

Modelling of Energy in Transit Using Python

Krishna Gajula, Vikrant Sharma, Basant Sharma, Dhananjay R. Mishra, Pankaj Dumka*
 Department of Mechanical Engineering
 Jaypee University of Engineering and Technology
 A.B. Road, Raghogarh-473226, Guna (India)

Abstract:- In this paper, an attempt has been made to develop a Python module for evaluating energy in transit (viz., heat and work). The work considered in this article is non-dissipative displacement work. Functions for constant pressure, volume, and temperature were developed, along with adiabatic and polytropic processes. Also, the sensible and latent heat were also transformed into functions. The modules NumPy and Matplotlib were used to perform the stipulated task. The correctness of codes was checked against four different numerical problems, and it has been observed that the program results match exactly with the results in the literature. As a result, the developed functions thus developed have shown high accuracy with the least effort and error in all the cases.

Keywords:- Energy in Transit; Thermodynamic Processes; Heat; Work; Python Programming; NumPy; Matplotlib.

I. INTRODUCTION

The concept of energy in transit describes that the system cannot store the energy in the form of work or heat it will be in a transition state, which means there will be an interaction between the system and its surroundings. Here mainly discussion of work and heat interactions between the system and surroundings for a closed system has been done. In this closed system, there will be a energy transfer, but no mass transfer occurs. On the other hand, in an open system, there will be a transfer of mass; as the mass is being transferred, there won't be a fixed amount of matter to gain or lose energy.

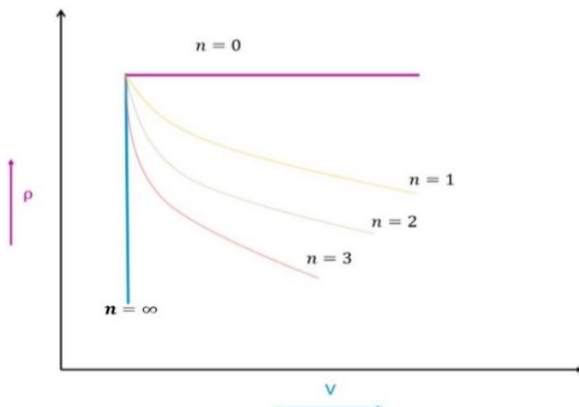


Fig. 1: Variation of closed system work as a function of index n .

Thermodynamic work is said to be done by a system if the sole effect on the things external to the system can be reduced to raising weight [1]. The weight may not be raised in actual cases by the effect can be that of raising weights. Based

on the direction, the work can be done by or on the system. The former is considered positive, whereas the latter is considered negative. Work is a path function because it not only depends on the initial and final state but also on the path followed between two states. The work can further be divided into many parts; one such category of work is non-dissipative/displacement work. For a closed system, it's the area under the $p - v$ graph as this is non-dissipative, so the path is quasistatic/reversible.

For a general polytropic process represented by process $pv^n = \text{constant}$, the work done depends on the index n . Fig. 1. Shows the impact of the variation of n on the area under the curve (work done by a closed system), whereas Table 1. tells about the different processes based on the index n .

TABLE I: Index n and its equivalent process [1]

| n | Process | $p - v$ relation |
|----------|---|------------------|
| 0 | Isobaric process ($p=\text{constant}$) | $p = c$ |
| 1 | Isothermal process for an ideal gas ($T=\text{constant}$) | $pv = c$ |
| 1.4 | Adiabatic process (no heat interaction) | $pv^\gamma = c$ |
| ∞ | Isochoric process (constant volume) | $v = c$ |

The general procedure is to integrate the process from some initial state to the final state to obtain the work interaction, as shown in Eq. (1). Therefore, the final expressions of work interaction in all the processes are listed in Table 2.

$$W = \int_1^2 p dv \tag{1}$$

TABLE II: Work in different processes

| Process | $p - v$ relation |
|--|---|
| Isobaric process ($p = c$) | $P(v_2 - v_1)$ |
| Isothermal process for an ideal gas ($pv = c$) | $p_1 v_1 \ln \frac{v_2}{v_1} = p_1 v_1 \ln \frac{p_1}{p_2}$ |
| Polytropic process ($pv^n = c$) | $\frac{p_1 v_1 - p_2 v_2}{n - 1}$ |
| Adiabatic process ($pv^\gamma = c$) | $\frac{p_1 v_1 - p_2 v_2}{\gamma - 1}$ |
| Isochoric process ($v = c$) | 0 |

Heat transfer is defined as the energy transformation between the system and surroundings due to the temperature difference. It is majorly categorized into conduction, convection, and radiation. In conduction, heat transfer is caused without bulk motion of the matter. Whereas in convection, the movement of matter causes heat transfer. And in radiation, the heat transfer is caused due to electromagnetic radiation. Moreover, it does not require any medium.

Heat is not a conservative property of a system, and it is a path function. The heat gained by the system is considered positive, whereas the one who is leaving the system is negative. An adiabatic process is the one in which the system does not interact with the surroundings in terms of heat. Specific heat is the amount of energy in the form of heat required to raise the temperature of a unit mass of a substance by a unit degree. Eq. (2) shows the expression for specific heat. The product of mass and specific heat is called the heat capacity of a substance, and it is denoted by C ($C = m \times c$). The specific heat remains independent of the process for solids and liquids, whereas, for gases, it must qualify the process. So, it is different for constant pressure and volume processes for a gas.

$$Q = mc\Delta t \tag{2}$$

Latent heat is the amount of energy in the form of heat required to cause the phase change of the unit mass of a substance. During latent heat transformation, neither temperature nor pressure changes. If L is the latent heat, then the total heat interaction for a unit mass of a substance during the phase change is given by:

$$Q = mL \tag{3}$$

Normally equations in Table 1 are solved using hand calculations. So, the chances of error flow into the calculations are there. However, if the process of computations is automated with the help of computer programmes, then the chances of error in moving into the calculation can be removed. Moreover, to better understand the process, one should also draw the equivalent p-v diagrams, which is sometimes difficult to do by hand calculations. However, for programs, it is just a matter of some data.

Here comes the importance of Python programming, which is very easy to implement [2–5]. The strength of Python lies in its modules, specially NumPy, Sympy, and Matplotlib, which are capable of solving any problem numerically and symbolically and plotting the data with only a few lines of code [6–10]. The power of NumPy lies in its array of objects. With the help of NumPY, one can write an understandable code which can be easily extended while keeping the code as simple as possible. The NumPy can do multiple tasks like limit evaluation, derivative & integral computations, equation solutions, matrix operation, ordinary differential equation (ODE) solution and much more [11,12]. The module named ‘pylab’ [13–15] has both ‘Numpy’ and ‘Matplotlib’ in it; hence, only one module has to be called to perform both the task of data calculation and visualization.

This research article uses Python programming to do work and heat calculations. The functions were developed for different processes, and the implementation has been explained. The developed functions will also plot the process. This article will equip the readers with a basic understanding of how-to implement the programming in basic thermodynamics.

II. IMPLEMENTATION OF WORK AND HEAT IN PYTHON

Table 3 shows the different python program functions developed to obtain work and heat interaction.

TABLE III: Python code development

| Explanation | Code |
|---|--|
| PyLab is imported | <code>from pylab import *</code> |
| Setting font and font size | <code>font = {'family' : 'Times New Roman', 'size' : 38}</code> <code>rc('font', **font)</code> |
| Function for constant volume is created | <pre>#-----# # Constant Volume Process #-----# def w_cv(p1,p2,v): """ Function for constant volume process Input: Pressure limits p1 & p2 (in kPa) and volume (m^3) Output: Work output and p-v plot """ figure(1,figsize=(20,15),dpi=300) p=linspace(min(p1,p2),max(p1,p2),30) v=zeros(30)+v plot(v,p,'k-',linewidth=4) xlabel('Volume (m\$^3\$)') ylabel('Pressure (kPa)') show() return 0</pre> |
| Function for constant pressure is created | <pre>#-----# ---# Constant Pressure Process #-----# def w_cp(p,v1,v2): """ Function for constant pressure process Input : Volume limits v1 & v2 (m^3) and pressure (kPa) Output: Work output and p-v plot """ work = p*(v2-v1) figure(2,figsize=(20,15),dpi=300)</pre> |

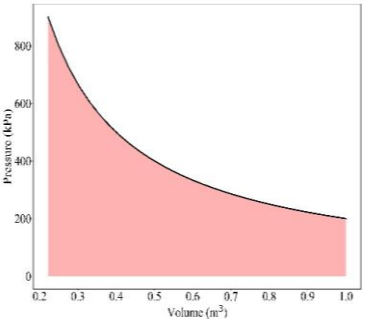
| | | | |
|---|--|--|--|
| | <pre>v=linspace(min(v1,v2),max(v1,v2),30) p=zeros(30)+p plot(v,p,'k-',linewidth=4) fill_between(v,p,color='red',alpha=0.3) xlabel('Volume (m^3)') ylabel('Pressure (kPa)') show() return round(work,3)</pre> | | <pre>work = (p1*v1-p2*v2)/(n-1) figure(2,figsize=(20,15),dpi=300) v=linspace(min(v1,v2),max(v1,v2),30) c=p1*v1**n p=c/v**n plot(v,p,'k-',linewidth=4) fill_between(v,p,color='red',alpha=0.3) xlabel('Volume (m^3)') ylabel('Pressure (kPa)') show() return round(work,3)</pre> |
| <p>Function for constant temperature is created</p> | <pre>#----- # Constant Temperature # Process #----- def w_ct(p1,v1,v2): """ Function for constant temperature process Input : Initial pressure (kPa) and volume (m^3) and final volume (m^3) Output: Work output and p-v plot """ work = p1*v1*log(v2/v1) figure(2,figsize=(20,15),dpi=300) v=linspace(min(v1,v2),max(v1,v2),30) c=p1*v1 p=c/v plot(v,p,'k-',linewidth=4) fill_between(v,p,color='red',alpha=0.3) xlabel('Volume (m^3)') ylabel('Pressure (kPa)') show() return round(work,3)</pre> | <p>Function for sensible heat interaction</p> <pre>#----- # Sensible Heat Transfer #----- def q_sh(m,c,T1,T2): """ Function for the evaluation of sensible heat transfer Input : mass (m), sp. heat (c), initial temp (T1), final temp (T2) Output: heat transfer """ return m*c*(T2-T1)</pre> | <pre>#----- # Heat transfer during phase # change #----- def q_mel_sol(m,Ti,Tf,T_pc,c_bpc,c_apc,L): """ Function for the evaluation of heat transfer during phase change Input : mass (m), initial temp (Ti), final temp (Tf), phase change temp (T_pc), sp. heat below phase change (c_bpc), sp. heat above phase change (c_apc), latent heat (L) Output: heat interaction """ if Ti>Tf: print('Process is either freezing or condensation') return m*(c_bpc*(T_pc-Ti)- L+c_apc*(Tf-T_pc)) else:</pre> |
| <p>Function for the polytropic process is created</p> | <pre>#----- # Polytropic Process #----- def w_pol(p1,p2,v1,v2,n): """ Function for polytropic process Input : p1 & p2 (kPa), v1 & v2 (m^3), and n (m^3) If the process is adiabatic write n=1.4 Output: Work output and p-v plot """</pre> | <p>Function for the heat transfer process involved in both sensible and latent heat</p> <pre>#----- # Heat transfer during phase # change #----- def q_mel_sol(m,Ti,Tf,T_pc,c_bpc,c_apc,L): """ Function for the evaluation of heat transfer during phase change Input : mass (m), initial temp (Ti), final temp (Tf), phase change temp (T_pc), sp. heat below phase change (c_bpc), sp. heat above phase change (c_apc), latent heat (L) Output: heat interaction """ if Ti>Tf: print('Process is either freezing or condensation') return m*(c_bpc*(T_pc-Ti)- L+c_apc*(Tf-T_pc)) else:</pre> | <pre>#----- # Heat transfer during phase # change #----- def q_mel_sol(m,Ti,Tf,T_pc,c_bpc,c_apc,L): """ Function for the evaluation of heat transfer during phase change Input : mass (m), initial temp (Ti), final temp (Tf), phase change temp (T_pc), sp. heat below phase change (c_bpc), sp. heat above phase change (c_apc), latent heat (L) Output: heat interaction """ if Ti>Tf: print('Process is either freezing or condensation') return m*(c_bpc*(T_pc-Ti)- L+c_apc*(Tf-T_pc)) else:</pre> |

| | |
|---|---|
| | <pre>print('Process is either melting or vaporization') return m*(c_bpc*(T_pc- Ti)+L+c_apc*(Tf-T_pc))</pre> |
| Function for stream function is developed | <pre>def stream_fun(ψ): u=diff(ψ,y) v=-diff(ψ,x) return u,v</pre> |

With the help of the above-developed functions, one can easily obtain the heat and work interactions for different problems and plots. The following examples will demonstrate the use of the developed functions.

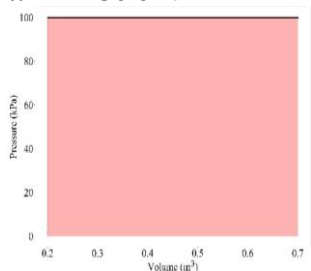
Example 1: Air having a mass of 1.2 kg is compressed from 0.2 MPa to 0.9 MPa according to the process $p v = c$. The initial gas density is 1.2 kg/m^3 . First, work done has to be obtained.

Python Approach:

| Code | Program Output |
|---|---|
| <pre># Input Data m=1.2 #kg p1=0.2E3 #kPa p2=0.9E3 #kPa ρ1=1.2 #kg/m^3 # Volume calculation at point 1 v1=m/ρ1 # Calculation of volume at point 2 v2=(p1*v1/p2) # The process is isothermal # The function used is w_ct Work = w_ct(p1,v1,v2) print('Work output =',Work)</pre> | <pre>Out[11]: -300.815 OR w = -300.815 kJ</pre>  |

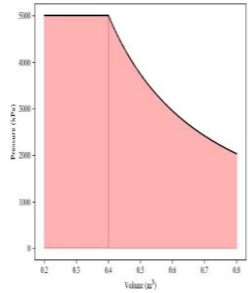
Example 2: Air with a mass of 1.2 kg at a pressure of 100 kPa is compressed isobarically from 0.7 m^3 to 0.2 m^3 . Work done has to be obtained.

Python Approach:

| Code | Program Output |
|--|---|
| <pre># Input Data p1=p2=100 #kPa v1=0.7 #m^3 v2=0.2 #m^3 # The process is isobaric # The function used is w_cp work = w_cp(p1,v1,v2) work</pre> | <pre>Out[13]: -50.0 OR w = -50.0 kJ</pre>  |

Example 3: A closed system performs two processes, viz. isobaric followed by polytropic processes (having index 1.3). The pressure at the beginning of the isobaric process is 50 bar, and the volume is 0.3 m^3 which increases to 0.4 m^3 at the end of the process. The volume at the end of a polytropic expansion is 0.8 m^3 . Total work has to be obtained.

Python Approach:

| Code | Program Output |
|---|---|
| <pre># Input data p1=p2=50E2 #kPa v1=0.2 #m^3 v2=0.4 #m^3 v3=0.8 #m^3 n=1.3 #polytropic index # evaluation of pressure at point 3 p3=p2*(v2/v3)**n # work has to be split into two parts # One for isobaric and the other for polytropic w_ib=w_cp(p1,v1,v2) w_ib w_pt=w_pol(p2,p3,v2,v3,n) w_pt font = {'family' : 'Times New Roman','size' : 10} rc('font', **font) figure(3,dpi=300) v=linspace(v1,v2,30) p=ones(30)*p1 plot(v,p,'k-') fill_between(v,p,color='red',alph a=0.3) v=linspace(v2,v3,30) c=p2*v2**n p=c/v**n</pre> | <pre>Out[17]: 2251.6 OR w = 2251.651 kJ</pre>  |

```

plot(v,p,'k-')
fill_between(v,p,color='red',alpha=0.3)

xlabel("Volume (m$^3$)")
ylabel("Pressure (kPa)")
savefig("Q3.jpg")
show()

work=w_ib+w_pt
work
    
```

Example 4: 650 kg of Tuna at 5°C are to be frozen and stored at -12°C. The specific heat above and below the freezing points (-2°C) are 3.182 and 1.717 kJ/kg-K, respectively. For a latent heat of fusion of 234.5 kJ/kg, the net heat removed from Tuna is to be evaluated.

Python Approach:

| Code | Program Output |
|---|---|
| <pre> # Input data m=650 #kg Ti=5 #deg C Tf=-12 #deg C T_pc=-2 #deg C c_bpc=3.182 #kJ/kg c_apc=1.717 #kJ/kg L=234.5 # Function calling Q_net=q_mel_sol(m,Ti,Tf,T_pc,c_bpc,c_apc,L) Q_net </pre> | <pre> Out[20]: -178063.6 OR Q = -178063.0 kJ </pre> |

III. CONCLUSION

In this research article, an attempt has been made to automate the process of heat and work for various thermodynamic processes with the help of Python programming. NumPy and Matplotlib (Pylab) have been used to develop functions for evaluating and plotting various heat and work interactions. The developed functions were tested against four examples, and it has been observed that the program results match exactly with the literature. Therefore, this research article will help beginners of thermodynamics to solve the problems related to work and heat transfer using Python. Furthermore, they can interpret the result using the article's approach.

REFERENCES

[1]. Nag PK. Engineering thermodynamics. Tata McGraw Hill; 2013.
 [2]. Dumka P, Sharma S, Gautam H, Mishra DR. Finite Volume Modelling of an Axisymmetric Cylindrical Fin using Python. Res Appl Therm Eng 2021;4:1–11.

[3]. Huei YC. Benefits and introduction to python programming for freshmen students using inexpensive robots. Proc. IEEE Int. Conf. Teaching, Assess. Learn. Eng. Learn. Futur. Now, TALE 2014, 2015, p. 12–7. doi:10.1109/TALE.2014.7062611.
 [4]. Moruzzi G. Python Basics and the Interactive Mode. Essent. Python Phys., Cham: Springer International Publishing; 2020, p. 1–39. doi:10.1007/978-3-030-45027-4_1.
 [5]. Dumka P, Pawar PS, Sauda A, Shukla G, Mishra DR. Application of He’s homotopy and perturbation method to solve heat transfer equations: A python approach. Adv Eng Softw 2022;170:103160. doi:10.1016/j.advengsoft.2022.103160.
 [6]. Cywiak M, Cywiak D. SymPy. Multi-Platform Graph. Program. with Kivy Basic Anal. Program. 2D, 3D, Stereosc. Des., Berkeley, CA: Apress; 2021, p. 173–90. doi:10.1007/978-1-4842-7113-1_11.
 [7]. Meurer A, Smith CP, Paprocki M, Čertík O, Kirpichev SB, Rocklin M, et al. SymPy: Symbolic computing in python. PeerJ Comput Sci 2017;2017:1–27. doi:10.7717/peerj-cs.103.
 [8]. Johansson R. Numerical python: Scientific computing and data science applications with numpy, SciPy and matplotlib, Second edition. Apress, Berkeley, CA; 2018. doi:10.1007/978-1-4842-4246-9.
 [9]. Dumka P, Chauhan R, Singh A, Singh G, Mishra D. Implementation of Buckingham ’ s Pi theorem using Python. Adv Eng Softw 2022;173:103232. doi:10.1016/j.advengsoft.2022.103232.
 [10]. Dumka P, Rana K, Pratap S, Tomar S, Pawar PS, Mishra DR. Modelling air standard thermodynamic cycles using python. Adv Eng Softw 2022;172:103186. doi:10.1016/j.advengsoft.2022.103186.
 [11]. Huang C. Python Solver for Stochastic Differential Equations 2011;34:1–13.
 [12]. Pawar PS, Mishra DR, Dumka P. Solving First Order Ordinary Differential Equations using Least Square Method: A comparative study. Int J Innov Sci Res Technol 2022;7:857–64.
 [13]. Ranjani J, Sheela A, Pandi Meena K. Combination of NumPy, SciPy and Matplotlib/Pylab-A good alternative methodology to MATLAB-A Comparative analysis. Proc. 1st Int. Conf. Innov. Inf. Commun. Technol. ICICT 2019, 2019, p. 1–5. doi:10.1109/ICICT1.2019.8741475.
 [14]. Kanagachidambaresan GR, Manohar Vinoothna G. Visualizations. In: Prakash KB, Kanagachidambaresan GR, editors. EAI/Springer Innov. Commun. Comput., Cham: Springer International Publishing; 2021, p. 15–21. doi:10.1007/978-3-030-57077-4_3.
 [15]. Pawar PS, Mishra DR, Dumka P, Pradesh M. OBTAINING EXACT SOLUTIONS OF VISCO-INCOMPRESSIBLE PARALLEL FLOWS USING PYTHON. Int J Eng Appl Sci Technol 2022;6:213–7.