How Quickly Does Water Cool?

Stan Wagon
Department of Mathematics and Computer Science
Macalester College
St. Paul, MN 55105

wagon@macalester.edu

Robert Portmann
325 Broadway
Boulder, CO 80305

1. Newton's Law of Cooling

Newton's Law of Cooling states that the rate of change of temperature of an object is proportional to the temperature difference between it and the surrounding medium; using T_{ambient} for the ambient temperature, the law is $dT/dt = -K(T - T_{\text{ambient}})$, where T is temperature, t is time, and K is a constant related to efficiency of heat transfer. Most mathematicians, when asked for the rule that governs the cooling of hot water to room temperature, will say that Newton's Law applies and so the decline is a simple exponential decay. Like many teachers of calculus and differential equations, the first author has gathered some data and tried to model it by this law. But it cannot be done: the data do *not* follow the simple exponential form that the law suggests. The correct model is quite a bit more complicated, and we do not give a definitive solution here, but show how one might try to model evaporation with the help of *Mathematica*'s ability to fit parameters to the numerical solution of differential equations.

2. The Best-Fit Exponential

Here is some data obtained by hand after pouring boiling water into an aluminum pot. The first coordinates are the times in seconds; the second are degrees Fahrenheit.

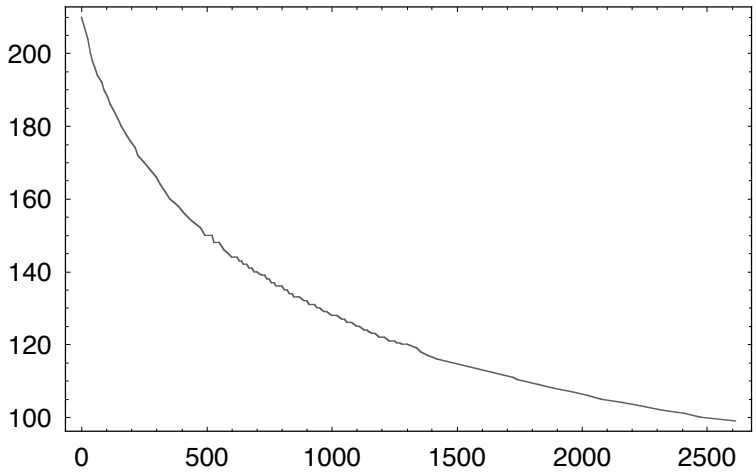
```
In[1]:= data = {{0, 210}, {25, 204}, {34, 200}, {42, 198}, {54, 196},
        {63, 194}, {80, 192}, {89, 190}, {103, 188}, {115, 186}, {131, 184},
        {145, 182}, {158, 180}, {175, 178}, {195, 176}, {213, 174},
        {225, 172}, {250, 170}, {274, 168}, {298, 166}, {315, 164},
        {335, 162}, {353, 160}, {387, 158}, {411, 156}, {440, 154},
        {475, 152}, {492, 150}, {520, 150}, {530, 148}, {550, 148},
        {560, 147}, {570, 146}, {585, 145}, {600, 144}, {610, 144},
        {620, 144}, {630, 143}, {640, 143}, {647, 142}, {660, 142},
        {670, 141}, {680, 141}, {690, 140}, {700, 140}, {710, 139.5},
        {720, 139}, {730, 139}, {740, 138}, {750, 138}, {760, 137},
        {770, 137}, {778, 136}, {790, 136}, {800, 136}, {810, 135},
        {820, 135}, {830, 134}, {840, 134}, {850, 133}, {860, 133},
        {870, 133}, {880, 132.5}, {890, 132}, {900, 132}, {910, 131},
        {920, 131}, {930, 131}, {940, 130}, {950, 130}, {960, 129.5},
        {970, 129}, {980, 129}, {990, 128.5}, {1000, 128}, {1010, 128},
        {1020, 128}, {1030, 127.5}, {1040, 127}, {1050, 127}, {1060, 126},
        {1070, 126}, {1080, 126}, {1090, 125.5}, {1100, 125}, {1110, 125},
        {1120, 124.5}, {1130, 124}, {1140, 124}, {1150, 123.5}, {1160, 123},
```

```
{1170, 123}, {1180, 122.5}, {1190, 122}, {1200, 122}, {1210, 122},
{1220, 121.5}, {1230, 121}, {1240, 121}, {1250, 121}, {1260, 120.5},
{1270, 120.5}, {1280, 120}, {1300, 120}, {1340, 119}, {1356, 118},
{1380, 117}, {1418, 116}, {1725, 111}, {1740, 110.5}, {1825, 109},
{1890, 108}, {1955, 107}, {2022, 106}, {2077, 105}, {2160, 104},
{2244, 103}, {2320, 102}, {2408, 101}, {2473, 100}, {2613, 99}};
```

We define the times and temperature, as well as the last time, the first temperature, and the ambient temperature, which was 79°.

In[2]:=
```
{time, temp} = Transpose[N[data]];
t_max = Last[time]; temp_0 = temp[[1]]; ambient = 79;
```

In[4]:=
```
ListPlot[data];
```



Though it is easily done by hand, we use **DSolve** to solve the simple differential equation in Newton's Law.

In[5]:=
```
NewtonModel[K_][t_] := Evaluate[Simplify[T[t] /. DSolve[{
        T'[t] == -K (T[t] - ambient), T[0] == temp_0}, T[t], t][[1]]
  ]];

NewtonModel[K][t]
```

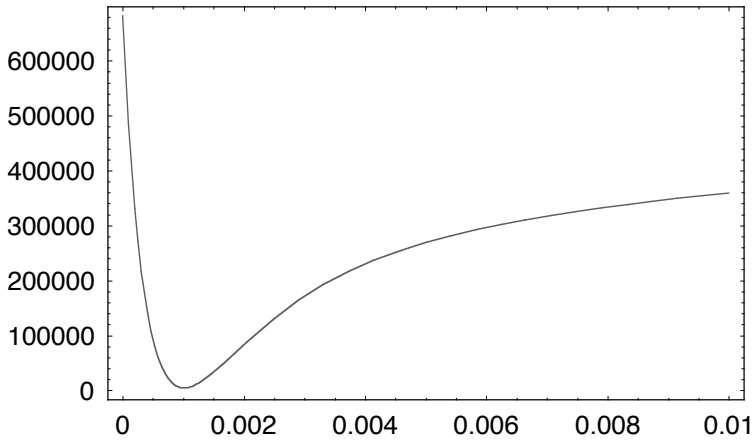
Out[6]=
$$79. + 131.\, e^{-K\,t}$$

We now set up the residual corresponding to a value of the parameter K, which is a coefficient of heat conductance.

In[7]:=
```
residual[K_] := temp - NewtonModel[K][time];
```

A quick plot shows that the sum of squares attains a clear minimum.

In[8]:=
```
Plot[residual[K] . residual[K], {K, 0, 0.01}];
```



Now we wish to find the value of K so that the model is the best sum-of-squares estimate to the data. Note that K is not the true air-water conduction coefficient, whose use would require knowing the shape of the pan and the amount of the water, but is simply the value appropriate for this particular experiment. We can use **FindMinimum**, which requires a starting value and then finds a local minimum. **FindMinimum** should recognize the objective as being a sum of squares and use the **LevenbergMarquardt** method, which is especially robust for such problems. But it does not hurt to ask for that method explicitly. A seed of 0 works in this case, but in other modeling situations one may want a positive seed somewhere in the vicinity of the answer.

```
In[9]:= FindMinimum[residual[K].residual[K],
          {K, 0}, Method → "LevenbergMarquardt"] // Timing

Out[9]= {0.12 Second, {4775.83, {K → 0.00101362}}}
```

An alternative, somewhat faster, approach is to use **FindFit**, new in *Mathematica* version 5. It works better when we give a seed for the parameter k.

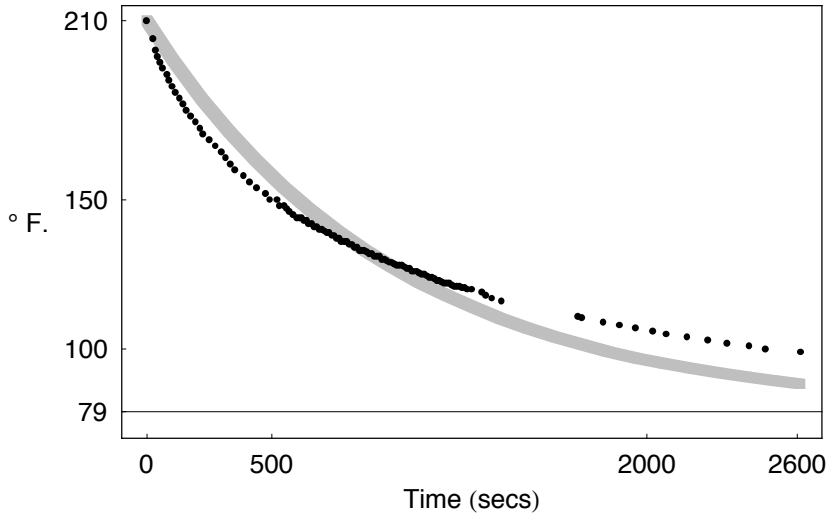
```
In[10]:= (fit = FindFit[data, NewtonModel[K][t], {{K, 0}}, t]) // Timing

Out[10]= {0.01 Second, {K → 0.00101362}}
```

We define the best-fitting model.

```
In[11]:= NewtonModelBest = NewtonModel [K][t] /. fit;
```

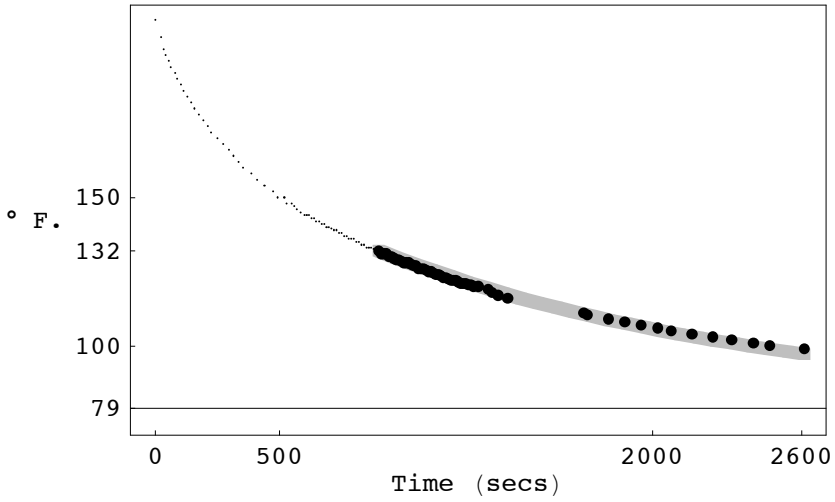
And we compare the model to the data. This "best fit" is horrible.

```
In[12]:= Plot[NewtonModelBest, {t, 0, t_max},
           PlotStyle → {Thickness[0.015], GrayLevel[0.7]},
           Epilog → {PointSize[0.01], Point /@ data},
           FrameLabel → {"Time (secs)", "° F."},
           RotateLabel → False, Frame → True,
           Axes → False, GridLines → {{}, {{ambient, {}}}},
           PlotRange → {{-100, 2700}, {70, 215}},
           FrameTicks → {{0, 500, 2000, 2600},
             {temp_0, 100, 150, ambient}, None, None}];
```



It is evident that the temperatures do not follow a rule as simple as Newton's Law. Yet it appears equally evident from the data that there is some sort of pattern in the decreasing temperatures.

Exercise. Use Newton's Law to fit various subsets of the data, such as the first 50 points, the last 50, or the middle 50 (making sure to change the initial temperature appropriately). The diagram below shows the best fit when the last 57 points are used. Such experiments show how difficult interpretation can be. The fits are generally good. Does that support the view that Newton's Law is a good model when the temperature does not vary too much and the difference between the initial and ambient temperatures is not too great? Or is the fit better only because the smaller data sets are more nearly linear?



3. An Oily Solution

It turns out that the heat loss from hot water due to evaporation is considerable; this effect is completely ignored by Newton's Law. Before going into the details of evaporation, we show what happens when a thin sheet of oil is placed over the water. This cuts out almost all of the evaporation. Macalester College student Tak Iwanaga used some physics software (LoggerPro™) to obtain data in this case. His ambient temperature was 68° and he tracked the temperature every 6 seconds for one hour. Here we useevery 10th point only since that is more than enough to get the fit.

```
dataOil = {{6., 188}, {66., 186.62}, {126., 185.},
  {186., 183.02}, {246., 181.22}, {306., 179.06},
  {366., 177.08}, {426., 174.92}, {486., 172.94},
  {546., 170.96}, {606., 168.98}, {666., 167.},
  {726., 165.20}, {786., 163.22}, {846., 161.24},
  {906., 159.44}, {966., 157.64}, {1026., 155.84},
  {1086., 154.22}, {1146., 152.42}, {1206., 150.8},
  {1266., 149.36}, {1326., 147.74}, {1386., 146.3},
  {1446., 144.68}, {1506., 143.24}, {1566., 141.8},
  {1626., 140.54}, {1686., 139.10}, {1746., 137.84},
  {1806., 136.58}, {1866., 135.32}, {1926., 134.06`},
  {1986., 132.8}, {2046., 131.54}, {2106., 130.46},
  {2166., 129.2`}, {2226., 128.12}, {2286., 127.22},
  {2346., 126.14}, {2406., 125.06}, {2466., 123.98},
  {2526., 123.08}, {2586., 122.}, {2646., 121.10},
  {2706., 120.2}, {2766., 119.30}, {2826., 118.4},
  {2886., 117.5}, {2946., 116.78}, {3006., 115.88},
  {3066., 114.98}, {3126., 114.26}, {3186., 113.54},
  {3246., 112.82}, {3306., 112.10}, {3366., 111.38},
  {3426., 110.66}, {3486., 109.94}, {3546., 109.22}};
```

In[14]:=
```
{timeOil, tempOil} = Transpose[dataOil];
```

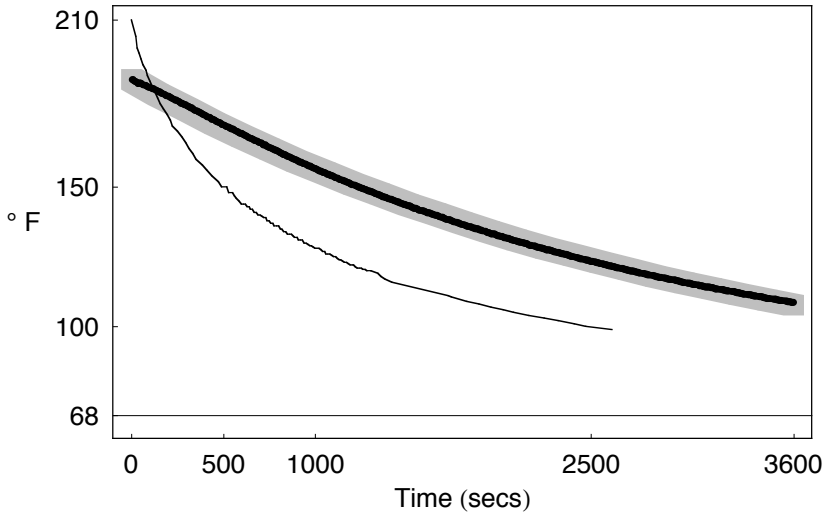
```
tempOil0 = 188.6; toilmax = 3546;
ambientOil = 68;
NewtonModelOil[K_][t_] :=
 Evaluate[Simplify[T[t] /. DSolve[{
       T'[t] == -K (T[t] - ambientOil),
       T[0] == tempOil0}, T[t], t][[1]]]

NewtonModelOil[K][t]
```

Out[16]= $68. + 120.6\, e^{-K t}$

In[17]:=
```
residualOil[K_] := tempOil - NewtonModelOil[K][timeOil];
fitOil = FindFit[dataOil, NewtonModelOil[K][t], {{K, 0}}, t]
```

Out[18]= $\{K \to 0.000308649\}$

In[19]:=
```
Plot[NewtonModelOil [K][t] /. fitOil, {t, 0, toilmax},
 PlotStyle → {Thickness[0.03], GrayLevel[0.7]},
 Epilog →
  {PointSize[0.01], Point /@ dataOil, Line @ data},
 FrameLabel → {"Time (secs)", "° F"},
 RotateLabel → False, Frame → True,
 Axes → False, GridLines → {{}, {{ambientOil, {}}}},
 PlotRange → {{-100, 3700}, {60, 215}},
 FrameTicks → {{0, 500, 1000, 2500, 3600},
   {temp0, 100, 150, ambientOil}, None, None}];
```



The fit is excellent, showing that Newton's Law works very well when evaporation is eliminated. The lower curve in the figure is the data from §2; it is clear that the cooling is much more rapid when there is evaporation.

4. Modeling Evaporation

To set up a differential equation that models evaporation, we need some elementary thermodynamics. Evaporative heat loss will remove approximately 2260 Joules of heat for every gram of water that evaporates; this is called the *latent heat of evaporation* (per gram; we will work per molecule below). As a first attempt at a model, we assume that the rate of evaporation (gram/second) is proportional to the difference between the vapor pressure at the water surface and that in the air. We will see that this assumption is in fact problematic. The vapor pressure at the water surface is governed by the Clausius–Clapeyron equation $dp/dT = L/(T\Delta V)$, where p is the vapor pressure of water at the surface, T is the temperature (in Kelvin), L the latent heat (per molecule), and ΔV the change in volume (per molecule). Our treatment here is necessarily abbreviated, for more details see [1]. We can use the ideal gas law ($pV = NkT$, where N is the number of molecules and k the Boltzmann constant, so $\Delta V = kT/p$) to solve this for $p(T)$ if we assume that ΔV is entirely from the gas (a good assumption for water) and that L is independent of T (reasonable over small temperature ranges); separation of variables yields $p(T) = Ce^{-L/(kT)}$. Here is how this is done using **DSolve**.

```
In[20]:= p[T] /. First[DSolve[p'[T] == L/T p[T]/(k T), p[T], T]]

Out[20]= E^(-L/(k T)) C[1]
```

Next we need the vapor pressure of water vapor in the air, which can be written in terms of relative humidity and the expression for $p(T)$ just found as $RHCe^{-L/(kT_{\text{ambient}})}$, where RH is relative humidity. Summing up, and absorbing C into the proportionality constant, we get:

$$\text{Rate of evaporative heat loss } (T) = K_2(e^{-L/(kT)} - RHe^{-L/(kT_{\text{ambient}})})$$

In reality the latent heat is not exactly constant (it varies by about 10% between the freezing and boiling points), but since the evaporative heat loss is largest near the boiling point (exponentially so), we use the value for that temperature. We now calculate the constant (L/k) in the exponential. We use

Mathematica to do this, keeping track of units as a check (noting that molecules are not considered a unit, and so they disappear). Reminder: One mole of water weighs 18 grams (16 from oxygen, 1 from each of the two hydrogen atoms). A mole of anything consists of an Avogadro constant number of molecules of the substance.

```
In[21]:= Needs /@ {"Miscellaneous`PhysicalConstants`",
           "Miscellaneous`Units`"};

         L = 2260 Joule / Gram;
                          L (18 Gram / Mole)
         c = ------------------------------------------
             AvogadroConstant BoltzmannConstant
```

```
Out[23]= 4892.67 Kelvin
```

```
In[24]:= c = c[[1]];
```

Now we turn to *Mathematica* to set up and solve the evaporative model, redoing the preceding computation using constants in the **PhysicalConstants** package. First we convert the temperatures to the Kelvin scale.

```
In[25]:= {tempK, tempK0, ambientK} = ConvertTemperature[
           {temp, temp0, ambient}, Fahrenheit, Kelvin];
```

Now we set up the equation.

```
In[26]:= evaporationEqn =
           (T'[t] == -K1 (T[t] - ambientK) - K2 ( e^(-c/T[t]) - RH e^(-c/ambientK) ))
```

$$\text{Out[26]= } T'[t] == -\left(e^{-\frac{4892.67}{T[t]}} - 7.93672 \times 10^{-8}\ RH\right) K_2 - K_1 \left(-\frac{53867}{180} + T[t]\right)$$

We did not record *RH*, but the data were taken on a humid September day in St. Paul. We will estimate $RH = 0.75$.

```
In[27]:= RH = 0.75;
```

Because of the exponential term, this equation cannot be solved algebraically, but **NDSolve** has no difficulty, and we set up the model to call on that.

```
In[28]:= EvaporationModel[K1_?NumericQ, K2_?NumericQ][t_] :=
           T[t] /. First[NDSolve[{evaporationEqn /. {K1 → K1, K2 → K2},
               T[0] == tempK0}, T[t], {t, 0, tmax}]];

         evaporationResidual[K1_?NumericQ, K2_?NumericQ] :=
           (EvaporationModel[K1, K2][t] /. t → time) - tempK
```

Here it is easiest to use **FindMinimum** to minimize the sum of squares, since the model is given not by a formula, but by a numerically solved differential equation.

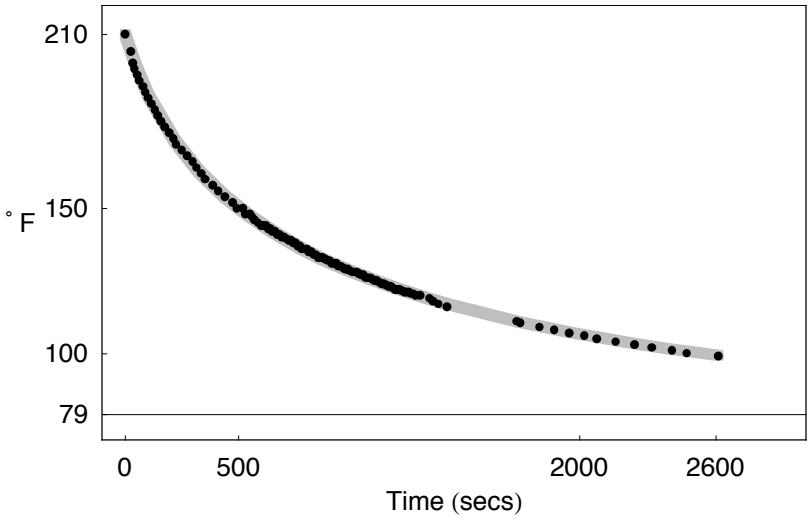
```
In[30]:= (fit = FindMinimum[evaporationResidual[K1, K2].
           evaporationResidual[K1, K2], {K1, 0.001},
          {K2, 10000}, Method → "LevenbergMarquardt"]) // Timing

Out[30]= {0.91 Second, {13.1827, {K1 → -0.0000688032, K2 → 78172.4}}}
```

Ouch! The negative value of K_1 is a physical impossibility, even though the fit is excellent. But let us press on. For convenience, we do the plotting in degrees Fahrenheit. Now the fitting curve is an interpolating function.

```
In[31]:= bestEvaporationModel =
          N[Expand[ConvertTemperature[T[t], Kelvin, Fahrenheit]]] /.
           First[NDSolve[{evaporationEqn /. fit[[2]], T[0] == tempK0},
             T[t], {t, 0, tmax}]]

Out[31]= -459.67 + 1.8 InterpolatingFunction[{{0., 2613.}}, <>][t]
```

And now we plot the model and the data. The fit is remarkably good.

```
In[32]:= Plot[bestEvaporationModel, {t, 0, tmax},
          Frame → True, PlotRange → {{-100, 3000}, {70, 220}},
          Axes → None, GridLines → {{}, {{ambient, {}}}},
          FrameLabel → {"Time (secs)", "°F"}, RotateLabel → False,
          FrameTicks → {{0, 500, 2000, 2600}, {79, 100, 150, 210},
            None, None},
          PlotStyle → {Thickness[0.015], GrayLevel[0.7]},
          Epilog →
           {PointSize[0.0125], Point /@ Transpose[{time, temp}] }];
```



But what can we make of K_1, whose negativity violates the laws of thermodynamics by suggesting that the water gets hotter by virtue of its presence in the cool air? The most likely problem is that our simple model (the proportionality assumption) is not adequate near the boiling point. There are many complicated factors that affect heat transportation, such as air movement, boundary layer dissipation, and diffusion, and our use of a single linear relationship appears to be inadequate. In the next section

we suggest some further experiments, but we also hope that our experiments might inspire readers to come up with a better mathematical model.

There is one interesting aspect of the evaporation model worth noting, regardless of the differential equation one comes up with as a model: the equilibrium temperature (obtained by setting the right side of the differential equation to 0) should be less, but only a little less, than the ambient temperature. In fact, under typical relative humidity, one can easily observe this by comparing air temperature to the temperature of a pot of water that has had several hours to equilibrate. We found that the water temperature was about 1° F less than the air, but that the difference disappeared when the water was coated with oil.

5. Further Experiments

There is clearly much room for further experimentation, and we encourage readers who are so inclined to carry some out. Those with access to a physics lab will know that the measuring of the temperature can be fully automated, which eases the task quite a bit.

1. Model the temperature as a cold object warms up to room temperature. Such an experiment might help resolve the question whether Newton's Law works better when the temperature range is small. To get a large temperature range, try starting with a liquid (e.g., alcohol) that has a high boiling point and does not freeze at, say, 0° F; watch it warm to, say 80° F and see if Newton applies throughout the range.

2. A more complicated but potentially very useful experiment would be to measure the weight of water in the pan throughout the cooling process. For then one could use the standard latent heat of evaporation (1 gram of water requires 2260 Joules to evaporate) along with the heat capacity of water (4.2 Joules per gram) to directly model the temperature change due only to evaporation. See if Newton's Law models what's left over.

3. Perform water and water-with-oil experiments using identical amounts of water and identical pans. Then the oil experiment should lead to an estimate of the conductive coefficient that ought to be quite close to the conduction coefficient that arises from a successful evaporation model.

4. Does radiation play a role? One can try to model radiation by adding a term proportional to T^4 to the rate equation (the Stefan–Boltzman law).

4. A nice puzzle to investigate is the following: If some cold milk is to be added to hot coffee with the goal of getting the coffee down to a certain temperature, when should the milk be added so that the desired temperature is reached as soon as possible?

Acknowledgement. The authors thank B. J. Stanbery, President of HelioVolt, Inc., who pointed out that evaporation was critical and suggested the slick trick of an oil coating as a way to prove it, Rob Knapp (Wolfram Research, Inc.) for some help with **`FindFit`**, and a referee for some very helpful physical observations. We also thank Tak Iwanaga who carried out several experiments as part of a modeling course at Macalester College.

Reference

1. C. Kittel and H. Kroemer, *Thermal Physics*, W. H. Freeman, New York, 1980.