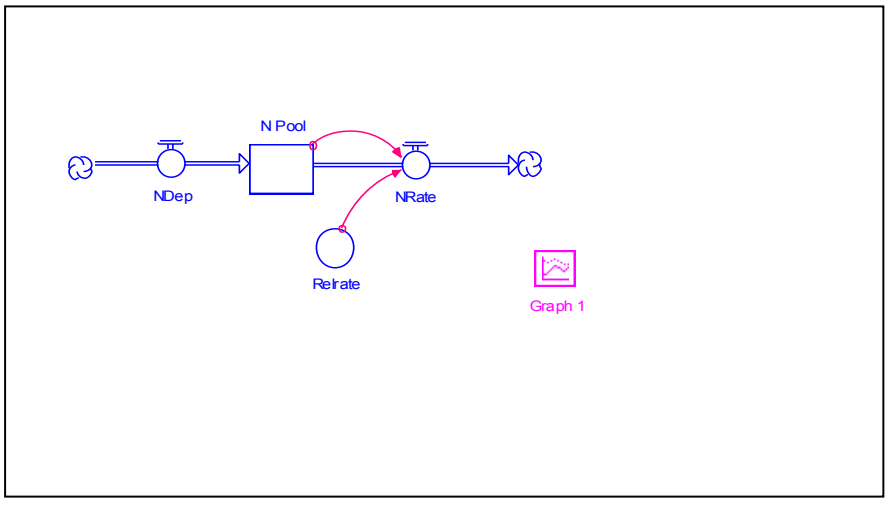
Partial differential equations: Heat flow

2 Numerical solutions of partial differential equations

During the course Systems Thinking, Scenarios & Indicators for Sustainable Development you have been introduced to models implemented in Stella. By drawing relational diagrams, and assessing the rate equations you were able to solve problems. An example of such a model is for instance:



After entering the rate equations and all model parameters, the program source code was generated automatically and the calculation including the integration were carried out. The source code of this example was:

```
INITIALISATION
INIT N_Pool = 100

PARAMETERS
Relrate = 0.01

INFLOWS:
NDep = 40/365

OUTFLOWS:
NRate = Relrate*N_Pool


N_Pool(t) = N_Pool(t - dt) + (NDep - NRate) * dt
```

Stella as a modelling language is very useful in the design of new models and the development of concepts. However, it is difficult to use for systems with many state variables, because you then get very complex relational diagrams. Also for discretized systems (where a system is spatially split up in equally behaving sub-systems) Stella is not an optimal modelling tool.

During mathematics courses you have encountered partial differential equations, hereafter referred to as "pde". You also learned some techniques to solve these equations analytically. In reality, however, **many pde's only have analytical solutions for very specific boundary conditions**. And nature does not behave in accordance to these specific boundary conditions.

In this chapter the numerical solution of pde's will be explained using the heat flow equation. As for any matrix calculation, MATLAB is very well suited for solving partial differential equations.

2.1 Theory

An example of a process that translates into a partial differential equation is heat flow through the soil. Its equation is:

$$C * \frac{\partial T}{\partial t} = \lambda * \frac{\partial^2 T}{\partial z^2} \quad (2.1)$$

T	temperature	[^oC]
t	time	[day]
z	depth in the soil column	[m]
C	volumetric heat capacity	[J $^oC^{-1}$ m^{-3}]
λ	heat conductivity	[J m^{-1} $^oC^{-1}$ day^{-1}]

Given specific boundary conditions, this equation could be solved analytically. If, for example, the soil column is an infinite slab and surface temperature, T_s, varies according to:

$$T_s = T_0 + T_a * \sin(2 * \pi * t) \quad (2.2)$$

T_0 = average temperature
T_a = amplitude of daily temperature changes,

the solution to the partial differential equation becomes:

$$T(t,z) = T_a * e^{-zd} * \sin\left(2 * \pi * t - \frac{z}{d}\right) + T_0 \text{ with } d = \sqrt{\lambda/(C * \pi)} \quad (2.3)$$

This solution is the space-time evolution of the soil temperature, T(t,z), but it has a limited value. The solution is only valid with specific boundary conditions and for situations where heat capacity as well as heat conductivity are constant in space and time.

Exercise 2.1.1 *On what will heat capacity and heat conductivity in reality depend? Why would they change in space and time?*

2.2 Using a discrete approach

We will now discuss the numerical background of a computer simulation model for calculating heat transport in the soil and the temporal evolution of temperature throughout a one-dimensional soil column. The computer simulation model is a numerical solution of equation 2.1 for a given initial temperature profile and given boundary conditions.

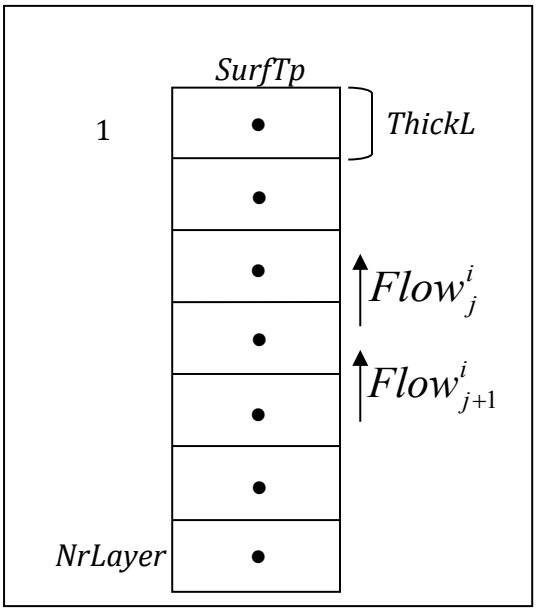


Figure 2.1: Discrete approach to the heat flow equation.

To solve the pde we can use numerical methods to create an algorithm that calculates change in temperature throughout the column as a function of time and depth. We then work with discrete steps of Δz and Δt instead of infinitesimally small time steps ∂z and ∂t. As you can see in Figure 2.1, we divide the soil column in a number of layers, NrLayer, of equal thickness, ThickL. The temperature of a layer is constant throughout the layer and is represented by the temperature of a **node** in the middle of the layer.

The flows are represented by the arrows and are valid between the cells. $Flow^i_{j+1}$ is the flow between nodes j and j+1.

It is now possible to write a separate heat balance for each individual layer.

Let's consider the heat balance of layer j as determined by the conditions at time i. We call the heat capacity of each layer HeatCap [J $^oC^{-1}$ m^{-3}] and heat conductivity between two layers Conduc [J m^{-1} $^oC^{-1}$ d^{-1}]. For the time being, HeatCap and Conduc are assumed to be constant in time and equal for each layer.

We first relate the volumetric heat content of a layer and its temperature:

$$\text{HeatCont}^i_j = \text{T}^i_j * \text{HeatCap} * \text{ThickL} \qquad (2.4)$$

HeatCont is the state variable that we are interested in, and of which we will calculate the heat balance.

Exercise 2.2.1 *What are the units of HeatCont? Given we are more interested in temperature and not heat content, why don't we calculate the balance of temperature immediately?*

A temperature difference between layers j and (j+1) or (j-1) causes heat to flow between these layers. We will define this flow as positive if its direction is upward and negative if it is downward. The heat flow depends on the temperature difference between subsequent layers as well as the distance between the nodes of the layers (ThickL) and the conductivity (Conduc) over the flow distance. Flow thus becomes:

$$\text{Flow}^i_j = \frac{\text{Conduc} * (T^i_j - T^i_{j-1})}{\text{ThickL}} \qquad (2.5)$$

Exercise 2.2.2 *What are the units of heat flow?*

This equation can be applied to all layers, except the uppermost and lowermost layers. We will impose the same forcing function on the uppermost layer as we did in the analytical solution for the surface boundary. We presume the surface temperature to vary according to:

$$\text{SurfTp} = \text{SurfAvTp} + \text{SurfAmp} * \sin(2 * \pi * \text{Time}) \quad (2.6)$$

The heat flow between the uppermost layer and the surface then equals

$$\text{Flow}_1^i = \frac{\text{Conduc} * (T_1^i - \text{SurfTp}^i)}{0.5 * \text{ThickL}} \quad (2.7)$$

Exercise 2.2.3 *Why do we divide by 0.5*ThickL in (2.7)?*

The impact of daily temperature fluctuations does not propagate deeply into the soil. Changes in temperature decrease with depth. We will therefore state that the heat flow through the lowest layer of the column equals zero.

$$\text{Flow}_{\text{NrLayer+1}}^i = 0 \quad (2.8)$$

The flow balance returns a net flow for each layer, which is equal to:

$$\text{NFlow}_j^i = \text{Flow}_{\text{j+1}}^i - \text{Flow}_j^i \quad (2.9)$$

Integrating net flow over time step dt then gives us an approach for the change of heat content of layer j over the time step. When we add this change to the content at time i, we get the heat content at time i+1:

$$\text{HeatCont}_j^{\text{i+1}} = \text{HeatCont}_j^i + \text{NFlow}_j^i * \text{dt} \quad (2.10)$$

We now rewrite equation 2.4 to calculate the temperature of layer j:

$$T_j^{\text{i+1}} = \text{HeatCont}_j^{\text{i+1}} / (\text{HeatCap} * \text{ThickL}) \quad (2.11)$$

Putting all these calculations in the right order enables us to calculate the temperature for each layer at time t+Δt and, subsequently, at all times t+2Δt, t+3Δt and so on.

Obtaining a numerical solution in this manner still involves some boundary conditions. We need the initial temperatures throughout the soil column and the temperature or heat fluxes at the surface.

The quality of the numerical solution largely depends on the sizes of Δz and Δt. Taking them infinitely small will cause the numerical solution to converge to the analytical one. This, however, would take an infinite time to compute. Generally, we want to find some balance between desired accuracy and length of computation time.

2.3 Writing a simulation program in MATLAB

A major use of computers in science is in finding numerical solutions to problems that have no analytical solutions. In this section we will look at a MATLAB code that implements the numerical solution of the heat diffusion equation of the previous section. You can find the code enclosed at the end of this chapter.

The first part of the program consists of an initialization of all relevant constants and variables. (Though, strictly speaking, MATLAB does not require the initialization of variables if they have the value '0'. For reason of good convention with other programming languages, however, we will initialize all variables anyway. Furthermore, it improves the readability of your model.)

Exercise 2.3.1 *Describe the characteristics of constants, control constants, system parameters, control variables and system variables. What are their differences?*

The second large block in the code is often referred to as the dynamic block. In this block, calculations of the dynamic simulation are carried out. The solution to the heat equation we derived in the previous section starts by calculating the rates of change (NFlow) of the HeatCont for each layer. It uses this information at time t to compute the HeatCont at time $t+\Delta t$ from which the temperature in each layer at time $t+\Delta t$ is derived. Note that the order of the statements is relevant: first rates, then states. Change in temperature throughout the soil is caused by the driving variation in SurfTp. Its value must be known before we can compute other variables.

The dynamic block contains three loops inside a larger time loop. We will look at the inner three loops first. The first one computes the heat flow for each layer. The second one computes the net flow. Calculating NFlow for layer k requires knowledge of Flow for layer k as well as Flow for layer k+1. It easy to see that it is not possible to calculate NFlow in the first loop, right after Flow. The last loop integrates NFlow over a time step dt to calculate the new HeatCont at time t+dt and subsequently the new temperature.

Using only the inner three loops (or single statements), the model will run only one time step. Since our interest is in a larger time range, we use the outer 'while'-loop. Every time this loop executes, the variable 'Time' is incremented with one time step and the new temperature is calculated at the incremented time. This way the time evolution of temperature is calculated throughout the soil column.

Exercise 2.3.2 *The calculations of Flow, NFlow, HeatCont and Temp still need to be completed in the program. Complete these, run the program and check whether the result is what you expect by using plot(StoreTemp(:,2:26)).*

Exercise 2.3.3 *Flow of all layers is calculated in what we call an explicit for-loop. MATLAB does not require use of an explicit for-loop to complete this action. The remark in the code shows an alternative notation. Replace the for-loop by this single statement, and also alter the other for-loops. Save your program under a different name and check whether the result is the same as in the former program.*

Exercise 2.3.4 *Now check the speed of calculation. You can determine the calculation time by inserting 'tic' at the beginning of your program and 'toc' at the end (see help tic and toc). Compare the calculation time of both programs; do they differ? (Run the programs a few times as some difference can occur for identical runs.)*

Exercise 2.3.5 *Motivate what you prefer, the explicit loop or the single statement?*

A third block generates output files or the visualization of the model results. This block can occur at the end of the program if we are only interested in the final result. However, often we are interested in dynamic results throughout the simulation period. Then, this block must be located inside the time-loop (as it is now).

Exercise 2.3.6 *Now remove the '%'s in the visualization part of the code and run the model. Explain what happens if, first, you remove the statement pause(0.2), and second, when you replace this statement with 'drawnow'.*

Exercise 2.3.7 *Now display the simulation results with 'plot' instead of 'imagesc'. Which method "plot" or "imagesc" do you find more informative? Include this motivation as a remark in your program. Also apply this visualization method and add the required legend, axes etc.*

The main reason to use numerical solutions instead of analytical solution is the possibility to apply any variation in dynamic boundary conditions, which we have not done until now. Now we make the boundary conditions more flexible in two steps. First, we change the initial conditions to non-constant, then we introduce the measured temporal variation of the surface temperature.

Exercise 2.3.8 *soiltemp.txt, (download from Blackboard) contains the measured soil temperature at every 2 cm (at the middle of the layers of the column). You can import data with:*

```
% Look up the file soiltemp.txt in the current directory
importdata('soiltemp.txt');
```

a) Make a quick visualization and interpret the soil temperature.
b) Now replace the initialization of soil temperature in the program by the measured temperature.
c) Make a plot with depth on the vertical axis and temperature on the horizontal axis and interpret the results. Make sure the y-axis is reversed (i.e. that the deepest layer is at the bottom) using set(gca,'Ydir','reverse') after the plot command. Let this plot change in time while you run the program. Report only the last plot.
d) Now make a new plot, analogous to the previous one (with the same axes), but instead displays all the temperature profiles for the whole simulation period in the same figure.

The soil surface temperature was also measured for a period from day 230 to 243 *(0 = 1 Jan 00:00)*. The measurements are contained in the file Stemp.txt (first column is time [day], second column is Surface temperature [^{o}C]). If you have a look at the numbers of the first 10 rows, you will see that the soil temperature is not measured at times you need them in your program. Thus you must interpolate. You will be guided step by step.

Exercise 2.3.9
a) Make a quick visualization of the soil surface temperature.
b) Load Stemp.txt in the correct place in the program.
c) Now interpolate SurfTp at Time with a linear interpolation (see doc interp1).
d) Don't forget to change StartTime and EndTime.
e) Interpret the result.

```
%
% Model HeatFlow
% Numerical solution of the 1D heat diffusion equation
% Jan, 2003 by Tessa van Wijnen and Willem Bouten
%
%%%%%%%%%%%%%% INITIALISATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all

% Constants
Omega = 2*pi;

% Control Constants
StartTime= 0;                               %   [day]
EndTime = 1;                                %   [day]
dt = 0.001;                                 %   [day]
PlotStep = 0.02;                            %   [day]
StoreStep = 0.01;                           %   [day]
ThickL = 0.02;                              %   [m]
NrLayer = 25;

% System Parameters
HeatCap = 2.5e5;                            %   [J/oC/m3]
Conduc = 8640;                              %   [J/oC/m/dag]
SurfAvTp = 20;                              %   [oC]
SurfAmp = 10;                               %   [oC]

% Control Variables
NrStore = 1;
PlotTime = PlotStep;
StoreTime = StoreStep;
Time = StartTime;

% System Variables
SurfTp = SurfAvTp;
Flow = zeros(NrLayer+1,1);
Temp = ones(NrLayer,1)*SurfAvTp;
HeatCont(1:NrLayer) = Temp(1:NrLayer)*HeatCap*ThickL;

%%%%%%%%%%%%%% DYNAMIC %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
while Time < EndTime

  % Dynamic boundary conditions
  SurfTp = SurfAvTp+(SurfAmp*sin(Omega*Time)) ;
  Flow(1) = Conduc*(Temp(1)-SurfTp)/(0.5*ThickL);
  Flow(NrLayer+1) = 0;

  % Rates: Flow and net flow calculations
  for k = 2 : NrLayer
     Flow(k) = %%%%%%% INTRODUCE THIS STATEMENT %%%%%%%%%%%%%%
  end
  % this for-loop could also be replaced by:
  % Flow(2:NrLayer) = (Temp(2:NrLayer)-Temp(1:NrLayer-1))*Conduc/ThickL;

  for k = 1 : NrLayer
    NFlow(k) = %%%%%%% INTRODUCE THIS STATEMENT %%%%%%%%%%%%%%
  end

  % States: Integration step
  for k = 1 : NrLayer
    HeatCont(k) = %%%% INTRODUCE THIS STATEMENT %%%%%%%%%%%%%%
    Temp(k) = %%%%%%%% INTRODUCE THIS STATEMENT %%%%%%%%%%%%%%
  end
  Time = Time+dt;

  %%%%%%%%%%%%%%%% STORING AND VISUALIZATION %%%%%%%%%%%%%%%%%
  StoreTime = StoreTime - dt;
  if StoreTime <= 0
    StoreTemp(NrStore,:) = [Time Temp];
    NrStore = NrStore+1;
    StoreTime = StoreStep;
  end %if StoreTime <= 0

  % These statements should be activated for dynamic visualization
  % PlotTime = PlotTime-dt;
  % if PlotTime <= 0
  %   ImTemp= rot90(Temp,3);
  %   imagesc(ImTemp,[10 30]),colorbar
  %   xlabel(Time)
  %   pause(0.2)
  %   PlotTime = PlotStep;
  % end % if PlotTime <= 0
end %while Time < EndTime

% save the simulation results in a file
save storetemp.txt StoreTemp -ascii
```