Stefan-Boltzman Radiation Law

Zekeriya Gökhan Evelek
Boğaziçi University • Physics Department

***Abstract*—In this experiment, we have tested both the inverse square law and Stefan Boltzmann Radiation Law. We started by measuring lamp's(radiation source for part 1 and 2) resistance at room temperature to estimate the T as a function of Radiation. We showed that the radiation is proportional to 1.95±0.117, which is 0.38 σs away from the expected value 2. Then, we have measured the n value, proportionality of Temperature with the radiation, for various distances at high temperature and low temperature and have used exponential and log-scale line fits to get n as $n = 4.040 \pm 0.022$ which is 1.82 σs away form the expected value of 4.**

I. Theory

In 1877, Josef Stephan has deduced that the thermal radiation is related to the fourth power of the temperature from the measruements of infrared emissions by John Tydall in 1860 and he has published it on his paper called Über die Beziehung zwischen der Wärmestrahlung und der Temperatur (On the relationship between thermal radiation and temperature). [1]. Later on the theory is derived by Ludwig Boltzmann by using the works of Adolfo Bartoli. It was not so later that the theory was fully verified experimentally. This theory was used to explain many things such as calculating the temperature of stars. [2]

The energy of a photon:

$$E = pc = \hbar kc = \hbar w \tag{1}$$

so

$$k = \frac{w}{c} \tag{2}$$

density of states for photon:

$$g(k)dk = \frac{GV}{(2\pi)^3} 4\pi \frac{w^2}{c^2} \frac{1}{c} dw \tag{3}$$

G is the spin factor and 2 for photon

$$g(w)dw = \frac{V}{\pi^2 c^3} w^2 dw \tag{4}$$

energy density for a photon can be calculated from the formula:

$$e(w)\delta w = g(w)\delta w x f(w) x \epsilon(w) \tag{5}$$

$$e(w)dw = \frac{V}{\pi^2 c^3} w^2 dw \frac{1}{\exp(\frac{\hbar w}{k_B T}} - 1)\hbar w \tag{6}$$

so the total energy is calculated as

$$E = \frac{V}{\pi^2 c^3} \int_0^\infty w^3 \exp(\frac{\hbar w}{k_B T} - 1) dw \tag{7}$$

the energy per volume is:

$$\frac{E}{V} = \frac{{k_B}^4 \pi^2}{15\hbar^3 c^3} T^4 \tag{8}$$

$$\frac{E}{N} = \frac{4\sigma}{c} T^4 \tag{9}$$

where $\hbar$ is reduced Planck constant, V is the volume, c is the speed of light, k_B is the Boltzmann constant, w is the angular frequency, p is the momentum, k is the angular wavenumber, σ is the Stefan Boltzmann constant, T is the absolute temperature

Equation 9 gives the Stefan Boltzmann Law. It is assumed in the theory that photons are radiated in every direction so in order to detect the radiated photons, we need to use the inverse square law which implies that the intensity of the radiated photons is inversely proportional to the distance because of the surface area of sphere:

$$I = \frac{P}{A} = \frac{P}{4\pi r^2} \tag{10}$$

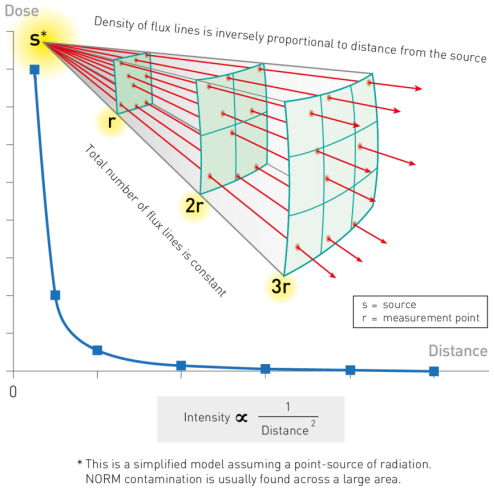

Fig. 1. visualization of the inverse square law

II. Method

We have conducted the experiment in 3 different parts, in addition to the determination of the resistance in the room temperature. Then we have tested the inverse square law by measuring for different distances. Thirdly, we have measured V_{sensor}, V_{lamp},I_{lamp}, at high temperatures for two different distances and finally V_{sensor} and T values at low temperatures, which are around 200-300 °C.

A. Part 0- Determination of R_{300K}

In this part we have calculated the resistance of the lamp by measuring its V and I values, at room temperature around 300K, which is to be used in the later calculations.

1) The voltage is applied to Stefan Boltzmann Lamp
2) The voltage and current values are read

B. Part 1- Inverse Square Law

In this part we have measured V_{sensor} values for 14 different distances to test the inverse square law

1) The voltage is applied to Stefan Boltzmann Lamp
2) The lamp creates radiation
3) The amplified(10^4) voltage values read for the radiation sensor
4) The lamp is repositined and the same process has been done again

C. Part 2- High Temperature Measurements

In this part the lamp is applied a high voltage to create high temperatures and V_{sensor}, V_{lamp},I_{lamp} are measured.

1) The lamp and radiation sensor are placed 7 cm and 10 cm apart
2) Relatively high voltage(around 2-3 V) is applied to Stefan Boltzmann Lamp
3) The lamp creates radiation
4) Voltage and current values on the lamp are read.
5) The amplified(10^4) voltage values read for the radiation sensor
6) The same process happens with increased voltage values to obtain 10 different data for 7 cm distance and 9 different data for 10 cm distance(there is no reason behind this, we have just forgot take one more in the 10 cm case)

D. Part 3- Low Temperature Measurements

In this part, we have Measured temperature and voltage values on the sensor at low temperatures(around 200-300 °C)

1) The electrical owen is heated
2) Voltage values and the temperature values are read for the radiation sensor which is placed right in front of the owen

III. The Experimental Setup

Here is the full list of apparatus we have used in the experiment

- Stefan Boltzmann Lamp
- Power Supply
- Voltage Amplifier
- Radiation Sensor
- Ruler
- Voltmeter 15V DC
- Ammeter 3A DC
- Optical Bench
- Electrical Owen
- Thermometer
- 1 kΩ resistor
- Connecting Leads

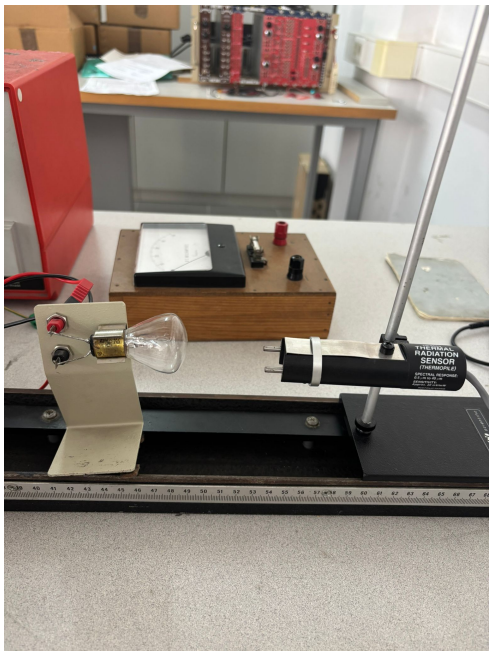

Fig. 2. Lamp and the radiation sensor

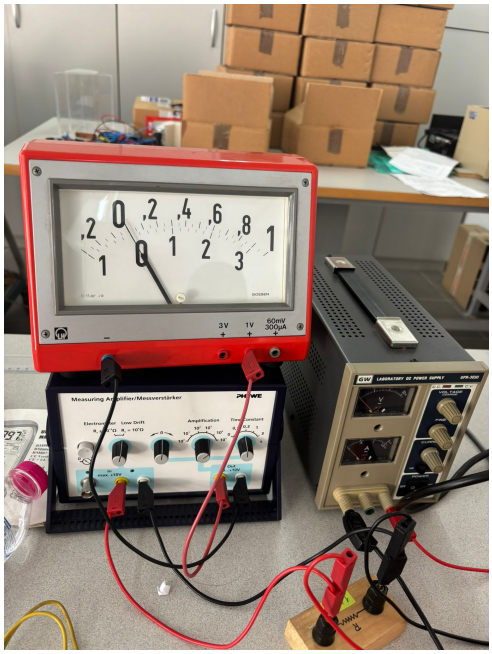

Fig. 3. Voltage Amplifier and lamp sources

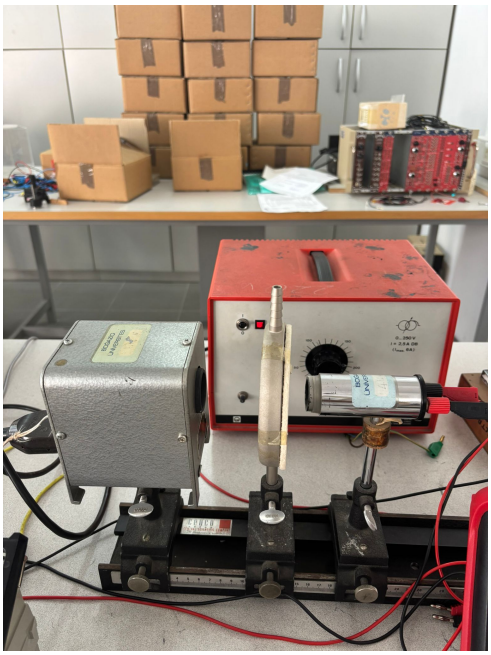

Fig. 4. Electrical owen and sensor

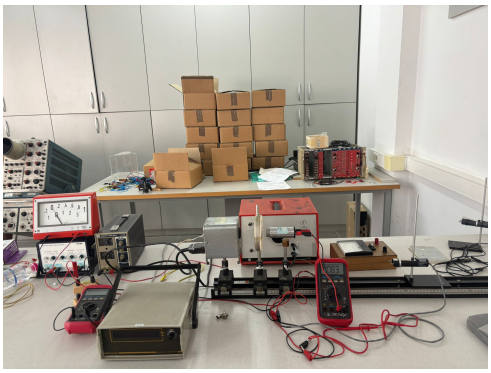

Fig. 5. the Apparatus

IV. The Data

Here's the dataset below

TABLE I
R_{300K} MEASUREMENT AT ROOM TEMPERATURE

Voltage (mV)	Current (mA)
34.7 ± 0.14	109 ± 1
43.8 ± 0.18	137 ± 1
50.0 ± 0.19	156 ± 1
56.6 ± 0.21	176 ± 1
63.2 ± 0.23	195 ± 1
68.9 ± 0.24	211 ± 1
77.8 ± 0.27	237 ± 1
85.5 ± 0.30	258 ± 1
95.4 ± 0.33	285 ± 1

V. The Analysis

In the analysis, we have used the following error propagation formula if there is no correlation between the variables:[3]

$$\sigma_f = \sqrt{(\frac{\partial f}{\partial x})^2(\sigma_x)^2 + (\frac{\partial f}{\partial y})^2(\sigma_y)^2 + (\frac{\partial f}{\partial z})^2(\sigma_z)^2 + ...} \quad (11)$$

TABLE II
GIVEN T AND R/R_{300K} VALUES [3]

Temperature (K)	R/R_{300K}
300 ± 100	1.00 ± 0.01
400 ± 100	1.43 ± 0.01
500 ± 100	1.87 ± 0.01
600 ± 100	2.34 ± 0.01
700 ± 100	2.85 ± 0.01
800 ± 100	3.36 ± 0.01
900 ± 100	3.88 ± 0.01
1000 ± 100	4.41 ± 0.01
1100 ± 100	4.95 ± 0.01
1200 ± 100	5.48 ± 0.01
1300 ± 100	6.03 ± 0.01
1400 ± 100	6.58 ± 0.01
1500 ± 100	7.14 ± 0.01
1600 ± 100	7.71 ± 0.01
1700 ± 100	8.28 ± 0.01
1800 ± 100	8.86 ± 0.01
1900 ± 100	9.44 ± 0.01
2000 ± 100	10.03 ± 0.01
2100 ± 100	10.63 ± 0.01
2200 ± 100	11.24 ± 0.01
2300 ± 100	11.84 ± 0.01
2400 ± 100	12.46 ± 0.01
2500 ± 100	13.08 ± 0.01
2600 ± 100	13.72 ± 0.01
2700 ± 100	14.34 ± 0.01
2800 ± 100	14.99 ± 0.01
2900 ± 100	15.63 ± 0.01
3000 ± 100	16.29 ± 0.01
3100 ± 100	16.95 ± 0.01
3200 ± 100	17.62 ± 0.01
3300 ± 100	18.28 ± 0.01
3400 ± 100	18.97 ± 0.01
3500 ± 100	19.66 ± 0.01

TABLE III
INVERSE SQUARE LAW MEASUREMENTS
LAMB VOLTAGE: 3.355 ± 0.001 V
LAMB CURRENT: 1.565 ± 0.001 A
VOLTAGE AMPLIFICATION FOR V_{sensor}: 10^4

Distance (cm)	V_sensor(V)
32.7 ± 0.1	0.06 ± 0.05
24.2 ± 0.1	0.10 ± 0.05
19.4 ± 0.1	0.15 ± 0.05
17.2 ± 0.1	0.20 ± 0.05
15.4 ± 0.1	0.25 ± 0.05
13.6 ± 0.1	0.30 ± 0.05
12.5 ± 0.1	0.35 ± 0.05
11.8 ± 0.1	0.40 ± 0.05
11.3 ± 0.1	0.45 ± 0.05
10.5 ± 0.1	0.50 ± 0.05
10.0 ± 0.1	0.55 ± 0.05
9.7 ± 0.1	0.60 ± 0.05
9.0 ± 0.1	0.70 ± 0.05
8.3 ± 0.1	0.80 ± 0.05

We have first plotted V vs I for the lamp so that we can get the resistance of it at room temperature around 300K:

TABLE IV
HIGH TEMPERATURE MEASUREMENTS FOR 7CM DISTANCE
VOLTAGE AMPLIFICATION FOR V_{sensor}: 10^4

V_sensor (V)	Lamp Voltage (V)	Lamp Current (A)
1.00 ± 0.05	3.932 ± 0.001	1.689 ± 0.001
0.95 ± 0.05	3.721 ± 0.001	1.644 ± 0.001
0.90 ± 0.05	3.661 ± 0.001	1.631 ± 0.001
0.85 ± 0.05	3.417 ± 0.001	1.579 ± 0.001
0.80 ± 0.05	3.246 ± 0.001	1.542 ± 0.001
0.75 ± 0.05	3.114 ± 0.001	1.513 ± 0.001
0.70 ± 0.05	2.953 ± 0.001	1.477 ± 0.001
0.65 ± 0.05	2.808 ± 0.001	1.444 ± 0.001
0.60 ± 0.05	2.648 ± 0.001	1.407 ± 0.001
0.50 ± 0.05	2.245 ± 0.001	1.311 ± 0.001

TABLE V
HIGH TEMPERATURE MEASUREMENTS FOR 10CM DISTANCE
VOLTAGE AMPLIFICATION FOR V_{sensor}: 10^4

V_sensor (V)	Lamp Voltage (V)	Lamp Current (A)
1.00 ± 0.05	4.860 ± 0.001	1.875 ± 0.001
0.95 ± 0.05	4.580 ± 0.001	1.818 ± 0.001
0.90 ± 0.05	4.390 ± 0.001	1.782 ± 0.001
0.85 ± 0.05	4.200 ± 0.001	1.742 ± 0.001
0.80 ± 0.05	3.990 ± 0.001	1.700 ± 0.001
0.75 ± 0.05	3.830 ± 0.001	1.665 ± 0.001
0.70 ± 0.05	3.660 ± 0.001	1.630 ± 0.001
0.65 ± 0.05	3.493 ± 0.001	1.594 ± 0.001
0.60 ± 0.05	3.298 ± 0.001	1.553 ± 0.001

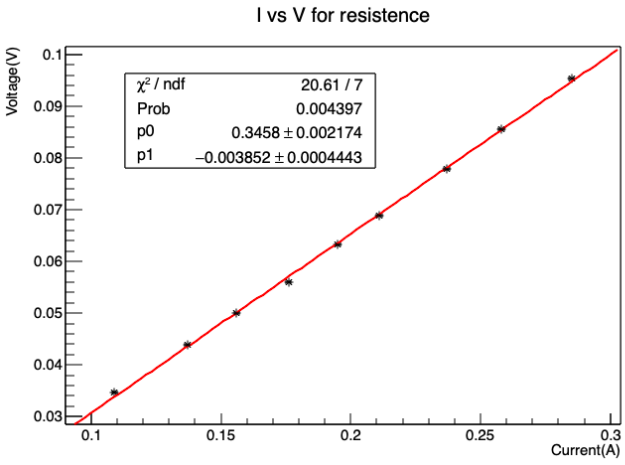

Fig. 6. Determination of Reference R_{300K} from I vs V measurements at room temperature around 300K

Here we have found $R_{300K} = 0.3458 \pm 0.002174$.

By using the dataset at table 2, we have plotted R/R_{300K} vs T.

TABLE VI
LOW TEMPERATURE MEASUREMENTS

Temperature (°C)	V_sensor (V)
250.6 ± 0.1	4.8 ± 0.05
252.1 ± 0.1	4.9 ± 0.05
255.3 ± 0.1	5.0 ± 0.05
258.6 ± 0.1	5.1 ± 0.05
261.1 ± 0.1	5.2 ± 0.05
263.0 ± 0.1	5.3 ± 0.05
269.4 ± 0.1	5.5 ± 0.05
272.9 ± 0.1	5.7 ± 0.05
278.3 ± 0.1	5.9 ± 0.05
282.6 ± 0.1	6.1 ± 0.05
286.9 ± 0.1	6.3 ± 0.05
291.3 ± 0.1	6.5 ± 0.05
295.6 ± 0.1	6.7 ± 0.05
299.8 ± 0.1	6.9 ± 0.05
304.1 ± 0.1	7.1 ± 0.05
308.2 ± 0.1	7.3 ± 0.05
312.0 ± 0.1	7.5 ± 0.05
316.1 ± 0.1	7.7 ± 0.05
320.0 ± 0.1	7.9 ± 0.05
324.0 ± 0.1	8.1 ± 0.05

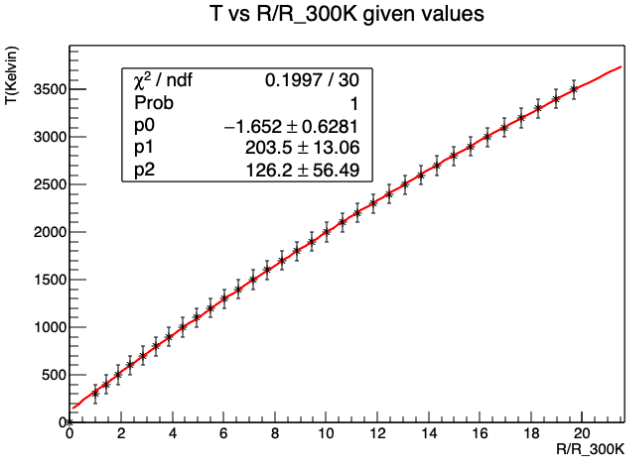

Fig. 7. Second-order polynomial-fit for R/R_{300K} vs T taken from table 2

In order to get a better χ^2 value, that is less error, we have used a second order polynomial for our fit and we had the following estimation: $T(R/R_{300K}) = -1.652(R/R_{300K})^2 + 203.5(R/R_{300K}) + 126.2$

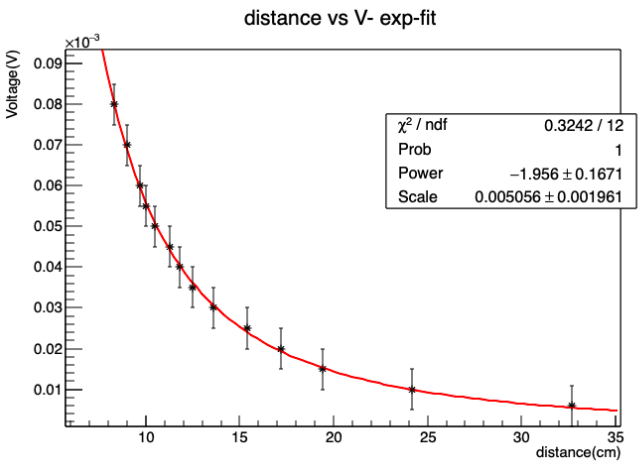

Fig. 8. exponential fit for Distance vs V_{sensor}

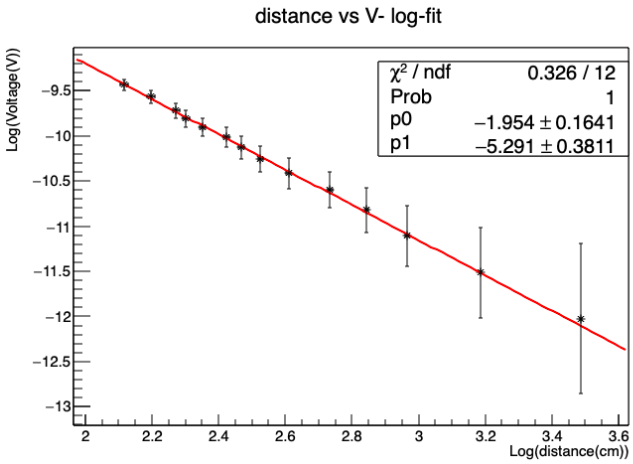

Fig. 9. line fit fit for log(Distance) vs log(V_{sensor})

The exponential and the log-scale linear fits for the part 1 are have reasonable χ^2 values and we got $R \propto (\frac{1}{distance})^{1.956 \pm 0.1671}$ for the exponential-fit and $R \propto (\frac{1}{distance})^{1.954 \pm 0.1641}$ for the log-scale fit.

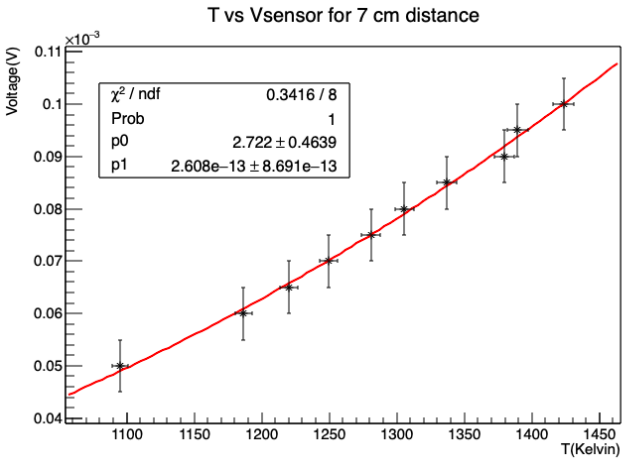

Fig. 10. exponential fit at high temperature for 7 cm distance

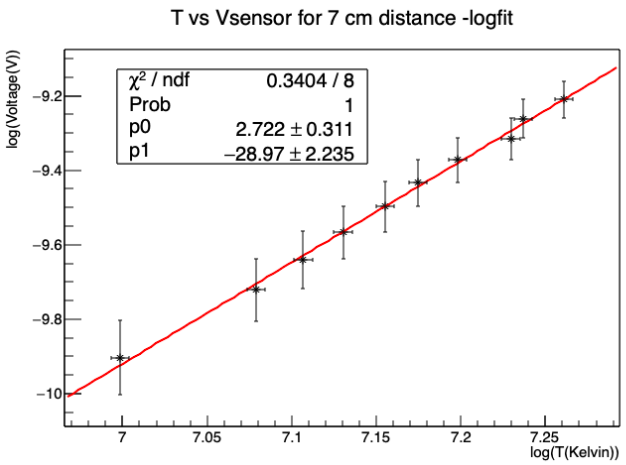

Fig. 11. log-scale line fit at high temperature for 7 cm distance

Both the exponential and the log-scale line-fits gave us reasonable χ^2 values and we got $R \propto T^{2.722 \pm 0.4639}$ for the exponential fit and $R \propto T^{2.722 \pm 0.311}$ for the log-scale fit, at high temperatures and 7 cm distance.

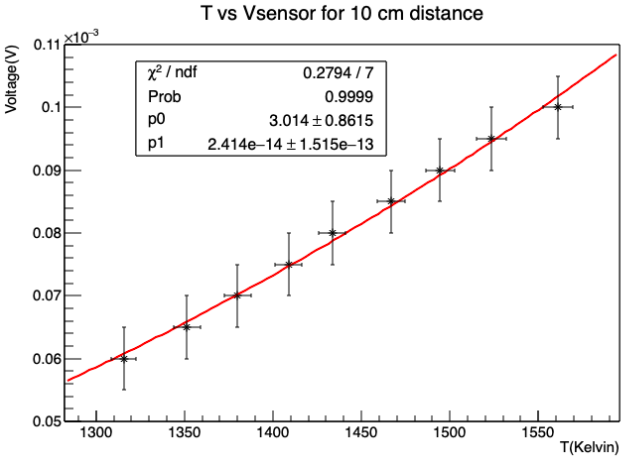

Fig. 12. exponential fit at high temperature for 10 cm distance

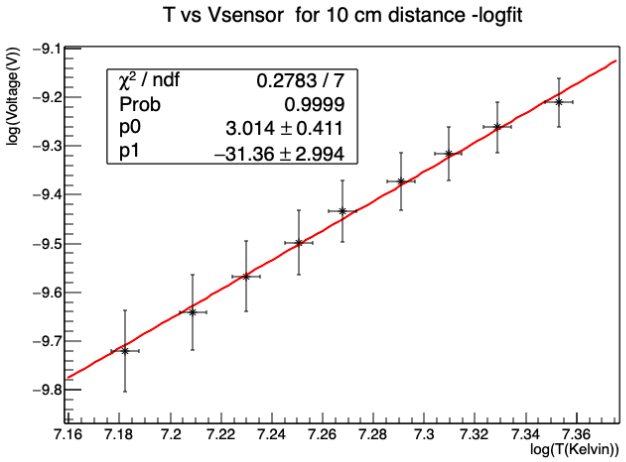

Fig. 13. log-scale line fit at high temperature for 10 cm distance

Again, our fits gave us reasonable χ^2 values and we got $R \propto T^{3.014\pm0.8615}$ for the exponential fit and $R \propto T^{3.014\pm0.411}$ for the log-scale fit, at high temperatures and 10 cm distance.

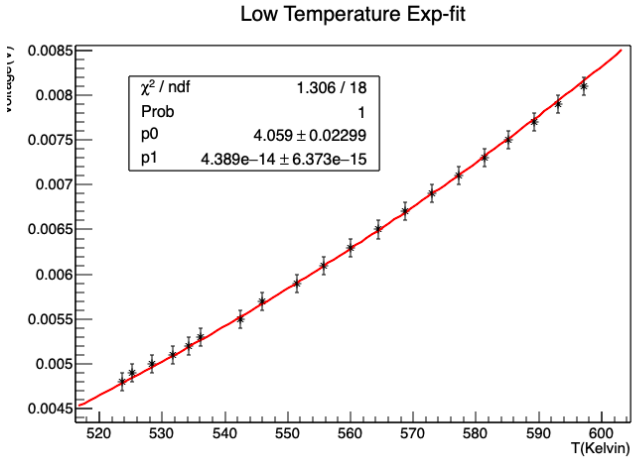

Fig. 14. exponential fit at low temperature

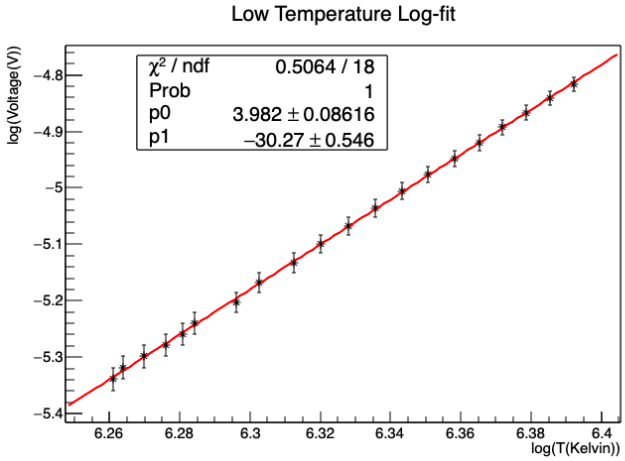

Fig. 15. log-scale line fit at low temperature

Although we got again a relatively χ^2 for exponential fit, both errors are reasonable and we get $R \propto T^{4.059\pm0.0.02299}$ for the exponential fit and $R \propto T^{3.982\pm0.08616}$ for the log-scale fit, at low temperatures.

VI. The Result

We have calculated the resistance of the lamp at room temperature as $R_{300K} = 0.3458 \pm 0.002174$ in the first place and then have used it in the estimation of the T as a function of $(R/R_{300K}$ as a second order polynomial and the result is $T(R/R_{300K}) = -1.652(R/R_{300K})^2 + 203.5(R/R_{300K}) + 126.2$.

Secondly, we have tested the inverse square law and found the proportionality as 1.956 ± 0.1671,0.009 σs away from 2, from exponential fit, and got 1.956 ± 0.1641, again 0.009 σs away, from log-scale line fit.

The expected proportionality of the radiation to the Temperature is 4.In the high temperature case, we have found the proportionality of the radiation the temperature at 7 cm distance as 2.722 ± 0.4639, 2.75 σs away, 2.722 ± 0.311, 4.11 σs away, from exponential and log-scale line fits respectively. Moreover, for the 10 cm distance at high temperatures we have found the proportionality n as 3.014 ± 0.8615,1.14 σs away ,3.014 ± 0.411,σs away 2.40, from exponential and log-scale line fits respectively.

Finally, at low temperatures, we have found n to be $4.059 \pm 0.0.02299$, 2.57 σs away, 3.982 ± 0.08616, 0.21 σs away.

Our final aim is to calculate the weighted average of the those 6 n values we got from fits. The weighted average $\bar{x}$ of a set of values x_i with uncertainties σ_i is calculated as:

$$\bar{x} = \frac{\sum\left(\frac{x_i}{\sigma_i^2}\right)}{\sum\left(\frac{1}{\sigma_i^2}\right)}$$

The uncertainty of the weighted average, $\sigma_{\bar{x}}$, is given by:

$$\sigma_{\bar{x}} = \sqrt{\frac{1}{\sum\left(\frac{1}{\sigma_i^2}\right)}}$$

Then using those formulas we got $n_{weighted} = 4.040 \pm 0.022$ for Stefan Boltzmann Radiation Law and n = 1.95 ± 0.117 for inverse square law.

VII. The Conclusion

We have calculated, for reference, the resistance of the lamp and used it to get the temperature as a function of radiation successfully. The experimental test of the inverse square law seems to be very successful as well, by its closeness to 2. In the high temperature case, the proportionality with the temperature diverged slightly from the expected value of 3. One of the reasons for this is that our exponential fit has an y-intercept bigger than 0, which we do not expect because we have used an exponential fit in the form of $scale * x^{(}power)$. Therefore, we deduced that the voltage amplifier was calibrated properly so that it is biased to give higher voltages than its real values. In the low temperature exponential fit, we almost exactly the desired value in the log scale-fit but in the exponential fit there is a relatively high error because we took the uncertainty of the temperature as the lowest significant figure rather than the actual one. It should also be noted that, we did not measure and calculate the background radiation as we have been measuring at higher than 7 cm's. In general, we have condacted a successful experiment if the weighted averages of the n's are to be considered.

References

[1] *article*. URL: https://babel.hathitrust.org/cgi/pt?id=uc1.a0002763670&view=1up&seq=327 (visited on 11/09/2024).

[2] *wiki*. URL: https://en.wikipedia.org/wiki/Stefan%E2%80%93Boltzmann_law (visited on 11/09/2024).

[3] E. Gülmez. *Advanced Physics Experiments*. 1st. Boğaziçi University Publications, 1999.

VIII. Appendix

I have used the ROOT'S built'in function to get linear(figures 6,9,11,13 and 15),polnymomial(figure 7) and exponential fits(figures 8,10,12 and 14). Here is the full code that includes the datasets below:

```
{
// PART 0
const int ndata0 = 9;
float erI0[ndata0] =
    {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001};
float voltages0[ndata0] = {
  0.0347,0.0438,0.050,0.0566,0.0632,0.0689,0.0778,0.0885,0.0994
   };

float currents0[ndata0] = {
  0.109,0.137,0.156,0.176,0.195,0.211,0.237,0.258,0.285
   };

float erV0[ndata0];

for (int p = 0; p < ndata0; ++p) {
   if (voltages0[p] <= 0.4)
{
erV0[p] = (3 * voltages0[p] / 100) + 0.0004;
}
else if (voltages0[p] <= 4)
{
erV0[p] = (5 * voltages0[p] / 100) + 0.003;
}
else if (voltages0[p] <= 40)
{
erV0[p] = (5 * voltages0[p] / 100) + 0.03;
}
else if (voltages0[p] <= 400)
{
   erV0[p] = (5 * voltages0[p] / 100) + 0.3;
}
else
{
erV0[p] = (voltages0[p] / 100) + 4;
}

}

TGraphErrors *part0 = new TGraphErrors(ndata0
    ,currents0,voltages0,erI0,erV0);
part0->Draw("A*");
part0->GetXaxis()->SetTitle("Current(A)");
part0->GetYaxis()->SetTitle("Voltage(V)");
part0->SetTitle("I vs V for resistence");


TF1 *p0Linefit = new
    TF1("p0Linefit","[0]*x+[1]");
p0Linefit->SetParameters(3.1416,2.7182);
part0->Fit(p0Linefit);
TCanvas *c = new TCanvas();

gStyle->SetOptFit(1111);

double Rroom = p0Linefit->GetParameter(0);
double Rroom_error = p0Linefit->GetParError(0);

cout<<"resistance at room temperature 300K is
    = "<<Rroom<<"+-"<<Rroom_error<< "\n";
```

```
// T VS R/R_300K

const int ntable = 34;
float Ttable[ntable] =
    {300,400,500,600,700,800,900,1000,1100,1200,1300,140
2500,2600,2700,2800,2900,3000,3100,3200,3300,3400,3500};
float TtableEr[ntable] =
    {100,100,100,100,100,100,100,100,100,100,100,100,100,100
100,100,100,100,100,100,100,100,100,100,100,100};
float Rtable[ntable] =
    {1.0,1.43,1.87,2.34,2.85,3.36,3.88,4.41,4.95,5.48,6.
13.08,13.72,14.34,14.99,15.63,16.29,16.95,17.62,18.28,18
float RtableEr[ntable] =
    {0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0
0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01};

TGraphErrors *table = new
    TGraphErrors(ntable,Rtable,Ttable,RtableEr,TtableEr)
table->Draw("A*");
table->GetXaxis()->SetTitle("R/R_300K");
table->GetYaxis()->SetTitle("T(Kelvin)");
table->SetTitle("T vs R/R_300K given values");


TF1 *tablepoly = new TF1("tableLine","[0]*x*x
    + [1]*x + [2]");
tablepoly->SetParameters(1,1,1);
table->Fit(tablepoly);
TCanvas *c2 = new TCanvas();

double tSecond = tablepoly->GetParameter(0);
double tFirst = tablepoly->GetParameter(1);
double tZero = tablepoly->GetParameter(2);

cout<<"T(R/R_300K) =
    "<<tSecond<<"(R/R_300K)^2"<<tFirst<<"R/R_300K"<<tZer
    "\n";



// PART 1

const int ndata1 = 14;
float amp = 1e4;
float vLamb1[ndata1] =
    {0.06,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0
float vLamb1err[ndata1] =
    {0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0

float vLamb1log[ndata1];
float vLamb1logerr[ndata1];



float d1[ndata1] =
    {32.7,24.2,19.4,17.2,15.4,13.6,12.5,11.8,11.3,10.5,1
float d1err[ndata1] =
    {0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1

float d1log[ndata1];
float d1logerr[ndata1];

for (int i = 0; i < 14; ++i) {
```

```
   vLamb1[i] = vLamb1[i] / amp;
   vLamb1err[i] = vLamb1err[i] / amp;
   vLamb1log[i] = log(vLamb1[i]);
   vLamb1logerr[i] = vLamb1err[i] / vLamb1[i];
   d1log[i] = log(d1[i]);
   d1logerr[i] = d1err[i] / d1[i];

}


TGraphErrors *p1_exp = new
    TGraphErrors(ndata1,d1,vLamb1,d1err,vLamb1err);
p1_exp->Draw("A*");
p1_exp->GetXaxis()->SetTitle("distance(cm)");
p1_exp->GetYaxis()->SetTitle("Voltage(V)");
p1_exp->SetTitle("distance vs V- exp-fit ");


TF1 *exp = new TF1("exp","[1]*pow(x,[0])");
exp->SetParNames("Power", "Scale");
p1_exp->Fit(exp);

TCanvas *c3 = new TCanvas();



float n = exp->GetParameter(0);
float n_error = exp->GetParError(0);
cout << "Power = "<< n << "+- "<<n_error<<"\n";

TGraphErrors *p1log = new
    TGraphErrors(ndata1,d1log,vLamb1log,d1logerr,vLamb1logerr);
p1log->Draw("A*");
p1log->GetXaxis()->SetTitle("Log(distance(cm))");
p1log->GetYaxis()->SetTitle("Log(Voltage(V))");
p1log->SetTitle("distance vs V- log-fit ");


TF1 *p1Linefit = new
    TF1("p1Linefit","[0]*x+[1]");
p1log->Fit(p1Linefit);
TCanvas *c4 = new TCanvas();

// PART 2

// data for cm


const int ndata2 = 10;
float p2Vsensor7cm[ndata2] =
    {1.0,0.95,0.90,0.85,0.80,0.75,0.70,0.65,0.60,0.50};
float p2Vlamb7cm[ndata2] =
    {3.932,3.721,3.661,3.417,3.246,3.114,2.953,2.808,2.64[illegible],2.345};
float p2Ilamb7cm[ndata2]=
    {1.689,1.644,1.631,1.579,1.542,1.513,1.477,1.444,1.407,1.317};

//data for 10 cm
float p2Vsensor10cm[9]=
    {1.0,0.95,0.90,0.85,0.80,0.75,0.70,0.65,0.60};
float p2Vlamb10cm[9]=
    {4.86,4.58,4.39,4.20,3.99,3.83,3.66,3.493,3.298};
float p2Ilamb10cm[9]=
    {1.875,1.818,1.782,1.742,1.700,1.665,1.630,1.594,1.553};

// floats to be filled

float
    p2Vsensor7cmerr[ndata2],p2Vlamb7cmerr[ndata2],p2Ilamb7cmerr[ndata2],
    p2Rlamb7cm[ndata2],p2Rlamb7cmerr[ndata2],T7cm[ndata2],T7
    ;

float
    p2Vsensor10cmerr[9],p2Vlamb10cmerr[9],p2Ilamb10cmerr
    p2Rlamb10cm[9],p2Rlamb10cmerr[9],T10cm[9],T10cmerr[9]
    ;

// log floats

float T7cmlog[ndata2], T7cmlogerr[ndata2],
    T10cmlog[9],T10cmlogerr[9],
    p2Vsensor7cmlog[ndata2],p2Vsensor7cmlogerr[ndata2],
    p2Vsensor10cmlog[9],p2Vsensor10cmlogerr[9];

// for 7 cm fgetting errors and R values

for (int j = 0; j < 10; ++j) {

    // sensor in terms of voltage
    p2Vsensor7cm[j] = p2Vsensor7cm[j] / amp;
    // errors of measurements
    p2Vsensor7cmerr[j] = 0.05 / amp;
    p2Vlamb7cmerr[j] = 0.001;
    p2Ilamb7cmerr[j] = 0.001;
    //resistance calculations
    p2Rlamb7cm[j] = p2Vlamb7cm[j] /
        p2Ilamb7cm[j];
    p2Rlamb7cmerr[j] = p2Rlamb7cm[j] *
        sqrt(pow(p2Vlamb7cmerr[j]/p2Vlamb7cm[j],2)+
    pow(p2Ilamb7cmerr[j]/p2Ilamb7cm[j],2));


    // getting T and its error

    T7cm[j] = tSecond *
        (p2Rlamb7cm[j]/Rroom)*(p2Rlamb7cm[j]/Rroom)
        +
    tFirst * (p2Rlamb7cm[j]/Rroom) + tZero;

    double dTdR = tSecond * (2 * p2Rlamb7cm[j])
        / (Rroom * Rroom) + tFirst / Rroom;
    double dTdRroom = -tSecond * (2 *
        p2Rlamb7cm[j] * p2Rlamb7cm[j]) / (Rroom
        * Rroom * Rroom) - tFirst *
        p2Rlamb7cm[j] / (Rroom * Rroom);

    T7cmerr[j] = sqrt((dTdR * p2Rlamb7cmerr[j])
    * (dTdR * p2Rlamb7cmerr[j]) +
     (dTdRroom * Rroom_error) * (dTdRroom *
     Rroom_error));
    cout<<"T7cm T values are:
        "<<T7cm[j]<<"+-"<<T7cmerr[j]<<"\n";

    //filling logs
    p2Vsensor7cmlog[j] = log(p2Vsensor7cm[j]);
    p2Vsensor7cmlogerr[j] = p2Vsensor7cmerr[j]
        / p2Vsensor7cm[j];
    T7cmlog[j] = log(T7cm[j]);
    T7cmlogerr[j] = T7cmerr[j]/T7cm[j];
}
//p2Ilamb7cmerr[ndata2],errors and R values
```

```
for (int k = 0; k < 9; ++k) {

   // sensor in terms of voltage
   p2Vsensor10cm[k] = p2Vsensor10cm[k] / amp;
   // errors of measurements
   p2Vsensor10cmerr[k] = 0.05 / amp;
   p2Vlamb10cmerr[k] = 0.001;
   p2Ilamb10cmerr[k] = 0.001;
   //resistance calculations
   p2Rlamb10cm[k] = p2Vlamb10cm[k] /
       p2Ilamb10cm[k];
   p2Rlamb10cmerr[k] = p2Rlamb10cm[k] *
       sqrt(pow(p2Vlamb10cmerr[k]/p2Vlamb10cm[k],2)+
   pow(p2Ilamb10cmerr[k]/p2Ilamb10cm[k],2));



   // getting T and its error

   T10cm[k] = tSecond *
       (p2Rlamb10cm[k]/Rroom)*(p2Rlamb10cm[k]/Rroom)
       +
   tFirst * (p2Rlamb10cm[k]/Rroom) + tZero;


   double dTdR_10cm = tSecond * (2 *
       p2Rlamb10cm[k]) / (Rroom * Rroom) +
       tFirst / Rroom;
   double dTdRroom_10cm = -tSecond * (2 *
       p2Rlamb10cm[k] * p2Rlamb10cm[k]) /
       (Rroom * Rroom * Rroom) - tFirst *
       p2Rlamb10cm[k] / (Rroom * Rroom);

   T10cmerr[k] = sqrt((dTdR_10cm *
       p2Rlamb10cmerr[k]) * (dTdR_10cm *
       p2Rlamb10cmerr[k]) +
    (dTdRroom_10cm * Rroom_error) *
        (dTdRroom_10cm * Rroom_error));


   // filling logs

   p2Vsensor10cmlog[k] = log(p2Vsensor10cm[k]);
   p2Vsensor10cmlogerr[k] =
       p2Vsensor10cmerr[k] / p2Vsensor10cm[k];

   T10cmlog[k] = log(T10cm[k]);
   T10cmlogerr[k] = T10cmerr[k]/T10cm[k];

   cout<<"for 7 cm T values are:
       "<<T10cm[k]<<"+-"<<T10cmerr[k]<<"\n";

}


// graphs for measurements of 7 cm distance

 // exponantial fit

   TGraphErrors *p2exp_7cm = new
       TGraphErrors(ndata2,T7cm,p2Vsensor7cm,T7cmerr,p2Vsensor7cmerr);
   p2exp_7cm->Draw("A*");
   p2exp_7cm->GetXaxis()->SetTitle("T(Kelvin)");
   p2exp_7cm->GetYaxis()->SetTitle("Voltage(V)");
   p2exp_7cm->SetTitle("T vs Vsensor for 7 cm
       distance");


   TF1 *exp_7cm = new
       TF1("exp7","[1]*TMath::Power(x,[0])");
   exp_7cm->SetParameter(1, 1.70743068e-17);
   exp_7cm->SetParameter(0, 4);
   p2exp_7cm->Fit(exp_7cm);
   exp_7cm->SetParNames("Power", "Scale");


   TCanvas *c5 = new TCanvas();

// log fit

   TGraphErrors *p2log_7cm = new
       TGraphErrors(ndata2,T7cmlog,p2Vsensor7cmlog,T7cml
   p2log_7cm->Draw("A*");
   p2log_7cm->GetXaxis()->SetTitle("log(T(Kelvin))");
   p2log_7cm->GetYaxis()->SetTitle("log(Voltage(V))");
   p2log_7cm->SetTitle("T vs Vsensor for 7 cm
       distance -logfit");

   TF1 *log_7cm = new TF1("log7","[0]*x+[1]");
   p2log_7cm->Fit(log_7cm);
   log_7cm->SetParNames("slope", "intercept");


   TCanvas *c6= new TCanvas();

   // graphs for measurements of 10 cm distance

 // exponantial fit

    const int n10cm = 9;

   TGraphErrors *p2exp_10cm = new
       TGraphErrors(n10cm,T10cm,p2Vsensor10cm,T10cmerr,p
   p2exp_10cm->Draw("A*");
   p2exp_10cm->GetXaxis()->SetTitle("T(Kelvin)");
   p2exp_10cm->GetYaxis()->SetTitle("Voltage(V)");
   p2exp_10cm->SetTitle("T vs Vsensor for 10
       cm distance");

   TF1 *exp_10cm = new
       TF1("exp10","[1]*pow(x,[0])");
   exp_10cm->SetParameter(1, 1.70743068e-17);
   exp_10cm->SetParameter(0, 4);
   p2exp_10cm->Fit(exp_10cm);
   exp_10cm->SetParNames("Power", "Scale");

   TCanvas *c7 = new TCanvas();


   // log fit

   TGraphErrors *p2log_10cm = new
       TGraphErrors(n10cm,T10cmlog,p2Vsensor10cmlog,T10c
   p2log_10cm->Draw("A*");
   p2log_10cm->GetXaxis()->SetTitle("log(T(Kelvin))");
   p2log_10cm->GetYaxis()->SetTitle("log(Voltage(V))");
   p2log_10cm->SetTitle("T vs Vsensor for 10
       cm distance -logfit");


   TF1 *log_10cm = new
       TF1("log10","[0]*x+[1]");
   p2log_10cm->Fit(log_10cm);
   log_10cm->SetParNames("slope", "intercept");
```

```
   TCanvas *c8= new TCanvas();



// PART 3
                                                          }
const int ndata3 =20;
float p3T[ndata3] =
    {250.6,252.1,255.3,258.6,261.1,263.0,269.4,272.9,278.3,282.6,286.9,291.3,295.6,299.8,
304.1,308.2,312.0,316.1,320.0,324.0};
float p3Vsensor[ndata3] =
    {4.8,4.9,5.0,5.1,5.2,5.3,5.5,5.7,5.9,6.1,6.3,
6.5,6.7,6.9,7.1,7.3,7.5,7.7,7.9,8.1};

float
    p3Terr[ndata3],p3Vsensorerr[ndata3],p3Tlog[ndata3],p3Vlog[ndata3],p3Tlogerr[ndata3],p3Vlogerr[ndata3];

for (int m = 0; m < ndata3; ++m) {

p3T[m] = p3T[m] + 273.1;
p3Terr[m] = 0.1;
p3Vsensor[m] = p3Vsensor[m] / 1000;
p3Vsensorerr[m] = 0.1 /1000;
p3Tlog[m] = log(p3T[m]);
p3Vlog[m] = log(p3Vsensor[m]);


p3Tlogerr[m] = p3Terr[m]/p3T[m];
p3Vlogerr[m] = p3Vsensorerr[m]/p3Vsensor[m];

}

// part 3 exponantial
   TGraphErrors *p3exp = new
       TGraphErrors(ndata3,p3T,p3Vsensor,p3Terr,p3Vsensorerr);
   p3exp->Draw("A*");
   p3exp->GetXaxis()->SetTitle("T(Kelvin)");
   p3exp->GetYaxis()->SetTitle("Voltage(V)");
   p3exp->SetTitle("Low Temperature Exp-fit");

   TF1 *lastexp = new
       TF1("lastexp","[1]*pow(x,[0])");
   lastexp->SetParameter(1, 1.70743068e-17);
   lastexp->SetParameter(0, 4);
   p3exp->Fit(lastexp);
   lastexp->SetParNames("Power", "Scale");

   TCanvas *c9 = new TCanvas();

// part 3 log

TGraphErrors *p3log = new
    TGraphErrors(ndata3,p3Tlog,p3Vlog,p3Tlogerr,p3Vlogerr);
   p3log->Draw("A*");
   p3log->GetXaxis()->SetTitle("log(T(Kelvin))");
   p3log->GetYaxis()->SetTitle("log(Voltage(V))");
   p3log->SetTitle("Low Temperature Log-fit");

   TF1 *lastlog = new
       TF1("lastlog","[0]*x+[1]");
   p3log->Fit(lastlog);
   lastlog->SetParNames("slope", "intercept");

   TCanvas *c10 = new TCanvas();
```