Heat Transport Mechanisms for an End-Heated Aluminum Rod

James Wasteneys, 34140137, Group 13

July 6, 2015

Contents

1 Introduction

In this lab multiple experiments were conducted to determine the various constants needed in order to accurately model the heat transfer mechanisms in an aluminum rod. The main heat transfer mechanisms that were used to model these experiments are conduction, convection, radiation and evaporation. Thermal conduction is a transfer of energy through solid bodies, and can be described by the equation $q = \frac{\kappa A \Delta T}{d}$ where q is heat transferred per unit time, κ is the thermal conductivity constant, ΔT is the temperature difference, A is the cross-sectional area and d is the distance over which the heat is conducting. Convection is the transfer of heat caused by the flow of fluids due to changes in density through heating. Convection is a much more complicated transfer mechanism to model as it has many different factors that it depends on, such as the geometry of the system or any air currents. The equation used in this lab to model convection is $q = \kappa_c A \Delta T$ where κ_c is the convective heat transfer coefficient. Black body radiation is process where thermal radiation (usually in the infrared spectrum for room temperature objects) is given off by material based on it's temperature. Most objects are not true black-bodies but they can be modeled as such by applying an emissivity constant. In this lab black body radiation will be modeled according to the Stefan-Boltzmann Law : $q = \epsilon \sigma A(T^4 - T_{amb}^4)$ where ϵ is the emissivity σ is the Stefan-Bolztmann Constant, A is the area, T is the temperature, and T_{amb} is the temperature of the surroundings. Evaporative heat loss was also investigated in one of the experiments, although it was only estimated due to the complexity of the calculation.

2 Experimental setup and procedures

2.1 Calibration

Temperature measurements in this lab were made with TMP35 low voltage temperature sensors. It was found that these sensors did not behave consistently and that two sensors at the same temperature would produce different voltages. In order to calibrate them, the sensors were placed into a plastic bag and then into a water bath at slightly below room temperature and were continuously monitored while the water was slowly brought to a boil. The data from the different sensors was then averaged and scaled according to the equation given by the sensor data sheet to provide the best estimate of the temperature of the water with respect to time. Each individual TMP35 was then calibrated to this average by doing a least squares fit of the averaged values vs the values of an individual sensor. This gave a scaling factor as well as an offset that can be applied to the data collected in each experiment in order to give consistency between measurements.

2.2 Horizontal Bare Rod

In this experiment, the rod was held in place horizontally using a laboratory stand and a power resistor was fastened to one end using a screw and a dab of thermal paste. TMP35 temperature sensors were placed at five positions along the rod, one at each end and three in the middle, held in place using electrical tape. An additional sensor was placed in a breadboard sitting on the lab bench and was used to measure the ambient temperature. The power resistor was then supplied with a certain amount of power from a laboratory power supply and the sensors were monitored using an Arduino micro-controller until the temperature of the rod reached a steady state for a significant period of time. The TMP35 sensors were powered between 5V and ground using a laboratory power supply. The Arduino micro-controller ground and 5V pins were shorted to the power supply ground and 5V respectively in order to keep consistency between the output voltage of the TMP35 sensors and the measurements through the analog IO pins of the Arduino. The data was read off the Arduino in real time and imported into MATLAB.

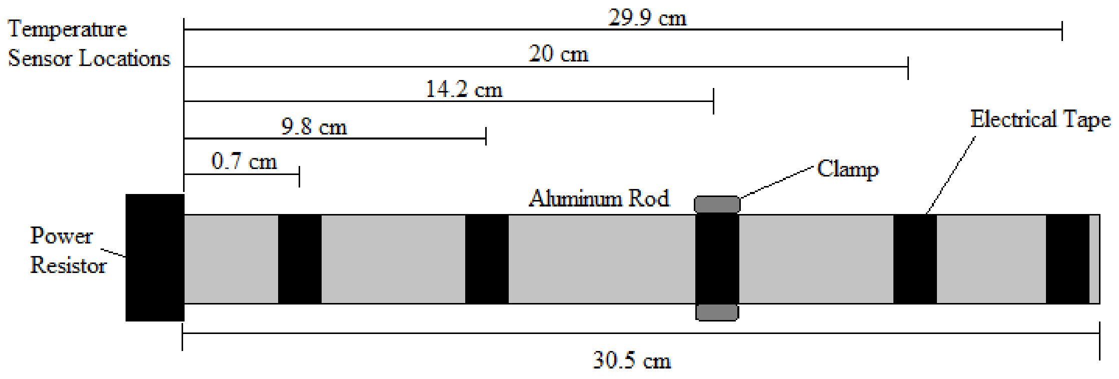


Figure 1: Horizontal Bare Rod

2.3 Vertical Bare Rod

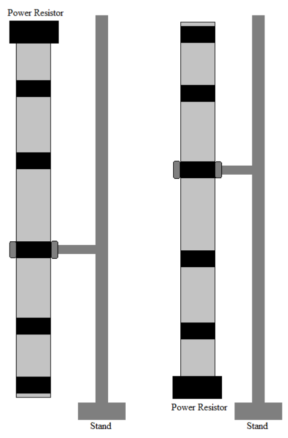


Figure 2: Vertical Bare Rod

This experiment was identical to the horizontal bare rod except in this case the rod was position vertically, once with the power resistor at the top, and once with it at the bottom.

2.4 Insulated Rod

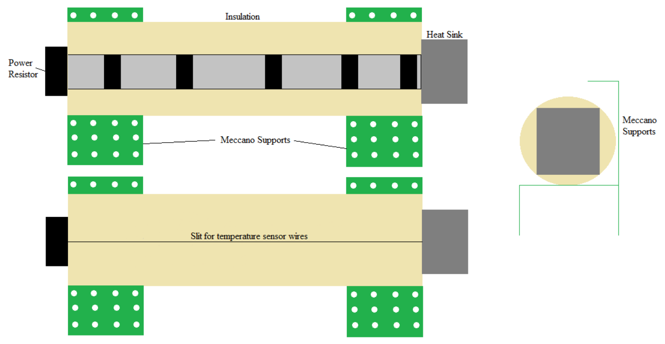


Figure 3: Insulated Rod

In this experiment the rod was completely encased in thick insulation that should have blocked out most heat transfer throughout the length of the rod. On the end opposite the power resistor a heat sink was placed to increase the rate of change of temperature along the length of the rod.

2.5 Rod Painted Black

In this experiment the aluminum rod was painted with black spray paint, which should increase the emissivity of it's surface. Aside from the paint this experiment was identical to the horizontal bare rod experiment.

2.6 Moist Rod

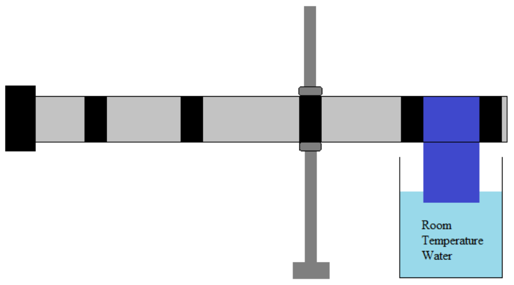


Figure 4: Moist Rod

This experiment was again identical to the horizontal bare rod experiment except for a 9.9cm strip of moist fabric which was placed over a section of the rod between the two temperature sensor position furthest from the power resistor.

3 Analysis

The data collected from the various experiments was analyzed using finite difference simulations in MATLAB (Appendix A for Steady State, B for Transient).

3.1 Transient

Expt.	k	δk	k_c	δk_c	ε	$\delta\varepsilon$	c	δc	δT
Insulated	189	70	0	1	0.39	0.12	988	200	5.62
Bare	278	100	14	6	1	0.75	1353	550	2.66
Vert. (Top)	206	160	16.5	6.5	0.3	0.7	1403	500	2.4
Vert. (Bot.)	270	130	12.4	3	0.3	0.45	1059	550	4.14
Moist	191	80	17.6	9	1	0.6	1381	600	3.45
Black	163	75	16.4	4	1	0.5	1563	450	2

(All values above are in standard SI units.)

The uncertainties listed above were calculated with methods described in section 4.3

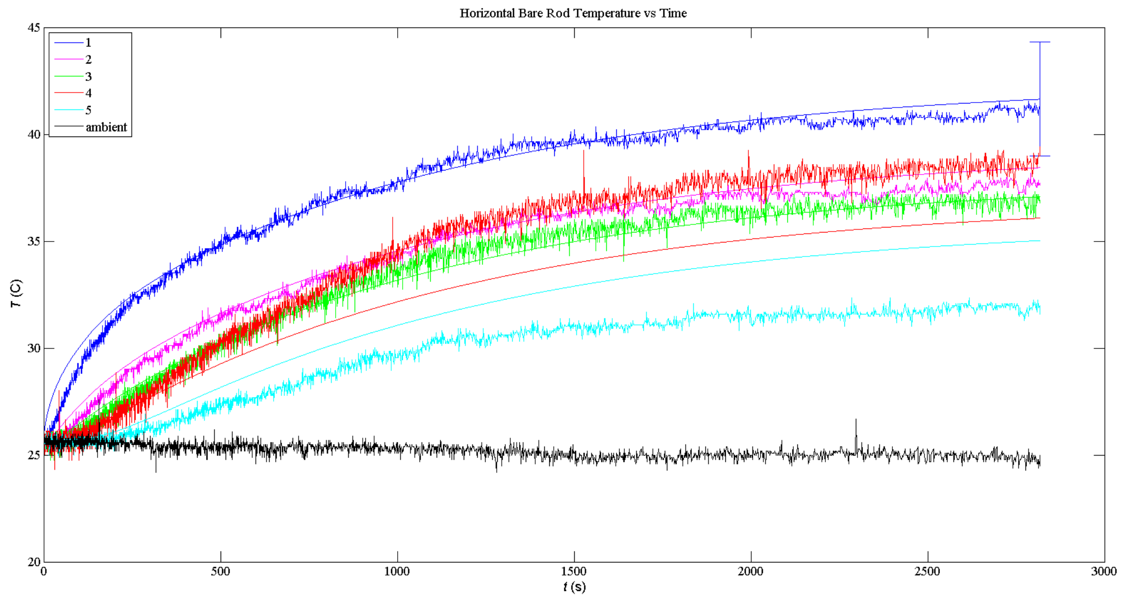


Figure 5: Bare Rod Transient Fit

The k_c and ε values for the insulated rod listed above apply to the inside of the rod. The model calculated different values of k_c and ε at the end of the rod: $k_c = 375 \pm 100 \frac{W}{m^2 * K}$, $\varepsilon = 1 \pm 1$. The large value of k_c is justified since the surface area of the heatsink was not modeled: the true k_c is likely much smaller as the surface area of the heatsink is larger.

The moist rod run had an additional parameter, the power loss from evaporation, which came out to be $P_{loss} = 0 \pm 0.8W$. Using the latent heat of evaporation of water (2257 J/g[**?**]), the mass loss rate comes out to be between 0 and $3.54 * 10^{-4}$ g/s, or about 1.27 g/h. This seems reasonable given the low power inputs into the system.

3.2 Steady State

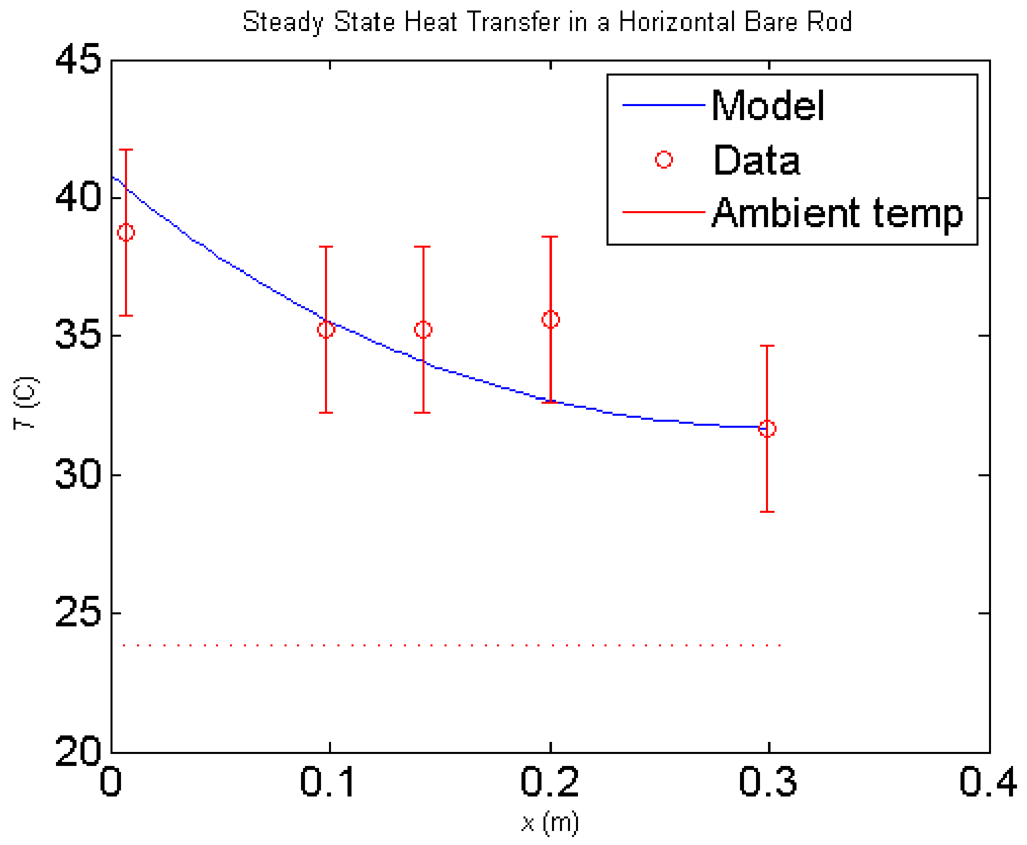


Figure 6: Bare Rod Steady State Fit

Expt.	k	δk	k_c	δk_c	ε	$\delta\varepsilon$
Insulated	180	30	-	-	-	-
Bare	200	30	20	7	0.2	0.2
Vert. (Top)	200	20	20	9	0.2	0.2
Vert. (Bot.)	200	25	25	9	0.2	0.2
Moist	200	30	20	9	0.2	0.2
Black	180	35	26	8	0.9	0.2

(All values are in standard SI units) The uncertainties listed above were estimated by eye from degree to which they changed the fit and there similarities to independently found values.

For the insulated rod the heat loss through the insulation was simulated as being proportional to temperature difference by applying a ”fudge factor” (Appendix A.3).

4 Uncertainty Analysis

4.1 Sources

There were several sources of error in these experiments, some of them are due to environmental factors, and others due to the measurement equipment.

4.1.1 Environmental Factors

Environmental factors provide error that are extremely difficult or impossible to account for in the models. An environmental factor that influences convection is the presence of air currents. Air currents affect the rate of convection however the models used only take into account natural convection (i.e. in the absence of external air currents). Another source of error is that the temperature measured at the ambient temperature sensor may not reflect the effective temperature of the entire room.

4.1.2 Measuring Equipment

The TMP35 sensors themselves created a large degree of error. Despite many attempts to calibrate the different sensors relative to one another results were not found to be consistent between different experiments or calibrations. The TMP35 sensors also have a stated error of $\pm 2°C$ and a linearity of $\pm 0.5°C$. There is also potential for a certain degree of self-heating in the TMP35 sensors due to current flow that could affect the measurement, although the manufacturer states that this is relatively low. There are also losses of precision when the voltages from the sensors are measured with the Arduino analog IO pins, which have an ADC that converts the analog voltage to a digital value between 0 and 1023.

4.2 Offsets and Scaling

It was found that the TMP35 senors have significant temperature offsets between individual sensors. On top of this, the sensors also do not scale with temperature to the same degree. In order to compensate for this the sensors were calibrated by placing them all into a water bath that was then heated from slightly below room temperature to $100°C$. The data collected from the sensors was then averaged at each point in time to provide a best estimate of the temperature with respect to time. Each individual temperature sensor was then curve fit using a linear least squares fit to provide an offset and scaling factor for each individual sensor.

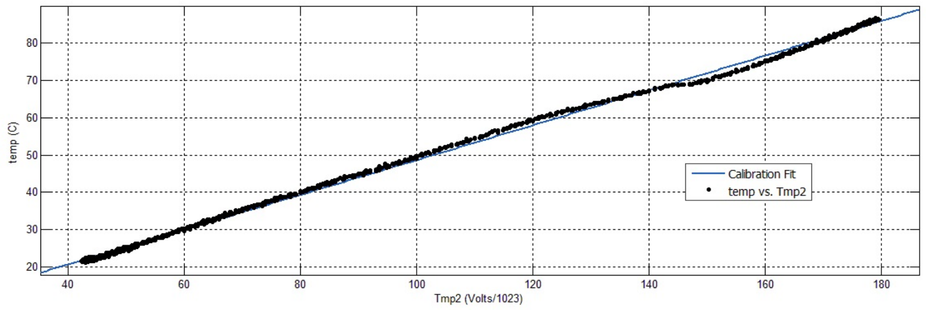
4.3 Chi-Squared Analysis

The uncertainty in each temperature measurement was estimated using χ^2 as a benchmark: it was assumed that χ^2 would be approximately equal to 1 per degree of freedom, and this determined the uncertainty of the measurements in a particular run.

Note that we also ignored the number of parameters when doing this calculation, as there were many more points being fitted than parameters (on the order of 20000 points and 5 parameters).

The standard deviation in each parameter was then estimated by varying the parameter until the χ^2 grew to 2 per degree of freedom: the difference in the result from the original value of the parameter was used as the standard deviation. This results in parameters with larger effects on the model having lower associated uncertainties.

Figure 7: Example of Calibration



Unfortunately, the sensors were still found to be inconsistent between experiments, so this method was insufficient. To make up for the discrepancy, during each individual experiment the sensors had an additional offset applied based off of the differences in the data before heating began and when each sensor was assumed to be at an equal temperature.

4.4 Noise

Much of the collected data was very noisy, but due to the large volume of collected points, most of it could be averaged out. occasionally, there are spikes in the data, that are reflected in all of the TMP35 sensors simultaneously, suggesting some disturbance, potentially with the Arduino board.

5 Summary of results

The final calculated values for the various constants averaged between experiments lie in the following ranges

- $180 < k < 240 \frac{W}{m*K}$
- $0.1 < \varepsilon < 0.3$ (sandblasted)
- $0.8 < \varepsilon < 1.0$ (Painted black)
- $1113 < c < 1188 \frac{J}{kg*K}$
- $16 < k_c < 25 \frac{W}{m^2*K}$ (for horizontal rod)
- $12 < k_c < 16$ (for vertical rod)
- $0 < P_{evap} < 0.8W$
- $Powerfraction = 90.4\%$

The power fraction was calculated from the steady state simulations, by dividing the power calculated to be going into the end of the rod by the power being put into the power resistor.

A
MATLAB Steady State Simulations and Auxiliary Functions

A.1 Bare Rod

Contents

Horizontal Bare rod Steady State simulation

```
%2015.06.03
%ENPH 257 Lab - Group 13
```

```
clear all;
close all;
%Load the results
load('June3TransientSteadyState');
```

```
radius = 0.0111; %m
length = 0.305; %m
nstep = 50;
dx = length/nstep;%m
```

```
%Thermo constants
k = 200; %W / (m * K) - conduction
sigma = 5.67e-8;%W / (m^2 * K^4) stefan-boltzman const
emsv = 0.2; % emissivity
alpha = 1.9e-5;  %m^2/s kinematic viscosity of air
g = 9.81; %m/s^2
kc = 20; %W / (m^2 * K)
emsv_elec_tape = 0.95;
width_tape = .020;%m, width of the electrical tape
pwrR_Area = ((.0155*.0207) + 2*(.0155*.002) + 2*(.0207*.002));%m^2, area of pwr resistor
emsvR = 0.8;%emsivity of power resistor
```

```
%measurement points
h5 = 0.006;%m, distance from endhole
h4 = 0.105;%m, distance from endhole
h3 = 0.163;%m, distance from endhole
h2 = 0.207;%m, distance from endhole
h1 = 0.298;%m, distance from endhole

t1st = h1 - width_tape/2;
t2st = h2 - width_tape/2;
t2end = h2 + width_tape/2;
t3st = h3 - width_tape/2;
t3end = h3 + width_tape/2;
t4end = h4 + width_tape/2;
t4st = h4 - width_tape/2;
```

```
t5end = h5 + width_tape/2;
```

```
readRangeStart = 1;
readRangeEnd = 400;
sensorDataC = 1:6;

offset = offsetCalculator('June3TransientHeating-Test2-ALLSYSTEMSNOMINAL',80,6);
calibratedData = Calibrate(readings,readRangeStart,readRangeEnd,6);%calibrates data in reading rang
e
for i = 1:6
    sensorDataC(i) = mean(calibratedData(i,:)) + offset(i);%C, averages temperature at each sensor
and applies additional offset
end

sensorPos = [h1 h2 h3 h4 h5]; %from end hole

x = 1:nstep;%just placeholder data
T = 1:nstep;%just placeholder data

Tamb = sensorDataC(6)+273;%K
T(1) = sensorDataC(5)+273;%K
```

```
%End conditions
```

$$P_{conv} = k_c(Area)dx(T(1) - T_{amb})$$

```
P_conv_end = kc * pi * radius^2 * (T(1) - Tamb);
P_conv_cyl = kc * 2 * pi * radius * dx * (T(1) - Tamb);%convection power loss for the cylindrical p
art of the end of the rod
```

$$P_{rad} = \sigma(Area)dx(T(1)^4 - T_{amb}^4)$$

```
P_rad_end = sigma * emsv * pi * radius^2 *(T(1)^4 - Tamb^4);
P_rad_cyl = sigma * emsv_elec_tape * 2 * pi * radius * dx *(T(1)^4 - Tamb^4);

P_out = P_conv_end + P_conv_cyl + P_rad_end + P_rad_cyl;
P_in = P_out;
x(1) = dx;
```

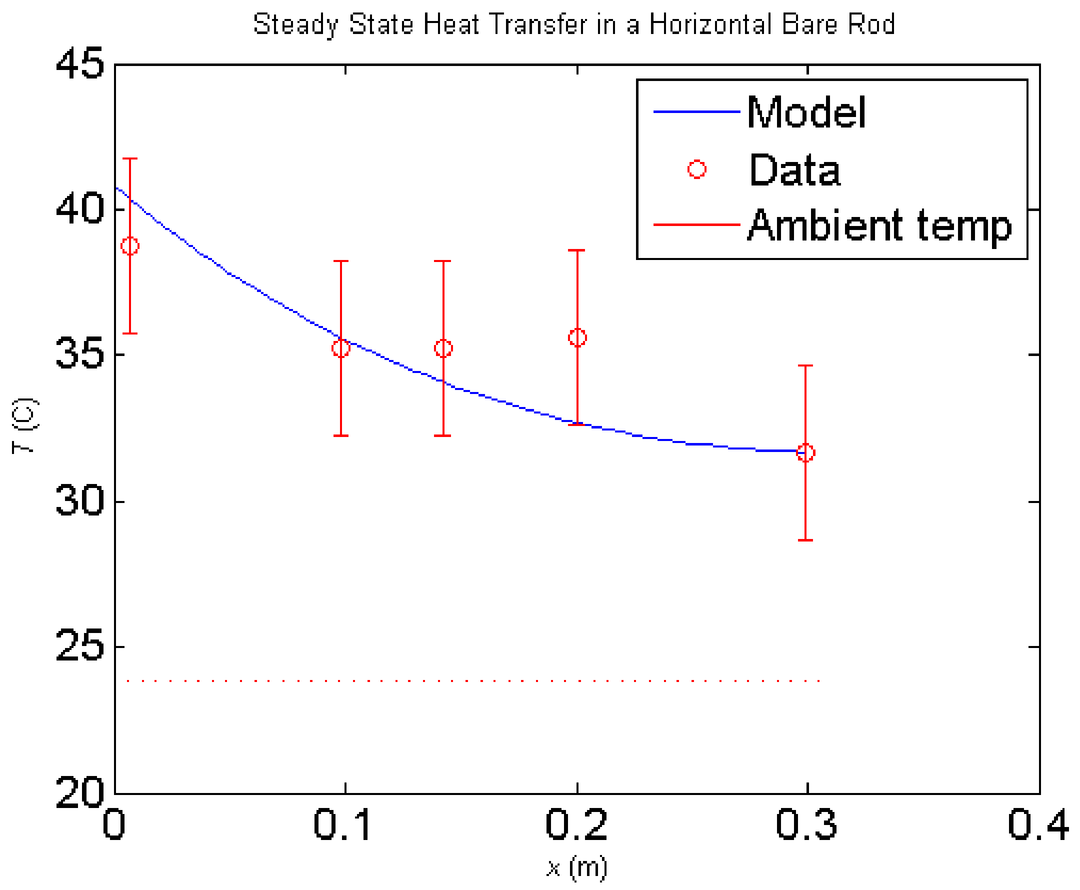
```
for i = 2:nstep
   x(i) = i * dx;
   P_out = P_in;
   T(i) = T(i-1);

   %is the slice covered in electrical tape or not?
   if (x(i) < t5end) || (x(i) > t4st && x(i) < t4end) || (x(i) > t3st && x(i) < t3end) || (x(i) > t
2st && x(i) < t2end) || x(i) > t1st
```

```
        P_conv_cyl = kc * 2 * pi * radius * dx *(T(i) - Tamb);
        P_rad = emsv_elec_tape * sigma * (2*pi*radius)*dx*(T(i)^4-Tamb^4);
        P_loss = P_conv_cyl + P_rad;
    else
        P_conv_cyl = kc * 2 * pi * radius * dx *(T(i) - Tamb);
        P_rad = emsv * sigma * (2*pi*radius)*dx*(T(i)^4-Tamb^4);
        P_loss = P_conv_cyl + P_rad;
    end

    P_in = P_out + P_loss;
    dT = P_in * dx/(k * pi * radius^2);
    T(i) = T(i) + dT;
end
```

```
pwrR_rod = P_in;%the power going into the rod is the power going into the last slice (which is the
slice adjancent to the power resistor)
pwrR_tot = 9*.6;%W, 9V*0.6A, this should equal the power loss plus the power in
pwrFract  = pwrR_rod/(pwrR_tot);%fraction of power going into rod
display(pwrFract);
```

```
pwrFract =

   0.964644349795173
```

```
figure
plot(length - x,T-273);
hold on
errorbar(length - sensorPos,sensorDataC(1:5),[3 3 3 3 3],'ro');
plot(x,Tamb-273,'r');
title('Steady State Heat Transfer in a Horizontal Bare Rod');
legend('Model','Data','Ambient temp');
xlabel('{\it x} (m)')
ylabel('{\it T} (C)')
set(gca, 'FontSize', 16)
set(gca, 'FontName', 'TimesRoman')
```



Published with MATLAB® R2012b

A.2 Black Rod

Contents

Horizontal Black rod Steady State simulation

```
%2015.06.03
%ENPH 257 Lab - Group 13
```

```
clear all;
close all;
%Load the results
load('June17BlackRodHorizontal-SteadyState');
```

```
radius = 0.0111; %m
length = 0.305; %m
nstep = 50;
dx = length/nstep;%m
```

```
%Thermo constants
k = 180; %W / (m * K) - conduction
sigma = 5.67e-8;%W / (m^2 * K^4) stefan-boltzman const
emsv = 0.9; % emissivity
fudge = 1.0; %fudge factor for convection
alpha = 1.9e-5;  %m^2/s kinematic viscosity of air
g = 9.81; %m/s^2
kc = 26; %W / (m^2 * K)
emsv_elec_tape = 0.95;
width_tape = .020;%m, width of the electrical tape
pwrR_Area = ((.0155*.0207) + 2*(.0155*.002) + 2*(.0207*.002));%m^2, area of pwr resistor
emsvR = 0.8;%emsivity of power resistor
```

```
%measurement points
h5 = 0.006;%m, distance from endhole
h4 = 0.105;%m, distance from endhole
h3 = 0.163;%m, distance from endhole
h2 = 0.207;%m, distance from endhole
h1 = 0.298;%m, distance from endhole

t1st = h1 - width_tape/2;
t2st = h2 - width_tape/2;
t2end = h2 + width_tape/2;
t3st = h3 - width_tape/2;
t3end = h3 + width_tape/2;
t4end = h4 + width_tape/2;
```

```
t4st = h4 - width_tape/2;
t5end = h5 + width_tape/2;
```

```
readRangeStart = 1;
readRangeEnd = 700;
sensorDataC = 1:6;

offset = offsetCalculator('June17BlackRodHorizontal-Heating',60,6);
calibratedData = Calibrate(readings,readRangeStart,readRangeEnd,6);%calibrates data in reading rang
e
for i = 1:6
    sensorDataC(i) = mean(calibratedData(i,:)) + offset(i);%C, averages temperature at each sensor
and applies additional offset
end

sensorPos = [h1 h2 h3 h4 h5]; %from end hole

x = 1:nstep;%just placeholder data
T = 1:nstep;%just placeholder data

Tamb = sensorDataC(6)+273;%K
T(1) = sensorDataC(5)+273;%K
```

```
%End conditions
```

$$P_{conv} = k_c(Area)dx(T(1) - T_{amb})$$

```
P_conv_end = kc * pi * radius^2 * (T(1) - Tamb);
P_conv_cyl = kc * 2 * pi * radius * dx * (T(1) - Tamb);%convection power loss for the cylindrical p
art of the end of the rod
```

$$P_{rad} = \sigma(Area)dx(T(1)^4 - T_{amb}^4)$$

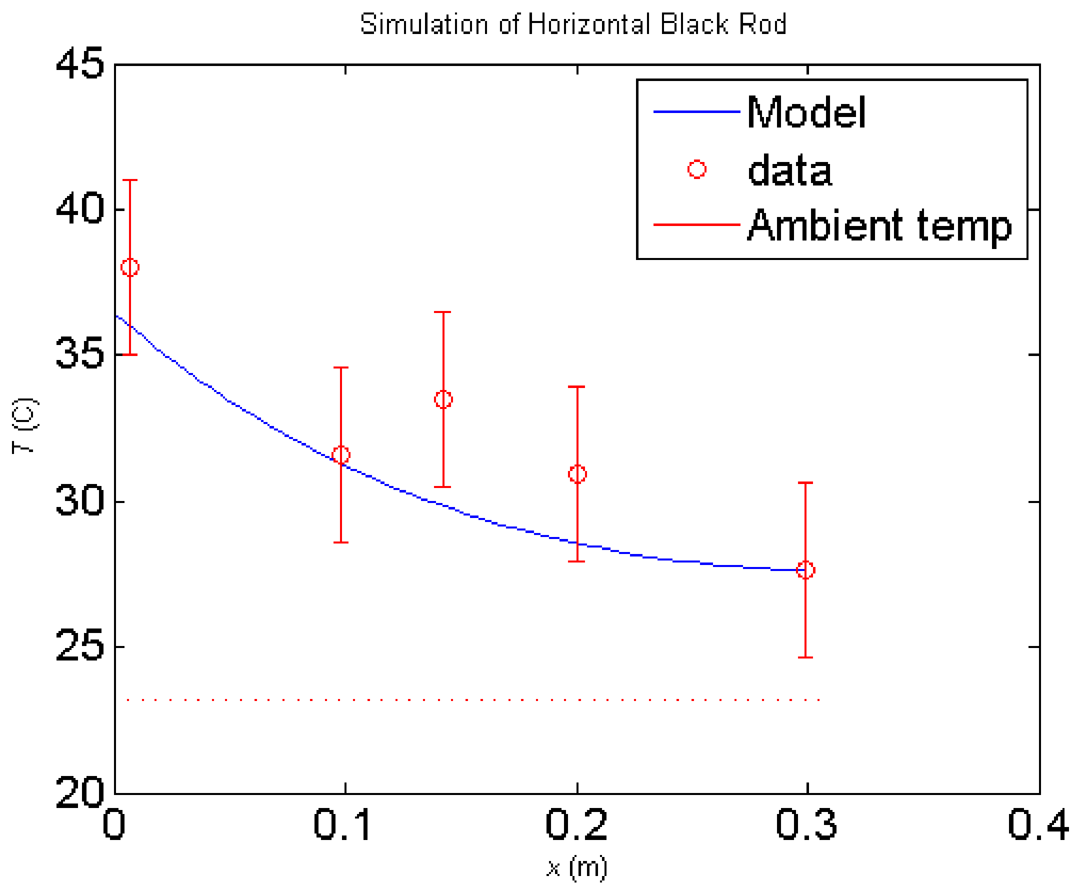
```
P_rad_end = sigma * emsv * pi * radius^2 *(T(1)^4 - Tamb^4);
P_rad_cyl = sigma * emsv_elec_tape * 2 * pi * radius * dx *(T(1)^4 - Tamb^4);

P_out = P_conv_end + P_conv_cyl + P_rad_end + P_rad_cyl;
P_in = P_out;
x(1) = dx;
```

```
for i = 2:nstep
   x(i) = i * dx;
   P_out = P_in;
   T(i) = T(i-1);

   %is the slice covered in electrical tape or not?
   if (x < t5end) | (x > t4st & x < t4end) | (x > t3st & x < t3end) | (x > t2st & x < t2end) | (x >
 t1st)
       P_conv_cyl = kc * 2 * pi * radius * dx *(T(i) - Tamb);
       P_rad = emsv_elec_tape * sigma * (2*pi*radius)*dx*(T(i)^4-Tamb^4);
       P_loss = P_conv_cyl + P_rad;
   else
       P_conv_cyl = kc * 2 * pi * radius * dx *(T(i) - Tamb);
       P_rad = emsv * sigma * (2*pi*radius)*dx*(T(i)^4-Tamb^4);
       P_loss = P_conv_cyl + P_rad;
   end

   P_in = P_out + P_loss;
   dT = P_in * dx/(k * pi * radius^2);
   T(i) = T(i) + dT;
end
```

```
%pwrR_loss = emsvR * sigma * pwrR_Area *dx*(T(nstep)^4-Tamb^4) + kc * pwrR_Area * dx *(T(nstep) - T
amb);%power loss from power resistor
pwrR_rod = P_in;%the power going into the rod is the power going into the last slice (which is the
slice adjancent to the power resistor)
pwrR_tot = 9*.6;%W, 9V*0.6A, this should equal the power loss plus the power in
pwrFract  = pwrR_rod/(pwrR_tot);%fraction of power going into rod
display(pwrFract);
```

```
pwrFract =

   0.892475316979505
```

```
figure
plot(length - x,T-273);
hold on
errorbar(length - sensorPos,sensorDataC(1:5),[3 3 3 3 3],'ro');
plot(x,Tamb-273,'r');
title('Simulation of Horizontal Black Rod');
legend('Model','data','Ambient temp');
xlabel('{\it x} (m)')
ylabel('{\it T} (C)')
set(gca, 'FontSize', 16)
set(gca, 'FontName', 'TimesRoman')
```



Published with MATLAB® R2012b

A.3 Insulated Rod

Contents

Insulated Rod Simulation

```
%ENPH 257 -group 13
```

```
close all;
%load data
load('June1SteadyStateInsulatedHeating.mat')
```

```
radius = 0.0111; %m,
length = 0.305; %m
nstep = 50;
dx = length/nstep;%m
```

```
%Thermo constants
k = 180; %W / (m * K) - conduction
insulFudge = 4; %fudge factor for power loss through insulation
pwrFrac = .9;%power fraction
```

```
%measurement points
h5 = length - 0.006;%m, distance from power resistor
h4 = length - 0.105;%m, distance from power resistor
h3 = length - 0.163;%m, distance from power resistor
h2 = length - 0.207;%m, distance from power resistor
h1 = length - 0.298;%m, distance from power resistor
```

Calibration

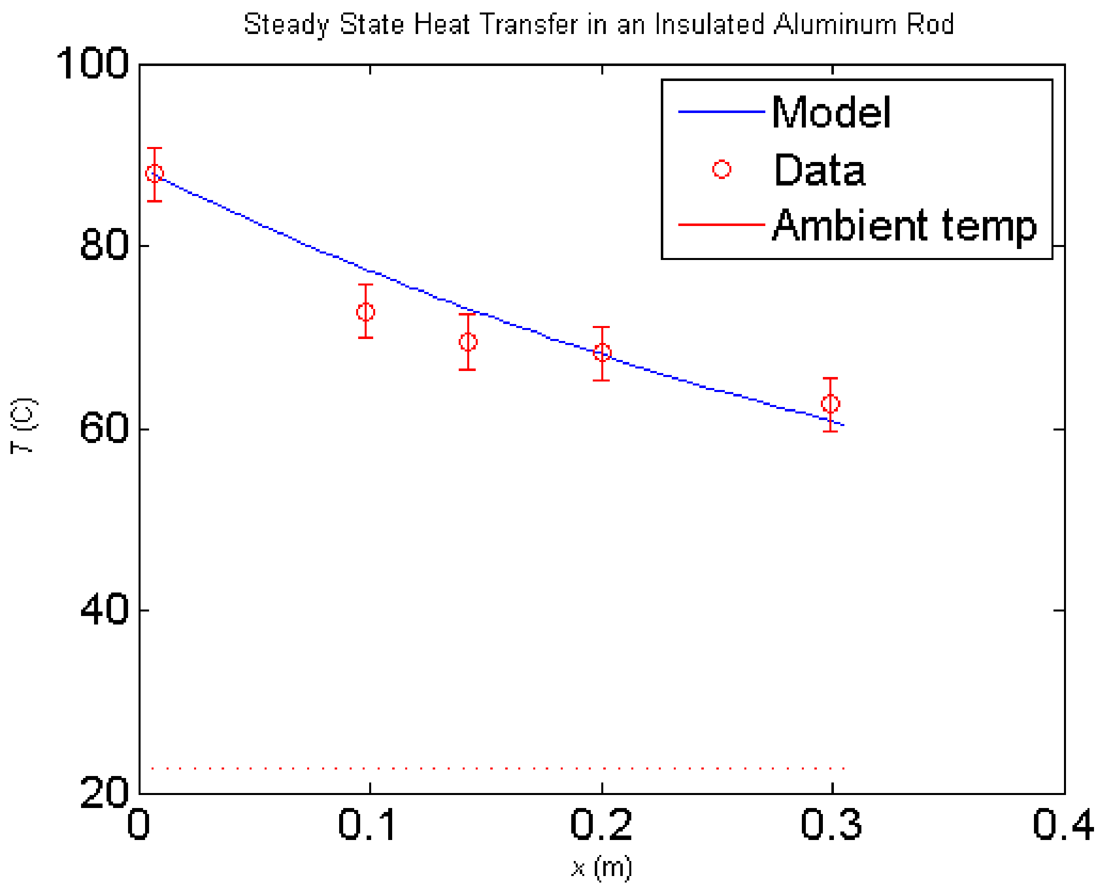
```
readRangeStart = 1;
readRangeEnd = 50;
sensorDataC = 1:6;
offset = offsetCalculator('June1TransientInsulatedHeating.mat',200,6);
calibratedData = Calibrate(readings,readRangeStart,readRangeEnd,6);%calibrates data in reading rang
e
for i = 1:6
    sensorDataC(i) = mean(calibratedData(i,:)) + offset(i);%C, averages temperature at each sensor
and applies additional offset
end
sensorPos = [h1 h2 h3 h4 h5]; %from end hole

x = 1:nstep;%just placeholder data
T = 1:nstep;%just placeholder data
```

```
Tamb = sensorDataC(6)+273;%K
T(1) = sensorDataC(1)+273;%K
```

```
%Start conditions
resistorPwr = 12*0.8;%W, 12V 0.8A
P_in = resistorPwr*pwrFrac;
P_out = P_in;
x(1) = dx;


for i = 2:nstep
    x(i) = i*dx;
    T(i) = T(i-1);
    P_in(i) = P_out;
    P_loss = insulFudge*(2*pi*radius*dx)*(T(i) - Tamb);
    P_out = P_in(i) - P_loss;
    dT = P_in(i) * dx/(k * pi * radius^2);
    T(i) = T(i) - dT;
end
display(P_out);%power going out heat sink
```

```
P_out =

   4.446466858327812
```

```
figure
plot(x,T-273);
hold on
errorbar(sensorPos,sensorDataC(1:5),[3 3 3 3 3],'ro');
plot(x,Tamb-273,'r');
title('Steady State Heat Transfer in an Insulated Aluminum Rod');
xlabel('{\it x} (m)')
ylabel('{\it T} (C)')
set(gca, 'FontSize', 16)
set(gca, 'FontName', 'TimesRoman')

legend('Model','Data','Ambient temp');
```



Published with MATLAB® R2012b

A.4 Vertical Rod, Heated From the Bottom

Contents

Vertical Bare Rod - Top Heating Steady State simulation

```
%2015.06.03
%ENPH 257 Lab - Group 13
```

```
clear all;
close all;
%Load the results
load('june10VertBareRodSteadyState-TopPowered');
```

```
radius = 0.0111; %m
length = 0.305; %m
nstep = 50;
dx = length/nstep;%m
```

```
%Thermo constants
k = 200; %W / (m * K) - conduction
sigma = 5.67e-8;%W / (m^2 * K^4) stefan-boltzman const
emsv = 0.2; % emissivity
fudge = 1.0; %fudge factor for convection
alpha = 1.9e-5;  %m^2/s kinematic viscosity of air
g = 9.81; %m/s^2
kc = 20; %W / (m^2 * K)
emsv_elec_tape = 0.95;
width_tape = .020;%m, width of the electrical tape
pwrR_Area = ((.0155*.0207) + 2*(.0155*.002) + 2*(.0207*.002));%m^2, area of pwr resistor
emsvR = 0.8;%emsivity of power resistor
```

```
%measurement points
h5 = 0.006;%m, distance from endhole
h4 = 0.105;%m, distance from endhole
h3 = 0.163;%m, distance from endhole
h2 = 0.207;%m, distance from endhole
h1 = 0.298;%m, distance from endhole

t1st = h1 - width_tape/2;
t2st = h2 - width_tape/2;
t2end = h2 + width_tape/2;
t3st = h3 - width_tape/2;
t3end = h3 + width_tape/2;
t4end = h4 + width_tape/2;
```

```
t4st = h4 - width_tape/2;
t5end = h5 + width_tape/2;
```

```
readRangeStart = 1;
readRangeEnd = 100;
sensorDataC = 1:6;

offset = offsetCalculator('June10VertBareRodHeating-TopPower',20,1);
calibratedData = Calibrate(readings,readRangeStart,readRangeEnd,1);%calibrates data in reading rang
e
for i = 1:6
    sensorDataC(i) = mean(calibratedData(i,:)) + offset(i);%C, averages temperature at each sensor
and applies additional offset
end

sensorPos = [h1 h2 h3 h4 h5]; %from end hole

x = 1:nstep;%just placeholder data
T = 1:nstep;%just placeholder data

Tamb = sensorDataC(6)+273;%K
T(1) = sensorDataC(5)+273;%K
```

```
%End conditions
```

$$P_{conv} = k_c(Area)dx(T(1) - T_{amb})$$

```
P_conv_end = kc * pi * radius^2 * (T(1) - Tamb);
P_conv_cyl = kc * 2 * pi * radius * dx * (T(1) - Tamb);%convection power loss for the cylindrical p
art of the end of the rod
```

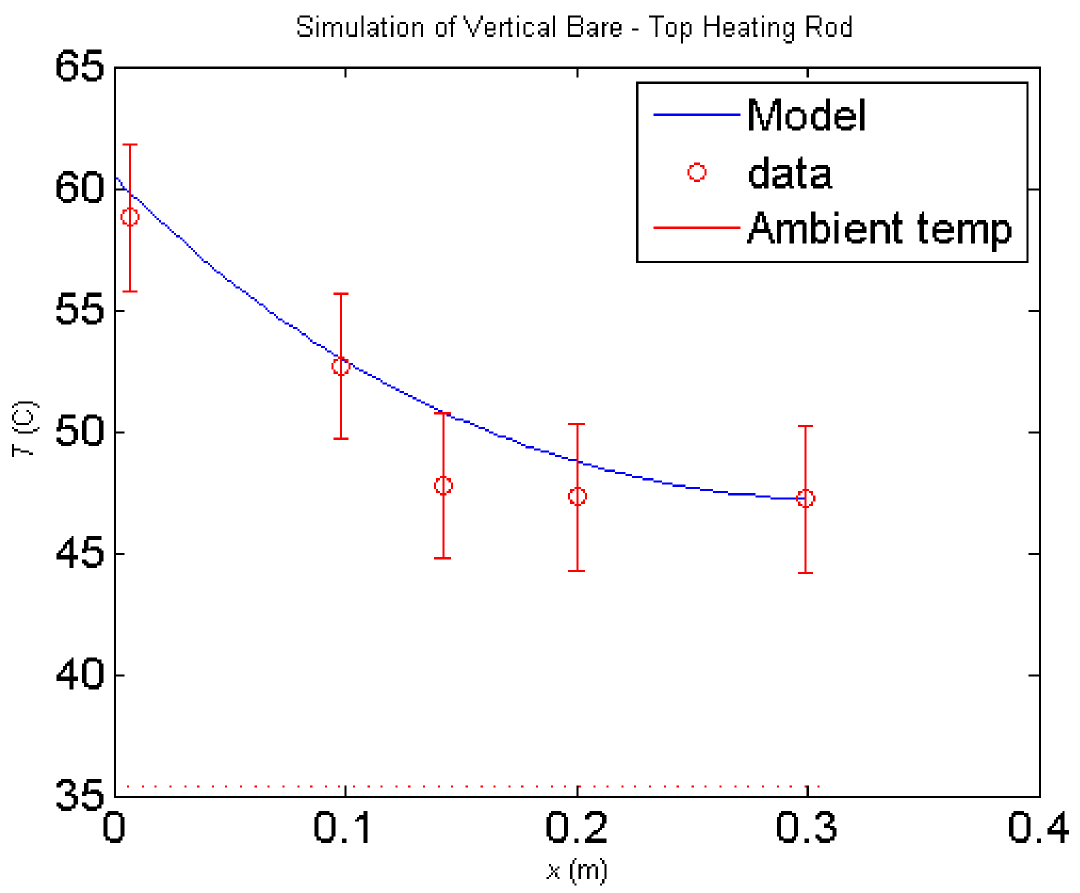
$$P_{rad} = \sigma(Area)dx(T(1)^4 - T_{amb}^4)$$

```
P_rad_end = sigma * emsv * pi * radius^2 *(T(1)^4 - Tamb^4);
P_rad_cyl = sigma * emsv_elec_tape * 2 * pi * radius * dx *(T(1)^4 - Tamb^4);

P_out = P_conv_end + P_conv_cyl + P_rad_end + P_rad_cyl;
P_in = P_out;
x(1) = dx;
```

```
for i = 2:nstep
   x(i) = i * dx;
   P_out = P_in;
   T(i) = T(i-1);

   %is the slice covered in electrical tape or not?
   if (x < t5end) | (x > t4st & x < t4end) | (x > t3st & x < t3end) | (x > t2st & x < t2end) | (x >
 t1st)
       P_conv_cyl = kc * 2 * pi * radius * dx *(T(i) - Tamb);
       P_rad = emsv_elec_tape * sigma * (2*pi*radius)*dx*(T(i)^4-Tamb^4);
       P_loss = P_conv_cyl + P_rad;
   else
       P_conv_cyl = kc * 2 * pi * radius * dx *(T(i) - Tamb);
       P_rad = emsv * sigma * (2*pi*radius)*dx*(T(i)^4-Tamb^4);
       P_loss = P_conv_cyl + P_rad;
   end

   P_in = P_out + P_loss;
   dT = P_in * dx/(k * pi * radius^2);
   T(i) = T(i) + dT;
end
```

```
%pwrR_loss = emsvR * sigma * pwrR_Area *dx*(T(nstep)^4-Tamb^4) + kc * pwrR_Area * dx *(T(nstep) - T
amb);%power loss from power resistor
pwrR_rod = P_in;%the power going into the rod is the power going into the last slice (which is the
slice adjancent to the power resistor)
pwrR_tot = 9*.6;%W, 9V*0.6A, this should equal the power loss plus the power in
pwrFract  = pwrR_rod/(pwrR_tot);%fraction of power going into rod
```

```
figure
plot(length - x,T-273);
hold on
errorbar(length - sensorPos,sensorDataC(1:5),[3 3 3 3 3],'ro');
plot(x,Tamb-273,'r');
title('Simulation of Vertical Bare - Top Heating Rod');
legend('Model','data','Ambient temp');
xlabel('{\it x} (m)')
ylabel('{\it T} (C)')
set(gca, 'FontSize', 16)
set(gca, 'FontName', 'TimesRoman')
```



Published with MATLAB® R2012b

A.5 Vertical Rod, Heated From the Top

Contents

Vertical Bare Rod - Bottom Heating Steady State simulation

```
%2015.06.03
%ENPH 257 Lab - Group 13
```

```
clear all;
close all;
%Load the results
load('June10VertBareRodSteadyState-BottomPowered');
```

```
radius = 0.0111; %m
length = 0.305; %m
nstep = 50;
dx = length/nstep;%m
```

```
%Thermo constants
k = 200; %W / (m * K) - conduction
sigma = 5.67e-8;%W / (m^2 * K^4) stefan-boltzman const
emsv = 0.2; % emissivity
fudge = 1.0; %fudge factor for convection
alpha = 1.9e-5;  %m^2/s kinematic viscosity of air
g = 9.81; %m/s^2
kc = 25; %W / (m^2 * K)
emsv_elec_tape = 0.95;
width_tape = .020;%m, width of the electrical tape
pwrR_Area = ((.0155*.0207) + 2*(.0155*.002) + 2*(.0207*.002));%m^2, area of pwr resistor
emsvR = 0.8;%emsivity of power resistor
```

```
%measurement points
h5 = 0.006;%m, distance from endhole
h4 = 0.105;%m, distance from endhole
h3 = 0.163;%m, distance from endhole
h2 = 0.207;%m, distance from endhole
h1 = 0.298;%m, distance from endhole

t1st = h1 - width_tape/2;
t2st = h2 - width_tape/2;
t2end = h2 + width_tape/2;
t3st = h3 - width_tape/2;
t3end = h3 + width_tape/2;
t4end = h4 + width_tape/2;
```

```
t4st = h4 - width_tape/2;
t5end = h5 + width_tape/2;
```

```
readRangeStart = 1;
readRangeEnd = 450;
sensorDataC = 1:6;

offset = offsetCalculator('June10VertBareRodHeating-BottomPowered',20,1);
calibratedData = Calibrate(readings,readRangeStart,readRangeEnd,1);%calibrates data in reading rang
e
for i = 1:6
    sensorDataC(i) = mean(calibratedData(i,:)) + offset(i);%C, averages temperature at each sensor
and applies additional offset
end

sensorPos = [h1 h2 h3 h4 h5]; %from end hole

x = 1:nstep;%just placeholder data
T = 1:nstep;%just placeholder data

Tamb = sensorDataC(6)+273;%K
T(1) = sensorDataC(5)+273;%K
```

```
%End conditions
```

$$P_{conv} = k_c(Area)dx(T(1) - T_{amb})$$

```
P_conv_end = kc * pi * radius^2 * (T(1) - Tamb);
P_conv_cyl = kc * 2 * pi * radius * dx * (T(1) - Tamb);%convection power loss for the cylindrical p
art of the end of the rod
```

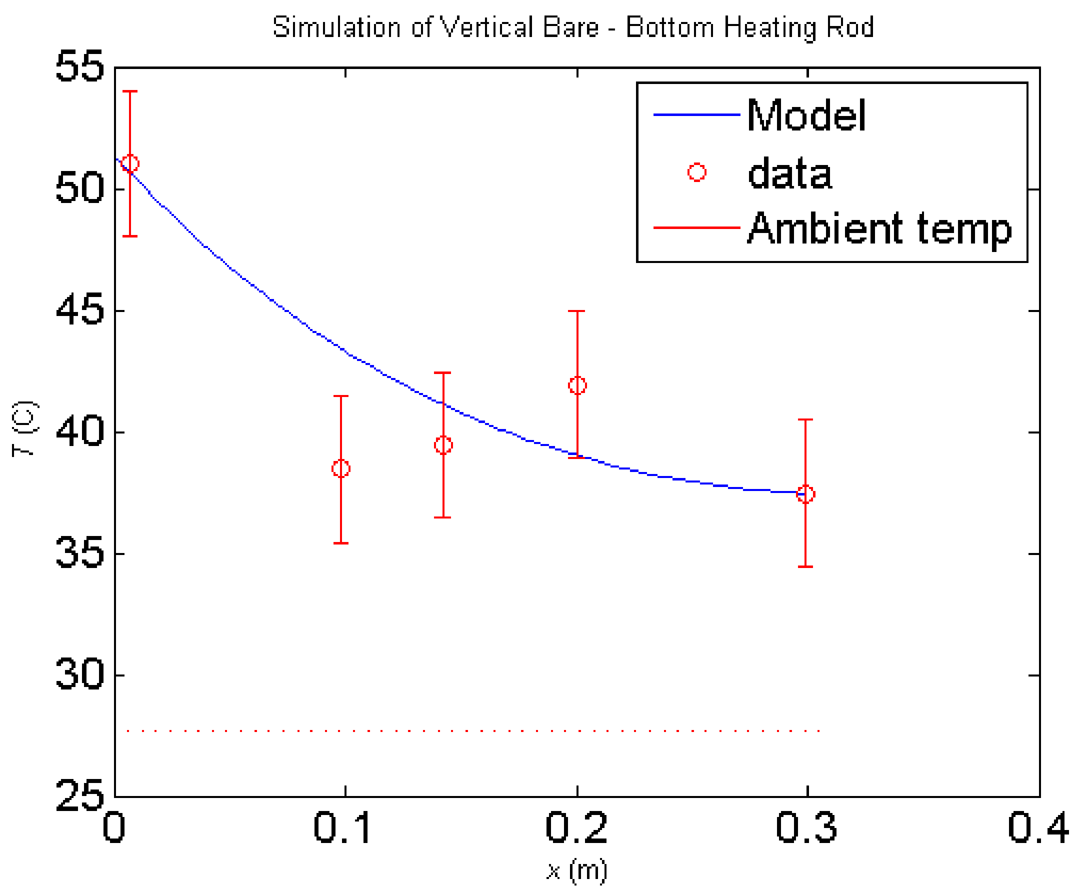
$$P_{rad} = \sigma(Area)dx(T(1)^4 - T_{amb}^4)$$

```
P_rad_end = sigma * emsv * pi * radius^2 *(T(1)^4 - Tamb^4);
P_rad_cyl = sigma * emsv_elec_tape * 2 * pi * radius * dx *(T(1)^4 - Tamb^4);

P_out = P_conv_end + P_conv_cyl + P_rad_end + P_rad_cyl;
P_in = P_out;
x(1) = dx;
```

```
for i = 2:nstep
   x(i) = i * dx;
   P_out = P_in;
   T(i) = T(i-1);

   %is the slice covered in electrical tape or not?
   if (x < t5end) | (x > t4st & x < t4end) | (x > t3st & x < t3end) | (x > t2st & x < t2end) | (x >
 t1st)
       P_conv_cyl = kc * 2 * pi * radius * dx *(T(i) - Tamb);
       P_rad = emsv_elec_tape * sigma * (2*pi*radius)*dx*(T(i)^4-Tamb^4);
       P_loss = P_conv_cyl + P_rad;
   else
       P_conv_cyl = kc * 2 * pi * radius * dx *(T(i) - Tamb);
       P_rad = emsv * sigma * (2*pi*radius)*dx*(T(i)^4-Tamb^4);
       P_loss = P_conv_cyl + P_rad;
   end

   P_in = P_out + P_loss;
   dT = P_in * dx/(k * pi * radius^2);
   T(i) = T(i) + dT;
end
```

```
%pwrR_loss = emsvR * sigma * pwrR_Area *dx*(T(nstep)^4-Tamb^4) + kc * pwrR_Area * dx *(T(nstep) - T
amb);%power loss from power resistor
pwrR_rod = P_in;%the power going into the rod is the power going into the last slice (which is the
slice adjancent to the power resistor)
pwrR_tot = 9*.6;%W, 9V*0.6A, this should equal the power loss plus the power in
pwrFract  = pwrR_rod/(pwrR_tot);%fraction of power going into rod
```

```
figure
plot(length - x,T-273);
hold on
errorbar(length - sensorPos,sensorDataC(1:5),[3 3 3 3 3],'ro');
plot(x,Tamb-273,'r');
title('Simulation of Vertical Bare - Bottom Heating Rod');
legend('Model','data','Ambient temp');
xlabel('{\it x} (m)')
ylabel('{\it T} (C)')
set(gca, 'FontSize', 16)
set(gca, 'FontName', 'TimesRoman')
```



Published with MATLAB® R2012b

A.6 Moist Rod

Horizontal Moist rod Steady State simulation

```
%2015.06.03
%ENPH 257 Lab - Group 13
```

```
clear all;
close all;
%Load the results
load('June12MoistRodSteadyState.mat');
```

```
radius = 0.0111; %m
length = 0.305; %m
nstep = 50;
dx = length/nstep;%m
```

```
%Thermo constants
k = 200; %W / (m * K) - conduction
sigma = 5.67e-8;%W / (m^2 * K^4) stefan-boltzman const
emsv = 0.2; %emissivity
moist_kc = 5.0; %W / (m^2 * K)
fudgeE = 1.5; %fudge factor for evaporation
alpha = 1.9e-5;%m^2/s kinematic viscosity of air
g = 9.81; %m/s^2
kc = 20.0; %W / (m^2 * K)
emsv_elec_tape = 0.95;
emsv_cloth = 0.9;
width_tape = .020;%m, width of the electrical tape
pwrR_Area = ((15.5*20.7) + 2*(15.5*2) + 2*(20.7*2))*10^-6;%m^2, area of pwr resistor
emsvR = 0.8;%emsivity of power resistor
latentHeatWaterEvap = 2260; %J/g
totalEvapPowerLoss = 0.0; %0.0 Value is a placeholder until loop is exicuted
```

```
%measurement points
h5 = 0.006;%m, distance from endhole
h4 = 0.105;%m, distance from endhole
h3 = 0.163;%m, distance from endhole
h2 = 0.207;%m, distance from endhole
h1 = 0.298;%m, distance from endhole

t1st = h1 - width_tape/2;
t2st = h2 - width_tape/2;
t2end = h2 + width_tape/2;
t3st = h3 - width_tape/2;
t3end = h3 + width_tape/2;
t4end = h4 + width_tape/2;
t4st = h4 - width_tape/2;
t5end = h5 + width_tape/2;
```

```
ambPin = 1;
readRangeStart = 1;
readRangeEnd = 400;
sensorDataC = 1:6;

offset = moistOffsetCalculator('June12MoistRodHeating.mat',120,ambPin);
calibratedData = moistCalibrate(readings,readRangeStart,readRangeEnd,ambPin);%calibrates data
 in reading range
for i = 1:6
    sensorDataC(i) = mean(calibratedData(i,:)) + offset(i);%C, averages temperature at each s
ensor and applies additional offset
end

sensorPos = [h1 h2 h3 h4 h5]; %from end hole

x = 1:nstep;%just placeholder data
T = 1:nstep;%just placeholder data

Tamb = sensorDataC(6)+273;%K
T(1) = sensorDataC(5)+273;%K
```

```
%End conditions

P_conv_end = kc * pi * radius^2 * (T(1) - Tamb);
% $P_{conv} = k_c  (2 \pi r) dx (T(1)^4 - T_{amb}^4)$
P_conv_an = kc * 2 * pi * radius * dx * (T(1) - Tamb);%convection power loss for the annulus
of the end of the rod
P_rad_end = sigma * emsv * pi * radius^2 *(T(1)^4 - Tamb^4);
P_rad_an = sigma * emsv_elec_tape * 2 * pi * radius * dx *(T(1)^4 - Tamb^4);

P_out = P_conv_end + P_conv_an + P_rad_end + P_rad_an;
P_in = P_out;
x(1) = dx;
```

```
for i = 2:nstep
   x(i) = i * dx;
   P_out = P_in;
   T(i) = T(i-1);



   %is the slice on the tape or not?
 if (x(i) < t5end) || (x(i) > t4st && x(i) < t4end) || (x(i) > t3st && x(i) < t3end) || (x(i)
 > t2st && x(i) < t2end) || (x(i) > t1st)
       P_conv = kc * 2 * pi * radius * dx *(T(i) - Tamb);
       P_rad = emsv_elec_tape * sigma * (2*pi*radius)*dx*(T(i)^4-Tamb^4);
       P_loss = P_conv + P_rad;
 else

     if (x(i) < t4st && x(i) > t5end )
          P_conv = moist_kc * 2 * pi * radius * dx *(T(i) - Tamb);
          P_rad = emsv_cloth * sigma * (2*pi*radius) * dx * (T(i)^4-Tamb^4);
          P_evap = fudgeE * dx * (T(i)-Tamb);
```

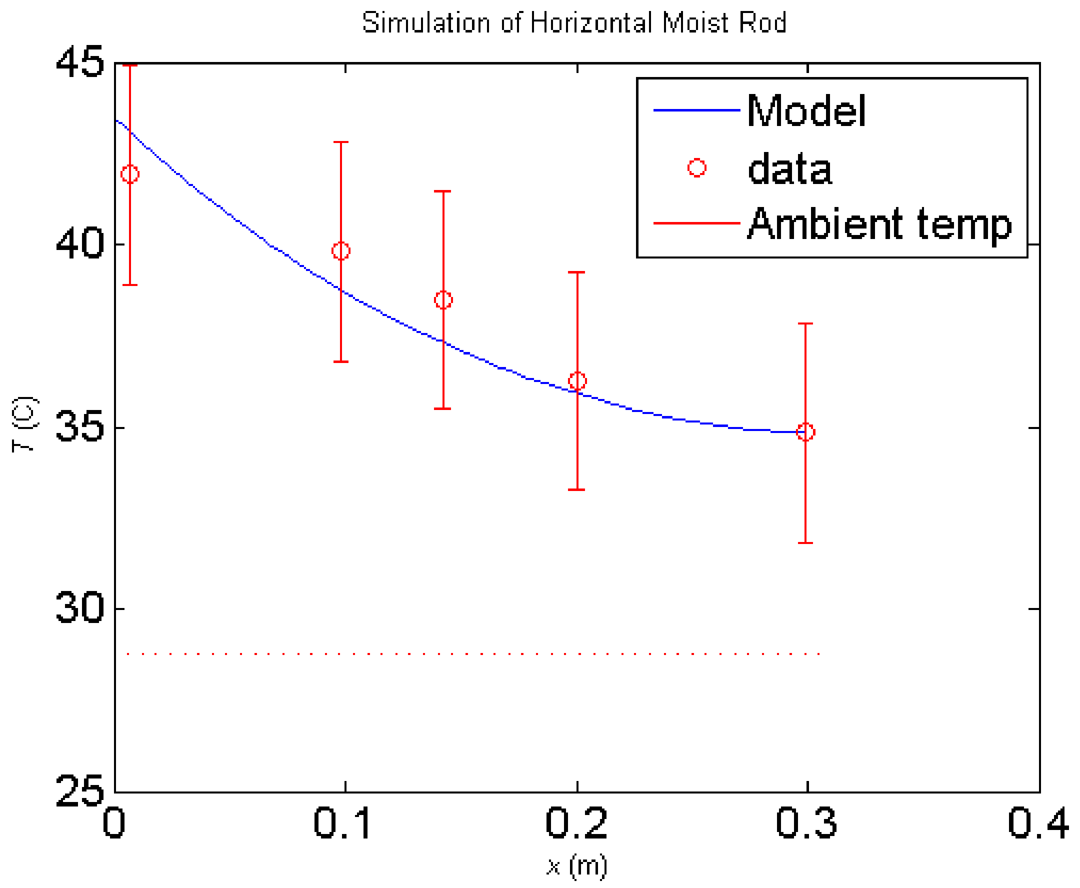
```
            P_loss = P_conv + P_rad + P_evap;

            totalEvapPowerLoss = totalEvapPowerLoss + P_evap;

        else

            P_conv = kc * 2 * pi * radius * dx *(T(i) - Tamb);
            P_rad = emsv * sigma * (2*pi*radius)*dx*(T(i)^4-Tamb^4);
            P_loss = P_conv + P_rad;
        end
    end

    P_in = P_out + P_loss;
    dT = P_in * dx/(k * pi * radius^2);
    T(i) = T(i) + dT;
end
```

```
pwrR_loss = emsvR * sigma * pwrR_Area *dx*(T(nstep)^4-Tamb^4) + kc * pwrR_Area * dx *(T(nstep
) - Tamb);%power loss from power resistor
pwrR_rod = P_in;%the power going into the rod is the power going into the last slice (which i
s the slice adjancent to the power resistor)
pwrR_tot = 9*.6;%W, 9V*0.6A, this should equal the power loss plus the power in
pwrFract  = pwrR_rod/(pwrR_tot);%fraction of power going into rod
display(pwrR_tot)
display(pwrR_loss+pwrR_rod);
```

```
pwrR_tot =

    5.4000


ans =

    4.7125
```

```
figure
plot(length - x,T-273);
hold on
errorbar(length - sensorPos,sensorDataC(1:5),[3 3 3 3 3],'ro');
plot(x,Tamb-273,'r');
title('Simulation of Horizontal Moist Rod');
legend('Model','data','Ambient temp');
xlabel('{\it x} (m)')
ylabel('{\it T} (C)')
set(gca, 'FontSize', 16)
set(gca, 'FontName', 'TimesRoman')

waterEvapPerSec = totalEvapPowerLoss / latentHeatWaterEvap %g/s
display(totalEvapPowerLoss)%W
```

```
waterEvapPerSec =

   3.3383e-04


totalEvapPowerLoss =

    0.7545
```



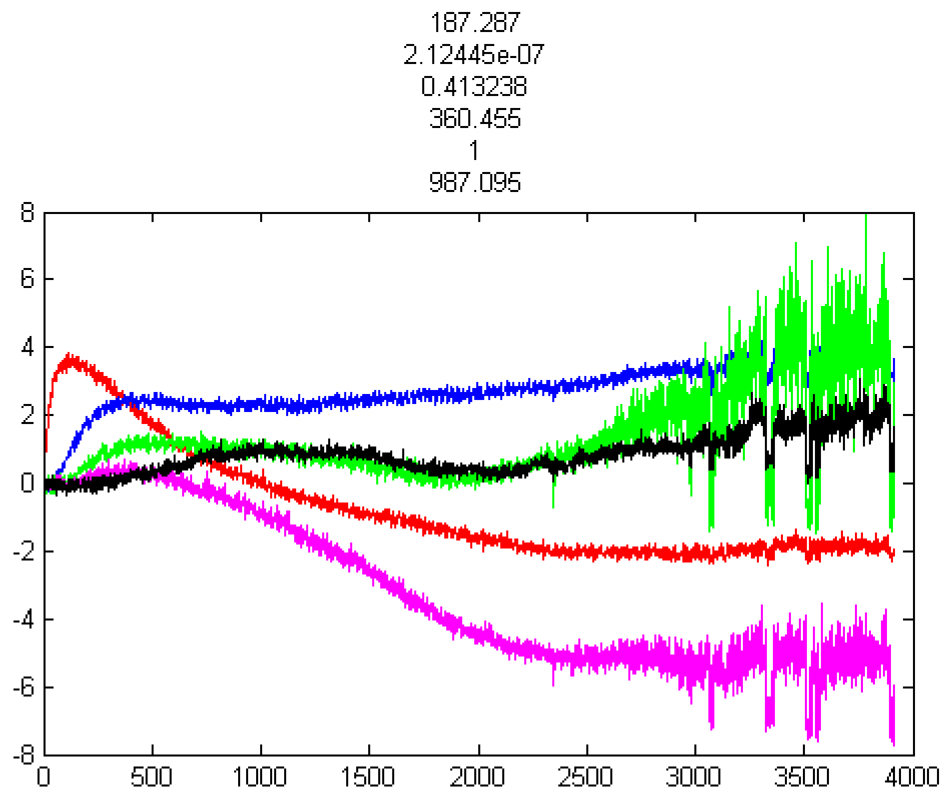
Published with MATLAB® R2013a

B MATLAB Transient Simulations and Auxiliary Functions

B.1 Model Function

```
load('june1TransientInsulatedHeating.mat');
lb = [0 0 0 0 0 100];
ub = [Inf Inf 1 Inf 1 10000];
x = [190 0 0 375 1 988]; %initial guess
offsets2 = [0.3312 2.9439 0.75 3.3023 -1.0658 0];
%parameters: 1st is conduction constant, 2nd is convection inside tube,
%3rd is emissivity inside tube, 4th is convection outside tube, 5th is
%emissivity outside tube

tOffset = 59.1847;
reading1 = 220;
readingF = 4130;
amb1 = 0;
Pin = 9.9;
eq = 0;
iceEnd = 0;
blackRod = 0;
moistRod = 0;
%make sure readings are loaded here
[x, errsum] = lsqnonlin(@(x)transientFinDiffFuncNonLin(x, readings, tOffset, ...
    reading1, readingF, offsets2, amb1, Pin, eq, iceEnd, blackRod, moistRod), ...
    x, lb, ub);
x
errsum
```

```
Local minimum possible.

lsqnonlin stopped because the final change in the sum of squares relative
its initial value is less than the default value of the function tolerance




x =

  187.2870    0.0000    0.4132  360.4545    1.0000  987.0955


errsum =

   1.2364e+05
```



Published with MATLAB® R2013a

B.2 Use of Model Function

```
function [F, T] = transientFinDiffFuncNonLin(param, readings, tOffset, ...
reading1, readingF, offsets2, amb1, Pin, eq, iceEnd, blackRod, moistRod)
    %Finite difference calculations for aluminum rod transients
    %param meanings: 1=k, 2=kc, 3=eps, 4=kcEnd, 5=epsEnd, 6=c. Parameters
    %that are not used in a particular scenario are
    %ignored.
    %Other parameters:
    %readings: Data from sensors
    %tOffset: starting time for heating
    %reading1: starting reading for heating
    %readingF: final reading to use
    %offsets2: secondary offset to ensure all sensors are identical
    %initially
    %amb1: whether the ambient pin is 1 (if false, 6)
    %Pin: input power to rod
    %eq: whether kc and eps are equal inside and at end of rod
    %iceEnd: whether the end of the rod is 0 degrees C
    %blackRod: for black rod experiment: shuffles sensors
    %Return values:
    %F: a matrix containing the errors in each reading
    %T: a matrix containing temperatures for the sensor locations.

    %%Temp sensor calibration stuff
    factors = [1.79 1.81 1.53 1.46 2.06 2];
    offsets = [4.14 2.25 0.16 3.64 -0.35 0];

    %%shuffle sensors for black rod
    if(blackRod)
        factors = factors([2 3 5 4 1 6]);
        offsets = offsets([2 3 5 4 1 6]);
    end

    sensorPos = [1 6 9 12 19];

    %%Set up constants
    sigma = 5.670e-8; % W /^-2 K^-4
    c = param(6); %specific heat capacity
    rho = 2700; %kg/m^3

    L = 0.305; %length
    r = 0.0111; %radius
    A = pi*r^2;

    %%Handle case for differing ambient pins
    if(amb1)
        Tamb = mean(squeeze(readings(1, 1, 1:35)))/2; %ambient temperature
        factors = [2 factors(1:5)];
        offsets = [0 offsets(1:5)];
    else
        Tamb = mean(squeeze(readings(1, 6, 1:35)))/2;
    end
```

```
nstepsT = ceil(readings(3, 6, readingF));

time = nstepsT; %seconds

nstepsX = 19;

dt = time / nstepsT;
dx = L / nstepsX;

m = dx*A*rho;

T = zeros(nstepsT + 1, nstepsX + 1); %K
P = zeros(nstepsT + 1, nstepsX + 1); %W

T(1, :) = Tamb;
P(:, 1) = Pin;

%%Iteration step
for i = 1:(nstepsT)
    for j = 1:(nstepsX + 1)
        if j == nstepsX + 1
            S = A;
            Pcond = 0;
            if(eq)
                eps = param(3);
                kc = param(2);
            else
                eps = param(5);
                kc = param(4);
            end
            if(iceEnd) Tamb = 0; end;%(ice water)
        else
            S = dx*pi*r*2;
            Pcond = -param(1)*A*(T(i, j) - T(i, j+1))/dx;
            eps = param(3);
            kc = param(2); %represents heat loss through insulation
            %actually probably represents conduction
            if(iceEnd) Tamb = T(1, 1); end; %(ice water)
        end
        Pconv = -S * (T(i, j) - Tamb) * kc;
        Prad = -eps * S * sigma * ((T(i, j) + 273.15)^4 - (Tamb+273.15)^4);
        Ptot = Pcond + Pconv + Prad + P(i, j);
        if(j >= 12 && j < 19 && moistRod)
            Ptot = Pcond + Pconv/2 + Prad - param(7) / 6 + P(i, j);
        end
        P(i, j+1) = -Pcond;
        T(i+1, j) = Ptot*dt/(c*m) + T(i, j);
    end
end
rng = reading1:readingF;

%%Calculate errors
for i = 1:5
    j = i+amb1;
```

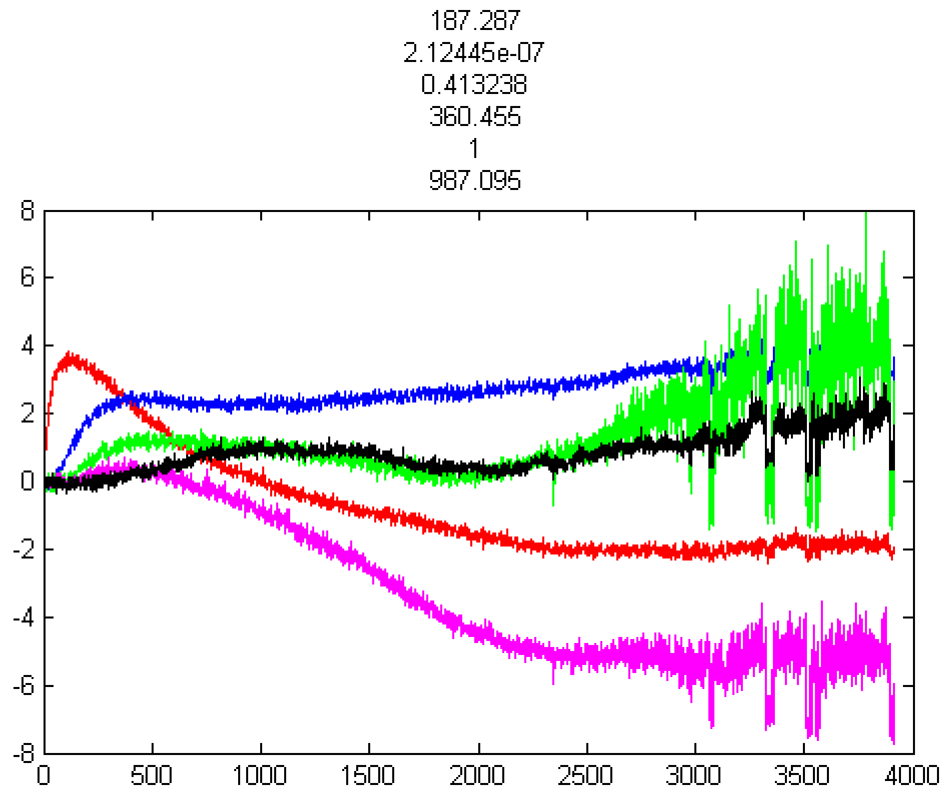
```
        times = squeeze(readings(3, j, rng));
        values = squeeze(readings(1, j, rng));
        %if(i == 4); j = 3; elseif(i == 3); j=4; else j = i; end
        F(j-amb1, :) = interp1((1:length(T(:,i)))+tOffset, ...
            T(:,sensorPos(i)), times)-(values-offsets(j))/factors(j)+offsets2(j);
    end

    %%Plot residuals
    p = plot(F');
    title(param);

    clrs = 'rbgmk';
    for i=1:numel(p)
        set(p(i), 'color', clrs(i));
    end
    pause(.01);
end
```

Published with MATLAB® R2013a

B.3 Plotting

Table of Contents

```
[F, T] = transientFinDiffFuncNonLin(x, readings, tOffset, reading1, readingF, ...
    offsets2, amb1, Pin, eq, iceEnd, blackRod, moistRod);
```



Calculate sum of errors squared.

```
errsum = sum(sum(F.^2))
```

```
errsum =

   1.2364e+05
```

Calculate error per degree of freedom. Since we have so many points we ignore the number of parameter values.

```
errSumPer = errsum/(readingF-reading1)
```

```
errSumPer =

   31.6207
```

Calculate approximate uncertainty of each measurement.

```
uncertainty = sqrt(errSumPer)
```

```
uncertainty =

    5.6232
```

Parameters of this fit: [k kcInner epsInner kcOuter epsOuter c]

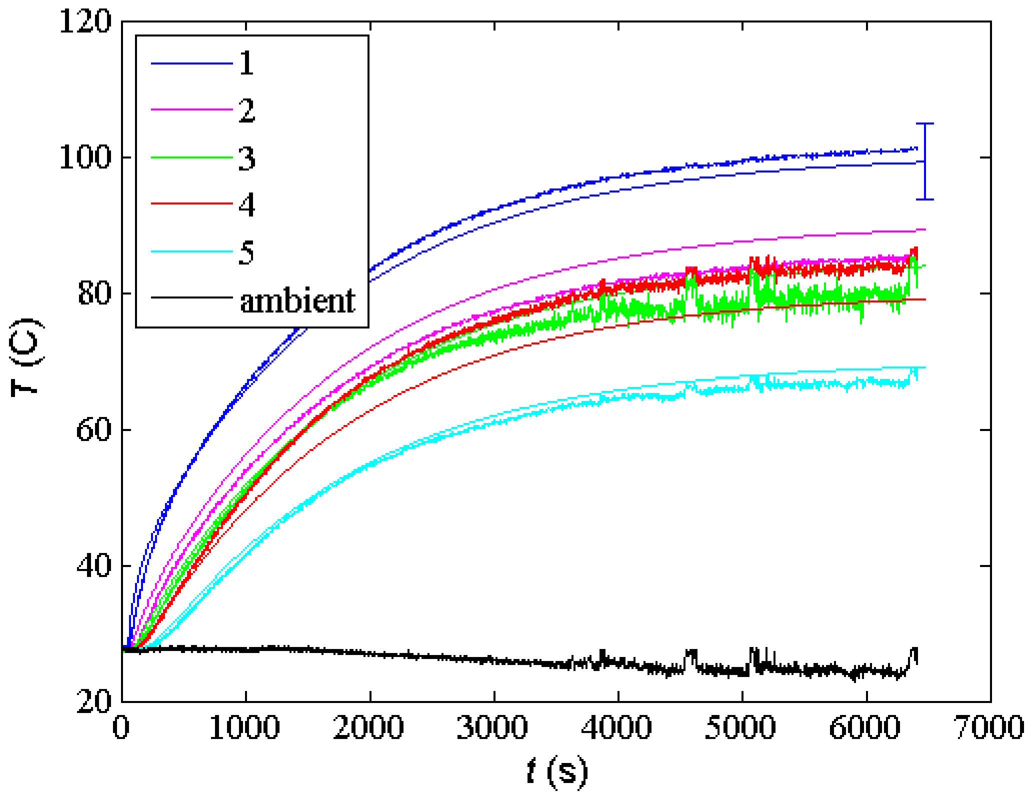
```
x
```

```
x =

  187.2870    0.0000    0.4132  360.4545    1.0000  987.0955
```

Plot results.

```
colors = 'kbmgrck';
sensorPos = [1 6 9 12 19];
factors = [1.79 1.81 1.53 1.46 2.06 2];
offsets = [4.14 2.25 0.16 3.64 -0.35 0];

if(blackRod)
    %shuffle sensors
    factors = factors([2 3 5 4 1 6]);
    offsets = offsets([2 3 5 4 1 6]);
end

if(amb1)
```

```
    factors = [2 factors(1:5)];
    offsets = [0 offsets(1:5)];
end
figure(2);
hold off
for i=1:6
    lines(i) = plot(squeeze(readings(3, i, :)), (squeeze(readings(1, i, :)) ...
        - offsets(i)) / factors(i) - offsets2(i), colors(i+1-amb1));
    hold on
    if(i < 6)
        plot((1:length(T(:, i)))+tOffset, T(:, sensorPos(i)), colors(i+1));
        if(i < 2); errorbar(length(T(:, i))+tOffset, T(length(T(:, i)), ...
                sensorPos(i)), uncertainty); end;
    end
end
hold off;
if(amb1)
    legend(lines, 'ambient', '1', '2', '3','4', '5', 'Location', 'northwest');
else
    legend(lines, '1', '2', '3','4', '5', 'ambient', 'Location', 'northwest');
end

xlabel('{\it t} (s)')
ylabel('{\it T} (C)')
set(gca, 'FontSize', 14)
set(gca, 'FontName', 'Times New Roman')
```



Published with MATLAB® R2013a

B.4 Fits

C Calibration

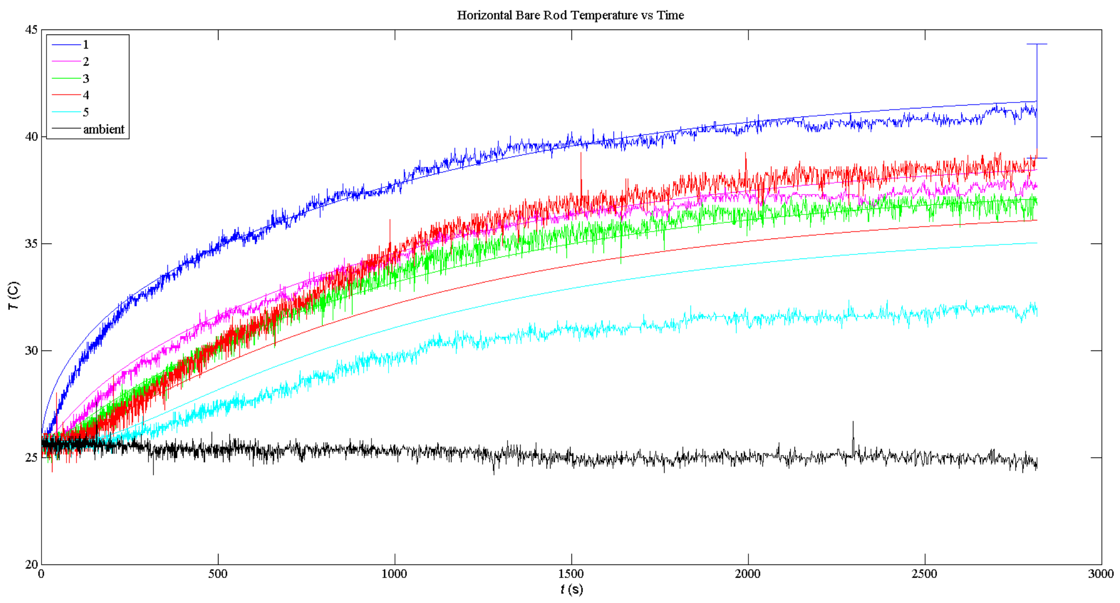


Figure 8: Bare Rod

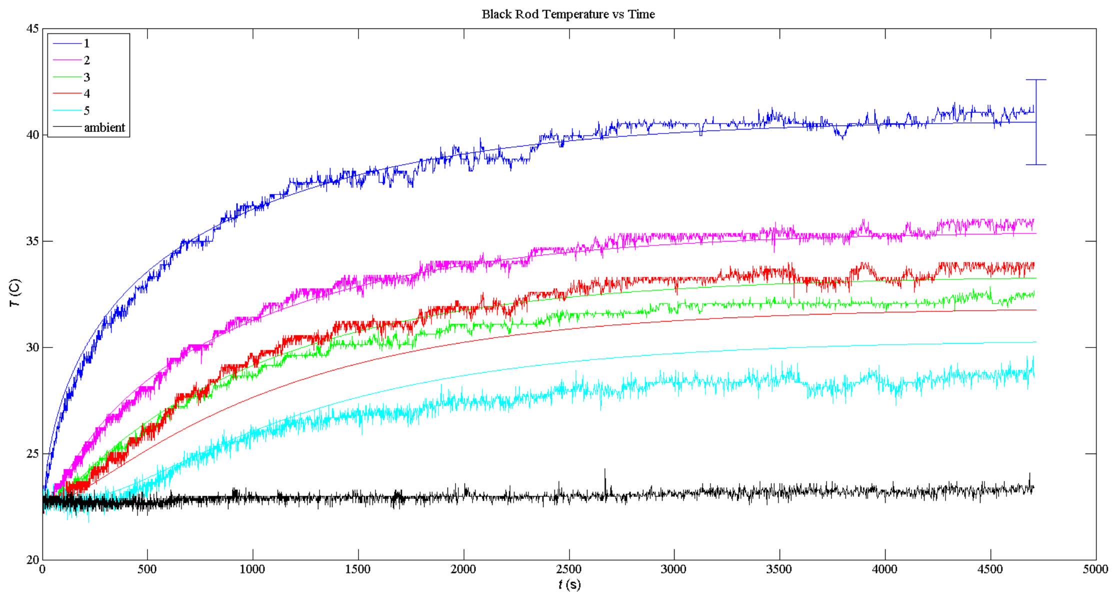


Figure 9: Black Painted Rod

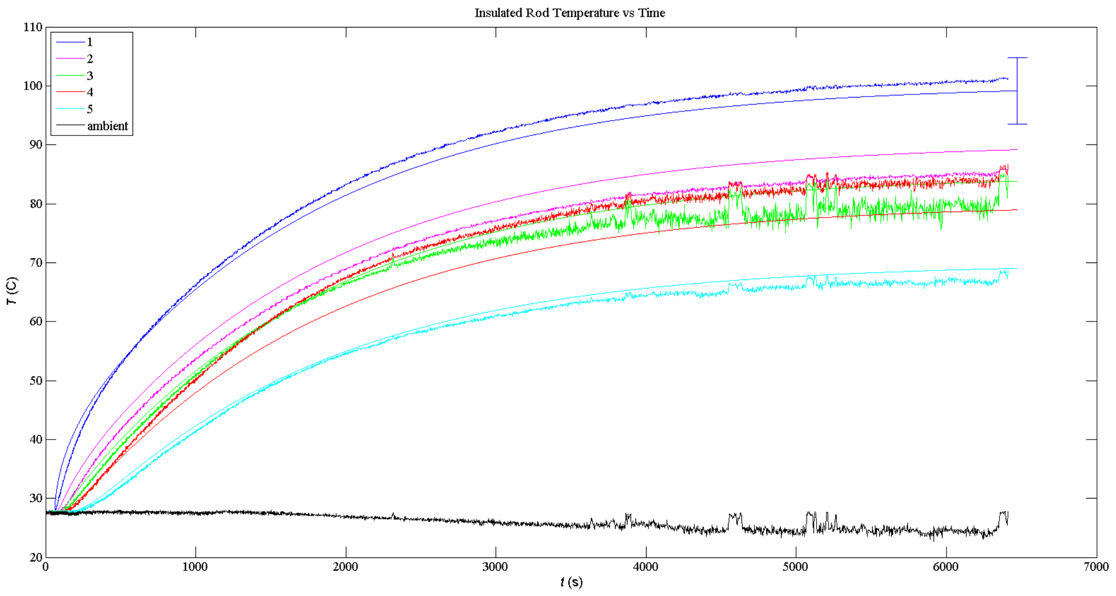


Figure 10: Insulated Rod

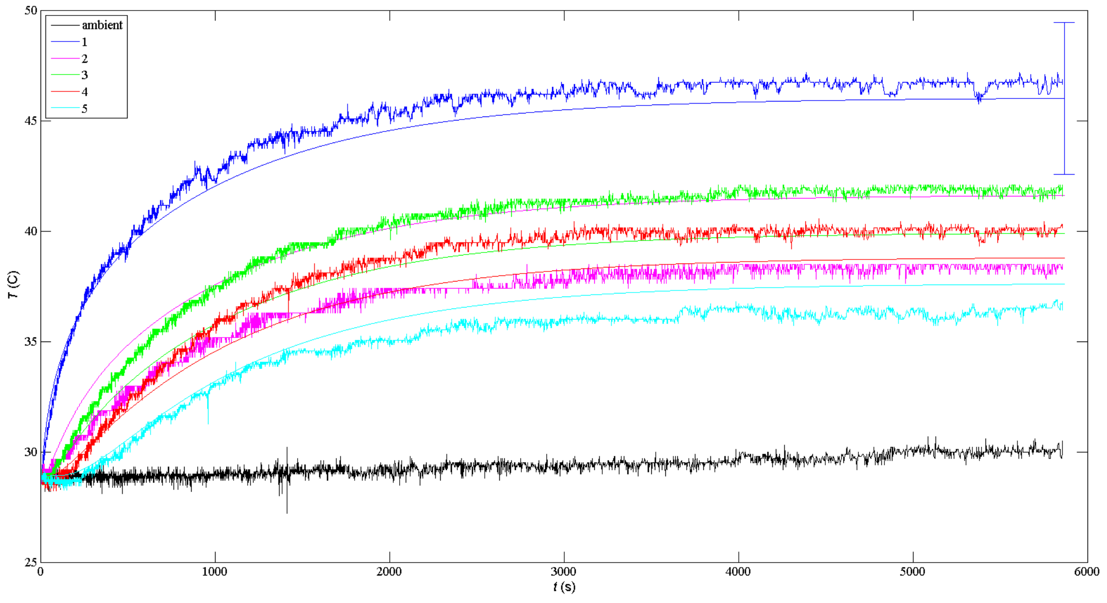


Figure 11: Moist Rod

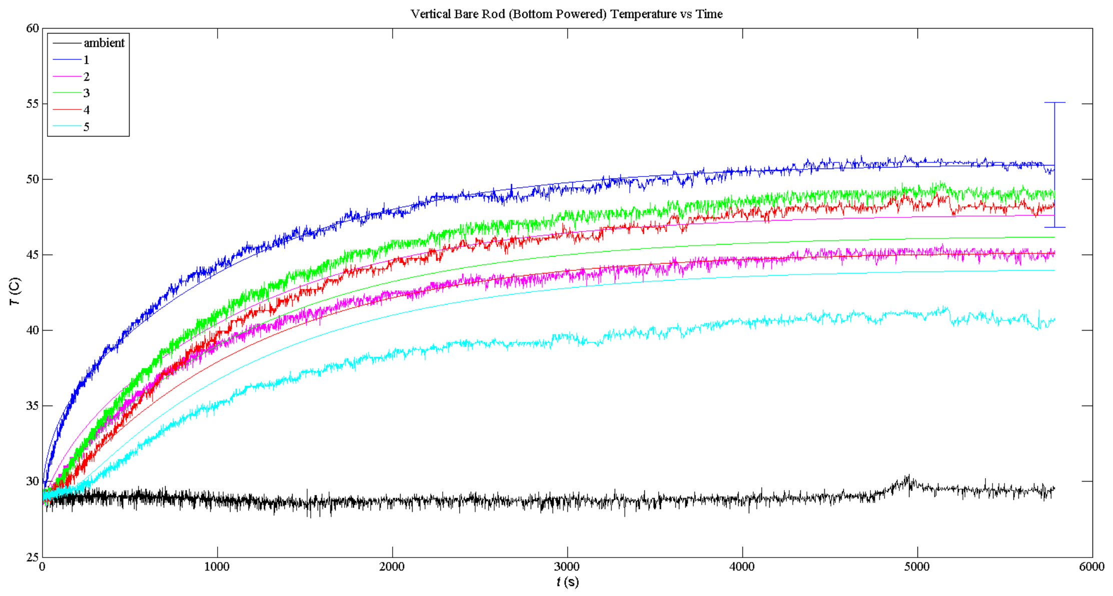


Figure 12: Vertical Bottom Heated Rod

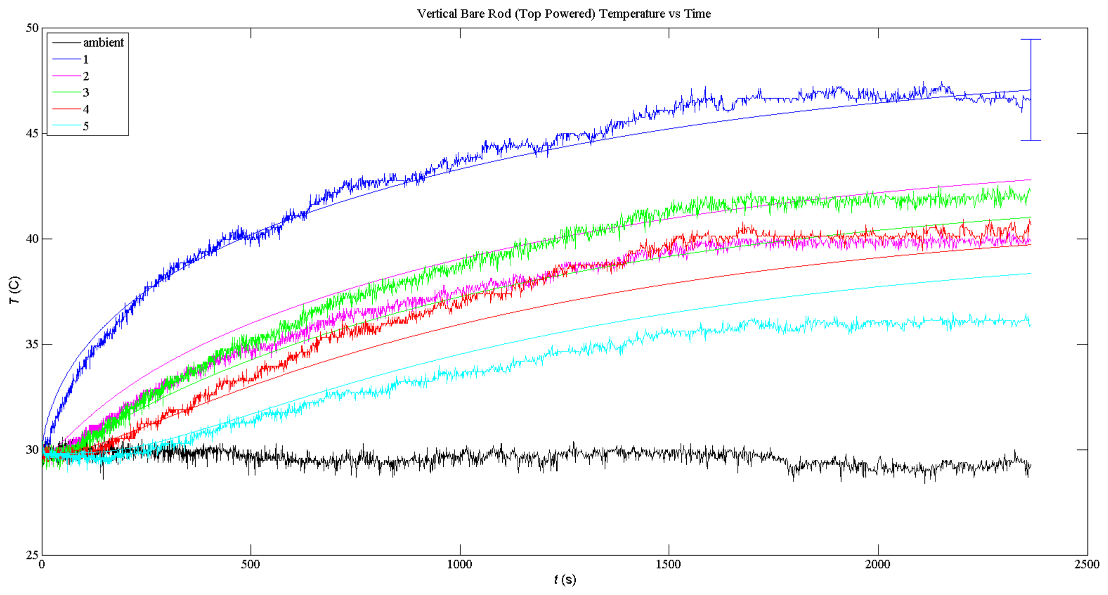


Figure 13: Vertical Top Heated Rod

Data Calibration

```
%ENPH 257 - group 13
%2015.06.27
%Takes:     filename: The file name of the data you want to calibrate
%           rangeStart: start of data range
%           rangeEnd: end of data Range
%           ambPin: the ambient Temperature pin
%Returns: The temperature of each sensor
```

```
function Tsensor = Calibrate(readings,rangeStart,rangeEnd,ambPin)

pin = [1:(ambPin-1) (ambPin+1):6];%pins in order that are NOT the ambient temp pin
Tsensor(1,:) = 0.4737*squeeze(readings(1,pin(1),rangeStart:rangeEnd)) - 1.02;

Tsensor(2,:) = 0.4525*squeeze(readings(1,pin(2),rangeStart:rangeEnd)) + .8524;

Tsensor(3,:) = 0.5492*squeeze(readings(1,pin(3),rangeStart:rangeEnd)) + 1.111;

Tsensor(4,:) = 0.5179*squeeze(readings(1,pin(4),rangeStart:rangeEnd)) + 2.053;

Tsensor(5,:) = 0.4577*squeeze(readings(1,pin(5),rangeStart:rangeEnd)) - 2.049;

Tsensor(6,:) = (500/1023)*squeeze(readings(1,ambPin,rangeStart:rangeEnd));

end
```

```
Error using Calibrate (line 13)
Not enough input arguments.
```

Published with MATLAB® R2012b

```
%%Post Calibration Offset Calculator
%Calculates an additional offset on the data based on the average
%temperature of all sensors before heating begins and all the temperature sensors
%are assumed to be at the same temperature

%Takes: transientDataFile: a string of the filename for the heating data
%       readRange: how many data points to average
%       ambPin: the number of the ambient temp in the readings
%Returns: a calibrated offset for each temp sensor
```

```
function offset = offsetCalculator(transientDataFile,readRange,ambPin)
load(transientDataFile);
calibratedData = Calibrate(readings(1,[1:(ambPin-1) (ambPin+1):6 ambPin],:),1,readRange,ambPin);%ca
librates data in readRange
offset1 = 1:6;
sensorDataC = 1:6;

for i = 1:6
    sensorDataC(i) = mean(calibratedData(i,:));%C, averages temperature at each sensor
end

averageTemp = mean(sensorDataC(:));

for i = 1:6
    offset1(i) = averageTemp - sensorDataC(i);
end

offset = offset1;
end
```

```
Error using offsetCalculator (line 13)
Not enough input arguments.
```

Published with MATLAB® R2012b