

First Law of Thermodynamics for Closed System: A Python Approach

¹Krishna Gajula, ²Vikrant Sharma, ³Dhananjay R. Mishra, ^{4*}Pankaj Dumka
^{1,2,3,4}Department of Mechanical Engineering, Jaypee University of Engineering and Technology, A.B. Road, Raghogarh-473226, Guna (India)

***Corresponding Author**
E-mail Id: p.dumka.ipeec@gmail.com

ABSTRACT

In this paper, an attempt has been made to develop a Python module for evaluating the first law of thermodynamics, which includes the process of work done and amount of heat gained or lost by the system and the amount of internal energy stored. The modules NumPy and Matplotlib were used to perform the stipulated task. In addition, the correctness of codes was checked against different numerical problems, and it has been observed that the program results match exactly with the results in the literature. As a result, the functions thus developed have shown high accuracy with the least effort and error in all the cases.

Keywords: Python module, NumPy and Matplotlib.

1. Introduction

Rudolf Clausius gave the first law of thermodynamics in 1850. The first law of thermodynamics describes that the total energy in any system is a conserved quantity. In this system, energy cannot be created or destroyed. If we consider a closed system, the work and heat are the

forms of energy. Let us consider the closed cyclic process in which the work done by the system or on the system is directly proportional to the heat transfer along the path; the proportionality constant is called the joules equivalent or the mechanical equivalent of heat [1].

$$\Sigma W_{cycle} = J \Sigma Q_{cycle} \quad (1)$$

Or, it can also be expressed in the form [1]:

$$\oint dW = J \oint dQ \quad (2)$$

The above equation is applicable for systems undergoing a cyclic process. Still, if a system undergoes a change of state (i.e., a process) during which both heat

transfer and work transfer are involved, then the net energy transfer will be stored or accumulated within the system as shown in Eq (3) [1]Nag.

$$Q - W = \Delta E \quad (3)$$

The stored energy has a definite value at an intermediate state of a system; hence, it is a point function, and energy E is an

extensive property. The specific energy, i.e. energy per unit mass, is written as:

$$e = \frac{E}{M} \quad (4)$$

The ΔE is the stored amount of energy that can be split down into two modes. Macro and Microscopic modes.

Macroscopic energy mode includes macroscopic kinetic energy and potential energy of a system, and Microscopic energy mode includes various forms like Translational KE, Rotational KE,

Vibrational energy, Nuclear binding energy, Electron energy and Internal energy. But we will consider only internal energy as stored energy because when we compare the values of other forms of energy, they are quite negligible. So, the modified equation of the first law of thermodynamics for a closed system undergoing a process is given as:

$$Q = \Delta U + W \quad (5)$$

U is an extensive property of the system. The specific internal energy u equals $\frac{U}{m}$, and its unit is Joule per Kg. The differential form of the first law of thermodynamics is represented as [1]:

$$dQ = dU + dW \quad (6)$$

If we consider the work done as a non-dissipative $p dv$ work, then the equation of the first law of thermodynamics becomes [1,2]:

$$dW = dU + p dv \quad (7)$$

Specific heat at constant volume is defined as the rate of change of specific internal energy with respect to the temperature when the volume is kept constant.

$$c_v = \left(\frac{\partial u}{\partial t} \right)_v \quad (8)$$

The first law may be written as $dQ = du + p dv$ for a process without work other than $p dv$ work. Now, $p dv = 0$ as volume remains constant.

$$Q_v = \int_{T_1}^{T_2} c_v dT \quad (9)$$

For constant c_v , the internal energy and heat transfer can be evaluated as:

$$Q_v = \Delta u = c_v(T_2 - T_1) \quad (10)$$

Enthalpy is an accounting property which is defined as the sum of internal energy and PV product; in specific sense it is defined as:

$$h = u + pV \quad (11)$$

Specific heat at constant pressure is defined as the rate of change of specific enthalpy with respect to temperature when the pressure is held constant as [2]:

$$C_p = \left(\frac{\partial h}{\partial t} \right)_p \quad (12)$$

Solving the thermodynamic problems with the help of the above equations on pen and paper is sometimes boring, time-consuming, and may also lead to numerical calculations. The task of problem solving can be made error free if one automates the above process via a computer program. Moreover, the computations will be very fast compared to the hand calculations, and will be accurate. To simplify the solution and implement the thermodynamic equations mentioned above, the Python programming language is most suited as its syntax is very easy and its modules are very powerful [3–6]. To solve problems symbolically, SymPy is a very powerful tool which is also written in Python [7–10]. It is so powerful that once an equation is entered, it can integrate, differentiate, simplify expression, evaluate limit, and much more with great ease. Comprehensible code writing is possible with the help of SymPy without losing its simplicity [11]. Apart from SymPy, NumPy and Matplotlib are very useful modules for numerical computations and data visualization [7,12–15]. The strength of NumPy is 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 also perform multiple tasks like matrix operation, equation solutions, limit evaluation, derivative & integral computations, ordinary differential equation (ODEs)

solution, etc. [11,16]. The module named 'pylab' [17–20] 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.

In this research article, the authors have used Python programming to solve numerically the thermodynamic problems involving energy conservation, i.e., the first law. The codes were developed along with their implementation for different aspects of the first law for closed systems viz cycle or process. This article will equip the readers with a basic understanding of how to implement the programming to solve the first law of thermodynamics.

2. Implementing the First law for a closed system in Python

In this section, different numerical problems have been solved with the help of Python programming. The problems discussed here range from numeric calculations to symbolic computations.

Example 1: If the amount of heat transferred by the air conditioner is 234 kJ and the energy consumed in one hour is 456 kJ, then find the amount of work done in one hour and the nature of work done [1].

Solution: Methodology will first input the values of Q and E and then, by using Eq. (3), evaluate the W. Once W is known, then compare whether it is less than, greater than or equal to zero.

Code	Output
<pre>Q= float(input("enter the heat transfer : ")) ΔE= float(input("enter the internal energy: ")) W= Q-ΔE if (W>0): print("Work = ",W) print("work is done by the system") elif (W<0): print("Work = ",W) print("work is done on the system") else: print("No work interaction")</pre>	<pre>enter the heat transfer : 234 enter the internal energy: 456 Work = -222.0 work is done on the system</pre>

Example 2: If the gas contained in a piston-cylinder performs the work of 200 kJ and the heat lost by the cylinder is 100 kJ. Then what is the change in internal energy of the piston cylinder [1]?

Solution: Methodology will first input the values of Q and W and then, by using Eq. (3), evaluate the ΔE .

CODE	OUTPUT
<pre>Q= float(input("enter the heat transfer : ")) W= float(input("enter the work done : ")) ΔE= Q-W print("ΔE = ",ΔE)</pre>	<pre>enter the heat transfer100 enter the work done 200 ΔE = -100.0</pre>

Example 3: The heat capacity at the constant volume of a certain system is 1.5 kJ/kgK, and the system performs the work of 500 kJ due to a change in its temperature from 30°C to 70°C. Then what does the system do the net heat transfer?

Solution: Methodology will be to first input the values of c_v , T_1 , and T_2 and evaluate ΔU with the help of Eq. (10). Once ΔU is known the value of W has to be inserted to evaluate Q with the help of Eq. (3) evaluate the ΔE .

CODE	OUTPUT
<pre>Cv= float(input("enter the specific heat at constant volume: ")) T1= float(input("enter the initial temp: ")) T2= float(input("enter the final temp: ")) ΔU= Cv*(T2-T1) print("ΔU : ",ΔU) W= float(input("enter the work done : ")) Q= ΔU+W print("Q = ",Q)</pre>	<pre>enter the specific heat at constant volume: 1.5 enter the intial temp: 30 enter the final temp: 70 ΔU : 60.0 enter the workdone500 Q = 560.0</pre>

Example 4: The properties of a certain fluid are related as follows:

$$u = 196 + 0.718 \times T$$

$$pv = 0.287 (T + 273)$$

Where u is the specific internal energy (kJ/kg), T is in °C, p is pressure (kN/m²), and v is the specific volume (m³/kg). For this fluid, find c_p and c_v [1]?

Solution: The methodology will be to make functions of u and pv by using Function() function in SymPy and then supplying the expressions. Then h is obtained by adding them and using the diff() function to get the specific heats.

CODE	OUTPUT
<pre> from sympy import* from sympy.abc import* T= Symbol('T') u= Function('u')(T) u= (input("Enter u : ")) pv= Function('pv')(T) pv= (input("Enter pv : ")) h=u+" "+pv print("h = ",h) cp=diff(h,T) cv= diff(u,T) print(f"cv= {cv}\ncp={cp}") </pre>	<pre> Enter u : 196+0.718*T Enter pv : 0.287*(T+273) h = 196+0.718*T+0.287*(T+273) cv= 0.7180000000000000 cp=1.0050000000000000 </pre>

Example 5: The heat capacity at a constant pressure of a certain system is a function of temperature only and may be expressed as

$$C_p = 2.093 + \frac{41.87}{t + 100} J/^{\circ}C$$

where t is the temperature of the system in $^{\circ}C$. The system is heated while it is maintained at a pressure of 1 atmosphere until its volume increases from 2000 to 2400 cm^3 and its temperature increases from $0^{\circ}C$ to $100^{\circ}C$ [1].

- How much does the internal energy of the system increase?
- Find the magnitude of heat interaction.

Solution: The methodology will be to make functions of c_p and then supply the expressions to it. Then integrate() function is used to integrate it within limits to obtain Q . Then pressure and volumes are entered, and as Q is known, then ΔU can be evaluated easily by Eq. (5).

CODE	OUTPUT
<pre> from sympy import* from sympy.abc import* Cp = Function(T) Cp= input('enter the Cp as a function of T') Q=float(integrate(Cp, (T,0,100)))) P= float(input('enter the pressure : ')) V1= float(input('enter the V1: ')) V2= float(input('enter the V2: ')) W= float(P*(V2-V1)) ΔU=Q-W print("Q = ",round(Q,3)) print("ΔU = ",round(ΔU,3)) </pre>	<pre> enter the Cp as a function of T:2.093+41.87/(T+100) enter the pressure : 101325 enter the V1: 0.002 enter the V2: 0.0024 Q = 238.322 ΔU = 197.792 </pre>

Example 6: A gas undergoes a thermodynamic cycle consisting of the following processes [1]:

- Process 1-2: Constant pressure $p = 1.4$ bar, $V_1 = 0.028$ m^3 , $W_{12} = 10.5$ kJ.

- Process 2-3: Compression with $pV = \text{Constant}$, $U_3 = U_2$.
 - Process 3-1: Constant volume, $U_1 - U_3 = 26 \text{ kJ}$. There are no significant changes in KE and PE.
- (a) Calculate the network for the cycle in kJ. (b) Calculate the heat transfer for process 1-2 (c) Show that $2Q = 2W$ cycle.

Solution: Methodology will first define the no. of process and give the numerical data to different properties. Then NumPy arrays of Q , ΔU , and W are made. Next, and based on the values given for other

processes, supply the data to different arrays. And finally, use $\Sigma Q = \Sigma W$ to obtain the remaining heat and work interactions.

CODE	OUTPUT
<pre> from numpy import * # Number of processes #----- n=3 #Thermodynamic property data #----- p1=p2=140 v1=v3=0.028 # Array creation #----- Q=zeros(n) w=zeros(n) Δu=zeros(n) # Process 1-2 #----- w[0]=10.5 v2=(w[0]/p1)+v1 # Process 2-3 #----- w[1]=p2*v2*log(v3/v2) Δu[1]=0 Q[1]=Δu[1]+w[1] # Process 3-1 #----- w[-1]=0 Δu[-1]=-26.4 # Cycl integral of u=0 #----- Δu[0]=-sum(Δu[1:]) Q[0]=Δu[0]+w[0] # From first law ΣQ=Σw #----- Q[-1]=sum(w)-sum(Q[:2]) print("Q :",Q) print("W :",w) print("Δu :",Δu) </pre>	<pre> Q : [74.02792292 -150. 22.] W : [-103.97207708 0. 50.] Δu : [178. -150. -28.] </pre>

Example 7: A gas undergoes a thermodynamic cycle consisting of three processes beginning at an initial state where $p_1 = 1 \text{ bar}$, $V_1 = 1.5 \text{ m}^3$ and $U_1 = 512 \text{ kJ}$. The processes are as follows [1]:

- Process 1-2: Compression with $pV = \text{constant}$ to $p_2 = 2 \text{ bar}$, $U_2 = 690 \text{ kJ}$
- Process 2-3: $W_{23} = 0$, $Q_{23} = -150 \text{ kJ}$
- Process 3-1: $W_{31} = +50 \text{ kJ}$. Neglecting KE and PE changes determine the heat interactions Q_{12} and Q_{31} .

Solution: In this problem, a similar solution methodology will also be used as in the above example.

CODE	OUTPUT
<pre> from numpy import * # Number of processes #----- n=3 #Thermodynamic property data #----- p1=100 p2=200 v1=1.5 # Array creation #----- Q=zeros(n) w=zeros(n) Δu=zeros(n) # Process 1-2 #----- Δu[0]=690-512 w[0]=p1*v1*log(p1/p2) Q[0]=Δu[0]+w[0] # Process 2-3 #----- w[1]=0 Q[1]=-150 Δu[1]=Q[1]-w[1] # Process 3-1 #----- w[-1]=50 # Cycl integral of u=0 #----- Δu[-1]=-sum(Δu[:2]) # From first law ΣQ=Σw #----- Q[-1]=sum(w)-sum(Q[:2]) print("Q : ",Q) print("W : ",w) print("Δu : ",Δu) </pre>	<pre> Q : [74.02792292 -150. 22.] W : [-103.97207708 0. 50.] Δu : [178. -150. -28.] </pre>

Example 8: A fluid is confined in a cylinder by a spring-loaded, frictionless piston so that the pressure in the fluid is a linear function of the volume ($p = a +$

bV). The following equation gives the internal energy of the fluid:

$$U = 34 + 3.15 pV$$

where U is in kJ, p in kPa, and V in m^3 . If the fluid changes from an initial state of 170 kPa, 0.03 m^3 to a final state of 400 kPa, 0.06 m^3 , with no work other than that done on the piston, find the direction and magnitude of the work and heat transfer [1].

Solution: First, the functions of u and p are made. Then based on the p and v

values at the start and end, solve the equations for a and b . Once a and b are known integrate the expression for p within v_1 and v_2 to obtain work. Also, the change in internal energy can also be brought once limits are replaced in the function of u . Finally, heat interaction is obtained by Eq. (5).

Code	Output
<pre># Importing functions from sympy import * from sympy.abc import * # Defining Symbols vi,vf,pi,pf=symbols("vi vf pi pf") # Defining functions of p and u p=Function('p')(v) p=lambda a,b,v: a+b*v u=lambda p,v: 34+3.15*p*v # Equations to be solved for a and b eq1=Eq(p(a,b,vi),pi) eq2=Eq(p(a,b,vf),pf) a,b=solve([eq1,eq2],(a,b)).values() # Numerical calculations # values for initial and final p and v p1=170;v1=0.03;p2=400;v2=0.06 # obtaining values of a and b a=a.subs({pi:p1,vi:v1,pf:p2,vf:v2}) b=b.subs({pi:p1,vi:v1,pf:p2,vf:v2}) print(f"a = {a}; b = {b}") # obtaining work work=integrate(a+b*v,(v,[v1,v2])) print('W = ',round(work,2)) # obtaining change in internal energy Δu=u(p2,v2)-u(p1,v1) print("Δu = ",Δu) # Evaluation of heat Q=Δu+work print("Q = ", round(Q,2))</pre>	<pre>a = -60.0000000000000; b = 7666.66666666667 W = 8.55 Δu = 59.535 Q = 68.08</pre>

3. CONCLUSION

In this research article, an attempt has been made to automate the process of finding the amount of internal energy stored by the system and the amount of heat energy gained or lost by the system with the help of Python programming. NumPy, SciPy and Matplotlib(Pylab) have been used to develop functions for evaluating the work, heat, specific heats etc., for different processes.

The created functions were tested by taking certain eight examples, and it has been observed that the programme results match exactly with the literature. This research article will help beginners of thermodynamics to solve the problems related to the first law of thermodynamics (for a closed system) using Python. Furthermore, they can interpret the result by using the approach adopted in the research article.

REFERENCES

1. Nag PK. Engineering thermodynamics. Tata McGraw Hill; 2013.
2. Incropera FP, DeWitt DP, Bergman TL, Lavine AS, Incropera, F.P., Dewitt D.P., Bergman, T., Lavine A. Fundamentals of Heat and Mass Transfer. 8th ed. Hoboken, NJ: Wiley's; 2007. doi:10.1016/j.applthermaleng.2011.03.022.
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. Lin JWB. Why python is the next wave in earth sciences computing. Bull Am Meteorol Soc 2012;93:1823–4. doi:10.1175/BAMS-D-12-00148.1.
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. Dumka P, Deo A, Gajula K, Sharma V, Chauhan R, Mishra DR. Load and Load Duration Curves Using Python. Int J All Res Educ Sci Methods 2022;10:2127–34.
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. Rocklin M, Terrel AR. Symbolic statistics with SymPy. Comput Sci Eng 2012;14:88–93. doi:10.1109/MCSE.2012.56.
9. Rocklin M. Uncertainty Modeling with SymPy Stats. Proc 11th Python Sci Conf 2012:51–5. doi:10.25080/majora-54c7f2c8-009.
10. 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.
11. Huang C. Python Solver for Stochastic Differential Equations 2011;34:1–13.
12. 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.
13. 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.
14. 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
15. 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
16. 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.
17. 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.
18. 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.
19. 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.
20. Gajula K, Sharma V, Sharma B, Mishra DR, Dumka P. Modelling of Energy in Transit Using Python. Int J Innov Sci Res Technol 2022;7:1152–6.