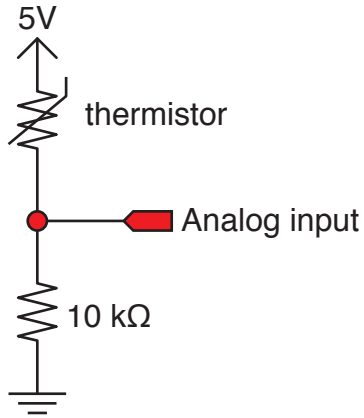
Curve Fits to Calibration Data of a Thermistor in a Voltage Divider

Portland State University
Department of Mechanical Engineering

March 3, 2019

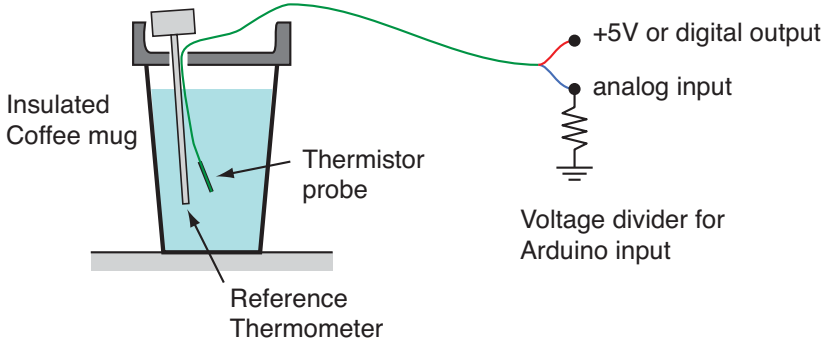
Thermistor Calibration Measurements

The thermistor is located in the upper leg of a voltage divider.



The thermistor is calibrated with a kitchen thermometer as the reference.

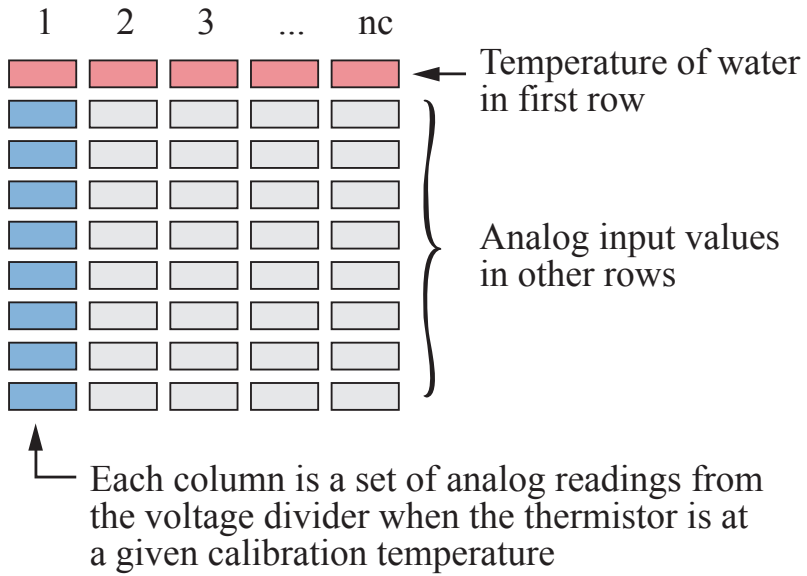
A calibration data set is obtained by filling the coffee mug with water at different temperatures.



Calibration Data Set

Data storage

- Readings of the voltage divider are averaged and displayed in the Arduino Serial Monitor.
- Displayed data is copied and pasted into columns of an Excel spreadsheet.
- The first row of the spreadsheet is the temperature for the data in its column.
- Columns under the first row are the averaged raw analog input values.



Save the Excel worksheet, and then export it (save as) a tab-delimited text file.

Reading data into MATLAB from a single file requires all columns have the same length.

Analysis of Calibration Data with MATLAB

The analysis of the calibration data involves these steps

1. Read the data into MATLAB.
2. Plot histograms of the raw readings to determine the variability of the calibration readings.
3. Compute the mean of the raw readings
4. Curve fit the raw voltage divider readings as a function of temperature
5. Swap the roles of the data to curve fit the temperature as a function of voltage divider readings.

Step 4 summarizes the calibration experiment.

Step 5 produces the calibration equation used in the temperature control algorithm.

Read the Calibration Data into MATLAB

The `load` function reads an array of plain text data into a matrix

```
D = load('thermistor_data.txt')
```

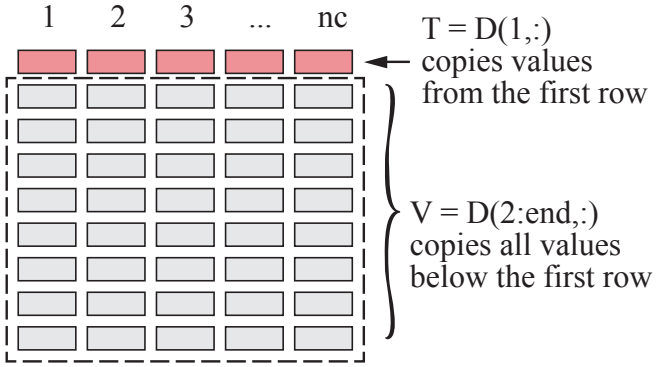
`D` is a matrix. The first row of `D` is the temperature.

```
T = D(1,:);
```

The remaining rows are voltage divider readings

```
V = D(2:end,:);
```

The first step in the analysis is to look at statistics of data in the columns of `V`.



Read the Calibration Data into MATLAB

Compute the average of each column in `V`

```
Vave = mean(V);
```

`Vave` is a row vector with `nc` elements, where `nc` is the number of columns in `V`.

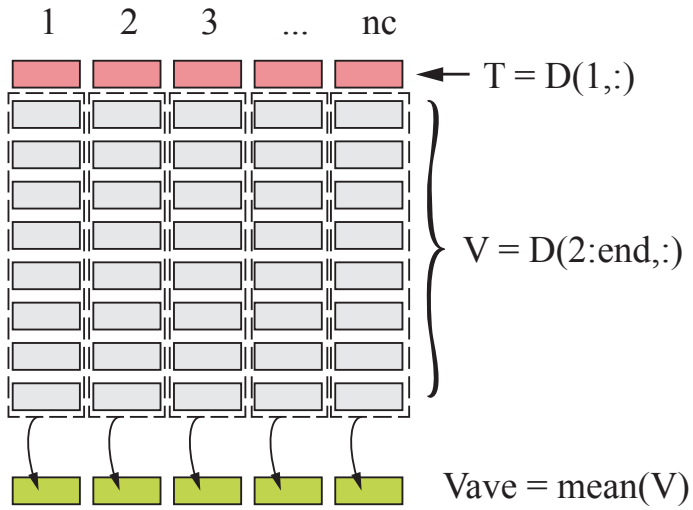
Both `T` and `Vave` have `nc` elements.

Calibration curve fits are obtained for

$$\texttt{Vave} = f(\texttt{T})$$

and

$$\texttt{T} = g(\texttt{Vave})$$



Histograms of Raw Data

Check Quality of Data at Each temperature

- Although it is easy to compute `Vave`, is the mean truly representative of the data?
- Histograms of raw data allow us to assess data quality.
- Each column is a set of readings at a given calibration temperature. Therefore, we make histograms of data in each column.

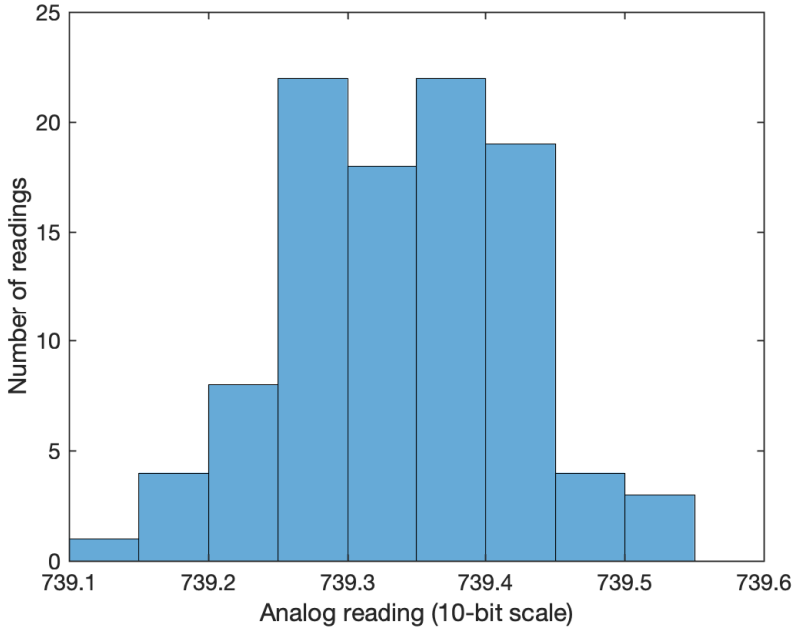
Histograms of the Calibration Data (1)

`V(:,1)` is the vector of data in the first column of `V`. The `:` is a *wildcard* meaning *all of the row indices*.

It is easy to plot a histogram of the data in the first column.

```
histogram(V(:,1))
xlabel('Analog reading (10-bit scale)')
ylabel('Number of readings')
```

There *appears* to be some spread in the data, *but look at the range*!



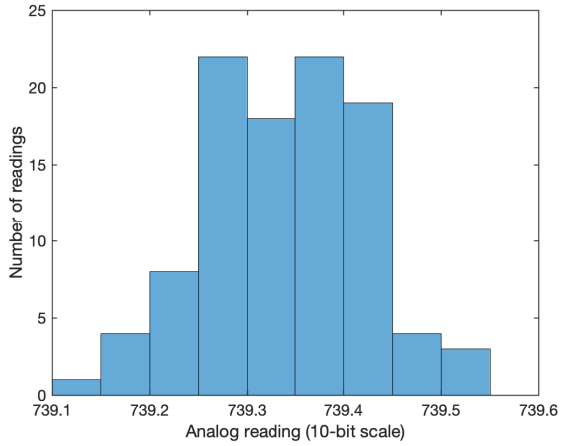
The entire set of readings at this temperature fall between 739.1 and 739.6.

> Remember that the `analogRead` function returns integers in the range $0 - 1023$. The non-integer values occur because each of the readings in the data set is an *average* of 20 readings.

Histograms of the Calibration Data (2)

There is *no significant variability* for the data in the first column.

Let's check the other columns (other temperatures).

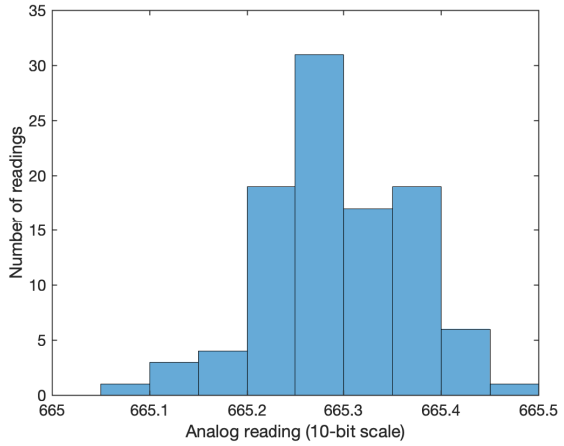


A histogram of data in the second column is created with

```
histogram(V(:,2))
```

which produces the plot to the right.

Again, the readings fall in a very narrow of analog input values.



Histograms of the Calibration Data (3)

We can automate the creation of histograms with a loop over all columns in `V`.

First, extract the number of columns for the data set with the `size` command

```
nc = size(V,2)
```

With `nc` known, loop over the columns in `V`

```
nc = size(V,2)                                  %  nc is # of columns in V
for j=1:nc                                      %  Loop over all columns
   figure                                       %  Open a new plot window
   histogram(V(:,j))                            %  Histogram of data in column j
   xlabel('Analog reading (10-bit scale)')
   ylabel('Number of readings')
end
```

The preceding code automatically adjusts to the number of columns and rows in `V`.

Histograms of the Calibration Data (4)

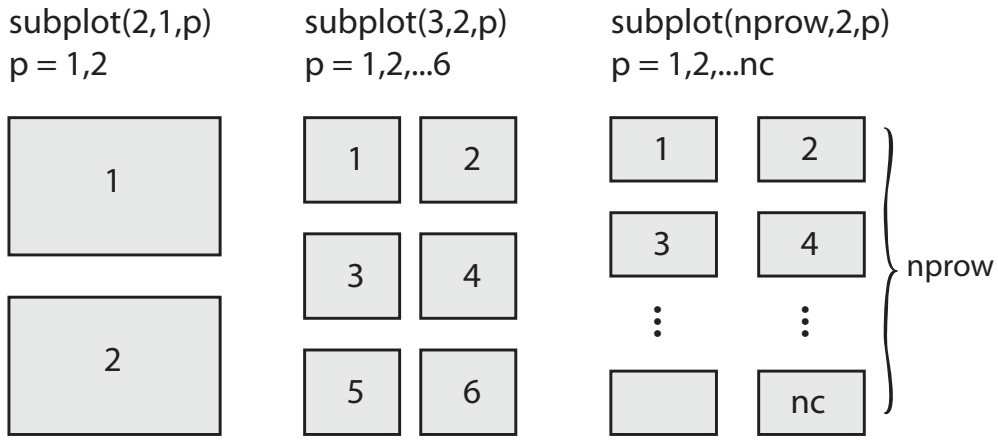
Combine the histograms in a single window with the `subplot` command to create an array of plots. The syntax is

```
subplot(nrow,ncol,p)
```

where *nrow* is the number of rows in the array, *ncol* is the number of columns in the array, and *p* is the index for a specific plot in the array.

The diagram to the right shows three possible layouts of subplots.

Note that p counts across the rows first.



Histograms of the Calibration Data (5)

The following code creates an `nprow` $\times$ 2 array of histograms from the columns of `V`.

```
nc = size(V,2);          %  number of columns
nprow = ceil(nc/2);      %  number of rows for array of subplots

for j=1:nc
  subplot(nprow,2,j);    %  Move to the next subplot
  histogram(V(:,j));     %  Bin the data and plot the histogram
  ylim( [0, 40] );       %  Set the same y-axis limits in all plots
end
```

The `thermistor_histograms` function (on the next slide) mass-produces histograms: one for each column in the data set.

Histograms of the Calibration Data (6)

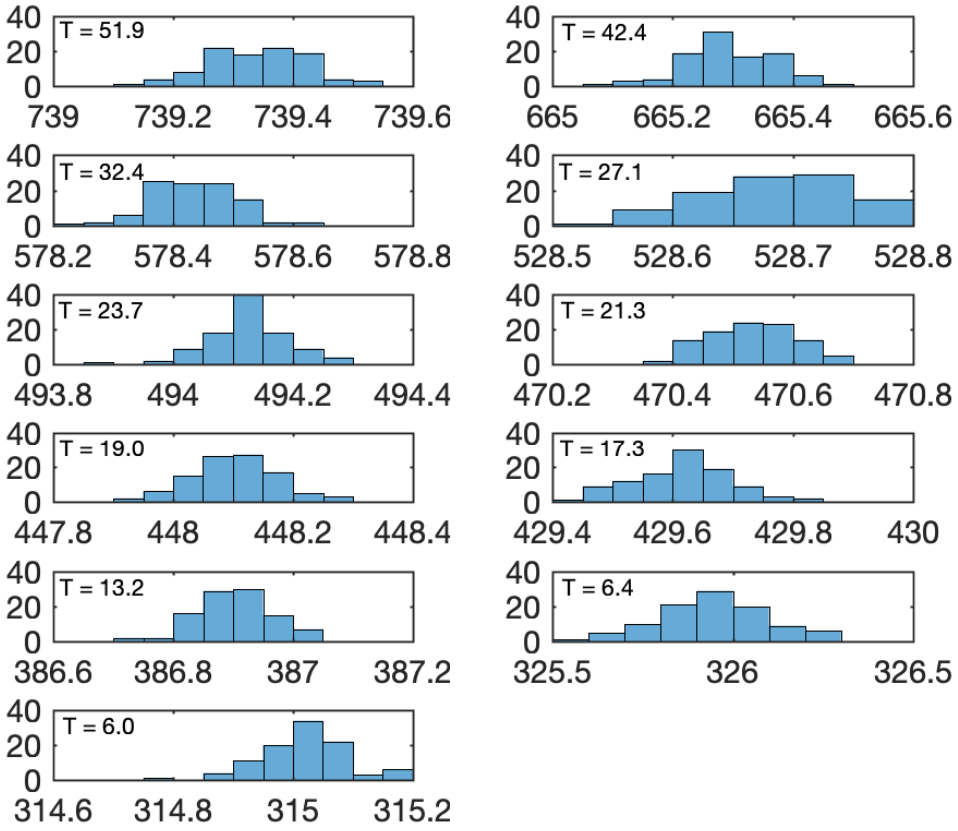
```
function thermistor_histograms(fname)
% thermistor_histograms  Plot histograms of calibration data for fish tank thermistor

if nargin<1, fname='thermistor_data.txt';  end   %  Default name of data file

D = load(fname);    %  Load data from text file
T = D(1,:);         %  Temperature values in first row
V = D(2:end,:);     %  Voltage data in columns from second row until end
Vave = mean(V);     %  Mean of data in each column
Vmed = median(V);   %  Median of data in each column
Vstd = std(V);      %  Standard deviation of each column
nc = size(V,2);     %  Number of columns in V
nprow = ceil(nc/2); %  Number of rows in the array of histograms

fprintf('\nCalibration data for %d readings at each T\n',size(V,1));
fprintf('\n    T(C)   mean(V) median(V) std_dev(V)\n');
for j=1:nc
  subplot(nprow,2,j);         %  Move to the next subplot
  histogram(V(:,j));          %  Bin the data and plot the histogram
  ylim( [0, 40] );            %  Set the same y-axis limits in all plots
  text(min(V(:,j))+0.02, 31, sprintf('T = %-6.1f',T(j)));    %  label the subplot
  fprintf('%8.1f %8.1f %8.1f %8.2f\n',T(j),Vave(j),Vmed(j),Vstd(j));
end
```

Variability is negligible at all temperatures this data set



Conclusion from Histograms

- The thermistor calibration data shows little variability at each temperature.
- Since the data is well-clustered about the mean at each temperature, the mean analog reading is a reliable data source for the calibration.

Note that the thermistor calibration experiment is both easier to carry out and it results in cleaner data than the calibration of the salinity sensor.

Curve Fits to Calibration Data

Summary of Calibration Data

The `thermistor_histograms` function prints summary statistics for each column of data. Each column contains the analog readings for a given calibration temperature.

```
>> thermistor_histograms

Calibration data for 101 readings at each T

    T(C)   mean(V) median(V) std_dev(V)
    51.9    739.3    739.3     0.08
    42.4    665.3    665.3     0.08
    32.4    578.4    578.4     0.07
    27.1    528.7    528.7     0.06
    23.7    494.1    494.1     0.07
    21.3    470.5    470.5     0.07
    19.0    448.1    448.1     0.07
    17.3    429.6    429.6     0.08
    13.2    386.9    386.9     0.07
     6.4    325.9    325.9     0.15
     6.0    315.0    315.0     0.07
```

The next step is to create curve fits between `T` and `mean(V)`.

Calibration Curve Fit Equation

The calibration process suggests a curve fit of the form $V = f(T)$: the input (independent variable) is the water temperature and the output (dependent variable) is the analog reading on the Arduino.

For the calibration experiment we could use a polynomial for $V = f(T)$.

$$V = f(T) = c_1 T^n + c_2 T^{n-1} + \cdots + c_{n-1} T + c_n. \tag{1}$$

However, to control the temperature of the fish tank, we will need $T = g(V)$.

$$T = g(V) = a_1 V^n + a_2 V^{n-1} + \cdots + a_{n-1} V + a_n. \tag{2}$$

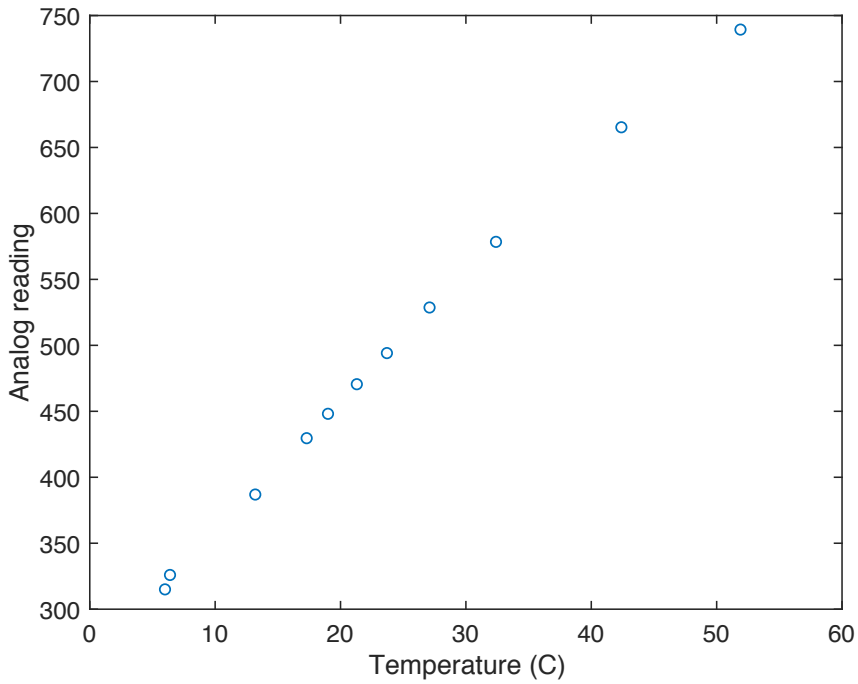
In the equations on this page, V could be the voltage output of the voltage divider. Or we could choose V to be the 10-bit value returned by the `analogRead` function during the calibration measurements.

Forward curve fit: $V = f(T)$

Polynomial Curve Fit to $V = f(T)$

Calibration data is stored in the `T` and `Vave` vectors. Plot it:

```
plot(Vave,T,'o');
```



The `polyfit` command creates a polynomial curve fit. Try a polynomial of degree 2.

$$V = c_1 T^2 + c_2 T + c_3$$

The following commands store and print the fit coefficients in vector `c`.

```
c = polyfit(T,Vave,2);
fprintf('  %14.7e',c);   fprintf('\n');
```

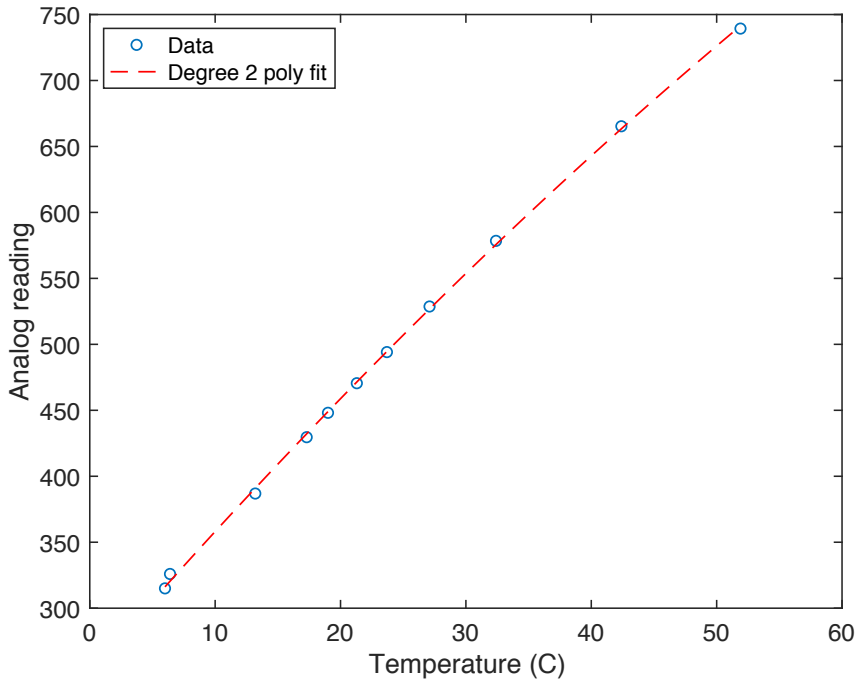
```
  -2.8959576e-02   1.0939851e+01   2.5159214e+02
```

Polynomial Curve Fit to $V = f(T)$

Given the fit coefficients in `c`, use `polyval` to evaluate and plot the fit function over the range of input `T` values.

```
Tfit = linspace(min(T),max(T));
Vfit = polyval(c,Tfit);

plot(T,Vave,'o',Tfit,Vfit,'r--');
xlabel('Temperature (C)');
ylabel('Analog output');
```

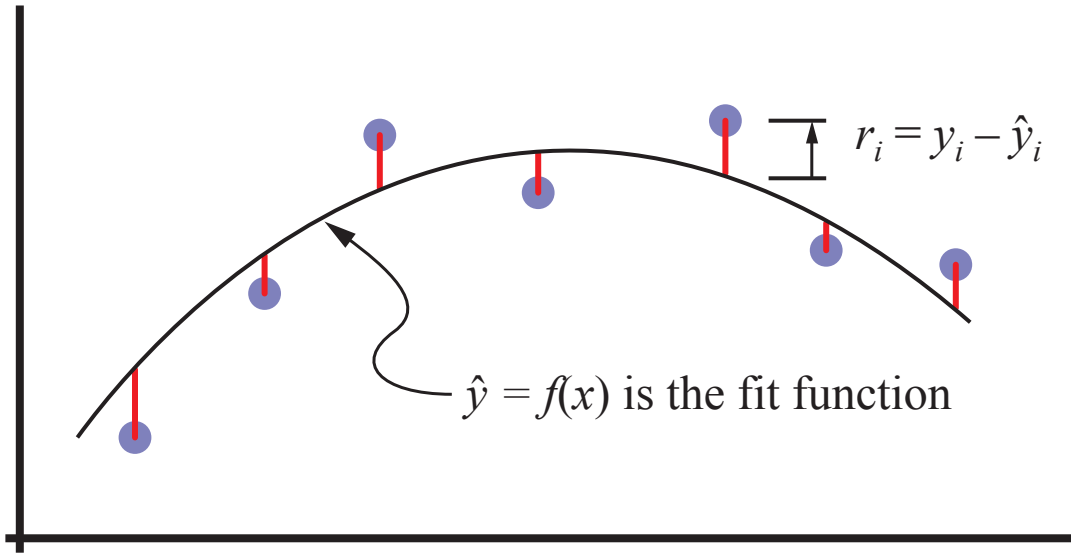


Residual of the Curve Fit

We often use R^2 as an indicator of the fit. There is also information in the residuals

$$r_i = y_i - \hat{y}_i$$

where (x_i, y_i) are the measured data and $\hat{y}_i = f(x_i)$ is the fit function evaluated at x_i.



The R^2 Statistic

R^2 is a measure of how well the fit function follows the trend in the data. $0 \leq R^2 \leq 1$.

$\hat{y}_i$ is the value of the fit function at the known (measured, input) data points, x_i.

For a line fit: $\hat{y}_i = c_1 x_i + c_2$

For a quadratic fit: $\hat{y}_i = c_1 x_i^2 + c_2 x_i + c_3$

In general: $\hat{y}_i = f(x_i)$, where $f(x)$ is the fit function

$$R^2 = \frac{\sum (\hat{y}_i - \bar{y})^2}{\sum (y_i - \bar{y})^2} = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$$

When $R^2 \approx 1$ the fit function follows the trend of the data.

When $R^2 \approx 0$ the fit is not significantly better than approximating the data by its mean.

The R^2 Statistic and the Residuals

Since

$$r_i = y_i - \hat{y}_i$$

and

$$R^2 = 1 - \frac{\sum (y_i - \hat{y})^2}{\sum (y_i - \bar{y})^2}$$

we can replace part of the numerator of the second term with

$$R^2 = 1 - \frac{\sum r_i^2}{\sum (y_i - \bar{y})^2}$$

R^2 and Residuals for the Polynomial Curve Fit to $V = f(T)$

Given the fit coefficients in `c`, use `polyval` to evaluate the residuals and R^2

```
rfit = Vave - polyval(c,T);                          %  Residual
R2 = 1 - sum(rfit.^2)/sum((Vave-mean(Vave)).^2);     %  R^2 statistic

fprintf('R^2 = %9.5f\n',R2);
fprintf('Max absolute deviation = %6.2f in V units\n',max(abs(rfit)));
```

Running the preceding code snippet produces this text output

```
R^2 =   0.99957
Max absolute deviation =   5.52 in V units
```

The quadratic curve fit produces a very good (very close to 1.0) value of R^2.

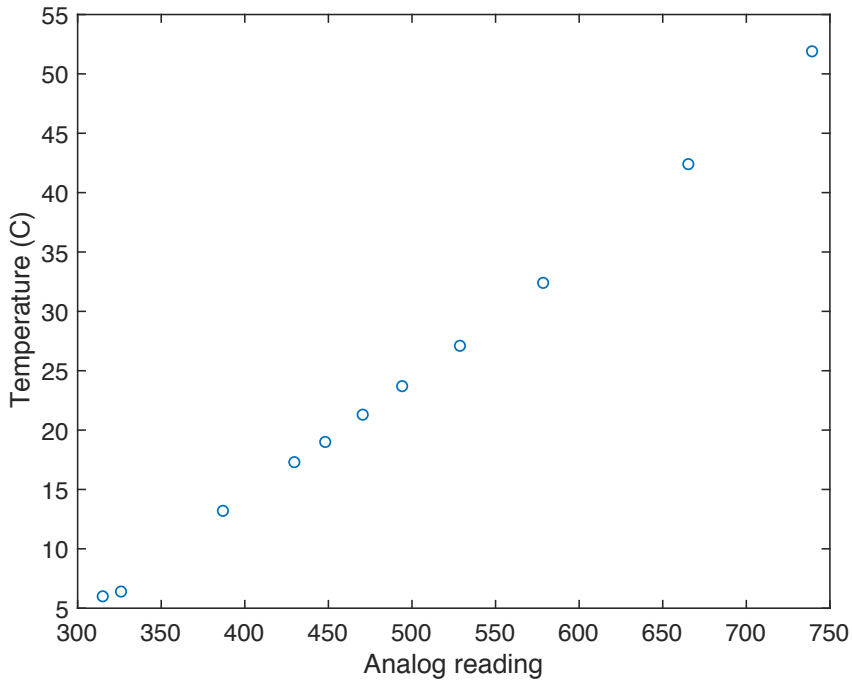
The maximum deviation of the curve fit from the given data is 5.5 units on the analog input scale, or $5.5/1023 = 5\%$ of the full scale reading.

Reverse curve fit: $T = g(V)$

Polynomial Curve Fit to $T = g(V)$

Calibration data is stored in the `T` and `Vave` vectors. Plot it:

```
plot(Vave,T,'o');
```



The `polyfit` command creates a polynomial curve fit. Try a polynomial of degree 2.

$$T = b_1 V^2 + b_2 V + b_3$$

The following commands store and print the fit coefficients in vector `b`.

```
b = polyfit(Vave,T,2);
fprintf('  %14.7e',b);   fprintf('\n');
```

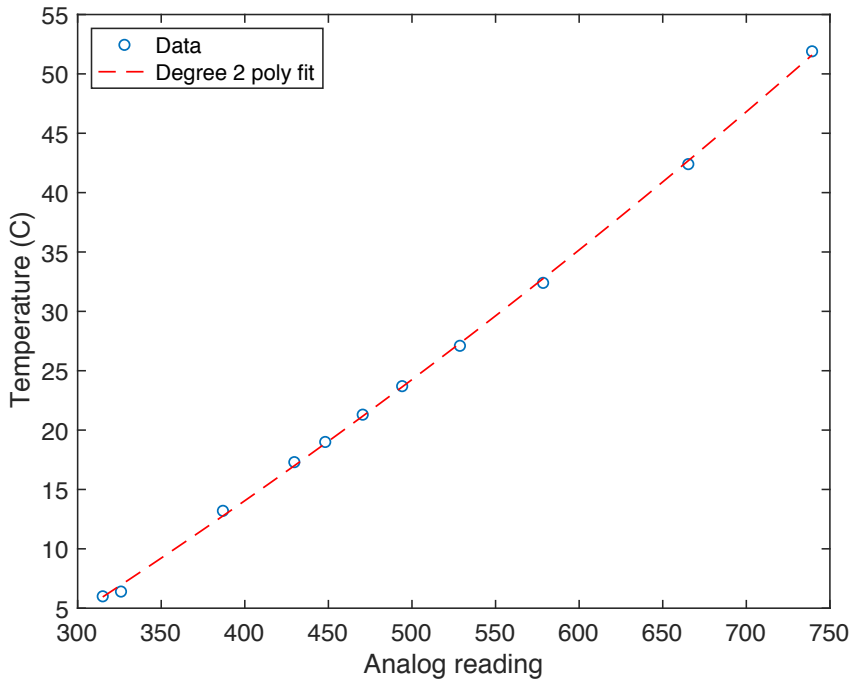
```
   1.5027635e+00   1.4261910e+01  -1.9573998e+01
```

Polynomial Curve Fit to $T = g(V)$

Given the fit coefficients in `b`, use `polyval` to evaluate and plot the fit function over the range of input `T` values.

```
Vfit = linspace(min(Vave),max(Vave));
Tfit = polyval(b,Vfit);

plot(Vave,T,'o',Vfit,Tfit,'r--');
xlabel('Analog output');
ylabel('Temperature (C)');
```



R^2 and Residuals for the Polynomial Curve Fit to $T = g(V)$

Given the fit coefficients in `b`, use `polyval` to evaluate the residuals and R^2

```
rfit = T - polyval(b,Vave);                          %  Residual
R2 = 1 - sum(rfit.^2)/sum((T-mean(T)).^2);           %  R^2 statistic

fprintf('R^2 = %9.5f\n',R2);
fprintf('Max absolute deviation = %6.2f (C))\n',max(abs(rfit)));
```

Running the preceding code snippet produces this text output

```
R^2 =   0.99951
Max absolute deviation =   0.56 (C)
```

Again, the quadratic curve fit produces a very good (very close to 1.0) value of R^2.

The maximum deviation of the curve fit from the given data is 0.6 $^\circ$C, which is acceptable for our purposes.

MATLAB Code for Curve Fits (1)

```
function thermistor_curve_fits(npoly,fname)
% thermistor_curve_fits  Polynomial curve fits of thermistor calibration
%                        data: V = f(T) and T = g(V)

% -- Supply default values for npoly and fname
if nargin<1, npoly=2;  end
if nargin<2, fname='thermistor_data.txt';  end

D = load(fname);    %  Load data from text file
T = D(1,:);         %  Temperature values in first row
V = D(2:end,:);     %  Raw analog readings in columns from second row until end
Vave = mean(V);     %  Mean of data in each column
Vmed = median(V);   %  Median of data in each column
Vstd = std(V);      %  Standard deviation of each column
nc = size(V,2);     %  Number of columns in V

% -- Print summary data
fprintf('\n    T(C)   mean(V) median(V) std_dev(V)\n');
for j=1:nc
  fprintf('%8.1f %8.1f %8.1f %8.2f\n',T(j),Vave(j),Vmed(j),Vstd(j));
end

%  Code continues in part 2 ...
```

MATLAB Code for Curve Fits (2)

```
% ... Continued from part 1

% ------------------------------------------------------------------------
% -- Perform curve fit:  V = f(T).   npoly is the degree of the polynomial
c = polyfit(T,Vave,npoly);
fprintf('\nCurve fit coefficients for V = f(T)\n');
fprintf('  %14.7e',c);   fprintf('\n');

rfit = Vave - polyval(c,T);                          %  Residual
R2 = 1 - sum(rfit.^2)/sum((Vave-mean(Vave)).^2);     %  R^2 statistic
fprintf('R^2 = %9.5f\n',R2);
fprintf('Max absolute deviation = %6.2f in V units\n',max(abs(rfit)));

% -- Evaluate and plot the curve fit
Tfit = linspace(min(T),max(T));       %  100 points in range min(T) <= T <= max(T)
vfit = polyval(c,Tfit);               %  Polynomial evaluated at those 100 points
figure('Name','Fit V = f(T)');        %  Open a new, named figure window
plot(T,Vave,'o',Tfit,vfit,'r--');
xlabel('Temperature (C)');   ylabel('Analog reading');
legend('Data',sprintf('Degree %d poly fit',npoly),'Location','Northwest');

%  Code continues in part 3 ...
```

MATLAB Code for Curve Fits (3)

```
% ... Continued from part 2

% ------------------------------------------------------------------------
% -- Perform curve fit:  T = g(V).   npoly is the degree of the polynomial
b = polyfit(Vave,T,npoly);
fprintf('\nCurve fit coefficients for T = g(V)\n');
fprintf('  %14.7e',b);   fprintf('\n');

rfit = T - polyval(b,Vave);                          %  Residual
R2 = 1 - sum(rfit.^2)/sum((T-mean(T)).^2);           %  R^2 statistic
fprintf('R^2 = %9.5f\n',R2);
fprintf('Max absolute deviation = %6.2f (C)\n',max(abs(rfit)));

% -- Plot the curve fit
vfit = linspace(min(Vave),max(Vave));   %  100 points in range min(Vave) <= v <= max(Vave)
Tfit = polyval(b,vfit);                 %  Polynomial evaluated at those 100 points
figure('Name','Fit T = g(V)');          %  Open a new, named figure window
plot(Vave,T,'o',vfit,Tfit,'r--');
xlabel('Analog reading');    ylabel('Temperature (C)');
legend('Data',sprintf('Degree %d poly fit',npoly),'Location','Northwest');

end
```

Summary

- MATLAB is used to analyze the thermistor calibration data
- Histograms of readings at a fixed temperature show little variation in data

$\Longrightarrow$ Thermistor readings are much more stable than salinity sensor readings

- The forward curve fit $V = f(T)$ characterizes the calibration experiments
- The reverse curve fit $T = g(V)$ is useful for fish tank control
- Quadratic curve fits are sufficient for $T = f(V)$ and $V = g(T)$ *for the sample data*

NOTE: Numerical values in these slides, especially the curve fit coefficients, are only for demonstration purposes. You need to analyze your data and produce curve fits based on that data alone.