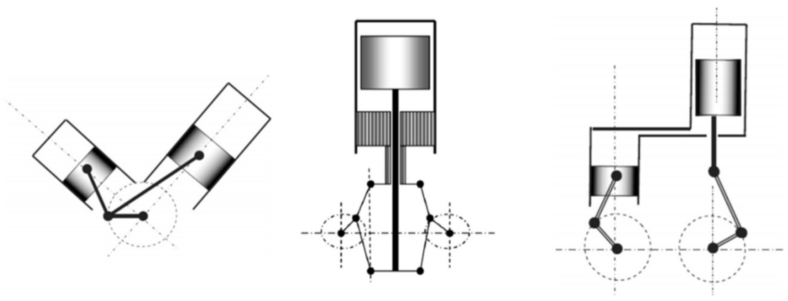


Figure 1.5: Stirling engine mechanisms that generates sinusoidal piston motion (Ahmadi, Ahmadi and Pourfayaz, 2017)

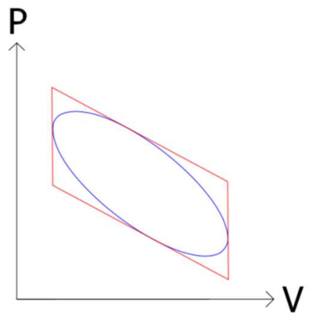


Figure 1.6: Illustration of PV diagrams of Stirling engine with sinusoidal piston motion and Stirling engine with ideal piston motion. Red color curve is the PV diagram of the Stirling engine with ideal piston motion. (Nicol-Seto, 2021)

1.2 Importance of the Study

Due to the urgency of the environmental issue caused by the global warming phenomena, it is very important to have methods, strategies or ideas that would help to improve the energy efficiencies of the machines. Waste heat from process heating by factories are some of the factors that contribute to the energy wastage in the factories. Applying Stirling engine to reuse some of the energy from the waste heat is one of the methods to reduce energy wastage. Approaches like introducing non-sinusoidal motion to the Stirling engine pistons are possible to improve the efficiency of the Stirling engine, and eventually extract more useful energy from the waste heat. It is the same for improvement of solar energy extraction using Stirling engine with non-sinusoidal piston motions. As the Stirling engine can run continuously as long as the heat sources are stable, therefore small improvement in the engine efficiency can generate more energy in long operation time. Hence, development of simulation model can speed up the design of the Stirling engine with non-sinusoidal piston motions.

1.3 Problem Statement

There are many variations of non-sinusoidal piston motions that a gamma configuration Stirling engine can adopt to increase the amount of heat that can be absorbed and improve the engine output. To evaluate the gamma configuration Stirling engine performance under many variations of non-sinusoidal motions through experimental methods is time consuming, since the development time for different experimental setups is very long and it may cause wastage in the materials when building the prototypes. There are many existing mathematical models of Stirling Engine. However, the models are very general and does not focus on the customized piston motion.

1.4 Aims and Objectives

The aim of this project is to study the improvement of the Stirling engine work output and its thermal efficiency under non-sinusoidal piston motion. To achieve this aim, the objectives of this project are:

1) To develop a mathematical model of gamma configuration Stirling engine with non-sinusoidal piston motion that focus on studying the engine work output and its thermal efficiency.

2) To assess the performance of the gamma configuration Stirling engine under different non-sinusoidal motions on engine work output and its thermal efficiency.
3) To compare the non-sinusoidal Stirling engine mathematical model built and the sinusoidal gamma configuration Stirling engine on engine work output and its thermal efficiency.
4) To verify the Stirling engine mathematical model by comparing the output with the real gamma configuration Stirling engine.

1.5 Scope and Limitation of the Study

In this project, the simulation models developed do not include losses analysis of the Stirling engine. Engine heat losses and mechanical losses of the mechanisms are not included in the simulation. The analysis of the simulation results only focuses on the thermal efficiency and work output of the simulated Stirling engine.

1.6 Contribution of the Study

This project develops a computer simulation model of the Stirling engine with non-sinusoidal piston motions that helps in predicting the engine thermal efficiency improvement at different parameters such as temperature at the cold space and hot space; engine rotational speed; engine dimension information and etc.

1.7 Outline of the Report

This report starts with the introduction of the Stirling engine; problem statement and objectives of this project; literature review on the types of thermodynamics models that can predict the Stirling engine performance; literature review on the different methods of generating non-sinusoidal piston motion; methods to develop mathematical model of the non-sinusoidal mechanisms; methods to develop mathematical model of the Stirling engine thermodynamics; analysis and discussion of the simulation result; possible future improvement needed for the project; conclusion of the project; references and appendices.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Ideal gamma Stirling engine operation starts with the isothermal compression process by reducing power piston volume while displacer piston not moving, as shown in Figure 2.1. During the isothermal compression process, the gas pressure inside the engine increased while gas temperature at cold space is kept constant by heat transfer to the constant temperature cooler.

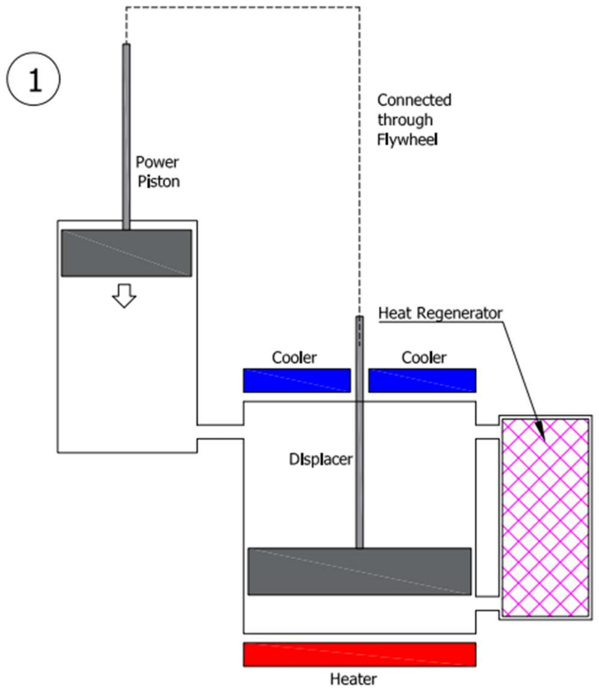


Figure 2.1: Ideal gamma Stirling engine operation step 1 (Kongtragool and Wongwises, 2003)

After the compression process, displacer piston pushes the high pressure cold temperature gas to the regenerator that heats up the gas and further increase the gas pressure. As the overall volume of the engine does not changed, the heating process is isochoric (constant volume).

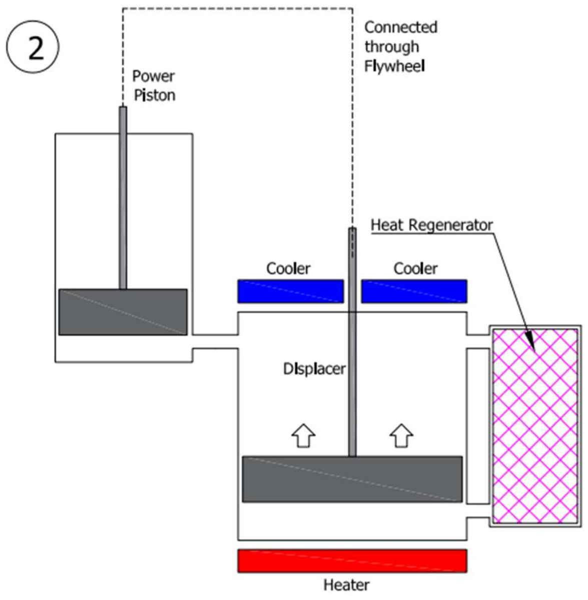


Figure 2.2: Ideal gamma Stirling engine operation step 2 (Kongtragool and Wongwises, 2003)

After the heating process, the heated high pressure gas is allowed to expand through power piston cylinder. During the expansion process, gas temperature at the hot space is kept constant by heat transfer from the constant temperature heater.

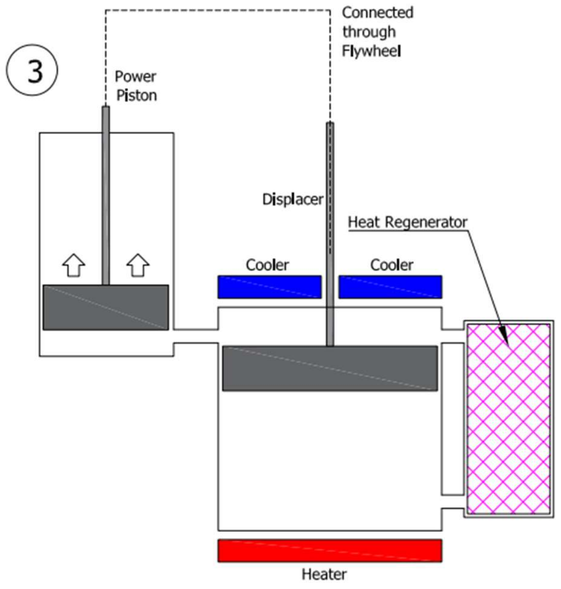


Figure 2.3: Ideal gamma Stirling engine operation step 3 (Kongtragool and Wongwises, 2003)

After the expansion process, the lower pressure gas is cooled down by transferring the gas through the regenerator by moving the displacer piston. As the overall volume of the engine does not changed, the cooling process is isochoric (constant volume). After the cooling process, the engine operation will be repeated again with the compression process.

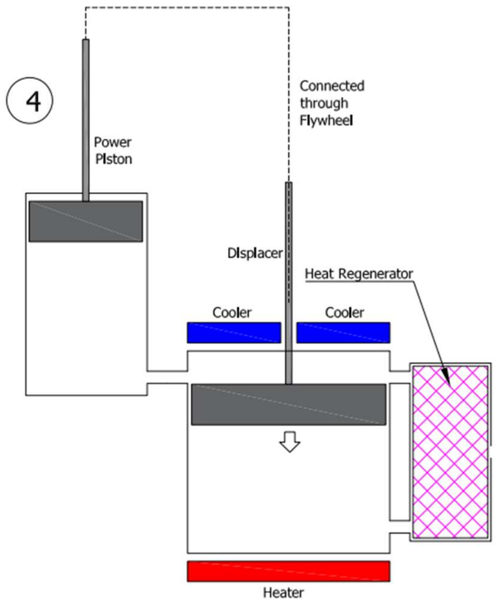


Figure 2.4: Ideal gamma Stirling engine operation step 4 (Kongtragool and Wongwises, 2003)

By refering to Figure 1.1 and Figure 2.1, ideal Stirling cycle starts with a constant low temperature compression process where it can be formulated by the following equation (Martini, 1983):

$$W_{compression(isothermal)} = P_1V_1 \ln\frac{V_2}{V_1} = M_{gas}R_{specific}T_1 \ln\frac{V_2}{V_1} \tag{2.1}$$

where

$W_{compression(isothermal}$	=	work done during the process
P_1	=	pressure of gas at the start of the process
V_1	=	gas volume at the start of the process
V_2	=	gas volume at the end of the process

M_{gas}	=	mass of gas going through the process
$R_{specific}$	=	specific gas constant
T_1	=	temperature of gas

It is followed by a constant volume heating process where heat energy stored in an ideal regenerator is transferred to the lower temperature gas that flows through it.

$$Q_{heating(isochoric)} = M_{gas} c_V (T_3 - T_2) \tag{2.2}$$

where

$Q_{heating(isochoric)}$	=	heat transfer by regenerator
M_{gas}	=	mass of gas going through the process
c_V	=	constant volume specific heat capacity at average temperature
T_2	=	temperature of gas at the start of the process
T_3	=	temperature of gas at the end of the process

The following processes, isothermal expansion process and isochoric cooling process, are the reverse of the above 2 processes.

$$W_{expansion(isothermal)} = P_3 V_3 \ln\frac{V_4}{V_3} = M_{gas} R_{specific} T_3 \ln\frac{V_4}{V_3} \tag{2.3}$$

$$Q_{cooling(isochoric)} = M_{gas} c_V (T_1 - T_4) \tag{2.4}$$

$$V_1 = V_4 \tag{2.5}$$

$$V_2 = V_3 \tag{2.6}$$

$$T_1 = T_2 \tag{2.7}$$

$$T_3 = T_4 \tag{2.8}$$

To study the Stirling engine in more details, many numerical or mathematical thermodynamics models of Stirling engine were developed. These thermodynamics models can be categorized into 1st order models, 2nd order models, 3rd order models and higher order models (Middleton, 2021). Some examples of thermodynamics models are Schmidt model (1st order model), Urieli and Berchowitz model (2nd order model), SIMPLE analysis (2nd order model) and the polytropic analysis of Stirling engine with various losses or PSVL (2nd order model). 3rd order models and higher order models are helpful in the final engine parameter optimisation, but they are not suitable for the Stirling engine early stage design process due to its complexity, and they will not be included in this review.

2.2 Stirling Engine Thermodynamics Models

Thermodynamics models are developed to simulate the internal conditions and performance of the Stirling engine under different operating conditions. Some information that can be extracted from the thermodynamics model simulation are gas temperature at different parts of the engine, internal gas pressure, work done by the engine, and heat transfer inside the engine.

2.2.1 Schmidt Model

Schmidt analysis or Schmidt model of Stirling engine is introduced by Gustav Schmidt in 1871. Schmidt model is categorised as 1st order model of Stirling engine as it provides closed-form solution in modeling the Stirling engine (Middleton, 2021). Table 2.1 shows the assumptions made when using Schmidt model to simulate the Stirling engine performance. Schmidt model is a semi-ideal isothermal simulation model of Stirling engine. According to assumption 11 in Table 2.1, the gas temperature change in compression space and expansion space is assumed to be zero throughout the engine cycle.

Table 2.1: Schmidt model assumptions (Middleton, 2021)

No.	Schmidt Model Assumptions
1	Regenerator is perfect
2	Instantaneous pressure of gas inside the engine spaces is homogeneous
3	Gas inside engine behaves like ideal gas
4	The gas inside engine is not leak to the surrounding
5	The temperature at heat exchangers are constant.
6	Cylinder wall and piston temperature do not change
7	The gas is in prefect mix in compression space and expansion space
8	Dead volume temperature is constant
9	Rotational speed of engine is constant
10	Gas flow is in steady state
11	Gas in compression space / cold space is in isothermal; gas in expansion space / hot space is in isothermal
12	Temperature gradient across the regenerator is linear
13	Kinetic and potential energy are negligible

Schmidt analysis of Stirling engine starts with mass balance equation that is shown below (Martini, 1983):

$$\mathrm{M_{engine}} = \mathrm{m_{hot}} + \mathrm{m_{cold}} + \mathrm{m_{regenerator}} \tag{2.9}$$

where

M_{engine}	=	total mass of gas inside engine
m_{hot}	=	mass of gas in engine hot space
m_{cold}	=	mass of gas in engine cold space
$m_{regenerator}$	=	mass of gas in regenerator

Schmidt model divides the internal volume of the engine into 3 spaces with different temperatures, which are hot space, cold space and regenerator space. The temperature in these 3 spaces will remain constant throughout the engine cycle. Temperature of gas inside regenerator space can be estimated using the log mean formulation (Martini, 1983):

$$T_{regenerator} = \frac{(T_{hot} - T_{cold})}{\ln\left(\frac{T_{hot}}{T_{cold}}\right)} \tag{2.10}$$

where

$T_{regenerator}$	=	temperature of gas inside regenerator space
T_{hot}	=	temperature of gas inside hot space
T_{cold}	=	temperature of gas inside cold space

By applying the ideal gas law to the mass of gas in the 3 spaces, the pressure of the gas inside engine can be determined (Middleton, 2021).

$$\mathrm{PV} = \mathrm{m}R_{specific}T \tag{2.11}$$

$$\mathrm{m}_{\mathrm{regenerator}} = \frac{\mathrm{PV}_{\mathrm{regenerator}}}{R_{specific}T_{\mathrm{regenerator}}} \tag{2.12}$$

$$\mathrm{m}_{\mathrm{hot}} = \frac{\mathrm{PV}_{\mathrm{hot}}}{R_{specific}T_{\mathrm{hot}}} \tag{2.13}$$

$$\mathrm{m}_{\mathrm{cold}} = \frac{\mathrm{PV}_{\mathrm{cold}}}{R_{specific}T_{\mathrm{cold}}} \tag{2.14}$$

$$\mathrm{P} = \frac{\mathrm{M}_{\mathrm{engine}}R_{specific}}{\frac{\mathrm{V}_{\mathrm{cold}}}{T_{\mathrm{cold}}} + \frac{\mathrm{V}_{\mathrm{regenerator}}}{T_{\mathrm{regenerator}}} + \frac{\mathrm{V}_{\mathrm{hot}}}{T_{\mathrm{hot}}}} \tag{2.15}$$

After the pressure of gas inside the engine is determined, the work done by the engine in one cycle can be calculated from the equation below using numerical integration (Martini, 1983).

$$\mathrm{W} = \oint \mathrm{P}\left(\frac{\mathrm{dV}_{\mathrm{hot}}}{d\theta} + \frac{\mathrm{dV}_{\mathrm{cold}}}{d\theta}\right) d\theta \tag{2.16}$$

where

W = work done by engine in one cycle

If $\frac{dV_{hot}}{d\theta}$ and $\frac{dV_{cold}}{d\theta}$ are functions are sinusoidal-like and they are not complex, W can be solved analytically, else, numerical calculation using computer is needed.

Simulation result of Schmidt model is expected to has large error when compared with the real Stirling engine performance due to its idealized working conditions. However, it is simple and it can give early insight on the performance of the engine under certain engine parameters such as piston size, heater temperature, cooler temperature and etc.

2.2.2 Urieli and Berchowitz Model

Urieli and Berchowitz (1984) proposed a 2nd order Stirling engine numerical model where the engine compression and expansion spaces are assumed to be adiabatic instead of isothermal. Expansion / compression space that are not too close to the heater /cooler is difficult to maintain its temperature during compression and expansion processes, especially in a real Stirling engine that runs at high speed. Assumptions made in Urieli and Berchowitz model to simulate Stirling engine are listed in Table 2.2.

Table 2.2: Urieli and Berchowitz model assumptions (Babaelahi and Sayyaadi, 2015)

No.	Urieli and Berchowitz Model Assumptions
1	Regenerator is perfect
2	Instantaneous pressure of gas inside the engine spaces is homogeneous
3	Gas inside engine behaves like ideal gas
4	The gas inside engine is not leak to the surrounding
5	The temperature at heat exchangers are constant.
6	Cylinder wall and piston temperature do not change
7	The gas is in prefect mix in compression space and expansion space
8	Rotational speed of engine is constant
9	Gas flow is in steady state
10	Processes in compression & expansion spaces are adiabatic
11	Temperature gradient across the regenerator is linear
12	Kinetic and potential energy are negligible

Instead of dividing the engine internal space into 3 sections, Urieli and Berchowitz model separates them into 5 sections. Hot/cold volume is divided into heat/cooler space where temperature change in this region will be zero; and compression/expansion space where temperature of gas in this space will varies.

$$\mathrm{M_{engine}} = \mathrm{m_{comp}} + \mathrm{m_{exp}} + \mathrm{m_{regenerator}} + \mathrm{m_h} + \mathrm{m_k} \tag{2.17}$$

$$\mathrm{V_{hot}} = \mathrm{V_{displacer(hot)}} + \mathrm{V_{dead(hot)}} + \mathrm{V_h} \tag{2.18}$$

$$\mathrm{V_{cold}} = \mathrm{V_{displacer(cold)}} + \mathrm{V_{dead(cold)}} + \mathrm{V_k} + \mathrm{V_{power(cold)}} \tag{2.19}$$

where

M_{engine}	=	total mass of gas inside engine
m_{comp}	=	mass of gas in engine piston compression space
m_{exp}	=	mass of gas in engine piston expansion space
m_h	=	mass of gas at heater
m_k	=	mass of gas at cooler

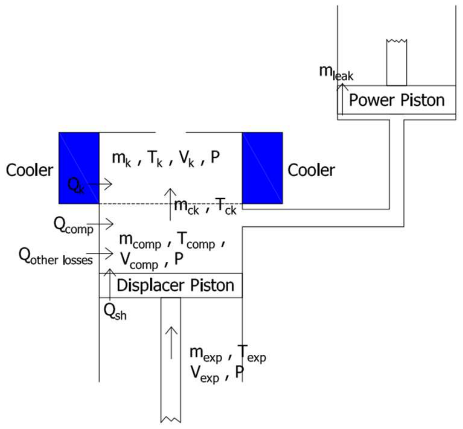


Figure 2.5: Illustration of mass and energy flow between compression space and cooler space. (Furmanek and Kropiwnicki, 2022)

In this model, each of the 5 spaces in the engine are set as a control volume, where each control volume has its own temperature and gas mass. In each control volume, the gas flow inside the control volume is assumed to be always in steady state. For each control volume, mass and energy balance equations can be constructed to study the condition of the gas in that volume. For example, by referring to Figure 2.5, simple energy balance equation of compression space of a gamma Stirling engine is shown below (Babaelahi and Sayyaadi, 2015):

$$\frac{dQ_{comp}}{d\theta}+\frac{dQ_{sh}}{d\theta}+\left(\dot{m}_{in}c_pT_{in}+\dot{m}_{out}c_pT_{out}\right)+\frac{dQ_{other\ losses}}{d\theta} = \frac{dW_{comp}}{d\theta}+c_V\frac{d\left(m_{comp}T_{comp}\right)}{d\theta} \tag{2.20}$$

where

Q_{comp}	=	heat energy transfer into the engine space (positive value if energy flow into the space and vice versa)
Q_{sh}	=	shuttle heat loss (always positive value for compression space, vice versa for expansion space)
$\dot{m}_{in}$	=	mass flow rate into the space per engine angle (always positive value)
$\dot{m}_{out}$	=	mass flow rate out of the space per engine angle (always negative value)
c_p	=	constant pressure specific heat capacity
$Q_{other\ losses}$	=	other energy losses (always negative value)
W_{comp}	=	work done (positive value if work transfer out of the space, negative value if work is put into the space)

As the compression process is assumed to be adiabatic, no heat transfer and losses occurred in the control volume.

$$\left(-\dot{m}_{ck}c_pT_{ck}\right)=P\frac{dV_{comp}}{d\theta}+c_V\frac{d\left(\frac{PV_{comp}}{R_{spec}}\right)}{d\theta} \tag{2.21}$$

$$\left(\frac{dm_{comp}}{d\theta}c_pT_{ck}\right)=P\frac{dV_{comp}}{d\theta}+\frac{c_V}{R_{spec}}\frac{dP}{d\theta}V_{comp}+\frac{c_V}{R_{spec}}\frac{dV_{comp}}{d\theta}P \tag{2.22}$$

where

$\dot{m}_{ck}$	=	mass flow from compression space to cooler space per engine angle (always positive value from compression space to cooler space, and

vice versa)

V_{comp}	=	Volume of compression space

Using the relationship of R_{spec}, c_p and c_V shown below, the above equation can be further simplified.

$$R_{spec} = c_p - c_V \tag{2.23}$$

$$\gamma = k_{isentropic} = \frac{c_p}{c_V} \tag{2.24}$$

$$\left(\frac{dm_{comp}}{d\theta}\right) = \left(\frac{R_{spec} + c_V}{R_{spec} c_p T_{ck}}\right) P \frac{dV_{comp}}{d\theta} + \frac{c_V}{R_{spec} c_p T_{ck}} \frac{dP}{d\theta} V_{comp} \tag{2.25}$$

$$\left(\frac{dm_{comp}}{d\theta}\right) = \left(\frac{1 - \frac{1}{\gamma} + \frac{1}{\gamma}}{R_{spec} T_{ck}}\right) P \frac{dV_{comp}}{d\theta} + \frac{\frac{dP}{d\theta} V_{comp} \frac{1}{\gamma}}{R_{spec} T_{ck}} \tag{2.26}$$

$$\left(\frac{dm_{comp}}{d\theta}\right) = \left(\frac{P \frac{dV_{comp}}{d\theta} + \frac{1}{\gamma} \frac{dP}{d\theta} V_{comp}}{R_{spec} T_{ck}}\right) \tag{2.27}$$

where

γ = heat capacity ratio of the gas

Similar steps are repeated on expansion space to determine the mass changes in expansion space.

$$\left(\frac{dm_{exp}}{d\theta}\right) = \left(\frac{P \frac{dV_{exp}}{d\theta} + \frac{1}{\gamma} \frac{dP}{d\theta} V_{exp}}{R_{spec} T_{he}}\right) \tag{2.28}$$

$\left(\frac{dm_{comp}}{d\theta}\right)$ and $\left(\frac{dm_{exp}}{d\theta}\right)$ are 2 new equations obtained from the energy balance equations of compression and expansion spaces. Next, $\frac{dP}{d\theta}$ in $\left(\frac{dm_{comp}}{d\theta}\right)$ and $\left(\frac{dm_{exp}}{d\theta}\right)$ equations and engine internal pressure, P, can be determined using the mass flow balance equation.

$$\mathrm{M_{engine}} = \mathrm{m_{comp}} + \mathrm{m_{exp}} + \mathrm{m_{regenerator}} + \mathrm{m_h} + \mathrm{m_k} + \mathrm{m_{leak}} \tag{2.29}$$

$$\mathrm{m_{regenerator}} = \frac{\mathrm{PV_{regenerator}}}{R_{specific}T_{\mathrm{regenerator}}} \tag{2.30}$$

$$\mathrm{m_h} = \frac{\mathrm{PV_h}}{R_{specific}T_{\mathrm{h}}} \tag{2.31}$$

$$\mathrm{m_k} = \frac{\mathrm{PV_k}}{R_{specific}T_{\mathrm{k}}} \tag{2.32}$$

$$\mathrm{m_{comp}} = \frac{\mathrm{PV_{comp}}}{R_{specific}T_{\mathrm{comp}}} \tag{2.33}$$

$$\mathrm{m_{exp}} = \frac{\mathrm{PV_{exp}}}{R_{specific}T_{\mathrm{exp}}} \tag{2.34}$$

$$\mathrm{P} = \frac{\mathrm{M_{engine}}R_{specific}}{\frac{\mathrm{V_{comp}}}{T_{\mathrm{comp}}} + \frac{\mathrm{V_{exp}}}{T_{\mathrm{exp}}} + \frac{\mathrm{V_h}}{T_{\mathrm{h}}} + \frac{\mathrm{V_{regenerator}}}{T_{\mathrm{regenerator}}} + \frac{\mathrm{V_k}}{T_{\mathrm{k}}}} \tag{2.35}$$

where

$\mathrm{m_{leak}}$ = mass of gas leaked out of the engine

Using ideal gas law and assume gas leakage is zero, $\frac{\mathrm{dP}}{\mathrm{d\theta}}$ equation can be obtained.

$$\frac{\mathrm{dm_{leak}}}{\mathrm{d\theta}} = 0 \tag{2.36}$$

$$\frac{\mathrm{d}}{\mathrm{d\theta}}(\mathrm{m_i}) = \frac{\mathrm{d}}{\mathrm{d\theta}}\left(\frac{\mathrm{PV_i}}{R_{specific}T_{\mathrm{i}}}\right);\ \mathrm{i = h, regenerator, k} \tag{2.37}$$

$$\frac{\mathrm{dm_i}}{\mathrm{d\theta}} = \frac{\mathrm{V_i}}{R_{specific}T_{\mathrm{i}}}\frac{\mathrm{dP}}{\mathrm{d\theta}};\ \mathrm{i = h, regenerator, k} \tag{2.38}$$

$$\frac{\mathrm{d}}{\mathrm{d\theta}}\left(\mathrm{M_{engine}}\right) = \frac{\mathrm{d}}{\mathrm{d\theta}}\left(\mathrm{m_{comp}} + \mathrm{m_{exp}} + \mathrm{m_{regenerator}} + \mathrm{m_h} + \mathrm{m_k} + \mathrm{m_{leak}}\right) \tag{2.39}$$

$$\frac{\mathrm{dm_{comp}}}{\mathrm{d\theta}} + \frac{\mathrm{dm_{exp}}}{\mathrm{d\theta}} + \frac{\mathrm{dm_{regenerator}}}{\mathrm{d\theta}} + \frac{\mathrm{dm_h}}{\mathrm{d\theta}} + \frac{\mathrm{dm_k}}{\mathrm{d\theta}} + \frac{\mathrm{dm_{leak}}}{\mathrm{d\theta}} = 0 \tag{2.40}$$

$$\frac{\mathrm{dP}}{\mathrm{d\theta}} = \frac{-\mathrm{P}\gamma\left(\frac{\mathrm{dV_{comp}}}{\mathrm{d\theta}} + \frac{\mathrm{dV_{exp}}}{\mathrm{d\theta}}\right)}{\frac{\mathrm{V_{comp}}}{T_{\mathrm{ck}}} + \frac{\mathrm{V_{exp}}}{T_{\mathrm{he}}} + \gamma\left(\frac{\mathrm{V_h}}{T_{\mathrm{h}}} + \frac{\mathrm{V_{regenerator}}}{T_{\mathrm{regenerator}}} + \frac{\mathrm{V_k}}{T_{\mathrm{k}}}\right)} \tag{2.41}$$

Changes of the temperature in compression and expansion space can be calculated from the differentiation of the ideal gas law equation.

$$T = \frac{\mathrm{PV}}{\mathrm{m}R_{specific}} \tag{2.42}$$

$$\frac{\mathrm{d}T}{\mathrm{d}\theta} = \frac{\mathrm{d}}{\mathrm{d}\theta}\left(\frac{\mathrm{PV}}{\mathrm{m}R_{specific}}\right) \tag{2.43}$$

$$\frac{\mathrm{d}T}{\mathrm{d}\theta} = \mathrm{T}\left[\frac{\left(\frac{\mathrm{dP}}{\mathrm{d}\theta}\right)}{\mathrm{P}} + \frac{\left(\frac{\mathrm{dV}}{\mathrm{d}\theta}\right)}{\mathrm{V}} - \frac{\left(\frac{\mathrm{dm}}{\mathrm{d}\theta}\right)}{\mathrm{m}}\right] \tag{2.44}$$

Heat transfer ($\frac{\mathrm{d}Q}{\mathrm{d}\theta}$) in heater, cooler and regenerator can be determined by the energy balance equation, using the same steps as the compression space analysis shown above. Table 2.3 shows the ordinary differential equations (ODEs) used in Urieli and Berchowitz model to simulate the Stirling engine performance, where some of the equations are already shown above.

Table 2.3: Urieli and Berchowitz model ODEs (Furmanek and Kropiwnicki, 2022)

No.	Urieli and Berchowitz Model ODEs
1	$\mathrm{P} = \frac{\mathrm{M_{engine}}R_{specific}}{\frac{\mathrm{V_{comp}}}{T_{\mathrm{comp}}} + \frac{\mathrm{V_{exp}}}{T_{\mathrm{exp}}} + \frac{\mathrm{V_h}}{T_{\mathrm{h}}} + \frac{\mathrm{V_{regenerator}}}{T_{\mathrm{regenerator}}} + \frac{\mathrm{V_k}}{T_{\mathrm{k}}}}$
2	$\frac{\mathrm{dP}}{\mathrm{d}\theta} = \frac{-\mathrm{P}\gamma\left(\frac{\mathrm{dV_{comp}}}{\mathrm{d}\theta} + \frac{\mathrm{dV_{exp}}}{\mathrm{d}\theta}\right)}{\frac{\mathrm{V_{comp}}}{T_{\mathrm{ck}}} + \frac{\mathrm{V_{exp}}}{T_{\mathrm{he}}} + \gamma\left(\frac{\mathrm{V_h}}{T_{\mathrm{h}}} + \frac{\mathrm{V_{regenerator}}}{T_{\mathrm{regenerator}}} + \frac{\mathrm{V_k}}{T_{\mathrm{k}}}\right)}$
3	$\mathrm{m_i} = \frac{\mathrm{PV_i}}{R_{specific}T_{\mathrm{i}}}$; $i = exp, comp, \mathrm{h}, \mathrm{regenerator}, \mathrm{k}$
4	$\left(\frac{\mathrm{dm_{exp}}}{\mathrm{d}\theta}\right) = \left(\frac{\mathrm{P}\frac{\mathrm{dV_{exp}}}{\mathrm{d}\theta} + \frac{1}{\gamma}\frac{\mathrm{dP}}{\mathrm{d}\theta}\mathrm{V_{exp}}}{\mathrm{R_{spec}T_{he}}}\right)$
5	$\left(\frac{\mathrm{dm_{comp}}}{\mathrm{d}\theta}\right) = \left(\frac{\mathrm{P}\frac{\mathrm{dV_{comp}}}{\mathrm{d}\theta} + \frac{1}{\gamma}\frac{\mathrm{dP}}{\mathrm{d}\theta}\mathrm{V_{comp}}}{\mathrm{R_{spec}T_{ck}}}\right)$

6	$\frac{\mathrm{dm_i}}{\mathrm{d\theta}} = \frac{\mathrm{V_i}}{R_{specific}T_\mathrm{i}}\frac{\mathrm{dP}}{\mathrm{d\theta}};\ \mathrm{i = h, regenerator, k}$
7	$\dot{\mathrm{m}}_\mathrm{ck} = -\frac{\mathrm{dm_{comp}}}{\mathrm{d\theta}}$
8	$\dot{\mathrm{m}}_\mathrm{he} = \frac{\mathrm{dm_{exp}}}{\mathrm{d\theta}}$
9	$\dot{\mathrm{m}}_\mathrm{kr} = \dot{\mathrm{m}}_\mathrm{ck} - \frac{\mathrm{dm_k}}{\mathrm{d\theta}}$
10	$\dot{\mathrm{m}}_\mathrm{rh} = \dot{\mathrm{m}}_\mathrm{he} - \frac{\mathrm{dm_h}}{\mathrm{d\theta}}$
11	If $\dot{\mathrm{m}}_\mathrm{ck}$>0, $\mathrm{T_{ck} = T_{comp}}$, else $\mathrm{T_{ck} = T_k}$
11	If $\dot{\mathrm{m}}_\mathrm{he}$>0, $\mathrm{T_{he} = T_h}$, else $\mathrm{T_{he} = T_{exp}}$
13	$\frac{\mathrm{dT_{exp}}}{\mathrm{d\theta}} = \mathrm{T_{exp}}\left[\frac{\left(\frac{\mathrm{dP}}{\mathrm{d\theta}}\right)}{\mathrm{P}} + \frac{\left(\frac{\mathrm{dV_{exp}}}{\mathrm{d\theta}}\right)}{\mathrm{V_{exp}}} - \frac{\left(\frac{\mathrm{dm_{exp}}}{\mathrm{d\theta}}\right)}{\mathrm{m_{exp}}}\right]$
14	$\frac{\mathrm{dT_{comp}}}{\mathrm{d\theta}} = \mathrm{T_{exp}}\left[\frac{\left(\frac{\mathrm{dP}}{\mathrm{d\theta}}\right)}{\mathrm{P}} + \frac{\left(\frac{\mathrm{dV_{comp}}}{\mathrm{d\theta}}\right)}{\mathrm{V_{comp}}} - \frac{\left(\frac{\mathrm{dm_{comp}}}{\mathrm{d\theta}}\right)}{\mathrm{m_{comp}}}\right]$
15	$\mathrm{T_{kr}}$=$\mathrm{T_k}$; $\mathrm{T_{rh}}$=$\mathrm{T_h}$
16	$\frac{\mathrm{dQ_k}}{\mathrm{d\theta}} = \mathrm{V_k}\frac{\mathrm{c_V}}{\mathrm{R}}\mathrm{dP} - \mathrm{c_p}(\mathrm{T_{ck}}\dot{\mathrm{m}}_\mathrm{ck} - \mathrm{T_{kr}}\dot{\mathrm{m}}_\mathrm{kr})$
17	$\frac{\mathrm{dQ_{regenerator}}}{\mathrm{d\theta}} = \mathrm{V_{regenerator}}\frac{\mathrm{c_V}}{\mathrm{R}}\mathrm{dP} - \mathrm{c_p}(\mathrm{T_{kr}}\dot{\mathrm{m}}_\mathrm{kr} - \mathrm{T_{rh}}\dot{\mathrm{m}}_\mathrm{hr})$
18	$\frac{\mathrm{dQ_h}}{\mathrm{d\theta}} = \mathrm{V_h}\frac{\mathrm{c_V}}{\mathrm{R}}\mathrm{dP} - \mathrm{c_p}(\mathrm{T_{rh}}\dot{\mathrm{m}}_\mathrm{rh} - \mathrm{T_{he}}\dot{\mathrm{m}}_\mathrm{he})$
19	$\frac{\mathrm{dW_{exp}}}{\mathrm{d\theta}} = \mathrm{PdV_{exp}}$
20	$\frac{\mathrm{dW_{comp}}}{\mathrm{d\theta}} = \mathrm{PdV_{comp}}$

Using the equations in Table 2.3, engine parameters, $\mathrm{T_{exp}}$, $\mathrm{T_{comp}}$, $\mathrm{Q_k}$, $\mathrm{Q_{regenerator}}$, $\mathrm{Q_h}$, $\mathrm{W_{exp}}$ and $\mathrm{W_{comp}}$ can be obtained by applying Runge-Kutta fourth order (RK4) method on these differential equations $\frac{\mathrm{dT_{exp}}}{\mathrm{d\theta}}$, $\frac{\mathrm{dT_{comp}}}{\mathrm{d\theta}}$, $\frac{\mathrm{dQ_k}}{\mathrm{d\theta}}$, $\frac{\mathrm{dQ_{regenerator}}}{\mathrm{d\theta}}$, $\frac{\mathrm{dQ_h}}{\mathrm{d\theta}}$, $\frac{\mathrm{dW_{exp}}}{\mathrm{d\theta}}$ and $\frac{\mathrm{dW_{comp}}}{\mathrm{d\theta}}$ (Furmanek and Kropiwnicki, 2022). The analysis will be done several iterations until the $\mathrm{T_{exp}}$ and $\mathrm{T_{comp}}$ at 0° engine shaft angle and 360° engine shaft angle are the same, where the engine in the simulation is in steady condition. At the start of the analysis, $\mathrm{T_{exp}}$ and $\mathrm{T_{comp}}$ are set to be the same as $\mathrm{T_h}$ and $\mathrm{T_k}$; and M_{gas} information needs to be given by the user. At the end of each iteration, the engine parameters at this iteration will be the initial condition of next iteration.

Simulation result of Stirling engine using Urieli and Berchowitz model is expected to has higher accuracy than the Schmidt model in many cases. Keeping the temperature constant in entire hot space and entire cold space inside the engine are difficult especially when the engine run at high speed. When the running speed is high, the rate of heat transfer at cooler and heater cannot keep up with the changing of temperature due to the volume and pressure changes. Therefore, adiabatic assumption near the piston cylinder space makes more sense if losses are not considered. The Urieli and Berchowitz model simulation results may have differences if the expansion space, heater space, compression space and cooler space are defined differently.

Urieli and Berchowitz (1984) also proposed to improve the simulation result by correcting the calculated engine parameters by considering the losses. This analysis is called SIMPLE analysis.

2.2.3 Losses Analysis of Stirling Engine

Schmidt model and Urieli and Berchowitz model are two thermodynamics models that simulates the Stirling engine based on idealized condition of the gas and assuming no energy losses to the surrounding. Losses analysis can be added to the thermodynamics models to obtain simulation results closer to a real Stirling engine. Table 2.4 summarized some losses of a real Stirling engine.

Applying finite speed thermodynamics analysis to correct the engine work done by the pistons can improve the accuracy of the engine simulation (Furmanek and Kropiwnicki, 2022). Unlike classical thermodynamics analysis that assume the gas condition is always homogenous and isotropic, finite speed thermodynamics analysis takes into account the effects by the gas molecules average velocity and the piston velocity (Petrescu et al., 2016). The processes are now irreversible. Work losses due to piston finite speed and gas mechanical friction losses at the compression or expansion space can be shown in below equation (Babaelahi and Sayyaadi, 2015):

$$W_{\text{loss(finite speed and friction)}} = \int \left(\pm a_{\text{finite}} \frac{u_{\text{piston}}}{c_{\text{gas speed}}} \pm \frac{\Delta P_{\text{friction}}}{P} \right) dV \quad (2.45)$$

$$a_{\text{finite}} = \sqrt{3\gamma} \quad (2.46)$$

$$c_{\text{gas speed}} = \sqrt{3RT} \quad (2.47)$$

Where

u_{piston}	=	piston velocity
$c_{\text{gas speed}}$	=	gas molecules average speed
$\Delta P_{\text{friction}}$	=	pressure drop due to mechanical friction

'+' sign for compression space analysis

'-' sign for expansion space analysis

There are also pressure or work losses at the heat exchangers due to the friction effect on the gas flow (Furmanek and Kropiwnicki, 2022). This throttle losses at each heat exchangers can be described by the following equation (Babaelahi and Sayyaadi, 2015):

$$\Delta P_{\text{throttle}} = -\frac{2 f_{\text{Re}} \mu_{\text{gas}} u_{\text{gas}} V}{{d_{\text{hydraulic}}}^{2} A_{\text{cross-section}}} \quad (2.48)$$

where

f_{Re}	=	Reynolds friction factor
μ_{gas}	=	viscosity of gas
u_{gas}	=	velocity of gas
V	=	volume of gas in heat exchanger (cooler, heater or regenerator)
$d_{\text{hydraulic}}$	=	hydraulic diameter
$A_{\text{cross-section}}$	=	cross sectional area

Another factor that contributes to the losses in the engine is the regenerator efficiency. Regenerator efficiency can be calculated using the below equation (Furmanek and Kropiwnicki, 2022):

$$\varepsilon_{regenerator} = \frac{NTU_{regenerator}}{1 + NTU_{regenerator}} \tag{2.49}$$

$$NTU_{regenerator} = \frac{(St)L_{regenerator}}{\mathrm{d}_{\mathrm{hydraulic(regenerator)}}} \tag{2.50}$$

$$St = 0.023Re^{-0.2}Pr^{-0.6} \tag{2.51}$$

where

$\varepsilon_{regenerator}$	=	Regenerator efficiency
$NTU_{regenerator}$	=	number of transfer unit
$L_{regenerator}$	=	regenerator length
$\mathrm{d}_{\mathrm{hydraulic(regenerator)}}$	=	hydraulic diameter
St	=	Stanton number
Re	=	Reynolds number
Pr	=	Prandtl number

The non-ideal regenerator will cause the gas temperature in heater / cooler space to be slightly different from the heater/cooler wall temperature (Babaelahi and Sayyaadi, 2015).

$$T_{h(new)} = T_{h(wall)} - \frac{f_{rotation}}{h_h A_{wh}}\left[Q_{h(initial)} + Q_{r(ideal)}\left(1 - \varepsilon_{regenerator}\right)\right] \tag{2.52}$$

$$T_{k(new)} = T_{k(wall)} - \frac{f_{rotation}}{h_k A_{wk}}\left[Q_{k(initial)} - Q_{r(ideal)}\left(1 - \varepsilon_{regenerator}\right)\right] \tag{2.53}$$

$$h_{h,k} = \frac{0.0791\mu_{gas}c_p Re^{0.75}}{2\mathrm{d}_{\mathrm{hydraulic}}\mathrm{Pr}} \tag{2.54}$$

where

$T_{(new)}$	=	corrected temperature
$T_{(wall)}$	=	wall temperature
$f_{rotation}$	=	frequency of engine rotation
$\mathrm{A_w}$	=	wetted area
$h_{k,h}$	=	heat transfer coefficient

The temperature difference between 2 sides of the piston will induce shuttle conduction heat loss. Heat transfer of gas from 2 sides occurred at the gap between piston and cylinder wall where distance between 2 sides is the lowest. Shuttle heat loss at displacer piston can be estimated by (Martini, 1983):

$$Q_{shuttle} = \frac{Y_k Z_k {S_{piston}}^2 k_{gas} d_{displacer} (T_{exp} - T_{comp})}{J_{gap} L_{displacer}} \tag{2.55}$$

where

Y_k	=	friction and wall properties
Z_k	=	wave form factor, if it is sinusoidal wave form, it can be approximated as π/8
S_{piston}	=	piston stroke
k_{gas}	=	gas thermal conductivity
$d_{displacer}$	=	piston diameter
J_{gap}	=	gap between piston and wall
$L_{displacer}$	=	piston length

Gas leakage at the engine is another loss that contributes to the inaccuracy of the thermodynamics model. Mass of gas leakage can be estimated as (Urieli, I. and Berchowitz, 1983):

$$m_{leak} = \pi d_{piston} \frac{P + P_{Buffer}}{4 R_{specific} T_{gas}} \left(u_{piston} J_{gap} - \frac{{J_{gap}}^3}{6 \mu_{gas}} \frac{P - P_{Buffer}}{L_{piston}} \right) \tag{2.56}$$

where

d_{piston}	=	piston diameter
L_{piston}	=	piston length
T_{gas}	=	gas temperature
P_{Buffer}	=	assumed to be 100kPa

Table 2.4: Losses in real Stirling engine (Nicol-Seto, 2021)

No.	Losses in Real Stirling Engine
1	Piston finite speed effect
2	Gas pressure losses due to friction effect at piston spaces
3	Gas pressure losses at heat exchangers
4	Non-ideal regenerator, cooler and heater
5	Heat losses due to conduction (including shuttle loss)
6	Mass leakage
7	Gas compressibility
8	Non-ideal piston motion
9	Difficulties in thermal control at heater and cooler
10	Friction losses of mechanical parts

2.2.4 Polytropic Analysis of Stirling engine with Various Losses

Babaelahi and Sayyaadi (2015) has proposed the adding of polytropic heat loss analysis in the simulation of Stirling engine. By referring to Figure 2.5, the control volume analysis will be applied to the compression space and expansion space similar to Urieli and Berchowitz model. However, the heat transfer $\frac{dQ_{comp}}{d\theta}$ is no longer zero as the process is no longer adiabatic.

$$Q_{comp(polytopic)} = m_{comp}c_{n(comp)}\left(T_0 - T_{comp}\right) \tag{2.57}$$

$$c_n = c_v \frac{n_{polytopic} - k_{isentopic}}{n_{polytopic} - 1} \tag{2.58}$$

$$n_{polytopic} = -\frac{V\frac{dP}{d\theta}}{P\frac{dV}{d\theta}} \tag{2.59}$$

$$\frac{dQ_{comp(polytopic)}}{d\theta} = \frac{dm_{comp}}{d\theta}c_{n(comp)}\left(T_0 - T_{comp}\right) - m_{comp}c_{n(comp)}\left(\frac{dT_{comp}}{d\theta}\right) \tag{2.60}$$

$$\frac{dQ_{exp(polytopic)}}{d\theta} = \frac{dm_{exp}}{d\theta} c_{n(exp)}\left(T_0 - T_{exp}\right) - m_{exp}c_{n(exp)}\left(\frac{dT_{exp}}{d\theta}\right) \tag{2.61}$$

where

T_0	=	surrounding temperature
$n_{polytopic}$	=	polytropic index
$k_{isentopic}$	=	γ = heat capacity ratio

Adding $\frac{dQ_{comp}}{d\theta}$, $\frac{dQ_{exp}}{d\theta}$, shuttle heat loss and mass leakage, in the energy balance and mass balance analysis will generates new set of equations as in Table 2.3. Similar to Urieli and Berchowitz model, RK4 method will be used to obtain engine parameters, T_{exp}, T_{comp}, Q_k, $Q_{regenerator}$, Q_h, W_{exp} and W_{comp}. The analysis will do several iterations until the T_{exp}, T_{comp}, $n_{polytopic(exp)}$ and $n_{polytopic(comp)}$ at 0^o engine shaft angle and 360^o engine shaft angle are the same. To make the simulation result more accurate, losses analysis

2.2.5 Comparison between Experimental Results and Simulation Results of Different Thermodynamics Models

There are more thermodynamics models of Stirling engine other than the above-mentioned ones. Results of different simulation models at different operating condition of gamma Stirling engine were studied by many researchers.

Li, Grosu and Queiros-Condé (2016) have compared the simulation result of their thermodynamics model with a real solar powered gamma Stirling engine prototype as shown in Figure 2.6. Thermodynamics model used is isothermal model with finite speed analysis. Non-ideal regenerator, piston finite speed effect, gas hysteresis loss, mass leakage loss, heat conduction loss, shuttle heat loss are considered in their model. The real Stirling engine data and simulation data are shown in Table 2.5.

Sowale et al. (2018) has compared the simulation result of the thermodynamics model built by them with the experimental results obtained by Gheith, Aloui and Nasrallah (2012). Their model has included heat conduction loss, shuttle loss, non-ideal regenerator in the analysis in their initial adiabatic model. 6 control volumes are used in their model instead of 5 control volumes, where generator space is divided into 2 control volumes. Engine data at different initial charging pressure (initial gas mass) are used for comparison. Table 2.6 shows the Stirling engine data of their thermodynamics model.

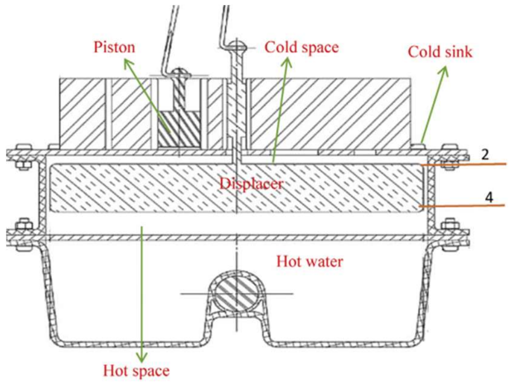


Figure 2.6: Illustration of solar powered gamma Stirling engine by Li, Grosu and Queiros-Condé (2016)

Table 2.5: Stirling engine data by Li, Grosu and Queiros-Condé (2016)

Description	Data
Temperature at Expansion Volume	317.9 K
Temperature at Compression Volume	305.6 K
Temperature at Cold Sink	303.9 K
Environment Temperature	299.1 K
Gas	Air
Engine Frequency	0.4189
Displacer Piston Stroke	9mm
Power Piston Stroke	21mm
Phase shift between pistons	90°
Initial engine pressure	101325 Pa
Power piston length	21.3mm
Power piston diameter	22mm
Displacer piston length	24.9mm
Cylinder inner diameter	176mm

Displacer piston diameter	172mm
Power Piston Clearance	0.08mm
Simulated Shaft Work per cycle after deduction of heat losses and work losses	0.00351816 J
Experimental Shaft Work per cycle	0.003213 J
Simulation result error	9.49%

Table 2.6: Stirling engine data by Sowale et al. (2018)

Description	Data
Experimental Mean Pressure (bar)	10
Temperature at Heater (°C)	400
Temperature at Cooler (°C)	15
Phase shift between pistons (°)	90
Gas	Air
Engine Rotational Speed (rpm)	600
Displacer Piston Stroke (meter)	0.120
Power Piston Stroke (meter)	0.145
Displacer piston diameter (meter)	0.095
Power piston diameter (meter)	0.08
Outer diameter of regenerator (meter)	0.134
Inner diameter of regenerator (meter)	0.098
Length of regenerator (meter)	0.05
Regenerator Material	Stainless Steel 304L
Regenerator Porosity	0.9

Regenerator Density (g/cm^3)	7.85
Regenerator Thermal capacity [J/(kg K)]	477
Regenerator Thermal conductivity [W/(m K)]	26
Experimental Brake Power (W) at initial charge pressure of 3bar	150
Simulation Brake Power (W) at initial charge pressure of 3bar	146
Experimental Brake Power (W) at initial charge pressure of 5bar	275
Simulation Brake Power (W) at initial charge pressure of 5bar	269
Experimental Brake Power (W) at initial charge pressure of 8bar	308
Simulation Brake Power (W) at initial charge pressure of 8bar	303

Araoz et al. (2015) has compared their thermodynamics model simulation result with the performance of the prototype Genoa gamma Stirling engine built by GENOA Stirling S.R.L. company. Their thermodynamics model starts with adiabatic model analysis and has included energy losses analysis to improve the accuracy of the simulation result. Table 2.7 shows the Stirling engine data of their thermodynamics model.

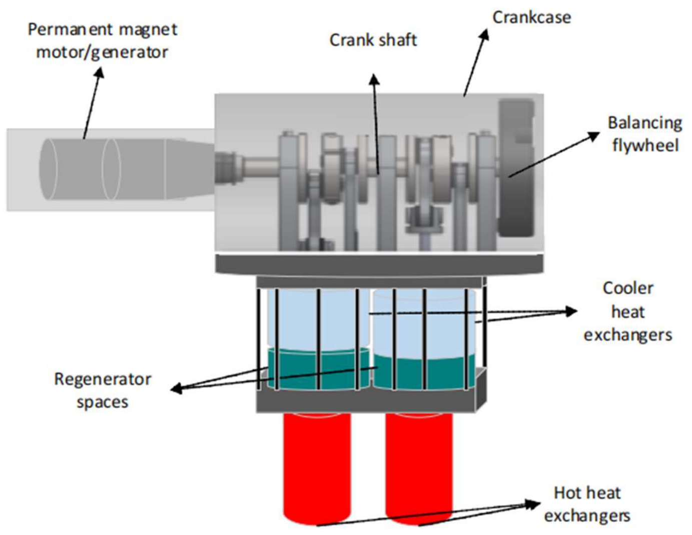


Figure 2.7: Illustration of Genoa engine (Araoz et al., 2015)

Table 2.7: Stirling engine data by Araoz et al. (2015)

Description	Data
Measured Frequency (Hz)	5.17
Dead Volume ratio (displacer dead volume/displacer swept volume)	1.3353
Dead Volume ratio (power piston swept volume/displacer swept volume)	0.3684
Charging Pressure (bar)	12.5
Measured T_h (K)	816.4
Measured T_k (K)	322.4
Experimental Brake Power (W)	54.72
Simulated Brake Power (W)	53.59

Alfarawi, Al-Dadah and Mahmoud (2016) has compared their thermodynamics model simulation result with the performance of the gamma Stirling engine designed by Dieter Viebach. Their thermodynamics model has included shuttle loss, conduction loss, pumping loss and mechanical loss in the analysis. Table 2.8 shows the Stirling engine data of their thermodynamics model.

Table 2.8: Stirling engine data by Araoz et al. Alfarawi, Al-Dadah and Mahmoud (2016)

Description	Data
Rotational Speed (rpm)	500
Piston Stroke (mm)	75
Bore diameter of power piston (mm)	85
Bore diameter of displacer piston (mm)	96
Charge pressure (bar)	10
Gas	N_2
Type of regenerator	Random Fiber
Regenerator wire diameter (micron)	31
Regenerator Porosity	0.9
Temperature at Heater (°C)	650
Temperature at Cooler (°C)	15
Compression ratio	1.3

Laazaar and Boutammachte (2022) has compare their thermodynamics model simulation result with the GPU-3 Stirling engine performance. Their model is based on Urieli and Berchowitz adiabatic model and added non-ideal regenerator loss, piston finite speed effect, heat exchangers pressure drops, mass leakage loss, heat conduction loss and shuttle heat loss in their model analysis. Table 2.9 shows the dimension data and operation data of GPU-3 engine while Table 2.10 shows the simulations results of different thermodynamics models.

In summary, isothermal model and adiabatic model of Stirling engine are the bases of many second order Stirling engine thermodynamic models mentioned above. With the addition of losses analysis, the simulated results of the engine can be very close to the performance of a real Stirling engine with acceptable errors.

Table 2.9: Dimension data and operation data of GPU-3 engine (Urieli and Berchowitz, 1984)

Description	Data
Temperature at Heater (K)	977
Temperature at Cooler (K)	288
Engine Frequency (Hz)	41.7

Gas mass (g)	1.1362
Phase shift between pistons (°)	90
Gas	Helium
Piston Stroke (mm)	31.2
Piston Bore (mm)	69.9
Expansion Volume Clearance (mm^3)	30.52
Compression Volume Clearance (mm^3)	28.68
Eccentricity (mm)	20.8
Volume of heater (cm^3)	70.88
Heater average tube length (mm)	245.3
Heater tube external diameter (mm)	4.83
Heater tube internal diameter (mm)	3.02
Heater tube number	40
Volume of regenerator (cm^3)	50.55
Regenerator Length (mm)	22.6
Regenerator internal diameter (mm)	22.6
Regenerator tube diameter (mm)	0.04
Regenerator number	8
Regenerator Porosity	0.697
Regenerator Material	Stainless Steel
Cooler Volume (cm^3)	13.18
Cooler average tube length (mm)	46.1
Cooler tube external diameter (mm)	1.59
Cooler tube internal diameter (mm)	1.09
Cooler tube number	312

Table 2.10: Simulation Results of GPU-3 engine (Laazaar and Boutammachte, 2022)

Thermodynamics model	Output Power per cycle (kW)	Error in Power (%)	Thermal Efficiency (%)	Error in Efficiency (%)
Real GPU-3 engine	3	-	21.3	-

Laazaar and Boutammachte model	3.9	30.00	24.19	2.89
Ideal adiabatic	8.3	176.60	62.3	41
Ideal polytropic	7.73	157.60	60.36	39.06
Urieli &Berchowitz (Simple analysis)	6.70	123.30	52.50	31.20
Developed analytical isothermal model	6.09	103.00	52.90	31.60
FST	4.8	60.00	29.27	7.97
Timoumi	4.27	42.30	38.50	17.20
Third order analysis	4.26	42.00	42.00	20.70
Simple polytropic	4.16	38.66	25.23	3.93
CAFS	4.11	37.00	36.20	14.90

2.3 Stirling Engine Improvement With Efficient Regenerator

Different types of regenerators are studied by many authors to improve the Stirling engine performance. Without an efficient regenerator, constant volume heating / cooling process at the regenerator cannot keep the temperature at heater /cooler space same as the wall temperature of the heater / cooler. Beside the flow losses at the regenerator, more heat energy transfer to / from the heater/cooler causes the drop in engine thermal efficiency.

In general, a regenerator can be defined as a porous solid matrix that allows heat exchange between the solid matrix and the gas that flows through it. Conventional regenerator materials are woven screen / wire mesh and random fiber (Yu et al., 2022). Conventional regenerator has large surface area over volume ratio and good thermal transfer coefficient. However, the flow loss or pressure drop of the gas that flows through the regenerator is also high due to the flow separation in the matrix and the eddy generated inside the flow (Yu et al., 2022). To reduce the flow losses in the regenerator, Yu et al. (2022) propose a new annular constructal bifurcation regenerator

(CBR) as shown in Figure 2.8. In CBR, the wire mesh is aligned and slanted towards the flow direction to give a smoother flow of gas through the regenerator while keeping high contact surface area between the wire mesh and the gas. The CBR is shown to has much lower flow losses than conventional wire mesh regenrator while only providing slightly lower thermal performance than conventional regenerator (Yu et al., 2022).

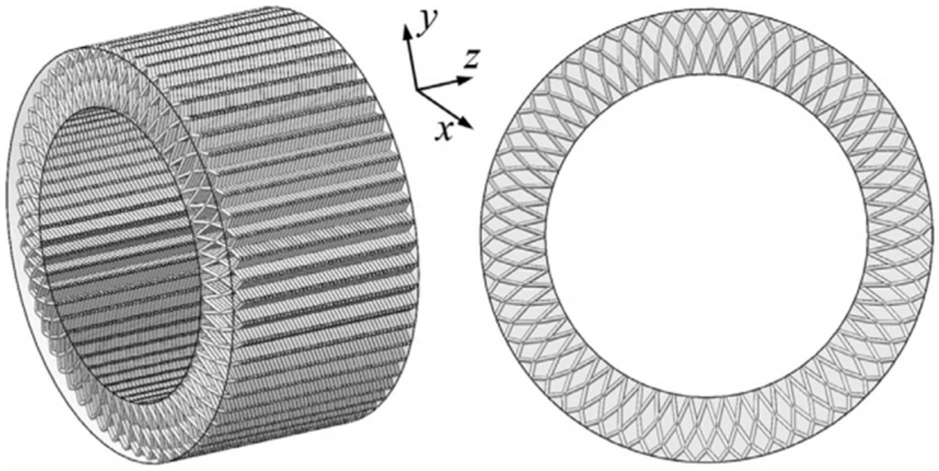


Figure 2.8: Annular CBR (Yu et. al., 2022)

Wang et al. (2021) has studied the performance of the pin-array stack regenerator for Stirling engine. According to Wang et al. (2021), pin-array stack regenerator has low flow resistance due to the pin-array that aligned parallel to the flow as shown in Figure 2.9, while having acceptable thermal transfer coefficient of up to 10^5 W/(m^2 K).

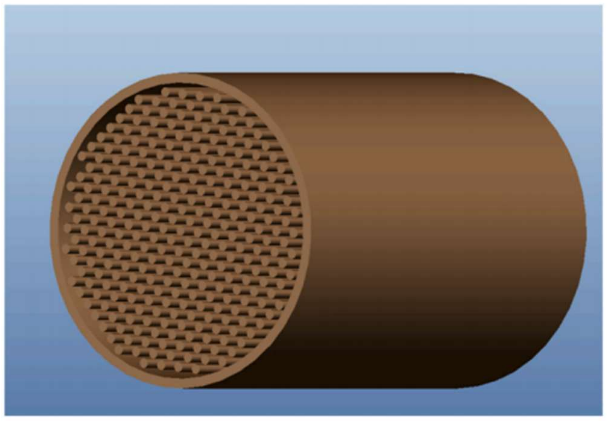


Figure 2.9: Pin-array stack regenerator (Wang et al., 2021)

With proper design, parallel plate regenerator (PPR) can have regenerative effectiveness as high as 93.9% and with flow resistance much less than conventional wire mesh regenrator (Liu et al., 2022). Longer plate and lower plate thickness increases the regenerative effectiveness while appropriate plate gap gives low flow resistance.

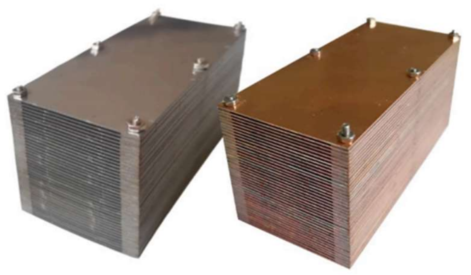


Figure 2.10: Parallel plate regenerator (Liu et al., 2022)

Sinusoidal corrugated-channel regenerator is introduced by Yu et al. (2024) to improve the performance of the Stirling engine. It has low flow resistance by reduces the flow separation and flow stagnation of the gas passing through when the channels are aligned to the gas flow direction. Its inclining surface towards the gas flow direction also allows good regenerative effectiveness due to high surface contact with the gas.

In summary, the design of regenerator needs to balance between the regenerative effectiveness and the flow resistance. Good regenerator needs high regenerative effectiveness while having an acceptable flow losses.

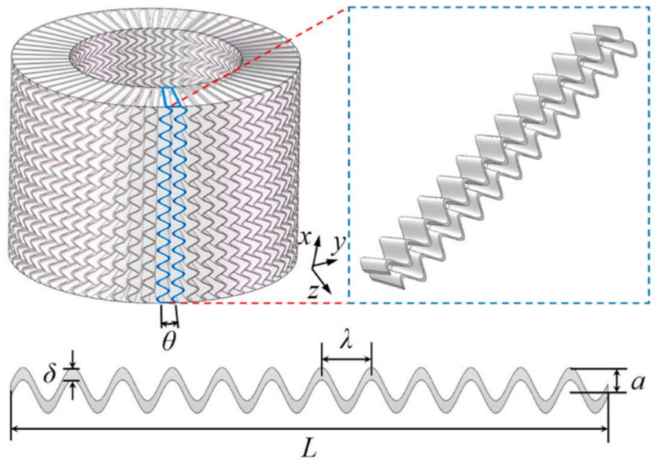


Figure 2.11: Sinusoidal corrugated-channel regenerator (Yu et al., 2024)

2.4 Stirling Engine Improvement With Non-sinusoidal Piston Motion

One of losses that decreases the efficiency and work output of the Stirling engine is the non-ideal piston motion (Nicol-Seto and Nobes, 2021). Various mechanisms have been proposed by researchers to improve Stirling engine output.

2.4.1 Free Piston Mechanism

The idea of using free pistons mechanism to improve Stirling engine efficiency are studied by many authors such as Walker et al. (1985), Dai et al. (2021), Gopal, Duke, and Clucas (2009) and Gopal (2012). The free piston mechanism decouples displacer piston from the engine main shaft, and its motion is controlled by DC servomotor or other electrical/magnetic drive unit. Power piston motion is also possible to be altered using the electrical/magnetic control mechanism to be closer to ideal piston motion.

A free piston Stirling engine proposed by Gopal (2012) uses an electrical linear servo drive unit to generate displacer piston motion with dwells that is more resembles to the ideal piston motion. Figure 2.12 shows a free piston Stirling engine propose by Gopal (2012), while Figure 2.13 shows the piston motion of the free piston Stirling engine propose. It is reported that a 13.5% improvement of efficiency at 750rpm of engine rotation when compared with conventional sinusoidal displacer piston motion. (Gopal, 2012).

Free piston Stirling engine has the flexibility of generating any kind of piston motion using electronic control system. However, the building of the electronic control system is expensive and the electrical losses may adding the complexity of the efficiency analysis.

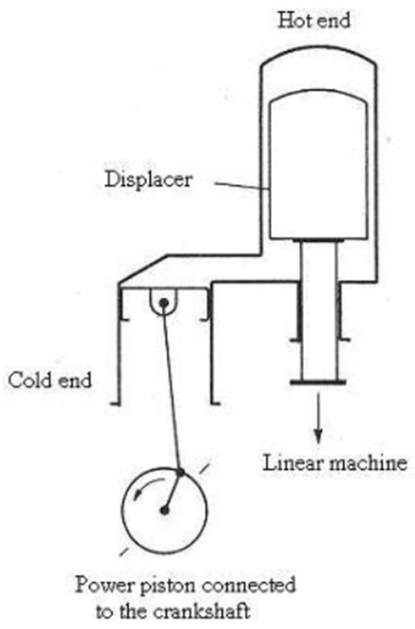


Figure 2.12: Free piston Stirling engine concept proposed by Gopal (2012)

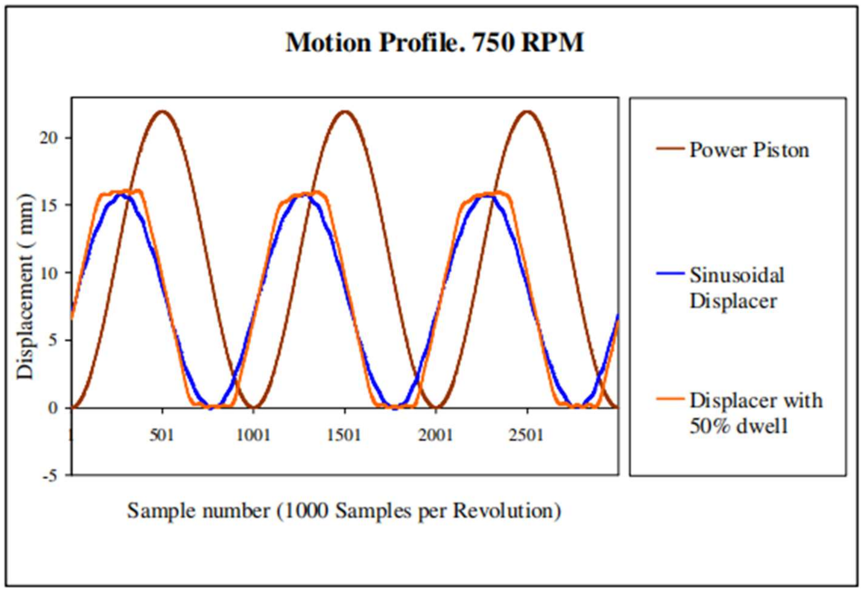


Figure 2.13: Free piston Stirling engine piston motion (Gopal, 2012)

2.4.2 7-bar linkage mechanism

Dehelean and Ciupe (2009) propose 7-bar linkage mechanism that able to generate a single dwell on the compression cylinder motion of an Alpha Stirling Engine to enable more gas expansion work goes to the engine power output. Figure 2.14 and Figure 2.15 show the proposed 7-bar linkage mechanism and the generated piston motions.

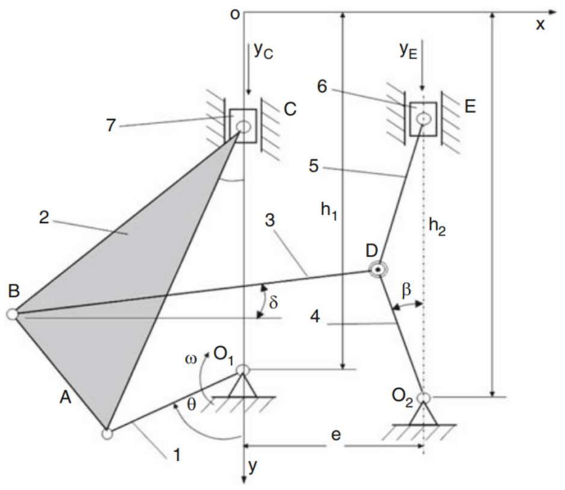


Figure 2.14: 7-bar linkage mechanism that generate single dwell (Dehelean and Ciupe, 2009)

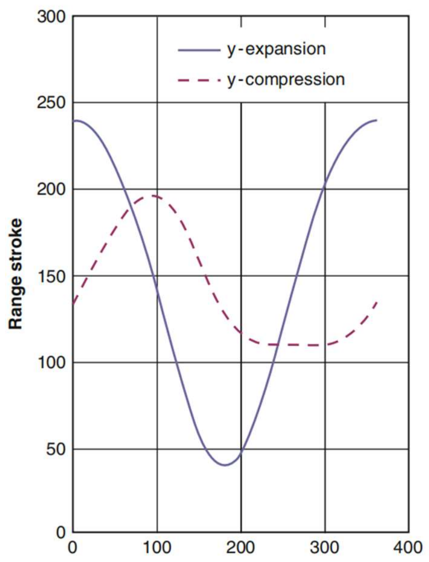


Figure 2.15: Piston motions of Stirling engine with 7-bar linkage mechanism (Dehelean and Ciupe, 2009)

Dehelean et al. (2010) infer that introduction of dwell in the compression cylinder motion of an alpha Stirling engine can improve the indicated power of the engine by observing the change in PV diagram. However, experimental work to prove the performance of a real Stirling engine using the 7-bar linkage mechanism is less studied. It is also observed that the improvement of indicated power using the 7-bar linkage mechanism proposed maybe small since it can only make small part of the piston motion similar to the ideal piston motion of Stirling engine.

2.4.3 Non-circular gear mechanism

Fang et al. (1996) has proposed the use of oval elliptical gears to generate more ideal volume variation in the compression and expansion spaces.

Experimental work on Stirling engine with oval gear drive is done by Nicol-Seto and Nobes (2021) to study the improvement of the engine efficiency. Figure 2.16 shows the schematic diagram of the prototype Stirling engine. By using oval gears (e=1/5), the power output improves to 5.84W from the conventional sinusoidal 5.61W. Nicol-Seto and Nobes (2021) also stated that some design of the oval gears such as e=1/3 may reduce the engine output instead of improving the engine output. According to Nicol-Seto and Nobes (2021), the reduction of engine output is suspected to be caused by the too much decrease in heating/cooling time in the cycle while the expansion/compression time increases.

Although the use of non-circular gears can improve the engine performance, the construction of the complex shape gears for different piston motion is complex and difficult, especially the gear teeth. Building prototype of complex shape gears may take a long time and it requires manufacturing methods such as 3D printing, CNC machining or casting.

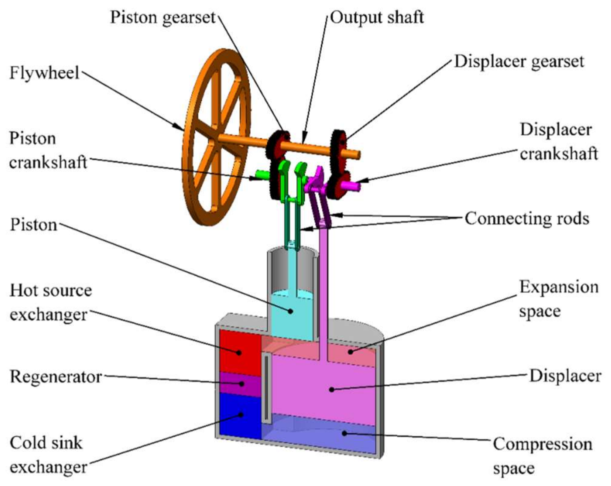


Figure 2.16: Schematic diagram of Stirling engine with oval gear drive (Nicol-Seto and Nobes, 2021)

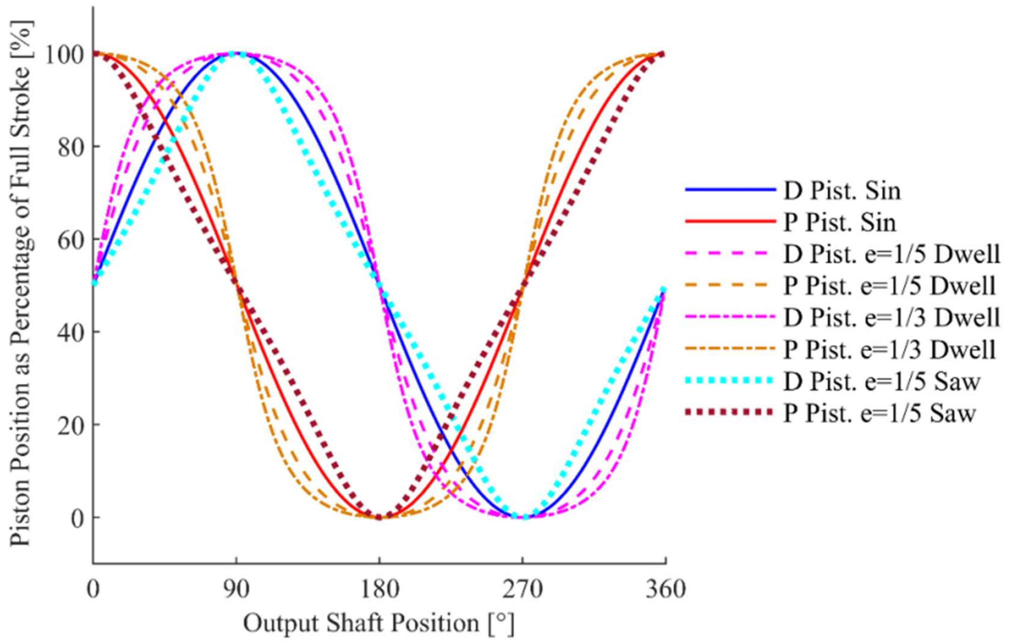


Figure 2.17: Piston motions of Stirling engine with oval gear drive (Nicol-Seto and Nobes, 2021)

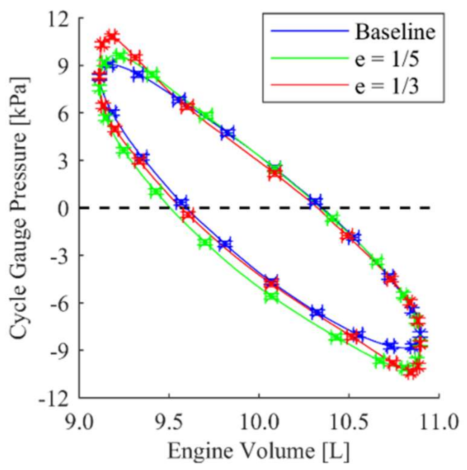


Figure 2.18: PV diagram of Stirling engine with oval gear drive (Nicol-Seto and Nobes, 2021)

2.4.4 Modified Scotch Yoke Mechanism

Melvin and Thomas (1993) have proposed the use of modified Scotch Yoke mechanism to generate double dwells in piston motions of a 2 pistons internal combustion engine.

Middleton (2021) has suggested the use of modified Scotch Yoke mechanism to generate customized piston motion for Stirling engine. However, not much experimental or simulation works have been published to study the engine performance.

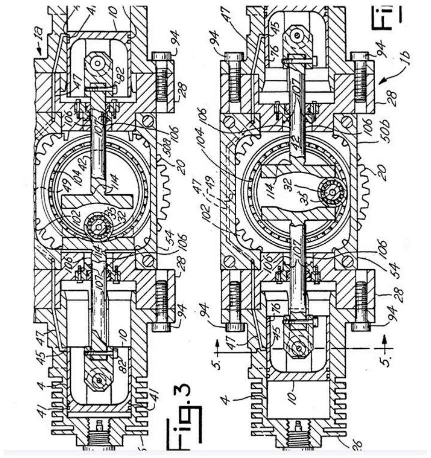


Figure 2.19: Modified Scotch Yoke mechanism (Melvin and Thomas, 1993)

2.4.5 Cam-follower Mechanism

Wong and Goh (2020), Wong (2019) and Boutammachte and Knorr (2012) have proposed cam-follower mechanism to generate non-sinusoidal motions with dwell features to improve the Stirling engine performance. A form-closed/groove cam with oscillating follower is proposed by Wong and Goh (2020) to generate double dwell piston motions to study the improvement in the gamma Stirling engine performance. Table 2.11 shows the experimental data of the Stirling engine with groove cam - oscillating follower piston drives. The Stirling engine test rig, PV diagram and test cam profile are shown in Figure 2.20, Figure 2.21 and Figure 2.22. By using a non-sinusoidal RDFD cam with 135 ° displacer dwell (135_45Ovrlp cam), Wong and Goh (2020) found out that there is a 35% improvement in thermal efficiency and increase in power output when compared to engine running under sinusoidal motion.

Using cam-follower mechanism to generate non-sinusoidal motion in Stirling engine shows good improvement on the indicated power and the thermal efficiency of the engine; good flexibility in piston motion design; and the construction processes of a cam mechanism are simple. Large mass at the cam is one of the factors that affect the output power of the engine. However, cam with large mass is possible to provide enough momentum or inertia to the system for smooth operation, and without the need to add additional flywheel.

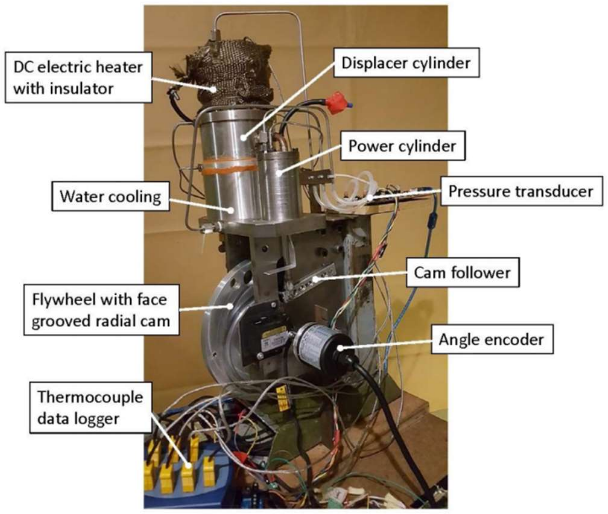


Figure 2.20: Stirling engine test rig with non-sinusoidal piston motion (Wong and Goh, 2020)

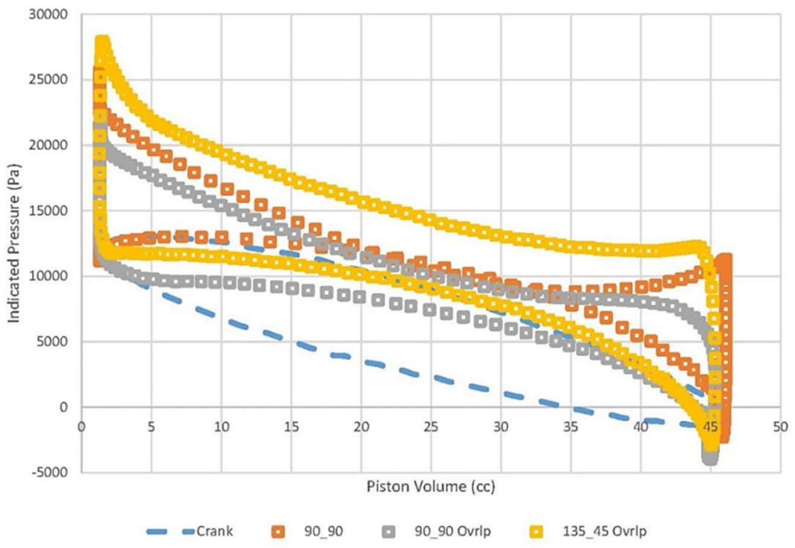
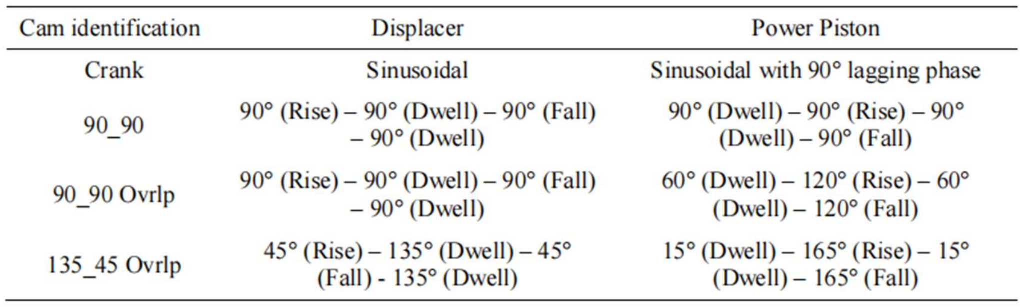


Figure 2.21: Stirling engine PV diagram with non-sinusoidal piston motion (Wong and Goh, 2020)

Table 2.11: Experimental data of non-sinusoidal Stirling engine (Wong and Goh, 2020)

Description	Data
Heater temperature	400°C
Cooler temperature	40°C
Engine frequency	1.8Hz
Engine Volume	267cc
Phase angle	90°+15°
Piston Stroke	35mm
Displacer Piston Bore	60mm
Power Piston Bore	40mm
Working gas	Air
Engine Output Power with conventional sinusoidal piston motion (W)	0.448
Engine Thermal Efficiency with conventional sinusoidal piston motion	12.81×10^{-4}
Engine Output Power with 135_45Ovrlp cam (W)	0.613
Engine Thermal Efficiency with 135_45Ovrlp cam	17.43×10^{-4}



Cam identification	Displacer	Power Piston
Crank	Sinusoidal	Sinusoidal with 90° lagging phase
90_90	90° (Rise) – 90° (Dwell) – 90° (Fall) – 90° (Dwell)	90° (Dwell) – 90° (Rise) – 90° (Dwell) – 90° (Fall)
90_90 Ovrlp	90° (Rise) – 90° (Dwell) – 90° (Fall) – 90° (Dwell)	60° (Dwell) – 120° (Rise) – 60° (Dwell) – 120° (Fall)
135_45 Ovrlp	45° (Rise) – 135° (Dwell) – 45° (Fall) - 135° (Dwell)	15° (Dwell) – 165° (Rise) – 15° (Dwell) – 165° (Fall)

Figure 2.22: Tested cam profile (Wong and Goh, 2020)

2.5 Summary

First or Second order thermodynamics models like Schmidt model and Urieli & Berchowitz model are able to simulate the pressure and temperature condition in the Stirling engine at any time during the steady operation. By collecting the pressure, temperature and volume information of the Stirling engine along an operation cycle, the indicated work/power per negine cycle and its thermal efficiency can be computed. The accuracy of the thermodynamics model are depends on the degree of idealisation of the engine operation. To simulate the real Stirling engine with great accuracy, detail losses analysis can be applied to the ideal isothermal and adiabtic simulation models to correct the simulated idealised engine conditions. However, in the early stage of the engine design where many parameters of the engine are not yet determined, ideal thermodynamics models such as Schmidt model and Urieli & Berchowitz model are good enough to study the behaviour of the engine.

The design of efficient regenerator can increase the Stirling engine performance. Computer aided 3D modeling software and computational fluid dynamics software can help the design process of regenerator and simulate the regenerator performance. By inputting the effectiveness of the regenerator into the Stirling engine thermodynamics model with losses analysis, the increase in the Stirling engine performance can be analysed.

However, less work has been done to develop a set of Stirling engine simulation model or a work flow of Stirling engine simulation that can study the improvement of Stirling engine performance under different non-sinusoidal piston motions, generated by different mechanisms. Cam-follower and modified Scotch Yoke are among mechanisms with little study on their effectiveness in improving the Stirling engine performance using non-sinusoidal piston motions. Unlike conventional sinusidal motion generation mechanism such as slider-crank mechanism, there is little or none Stirling engine simulation models that apply these two mechanisms in piston motion generation in their engine performance simulations. Development of computer simulation model of gamma configuration Stirling engine with non-sinusoidal motion with these mechanisms can help to improve the Stirling engine design process and help to analyse the performance of the newly designed engine.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

In this project, 3 sets of computer simulation programs of Stirling engine at three different conditions will be written in Matlab code. The first set of simulation program simulate the GPU-3 engine using thermodynamics model; compare simulation result with GPU-3 engine available data; and to validate the computer code/program written is capable of simulating the engine. The second set of computer simulation programs are to simulate the non-sinusoidal Stirling engine and checked with the experimental results by Wong and Goh (2020). The third set of computer simulation programs is to propose other method of generating non-sinusoidal piston motion in Stirling engine which is the modified Scotch Yoke mechanism. Future experimental work is needed to verify the designed modified Scotch Yoke mechanism for Stirling engine.

The computer simulation program can be divided into several sections. The first section of the program is the kinematics analysis of the piston driving mechanism. The second section of the program is the thermodynamics model. Two thermodynamics models will be studied in this project, which are Schmidt model / isothermal model; and Urieli and Berchowitz model / adiabatic model. Losses analysis is not included in the computer simulation program built in this project. Losses analysis requires accurate and precise information from a prototype Stirling engine (physical or 3D CAD model) to model the engine correctly. Losses analysis and other higher order models are more suitable for optimization of the engine parameters to increase the efficiency of the engine. Isothermal model and adiabatic model are more suitable in the early-stage design of Stirling engine when many information is still unknown (Furmanek and Kropiwnicki, 2022). Summary of the simulation programs is shown in Table 3.1.

Table 3.1: Summary of simulation programs

Simulation	Description	Purpose	Computer Program
1	Beta configuration GPU-3 engine simulation	Compare the simulation result of GPU-3 engine with other authors to verify the thermodynamic models built	Appendix A
2	Stirling engine with form-closed cam-oscillating follower (cycloidal) simulation	To compare simulation result of Stirling engine with form-closed cam-oscillating follower (cycloidal) with experimental result of real Stirling engine prototype	Appendix B
2.1	Stirling engine with sinusoidal piston motion simulation	To compare simulation result of Stirling engine with form-closed cam-oscillating follower (cycloidal) mechanism with conventional sinusoidal one	Appendix C
2.2	Stirling engine with form-closed cam-oscillating follower (polynomial) simulation	To improve the operation of the cam designed using cycloidal method	Appendix D
3	Stirling engine with modified Scotch Yoke (MSY) simulation	To simulate the performance of a new conceptual Stirling engine with modified Scotch Yoke (MSY) mechanism	Appendix E
3.1	Stirling engine with slider-crank simulation to compare with MSY	To compare simulation result of Stirling engine with MSY mechanism with conventional sinusoidal one	Appendix F

3.2 Kinematics Analysis of Piston Driving Mechanism

Four types of piston motion will be studied in this project: Sinusoidal piston motion; near Sinusoidal piston motion by slider-crank mechanism; double dwell piston motion by groove cam with oscillating follower; and double dwell piston motion by modified Scotch Yoke mechanism. Volume of engine spaces such as compression and expansion spaces can be calculated using piston position and velocity information.

$$V_{sw\_displacer\_hot} = \pi {r_{displacer}}^2 h_{stroke(displacer)} \tag{3.1}$$

$$V_{sw\_displacer\_cold} = V_{sw\_displacer\_hot} - \pi {r_{displacer\_drive\_rod}}^2 h_{stroke(displacer)} \tag{3.2}$$

$$\mathrm{V_{cold\_displacer}} = V_{sw\_displacer\_cold} - \pi\left({r_{displacer}}^2 - {r_{displacer\_drive\_rod}}^2\right) x_{displacer} \tag{3.3}$$

$$V_{exp} = V_{displacer\_clearance\_hot} + \pi {r_{displacer}}^2 x_{displacer} \tag{3.4}$$

$$V_{hot} = V_h + V_{exp} \tag{3.5}$$

$$\frac{dV_{exp}}{d\theta} = \frac{dV_{hot}}{d\theta} = \pi {r_{displacer}}^2 v_{displacer} \tag{3.6}$$

$$V_{comp} = V_{displacer\_clearance\_cold} + V_{power\_clearance} + \pi {r_{power}}^2 x_{power} + \mathrm{V_{cold\_displacer}} \tag{3.7}$$

$$V_{hot} = V_h + V_{exp} \tag{3.8}$$

$$\frac{dV_{exp}}{d\theta} = \frac{dV_{hot}}{d\theta} = \pi {r_{displacer}}^2 v_{displacer} \tag{3.9}$$

$$V_{comp} = V_{displacer\_clearance\_cold} + V_{power\_clearance} + \pi {r_{power}}^2 x_{power} + \mathrm{V_{cold\_displacer}} \tag{3.10}$$

$$V_{cold} = V_k + V_{comp} \tag{3.11}$$

$$\frac{dV_{comp}}{d\theta} = \frac{dV_{cold}}{d\theta} = \pi {r_{power}}^2 v_{power} - \pi\left({r_{displacer}}^2 - {r_{displacer\_drive\_rod}}^2\right) v_{displacer} \tag{3.12}$$

where

V_{sw}	=	swept volume
r	=	piston bore radius

$r_{displacer\_drive\_rod}$	=	piston drive rod radius
h_{stroke}	=	piston stroke
x_{power}	=	power piston position; 0 when near the end of cylinder
v_{power}	=	power piston velocity
$x_{displacer}$	=	displacer piston position; 0 when near the end of cyclinder at hot side
$v_{displacer}$	=	displacer piston velocity

3.2.1 Sinusoidal Piston Motion

Piston position and velocity of sinusoidal motion mechanism can be described by equations below:

$$x_{power(sinusoidal)} = \frac{h_{stroke(power)}}{2}\cos(\theta) \quad (3.13)$$

$$v_{power(sinusoidal)} = -\omega\frac{h_{stroke(power)}}{2}\sin(\theta) \quad (3.14)$$

$$x_{displacer(sinusoidal)} = \frac{h_{stroke(displacer)}}{2}\cos(\theta + 90°) \quad (3.15)$$

$$v_{displacer(sinusoidal)} = -\omega\frac{h_{stroke(displacer)}}{2}\sin(\theta + 90°) \quad (3.16)$$

where

ω	=	engine rotational speed in rad/s
θ	=	engine shaft angle

3.2.2 Slider-crank Mechanism

Figure 3.1 shows the schematic diagram of a slider-crank mechanism. Piston position, velocity and acceleration of slider-crank mechanism can be described by equations below (Norton, 2007):

$$\mathrm{r}_{\mathrm{crank(power)}} = \frac{h_{stroke(power)}}{2} \quad (3.17)$$

$$\mathrm{r}_{\mathrm{crank}(displacer)} = \frac{h_{stroke(displacer)}}{2} \quad (3.18)$$

$$x_{power(crank)} = \mathrm{r}_{\mathrm{crank(power)}} \cos(\theta) + L_{conrod}\left[1 - \left(\frac{\mathrm{r}_{\mathrm{crank(power)}}}{L_{conrod}} \sin(\theta)\right)^2\right]^{1/2} \tag{3.19}$$

$$v_{power(crank)} = -\omega \mathrm{r}_{\mathrm{crank(power)}} \left\{ \sin(\theta) + \frac{\mathrm{r}_{\mathrm{crank(power)}}}{2L_{conrod}} \frac{\sin(2\theta)}{\left[1 - \left(\frac{\mathrm{r}_{\mathrm{crank(power)}}}{L_{conrod}} \sin(\theta)\right)^2\right]^{1/2}} \right\} \tag{3.20}$$

$$a_{power(crank)} = -\omega^2 \mathrm{r}_{\mathrm{crank(power)}} \left\{ \cos(\theta) - \left[\frac{\mathrm{r}_{\mathrm{crank(power)}}\left({L_{conrod}}^2[1 - 2(\cos\theta)^2] - {\mathrm{r}_{\mathrm{crank(power)}}}^2(\sin\theta)^4\right)}{\left\{{L_{conrod}}^2 - \left(\mathrm{r}_{\mathrm{crank(power)}} \sin\theta\right)^2\right\}^{3/2}} \right] \right\} \tag{3.21}$$

$$x_{displacer(crank)} = \mathrm{r}_{\mathrm{crank}(displacer)} \cos(\theta + 90°) + L_{conrod}\left[1 - \left(\frac{\mathrm{r}_{\mathrm{crank}(displacer)}}{L_{conrod}} \sin(\theta + 90°)\right)^2\right]^{1/2} \tag{3.22}$$

$$v_{displacer(crank)} = -\omega \mathrm{r}_{\mathrm{crank}(displacer)} \left\{ \sin(\theta + 90°) + \frac{\mathrm{r}_{\mathrm{crank}(displacer)}}{2L_{conrod}} \frac{\sin(2(\theta + 90°))}{\left[1 - \left(\frac{\mathrm{r}_{\mathrm{crank}(displace\ )}}{L_{conrod}} \sin(\theta + 90°)\right)^2\right]^{1/2}} \right\} \tag{3.23}$$

$$a_{displacer(crank)} = -\omega^2 \mathrm{r}_{\mathrm{crank}(displacer)} \left\{ \cos(\theta + 90°) - \left[\frac{\mathrm{r}_{\mathrm{crank}(displacer)}\left(L_{conrod}{}^2[1 - 2(\cos(\theta+90°))^2] - \mathrm{r}_{\mathrm{crank}(displacer)}{}^2(\sin(\theta+90°))^4\right)}{\left\{L_{conrod}{}^2 - \left(\mathrm{r}_{\mathrm{crank}(displacer)}\sin(\theta+90°)\right)^2\right\}^{3/2}} \right] \right. \tag{3.24}$$

where

$\mathrm{r}_{\mathrm{crank}}$	=	length of crank
L_{conrod}	=	length of connection rod
a	=	acceleration

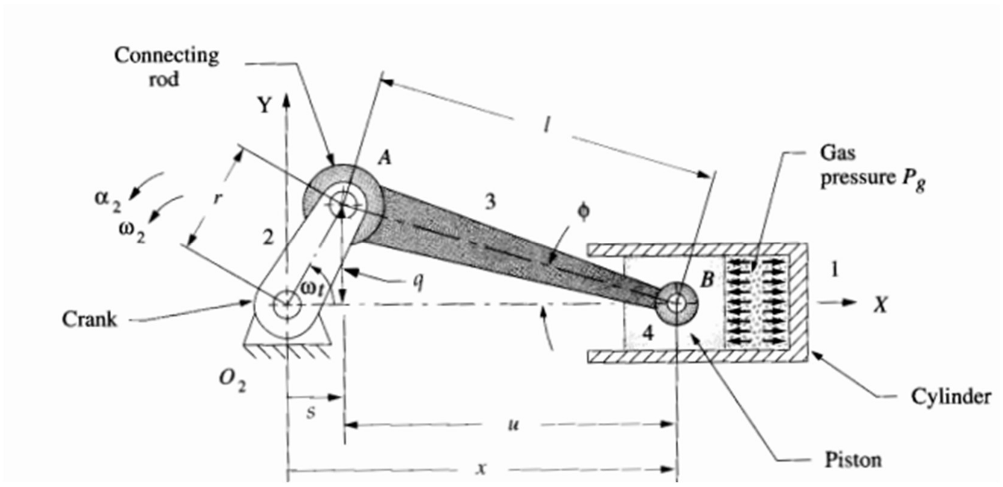


Figure 3.1: Schematic diagram of slider-crank mechanism (Norton, 2007)

If the engine pressure or gas pressure is obtained from the thermodynamics model, gas torque exerted on the engine can be approximated by the below equations.

$$\mathrm{T}_{\mathrm{gas(power)}} = \mathrm{T}_{\mathrm{g21}} \cong \mathrm{PA}_{\mathrm{piston(power)}}\mathrm{r}_{\mathrm{crank(power)}} \sin\theta \left(1 + \frac{\mathrm{r}_{\mathrm{crank(power)}}}{L_{conrod}}\cos(\theta)\right) \tag{3.25}$$

$$\mathrm{T}_{\mathrm{gas(dispalcer)}} = \mathrm{T}_{\mathrm{g21}} \cong \mathrm{PA}_{\mathrm{piston(dispalcer)}}\mathrm{r}_{\mathrm{crank(dispalcer)}} \sin(\theta + 90°) \left(1 + \frac{\mathrm{r}_{\mathrm{crank(dispalcer)}}}{L_{conrod}}\cos(\theta + 90°)\right) \tag{3.26}$$

$$\mathrm{T}_{\mathrm{gas(total)}} = \mathrm{T}_{\mathrm{gas(power)}} + \mathrm{T}_{\mathrm{gas(dispalcer)}} \tag{3.27}$$

where

T_{gas} = gas torque

Total torque experienced by the engine shaft is adding inertia torque and gas torque. Only gas torque which is caused by the gas pressure is calculated in this project, since inertia torque which is caused by the motion of mass of the mechanism itself is not part of the scope of this project. Inertia torque information is only needed when the design of the mechanism enters a later stage where stress analysis, balance analysis and other analysis are needed.

3.2.3 Form-closed Cam with Oscillating Follower Mechanism

The steps to design a form-closed cam with oscillating follower mechanism for Stirling engine non-sinusoidal piston motion are proposed by Wong (2019). The design starts with the synthesis of the oscillating follower that is connected to a slider-crank mechanism. Figure 3.2 shows the oscillating follower schematic diagram.

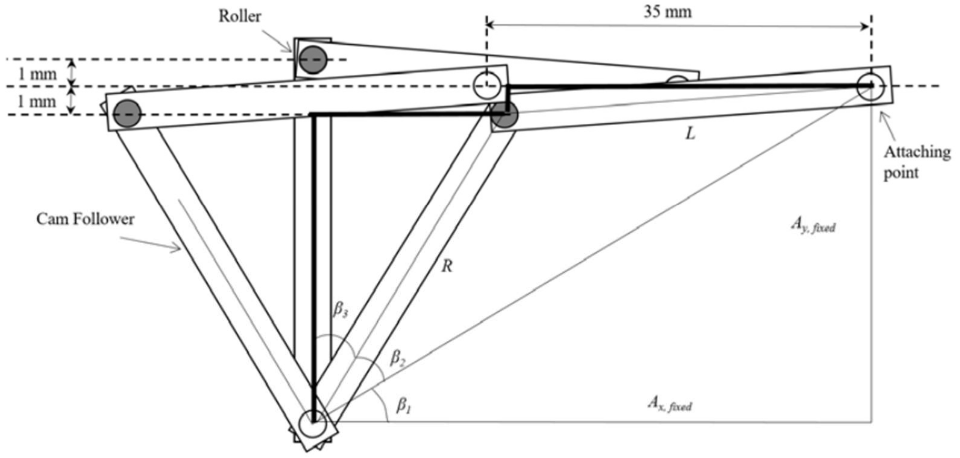


Figure 3.2: Schematic diagram of slider-crank mechanism for 35mm stroke (Wong, 2019)

From Figure 3.1, piston block will be connected to the attaching point of the oscillating follower. The form-closed cam / groove cam center can be placed at the positive direction of y-axis from oscillating follower center with a distance c_{cam}. The synthesis of this oscillating follower starts with setting the maximum vertical offset of joint 2 roller. In this project, it will be ±1mm. The second information needed is the piston stroke. For case in Figure 3.1, the stroke will be 35mm. This value can be changed according to the engine parameters. The geometric information of the oscillating follower can be calculated using the below equations:

$$L_{conrod(oscillating)} = \left[h_{stroke}{}^2 + \left(\frac{y_{max(joint2)}}{2}\right)^2\right]^{1/2} \tag{3.28}$$

$$L_{follower(oscillating)} = r_{crank(oscillating)} = \frac{y_{max(joint2)}}{1 - \cos\left(\frac{\beta_{stroke(follower)}}{2}\right)} \tag{3.29}$$

where

$L_{conrod(oscillating)}$	=	conrod length
$L_{follower(oscillating)}$	=	follower length
$y_{max(joint2)}$	=	maximum vertical offset of joint 2 roller
$\beta_{stroke(follower)}$	=	maximum swing angle of the follower

Based on Figure 3.1, a quadratic equation can be formed by applying the Pythagoras theorem on the triangles. By using the quadratic formula and trigonometry rule, $\beta_{stroke(follower)}$ can be solved.

$$\beta_{stroke(follower)\_part1} = \left[y_{max(joint2)}\right]^2 + \left(\frac{h_{stroke}}{2}\right)^2 \tag{3.30}$$

$$\beta_{stroke(follower)\_part2} = -2\left(\frac{h_{stroke}}{2}\right)^2 \tag{3.31}$$

$$\beta_{stroke(follower)\_part3} = \left(\frac{h_{stroke}}{2}\right)^2 - \left[y_{max(joint2)}\right]^2 \tag{3.32}$$

$$\begin{aligned}\beta_{stroke(follower)\_part4} \\ = \Big[\left(\beta_{stroke(follower)\_part2}\right)^2 \\ - 4\beta_{stroke(follower)\_part1}\left(\beta_{stroke(follower)\_part3}\right)\Big]^{1/2}\end{aligned} \tag{3.33}$$

$$\beta_{stroke(follower)\_part5} = \frac{-\beta_{stroke(follower)\_part2} - \beta_{stroke(follower)\_part4}}{2\beta_{stroke(follower)\_part1}} \tag{3.34}$$

$$\beta_{stroke(follower)} = 2\left[\cos^{-1}\left(\beta_{stroke(follower)\_part5}\right)\right] \tag{3.35}$$

After the design of the oscillating follower and calculated the $\beta_{stroke(follower)}$, SVAJ (displacement, velocity, acceleration and jerk) analysis of the form-closed cam can be done. SVAJ analysis for cam is important to know whether the cam fulfills the fundamental law of cam design (except running at very low speed). Fundamental law of cam design stated that the velocity and acceleration curve of the follower needs to be continuous for across entire cycle of cam angle (Norton, 2007).

3.2.3.1 SVAJ Analysis of Cam using Cycloidal Function

SVAJ analysis of cam using cycloidal function is used by Wong (2019) to design the cam groove profile for double dwell piston motion. Cycloidal function on the rising curve of the SVAJ diagrams is shown below (Norton, 2007).

$$\mathrm{s}_{\mathrm{follower(rising)}} = \beta_{stroke(follower)}\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}} - \frac{1}{2\pi}\sin\left(2\pi\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right)\right] \tag{3.36}$$

$$\mathrm{v}_{\mathrm{follower(rising)}} = \frac{\beta_{stroke(follower)}}{\beta_{\mathrm{period(rising)}}}\left[1 - \cos\left(2\pi\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right)\right] \tag{3.37}$$

$$\mathrm{a}_{\mathrm{follower(rising)}} = 2\pi\frac{\beta_{stroke(follower)}}{\beta_{\mathrm{period(rising)}}{}^{2}}\sin\left(2\pi\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right) \tag{3.38}$$

$$\mathrm{j}_{\mathrm{follower(rising)}} = 4\pi^2\frac{\beta_{stroke(follower)}}{\beta_{\mathrm{period(rising)}}{}^{3}}\cos\left(2\pi\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right) \tag{3.39}$$

where

$\mathrm{s}_{\mathrm{follower(rising)}}$	=	follower angle in rad
β_{period}	=	rising period in rad
θ_{cam}	=	cam angle (value from 0 to β_{period})

The falling curve of SVAJ diagrams are "mirror" version of rising curve.

$$\mathrm{s}_{\mathrm{follower(falling)}} = \beta_{stroke(follower)} - \mathrm{s}_{\mathrm{follower(rising)}} \tag{3.40}$$

$$\mathrm{v}_{\mathrm{follower(falling)}} = -\mathrm{v}_{\mathrm{follower(rising)}} \tag{3.41}$$

$$\mathrm{a}_{\mathrm{follower(falling)}} = -\mathrm{a}_{\mathrm{follower(rising)}} \tag{3.42}$$

$$\mathrm{j}_{\text{follower(falling)}} = -\mathrm{j}_{\text{follower(rising)}} \tag{3.43}$$

The S diagram of the follower at power piston can be plotted by joining the piecewise functions.

$$\mathrm{s}_{\text{follower(power)}} = \left\{\mathrm{s}_{\text{follower(falling)}}, 0, \mathrm{s}_{\text{follower(rising)}}, \beta_{stroke(follower)}\right\} \tag{3.44}$$

where

$\mathrm{s}_{\text{follower(power)}}$ = follower position in rad

Similar procedure can be done to plot other SVAJ diagrams. SVAJ diagrams of the follower at displacer piston is just shifting the power piston follower diagrams by 90 degrees to the left.

3.2.3.2 SVAJ Analysis of Cam using Polynomial Function

Besides cycloidal function, other functions such as modified trapezoidal function and spline functions can be used in the design of double dwell motion. Norton (2007) suggested the use of 4-5-6-7 polynomial function in the double dwell cam SVAJ analysis as it ensures the SVAJ functions to be continuous for the whole cycle. The highest order of the polynomial function is chosen as 7 to ensure the J diagram to be continuous and smooth after several differentiations of S diagram. Using 4-5-6-7 polynomial function on rising curve of the SVAJ diagram starts with a general polynomial function.

$$\begin{aligned}\mathrm{s}_{\text{follower(rising)}} = {} & \mathrm{C}_0 + \mathrm{C}_1 \frac{\theta_{\text{cam}}}{\beta_{\text{period(rising)}}} + \mathrm{C}_2 \left[\frac{\theta_{\text{cam}}}{\beta_{\text{period(rising)}}}\right]^2 \\ & + \mathrm{C}_3 \left[\frac{\theta_{\text{cam}}}{\beta_{\text{period(rising)}}}\right]^3 + \mathrm{C}_4 \left[\frac{\theta_{\text{cam}}}{\beta_{\text{period(rising)}}}\right]^4 \\ & + \mathrm{C}_5 \left[\frac{\theta_{\text{cam}}}{\beta_{\text{period(rising)}}}\right]^5 + \mathrm{C}_6 \left[\frac{\theta_{\text{cam}}}{\beta_{\text{period(rising)}}}\right]^6 \\ & + \mathrm{C}_7 \left[\frac{\theta_{\text{cam}}}{\beta_{\text{period(rising)}}}\right]^7\end{aligned} \tag{3.45}$$

$$\mathrm{v}_{\mathrm{follower(rising)}} = \frac{1}{\beta_{\mathrm{period(rising)}}}\left\{C_1 + 2C_2\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}} + 3C_3\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right]^2 + 4C_4\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right]^3 + 5C_5\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right]^4 + 6C_6\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right]^5 + 7C_7\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right]^6\right\} \tag{3.46}$$

$$\mathrm{a}_{\mathrm{follower(rising)}} = \frac{1}{\beta_{\mathrm{period(rising)}}{}^2}\left\{2C_2 + 6C_3\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right] + 12C_4\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right]^2 + 20C_5\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right]^3 + 30C_6\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right]^4 + 42C_7\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right]^5\right\} \tag{3.47}$$

$$\mathrm{j}_{\mathrm{follower(rising)}} = \frac{1}{\beta_{\mathrm{period(rising)}}{}^3}\left\{6C_3 + 24C_4\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right] + 60C_5\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right]^2 + 120C_6\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right]^3 + 210C_7\left[\frac{\theta_{\mathrm{cam}}}{\beta_{\mathrm{period(rising)}}}\right]^4\right\} \tag{3.48}$$

The polynomial function can be simplified by applying boundary conditions of the curve.

$$\text{When } \theta_{\mathrm{cam}} = 0,\ \mathrm{s}_{\mathrm{follower(rising)}} = 0;\ \mathrm{v}_{\mathrm{follower(rising)}} = 0;\ \mathrm{a}_{\mathrm{follower(rising)}} = 0;\ \mathrm{j}_{\mathrm{follower(rising)}} = 0 \tag{3.49}$$

$$\text{When } \beta_{\mathrm{period(rising)}} = 0,\ \mathrm{s}_{\mathrm{follower(rising)}} = \beta_{stroke(follower)};\ \mathrm{v}_{\mathrm{follower(rising)}} = 0;\ \mathrm{a}_{\mathrm{follower(rising)}} = 0;\ \mathrm{j}_{\mathrm{follower(rising)}} = 0 \tag{3.50}$$

For example, S diagram can be simplified to:

$$s_{\text{follower(rising)}} = \beta_{stroke(follower)} \left[35 \left[\frac{\theta_{\text{cam}}}{\beta_{\text{period(rising)}}} \right]^4 - 84 \left[\frac{\theta_{\text{cam}}}{\beta_{\text{period(rising)}}} \right]^5 + 70 \left[\frac{\theta_{\text{cam}}}{\beta_{\text{period(rising)}}} \right]^6 - 20 \left[\frac{\theta_{\text{cam}}}{\beta_{\text{period(rising)}}} \right]^7 \right] \tag{3.51}$$

Similar steps from the cycloidal function section can be done here to plot SVAJ diagrams of the cam-follower.

3.2.3.3 Pressure Angle and Profile of Cam

Cam pressure angle is the angle between the “cam center - follower roller (joint2)” line and the tangent line of the cam curve surface. This is the angle where the force from the follower applies on the cam. Figure 3.3 shows a general schematic diagram of the cam-oscillating follower that shows the important angles and lengths.

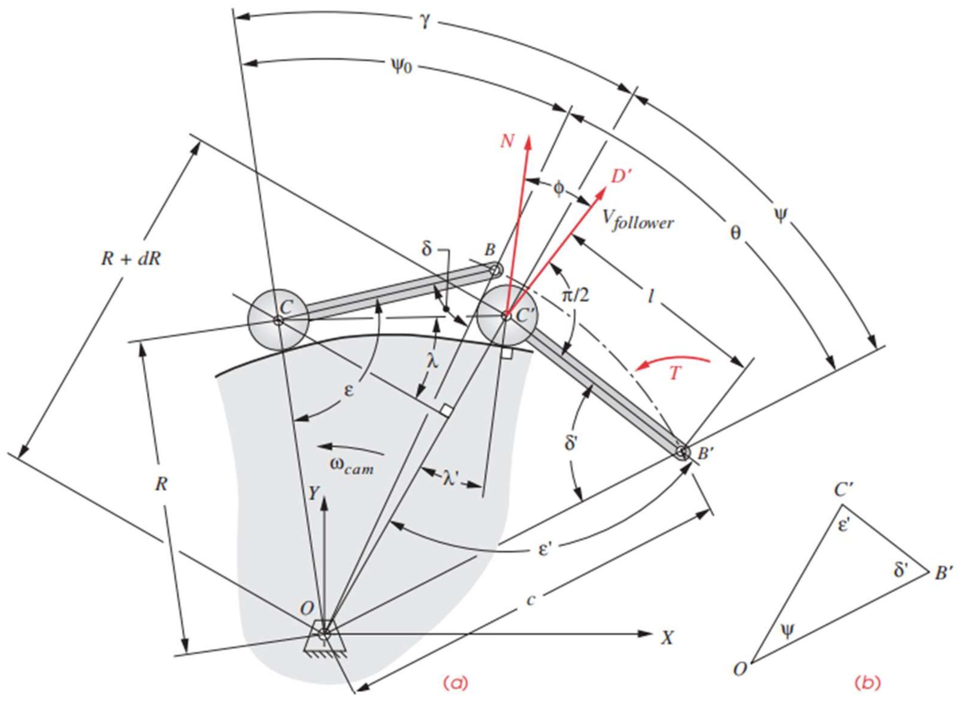


Figure 3.3: Schematic diagram of the cam-oscillating follower (Norton, 2002)

Calculation of pressure angle for the cam at power piston is shown below:

$$\delta_0 = -\beta_{stroke(follower)}/2 \tag{3.52}$$

$$\delta_{\text{follower}} = \delta_0 + \text{s}_{\text{follower}} \tag{3.53}$$

$$\begin{aligned}\text{R}_{\text{cam\_curve}} = \Big[&{L_{follower(oscillating)}}^2 + {c_{cam}}^2 \\ &- 2L_{follower(oscillating)}(c_{cam})\cos(\delta_{\text{follower}})\Big]^{1/2}\end{aligned} \tag{3.54}$$

$$\varepsilon_{cam} = \sin^{-1}\left(\frac{c_{cam}}{\text{R}_{\text{cam\_curve}}}\sin(\delta_{\text{follower}})\right) \tag{3.55}$$

$$\varphi_{\text{cam}} = \cos^{-1}\left[\frac{{c_{cam}}^2 + {\text{R}_{\text{cam\_curve}}}^2 - {L_{follower(oscillating)}}^2}{2\text{R}_{\text{cam\_curve}}c_{cam}}\right] \tag{3.56}$$

$$\gamma_{\text{cam}} = \varphi_{\text{cam(initial)}} - \varphi_{\text{cam}} + \theta \tag{3.57}$$

$$\emptyset_{\text{low\_dwell}} = \frac{\pi}{2} - \varepsilon_{cam\ (low\_dwell)} \tag{3.58}$$

$$\begin{aligned}&\emptyset_{\text{rising(part1)}} \\ &= \frac{{\text{R}_{\text{cam\_curve(rising)}}}^2}{L_{follower(oscillating)}c_{cam}\sin(\delta_{\text{follower(rising)}})\text{v}_{\text{follower(rising)}}}\end{aligned} \tag{3.59}$$

$$\emptyset_{\text{rising(part2)}} = \frac{{c_{cam}}^2 - {\text{R}_{\text{cam\_curve(rising)}}}^2 - {L_{follower(oscillating)}}^2}{2\text{R}_{\text{cam\_curve(rising)}}c_{cam}\sin\big(\varphi_{\text{cam(rising)}}\big)} \tag{3.60}$$

$$\emptyset_{\text{rising}} = \frac{\pi}{2} - \varepsilon_{cam\ (rising)} + \tan^{-1}\frac{1}{\emptyset_{\text{rising(part1)}} - \emptyset_{\text{rising(part1)}}} \tag{3.61}$$

$$\emptyset_{\text{high\_dwell}} = -\frac{\pi}{2} + \varepsilon_{cam\ (low\_dwell)} \tag{3.62}$$

$$\begin{aligned}&\emptyset_{\text{falling(part1)}} \\ &= \frac{{\text{R}_{\text{cam\_curve(falling)}}}^2}{L_{follower(oscillating)}c_{cam}\sin(\delta_{\text{follower(falling)}})\text{v}_{\text{follower(falling)}}}\end{aligned} \tag{3.63}$$

$$\emptyset_{\text{falling(part2)}} = \frac{{c_{cam}}^2 - {\text{R}_{\text{cam\_curve(falling)}}}^2 - {L_{follower(oscillating)}}^2}{2\text{R}_{\text{cam\_curve(falling)}}c_{cam}\sin\big(\varphi_{\text{cam(falling)}}\big)} \tag{3.64}$$

$$\emptyset_{\text{falling}} = -\frac{\pi}{2} + \varepsilon_{cam\ (\text{falling})} + \tan^{-1}\frac{1}{\emptyset_{\text{falling(part1)}} + \emptyset_{\text{falling(part1)}}} \tag{3.65}$$

$$\emptyset_{\text{cam(power)}} = \left\{\emptyset_{\text{falling}}, \emptyset_{\text{low\_dwell}}, \emptyset_{\text{rising}}, \emptyset_{\text{high\_dwell}}\right\} \tag{3.66}$$

where

$\emptyset_{\text{cam(power)}}$ = pressure angle

Pressure angle for the cam at the displacer piston can be calculated with similar steps. Oscillating cam groove profile can be determined using $\mathrm{R_{cam\_curve}}$ information.

$$\mathrm{Profile_x} = \mathrm{R_{cam\_curve}} \cos(\theta + \delta_0) \tag{3.67}$$

$$\mathrm{Profile_y} = \mathrm{R_{cam\_curve}} \sin(\theta + \delta_0) \tag{3.68}$$

where

$\mathrm{Profile_x}$ = curve profile x coordinate

$\mathrm{Profile_y}$ = curve profile y coordinate

3.2.3.4 Kinematics Analysis of Piston

The piston displacement, velocity and acceleration can be calculated using the below equations which is similar to the analysis done on slider-crank mechanism (Norton, 2007).

$$\begin{aligned} x_{piston} = {} & L_{follower(oscillating)} \cos\theta_{2(oscillating)} \\ & - L_{conrod(oscillating)} \cos\theta_{3(oscillating)} \end{aligned} \tag{3.69}$$

$$\theta_{2(oscillating)} = \frac{\pi}{2} + \beta_{stroke(follower)}/2 - \mathrm{s_{follower}} \tag{3.70}$$

$$\begin{aligned} & \theta_{3(oscillating)} \\ & = \sin^{-1}\left[\frac{L_{follower(oscillating)}\cos\theta_{2(oscillating)} - \left(L_{follower(oscillating)} - \frac{y_{max(joint2)}}{2}\right)}{L_{conrod(oscillating)}}\right] \end{aligned} \tag{3.71}$$

$$\begin{aligned} v_{piston} = {} & -L_{follower(oscillating)}(-\mathrm{v_{follower}}) \sin\theta_{2(oscillating)} \\ & + L_{conrod(oscillating)}\omega_{3(oscillating)} \sin\theta_{3(oscillating)} \end{aligned} \tag{3.72}$$

$$\omega_{3(oscillating)} = \frac{L_{follower(oscillating)}(-\mathrm{v_{follower}}) \cos\theta_{2(oscillating)}}{L_{conrod(oscillating)} \cos\theta_{3(oscillating)}} \tag{3.73}$$

$$\begin{aligned} a_{piston} = {} & -L_{follower(oscillating)}\mathrm{a_{follower}} \sin\theta_{2(oscillating)} \\ & - L_{follower(oscillating)}(-\mathrm{v_{follower}})^2 \sin\theta_{2(oscillating)} \\ & + L_{conrod(oscillating)}\alpha_{3(oscillating)} \sin\theta_{3(oscillating)} \\ & + L_{conrod(oscillating)}\left(\omega_{3(oscillating)}\right)^2 \sin\theta_{3(oscillating)} \end{aligned} \tag{3.74}$$

$$\alpha_{3(oscillating)} = \Big[L_{follower(oscillating)}\mathrm{a}_{\mathrm{follower}} \cos\theta_{2(oscillating)} - L_{follower(oscillating)}(-\mathrm{v}_{\mathrm{follower}})^2 \sin\theta_{2(oscillating)} + L_{conrod(oscillating)}\big(\omega_{3(oscillating)}\big)^2 \sin\theta_{3(oscillating)}\Big] / \big(L_{conrod(oscillating)} \cos\theta_{3(oscillating)}\big) \tag{3.75}$$

With piston displacement and velocity information, engine internal volume information can be calculated.

3.2.3.5 Gas Torque Calculation

Gas Torque exerted on the oscillating follower can be analysed similar to slider-crank dynamics analysis shown above. If total torque exerted on the engine shaft is needed, dynamic force matrix needed to be constructed to calculate the total torque (T_{21}). Figure 3.4 shows the dynamics analysis matrix for general slider-crank mechanism.

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ -R_{12_y} & R_{12_x} & -R_{32_y} & R_{32_x} & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & R_{23_y} & -R_{23_x} & -R_{43_y} & R_{43_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & \pm\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} F_{12_x} \\ F_{12_y} \\ F_{32_x} \\ F_{32_y} \\ F_{43_x} \\ F_{43_y} \\ F_{14_y} \\ T_{12} \end{bmatrix} = \begin{bmatrix} m_2 a_{G_{2_x}} \\ m_2 a_{G_{2_y}} \\ I_{G_2}\alpha_2 \\ m_3 a_{G_{3_x}} \\ m_3 a_{G_{3_y}} \\ I_{G_3}\alpha_3 \\ m_4 a_{G_{4_x}} - F_{P_x} \\ -F_{P_y} \end{bmatrix}$$

Figure 3.4: Dynamics force matrix of the slider-crank mechanism (Norton, 2007)

3.2.4 Modified Scotch Yoke Mechanism

The design procedure for modified Scotch Yoke mechanism in this project is very similar to cam-follower. As shown in Figure 3.4, the falling curve / rising curve are divided into 3 sections. The first section is the transition curve from dwell to rising/falling. The second section is the falling or rising cosine curve. The third section is the transition curve from rising/falling to dwell. The second section curve will appear to be straight line in the slot curve of Scotch Yoke. The first section and third section curve can be drafted using steps similar to cam-follower polynomial curve construction. The difference is the boundary conditions for the general polynomial function are different. As 4-5-6-7 polynomial function and the normal cosine curve will be continuous in SVAJ curves, the noise of the mechanism when running can be reduced. Since piecewise 4-5-6-7 polynomial function, cosine function and flat function are all differentiable for at least three times, all SVAJ plots are expected to be continuous and finite.

After designing the piston motion, slot curve of the modified Scotch Yoke can be generated using the following equations:

$$\mathrm{r_{rotation\_slot}} = \frac{\mathrm{h_{stroke}}}{2\sin\left(\beta_{\mathrm{period(moving)}}\pi\right)} \tag{3.76}$$

$$\delta_{\mathrm{slot\_angle}} = \tan^{-1}\frac{\mathrm{r_{rotation\_slot}}(\cos\theta_{\mathrm{i+1}} - \cos\theta_{\mathrm{i}}) - \left(x_{piston(i+1)} - x_{piston(i)}\right)}{x_{piston(i+1)} - x_{piston(i)}} \tag{3.77}$$

$$\mathrm{Curve_{slot\_x}} = \mathrm{r_{rotation\_slot}}\sin\theta \tag{3.78}$$

$$\mathrm{Curve_{slot\_y(i+1)}} = \mathrm{Curve_{slot\_y(i)}} + \left(\mathrm{Curve_{slot\_x(I+1)}} - \mathrm{Curve_{slot\_x(i)}}\right)\tan\delta_{\mathrm{slot\_angle}} \tag{3.79}$$

where

$\mathrm{Curve_{slot\_x}}$ = curve profile x coordinate

$\mathrm{Curve_{slot\_y}}$ = curve profile y coordinate

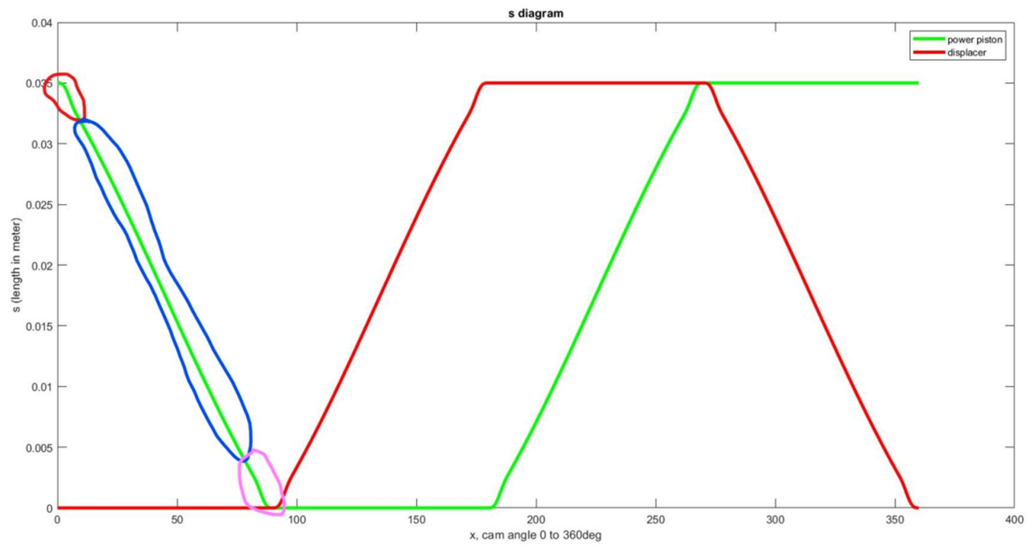


Figure 3.4: Power piston falling curve of modified Scotch Yoke Mechanism. The falling curve are divided into three sections which are circled in the figure.

After obtaining the pressure information from the thermodynamics model, the gas torque exerted by only y-direction force of the gas pressure can be calculated:

$$T_{gas(y)} = P\pi\frac{{D_{piston}}^{2}}{4}\mathrm{Curve}_{\mathrm{slot\_x}} \tag{3.80}$$

3.3 Building of Thermodynamics Models in Computer Program

Two types of thermodynamics models will be built using Matlab code in this project, which are Schmidt model and the Urieli and Berchowitz model. The compression space volume and expansion space volume obtained from the previous section (piston driving mechanism) will be inserted into the thermodynamics model together with the other information (including heater/cooler temperature information) to simulate the Stirling engine condition / performance.

3.3.1 Schmidt model

The Schmidt model is an isothermal model of Stirling engine that is closer to the idea of an ideal Stirling engine where the temperatures in the compression and expansion space are always constant. The modeling of Schmidt model is simpler, as it requires less design information than other types of models.

Information needed by the Schmidt model to start simulation including type of gas inside the engine; total mass of gas inside the engine, M_{gas}; volume information of different parts of the engine, $V_{regenerator}, V_{hot}, V_{cold}, dV_{ho}$, dV_{cold}; heat exchanger temperatures, T_h, T_k.

After the gas is known, gas constant, R_{spec} and the specific heat capacities, c_p & c_v at average temperature and pressure are to be determined from property tables and charts or using empirical equation to make approximation.

Using the equations from section 2.2.1, engine pressure and work output can be calculated. Numerical method Simpson's rule is applied in this project to solve the integration problem of the work equation, W. Simpson's rule equation is shown below (Abramowitz and Stegun, 1968):

$$S_n = \frac{h_{step}}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)] \tag{3.81}$$

where

h_{step} = increment magnitude of the engine angle in rad

After the work output is determined, thermal efficiency of the engine can be calculated:

$$\eta_{Schmi} = \frac{W_{net}}{Q_{hot}} = \frac{W_{net}}{W_{hot}} = \frac{\oint \mathrm{P}\left(\frac{\mathrm{dV_{hot}}}{d\theta} + \frac{\mathrm{dV_{cold}}}{d\theta}\right) d\theta}{\oint \mathrm{P}\left(\frac{\mathrm{dV_{hot}}}{d\theta}\right) d\theta} \tag{3.82}$$

3.3.2 Urieli and Berchowitz model

Urieli and Berchowitz model is more resembles the real Stirling engine as isothermal compression or expansion is difficult to achieve, especially if the engine is running at a high speed.

Figure 3.5 shows the flow chart of steps to simulate Stirling engine using Urieli and Berchowitz model. Engine dimensions and operation data are to be provided by the user. V and dV values are to be determined based on the piston driving mechanism information. Initial conditions of some temperatures are shown below:

$$T_{comp(initial)} = T_k \tag{3.83}$$

$$T_{exp(initial)} = T_h \tag{3.84}$$

$$T_{ck(initial)} = T_k \tag{3.85}$$

$$T_{he(initial)} = T_h \tag{3.86}$$

$$T_{kr} = T_k \tag{3.87}$$

$$T_{rh} = T_h \tag{3.88}$$

Using the equations in section 2.2.2 and running through the steps in Figure 3.5, engine pressure, compression space temperature, expansion space temperature, engine work and heat transfer at heat exchangers can be simulated.

Differential equations of T, Q and W are to be solved numerically using RK4 method. General RK4 method formulation is shown below (Abramowitz and Stegun, 1968):

$$k_{1y} = h_{step} y'(x_n, y_n, z_n, \dots) \tag{3.89}$$

$$k_{1z} = h_{step} z'(x_n, y_n, z_n, \dots) \tag{3.90}$$

$$k_{2y} = h_{step} y'(x_n + \frac{h_{step}}{2}, y_n + \frac{k_{1y}}{2}, z_n + \frac{k_{1z}}{2}, \dots) \tag{3.91}$$

$$k_{2z} = h_{step} z'(x_n + \frac{h_{step}}{2}, y_n + \frac{k_{1y}}{2}, z_n + \frac{k_{1z}}{2}, \dots) \tag{3.92}$$

$$k_{3y} = h_{step} y'(x_n + \frac{h_{step}}{2}, y_n + \frac{k_{2y}}{2}, z_n + \frac{k_{2z}}{2}, \dots) \tag{3.93}$$

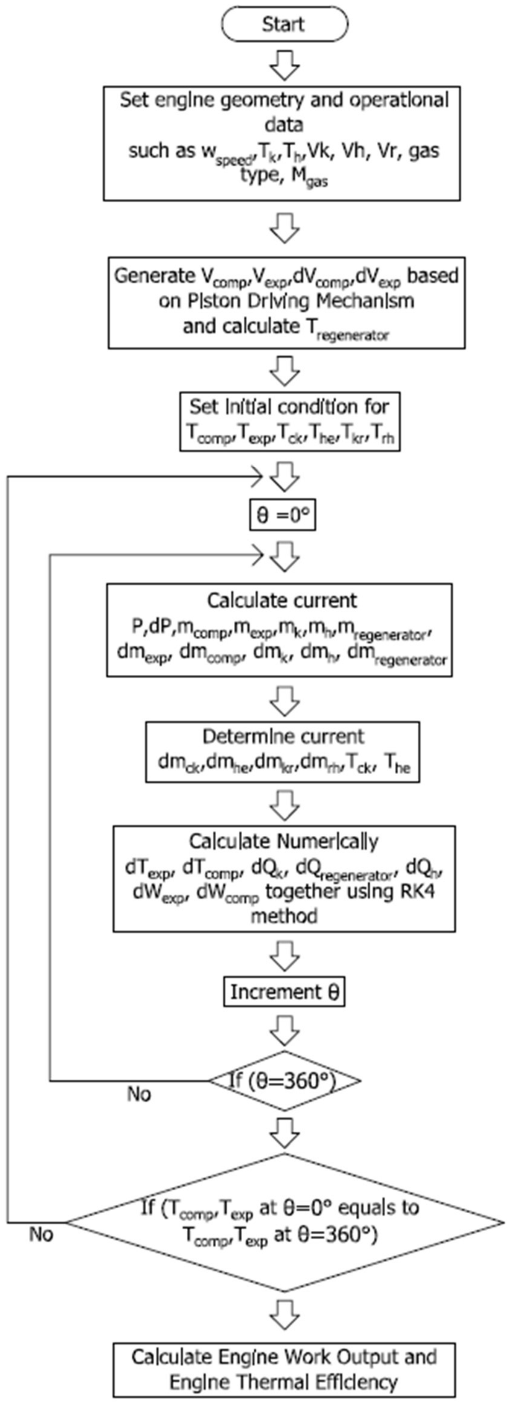
$$k_{3z} = h_{step} z'(x_n + \frac{h_{step}}{2}, y_n + \frac{k_{2y}}{2}, z_n + \frac{k_{2z}}{2}, \dots) \tag{3.94}$$

$$k_{4y} = h_{step} y'(x_n + h_{step}, y_n + k_{3y}, z_n + k_{3z}, \dots) \tag{3.95}$$

$$k_{4z} = h_{step} z'(x_n + h_{step}, y_n + k_{3y}, z_n + k_{3z}, \dots) \tag{3.96}$$

$$y_{n+1} = y_n + \frac{1}{6}(k_{1y} + 2k_{2y} + 2k_{3y} + k_{4y}) \tag{3.97}$$

$$z_{n+1} = z_n + \frac{1}{6}(k_{1z} + 2k_{2z} + 2k_{3z} + k_{4z}) \tag{3.98}$$



Start
Set engine geometry and operational data such as w_{speed}, T_k, T_h, Vk, Vh, Vr, gas type, M_{gas}
Generate V_{comp}, V_{exp}, dV_{comp}, dV_{exp} based on Piston Driving Mechanism and calculate $T_{regenerator}$
Set initial condition for T_{comp}, T_{exp}, T_{ck}, T_{he}, T_{kr}, T_{rh}
θ =0°
Calculate current P, dP, m_{comp}, m_{exp}, m_k, m_h, $m_{regenerator}$, dm_{exp}, dm_{comp}, dm_k, dm_h, $dm_{regenerator}$
Determine current dm_{ck}, dm_{he}, dm_{kr}, dm_{rh}, T_{ck}, T_{he}
Calculate Numerically dT_{exp}, dT_{comp}, dQ_k, $dQ_{regenerator}$, dQ_h, dW_{exp}, dW_{comp} together using RK4 method
Increment θ
If (θ=360°)
No
If (T_{comp}, T_{exp} at θ=0° equals to T_{comp}, T_{exp} at θ=360°)
No
Calculate Engine Work Output and Engine Thermal Efficiency

Figure 3.5: Flow chart of Urieli and Berchowitz model analysis steps

After obtaining the W and Q values, and the model enters a steady state, engine thermal efficiency can be determined.

$$\eta_{Urieli} = \frac{W_{comp} + W_{exp}}{Q_h} \tag{3.99}$$

3.4 Summary

In simulation program 1, Stirling engine with sinusoidal piston motion is simulated using Schmidt model and Urieli and Berchowitz model to compare with the GPU-3 real Striling engine performance. In simulation program 2, Stirling engine with double dwell piston motion driven by groove cam with oscillating follower is simulated using Schmidt model and Urieli and Berchowitz model to compare with the experimental results published by Wong and Goh (2020). In simulation program 3, Stirling engine with double dwell piston motion driven by modified Scotch yoke mechanism is simulated using Schmidt model and Urieli and Berchowitz model to study the engine performance.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

Three sets of Stirling engine simulation program written in Matlab code are used to simulate the sinusoidal motion Stirling engine, non-sinusoidal motion Stirling engine with cam-follower mechanism and non-sinusoidal motion Stirling engine with modified Scotch yoke mechanism. The written computer programs are shown in the appendices of this report, Appendix A until Appendix F. Two thermodynamics models were used in each program to study the engine performance. Simulation results of the Stirling engine will include the piston motion diagram, engine internal pressure diagram, engine internal volume diagram, PV diagram, engine indicated power output per cycle and engine thermal efficiency.

When comparing the simulated results with the experimental ones, indicated power error and thermal efficiency error can be calculated.

$$\begin{aligned}&Indicated\ Power\ Error\\&= \frac{Simulated\ Indicated\ Power - Experimental\ Indicated\ Power}{Experimental\ Indicated\ Power} x100\%\end{aligned} \tag{4.1}$$

$$\begin{aligned}&Thermal\ Efficiency\ Error\\&= Simulated\ Thermal\ Efficiency - Experimental\ Thermal\ Efficiency\end{aligned} \tag{4.2}$$

4.2 Simulation 1: Verification of Simulation Model

Schmidt model and Urieli and Berchowitz model shown in Appendix A computer program are used to simulate the GPU-3 beta Stirling engine according to the dimension and operation data in Table 2.9. The piston motion of the engine simulated is an exact sinusoidal. Figure 4.1 to Figure 4.4, and Table 4.1 show the simulation results from the computer program written.

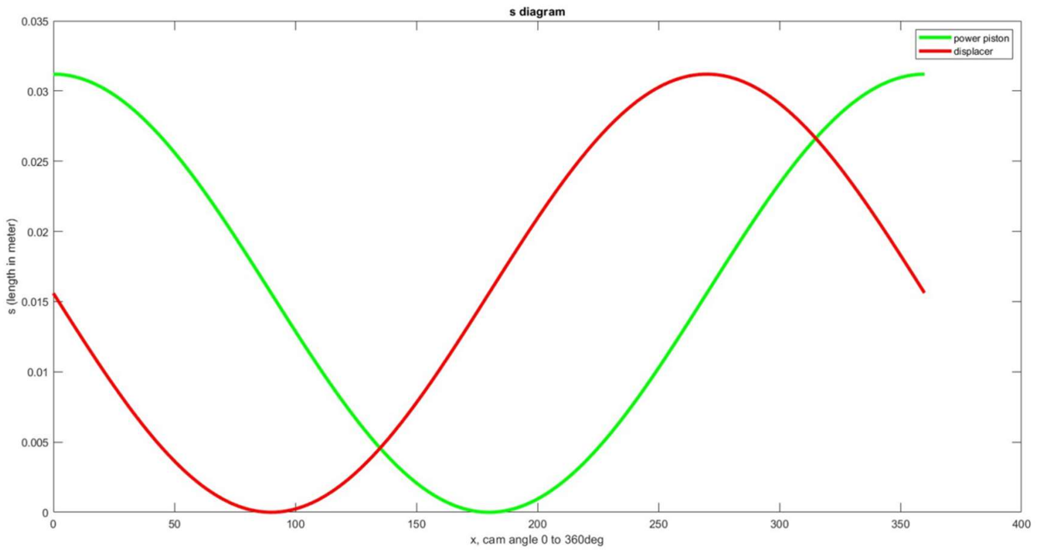


Figure 4.1: Piston motion diagram of simulation 1

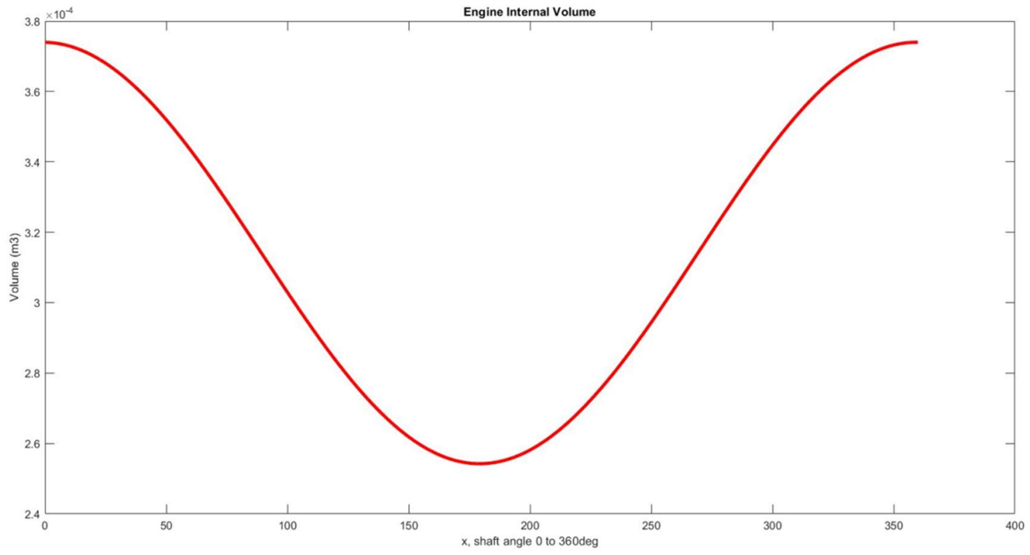


Figure 4.2: Engine internal volume of simulation 1

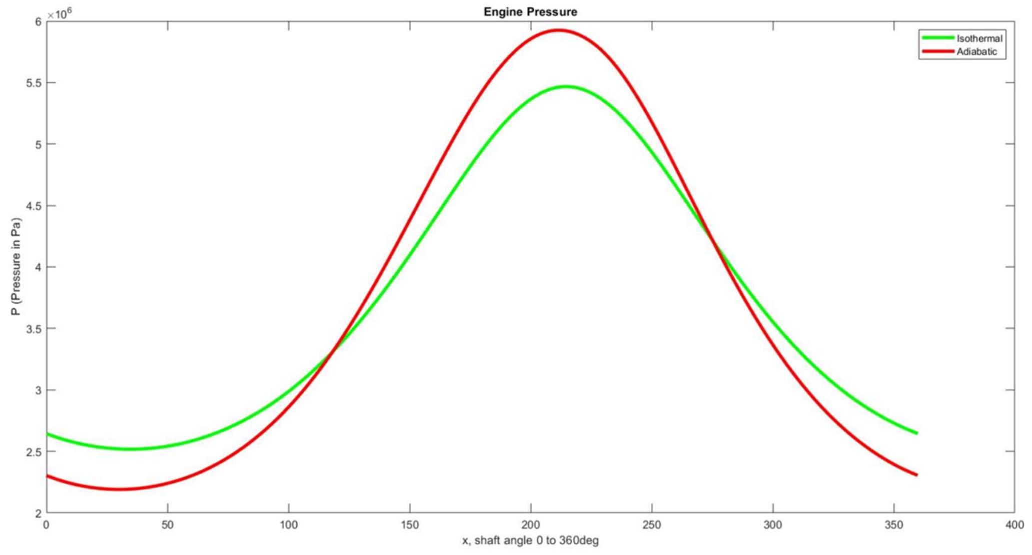


Figure 4.3: Engine internal pressure of simulation 1

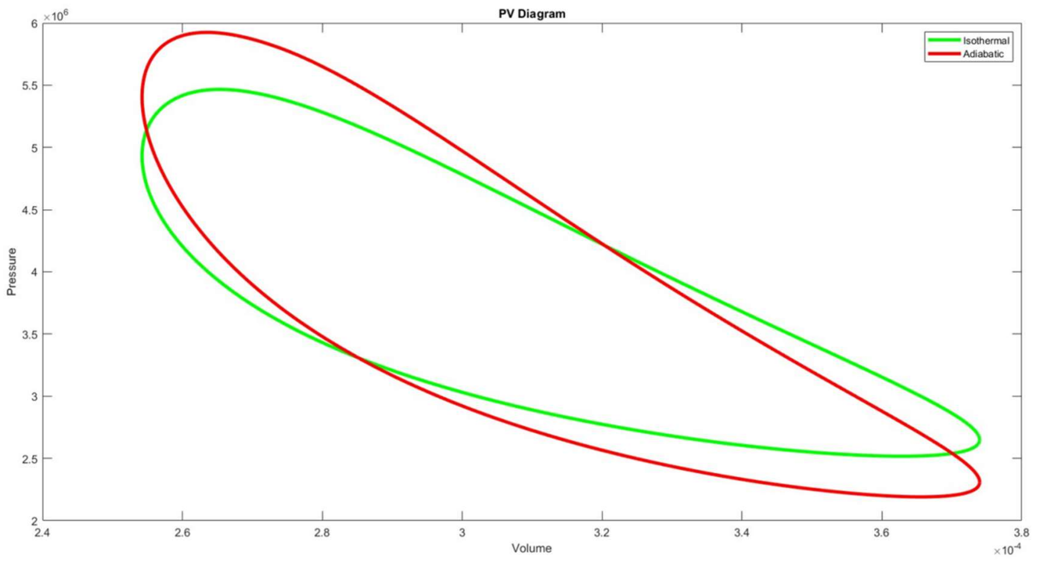


Figure 4.4: PV diagram of simulation 1

Table 4.1: Simulated engine performance of simulation 1

Description	Schmidt Model Simulation Data	Urieli and Berchowitz model Simulation Data
Engine Indicated Work (J) per revolution	154.8903	177.4512
Engine Indicated Power (W)	6458.9	7399.7
Engine Thermal Efficiency (%)	70.52	60.11
Indicated power error when compared to experimental result (%)	115.30	146.66
Thermal efficiency error when compared to experimental result (%)	49.22	38.81

Based on the simulation result in Table 4.1, the indicated output power and the engine thermal efficiency simulated by the written computer program in Appendix A (Schmidt model and the Urieli and Berchowitz model) have large indicated power errors and large thermal efficiency errors when compared with the experimental data of GPU-3 beta Stirling engine. However, it is shown in Table 2.10 that ideal thermodynamics models such as the ideal adiabatic analysis and ideal polytropic model by other authors also have large indicated power error as high as 176.7% and 157.6%; and large thermal efficiency error of 41% and 39.06%.

According to the simulation results of computer program in Appendix A and other ideal thermodynamics models by other authors, it is clear that the results from the simulation using ideal processes such as adiabatic and isothermal processes without considering any losses may have large differences from the experimental results. The indicated power error can be as large as around 110% to 180%, while the thermal efficiency error can range between 38% to 50%.

It is also can be observed that there are some slight differences between the PV diagrams, indicated powers and thermal efficiencies of Schmidt model and that of the Urieli and Berchowitz model. From Figure 4.3, the pressure changes across the engine cycle for Schmidt model and that of Urieli and Berchowitz model are different. If the pressure equations of these 2 models are compared, it can be found that their differences is the fluctuation of temperature values in compression space and expansion space in Urieli and Berchowitz model. Referring to the energy balance equation from section 2.2.2, it can be shown that some work/heat energy goes to/from the internal energy of the gas (rise or fall of the gas temperature) in the adiabatic model of Urieli and Berchowitz; while heat energy/work in isothermal model of Schmidt analysis transfer interchangeability (reversible process) without the change in gas internal energy (assume the gas behave like an ideal gas). Therefore, the work done by the gas for these 2 models are different when the engine pressures are different in the engine cycle.

It is also found out that the thermal efficiency simulated by Schmidt model is very close to the ideal Stirling / Carnot efficiency (70.52% at 288K cooler temperature and 977K heater temperature) while thermal efficiency simulated by Urieli and Berchowitz model has a difference of 10.41% from the Carnot efficiency. The thermal efficiency simulated by Schmidt model is close to the Carnot efficiency due to the assumed reversible isothermal compression/expansion process in the engine.

4.3 Simulation 2: Non-sinusoidal Piston Motion Stirling Engine Performance

Computer program in Appendix B uses the dimension and operation data of the Stirling engine prototype of Wong and Goh (2020) shown in Table 2.11 to simulate the indicated power and engine thermal efficiency. Figure 4.5 to Figure 4.8 and Table 4.2 show the simulation results from the computer program written.

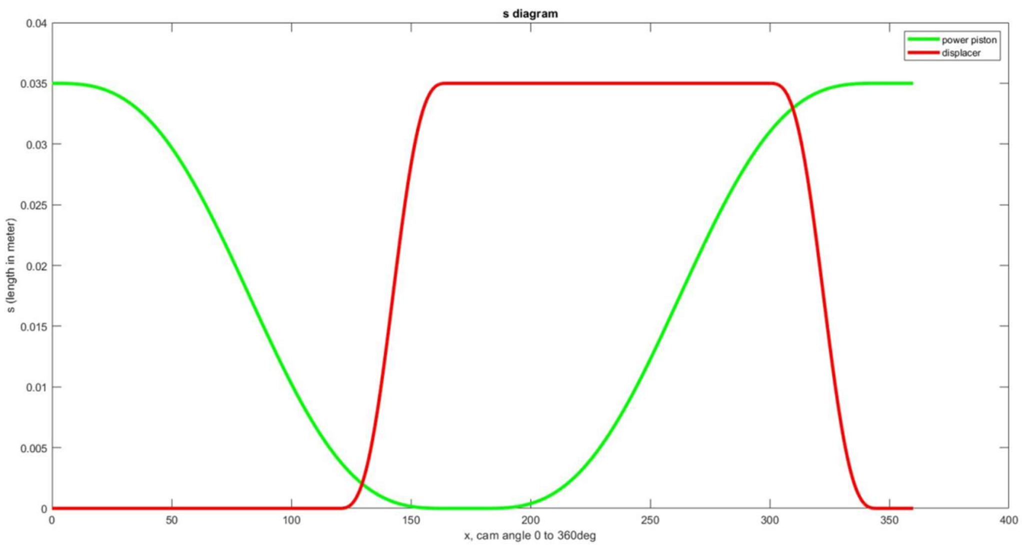


Figure 4.5: Piston motion diagram of simulation 2

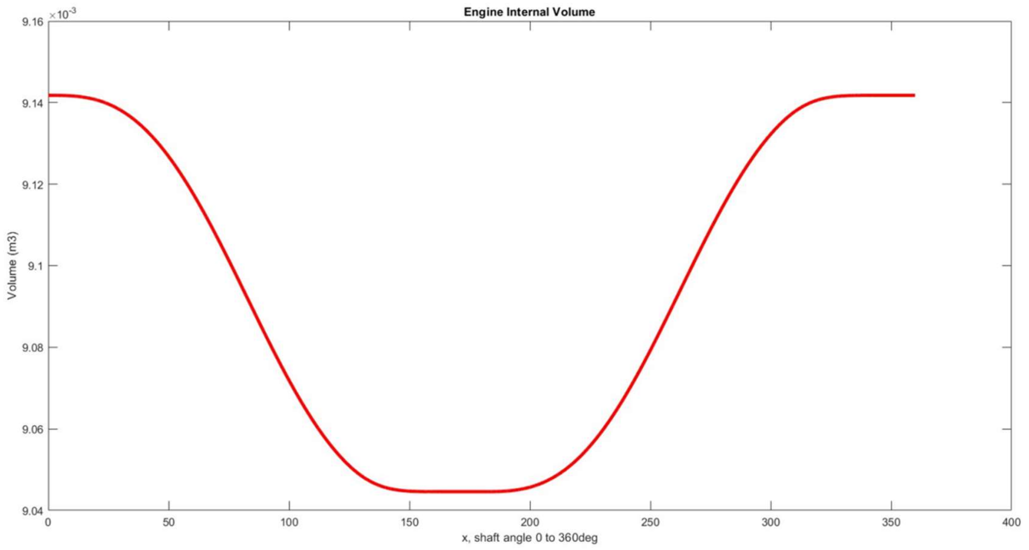


Figure 4.6: Engine internal volume of simulation 2

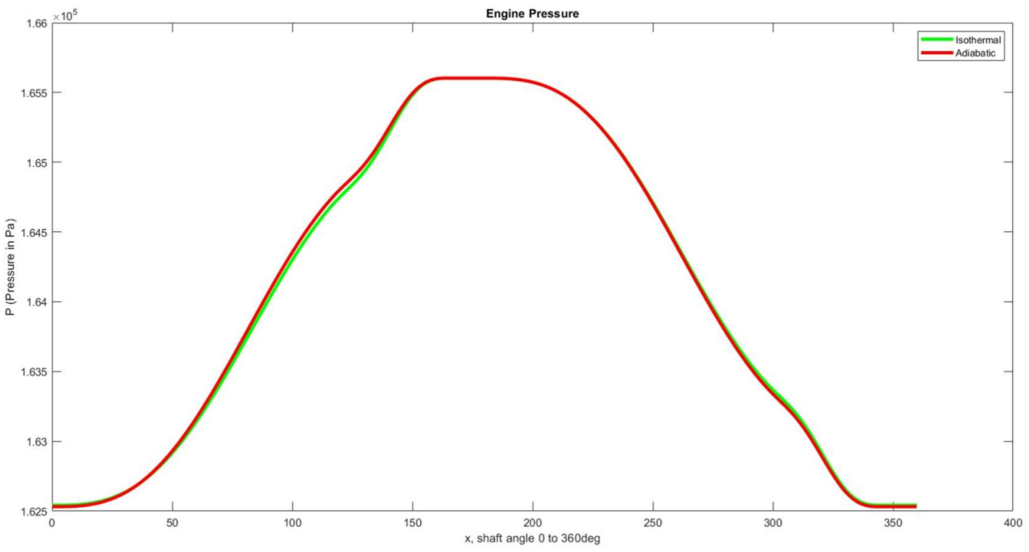


Figure 4.7: Engine pressure of simulation 2

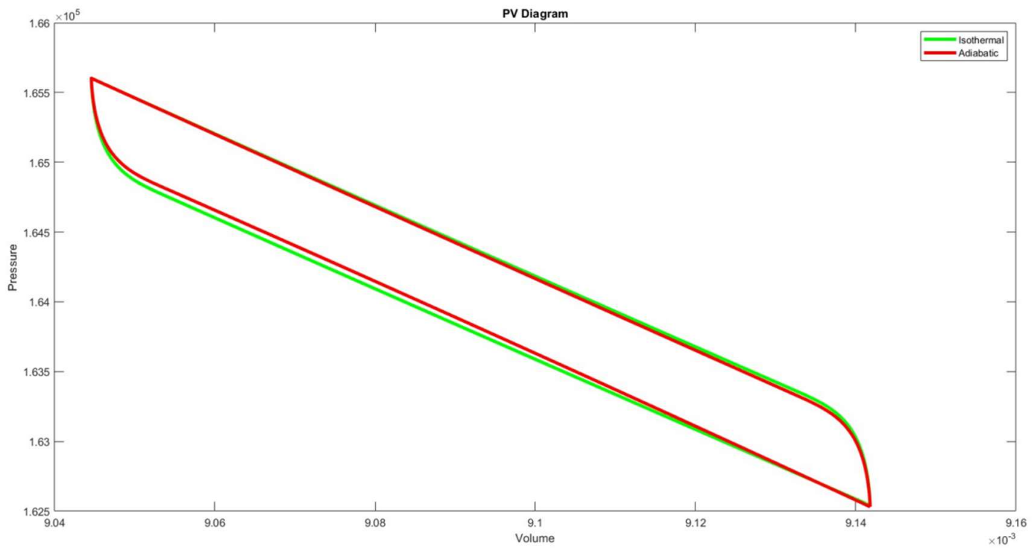


Figure 4.8: PV diagram of simulation 2

Table 4.2: Simulated engine performance of simulation 2

Description	Schmidt model Simulation Data	Urieli and Berchowitz model Simulation Data
Engine Indicated Work (J) per revolution	0.6389	0.5812
Engine Indicated Power (W)	1.1499	1.0461
Engine Thermal Efficiency (%)	53.49	0.0321
Indicated power error when compared to experimental result (%)	87.58	70.65
Thermal efficiency error when compared to experimental result (%)	53.32	-0.1422

Based on the simulation result in Table 4.2, when it is compared with experimental results by Wong and Goh (2020), similar to simulation 1, the indicated power error and thermal efficiency error are large due to the idealised assumptions made in the thermodynamics models. The assumptions made on heater space, cooler space, regenerator space and gass mass in the simulation also contribute to the errors in the simulation result as these informations are not clearly defined in the prototype engine.

After obtaining engine pressure, the gas torque exerted on the oscillating followers is calculated and shown in Figure 4.9. Gas torque information is important in the calculation of the mechanical power output from the indicated power output. It is also important in the design of the balancing method. Besides simulating the engine performance, the computer program in Appendix B also helps in the design of groove cam as shown in Figure 4.10 and Figure 4.11.

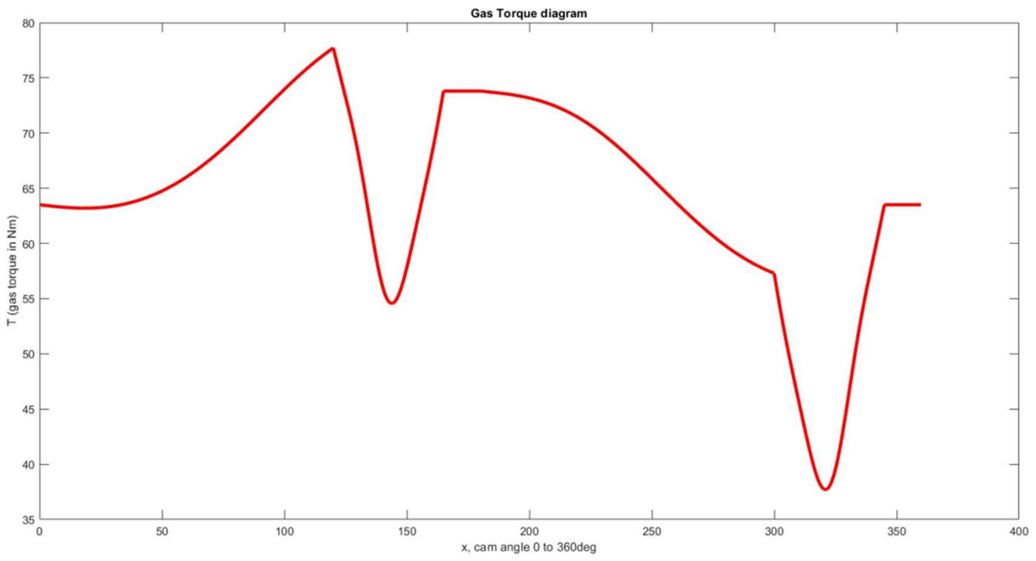


Figure 4.9: Gas Torque diagram of simulation 2

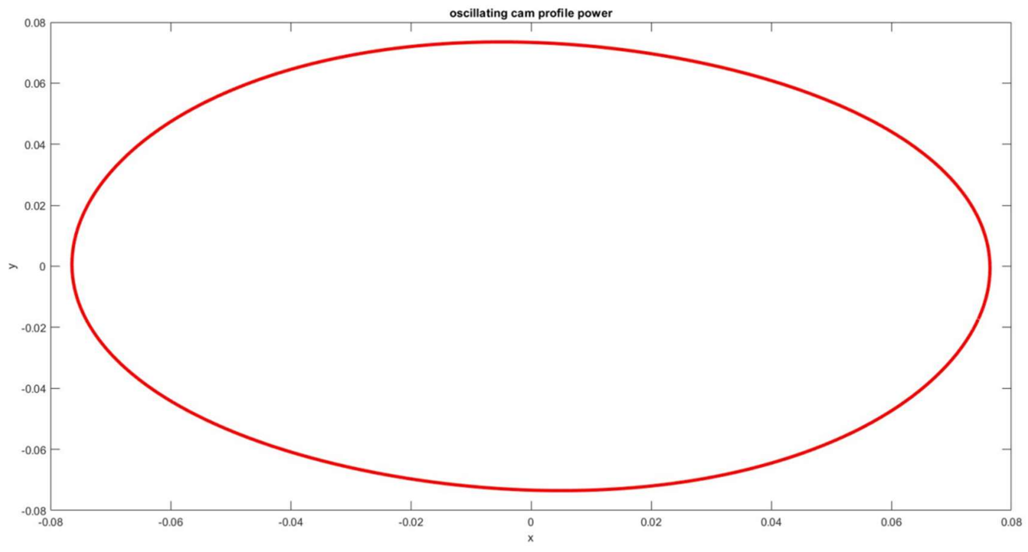


Figure 4.10: Cam groove profile at power piston of simulation 2

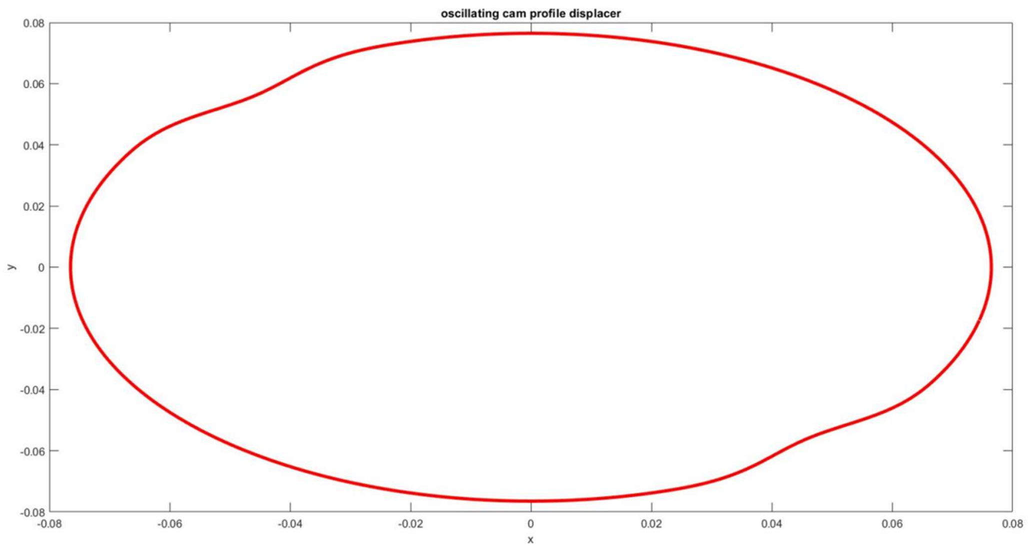


Figure 4.11: Cam groove profile at displacer piston of simulation 2

4.3.1 Simulation 2.1: Comparison with Sinusoidal Piston Motion

The computer program in Appendix C simulates the sinusoidal Stirling engine with similar dimension of the Stirling engine by Wong and Goh (2020) and operation data as Simulation 2. Figure 4.12 to Figure 4.15 and Table 4.3 show the simulation results from the computer program written.

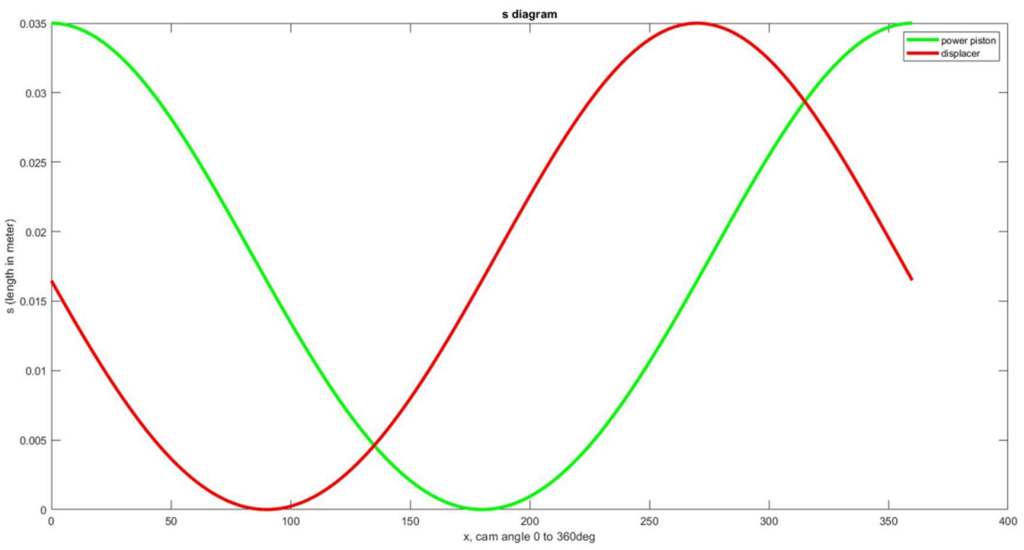


Figure 4.12: Piston motion diagram of simulation 2.1

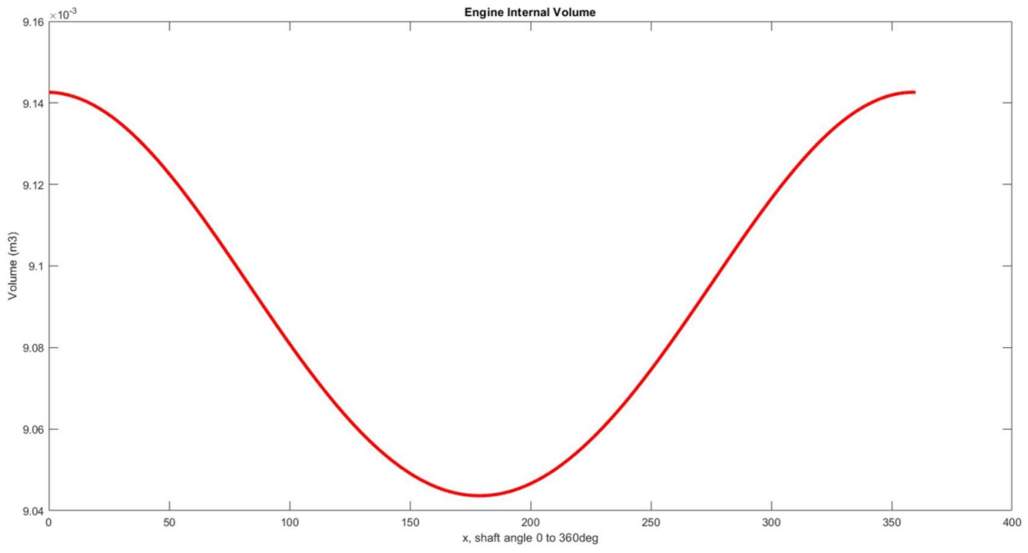


Figure 4.13: Engine internal volume diagram of simulation 2.1

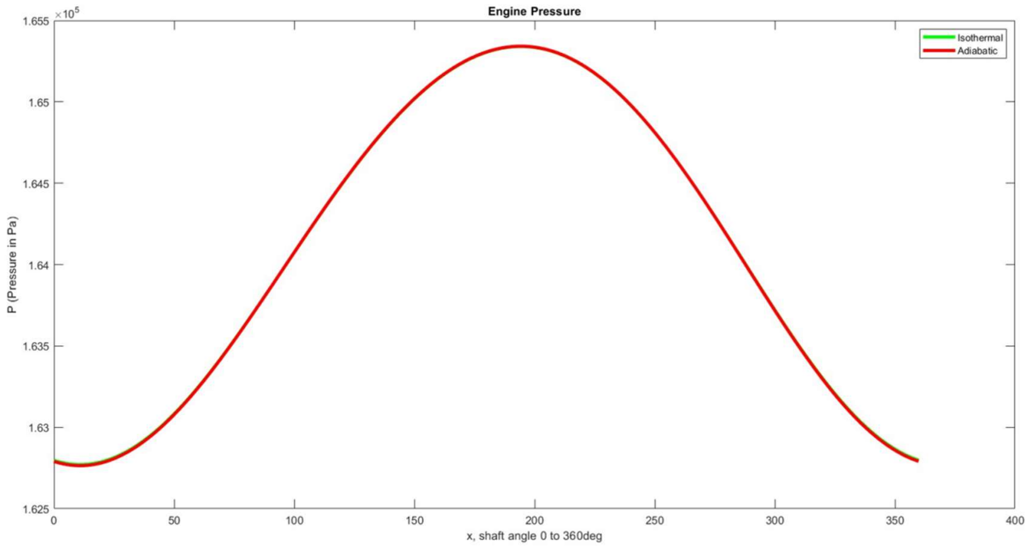


Figure 4.14: Engine pressure diagram of simulation 2.1

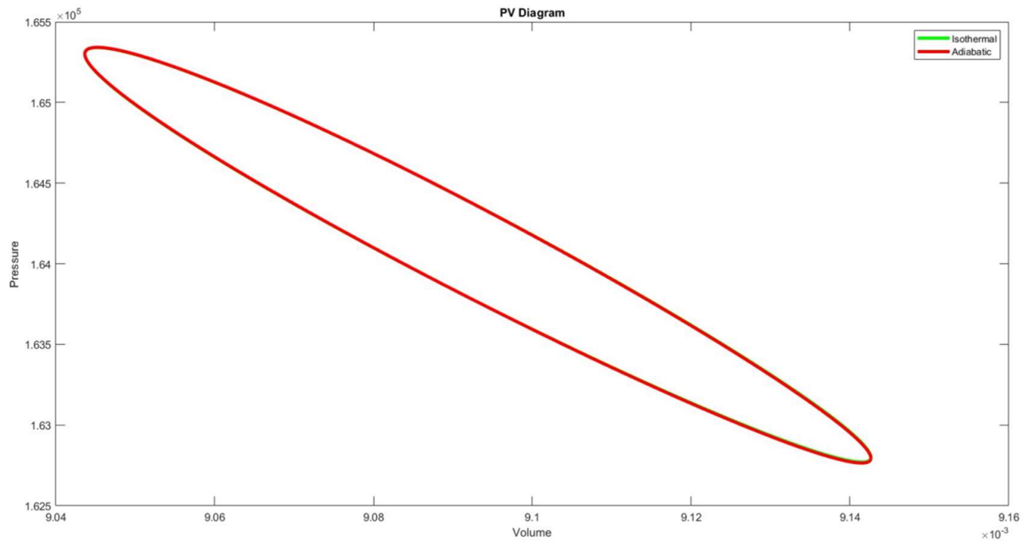


Figure 4.15: PV diagram of simulation 2.1

Table 4.3: Simulated engine performance of simulation 2.1

Description	Schmidt model Simulation Data	Urieli and Berchowitz model Simulation Data
Engine Indicated Work (J) per revolution	0.0464	0.0463
Engine Indicated Power (W)	0.0834	0.0834
Engine Thermal Efficiency (%)	53.49	53.29
Indicated power error when compared to experimental result(%)	-81.38	-81.38
Thermal efficiency error when compared to experimental result(%)	53.36	53.16

By comparing the data in Table 4.2 and Table 4.3, it is found that the engine indicated power for non-sinusoidal piston motion is greater than the sinusoidal piston motion by 1.0665W in Schmidt model simulation and 0.9627W in Urieli and Berchowitz model Simulation. The increase in the indicated output is the result of the expanding area of the PV curve when non-sinusoidal piston motion is introduced. The difference can be seen by comparing Figure 4.8 and Figure 4.15.

In addition, There is no significant difference in the thermal efficiency between Simulation 2 and Simulation 2.1, while the experimental work done by Wong and Goh (2020) shows a difference of around 30%. It is probably due to the absence of losses analysis in the model built to simulate more accurate thermal efficiencies for comparison. Similar to Simulation 2, the large indicated power error and thermal efficiency error when compared to the experimental work are contributed by the idealised conditions in the thermodynamics model and the assumption made on the on heater space, cooler space, regenerator space and gass mass in the simulation.

Similar to Simulation 1, the thermal efficiency simulated by Schmidt model is very close to the Carnot efficiency (53.49% at 313K cooler temperature and 673K heater temperature) while thermal efficiency simulated by Urieli and Berchowitz model has a difference of 0.2% from the Carnot efficiency.

4.4 Simulation 3: Non-sinusoidal Piston Motion with modified Scotch Yoke Mechanism

The computer program in Appendix E simulates the non-sinusoidal Stirling engine with modified Scotch Yoke (MSY). Figure 4.16 to Figure 4.19 and Table 4.4 show the simulation results from the computer program written.

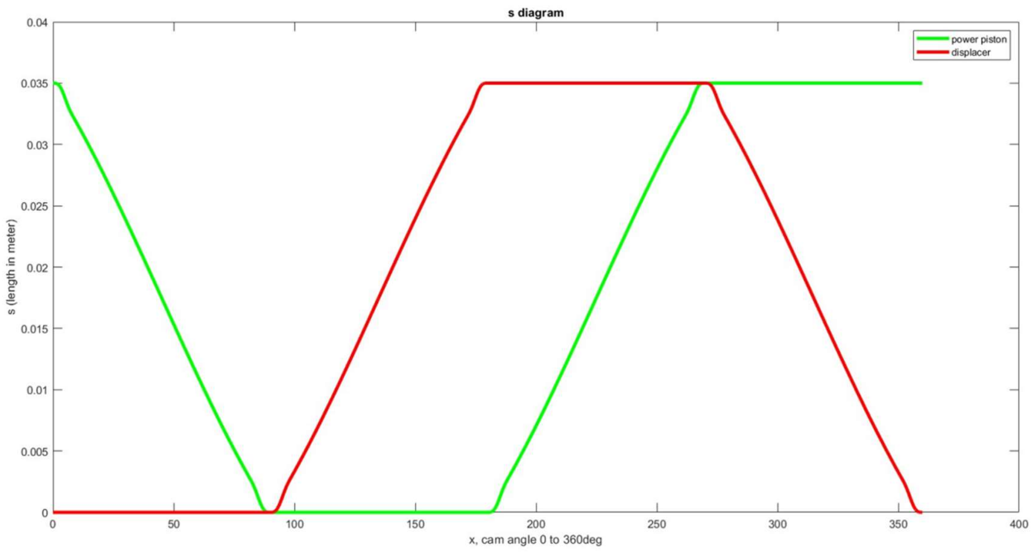


Figure 4.16: Piston motion diagram of simulation 3

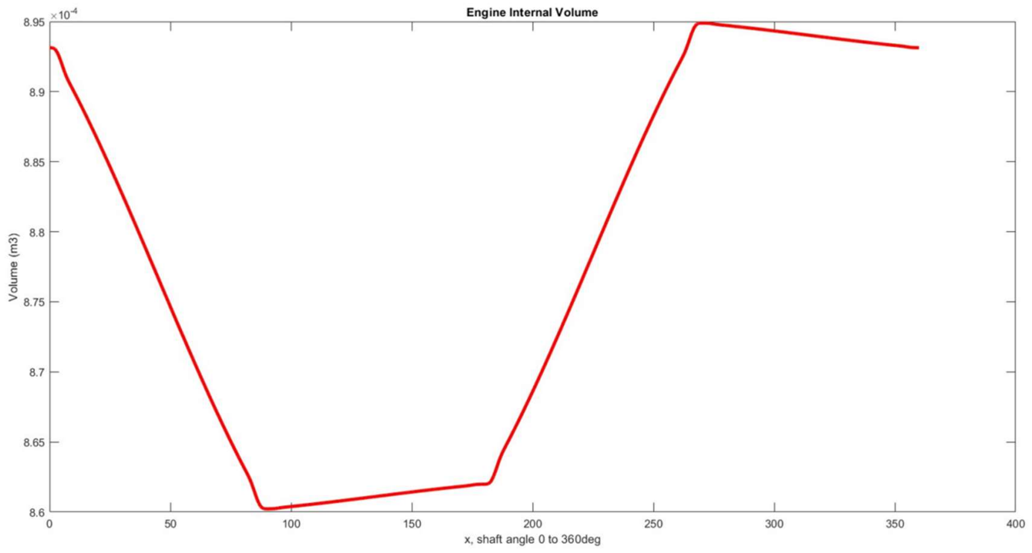


Figure 4.17: Engine internal volume diagram of simulation 3

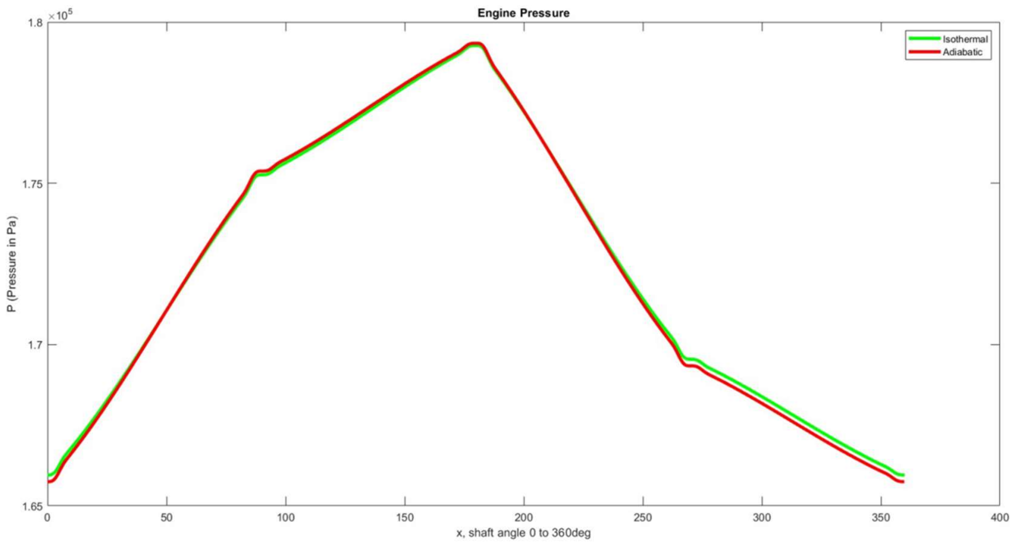


Figure 4.18: Engine pressure diagram of simulation 3

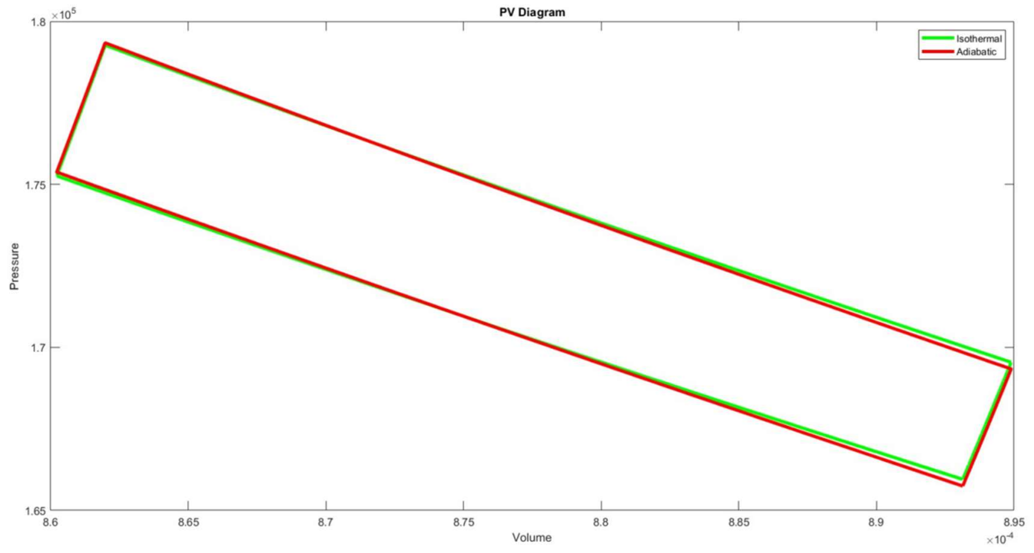


Figure 4.19: PV diagram of simulation 3

Table 4.4: Simulated engine performance of simulation 3

Description	Schmidt model Simulation Data	Urieli and Berchowitz model Simulation Data
Engine Indicated Work (J) per revolution	0.1416	0.1410
Engine Indicated Power (W)	0.2548	0.2539
Engine Thermal Efficiency (%)	33.83	32.65

According to Figure 4.16, the introduced MSY mechanism can generate piston motion that are close to the shape of the ideal piston motion of Stirling engine. When the piston motion of the mechanism is closer to the ideal piston motion, the work output of the engine can be increased. One of the disadvantages of the use of modified Scotch Yoke is that it may introduce more jerk during the transition from dwell to rising/falling due to the addition of the high order polynomial transition as shown in the displacement diagram (Figure 4.16) and the jerk diagram (Figure 4.20). It is observed that the jerk diagram is continuous although there is large change in values at the transition curve along the engine cycle. The polynomial transition curve eliminate the discontinuity of the jerk diagram but increases the jerk in the system. In addition, computer program in Appendix E also includes the design of modified Scotch Yoke slot curve, where the results is shown in Figure 4.21.

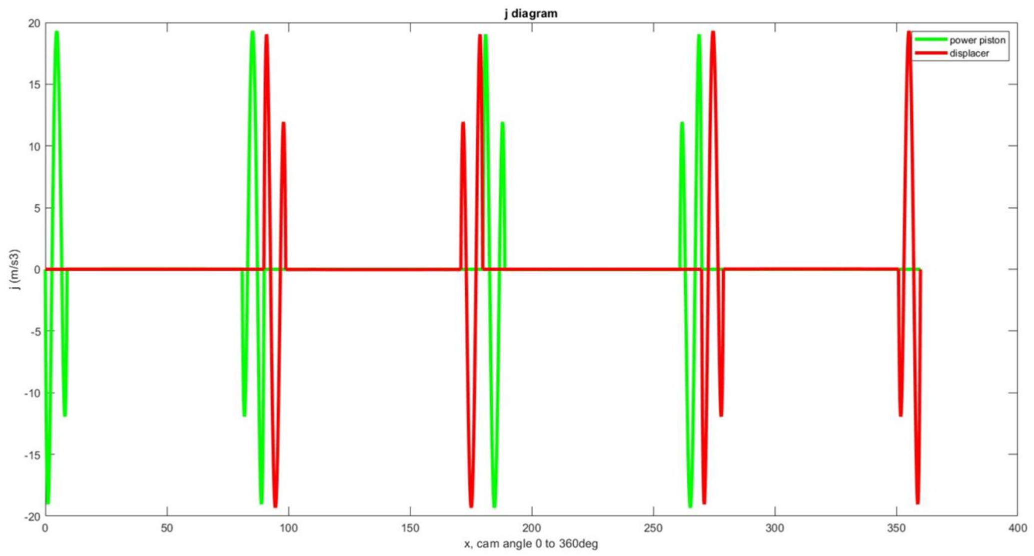


Figure 4.20: Modified Scotch Yoke piston jerk diagram

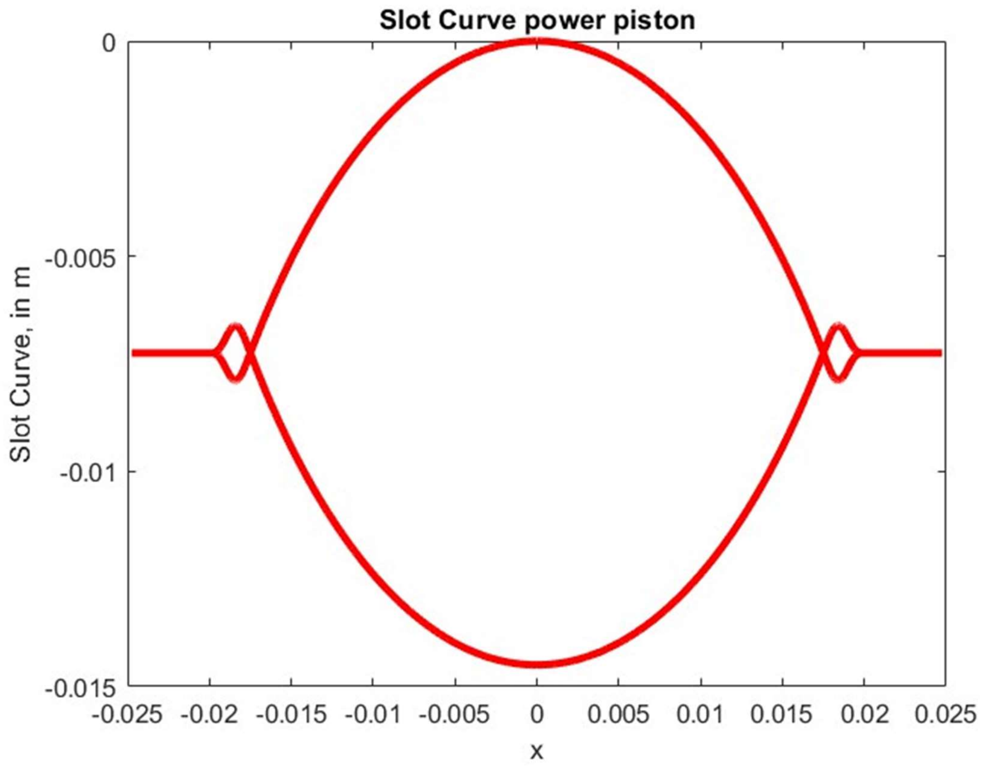


Figure 4.21: Modified Scotch Yoke slot curve of simulation 3

4.5 Simulation 3.1: Comparison with Sinusoidal Piston Motion

The computer program in Appendix F (Simulation 3.1) simulates the sinusoidal Stirling engine with same operation data as Simulation 3. The simulation results by Simulation 3 and Simulation 3.1 are compared to analyse the change in engine performance. Figure 4.22 to Figure 4.25 and Table 4.5 show the simulation results from Simulation 3.1.

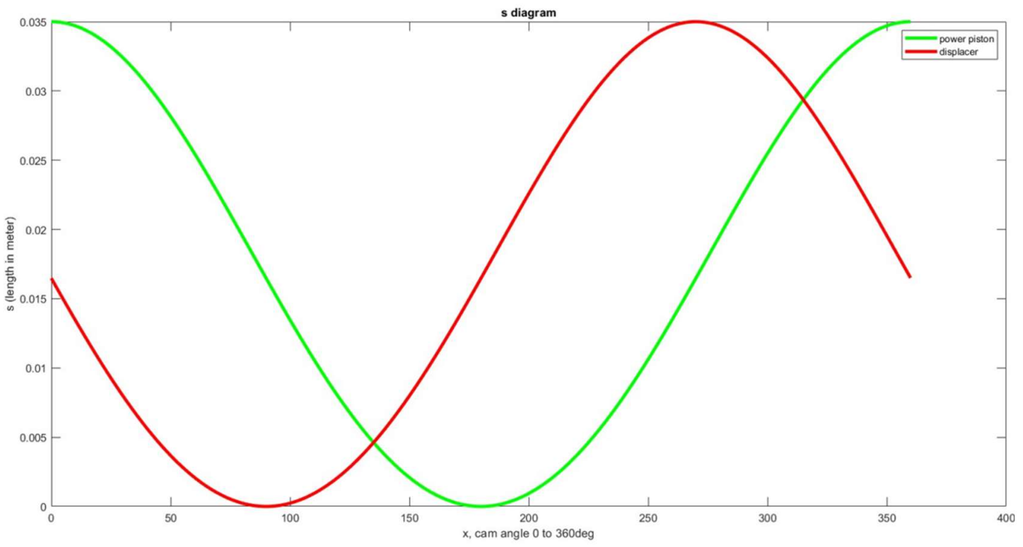


Figure 4.22: Piston motion diagram of simulation 3.1

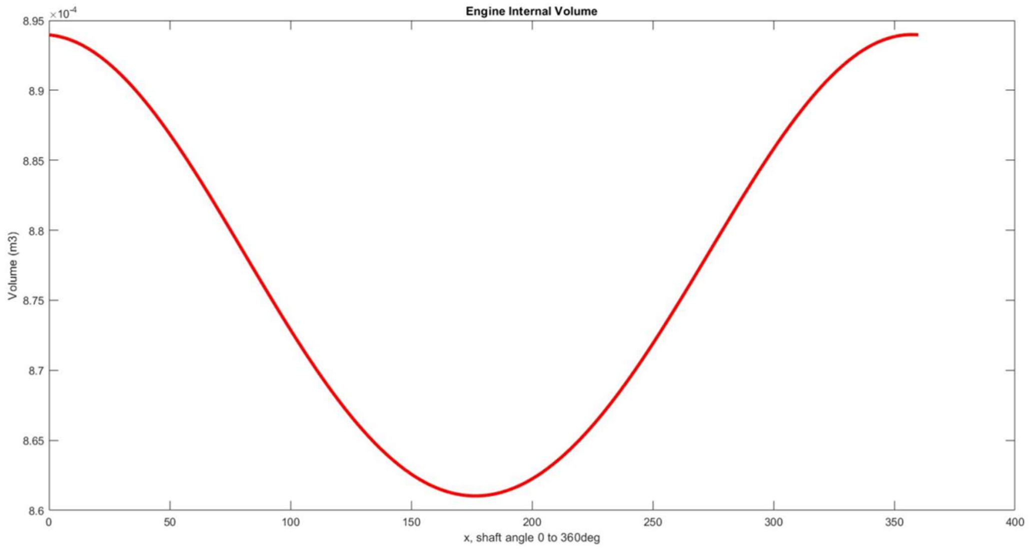


Figure 4.23: Engine internal volume diagram of simulation 3.1

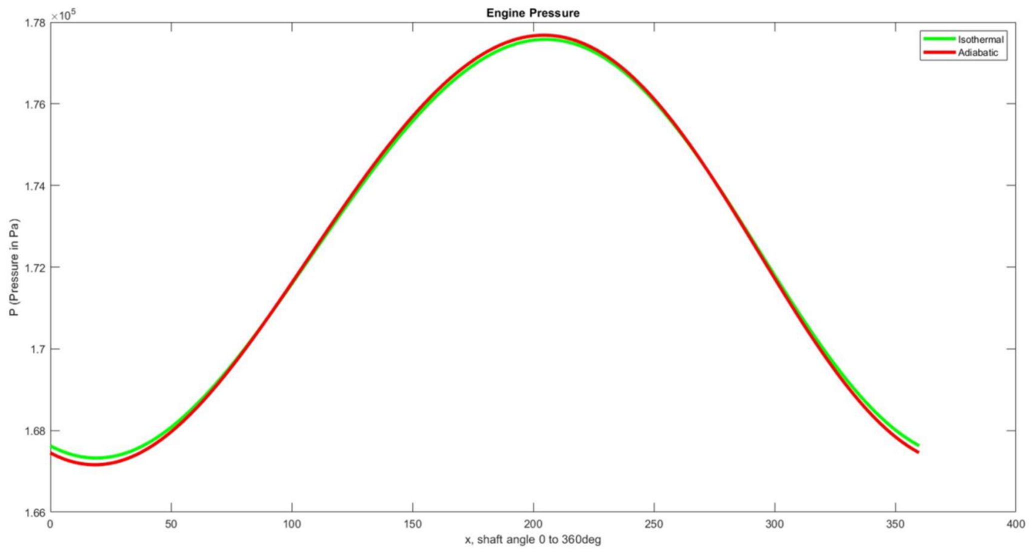


Figure 4.24: Engine pressure diagram of simulation 3.1

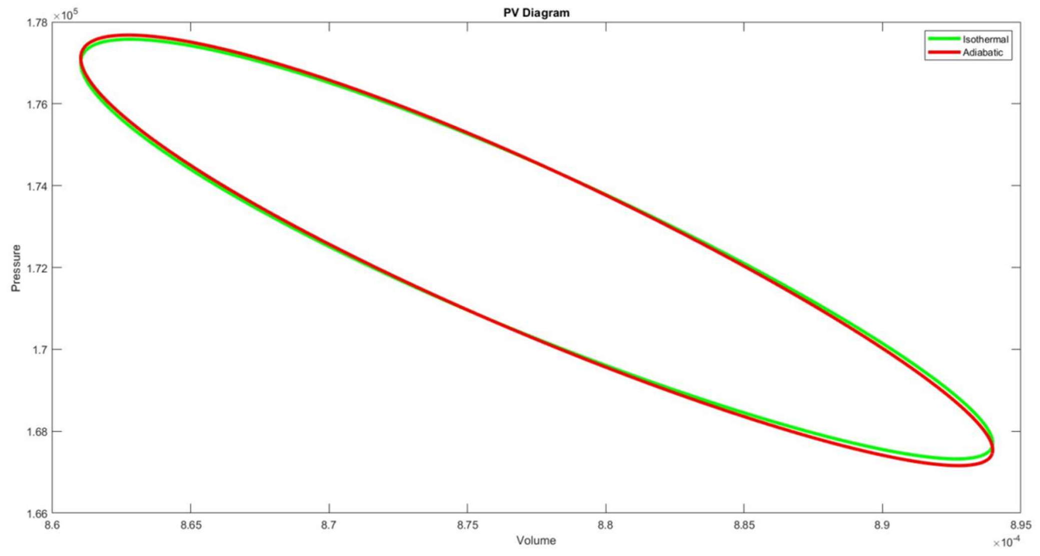


Figure 4.25: PV diagram of simulation 3.1

Table 4.5: Simulated engine performance of simulation 3.1

Description	Schmidt model Simulation Data	Urieli and Berchowitz model Simulation Data
Engine Indicated Work (J) per revolution	0.1112	0.1117
Engine Indicated Power (W)	0.2002	0.2010

Engine Thermal Efficiency (%)	33.83	32.96

By comparing the data between Table 4.4 and 4.5, the indicated power increased by around 26% while the thermal efficiency is decreased by less than 1% when using non-sinusoidal piston motion with MSY mechanism. Applying MSY mechanism in piston motion generation that are closer to the ideal piston motion can greatly increase the indicated power by expanding the PV diagram of engine. The small drop in thermal efficiency may be contributed by the small volume fluctuation at the transition of piston motion from dwell to sinusoidal rising/falling.

Similar to Simulation 1, the thermal efficiency simulated by Schmidt model is very close to the Carnot efficiency (33.83% at 313K cooler temperature and 473K heater temperature) while thermal efficiency simulated by Urieli and Berchowitz model has a difference of 0.87% from the Carnot efficiency.

4.6 Simulation 2.2: Improvement in the Cam Design

It is found that the jerk diagram the pistons in Simulation 2 is discontinuous as shown in Figure 4.26. From the figure, the jerk diagram is not smooth. An improvement can be made to the design of cam by replacing the cycloidal method with polynomial method to smoothen the jerk diagram as in Simulation 2.2.

As mentioned in section 3.2.3.2, using polynomial function in the design of double dwell cam profile has the advantage of getting smooth jerk diagram. Figure 4.26 and Figure 4.28 show the new piston motion diagram and the new jerk diagram after using polynomial function instead of cycloidal function (refer to Appendix D for the computer program). The new jerk diagram is now smooth and continuous. One thing to note is that the power piston jerk is much smaller than the displacer piston due to the difference in the steepness of the piston motion curve as their dwell periods are different. The amplitude of the piston jerk by polynomial method is observed to have much larger values than cycloidal method. This increase of jerk amplitude in the polynomial method is similar to the observation made by Norton (2007). Polynomial method has better vibration control than cycloidal method in exchange of larger theoretical peak acceleration and eventually larger jerk (Norton, 2007). Larger peak acceleration may increase the inertia forces at the joints, however, less vibration can reduce the wear and tear of the mechanism especially when the engine runs at high speed.

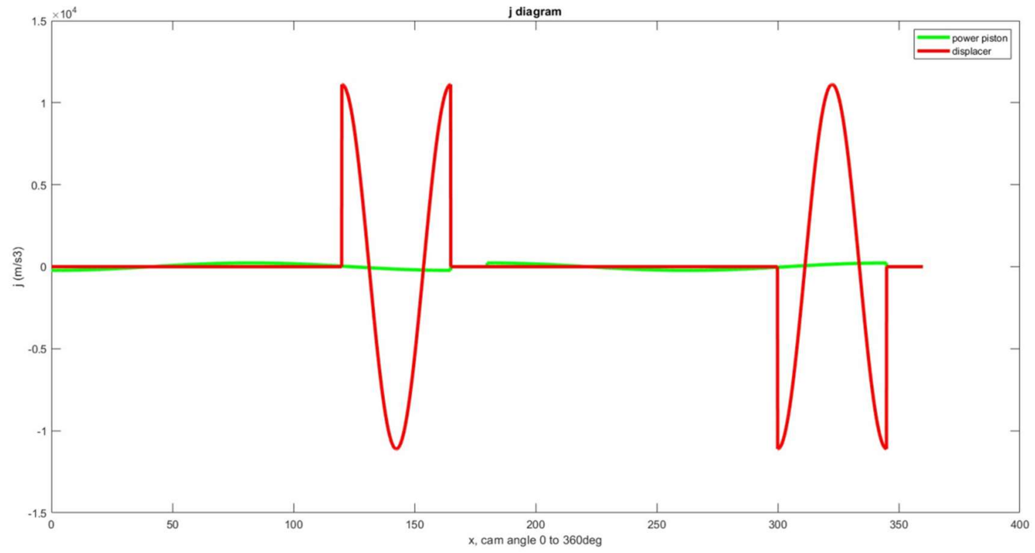


Figure 4.26: Jerk diagram of simulation 2

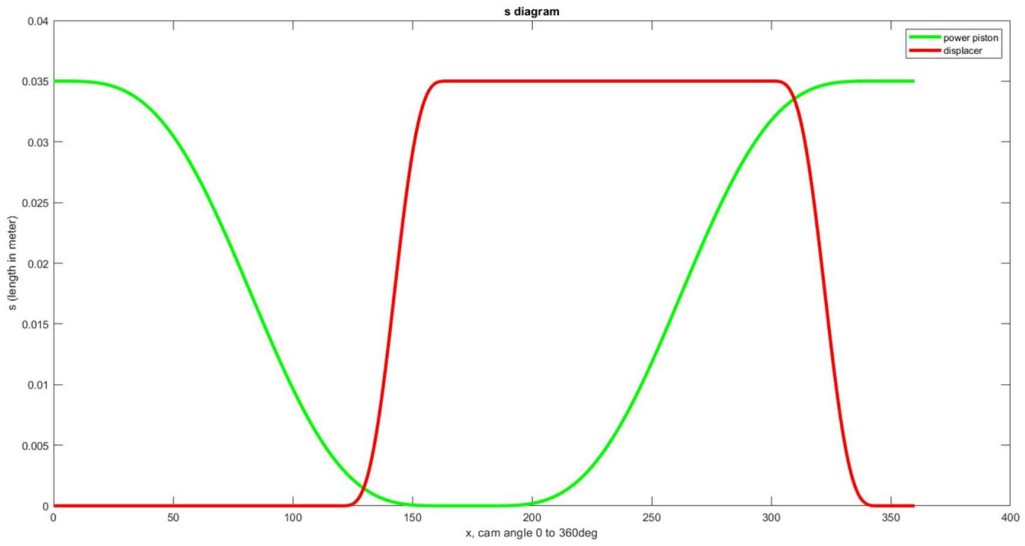


Figure 4.27: Piston motion diagram of simulation 2.2

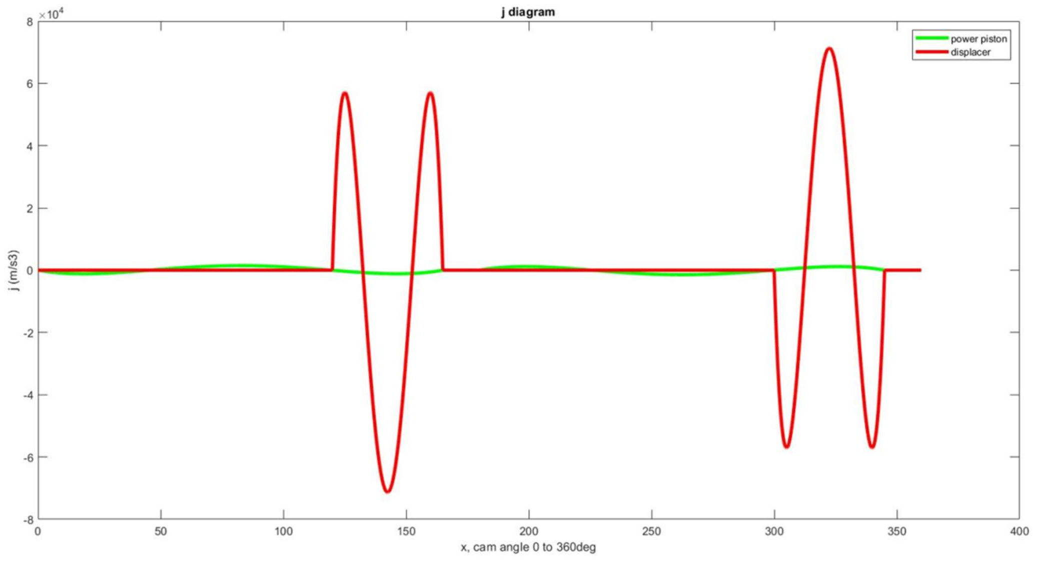


Figure 4.28: Jerk diagram of simulation 2.2

Table 4.6: Simulated engine performance of simulation 2.2

Description	Schmidt model Simulation Data	Urieli and Berchowitz model Simulation Data
Engine Indicated Work (J) per revolution	0.6505	0.5919
Engine Indicated Power (W)	1.1708	1.0654
Engine Thermal Efficiency (%)	53.49	0.0324

Using polynomial function in the cam design also will not has large effect on the engine performance. Comparing the results in Table 4.6 and Table 4.2, the difference in engine indicated power between two types of cam design methods is less than 2% (slight increase) while there is no significant change in the engine thermal efficiency.

4.7 Summary

Simulation programs used in this research are capable of simulating Stirling engines with acceptable accuracy. Simulation 1 showed that the programs used are very similar to other researchers, therefore the programs are verified of its accuracy. Just to note, any ideal thermodynamic model analysis without consideration of losses will have large errors in their simulated results when compared to the real engine data as shown by Laazaar and Boutammachte (2022), where the power error can be more than 100% while the thermal efficiency error can be more than 50%.

Introducing non-sinusoidal piston motions can improve a Stirling engine's performance. For radial cam-follower mechanism, by using Schmidt model simulation, it shows an increase of 12.79% indicated power from 0.0834W (sinusoidal) to 1.1499W (non-sinusoidal) when using. For MSY mechanism, by using Schmidt model simulation, it shows an increase of 27.27% indicated power from 0.2002W (sinusoidal) to 0.2548W (non-sinusoidal) when the engine is simulated using Schmidt model. The non-sinusoidal piston motions generated by radial cam-follower and MSY can expand the P-V curve of the Stirling engine, and eventually increase the indicated power. The increase in engine indicated power by expanding the PV curve using non-sinusoidal piston motion is also shown in the experimental work by Nicol-Seto and Nobes (2021), where the power output of the engine increase by 1.4% when e=1/3 oval gears are used to replace the conventional sinusoidal mechanism. Wong and Goh (2020) also shows the improvement of indicated power by 37% when RDFD cam mechanism is used.

The improvement of Stirling engine performance by cam-follower and modified Scotch Yoke are comparable in simulation. However, modified Scotch Yoke will introduce more jerk to the system at the transition of rising/falling to dwell.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

Thermodynamic models of Stirling engine, Schmidt model (isothermal model) and Urieli and Berchowitz model (adiabatic model) had successfully built as listed in the computer programs in the appendices to simulate the Stirling engine with non-sinusoidal piston motion. Stirling engine output work and the thermal efficiency had been successfully simulated by the computer programs to study the engine performance.

The performance of the gamma Stirling engine under non-sinusoidal piston motion generated by radial cam/groove cam/form-closed cam with oscillating follower and the modified Scotch yoke mechanism had improved in terms of indicated power and the thermal efficiency according to the simulation results. Stirling engine under non-sinusoidal piston motion generated by groove cam with oscillating follower can increase the indicated power by 1.0665W while no significant change in the thermal efficiency when compared to the sinusoidal piston motion engine during the simulation. Non-sinusoidal piston motion generated by modified Scotch yoke mechanism is different from the piston motion generated by groove cam with oscillating follower. According the simulation result, applying modified Scotch yoke mechanism to the Stirling engine can increase the indicated output power by around 26% and decrease the engine thermal efficiency by less than 1% when it is compared to the conventional near sinusoidal crank-slider mechanism.

Comparison between the numerical model of the gamma Stirling engine under non-sinusoidal piston motion generated by groove cam with oscillating follower and the experimental results of the engine prototype built by Wong and Goh (2020) are studied in this project. The error in indicated power is 87.58% and the thermal efficiency error is 53.32%. Absence of losses analysis, and assumption made on the heat exchanger spaces and gas mass are among possible contributions to these large errors. In summary, work output and thermal efficiency of the real Stirling engine will always much lower than the simulated results due to the losses. If the simulated indicated power of the engine is 2kW, the real Stirling engine indicated power will be much smaller at around 0.8kW if a 150% error is expected.

5.2 Recommendations for future work

More accurate engine dimension data and gas information such as total mass of gas and volume of gas in the heat exchanges are needed to be obtained from the real Stirling engine prototype when comparing the experimental data and the simulation result. More accurate comparison can be done with more accurate engine dimension data and gas information.

Besides calculating the gas torque of the engine, inertia toque calculation can be added into the simulation model in the future work by enquiring the mechanism mass and dimension information from the user. With gas torque and inertia torque calculated, this information can help the balancing work and stress analysis in the future.

Human machine interface of the computer program can be improved in the future work to make the process of inputting the user defined engine data into computer program more convenient. A dialog box can be displayed to prompt user to key in the necessary data instead of going to the program code to key in the data.

Losses analysis as shown by many authors can improve the accuracy of the simulation/mathematical/numerical model of the Stirling engine. Future work can be done to add losses analysis to the adiabatic model of the Stirling engine. It is also possible to implement optimization algorithms/evolutionary algorithms in the help of optimizing the engine parameters so that the performance of the engine can be improved.

REFERENCES

Abramowitz, M. and Stegun, I.A., 1968. *Handbook of mathematical functions with formulas, graphs, and mathematical tables (Vol. 55)*. Washington: US Government printing office.

Ahmadi, M.H., Ahmadi, M.A. and Pourfayaz, F., 2017. Thermal models for analysis of performance of Stirling engine: A review. *Renewable and Sustainable Energy Reviews*, 68, pp.168-184.

Alfarawi, S., Al-Dadah, R. and Mahmoud, S., 2016. Enhanced thermodynamic modelling of a gamma-type Stirling engine. *Applied Thermal Engineering*, 106, pp.1380-1390.

Araoz, J.A., Cardozo, E., Salomon, M., Alejo, L. and Fransson, T.H., 2015. Development and validation of a thermodynamic model for the performance analysis of a gamma Stirling engine prototype. *Applied Thermal Engineering*, 83, pp.16-30.

Babaelahi, M. and Sayyaadi, H., 2015. A new thermal model based on polytropic numerical simulation of Stirling engines. *Applied Energy*, 141, pp.143-159.

Boutammachte, N. and Knorr, J., 2012. Field-test of a solar low delta-T Stirling engine. *Solar energy*, 86(6), pp.1849-1856.

Cengel, Y.A., Boles, M.A. and Kanoğlu, M., 2011. *Thermodynamics: an engineering approach (Vol. 5)*. New York: McGraw-hill.

Dai, Z., Wang, C., Zhang, D., Tian, W., Qiu, S. and Su, G.H., 2021. Design and analysis of a free-piston stirling engine for space nuclear power reactor. *Nuclear Engineering and Technology*, 53(2), pp.637-646.

Dehelean, N.M., Ciupe, V. and Lovasz, E.C., 2009. A Digital Model of a Dwell Mechanism for Alpha-Stirling Engine. *In SYROM 2009: Proceedings of the 10th IFToMM International Symposium on Science of Mechanisms and Machines*, Brasov, 12-15 October, 2009. Netherlands: Springer. pp. 521-527.

Dehelean, N.M. and Ciupe, V., 2009. The Analysis of a Dwell Mechanism for Alpha-Stirling Engine. *In SYROM 2009: Proceedings of the 10th IFToMM International Symposium on Science of Mechanisms and Machines*, Brasov, 12-15 October, 2009. Netherlands: Springer, pp. 511-519.

Fang, H.W., Herold, K.E., Holland, H.M. and Beach, E.H., 1996. A novel Stirling engine with am elliptic drive. *In IECEC 96, Proceedings of the 31st Intersociety Energy Conversion Engineering Conference*. Washington, 11-16 August, 1996. New Jersey: IEEE, Vol. 2, pp. 1232-1237.

Furmanek, M. and Kropiwnicki, J., 2022. Stirling engines-the state of technology development and computational models. *Combustion Engines*, 61.

Gheith, R., Aloui, F. and Nasrallah, S.B., 2012. Study of the regenerator constituting material influence on a Gamma type Stirling engine. *Journal of Mechanical Science and Technology*, 26, pp.1251-1255.

Gopal, V.K., 2012. *Active Stirling Engine*. PhD Thesis, University of Canterbury. New Zealand.

Gopal, V.K., Duke, R. and Clucas, D., 2009. Active stirling engine. *In TENCON 2009-2009 IEEE Region 10 Conference*. 23-26 November 2009, Singapore. New Jersey: IEEE. pp. 1-6.

Kongtragool, B. and Wongwises, S., 2003. A review of solar-powered Stirling engines and low temperature differential Stirling engines. *Renewable and Sustainable energy reviews*, 7(2), pp.131-154.

Laazaar, K. and Boutammachte, N., 2022. Development of a new technique of waste heat recovery in cement plants based on Stirling engine technology. *Applied Thermal Engineering*, 210, p.118316.

Li, R., Grosu, L. and Queiros-Condé, D., 2016. Losses effect on the performance of a Gamma type Stirling engine. *Energy Conversion and Management*, 114, pp.28-37.

Liu, M., Zhang, B., Han, D., Du, X. and Wang, H., 2022. Experimental study on regenerative effectiveness and flow characteristics of parallel-plate regenerator in Stirling engine. *Applied Thermal Engineering*, 217, p.119139.

Martini, W.R., 1983. *Stirling engine design manual*. No. NASA-CR-168088. Washington: NAZA.

Melvin A. Vaux and Thomas R. Denner, 1993. *Dwelling scotch yoke engine*. US patent US5331926A.

Middleton, S.M.W., 2021. *A modular numerical model for Stirling engines and single-phase thermodynamic machines*. Master Thesis, University of Alberta. Canada.

Nicol-Seto, M. and Nobes, D., 2021. Experimental evaluation of piston motion modification to improve the thermodynamic power output of a low temperature gamma Stirling engine. *In E3S Web of Conferences Vol. 313, 19° International Stirling Engine Conference*. 2021. Les Ulis: EDP Sciences., pp. 04002.

Nicol-Seto, M.E., 2021. *Investigation of a Drive Mechanism Modification to Increase Thermodynamic Power of a Low Temperature Differential Gamma Type Stirling Engine*. Master Thesis, University of Alberta. Canada.

Norton, R.L., 2002. *Cam design and manufacturing handbook*. South Norwalk: Industrial Press Inc..

Norton, R.L., 2007. *Kinematics and dynamics of machinery*. Second Edition. New York: McGraw-Hill.

Petrescu, S., Borcila, B., Costea, M., Banches, E., Popescu, G., Boriaru, N., Stanciu, C. and Dobre, C., 2016, August. Concepts and fundamental equations in Thermodynamics with Finite Speed. *In IOP Conference Series: Materials Science and Engineering Vol. 147, No. 1*. 2016. Bristol: IOP Publishing. pp. 012144.

Rahmati, A., Varedi-Koulaei, S.M., Ahmadi, M.H. and Ahmadi, H., 2020. Dimensional synthesis of the Stirling engine based on optimizing the output work by evolutionary algorithms. *Energy Reports*, 6, pp.1468-1486.

Sowale, A., Kolios, A.J., Fidalgo, B., Somorin, T., Parker, A., Williams, L., Collins, M., McAdam, E. and Tyrrel, S., 2018. Thermodynamic analysis of a gamma type Stirling engine in an energy recovery system. *Energy conversion and management*, 165, pp.528-540.

Urieli, I. and Berchowitz, D.M., 1984. *Stirling cycle engine analysis*. London: Adam Hilger Limited.

Tavakolpour-Saleh, A.R., Zare, S.H. and Bahreman, H., 2017. A novel active free piston Stirling engine: Modeling, development, and experiment. *Applied energy*, 199, pp.400-415.

Walker, G., Senft, J.R., Walker, G. and Senft, J.R., 1985. Free-piston Stirling engines. *Springer Berlin Heidelberg*, pp.23-99.

Wang, H., Zhang, B., Liu, M. and Rao, Z., 2021. Analytical solution of heat transfer performance of pin-array stack regenerator in stirling cycle. *International Journal of Thermal Sciences*, 167, p.107015.

Wong, H.M., 2019. *A novel cam drive mechanism for displacer and piston motion control of a stirling engine*. PhD Thesis, Universiti Tunku Abdul Rahman. Malaysia.

Wong, H.M. and Goh, S.Y., 2020. Experimental comparison of sinusoidal motion and non-sinusoidal motion of rise-dwell-fall-dwell in a Stirling engine. *Journal of Mechanical Engineering and Sciences*, 14(3), pp.6971-6981.

Yu, M., Shi, C., Liu, Z. and Liu, W., 2022. Design and multi-objective optimization of a new annular constructal bifurcation Stirling regenerator using response surface methodology. *International Journal of Heat and Mass Transfer*, 195, pp.123129.

Yu, M., Xu, L., Cui, H., Liu, Z. and Liu, W., 2024. Characteristics and potential of a novel inclined-flow stirling regenerator constructed by sinusoidal corrugated channels. *Energy*, 288, p.129686.

APPENDICES

APPENDIX A: Computer Simulation Program 1 - Matlab Code for GPU-3 engine Simulation

```
%Clear all the previous data
clear all
close all
clc
%-------------------------
%User Defined Parameters
engine_power_stroke=0.0312; %Power Piston stroke
engine_displacer_stroke=0.0312; %Displacer Piston stroke
Diameter_piston_power=0.0699; %Power Piston Diameter
Diameter_piston_displacer=0.0699; %Displacer Piston Diameter
theta_engine_rad_step=4000; %Engine Angle discrete small steps for
numerical calculation. Note : Using RK4 method will reduce the steps by
half
w_engine=2*pi*41.7; %engine rotational speed; w=2(pi)(frequency)
T_h = 977; %degree K, heater temp.
T_k = 288; %degree K, cooler temp.
T_0 = 25+273; %degree K, ambient temp.
r_displacer_drive_rod=0.008/2; %Diameter of Displacer Piston drive rod in
meter
V_r=50.55*10^(-6); %Regenerator Dead Volume
V_k=13.8*10^(-6); %Cooler Dead Volume
V_h=70.88*10^(-6); %Heater Dead Volume
V_displacer_clearance_hot=30.52*10^(-9); %dead volume in displacer piston
at hot side
V_displacer_clearance_cold=28.68*10^(-9); %dead volume in displacer piston
at cold side
V_power_clearance=0*10^(-6); %dead volume in power piston
R_spec_gas=2076.9; %J/g/K*1000=J/kg/K, specific gas constant of helium at
ambient
density_gas=0.178; %kg/m3
M_gas_engine_initial=1.1362*10^(-3); %Total mass of gas in engine (in kg)
C_p_gas=5192.6; %constant pressure specific heat capacity (J/kg/K) ;Helium
Gas
```

```
C_v_gas=3115.6; %constant volume specific heat capacity (J/kg/K) ;Helium
Gas
k_isentropic=C_p_gas/C_v_gas;
%------------------------------
w_engine_rpm=w_engine/2/pi*60;
theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
theta_engine_rad=0;
for i=1:(theta_engine_rad_step-1)
theta_engine_rad=[theta_engine_rad
theta_engine_rad(1,i)+theta_engine_rad_step_size];
end
theta_engine_deg=theta_engine_rad/pi*180;
%Sinusoidal Piston Motion
engine_stroke=engine_power_stroke;
theta_crank_rad=theta_engine_rad;
x_piston_sinusoidal=engine_stroke/2*cos(theta_engine_rad);
v_piston_sinusoidal=-engine_stroke/2*w_engine.*(sin(theta_engine_rad));
a_piston_sinusoidal=-
engine_stroke/2*(w_engine.^2).*(cos(theta_engine_rad));
x_piston_power=x_piston_sinusoidal-min(x_piston_sinusoidal);
v_piston_power=v_piston_sinusoidal/w_engine; %length/rad
engine_stroke=engine_displacer_stroke;
x_piston_sinusoidal=engine_stroke/2*cos(theta_engine_rad);
v_piston_sinusoidal=-engine_stroke/2*w_engine.*(sin(theta_engine_rad));
a_piston_sinusoidal=-
engine_stroke/2*(w_engine.^2).*(cos(theta_engine_rad));
x_piston_sinusoidal=[x_piston_sinusoidal((1/4*theta_engine_rad_step+1):the
ta_engine_rad_step) x_piston_sinusoidal(1:(1/4*theta_engine_rad_step))];
x_piston_displacer=x_piston_sinusoidal-min(x_piston_sinusoidal);
v_piston_sinusoidal=[v_piston_sinusoidal((1/4*theta_engine_rad_step+1):the
ta_engine_rad_step) v_piston_sinusoidal(1:(1/4*theta_engine_rad_step))];
v_piston_displacer=v_piston_sinusoidal/w_engine; %length/rad
%Stirling Engine Piston Motion Display
figure
plot(theta_engine_deg,x_piston_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,x_piston_displacer,'r','LineWidth',3)
title('s diagram')
xlabel('x, cam angle 0 to 360deg')
```

```
ylabel('s (length in meter)')
legend('power piston','displacer')
%Stirling Engine Characteristics Initialisation
T_regenerator=(T_h-T_k)./(log(T_h./T_k));
r_piston_power=Diameter_piston_power/2; %in meter;
r_piston_displacer=Diameter_piston_displacer/2; %in meter
V_sw_hot=pi*(r_piston_displacer^2)*engine_displacer_stroke;
V_sw_cold=V_sw_hot-pi*(r_displacer_drive_rod^2)*engine_displacer_stroke;
V_sw_power=pi*(r_piston_power^2)*engine_power_stroke;
%Stirling Engine Piston Volume Calculation
V_hot_displacer=pi*(r_piston_displacer^2)*x_piston_displacer;
V_power=pi*(r_piston_power^2)*x_piston_power;
V_cold_displacer=V_sw_cold-pi*(r_piston_displacer^2-
r_displacer_drive_rod^2)*x_piston_displacer;
dV_hot_displacer=pi*(r_piston_displacer^2)*v_piston_displacer;
V_cold_engine=V_power+V_cold_displacer;
dV_cold_engine=pi*(r_piston_power^2)*v_piston_power-
pi*(r_piston_displacer^2-r_displacer_drive_rod^2)*v_piston_displacer;
%Integrate into Schmidt Model
V_hot=V_h+V_displacer_clearance_hot+V_hot_displacer;
dV_hot=dV_hot_displacer;
V_cold=V_k+V_displacer_clearance_cold+V_power_clearance+V_cold_engine;
dV_cold=dV_cold_engine;
%Integrate into Urieli Model
V_exp=V_displacer_clearance_hot+V_hot_displacer;
dV_exp=dV_hot_displacer;
V_comp=V_displacer_clearance_cold+V_power_clearance+V_cold_engine;
dV_comp=dV_cold_engine;
%Schmidt Model
P_engine_Schmidt=M_gas_engine_initial*R_spec_gas./(V_cold/T_k+V_r/T_regene
rator+V_hot/T_h);
m_regenerator_Schmidt=V_r*P_engine_Schmidt*(log(T_h/T_k))/(R_spec_gas*(T_h
-T_k));
Volume_engine_Schmidt=V_hot+V_cold+V_r;
Simpson_multiple=[];
for i=1:((theta_engine_rad_step)/2)
Simpson_multiple=[Simpson_multiple;2;4];
end
Simpson_multiple=[Simpson_multiple;1];
```

```
Simpson_multiple(1,1)=1;
Work_engine_part1=P_engine_Schmidt.*(dV_cold+dV_hot);
Work_engine_part1=[Work_engine_part1 Work_engine_part1(1,1)];
h=2*pi/theta_engine_rad_step;
Work_engine_Schmidt=h/3*Work_engine_part1*Simpson_multiple;
Words = 'Engine Work in J per revolution (Schmidt)';
disp(Words)
disp(Work_engine_Schmidt)
Work_done_Schmidt_power=Work_engine_Schmidt*w_engine_rpm/60;
Words = 'Engine Work in W (Schmidt)';
disp(Words)
disp(Work_done_Schmidt_power)
Work_engine_part2=P_engine_Schmidt.*(dV_hot);
Work_engine_part2=[Work_engine_part2 Work_engine_part2(1,1)];
h=2*pi/theta_engine_rad_step;
Work_engine_Schmidt_hot=h/3*Work_engine_part2*Simpson_multiple;
Q_hot_Schmidt=Work_engine_Schmidt_hot;
Engine_Efficiency_Schmidt=Work_engine_Schmidt/Q_hot_Schmidt;
Words = 'Engine Efficiency (Schmidt)';
disp(Words)
disp(Engine_Efficiency_Schmidt)
%Urieli model - ODEs in adiabatic analysis
%Separate odd and even rows for rk4 method 1/2step calculation
theta_engine_rad_step=theta_engine_rad_step/2;
theta_engine_rad_step_size=theta_engine_rad_step_size*2;
for i=1:theta_engine_rad_step
V_exp_odd(1,i)=V_exp(1,i*2-1);
dV_exp_odd(1,i)=dV_exp(1,i*2-1);
V_comp_odd(1,i)=V_comp(1,i*2-1);
dV_comp_odd(1,i)=dV_comp(1,i*2-1);
V_exp_even(1,i)=V_exp(1,i*2);
dV_exp_even(1,i)=dV_exp(1,i*2);
V_comp_even(1,i)=V_comp(1,i*2);
dV_comp_even(1,i)=dV_comp(1,i*2);
theta_engine_deg_odd(1,i)=theta_engine_deg(1,i*2-1);
end
V_exp=V_exp_odd;
dV_exp=dV_exp_odd;
V_comp=V_comp_odd;
```

```
dV_comp=dV_comp_odd;
Temperature_term1=V_k/T_k+V_r/T_regenerator+V_h/T_h;
T_comp=ones(1,theta_engine_rad_step+1).*T_k;
T_exp=ones(1,theta_engine_rad_step+1).*T_h;
T_ck=T_comp(1,1);
T_he=T_exp(1,1);
Q_h=zeros(1,theta_engine_rad_step);
Q_k=zeros(1,theta_engine_rad_step);
Q_r=zeros(1,theta_engine_rad_step);
W_exp=zeros(1,theta_engine_rad_step);
W_comp=zeros(1,theta_engine_rad_step);
T_kr=T_k;
T_rh=T_h;
for j=1:50
for i=1:(theta_engine_rad_step)
if i==1
M_gas_engine=M_gas_engine_initial;
end
P_engine_Urieli(1,i)=M_gas_engine*R_spec_gas/(V_comp(1,i)/T_comp(1,i)+Temp
erature_term1+V_exp(1,i)/T_exp(1,i));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli(1,i)*(dV_comp(1,i)/T_ck+dV_exp(1,i)/T_he);
dP_engine_Urieli_term2=V_comp(1,i)/T_ck+k_isentropic*Temperature_term1+V_e
xp(1,i)/T_he;
dP_engine_Urieli(1,i)=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli(1,i)*V_comp(1,i)/(R_spec_gas*T_comp(1,i));
m_k=P_engine_Urieli(1,i)*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli(1,i)*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli(1,i)*V_h/(R_spec_gas*T_h);
m_exp=P_engine_Urieli(1,i)*V_exp(1,i)/(R_spec_gas*T_exp(1,i));
dm_exp(1,i)=(P_engine_Urieli(1,i)*dV_exp(1,i)+V_exp(1,i)*dP_engine_Urieli(
1,i)/k_isentropic)/(R_spec_gas*T_he);
dm_comp(1,i)=(P_engine_Urieli(1,i)*dV_comp(1,i)+V_comp(1,i)*dP_engine_Urie
li(1,i)/k_isentropic)/(R_spec_gas*T_ck);
dm_k(1,i)=m_k*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
dm_r(1,i)=m_r*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
dm_h(1,i)=m_h*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
m_dot_ck=-dm_comp(1,i);
m_dot_he=dm_exp(1,i);
```

```
m_dot_kr=m_dot_ck-dm_k(1,i);
m_dot_rh=m_dot_he+dm_h(1,i);
if m_dot_ck>0
T_ck=T_comp(1,i);
else
T_ck=T_k;
end
if m_dot_he>0
T_he=T_h;
else
T_he=T_exp(1,i);
end
dT_exp=T_exp(1,i)*(dP_engine_Urieli(1,i)/P_engine_Urieli(1,i)+dV_exp(1,i)/
V_exp(1,i)-dm_exp(1,i)/m_exp);
dT_comp=T_comp(1,i)*(dP_engine_Urieli(1,i)/P_engine_Urieli(1,i)+dV_comp(1,
i)/V_comp(1,i)-dm_comp(1,i)/m_comp);
dQ_k=V_k*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_ck*m_dot_ck-
T_kr*m_dot_kr);
dQ_r=V_r*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_kr*m_dot_kr-
T_rh*m_dot_rh);
dQ_h=V_h*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_rh*m_dot_rh-
T_he*m_dot_he);
dW_exp=P_engine_Urieli(1,i)*dV_exp(1,i);
dW_comp=P_engine_Urieli(1,i)*dV_comp(1,i);
k1_exp=theta_engine_rad_step_size*dT_exp;
k1_comp=theta_engine_rad_step_size*dT_comp;
k1_dQk=theta_engine_rad_step_size*dQ_k;
k1_dQr=theta_engine_rad_step_size*dQ_r;
k1_dQh=theta_engine_rad_step_size*dQ_h;
k1_dWe=theta_engine_rad_step_size*dW_exp;
k1_dWc=theta_engine_rad_step_size*dW_comp;
T_ck_half=T_ck;
T_he_half=T_he;
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp_even(1,i)/(T_comp(1,i
)+0.5*k1_comp)+Temperature_term1+V_exp_even(1,i)/(T_exp(1,i)+0.5*k1_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp_even(1,i)/T_ck_half+dV_exp_even
(1,i)/T_he_half);
```

```
dP_engine_Urieli_term2=V_comp_even(1,i)/T_ck_half+k_isentropic*Temperature
_term1+V_exp_even(1,i)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp_even(1,i)/(R_spec_gas*(T_comp(1,i)+0.5*
k1_comp));
m_exp=P_engine_Urieli_half*V_exp_even(1,i)/(R_spec_gas*(T_exp(1,i)+0.5*k1_
exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp_even(1,i)+V_exp_even(1,i)*dP_engi
ne_Urieli_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp_even(1,i)+V_comp_even(1,i)*dP_e
ngine_Urieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+0.5*k1_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+0.5*k1_exp;
end
dT_exp=(T_exp(1,i)+0.5*k1_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half
+dV_exp_even(1,i)/V_exp_even(1,i)-dm_exp_half/m_exp);
k2_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+0.5*k1_comp)*(dP_engine_Urieli_half/P_engine_Urieli_h
alf+dV_comp_even(1,i)/V_comp_even(1,i)-dm_comp_half/m_comp);
k2_comp=theta_engine_rad_step_size*dT_comp;
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
```

```
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp_even(1,i);
dW_comp=P_engine_Urieli_half*dV_comp_even(1,i);
k2_dQk=theta_engine_rad_step_size*dQ_k;
k2_dQr=theta_engine_rad_step_size*dQ_r;
k2_dQh=theta_engine_rad_step_size*dQ_h;
k2_dWe=theta_engine_rad_step_size*dW_exp;
k2_dWc=theta_engine_rad_step_size*dW_comp;
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp_even(1,i)/(T_comp(1,i
)+0.5*k2_comp)+Temperature_term1+V_exp_even(1,i)/(T_exp(1,i)+0.5*k2_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp_even(1,i)/T_ck_half+dV_exp_even
(1,i)/T_he_half);
dP_engine_Urieli_term2=V_comp_even(1,i)/T_ck_half+k_isentropic*Temperature
_term1+V_exp_even(1,i)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp_even(1,i)/(R_spec_gas*(T_comp(1,i)+0.5*
k2_comp));
m_exp=P_engine_Urieli_half*V_exp_even(1,i)/(R_spec_gas*(T_exp(1,i)+0.5*k2_
exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp_even(1,i)+V_exp_even(1,i)*dP_engi
ne_Urieli_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp_even(1,i)+V_comp_even(1,i)*dP_e
ngine_Urieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+0.5*k2_comp;
```

```
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+0.5*k2_exp;
end
dT_exp=(T_exp(1,i)+0.5*k2_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half
+dV_exp_even(1,i)/V_exp_even(1,i)-dm_exp_half/m_exp);
k3_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+0.5*k2_comp)*(dP_engine_Urieli_half/P_engine_Urieli_h
alf+dV_comp_even(1,i)/V_comp_even(1,i)-dm_comp_half/m_comp);
k3_comp=theta_engine_rad_step_size*dT_comp;
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp_even(1,i);
dW_comp=P_engine_Urieli_half*dV_comp_even(1,i);
k3_dQk=theta_engine_rad_step_size*dQ_k;
k3_dQr=theta_engine_rad_step_size*dQ_r;
k3_dQh=theta_engine_rad_step_size*dQ_h;
k3_dWe=theta_engine_rad_step_size*dW_exp;
k3_dWc=theta_engine_rad_step_size*dW_comp;
i2=i+1;
if i2>theta_engine_rad_step
i2=i2-theta_engine_rad_step;
end
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp(1,i2)/(T_comp(1,i)+k3
_comp)+Temperature_term1+V_exp(1,i2)/(T_exp(1,i)+k3_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp(1,i2)/T_ck_half+dV_exp(1,i2)/T_
he_half);
dP_engine_Urieli_term2=V_comp(1,i2)/T_ck_half+k_isentropic*Temperature_ter
m1+V_exp(1,i2)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
```

```
m_comp=P_engine_Urieli_half*V_comp(1,i2)/(R_spec_gas*(T_comp(1,i)+k3_comp)
);
m_exp=P_engine_Urieli_half*V_exp(1,i2)/(R_spec_gas*(T_exp(1,i)+k3_exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp(1,i2)+V_exp(1,i2)*dP_engine_Uriel
i_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp(1,i2)+V_comp(1,i2)*dP_engine_Ur
ieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+k3_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+k3_exp;
end
dT_exp=(T_exp(1,i)+k3_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half+dV_
exp(1,i2)/V_exp(1,i2)-dm_exp_half/m_exp);
k4_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+k3_comp)*(dP_engine_Urieli_half/P_engine_Urieli_half+
dV_comp(1,i2)/V_comp(1,i2)-dm_comp_half/m_comp);
k4_comp=theta_engine_rad_step_size*dT_comp;
T_exp(1,i+1)=T_exp(1,i)+1/6*(k1_exp+2*k2_exp+2*k3_exp+k4_exp);
T_comp(1,i+1)=T_comp(1,i)+1/6*(k1_comp+2*k2_comp+2*k3_comp+k4_comp);
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
```

```
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp(1,i2);
dW_comp=P_engine_Urieli_half*dV_comp(1,i2);
k4_dQk=theta_engine_rad_step_size*dQ_k;
k4_dQr=theta_engine_rad_step_size*dQ_r;
k4_dQh=theta_engine_rad_step_size*dQ_h;
k4_dWe=theta_engine_rad_step_size*dW_exp;
k4_dWc=theta_engine_rad_step_size*dW_comp;
Q_k(1,i+1)=Q_k(1,i)+1/6*(k1_dQk+2*k2_dQk+2*k3_dQk+k4_dQk);
Q_r(1,i+1)=Q_r(1,i)+1/6*(k1_dQr+2*k2_dQr+2*k3_dQr+k4_dQr);
Q_h(1,i+1)=Q_h(1,i)+1/6*(k1_dQh+2*k2_dQh+2*k3_dQh+k4_dQh);
W_exp(1,i+1)=W_exp(1,i)+1/6*(k1_dWe+2*k2_dWe+2*k3_dWe+k4_dWe);
W_comp(1,i+1)=W_comp(1,i)+1/6*(k1_dWc+2*k2_dWc+2*k3_dWc+k4_dWc);
if i==theta_engine_rad_step
T_exp_diff(1,j)=T_exp(1,theta_engine_rad_step+1)-T_exp(1,1);
T_exp(1,1)=T_exp(1,theta_engine_rad_step+1);
T_exp(:,theta_engine_rad_step+1)=[];
T_comp_diff(1,j)=T_comp(1,theta_engine_rad_step+1)-T_comp(1,1);
T_comp(1,1)=T_comp(1,theta_engine_rad_step+1);
T_comp(:,theta_engine_rad_step+1)=[];
end
end
end
Work_done_Urieli=W_exp(1,theta_engine_rad_step+1)+W_comp(1,theta_engine_ra
d_step+1);
Engine_Efficiency_Urieli=Work_done_Urieli/Q_h(1,theta_engine_rad_step+1);
Words = 'Engine Work in J per revolution (Urieli)';
disp(Words)
disp(Work_done_Urieli)
Work_done_Urieli_power=Work_done_Urieli*w_engine_rpm/60;
Words = 'Engine Work in W (Urieli)';
disp(Words)
disp(Work_done_Urieli_power)
Words = 'Engine Efficiency';
disp(Words)
disp(Engine_Efficiency_Urieli)
%-----------------------------------
% PV Diagram Display:
```

```
Volume_engine_Urieli=V_comp+V_exp+V_r+V_h+V_k;
figure
plot(Volume_engine_Schmidt,P_engine_Schmidt,'g','LineWidth',3)
hold on
plot(Volume_engine_Urieli,P_engine_Urieli,'r','LineWidth',3)
title('PV Diagram')
xlabel('Volume')
ylabel('Pressure')
legend('Isothermal','Adiabatic')
%--------------------------------
figure
plot(theta_engine_deg,P_engine_Schmidt,'g','LineWidth',3)
hold on
plot(theta_engine_deg_odd,P_engine_Urieli,'r','LineWidth',3)
title('Engine Pressure')
xlabel('x, shaft angle 0 to 360deg')
ylabel('P (Pressure in Pa)')
legend('Isothermal','Adiabatic')
figure
plot(theta_engine_deg,Volume_engine_Schmidt,'r','LineWidth',3)
title('Engine Internal Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
```

APPENDIX B: Computer Simulation Program 2a - Matlab Code for Stirling engine with form-closed cam-oscillating follower (cycloidal) simulation

```
%Clear all the previous data
clear all
close all
clc
%User Defined Parameters
engine_power_stroke=0.035;
engine_displacer_stroke=0.035;
Diameter_piston_power=0.060;
Diameter_piston_displacer=0.040;
w_engine=2*pi*1.8; %w=2(pi)(frequency)
theta_engine_rad_step=4000;
theta_engine_rad_step_original=theta_engine_rad_step;
%Power Piston Cam Parameters
rising_period_power=165/360;
falling_period_power=165/360;
dwell_1_power=15/360;
dwell_2_power=15/360;
Roller_follower_radius_power=0.012/2;
c_cam_power=0.15; %distance between follower center and cam center
%Displacer Piston Cam Parameters
rising_period_displacer=(45)/360; % period for rising
falling_period_displacer=(45)/360; % period for falling
dwell_1_displacer=(135)/360; % period for dwell 1
dwell_2_displacer=(135)/360; % period for dwell 2
Roller_follower_radius_displacer=0.012/2;
c_cam_displacer=0.15; %distance between follower center and cam center
dwell_cam_overlap_deg_displacer=15;
T_h = 400+273; %degree K, heater temp.
T_k = 40+273; %degree K, cooler temp.
T_0 = 25+273; %degree K, ambient temp.
r_displacer_drive_rod=0.008/2; %in meter
V_r=(3000)*10^(-6); %Regenerator Dead Volume
V_k=(3000)*10^(-6); %Cooler Dead Volume
V_h=(3000)*10^(-6); %Heater Dead Volume
V_displacer_clearance_hot=30*10^(-8);
V_displacer_clearance_cold=30*10^(-8);
```

```
V_power_clearance=0*10^(-9);
%R_spec_gas=2076.9; %J/g/K*1000=J/kg/K, specific gas constant of helium at
ambient
%density_gas=0.178; %kg/m3
R_spec_gas=287.052874; %J/g/K*1000=J/kg/K, specific gas constant of air at
ambient
density_gas=1.293; %kg/m3; at ambient
%Specific Heat Capacity
%Air
Molar_mass_gas=28.97; %kg/kmol
a_cp=28.11;
b_cp=0.1967*10^(-2);
c_cp=0.4802*10^(-5);
d_cp=-1.966*10^(-9);
T_gas=(T_h+T_k)/2; %in degree K
C_p_gas=(a_cp+b_cp*T_gas+c_cp*T_gas^2+d_cp*T_gas^3); %in kJ/(kmol*K),
ideal gas, from 273K to 1800K, at ambient
C_p_gas=C_p_gas/Molar_mass_gas*1000; %in J/(kg*K)
C_v_gas=C_p_gas-R_spec_gas;
k_isentropic=C_p_gas/C_v_gas; %gamma or k, index of isentropic
%Cam Follower Geometry Data for force analysis
%Power Piston
R_cg2_oscillating_power=0.02;
theta_cg2_oscillating_rad_power=0/180*pi;
m_cg2_oscillating_power=0.5;
I_cg2_oscillating_power=0.006;
R_cg3_oscillating_power=0.02;
theta_cg3_oscillating_rad_power=0/180*pi;
m_cg3_oscillating_power=0.8;
I_cg3_oscillating_power=0.011;
theta_cg4_oscillating_rad_power=0/180*pi;
m_cg4_oscillating_power=0.5;
I_cg4_oscillating_power=0.005;
%Displacer Piston
R_cg2_oscillating_displacer=0.02;
theta_cg2_oscillating_rad_displacer=0/180*pi;
m_cg2_oscillating_displacer=0.5;
I_cg2_oscillating_displacer=0.006;
R_cg3_oscillating_displacer=0.02;
```

```
theta_cg3_oscillating_rad_displacer=0/180*pi;
m_cg3_oscillating_displacer=0.8;
I_cg3_oscillating_displacer=0.011;
theta_cg4_oscillating_rad_displacer=0/180*pi;
m_cg4_oscillating_displacer=0.5;
I_cg4_oscillating_displacer=0.005;
%-----------------------------------------------
%Define/Initialize values for engine kinematics
w_engine=2*pi*1.8; %w=2(pi)(frequency)
w_engine_rpm=w_engine/2/pi*60;
theta_engine_rad_step=4000;
theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
theta_engine_rad=0;
for i=1:(theta_engine_rad_step-1)
theta_engine_rad=[theta_engine_rad
theta_engine_rad(1,i)+theta_engine_rad_step_size];
end
theta_engine_deg=theta_engine_rad/pi*180;
%-------------------------------------------
%Cam-Oscillating Follower Analysis
%Power Piston
%Follower-Slider-crank Geometry Synthesis
engine_stroke=engine_power_stroke;
joint2_y_max_displacement=0.002; %2mm
L_conrod_oscillating_cam=(engine_stroke^2+(joint2_y_max_displacement/2)^2)
^(1/2); %L notation
l_follower_part1=joint2_y_max_displacement^2+(engine_stroke/2)^2;
l_follower_part2=-2*((engine_stroke/2)^2);
l_follower_part3=((engine_stroke/2)^2)-joint2_y_max_displacement^2;
l_follower_part4=(l_follower_part2^2-
4*l_follower_part1*l_follower_part3)^(1/2);
l_follower_part5=(-l_follower_part2-
l_follower_part4)/(2*l_follower_part1);
Follower_angle_stroke_rad=2*acos(l_follower_part5);
Follower_angle_stroke_deg=Follower_angle_stroke_rad/pi*180;
l_follower=joint2_y_max_displacement/(1-
cos(Follower_angle_stroke_rad/2)); %B notation
%Define/Initialize values
h_cam=Follower_angle_stroke_rad;
```

```
cam_velocity_rpm=w_engine_rpm;
cam_velocity_rad=w_engine;
rising_period=rising_period_power;
falling_period=falling_period_power;
rising_steps=round(theta_engine_rad_step*rising_period);
falling_steps=round(theta_engine_rad_step*rising_period);
dwell_1=dwell_1_power;
dwell_2=dwell_2_power;
dwell_period1_size=round(dwell_1*theta_engine_rad_step);
dwell_period2_size=round(dwell_2*theta_engine_rad_step);
if
((rising_steps+falling_steps+dwell_period1_size+dwell_period2_size)>theta_
engine_rad_step)
dwell_period1_size=dwell_period1_size-1;
dwell_period2_size=dwell_period2_size-1;
end
beta1_cam=rising_steps*theta_engine_rad_step_size; % rising duration
beta2_cam=falling_steps*theta_engine_rad_step_size; % falling duration
beta_cam=[beta1_cam beta2_cam];
dwell_period1=dwell_period1_size*theta_engine_rad_step_size;
dwell_period2=dwell_period2_size*theta_engine_rad_step_size;
dwell_1=zeros(1,dwell_period1_size);
dwell_2=zeros(1,dwell_period2_size);
dwell_2_s=h_cam*ones(1,dwell_period2_size);
%Condition : dwell_period1+beta1_cam+dwell_period2+beta2_cam=2*pi
theta_cam=[0 0;0 0];
for k=1:2
if k==1
small_steps=rising_steps;
end
if k==2
small_steps=falling_steps;
end
for i=1:(small_steps-1)
theta_cam(k,i+1)=theta_cam(k,i)+theta_engine_rad_step_size;
end
for j=1:small_steps
%Normalized cam angle value (x_cam) Caiculation
x_cam=theta_cam(k,j)/beta_cam(1,k);
```

```
%Cycloidal Caiculation
s_cam_rising(1,j)=h_cam*(x_cam-1/(2*pi)*sin(2*pi*x_cam));
v_cam_rising(1,j)=(h_cam/beta_cam(1,k))*(1-cos(2*pi*x_cam));
a_cam_rising(1,j)=((h_cam/beta_cam(1,k))^2)*2*pi*sin(2*pi*x_cam);
j_cam_rising(1,j)=((h_cam/beta_cam(1,k))^3)*4*(pi^2)*cos(2*pi*x_cam);
s_cam_falling(1,j)=h_cam-h_cam*(x_cam-1/(2*pi)*sin(2*pi*x_cam));
v_cam_falling(1,j)=-(h_cam/beta_cam(1,k))*(1-cos(2*pi*x_cam));
a_cam_falling(1,j)=-((h_cam/beta_cam(1,k))^2)*2*pi*sin(2*pi*x_cam);
j_cam_falling(1,j)=-((h_cam/beta_cam(1,k))^3)*4*(pi^2)*cos(2*pi*x_cam);
end
end
%Piston Motion pattern (displacer/power)
plot_x_rad=theta_engine_rad;
plot_x=theta_engine_deg;
plot_y_s=[s_cam_falling(1,:) dwell_1 s_cam_rising(1,:) dwell_2_s];
plot_y_v=[v_cam_falling(1,:) dwell_1 v_cam_rising(1,:) dwell_2];
plot_y_a=[a_cam_falling(1,:) dwell_1 a_cam_rising(1,:) dwell_2];
plot_y_j=[j_cam_falling(1,:) dwell_1 j_cam_rising(1,:) dwell_2];
s_cam_real=plot_y_s;
v_cam_real=plot_y_v*cam_velocity_rad;
a_cam_real=plot_y_a*(cam_velocity_rad^2);
j_cam_real=plot_y_j*(cam_velocity_rad^3);
j_cam_power=j_cam_real;
%Cam sizing of oscillating follower
%Initialize cam & oscillating follower geometry
Roller_follower_radius=Roller_follower_radius_power;
c_cam=c_cam_power; %distance between follower center and cam center
delta_initial_deg=-Follower_angle_stroke_deg/2;
delta_initial_rad=delta_initial_deg/180*pi;
delta_follower=delta_initial_rad+s_cam_real;
R_cam=(l_follower^2+c_cam^2-
2*l_follower*c_cam*cos(delta_follower)).^(0.5);
epsilon_cam=asin(c_cam./R_cam.*sin(delta_follower));
psi_cam=acos((c_cam^2+R_cam.^2-l_follower^2)./(2*R_cam*c_cam));
psi_cam_initial=psi_cam(1,1);
gamma_cam=psi_cam_initial-psi_cam+plot_x_rad;
%Pressure Angle Calculation
period_start=1;
period_stop=period_start+falling_steps-1;
```

```
phi_falling_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*
sin(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start
:period_stop));
phi_falling_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_falling=-
pi/2+epsilon_cam(1,period_start:period_stop)+atan(1./(phi_falling_part1+ph
i_falling_part2));
period_start=period_stop+1;
period_stop=period_start+dwell_period1_size-1;
phi_low_dwell=pi/2-epsilon_cam(1,period_start:period_stop);
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
phi_rising_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*s
in(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start:
period_stop));
phi_rising_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_rising=pi/2-
epsilon_cam(1,period_start:period_stop)+atan(1./(phi_rising_part1-
phi_rising_part2));
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
phi_high_dwell=-pi/2+epsilon_cam(1,period_start:period_stop);
phi_cam=[phi_falling phi_low_dwell phi_rising phi_high_dwell];
%Radius of curvature calculation
period_start=1;
period_stop=period_start+falling_steps-1;
rho_pitch_falling_part1=(c_cam^2)*(sin(delta_follower(1,period_start:perio
d_stop))).^2;
rho_pitch_falling_part2=a_cam_real(1,period_start:period_stop).*cos(phi_fa
lling)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:
period_stop)-1).*sin(phi_falling);
rho_pitch_falling_part3=l_follower*cos(phi_falling).*rho_pitch_falling_par
t2;
rho_pitch_falling_part4=c_cam*sin(phi_falling)-rho_pitch_falling_part3;
rho_pitch_falling_part5=cos(phi_falling).*rho_pitch_falling_part4;
```

```
rho_pitch_falling=rho_pitch_falling_part1./rho_pitch_falling_part5;
period_start=period_stop+1;
period_stop=period_start+dwell_period1_size-1;
rho_pitch_low_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
rho_pitch_rising_part1=(c_cam^2)*(sin(delta_follower(1,period_start:period
_stop))).^2;
rho_pitch_rising_part2=a_cam_real(1,period_start:period_stop).*cos(phi_ris
ing)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:pe
riod_stop)-1).*sin(phi_rising);
rho_pitch_rising_part3=l_follower*cos(phi_rising).*rho_pitch_rising_part2;
rho_pitch_rising_part4=c_cam*sin(phi_rising)-rho_pitch_rising_part3;
rho_pitch_rising_part5=cos(phi_rising).*rho_pitch_rising_part4;
rho_pitch_rising=rho_pitch_rising_part1./rho_pitch_rising_part5;
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
rho_pitch_high_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
rho_pitch=[rho_pitch_falling rho_pitch_low_dwell rho_pitch_rising
rho_pitch_high_dwell];
rho_c=rho_pitch-Roller_follower_radius;
%Position Analysis
oscillating_follower_link_a=l_follower;
oscillating_follower_link_b=L_conrod_oscillating_cam;
oscillating_follower_link_c=l_follower-joint2_y_max_displacement/2;
theta2_oscillating=pi/2+(Follower_angle_stroke_rad/2-s_cam_real);
theta2_oscillating_deg=theta2_oscillating/pi*180;
theta3_oscillating=asin((oscillating_follower_link_a*sin(theta2_oscillatin
g)-oscillating_follower_link_c)/oscillating_follower_link_b);
d_oscillating=oscillating_follower_link_a*cos(theta2_oscillating)-
oscillating_follower_link_b*cos(theta3_oscillating);
d_oscillating_normalize=d_oscillating-min(d_oscillating);
%Velocity Analysis
w2_oscillating=-v_cam_real; %clockwise is negative
w3_oscillating=oscillating_follower_link_a/oscillating_follower_link_b;
```

```
w3_oscillating=w3_oscillating*w2_oscillating.*cos(theta2_oscillating)./cos
(theta3_oscillating);
d_dot_oscillating=-
oscillating_follower_link_a*w2_oscillating.*sin(theta2_oscillating);
d_dot_oscillating=d_dot_oscillating+oscillating_follower_link_b*w3_oscilla
ting.*sin(theta3_oscillating);
%Acceleration Analysis
alpha2_oscillating=a_cam_real;
alpha3_oscillating=oscillating_follower_link_a*alpha2_oscillating.*cos(the
ta2_oscillating);
alpha3_oscillating=alpha3_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
alpha3_oscillating=alpha3_oscillating+oscillating_follower_link_b*(w3_osci
llating.^2).*sin(theta3_oscillating);
alpha3_oscillating=alpha3_oscillating./(oscillating_follower_link_b*cos(th
eta3_oscillating));
d_double_dot_oscillating=-
oscillating_follower_link_a*alpha2_oscillating.*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*alpha3_oscillating.*sin(theta3_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*(w3_oscillating.^2).*sin(theta3_oscillating);
x_piston_power=d_oscillating_normalize;
v_piston_power=d_dot_oscillating;
%Displacer Piston
s_cam_rising=[];
v_cam_rising=[];
a_cam_rising=[];
j_cam_rising=[];
s_cam_falling=[];
v_cam_falling=[];
a_cam_falling=[];
j_cam_falling=[];
%Follower-Slider-crank Geometry Synthesis
engine_stroke=engine_displacer_stroke;
joint2_y_max_displacement=0.002; %2mm
```

```
L_conrod_oscillating_cam=(engine_stroke^2+(joint2_y_max_displacement/2)^2)
^(1/2); %L notation
l_follower_part1=joint2_y_max_displacement^2+(engine_stroke/2)^2;
l_follower_part2=-2*((engine_stroke/2)^2);
l_follower_part3=((engine_stroke/2)^2)-joint2_y_max_displacement^2;
l_follower_part4=(l_follower_part2^2-
4*l_follower_part1*l_follower_part3)^(1/2);
l_follower_part5=(-l_follower_part2-
l_follower_part4)/(2*l_follower_part1);
Follower_angle_stroke_rad=2*acos(l_follower_part5);
Follower_angle_stroke_deg=Follower_angle_stroke_rad/pi*180;
l_follower=joint2_y_max_displacement/(1-
cos(Follower_angle_stroke_rad/2)); %B notation
%Define/Initialize values
h_cam=Follower_angle_stroke_rad;
cam_velocity_rpm=w_engine_rpm;
cam_velocity_rad=w_engine;
rising_period=rising_period_displacer; % period for rising
falling_period=falling_period_displacer; % period for falling
rising_steps=round(theta_engine_rad_step*rising_period);
falling_steps=round(theta_engine_rad_step*rising_period);
dwell_1=dwell_1_displacer;
dwell_2=dwell_2_displacer;
dwell_period1_size=round(dwell_1*theta_engine_rad_step);
dwell_period2_size=round(dwell_2*theta_engine_rad_step);
if
((rising_steps+falling_steps+dwell_period1_size+dwell_period2_size)>theta_
engine_rad_step)
dwell_period1_size=dwell_period1_size-1;
dwell_period2_size=dwell_period2_size-1;
end
%theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
beta1_cam=rising_steps*theta_engine_rad_step_size; % rising duration
beta2_cam=falling_steps*theta_engine_rad_step_size; % falling duration
beta_cam=[beta1_cam beta2_cam];
dwell_period1=dwell_period1_size*theta_engine_rad_step_size;
dwell_period2=dwell_period2_size*theta_engine_rad_step_size;
dwell_1=zeros(1,dwell_period1_size);
dwell_2=zeros(1,dwell_period2_size);
```

```
dwell_2_s=h_cam*ones(1,dwell_period2_size);
%Condition : dwell_period1+beta1_cam+dwell_period2+beta2_cam=2*pi
theta_cam=[0 0;0 0];
for k=1:2
if k==1
small_steps=rising_steps;
end
if k==2
small_steps=falling_steps;
end
for i=1:(small_steps-1)
theta_cam(k,i+1)=theta_cam(k,i)+theta_engine_rad_step_size;
end
for j=1:small_steps
%Normalized cam angle value (x_cam) Caiculation
x_cam=theta_cam(k,j)/beta_cam(1,k);
%Cycloidal Caiculation
s_cam_rising(1,j)=h_cam*(x_cam-1/(2*pi)*sin(2*pi*x_cam));
v_cam_rising(1,j)=(h_cam/beta_cam(1,k))*(1-cos(2*pi*x_cam));
a_cam_rising(1,j)=((h_cam/beta_cam(1,k))^2)*2*pi*sin(2*pi*x_cam);
j_cam_rising(1,j)=((h_cam/beta_cam(1,k))^3)*4*(pi^2)*cos(2*pi*x_cam);
s_cam_falling(1,j)=h_cam-h_cam*(x_cam-1/(2*pi)*sin(2*pi*x_cam));
v_cam_falling(1,j)=-(h_cam/beta_cam(1,k))*(1-cos(2*pi*x_cam));
a_cam_falling(1,j)=-((h_cam/beta_cam(1,k))^2)*2*pi*sin(2*pi*x_cam);
j_cam_falling(1,j)=-((h_cam/beta_cam(1,k))^3)*4*(pi^2)*cos(2*pi*x_cam);
end
end
%Piston Motion pattern (displacer/power)
plot_x_rad=theta_engine_rad;
plot_x=theta_engine_deg;
dwell_cam_overlap_deg=dwell_cam_overlap_deg_displacer;
dwell_cam_overlap_step=round(dwell_cam_overlap_deg/360*theta_engine_rad_st
ep);
plot_y_s=[dwell_1 s_cam_rising(1,:) dwell_2_s s_cam_falling(1,:)];
plot_y_v=[dwell_1 v_cam_rising(1,:) dwell_2 v_cam_falling(1,:)];
plot_y_a=[dwell_1 a_cam_rising(1,:) dwell_2 a_cam_falling(1,:)];
plot_y_j=[dwell_1 j_cam_rising(1,:) dwell_2 j_cam_falling(1,:)];
s_cam_real=circshift(plot_y_s,-dwell_cam_overlap_step);
v_cam_real=circshift(plot_y_v*cam_velocity_rad,-dwell_cam_overlap_step);
```

```
a_cam_real=circshift(plot_y_a*(cam_velocity_rad^2),-
dwell_cam_overlap_step);
j_cam_real=circshift(plot_y_j*(cam_velocity_rad^3),-
dwell_cam_overlap_step);
j_cam_displacer=j_cam_real;
%Cam sizing of oscillating follower
%Initialize cam & oscillating follower geometry
Roller_follower_radius=Roller_follower_radius_displacer;
c_cam=c_cam_displacer; %distance between follower center and cam center
delta_initial_deg=-Follower_angle_stroke_deg/2;
delta_initial_rad=delta_initial_deg/180*pi;
delta_follower=delta_initial_rad+s_cam_real;
R_cam=(l_follower^2+c_cam^2-
2*l_follower*c_cam*cos(delta_follower)).^(0.5);
epsilon_cam=asin(c_cam./R_cam.*sin(delta_follower));
psi_cam=acos((c_cam^2+R_cam.^2-l_follower^2)./(2*R_cam*c_cam));
psi_cam_initial=psi_cam(1,1);
gamma_cam=psi_cam_initial-psi_cam+plot_x_rad;
%Pressure Angle Calculation
period_start=1;
period_stop=period_start+dwell_period1_size-1;
phi_low_dwell=pi/2-epsilon_cam(1,period_start:period_stop);
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
phi_rising_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*s
in(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start:
period_stop));
phi_rising_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_rising=pi/2-
epsilon_cam(1,period_start:period_stop)+atan(1./(phi_rising_part1-
phi_rising_part2));
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
phi_high_dwell=-pi/2+epsilon_cam(1,period_start:period_stop);
period_start=period_stop+1;
period_stop=period_start+falling_steps-1;
```

```
phi_falling_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*
sin(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start
:period_stop));
phi_falling_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_falling=-
pi/2+epsilon_cam(1,period_start:period_stop)+atan(1./(phi_falling_part1+ph
i_falling_part2));
phi_cam=[phi_low_dwell phi_rising phi_high_dwell phi_falling];
%Radius of curvature calculation
period_start=1;
period_stop=period_start+dwell_period1_size-1;
rho_pitch_low_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
rho_pitch_rising_part1=(c_cam^2)*(sin(delta_follower(1,period_start:period
_stop))).^2;
rho_pitch_rising_part2=a_cam_real(1,period_start:period_stop).*cos(phi_ris
ing)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:pe
riod_stop)-1).*sin(phi_rising);
rho_pitch_rising_part3=l_follower*cos(phi_rising).*rho_pitch_rising_part2;
rho_pitch_rising_part4=c_cam*sin(phi_rising)-rho_pitch_rising_part3;
rho_pitch_rising_part5=cos(phi_rising).*rho_pitch_rising_part4;
rho_pitch_rising=rho_pitch_rising_part1./rho_pitch_rising_part5;
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
rho_pitch_high_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
period_start=period_stop+1;
period_stop=period_start+falling_steps-1;
rho_pitch_falling_part1=(c_cam^2)*(sin(delta_follower(1,period_start:perio
d_stop))).^2;
rho_pitch_falling_part2=a_cam_real(1,period_start:period_stop).*cos(phi_fa
lling)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:
period_stop)-1).*sin(phi_falling);
```

```
rho_pitch_falling_part3=l_follower*cos(phi_falling).*rho_pitch_falling_par
t2;
rho_pitch_falling_part4=c_cam*sin(phi_falling)-rho_pitch_falling_part3;
rho_pitch_falling_part5=cos(phi_falling).*rho_pitch_falling_part4;
rho_pitch_falling=rho_pitch_falling_part1./rho_pitch_falling_part5;
rho_pitch=[rho_pitch_low_dwell rho_pitch_rising rho_pitch_high_dwell
rho_pitch_falling];
rho_c=rho_pitch-Roller_follower_radius;
%Position Analysis
oscillating_follower_link_a=l_follower;
oscillating_follower_link_b=L_conrod_oscillating_cam;
oscillating_follower_link_c=l_follower-joint2_y_max_displacement/2;
theta2_oscillating=pi/2+(Follower_angle_stroke_rad/2-s_cam_real);
theta2_oscillating_deg=theta2_oscillating/pi*180;
theta3_oscillating=asin((oscillating_follower_link_a*sin(theta2_oscillatin
g)-oscillating_follower_link_c)/oscillating_follower_link_b);
d_oscillating=oscillating_follower_link_a*cos(theta2_oscillating)-
oscillating_follower_link_b*cos(theta3_oscillating);
d_oscillating_normalize=d_oscillating-min(d_oscillating);
%Velocity Analysis
w2_oscillating=-v_cam_real; %clockwise is negative
w3_oscillating=oscillating_follower_link_a/oscillating_follower_link_b;
w3_oscillating=w3_oscillating*w2_oscillating.*cos(theta2_oscillating)./cos
(theta3_oscillating);
d_dot_oscillating=-
oscillating_follower_link_a*w2_oscillating.*sin(theta2_oscillating);
d_dot_oscillating=d_dot_oscillating+oscillating_follower_link_b*w3_oscilla
ting.*sin(theta3_oscillating);
%Acceleration Analysis
alpha2_oscillating=a_cam_real;
alpha3_oscillating=oscillating_follower_link_a*alpha2_oscillating.*cos(the
ta2_oscillating);
alpha3_oscillating=alpha3_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
alpha3_oscillating=alpha3_oscillating+oscillating_follower_link_b*(w3_osci
llating.^2).*sin(theta3_oscillating);
alpha3_oscillating=alpha3_oscillating./(oscillating_follower_link_b*cos(th
eta3_oscillating));
```

```
d_double_dot_oscillating=-
oscillating_follower_link_a*alpha2_oscillating.*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*alpha3_oscillating.*sin(theta3_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*(w3_oscillating.^2).*sin(theta3_oscillating);
x_piston_displacer=d_oscillating_normalize;
v_piston_displacer=d_dot_oscillating;
%Stirling Engine Piston Motion Display
figure
plot(theta_engine_deg,x_piston_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,x_piston_displacer,'r','LineWidth',3)
title('s diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('s (length in meter)')
legend('power piston','displacer')
figure
plot(theta_engine_deg,v_piston_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,v_piston_displacer,'r','LineWidth',3)
title('v diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('v (m/s)')
legend('power piston','displacer')
figure
plot(theta_engine_deg,j_cam_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,j_cam_displacer,'r','LineWidth',3)
title('j diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('j (m/s3)')
legend('power piston','displacer')
%Stirling Engine Characteristics Initialisation
%Define/Initialize values for engine characteristics
T_regenerator=(T_h-T_k)./(log(T_h./T_k));
r_piston_power=Diameter_piston_power/2; %in meter;
```

```
r_piston_displacer=Diameter_piston_displacer/2; %in meter
V_sw_hot=pi*(r_piston_displacer^2)*engine_displacer_stroke;
V_sw_cold=V_sw_hot-pi*(r_displacer_drive_rod^2)*engine_displacer_stroke;
V_sw_power=pi*(r_piston_power^2)*engine_power_stroke;
Total_volume_engine_ambient=V_sw_power+V_power_clearance+V_r+V_k+V_h+V_sw_
cold+V_displacer_clearance_cold;
M_gas_engine_initial=density_gas*Total_volume_engine_ambient; %in kg
%Stirling Engine Piston Volume Calculation
V_hot_displacer=pi*(r_piston_displacer^2)*x_piston_displacer;
V_power=pi*(r_piston_power^2)*x_piston_power;
V_cold_displacer=V_sw_cold-pi*(r_piston_displacer^2-
r_displacer_drive_rod^2)*x_piston_displacer;
dV_hot_displacer=pi*(r_piston_displacer^2)*v_piston_displacer;
V_cold_engine=V_power+V_cold_displacer;
dV_cold_engine=pi*(r_piston_power^2)*v_piston_power-
pi*(r_piston_displacer^2-r_displacer_drive_rod^2)*v_piston_displacer;
%Integrate into Schmidt Model
V_hot=V_h+V_displacer_clearance_hot+V_hot_displacer;
dV_hot=dV_hot_displacer;
V_cold=V_k+V_displacer_clearance_cold+V_power_clearance+V_cold_engine;
dV_cold=dV_cold_engine;
%Integrate into Urieli Model
V_exp=V_displacer_clearance_hot+V_hot_displacer;
dV_exp=dV_hot_displacer;
V_comp=V_displacer_clearance_cold+V_power_clearance+V_cold_engine;
dV_comp=dV_cold_engine;
%Schmidt Model for Initial value prediction
P_engine_Schmidt=M_gas_engine_initial*R_spec_gas./(V_cold/T_k+V_r/T_regene
rator+V_hot/T_h);
m_regenerator_Schmidt=V_r*P_engine_Schmidt*(log(T_h/T_k))/(R_spec_gas*(T_h
-T_k));
Volume_engine_Schmidt=V_hot+V_cold+V_r;
Simpson_multiple=[];
for i=1:((theta_engine_rad_step)/2)
Simpson_multiple=[Simpson_multiple;2;4];
end
Simpson_multiple=[Simpson_multiple;1];
Simpson_multiple(1,1)=1;
Work_engine_part1=P_engine_Schmidt.*(dV_cold+dV_hot);
```

```
Work_engine_part1=[Work_engine_part1 Work_engine_part1(1,1)];
h=2*pi/theta_engine_rad_step;
Work_engine_Schmidt=h/3*Work_engine_part1*Simpson_multiple;
Words = 'Engine Work in J per revolution (Schmidt)';
disp(Words)
disp(Work_engine_Schmidt)
Work_done_Schmidt_power=Work_engine_Schmidt*w_engine_rpm/60;
Words = 'Engine Work in W (Schmidt)';
disp(Words)
disp(Work_done_Schmidt_power)
Work_engine_part2=P_engine_Schmidt.*(dV_hot);
Work_engine_part2=[Work_engine_part2 Work_engine_part2(1,1)];
h=2*pi/theta_engine_rad_step;
Work_engine_Schmidt_hot=h/3*Work_engine_part2*Simpson_multiple;
Q_hot_Schmidt=Work_engine_Schmidt_hot;
Engine_Efficiency_Schmidt=Work_engine_Schmidt/Q_hot_Schmidt;
Words = 'Engine Efficiency (Schmidt)';
disp(Words)
disp(Engine_Efficiency_Schmidt)
%Urieli model - ODEs in adiabatic analysis
%Separate odd and even rows for rk4 method 1/2step calculation
theta_engine_rad_step=theta_engine_rad_step/2;
theta_engine_rad_step_size=theta_engine_rad_step_size*2;
for i=1:theta_engine_rad_step
V_exp_odd(1,i)=V_exp(1,i*2-1);
dV_exp_odd(1,i)=dV_exp(1,i*2-1);
V_comp_odd(1,i)=V_comp(1,i*2-1);
dV_comp_odd(1,i)=dV_comp(1,i*2-1);
V_exp_even(1,i)=V_exp(1,i*2);
dV_exp_even(1,i)=dV_exp(1,i*2);
V_comp_even(1,i)=V_comp(1,i*2);
dV_comp_even(1,i)=dV_comp(1,i*2);
theta_engine_deg_odd(1,i)=theta_engine_deg(1,i*2-1);
end
V_exp=V_exp_odd;
dV_exp=dV_exp_odd;
V_comp=V_comp_odd;
dV_comp=dV_comp_odd;
figure
```

```
plot(theta_engine_deg_odd,V_exp,'r','LineWidth',3)
title('Expansion Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
figure
plot(theta_engine_deg_odd,V_comp,'r','LineWidth',3)
title('Compression Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
figure
plot(theta_engine_deg_odd,dV_exp,'g','LineWidth',3)
hold on
plot(theta_engine_deg_odd,dV_comp,'r','LineWidth',3)
title('Volume Change')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume Change (m3/s)')
%Initialize values
Temperature_term1=V_k/T_k+V_r/T_regenerator+V_h/T_h;
T_comp=ones(1,theta_engine_rad_step+1).*T_k;
T_exp=ones(1,theta_engine_rad_step+1).*T_h;
T_ck=T_comp(1,1);
T_he=T_exp(1,1);
Q_h=zeros(1,theta_engine_rad_step);
Q_k=zeros(1,theta_engine_rad_step);
Q_r=zeros(1,theta_engine_rad_step);
W_exp=zeros(1,theta_engine_rad_step);
W_comp=zeros(1,theta_engine_rad_step);
T_kr=T_k;
T_rh=T_h;
for j=1:50
for i=1:(theta_engine_rad_step)
if i==1
M_gas_engine=M_gas_engine_initial;
end
P_engine_Urieli(1,i)=M_gas_engine*R_spec_gas/(V_comp(1,i)/T_comp(1,i)+Temp
erature_term1+V_exp(1,i)/T_exp(1,i));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli(1,i)*(dV_comp(1,i)/T_ck+dV_exp(1,i)/T_he);
```

```
dP_engine_Urieli_term2=V_comp(1,i)/T_ck+k_isentropic*Temperature_term1+V_e
xp(1,i)/T_he;
dP_engine_Urieli(1,i)=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli(1,i)*V_comp(1,i)/(R_spec_gas*T_comp(1,i));
m_k=P_engine_Urieli(1,i)*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli(1,i)*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli(1,i)*V_h/(R_spec_gas*T_h);
m_exp=P_engine_Urieli(1,i)*V_exp(1,i)/(R_spec_gas*T_exp(1,i));
dm_exp(1,i)=(P_engine_Urieli(1,i)*dV_exp(1,i)+V_exp(1,i)*dP_engine_Urieli(
1,i)/k_isentropic)/(R_spec_gas*T_he);
dm_comp(1,i)=(P_engine_Urieli(1,i)*dV_comp(1,i)+V_comp(1,i)*dP_engine_Urie
li(1,i)/k_isentropic)/(R_spec_gas*T_ck);
dm_k(1,i)=m_k*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
dm_r(1,i)=m_r*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
dm_h(1,i)=m_h*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
m_dot_ck=-dm_comp(1,i);
m_dot_he=dm_exp(1,i);
m_dot_kr=m_dot_ck-dm_k(1,i);
m_dot_rh=m_dot_he+dm_h(1,i);
if m_dot_ck>0
T_ck=T_comp(1,i);
else
T_ck=T_k;
end
if m_dot_he>0
T_he=T_h;
else
T_he=T_exp(1,i);
end
dT_exp=T_exp(1,i)*(dP_engine_Urieli(1,i)/P_engine_Urieli(1,i)+dV_exp(1,i)/
V_exp(1,i)-dm_exp(1,i)/m_exp);
dT_comp=T_comp(1,i)*(dP_engine_Urieli(1,i)/P_engine_Urieli(1,i)+dV_comp(1,
i)/V_comp(1,i)-dm_comp(1,i)/m_comp);
dQ_k=V_k*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_ck*m_dot_ck-
T_kr*m_dot_kr);
dQ_r=V_r*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_kr*m_dot_kr-
T_rh*m_dot_rh);
dQ_h=V_h*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_rh*m_dot_rh-
T_he*m_dot_he);
```

```
dW_exp=P_engine_Urieli(1,i)*dV_exp(1,i);
dW_comp=P_engine_Urieli(1,i)*dV_comp(1,i);
k1_exp=theta_engine_rad_step_size*dT_exp;
k1_comp=theta_engine_rad_step_size*dT_comp;
k1_dQk=theta_engine_rad_step_size*dQ_k;
k1_dQr=theta_engine_rad_step_size*dQ_r;
k1_dQh=theta_engine_rad_step_size*dQ_h;
k1_dWe=theta_engine_rad_step_size*dW_exp;
k1_dWc=theta_engine_rad_step_size*dW_comp;
T_ck_half=T_ck;
T_he_half=T_he;
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp_even(1,i)/(T_comp(1,i
)+0.5*k1_comp)+Temperature_term1+V_exp_even(1,i)/(T_exp(1,i)+0.5*k1_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp_even(1,i)/T_ck_half+dV_exp_even
(1,i)/T_he_half);
dP_engine_Urieli_term2=V_comp_even(1,i)/T_ck_half+k_isentropic*Temperature
_term1+V_exp_even(1,i)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp_even(1,i)/(R_spec_gas*(T_comp(1,i)+0.5*
k1_comp));
m_exp=P_engine_Urieli_half*V_exp_even(1,i)/(R_spec_gas*(T_exp(1,i)+0.5*k1_
exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp_even(1,i)+V_exp_even(1,i)*dP_engi
ne_Urieli_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp_even(1,i)+V_comp_even(1,i)*dP_e
ngine_Urieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+0.5*k1_comp;
```

```
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+0.5*k1_exp;
end
dT_exp=(T_exp(1,i)+0.5*k1_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half
+dV_exp_even(1,i)/V_exp_even(1,i)-dm_exp_half/m_exp);
k2_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+0.5*k1_comp)*(dP_engine_Urieli_half/P_engine_Urieli_h
alf+dV_comp_even(1,i)/V_comp_even(1,i)-dm_comp_half/m_comp);
k2_comp=theta_engine_rad_step_size*dT_comp;
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp_even(1,i);
dW_comp=P_engine_Urieli_half*dV_comp_even(1,i);
k2_dQk=theta_engine_rad_step_size*dQ_k;
k2_dQr=theta_engine_rad_step_size*dQ_r;
k2_dQh=theta_engine_rad_step_size*dQ_h;
k2_dWe=theta_engine_rad_step_size*dW_exp;
k2_dWc=theta_engine_rad_step_size*dW_comp;
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp_even(1,i)/(T_comp(1,i
)+0.5*k2_comp)+Temperature_term1+V_exp_even(1,i)/(T_exp(1,i)+0.5*k2_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp_even(1,i)/T_ck_half+dV_exp_even
(1,i)/T_he_half);
dP_engine_Urieli_term2=V_comp_even(1,i)/T_ck_half+k_isentropic*Temperature
_term1+V_exp_even(1,i)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp_even(1,i)/(R_spec_gas*(T_comp(1,i)+0.5*
k2_comp));
m_exp=P_engine_Urieli_half*V_exp_even(1,i)/(R_spec_gas*(T_exp(1,i)+0.5*k2_
exp));
```

```
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp_even(1,i)+V_exp_even(1,i)*dP_engi
ne_Urieli_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp_even(1,i)+V_comp_even(1,i)*dP_e
ngine_Urieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+0.5*k2_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+0.5*k2_exp;
end
dT_exp=(T_exp(1,i)+0.5*k2_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half
+dV_exp_even(1,i)/V_exp_even(1,i)-dm_exp_half/m_exp);
k3_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+0.5*k2_comp)*(dP_engine_Urieli_half/P_engine_Urieli_h
alf+dV_comp_even(1,i)/V_comp_even(1,i)-dm_comp_half/m_comp);
k3_comp=theta_engine_rad_step_size*dT_comp;
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp_even(1,i);
dW_comp=P_engine_Urieli_half*dV_comp_even(1,i);
k3_dQk=theta_engine_rad_step_size*dQ_k;
```

```
k3_dQr=theta_engine_rad_step_size*dQ_r;
k3_dQh=theta_engine_rad_step_size*dQ_h;
k3_dWe=theta_engine_rad_step_size*dW_exp;
k3_dWc=theta_engine_rad_step_size*dW_comp;
i2=i+1;
if i2>theta_engine_rad_step
i2=i2-theta_engine_rad_step;
end
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp(1,i2)/(T_comp(1,i)+k3
_comp)+Temperature_term1+V_exp(1,i2)/(T_exp(1,i)+k3_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp(1,i2)/T_ck_half+dV_exp(1,i2)/T_
he_half);
dP_engine_Urieli_term2=V_comp(1,i2)/T_ck_half+k_isentropic*Temperature_ter
m1+V_exp(1,i2)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp(1,i2)/(R_spec_gas*(T_comp(1,i)+k3_comp)
);
m_exp=P_engine_Urieli_half*V_exp(1,i2)/(R_spec_gas*(T_exp(1,i)+k3_exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp(1,i2)+V_exp(1,i2)*dP_engine_Uriel
i_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp(1,i2)+V_comp(1,i2)*dP_engine_Ur
ieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+k3_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
```

```
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+k3_exp;
end
dT_exp=(T_exp(1,i)+k3_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half+dV_
exp(1,i2)/V_exp(1,i2)-dm_exp_half/m_exp);
k4_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+k3_comp)*(dP_engine_Urieli_half/P_engine_Urieli_half+
dV_comp(1,i2)/V_comp(1,i2)-dm_comp_half/m_comp);
k4_comp=theta_engine_rad_step_size*dT_comp;
T_exp(1,i+1)=T_exp(1,i)+1/6*(k1_exp+2*k2_exp+2*k3_exp+k4_exp);
T_comp(1,i+1)=T_comp(1,i)+1/6*(k1_comp+2*k2_comp+2*k3_comp+k4_comp);
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp(1,i2);
dW_comp=P_engine_Urieli_half*dV_comp(1,i2);
k4_dQk=theta_engine_rad_step_size*dQ_k;
k4_dQr=theta_engine_rad_step_size*dQ_r;
k4_dQh=theta_engine_rad_step_size*dQ_h;
k4_dWe=theta_engine_rad_step_size*dW_exp;
k4_dWc=theta_engine_rad_step_size*dW_comp;
Q_k(1,i+1)=Q_k(1,i)+1/6*(k1_dQk+2*k2_dQk+2*k3_dQk+k4_dQk);
Q_r(1,i+1)=Q_r(1,i)+1/6*(k1_dQr+2*k2_dQr+2*k3_dQr+k4_dQr);
Q_h(1,i+1)=Q_h(1,i)+1/6*(k1_dQh+2*k2_dQh+2*k3_dQh+k4_dQh);
W_exp(1,i+1)=W_exp(1,i)+1/6*(k1_dWe+2*k2_dWe+2*k3_dWe+k4_dWe);
W_comp(1,i+1)=W_comp(1,i)+1/6*(k1_dWc+2*k2_dWc+2*k3_dWc+k4_dWc);
if i==theta_engine_rad_step
T_exp_diff(1,j)=T_exp(1,theta_engine_rad_step+1)-T_exp(1,1);
T_exp(1,1)=T_exp(1,theta_engine_rad_step+1);
T_exp(:,theta_engine_rad_step+1)=[];
T_comp_diff(1,j)=T_comp(1,theta_engine_rad_step+1)-T_comp(1,1);
T_comp(1,1)=T_comp(1,theta_engine_rad_step+1);
T_comp(:,theta_engine_rad_step+1)=[];
end
end
```

```
end
Work_done_Urieli=W_exp(1,theta_engine_rad_step+1)+W_comp(1,theta_engine_ra
d_step+1);
Engine_Efficiency_Urieli=Work_done_Urieli/Q_h(1,theta_engine_rad_step+1);
Words = 'Engine Work in J per revolution (Urieli)';
disp(Words)
disp(Work_done_Urieli)
Work_done_Urieli_power=Work_done_Urieli*w_engine_rpm/60;
Words = 'Engine Work in W (Urieli)';
disp(Words)
disp(Work_done_Urieli_power)
Words = 'Engine Efficiency Urieli';
disp(Words)
disp(Engine_Efficiency_Urieli)
% PV Diagram Display:
Volume_engine_Urieli=V_comp+V_exp+V_r+V_h+V_k;
figure
plot(Volume_engine_Schmidt,P_engine_Schmidt,'g','LineWidth',3)
hold on
plot(Volume_engine_Urieli,P_engine_Urieli,'r','LineWidth',3)
title('PV Diagram')
xlabel('Volume')
ylabel('Pressure')
legend('Isothermal','Adiabatic')
figure
plot(theta_engine_deg,P_engine_Schmidt,'g','LineWidth',3)
hold on
plot(theta_engine_deg_odd,P_engine_Urieli,'r','LineWidth',3)
title('Engine Pressure')
xlabel('x, shaft angle 0 to 360deg')
ylabel('P (Pressure in Pa)')
legend('Isothermal','Adiabatic')
figure
plot(theta_engine_deg,Volume_engine_Schmidt,'r','LineWidth',3)
%hold on
%plot(theta_engine_deg_odd,Volume_engine_Urieli,'r','LineWidth',3)
title('Engine Internal Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
```

```
%-----------------------
%Power Piston
theta_engine_rad_step=theta_engine_rad_step_original;
theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
theta_engine_rad=0;
for i=1:(theta_engine_rad_step-1)
theta_engine_rad=[theta_engine_rad
theta_engine_rad(1,i)+theta_engine_rad_step_size];
end
theta_engine_deg=theta_engine_rad/pi*180;
%Follower-Slider-crank Geometry Synthesis
engine_stroke=engine_power_stroke;
joint2_y_max_displacement=0.002; %2mm
L_conrod_oscillating_cam=(engine_stroke^2+(joint2_y_max_displacement/2)^2)
^(1/2); %L notation
l_follower_part1=joint2_y_max_displacement^2+(engine_stroke/2)^2;
l_follower_part2=-2*((engine_stroke/2)^2);
l_follower_part3=((engine_stroke/2)^2)-joint2_y_max_displacement^2;
l_follower_part4=(l_follower_part2^2-
4*l_follower_part1*l_follower_part3)^(1/2);
l_follower_part5=(-l_follower_part2-
l_follower_part4)/(2*l_follower_part1);
Follower_angle_stroke_rad=2*acos(l_follower_part5);
Follower_angle_stroke_deg=Follower_angle_stroke_rad/pi*180;
l_follower=joint2_y_max_displacement/(1-
cos(Follower_angle_stroke_rad/2)); %B notation
%Define/Initialize values
h_cam=Follower_angle_stroke_rad;
cam_velocity_rpm=w_engine_rpm;
cam_velocity_rad=w_engine;
rising_period=rising_period_power;
falling_period=falling_period_power;
rising_steps=round(theta_engine_rad_step*rising_period);
falling_steps=round(theta_engine_rad_step*rising_period);
dwell_1=dwell_1_power;
dwell_2=dwell_2_power;
dwell_period1_size=round(dwell_1*theta_engine_rad_step);
dwell_period2_size=round(dwell_2*theta_engine_rad_step);
```

```
if
((rising_steps+falling_steps+dwell_period1_size+dwell_period2_size)>theta_
engine_rad_step)
dwell_period1_size=dwell_period1_size-1;
dwell_period2_size=dwell_period2_size-1;
end
%theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
beta1_cam=rising_steps*theta_engine_rad_step_size; % rising duration
beta2_cam=falling_steps*theta_engine_rad_step_size; % falling duration
beta_cam=[beta1_cam beta2_cam];
dwell_period1=dwell_period1_size*theta_engine_rad_step_size;
dwell_period2=dwell_period2_size*theta_engine_rad_step_size;
dwell_1=zeros(1,dwell_period1_size);
dwell_2=zeros(1,dwell_period2_size);
dwell_2_s=h_cam*ones(1,dwell_period2_size);
%Condition : dwell_period1+beta1_cam+dwell_period2+beta2_cam=2*pi
theta_cam=[0 0;0 0];
for k=1:2
if k==1
small_steps=rising_steps;
end
if k==2
small_steps=falling_steps;
end
for i=1:(small_steps-1)
theta_cam(k,i+1)=theta_cam(k,i)+theta_engine_rad_step_size;
end
for j=1:small_steps
%Normalized cam angle value (x_cam) Caiculation
x_cam=theta_cam(k,j)/beta_cam(1,k);
%Cycloidal Caiculation
s_cam_rising(1,j)=h_cam*(x_cam-1/(2*pi)*sin(2*pi*x_cam));
v_cam_rising(1,j)=(h_cam/beta_cam(1,k))*(1-cos(2*pi*x_cam));
a_cam_rising(1,j)=((h_cam/beta_cam(1,k))^2)*2*pi*sin(2*pi*x_cam);
j_cam_rising(1,j)=((h_cam/beta_cam(1,k))^3)*4*(pi^2)*cos(2*pi*x_cam);
s_cam_falling(1,j)=h_cam-h_cam*(x_cam-1/(2*pi)*sin(2*pi*x_cam));
v_cam_falling(1,j)=-(h_cam/beta_cam(1,k))*(1-cos(2*pi*x_cam));
a_cam_falling(1,j)=-((h_cam/beta_cam(1,k))^2)*2*pi*sin(2*pi*x_cam);
j_cam_falling(1,j)=-((h_cam/beta_cam(1,k))^3)*4*(pi^2)*cos(2*pi*x_cam);
```

```
end
end
%Piston Motion pattern (displacer/power)
plot_x_rad=theta_engine_rad;
plot_x=theta_engine_deg;
plot_y_s=[s_cam_falling(1,:) dwell_1 s_cam_rising(1,:) dwell_2_s];
plot_y_v=[v_cam_falling(1,:) dwell_1 v_cam_rising(1,:) dwell_2];
plot_y_a=[a_cam_falling(1,:) dwell_1 a_cam_rising(1,:) dwell_2];
plot_y_j=[j_cam_falling(1,:) dwell_1 j_cam_rising(1,:) dwell_2];
s_cam_real=plot_y_s;
v_cam_real=plot_y_v*cam_velocity_rad;
a_cam_real=plot_y_a*(cam_velocity_rad^2);
j_cam_real=plot_y_j*(cam_velocity_rad^3);
%Cam sizing of oscillating follower
%Initialize cam & oscillating follower geometry
Roller_follower_radius=Roller_follower_radius_power;
c_cam=c_cam_power; %distance between follower center and cam center
delta_initial_deg=-Follower_angle_stroke_deg/2;
delta_initial_rad=delta_initial_deg/180*pi;
delta_follower=delta_initial_rad+s_cam_real;
R_cam=(l_follower^2+c_cam^2-
2*l_follower*c_cam*cos(delta_follower)).^(0.5);
epsilon_cam=asin(c_cam./R_cam.*sin(delta_follower));
psi_cam=acos((c_cam^2+R_cam.^2-l_follower^2)./(2*R_cam*c_cam));
psi_cam_initial=psi_cam(1,1);
gamma_cam=psi_cam_initial-psi_cam+plot_x_rad;
%Pressure Angle Calculation
period_start=1;
period_stop=period_start+falling_steps-1;
phi_falling_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*
sin(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start
:period_stop));
phi_falling_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_falling=-
pi/2+epsilon_cam(1,period_start:period_stop)+atan(1./(phi_falling_part1+ph
i_falling_part2));
period_start=period_stop+1;
```

```
period_stop=period_start+dwell_period1_size-1;
phi_low_dwell=pi/2-epsilon_cam(1,period_start:period_stop);
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
phi_rising_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*s
in(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start:
period_stop));
phi_rising_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_rising=pi/2-
epsilon_cam(1,period_start:period_stop)+atan(1./(phi_rising_part1-
phi_rising_part2));
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
phi_high_dwell=-pi/2+epsilon_cam(1,period_start:period_stop);
phi_cam=[phi_falling phi_low_dwell phi_rising phi_high_dwell];
%Radius of curvature calculation
period_start=1;
period_stop=period_start+falling_steps-1;
rho_pitch_falling_part1=(c_cam^2)*(sin(delta_follower(1,period_start:perio
d_stop))).^2;
rho_pitch_falling_part2=a_cam_real(1,period_start:period_stop).*cos(phi_fa
lling)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:
period_stop)-1).*sin(phi_falling);
rho_pitch_falling_part3=l_follower*cos(phi_falling).*rho_pitch_falling_par
t2;
rho_pitch_falling_part4=c_cam*sin(phi_falling)-rho_pitch_falling_part3;
rho_pitch_falling_part5=cos(phi_falling).*rho_pitch_falling_part4;
rho_pitch_falling=rho_pitch_falling_part1./rho_pitch_falling_part5;
period_start=period_stop+1;
period_stop=period_start+dwell_period1_size-1;
rho_pitch_low_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
rho_pitch_rising_part1=(c_cam^2)*(sin(delta_follower(1,period_start:period
_stop))).^2;
```

```
rho_pitch_rising_part2=a_cam_real(1,period_start:period_stop).*cos(phi_ris
ing)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:pe
riod_stop)-1).*sin(phi_rising);
rho_pitch_rising_part3=l_follower*cos(phi_rising).*rho_pitch_rising_part2;
rho_pitch_rising_part4=c_cam*sin(phi_rising)-rho_pitch_rising_part3;
rho_pitch_rising_part5=cos(phi_rising).*rho_pitch_rising_part4;
rho_pitch_rising=rho_pitch_rising_part1./rho_pitch_rising_part5;
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
rho_pitch_high_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
rho_pitch=[rho_pitch_falling rho_pitch_low_dwell rho_pitch_rising
rho_pitch_high_dwell];
rho_c=rho_pitch-Roller_follower_radius;
%Position Analysis
oscillating_follower_link_a=l_follower;
oscillating_follower_link_b=L_conrod_oscillating_cam;
oscillating_follower_link_c=l_follower-joint2_y_max_displacement/2;
theta2_oscillating=pi/2+(Follower_angle_stroke_rad/2-s_cam_real);
theta2_oscillating_deg=theta2_oscillating/pi*180;
theta3_oscillating=asin((oscillating_follower_link_a*sin(theta2_oscillatin
g)-oscillating_follower_link_c)/oscillating_follower_link_b);
d_oscillating=oscillating_follower_link_a*cos(theta2_oscillating)-
oscillating_follower_link_b*cos(theta3_oscillating);
d_oscillating_normalize=d_oscillating-min(d_oscillating);
%Velocity Analysis
w2_oscillating=-v_cam_real; %clockwise is negative
w3_oscillating=oscillating_follower_link_a/oscillating_follower_link_b;
w3_oscillating=w3_oscillating*w2_oscillating.*cos(theta2_oscillating)./cos
(theta3_oscillating);
d_dot_oscillating=-
oscillating_follower_link_a*w2_oscillating.*sin(theta2_oscillating);
d_dot_oscillating=d_dot_oscillating+oscillating_follower_link_b*w3_oscilla
ting.*sin(theta3_oscillating);
%Acceleration Analysis
alpha2_oscillating=a_cam_real;
alpha3_oscillating=oscillating_follower_link_a*alpha2_oscillating.*cos(the
ta2_oscillating);
```

```
alpha3_oscillating=alpha3_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
alpha3_oscillating=alpha3_oscillating+oscillating_follower_link_b*(w3_osci
llating.^2).*sin(theta3_oscillating);
alpha3_oscillating=alpha3_oscillating./(oscillating_follower_link_b*cos(th
eta3_oscillating));
d_double_dot_oscillating=-
oscillating_follower_link_a*alpha2_oscillating.*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*alpha3_oscillating.*sin(theta3_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*(w3_oscillating.^2).*sin(theta3_oscillating);
%Force Analysis
R_cg2_oscillating=R_cg2_oscillating_power;
theta_cg2_oscillating_rad=theta_cg2_oscillating_rad_power;
m_cg2_oscillating=m_cg2_oscillating_power;
I_cg2_oscillating=I_cg2_oscillating_power;
R_cg3_oscillating=R_cg3_oscillating_power;
theta_cg3_oscillating_rad=theta_cg3_oscillating_rad_power;
m_cg3_oscillating=m_cg3_oscillating_power;
I_cg3_oscillating=I_cg3_oscillating_power;
theta_cg4_oscillating_rad=theta_cg4_oscillating_rad_power;
R_cg4_oscillating=oscillating_follower_link_c/cos(theta_cg4_oscillating_ra
d);
m_cg4_oscillating=m_cg4_oscillating_power;
I_cg4_oscillating=I_cg4_oscillating_power;
A_x_cg2_oscillating=R_cg2_oscillating*alpha2_oscillating.*(-
sin(theta2_oscillating+theta_cg2_oscillating_rad))-
R_cg2_oscillating*(w2_oscillating.^2).*cos(theta2_oscillating+theta_cg2_os
cillating_rad);
A_y_cg2_oscillating=R_cg2_oscillating*alpha2_oscillating.*(cos(theta2_osci
llating+theta_cg2_oscillating_rad))-
R_cg2_oscillating*(w2_oscillating.^2).*sin(theta2_oscillating+theta_cg2_os
cillating_rad);
A_x_cg3_J3_oscillating=R_cg3_oscillating*alpha3_oscillating.*(-
sin(theta3_oscillating+theta_cg3_oscillating_rad))-
```

```
R_cg3_oscillating*(w3_oscillating.^2).*cos(theta3_oscillating+theta_cg3_os
cillating_rad);
A_y_cg3_J3_oscillating=R_cg3_oscillating*alpha3_oscillating.*(cos(theta3_o
scillating+theta_cg3_oscillating_rad))-
R_cg3_oscillating*(w3_oscillating.^2).*sin(theta3_oscillating+theta_cg3_os
cillating_rad);
A_x_J2_oscillating=-
l_follower*alpha2_oscillating.*sin(theta2_oscillating)-
l_follower*(w2_oscillating.^2).*cos(theta2_oscillating);
A_y_J2_oscillating=l_follower*alpha2_oscillating.*cos(theta2_oscillating)-
l_follower*(w2_oscillating.^2).*sin(theta2_oscillating);
A_x_J3_oscillating=d_double_dot_oscillating;
A_y_J3_oscillating=0;
A_x_cg3_oscillating=A_x_J3_oscillating+A_x_cg3_J3_oscillating;
A_y_cg3_oscillating=A_y_J3_oscillating+A_y_cg3_J3_oscillating;
%Link 2 cg_joint position + Oscillating cam exerted force on follower arm
direction
theta_cg2_J1_oscillating=theta2_oscillating+theta_cg2_oscillating_rad;
theta_J1_cg2_oscillating=theta_cg2_J1_oscillating-pi;
position_J2_cg2_x=R_cg2_oscillating*sin(theta_J1_cg2_oscillating)+l_follow
er*sin(theta2_oscillating);
position_J2_cg2_y=R_cg2_oscillating*cos(theta_J1_cg2_oscillating)+l_follow
er*cos(theta2_oscillating);
%Link 3 cg_joint position
theta_cg3_J3_oscillating=theta3_oscillating+theta_cg3_oscillating_rad;
theta_J3_cg3_oscillating=theta_cg3_J3_oscillating-pi;
position_J2_cg3_x=R_cg3_oscillating*sin(theta_J3_cg3_oscillating)+L_conrod
_oscillating_cam*sin(theta3_oscillating);
position_J2_cg3_y=R_cg3_oscillating*cos(theta_J3_cg3_oscillating)+L_conrod
_oscillating_cam*cos(theta3_oscillating);
%Link 4 cg_joint position
theta_cg4_J4_oscillating=pi/2+theta_cg4_oscillating_rad;
theta_J4_cg4_oscillating=theta_cg4_J4_oscillating-pi;
position_J3_cg4_x=R_cg4_oscillating*sin(theta_J4_cg4_oscillating)+oscillat
ing_follower_link_c*sin(pi/2);
position_J3_cg4_y=R_cg4_oscillating*cos(theta_J4_cg4_oscillating)+oscillat
ing_follower_link_c*cos(pi/2);
%Matrix Construction
F_p_x_oscillating=-P_engine_Schmidt*pi*((Diameter_piston_power)^2)/4;
```

```
F_p_y_oscillating=0;
R_12y=R_cg2_oscillating*cos(theta_J1_cg2_oscillating);
R_12x=R_cg2_oscillating*sin(theta_J1_cg2_oscillating);
R_32y=position_J2_cg2_y;
R_32x=position_J2_cg2_x;
R_23y=position_J2_cg3_x;
R_23x=position_J2_cg3_y;
R_43y=R_cg3_oscillating*cos(theta_J3_cg3_oscillating);
R_43x=R_cg3_oscillating*sin(theta_J3_cg3_oscillating);
R_34y=position_J3_cg4_y;
R_34x=position_J3_cg4_x;
R_14y=R_cg4_oscillating*cos(theta_J4_cg4_oscillating);
R_14x=R_cg4_oscillating*sin(theta_J4_cg4_oscillating);
alpha4_oscillating=0;
A_x_cg4_oscillating=d_double_dot_oscillating;
A_y_cg4_oscillating=0;
for i=1:theta_engine_rad_step
sign_oscillating=1;
if d_dot_oscillating(1,i)<0
sign_oscillating=-1;
else
sign_oscillating=1;
end
friction_coefficient_oscillating=0;
Matrix1_oscillating=[1 0 1 0 0 0 0 0;0 1 0 1 0 0 0 0;-R_12y(1,i)
R_12x(1,i) -R_32y(1,i) R_32x(1,i) 0 0 0 1;0 0 -1 0 1 0 0 0;0 0 0 -1 0 1 0
0];
Matrix1_oscillating=[Matrix1_oscillating;0 0 R_23y(1,i) -R_23x(1,i) -
R_43y(1,i) R_43x(1,i) 0 0;0 0 0 0 -1 0
sign_oscillating*friction_coefficient_oscillating 0;0 0 0 0 0 -1 1 0];
Matrix2_oscillating=[m_cg2_oscillating*A_x_cg2_oscillating(1,i);m_cg2_osci
llating*A_y_cg2_oscillating(1,i)];
Matrix2_oscillating=[Matrix2_oscillating;I_cg2_oscillating*alpha2_oscillat
ing(1,i)];
Matrix2_oscillating=[Matrix2_oscillating;m_cg3_oscillating*A_x_cg3_oscilla
ting(1,i);m_cg3_oscillating*A_y_cg3_oscillating(1,i);I_cg3_oscillating*alp
ha3_oscillating(1,i)];
Matrix2_oscillating=[Matrix2_oscillating;m_cg4_oscillating*A_x_cg4_oscilla
ting(1,i)-F_p_x_oscillating(1,i);-F_p_y_oscillating];
```

```
Matrix3_oscillating(:,i)=inv(Matrix1_oscillating)*Matrix2_oscillating;
T_12_oscillating=Matrix3_oscillating(8,:);
end
T_12_oscillating_power=T_12_oscillating;
R_cam_power=R_cam;
delta_initial_rad_power=delta_initial_rad;
%Displacer Piston
s_cam_rising=[];
v_cam_rising=[];
a_cam_rising=[];
j_cam_rising=[];
s_cam_falling=[];
v_cam_falling=[];
a_cam_falling=[];
j_cam_falling=[];
%Follower-Slider-crank Geometry Synthesis
engine_stroke=engine_displacer_stroke;
joint2_y_max_displacement=0.002; %2mm
L_conrod_oscillating_cam=(engine_stroke^2+(joint2_y_max_displacement/2)^2)
^(1/2); %L notation
l_follower_part1=joint2_y_max_displacement^2+(engine_stroke/2)^2;
l_follower_part2=-2*((engine_stroke/2)^2);
l_follower_part3=((engine_stroke/2)^2)-joint2_y_max_displacement^2;
l_follower_part4=(l_follower_part2^2-
4*l_follower_part1*l_follower_part3)^(1/2);
l_follower_part5=(-l_follower_part2-
l_follower_part4)/(2*l_follower_part1);
Follower_angle_stroke_rad=2*acos(l_follower_part5);
Follower_angle_stroke_deg=Follower_angle_stroke_rad/pi*180;
l_follower=joint2_y_max_displacement/(1-
cos(Follower_angle_stroke_rad/2)); %B notation
%Define/Initialize values
h_cam=Follower_angle_stroke_rad;
cam_velocity_rpm=w_engine_rpm;
cam_velocity_rad=w_engine;
rising_period=rising_period_displacer; % period for rising
falling_period=falling_period_displacer; % period for falling
rising_steps=round(theta_engine_rad_step*rising_period);
falling_steps=round(theta_engine_rad_step*rising_period);
```

```
dwell_1=dwell_1_displacer;
dwell_2=dwell_2_displacer;
dwell_period1_size=round(dwell_1*theta_engine_rad_step);
dwell_period2_size=round(dwell_2*theta_engine_rad_step);
if
((rising_steps+falling_steps+dwell_period1_size+dwell_period2_size)>theta_
engine_rad_step)
dwell_period1_size=dwell_period1_size-1;
dwell_period2_size=dwell_period2_size-1;
end
beta1_cam=rising_steps*theta_engine_rad_step_size; % rising duration
beta2_cam=falling_steps*theta_engine_rad_step_size; % falling duration
beta_cam=[beta1_cam beta2_cam];
dwell_period1=dwell_period1_size*theta_engine_rad_step_size;
dwell_period2=dwell_period2_size*theta_engine_rad_step_size;
dwell_1=zeros(1,dwell_period1_size);
dwell_2=zeros(1,dwell_period2_size);
dwell_2_s=h_cam*ones(1,dwell_period2_size);
%Condition : dwell_period1+beta1_cam+dwell_period2+beta2_cam=2*pi
theta_cam=[0 0;0 0];
for k=1:2
if k==1
small_steps=rising_steps;
end
if k==2
small_steps=falling_steps;
end
for i=1:(small_steps-1)
theta_cam(k,i+1)=theta_cam(k,i)+theta_engine_rad_step_size;
end
for j=1:small_steps
%Normalized cam angle value (x_cam) Caiculation
x_cam=theta_cam(k,j)/beta_cam(1,k);
%Cycloidal Caiculation
s_cam_rising(1,j)=h_cam*(x_cam-1/(2*pi)*sin(2*pi*x_cam));
v_cam_rising(1,j)=(h_cam/beta_cam(1,k))*(1-cos(2*pi*x_cam));
a_cam_rising(1,j)=((h_cam/beta_cam(1,k))^2)*2*pi*sin(2*pi*x_cam);
j_cam_rising(1,j)=((h_cam/beta_cam(1,k))^3)*4*(pi^2)*cos(2*pi*x_cam);
s_cam_falling(1,j)=h_cam-h_cam*(x_cam-1/(2*pi)*sin(2*pi*x_cam));
```

```
v_cam_falling(1,j)=-(h_cam/beta_cam(1,k))*(1-cos(2*pi*x_cam));
a_cam_falling(1,j)=-((h_cam/beta_cam(1,k))^2)*2*pi*sin(2*pi*x_cam);
j_cam_falling(1,j)=-((h_cam/beta_cam(1,k))^3)*4*(pi^2)*cos(2*pi*x_cam);
end
end
%Piston Motion pattern (displacer/power)
plot_x_rad=theta_engine_rad;
plot_x=theta_engine_deg;
dwell_cam_overlap_deg=dwell_cam_overlap_deg_displacer;
dwell_cam_overlap_step=round(dwell_cam_overlap_deg/360*theta_engine_rad_st
ep);
plot_y_s=[dwell_1 s_cam_rising(1,:) dwell_2_s s_cam_falling(1,:)];
plot_y_v=[dwell_1 v_cam_rising(1,:) dwell_2 v_cam_falling(1,:)];
plot_y_a=[dwell_1 a_cam_rising(1,:) dwell_2 a_cam_falling(1,:)];
plot_y_j=[dwell_1 j_cam_rising(1,:) dwell_2 j_cam_falling(1,:)];
s_cam_real=circshift(plot_y_s,-dwell_cam_overlap_step);
v_cam_real=circshift(plot_y_v*cam_velocity_rad,-dwell_cam_overlap_step);
a_cam_real=circshift(plot_y_a*(cam_velocity_rad^2),-
dwell_cam_overlap_step);
j_cam_real=circshift(plot_y_j*(cam_velocity_rad^3),-
dwell_cam_overlap_step);
%Cam sizing of oscillating follower
%Initialize cam & oscillating follower geometry
Roller_follower_radius=Roller_follower_radius_displacer;
c_cam=c_cam_displacer; %distance between follower center and cam center
delta_initial_deg=-Follower_angle_stroke_deg/2;
delta_initial_rad=delta_initial_deg/180*pi;
delta_follower=delta_initial_rad+s_cam_real;
R_cam=(l_follower^2+c_cam^2-
2*l_follower*c_cam*cos(delta_follower)).^(0.5);
epsilon_cam=asin(c_cam./R_cam.*sin(delta_follower));
psi_cam=acos((c_cam^2+R_cam.^2-l_follower^2)./(2*R_cam*c_cam));
psi_cam_initial=psi_cam(1,1);
gamma_cam=psi_cam_initial-psi_cam+plot_x_rad;
%Pressure Angle Calculation
period_start=1;
period_stop=period_start+dwell_period1_size-1;
phi_low_dwell=pi/2-epsilon_cam(1,period_start:period_stop);
period_start=period_stop+1;
```

```
period_stop=period_start+rising_steps-1;
phi_rising_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*s
in(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start:
period_stop));
phi_rising_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_rising=pi/2-
epsilon_cam(1,period_start:period_stop)+atan(1./(phi_rising_part1-
phi_rising_part2));
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
phi_high_dwell=-pi/2+epsilon_cam(1,period_start:period_stop);
period_start=period_stop+1;
period_stop=period_start+falling_steps-1;
phi_falling_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*
sin(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start
:period_stop));
phi_falling_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_falling=-
pi/2+epsilon_cam(1,period_start:period_stop)+atan(1./(phi_falling_part1+ph
i_falling_part2));
phi_cam=[phi_low_dwell phi_rising phi_high_dwell phi_falling];
%Radius of curvature calculation
period_start=1;
period_stop=period_start+dwell_period1_size-1;
rho_pitch_low_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
rho_pitch_rising_part1=(c_cam^2)*(sin(delta_follower(1,period_start:period
_stop))).^2;
rho_pitch_rising_part2=a_cam_real(1,period_start:period_stop).*cos(phi_ris
ing)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:pe
riod_stop)-1).*sin(phi_rising);
rho_pitch_rising_part3=l_follower*cos(phi_rising).*rho_pitch_rising_part2;
```

```
rho_pitch_rising_part4=c_cam*sin(phi_rising)-rho_pitch_rising_part3;
rho_pitch_rising_part5=cos(phi_rising).*rho_pitch_rising_part4;
rho_pitch_rising=rho_pitch_rising_part1./rho_pitch_rising_part5;
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
rho_pitch_high_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
period_start=period_stop+1;
period_stop=period_start+falling_steps-1;
rho_pitch_falling_part1=(c_cam^2)*(sin(delta_follower(1,period_start:perio
d_stop))).^2;
rho_pitch_falling_part2=a_cam_real(1,period_start:period_stop).*cos(phi_fa
lling)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:
period_stop)-1).*sin(phi_falling);
rho_pitch_falling_part3=l_follower*cos(phi_falling).*rho_pitch_falling_par
t2;
rho_pitch_falling_part4=c_cam*sin(phi_falling)-rho_pitch_falling_part3;
rho_pitch_falling_part5=cos(phi_falling).*rho_pitch_falling_part4;
rho_pitch_falling=rho_pitch_falling_part1./rho_pitch_falling_part5;
rho_pitch=[rho_pitch_low_dwell rho_pitch_rising rho_pitch_high_dwell
rho_pitch_falling];
rho_c=rho_pitch-Roller_follower_radius;
%Position Analysis
oscillating_follower_link_a=l_follower;
oscillating_follower_link_b=L_conrod_oscillating_cam;
oscillating_follower_link_c=l_follower-joint2_y_max_displacement/2;
theta2_oscillating=pi/2+(Follower_angle_stroke_rad/2-s_cam_real);
theta2_oscillating_deg=theta2_oscillating/pi*180;
theta3_oscillating=asin((oscillating_follower_link_a*sin(theta2_oscillatin
g)-oscillating_follower_link_c)/oscillating_follower_link_b);
d_oscillating=oscillating_follower_link_a*cos(theta2_oscillating)-
oscillating_follower_link_b*cos(theta3_oscillating);
d_oscillating_normalize=d_oscillating-min(d_oscillating);
%Velocity Analysis
w2_oscillating=-v_cam_real; %clockwise is negative
w3_oscillating=oscillating_follower_link_a/oscillating_follower_link_b;
w3_oscillating=w3_oscillating*w2_oscillating.*cos(theta2_oscillating)./cos
(theta3_oscillating);
```

```
d_dot_oscillating=-
oscillating_follower_link_a*w2_oscillating.*sin(theta2_oscillating);
d_dot_oscillating=d_dot_oscillating+oscillating_follower_link_b*w3_oscilla
ting.*sin(theta3_oscillating);
%Acceleration Analysis
alpha2_oscillating=a_cam_real;
alpha3_oscillating=oscillating_follower_link_a*alpha2_oscillating.*cos(the
ta2_oscillating);
alpha3_oscillating=alpha3_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
alpha3_oscillating=alpha3_oscillating+oscillating_follower_link_b*(w3_osci
llating.^2).*sin(theta3_oscillating);
alpha3_oscillating=alpha3_oscillating./(oscillating_follower_link_b*cos(th
eta3_oscillating));
d_double_dot_oscillating=-
oscillating_follower_link_a*alpha2_oscillating.*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*alpha3_oscillating.*sin(theta3_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*(w3_oscillating.^2).*sin(theta3_oscillating);
%Force Analysis
R_cg2_oscillating=R_cg2_oscillating_displacer;
theta_cg2_oscillating_rad=theta_cg2_oscillating_rad_displacer;
m_cg2_oscillating=m_cg2_oscillating_displacer;
I_cg2_oscillating=I_cg2_oscillating_displacer;
R_cg3_oscillating=R_cg3_oscillating_displacer;
theta_cg3_oscillating_rad=theta_cg3_oscillating_rad_displacer;
m_cg3_oscillating=m_cg3_oscillating_displacer;
I_cg3_oscillating=I_cg3_oscillating_displacer;
theta_cg4_oscillating_rad=theta_cg4_oscillating_rad_displacer;
R_cg4_oscillating=oscillating_follower_link_c/cos(theta_cg4_oscillating_ra
d);
m_cg4_oscillating=m_cg4_oscillating_displacer;
I_cg4_oscillating=I_cg4_oscillating_displacer;
A_x_cg2_oscillating=R_cg2_oscillating*alpha2_oscillating.*(-
sin(theta2_oscillating+theta_cg2_oscillating_rad))-
```

```
R_cg2_oscillating*(w2_oscillating.^2).*cos(theta2_oscillating+theta_cg2_os
cillating_rad);
A_y_cg2_oscillating=R_cg2_oscillating*alpha2_oscillating.*(cos(theta2_osci
llating+theta_cg2_oscillating_rad))-
R_cg2_oscillating*(w2_oscillating.^2).*sin(theta2_oscillating+theta_cg2_os
cillating_rad);
A_x_cg3_J3_oscillating=R_cg3_oscillating*alpha3_oscillating.*(-
sin(theta3_oscillating+theta_cg3_oscillating_rad))-
R_cg3_oscillating*(w3_oscillating.^2).*cos(theta3_oscillating+theta_cg3_os
cillating_rad);
A_y_cg3_J3_oscillating=R_cg3_oscillating*alpha3_oscillating.*(cos(theta3_o
scillating+theta_cg3_oscillating_rad))-
R_cg3_oscillating*(w3_oscillating.^2).*sin(theta3_oscillating+theta_cg3_os
cillating_rad);
A_x_J2_oscillating=-
l_follower*alpha2_oscillating.*sin(theta2_oscillating)-
l_follower*(w2_oscillating.^2).*cos(theta2_oscillating);
A_y_J2_oscillating=l_follower*alpha2_oscillating.*cos(theta2_oscillating)-
l_follower*(w2_oscillating.^2).*sin(theta2_oscillating);
A_x_J3_oscillating=d_double_dot_oscillating;
A_y_J3_oscillating=0;
A_x_cg3_oscillating=A_x_J3_oscillating+A_x_cg3_J3_oscillating;
A_y_cg3_oscillating=A_y_J3_oscillating+A_y_cg3_J3_oscillating;
%Link 2 cg_joint position + Oscillating cam exerted force on follower arm
direction
theta_cg2_J1_oscillating=theta2_oscillating+theta_cg2_oscillating_rad;
theta_J1_cg2_oscillating=theta_cg2_J1_oscillating-pi;
position_J2_cg2_x=R_cg2_oscillating*sin(theta_J1_cg2_oscillating)+l_follow
er*sin(theta2_oscillating);
position_J2_cg2_y=R_cg2_oscillating*cos(theta_J1_cg2_oscillating)+l_follow
er*cos(theta2_oscillating);
%Link 3 cg_joint position
theta_cg3_J3_oscillating=theta3_oscillating+theta_cg3_oscillating_rad;
theta_J3_cg3_oscillating=theta_cg3_J3_oscillating-pi;
position_J2_cg3_x=R_cg3_oscillating*sin(theta_J3_cg3_oscillating)+L_conrod
_oscillating_cam*sin(theta3_oscillating);
position_J2_cg3_y=R_cg3_oscillating*cos(theta_J3_cg3_oscillating)+L_conrod
_oscillating_cam*cos(theta3_oscillating);
%Link 4 cg_joint position
```

```
theta_cg4_J4_oscillating=pi/2+theta_cg4_oscillating_rad;
theta_J4_cg4_oscillating=theta_cg4_J4_oscillating-pi;
position_J3_cg4_x=R_cg4_oscillating*sin(theta_J4_cg4_oscillating)+oscillat
ing_follower_link_c*sin(pi/2);
position_J3_cg4_y=R_cg4_oscillating*cos(theta_J4_cg4_oscillating)+oscillat
ing_follower_link_c*cos(pi/2);
%Matrix Construction
F_p_x_oscillating=-P_engine_Schmidt*pi*((Diameter_piston_displacer)^2)/4;
F_p_y_oscillating=0;
R_12y=R_cg2_oscillating*cos(theta_J1_cg2_oscillating);
R_12x=R_cg2_oscillating*sin(theta_J1_cg2_oscillating);
R_32y=position_J2_cg2_y;
R_32x=position_J2_cg2_x;
R_23y=position_J2_cg3_x;
R_23x=position_J2_cg3_y;
R_43y=R_cg3_oscillating*cos(theta_J3_cg3_oscillating);
R_43x=R_cg3_oscillating*sin(theta_J3_cg3_oscillating);
R_34y=position_J3_cg4_y;
R_34x=position_J3_cg4_x;
R_14y=R_cg4_oscillating*cos(theta_J4_cg4_oscillating);
R_14x=R_cg4_oscillating*sin(theta_J4_cg4_oscillating);
alpha4_oscillating=0;
A_x_cg4_oscillating=d_double_dot_oscillating;
A_y_cg4_oscillating=0;
for i=1:theta_engine_rad_step
sign_oscillating=1;
if d_dot_oscillating(1,i)<0
sign_oscillating=-1;
else
sign_oscillating=1;
end
friction_coefficient_oscillating=0;
Matrix1_oscillating=[1 0 1 0 0 0 0 0;0 1 0 1 0 0 0 0;-R_12y(1,i)
R_12x(1,i) -R_32y(1,i) R_32x(1,i) 0 0 0 1;0 0 -1 0 1 0 0 0;0 0 0 -1 0 1 0
0];
Matrix1_oscillating=[Matrix1_oscillating;0 0 R_23y(1,i) -R_23x(1,i) -
R_43y(1,i) R_43x(1,i) 0 0;0 0 0 0 -1 0
sign_oscillating*friction_coefficient_oscillating 0;0 0 0 0 0 -1 1 0];
```

```
Matrix2_oscillating=[m_cg2_oscillating*A_x_cg2_oscillating(1,i);m_cg2_osci
llating*A_y_cg2_oscillating(1,i)];
Matrix2_oscillating=[Matrix2_oscillating;I_cg2_oscillating*alpha2_oscillat
ing(1,i)];
Matrix2_oscillating=[Matrix2_oscillating;m_cg3_oscillating*A_x_cg3_oscilla
ting(1,i);m_cg3_oscillating*A_y_cg3_oscillating(1,i);I_cg3_oscillating*alp
ha3_oscillating(1,i)];
Matrix2_oscillating=[Matrix2_oscillating;m_cg4_oscillating*A_x_cg4_oscilla
ting(1,i)-F_p_x_oscillating(1,i);-F_p_y_oscillating];
Matrix3_oscillating(:,i)=inv(Matrix1_oscillating)*Matrix2_oscillating;
T_12_oscillating=Matrix3_oscillating(8,:);
end
T_12_oscillating_displacer=T_12_oscillating;
R_cam_displacer=R_cam;
delta_initial_rad_displacer=delta_initial_rad;
figure
plot(plot_x,(T_12_oscillating_power),'r','LineWidth',3)
title('Power Piston Gas Torque diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('T (gas torque in Nm)')
figure
plot(plot_x,(T_12_oscillating_displacer),'r','LineWidth',3)
title('Displacer Gas Torque diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('T (gas torque in Nm)')
figure
plot(plot_x,(T_12_oscillating_power+T_12_oscillating_displacer),'r','LineW
idth',3)
title('Gas Torque diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('T (gas torque in Nm)')
% Oscillating Cam profile power
oscillating_cam_profile_power_x=R_cam_power.*cos(theta_engine_rad+delta_in
itial_rad_power);
oscillating_cam_profile_power_y=R_cam_power.*sin(theta_engine_rad+delta_in
itial_rad_power);
oscillating_cam_profile_displacer_x=R_cam_displacer.*cos(theta_engine_rad+
delta_initial_rad_displacer);
```

```
oscillating_cam_profile_displacer_y=R_cam_displacer.*sin(theta_engine_rad+
delta_initial_rad_displacer);
figure
plot(oscillating_cam_profile_power_x,oscillating_cam_profile_power_y,'r','
LineWidth',3)
title('oscillating cam profile power')
xlabel('x')
ylabel('y')
figure
plot(oscillating_cam_profile_displacer_x,oscillating_cam_profile_displacer
_y,'r','LineWidth',3)
title('oscillating cam profile displacer')
xlabel('x')
ylabel('y')
```

APPENDIX C: Computer Simulation Program 2b - Matlab Code for Stirling engine with sinusoidal piston motion simulation

```
%Clear all the previous data
clear all
close all
clc
%User Defined Parameters
engine_power_stroke=0.035;
engine_displacer_stroke=0.035;
Diameter_piston_power=0.060;
Diameter_piston_displacer=0.040;
w_engine=2*pi*1.8; %w=2(pi)(frequency)
theta_engine_rad_step=4000;
```

```
theta_engine_rad_step_original=theta_engine_rad_step;
L_conrod=0.15; %Slider-crank conrod length
T_h = 400+273; %degree K, heater temp.
T_k = 40+273; %degree K, cooler temp.
T_0 = 25+273; %degree K, ambient temp.
r_displacer_drive_rod=0.008/2; %in meter
V_r=(3000)*10^(-6); %Regenerator Dead Volume
V_k=(3000)*10^(-6); %Cooler Dead Volume
V_h=(3000)*10^(-6); %Heater Dead Volume
V_displacer_clearance_hot=30*10^(-8);
V_displacer_clearance_cold=30*10^(-8);
V_power_clearance=0*10^(-9);
%R_spec_gas=2076.9; %J/g/K*1000=J/kg/K, specific gas constant of helium at
ambient
%density_gas=0.178; %kg/m3
R_spec_gas=287.052874; %J/g/K*1000=J/kg/K, specific gas constant of air at
ambient
density_gas=1.293; %kg/m3; at ambient
%Specific Heat Capacity
%Air
Molar_mass_gas=28.97; %kg/kmol
a_cp=28.11;
b_cp=0.1967*10^(-2);
c_cp=0.4802*10^(-5);
d_cp=-1.966*10^(-9);
T_gas=(T_h+T_k)/2; %in degree K
C_p_gas=(a_cp+b_cp*T_gas+c_cp*T_gas^2+d_cp*T_gas^3); %in kJ/(kmol*K),
ideal gas, from 273K to 1800K, at ambient
C_p_gas=C_p_gas/Molar_mass_gas*1000; %in J/(kg*K)
C_v_gas=C_p_gas-R_spec_gas;
k_isentropic=C_p_gas/C_v_gas; %gamma or k, index of isentropic
%-------------------------------------------------------------
%Define/Initialize values for engine kinematics
w_engine_rpm=w_engine/2/pi*60;
theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
theta_engine_rad=0;
for i=1:(theta_engine_rad_step-1)
theta_engine_rad=[theta_engine_rad
theta_engine_rad(1,i)+theta_engine_rad_step_size];
```

```
end
theta_engine_deg=theta_engine_rad/pi*180;
%Crank slider Analysis
%Power Piston Crank Slider
%Define/Initialize values
engine_stroke=engine_power_stroke;
r_crank=engine_stroke/2;
w_crank=w_engine;
theta_crank_rad=theta_engine_rad;
% Piston Position:-
x_piston_crank=r_crank*cos(theta_crank_rad)+L_conrod*((1-
(r_crank/L_conrod*sin(theta_crank_rad)).^2).^(1/2));
%Assume steady state analysis where w_crank is constant:-
v_piston_crank=-
r_crank*w_crank.*(sin(theta_crank_rad)+r_crank/2/L_conrod*sin(2*theta_cran
k_rad)./((1-(r_crank/L_conrod*sin(theta_crank_rad)).^2).^(1/2)));
a_piston_crank=-r_crank*(w_crank.^2).*(cos(theta_crank_rad)-
r_crank*(L_conrod^2*(1-2*(cos(theta_crank_rad).^2))-
r_crank^2*(sin(theta_crank_rad)).^4)./(L_conrod^2-
(r_crank*sin(theta_crank_rad)).^2).^(3/2));
%Integrate into Stirling Engine Power Piston Configuration
x_piston_power=x_piston_crank-min(x_piston_crank);
v_piston_power=v_piston_crank/w_engine; %length/rad
%Displacer Piston Crank Slider
engine_stroke=engine_displacer_stroke;
r_crank=engine_stroke/2;
% Piston Position:-
x_piston_crank=r_crank*cos(theta_crank_rad)+L_conrod*((1-
(r_crank/L_conrod*sin(theta_crank_rad)).^2).^(1/2));
%Assume steady state analysis where w_crank is constant:-
v_piston_crank=-
r_crank*w_crank.*(sin(theta_crank_rad)+r_crank/2/L_conrod*sin(2*theta_cran
k_rad)./((1-(r_crank/L_conrod*sin(theta_crank_rad)).^2).^(1/2)));
a_piston_crank=-r_crank*(w_crank.^2).*(cos(theta_crank_rad)-
r_crank*(L_conrod^2*(1-2*(cos(theta_crank_rad).^2))-
r_crank^2*(sin(theta_crank_rad)).^4)./(L_conrod^2-
(r_crank*sin(theta_crank_rad)).^2).^(3/2));
%Integrate into Stirling Engine Displacer Piston Configuration
```

```
x_piston_crank=[x_piston_crank((1/4*theta_engine_rad_step+1):theta_engine_
rad_step) x_piston_crank(1:(1/4*theta_engine_rad_step))];
x_piston_displacer=x_piston_crank-min(x_piston_crank);
v_piston_crank=[v_piston_crank((1/4*theta_engine_rad_step+1):theta_engine_
rad_step) v_piston_crank(1:(1/4*theta_engine_rad_step))];
v_piston_displacer=v_piston_crank/w_engine; %length/rad
%Stirling Engine Piston Motion Display
figure
plot(theta_engine_deg,x_piston_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,x_piston_displacer,'r','LineWidth',3)
title('s diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('s (length in meter)')
legend('power piston','displacer')
figure
plot(theta_engine_deg,v_piston_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,v_piston_displacer,'r','LineWidth',3)
title('v diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('v (m/s)')
legend('power piston','displacer')
%Stirling Engine Characteristics Initialisation
%Define/Initialize values for engine characteristics
T_regenerator=(T_h-T_k)./(log(T_h./T_k));
r_piston_power=Diameter_piston_power/2; %in meter;
r_piston_displacer=Diameter_piston_displacer/2; %in meter
V_sw_hot=pi*(r_piston_displacer^2)*engine_displacer_stroke;
V_sw_cold=V_sw_hot-pi*(r_displacer_drive_rod^2)*engine_displacer_stroke;
V_sw_power=pi*(r_piston_power^2)*engine_power_stroke;
Total_volume_engine_ambient=V_sw_power+V_power_clearance+V_r+V_k+V_h+V_sw_
cold+V_displacer_clearance_cold;
M_gas_engine_initial=density_gas*Total_volume_engine_ambient; %in kg
%Stirling Engine Piston Volume Calculation
V_hot_displacer=pi*(r_piston_displacer^2)*x_piston_displacer;
V_power=pi*(r_piston_power^2)*x_piston_power;
V_cold_displacer=V_sw_cold-pi*(r_piston_displacer^2-
r_displacer_drive_rod^2)*x_piston_displacer;
```

```
dV_hot_displacer=pi*(r_piston_displacer^2)*v_piston_displacer;
V_cold_engine=V_power+V_cold_displacer;
dV_cold_engine=pi*(r_piston_power^2)*v_piston_power-
pi*(r_piston_displacer^2-r_displacer_drive_rod^2)*v_piston_displacer;
%Integrate into Schmidt Model
V_hot=V_h+V_displacer_clearance_hot+V_hot_displacer;
dV_hot=dV_hot_displacer;
V_cold=V_k+V_displacer_clearance_cold+V_power_clearance+V_cold_engine;
dV_cold=dV_cold_engine;
%Integrate into Urieli Model
V_exp=V_displacer_clearance_hot+V_hot_displacer;
dV_exp=dV_hot_displacer;
V_comp=V_displacer_clearance_cold+V_power_clearance+V_cold_engine;
dV_comp=dV_cold_engine;
%Schmidt Model
P_engine_Schmidt=M_gas_engine_initial*R_spec_gas./(V_cold/T_k+V_r/T_regene
rator+V_hot/T_h);
m_regenerator_Schmidt=V_r*P_engine_Schmidt*(log(T_h/T_k))/(R_spec_gas*(T_h
-T_k));
%Note : W_engine=Integral of (P*(d_V_c/d_theta+d_V_e/d_theta)*d_theta)
Volume_engine_Schmidt=V_hot+V_cold+V_r;
Simpson_multiple=[];
for i=1:((theta_engine_rad_step)/2)
Simpson_multiple=[Simpson_multiple;2;4];
end
Simpson_multiple=[Simpson_multiple;1];
Simpson_multiple(1,1)=1;
Work_engine_part1=P_engine_Schmidt.*(dV_cold+dV_hot);
Work_engine_part1=[Work_engine_part1 Work_engine_part1(1,1)];
h=2*pi/theta_engine_rad_step;
Work_engine_Schmidt=h/3*Work_engine_part1*Simpson_multiple;
Words = 'Engine Work in J per revolution (Schmidt)';
disp(Words)
disp(Work_engine_Schmidt)
Work_done_Schmidt_power=Work_engine_Schmidt*w_engine_rpm/60;
Words = 'Engine Work in W (Schmidt)';
disp(Words)
disp(Work_done_Schmidt_power)
Work_engine_part2=P_engine_Schmidt.*(dV_hot);
```

```
Work_engine_part2=[Work_engine_part2 Work_engine_part2(1,1)];
h=2*pi/theta_engine_rad_step;
Work_engine_Schmidt_hot=h/3*Work_engine_part2*Simpson_multiple;
Q_hot_Schmidt=Work_engine_Schmidt_hot;
Engine_Efficiency_Schmidt=Work_engine_Schmidt/Q_hot_Schmidt;
Words = 'Engine Efficiency (Schmidt)';
disp(Words)
disp(Engine_Efficiency_Schmidt)
%Urieli model - ODEs in adiabatic analysis
%Separate odd and even rows for rk4 method 1/2step calculation
theta_engine_rad_step=theta_engine_rad_step/2;
theta_engine_rad_step_size=theta_engine_rad_step_size*2;
for i=1:theta_engine_rad_step
V_exp_odd(1,i)=V_exp(1,i*2-1);
dV_exp_odd(1,i)=dV_exp(1,i*2-1);
V_comp_odd(1,i)=V_comp(1,i*2-1);
dV_comp_odd(1,i)=dV_comp(1,i*2-1);
V_exp_even(1,i)=V_exp(1,i*2);
dV_exp_even(1,i)=dV_exp(1,i*2);
V_comp_even(1,i)=V_comp(1,i*2);
dV_comp_even(1,i)=dV_comp(1,i*2);
theta_engine_deg_odd(1,i)=theta_engine_deg(1,i*2-1);
end
V_exp=V_exp_odd;
dV_exp=dV_exp_odd;
V_comp=V_comp_odd;
dV_comp=dV_comp_odd;
figure
plot(theta_engine_deg_odd,V_exp,'r','LineWidth',3)
title('Expansion Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
figure
plot(theta_engine_deg_odd,V_comp,'r','LineWidth',3)
title('Compression Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
figure
plot(theta_engine_deg_odd,dV_exp,'g','LineWidth',3)
```

```
hold on
plot(theta_engine_deg_odd,dV_comp,'r','LineWidth',3)
title('Volume Change')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume Change (m3/s)')
%Initialize values
Temperature_term1=V_k/T_k+V_r/T_regenerator+V_h/T_h;
T_comp=ones(1,theta_engine_rad_step+1).*T_k;
T_exp=ones(1,theta_engine_rad_step+1).*T_h;
T_ck=T_comp(1,1);
T_he=T_exp(1,1);
Q_h=zeros(1,theta_engine_rad_step);
Q_k=zeros(1,theta_engine_rad_step);
Q_r=zeros(1,theta_engine_rad_step);
W_exp=zeros(1,theta_engine_rad_step);
W_comp=zeros(1,theta_engine_rad_step);
T_kr=T_k;
T_rh=T_h;
for j=1:50
for i=1:(theta_engine_rad_step)
if i==1
M_gas_engine=M_gas_engine_initial;
end
P_engine_Urieli(1,i)=M_gas_engine*R_spec_gas/(V_comp(1,i)/T_comp(1,i)+Temp
erature_term1+V_exp(1,i)/T_exp(1,i));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli(1,i)*(dV_comp(1,i)/T_ck+dV_exp(1,i)/T_he);
dP_engine_Urieli_term2=V_comp(1,i)/T_ck+k_isentropic*Temperature_term1+V_e
xp(1,i)/T_he;
dP_engine_Urieli(1,i)=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli(1,i)*V_comp(1,i)/(R_spec_gas*T_comp(1,i));
m_k=P_engine_Urieli(1,i)*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli(1,i)*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli(1,i)*V_h/(R_spec_gas*T_h);
m_exp=P_engine_Urieli(1,i)*V_exp(1,i)/(R_spec_gas*T_exp(1,i));
dm_exp(1,i)=(P_engine_Urieli(1,i)*dV_exp(1,i)+V_exp(1,i)*dP_engine_Urieli(
1,i)/k_isentropic)/(R_spec_gas*T_he);
dm_comp(1,i)=(P_engine_Urieli(1,i)*dV_comp(1,i)+V_comp(1,i)*dP_engine_Urie
li(1,i)/k_isentropic)/(R_spec_gas*T_ck);
```

```
dm_k(1,i)=m_k*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
dm_r(1,i)=m_r*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
dm_h(1,i)=m_h*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
m_dot_ck=-dm_comp(1,i);
m_dot_he=dm_exp(1,i);
m_dot_kr=m_dot_ck-dm_k(1,i);
m_dot_rh=m_dot_he+dm_h(1,i);
if m_dot_ck>0
T_ck=T_comp(1,i);
else
T_ck=T_k;
end
if m_dot_he>0
T_he=T_h;
else
T_he=T_exp(1,i);
end
dT_exp=T_exp(1,i)*(dP_engine_Urieli(1,i)/P_engine_Urieli(1,i)+dV_exp(1,i)/
V_exp(1,i)-dm_exp(1,i)/m_exp);
dT_comp=T_comp(1,i)*(dP_engine_Urieli(1,i)/P_engine_Urieli(1,i)+dV_comp(1,
i)/V_comp(1,i)-dm_comp(1,i)/m_comp);
dQ_k=V_k*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_ck*m_dot_ck-
T_kr*m_dot_kr);
dQ_r=V_r*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_kr*m_dot_kr-
T_rh*m_dot_rh);
dQ_h=V_h*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_rh*m_dot_rh-
T_he*m_dot_he);
dW_exp=P_engine_Urieli(1,i)*dV_exp(1,i);
dW_comp=P_engine_Urieli(1,i)*dV_comp(1,i);
k1_exp=theta_engine_rad_step_size*dT_exp;
k1_comp=theta_engine_rad_step_size*dT_comp;
k1_dQk=theta_engine_rad_step_size*dQ_k;
k1_dQr=theta_engine_rad_step_size*dQ_r;
k1_dQh=theta_engine_rad_step_size*dQ_h;
k1_dWe=theta_engine_rad_step_size*dW_exp;
k1_dWc=theta_engine_rad_step_size*dW_comp;
T_ck_half=T_ck;
T_he_half=T_he;
```

```
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp_even(1,i)/(T_comp(1,i
)+0.5*k1_comp)+Temperature_term1+V_exp_even(1,i)/(T_exp(1,i)+0.5*k1_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp_even(1,i)/T_ck_half+dV_exp_even
(1,i)/T_he_half);
dP_engine_Urieli_term2=V_comp_even(1,i)/T_ck_half+k_isentropic*Temperature
_term1+V_exp_even(1,i)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp_even(1,i)/(R_spec_gas*(T_comp(1,i)+0.5*
k1_comp));
m_exp=P_engine_Urieli_half*V_exp_even(1,i)/(R_spec_gas*(T_exp(1,i)+0.5*k1_
exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp_even(1,i)+V_exp_even(1,i)*dP_engi
ne_Urieli_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp_even(1,i)+V_comp_even(1,i)*dP_e
ngine_Urieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+0.5*k1_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+0.5*k1_exp;
end
dT_exp=(T_exp(1,i)+0.5*k1_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half
+dV_exp_even(1,i)/V_exp_even(1,i)-dm_exp_half/m_exp);
k2_exp=theta_engine_rad_step_size*dT_exp;
```

```
dT_comp=(T_comp(1,i)+0.5*k1_comp)*(dP_engine_Urieli_half/P_engine_Urieli_h
alf+dV_comp_even(1,i)/V_comp_even(1,i)-dm_comp_half/m_comp);
k2_comp=theta_engine_rad_step_size*dT_comp;
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp_even(1,i);
dW_comp=P_engine_Urieli_half*dV_comp_even(1,i);
k2_dQk=theta_engine_rad_step_size*dQ_k;
k2_dQr=theta_engine_rad_step_size*dQ_r;
k2_dQh=theta_engine_rad_step_size*dQ_h;
k2_dWe=theta_engine_rad_step_size*dW_exp;
k2_dWc=theta_engine_rad_step_size*dW_comp;
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp_even(1,i)/(T_comp(1,i
)+0.5*k2_comp)+Temperature_term1+V_exp_even(1,i)/(T_exp(1,i)+0.5*k2_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp_even(1,i)/T_ck_half+dV_exp_even
(1,i)/T_he_half);
dP_engine_Urieli_term2=V_comp_even(1,i)/T_ck_half+k_isentropic*Temperature
_term1+V_exp_even(1,i)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp_even(1,i)/(R_spec_gas*(T_comp(1,i)+0.5*
k2_comp));
m_exp=P_engine_Urieli_half*V_exp_even(1,i)/(R_spec_gas*(T_exp(1,i)+0.5*k2_
exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp_even(1,i)+V_exp_even(1,i)*dP_engi
ne_Urieli_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp_even(1,i)+V_comp_even(1,i)*dP_e
ngine_Urieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
```

```
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+0.5*k2_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+0.5*k2_exp;
end
dT_exp=(T_exp(1,i)+0.5*k2_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half
+dV_exp_even(1,i)/V_exp_even(1,i)-dm_exp_half/m_exp);
k3_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+0.5*k2_comp)*(dP_engine_Urieli_half/P_engine_Urieli_h
alf+dV_comp_even(1,i)/V_comp_even(1,i)-dm_comp_half/m_comp);
k3_comp=theta_engine_rad_step_size*dT_comp;
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp_even(1,i);
dW_comp=P_engine_Urieli_half*dV_comp_even(1,i);
k3_dQk=theta_engine_rad_step_size*dQ_k;
k3_dQr=theta_engine_rad_step_size*dQ_r;
k3_dQh=theta_engine_rad_step_size*dQ_h;
k3_dWe=theta_engine_rad_step_size*dW_exp;
k3_dWc=theta_engine_rad_step_size*dW_comp;
i2=i+1;
if i2>theta_engine_rad_step
i2=i2-theta_engine_rad_step;
end
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp(1,i2)/(T_comp(1,i)+k3
_comp)+Temperature_term1+V_exp(1,i2)/(T_exp(1,i)+k3_exp));
```

```
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp(1,i2)/T_ck_half+dV_exp(1,i2)/T_
he_half);
dP_engine_Urieli_term2=V_comp(1,i2)/T_ck_half+k_isentropic*Temperature_ter
m1+V_exp(1,i2)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp(1,i2)/(R_spec_gas*(T_comp(1,i)+k3_comp)
);
m_exp=P_engine_Urieli_half*V_exp(1,i2)/(R_spec_gas*(T_exp(1,i)+k3_exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp(1,i2)+V_exp(1,i2)*dP_engine_Uriel
i_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp(1,i2)+V_comp(1,i2)*dP_engine_Ur
ieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+k3_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+k3_exp;
end
dT_exp=(T_exp(1,i)+k3_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half+dV_
exp(1,i2)/V_exp(1,i2)-dm_exp_half/m_exp);
k4_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+k3_comp)*(dP_engine_Urieli_half/P_engine_Urieli_half+
dV_comp(1,i2)/V_comp(1,i2)-dm_comp_half/m_comp);
k4_comp=theta_engine_rad_step_size*dT_comp;
```

```
T_exp(1,i+1)=T_exp(1,i)+1/6*(k1_exp+2*k2_exp+2*k3_exp+k4_exp);
T_comp(1,i+1)=T_comp(1,i)+1/6*(k1_comp+2*k2_comp+2*k3_comp+k4_comp);
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp(1,i2);
dW_comp=P_engine_Urieli_half*dV_comp(1,i2);
k4_dQk=theta_engine_rad_step_size*dQ_k;
k4_dQr=theta_engine_rad_step_size*dQ_r;
k4_dQh=theta_engine_rad_step_size*dQ_h;
k4_dWe=theta_engine_rad_step_size*dW_exp;
k4_dWc=theta_engine_rad_step_size*dW_comp;
Q_k(1,i+1)=Q_k(1,i)+1/6*(k1_dQk+2*k2_dQk+2*k3_dQk+k4_dQk);
Q_r(1,i+1)=Q_r(1,i)+1/6*(k1_dQr+2*k2_dQr+2*k3_dQr+k4_dQr);
Q_h(1,i+1)=Q_h(1,i)+1/6*(k1_dQh+2*k2_dQh+2*k3_dQh+k4_dQh);
W_exp(1,i+1)=W_exp(1,i)+1/6*(k1_dWe+2*k2_dWe+2*k3_dWe+k4_dWe);
W_comp(1,i+1)=W_comp(1,i)+1/6*(k1_dWc+2*k2_dWc+2*k3_dWc+k4_dWc);
if i==theta_engine_rad_step
T_exp_diff(1,j)=T_exp(1,theta_engine_rad_step+1)-T_exp(1,1);
T_exp(1,1)=T_exp(1,theta_engine_rad_step+1);
T_exp(:,theta_engine_rad_step+1)=[];
T_comp_diff(1,j)=T_comp(1,theta_engine_rad_step+1)-T_comp(1,1);
T_comp(1,1)=T_comp(1,theta_engine_rad_step+1);
T_comp(:,theta_engine_rad_step+1)=[];
%Q_k(:,theta_engine_rad_step+1)=[];
%Q_r(:,theta_engine_rad_step+1)=[];
%Q_h(:,theta_engine_rad_step+1)=[];
%W_exp(:,theta_engine_rad_step+1)=[];
%W_comp(:,theta_engine_rad_step+1)=[];
end
end
end
Work_done_Urieli=W_exp(1,theta_engine_rad_step+1)+W_comp(1,theta_engine_ra
d_step+1);
Engine_Efficiency_Urieli=Work_done_Urieli/Q_h(1,theta_engine_rad_step+1);
Words = 'Engine Work in J per revolution (Urieli)';
```

```
disp(Words)
disp(Work_done_Urieli)
Work_done_Urieli_power=Work_done_Urieli*w_engine_rpm/60;
Words = 'Engine Work in W (Urieli)';
disp(Words)
disp(Work_done_Urieli_power)
Words = 'Engine Efficiency Urieli';
disp(Words)
disp(Engine_Efficiency_Urieli)
% PV Diagram Display:
Volume_engine_Urieli=V_comp+V_exp+V_r+V_h+V_k;
figure
plot(Volume_engine_Schmidt,P_engine_Schmidt,'g','LineWidth',3)
hold on
plot(Volume_engine_Urieli,P_engine_Urieli,'r','LineWidth',3)
title('PV Diagram')
xlabel('Volume')
ylabel('Pressure')
legend('Isothermal','Adiabatic')
figure
plot(theta_engine_deg,P_engine_Schmidt,'g','LineWidth',3)
hold on
plot(theta_engine_deg_odd,P_engine_Urieli,'r','LineWidth',3)
title('Engine Pressure')
xlabel('x, shaft angle 0 to 360deg')
ylabel('P (Pressure in Pa)')
legend('Isothermal','Adiabatic')
figure
plot(theta_engine_deg,Volume_engine_Schmidt,'r','LineWidth',3)
%hold on
%plot(theta_engine_deg_odd,Volume_engine_Urieli,'r','LineWidth',3)
title('Engine Internal Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
% Gas Torque Calculation:-
pressure_gas_crank=P_engine_Schmidt;
Area_piston_crank=pi*Diameter_piston_power^2/4;
engine_stroke=engine_power_stroke;
r_crank=engine_stroke/2;
```

```
theta_crank_rad=theta_engine_rad;
for i=1:theta_engine_rad_step
theta_crank_rad_odd(1,i)=theta_crank_rad(1,i*2-1);
end
theta_crank_rad=theta_crank_rad_odd;
tangent_Phi_conrod_rad=r_crank*sin(theta_crank_rad)./(L_conrod*((1-
(r_crank/L_conrod*sin(theta_crank_rad)).^2).^(1/2)));
tangent_Phi_conrod_deg=tangent_Phi_conrod_rad*180/pi;
Force_gas_crank_piston=pressure_gas_crank*Area_piston_crank;
Area_piston_crank_displacer=pi*Diameter_piston_displacer^2/4;
tangent_Phi_conrod_rad_displacer=r_crank*sin(theta_crank_rad+pi/2)./(L_con
rod*((1-(r_crank/L_conrod*sin(theta_crank_rad+pi/2)).^2).^(1/2)));
tangent_Phi_conrod_deg_displacer=tangent_Phi_conrod_rad_displacer*180/pi;
Force_gas_crank_piston_displacer=pressure_gas_crank*Area_piston_crank_disp
lacer;
for i=1:theta_engine_rad_step
Force_gas_crank_piston_vector=[-Force_gas_crank_piston(1,i) 0 0];
Force_gas_crank_32_vector=[-Force_gas_crank_piston(1,i)
Force_gas_crank_piston(1,i).*tangent_Phi_conrod_rad(1,i) 0];
R_A_crank=[r_crank*cos(theta_crank_rad(1,i))
r_crank*sin(theta_crank_rad(1,i)) 0];
Torque_driving_crank_21_vector =
cross(R_A_crank,Force_gas_crank_32_vector);
Torque_driving_crank_21(1,i)=Torque_driving_crank_21_vector(1,3);
Force_gas_crank_32_vector_displacer=[-
Force_gas_crank_piston_displacer(1,i)
Force_gas_crank_piston_displacer(1,i).*tangent_Phi_conrod_rad_displacer(1,
i) 0];
R_A_crank_displacer=[r_crank*cos(theta_crank_rad(1,i)+pi/2)
r_crank*sin(theta_crank_rad(1,i)+pi/2) 0];
Torque_driving_crank_21_vector_displacer =
cross(R_A_crank_displacer,Force_gas_crank_32_vector_displacer);
Torque_driving_crank_21_displacer(1,i)=Torque_driving_crank_21_vector_disp
lacer(1,3);
end
figure
plot(theta_crank_rad*180/pi,(Torque_driving_crank_21+Torque_driving_crank_
21_displacer),'r','LineWidth',3)
title('Gas Torque')
```

```
xlabel('theta, crank angle 0 to 360deg')
ylabel('T21, torque in Nm')
```

APPENDIX D: Computer Simulation Program 2c - Matlab Code for Stirling engine with form-closed cam-oscillating follower (polynomial) simulation

```
%Clear all the previous data
clear all
close all
clc
%User Defined Parameters
engine_power_stroke=0.035;
engine_displacer_stroke=0.035;
Diameter_piston_power=0.060;
Diameter_piston_displacer=0.040;
w_engine=2*pi*1.8; %w=2(pi)(frequency)
theta_engine_rad_step=4000;
theta_engine_rad_step_original=theta_engine_rad_step;
%Power Piston Cam Parameters
rising_period_power=165/360;
falling_period_power=165/360;
dwell_1_power=15/360;
dwell_2_power=15/360;
Roller_follower_radius_power=0.012/2;
c_cam_power=0.15; %distance between follower center and cam center
%Displacer Piston Cam Parameters
rising_period_displacer=(45)/360; % period for rising
falling_period_displacer=(45)/360; % period for falling
dwell_1_displacer=(135)/360; % period for dwell 1
dwell_2_displacer=(135)/360; % period for dwell 2
Roller_follower_radius_displacer=0.012/2;
c_cam_displacer=0.15; %distance between follower center and cam center
dwell_cam_overlap_deg_displacer=15;
T_h = 400+273; %degree K, heater temp.
T_k = 40+273; %degree K, cooler temp.
T_0 = 25+273; %degree K, ambient temp.
r_displacer_drive_rod=0.008/2; %in meter
V_r=(3000)*10^(-6); %Regenerator Dead Volume
V_k=(3000)*10^(-6); %Cooler Dead Volume
V_h=(3000)*10^(-6); %Heater Dead Volume
V_displacer_clearance_hot=30*10^(-8);
V_displacer_clearance_cold=30*10^(-8);
```

```
V_power_clearance=0*10^(-9);
%R_spec_gas=2076.9; %J/g/K*1000=J/kg/K, specific gas constant of helium at
ambient
%density_gas=0.178; %kg/m3
R_spec_gas=287.052874; %J/g/K*1000=J/kg/K, specific gas constant of air at
ambient
density_gas=1.293; %kg/m3; at ambient
%Specific Heat Capacity
%Air
Molar_mass_gas=28.97; %kg/kmol
a_cp=28.11;
b_cp=0.1967*10^(-2);
c_cp=0.4802*10^(-5);
d_cp=-1.966*10^(-9);
T_gas=(T_h+T_k)/2; %in degree K
C_p_gas=(a_cp+b_cp*T_gas+c_cp*T_gas^2+d_cp*T_gas^3); %in kJ/(kmol*K),
ideal gas, from 273K to 1800K, at ambient
C_p_gas=C_p_gas/Molar_mass_gas*1000; %in J/(kg*K)
C_v_gas=C_p_gas-R_spec_gas;
k_isentropic=C_p_gas/C_v_gas; %gamma or k, index of isentropic
%Cam Follower Geometry Data for force analysis
%Power Piston
R_cg2_oscillating_power=0.02;
theta_cg2_oscillating_rad_power=0/180*pi;
m_cg2_oscillating_power=0.5;
I_cg2_oscillating_power=0.006;
R_cg3_oscillating_power=0.02;
theta_cg3_oscillating_rad_power=0/180*pi;
m_cg3_oscillating_power=0.8;
I_cg3_oscillating_power=0.011;
theta_cg4_oscillating_rad_power=0/180*pi;
m_cg4_oscillating_power=0.5;
I_cg4_oscillating_power=0.005;
%Displacer Piston
R_cg2_oscillating_displacer=0.02;
theta_cg2_oscillating_rad_displacer=0/180*pi;
m_cg2_oscillating_displacer=0.5;
I_cg2_oscillating_displacer=0.006;
R_cg3_oscillating_displacer=0.02;
```

```
theta_cg3_oscillating_rad_displacer=0/180*pi;
m_cg3_oscillating_displacer=0.8;
I_cg3_oscillating_displacer=0.011;
theta_cg4_oscillating_rad_displacer=0/180*pi;
m_cg4_oscillating_displacer=0.5;
I_cg4_oscillating_displacer=0.005;
%-----------------------------------------------
%Define/Initialize values for engine kinematics
w_engine_rpm=w_engine/2/pi*60;
theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
theta_engine_rad=0;
for i=1:(theta_engine_rad_step-1)
theta_engine_rad=[theta_engine_rad
theta_engine_rad(1,i)+theta_engine_rad_step_size];
end
theta_engine_deg=theta_engine_rad/pi*180;
%Cam-Oscillating Follower Analysis
%Power Piston
%Follower-Slider-crank Geometry Synthesis
engine_stroke=engine_power_stroke;
joint2_y_max_displacement=0.002; %2mm
L_conrod_oscillating_cam=(engine_stroke^2+(joint2_y_max_displacement/2)^2)
^(1/2); %L notation
l_follower_part1=joint2_y_max_displacement^2+(engine_stroke/2)^2;
l_follower_part2=-2*((engine_stroke/2)^2);
l_follower_part3=((engine_stroke/2)^2)-joint2_y_max_displacement^2;
l_follower_part4=(l_follower_part2^2-
4*l_follower_part1*l_follower_part3)^(1/2);
l_follower_part5=(-l_follower_part2-
l_follower_part4)/(2*l_follower_part1);
Follower_angle_stroke_rad=2*acos(l_follower_part5);
Follower_angle_stroke_deg=Follower_angle_stroke_rad/pi*180;
l_follower=joint2_y_max_displacement/(1-cos(Follower_angle_stroke_rad/2));
%Define/Initialize values
k_cam=8; %for 4-5-6-7 Polynomial
n_cam=k_cam-1; %Polynomial degree is 7
s_cam=0;
h_cam=Follower_angle_stroke_rad;
cam_velocity_rpm=w_engine_rpm;
```

```
cam_velocity_rad=w_engine;
rising_period=rising_period_power;
falling_period=falling_period_power;
rising_steps=round(theta_engine_rad_step*rising_period);
falling_steps=round(theta_engine_rad_step*rising_period);
dwell_1=dwell_1_power;
dwell_2=dwell_2_power;
dwell_period1_size=round(dwell_1*theta_engine_rad_step);
dwell_period2_size=round(dwell_2*theta_engine_rad_step);
if
((rising_steps+falling_steps+dwell_period1_size+dwell_period2_size)>theta_
engine_rad_step)
dwell_period1_size=dwell_period1_size-1;
dwell_period2_size=dwell_period2_size-1;
end
beta1_cam=rising_steps*theta_engine_rad_step_size; % rising duration
beta2_cam=falling_steps*theta_engine_rad_step_size; % falling duration
beta_cam=[beta1_cam beta2_cam];
dwell_period1=dwell_period1_size*theta_engine_rad_step_size;
dwell_period2=dwell_period2_size*theta_engine_rad_step_size;
dwell_1=zeros(1,dwell_period1_size);
dwell_2=zeros(1,dwell_period2_size);
dwell_2_s=h_cam*ones(1,dwell_period2_size);
%Condition : dwell_period1+beta1_cam+dwell_period2+beta2_cam=2*pi
%Constants that is generated during differentiation of equation when power
lowers (D_coefficient_cam) Calculation
D_coefficient_cam=ones(k_cam,4);
for i=1:k_cam
D_coefficient_cam(i,2)=i;
end
Temporary_array1=[1;D_coefficient_cam(:,2)];
Temporary_array1(k_cam+1,:)=[];
D_coefficient_cam(:,2)=Temporary_array1;
Temporary_array1=[];
D_coefficient_cam_power=D_coefficient_cam(:,2);
for i=1:2
Temporary_array1=[1;D_coefficient_cam_power];
Temporary_array1(k_cam+1,:)=[];
D_coefficient_cam(:,2+i)=D_coefficient_cam(:,1+i).*Temporary_array1;
```

```
D_coefficient_cam_power=Temporary_array1;
Temporary_array1=[];
end
theta_cam=[0 0;0 0];
for k=1:2
if k==1
small_steps=rising_steps;
end
if k==2
small_steps=falling_steps;
end
for i=1:(small_steps-1)
theta_cam(k,i+1)=theta_cam(k,i)+theta_engine_rad_step_size;
end
for j=1:small_steps
%Normalized cam angle value (x_cam) Caiculation
x_cam=theta_cam(k,j)/beta_cam(1,k);
%Power terms of Normalized cam angle (P_terms_cam) Caiculation
P_terms_cam=[];
for i=1:k_cam
P_terms_cam(1,i)=x_cam.^(i-1);
end
Temporary_array1=[0 P_terms_cam(1,:)];
Temporary_array1(:,k_cam+1)=[];
P_terms_cam(2,:)=Temporary_array1;
Temporary_array1=[0 P_terms_cam(2,:)];
Temporary_array1(:,k_cam+1)=[];
P_terms_cam(3,:)=Temporary_array1;
Temporary_array1=[0 P_terms_cam(3,:)];
Temporary_array1(:,k_cam+1)=[];
P_terms_cam(4,:)=Temporary_array1;
Temporary_array1=[];
% Boundary Conditions:-
%{
1)At Rising, When theta_cam=0; s_cam=0; v_cam=0; a_cam=0; j_cam=0;
2)At Rising, When theta_cam=beta1_cam; s_cam=h_cam; v_cam=0; a_cam=0;
j_cam=0;
1)At Falling, When theta_cam=0; s_cam=h_cam; v_cam=0; a_cam=0; j_cam=0;
2)At Falling, When theta_cam=beta2_cam; s_cam=0; v_cam=0; a_cam=0;
```

```
j_cam=0;
%}
%C_coefficient_cam Caiculation
C_coefficient_cam=[0;0;0;0;(35*h_cam);(-84*h_cam);(70*h_cam);(-20*h_cam)];
%svaj Caiculation
s_cam_rising(1,j)=P_terms_cam(1,:)*(C_coefficient_cam(:,1).*D_coefficient_
cam(:,1));
v_cam_rising(1,j)=(1/beta_cam(1,k))*P_terms_cam(2,:)*(C_coefficient_cam(:,
1).*D_coefficient_cam(:,2));
a_cam_rising(1,j)=((1/beta_cam(1,k))^2)*P_terms_cam(3,:)*(C_coefficient_ca
m(:,1).*D_coefficient_cam(:,3));
j_cam_rising(1,j)=((1/beta_cam(1,k))^3)*P_terms_cam(4,:)*(C_coefficient_ca
m(:,1).*D_coefficient_cam(:,4));
s_cam_falling(1,j)=h_cam-
P_terms_cam(1,:)*(C_coefficient_cam(:,1).*D_coefficient_cam(:,1));
v_cam_falling(1,j)=-
(1/beta_cam(1,k))*P_terms_cam(2,:)*(C_coefficient_cam(:,1).*D_coefficient_
cam(:,2));
a_cam_falling(1,j)=-
((1/beta_cam(1,k))^2)*P_terms_cam(3,:)*(C_coefficient_cam(:,1).*D_coeffici
ent_cam(:,3));
j_cam_falling(1,j)=-
((1/beta_cam(1,k))^3)*P_terms_cam(4,:)*(C_coefficient_cam(:,1).*D_coeffici
ent_cam(:,4));
end
end
%Piston Motion pattern (displacer/power)
plot_x_rad=theta_engine_rad;
plot_x=theta_engine_deg;
plot_y_s=[s_cam_falling(1,:) dwell_1 s_cam_rising(1,:) dwell_2_s];
plot_y_v=[v_cam_falling(1,:) dwell_1 v_cam_rising(1,:) dwell_2];
plot_y_a=[a_cam_falling(1,:) dwell_1 a_cam_rising(1,:) dwell_2];
plot_y_j=[j_cam_falling(1,:) dwell_1 j_cam_rising(1,:) dwell_2];
s_cam_real=plot_y_s;
v_cam_real=plot_y_v*cam_velocity_rad;
a_cam_real=plot_y_a*(cam_velocity_rad^2);
j_cam_real=plot_y_j*(cam_velocity_rad^3);
j_cam_power=j_cam_real;
%Cam sizing of oscillating follower
```

```
%Initialize cam & oscillating follower geometry
Roller_follower_radius=Roller_follower_radius_power;
c_cam=c_cam_power; %distance between follower center and cam center
delta_initial_deg=-Follower_angle_stroke_deg/2;
delta_initial_rad=delta_initial_deg/180*pi;
delta_follower=delta_initial_rad+s_cam_real;
R_cam=(l_follower^2+c_cam^2-
2*l_follower*c_cam*cos(delta_follower)).^(0.5);
epsilon_cam=asin(c_cam./R_cam.*sin(delta_follower));
psi_cam=acos((c_cam^2+R_cam.^2-l_follower^2)./(2*R_cam*c_cam));
psi_cam_initial=psi_cam(1,1);
gamma_cam=psi_cam_initial-psi_cam+plot_x_rad;
%Pressure Angle Calculation
period_start=1;
period_stop=period_start+falling_steps-1;
phi_falling_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*
sin(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start
:period_stop));
phi_falling_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_falling=-
pi/2+epsilon_cam(1,period_start:period_stop)+atan(1./(phi_falling_part1+ph
i_falling_part2));
period_start=period_stop+1;
period_stop=period_start+dwell_period1_size-1;
phi_low_dwell=pi/2-epsilon_cam(1,period_start:period_stop);
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
phi_rising_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*s
in(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start:
period_stop));
phi_rising_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_rising=pi/2-
epsilon_cam(1,period_start:period_stop)+atan(1./(phi_rising_part1-
phi_rising_part2));
period_start=period_stop+1;
```

```
period_stop=period_start+dwell_period2_size-1;
phi_high_dwell=-pi/2+epsilon_cam(1,period_start:period_stop);
phi_cam=[phi_falling phi_low_dwell phi_rising phi_high_dwell];
%Radius of curvature calculation
period_start=1;
period_stop=period_start+falling_steps-1;
rho_pitch_falling_part1=(c_cam^2)*(sin(delta_follower(1,period_start:perio
d_stop))).^2;
rho_pitch_falling_part2=a_cam_real(1,period_start:period_stop).*cos(phi_fa
lling)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:
period_stop)-1).*sin(phi_falling);
rho_pitch_falling_part3=l_follower*cos(phi_falling).*rho_pitch_falling_par
t2;
rho_pitch_falling_part4=c_cam*sin(phi_falling)-rho_pitch_falling_part3;
rho_pitch_falling_part5=cos(phi_falling).*rho_pitch_falling_part4;
rho_pitch_falling=rho_pitch_falling_part1./rho_pitch_falling_part5;
period_start=period_stop+1;
period_stop=period_start+dwell_period1_size-1;
rho_pitch_low_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
rho_pitch_rising_part1=(c_cam^2)*(sin(delta_follower(1,period_start:period
_stop))).^2;
rho_pitch_rising_part2=a_cam_real(1,period_start:period_stop).*cos(phi_ris
ing)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:pe
riod_stop)-1).*sin(phi_rising);
rho_pitch_rising_part3=l_follower*cos(phi_rising).*rho_pitch_rising_part2;
rho_pitch_rising_part4=c_cam*sin(phi_rising)-rho_pitch_rising_part3;
rho_pitch_rising_part5=cos(phi_rising).*rho_pitch_rising_part4;
rho_pitch_rising=rho_pitch_rising_part1./rho_pitch_rising_part5;
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
rho_pitch_high_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
rho_pitch=[rho_pitch_falling rho_pitch_low_dwell rho_pitch_rising
rho_pitch_high_dwell];
```

```
rho_c=rho_pitch-Roller_follower_radius;
%Position Analysis
oscillating_follower_link_a=l_follower;
oscillating_follower_link_b=L_conrod_oscillating_cam;
oscillating_follower_link_c=l_follower-joint2_y_max_displacement/2;
theta2_oscillating=pi/2+(Follower_angle_stroke_rad/2-s_cam_real);
theta2_oscillating_deg=theta2_oscillating/pi*180;
theta3_oscillating=asin((oscillating_follower_link_a*sin(theta2_oscillatin
g)-oscillating_follower_link_c)/oscillating_follower_link_b);
d_oscillating=oscillating_follower_link_a*cos(theta2_oscillating)-
oscillating_follower_link_b*cos(theta3_oscillating);
d_oscillating_normalize=d_oscillating-min(d_oscillating);
%Velocity Analysis
w2_oscillating=-v_cam_real; %clockwise is negative
w3_oscillating=oscillating_follower_link_a/oscillating_follower_link_b;
w3_oscillating=w3_oscillating*w2_oscillating.*cos(theta2_oscillating)./cos
(theta3_oscillating);
d_dot_oscillating=-
oscillating_follower_link_a*w2_oscillating.*sin(theta2_oscillating);
d_dot_oscillating=d_dot_oscillating+oscillating_follower_link_b*w3_oscilla
ting.*sin(theta3_oscillating);
%Acceleration Analysis
alpha2_oscillating=a_cam_real;
alpha3_oscillating=oscillating_follower_link_a*alpha2_oscillating.*cos(the
ta2_oscillating);
alpha3_oscillating=alpha3_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
alpha3_oscillating=alpha3_oscillating+oscillating_follower_link_b*(w3_osci
llating.^2).*sin(theta3_oscillating);
alpha3_oscillating=alpha3_oscillating./(oscillating_follower_link_b*cos(th
eta3_oscillating));
d_double_dot_oscillating=-
oscillating_follower_link_a*alpha2_oscillating.*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*alpha3_oscillating.*sin(theta3_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*(w3_oscillating.^2).*sin(theta3_oscillating);
```

```
x_piston_power=d_oscillating_normalize;
v_piston_power=d_dot_oscillating;
%Displacer Piston
s_cam_rising=[];
v_cam_rising=[];
a_cam_rising=[];
j_cam_rising=[];
s_cam_falling=[];
v_cam_falling=[];
a_cam_falling=[];
j_cam_falling=[];
%Follower-Slider-crank Geometry Synthesis
engine_stroke=engine_displacer_stroke;
joint2_y_max_displacement=0.002; %2mm
L_conrod_oscillating_cam=(engine_stroke^2+(joint2_y_max_displacement/2)^2)
^(1/2); %L notation
l_follower_part1=joint2_y_max_displacement^2+(engine_stroke/2)^2;
l_follower_part2=-2*((engine_stroke/2)^2);
l_follower_part3=((engine_stroke/2)^2)-joint2_y_max_displacement^2;
l_follower_part4=(l_follower_part2^2-
4*l_follower_part1*l_follower_part3)^(1/2);
l_follower_part5=(-l_follower_part2-
l_follower_part4)/(2*l_follower_part1);
Follower_angle_stroke_rad=2*acos(l_follower_part5);
Follower_angle_stroke_deg=Follower_angle_stroke_rad/pi*180;
l_follower=joint2_y_max_displacement/(1-
cos(Follower_angle_stroke_rad/2)); %B notation
%Define/Initialize values
k_cam=8; %for 4-5-6-7 Polynomial
n_cam=k_cam-1; %Polynomial degree is 7
s_cam=0;
h_cam=Follower_angle_stroke_rad;
cam_velocity_rpm=w_engine_rpm;
cam_velocity_rad=w_engine;
rising_period=rising_period_displacer; % period for rising
falling_period=falling_period_displacer; % period for falling
rising_steps=round(theta_engine_rad_step*rising_period);
falling_steps=round(theta_engine_rad_step*rising_period);
dwell_1=dwell_1_displacer;
```

```
dwell_2=dwell_2_displacer;
dwell_period1_size=round(dwell_1*theta_engine_rad_step);
dwell_period2_size=round(dwell_2*theta_engine_rad_step);
if
((rising_steps+falling_steps+dwell_period1_size+dwell_period2_size)>theta_
engine_rad_step)
dwell_period1_size=dwell_period1_size-1;
dwell_period2_size=dwell_period2_size-1;
end
%theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
beta1_cam=rising_steps*theta_engine_rad_step_size; % rising duration
beta2_cam=falling_steps*theta_engine_rad_step_size; % falling duration
beta_cam=[beta1_cam beta2_cam];
dwell_period1=dwell_period1_size*theta_engine_rad_step_size;
dwell_period2=dwell_period2_size*theta_engine_rad_step_size;
dwell_1=zeros(1,dwell_period1_size);
dwell_2=zeros(1,dwell_period2_size);
dwell_2_s=h_cam*ones(1,dwell_period2_size);
%Condition : dwell_period1+beta1_cam+dwell_period2+beta2_cam=2*pi
%Constants that is generated during differentiation of equation when power
lowers (D_coefficient_cam) Calculation
D_coefficient_cam=ones(k_cam,4);
for i=1:k_cam
D_coefficient_cam(i,2)=i;
end
Temporary_array1=[1;D_coefficient_cam(:,2)];
Temporary_array1(k_cam+1,:)=[];
D_coefficient_cam(:,2)=Temporary_array1;
Temporary_array1=[];
D_coefficient_cam_power=D_coefficient_cam(:,2);
for i=1:2
Temporary_array1=[1;D_coefficient_cam_power];
Temporary_array1(k_cam+1,:)=[];
D_coefficient_cam(:,2+i)=D_coefficient_cam(:,1+i).*Temporary_array1;
D_coefficient_cam_power=Temporary_array1;
Temporary_array1=[];
end
theta_cam=[0 0;0 0];
for k=1:2
```

```
if k==1
small_steps=rising_steps;
end
if k==2
small_steps=falling_steps;
end
for i=1:(small_steps-1)
theta_cam(k,i+1)=theta_cam(k,i)+theta_engine_rad_step_size;
end
for j=1:small_steps
%Normalized cam angle value (x_cam) Caiculation
x_cam=theta_cam(k,j)/beta_cam(1,k);
%Power terms of Normalized cam angle (P_terms_cam) Caiculation
P_terms_cam=[];
for i=1:k_cam
P_terms_cam(1,i)=x_cam.^(i-1);
end
Temporary_array1=[0 P_terms_cam(1,:)];
Temporary_array1(:,k_cam+1)=[];
P_terms_cam(2,:)=Temporary_array1;
Temporary_array1=[0 P_terms_cam(2,:)];
Temporary_array1(:,k_cam+1)=[];
P_terms_cam(3,:)=Temporary_array1;
Temporary_array1=[0 P_terms_cam(3,:)];
Temporary_array1(:,k_cam+1)=[];
P_terms_cam(4,:)=Temporary_array1;
Temporary_array1=[];
% Boundary Conditions:-
%{
1)At Rising, When theta_cam=0; s_cam=0; v_cam=0; a_cam=0; j_cam=0;
2)At Rising, When theta_cam=beta1_cam; s_cam=h_cam; v_cam=0; a_cam=0;
j_cam=0;
1)At Falling, When theta_cam=0; s_cam=h_cam; v_cam=0; a_cam=0; j_cam=0;
2)At Falling, When theta_cam=beta2_cam; s_cam=0; v_cam=0; a_cam=0;
j_cam=0;
%}
%C_coefficient_cam Caiculation
C_coefficient_cam=[0;0;0;0;(35*h_cam);(-84*h_cam);(70*h_cam);(-20*h_cam)];
%svaj Caiculation
```

```
s_cam_rising(1,j)=P_terms_cam(1,:)*(C_coefficient_cam(:,1).*D_coefficient_
cam(:,1));
v_cam_rising(1,j)=(1/beta_cam(1,k))*P_terms_cam(2,:)*(C_coefficient_cam(:,
1).*D_coefficient_cam(:,2));
a_cam_rising(1,j)=((1/beta_cam(1,k))^2)*P_terms_cam(3,:)*(C_coefficient_ca
m(:,1).*D_coefficient_cam(:,3));
j_cam_rising(1,j)=((1/beta_cam(1,k))^3)*P_terms_cam(4,:)*(C_coefficient_ca
m(:,1).*D_coefficient_cam(:,4));
s_cam_falling(1,j)=h_cam-
P_terms_cam(1,:)*(C_coefficient_cam(:,1).*D_coefficient_cam(:,1));
v_cam_falling(1,j)=-
(1/beta_cam(1,k))*P_terms_cam(2,:)*(C_coefficient_cam(:,1).*D_coefficient_
cam(:,2));
a_cam_falling(1,j)=-
((1/beta_cam(1,k))^2)*P_terms_cam(3,:)*(C_coefficient_cam(:,1).*D_coeffici
ent_cam(:,3));
j_cam_falling(1,j)=-
((1/beta_cam(1,k))^3)*P_terms_cam(4,:)*(C_coefficient_cam(:,1).*D_coeffici
ent_cam(:,4));
end
end
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++
%Piston Motion pattern (displacer/power)
plot_x_rad=theta_engine_rad;
plot_x=theta_engine_deg;
dwell_cam_overlap_deg=dwell_cam_overlap_deg_displacer;
dwell_cam_overlap_step=round(dwell_cam_overlap_deg/360*theta_engine_rad_st
ep);
plot_y_s=[dwell_1 s_cam_rising(1,:) dwell_2_s s_cam_falling(1,:)];
plot_y_v=[dwell_1 v_cam_rising(1,:) dwell_2 v_cam_falling(1,:)];
plot_y_a=[dwell_1 a_cam_rising(1,:) dwell_2 a_cam_falling(1,:)];
plot_y_j=[dwell_1 j_cam_rising(1,:) dwell_2 j_cam_falling(1,:)];
s_cam_real=circshift(plot_y_s,-dwell_cam_overlap_step);
v_cam_real=circshift(plot_y_v*cam_velocity_rad,-dwell_cam_overlap_step);
a_cam_real=circshift(plot_y_a*(cam_velocity_rad^2),-
dwell_cam_overlap_step);
j_cam_real=circshift(plot_y_j*(cam_velocity_rad^3),-
dwell_cam_overlap_step);
```

```
j_cam_displacer=j_cam_real;
%Cam sizing of oscillating follower
%Initialize cam & oscillating follower geometry
Roller_follower_radius=Roller_follower_radius_displacer;
c_cam=c_cam_displacer; %distance between follower center and cam center
delta_initial_deg=-Follower_angle_stroke_deg/2;
delta_initial_rad=delta_initial_deg/180*pi;
delta_follower=delta_initial_rad+s_cam_real;
R_cam=(l_follower^2+c_cam^2-
2*l_follower*c_cam*cos(delta_follower)).^(0.5);
epsilon_cam=asin(c_cam./R_cam.*sin(delta_follower));
psi_cam=acos((c_cam^2+R_cam.^2-l_follower^2)./(2*R_cam*c_cam));
psi_cam_initial=psi_cam(1,1);
gamma_cam=psi_cam_initial-psi_cam+plot_x_rad;
%Pressure Angle Calculation
period_start=1;
period_stop=period_start+dwell_period1_size-1;
phi_low_dwell=pi/2-epsilon_cam(1,period_start:period_stop);
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
phi_rising_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*s
in(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start:
period_stop));
phi_rising_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_rising=pi/2-
epsilon_cam(1,period_start:period_stop)+atan(1./(phi_rising_part1-
phi_rising_part2));
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
phi_high_dwell=-pi/2+epsilon_cam(1,period_start:period_stop);
period_start=period_stop+1;
period_stop=period_start+falling_steps-1;
phi_falling_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*
sin(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start
:period_stop));
```

```
phi_falling_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_falling=-
pi/2+epsilon_cam(1,period_start:period_stop)+atan(1./(phi_falling_part1+ph
i_falling_part2));
phi_cam=[phi_low_dwell phi_rising phi_high_dwell phi_falling];
%Radius of curvature calculation
period_start=1;
period_stop=period_start+dwell_period1_size-1;
rho_pitch_low_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
rho_pitch_rising_part1=(c_cam^2)*(sin(delta_follower(1,period_start:period
_stop))).^2;
rho_pitch_rising_part2=a_cam_real(1,period_start:period_stop).*cos(phi_ris
ing)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:pe
riod_stop)-1).*sin(phi_rising);
rho_pitch_rising_part3=l_follower*cos(phi_rising).*rho_pitch_rising_part2;
rho_pitch_rising_part4=c_cam*sin(phi_rising)-rho_pitch_rising_part3;
rho_pitch_rising_part5=cos(phi_rising).*rho_pitch_rising_part4;
rho_pitch_rising=rho_pitch_rising_part1./rho_pitch_rising_part5;
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
rho_pitch_high_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
period_start=period_stop+1;
period_stop=period_start+falling_steps-1;
rho_pitch_falling_part1=(c_cam^2)*(sin(delta_follower(1,period_start:perio
d_stop))).^2;
rho_pitch_falling_part2=a_cam_real(1,period_start:period_stop).*cos(phi_fa
lling)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:
period_stop)-1).*sin(phi_falling);
rho_pitch_falling_part3=l_follower*cos(phi_falling).*rho_pitch_falling_par
t2;
rho_pitch_falling_part4=c_cam*sin(phi_falling)-rho_pitch_falling_part3;
```

```
rho_pitch_falling_part5=cos(phi_falling).*rho_pitch_falling_part4;
rho_pitch_falling=rho_pitch_falling_part1./rho_pitch_falling_part5;
rho_pitch=[rho_pitch_low_dwell rho_pitch_rising rho_pitch_high_dwell
rho_pitch_falling];
rho_c=rho_pitch-Roller_follower_radius;
%Position Analysis
oscillating_follower_link_a=l_follower;
oscillating_follower_link_b=L_conrod_oscillating_cam;
oscillating_follower_link_c=l_follower-joint2_y_max_displacement/2;
theta2_oscillating=pi/2+(Follower_angle_stroke_rad/2-s_cam_real);
theta2_oscillating_deg=theta2_oscillating/pi*180;
theta3_oscillating=asin((oscillating_follower_link_a*sin(theta2_oscillatin
g)-oscillating_follower_link_c)/oscillating_follower_link_b);
d_oscillating=oscillating_follower_link_a*cos(theta2_oscillating)-
oscillating_follower_link_b*cos(theta3_oscillating);
d_oscillating_normalize=d_oscillating-min(d_oscillating);
%Velocity Analysis
w2_oscillating=-v_cam_real; %clockwise is negative
w3_oscillating=oscillating_follower_link_a/oscillating_follower_link_b;
w3_oscillating=w3_oscillating*w2_oscillating.*cos(theta2_oscillating)./cos
(theta3_oscillating);
d_dot_oscillating=-
oscillating_follower_link_a*w2_oscillating.*sin(theta2_oscillating);
d_dot_oscillating=d_dot_oscillating+oscillating_follower_link_b*w3_oscilla
ting.*sin(theta3_oscillating);
%Acceleration Analysis
alpha2_oscillating=a_cam_real;
alpha3_oscillating=oscillating_follower_link_a*alpha2_oscillating.*cos(the
ta2_oscillating);
alpha3_oscillating=alpha3_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
alpha3_oscillating=alpha3_oscillating+oscillating_follower_link_b*(w3_osci
llating.^2).*sin(theta3_oscillating);
alpha3_oscillating=alpha3_oscillating./(oscillating_follower_link_b*cos(th
eta3_oscillating));
d_double_dot_oscillating=-
oscillating_follower_link_a*alpha2_oscillating.*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
```

```
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*alpha3_oscillating.*sin(theta3_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*(w3_oscillating.^2).*sin(theta3_oscillating);
x_piston_displacer=d_oscillating_normalize;
v_piston_displacer=d_dot_oscillating;
%Stirling Engine Piston Motion Display
figure
plot(theta_engine_deg,x_piston_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,x_piston_displacer,'r','LineWidth',3)
title('s diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('s (length in meter)')
legend('power piston','displacer')
figure
plot(theta_engine_deg,v_piston_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,v_piston_displacer,'r','LineWidth',3)
title('v diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('v (m/s)')
legend('power piston','displacer')
figure
plot(theta_engine_deg,j_cam_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,j_cam_displacer,'r','LineWidth',3)
title('j diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('j (m/s3)')
legend('power piston','displacer')
%Stirling Engine Characteristics Initialisation
%Define/Initialize values for engine characteristics
T_regenerator=(T_h-T_k)./(log(T_h./T_k));
r_piston_power=Diameter_piston_power/2; %in meter;
r_piston_displacer=Diameter_piston_displacer/2; %in meter
V_sw_hot=pi*(r_piston_displacer^2)*engine_displacer_stroke;
V_sw_cold=V_sw_hot-pi*(r_displacer_drive_rod^2)*engine_displacer_stroke;
V_sw_power=pi*(r_piston_power^2)*engine_power_stroke;
```

```
Total_volume_engine_ambient=V_sw_power+V_power_clearance+V_r+V_k+V_h+V_sw_
cold+V_displacer_clearance_cold;
M_gas_engine_initial=density_gas*Total_volume_engine_ambient; %in kg
%Stirling Engine Piston Volume Calculation
V_hot_displacer=pi*(r_piston_displacer^2)*x_piston_displacer;
V_power=pi*(r_piston_power^2)*x_piston_power;
V_cold_displacer=V_sw_cold-pi*(r_piston_displacer^2-
r_displacer_drive_rod^2)*x_piston_displacer;
dV_hot_displacer=pi*(r_piston_displacer^2)*v_piston_displacer;
V_cold_engine=V_power+V_cold_displacer;
dV_cold_engine=pi*(r_piston_power^2)*v_piston_power-
pi*(r_piston_displacer^2-r_displacer_drive_rod^2)*v_piston_displacer;
%Integrate into Schmidt Model
V_hot=V_h+V_displacer_clearance_hot+V_hot_displacer;
dV_hot=dV_hot_displacer;
V_cold=V_k+V_displacer_clearance_cold+V_power_clearance+V_cold_engine;
dV_cold=dV_cold_engine;
%Integrate into Urieli Model
V_exp=V_displacer_clearance_hot+V_hot_displacer;
dV_exp=dV_hot_displacer;
V_comp=V_displacer_clearance_cold+V_power_clearance+V_cold_engine;
dV_comp=dV_cold_engine;
%Schmidt Model
P_engine_Schmidt=M_gas_engine_initial*R_spec_gas./(V_cold/T_k+V_r/T_regene
rator+V_hot/T_h);
m_regenerator_Schmidt=V_r*P_engine_Schmidt*(log(T_h/T_k))/(R_spec_gas*(T_h
-T_k));
%Note : W_engine=Integral of (P*(d_V_c/d_theta+d_V_e/d_theta)*d_theta)
Volume_engine_Schmidt=V_hot+V_cold+V_r;
Simpson_multiple=[];
for i=1:((theta_engine_rad_step)/2)
Simpson_multiple=[Simpson_multiple;2;4];
end
Simpson_multiple=[Simpson_multiple;1];
Simpson_multiple(1,1)=1;
Work_engine_part1=P_engine_Schmidt.*(dV_cold+dV_hot);
Work_engine_part1=[Work_engine_part1 Work_engine_part1(1,1)];
h=2*pi/theta_engine_rad_step;
Work_engine_Schmidt=h/3*Work_engine_part1*Simpson_multiple;
```

```
Words = 'Engine Work in J per revolution (Schmidt)';
disp(Words)
disp(Work_engine_Schmidt)
Work_done_Schmidt_power=Work_engine_Schmidt*w_engine_rpm/60;
Words = 'Engine Work in W (Schmidt)';
disp(Words)
disp(Work_done_Schmidt_power)
Work_engine_part2=P_engine_Schmidt.*(dV_hot);
Work_engine_part2=[Work_engine_part2 Work_engine_part2(1,1)];
h=2*pi/theta_engine_rad_step;
Work_engine_Schmidt_hot=h/3*Work_engine_part2*Simpson_multiple;
Q_hot_Schmidt=Work_engine_Schmidt_hot;
Engine_Efficiency_Schmidt=Work_engine_Schmidt/Q_hot_Schmidt;
Words = 'Engine Efficiency (Schmidt)';
disp(Words)
disp(Engine_Efficiency_Schmidt)
%Urieli model - ODEs in adiabatic analysis
%Separate odd and even rows for rk4 method 1/2step calculation
theta_engine_rad_step=theta_engine_rad_step/2;
theta_engine_rad_step_size=theta_engine_rad_step_size*2;
for i=1:theta_engine_rad_step
V_exp_odd(1,i)=V_exp(1,i*2-1);
dV_exp_odd(1,i)=dV_exp(1,i*2-1);
V_comp_odd(1,i)=V_comp(1,i*2-1);
dV_comp_odd(1,i)=dV_comp(1,i*2-1);
V_exp_even(1,i)=V_exp(1,i*2);
dV_exp_even(1,i)=dV_exp(1,i*2);
V_comp_even(1,i)=V_comp(1,i*2);
dV_comp_even(1,i)=dV_comp(1,i*2);
theta_engine_deg_odd(1,i)=theta_engine_deg(1,i*2-1);
end
V_exp=V_exp_odd;
dV_exp=dV_exp_odd;
V_comp=V_comp_odd;
dV_comp=dV_comp_odd;
figure
plot(theta_engine_deg_odd,V_exp,'r','LineWidth',3)
title('Expansion Volume')
xlabel('x, shaft angle 0 to 360deg')
```

```
ylabel('Volume (m3)')
figure
plot(theta_engine_deg_odd,V_comp,'r','LineWidth',3)
title('Compression Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
figure
plot(theta_engine_deg_odd,dV_exp,'g','LineWidth',3)
hold on
plot(theta_engine_deg_odd,dV_comp,'r','LineWidth',3)
title('Volume Change')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume Change (m3/s)')
%Initialize values
Temperature_term1=V_k/T_k+V_r/T_regenerator+V_h/T_h;
T_comp=ones(1,theta_engine_rad_step+1).*T_k;
T_exp=ones(1,theta_engine_rad_step+1).*T_h;
T_ck=T_comp(1,1);
T_he=T_exp(1,1);
Q_h=zeros(1,theta_engine_rad_step);
Q_k=zeros(1,theta_engine_rad_step);
Q_r=zeros(1,theta_engine_rad_step);
W_exp=zeros(1,theta_engine_rad_step);
W_comp=zeros(1,theta_engine_rad_step);
T_kr=T_k;
T_rh=T_h;
for j=1:50
for i=1:(theta_engine_rad_step)
if i==1
M_gas_engine=M_gas_engine_initial;
end
P_engine_Urieli(1,i)=M_gas_engine*R_spec_gas/(V_comp(1,i)/T_comp(1,i)+Temp
erature_term1+V_exp(1,i)/T_exp(1,i));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli(1,i)*(dV_comp(1,i)/T_ck+dV_exp(1,i)/T_he);
dP_engine_Urieli_term2=V_comp(1,i)/T_ck+k_isentropic*Temperature_term1+V_e
xp(1,i)/T_he;
dP_engine_Urieli(1,i)=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli(1,i)*V_comp(1,i)/(R_spec_gas*T_comp(1,i));
```

```
m_k=P_engine_Urieli(1,i)*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli(1,i)*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli(1,i)*V_h/(R_spec_gas*T_h);
m_exp=P_engine_Urieli(1,i)*V_exp(1,i)/(R_spec_gas*T_exp(1,i));
dm_exp(1,i)=(P_engine_Urieli(1,i)*dV_exp(1,i)+V_exp(1,i)*dP_engine_Urieli(
1,i)/k_isentropic)/(R_spec_gas*T_he);
dm_comp(1,i)=(P_engine_Urieli(1,i)*dV_comp(1,i)+V_comp(1,i)*dP_engine_Urie
li(1,i)/k_isentropic)/(R_spec_gas*T_ck);
dm_k(1,i)=m_k*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
dm_r(1,i)=m_r*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
dm_h(1,i)=m_h*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
m_dot_ck=-dm_comp(1,i);
m_dot_he=dm_exp(1,i);
m_dot_kr=m_dot_ck-dm_k(1,i);
m_dot_rh=m_dot_he+dm_h(1,i);
if m_dot_ck>0
T_ck=T_comp(1,i);
else
T_ck=T_k;
end
if m_dot_he>0
T_he=T_h;
else
T_he=T_exp(1,i);
end
dT_exp=T_exp(1,i)*(dP_engine_Urieli(1,i)/P_engine_Urieli(1,i)+dV_exp(1,i)/
V_exp(1,i)-dm_exp(1,i)/m_exp);
dT_comp=T_comp(1,i)*(dP_engine_Urieli(1,i)/P_engine_Urieli(1,i)+dV_comp(1,
i)/V_comp(1,i)-dm_comp(1,i)/m_comp);
dQ_k=V_k*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_ck*m_dot_ck-
T_kr*m_dot_kr);
dQ_r=V_r*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_kr*m_dot_kr-
T_rh*m_dot_rh);
dQ_h=V_h*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_rh*m_dot_rh-
T_he*m_dot_he);
dW_exp=P_engine_Urieli(1,i)*dV_exp(1,i);
dW_comp=P_engine_Urieli(1,i)*dV_comp(1,i);
k1_exp=theta_engine_rad_step_size*dT_exp;
k1_comp=theta_engine_rad_step_size*dT_comp;
```

```
k1_dQk=theta_engine_rad_step_size*dQ_k;
k1_dQr=theta_engine_rad_step_size*dQ_r;
k1_dQh=theta_engine_rad_step_size*dQ_h;
k1_dWe=theta_engine_rad_step_size*dW_exp;
k1_dWc=theta_engine_rad_step_size*dW_comp;
T_ck_half=T_ck;
T_he_half=T_he;
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp_even(1,i)/(T_comp(1,i
)+0.5*k1_comp)+Temperature_term1+V_exp_even(1,i)/(T_exp(1,i)+0.5*k1_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp_even(1,i)/T_ck_half+dV_exp_even
(1,i)/T_he_half);
dP_engine_Urieli_term2=V_comp_even(1,i)/T_ck_half+k_isentropic*Temperature
_term1+V_exp_even(1,i)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp_even(1,i)/(R_spec_gas*(T_comp(1,i)+0.5*
k1_comp));
m_exp=P_engine_Urieli_half*V_exp_even(1,i)/(R_spec_gas*(T_exp(1,i)+0.5*k1_
exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp_even(1,i)+V_exp_even(1,i)*dP_engi
ne_Urieli_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp_even(1,i)+V_comp_even(1,i)*dP_e
ngine_Urieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+0.5*k1_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
```

```
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+0.5*k1_exp;
end
dT_exp=(T_exp(1,i)+0.5*k1_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half
+dV_exp_even(1,i)/V_exp_even(1,i)-dm_exp_half/m_exp);
k2_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+0.5*k1_comp)*(dP_engine_Urieli_half/P_engine_Urieli_h
alf+dV_comp_even(1,i)/V_comp_even(1,i)-dm_comp_half/m_comp);
k2_comp=theta_engine_rad_step_size*dT_comp;
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp_even(1,i);
dW_comp=P_engine_Urieli_half*dV_comp_even(1,i);
k2_dQk=theta_engine_rad_step_size*dQ_k;
k2_dQr=theta_engine_rad_step_size*dQ_r;
k2_dQh=theta_engine_rad_step_size*dQ_h;
k2_dWe=theta_engine_rad_step_size*dW_exp;
k2_dWc=theta_engine_rad_step_size*dW_comp;
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp_even(1,i)/(T_comp(1,i
)+0.5*k2_comp)+Temperature_term1+V_exp_even(1,i)/(T_exp(1,i)+0.5*k2_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp_even(1,i)/T_ck_half+dV_exp_even
(1,i)/T_he_half);
dP_engine_Urieli_term2=V_comp_even(1,i)/T_ck_half+k_isentropic*Temperature
_term1+V_exp_even(1,i)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp_even(1,i)/(R_spec_gas*(T_comp(1,i)+0.5*
k2_comp));
m_exp=P_engine_Urieli_half*V_exp_even(1,i)/(R_spec_gas*(T_exp(1,i)+0.5*k2_
exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
```

```
dm_exp_half=(P_engine_Urieli_half*dV_exp_even(1,i)+V_exp_even(1,i)*dP_engi
ne_Urieli_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp_even(1,i)+V_comp_even(1,i)*dP_e
ngine_Urieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+0.5*k2_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+0.5*k2_exp;
end
dT_exp=(T_exp(1,i)+0.5*k2_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half
+dV_exp_even(1,i)/V_exp_even(1,i)-dm_exp_half/m_exp);
k3_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+0.5*k2_comp)*(dP_engine_Urieli_half/P_engine_Urieli_h
alf+dV_comp_even(1,i)/V_comp_even(1,i)-dm_comp_half/m_comp);
k3_comp=theta_engine_rad_step_size*dT_comp;
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp_even(1,i);
dW_comp=P_engine_Urieli_half*dV_comp_even(1,i);
k3_dQk=theta_engine_rad_step_size*dQ_k;
k3_dQr=theta_engine_rad_step_size*dQ_r;
k3_dQh=theta_engine_rad_step_size*dQ_h;
k3_dWe=theta_engine_rad_step_size*dW_exp;
```

```
k3_dWc=theta_engine_rad_step_size*dW_comp;
i2=i+1;
if i2>theta_engine_rad_step
i2=i2-theta_engine_rad_step;
end
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp(1,i2)/(T_comp(1,i)+k3
_comp)+Temperature_term1+V_exp(1,i2)/(T_exp(1,i)+k3_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp(1,i2)/T_ck_half+dV_exp(1,i2)/T_
he_half);
dP_engine_Urieli_term2=V_comp(1,i2)/T_ck_half+k_isentropic*Temperature_ter
m1+V_exp(1,i2)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp(1,i2)/(R_spec_gas*(T_comp(1,i)+k3_comp)
);
m_exp=P_engine_Urieli_half*V_exp(1,i2)/(R_spec_gas*(T_exp(1,i)+k3_exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp(1,i2)+V_exp(1,i2)*dP_engine_Uriel
i_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp(1,i2)+V_comp(1,i2)*dP_engine_Ur
ieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+k3_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+k3_exp;
```

```
end
dT_exp=(T_exp(1,i)+k3_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half+dV_
exp(1,i2)/V_exp(1,i2)-dm_exp_half/m_exp);
k4_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+k3_comp)*(dP_engine_Urieli_half/P_engine_Urieli_half+
dV_comp(1,i2)/V_comp(1,i2)-dm_comp_half/m_comp);
k4_comp=theta_engine_rad_step_size*dT_comp;
T_exp(1,i+1)=T_exp(1,i)+1/6*(k1_exp+2*k2_exp+2*k3_exp+k4_exp);
T_comp(1,i+1)=T_comp(1,i)+1/6*(k1_comp+2*k2_comp+2*k3_comp+k4_comp);
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp(1,i2);
dW_comp=P_engine_Urieli_half*dV_comp(1,i2);
k4_dQk=theta_engine_rad_step_size*dQ_k;
k4_dQr=theta_engine_rad_step_size*dQ_r;
k4_dQh=theta_engine_rad_step_size*dQ_h;
k4_dWe=theta_engine_rad_step_size*dW_exp;
k4_dWc=theta_engine_rad_step_size*dW_comp;
Q_k(1,i+1)=Q_k(1,i)+1/6*(k1_dQk+2*k2_dQk+2*k3_dQk+k4_dQk);
Q_r(1,i+1)=Q_r(1,i)+1/6*(k1_dQr+2*k2_dQr+2*k3_dQr+k4_dQr);
Q_h(1,i+1)=Q_h(1,i)+1/6*(k1_dQh+2*k2_dQh+2*k3_dQh+k4_dQh);
W_exp(1,i+1)=W_exp(1,i)+1/6*(k1_dWe+2*k2_dWe+2*k3_dWe+k4_dWe);
W_comp(1,i+1)=W_comp(1,i)+1/6*(k1_dWc+2*k2_dWc+2*k3_dWc+k4_dWc);
if i==theta_engine_rad_step
T_exp_diff(1,j)=T_exp(1,theta_engine_rad_step+1)-T_exp(1,1);
T_exp(1,1)=T_exp(1,theta_engine_rad_step+1);
T_exp(:,theta_engine_rad_step+1)=[];
T_comp_diff(1,j)=T_comp(1,theta_engine_rad_step+1)-T_comp(1,1);
T_comp(1,1)=T_comp(1,theta_engine_rad_step+1);
T_comp(:,theta_engine_rad_step+1)=[];
end
end
end
Work_done_Urieli=W_exp(1,theta_engine_rad_step+1)+W_comp(1,theta_engine_ra
d_step+1);
```

```
Engine_Efficiency_Urieli=Work_done_Urieli/Q_h(1,theta_engine_rad_step+1);
Words = 'Engine Work in J per revolution (Urieli)';
disp(Words)
disp(Work_done_Urieli)
Work_done_Urieli_power=Work_done_Urieli*w_engine_rpm/60;
Words = 'Engine Work in W (Urieli)';
disp(Words)
disp(Work_done_Urieli_power)
Words = 'Engine Efficiency Urieli';
disp(Words)
disp(Engine_Efficiency_Urieli)
%-------------------------------------------------------------------------
--------------------------------------------------
% PV Diagram Display:
Volume_engine_Urieli=V_comp+V_exp+V_r+V_h+V_k;
figure
plot(Volume_engine_Schmidt,P_engine_Schmidt,'g','LineWidth',3)
hold on
plot(Volume_engine_Urieli,P_engine_Urieli,'r','LineWidth',3)
title('PV Diagram')
xlabel('Volume')
ylabel('Pressure')
legend('Isothermal','Adiabatic')
figure
plot(theta_engine_deg,P_engine_Schmidt,'g','LineWidth',3)
hold on
plot(theta_engine_deg_odd,P_engine_Urieli,'r','LineWidth',3)
title('Engine Pressure')
xlabel('x, shaft angle 0 to 360deg')
ylabel('P (Pressure in Pa)')
legend('Isothermal','Adiabatic')
figure
plot(theta_engine_deg,Volume_engine_Schmidt,'r','LineWidth',3)
%hold on
%plot(theta_engine_deg_odd,Volume_engine_Urieli,'r','LineWidth',3)
title('Engine Internal Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
%Power Piston
```

```
theta_engine_rad_step=theta_engine_rad_step_original;
theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
theta_engine_rad=0;
for i=1:(theta_engine_rad_step-1)
theta_engine_rad=[theta_engine_rad
theta_engine_rad(1,i)+theta_engine_rad_step_size];
end
theta_engine_deg=theta_engine_rad/pi*180;
%Follower-Slider-crank Geometry Synthesis
engine_stroke=engine_power_stroke;
joint2_y_max_displacement=0.002; %2mm
L_conrod_oscillating_cam=(engine_stroke^2+(joint2_y_max_displacement/2)^2)
^(1/2); %L notation
l_follower_part1=joint2_y_max_displacement^2+(engine_stroke/2)^2;
l_follower_part2=-2*((engine_stroke/2)^2);
l_follower_part3=((engine_stroke/2)^2)-joint2_y_max_displacement^2;
l_follower_part4=(l_follower_part2^2-
4*l_follower_part1*l_follower_part3)^(1/2);
l_follower_part5=(-l_follower_part2-
l_follower_part4)/(2*l_follower_part1);
Follower_angle_stroke_rad=2*acos(l_follower_part5);
Follower_angle_stroke_deg=Follower_angle_stroke_rad/pi*180;
l_follower=joint2_y_max_displacement/(1-
cos(Follower_angle_stroke_rad/2)); %B notation
%Define/Initialize values
k_cam=8; %for 4-5-6-7 Polynomial
n_cam=k_cam-1; %Polynomial degree is 7
s_cam=0;
h_cam=Follower_angle_stroke_rad;
cam_velocity_rpm=w_engine_rpm;
cam_velocity_rad=w_engine;
rising_period=rising_period_power;
falling_period=falling_period_power;
rising_steps=round(theta_engine_rad_step*rising_period);
falling_steps=round(theta_engine_rad_step*rising_period);
dwell_1=dwell_1_power;
dwell_2=dwell_2_power;
dwell_period1_size=round(dwell_1*theta_engine_rad_step);
dwell_period2_size=round(dwell_2*theta_engine_rad_step);
```

```
if
((rising_steps+falling_steps+dwell_period1_size+dwell_period2_size)>theta_
engine_rad_step)
dwell_period1_size=dwell_period1_size-1;
dwell_period2_size=dwell_period2_size-1;
end
%theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
beta1_cam=rising_steps*theta_engine_rad_step_size; % rising duration
beta2_cam=falling_steps*theta_engine_rad_step_size; % falling duration
beta_cam=[beta1_cam beta2_cam];
dwell_period1=dwell_period1_size*theta_engine_rad_step_size;
dwell_period2=dwell_period2_size*theta_engine_rad_step_size;
dwell_1=zeros(1,dwell_period1_size);
dwell_2=zeros(1,dwell_period2_size);
dwell_2_s=h_cam*ones(1,dwell_period2_size);
%Condition : dwell_period1+beta1_cam+dwell_period2+beta2_cam=2*pi
% Constants that is generated during differentiation of equation when
power lowers (D_coefficient_cam) Calculation
D_coefficient_cam=ones(k_cam,4);
for i=1:k_cam
D_coefficient_cam(i,2)=i;
end
Temporary_array1=[1;D_coefficient_cam(:,2)];
Temporary_array1(k_cam+1,:)=[];
D_coefficient_cam(:,2)=Temporary_array1;
Temporary_array1=[];
D_coefficient_cam_power=D_coefficient_cam(:,2);
for i=1:2
Temporary_array1=[1;D_coefficient_cam_power];
Temporary_array1(k_cam+1,:)=[];
D_coefficient_cam(:,2+i)=D_coefficient_cam(:,1+i).*Temporary_array1;
D_coefficient_cam_power=Temporary_array1;
Temporary_array1=[];
end
theta_cam=[0 0;0 0];
for k=1:2
if k==1
small_steps=rising_steps;
end
```

```
if k==2
small_steps=falling_steps;
end
for i=1:(small_steps-1)
theta_cam(k,i+1)=theta_cam(k,i)+theta_engine_rad_step_size;
end
for j=1:small_steps
%Normalized cam angle value (x_cam) Caiculation
x_cam=theta_cam(k,j)/beta_cam(1,k);
%Power terms of Normalized cam angle (P_terms_cam) Caiculation
P_terms_cam=[];
for i=1:k_cam
P_terms_cam(1,i)=x_cam.^(i-1);
end
Temporary_array1=[0 P_terms_cam(1,:)];
Temporary_array1(:,k_cam+1)=[];
P_terms_cam(2,:)=Temporary_array1;
Temporary_array1=[0 P_terms_cam(2,:)];
Temporary_array1(:,k_cam+1)=[];
P_terms_cam(3,:)=Temporary_array1;
Temporary_array1=[0 P_terms_cam(3,:)];
Temporary_array1(:,k_cam+1)=[];
P_terms_cam(4,:)=Temporary_array1;
Temporary_array1=[];
% Boundary Conditions:-
%{
1)At Rising, When theta_cam=0; s_cam=0; v_cam=0; a_cam=0; j_cam=0;
2)At Rising, When theta_cam=beta1_cam; s_cam=h_cam; v_cam=0; a_cam=0;
j_cam=0;
1)At Falling, When theta_cam=0; s_cam=h_cam; v_cam=0; a_cam=0; j_cam=0;
2)At Falling, When theta_cam=beta2_cam; s_cam=0; v_cam=0; a_cam=0;
j_cam=0;
%}
%C_coefficient_cam Caiculation
C_coefficient_cam=[0;0;0;0;(35*h_cam);(-84*h_cam);(70*h_cam);(-20*h_cam)];
%svaj Caiculation
s_cam_rising(1,j)=P_terms_cam(1,:)*(C_coefficient_cam(:,1).*D_coefficient_
cam(:,1));
```

```
v_cam_rising(1,j)=(1/beta_cam(1,k))*P_terms_cam(2,:)*(C_coefficient_cam(:,
1).*D_coefficient_cam(:,2));
a_cam_rising(1,j)=((1/beta_cam(1,k))^2)*P_terms_cam(3,:)*(C_coefficient_ca
m(:,1).*D_coefficient_cam(:,3));
j_cam_rising(1,j)=((1/beta_cam(1,k))^3)*P_terms_cam(4,:)*(C_coefficient_ca
m(:,1).*D_coefficient_cam(:,4));
s_cam_falling(1,j)=h_cam-
P_terms_cam(1,:)*(C_coefficient_cam(:,1).*D_coefficient_cam(:,1));
v_cam_falling(1,j)=-
(1/beta_cam(1,k))*P_terms_cam(2,:)*(C_coefficient_cam(:,1).*D_coefficient_
cam(:,2));
a_cam_falling(1,j)=-
((1/beta_cam(1,k))^2)*P_terms_cam(3,:)*(C_coefficient_cam(:,1).*D_coeffici
ent_cam(:,3));
j_cam_falling(1,j)=-
((1/beta_cam(1,k))^3)*P_terms_cam(4,:)*(C_coefficient_cam(:,1).*D_coeffici
ent_cam(:,4));
end
end
%Piston Motion pattern (displacer/power)
plot_x_rad=theta_engine_rad;
plot_x=theta_engine_deg;
plot_y_s=[s_cam_falling(1,:) dwell_1 s_cam_rising(1,:) dwell_2_s];
plot_y_v=[v_cam_falling(1,:) dwell_1 v_cam_rising(1,:) dwell_2];
plot_y_a=[a_cam_falling(1,:) dwell_1 a_cam_rising(1,:) dwell_2];
plot_y_j=[j_cam_falling(1,:) dwell_1 j_cam_rising(1,:) dwell_2];
s_cam_real=plot_y_s;
v_cam_real=plot_y_v*cam_velocity_rad;
a_cam_real=plot_y_a*(cam_velocity_rad^2);
j_cam_real=plot_y_j*(cam_velocity_rad^3);
%Cam sizing of oscillating follower
%Initialize cam & oscillating follower geometry
Roller_follower_radius=Roller_follower_radius_power;
c_cam=c_cam_power; %distance between follower center and cam center
delta_initial_deg=-Follower_angle_stroke_deg/2;
delta_initial_rad=delta_initial_deg/180*pi;
delta_follower=delta_initial_rad+s_cam_real;
R_cam=(l_follower^2+c_cam^2-
2*l_follower*c_cam*cos(delta_follower)).^(0.5);
```

```
epsilon_cam=asin(c_cam./R_cam.*sin(delta_follower));
psi_cam=acos((c_cam^2+R_cam.^2-l_follower^2)./(2*R_cam*c_cam));
psi_cam_initial=psi_cam(1,1);
gamma_cam=psi_cam_initial-psi_cam+plot_x_rad;
%Pressure Angle Calculation
period_start=1;
period_stop=period_start+falling_steps-1;
phi_falling_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*
sin(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start
:period_stop));
phi_falling_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_falling=-
pi/2+epsilon_cam(1,period_start:period_stop)+atan(1./(phi_falling_part1+ph
i_falling_part2));
period_start=period_stop+1;
period_stop=period_start+dwell_period1_size-1;
phi_low_dwell=pi/2-epsilon_cam(1,period_start:period_stop);
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
phi_rising_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*s
in(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start:
period_stop));
phi_rising_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_rising=pi/2-
epsilon_cam(1,period_start:period_stop)+atan(1./(phi_rising_part1-
phi_rising_part2));
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
phi_high_dwell=-pi/2+epsilon_cam(1,period_start:period_stop);
phi_cam=[phi_falling phi_low_dwell phi_rising phi_high_dwell];
%Radius of curvature calculation
period_start=1;
period_stop=period_start+falling_steps-1;
rho_pitch_falling_part1=(c_cam^2)*(sin(delta_follower(1,period_start:perio
d_stop))).^2;
```

```
rho_pitch_falling_part2=a_cam_real(1,period_start:period_stop).*cos(phi_fa
lling)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:
period_stop)-1).*sin(phi_falling);
rho_pitch_falling_part3=l_follower*cos(phi_falling).*rho_pitch_falling_par
t2;
rho_pitch_falling_part4=c_cam*sin(phi_falling)-rho_pitch_falling_part3;
rho_pitch_falling_part5=cos(phi_falling).*rho_pitch_falling_part4;
rho_pitch_falling=rho_pitch_falling_part1./rho_pitch_falling_part5;
period_start=period_stop+1;
period_stop=period_start+dwell_period1_size-1;
rho_pitch_low_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
rho_pitch_rising_part1=(c_cam^2)*(sin(delta_follower(1,period_start:period
_stop))).^2;
rho_pitch_rising_part2=a_cam_real(1,period_start:period_stop).*cos(phi_ris
ing)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:pe
riod_stop)-1).*sin(phi_rising);
rho_pitch_rising_part3=l_follower*cos(phi_rising).*rho_pitch_rising_part2;
rho_pitch_rising_part4=c_cam*sin(phi_rising)-rho_pitch_rising_part3;
rho_pitch_rising_part5=cos(phi_rising).*rho_pitch_rising_part4;
rho_pitch_rising=rho_pitch_rising_part1./rho_pitch_rising_part5;
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
rho_pitch_high_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
rho_pitch=[rho_pitch_falling rho_pitch_low_dwell rho_pitch_rising
rho_pitch_high_dwell];
rho_c=rho_pitch-Roller_follower_radius;
%Position Analysis
oscillating_follower_link_a=l_follower;
oscillating_follower_link_b=L_conrod_oscillating_cam;
oscillating_follower_link_c=l_follower-joint2_y_max_displacement/2;
theta2_oscillating=pi/2+(Follower_angle_stroke_rad/2-s_cam_real);
theta2_oscillating_deg=theta2_oscillating/pi*180;
```

```
theta3_oscillating=asin((oscillating_follower_link_a*sin(theta2_oscillatin
g)-oscillating_follower_link_c)/oscillating_follower_link_b);
d_oscillating=oscillating_follower_link_a*cos(theta2_oscillating)-
oscillating_follower_link_b*cos(theta3_oscillating);
d_oscillating_normalize=d_oscillating-min(d_oscillating);
%Velocity Analysis
w2_oscillating=-v_cam_real; %clockwise is negative
w3_oscillating=oscillating_follower_link_a/oscillating_follower_link_b;
w3_oscillating=w3_oscillating*w2_oscillating.*cos(theta2_oscillating)./cos
(theta3_oscillating);
d_dot_oscillating=-
oscillating_follower_link_a*w2_oscillating.*sin(theta2_oscillating);
d_dot_oscillating=d_dot_oscillating+oscillating_follower_link_b*w3_oscilla
ting.*sin(theta3_oscillating);
%Acceleration Analysis
alpha2_oscillating=a_cam_real;
alpha3_oscillating=oscillating_follower_link_a*alpha2_oscillating.*cos(the
ta2_oscillating);
alpha3_oscillating=alpha3_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
alpha3_oscillating=alpha3_oscillating+oscillating_follower_link_b*(w3_osci
llating.^2).*sin(theta3_oscillating);
alpha3_oscillating=alpha3_oscillating./(oscillating_follower_link_b*cos(th
eta3_oscillating));
d_double_dot_oscillating=-
oscillating_follower_link_a*alpha2_oscillating.*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*alpha3_oscillating.*sin(theta3_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*(w3_oscillating.^2).*sin(theta3_oscillating);
%Force Analysis
R_cg2_oscillating=R_cg2_oscillating_power;
theta_cg2_oscillating_rad=theta_cg2_oscillating_rad_power;
m_cg2_oscillating=m_cg2_oscillating_power;
I_cg2_oscillating=I_cg2_oscillating_power;
R_cg3_oscillating=R_cg3_oscillating_power;
theta_cg3_oscillating_rad=theta_cg3_oscillating_rad_power;
```

```
m_cg3_oscillating=m_cg3_oscillating_power;
I_cg3_oscillating=I_cg3_oscillating_power;
theta_cg4_oscillating_rad=theta_cg4_oscillating_rad_power;
R_cg4_oscillating=oscillating_follower_link_c/cos(theta_cg4_oscillating_ra
d);
m_cg4_oscillating=m_cg4_oscillating_power;
I_cg4_oscillating=I_cg4_oscillating_power;
A_x_cg2_oscillating=R_cg2_oscillating*alpha2_oscillating.*(-
sin(theta2_oscillating+theta_cg2_oscillating_rad))-
R_cg2_oscillating*(w2_oscillating.^2).*cos(theta2_oscillating+theta_cg2_os
cillating_rad);
A_y_cg2_oscillating=R_cg2_oscillating*alpha2_oscillating.*(cos(theta2_osci
llating+theta_cg2_oscillating_rad))-
R_cg2_oscillating*(w2_oscillating.^2).*sin(theta2_oscillating+theta_cg2_os
cillating_rad);
A_x_cg3_J3_oscillating=R_cg3_oscillating*alpha3_oscillating.*(-
sin(theta3_oscillating+theta_cg3_oscillating_rad))-
R_cg3_oscillating*(w3_oscillating.^2).*cos(theta3_oscillating+theta_cg3_os
cillating_rad);
A_y_cg3_J3_oscillating=R_cg3_oscillating*alpha3_oscillating.*(cos(theta3_o
scillating+theta_cg3_oscillating_rad))-
R_cg3_oscillating*(w3_oscillating.^2).*sin(theta3_oscillating+theta_cg3_os
cillating_rad);
A_x_J2_oscillating=-
l_follower*alpha2_oscillating.*sin(theta2_oscillating)-
l_follower*(w2_oscillating.^2).*cos(theta2_oscillating);
A_y_J2_oscillating=l_follower*alpha2_oscillating.*cos(theta2_oscillating)-
l_follower*(w2_oscillating.^2).*sin(theta2_oscillating);
A_x_J3_oscillating=d_double_dot_oscillating;
A_y_J3_oscillating=0;
A_x_cg3_oscillating=A_x_J3_oscillating+A_x_cg3_J3_oscillating;
A_y_cg3_oscillating=A_y_J3_oscillating+A_y_cg3_J3_oscillating;
%Link 2 cg_joint position + Oscillating cam exerted force on follower arm
direction
theta_cg2_J1_oscillating=theta2_oscillating+theta_cg2_oscillating_rad;
theta_J1_cg2_oscillating=theta_cg2_J1_oscillating-pi;
position_J2_cg2_x=R_cg2_oscillating*sin(theta_J1_cg2_oscillating)+l_follow
er*sin(theta2_oscillating);
```

```
position_J2_cg2_y=R_cg2_oscillating*cos(theta_J1_cg2_oscillating)+l_follow
er*cos(theta2_oscillating);
%Link 3 cg_joint position
theta_cg3_J3_oscillating=theta3_oscillating+theta_cg3_oscillating_rad;
theta_J3_cg3_oscillating=theta_cg3_J3_oscillating-pi;
position_J2_cg3_x=R_cg3_oscillating*sin(theta_J3_cg3_oscillating)+L_conrod
_oscillating_cam*sin(theta3_oscillating);
position_J2_cg3_y=R_cg3_oscillating*cos(theta_J3_cg3_oscillating)+L_conrod
_oscillating_cam*cos(theta3_oscillating);
%Link 4 cg_joint position
theta_cg4_J4_oscillating=pi/2+theta_cg4_oscillating_rad;
theta_J4_cg4_oscillating=theta_cg4_J4_oscillating-pi;
position_J3_cg4_x=R_cg4_oscillating*sin(theta_J4_cg4_oscillating)+oscillat
ing_follower_link_c*sin(pi/2);
position_J3_cg4_y=R_cg4_oscillating*cos(theta_J4_cg4_oscillating)+oscillat
ing_follower_link_c*cos(pi/2);
%Matrix Construction
F_p_x_oscillating=-P_engine_Schmidt*pi*((Diameter_piston_power)^2)/4;
F_p_y_oscillating=0;
R_12y=R_cg2_oscillating*cos(theta_J1_cg2_oscillating);
R_12x=R_cg2_oscillating*sin(theta_J1_cg2_oscillating);
R_32y=position_J2_cg2_y;
R_32x=position_J2_cg2_x;
R_23y=position_J2_cg3_x;
R_23x=position_J2_cg3_y;
R_43y=R_cg3_oscillating*cos(theta_J3_cg3_oscillating);
R_43x=R_cg3_oscillating*sin(theta_J3_cg3_oscillating);
R_34y=position_J3_cg4_y;
R_34x=position_J3_cg4_x;
R_14y=R_cg4_oscillating*cos(theta_J4_cg4_oscillating);
R_14x=R_cg4_oscillating*sin(theta_J4_cg4_oscillating);
alpha4_oscillating=0;
A_x_cg4_oscillating=d_double_dot_oscillating;
A_y_cg4_oscillating=0;
for i=1:theta_engine_rad_step
sign_oscillating=1;
if d_dot_oscillating(1,i)<0
sign_oscillating=-1;
else
```

```
sign_oscillating=1;
end
friction_coefficient_oscillating=0;
Matrix1_oscillating=[1 0 1 0 0 0 0 0;0 1 0 1 0 0 0 0;-R_12y(1,i)
R_12x(1,i) -R_32y(1,i) R_32x(1,i) 0 0 0 1;0 0 -1 0 1 0 0 0;0 0 0 -1 0 1 0
0];
Matrix1_oscillating=[Matrix1_oscillating;0 0 R_23y(1,i) -R_23x(1,i) -
R_43y(1,i) R_43x(1,i) 0 0;0 0 0 0 -1 0
sign_oscillating*friction_coefficient_oscillating 0;0 0 0 0 0 -1 1 0];
Matrix2_oscillating=[m_cg2_oscillating*A_x_cg2_oscillating(1,i);m_cg2_osci
llating*A_y_cg2_oscillating(1,i)];
Matrix2_oscillating=[Matrix2_oscillating;I_cg2_oscillating*alpha2_oscillat
ing(1,i)];
Matrix2_oscillating=[Matrix2_oscillating;m_cg3_oscillating*A_x_cg3_oscilla
ting(1,i);m_cg3_oscillating*A_y_cg3_oscillating(1,i);I_cg3_oscillating*alp
ha3_oscillating(1,i)];
Matrix2_oscillating=[Matrix2_oscillating;m_cg4_oscillating*A_x_cg4_oscilla
ting(1,i)-F_p_x_oscillating(1,i);-F_p_y_oscillating];
Matrix3_oscillating(:,i)=inv(Matrix1_oscillating)*Matrix2_oscillating;
T_12_oscillating=Matrix3_oscillating(8,:);
end
T_12_oscillating_power=T_12_oscillating;
R_cam_power=R_cam;
delta_initial_rad_power=delta_initial_rad;
%Displacer Piston
s_cam_rising=[];
v_cam_rising=[];
a_cam_rising=[];
j_cam_rising=[];
s_cam_falling=[];
v_cam_falling=[];
a_cam_falling=[];
j_cam_falling=[];
%Follower-Slider-crank Geometry Synthesis
engine_stroke=engine_displacer_stroke;
joint2_y_max_displacement=0.002; %2mm
L_conrod_oscillating_cam=(engine_stroke^2+(joint2_y_max_displacement/2)^2)
^(1/2); %L notation
l_follower_part1=joint2_y_max_displacement^2+(engine_stroke/2)^2;
```

```
l_follower_part2=-2*((engine_stroke/2)^2);
l_follower_part3=((engine_stroke/2)^2)-joint2_y_max_displacement^2;
l_follower_part4=(l_follower_part2^2-
4*l_follower_part1*l_follower_part3)^(1/2);
l_follower_part5=(-l_follower_part2-
l_follower_part4)/(2*l_follower_part1);
Follower_angle_stroke_rad=2*acos(l_follower_part5);
Follower_angle_stroke_deg=Follower_angle_stroke_rad/pi*180;
l_follower=joint2_y_max_displacement/(1-
cos(Follower_angle_stroke_rad/2)); %B notation
%Define/Initialize values
k_cam=8; %for 4-5-6-7 Polynomial
n_cam=k_cam-1; %Polynomial degree is 7
s_cam=0;
h_cam=Follower_angle_stroke_rad;
cam_velocity_rpm=w_engine_rpm;
cam_velocity_rad=w_engine;
rising_period=rising_period_displacer; % period for rising
falling_period=rising_period_displacer; % period for falling
rising_steps=round(theta_engine_rad_step*rising_period);
falling_steps=round(theta_engine_rad_step*rising_period);
dwell_1=dwell_1_displacer;
dwell_2=dwell_2_displacer;
dwell_period1_size=round(dwell_1*theta_engine_rad_step);
dwell_period2_size=round(dwell_2*theta_engine_rad_step);
if
((rising_steps+falling_steps+dwell_period1_size+dwell_period2_size)>theta_
engine_rad_step)
dwell_period1_size=dwell_period1_size-1;
dwell_period2_size=dwell_period2_size-1;
end
%theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
beta1_cam=rising_steps*theta_engine_rad_step_size; % rising duration
beta2_cam=falling_steps*theta_engine_rad_step_size; % falling duration
beta_cam=[beta1_cam beta2_cam];
dwell_period1=dwell_period1_size*theta_engine_rad_step_size;
dwell_period2=dwell_period2_size*theta_engine_rad_step_size;
dwell_1=zeros(1,dwell_period1_size);
dwell_2=zeros(1,dwell_period2_size);
```

```
dwell_2_s=h_cam*ones(1,dwell_period2_size);
%Condition : dwell_period1+beta1_cam+dwell_period2+beta2_cam=2*pi
%Constants that is generated during differentiation of equation when power
lowers (D_coefficient_cam) Calculation
D_coefficient_cam=ones(k_cam,4);
for i=1:k_cam
D_coefficient_cam(i,2)=i;
end
Temporary_array1=[1;D_coefficient_cam(:,2)];
Temporary_array1(k_cam+1,:)=[];
D_coefficient_cam(:,2)=Temporary_array1;
Temporary_array1=[];
D_coefficient_cam_power=D_coefficient_cam(:,2);
for i=1:2
Temporary_array1=[1;D_coefficient_cam_power];
Temporary_array1(k_cam+1,:)=[];
D_coefficient_cam(:,2+i)=D_coefficient_cam(:,1+i).*Temporary_array1;
D_coefficient_cam_power=Temporary_array1;
Temporary_array1=[];
end
theta_cam=[0 0;0 0];
for k=1:2
if k==1
small_steps=rising_steps;
end
if k==2
small_steps=falling_steps;
end
for i=1:(small_steps-1)
theta_cam(k,i+1)=theta_cam(k,i)+theta_engine_rad_step_size;
end
for j=1:small_steps
%Normalized cam angle value (x_cam) Caiculation
x_cam=theta_cam(k,j)/beta_cam(1,k);
%Power terms of Normalized cam angle (P_terms_cam) Caiculation
P_terms_cam=[];
for i=1:k_cam
P_terms_cam(1,i)=x_cam.^(i-1);
end
```

```
Temporary_array1=[0 P_terms_cam(1,:)];
Temporary_array1(:,k_cam+1)=[];
P_terms_cam(2,:)=Temporary_array1;
Temporary_array1=[0 P_terms_cam(2,:)];
Temporary_array1(:,k_cam+1)=[];
P_terms_cam(3,:)=Temporary_array1;
Temporary_array1=[0 P_terms_cam(3,:)];
Temporary_array1(:,k_cam+1)=[];
P_terms_cam(4,:)=Temporary_array1;
Temporary_array1=[];
% Boundary Conditions:-
%{
1)At Rising, When theta_cam=0; s_cam=0; v_cam=0; a_cam=0; j_cam=0;
2)At Rising, When theta_cam=beta1_cam; s_cam=h_cam; v_cam=0; a_cam=0;
j_cam=0;
1)At Falling, When theta_cam=0; s_cam=h_cam; v_cam=0; a_cam=0; j_cam=0;
2)At Falling, When theta_cam=beta2_cam; s_cam=0; v_cam=0; a_cam=0;
j_cam=0;
%}
%C_coefficient_cam Caiculation
C_coefficient_cam=[0;0;0;0;(35*h_cam);(-84*h_cam);(70*h_cam);(-20*h_cam)];
%svaj Caiculation
s_cam_rising(1,j)=P_terms_cam(1,:)*(C_coefficient_cam(:,1).*D_coefficient_
cam(:,1));
v_cam_rising(1,j)=(1/beta_cam(1,k))*P_terms_cam(2,:)*(C_coefficient_cam(:,
1).*D_coefficient_cam(:,2));
a_cam_rising(1,j)=((1/beta_cam(1,k))^2)*P_terms_cam(3,:)*(C_coefficient_ca
m(:,1).*D_coefficient_cam(:,3));
j_cam_rising(1,j)=((1/beta_cam(1,k))^3)*P_terms_cam(4,:)*(C_coefficient_ca
m(:,1).*D_coefficient_cam(:,4));
s_cam_falling(1,j)=h_cam-
P_terms_cam(1,:)*(C_coefficient_cam(:,1).*D_coefficient_cam(:,1));
v_cam_falling(1,j)=-
(1/beta_cam(1,k))*P_terms_cam(2,:)*(C_coefficient_cam(:,1).*D_coefficient_
cam(:,2));
a_cam_falling(1,j)=-
((1/beta_cam(1,k))^2)*P_terms_cam(3,:)*(C_coefficient_cam(:,1).*D_coeffici
ent_cam(:,3));
```

```
j_cam_falling(1,j)=-
((1/beta_cam(1,k))^3)*P_terms_cam(4,:)*(C_coefficient_cam(:,1).*D_coeffici
ent_cam(:,4));
end
end
%Piston Motion pattern (displacer/power)
plot_x_rad=theta_engine_rad;
plot_x=theta_engine_deg;
dwell_cam_overlap_deg=dwell_cam_overlap_deg_displacer;
dwell_cam_overlap_step=round(dwell_cam_overlap_deg/360*theta_engine_rad_st
ep);
plot_y_s=[dwell_1 s_cam_rising(1,:) dwell_2_s s_cam_falling(1,:)];
plot_y_v=[dwell_1 v_cam_rising(1,:) dwell_2 v_cam_falling(1,:)];
plot_y_a=[dwell_1 a_cam_rising(1,:) dwell_2 a_cam_falling(1,:)];
plot_y_j=[dwell_1 j_cam_rising(1,:) dwell_2 j_cam_falling(1,:)];
s_cam_real=circshift(plot_y_s,-dwell_cam_overlap_step);
v_cam_real=circshift(plot_y_v*cam_velocity_rad,-dwell_cam_overlap_step);
a_cam_real=circshift(plot_y_a*(cam_velocity_rad^2),-
dwell_cam_overlap_step);
j_cam_real=circshift(plot_y_j*(cam_velocity_rad^3),-
dwell_cam_overlap_step);
%Cam sizing of oscillating follower
%Initialize cam & oscillating follower geometry
Roller_follower_radius=Roller_follower_radius_displacer;
c_cam=c_cam_displacer; %distance between follower center and cam center
delta_initial_deg=-Follower_angle_stroke_deg/2;
delta_initial_rad=delta_initial_deg/180*pi;
delta_follower=delta_initial_rad+s_cam_real;
R_cam=(l_follower^2+c_cam^2-
2*l_follower*c_cam*cos(delta_follower)).^(0.5);
epsilon_cam=asin(c_cam./R_cam.*sin(delta_follower));
psi_cam=acos((c_cam^2+R_cam.^2-l_follower^2)./(2*R_cam*c_cam));
psi_cam_initial=psi_cam(1,1);
gamma_cam=psi_cam_initial-psi_cam+plot_x_rad;
%Pressure Angle Calculation
period_start=1;
period_stop=period_start+dwell_period1_size-1;
phi_low_dwell=pi/2-epsilon_cam(1,period_start:period_stop);
period_start=period_stop+1;
```

```
period_stop=period_start+rising_steps-1;
phi_rising_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*s
in(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start:
period_stop));
phi_rising_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_rising=pi/2-
epsilon_cam(1,period_start:period_stop)+atan(1./(phi_rising_part1-
phi_rising_part2));
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
phi_high_dwell=-pi/2+epsilon_cam(1,period_start:period_stop);
period_start=period_stop+1;
period_stop=period_start+falling_steps-1;
phi_falling_part1=R_cam(1,period_start:period_stop).^2./(l_follower*c_cam*
sin(delta_follower(1,period_start:period_stop)).*v_cam_real(1,period_start
:period_stop));
phi_falling_part2=(c_cam^2-R_cam(1,period_start:period_stop).^2-
l_follower^2)./(2*R_cam(1,period_start:period_stop)*c_cam.*sin(psi_cam(1,p
eriod_start:period_stop)));
phi_falling=-
pi/2+epsilon_cam(1,period_start:period_stop)+atan(1./(phi_falling_part1+ph
i_falling_part2));
phi_cam=[phi_low_dwell phi_rising phi_high_dwell phi_falling];
%Radius of curvature calculation
period_start=1;
period_stop=period_start+dwell_period1_size-1;
rho_pitch_low_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
period_start=period_stop+1;
period_stop=period_start+rising_steps-1;
rho_pitch_rising_part1=(c_cam^2)*(sin(delta_follower(1,period_start:period
_stop))).^2;
rho_pitch_rising_part2=a_cam_real(1,period_start:period_stop).*cos(phi_ris
ing)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:pe
riod_stop)-1).*sin(phi_rising);
rho_pitch_rising_part3=l_follower*cos(phi_rising).*rho_pitch_rising_part2;
```

```
rho_pitch_rising_part4=c_cam*sin(phi_rising)-rho_pitch_rising_part3;
rho_pitch_rising_part5=cos(phi_rising).*rho_pitch_rising_part4;
rho_pitch_rising=rho_pitch_rising_part1./rho_pitch_rising_part5;
period_start=period_stop+1;
period_stop=period_start+dwell_period2_size-1;
rho_pitch_high_dwell=(c_cam^2+l_follower^2-
2*l_follower*c_cam*cos(delta_follower(1,period_start:period_stop))).^(0.5)
;
period_start=period_stop+1;
period_stop=period_start+falling_steps-1;
rho_pitch_falling_part1=(c_cam^2)*(sin(delta_follower(1,period_start:perio
d_stop))).^2;
rho_pitch_falling_part2=a_cam_real(1,period_start:period_stop).*cos(phi_fa
lling)+v_cam_real(1,period_start:period_stop).*(v_cam_real(1,period_start:
period_stop)-1).*sin(phi_falling);
rho_pitch_falling_part3=l_follower*cos(phi_falling).*rho_pitch_falling_par
t2;
rho_pitch_falling_part4=c_cam*sin(phi_falling)-rho_pitch_falling_part3;
rho_pitch_falling_part5=cos(phi_falling).*rho_pitch_falling_part4;
rho_pitch_falling=rho_pitch_falling_part1./rho_pitch_falling_part5;
rho_pitch=[rho_pitch_low_dwell rho_pitch_rising rho_pitch_high_dwell
rho_pitch_falling];
rho_c=rho_pitch-Roller_follower_radius;
%Position Analysis
oscillating_follower_link_a=l_follower;
oscillating_follower_link_b=L_conrod_oscillating_cam;
oscillating_follower_link_c=l_follower-joint2_y_max_displacement/2;
theta2_oscillating=pi/2+(Follower_angle_stroke_rad/2-s_cam_real);
theta2_oscillating_deg=theta2_oscillating/pi*180;
theta3_oscillating=asin((oscillating_follower_link_a*sin(theta2_oscillatin
g)-oscillating_follower_link_c)/oscillating_follower_link_b);
d_oscillating=oscillating_follower_link_a*cos(theta2_oscillating)-
oscillating_follower_link_b*cos(theta3_oscillating);
d_oscillating_normalize=d_oscillating-min(d_oscillating);
%Velocity Analysis
w2_oscillating=-v_cam_real; %clockwise is negative
w3_oscillating=oscillating_follower_link_a/oscillating_follower_link_b;
w3_oscillating=w3_oscillating*w2_oscillating.*cos(theta2_oscillating)./cos
(theta3_oscillating);
```

```
d_dot_oscillating=-
oscillating_follower_link_a*w2_oscillating.*sin(theta2_oscillating);
d_dot_oscillating=d_dot_oscillating+oscillating_follower_link_b*w3_oscilla
ting.*sin(theta3_oscillating);
%Acceleration Analysis
alpha2_oscillating=a_cam_real;
alpha3_oscillating=oscillating_follower_link_a*alpha2_oscillating.*cos(the
ta2_oscillating);
alpha3_oscillating=alpha3_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
alpha3_oscillating=alpha3_oscillating+oscillating_follower_link_b*(w3_osci
llating.^2).*sin(theta3_oscillating);
alpha3_oscillating=alpha3_oscillating./(oscillating_follower_link_b*cos(th
eta3_oscillating));
d_double_dot_oscillating=-
oscillating_follower_link_a*alpha2_oscillating.*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating-
oscillating_follower_link_a*(w2_oscillating.^2).*sin(theta2_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*alpha3_oscillating.*sin(theta3_oscillating);
d_double_dot_oscillating=d_double_dot_oscillating+oscillating_follower_lin
k_b*(w3_oscillating.^2).*sin(theta3_oscillating);
%Force Analysis
R_cg2_oscillating=R_cg2_oscillating_displacer;
theta_cg2_oscillating_rad=theta_cg2_oscillating_rad_displacer;
m_cg2_oscillating=m_cg2_oscillating_displacer;
I_cg2_oscillating=I_cg2_oscillating_displacer;
R_cg3_oscillating=R_cg3_oscillating_displacer;
theta_cg3_oscillating_rad=theta_cg3_oscillating_rad_displacer;
m_cg3_oscillating=m_cg3_oscillating_displacer;
I_cg3_oscillating=I_cg3_oscillating_displacer;
theta_cg4_oscillating_rad=theta_cg4_oscillating_rad_displacer;
R_cg4_oscillating=oscillating_follower_link_c/cos(theta_cg4_oscillating_ra
d);
m_cg4_oscillating=m_cg4_oscillating_displacer;
I_cg4_oscillating=I_cg4_oscillating_displacer;
A_x_cg2_oscillating=R_cg2_oscillating*alpha2_oscillating.*(-
sin(theta2_oscillating+theta_cg2_oscillating_rad))-
```

```
R_cg2_oscillating*(w2_oscillating.^2).*cos(theta2_oscillating+theta_cg2_os
cillating_rad);
A_y_cg2_oscillating=R_cg2_oscillating*alpha2_oscillating.*(cos(theta2_osci
llating+theta_cg2_oscillating_rad))-
R_cg2_oscillating*(w2_oscillating.^2).*sin(theta2_oscillating+theta_cg2_os
cillating_rad);
A_x_cg3_J3_oscillating=R_cg3_oscillating*alpha3_oscillating.*(-
sin(theta3_oscillating+theta_cg3_oscillating_rad))-
R_cg3_oscillating*(w3_oscillating.^2).*cos(theta3_oscillating+theta_cg3_os
cillating_rad);
A_y_cg3_J3_oscillating=R_cg3_oscillating*alpha3_oscillating.*(cos(theta3_o
scillating+theta_cg3_oscillating_rad))-
R_cg3_oscillating*(w3_oscillating.^2).*sin(theta3_oscillating+theta_cg3_os
cillating_rad);
A_x_J2_oscillating=-
l_follower*alpha2_oscillating.*sin(theta2_oscillating)-
l_follower*(w2_oscillating.^2).*cos(theta2_oscillating);
A_y_J2_oscillating=l_follower*alpha2_oscillating.*cos(theta2_oscillating)-
l_follower*(w2_oscillating.^2).*sin(theta2_oscillating);
A_x_J3_oscillating=d_double_dot_oscillating;
A_y_J3_oscillating=0;
A_x_cg3_oscillating=A_x_J3_oscillating+A_x_cg3_J3_oscillating;
A_y_cg3_oscillating=A_y_J3_oscillating+A_y_cg3_J3_oscillating;
%Link 2 cg_joint position + Oscillating cam exerted force on follower arm
direction
theta_cg2_J1_oscillating=theta2_oscillating+theta_cg2_oscillating_rad;
theta_J1_cg2_oscillating=theta_cg2_J1_oscillating-pi;
position_J2_cg2_x=R_cg2_oscillating*sin(theta_J1_cg2_oscillating)+l_follow
er*sin(theta2_oscillating);
position_J2_cg2_y=R_cg2_oscillating*cos(theta_J1_cg2_oscillating)+l_follow
er*cos(theta2_oscillating);
%Link 3 cg_joint position
theta_cg3_J3_oscillating=theta3_oscillating+theta_cg3_oscillating_rad;
theta_J3_cg3_oscillating=theta_cg3_J3_oscillating-pi;
position_J2_cg3_x=R_cg3_oscillating*sin(theta_J3_cg3_oscillating)+L_conrod
_oscillating_cam*sin(theta3_oscillating);
position_J2_cg3_y=R_cg3_oscillating*cos(theta_J3_cg3_oscillating)+L_conrod
_oscillating_cam*cos(theta3_oscillating);
%Link 4 cg_joint position
```

```
theta_cg4_J4_oscillating=pi/2+theta_cg4_oscillating_rad;
theta_J4_cg4_oscillating=theta_cg4_J4_oscillating-pi;
position_J3_cg4_x=R_cg4_oscillating*sin(theta_J4_cg4_oscillating)+oscillat
ing_follower_link_c*sin(pi/2);
position_J3_cg4_y=R_cg4_oscillating*cos(theta_J4_cg4_oscillating)+oscillat
ing_follower_link_c*cos(pi/2);
%Matrix Construction
F_p_x_oscillating=-P_engine_Schmidt*pi*((Diameter_piston_displacer)^2)/4;
F_p_y_oscillating=0;
R_12y=R_cg2_oscillating*cos(theta_J1_cg2_oscillating);
R_12x=R_cg2_oscillating*sin(theta_J1_cg2_oscillating);
R_32y=position_J2_cg2_y;
R_32x=position_J2_cg2_x;
R_23y=position_J2_cg3_x;
R_23x=position_J2_cg3_y;
R_43y=R_cg3_oscillating*cos(theta_J3_cg3_oscillating);
R_43x=R_cg3_oscillating*sin(theta_J3_cg3_oscillating);
R_34y=position_J3_cg4_y;
R_34x=position_J3_cg4_x;
R_14y=R_cg4_oscillating*cos(theta_J4_cg4_oscillating);
R_14x=R_cg4_oscillating*sin(theta_J4_cg4_oscillating);
alpha4_oscillating=0;
A_x_cg4_oscillating=d_double_dot_oscillating;
A_y_cg4_oscillating=0;
for i=1:theta_engine_rad_step
sign_oscillating=1;
if d_dot_oscillating(1,i)<0
sign_oscillating=-1;
else
sign_oscillating=1;
end
friction_coefficient_oscillating=0;
Matrix1_oscillating=[1 0 1 0 0 0 0 0;0 1 0 1 0 0 0 0;-R_12y(1,i)
R_12x(1,i) -R_32y(1,i) R_32x(1,i) 0 0 0 1;0 0 -1 0 1 0 0 0;0 0 0 -1 0 1 0
0];
Matrix1_oscillating=[Matrix1_oscillating;0 0 R_23y(1,i) -R_23x(1,i) -
R_43y(1,i) R_43x(1,i) 0 0;0 0 0 0 -1 0
sign_oscillating*friction_coefficient_oscillating 0;0 0 0 0 0 -1 1 0];
```

```
Matrix2_oscillating=[m_cg2_oscillating*A_x_cg2_oscillating(1,i);m_cg2_osci
llating*A_y_cg2_oscillating(1,i)];
Matrix2_oscillating=[Matrix2_oscillating;I_cg2_oscillating*alpha2_oscillat
ing(1,i)];
Matrix2_oscillating=[Matrix2_oscillating;m_cg3_oscillating*A_x_cg3_oscilla
ting(1,i);m_cg3_oscillating*A_y_cg3_oscillating(1,i);I_cg3_oscillating*alp
ha3_oscillating(1,i)];
Matrix2_oscillating=[Matrix2_oscillating;m_cg4_oscillating*A_x_cg4_oscilla
ting(1,i)-F_p_x_oscillating(1,i);-F_p_y_oscillating];
Matrix3_oscillating(:,i)=inv(Matrix1_oscillating)*Matrix2_oscillating;
T_12_oscillating=Matrix3_oscillating(8,:);
end
T_12_oscillating_displacer=T_12_oscillating;
R_cam_displacer=R_cam;
delta_initial_rad_displacer=delta_initial_rad;
figure
plot(plot_x,(T_12_oscillating_power),'r','LineWidth',3)
title('Power Piston Gas Torque diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('T (gas torque in Nm)')
figure
plot(plot_x,(T_12_oscillating_displacer),'r','LineWidth',3)
title('Displacer Gas Torque diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('T (gas torque in Nm)')
figure
plot(plot_x,(T_12_oscillating_power+T_12_oscillating_displacer),'r','LineW
idth',3)
title('Gas Torque diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('T (gas torque in Nm)')
% Oscillating Cam profile power
oscillating_cam_profile_power_x=R_cam_power.*cos(theta_engine_rad+delta_in
itial_rad_power);
oscillating_cam_profile_power_y=R_cam_power.*sin(theta_engine_rad+delta_in
itial_rad_power);
oscillating_cam_profile_displacer_x=R_cam_displacer.*cos(theta_engine_rad+
delta_initial_rad_displacer);
```

```
oscillating_cam_profile_displacer_y=R_cam_displacer.*sin(theta_engine_rad+
delta_initial_rad_displacer);
figure
plot(oscillating_cam_profile_power_x,oscillating_cam_profile_power_y,'r','
LineWidth',3)
title('oscillating cam profile power')
xlabel('x')
ylabel('y')
figure
plot(oscillating_cam_profile_displacer_x,oscillating_cam_profile_displacer
_y,'r','LineWidth',3)
title('oscillating cam profile displacer')
xlabel('x')
ylabel('y')
```

APPENDIX E: Computer Simulation Program 3a - Matlab Code for Stirling engine with modified Scotch Yoke (MSY) simulation

```
%Clear all the previous data
clear all
close all
clc
%User Defined Parameters
engine_power_stroke=0.035;
engine_displacer_stroke=0.035;
Diameter_piston_power=0.0346;
Diameter_piston_displacer=0.040;
w_engine=2*pi*1.8; %w=2(pi)(frequency)
theta_engine_rad_step=4000;
theta_engine_rad_step_original=theta_engine_rad_step;
%Power piston Slot Curve Parameters
dwell_period_power=0.25; % 25% period dwell at each side
Transition_period_power=9; % x deg for transition between flat and
sine/cos curve (key in multiple of "9")
%Displacer piston Slot Curve Parameters
dwell_period_displacer=0.25; % 25% period dwell at each side
```

```
Transition_period_displacer=9; % x deg for transition between flat and
sine/cos curve (key in multiple of "9")
T_h = 200+273; %degree K, heater temp.
T_k = 40+273; %degree K, cooler temp.
T_0 = 25+273; %degree K, ambient temp.
r_displacer_drive_rod=0.008/2; %in meter
V_r=(1)*10^(-6); %Regenerator Dead Volume
V_k=(1)*10^(-6); %Cooler Dead Volume
V_h=(800)*10^(-6); %Heater Dead Volume
V_displacer_clearance_hot=800*10^(-8);
V_displacer_clearance_cold=800*10^(-8);
V_power_clearance=0*10^(-9);
%R_spec_gas=2076.9; %J/g/K*1000=J/kg/K, specific gas constant of helium at
ambient
%density_gas=0.178; %kg/m3
R_spec_gas=287.052874; %J/g/K*1000=J/kg/K, specific gas constant of air at
ambient
density_gas=1.293; %kg/m3; at ambient
%Specific Heat Capacity
%Air
Molar_mass_gas=28.97; %kg/kmol
a_cp=28.11;
b_cp=0.1967*10^(-2);
c_cp=0.4802*10^(-5);
d_cp=-1.966*10^(-9);
T_gas=(T_h+T_k)/2; %in degree K
C_p_gas=(a_cp+b_cp*T_gas+c_cp*T_gas^2+d_cp*T_gas^3); %in kJ/(kmol*K),
ideal gas, from 273K to 1800K, at ambient
C_p_gas=C_p_gas/Molar_mass_gas*1000; %in J/(kg*K)
C_v_gas=C_p_gas-R_spec_gas;
k_isentropic=C_p_gas/C_v_gas; %gamma or k, index of isentropic
%------------------------------------------------------------
%Define/Initialize values for engine kinematics
w_engine_rpm=w_engine/2/pi*60;
theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
theta_engine_rad=0;
for i=1:(theta_engine_rad_step-1)
theta_engine_rad=[theta_engine_rad
theta_engine_rad(1,i)+theta_engine_rad_step_size];
```

```
end
theta_engine_deg=theta_engine_rad/pi*180;
%Modified Scotch Yoke Analysis
%Power Piston
engine_stroke=engine_power_stroke;
%Define/Initialize values
dwell_period=dwell_period_power;
Transition_period=Transition_period_power;
moving_period=(1-dwell_period*2)/2;
%Transition_angle_number=floor(Transition_period/90*(500));
Transition_angle_number=floor(Transition_period/90*(theta_engine_rad_step*
moving_period));
r_rotation=engine_stroke/2/sin(moving_period*pi);
moving_angle_falling=theta_engine_rad(moving_period/2*theta_engine_rad_ste
p+1+Transition_angle_number:moving_period*3/2*theta_engine_rad_step+1-
Transition_angle_number);
moving_angle_falling_deg=moving_angle_falling/pi*180;
moving_angle_rising=theta_engine_rad(moving_period*5/2*theta_engine_rad_st
ep+1+Transition_angle_number:moving_period*7/2*theta_engine_rad_step+1-
Transition_angle_number);
moving_angle_rising_deg=moving_angle_rising/pi*180;
% Piston Position:-
x_piston_scotch_yoke_falling=r_rotation+r_rotation*cos(moving_angle_fallin
g)-(r_rotation-r_rotation*sin(moving_period*pi));
x_piston_scotch_yoke_rising=r_rotation+r_rotation*cos(moving_angle_rising)
-(r_rotation-r_rotation*sin(moving_period*pi));
v_piston_scotch_yoke_falling=-r_rotation*sin(moving_angle_falling);
v_piston_scotch_yoke_rising=-r_rotation*sin(moving_angle_rising);
a_piston_scotch_yoke_falling=-r_rotation*cos(moving_angle_falling);
a_piston_scotch_yoke_rising=-r_rotation*cos(moving_angle_rising);
j_piston_scotch_yoke_falling=r_rotation*sin(moving_angle_falling);
j_piston_scotch_yoke_rising=r_rotation*sin(moving_angle_rising);
%Transition between dwell and rising
%Polynomial function transition rising
Beta_transition=Transition_period/180*pi;
polynomial_multiples=[1 1 1 1;4 5 6 7;12 20 30 42;24 60 120 210];
polynomial_multiples(2,:)=1/Beta_transition*polynomial_multiples(2,:);
polynomial_multiples(3,:)=1/(Beta_transition^2)*polynomial_multiples(3,:);
polynomial_multiples(4,:)=1/(Beta_transition^3)*polynomial_multiples(4,:);
```

```
C0=0;
polynomial_X(1,:)=r_rotation*cos((225+Transition_period)/180*pi)-
r_rotation*cos(225/180*pi)-C0;
polynomial_X(2,:)=-r_rotation*sin((225+Transition_period)/180*pi);
polynomial_X(3,:)=-r_rotation*cos((225+Transition_period)/180*pi);
polynomial_X(4,:)=r_rotation*sin((225+Transition_period)/180*pi);
C4_to_C7=inv(polynomial_multiples)*(polynomial_X);
%Transition angle rising
Transition_angle_rising=theta_engine_rad(moving_period*5/2*theta_engine_ra
d_step+1:moving_period*5/2*theta_engine_rad_step+Transition_angle_number);
Transition_angle_rising_deg=Transition_angle_rising/pi*180;
Transition_angle_rising_nom=Transition_angle_rising-
min(Transition_angle_rising);
Transition_angle_rising_nom_deg=Transition_angle_rising_nom/pi*180;
x_transition_rising_angle=Transition_angle_rising_nom/Beta_transition;
s_transition_rising=C4_to_C7(1,1)*x_transition_rising_angle.^4+C4_to_C7(2,
1)*x_transition_rising_angle.^5+C0;
s_transition_rising=s_transition_rising+C4_to_C7(3,1)*x_transition_rising_
angle.^6+C4_to_C7(4,1)*x_transition_rising_angle.^7;
v_transition_rising=C4_to_C7(1,1)*4*x_transition_rising_angle.^3+C4_to_C7(
2,1)*5*x_transition_rising_angle.^4;
v_transition_rising=v_transition_rising+C4_to_C7(3,1)*6*x_transition_risin
g_angle.^5+C4_to_C7(4,1)*7*x_transition_rising_angle.^6;
v_transition_rising=1/Beta_transition*v_transition_rising;
a_transition_rising=C4_to_C7(1,1)*12*x_transition_rising_angle.^2+C4_to_C7
(2,1)*20*x_transition_rising_angle.^3;
a_transition_rising=a_transition_rising+C4_to_C7(3,1)*30*x_transition_risi
ng_angle.^4+C4_to_C7(4,1)*42*x_transition_rising_angle.^5;
a_transition_rising=1/(Beta_transition^2)*a_transition_rising;
j_transition_rising=C4_to_C7(1,1)*24*x_transition_rising_angle+C4_to_C7(2,
1)*60*x_transition_rising_angle.^2;
j_transition_rising=j_transition_rising+C4_to_C7(3,1)*120*x_transition_ris
ing_angle.^3+C4_to_C7(4,1)*210*x_transition_rising_angle.^4;
j_transition_rising=1/(Beta_transition^3)*j_transition_rising;
%Transition between dwell and falling
%polynomial function transition falling
Beta_transition=Transition_period/180*pi;
polynomial_multiples=[1 1 1 1;4 5 6 7;12 20 30 42;24 60 120 210];
polynomial_multiples(2,:)=1/Beta_transition*polynomial_multiples(2,:);
polynomial_multiples(3,:)=1/(Beta_transition^2)*polynomial_multiples(3,:);
polynomial_multiples(4,:)=1/(Beta_transition^3)*polynomial_multiples(4,:);
C0=2*r_rotation*cos(45/180*pi);
polynomial_X(1,:)=r_rotation*cos((45+Transition_period)/180*pi)+r_rotation
*cos(45/180*pi)-C0;
polynomial_X(2,:)=-r_rotation*sin((45+Transition_period)/180*pi);
polynomial_X(3,:)=-r_rotation*cos((45+Transition_period)/180*pi);
polynomial_X(4,:)=r_rotation*sin((45+Transition_period)/180*pi);
C4_to_C7=inv(polynomial_multiples)*(polynomial_X);
%Transition angle transition falling
Transition_angle_falling=theta_engine_rad(moving_period*1/2*theta_engine_r
ad_step+1:moving_period*1/2*theta_engine_rad_step+Transition_angle_number)
;
Transition_angle_falling_deg=Transition_angle_falling/pi*180;
Transition_angle_falling_nom=Transition_angle_falling-
min(Transition_angle_falling);
Transition_angle_falling_nom_deg=Transition_angle_falling_nom/pi*180;
x_transition_falling_angle=Transition_angle_falling_nom/Beta_transition;
s_transition_falling=C4_to_C7(1,1)*x_transition_falling_angle.^4+C4_to_C7(
2,1)*x_transition_falling_angle.^5+C0;
s_transition_falling=s_transition_falling+C4_to_C7(3,1)*x_transition_falli
ng_angle.^6+C4_to_C7(4,1)*x_transition_falling_angle.^7;
v_transition_falling=C4_to_C7(1,1)*4*x_transition_falling_angle.^3+C4_to_C
7(2,1)*5*x_transition_falling_angle.^4;
v_transition_falling=v_transition_falling+C4_to_C7(3,1)*6*x_transition_fal
ling_angle.^5+C4_to_C7(4,1)*7*x_transition_falling_angle.^6;
v_transition_falling=1/Beta_transition*v_transition_falling;
a_transition_falling=C4_to_C7(1,1)*12*x_transition_falling_angle.^2+C4_to_
C7(2,1)*20*x_transition_falling_angle.^3;
a_transition_falling=a_transition_falling+C4_to_C7(3,1)*30*x_transition_fa
lling_angle.^4+C4_to_C7(4,1)*42*x_transition_falling_angle.^5;
a_transition_falling=1/(Beta_transition^2)*a_transition_falling;
j_transition_falling=C4_to_C7(1,1)*24*x_transition_falling_angle+C4_to_C7(
2,1)*60*x_transition_falling_angle.^2;
j_transition_falling=j_transition_falling+C4_to_C7(3,1)*120*x_transition_f
alling_angle.^3+C4_to_C7(4,1)*210*x_transition_falling_angle.^4;
j_transition_falling=1/(Beta_transition^3)*j_transition_falling;
%Transition between rising and dwell
%polynomial function transition rising2
```

```
Beta_transition=Transition_period/180*pi;
polynomial_multiples=[1 1 1 1;4 5 6 7;12 20 30 42;24 60 120 210];
polynomial_multiples(2,:)=1/Beta_transition*polynomial_multiples(2,:);
polynomial_multiples(3,:)=1/(Beta_transition^2)*polynomial_multiples(3,:);
polynomial_multiples(4,:)=1/(Beta_transition^3)*polynomial_multiples(4,:);
C0=r_rotation*cos((315-
Transition_period)/180*pi)+r_rotation*cos(45/180*pi);
C1=Beta_transition*(-r_rotation*sin((315-Transition_period)/180*pi));
C2=(Beta_transition^2)/2*(-r_rotation*cos((315-
Transition_period)/180*pi));
C3=(Beta_transition^3)/6*(r_rotation*sin((315-Transition_period)/180*pi));
polynomial_X(1,:)=2*r_rotation*cos(45/180*pi)-C0-C1-C2-C3;
polynomial_X(2,:)=0-C1/Beta_transition-2*C2/Beta_transition-
3*C3/Beta_transition;
polynomial_X(3,:)=0-2*C2/(Beta_transition^2)-6*C3/(Beta_transition^2);
polynomial_X(4,:)=0-6*C3/(Beta_transition^3);
C4_to_C7=inv(polynomial_multiples)*(polynomial_X);
%Transition angle rising2
Transition_angle_rising2=theta_engine_rad(moving_period*7/2*theta_engine_r
ad_step-
Transition_angle_number+2:moving_period*7/2*theta_engine_rad_step+1);
Transition_angle_rising2_deg=Transition_angle_rising2/pi*180;
Transition_angle_rising2_nom=Transition_angle_rising2-
(max(Transition_angle_rising2_deg)-Transition_period)/180*pi;
Transition_angle_rising2_nom_deg=Transition_angle_rising2_nom/pi*180;
x_transition_rising2_angle=Transition_angle_rising2_nom/Beta_transition;
s_transition_rising2=C4_to_C7(1,1)*x_transition_rising2_angle.^4+C4_to_C7(
2,1)*x_transition_rising2_angle.^5+C0+C1*x_transition_rising2_angle+C2*x_t
ransition_rising2_angle.^2;
s_transition_rising2=s_transition_rising2+C4_to_C7(3,1)*x_transition_risin
g2_angle.^6+C4_to_C7(4,1)*x_transition_rising2_angle.^7+C3*x_transition_ri
sing2_angle.^3;
v_transition_rising2=C4_to_C7(1,1)*4*x_transition_rising2_angle.^3+C4_to_C
7(2,1)*5*x_transition_rising2_angle.^4+C1+2*C2*x_transition_rising2_angle;
v_transition_rising2=v_transition_rising2+C4_to_C7(3,1)*6*x_transition_ris
ing2_angle.^5+C4_to_C7(4,1)*7*x_transition_rising2_angle.^6+3*C3*x_transit
ion_rising2_angle.^2;
v_transition_rising2=1/Beta_transition*v_transition_rising2;
```

```
a_transition_rising2=C4_to_C7(1,1)*12*x_transition_rising2_angle.^2+C4_to_
C7(2,1)*20*x_transition_rising2_angle.^3+2*C2+6*C3*x_transition_rising2_an
gle;
a_transition_rising2=a_transition_rising2+C4_to_C7(3,1)*30*x_transition_ri
sing2_angle.^4+C4_to_C7(4,1)*42*x_transition_rising2_angle.^5;
a_transition_rising2=1/(Beta_transition^2)*a_transition_rising2;
j_transition_rising2=C4_to_C7(1,1)*24*x_transition_rising2_angle+C4_to_C7(
2,1)*60*x_transition_rising2_angle.^2+6*C3;
j_transition_rising2=j_transition_rising2+C4_to_C7(3,1)*120*x_transition_r
ising2_angle.^3+C4_to_C7(4,1)*210*x_transition_rising2_angle.^4;
j_transition_rising2=1/(Beta_transition^3)*j_transition_rising2;
%Transition between falling and dwell
%polynomial function transition falling2
Beta_transition=Transition_period/180*pi;
polynomial_multiples=[1 1 1 1;4 5 6 7;12 20 30 42;24 60 120 210];
polynomial_multiples(2,:)=1/Beta_transition*polynomial_multiples(2,:);
polynomial_multiples(3,:)=1/(Beta_transition^2)*polynomial_multiples(3,:);
polynomial_multiples(4,:)=1/(Beta_transition^3)*polynomial_multiples(4,:);
C0=r_rotation*cos((135-Transition_period)/180*pi)-
r_rotation*cos(135/180*pi);
C1=Beta_transition*(-r_rotation*sin((135-Transition_period)/180*pi));
C2=(Beta_transition^2)/2*(-r_rotation*cos((135-
Transition_period)/180*pi));
C3=(Beta_transition^3)/6*(r_rotation*sin((135-Transition_period)/180*pi));
polynomial_X(1,:)=0-C0-C1-C2-C3;
polynomial_X(2,:)=0-C1/Beta_transition-2*C2/Beta_transition-
3*C3/Beta_transition;
polynomial_X(3,:)=0-2*C2/(Beta_transition^2)-6*C3/(Beta_transition^2);
polynomial_X(4,:)=0-6*C3/(Beta_transition^3);
C4_to_C7=inv(polynomial_multiples)*(polynomial_X);
%Transition angle falling2
Transition_angle_falling2=theta_engine_rad(moving_period*3/2*theta_engine_
rad_step-
Transition_angle_number+2:moving_period*3/2*theta_engine_rad_step+1);
Transition_angle_falling2_deg=Transition_angle_falling2/pi*180;
Transition_angle_falling2_nom=Transition_angle_falling2-
(max(Transition_angle_falling2_deg)-Transition_period)/180*pi;
Transition_angle_falling2_nom_deg=Transition_angle_falling2_nom/pi*180;
x_transition_falling2_angle=Transition_angle_falling2_nom/Beta_transition;
```

```
s_transition_falling2=C4_to_C7(1,1)*x_transition_falling2_angle.^4+C4_to_C
7(2,1)*x_transition_falling2_angle.^5+C0+C1*x_transition_falling2_angle+C2
*x_transition_falling2_angle.^2;
s_transition_falling2=s_transition_falling2+C4_to_C7(3,1)*x_transition_fal
ling2_angle.^6+C4_to_C7(4,1)*x_transition_falling2_angle.^7+C3*x_transitio
n_falling2_angle.^3;
v_transition_falling2=C4_to_C7(1,1)*4*x_transition_falling2_angle.^3+C4_to
_C7(2,1)*5*x_transition_falling2_angle.^4+C1+2*C2*x_transition_falling2_an
gle;
v_transition_falling2=v_transition_falling2+C4_to_C7(3,1)*6*x_transition_f
alling2_angle.^5+C4_to_C7(4,1)*7*x_transition_falling2_angle.^6+3*C3*x_tra
nsition_falling2_angle.^2;
v_transition_falling2=1/Beta_transition*v_transition_falling2;
a_transition_falling2=C4_to_C7(1,1)*12*x_transition_falling2_angle.^2+C4_t
o_C7(2,1)*20*x_transition_falling2_angle.^3+2*C2+6*C3*x_transition_falling
2_angle;
a_transition_falling2=a_transition_falling2+C4_to_C7(3,1)*30*x_transition_
falling2_angle.^4+C4_to_C7(4,1)*42*x_transition_falling2_angle.^5;
a_transition_falling2=1/(Beta_transition^2)*a_transition_falling2;
j_transition_falling2=C4_to_C7(1,1)*24*x_transition_falling2_angle+C4_to_C
7(2,1)*60*x_transition_falling2_angle.^2+6*C3;
j_transition_falling2=j_transition_falling2+C4_to_C7(3,1)*120*x_transition
_falling2_angle.^3+C4_to_C7(4,1)*210*x_transition_falling2_angle.^4;
j_transition_falling2=1/(Beta_transition^3)*j_transition_falling2;
dwell_period_size=theta_engine_rad_step*dwell_period;
dwell_1_half=zeros(1,dwell_period_size/2);
dwell_1_half2=zeros(1,dwell_period_size/2-1);
dwell_1_s_half=dwell_1_half+engine_stroke;
dwell_1_s_half2=dwell_1_half2+engine_stroke;
dwell_2=zeros(1,dwell_period_size-1);
s_modified_scotch_yoke=[s_transition_falling x_piston_scotch_yoke_falling
s_transition_falling2];
s_modified_scotch_yoke=[s_modified_scotch_yoke dwell_2 s_transition_rising
x_piston_scotch_yoke_rising];
s_modified_scotch_yoke=[s_modified_scotch_yoke s_transition_rising2
dwell_1_s_half2 dwell_1_s_half];
v_modified_scotch_yoke=[v_transition_falling v_piston_scotch_yoke_falling
v_transition_falling2];
```

```
v_modified_scotch_yoke=[v_modified_scotch_yoke dwell_2 v_transition_rising
v_piston_scotch_yoke_rising];
v_modified_scotch_yoke=[v_modified_scotch_yoke v_transition_rising2
dwell_1_half2 dwell_1_half];
a_modified_scotch_yoke=[a_transition_falling a_piston_scotch_yoke_falling
a_transition_falling2];
a_modified_scotch_yoke=[a_modified_scotch_yoke dwell_2 a_transition_rising
a_piston_scotch_yoke_rising];
a_modified_scotch_yoke=[a_modified_scotch_yoke a_transition_rising2
dwell_1_half2 dwell_1_half];
j_modified_scotch_yoke=[j_transition_falling j_piston_scotch_yoke_falling
j_transition_falling2];
j_modified_scotch_yoke=[j_modified_scotch_yoke dwell_2 j_transition_rising
j_piston_scotch_yoke_rising];
j_modified_scotch_yoke=[j_modified_scotch_yoke j_transition_rising2
dwell_1_half2 dwell_1_half];
x_piston_power=s_modified_scotch_yoke;
v_piston_power=v_modified_scotch_yoke; %length/rad
j_piston_power=j_modified_scotch_yoke;
%Slot Transition Curve Calculation
s_modified_scotch_yoke1=[dwell_1_s_half s_transition_falling
x_piston_scotch_yoke_falling s_transition_falling2];
s_modified_scotch_yoke1=[s_modified_scotch_yoke1 dwell_2
s_transition_rising x_piston_scotch_yoke_rising];
s_modified_scotch_yoke1=[s_modified_scotch_yoke1 s_transition_rising2
dwell_1_s_half2];
theta_engine_rad2=[theta_engine_rad theta_engine_rad(1,1)];
s_modified_scotch_yoke2=[s_modified_scotch_yoke1
s_modified_scotch_yoke1(1,1)];
Slot_Curve_Angle=[];
for i=1:theta_engine_rad_step
a1=r_rotation*(cos(theta_engine_rad2(1,i+1))-cos(theta_engine_rad2(1,i)));
b1=r_rotation*(sin(theta_engine_rad2(1,i+1))-sin(theta_engine_rad2(1,i)));
x1=s_modified_scotch_yoke2(1,i+1)-s_modified_scotch_yoke2(1,i);
x2(1,i)=s_modified_scotch_yoke2(1,i+1)-s_modified_scotch_yoke2(1,i);
Slot_Curve_Angle(1,i)=atan2((a1-x1),b1);
end
Slot_Curve_Angle_deg=Slot_Curve_Angle/pi*180;
Slot_Curve_x_axis=r_rotation*sin(theta_engine_rad);
```

```
Slot_Curve=[];
Slot_Curve(1,1)=0;
for i=1:(theta_engine_rad_step-1)
delta_x=Slot_Curve_x_axis(1,i+1)-Slot_Curve_x_axis(1,i);
Slot_Curve(1,i+1)=Slot_Curve(1,i)+delta_x.*tan(Slot_Curve_Angle(1,i));
end
Slot_Curve_x_axis_power=Slot_Curve_x_axis;
Slot_Curve_power=Slot_Curve;
Slot_Curve_Angle_power=Slot_Curve_Angle;
Slot_Curve_Angle_deg_power=Slot_Curve_Angle_deg;
figure
plot(Slot_Curve_x_axis_power,Slot_Curve_power,'r','LineWidth',3)
title('Slot Curve power piston')
xlabel('x')
ylabel('Slot Curve, in m')
s_modified_scotch_yoke=[];
%Displacer Piston
engine_stroke=engine_displacer_stroke;
%Define/Initialize values
dwell_period=dwell_period_displacer;
moving_period=(1-dwell_period*2)/2;
Transition_period=Transition_period_displacer;
Transition_angle_number=floor(Transition_period/90*(theta_engine_rad_step*
moving_period));
r_rotation=engine_stroke/2/sin(moving_period*pi);
moving_angle_falling=theta_engine_rad(moving_period/2*theta_engine_rad_ste
p+1+Transition_angle_number:moving_period*3/2*theta_engine_rad_step+1-
Transition_angle_number);
moving_angle_falling_deg=moving_angle_falling/pi*180;
moving_angle_rising=theta_engine_rad(moving_period*5/2*theta_engine_rad_st
ep+1+Transition_angle_number:moving_period*7/2*theta_engine_rad_step+1-
Transition_angle_number);
moving_angle_rising_deg=moving_angle_rising/pi*180;
% Piston Position:-
x_piston_scotch_yoke_falling=r_rotation+r_rotation*cos(moving_angle_fallin
g)-(r_rotation-r_rotation*sin(moving_period*pi));
x_piston_scotch_yoke_rising=r_rotation+r_rotation*cos(moving_angle_rising)
-(r_rotation-r_rotation*sin(moving_period*pi));
v_piston_scotch_yoke_falling=-r_rotation*sin(moving_angle_falling);
```

```
v_piston_scotch_yoke_rising=-r_rotation*sin(moving_angle_rising);
a_piston_scotch_yoke_falling=-r_rotation*cos(moving_angle_falling);
a_piston_scotch_yoke_rising=-r_rotation*cos(moving_angle_rising);
j_piston_scotch_yoke_falling=r_rotation*sin(moving_angle_falling);
j_piston_scotch_yoke_rising=r_rotation*sin(moving_angle_rising);
%Transition between dwell and rising
%polynomial function transition rising
Beta_transition=Transition_period/180*pi;
polynomial_multiples=[1 1 1 1;4 5 6 7;12 20 30 42;24 60 120 210];
polynomial_multiples(2,:)=1/Beta_transition*polynomial_multiples(2,:);
polynomial_multiples(3,:)=1/(Beta_transition^2)*polynomial_multiples(3,:);
polynomial_multiples(4,:)=1/(Beta_transition^3)*polynomial_multiples(4,:);
C0=0;
polynomial_X(1,:)=r_rotation*cos((225+Transition_period)/180*pi)-
r_rotation*cos(225/180*pi)-C0;
polynomial_X(2,:)=-r_rotation*sin((225+Transition_period)/180*pi);
polynomial_X(3,:)=-r_rotation*cos((225+Transition_period)/180*pi);
polynomial_X(4,:)=r_rotation*sin((225+Transition_period)/180*pi);
C4_to_C7=inv(polynomial_multiples)*(polynomial_X);
%Transition angle rising
Transition_angle_rising=theta_engine_rad(moving_period*5/2*theta_engine_ra
d_step+1:moving_period*5/2*theta_engine_rad_step+Transition_angle_number);
Transition_angle_rising_deg=Transition_angle_rising/pi*180;
Transition_angle_rising_nom=Transition_angle_rising-
min(Transition_angle_rising);
Transition_angle_rising_nom_deg=Transition_angle_rising_nom/pi*180;
x_transition_rising_angle=Transition_angle_rising_nom/Beta_transition;
s_transition_rising=C4_to_C7(1,1)*x_transition_rising_angle.^4+C4_to_C7(2,
1)*x_transition_rising_angle.^5+C0;
s_transition_rising=s_transition_rising+C4_to_C7(3,1)*x_transition_rising_
angle.^6+C4_to_C7(4,1)*x_transition_rising_angle.^7;
v_transition_rising=C4_to_C7(1,1)*4*x_transition_rising_angle.^3+C4_to_C7(
2,1)*5*x_transition_rising_angle.^4;
v_transition_rising=v_transition_rising+C4_to_C7(3,1)*6*x_transition_risin
g_angle.^5+C4_to_C7(4,1)*7*x_transition_rising_angle.^6;
v_transition_rising=1/Beta_transition*v_transition_rising;
a_transition_rising=C4_to_C7(1,1)*12*x_transition_rising_angle.^2+C4_to_C7
(2,1)*20*x_transition_rising_angle.^3;
```

```
a_transition_rising=a_transition_rising+C4_to_C7(3,1)*30*x_transition_risi
ng_angle.^4+C4_to_C7(4,1)*42*x_transition_rising_angle.^5;
a_transition_rising=1/(Beta_transition^2)*a_transition_rising;
j_transition_rising=C4_to_C7(1,1)*24*x_transition_rising_angle+C4_to_C7(2,
1)*60*x_transition_rising_angle.^2;
j_transition_rising=j_transition_rising+C4_to_C7(3,1)*120*x_transition_ris
ing_angle.^3+C4_to_C7(4,1)*210*x_transition_rising_angle.^4;
j_transition_rising=1/(Beta_transition^3)*j_transition_rising;
%Transition between dwell and falling
%polynomial function transition falling
Beta_transition=Transition_period/180*pi;
polynomial_multiples=[1 1 1 1;4 5 6 7;12 20 30 42;24 60 120 210];
polynomial_multiples(2,:)=1/Beta_transition*polynomial_multiples(2,:);
polynomial_multiples(3,:)=1/(Beta_transition^2)*polynomial_multiples(3,:);
polynomial_multiples(4,:)=1/(Beta_transition^3)*polynomial_multiples(4,:);
C0=2*r_rotation*cos(45/180*pi);
polynomial_X(1,:)=r_rotation*cos((45+Transition_period)/180*pi)+r_rotation
*cos(45/180*pi)-C0;
polynomial_X(2,:)=-r_rotation*sin((45+Transition_period)/180*pi);
polynomial_X(3,:)=-r_rotation*cos((45+Transition_period)/180*pi);
polynomial_X(4,:)=r_rotation*sin((45+Transition_period)/180*pi);
C4_to_C7=inv(polynomial_multiples)*(polynomial_X);
%Transition angle transition falling
Transition_angle_falling=theta_engine_rad(moving_period*1/2*theta_engine_r
ad_step+1:moving_period*1/2*theta_engine_rad_step+Transition_angle_number)
;
Transition_angle_falling_deg=Transition_angle_falling/pi*180;
Transition_angle_falling_nom=Transition_angle_falling-
min(Transition_angle_falling);
Transition_angle_falling_nom_deg=Transition_angle_falling_nom/pi*180;
x_transition_falling_angle=Transition_angle_falling_nom/Beta_transition;
s_transition_falling=C4_to_C7(1,1)*x_transition_falling_angle.^4+C4_to_C7(
2,1)*x_transition_falling_angle.^5+C0;
s_transition_falling=s_transition_falling+C4_to_C7(3,1)*x_transition_falli
ng_angle.^6+C4_to_C7(4,1)*x_transition_falling_angle.^7;
v_transition_falling=C4_to_C7(1,1)*4*x_transition_falling_angle.^3+C4_to_C
7(2,1)*5*x_transition_falling_angle.^4;
v_transition_falling=v_transition_falling+C4_to_C7(3,1)*6*x_transition_fal
ling_angle.^5+C4_to_C7(4,1)*7*x_transition_falling_angle.^6;
```

```
v_transition_falling=1/Beta_transition*v_transition_falling;
a_transition_falling=C4_to_C7(1,1)*12*x_transition_falling_angle.^2+C4_to_
C7(2,1)*20*x_transition_falling_angle.^3;
a_transition_falling=a_transition_falling+C4_to_C7(3,1)*30*x_transition_fa
lling_angle.^4+C4_to_C7(4,1)*42*x_transition_falling_angle.^5;
a_transition_falling=1/(Beta_transition^2)*a_transition_falling;
j_transition_falling=C4_to_C7(1,1)*24*x_transition_falling_angle+C4_to_C7(
2,1)*60*x_transition_falling_angle.^2;
j_transition_falling=j_transition_falling+C4_to_C7(3,1)*120*x_transition_f
alling_angle.^3+C4_to_C7(4,1)*210*x_transition_falling_angle.^4;
j_transition_falling=1/(Beta_transition^3)*j_transition_falling;
%Transition between rising and dwell
%polynomial function transition rising2
Beta_transition=Transition_period/180*pi;
polynomial_multiples=[1 1 1 1;4 5 6 7;12 20 30 42;24 60 120 210];
polynomial_multiples(2,:)=1/Beta_transition*polynomial_multiples(2,:);
polynomial_multiples(3,:)=1/(Beta_transition^2)*polynomial_multiples(3,:);
polynomial_multiples(4,:)=1/(Beta_transition^3)*polynomial_multiples(4,:);
C0=r_rotation*cos((315-
Transition_period)/180*pi)+r_rotation*cos(45/180*pi);
C1=Beta_transition*(-r_rotation*sin((315-Transition_period)/180*pi));
C2=(Beta_transition^2)/2*(-r_rotation*cos((315-
Transition_period)/180*pi));
C3=(Beta_transition^3)/6*(r_rotation*sin((315-Transition_period)/180*pi));
polynomial_X(1,:)=2*r_rotation*cos(45/180*pi)-C0-C1-C2-C3;
polynomial_X(2,:)=0-C1/Beta_transition-2*C2/Beta_transition-
3*C3/Beta_transition;
polynomial_X(3,:)=0-2*C2/(Beta_transition^2)-6*C3/(Beta_transition^2);
polynomial_X(4,:)=0-6*C3/(Beta_transition^3);
C4_to_C7=inv(polynomial_multiples)*(polynomial_X);
%Transition angle rising2
Transition_angle_rising2=theta_engine_rad(moving_period*7/2*theta_engine_r
ad_step-
Transition_angle_number+2:moving_period*7/2*theta_engine_rad_step+1);
Transition_angle_rising2_deg=Transition_angle_rising2/pi*180;
Transition_angle_rising2_nom=Transition_angle_rising2-
(max(Transition_angle_rising2_deg)-Transition_period)/180*pi;
Transition_angle_rising2_nom_deg=Transition_angle_rising2_nom/pi*180;
x_transition_rising2_angle=Transition_angle_rising2_nom/Beta_transition;
```

```
s_transition_rising2=C4_to_C7(1,1)*x_transition_rising2_angle.^4+C4_to_C7(
2,1)*x_transition_rising2_angle.^5+C0+C1*x_transition_rising2_angle+C2*x_t
ransition_rising2_angle.^2;
s_transition_rising2=s_transition_rising2+C4_to_C7(3,1)*x_transition_risin
g2_angle.^6+C4_to_C7(4,1)*x_transition_rising2_angle.^7+C3*x_transition_ri
sing2_angle.^3;
v_transition_rising2=C4_to_C7(1,1)*4*x_transition_rising2_angle.^3+C4_to_C
7(2,1)*5*x_transition_rising2_angle.^4+C1+2*C2*x_transition_rising2_angle;
v_transition_rising2=v_transition_rising2+C4_to_C7(3,1)*6*x_transition_ris
ing2_angle.^5+C4_to_C7(4,1)*7*x_transition_rising2_angle.^6+3*C3*x_transit
ion_rising2_angle.^2;
v_transition_rising2=1/Beta_transition*v_transition_rising2;
a_transition_rising2=C4_to_C7(1,1)*12*x_transition_rising2_angle.^2+C4_to_
C7(2,1)*20*x_transition_rising2_angle.^3+2*C2+6*C3*x_transition_rising2_an
gle;
a_transition_rising2=a_transition_rising2+C4_to_C7(3,1)*30*x_transition_ri
sing2_angle.^4+C4_to_C7(4,1)*42*x_transition_rising2_angle.^5;
a_transition_rising2=1/(Beta_transition^2)*a_transition_rising2;
j_transition_rising2=C4_to_C7(1,1)*24*x_transition_rising2_angle+C4_to_C7(
2,1)*60*x_transition_rising2_angle.^2+6*C3;
j_transition_rising2=j_transition_rising2+C4_to_C7(3,1)*120*x_transition_r
ising2_angle.^3+C4_to_C7(4,1)*210*x_transition_rising2_angle.^4;
j_transition_rising2=1/(Beta_transition^3)*j_transition_rising2;
%Transition between falling and dwell
%polynomial function transition falling2
Beta_transition=Transition_period/180*pi;
polynomial_multiples=[1 1 1 1;4 5 6 7;12 20 30 42;24 60 120 210];
polynomial_multiples(2,:)=1/Beta_transition*polynomial_multiples(2,:);
polynomial_multiples(3,:)=1/(Beta_transition^2)*polynomial_multiples(3,:);
polynomial_multiples(4,:)=1/(Beta_transition^3)*polynomial_multiples(4,:);
C0=r_rotation*cos((135-Transition_period)/180*pi)-
r_rotation*cos(135/180*pi);
C1=Beta_transition*(-r_rotation*sin((135-Transition_period)/180*pi));
C2=(Beta_transition^2)/2*(-r_rotation*cos((135-
Transition_period)/180*pi));
C3=(Beta_transition^3)/6*(r_rotation*sin((135-Transition_period)/180*pi));
polynomial_X(1,:)=0-C0-C1-C2-C3;
polynomial_X(2,:)=0-C1/Beta_transition-2*C2/Beta_transition-
3*C3/Beta_transition;
```

```
polynomial_X(3,:)=0-2*C2/(Beta_transition^2)-6*C3/(Beta_transition^2);
polynomial_X(4,:)=0-6*C3/(Beta_transition^3);
C4_to_C7=inv(polynomial_multiples)*(polynomial_X);
%Transition angle falling2
Transition_angle_falling2=theta_engine_rad(moving_period*3/2*theta_engine_
rad_step-
Transition_angle_number+2:moving_period*3/2*theta_engine_rad_step+1);
Transition_angle_falling2_deg=Transition_angle_falling2/pi*180;
Transition_angle_falling2_nom=Transition_angle_falling2-
(max(Transition_angle_falling2_deg)-Transition_period)/180*pi;
Transition_angle_falling2_nom_deg=Transition_angle_falling2_nom/pi*180;
x_transition_falling2_angle=Transition_angle_falling2_nom/Beta_transition;
s_transition_falling2=C4_to_C7(1,1)*x_transition_falling2_angle.^4+C4_to_C
7(2,1)*x_transition_falling2_angle.^5+C0+C1*x_transition_falling2_angle+C2
*x_transition_falling2_angle.^2;
s_transition_falling2=s_transition_falling2+C4_to_C7(3,1)*x_transition_fal
ling2_angle.^6+C4_to_C7(4,1)*x_transition_falling2_angle.^7+C3*x_transitio
n_falling2_angle.^3;
v_transition_falling2=C4_to_C7(1,1)*4*x_transition_falling2_angle.^3+C4_to
_C7(2,1)*5*x_transition_falling2_angle.^4+C1+2*C2*x_transition_falling2_an
gle;
v_transition_falling2=v_transition_falling2+C4_to_C7(3,1)*6*x_transition_f
alling2_angle.^5+C4_to_C7(4,1)*7*x_transition_falling2_angle.^6+3*C3*x_tra
nsition_falling2_angle.^2;
v_transition_falling2=1/Beta_transition*v_transition_falling2;
a_transition_falling2=C4_to_C7(1,1)*12*x_transition_falling2_angle.^2+C4_t
o_C7(2,1)*20*x_transition_falling2_angle.^3+2*C2+6*C3*x_transition_falling
2_angle;
a_transition_falling2=a_transition_falling2+C4_to_C7(3,1)*30*x_transition_
falling2_angle.^4+C4_to_C7(4,1)*42*x_transition_falling2_angle.^5;
a_transition_falling2=1/(Beta_transition^2)*a_transition_falling2;
j_transition_falling2=C4_to_C7(1,1)*24*x_transition_falling2_angle+C4_to_C
7(2,1)*60*x_transition_falling2_angle.^2+6*C3;
j_transition_falling2=j_transition_falling2+C4_to_C7(3,1)*120*x_transition
_falling2_angle.^3+C4_to_C7(4,1)*210*x_transition_falling2_angle.^4;
j_transition_falling2=1/(Beta_transition^3)*j_transition_falling2;
dwell_period_size=theta_engine_rad_step*dwell_period;
dwell_1_half=zeros(1,dwell_period_size/2);
dwell_1_half2=zeros(1,dwell_period_size/2-1);
```

```
dwell_1_s_half=dwell_1_half+engine_stroke;
dwell_1_s_half2=dwell_1_half2+engine_stroke;
dwell_2=zeros(1,dwell_period_size-1);
s_modified_scotch_yoke=[dwell_2 s_transition_rising
x_piston_scotch_yoke_rising];
s_modified_scotch_yoke=[s_modified_scotch_yoke s_transition_rising2
dwell_1_s_half2 dwell_1_s_half];
s_modified_scotch_yoke=[s_modified_scotch_yoke s_transition_falling
x_piston_scotch_yoke_falling s_transition_falling2];
v_modified_scotch_yoke=[dwell_2 v_transition_rising
v_piston_scotch_yoke_rising];
v_modified_scotch_yoke=[v_modified_scotch_yoke v_transition_rising2
dwell_1_half2 dwell_1_half];
v_modified_scotch_yoke=[v_modified_scotch_yoke v_transition_falling
v_piston_scotch_yoke_falling v_transition_falling2];
a_modified_scotch_yoke=[dwell_2 a_transition_rising
a_piston_scotch_yoke_rising];
a_modified_scotch_yoke=[a_modified_scotch_yoke a_transition_rising2
dwell_1_half2 dwell_1_half];
a_modified_scotch_yoke=[a_modified_scotch_yoke a_transition_falling
a_piston_scotch_yoke_falling a_transition_falling2];
j_modified_scotch_yoke=[dwell_2 j_transition_rising
j_piston_scotch_yoke_rising];
j_modified_scotch_yoke=[j_modified_scotch_yoke j_transition_rising2
dwell_1_half2 dwell_1_half];
j_modified_scotch_yoke=[j_modified_scotch_yoke j_transition_falling
j_piston_scotch_yoke_falling j_transition_falling2];
x_piston_displacer=s_modified_scotch_yoke;
v_piston_displacer=v_modified_scotch_yoke; %length/rad
j_piston_displacer=j_modified_scotch_yoke;
%Stirling Engine Piston Motion Display
figure
plot(theta_engine_deg,x_piston_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,x_piston_displacer,'r','LineWidth',3)
title('s diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('s (length in meter)')
legend('power piston','displacer')
```

```
figure
plot(theta_engine_deg,v_piston_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,v_piston_displacer,'r','LineWidth',3)
title('v diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('v (m/s)')
legend('power piston','displacer')
figure
plot(theta_engine_deg,j_piston_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,j_piston_displacer,'r','LineWidth',3)
title('j diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('j (m/s3)')
legend('power piston','displacer')
%Stirling Engine Characteristics Initialisation
%Define/Initialize values for engine characteristics
T_regenerator=(T_h-T_k)./(log(T_h./T_k));
r_piston_power=Diameter_piston_power/2; %in meter;
r_piston_displacer=Diameter_piston_displacer/2; %in meter
%V_sw_hot=120.88*10^(-6); %Swept Volume of expansion piston
%V_sw_power=113.14*10^(-6); %Swept Volume of compression piston
V_sw_hot=pi*(r_piston_displacer^2)*engine_displacer_stroke;
V_sw_cold=V_sw_hot-pi*(r_displacer_drive_rod^2)*engine_displacer_stroke;
V_sw_power=pi*(r_piston_power^2)*engine_power_stroke;
Total_volume_engine_ambient=V_sw_power+V_power_clearance+V_r+V_k+V_h+V_sw_
cold+V_displacer_clearance_cold;
M_gas_engine_initial=density_gas*Total_volume_engine_ambient; %in kg
%Stirling Engine Piston Volume Calculation
V_hot_displacer=pi*(r_piston_displacer^2)*x_piston_displacer;
V_power=pi*(r_piston_power^2)*x_piston_power;
V_cold_displacer=V_sw_cold-pi*(r_piston_displacer^2-
r_displacer_drive_rod^2)*x_piston_displacer;
dV_hot_displacer=pi*(r_piston_displacer^2)*v_piston_displacer;
V_cold_engine=V_power+V_cold_displacer;
dV_cold_engine=pi*(r_piston_power^2)*v_piston_power-
pi*(r_piston_displacer^2-r_displacer_drive_rod^2)*v_piston_displacer;
%Integrate into Schmidt Model
```

```
V_hot=V_h+V_displacer_clearance_hot+V_hot_displacer;
dV_hot=dV_hot_displacer;
V_cold=V_k+V_displacer_clearance_cold+V_power_clearance+V_cold_engine;
dV_cold=dV_cold_engine;
%Integrate into Urieli Model
V_exp=V_displacer_clearance_hot+V_hot_displacer;
dV_exp=dV_hot_displacer;
V_comp=V_displacer_clearance_cold+V_power_clearance+V_cold_engine;
dV_comp=dV_cold_engine;
%Schmidt Model for Initial value prediction
P_engine_Schmidt=M_gas_engine_initial*R_spec_gas./(V_cold/T_k+V_r/T_regene
rator+V_hot/T_h);
m_regenerator_Schmidt=V_r*P_engine_Schmidt*(log(T_h/T_k))/(R_spec_gas*(T_h
-T_k));
%Note : W_engine=Integral of (P*(d_V_c/d_theta+d_V_e/d_theta)*d_theta)
Volume_engine_Schmidt=V_hot+V_cold+V_r;
%Numerical differention (in progress)
%(f(x+h)-f(x))/h, h=step value
%Numerical Integration - Simpson's Rule (in progress)
%S_n=h/3*(f(x0)+4f(x_1)+2f(x_2)+4f(x_3)+....2f(x_(n-2))+4f(x_(n-
1))+f(x_n))
Simpson_multiple=[];
for i=1:((theta_engine_rad_step)/2)
Simpson_multiple=[Simpson_multiple;2;4];
end
Simpson_multiple=[Simpson_multiple;1];
Simpson_multiple(1,1)=1;
Work_engine_part1=P_engine_Schmidt.*(dV_cold+dV_hot);
Work_engine_part1=[Work_engine_part1 Work_engine_part1(1,1)];
h=2*pi/theta_engine_rad_step;
Work_engine_Schmidt=h/3*Work_engine_part1*Simpson_multiple;
Words = 'Engine Work in J per revolution (Schmidt)';
disp(Words)
disp(Work_engine_Schmidt)
Work_done_Schmidt_power=Work_engine_Schmidt*w_engine_rpm/60;
Words = 'Engine Work in W (Schmidt)';
disp(Words)
disp(Work_done_Schmidt_power)
Work_engine_part2=P_engine_Schmidt.*(dV_hot);
```

```
Work_engine_part2=[Work_engine_part2 Work_engine_part2(1,1)];
h=2*pi/theta_engine_rad_step;
Work_engine_Schmidt_hot=h/3*Work_engine_part2*Simpson_multiple;
Q_hot_Schmidt=Work_engine_Schmidt_hot;
Engine_Efficiency_Schmidt=Work_engine_Schmidt/Q_hot_Schmidt;
Words = 'Engine Efficiency (Schmidt)';
disp(Words)
disp(Engine_Efficiency_Schmidt)
%Urieli model - ODEs in adiabatic analysis
%Separate odd and even rows for rk4 method 1/2step calculation
theta_engine_rad_step=theta_engine_rad_step/2;
theta_engine_rad_step_size=theta_engine_rad_step_size*2;
for i=1:theta_engine_rad_step
V_exp_odd(1,i)=V_exp(1,i*2-1);
dV_exp_odd(1,i)=dV_exp(1,i*2-1);
V_comp_odd(1,i)=V_comp(1,i*2-1);
dV_comp_odd(1,i)=dV_comp(1,i*2-1);
V_exp_even(1,i)=V_exp(1,i*2);
dV_exp_even(1,i)=dV_exp(1,i*2);
V_comp_even(1,i)=V_comp(1,i*2);
dV_comp_even(1,i)=dV_comp(1,i*2);
theta_engine_deg_odd(1,i)=theta_engine_deg(1,i*2-1);
end
V_exp=V_exp_odd;
dV_exp=dV_exp_odd;
V_comp=V_comp_odd;
dV_comp=dV_comp_odd;
figure
plot(theta_engine_deg_odd,V_exp,'r','LineWidth',3)
title('Expansion Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
figure
plot(theta_engine_deg_odd,V_comp,'r','LineWidth',3)
title('Compression Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
figure
plot(theta_engine_deg_odd,dV_exp,'g','LineWidth',3)
```

```
hold on
plot(theta_engine_deg_odd,dV_comp,'r','LineWidth',3)
title('Volume Change')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume Change (m3/s)')
%Initialize values
Temperature_term1=V_k/T_k+V_r/T_regenerator+V_h/T_h;
T_comp=ones(1,theta_engine_rad_step+1).*T_k;
T_exp=ones(1,theta_engine_rad_step+1).*T_h;
T_ck=T_comp(1,1);
T_he=T_exp(1,1);
Q_h=zeros(1,theta_engine_rad_step);
Q_k=zeros(1,theta_engine_rad_step);
Q_r=zeros(1,theta_engine_rad_step);
W_exp=zeros(1,theta_engine_rad_step);
W_comp=zeros(1,theta_engine_rad_step);
T_kr=T_k;
T_rh=T_h;
for j=1:50
for i=1:(theta_engine_rad_step)
if i==1
M_gas_engine=M_gas_engine_initial;
end
P_engine_Urieli(1,i)=M_gas_engine*R_spec_gas/(V_comp(1,i)/T_comp(1,i)+Temp
erature_term1+V_exp(1,i)/T_exp(1,i));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli(1,i)*(dV_comp(1,i)/T_ck+dV_exp(1,i)/T_he);
dP_engine_Urieli_term2=V_comp(1,i)/T_ck+k_isentropic*Temperature_term1+V_e
xp(1,i)/T_he;
dP_engine_Urieli(1,i)=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli(1,i)*V_comp(1,i)/(R_spec_gas*T_comp(1,i));
m_k=P_engine_Urieli(1,i)*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli(1,i)*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli(1,i)*V_h/(R_spec_gas*T_h);
m_exp=P_engine_Urieli(1,i)*V_exp(1,i)/(R_spec_gas*T_exp(1,i));
dm_exp(1,i)=(P_engine_Urieli(1,i)*dV_exp(1,i)+V_exp(1,i)*dP_engine_Urieli(
1,i)/k_isentropic)/(R_spec_gas*T_he);
dm_comp(1,i)=(P_engine_Urieli(1,i)*dV_comp(1,i)+V_comp(1,i)*dP_engine_Urie
li(1,i)/k_isentropic)/(R_spec_gas*T_ck);
```

```
dm_k(1,i)=m_k*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
dm_r(1,i)=m_r*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
dm_h(1,i)=m_h*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
m_dot_ck=-dm_comp(1,i);
m_dot_he=dm_exp(1,i);
m_dot_kr=m_dot_ck-dm_k(1,i);
m_dot_rh=m_dot_he+dm_h(1,i);
if m_dot_ck>0
T_ck=T_comp(1,i);
else
T_ck=T_k;
end
if m_dot_he>0
T_he=T_h;
else
T_he=T_exp(1,i);
end
dT_exp=T_exp(1,i)*(dP_engine_Urieli(1,i)/P_engine_Urieli(1,i)+dV_exp(1,i)/
V_exp(1,i)-dm_exp(1,i)/m_exp);
dT_comp=T_comp(1,i)*(dP_engine_Urieli(1,i)/P_engine_Urieli(1,i)+dV_comp(1,
i)/V_comp(1,i)-dm_comp(1,i)/m_comp);
dQ_k=V_k*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_ck*m_dot_ck-
T_kr*m_dot_kr);
dQ_r=V_r*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_kr*m_dot_kr-
T_rh*m_dot_rh);
dQ_h=V_h*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_rh*m_dot_rh-
T_he*m_dot_he);
dW_exp=P_engine_Urieli(1,i)*dV_exp(1,i);
dW_comp=P_engine_Urieli(1,i)*dV_comp(1,i);
k1_exp=theta_engine_rad_step_size*dT_exp;
k1_comp=theta_engine_rad_step_size*dT_comp;
k1_dQk=theta_engine_rad_step_size*dQ_k;
k1_dQr=theta_engine_rad_step_size*dQ_r;
k1_dQh=theta_engine_rad_step_size*dQ_h;
k1_dWe=theta_engine_rad_step_size*dW_exp;
k1_dWc=theta_engine_rad_step_size*dW_comp;
T_ck_half=T_ck;
T_he_half=T_he;
```

```
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp_even(1,i)/(T_comp(1,i
)+0.5*k1_comp)+Temperature_term1+V_exp_even(1,i)/(T_exp(1,i)+0.5*k1_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp_even(1,i)/T_ck_half+dV_exp_even
(1,i)/T_he_half);
dP_engine_Urieli_term2=V_comp_even(1,i)/T_ck_half+k_isentropic*Temperature
_term1+V_exp_even(1,i)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp_even(1,i)/(R_spec_gas*(T_comp(1,i)+0.5*
k1_comp));
m_exp=P_engine_Urieli_half*V_exp_even(1,i)/(R_spec_gas*(T_exp(1,i)+0.5*k1_
exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp_even(1,i)+V_exp_even(1,i)*dP_engi
ne_Urieli_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp_even(1,i)+V_comp_even(1,i)*dP_e
ngine_Urieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+0.5*k1_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+0.5*k1_exp;
end
dT_exp=(T_exp(1,i)+0.5*k1_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half
+dV_exp_even(1,i)/V_exp_even(1,i)-dm_exp_half/m_exp);
k2_exp=theta_engine_rad_step_size*dT_exp;
```

```
dT_comp=(T_comp(1,i)+0.5*k1_comp)*(dP_engine_Urieli_half/P_engine_Urieli_h
alf+dV_comp_even(1,i)/V_comp_even(1,i)-dm_comp_half/m_comp);
k2_comp=theta_engine_rad_step_size*dT_comp;
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp_even(1,i);
dW_comp=P_engine_Urieli_half*dV_comp_even(1,i);
k2_dQk=theta_engine_rad_step_size*dQ_k;
k2_dQr=theta_engine_rad_step_size*dQ_r;
k2_dQh=theta_engine_rad_step_size*dQ_h;
k2_dWe=theta_engine_rad_step_size*dW_exp;
k2_dWc=theta_engine_rad_step_size*dW_comp;
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp_even(1,i)/(T_comp(1,i
)+0.5*k2_comp)+Temperature_term1+V_exp_even(1,i)/(T_exp(1,i)+0.5*k2_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp_even(1,i)/T_ck_half+dV_exp_even
(1,i)/T_he_half);
dP_engine_Urieli_term2=V_comp_even(1,i)/T_ck_half+k_isentropic*Temperature
_term1+V_exp_even(1,i)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp_even(1,i)/(R_spec_gas*(T_comp(1,i)+0.5*
k2_comp));
m_exp=P_engine_Urieli_half*V_exp_even(1,i)/(R_spec_gas*(T_exp(1,i)+0.5*k2_
exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp_even(1,i)+V_exp_even(1,i)*dP_engi
ne_Urieli_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp_even(1,i)+V_comp_even(1,i)*dP_e
ngine_Urieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
```

```
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+0.5*k2_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+0.5*k2_exp;
end
dT_exp=(T_exp(1,i)+0.5*k2_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half
+dV_exp_even(1,i)/V_exp_even(1,i)-dm_exp_half/m_exp);
k3_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+0.5*k2_comp)*(dP_engine_Urieli_half/P_engine_Urieli_h
alf+dV_comp_even(1,i)/V_comp_even(1,i)-dm_comp_half/m_comp);
k3_comp=theta_engine_rad_step_size*dT_comp;
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp_even(1,i);
dW_comp=P_engine_Urieli_half*dV_comp_even(1,i);
k3_dQk=theta_engine_rad_step_size*dQ_k;
k3_dQr=theta_engine_rad_step_size*dQ_r;
k3_dQh=theta_engine_rad_step_size*dQ_h;
k3_dWe=theta_engine_rad_step_size*dW_exp;
k3_dWc=theta_engine_rad_step_size*dW_comp;
i2=i+1;
if i2>theta_engine_rad_step
i2=i2-theta_engine_rad_step;
end
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp(1,i2)/(T_comp(1,i)+k3
_comp)+Temperature_term1+V_exp(1,i2)/(T_exp(1,i)+k3_exp));
```

```
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp(1,i2)/T_ck_half+dV_exp(1,i2)/T_
he_half);
dP_engine_Urieli_term2=V_comp(1,i2)/T_ck_half+k_isentropic*Temperature_ter
m1+V_exp(1,i2)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp(1,i2)/(R_spec_gas*(T_comp(1,i)+k3_comp)
);
m_exp=P_engine_Urieli_half*V_exp(1,i2)/(R_spec_gas*(T_exp(1,i)+k3_exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp(1,i2)+V_exp(1,i2)*dP_engine_Uriel
i_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp(1,i2)+V_comp(1,i2)*dP_engine_Ur
ieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+k3_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+k3_exp;
end
dT_exp=(T_exp(1,i)+k3_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half+dV_
exp(1,i2)/V_exp(1,i2)-dm_exp_half/m_exp);
k4_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+k3_comp)*(dP_engine_Urieli_half/P_engine_Urieli_half+
dV_comp(1,i2)/V_comp(1,i2)-dm_comp_half/m_comp);
k4_comp=theta_engine_rad_step_size*dT_comp;
```

```
T_exp(1,i+1)=T_exp(1,i)+1/6*(k1_exp+2*k2_exp+2*k3_exp+k4_exp);
T_comp(1,i+1)=T_comp(1,i)+1/6*(k1_comp+2*k2_comp+2*k3_comp+k4_comp);
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp(1,i2);
dW_comp=P_engine_Urieli_half*dV_comp(1,i2);
k4_dQk=theta_engine_rad_step_size*dQ_k;
k4_dQr=theta_engine_rad_step_size*dQ_r;
k4_dQh=theta_engine_rad_step_size*dQ_h;
k4_dWe=theta_engine_rad_step_size*dW_exp;
k4_dWc=theta_engine_rad_step_size*dW_comp;
Q_k(1,i+1)=Q_k(1,i)+1/6*(k1_dQk+2*k2_dQk+2*k3_dQk+k4_dQk);
Q_r(1,i+1)=Q_r(1,i)+1/6*(k1_dQr+2*k2_dQr+2*k3_dQr+k4_dQr);
Q_h(1,i+1)=Q_h(1,i)+1/6*(k1_dQh+2*k2_dQh+2*k3_dQh+k4_dQh);
W_exp(1,i+1)=W_exp(1,i)+1/6*(k1_dWe+2*k2_dWe+2*k3_dWe+k4_dWe);
W_comp(1,i+1)=W_comp(1,i)+1/6*(k1_dWc+2*k2_dWc+2*k3_dWc+k4_dWc);
if i==theta_engine_rad_step
T_exp_diff(1,j)=T_exp(1,theta_engine_rad_step+1)-T_exp(1,1);
T_exp(1,1)=T_exp(1,theta_engine_rad_step+1);
T_exp(:,theta_engine_rad_step+1)=[];
T_comp_diff(1,j)=T_comp(1,theta_engine_rad_step+1)-T_comp(1,1);
T_comp(1,1)=T_comp(1,theta_engine_rad_step+1);
T_comp(:,theta_engine_rad_step+1)=[];
end
end
end
Work_done_Urieli=W_exp(1,theta_engine_rad_step+1)+W_comp(1,theta_engine_ra
d_step+1);
Engine_Efficiency_Urieli=Work_done_Urieli/Q_h(1,theta_engine_rad_step+1);
Words = 'Engine Work in J per revolution (Urieli)';
disp(Words)
disp(Work_done_Urieli)
Work_done_Urieli_power=Work_done_Urieli*w_engine_rpm/60;
Words = 'Engine Work in W (Urieli)';
disp(Words)
disp(Work_done_Urieli_power)
Words = 'Engine Efficiency Urieli';
disp(Words)
disp(Engine_Efficiency_Urieli)
% PV Diagram Display:
Volume_engine_Urieli=V_comp+V_exp+V_r+V_h+V_k;
figure
plot(Volume_engine_Schmidt,P_engine_Schmidt,'g','LineWidth',3)
hold on
plot(Volume_engine_Urieli,P_engine_Urieli,'r','LineWidth',3)
title('PV Diagram')
xlabel('Volume')
ylabel('Pressure')
legend('Isothermal','Adiabatic')
figure
plot(theta_engine_deg,P_engine_Schmidt,'g','LineWidth',3)
hold on
plot(theta_engine_deg_odd,P_engine_Urieli,'r','LineWidth',3)
title('Engine Pressure')
xlabel('x, shaft angle 0 to 360deg')
ylabel('P (Pressure in Pa)')
legend('Isothermal','Adiabatic')
figure
plot(theta_engine_deg,Volume_engine_Schmidt,'r','LineWidth',3)
%hold on
%plot(theta_engine_deg_odd,Volume_engine_Urieli,'r','LineWidth',3)
title('Engine Internal Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
% Gas Torque Calculation
P_engine=P_engine_Schmidt;
shifting_size=size(dwell_1_s_half);
Slot_Curve_x_axis_power1=circshift(Slot_Curve_x_axis_power,-
shifting_size(1,2));
Slot_Curve_x_axis_power2=[Slot_Curve_x_axis_power1
Slot_Curve_x_axis_power1(1,1)];
Slot_Curve_power1=circshift(Slot_Curve_power,-shifting_size(1,2));
Slot_Curve_power2=[Slot_Curve_power1 Slot_Curve_power1(1,1)];
Slot_Curve_y_axis=r_rotation*cos(theta_engine_rad);
```

```
Slot_Curve_y_axis_power1=circshift(Slot_Curve_y_axis,-shifting_size(1,2));
Slot_Curve_gradient_power(1,1)=0;
for i=1:(theta_engine_rad_step*2)
Slot_Curve_gradient_power(1,i)=(Slot_Curve_power2(1,i+1)-
Slot_Curve_power2(1,i))/(Slot_Curve_x_axis_power2(1,i+1)-
Slot_Curve_x_axis_power2(1,i));
Slot_Curve_gradient_angle_power(1,i)=atan(Slot_Curve_gradient_power(1,i));
Slot_Curve_gradient_angle_power_deg(1,i)=Slot_Curve_gradient_angle_power(1
,i)/pi*180;
Gas_Force_y_power_MSY(1,i)=P_engine(1,i)*pi*(Diameter_piston_power^2)/4;
Gas_Torque_y_power_MSY(1,i)=Gas_Force_y_power_MSY(1,i)*Slot_Curve_x_axis_p
ower2(1,i);
%Resultant_force_power_angle(1,i)=pi/2+Slot_Curve_gradient_angle_power(1,i
);
%Gas_Force_x_power_MSY(1,i)=Gas_Force_y_power_MSY(1,i)/sin(Slot_Curve_grad
ient_angle_power(1,i))*cos(Slot_Curve_gradient_angle_power(1,i));
%Gas_Torque_x_power_MSY(1,i)=-
Gas_Force_x_power_MSY(1,i)*Slot_Curve_y_axis_power1(1,i);
end
figure
plot(theta_engine_deg,Slot_Curve_gradient_angle_power_deg,'r','LineWidth',
3)
title('Slot Curve Gradient Angle')
xlabel('Engine Angle')
ylabel('Gradient Angle')
figure
plot(theta_engine_deg,Gas_Torque_y_power_MSY,'r','LineWidth',3)
title('Gas Torque Exerted by Power Piston vertical force')
xlabel('Engine Angle')
ylabel('Gas Torque by Power Piston vertical force')
shifting_size=size([dwell_1_s_half s_transition_falling
x_piston_scotch_yoke_falling s_transition_falling2]);
Slot_Curve_x_axis_displacer1=circshift(Slot_Curve_x_axis_power,-
shifting_size(1,2));
Slot_Curve_x_axis_displacer2=[Slot_Curve_x_axis_displacer1
Slot_Curve_x_axis_displacer1(1,1)];
Slot_Curve_displacer1=circshift(Slot_Curve_power,-shifting_size(1,2));
Slot_Curve_displacer2=[Slot_Curve_displacer1 Slot_Curve_displacer1(1,1)];
Slot_Curve_y_axis=r_rotation*cos(theta_engine_rad);
```

```
Slot_Curve_y_axis_displacer1=circshift(Slot_Curve_y_axis,-
shifting_size(1,2));
Slot_Curve_gradient_displacer(1,1)=0;
for i=1:(theta_engine_rad_step*2)
Slot_Curve_gradient_displacer(1,i)=(Slot_Curve_displacer2(1,i+1)-
Slot_Curve_displacer2(1,i))/(Slot_Curve_x_axis_displacer2(1,i+1)-
Slot_Curve_x_axis_displacer2(1,i));
Slot_Curve_gradient_angle_displacer(1,i)=atan(Slot_Curve_gradient_displace
r(1,i));
Slot_Curve_gradient_angle_displacer_deg(1,i)=Slot_Curve_gradient_angle_dis
placer(1,i)/pi*180;
Gas_Force_y_displacer_MSY(1,i)=P_engine(1,i)*pi*(Diameter_piston_displacer
^2)/4;
Gas_Torque_y_displacer_MSY(1,i)=Gas_Force_y_displacer_MSY(1,i)*Slot_Curve_
x_axis_displacer2(1,i);
end
figure
plot(theta_engine_deg,Gas_Torque_y_displacer_MSY,'r','LineWidth',3)
title('Gas Torque Exerted by displacer Piston vertical force')
xlabel('Engine Angle')
ylabel('Gas Torque by displacer Piston vertical force')
Gas_Torque_y_MSY=Gas_Torque_y_displacer_MSY+Gas_Torque_y_power_MSY;
figure
plot(theta_engine_deg,Gas_Torque_y_MSY,'r','LineWidth',3)
title('Gas Torque Exerted by Pistons vertical force')
xlabel('Engine Angle')
ylabel('Gas Torque by Pistons vertical force')
```

APPENDIX F: Computer Simulation Program 3b - Matlab Code for Stirling engine with slider-crank simulation to compare with MSY

```
%Clear all the previous data
clear all
close all
clc
%User Defined Parameters
engine_power_stroke=0.035;
engine_displacer_stroke=0.035;
Diameter_piston_power=0.0346;
Diameter_piston_displacer=0.040;
w_engine=2*pi*1.8; %w=2(pi)(frequency)
theta_engine_rad_step=4000;
theta_engine_rad_step_original=theta_engine_rad_step;
L_conrod=0.15; %Slider-crank conrod length
T_h = 200+273; %degree K, heater temp.
T_k = 40+273; %degree K, cooler temp.
T_0 = 25+273; %degree K, ambient temp.
r_displacer_drive_rod=0.008/2; %in meter
V_r=(1)*10^(-6); %Regenerator Dead Volume
V_k=(1)*10^(-6); %Cooler Dead Volume
V_h=(800)*10^(-6); %Heater Dead Volume
V_displacer_clearance_hot=800*10^(-8);
V_displacer_clearance_cold=800*10^(-8);
V_power_clearance=0*10^(-9);
R_spec_gas=287.052874; %J/g/K*1000=J/kg/K, specific gas constant of air at
ambient
density_gas=1.293; %kg/m3; at ambient
%Specific Heat Capacity
%Air
Molar_mass_gas=28.97; %kg/kmol
a_cp=28.11;
b_cp=0.1967*10^(-2);
c_cp=0.4802*10^(-5);
d_cp=-1.966*10^(-9);
T_gas=(T_h+T_k)/2; %in degree K
C_p_gas=(a_cp+b_cp*T_gas+c_cp*T_gas^2+d_cp*T_gas^3); %in kJ/(kmol*K),
ideal gas, from 273K to 1800K, at ambient
```

```
C_p_gas=C_p_gas/Molar_mass_gas*1000; %in J/(kg*K)
C_v_gas=C_p_gas-R_spec_gas;
k_isentropic=C_p_gas/C_v_gas; %gamma or k, index of isentropic
%Define/Initialize values for engine kinematics
w_engine_rpm=w_engine/2/pi*60;
theta_engine_rad_step_size=2*pi/theta_engine_rad_step;
theta_engine_rad=0;
for i=1:(theta_engine_rad_step-1)
theta_engine_rad=[theta_engine_rad
theta_engine_rad(1,i)+theta_engine_rad_step_size];
end
theta_engine_deg=theta_engine_rad/pi*180;
%Crank slider Analysis
%Power Piston Crank Slider
%Define/Initialize values
engine_stroke=engine_power_stroke;
r_crank=engine_stroke/2;
w_crank=w_engine;
theta_crank_rad=theta_engine_rad;
% Piston Position:-
x_piston_crank=r_crank*cos(theta_crank_rad)+L_conrod*((1-
(r_crank/L_conrod*sin(theta_crank_rad)).^2).^(1/2));
%Assume steady state analysis where w_crank is constant:-
v_piston_crank=-
r_crank*w_crank.*(sin(theta_crank_rad)+r_crank/2/L_conrod*sin(2*theta_cran
k_rad)./((1-(r_crank/L_conrod*sin(theta_crank_rad)).^2).^(1/2)));
a_piston_crank=-r_crank*(w_crank.^2).*(cos(theta_crank_rad)-
r_crank*(L_conrod^2*(1-2*(cos(theta_crank_rad).^2))-
r_crank^2*(sin(theta_crank_rad)).^4)./(L_conrod^2-
(r_crank*sin(theta_crank_rad)).^2).^(3/2));
%Integrate into Stirling Engine Power Piston Configuration
x_piston_power=x_piston_crank-min(x_piston_crank);
v_piston_power=v_piston_crank/w_engine; %length/rad
%Displacer Piston Crank Slider
engine_stroke=engine_displacer_stroke;
r_crank=engine_stroke/2;
% Piston Position:-
x_piston_crank=r_crank*cos(theta_crank_rad)+L_conrod*((1-
(r_crank/L_conrod*sin(theta_crank_rad)).^2).^(1/2));
```

```
%Assume steady state analysis where w_crank is constant:-
v_piston_crank=-
r_crank*w_crank.*(sin(theta_crank_rad)+r_crank/2/L_conrod*sin(2*theta_cran
k_rad)./((1-(r_crank/L_conrod*sin(theta_crank_rad)).^2).^(1/2)));
a_piston_crank=-r_crank*(w_crank.^2).*(cos(theta_crank_rad)-
r_crank*(L_conrod^2*(1-2*(cos(theta_crank_rad).^2))-
r_crank^2*(sin(theta_crank_rad)).^4)./(L_conrod^2-
(r_crank*sin(theta_crank_rad)).^2).^(3/2));
%Integrate into Stirling Engine Displacer Piston Configuration
x_piston_crank=[x_piston_crank((1/4*theta_engine_rad_step+1):theta_engine_
rad_step) x_piston_crank(1:(1/4*theta_engine_rad_step))];
x_piston_displacer=x_piston_crank-min(x_piston_crank);
v_piston_crank=[v_piston_crank((1/4*theta_engine_rad_step+1):theta_engine_
rad_step) v_piston_crank(1:(1/4*theta_engine_rad_step))];
v_piston_displacer=v_piston_crank/w_engine; %length/rad
%Stirling Engine Piston Motion Display
%>>>>>>>>>>>>>>>>>>>>>
figure
plot(theta_engine_deg,x_piston_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,x_piston_displacer,'r','LineWidth',3)
title('s diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('s (length in meter)')
legend('power piston','displacer')
figure
plot(theta_engine_deg,v_piston_power,'g','LineWidth',3)
hold on
plot(theta_engine_deg,v_piston_displacer,'r','LineWidth',3)
title('v diagram')
xlabel('x, cam angle 0 to 360deg')
ylabel('v (m/s)')
legend('power piston','displacer')
%Stirling Engine Characteristics Initialisation
%Define/Initialize values for engine characteristics
T_regenerator=(T_h-T_k)./(log(T_h./T_k));
r_piston_power=Diameter_piston_power/2; %in meter;
r_piston_displacer=Diameter_piston_displacer/2; %in meter
%V_sw_hot=120.88*10^(-6); %Swept Volume of expansion piston
```

```
%V_sw_power=113.14*10^(-6); %Swept Volume of compression piston
V_sw_hot=pi*(r_piston_displacer^2)*engine_displacer_stroke;
V_sw_cold=V_sw_hot-pi*(r_displacer_drive_rod^2)*engine_displacer_stroke;
V_sw_power=pi*(r_piston_power^2)*engine_power_stroke;
Total_volume_engine_ambient=V_sw_power+V_power_clearance+V_r+V_k+V_h+V_sw_
cold+V_displacer_clearance_cold;
M_gas_engine_initial=density_gas*Total_volume_engine_ambient; %in kg
%Stirling Engine Piston Volume Calculation
V_hot_displacer=pi*(r_piston_displacer^2)*x_piston_displacer;
V_power=pi*(r_piston_power^2)*x_piston_power;
V_cold_displacer=V_sw_cold-pi*(r_piston_displacer^2-
r_displacer_drive_rod^2)*x_piston_displacer;
dV_hot_displacer=pi*(r_piston_displacer^2)*v_piston_displacer;
V_cold_engine=V_power+V_cold_displacer;
dV_cold_engine=pi*(r_piston_power^2)*v_piston_power-
pi*(r_piston_displacer^2-r_displacer_drive_rod^2)*v_piston_displacer;
%Integrate into Schmidt Model
V_hot=V_h+V_displacer_clearance_hot+V_hot_displacer;
dV_hot=dV_hot_displacer;
V_cold=V_k+V_displacer_clearance_cold+V_power_clearance+V_cold_engine;
dV_cold=dV_cold_engine;
%Integrate into Urieli Model
V_exp=V_displacer_clearance_hot+V_hot_displacer;
dV_exp=dV_hot_displacer;
V_comp=V_displacer_clearance_cold+V_power_clearance+V_cold_engine;
dV_comp=dV_cold_engine;
%Schmidt Model for Initial value prediction
P_engine_Schmidt=M_gas_engine_initial*R_spec_gas./(V_cold/T_k+V_r/T_regene
rator+V_hot/T_h);
m_regenerator_Schmidt=V_r*P_engine_Schmidt*(log(T_h/T_k))/(R_spec_gas*(T_h
-T_k));
%Note : W_engine=Integral of (P*(d_V_c/d_theta+d_V_e/d_theta)*d_theta)
Volume_engine_Schmidt=V_hot+V_cold+V_r;
Simpson_multiple=[];
for i=1:((theta_engine_rad_step)/2)
Simpson_multiple=[Simpson_multiple;2;4];
end
Simpson_multiple=[Simpson_multiple;1];
Simpson_multiple(1,1)=1;
```

```
Work_engine_part1=P_engine_Schmidt.*(dV_cold+dV_hot);
Work_engine_part1=[Work_engine_part1 Work_engine_part1(1,1)];
h=2*pi/theta_engine_rad_step;
Work_engine_Schmidt=h/3*Work_engine_part1*Simpson_multiple;
Words = 'Engine Work in J per revolution (Schmidt)';
disp(Words)
disp(Work_engine_Schmidt)
Work_done_Schmidt_power=Work_engine_Schmidt*w_engine_rpm/60;
Words = 'Engine Work in W (Schmidt)';
disp(Words)
disp(Work_done_Schmidt_power)
Work_engine_part2=P_engine_Schmidt.*(dV_hot);
Work_engine_part2=[Work_engine_part2 Work_engine_part2(1,1)];
h=2*pi/theta_engine_rad_step;
Work_engine_Schmidt_hot=h/3*Work_engine_part2*Simpson_multiple;
Q_hot_Schmidt=Work_engine_Schmidt_hot;
Engine_Efficiency_Schmidt=Work_engine_Schmidt/Q_hot_Schmidt;
Words = 'Engine Efficiency (Schmidt)';
disp(Words)
disp(Engine_Efficiency_Schmidt)
%Urieli model - ODEs in adiabatic analysis
%Separate odd and even rows for rk4 method 1/2step calculation
theta_engine_rad_step=theta_engine_rad_step/2;
theta_engine_rad_step_size=theta_engine_rad_step_size*2;
for i=1:theta_engine_rad_step
V_exp_odd(1,i)=V_exp(1,i*2-1);
dV_exp_odd(1,i)=dV_exp(1,i*2-1);
V_comp_odd(1,i)=V_comp(1,i*2-1);
dV_comp_odd(1,i)=dV_comp(1,i*2-1);
V_exp_even(1,i)=V_exp(1,i*2);
dV_exp_even(1,i)=dV_exp(1,i*2);
V_comp_even(1,i)=V_comp(1,i*2);
dV_comp_even(1,i)=dV_comp(1,i*2);
theta_engine_deg_odd(1,i)=theta_engine_deg(1,i*2-1);
end
V_exp=V_exp_odd;
dV_exp=dV_exp_odd;
V_comp=V_comp_odd;
dV_comp=dV_comp_odd;
```

```
figure
plot(theta_engine_deg_odd,V_exp,'r','LineWidth',3)
title('Expansion Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
figure
plot(theta_engine_deg_odd,V_comp,'r','LineWidth',3)
title('Compression Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
figure
plot(theta_engine_deg_odd,dV_exp,'g','LineWidth',3)
hold on
plot(theta_engine_deg_odd,dV_comp,'r','LineWidth',3)
title('Volume Change')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume Change (m3/s)')
%Initialize values
Temperature_term1=V_k/T_k+V_r/T_regenerator+V_h/T_h;
T_comp=ones(1,theta_engine_rad_step+1).*T_k;
T_exp=ones(1,theta_engine_rad_step+1).*T_h;
T_ck=T_comp(1,1);
T_he=T_exp(1,1);
Q_h=zeros(1,theta_engine_rad_step);
Q_k=zeros(1,theta_engine_rad_step);
Q_r=zeros(1,theta_engine_rad_step);
W_exp=zeros(1,theta_engine_rad_step);
W_comp=zeros(1,theta_engine_rad_step);
T_kr=T_k;
T_rh=T_h;
for j=1:50
for i=1:(theta_engine_rad_step)
if i==1
M_gas_engine=M_gas_engine_initial;
end
P_engine_Urieli(1,i)=M_gas_engine*R_spec_gas/(V_comp(1,i)/T_comp(1,i)+Temp
erature_term1+V_exp(1,i)/T_exp(1,i));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli(1,i)*(dV_comp(1,i)/T_ck+dV_exp(1,i)/T_he);
```

```
dP_engine_Urieli_term2=V_comp(1,i)/T_ck+k_isentropic*Temperature_term1+V_e
xp(1,i)/T_he;
dP_engine_Urieli(1,i)=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli(1,i)*V_comp(1,i)/(R_spec_gas*T_comp(1,i));
m_k=P_engine_Urieli(1,i)*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli(1,i)*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli(1,i)*V_h/(R_spec_gas*T_h);
m_exp=P_engine_Urieli(1,i)*V_exp(1,i)/(R_spec_gas*T_exp(1,i));
dm_exp(1,i)=(P_engine_Urieli(1,i)*dV_exp(1,i)+V_exp(1,i)*dP_engine_Urieli(
1,i)/k_isentropic)/(R_spec_gas*T_he);
dm_comp(1,i)=(P_engine_Urieli(1,i)*dV_comp(1,i)+V_comp(1,i)*dP_engine_Urie
li(1,i)/k_isentropic)/(R_spec_gas*T_ck);
dm_k(1,i)=m_k*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
dm_r(1,i)=m_r*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
dm_h(1,i)=m_h*dP_engine_Urieli(1,i)/P_engine_Urieli(1,i);
m_dot_ck=-dm_comp(1,i);
m_dot_he=dm_exp(1,i);
m_dot_kr=m_dot_ck-dm_k(1,i);
m_dot_rh=m_dot_he+dm_h(1,i);
if m_dot_ck>0
T_ck=T_comp(1,i);
else
T_ck=T_k;
end
if m_dot_he>0
T_he=T_h;
else
T_he=T_exp(1,i);
end
dT_exp=T_exp(1,i)*(dP_engine_Urieli(1,i)/P_engine_Urieli(1,i)+dV_exp(1,i)/
V_exp(1,i)-dm_exp(1,i)/m_exp);
dT_comp=T_comp(1,i)*(dP_engine_Urieli(1,i)/P_engine_Urieli(1,i)+dV_comp(1,
i)/V_comp(1,i)-dm_comp(1,i)/m_comp);
dQ_k=V_k*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_ck*m_dot_ck-
T_kr*m_dot_kr);
dQ_r=V_r*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_kr*m_dot_kr-
T_rh*m_dot_rh);
dQ_h=V_h*C_v_gas*dP_engine_Urieli(1,i)/R_spec_gas-C_p_gas*(T_rh*m_dot_rh-
T_he*m_dot_he);
```

```
dW_exp=P_engine_Urieli(1,i)*dV_exp(1,i);
dW_comp=P_engine_Urieli(1,i)*dV_comp(1,i);
k1_exp=theta_engine_rad_step_size*dT_exp;
k1_comp=theta_engine_rad_step_size*dT_comp;
k1_dQk=theta_engine_rad_step_size*dQ_k;
k1_dQr=theta_engine_rad_step_size*dQ_r;
k1_dQh=theta_engine_rad_step_size*dQ_h;
k1_dWe=theta_engine_rad_step_size*dW_exp;
k1_dWc=theta_engine_rad_step_size*dW_comp;
T_ck_half=T_ck;
T_he_half=T_he;
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp_even(1,i)/(T_comp(1,i
)+0.5*k1_comp)+Temperature_term1+V_exp_even(1,i)/(T_exp(1,i)+0.5*k1_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp_even(1,i)/T_ck_half+dV_exp_even
(1,i)/T_he_half);
dP_engine_Urieli_term2=V_comp_even(1,i)/T_ck_half+k_isentropic*Temperature
_term1+V_exp_even(1,i)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp_even(1,i)/(R_spec_gas*(T_comp(1,i)+0.5*
k1_comp));
m_exp=P_engine_Urieli_half*V_exp_even(1,i)/(R_spec_gas*(T_exp(1,i)+0.5*k1_
exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp_even(1,i)+V_exp_even(1,i)*dP_engi
ne_Urieli_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp_even(1,i)+V_comp_even(1,i)*dP_e
ngine_Urieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+0.5*k1_comp;
```

```
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+0.5*k1_exp;
end
T_ck_half=T_ck;
T_he_half=T_he;
dT_exp=(T_exp(1,i)+0.5*k1_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half
+dV_exp_even(1,i)/V_exp_even(1,i)-dm_exp_half/m_exp);
k2_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+0.5*k1_comp)*(dP_engine_Urieli_half/P_engine_Urieli_h
alf+dV_comp_even(1,i)/V_comp_even(1,i)-dm_comp_half/m_comp);
k2_comp=theta_engine_rad_step_size*dT_comp;
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp_even(1,i);
dW_comp=P_engine_Urieli_half*dV_comp_even(1,i);
k2_dQk=theta_engine_rad_step_size*dQ_k;
k2_dQr=theta_engine_rad_step_size*dQ_r;
k2_dQh=theta_engine_rad_step_size*dQ_h;
k2_dWe=theta_engine_rad_step_size*dW_exp;
k2_dWc=theta_engine_rad_step_size*dW_comp;
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp_even(1,i)/(T_comp(1,i
)+0.5*k2_comp)+Temperature_term1+V_exp_even(1,i)/(T_exp(1,i)+0.5*k2_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp_even(1,i)/T_ck_half+dV_exp_even
(1,i)/T_he_half);
dP_engine_Urieli_term2=V_comp_even(1,i)/T_ck_half+k_isentropic*Temperature
_term1+V_exp_even(1,i)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp_even(1,i)/(R_spec_gas*(T_comp(1,i)+0.5*
k2_comp));
```

```
m_exp=P_engine_Urieli_half*V_exp_even(1,i)/(R_spec_gas*(T_exp(1,i)+0.5*k2_
exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp_even(1,i)+V_exp_even(1,i)*dP_engi
ne_Urieli_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp_even(1,i)+V_comp_even(1,i)*dP_e
ngine_Urieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+0.5*k2_comp;
else
T_ck_half=T_k;
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+0.5*k2_exp;
end
dT_exp=(T_exp(1,i)+0.5*k2_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half
+dV_exp_even(1,i)/V_exp_even(1,i)-dm_exp_half/m_exp);
k3_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+0.5*k2_comp)*(dP_engine_Urieli_half/P_engine_Urieli_h
alf+dV_comp_even(1,i)/V_comp_even(1,i)-dm_comp_half/m_comp);
k3_comp=theta_engine_rad_step_size*dT_comp;
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp_even(1,i);
```

```
dW_comp=P_engine_Urieli_half*dV_comp_even(1,i);
k3_dQk=theta_engine_rad_step_size*dQ_k;
k3_dQr=theta_engine_rad_step_size*dQ_r;
k3_dQh=theta_engine_rad_step_size*dQ_h;
k3_dWe=theta_engine_rad_step_size*dW_exp;
k3_dWc=theta_engine_rad_step_size*dW_comp;
i2=i+1;
if i2>theta_engine_rad_step
i2=i2-theta_engine_rad_step;
end
P_engine_Urieli_half=M_gas_engine*R_spec_gas/(V_comp(1,i2)/(T_comp(1,i)+k3
_comp)+Temperature_term1+V_exp(1,i2)/(T_exp(1,i)+k3_exp));
dP_engine_Urieli_term1=-
k_isentropic*P_engine_Urieli_half*(dV_comp(1,i2)/T_ck_half+dV_exp(1,i2)/T_
he_half);
dP_engine_Urieli_term2=V_comp(1,i2)/T_ck_half+k_isentropic*Temperature_ter
m1+V_exp(1,i2)/T_he_half;
dP_engine_Urieli_half=dP_engine_Urieli_term1/dP_engine_Urieli_term2;
m_comp=P_engine_Urieli_half*V_comp(1,i2)/(R_spec_gas*(T_comp(1,i)+k3_comp)
);
m_exp=P_engine_Urieli_half*V_exp(1,i2)/(R_spec_gas*(T_exp(1,i)+k3_exp));
m_k=P_engine_Urieli_half*V_k/(R_spec_gas*T_k);
m_r=P_engine_Urieli_half*V_r/(R_spec_gas*T_regenerator);
m_h=P_engine_Urieli_half*V_h/(R_spec_gas*T_h);
dm_exp_half=(P_engine_Urieli_half*dV_exp(1,i2)+V_exp(1,i2)*dP_engine_Uriel
i_half/k_isentropic)/(R_spec_gas*T_he_half);
dm_comp_half=(P_engine_Urieli_half*dV_comp(1,i2)+V_comp(1,i2)*dP_engine_Ur
ieli_half/k_isentropic)/(R_spec_gas*T_ck_half);
dm_k_half=m_k*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_r_half=m_r*dP_engine_Urieli_half/P_engine_Urieli_half;
dm_h_half=m_h*dP_engine_Urieli_half/P_engine_Urieli_half;
m_dot_ck_half=-dm_comp_half;
m_dot_he_half=dm_exp_half;
m_dot_kr_half=m_dot_ck-dm_k_half;
m_dot_rh_half=m_dot_he+dm_h_half;
if m_dot_ck_half>0
T_ck_half=T_comp(1,i)+k3_comp;
else
T_ck_half=T_k;
```

```
end
if m_dot_he_half>0
T_he_half=T_h;
else
T_he_half=T_exp(1,i)+k3_exp;
end
dT_exp=(T_exp(1,i)+k3_exp)*(dP_engine_Urieli_half/P_engine_Urieli_half+dV_
exp(1,i2)/V_exp(1,i2)-dm_exp_half/m_exp);
k4_exp=theta_engine_rad_step_size*dT_exp;
dT_comp=(T_comp(1,i)+k3_comp)*(dP_engine_Urieli_half/P_engine_Urieli_half+
dV_comp(1,i2)/V_comp(1,i2)-dm_comp_half/m_comp);
k4_comp=theta_engine_rad_step_size*dT_comp;
T_exp(1,i+1)=T_exp(1,i)+1/6*(k1_exp+2*k2_exp+2*k3_exp+k4_exp);
T_comp(1,i+1)=T_comp(1,i)+1/6*(k1_comp+2*k2_comp+2*k3_comp+k4_comp);
dQ_k=V_k*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_ck_half*m_dot_ck_half-T_kr*m_dot_kr_half);
dQ_r=V_r*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_kr*m_dot_kr_half-T_rh*m_dot_rh_half);
dQ_h=V_h*C_v_gas*dP_engine_Urieli_half/R_spec_gas-
C_p_gas*(T_rh*m_dot_rh_half-T_he_half*m_dot_he_half);
dW_exp=P_engine_Urieli_half*dV_exp(1,i2);
dW_comp=P_engine_Urieli_half*dV_comp(1,i2);
k4_dQk=theta_engine_rad_step_size*dQ_k;
k4_dQr=theta_engine_rad_step_size*dQ_r;
k4_dQh=theta_engine_rad_step_size*dQ_h;
k4_dWe=theta_engine_rad_step_size*dW_exp;
k4_dWc=theta_engine_rad_step_size*dW_comp;
Q_k(1,i+1)=Q_k(1,i)+1/6*(k1_dQk+2*k2_dQk+2*k3_dQk+k4_dQk);
Q_r(1,i+1)=Q_r(1,i)+1/6*(k1_dQr+2*k2_dQr+2*k3_dQr+k4_dQr);
Q_h(1,i+1)=Q_h(1,i)+1/6*(k1_dQh+2*k2_dQh+2*k3_dQh+k4_dQh);
W_exp(1,i+1)=W_exp(1,i)+1/6*(k1_dWe+2*k2_dWe+2*k3_dWe+k4_dWe);
W_comp(1,i+1)=W_comp(1,i)+1/6*(k1_dWc+2*k2_dWc+2*k3_dWc+k4_dWc);
if i==theta_engine_rad_step
T_exp_diff(1,j)=T_exp(1,theta_engine_rad_step+1)-T_exp(1,1);
T_exp(1,1)=T_exp(1,theta_engine_rad_step+1);
T_exp(:,theta_engine_rad_step+1)=[];
T_comp_diff(1,j)=T_comp(1,theta_engine_rad_step+1)-T_comp(1,1);
T_comp(1,1)=T_comp(1,theta_engine_rad_step+1);
T_comp(:,theta_engine_rad_step+1)=[];
```

```
end
end
end
Work_done_Urieli=W_exp(1,theta_engine_rad_step+1)+W_comp(1,theta_engine_ra
d_step+1);
Engine_Efficiency_Urieli=Work_done_Urieli/Q_h(1,theta_engine_rad_step+1);
Words = 'Engine Work in J per revolution (Urieli)';
disp(Words)
disp(Work_done_Urieli)
Work_done_Urieli_power=Work_done_Urieli*w_engine_rpm/60;
Words = 'Engine Work in W (Urieli)';
disp(Words)
disp(Work_done_Urieli_power)
Words = 'Engine Efficiency Urieli';
disp(Words)
disp(Engine_Efficiency_Urieli)
% PV Diagram Display:
Volume_engine_Urieli=V_comp+V_exp+V_r+V_h+V_k;
figure
plot(Volume_engine_Schmidt,P_engine_Schmidt,'g','LineWidth',3)
hold on
plot(Volume_engine_Urieli,P_engine_Urieli,'r','LineWidth',3)
title('PV Diagram')
xlabel('Volume')
ylabel('Pressure')
legend('Isothermal','Adiabatic')
figure
plot(theta_engine_deg,P_engine_Schmidt,'g','LineWidth',3)
hold on
plot(theta_engine_deg_odd,P_engine_Urieli,'r','LineWidth',3)
title('Engine Pressure')
xlabel('x, shaft angle 0 to 360deg')
ylabel('P (Pressure in Pa)')
legend('Isothermal','Adiabatic')
figure
plot(theta_engine_deg,Volume_engine_Schmidt,'r','LineWidth',3)
title('Engine Internal Volume')
xlabel('x, shaft angle 0 to 360deg')
ylabel('Volume (m3)')
```

```
% Gas Torque Calculation:-
pressure_gas_crank=P_engine_Schmidt;
Area_piston_crank=pi*Diameter_piston_power^2/4;
engine_stroke=engine_power_stroke;
r_crank=engine_stroke/2;
theta_crank_rad=theta_engine_rad;
for i=1:theta_engine_rad_step
theta_crank_rad_odd(1,i)=theta_crank_rad(1,i*2-1);
end
theta_crank_rad=theta_crank_rad_odd;
tangent_Phi_conrod_rad=r_crank*sin(theta_crank_rad)./(L_conrod*((1-
(r_crank/L_conrod*sin(theta_crank_rad)).^2).^(1/2)));
tangent_Phi_conrod_deg=tangent_Phi_conrod_rad*180/pi;
Force_gas_crank_piston=pressure_gas_crank*Area_piston_crank;
Area_piston_crank_displacer=pi*Diameter_piston_displacer^2/4;
tangent_Phi_conrod_rad_displacer=r_crank*sin(theta_crank_rad+pi/2)./(L_con
rod*((1-(r_crank/L_conrod*sin(theta_crank_rad+pi/2)).^2).^(1/2)));
tangent_Phi_conrod_deg_displacer=tangent_Phi_conrod_rad_displacer*180/pi;
Force_gas_crank_piston_displacer=pressure_gas_crank*Area_piston_crank_disp
lacer;
for i=1:theta_engine_rad_step
Force_gas_crank_piston_vector=[-Force_gas_crank_piston(1,i) 0 0];
Force_gas_crank_32_vector=[-Force_gas_crank_piston(1,i)
Force_gas_crank_piston(1,i).*tangent_Phi_conrod_rad(1,i) 0];
R_A_crank=[r_crank*cos(theta_crank_rad(1,i))
r_crank*sin(theta_crank_rad(1,i)) 0];
Torque_driving_crank_21_vector =
cross(R_A_crank,Force_gas_crank_32_vector);
Torque_driving_crank_21(1,i)=Torque_driving_crank_21_vector(1,3);
Force_gas_crank_32_vector_displacer=[-
Force_gas_crank_piston_displacer(1,i)
Force_gas_crank_piston_displacer(1,i).*tangent_Phi_conrod_rad_displacer(1,
i) 0];
R_A_crank_displacer=[r_crank*cos(theta_crank_rad(1,i)+pi/2)
r_crank*sin(theta_crank_rad(1,i)+pi/2) 0];
Torque_driving_crank_21_vector_displacer =
cross(R_A_crank_displacer,Force_gas_crank_32_vector_displacer);
Torque_driving_crank_21_displacer(1,i)=Torque_driving_crank_21_vector_disp
lacer(1,3);
```

```
end
figure
plot(theta_crank_rad*180/pi,(Torque_driving_crank_21+Torque_driving_crank_
21_displacer),'r','LineWidth',3)
title('Gas Torque')
xlabel('theta, crank angle 0 to 360deg')
ylabel('T21, torque in Nm')
```