Experiment 3: kB

Madeleine Allen Edward Piper

2/6/2019

Recording the resistor values measured during lab.

```
#everything will be in ohms
short<- .03
shortError<-.001

k20<-20090
k20error<-1

k35 <- 35230 #secretly 35.2 but that would be an ugly variable name
k35error<-1

k100 <- 100700
k100error <- 1

k10 <- 999.05
k10error<- .01

k1 <- 998.17
k1error <- .01

k48 <- 48650 #secretly 48.7k but again that would be an ugly variable name
k48error<- 1
resistors<-c(k1,k10,k20,k35, k48,k100)
resistorserror<-c(k1error, k10error, k20error, k35error, k48error, k100error)
```

Experiment 3: Johnson Noise - Boltzmann Constant

```
experiment3data1<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/experiment3data1.csv")
```

Calculate Vmeas, V, and Vsystem

```
Vsys<- experiment3data1[1,7] #first row 7th column
VsysError <- experiment3data1[1,9]

Vmeask1<- (experiment3data1[2,7])
Vmeask10<-experiment3data1[3,7]
Vmeask20 <-experiment3data1[4,7]
Vmeask32<-experiment3data1[5,7]
Vmeask48<-experiment3data1[6,7]
Vmeask100<-experiment3data1[7,7]

Vmeas<-c(Vmeask1, Vmeask10, Vmeask20, Vmeask32, Vmeask48, Vmeask100)


VmeasError<-sqrt((sum(experiment3data1[2:7,9])^2))
```

```
V<- sqrt(-Vsys^2+Vmeas^2)
```

```
## Warning in sqrt(-Vsys^2 + Vmeas^2): NaNs produced
```

```
Verror<- sqrt(VmeasError^2+ VsysError^2)
```

Calculating G

```
gain <- data.frame(Frequency = vout1$x, Gain = (m_vout[2]/m_in))

capacitance <-87.875*(10^-12)
capacitanceError <-.594*(10^-12)
#df is just the x componenent

riemanSum <- function(fa,fb){
  area <-0.5*(125)*(fb-fa)+fa*125
  return(area)
}

#resistors<-read.csv("experiment3data1.csv")

C = capacitance
integrand <- data.frame(
  gain[2]/(1+(2*pi*C*vin1$x*short)^2),
  gain[2]/(1+(2*pi*C*vin1$x*k1)^2),
  gain[2]/(1+(2*pi*C*vin1$x*k10)^2),
  gain[2]/(1+(2*pi*C*vin1$x*k20)^2),
  gain[2]/(1+(2*pi*C*vin1$x*k35)^2),
  gain[2]/(1+(2*pi*C*vin1$x*k48)^2),
  gain[2]/(1+(2*pi*C*vin1$x*k100)^2)
  )
area <- data.frame(
  G1 =0,
  G2 =0,
  G3 =0,
  G4 =0,
  G5 =0,
  G6 =0,
  G7 =0

  )
for(i in 1:length(integrand))
{
    for(l in 1:398)
    {
      if(is.na(integrand[l+1,i]))
      {
        break
      }
      else
      {
        area[i] <- area[i]+ riemanSum(integrand[l,i],integrand[l+1,i])
```

```
      }
    }
}
```

So this returns a gain value G for each resistor (called "area")

```
#prepare the data for graphing
kb<- 1.38064852 *10^-23 #m2 kg s-2 K-1

area2<-area[2:7] #take away the short's data
resistors2 <-resistors[1:7]
y_value2<- (V^2)/(4*resistors2*area2)
y2<- unlist(y_value2, use.names=FALSE)

#calculate the temperatures
Temperature<- ((V^2)/(4*kb*area2*resistors2))/100
Temperature2<-unlist(Temperature, use.names = FALSE)
```

```
## [1] 71.82 73.19 74.53 74.65 75.43 78.52
```

```
Temperature<- ((V^2)/(4*kb*area2*resistors2))/100
Temperature2<-unlist(Temperature, use.names = FALSE)
Temperature2Error<-Temperature2*sqrt((V^2/Verror^2)+(resistors2/resistorserror)^2)
```

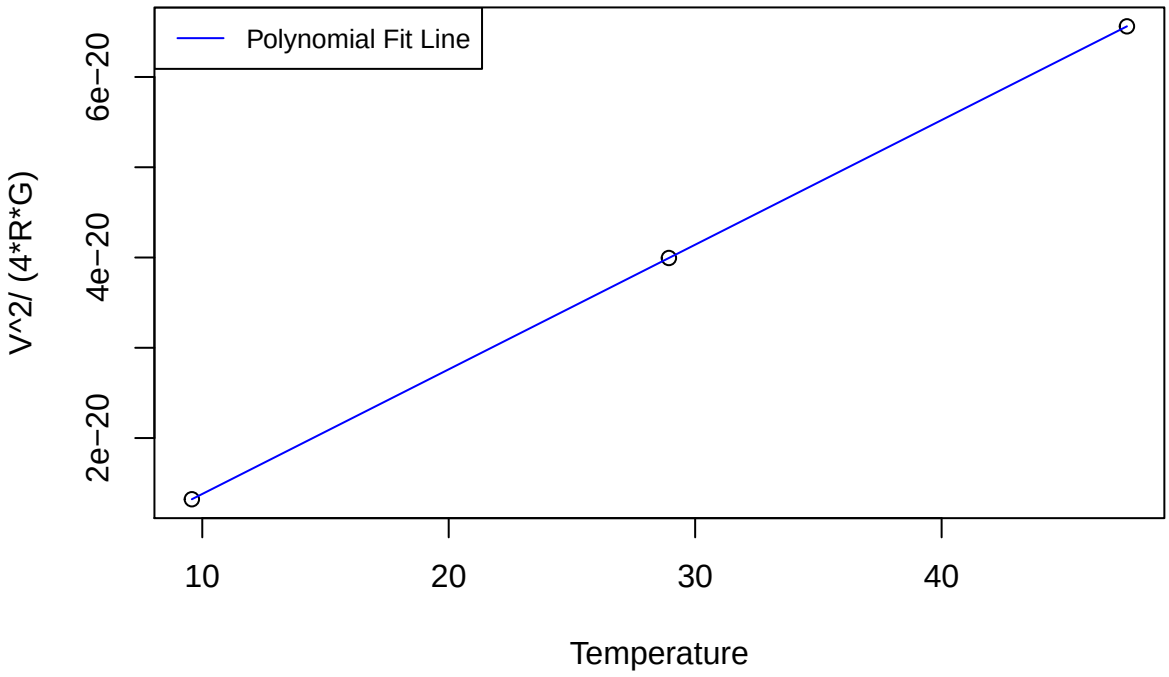
```
## Warning in resistors2/resistorserror: longer object length is not a
## multiple of shorter object length
```

```
## Warning in (V^2/Verror^2) + (resistors2/resistorserror)^2: longer object
## length is not a multiple of shorter object length
```

```
## Warning in Temperature2 * sqrt((V^2/Verror^2) + (resistors2/
## resistorserror)^2): longer object length is not a multiple of shorter
## object length
```

```
#2nd degree fit
fit <- lm(y2~Temperature2)

#plot same as above this time with a 2d fit line
plot(Temperature2,y2, main= "Gain and Voltage as a function of Temperature with a 2nd order fit", ylab=
lines(Temperature2, predict(fit, data.frame(Temperature2)), col="blue")
arrows(Temperature2, mean(Temperature2)-Temperature2Error, Temperature2, mean(Temperature2) + Temperatu
legend("topleft", legend=c("Polynomial Fit Line"),
       col=c("blue"), lty=1:2, cex=0.8)
```

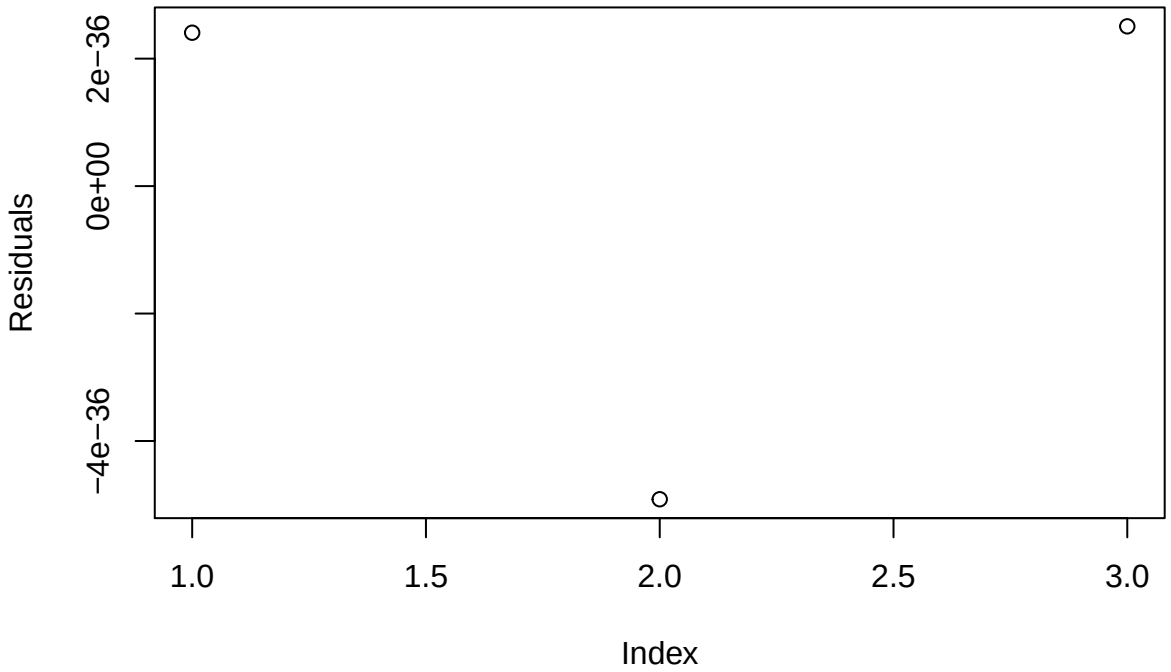



```
summary(fit)
```

```
## Warning in summary.lm(fit): essentially perfect fit: summary may be
## unreliable
```

```
##
## Call:
## lm(formula = y2 ~ Temperature2)
##
## Residuals:
##          4          5          6
##  2.407e-36 -4.914e-36  2.507e-36
##
## Coefficients:
##               Estimate Std. Error   t value Pr(>|t|)
## (Intercept)  0.000e+00  7.311e-36 0.000e+00        1
## Temperature2 1.381e-21  2.243e-37 6.155e+15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.019e-36 on 1 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 3.789e+31 on 1 and 1 DF,  p-value: < 2.2e-16
```

```
plot(fit$residuals, main = "Residuals of the fit line", ylab= "Residuals")
```



You have to divide by 100 for some reason in all of the results. Our result for the slope was 1.381*10^-21 but dividing by 100 gives us: 1.381*10^-23 with an error of