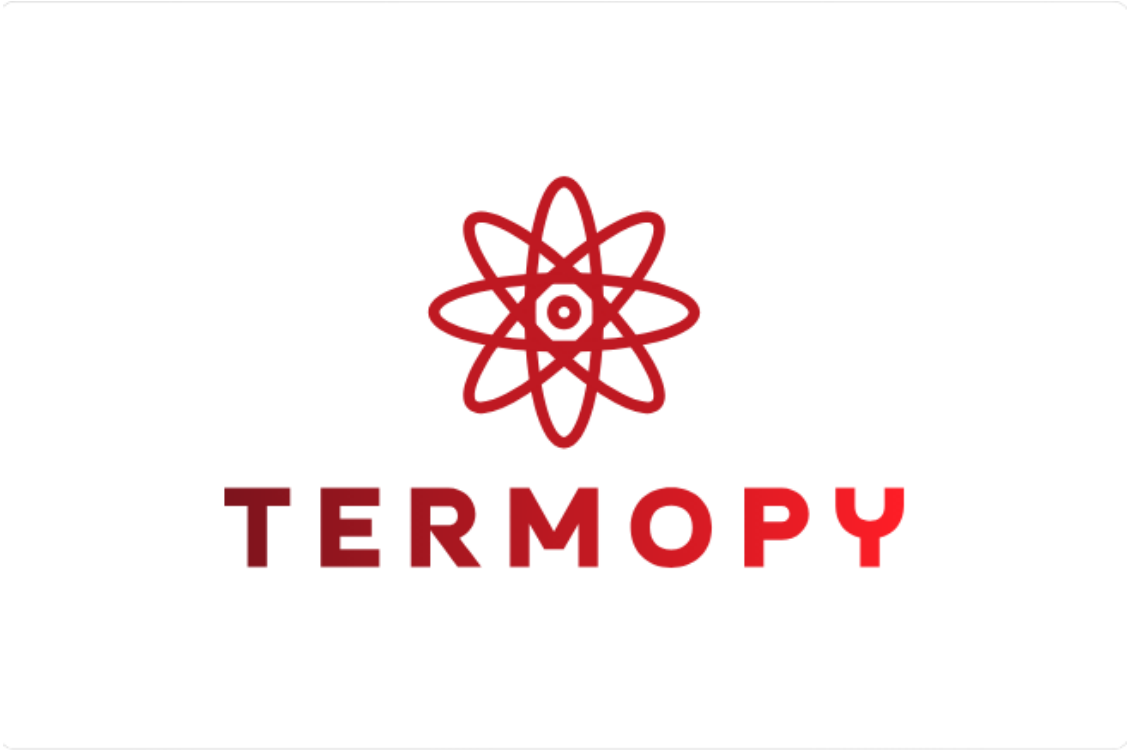
introduction

October 25, 2023



A Python library for modelling thermodynamic processes and cycles.

1 Introduction

What is Thermopy?

Thermopy is a Python library that allows the user to model thermodynamic processes and cycles. It is designed to be easy to use and to be able to model a wide range of processes and cycles. It is also designed to be easy to extend and add new processes and cycles.

1.1 Key Features

Process modelling - Termopy allows a user to quickly model a series of thermodynamic processes. Included in the library are:

- **Isobaric process** - A process where pressure is constant.
- **Isochoric process** - A process where volume is constant.
- **Isothermal process** - A process where temperature is constant.
- **Adiabatic process** - A process where no heat is transferred.

Process data - Thermopy allows the user to calculate the work done, heat transferred and change in internal energy for each process.

Cycle modeling - Thermopy allows the user to model a range of thermodynamic cycles. Included in the library are:

- **Carnot cycle** - A cycle consisting of two isothermal processes and two adiabatic processes.
- **Stirling cycle** - A cycle consisting of two isothermal processes and two isochoric processes.
- **Otto cycle** - A cycle consisting of two isochoric processes and two isobaric processes.
- **Diesel cycle** - A cycle consisting of two isochoric processes and two adiabatic processes.
- **Brayton cycle** - A cycle consisting of two isobaric processes and two adiabatic processes.
- **Rankine cycle** - A cycle consisting of two isobaric processes, one isentropic process and one isenthalpic process.

Cycle data - Thermopy allows the user to calculate the work done, heat transferred and change in internal energy for each process in the cycle. It also allows the user to calculate the thermal efficiency and the mean effective pressure of the cycle.

Graphing - Thermopy allows the user to graph the processes and cycles that they have modelled.

1.2 Installation

How to install Thermopy

Thermopy is currently in early development, and for that reason it is mostly still being tested. It is not yet available on PyPI, but it can be downloaded from my Github page Here. Once downloaded, simply run the following command in the command line:

```
conda install pip
pip install /path/to/TermoPy
```

1.3 Getting Started

1.3.1 How to use Thermopy

To use TermoPy, you can import the library using the following command:

```
import Thermopy as tp
```

Once imported, you can begin to model processes and cycles. To model a process, you must first create a fluid object. Fluid objects are the basis of the library, and are used to store the properties of the fluid that you are modelling. To create a fluid object, you must first import the fluid class from the library:

```
fluid = tp.Fluid(P = pressure, V = volume, T = temperature, n = moles, gas = name of gas)
```

Note that you only need to specify three of the initial properties, as the fourth can be calculated using the ideal gas equation. The gas parameter is optional, and if not specified will default to air. The gas parameter can be any 80 pre-defined gases.

Once you have created a fluid object, you can access the attributes of the fluid. a complete set of these is provided in the appendix “attributes”

To model a process, you can use methods from the fluid object. For example, to model an isobaric process, you can use the following command:

```
fluid.isobaric(V = final volume, T = final temperature)
```

Note that you can only specify one final state, the rest are calculated automatically. Also note that you cannot give a final pressure, as this is constant throughout the process. The same applies to the other single-variable static processes.

1.3.2 Visualizing Simulated Processes

Termopy provides a set of functions for visualizing the simulated thermodynamic processes. Two key functions for generating and displaying plots are "plot()" and "show()". Additionally, you can use the "save()" function to save plots as PNG files. These functions are designed to offer flexibility in creating and managing multiple plots before displaying or saving them all at once.

The plot() Function

The "plot()" function is used to create visual representations of the state vector of the thermodynamic process. When using the "plot()" function, you can specify the elements to be plotted by setting the "display" variable as a string of desired elements. TermoPy supports plotting of the following elements: - Pressure (P) - Volume (V) - Temperature (T) - Entropy (S)

it can be called using the following command:

```
tp.plot_process(fluid, display = "PV")
tp.show()
```

Plotting Options

1. **Single-Variable Plot (One-Character String):** If the "display" is set as a one-character string, it will generate a plot showing the change in the specified variable over time. You can also specify the time taken for each process, which will be represented on the plot.
2. **Two-Variable Plot (Two-Character String):** When "display" is set as a two-character string, the function will create a plot showing the change in the first variable over the change in the second variable. Similar to the single-variable plot, you can specify the time taken for each process.
3. **Three-Dimensional Plot (Three-Character String):** For a more advanced visualization, setting "display" as a three-character string results in a three-dimensional plot of the process within the given 3-axis space.

The show() Function

After using the "plot()" function to generate plots, you can use the "show()" function to display them in your Jupyter Notebook. This separation of functions allows you to create and customize multiple plots before presenting them. The "show()" function simplifies the process of viewing and analyzing the graphical representations of thermodynamic processes.

In summary, Termopy offers a versatile set of functions for visualizing thermodynamic processes. By utilizing "plot()" and "show()" in combination, you can generate and present graphical representations of your simulations, enhancing your understanding of the behavior of ideal gases in various processes.

Saving plots

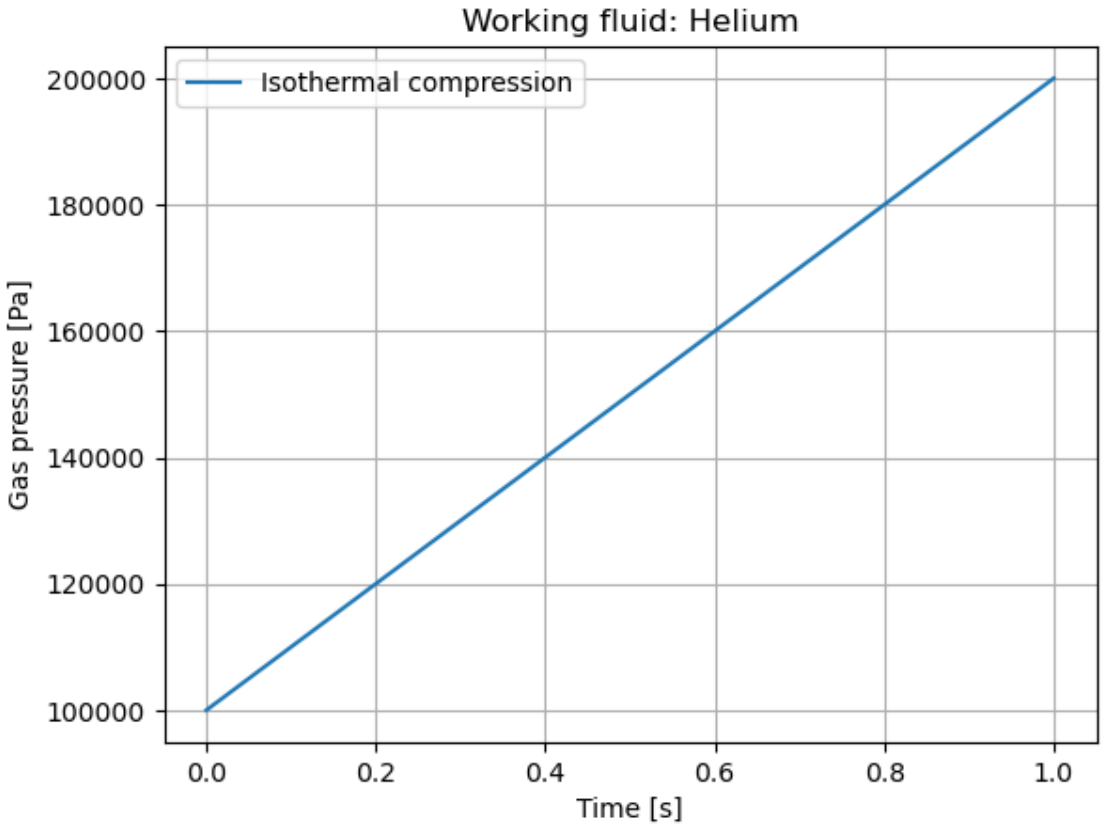
Saving plots is used by setting the "save" variable to TRUE when calling the show() function. You can further specify the name of the saved file by setting the "name" variable as a string. The file will be saved in the same directory as your program.

2 Examples of plotting

The type of plot is dependent on the length of the display string. If a single variable is specified, then the plot will be that variable plotted against time. If two variables are specified, then the plot will be one variable against the other. If three variables are specified, then the plot will be three dimensional.

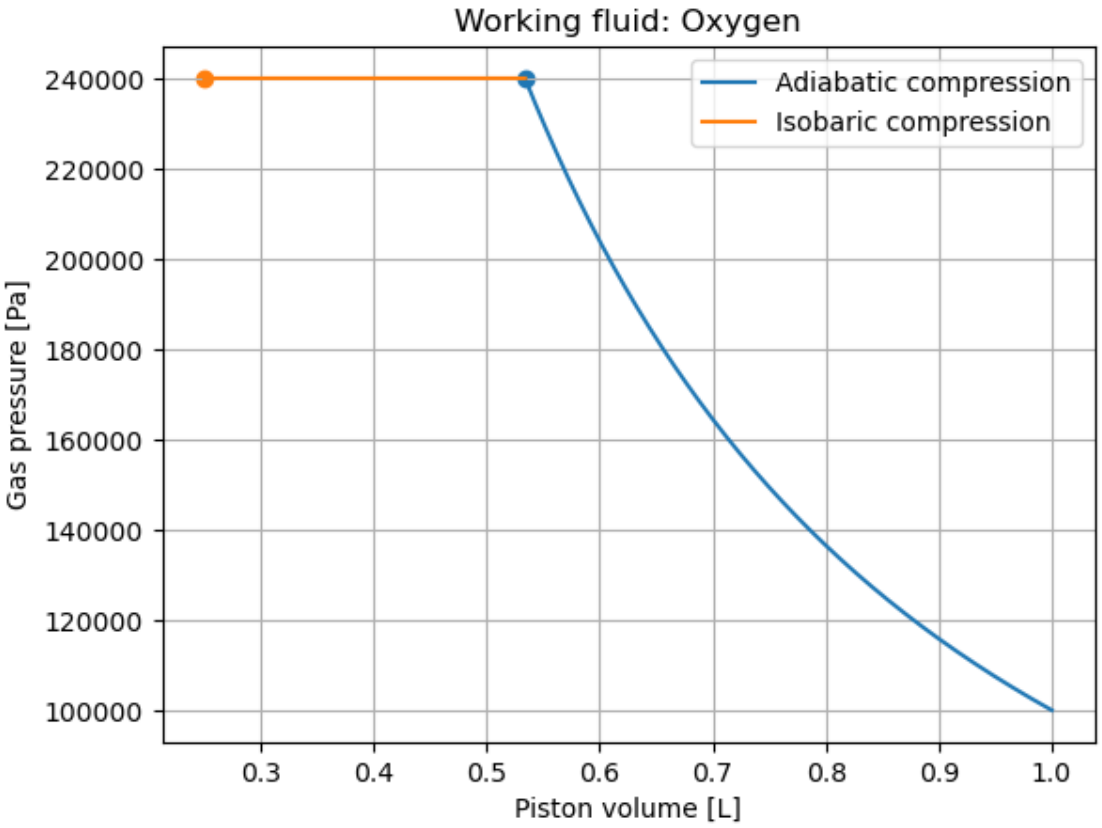
2.1 example of 1-variable plot

[1]:
```
import termoPy as tp

Fluid = tp.Fluid(P = 1e5, T = 300, V = 1e-3, gas = 'Helium')
Fluid.isothermal(P = 2e5)

tp.plot(Fluid, display='P')
tp.show()
```



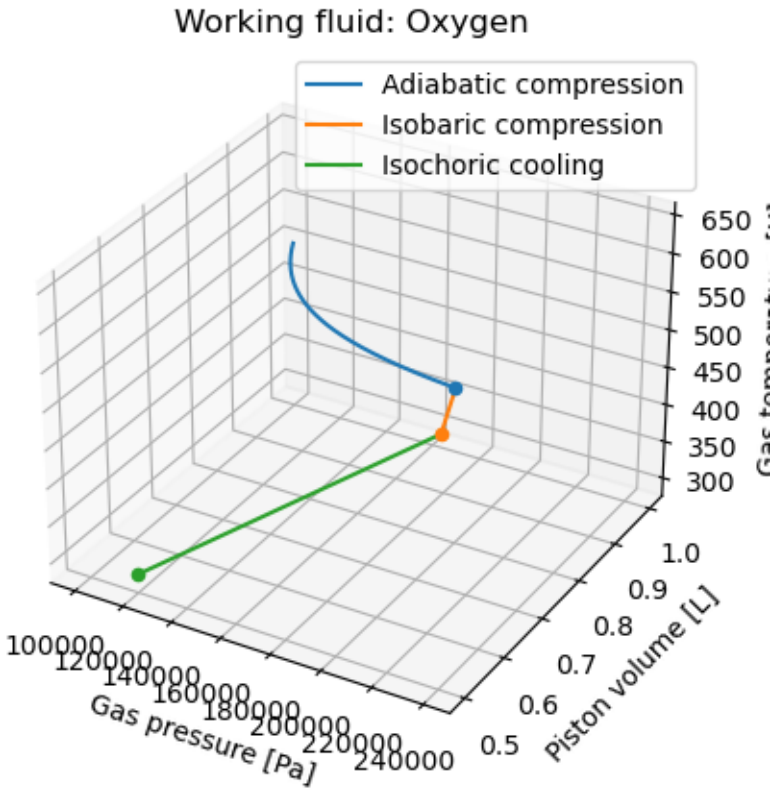
2.2 example of 2-variable plot

[2]:
```
O2 = tp.Fluid(P=1e5, V=1e-3, T=500, gas='O2')
O2.adiabatic(P=2.4e5, time=4)
O2.isobaric(T=300)

tp.plot(O2, display="VP")
tp.show()
```



2.3 Example of 3-variable plot

[3]:
```
O2 = tp.Fluid(P=1e5, V=1e-3, T=500, gas='O2')
O2.adiabatic(P=2.4e5, time=4)
O2.isobaric(T=600)
O2.isochoric(T=300)

tp.plot(O2, display="PVT")
tp.show()
```



3 Cycles

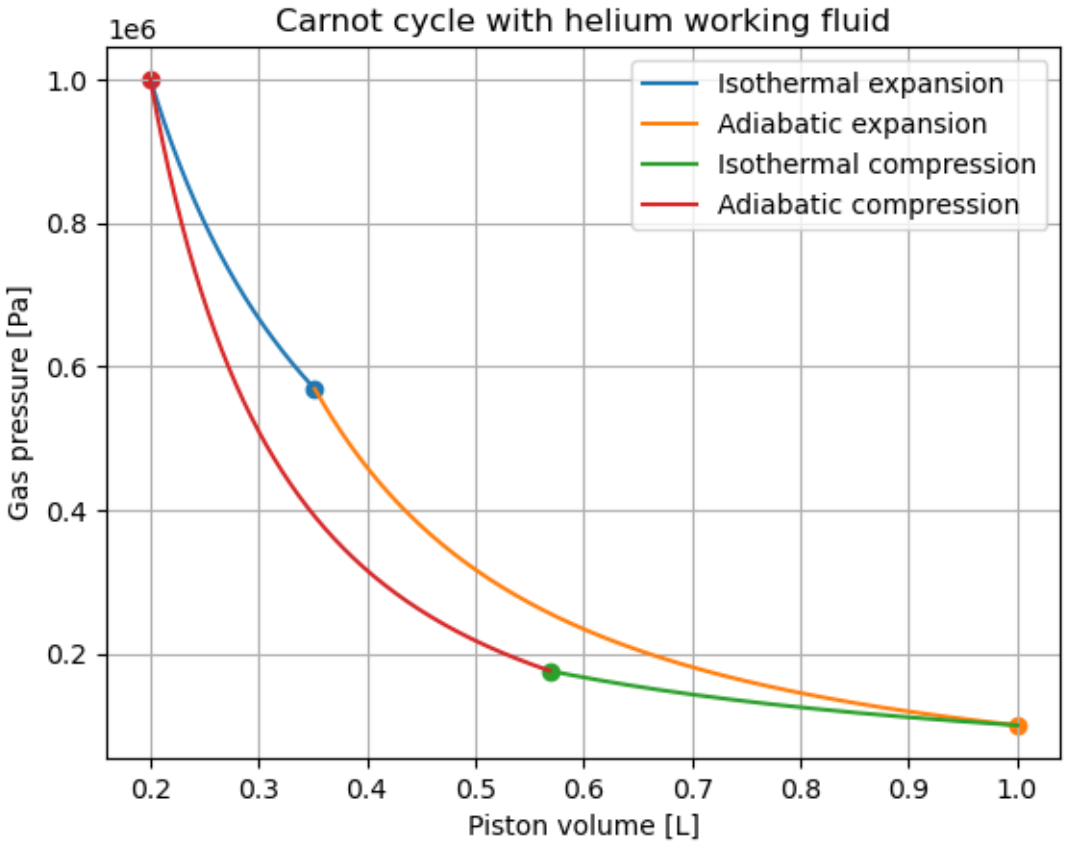
To model a cycle, you can use the following command:

```
cycle = tp.Cycle(T_hot = hot temperature, T_cold = cold temperature,
                 compression_ratio = compression ration of the piston,
                 P = pressure, V = volume, n = moles, gas = name of gas)
```

The difference with cycles as compared with fluids is that you must specify the hot and cold temperatures, as well as the compression ratio. The compression ratio is the ratio of the volume of the cylinder when the piston is at the bottom of the stroke to the volume of the cylinder when the piston is at the top of the stroke. The rest of the parameters are the same as for fluids. However the volume must be specified, as this is assumed to be known for the cycle.

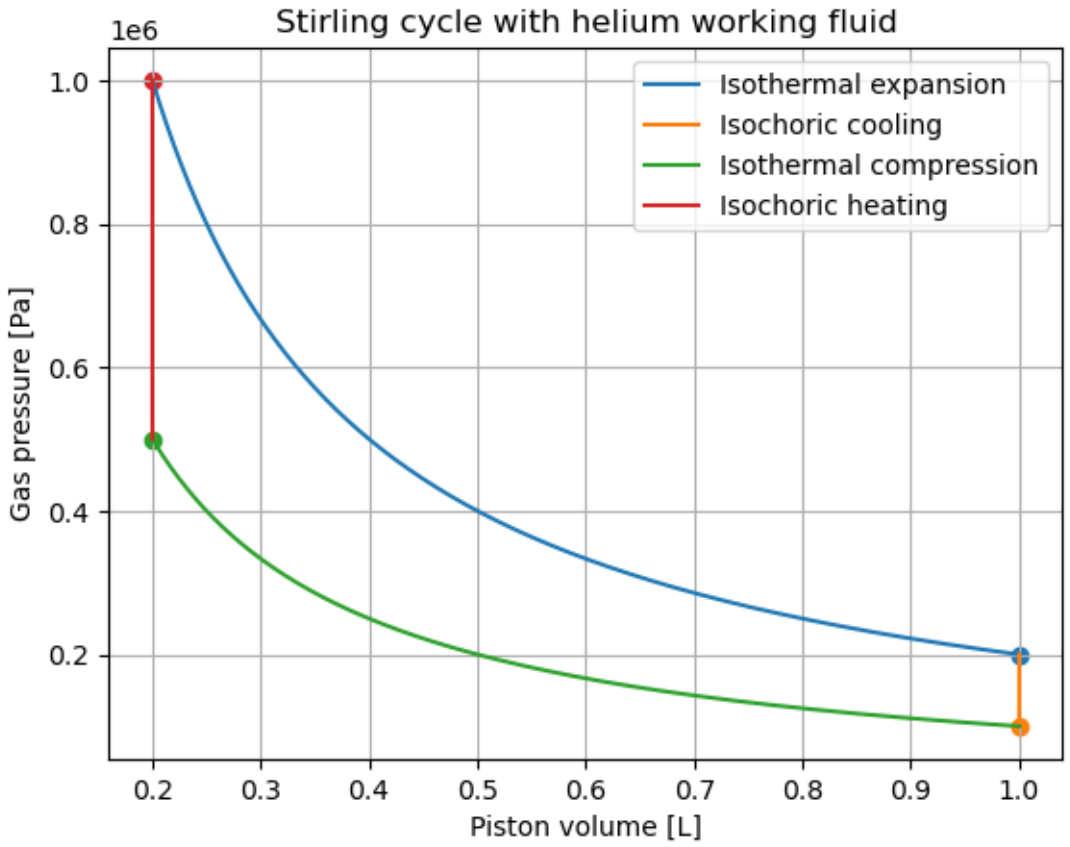
3.1 Example of a Carnot cycle

[6]:
```
carnot = tp.Carnot(T_cold=300, T_hot=600, compression_ratio=5,V=1e-3, P=1e6,␣
↪gas='Helium')

tp.plot(carnot, display='VP')
tp.show()
```



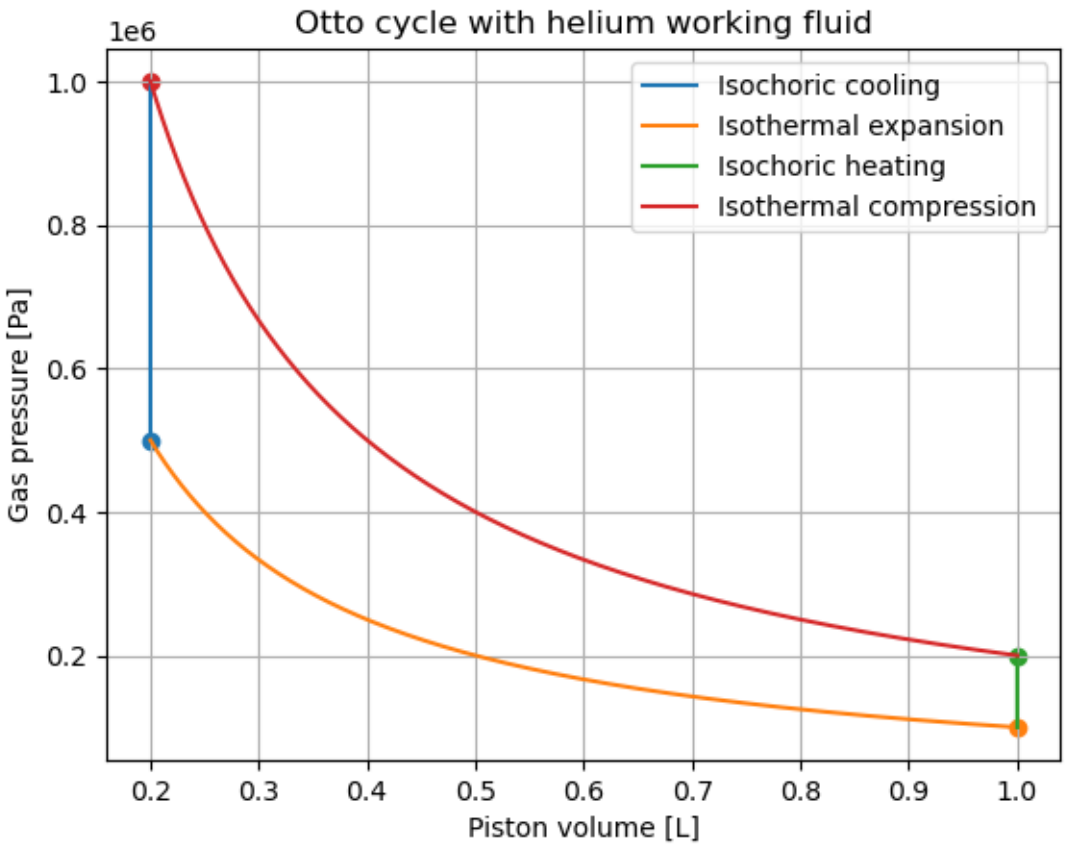
3.2 Example of a Stirling cycle

[5]:
```
stirling = tp.Stirling(T_cold=300, T_hot=600, compression_ratio=5,V=1e-3,␣
↪P=1e6, gas='Helium')

tp.plot(stirling, display='VP')
tp.show()
```



3.3 Example of an Otto cycle

[7]:
```
otto = tp.Otto(T_cold=300, T_hot=600, compression_ratio=5,V=1e-3, P=1e6,␣
↪gas='Helium')
tp.plot(otto, display='VP')
tp.show()
```


Otto cycle with helium working fluid
1e6
1.0
0.8
0.6
0.4
0.2
Gas pressure [Pa]
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Piston volume [L]
Isochoric cooling
Isothermal expansion
Isochoric heating
Isothermal compression

4 Overview of included attributes and methods

4.1 Vectors: Change throughout the process

State variables:

- `pressure`: pressure of the fluid
- `volume`: volume of the fluid
- `temperature`: temperature of the fluid
- `entropy`: entropy of the system
- `internal_energy`: internal energy of the system
- `heat`: represents the heat added by each process Q_{in}
- `work`: represents the work done on the fluid by each process W_{on}

Extensive variables:

- `rms`: root mean square velocity of the particles in the fluid
- `nv`: atomic density of the particles in the fluid
- `mean_free_path`: mean free path of the particles in the fluid
- `mean_collision_time`: mean collision time of the particles in the fluid
- `collision_rate`: collision rate of the particles in the fluid

Additional variables:

- `processes`: list of processes that the fluid has undergone

4.2 Scalars: Static throughout the process

- `M`: molar mass of the fluid
- `n`: number of moles of the fluid
- `Cv`: heat capacity at constant volume
- `Cp`: heat capacity at constant pressure
- `gamma`: ratio of heat capacities
- `diameter`: molecular diameter of the molecules in the fluid
- `atomic_mass`: mass of the atoms in the fluid
- `time_taken`: time taken for the processes to occur
- `heat_added`: total heat added to the fluid Q_{in}
- `heat_removed`: total heat removed from the fluid Q_{out}

Only for cycles:

- `T_hot`: temperature of the hot reservoir
- `T_cold`: temperature of the cold reservoir
- `compression_ratio`: ratio of the volume of the fluid at the end of the compression process to the volume of the fluid at the start of the compression process

4.3 Other

- `name`: name of the fluid
- `formula`: chemical formula of the fluid
- `title`: title of the cycle [only for cycles]
- `properties`: dictionary of properties of the fluid

4.4 Methods of the fluid class

the methods are named after the processes that the fluid undergoes, they take one final state variable and time taken as arguments and then calculate the state vectors throughout the process.

- `isothermal(P = pressure, V = volume, time = time taken)`: isothermal process
- `isobaric(T = temperature, V = volume, time = time taken)`: isobaric process
- `isochoric(T = temperature, P = pressure, time = time taken)`: isochoric process
- `adiabatic(P = pressure, V = volume, T = temperature, time = time taken)`: adiabatic process