Coffee Cooling Challenge #2

Daniel Huantes

September 27, 2019

1 Introduction

For this weeks challenge we used numerical methods in order to solve a first order seperable differential equation. The specific method we used was odeint, a function found in scipy's integrate module. The problem we dealt with was using the differential form of Newtons law of cooling

$$\frac{\mathrm{d}T(t)}{\mathrm{d}t} = -r(T(t) - T_{env}) \qquad (1)$$

with T(t) being the temperature of the coffee in our cup as a function of time, T$_{env}$ being the temperature of the room the cup was in, and r to be a physical constant defined as

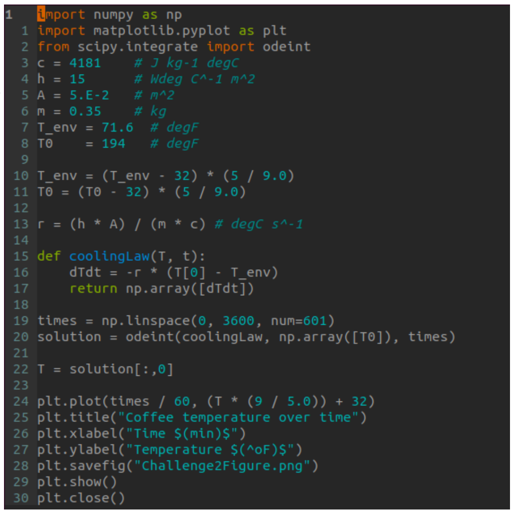
$$r = \frac{hA}{mc} \qquad (2)$$

We assumed that the room was large enough that any heat transferred from the cup to the environment through cooling would have a negligible effect on the temperature of the room. Our heat loss was modeled entirely through the area of the cup that was exposed to the air, and we assumed a container holding the coffee that would not lose any heat to the sides to the mug wall in order to simplify our problem.

2 Discussion

First, we set up our imports, numpy, scipy, matplotlib and our odeint function. Next we set up global variables representing the given quantities that were provided in the challenge document and videos. The global variables T_env and T0, representing T_{env} and $T(0)$ respectively, are now iniialized to a temperature in ^{o}Farenheight. The expressions following their initialization converts them to ^{o}Celcius, in order to be compatible with the units included. After this, I defined my function coolingLaw with two parameters, T and t, and had it return the temperature encapsulated in a numpy array. The function expects T to be a list-like argument, as it used it's zeroth index for T(t). The r constant and T$_{env}$ from Newton's law of cooling are referenced from the previously set up global variables. Odeint expects the method representing the differential to take two inputs, although there is an option to include additional arguments in a tuple at the end of the arguments of odeint to pass any further arguments to the function. Odeint's two other arguments other than the function it solves are the temperatures and times. Both were passed as numpy arrays, times being a list of linearly spaced points from 0 to 3600, representing our domain of interesti in seconds, 1 hour, and a list containing one member, the initial temperature of our coffee. We take the zeroth column of odeint's result and place it into a variable called T. Finally, I plotted T against times, using a vector operation inside of the plot call to convert times from seconds to minutes and another to convert our temperatures from ^{o}C to ^{o}F.

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

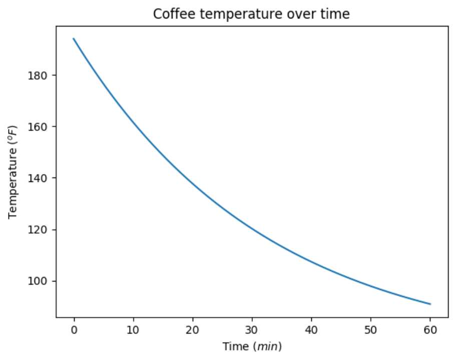


Figure 2: Plot generated from my code

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.