Shock Waves manuscript No.
(will be inserted by the editor)

Org-Mode Example Paper

Sean O'Byrne

University of New South Wales, Canberra, ACT, 2600, Australia

December 18, 2021

Abstract. This document illustrates a literate programming workflow using org-mode and org-babel to incorporate python and other code into a document. I have chosen the Shock Waves style file, but this could be relatively easily changed to something else. The example document that we will be using for our study will consider the question of whether adding milk immediately makes a cup of tea hotter or colder than adding it when it is about to be consumed at a later time.

Key words: tea, org-mode

Introduction

This document is designed to showcase how emacs Org-mode can be used to interact with code to allow for reproducible research. Most examples will be executed using the Python and lisp programming languages. I'll use the example of calculating the cooling of a cup of tea. The aim is to provide a simple template outlining the capabilities of org-mode for combining text production and computation. Each of the figures and tables in the document uses data generated by codes embedded in the document itself. References to calculated data within the text are also converted to numbers when the document is generated. In this way, the paper is self-documenting. Looking at the source code that generates the paper provides the reader with a transparent view of how the data within the paper was generated.

The Tea Cooling Problem

Let's say you just made yourself a cup of tea and you decide to add some milk to it before consuming it. However, being a forgetful academic, you have a tendency to consume it much later than when you make it. The obvious question is: to keep the tea as warm as possible for a delay of 10 minutes, is it better to add the milk when you initially make the tea or when you remember that you need to drink it at the later time?

You might think that such considerations do not matter, and that there is no standard way to make a cup of tea. As a matter of fact, there **is** a standard way of making a cup of tea [BS6008-1980, 1980]. According to this standard, the best results for infusion of tea occur when the temperature of the liquor (the mixture of tea extract and water) is 65 to 80 °C.

There is a fundamental difference between the two ways of adding the milk to tea:

- if you add the milk immediately, there is a sudden drop in temperature *before* the temperature decreases gradually over time due to cooling;
- if you add the milk after the delay, the sudden drop occurs after the cooling process has completed.

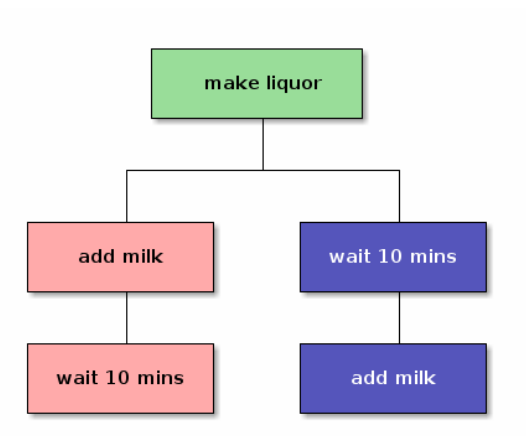


Fig. 1: Tea-making flowchart

So there are three things that can possibly happen

1. Adding milk immediately leads to a higher temperature after a given delay;
2. Adding milk after the cooling delay leads to a higher temperature;
3. There is no difference in the temperature between the two ways of adding the milk.

Both of the possible cooling sequences are summarised in Fig. 1. The question that we wish to answer in this paper is which of these outcomes is likely to happen for a particular combination of volume fraction of tea to milk

(in this case, 0.125), temperature of the tea (80 °C), temperature of the milk (20 °C) and ambient temperature (20 °C).

The Calculation

The first thing we need is a model of the tea cooling process. For the sake of simplicity here we will use Newton's Law of cooling [Winterton, 1999]. This equation proceeds from the logical idea that the rate of change of temperature as a liquid cools is proportional to the difference between the initial temperature and the temperature of the surroundings.

$$\frac{dT}{dt} = -k\left(T(t) - T_a\right) \tag{1}$$

where k is a rate constant and T_a is the ambient temperature of the surrounding air. The example code uses temperatures in degrees Celcius. If we integrate this differential equation, the solution is of the form

$$T(t) = Ce^{-kt} + T_a \tag{2}$$

where k is a dimensional constant indicating the rate at which the mixture cools and C is a constant of integration that can be found using the value of T at t=0. If the temperature of the tea is 80 °C initially, and the ambient temperature is 20 °C then $C = T_0 - T_a = 60$ °C. As the constant can be expressed in terms of the initial and ambient temperatures, we can calculate the temperature at any time using

$$T(t) = (T_0 - T_a)e^{-kt} + T_a \tag{3}$$

For the cooling tea, k has a value of 0.0004476 s^{-1}, ensuring a rather slow exponential decline in temperature over time.

To implement this calculation, we define a function that we call `cooling_law`, as defined below. This function can then be called within the document. We have defined it using the `:exports code` directive so that the code finds its way into the final pdf document. Using `:exports none` would suppress the output of the source code, which would be the more usual behaviour for such a code.

```
import math
def cooling_law(temp_out,temp_start,wait_time):
    k=0.0004476
    return temp_out \
+ (temp_start - temp_out)\
*math.exp(-1.0*k*wait_time)
return format(cooling_law \
      (temp_out,temp_start,wait_time),".2f")
```

This calculation can be called inline within a paragraph by using the `call` command. For example, we can find the output of our cooling law with inputs of 80 °C and 10 minutes (600 seconds) by using `call` and should get the answer of `65.87` °C. Note that this last number was produced by a call of the function and was not typed in directly. If you were to change the inputs to the function call and recompile the document, the number would change.

Calculating Cooling Using Tables

We can populate a table using a function by calling that function within the table. In this case, let's start at 80 °C and see what happens at different wait times.

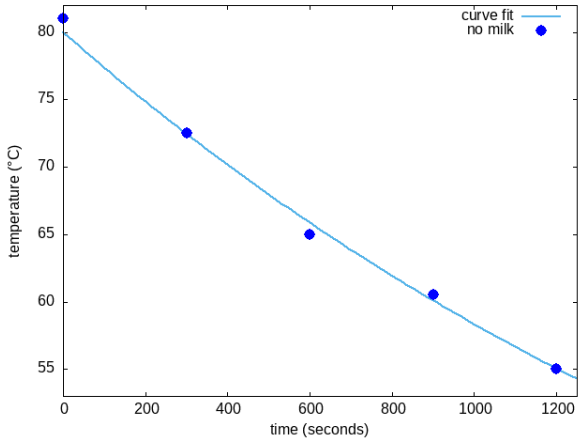
We will populate the temperature row of the table by using a lisp equivalent of the python function above. This is because lisp can directly be called to populate rows or columns of a table. I have also put some fake experimental data in the table to make it look like someone did an experiment.

time (s)	predicted T (° C)	measured T (° C)
0	80.00	81.0
300	72.46	72.5
600	65.87	65.0
900	60.11	60.5
1200	55.07	55.0

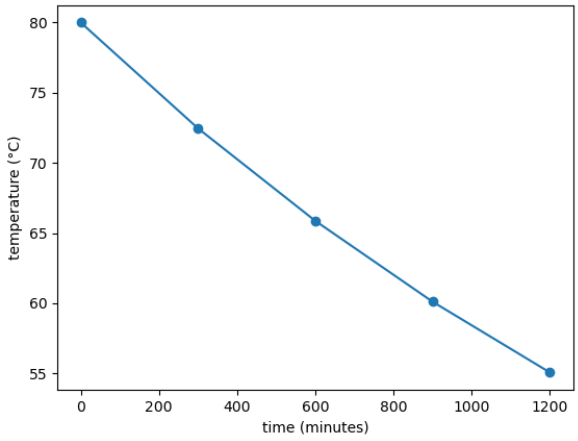
My preferred tool for plotting is gnuplot, because I have used it for a long time, and it's text-based. Org-mode allows us to plot the data we need to plot using org-babel gnuplot mode. There is also a built-in table plotting set of commands for org-tables, but the options for this are limited so I always use a gnuplot source block instead. The source block effectively works as a gnuplot script.

We are not limited to gnuplot, however. We can also take the data that we have calculated in the table and plot it in python using its matplotlib library. We could, of course, achieve the same thing in octave, matlab, R, C, Julia or whatever programming language that you prefer that is supported by org-babel. Note that in this case we have plotted points with straight-line segments between them, rather than plotting a separate fit curve as was done with the gnuplot graph. Provided (unlike me) you know enough python, you could do a curve of best fit or plot a function if you wanted to, as the source block effectively acts like any python script.

We can compare the cooling curve plots generated by gnuplot and python by plotting the two together, as shown in Fig. 2. The differences between the two plots indicate why it's not a great idea to mix plotting packages used to generate data for a paper, but it serves the purpose of illustrating the variety of ways such plots can be generated using org-mode. Note that I plotted the experimental data with a curve fit in gnuplot, but just plotted the points for the curve fit connected by points in python, because that's about all I can manage in python, but it gives the idea. I plotted the two graphs in raw latex using the `subfig` package, because there does not seem to be an easy built-in org-mode way of doing this and it illustrates how easily raw LaTeXcan be integrated into an org-mode file.



(a) Gnuplot



(b) Python

Fig. 2: Cooling curves plotted in different environments

Mixing

Now that we have a formula for the cooling of a liquid, we now need to determine the effect on temperature of mixing the tea and the milk. The temperature should be proportional to the difference in the temperatures (let's assume the milk is at room temperature) weighted to the relative liquid volumes:

$$T_{mix} = \frac{T_{tea}V_{tea} - T_{milk}V_{milk}}{V_{tea} + V_{milk}} \tag{4}$$

Again, I'm going to use a lisp function that can be used to populate the table

```
(defun tmix (temp_tea temp_milk vfrac)

  "Determines the final temperature of a
mixture of milk and tea based upon the
fractional volumes of the two mixture
components. vfrac is the fraction of the
mixture that is milk."
  (- (* temp_tea (- 1 vfrac))
     (* temp_milk vfrac)))
```

So, if we use this function assuming a volume fraction of 0.125 and a milk temperature of 20 °C, we can populate another table and make a plot comparing the cooling rate for the milk added after 10 minutes to the milk added immediately. Adding the milk immediately drops the first temperature from 80 °C to `67.5` °C.

t (s)	Milk later T (° C)	Milk first T (° C)
0	80.00	80.00
5	79.87	67.39
60	78.41	66.13
120	76.86	64.91
180	75.36	63.72
240	73.89	62.56
300	72.46	61.44
360	71.07	60.34
420	69.72	59.27
480	68.40	58.23
540	67.12	57.21
600	65.87	56.23
605	55.14	56.15

The comparison between these two cases can be seen in Fig. 3. The green curve corresponds to allowing the tea to cool for 10 minutes and then adding milk, while the blue curve shows the effect of adding milk immediately, reducing the temperature to `67.5` °C and then allowing it to cool for 10 minutes.

It is clear from Fig. 3 that the slope of the green cooling curve is higher than that of the blue curve. This finding is consistent with Newton's law, where the rate of change of temperature is proportional to the difference between the temperature of the liquid and the temperature of the surroundings, as expressed in Eq. 1.

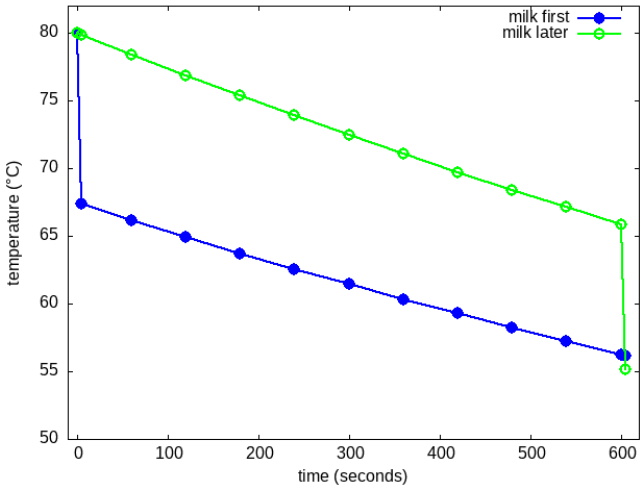


Fig. 3: Cooling comparison

Conclusions and Future Work

In this study we saw that, by a very small margin, if you leave a cup of tea for 10 minutes before adding milk, it ends up slightly colder than if you add the milk immediately, at least for our nominal conditions regarding the cooling rate of the liquid and the volume fraction of milk. Hopefully I have also shown that emacs org-mode, when combined with other free and open-source software can provide a completely traceable piece of research documentation for which all of the calculations are in the open.

This paper only scratches the surface of the capability of emacs as a system for performing robust and reproducible research. It has many capabilities in terms of project management and organisation that can also facilitate efficient completion and documentation of research projects.

In terms of future work, you might like to change some of the parameters in this paper and generate different graphs for different cooling rates, cooling times and ratios of milk to tea. More importantly, I hope that the paper will encourage you to try writing your next paper using this environment. Although there is quite a steep learning curve regarding how to make things work in the emacs/org-mode software ecosystem, the ability to harness a range of programming languages and other applications and integrate them consistently into a publication make it the most flexible platform for reproducible research that I am aware of.

References

BS6008-1980 (1980). Method for preparation of a liquor of tea for use in sensory tests. Standard, British Standard Institution, 2 Park St., London, W1A 2BS.

Winterton, R. (1999). Newton's law of cooling. *Contemporary Physics*, 40(3):205–212.