Uncertainty Quantification using EasyVVUQ

Anton Lebedev

May 28, 2024

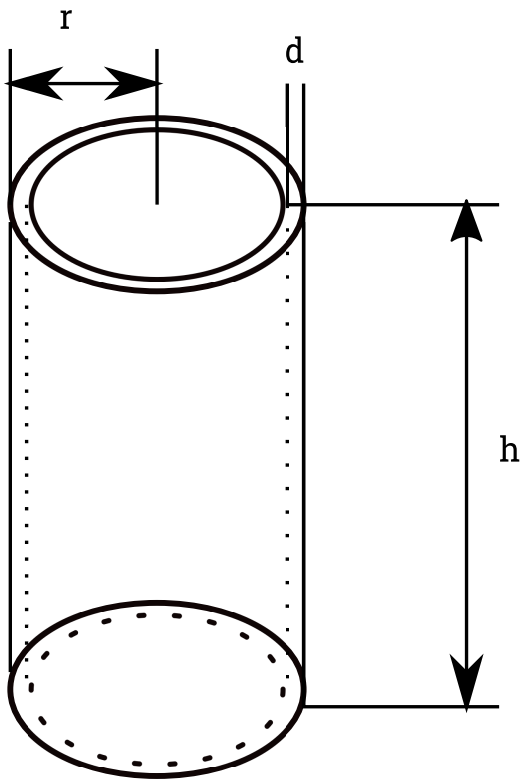

Figure 1: Simple cup model

0.1 Cooling a Cup of Water

0.1.1 Problem Description

Scalding hot coffee ($95[^\circ C]$) is poured into a take-away cardboard cup and deposited on wooden a table outside on a cool morning ($T_{out} = 5[^\circ C]$). The height of the cup is $14[cm]$ and its outer diameter is $D = 7[cm]$. The thickness of the walls and the bottom of the cup is $\approx 1.5[mm]$ and the lid's thickness is $\approx 0.5[mm]$.

We would like to know the evolution of the temperature of the coffee and the earliest likely time at which the coffee is easily consumable ($T \approx 45[^\circ C]$).

0.1.2 Assumptions

Most of the quantities specifying the system are estimated/crudely measured, as such they are erroneous/uncertain. Furthermore we make the following assumptions/simplifications:

1. The shape of the cup is a cylinder with an outer radius r_0, wall thickness d and height h (in general it would be a conic section).

2. The cup is closed by a polypropylene lid which is assumed to be just a disk of radius r_i, that is the inner radius of the cylinder.

3. No heat can flow through the bottom of the cup (the bottom is adiabatic).

4. The contents of the cup are essentially water.

Justifications of assumptions The justification for the shape simplification is that, due to symmetry, the shape of a conic section is similar to a cylinder. The adiabatic bottom of the cup is ensured by placing the cup on a wooden table, thereby improving the insulation of the bottom. As for the contents, coffee is essentially 'muddy water' and we do not expect its physical properties to differ from water's.

0.1.3 Formulae

To determine the heat flow out of or into the cup we require the heat conductivities for the materials λ_P (paper) and λ_{PP} (polypropylene, plastic) as well as the heat capacity per unit mass c_m of water:

$$\lambda_P = 0.15 \left[\frac{W}{m \cdot K}\right]$$
$$\lambda_{PP} = 0.23 \left[\frac{W}{m \cdot K}\right]$$
$$c_{m,W} = 4186 \left[\frac{J}{kg \cdot K}\right]$$

The heat flow through a surface is proportional to the surface's area A, the temperature difference across the surface $T_1 - T_2$, its heat conductivity λ and inversely proportional to the surface's thickness d:

$$\dot{Q} = \frac{\lambda \cdot A}{d}(T_1 - T_2) \tag{1}$$

For a cylinder's mantle the formula is slightly more elaborate:

$$\dot{Q} = \frac{2\pi h \lambda}{\ln(\frac{r_o}{r_i})}(T_1 - T_2) \tag{2}$$

where r_o, r_i are the outer and inner radii of the mantle and h its height.

The total flow of heat from the cup is then the sum of the flow through its walls and the lid.

Heat flow $\dot{Q}$ has the units of W since it is the change of thermal energy per unit time. The thermal energy stored in an object is given by its heat capacitance per unit mass c_m its mass m and the absolute temperature T (in Kelvin[1] of the object:

$$Q = c_m m T \ . \tag{3}$$

Since c_m is a constant and the mass of our coffee does not change we get:

$$\dot{Q} = c_m m \dot{T} \ . \tag{4}$$

Where the mass of water m can be determined using the volume of the cup and the waters density at $\rho = 1000[kg/m^3]$.

0.1.4 Implementation Task

Using the information provided above implement an EasyVVUQ campaign in Python and quantify the uncertainty of the temperature output depending on the uncertainty of the measurements of the cup radius, the environmental temperature and the coffees initial temperature.

For length measurements assume normally distributed measurements with a standard deviation of $0.5[mm]$. Temperature measurements can be assumed to be normally distributed, too, with a standard deviation of $5[^\circ C]$.

Start off by adding heat flow through the lid only and varying its thickness. Then proceed to add heat flow through the cup mantle and varying its thickness as well as the outside temperature.

Quantify, using first Sobol indices, the importance of each of the uncertain parameters for the final temperature.

Furthermore, determine (roughly, i.e., by visual inspection) the earliest, average and latest time at which the coffee will have reached a consumable temperature.

Extensions We have assumed $c_{m,W}$ to properly quantify the heat capacity of the coffee. Modify the set-up in a way as to allow a variation of this quantity and assume it to vary within 100 units around the given value.

0.1.5 Implementation Notes

The model and intergration routine are given below, where functions for the heat coefficients as well as the cup's volume calculations have to be

[1]For temperature differences: $1^\circ K = 1^\circ C$. But $0^\circ C$ corresponds to $273.15K$.

implemented.

```
def cupModel(time, T_0, T_env, r_o, d_cup, h, d_lid):
    # For the ODE integration
    from scipy.integrate import odeint
    heatCoeff = cylindricHeatCoefficient(r_i, r_o, h, lambdaPaper) \
    + planarHeatCoefficient(A, d_lid, lambdaPP );
    Vol = # cup volume

    alpha = heatCoeff / (rhoWater * Vol * c_m)

    # The equation describing the model
    def f(T, time, alpha, T_env):
        return alpha * (T_env - T)

    # Solving the equation by intergration
    temp = odeint(f, T_0, time, args=(alpha, T_env))[:, 0]

    # The output temperature
    return temp
```

Since the simulation is supposed to be disjoint from the EasyVVUQ campaign definition it has to be housed in a separate file, with the following added

```
if __name__=='__main__':
    json_input = sys.argv[1]
    with open(json_input, "r") as f:
        inputs = json.load(f)

    r_o = float(inputs['r_o'])
    d_cup = float(inputs['d_cup'])
    h = float(inputs['h_cup'])
    d_lid = float(inputs['d_lid'])
    t_env = float(inputs['t_env'])
    temp0 = float(inputs['T0'])

    t = np.linspace(0, 2500, 500)

    output_filename = inputs['out_file']

    te = cupModel(t, temp0, t_env, r_o, d_cup, h, d_lid)
```

```
    # output csv file
    np.savetxt(output_filename, te,
           delimiter=",", comments='',
           header='te')
```

To implement a UQ campaign you will need

```
import easyvvuq as uq
import chaospy as cp

import matplotlib.pyplot as plt
import numpy as np

uq.Campaign(name=<name>)
uq.encoders.GenericEncoder
uq.decoders.SimpleCSV
uq.actions.ExecuteLocal(<string containing a call to python and the input file>)
uq.actions.Actions(
                uq.actions.Encode(encoder),
                executor,
                uq.actions.Decode(decoder))

uq.sampling.PCESampler(vary=variablesToVary, polynomial_order=...)
uq.analysis.PCEAnalysis
```