BME 194: Applied Circuits Lab 1: Temperature Measurement

Kevin Karplus

December 31, 2012

1 Design Goal

The design goal for this lab is to build a temperature monitor that can record temperature measurements at frequent time intervals (say, once a second) using an Arduino microprocessor.

We want maximum sensitivity of the monitor at a specified operating temperature T_{op}, but the range should be at least 0°C to 100°C. (Note: T_{op} will not be announced until the lab period starts, so do your calculations symbolically—you'll need some numbers you won't have until halfway through the lab anyway.)

2 Background

Thermistors

A thermistor is a semiconductor device whose resistance varies with temperature. Read
`http://en.wikipedia.org/wiki/Electrical_resistance`
`http://en.wikipedia.org/wiki/Thermistor`

Thermistors are "negative thermal coefficient" devices, which means that the resistance decreases as the temperature increases (unlike most metals, which have a positive thermal coefficient of resistance, increasing resistance as temperature increases).

There are various formulas used for estimating the resistance at a particular temperature (or, equivalently, the temperature for a particular resistance). The simplest such formula, used on many specification sheets for thermistors is the "B" equation:

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} \ln\left(\frac{R}{R_{T_0}}\right) ,$$

where T is the temperature in degrees Kelvin, R is the resistance, B is a parameter that depends on the material used in the thermistor, and R_{T_0} is the resistance at some calibration temperature T_0. We can also write this as

$$R = R_{T_0} e^{B(1/T - 1/T_0)}$$

or

$$R = R_\infty e^{B/T} ,$$

where R_∞ is the projected "resistance at infinity", $R_{T_0} e^{-B/T_0}$.

For example, the thermistor we will use in this lab (NTCLE413E2103F520L) has a data sheet at `http://www.vishay.com/doc?29078` (Note: on future labs we won't be providing URLs for the data sheets—you'll be expected to look for, download, and read the data sheets without being specifically instructed to do so. We will help you try to understand the most important parts of each data sheet.)

The thermistor data sheet unpacks the part number to mean

NTC A negative temperature coefficient thermistor

L Leaded—that is, the devices has wires coming out of it (unlike some surface mount devices that are soldered directly to a printed-circuit board, for example)

E413 describes the wires and encapsulation: 30-gauge wires with the thermistor encapsulated in PVC and epoxy, rated for a maximum of 105°C. So this thermistor would not be suitable for measuring the temperature in an autoclave.

E2 Tin-alloy on the wires.

103 A nominal resistance of 10kΩ at 25°C. We'll see a lot of parts labeling where a number like 103 should be interpreted as 10.E3 or 10×10^3.

F The nominal resistance should be accurate to ±1%

520 The leads are 52mm long.

L The B-value is "low", between 3000 and 3500.

Further down in the data sheet, they give the $B_{25/85}$ value as 3435. This is the B-value that they got from a pair of measurements: the resistance at 25°C and the resistance at 85°C. They also provide a table of measured or calculated resistances (they don't say which) for the 10kΩ 1% low-B device we are using. You are not to use this table in the lab, but to make your own independent measurements—though you can check against this table to see if the measurements you are making are close to the specs.

The B equation is an ok approximation if you are keeping fairly close to the reference temperature, but using just 2 measurements to estimate B is a bad idea, particularly if you plan to use the thermistor at lower temperatures than either measurement. If you want to use the B equation, you need to fit the best values for B and R_{T_0} (or R_∞, the extrapolated resistance at infinite temperature) for a number of measurements across the temperature range you are interested in. When Prof. Karplus did this lab at home, he made 63 measurements, but that would take too long for this lab—10 measurements should be plenty, as long as they span the full range of interest and are made fairly accurately.

There is a better formula for approximating the relationship between thermistor resistance and temperature: the Steinhart-Hart equation
`http://en.wikipedia.org/wiki/Steinhart-Hart_equation`:

$$\frac{1}{T} = A + B\ln(R) + C(\ln(R))^3 \ .$$

Note that the B of this Steinhart-Hart equation is the inverse of the B in the B-equation and the spec sheet, just to be confusing. It is also much messier to invert the Steinhart-Hart equation to get R in terms of T.

Voltage dividers

A voltage divider consists of a serial connection of two resistances, as show in Figure 1. A voltage is applied across the pair of resistors and measured across one of the resistors. The formula for the measured voltage is simple:

$$V_{out} = V_{in}R_1/(R_1 + R_2) \ ,$$

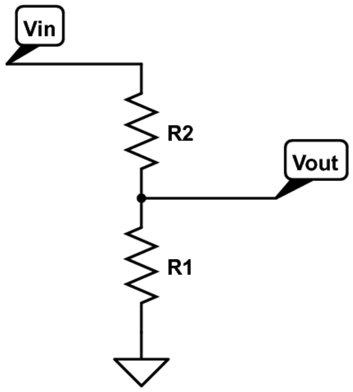

Figure 1: Voltage divider circuit. If there is no current through the output wire, then the current through the two resistances is the same, and the output voltage can be easily calculated by using ratios.

where R_1 is the resistance across which the voltage V_{out} is measured, R_2 is the other resistance, and V_{in} is the voltage across the two resistors in series.

You will use a voltage divider to convert the temperature-varying resistance of the thermistor to a temperature-varying voltage that can be measured by a voltmeter and by the Arduino.

If you want higher temperature to produce a higher voltage, which of R_1 and R_2 is the thermistor and which a fixed resistor?

Arduino Analog-to-Digital Converter

The Arduino microprocessor contains an analog-to-digital converter (ADC) with a 10-bit resolution (that is, it produces values from 0 to 1023) which represent voltages from one of the 6 analog input pins A0 through A6. It is not possible to measure voltages simultaneously on different inputs, but the microprocessor can switch the ADC rapidly from one input to another, so that measurements can be made at almost the same time.

The ADC is “ratiometric” which means that the output value is $1024 V_A / V_{Aref}$, where V_A is the voltage on the analog input pin, and V_{Aref} is a reference voltage being compared to. The default reference voltage in the data logger software we are using is the power-supply voltage to the Arduino board, which is nominally 5v, but could be quite different from that. It is possible to provide a reference voltage to the AREF pin of the Arduino, as long as the internal voltage reference is turned off by the software and the provided voltage is between 0.5v and the power-supply voltage.

For this lab, it will suffice to use the Arduino power supply as the input voltage to the voltage divider, so that the ADC converter will be reporting $1024 V_{out}/V_{in}$ for the voltage divider.

We have provided data logger software that can be run on the Arduino connected with a USB cable to a larger computer. On the computers in the lab, this software and documentation should be installed in directory C:\ProgramFiles\DataLogger\

You can get a copy to run on your own computer at
http://bitbucket.org/abe_k/arduino-data-logger/get/default.tar.bz2
(extensions .gz and .zip are also available, if your machine does not understand bzip format). Note: this is the latest version and may be a beta release. You can find earlier release versions at
https://bitbucket.org/abe_k/arduino-data-logger/downloads
under the “Tags” tab.

3 Pre-lab assignment

This lab consists mainly of measurements, but you do have one design task: to pick an appropriate resistor value for voltage divider. The goal is to have the maximum sensitivity of the output at particular temperature, which is a simple optimization problem. Before coming to the lab, you should go through the calculus needed to solve this problem for an arbitrary operating-point temperature T_{op}.

Because the resistors you have available come in only certain sizes with ratios of about 1.2 between sizes, you do not need to calculate the resistance very precisely—you just need an estimate close enough to pick the nearest standard resistance value. Because we don't need a precise calculation, you can use the simple model:

$$R = R_\infty e^{B/T}$$

for parameters R_∞ and B to be determined from your measurements.

Since the Arduino will be set up to measure V_{out}/V_{in}, figure out what that voltage ratio is as a function of temperature. The sensitivity is its derivative with respect to temperature (that is, how much does the voltage ratio change as the temperature changes). To maximize sensitivity, we have to set the derivative of sensitivity (that is, the second derivative of the voltage ratio) to zero. You should probably do one more derivative to ensure that this is maximum sensitivity, not minimum sensitivity.

Work through the calculus to figure out how to choose a fixed resistor value to give you the maximum sensitivity for an arbitrary operating temperature T_{op}. You should have the derivation and the result in your lab notebook before the lab starts, as there won't be time during lab to do the derivation.

Read the Agilent 34401A User's Guide to figure out how to use the meter to make resistance and voltage readings:
`http://cp.literature.agilent.com/litweb/pdf/34401-90004.pdf`

Read the gnuplot documentation and the gnuplot script so that you can modify the script to fit parameters for your measurements:
`http://users.soe.ucsc.edu/~karplus/bme194/w13/steinhart-hart.gnuplot`

Read the documentation on downloading programs to your Arduino board:
`http://arduino.cc/en/Guide/HomePage`
`http://arduino.cc/en/Guide/Environment`

4 Parts, tools, and equipment needed

For the first lab, we will be selling the parts kit and tool kit needed for the labs for the rest of the quarter. Please come with a checkbook or enough cash to pay for the kit(s).

Parts for this lab from kit:

- NTCLE413E2103F520L thermistor
- 4 alligator clips
- resistors
- wire (available in lab)
- breadboard (optional)

Parts students need to provide on their own:

- Arduino board
- USB cable for Arduino board
- flash drive to take results home

Tools for this lab:

- wire cutters
- wire strippers
- screwdriver (for alligator clips)
- thermometer

Equipment in lab:

- small cups for water baths
- secondary containment tubs to prevent spills
- thermoses with hot and cold water
- ice bucket
- hot pot to boil water
- power supply (for providing voltage to thermistor)
- multimeter (for measuring resistance and voltage)
- computer (with Arduino environment, data logger code, and gnuplot)

5 Procedures

Setting up

To connect up to the thermistor, it will be handy to have clip leads that can attach to the rather small and fragile wires on the thermistor. Cut a couple of pieces of wire about 2 feet (60cm) long. Strip the insulation off the last 5–6mm of each end, and tighten the screw on the alligator clip to hold the wire.

The thermistor has two leads that are very close together, which can be handy if you are soldering the thermistor to a printed circuit board, but which is inconvenient for this lab. Separate the two leads about 1cm by pulling the wires apart. It may be necessary to make a small cut between the two wires at the end, to start the split. Although a pocket knife or razor is good for this initial cut, it should be doable with the diagonal cutters in the toolkit.

We'll provide water at 3 temperatures (boiling, ice water, and roughly room temperature), which you can mix to get other temperatures. You'll make your water bath in disposable coffee cups (we don't have a sink here for washing glassware). Because water and electronics don't mix well, you must keep the coffee cup inside the secondary containment tub at all times.

In order to calibrate your thermistor, you will need to measure its temperature and resistance simultaneously. To measure the temperature, we'll use a liquid-filled student thermometer calibrated in °C. Fastening the thermometer and the thermistor together so that the bulbs are in contact will minimize the temperature difference between them.

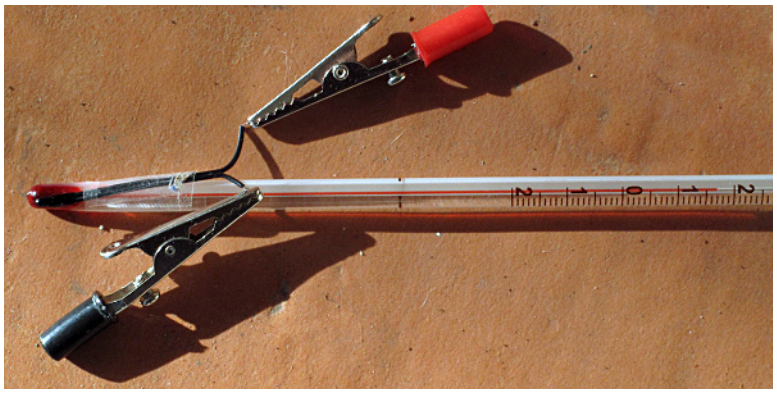

Figure 2: Thermistor with leads split for about 1cm, taped to a thermometer so that it is in contact with the bulb. The alligator clips are not attached to wires in this picture—that wiring should be done before the clips are attached to the thermistor.

Measuring resistance

You will need to use the bench multimeter to measure the resistance of the thermistor. Make sure that the probes are plugged into the high and low jacks labeled for voltage and resistance, not current. By convention, black probes are used for the "ground" and red probes for measured voltage. Although this polarity does not matter for measuring the thermistor resistance, it is good to get in the habit of connecting the probes up correctly.

You will need to select the resistance measurement (the button labeled Ω) and automatic continuous measurement. If the "MAN" for manual or "TRIG" for trigger lights are on on the display, try fiddling with the buttons (shift-trigger may help). Better would be to read the Agilent 34401A User's Guide
`http://cp.literature.agilent.com/litweb/pdf/34401-90004.pdf`

Depending on what sort of probe is connected to the multimeter, you may be able to connect them directly to the thermistor wires, or you may need to use alligator clip leads to connect them.

For making precise measurements, it helps to subtract off the measurement of the wires connecting up to the device being measured, by shorting them together and recording that measurement.

There is another technique that does an even better job of compensating for wiring resistance: using 4-wire measurement and Kelvin clips. For an explanation, see
`http://www.allaboutcircuits.com/vol_1/chpt_8/9.html`
Although the Agilent 34401A is capable of 4-wire measurement, we do not have the Kelvin clip probes for simultaneously measuring current and voltage, so this technique is not available to us. In any case, the resistances we're looking at on the thermistors are large enough compared to the wire resistance that we don't really need the extra accuracy obtainable with 4-wire measurement.

You will want to measure the temperature and resistance of the thermistor at the same time (since the uninsulated water baths will equilibrate to room temperature fairly quickly). Having one person hold and read the temperature, while the other records the temperature and resistance makes the recording easier. You can either record the measurements in a lab notebook, or type them directly into a computer file.

The computer file should start with a number of comment lines, giving the names of the people making the measurements, the date, the part being measured, and the headings for columns (the DataLogger code prompts you for this metadata, except for meanings of the column headings, which you should enter in the "notes"). Here is an example of the sort of heading I have in mind:

```
# Data collected Sat 23 Jun 2012
# by Kevin Karplus
# calibration for Vishay BC Components NTCLE413E2103F520L thermistor
# using Fluke 8060A multimeter
# degrees_F     kohm
192     1.30
190     1.37
186     1.452
180     1.65
176     1.718
...
```

Note: your measurements will be in °C not °F, and you will be measuring with a different meter. The "#" characters are there to hide the comments from gnuplot, which you can use for plotting and for fitting parameters of models to the data.

Fitting parameters with gnuplot

Gnuplot is a handy, free program for plotting data and fitting models. There are many other programs available with similar capabilities (some free, some cheap, some expensive), but gnuplot has been around for a long time and is likely to continue to be available for a long time to come.

Gnuplot can be used interactively to explore data, but can also be used with a script file that does everything consistently. I find it more useful, usually, to use a script file from the beginning, and edit it to make changes, rather than trying to remember all the commands I have typed in.

Because this is the first time most of you have used gnuplot, we'll provide a script file that you can edit to make minor changes (scaling of axes and where the data file is, for example):
`http://users.soe.ucsc.edu/~karplus/bme194/w13/steinhart-hart.gnuplot`
Figure 3 shows a typical output.

The provided script plots and fits the data with temperature as a function of resistance, because the Steinhart-Hart equation is simpler in that form than as resistance as a function of temperature (as one might wish to plot it for a physics experiment, as temperature is clearly the independent variable). The T vs. R plot is also the more useful plot for a calibration curve for using the thermistor later, where we want to change a measured resistance into the appropriate temperature value.

In future assignments, you'll have to do more of the scripting from scratch, so learn what each of the commands in this example file do. Gnuplot has a help command that you can use interactively to look at the documentation, and there is on-line documentation at
`http://www.gnuplot.info/documentation.html`

Measuring voltage

Choose an appropriate series resistor for the operating point T_{op} specified during the lab. Be sure to record T_{op}, the resistance chosen, and the schematic diagram for your circuit in your lab notebook.

You can hook up your series resistor and thermistor either using the alligator clip leads or using the breadboard. For a single connection like this, the breadboard does not offer any advantage, but for future, more complicated circuits you will find that the breadboard simplifies quick wiring and is more reliable than clip leads (though much less reliable than soldered connections).

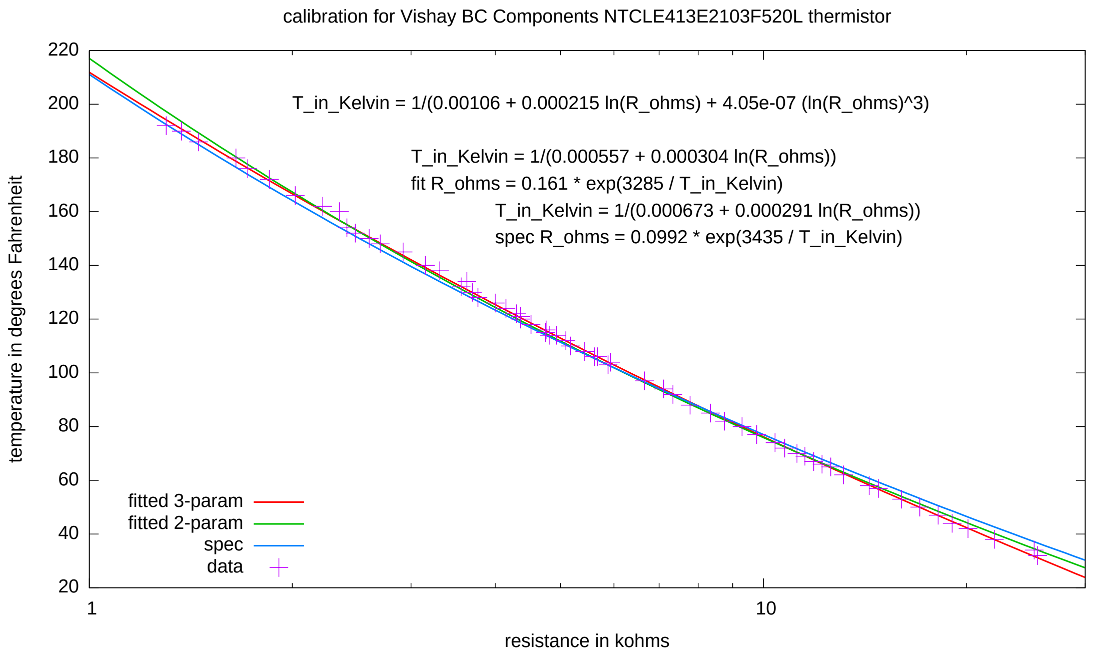

Figure 3: Plot produced from measurements with a Fahrenheit thermometer. Your plot should use Celsius measurements.

The adjustable 6v outputs on the Agilent E3631A power supplies will provide your input voltage divider. Before hooking up your resistor and thermistor, figure out how to set the voltage to 5v and measure it with the multimeter. Note that the "output on/off" button can be used to turn the power on and off to the circuit, without having to power down the Agilent E3631A itself. These power supplies are high-quality ones and should provide very accurate voltage sources. Once you've confirmed that the output voltage is correctly set on the binding posts that you expect, you can rely on the power supply's reporting of the voltage.

The color coding of the power supply terminals, with black for ground and red for the positive voltage, is standard (on the $\pm$25v supply they also use red for the negative supply, with black reserved for the common ground). Please use red wires for your positive power-supply and black wires for your ground throughout this course (and don't use those colors for any other signals). Following this convention will make your circuits much easier to check and debug.

Measure the temperature and voltage for a few different temperatures, both close to and far from T_{op}.

Recording Arduino measurements

We are providing some data logging software to use with the Arduino. This program was written by a high-school student, and he welcomes feedback (but be gentle, as it it his first project with real users). The program and documentation are available from the class web site:
`http://users.soe.ucsc.edu/~karplus/bme194/w13/DataLogger`
or directly from the BitBucket repository:
http://bitbucket.org/abe_k/arduino-data-logger/get/default.tar.bz2
(extensions tar.gz and .zip are also available, if your machine does not understand bzip format).

The software consists of two parts: a small program that runs on the Arduino and a Python program that runs on a laptop or desktop machine. The Arduino code makes measurements and communicates them over the USB connection to the Python program on host computer, which provides a graphical user interface, file I/O, and configuration information for the Arduino program.

To run the data logger, connect the Arduino to the computer with a USB cable, and open the arduino_data_logger.ino file in the Arduino environment. On the lab computers, this program should be in
`C:\ProgramFiles\DataLogger\arduino_data_logger\`
Download arduino_data_logger.ino to the Arduino. Once the program is downloaded, the Arduino will retain it in flash memory, even if power is removed. The Arduino environment can be closed after the download is done, as it is not needed again.

The python program is started with `python C:\ProgramFiles\DataLogger\datalogger\` (or whatever directory the data logger is installed in). Python looks for `__main__.py` in the directory.

The first thing to do is to configure what information you want recorded and when. For example, you may want to record the analog input A0 every 200 msec, and you may want to scale it so that 1.0 is full-scale (rather than the default, which is an integer from 0 to 1023, with 1024 representing the full-scale voltage).

You also have to decide what reference you will use for the analog-to-digital conversion. Since the A-to-D is ratiometric, and we are interested in measuring V_{out}/V_{in}, it is probably easiest to use the default reference (the 5V supply on the Arduino board) and hook that voltage up to V_{in} of your voltage divider.

If you want to use an external power supply as your V_{in}, you should probably connect it to the AREF pin and choose the external reference (but be careful, as hooking a power supply to AREF and using an internal reference can damage the Arduino). Quoting from
`http://arduino.cc/en/Reference/AnalogReference`

> Warning: Don't use anything less than 0V or more than 5V for external reference voltage on the AREF pin! If you're using an external reference on the AREF pin, you must set the analog reference to EXTERNAL before calling analogRead(). Otherwise, you will short together the active reference voltage (internally generated) and the AREF pin, possibly damaging the microcontroller on your Arduino board.
>
> Alternatively, you can connect the external reference voltage to the AREF pin through a 5K resistor, allowing you to switch between external and internal reference voltages. Note that the resistor will alter the voltage that gets used as the reference because there is an internal 32K resistor on the AREF pin. The two act as a voltage divider, so, for example, 2.5V applied through the resistor will yield 2.5 * 32 / (32 + 5) = 2.2V at the AREF pin.

Although the Arduino and ATMega328 documentation don't mention it, it appears that 0.5v is the lowest reference voltage that can be used. If the AREF voltage is too low, the chip will measure 1023 for all input voltages.

Once you have the configuration chosen and the wires for GND, V_{in}, and V_{out} hooked up, you are ready to start recording!

Record a couple minutes worth of data (say of hot water cooling down towards room temperature), and save it in a file. You should be able to plot the file easily with gnuplot, though getting proper scaling and labeling of the plot may take some effort after the lab time is over.

6 Demo and writeup

In the lab, the students need to show one of the instructors that they can correctly measure resistance and temperature, fit parameters of a model to the data, select an appropriate resistor for a given T_{op}, correctly measure voltage from a voltage divider, and record a time series of measurements with the Arduino.

A careful writeup of the lab is due the Monday after the lab.

Because of space limitations in the lab and to improve learning in the lab, all labs will be done by pairs of students (unless there are an odd number of people in class, in which case we will have a singleton, not a triple). For each lab the pairing will be different, so that no one has an unfair advantage or disadvantage from consistently being paired with a more or less competent partner.

For each lab, the partners have to choose whether to turn in a joint report with both names on it as co-authors, or separate reports with one author each, but explicitly acknowledging in writing the work done by the other partner. Both partners should keep their own lab notebooks, as they may not have access to their partner's lab notebook later in the quarter.

Note: the instructors are **not** the audience for the lab report. The report should be written to a bioengineer who has not read the assignment. It should start with a brief explanation of the problem to be solved, then explain how the problem was solved—what measurements were taken, what calculations were done, what design decisions were made, and what the final result was.

The lab report should contain at least the following (though not necessarily in this order):

- The name and number of the lab (for example, "Lab 1: thermistors").
- The authors of the report.
- The date.
- A statement of the problem.
- Any pre-lab analysis and design that was done.
- A plot of measurements made and any modeling done (for example, as the provided gnuplot script does for the temperature vs. resistance calibration curve). The plot should use the units measured (Ω or kΩ and °C, for example).
- A schematic for the voltage divider. If you want a free tool to help draw schematics, you can try the web-based CircuitLab: `http://www.circuitlab.com/editor/`
- A calibration plot of temperature vs. voltage for the voltage divider, showing both the data and the predicted voltage based on the selected resistance and the Steinhart-Hart parameters fit to the thermistor resistance measurements. Note that this plot will require you to do some scripting in gnuplot or a similar plotting tool.
- A plot of temperature vs. time from the Arduino output recording. Again, this plot will require some scripting.

To produce gnuplot outputs that can be incorporated into documents, I use a Makefile on a Linux or Mac OS X system that adds a line to the beginning of the file:

```
%.eps: %.gnuplot
        echo "set terminal postscript eps color" \
        | cat - $^ \
        | gnuplot \
        >$@

%.pdf: %.gnuplot
        echo "set terminal pdf color" \
        | cat - $^ \
        | gnuplot \
        >$@
```

On the School of Engineering Linux machines, I can use the "convert" program to convert to raster formats like png, should I need them (say for creating a web page). On the Macs, the command
`qlmanage -t -s 1000 -o . foo.pdf`
will create a foo.pdf.png image that is 1000 pixels wide from a pdf image.

7 Design Hints

Section 2 and Section 3 should have set you up with all the design hints you need for this assignment.