Assignment 1:
PDE - Heat Flow

Luca Bertoni
Alice Hartog

Environmental Systems Analysis

Faculty of Geoscience
Utrecht University
The Netherlands
February 22, 2021

2.1 Theory

In reality the heat capacity and heat conductivity are not constant values. In fact, they depend on the temperature gradient across the object. In this case, being the temperature a function of time, its gradient too is a function of time. Moreover, the temperature gradient decreases as we go deeper into the soil.

As a result, the heat capacity and heat conductivity depend on both space and time.

2.2 Discrete approach

2.2.1

HeatCont is measured in $[J/m^2]$.
Temperature is a quantity that is not conserved during a process. Therefore, a balance of temperatures cannot be done. On the contrary, the total amount of energy does not change. This means that the heat content allows to use the continuity equation and a proper balance can be calculated.

2.2.2

The unit of heat flow is $[J/(m^2 * day)]$.

2.2.3

We multiply ThickL by 0.5 in the denominator because when considering the uppermost layer the distance between node and surface is half the distance between two nodes.

2.3 Writing a simulation program in MATLAB

2.3.1

The Constants are parameters globally fixed and do not depend neither on the model nor on the physics of the problem. Control constants are parameters that are decided by the user depending on the required accuracy and computation time of the model. The value of these parameters remains fixed during the computation. System parameters are related to the physical properties of the system. Control variables are variables that regulate the flow of the model. System variables describe how the physics of the system evolves during the model steps (that is, during time).

2.3.2

The equations of Flow, NFlow, HeatCont and Temp are the following:

$$Flow_j^i = \frac{Conduc * (T_j^i - T_{j-1}^i)}{ThickL} \tag{1}$$

Written in MATLAB as: `Flow(k) = Conduc*(Temp(k)-Temp(k-1))/ThickL;`

$$NFlow_j^i = Flow_{j-1}^1 - Flow_j^i \tag{2}$$

Written in MATLAB as: `NFlow(k) = Flow(k+1) - Flow(k);`

$$HeatCont_j^{i+1} = HeatCont_j^i + NFlow_j^i * dt \tag{3}$$

Written in MATLAB as: `HeatCont(k) = HeatCont(k) + NFlow(k)*dt;`

$$Temp_j^{i+1} = \frac{HeatCont_j^{i+1}}{HeatCap * ThickL} \tag{4}$$

Written in MATLAB as: `Temp(k) = HeatCont(k)/(HeatCap*ThickL);`
Running the program with these equations we obtain this graph:

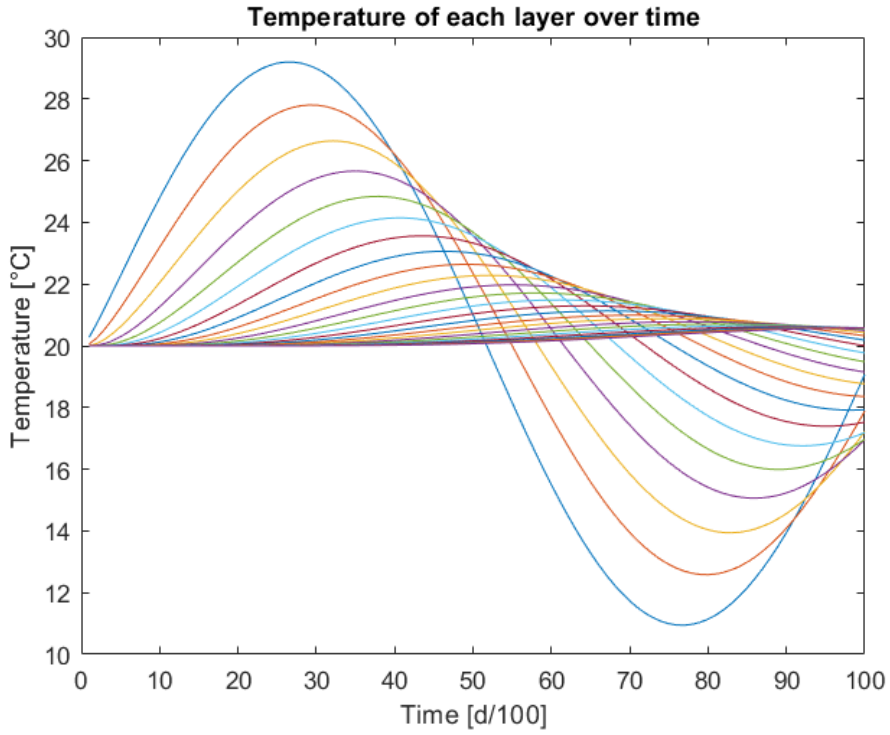

Figure 1: Representation showing temperature variation through time.

Figure 1 shows the variation of temperature in each soil layer, each of them represented by one line, through time.

The result shows a a sinusoidal temperature variation with time for each layer. As the depth (number of layer) increases, the amplitude of the sine decreases exponentially and its peak is delayed in time. Looking at the analytical solution of the heat equation with this specific boundary conditions, this behaviour is exactly what we expected.

2.3.3

The for-loops replaced by the single statement notation used in MATLAB become:

```
Flow(2:NrLayer) = (Temp(2:NrLayer)-Temp(1:NrLayer-1))*Conduc/ThickL;
NFlow(1:NrLayer) = Flow(2:NrLayer+1) - Flow(1:NrLayer);
HeatCont(1:NrLayer) = HeatCont(1:NrLayer)+NFlow(1:NrLayer)*dt;
Temp(1:NrLayer) = HeatCont(1:NrLayer)/(HeatCap*ThickL);
```

And the resulting graph is exactly the same as in Figure 1.

2.3.4

The calculation time of the two problems are a bit different: the first one, built with the for-loops, has an average run time of 0.3322 seconds, while to run the second one, build on the single statement function, the program needs on average 0.3071 seconds. The calculation is based on 10 runs per problem before calculating the average time. Of course the specific values differ for different computers and on different runs, but this means that the second problem takes less time compared to the first one.

2.3.5

The second way to write the for-loop is best since it takes less time to run and plot the chart compared to the first method. In this particular problem, a few hundredth seconds make not much difference, but in more complicated problems could mean significant saved time.

Moreover, from a design perspective, the built-in for loop makes the code much cleaner and easier to understand.

2.3.6

If the command `pause(0.2)` is removed, we can only see the last state of the system. This happens because the computational time of the program is so fast that the time evolution of the system cannot be appreciated by the human eye. Therefore, a short pause at every time step is needed.

Replacing the `pause(0.2)` command with `drawnow` allows to see the time evolution. It takes less time than using `pause(0.2)`, because if the time between the updates is less than 50 milliseconds, then the function discards the new updates. This results in a much faster time evolution, but at the expense of some information.

2.3.7

We find representation with `plot` to be more informative compared to the one with `imagesc`. Even though `imagesc` create a dynamic overview that makes the system evolution easy to visualize, it make more difficult to extrapolate quantitative information. On the other hand, `plot` allows to recognize with more details the temperature trend of each layer and their behaviour relative to each other.

Figure 2 shows again the temperature variation through time of each layer. The plot is the same obtained in Section 2.3.2.

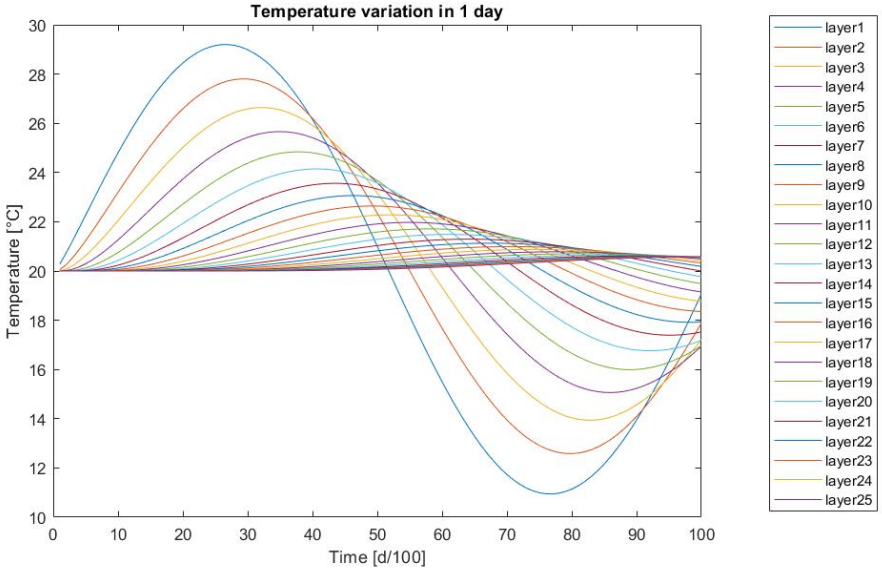

Figure 2: *Representation of temperature variation through time using the single statement.*

2.3.8

a) An experimental measure of the soil temperature as the depth increases is provided and shown in Figure 3. From the graph it can be noticed that the temperature at depth of 10 cm is greater than the surface one. This means that the surface temperature was initially greater than $\simeq 16.7\ ^{\circ}C$ and it experienced a rapid change. The quickness of the change is supported by the fact that the deeper layer of the soil are at a much lower temperature.

c) Initializing the soil temperature with the experimental data provided, the simulation is run. The temperature as a function of the depth changes over time in a sinusoidal way, with a phase shift as the depth increases. The last plot of the simulation is reported in Figure 4.

d) In Figure 5 all the temperature profiles of the soil throughout the simulation are displayed. For the sake of readability, only the timesteps are 5 times bigger (5/100 of a day).

2.3.9

a) In Figure 6 a quick visualization of the temperature of soil from day 230 to day 243 is reported.

b) and c) In Figure 7 the code for data preparation is reported.

d) In Figure 8 and in Figure 9 are reported, as well as the necessary changes that are implemented in order to make the simulation work.

Using the provided data as a surface temperature during time, during the whole simulation time the following temperature profiles are obtained (Figure 10).

Even though it is hard to provided an interpretation of the result, some insights can be obtained by comparing it with Figure 5. It is possible to notice that the temperature

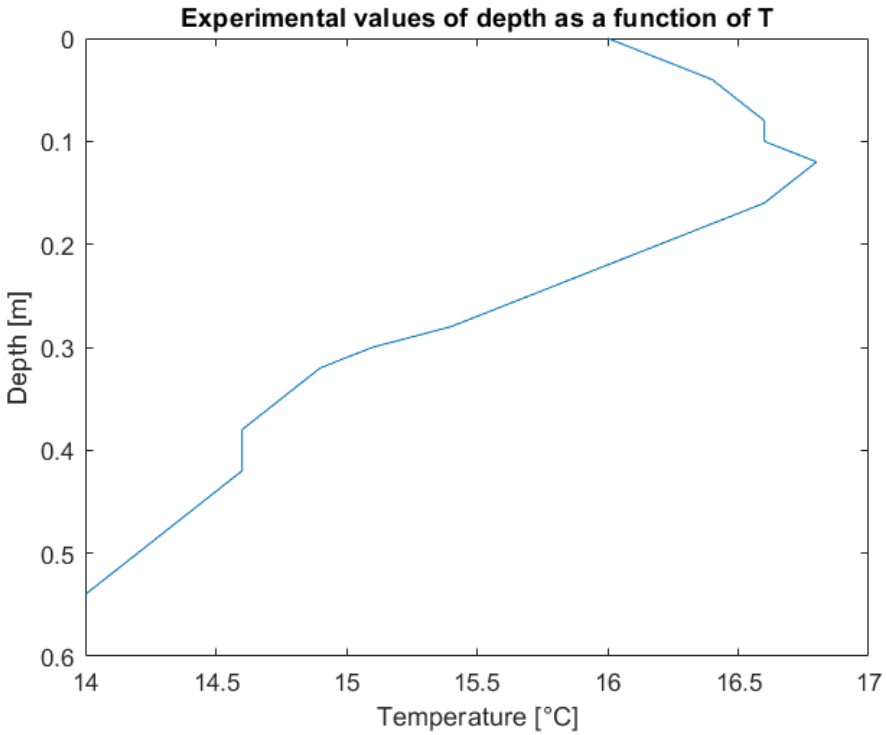

Figure 3: *Experimental values of temperature at different depths.*

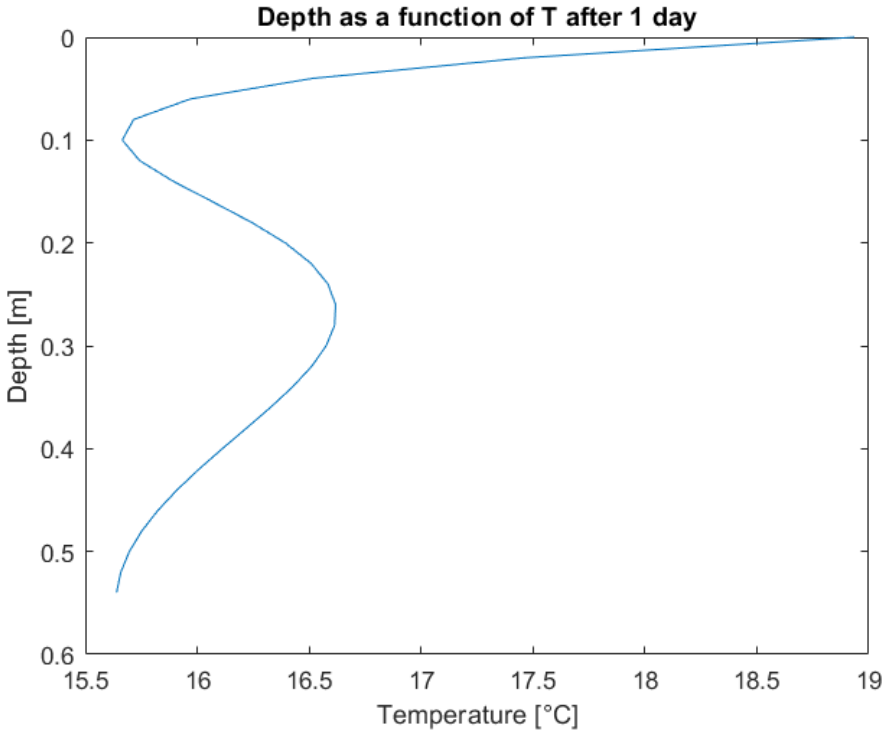

Figure 4: *Representation showing temperature trend variation through soil layers.*

range in the deeper layers is much wider when experimental data are used for the surface temperature. This means that, on average, the temperature variation is slower compared

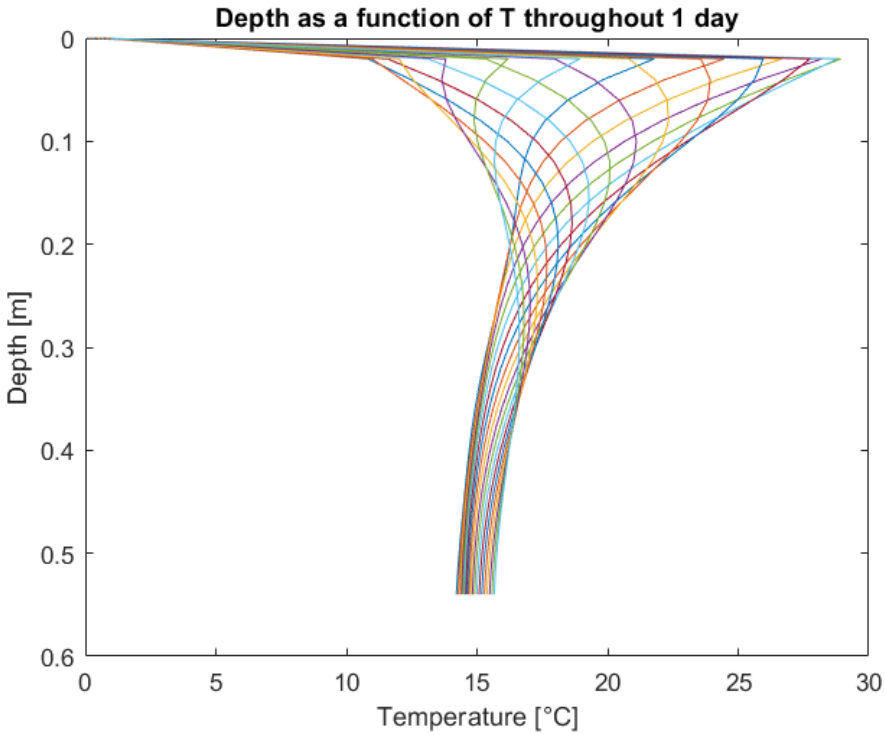

Figure 5: *Depth as a function of temperature throughout a 1 day simulation.*

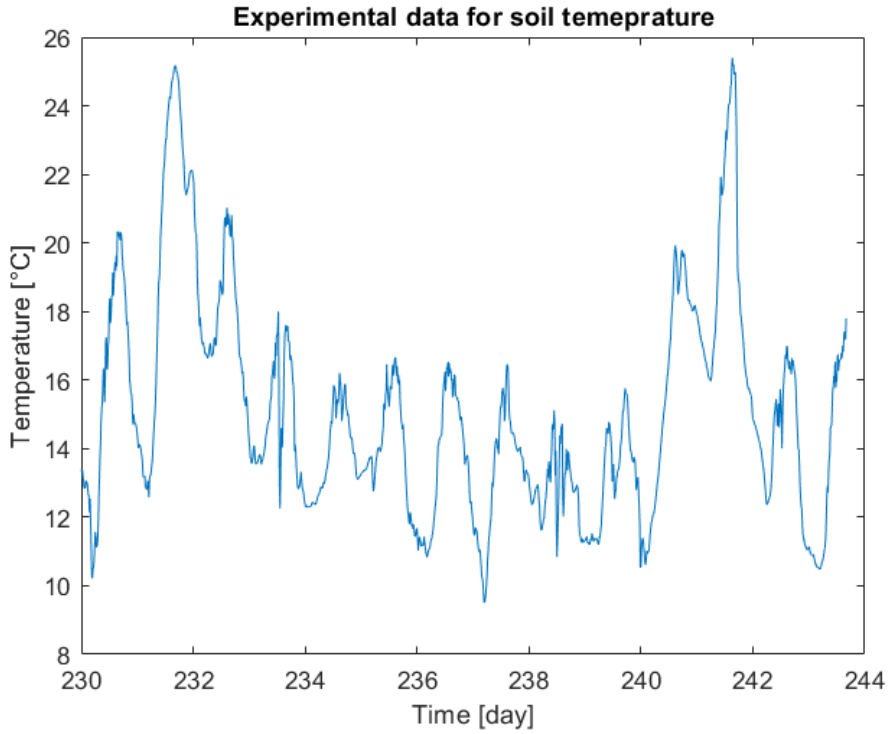

Figure 6: *Experimental data for soil temperature from the day 230 to the day 243.*

to the simulated one.

On the other hand, the uppermost layers behave in a similar way in both the simula-

```
% data preparation

data = readmatrix('STemp.txt');
total_time = linspace(230,243,13000);
surface_temperature = interp1(data(:,1),data(:,2), total_time);
```

Figure 7: *Loading of the file and data preparation*

```
% Constants
Omega = 2*pi;
% Control Constants
StartTime= 0; % [day]
EndTime = 13; % [day]
dt = 0.001; % [day]
```

Figure 8: *New StartTime and EndTime parameters.*

```
k = 1;
```

```
%%
%%%%%%%%%%%%%% DYNAMIC %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
while Time < EndTime && k < 13000



% Dynamic boundary conditions
SurfTp = surface_temperature(k) ;
Flow(1) = Conduc*(Temp(1)-SurfTp)/(0.5*ThickL);
```

Figure 9: *Other adjustment required in the code.*

tions.

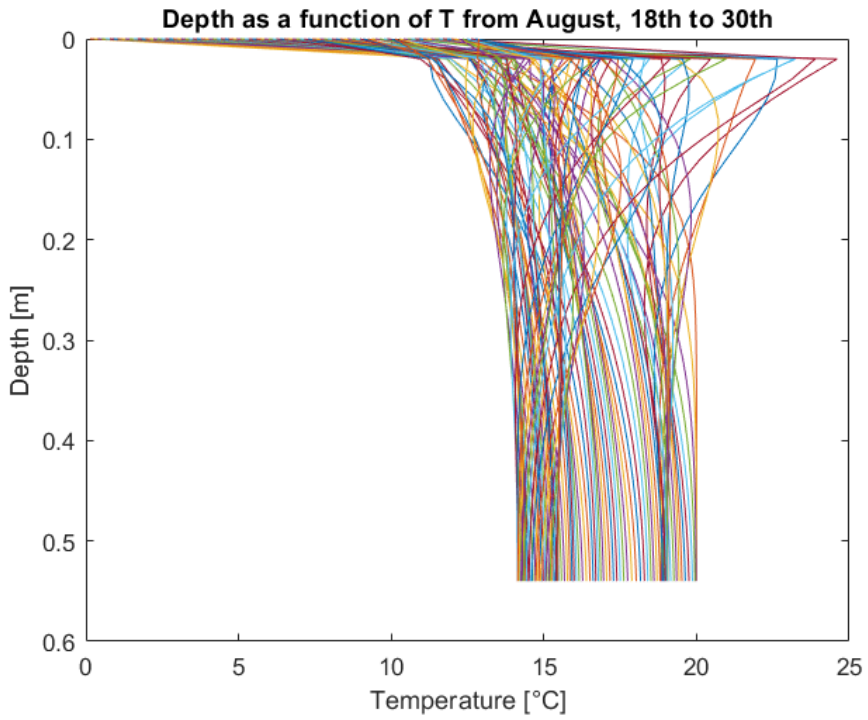

Figure 10: *Depth and temperature profiles of soil from August 18th to 30th using experimental data.*