# Coffee Cooling Challenge #3: This Time It's Personal

Daniel Huantes

October 1, 2019

## 1 Introduction

In this final Coffee Cooling Challenge submission I added terms of the differential form of Newton's law of cooling, in order to create a coupled differential equation that represents the cooling of a cup of coffee due to both the heat lost to the air above the liquid, and the sides of the cups surrounding it. The equation I used to model the rate of change in the coffee's temperature, $T$, with respect to time was

$$\frac{\mathrm{d}T}{\mathrm{d}t} = r_{env}(T_{env} - T) + r_c(T_{mug} - T) \tag{1}$$

with $T_{mug}$ being the temperature of the mug, changing with time, $T_{env}$ being the temperature of the environment, assumed to be constant over time. We assumed that the room holding the coffee-mug system was large enough that any heat disappated in it would have a negligible effect on it's total temperature. I modeled $\frac{\mathrm{d}T_{mug}}{t}$, the rate of change of the temperature of the mug using

$$\frac{\mathrm{d}T_{mug}}{\mathrm{d}t} = r_c(T - T_{mug}) + r_{ce}(T_{env} - T_{mug}) \tag{2}$$

Unlike our previous challenges, this version of the coffee cooling problem was more difficult as it involed a coupled differential equation, with both $T$ and $T_{mug}$ varying with time, and affecting each other. Another extra challenge added was the ability for the program to handle a variable volume of coffee entered in a more familiar unit (oz), as well as using measurements of a coffee cup I own in order to provide more reliable estimates of the cooling. Both of these issues improve upon the accuracy of the previous problem by introducing more real life factors that affect cooling, and the utility of the program, with being able to model different cups and volumes of coffee.

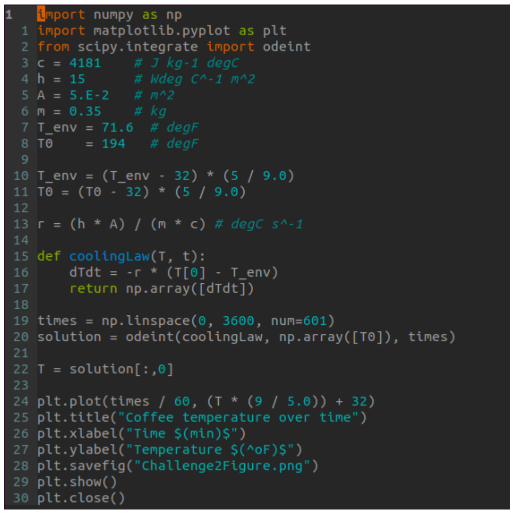
## 2 Discussion

My initial challenge in creating this version of the Coffee Cooling Challenge was creating a program that could take a variable volume of coffee V and with the physical parameters of the mug holding the coffee, could find the necessary volumes and areas required to calculate the 3 r coefficients in Newton's Law of Cooling I used. I was going to use a truncated cone [1] in order to model the geometry of a coffee mug such as my own, which although almost cylindrical, like many cups has a mouth wider than its base. My overall goal was to be able to develop a function in order to find the surface area of coffee in contact with the mug, it makes sense that every distinct volume of coffee that the cup could hold would generate it's own unique height and surface radius, however because of the nature of a cone's geometry, solving for any linear portion from a volume leads to a cubic function. Although the function only has a single meaningful root for our purposes, finding a 1 dimensional quantity, to be used as a linking point, from the surface area of the shape is also a quadratic. Then Patrick Cook heard of my struggles to find this equation, and over the weekend distracted himself from GRE prep with my puzzle, and solved it for me. The full solution is most easily represented in 3 equations, here with full credit and thanks to Patrick Cook, who answered my call: "Patrick, if you figure this out over the weekend don't tell me, I'm trying to work it out on my own"

$$\begin{aligned} h &= \frac{\mathbb{H}}{\frac{\mathbb{R}}{\mathbb{r}} - 1} \\ R &= \sqrt[3]{\frac{\mathbb{H}\mathbb{r}^3 - \mathbb{r}v + \mathbb{R}v}{\mathbb{H}}} \end{aligned} \tag{3}$$

[1] Actually a right circular frustum of a cone

## 3 Code



```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
c = 4181     # J kg-1 degC
h = 15       # Wdeg C^-1 m^2
A = 5.E-2    # m^2
m = 0.35     # kg
T_env = 71.6  # degF
T0     = 194   # degF

T_env = (T_env - 32) * (5 / 9.0)
T0 = (T0 - 32) * (5 / 9.0)

r = (h * A) / (m * c) # degC s^-1

def coolingLaw(T, t):
    dTdt = -r * (T[0] - T_env)
    return np.array([dTdt])

times = np.linspace(0, 3600, num=601)
solution = odeint(coolingLaw, np.array([T0]), times)

T = solution[:,0]

plt.plot(times / 60, (T * (9 / 5.0)) + 32)
plt.title("Coffee temperature over time")
plt.xlabel("Time $(min)$")
plt.ylabel("Temperature $(^oF)$")
plt.savefig("Challenge2Figure.png")
plt.show()
plt.close()
```

Figure 1: My Beautiful Code!

## 4 Results

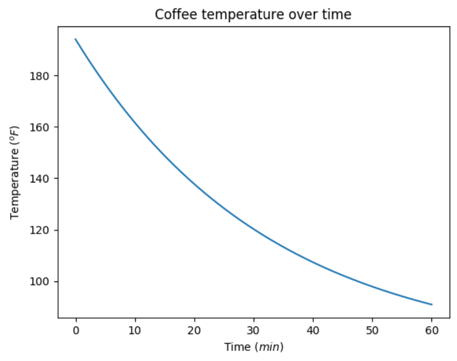


Figure 2: Plot generated from my code

The plots shows that as the difference in temperature between the coffee and the surroundings decrease, the rate at which the coffee cools decreases as well, which is consistent with our formula. We can be satisfied that at least under the assumptions we made, that we have successfully used numerical tools in order to solve a differential equation representing the cooling of a coffee cup.

## 5 Conclusion

We can tell that numerical methods to solve differentials are an extremely useful tool, because although the analytical solution to this specific function is fairly easily found, it does save us as programmers some work, as well as give us a tool to use in the future for more differentials. In further work, I plan to model thetemperatures of both the liquid inside the coffee and the cup in which it is held, as well as introducing ideas like variable volume of coffee.