National Public School, Indiranagar

Properties of the Temperature Distribution Curve of a Metal Bar

Class XI Physics Project 2019-2020

Anand Balivada 11 B
Aryan Mishra 11 B

Acknowledgements

We would like to thank our physics teacher Vinni Thomas ma’am for giving us this opportunity and providing valuable advice and perspectives about the investigations undertaken, and Sujata ma’am for providing us with materials and relevant components for our apparatus in the school lab. We would also like to thank our friends Sanidhya Dwivedi, who generously allowed us use his Arduino UNO board and with whose help we eliminated various possibilities for heat sources, and Anirudh S., who provided miscellaneous electronic equipment.

Finally, we would like to thank our parents for supporting our project, and with whom we had discussions that helped clarify our thought process, experimental design and improve various aspects of our investigation.

Contents

Introduction

In a metal bar, heat transfer is conductive and takes place in the direction opposite to the temperature gradient, i.e., from points of high temperature to low temperature.

After a metal bar is heated at the center for some time, and the heat source is removed, the heat diffuses across the bar, eventually distributing equally along its length. If there is no loss of heat from the ends of the bar through conduction, to the atmosphere by convection or as radiation, the total heat energy of the bar is conserved. The distribution of heat in the bar is a function of the position along the length of the bar and time elapsed (Heat per unit volume along the bar through time). The conservation of heat energy is equivalent to the statement that the area under the heat distribution curve is constant with respect to time. In section 3 (Theory), we outline how establishing this for the temperature distribution curve also yields conservation of energy.

Aim

We investigate the temperature distribution curves of a metal bar, heated at the center for a specified time interval, and experimentally establish the conservation of the area under the graphs of these curves and thus showing the conservation of energy in the bar. We measure the temperature at points along the length of the bar at specific points in time to generate 5 different temperature distribution curves for the bar. We compare these to numerical simulations for a one-dimensional bar under similar conditions, but with an imposed Neumann Boundary condition to represent insulation (ensuring conservation of area under the graphs).

We aim to experimentally establish the conservation of heat energy in a metal bar by showing that the area under the temperature distribution curve is constant with respect to time.

Theory

Consider a thin metal bar of length l (with endpoints at 0 and l), with constant mass density ρ and heat capacity c through-out its length and all instances in time, assumed to be isotropic (thermal expansion uniform in all directions). The variables x and t are the position along the bar and time elapsed respectively.

The change in internal energy per unit volume $\Delta Q(x,t) = \rho c \Delta T(x,t) \quad \dots (0)$
The change in temperature is directly proportional to that of the internal energy, and thus the area under the temperature distribution curve is proportional to the total heat energy.
Dividing by Δt and taking the limit $\Delta t \to 0$,

$$\frac{\partial Q}{\partial t} = \rho c \frac{\partial T}{\partial t} \qquad \dots (1)$$

For the heat flux q (Heat per unit time per unit area) through the point x at time t, we have Fourier's Law of Heat Conduction:

$$q(x,t) = -k\frac{\partial T}{\partial x} \quad \dots (2)$$

From conservation of energy, the sum of the internal energy per volume at a point and the amount of heat passing through the point per unit volume must be constant.

The amount of heat passing through the point is given by:

$$\frac{\partial}{\partial x}\int q(x,t)dt \ldots (3)$$

Our equation for the conservation of energy is:

$$\frac{\partial}{\partial x}\int q(x,t)dt + Q(x,t) = constant \ldots (4)$$

Differentiating the above with respect to t and rearranging, we obtain:

$$\frac{\partial Q}{\partial t} = -\frac{\partial q}{\partial x} \quad \ldots (5)$$

Substituting (2) in (5),

$$\frac{\partial Q}{\partial t} = k\frac{\partial^2 T}{\partial x^2} \quad \ldots (4)$$

Finally, substituting (2) in (6), we obtain the heat equation:

$$\frac{\partial T}{\partial t} = \frac{k}{c\rho}\frac{\partial^2 T}{\partial x^2} = \alpha\frac{\partial^2 T}{\partial x^2} \quad \ldots (7) \quad \text{where } \alpha = \frac{k}{c\rho} \text{ is the thermal diffusivity.}$$

The heat equation neglects radiative and convective heat loss, and we impose the Neumann Boundary Condition $\frac{\partial T}{\partial x}(0,t) = \frac{\partial T}{\partial x}(l,t) = 0$ ... (8) , to ensure no conductive heat loss.

The solution of (5) coupled with (8) satisfies $\int_0^l T(x,t)dx = constant \quad \ldots (9)$

See [2] for the derivation of (5) and a proof of (9).

The numerical simulations carried out were under these idealized conditions.

Materials Required

Apparatus: K-type Thermocouple and MAX6675 module (See [1] for details), Arduino Uno Board, Jumper wires, 2 equally leveled bricks, Water Bath/ Bucket filled with water, Metal bar/ ruler and candle.

Experimental Design

The experiment is carried out by measuring the temperature over 120 seconds (at 0, 24, 48, 72 and 96 seconds after removing heat source; leaving 24 seconds after) at seven equally spaced points that span the length of the ruler (two of these are the ends, and one is the centre).

We only possessed a single sensor, with which one cannot simultaneously measure the temperature at the seven different points, i.e., directly obtaining the temperature distribution. To overcome this limitation in resources, we devised a method to indirectly obtain the temperature distribution curves over time. We measure the temperature at one of the seven points on the scale over our desired time interval, cool the ruler to a fixed temperature, and replicate the same initial conditions (temperature at the centre of the ruler before heating, time of heating the ruler, and the temperature at the centre after heating and removing source) and repeat the process for all seven points.

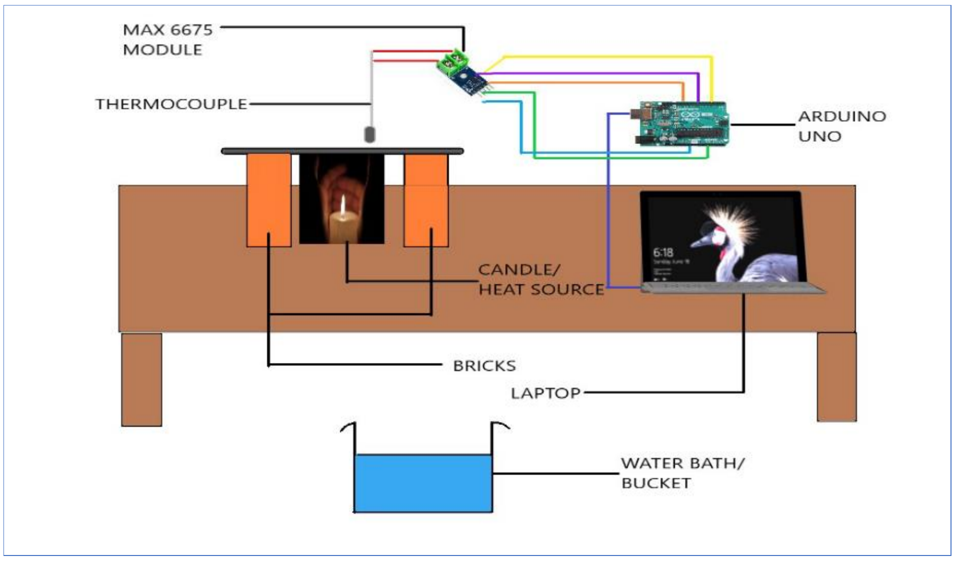


Fig 1. Schematic Representation of Experimental setup

The experiment generates 35 data points; 5 temperature readings over time at each of the seven points. While we couldn't directly measure the temperature distribution, the successful replication of the initial conditions allowed us to combine the temperature readings over the seven points at each instance in time. To ensure that the initial conditions could be replicated, we had to verify that a desired temperature could be attained at the centre of the ruler after heating the ruler for a fixed amount of time.

After measuring the temperature of the ruler's centre to be the same as the ambient temperature (302K), we heated it until it reached a desired temperature (here, 358K). We cooled it for a specified interval of time (after removing heat source and waiting 120 seconds, we cooled it in water and wiped it dry for 120 seconds) and found the temperature at the centre to be 307K. We heated the ruler again and this time, while the ruler was heating, the temperature at the centre was being measured in half-second intervals, and being logged and displayed along with the timestamp for the measurement, until 358K was reached. After cooling the ruler for the same specified amount of time (120 + 120), the centre temperature was the same as after the previous cooling (307K).

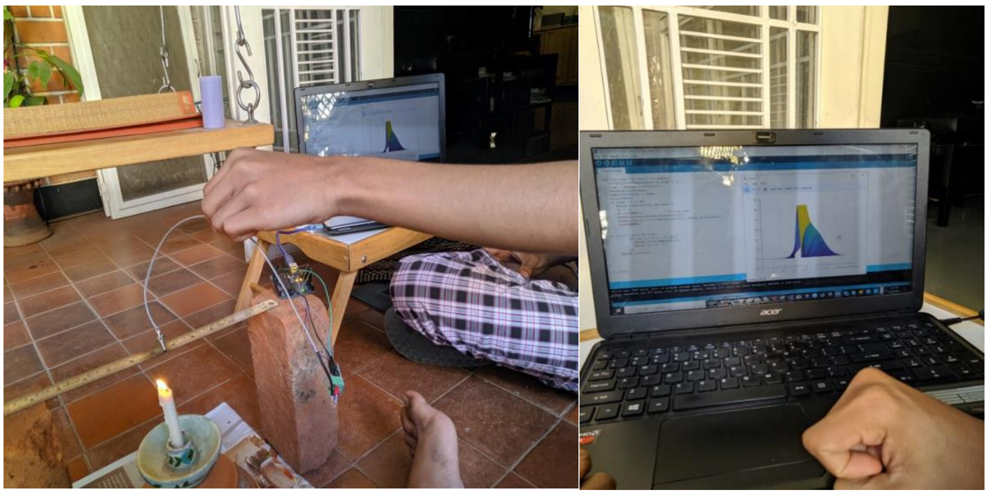
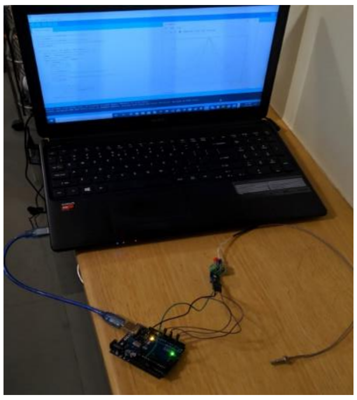


Fig 2. Experimental setup



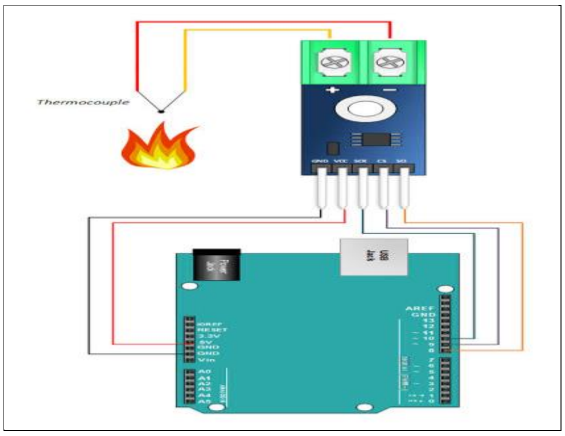


Fig 3. Arduino board-Laptop connections **Fig 4.** Schematic representation of Arduino board Connections

From the measurement log, we could find the time required to heat the ruler from 307K to 358K (14:09:54.442 - 14:12:04.290; ~130 seconds). We again heated the ruler until it reached 358K, and created another measurement log, from which we found that the time required to heat the ruler from 307K to 358K matched the previous log (14:15:57.373 - 14:18:06.728 ~130s). We standardized this time, along with the time we used to cool the scale (120+120), for all iterations of the experiment, thus ensuring replicability of the initial conditions. After cooling the ruler once again, we started the experiment.

Procedure

Note: We automated the time delays, measurement of temperatures, and tabulation of data using the Arduino UNO and used GNU octave for simulations.

1. Start heating the scale for a span of exactly 130 seconds.
2. Start noting down the temperature of the point at 0 cm mark taking 5 readings over 120 seconds, each reading after 24 seconds intervals.
3. Cool the bar down by immersing it in the water bath for 80 seconds and wipe it clean for 40 seconds.
4. Repeat the above procedure another 6 times, taking temperature readings at the points at 5.39cm, 10.78cm, 16.17cm, 21.56cm, 26.95cm, 32.34 cm marks.
5. Plot temperature vs. position graphs for each instance in time, and using code (ii) from Appendix A, calculate the area under each of these graphs.
6. Run the code from Appendix A in GNU Octave, to obtain the simulation plots, and the areas under the graphs.

Results

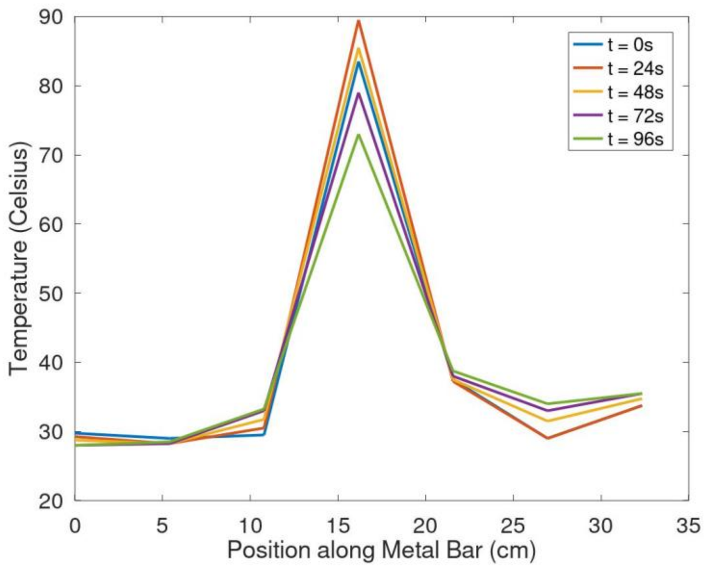


Fig 5. Experimentally obtained temperature distribution graphs for the bar at 24-second intervals.

Table 1. Experimentally obtained temperature distribution of bar over time and areas under the graphs (Celsius)

Time	**0cm**	**5.39cm**	**10.78cm**	**16.17cm**	**21.56cm**	**26.95cm**	**32.34cm**	**Area ([L][T])**
t = 0s	29.75	29.00	29.50	83.50	37.50	29.00	33.75	1294.9
t = 24s	29.25	28.25	30.50	89.50	37.25	29.00	33.75	1325.9
t = 48s	28.75	28.25	31.75	85.50	37.50	31.50	34.75	1327.3
t = 72s	28.00	28.25	33.00	79.00	38.00	33.00	35.50	1309.8
t = 96s	28.00	28.50	33.25	73.00	38.75	34.00	35.50	1289.6

Mean Area = 1309.5 $\pm$ 13.798

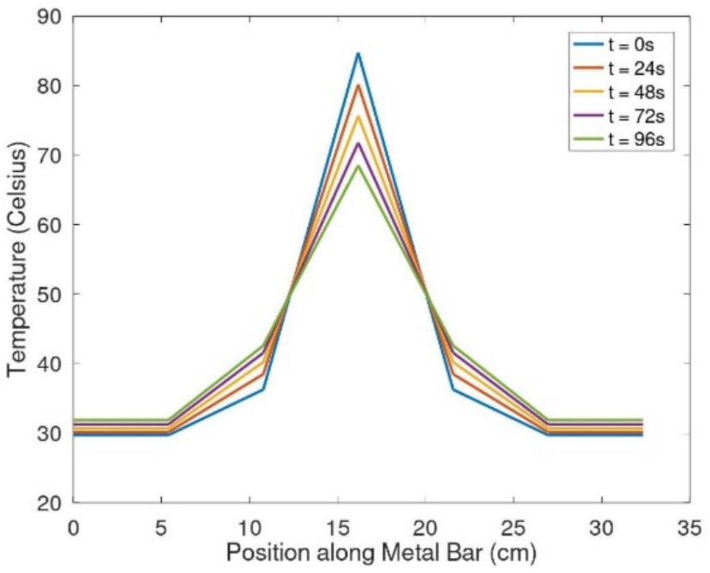


Fig 6. Simulated temperature distribution (after heating for 130 seconds and allowing to dissipate for 120 seconds), at 24 second intervals.

Table 2. Numerically simulated temperature distribution of bar over time and areas under graphs (Celsius)

Time	**0cm**	**5.39cm**	**10.78cm**	**16.17cm**	**21.56cm**	**26.95cm**	**32.34cm**	**Area ([L][T])**
t = 0s	29.569	29.569	35.470	84.763	35.470	29.569	29.569	1129.2
t = 24s	29.880	29.880	37.322	80.437	37.322	29.880	29.880	1130.6
t = 48s	30.255	30.255	38.841	76.648	38.841	30.255	30.255	1132.3
t = 72s	30.675	30.675	40.084	73.322	40.084	30.675	30.675	1134.3
t = 96s	31.603	31.603	41.913	67.810	41.913	31.603	31.603	1136.3

Mean Area = 1132.5±2.208

Conclusions and Further Directions

The results demonstrate the conservation of area under the experimentally obtained curves with an error of 1 %. The mean percentage error of the experimentally obtained curves from the simulated curves was 8.78 %.

These results are encouraging, and there are many further steps that can be taken to enhance the quality and accuracy of both the simulation and the experiment, and minimize the mean deviation. Using seven different sensors to measure the temperature distribution directly would be more accurate than the method used in this investigation. Furthermore, it would eliminate the need to carry out the experiment in many iterations, ensure the replicability of the initial condition, cooling with water, and minimize the time required for the entire experimental procedure. As a consequence of these improvements, multiple sensors would also allow taking readings at small time intervals (considerably smaller than 24 seconds) and at larger time scales (long enough to see the complete dissipation of heat throughout the ruler).

With regard to the simulation and calculation of area for experimental curves, we can interpolate the data points with continuous curves (See Chapter 3 of [3] for details on interpolation) rather than with lines between the curves, as done by Octave, and thus use more points for the numerical integration of these curves.

Apart from refining the methods used, we can even conduct different investigations, such as heating the ruler at multiple points initially, potentially exploring a myriad initial condition.

We hope to carry out at least some of these improvements in the future.

Precautions

1. Use a k-type thermocouple as the voltage output scales linearly with the temperature; this ensures the accuracy of the measurement.
2. Use two leveled bricks as support for the scale to make sure that the same point is heated during each iteration of the experiment.
3. Do not make direct contact with the ruler except at the ends.
4. Hold the thermocouple at a point before the rod, in order to prevent bias in the readings and burns due to high temperature.

References

[1] Thermocouple details: https://www.amazon.in/Robocraze-MAX6675-Thermocouple-Temperature-Arduino/dp/B07NSZ82QG

[2] Churchill, Ruel V., Brown, James Ward., Fourier Series and Boundary Value Problems.

[3] Flannery, B., Teukolsky, S., Vetterling, W., Press, W., Numerical Recipes in C.

[4] Strikwerda, John., Finite Difference Schemes for Partial Differential Equations.

Appendix A (Simulation and Area Calculation Code in GNU Octave)

Simulation of temperature distribution on metal ruler of length 32.34cm, using the 1-dimensional heat equation, with Neumann boundary condition of 0 on both ends. The ruler is heated from 307K (34 C) at the centre to 358K (85C) over 130 seconds, assuming a linear increase in temperature over this span of time. The heat source is removed and 120 seconds of heat transfer are simulated. The temperature distribution graphs presented in section 7 are taken as 0s, 24s,…,96s with respect to the time the heat source is removed.

For the formulation of and derivation of stability of the numerical scheme (Forward-Time, Central-Space Euler Method), see [4].

Simulation Code:

```
# Values of Physical Significance.

# Units: seconds (time), cm (length), Celsius (Temperature)

scale_length = 32.34

flame_temperature = 85
```

```
room_temperature = 29

base_temp = 34

heat_time = 130

cool_time = 120

time_scale = 1

tot_time = (heat_time + cool_time)*time_scale

alpha = 0.042                                        #thermal diffusivity

# Values relevant to numerical scheme (Forward-Time, Central-Space Euler Method for 1D Heat Equation with Neumann Boundary
Condition)

# Mesh sizes and interval lengths

scale_xmesh = 7/21

xmesh = ceil(21*scale_xmesh)

tmesh = floor(200*time_scale)

x = linspace(0,scale_length, xmesh);

t = linspace(0, tot_time, tmesh);

h = scale_length/xmesh

dt = tot_time/tmesh

l = dt/(h^2)

f = alpha*l                       # |f| < 0.5 for stability of scheme

disp(f)

# Initial Conditions

v = []

v(1, 1:floor(xmesh/2)) = room_temperature

v(1, ceil(xmesh/2)) = base_temp;

v(1, ceil(xmesh/2) + 1:xmesh) = room_temperature;

#phase 1; heating for heat_time seconds to generate appropriate initial conditions for the area-preservation verification step

for n = 1:ceil((heat_time/tot_time)*tmesh)

  for i = 2:xmesh-1

    v(n+1, i) = v(n, i) + f*(v(n, i+1) - 2*v(n, i) + v(n, i-1));

  endfor

  v(n+1, ceil(xmesh/2)) = ((flame_temperature-base_temp)/(heat_time))*t(n) + base_temp;        #Assuming a linear increase in temperature
at  # the centre

  #Neumann Boundary Condition of du/dx = 0:

  v(n + 1 , xmesh) = v(n + 1, xmesh-1);

  v(n+1, 1) = v(n+1, 2);

endfor

# phase 2; allowing heat to transfer through scale and to air without external heat source for tot_time - heat_time seconds

for n = ceil((heat_time/tot_time)*tmesh)+1:tmesh-1

  for i = 2:xmesh-1

    v(n+1, i) = v(n, i) + f*(v(n, i+1) - 2*v(n, i) + v(n, i-1));
  endfor
  #Neumann Boundary Condition of du/dx = 0:
  v(n + 1 , xmesh) = v(n + 1, xmesh-1);
  v(n+1, 1) = v(n+1, 2);
endfor
  # areas under the select curves
for n = ceil((heat_time/tot_time)*tmesh): ceil((cool_time/tot_time)*tmesh/5): tmesh
 a = 0
  for xi = 1:xmesh-1
    a += 0.5*h*(v(n, xi) + v(n, xi+1));
  endfor
  plot(x,t,v(n,1:xmesh))
  hold on
  disp(a)
endfor
```

Appendix B (Code for Arduino Uno and Thermocouple)

```
#include <max6675.h>
int soPin = 10;
int csPin = 7;
int sckPin = 4;

MAX6675 robojax(sckPin, csPin, soPin);
double temps[5][7];
int st = 0;
double temp1 = 0;
double temp2 = 0;
double calibrate_value;

void setup() {
  Serial.begin(9600);
  Serial.println("Robojax MAX6675");
  calibrate_value = robojax.readCelsius();
  Serial.println(calibrate_value);
    while (true){
      calibrate_value = robojax.readCelsius();
      Serial.println(calibrate_value);
        if ((calibrate_value >= 33) && (calibrate_value <= 35)){
          Serial.println("Start heating");
          break;
        }
        delay(500);
    }

  Serial.println("Start Heating now for 2 minutes 10 seconds");
  delay(130000);
  Serial.println("Stop Heating");

  for (int iter = 0; iter < 35; iter++){
    Serial.print("Temperature at point = ");
    temp1 = robojax.readCelsius();
    Serial.println(temp1);
    temps[iter%5][st] = temp1;
    delay(24000);
    if (iter % 5 == 4){
      Serial.println("Cool the scale");
      st += 1;
      delay(120000);          // cooling time
      Serial.println(robojax.readCelsius());
      Serial.println("Start Heating now for 2 minutes 10 seconds");
      delay(130000);
    }
}
  for (int i = 0; i <5; i++){
          for (int j = 0; j < 7; j++){
            Serial.print(temps[i][j]);
            Serial.print(" ");
           }
      Serial.println();
  }
}
void loop() {
}
```