Two Dimensional Heat Flow Apparatus

Patrick McDougall* and Eric Ayars

California State University, Chico

(Dated: September 16, 2013)

Abstract

We have created an apparatus to quantitatively measure two-dimensional heat flow in a metal plate using a grid of temperature sensors read by a microcontroller. Real-time temperature data is collected from the microcontroller by a computer for analysis and comparison with a computational model of the Heat Equation. The experimental data and computational model match closely. The microcontroller-based sensor array allows previously-unavailable levels of precision at very low cost, and the combination of measurement and modeling makes for an excellent apparatus for the undergraduate Advanced Lab course.

I. INTRODUCTION

Heat flow experiments on an undergraduate level are usually limited to the one-dimensional case. The most common experimental setup is to measure the temperature at multiple points along a copper pipe with fixed temperatures at the ends. With advances in microcontroller technology, it is possible to make significant improvements in both the precision and dimensionality of this basic apparatus. For the apparatus presented here, we are measuring the temperature at one hundred discrete points located on a 10×10 grid across the surface of a 30cm-square aluminum plate, shown in Fig **??**.

Temperatures at each point are measured every two seconds, and the resulting data can be compared with computational solutions of the heat flow equation.

The apparatus can be built for less than $300. This same technique can be adapted to many other sensor configurations: odd geometries, improved one-dimensional heat flow apparatus, or any situation in which it is desirable to measure temperature at a large number of points at high speed and good precision.

II. CONSTRUCTION

The key component to this apparatus is the Dallas Semiconductor DS18B20 temperature sensor. This device comes in a standard 3-pin TO-92 package (as well as various surface-mount options) and measures temperatures from $-55°$C to $+125°$C with either 9 or 12-bit precision. Rather than reporting temperature with an analog signal, these devices communicate via the "1-Wire" protocol. Every DS18B20 made comes with a unique hard-coded address, so it is possible to place multiple sensors on a single bus and still allow the control device to retrieve temperature measurements from individual sensors.[1]

The other component of the sensor network is the microcontroller. There are many options for the microcontroller; we chose to use an Arduino.[2] The Arduino is an open-source development board which includes power management, USB communications, and a user-friendly programming environment. It is based on the ATmega328p microcontroller, which has more than sufficient memory and speed for this application. There are numerous types of Arduino available: the particular variety we used was the Arduino Pro Mini[3] but any of the numerous options would work equally well.

The base of our apparatus was cut from a piece of 6mm aluminum plate. After cutting the plate square and thoroughly cleaning the surface, we glued 100 DS18B20 sensors on a grid distributed across one face of the plate, using "Arctic Silver" thermal epoxy[4]. Strips of 18mm Kapton tape were used for alignment guides and to provide insulation between the sensor leads and the plate. We then attached the microcontroller and ran power and ground wires from the controller to all of the sensors. (The controller is powered via the USB connection to the computer.)

Each DS18B20 temperature sensor has a unique address, but there is no external indication of what that address might be. So the next construction step was to program the microcontroller to determine those addresses. This program sent out a command on the signal bus for all temperature sensors on the bus to report back with their addresses. It would then send all observed addresses to the computer via the USB connection. We then temporarily connected each sensor, one at a time, to the control bus line and ran this program once for each sensor to develop an address map of the sensor array. Once the address map was constructed and carefully double-checked, we permanently soldered the bus line to all sensors.

The final step in the construction of the apparatus was to program the microcontroller. This required two distinct steps: the first was to write the address map to the non-volatile EEPROM on the microcontroller in left-to-right, top-to-bottom order. The second step was to write the final "operating" program to the microcontroller. This program first sends out a signal on the bus for all sensors to record the temperature. It then pulls the first address from EEPROM and sends a command for the sensor at that address to report its observation. The observed temperature is sent out the USB connection to the computer, and the microcontroller pulls the next address from EEPROM and repeats the process until it's polled all sensors. When all of the points have been measured the microcontroller repeats the process by once again sending a command to record the temperature. This continues indefinitely, for as long as the apparatus is connected to a USB port.

All programs used by the microcontroller and computer are available at `physics.csuchico.edu/~eayars/code/Thermoplate.zip`.

III. OPERATION

The completed apparatus sends each "frame" of temperature data as a single line of whitespace-delimited text data. Frames are sent at approximately 2-second intervals. Serial data in this form can be read by any computer system, and there are many good choices of language for reading and displaying the data. We chose to use Python, but LabVIEW or C/C++ would be equally good choices. With our Python program, we could display the thermal data as a contour map while saving each frame for future analysis and comparison with theory.

By default, the temperature sensors are set to 9-bit precision (roughly 0.3°C) and we left the device at that precision rather than switching to 12-bit so as to keep the frame rate as high as possible. Figure **??** shows a typical frame, as rendered by our Python program.

IV. THEORETICAL SETUP

Starting with the heat equation and assuming an adiabatic system gives Eq. (1). Because the temperatures involved are near the ambient room temperature and because aluminum has a very high thermal conductivity compared to the rate of heat transfer between the aluminum and the surrounding air, there is relatively little heat transfer to the environment and the adiabatic approximation is valid.

$$c_p \rho \frac{\partial T}{\partial t} - k\nabla^2(T) = 0 \tag{1}$$

Since there is no general analytical solution to the Eq. (1), we used the method of finite differences.

$$\frac{\partial T}{\partial t} \cong \frac{T_{i,j,n+1} - T_{i,j,n}}{\Delta t} \tag{2}$$

$$\nabla^2(T) \cong \frac{T_{i+1,j,n} - 2T_{i,j,n} + T_{i-1,j,n}}{(\Delta x)^2} + \frac{T_{i,j+1,n} - 2T_{i,j,n} + T_{i,j-1,n}}{(\Delta y)^2} \tag{3}$$

where i represents the index of the x coordinate, j the index of the y coordinate, and n the time step. For this apparatus we have an equal spatial step in the x and y directions, so $\Delta x = \Delta y$. Solving for the next $(n+1)$ temperature at some point (i, j), we obtain:

$$T_{i,j,n+1} = T_{i,j,n} + \frac{k\Delta t}{\Delta x^2 c_p \rho}(T_{i+1,j,n} + T_{i,j+1,n} - 4T_{i,j,n} + T_{i-1,j,n} + T_{i,j-1,n}) \tag{4}$$

We replace the constants with a single constant: $F_{0M} \equiv \frac{k\Delta t}{c_p \rho \Delta x^2}$, commonly referred to as the Mesh Fourier Number.[5]

The geometrical and physical terms in F_{0M} are constrained by our experimental apparatus, so the value of F_{0M} used determines the time step for the simulation. This replacement and a slight rearrangement of terms in Eq. 4 gives us Eq. 5.

$$T_{i,j,n+1} = (1 - 4F_{0M})T_{i,j,n} + F_{0M}(T_{i+1,j,n} + T_{i,j+1,n} + T_{i-1,j,n} + T_{i,j-1,n}) \tag{5}$$

For stability in the computational model, the value of F_{0M} must be less than $\frac{1}{4}$.

V. BOUNDARY CONDITIONS AND INTEGRATION METHODS

Equation 5 is reminiscent of the relaxation method[6] in that each step depends in some sense on the average of the surrounding points. And like the relaxation method, there is the question of what to do with boundary points. Recalling our earlier assumption that the system is adiabatic we can set the heat flow off the boundary to be zero. This heat flow condition would be met if the material extended beyond the boundary and the temperature of those extended points was the same as the temperature at the boundary points; zero temperature difference would result in zero heat flow. So for the sake of this calculation we use "virtual points" just beyond the boundary, each at the temperature of the adjacent boundary point. I.e.

$$T_{i,j_{max}+1,n} = T_{i,j_{max},n} \tag{6}$$

The experimental setup is an evenly spaced 10×10 grid (experimental grid), where each sensor is separated by 2.9 cm. In order for the finite difference approximation to be accurate, both Δt and Δx should be small. Δt is set by choosing a Mesh Fourier Number and Δx is dependent on the spacing of the experiment. However, Δx can be made smaller by creating a fine-scale grid for the simulation. We chose a 100×100 simulation grid for the calculation, and used cubic interpolation to fill in the initial values of the simulation grid based on the values from the 10×10 experimental grid.

Once this initial simulation grid was set, we merely applied Eq. (5) repeatedly, using the "virtual points" at the boundaries, to calculate the subsequent time evolution of the system. For the simulation grid chosen, $\Delta x = 0.29$ cm and $\Delta t = 0.0002$ s. Since the time and spatial step sizes for the simulation are not the same as the time and spatial steps for the experiment, we then extracted the relevant points from the appropriate frames of the simulation for comparison with the experimental data.

VI. RESULTS AND LABORATORY USE

After the simulation is complete, its results can be compared to the experimental data. The comparison here was done by looking at the percent difference of each point and averaging the magnitude of this difference across the experimental grid. Our plot of percent difference versus time is shown in Fig. **??**, and is very reasonable considering the assumptions made in forming the computational model.

This apparatus offers a distinct improvement over more traditional heat flow experiments in the physics Advanced Lab. In addition to the more interesting geometry, compared to the usual 1-D case, use of this apparatus presents an opportunity to provide students with a challenging but tractable computational modeling problem for comparison with the experiment. The sensitivity of the DS18B20 sensors allows useful experiments to be done at temperatures relatively close to ambient temperatures, which allows use of the adiabatic approximation without extensive insulation. Finally, this technique for controlling multiple DS18B20 sensors can be used for other experiments. The ability to use a large number of temperature sensors with a single microcontroller has the potential to be of use in many other laboratory applications.

VII. ACKNOWLEDGEMENTS

The authors would like to thank Daniel Lund for his assistance in constructing this apparatus, particularly his diligence in determining the internal addresses of all 100 sensors.

---

* Electronic address: pmcdougall@mail.csuchico.edu

[1] There is no *theoretical* limit on the number of sensors that could be tied to the same bus. The available memory of the microcontroller and the cumulative current draw of all the sensors on the data line do provide a *practical* limit, however, and 100 sensors is approaching that limit.

[2] `www.arduino.cc`

[3] `www.sparkfun.com`

[4] Available at computer supply and repair stores.

[5] Y. A. Cengel and A. J. Ghajar. *Heat and Mass Transfer: Fundamentals and Applications*. McGraw-Hill, fourth edition, 2011.

[6] M. DiStasio and W. C. McHarris. Electrostatic problems? relax! *American Journal of Physics*, 47(5):440–444, May 1979.