Master CompuPhys, University of Franche-Comté, semester 1
Academic year 2019-2020

Numerical project: Heat diffusion in homogeneous and inhomogenous media

Author: Vincent Ballenegger
Institut UTINAM, Université de Franche-Comté (vincent.ballenegger@univ-fcomte.fr)

1 Project description

The aim of this project is to develop a program to simulate numerically the diffusion of heat inside a metallic bar that is heated on one side. This bar can be homogeneous (made of a pure metal) or inhomogeneous because built from an inhomogeneous mixture of two metals. In the case of a homogeneous bar, the obtained numerical solution to the heat diffusion equation can be compared with detailed experimental measurements on aluminium and copper bars. No such measurement exist for *inhomogeneous* bars. This project has two goals:

1. Solve numerically the heat diffusion equation

$$\frac{\partial}{\partial t}T(x,t) = \alpha \frac{\partial^2}{\partial x^2}T(x,t) \tag{1}$$

   (where α is the thermal diffusivity[1]) in a homogeneous bar to obtain the time-dependent temperature profile $T(x,t)$ along the bar and compare it with experimental heating curves for Al and Cu bars. Discrepancies between the theoretical model and the experimental data should be discussed and improvements in the employed model can be suggested and tested.

2. Determine, by solving numerically the heat diffusion equation in 3 dimensions, the thermal behaviour of inhomogeneous bars in relation with their microscopic structure (grain size). The inhomogeneous bars are built by compressing a mixture of copper and iron "grains" and can be characterised by the typical size of the Cu and Fe domains. Iron happens to have a much lower thermal diffusivity than copper.

Two models for the bar will be considered: a simple 1-dimensional model at first and then a more general 3-dimensional model. The bar is assumed to have a rectangular cross-section. The 1d model is expected to be sufficient for simulating the heat diffusion in homogenous bars, while the 3d model will be necessary to study the influence of the size and distribution of the metallic domains. In this project, the heat equation will be solved numerically by using the **finite-difference method**, where the equation is discretized on a grid in time (with grid spacing Δt) and in space (with grid spacings $\Delta x, \Delta y$ and Δz).

2 Experiments on heat diffusion in Al and Cu bars

The experimental setup is shown in Figs. 1, 2 and 3. A metallic bar has an electric resistance integrated on its left side that heats the bar by dissipating a power of 12 W ($\pm$0.4 W). The bar is isolated laterally by a plastic encaging (thickness 5 mm) that surrounds it, apart for its lower surface which rests on a circuit board with 8 temperatures sensors. The sealing between the plastic encaging and the circuit board is air tight.The last 5.3 centimetres of the bar are not isolated and are located under a fan which can be turned on or off at will. The bar with its insulating plastic is put inside a closed metallic box. A computer program records, every second, the temperatures of the 8 sensors that

[1] See https://en.wikipedia.org/wiki/Thermal_diffusivity

are in contact with the lower surface of the bar. Fig.5 shows the temperatures of the 7 last sensors as function of time in various cases: Al and Cu bars, with or without air flow at the other bar end. The corresponding raw data files for these temperature curves can can be downloaded from `http://perso.utinam.cnrs.fr/~ballenegger/Docs/Experimental_data.zip`.

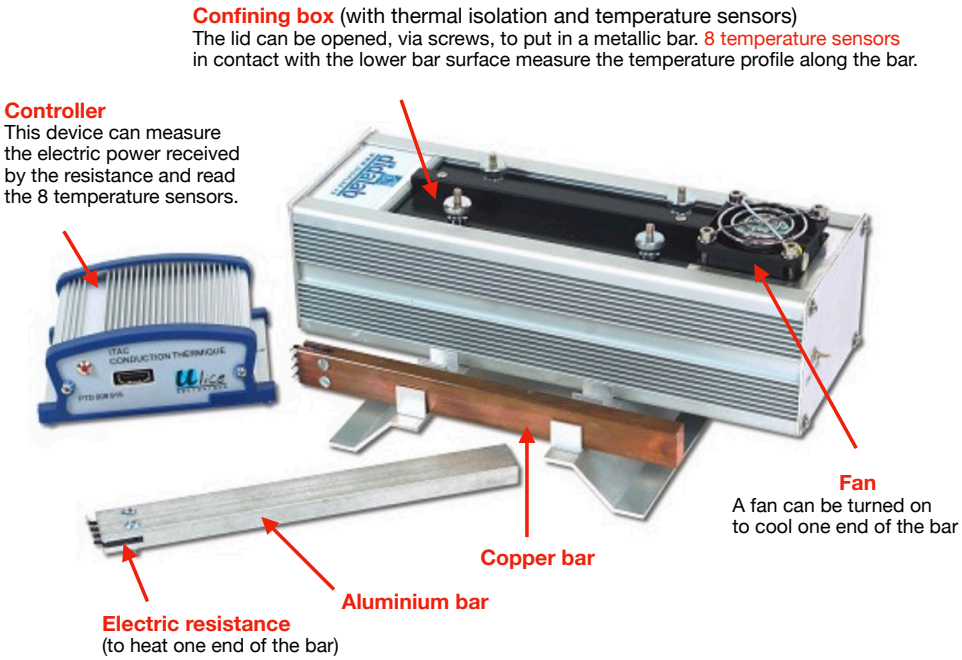

Figure 1: Experimental setup to measure the diffusion of heat in metallic bars.

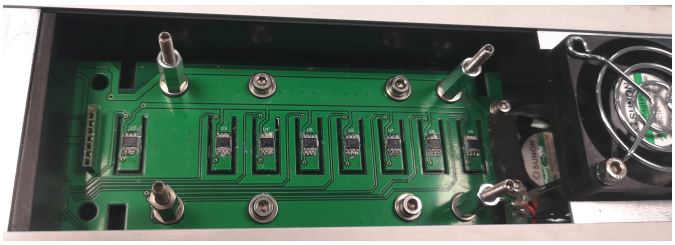

Figure 2: Circuit board inside the box showing the temperature sensors.

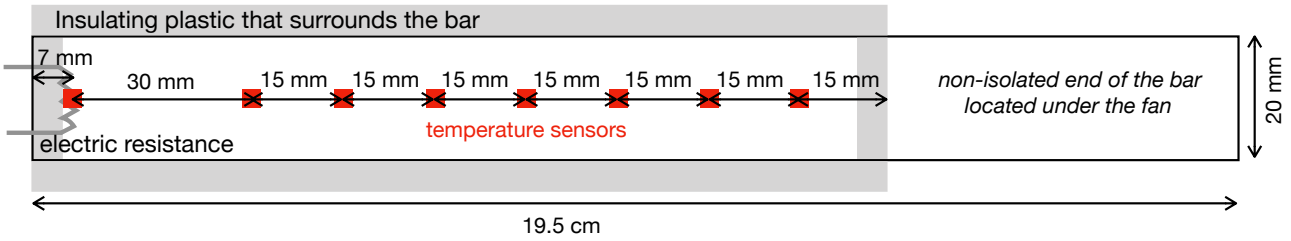

Figure 3: Top view of the bar showing its dimensions, its isolation, and the positions of the temperature sensors. The bar volume is $195 \times 20 \times 10$ mm^3.

3 Program specifications

Development environnement The programs must be written in the **Python** language. Numerical librairies like numpy, scipy, matplotlib, etc. can be used if needed. The plots can be created with matplotlib or with the plotting program Veusz.
Documentation can be found at
`https://docs.python.org/`
`http://perso.utinam.cnrs.fr/~ballenegger/Docs/Python_Cheatsheet_2018.pdf`
`https://matplotlib.org`
`https://veusz.github.io`

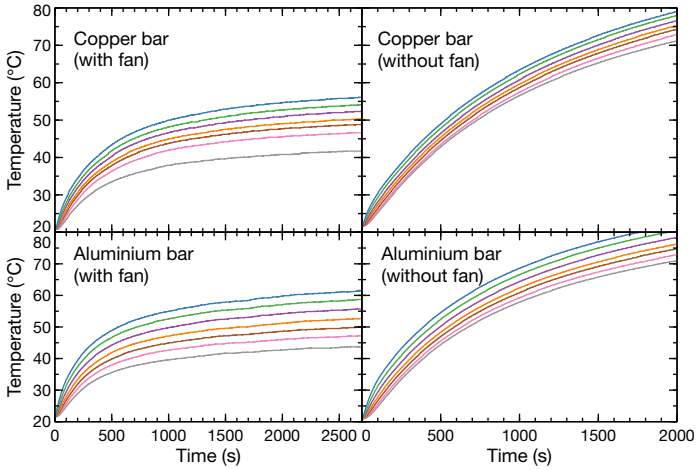

Figure 4: Experimental results for the temperature of the last 7 sensors for a copper bar (first row of plots) or an aluminium bar (second row) heated on one side by a power of 12 W; In the first column of plots, a fan cools down the other end of the bar, whereas this fan is switched off for the second column of plots.

Version alpha

At this stage, only homogeneous bars made of pure aluminium or copper are considered. The development starts by writing a program to compute numerically the solution of the heat diffusion equation for a 1-dimensional model of the bar using the **forward Euler scheme** (see §6.5). A generalisation to a 3d model is performed in a second stage. The programs should have the following features.

• **Modelisation in 1d** (program `heat_1d.py`)

1. (System definition) The initial temperature of the bar (in °C) is specified by a function `T_initial(x)` defined in the code. At each end of the bar, that is at $x = 0$ or at $x = L$, the boundary condition can be of the Dirichlet or of the Neumann type :

   - A Dirichlet boundary condition correspond to keeping the temperature at the boundary fixed at a constant value (denoted by T_0^{BC} at $x = 0$, resp. T_L^{BC} at $x = L$).
   - A Neumann boundary condition corresponds to keeping the derivative of the temperature fixed at a constant value. Physically this corresponds to fixing the value of the heat flux density[2] $\phi(x,t) = -\lambda \frac{\partial T}{\partial x}$ at the border (the constant value is denoted by ϕ_0^{BC} at $x = 0$, resp. ϕ_L^{BC} at $x = L$). In the previous formula, which is Fourier's law, λ is the thermal conductivity of the medium.

   A Neumann boundary condition at $x = 0$ can be used to account for the heat flux produced by the electric resistance. A bar that is isolated at its other end can be modelled by setting a Neumann boundary condition $\phi(\text{end point}, t) = 0$. The case where air is blown on this end can be modelled at first by setting a Dirichlet boundary condition, i.e. by assuming that the air flow keeps constant the temperature of the end of the bar.

2. The program starts by printing on the screen all the relevant parameters of the simulation, for instance:

```
System:
Bar length:  ...  cm
Bar width (length along y axis):  2 cm
Bar height (length along z axis):  1 cm
Material:  copper
-> thermal diffusivity:  alpha = ...
Left boundary condition:  [Neumann] phi(x = 0, all t) = ...  W/m2
Right boundary condition:  [Dirichlet] T(x = L, all t) = ...  degrees Celsius
Initial temperature profile T(x,0):  [uniform] T = ...  degrees
Simulation parameters:
Total simulation time:  T = ...  (s)
Nb of points for the discretisation in time:  M = ...
Nb of points for the discretisation in space:  N = ...
-> time step:  delta_t = ...
```

[2] We recall that the heat flux ϕS is the energy per second crossing an orthogonal surface with area S. The SI units of heat flux density ϕ are $\mathrm{J\,s^{-1}\,m^{-2}} = \mathrm{W\,m^{-2}}$.

```
-> spatial grid spacing:  deltaX = ...
etc.
```

All the relevant system and simulation parameters must be defined in easily identifiable variables at the beginning of the code. Reasonable default values should be assigned to these variables for all allowed uses of the program. Do **not** ask the user to enter values for the parameters via a python command `input("Enter a value:  ")`. Switching from one type of boundary condition (BC) to another [at any of the two ends of the bar] should be as easy as changing the value of a variable from 0 (representing the use of a Dirichlet BC) to 1 (representing the use of a Neuman BC), knowing that default values for $T_0^{\rm BC}$, $T_L^{\rm BC}$, $j_0^{\rm BC}$, $j_L^{\rm BC}$ must be defined in the code. These 4 values are of course not used all at the same time !

3. The program uses the **forward Euler scheme** (see §6.5) to solve numerically the heat diffusion equation. The calculations are performed internally using **reduced units** (see §6.3). The user can choose if the final results are shown in SI or in reduced units. Since the forward Euler scheme is stable only when the Fourier number $r = \alpha\Delta t/\Delta x^2 < 1/2$ (the reader is encouraged to verify that the numerical calculation does indeed diverge when $r \geq 1/2$), the program checks the value of r and stops immediately if its value is too large.

4. The program produces two main outputs:
   - a color plot of the temperature inside the bar as function of time, similar to the one in Fig.5. This plot is saved in a PDF file.
   - a text file (named for instance `Tsensors_sim.txt`) containing the temperatures, as function of simulation time, at the 8 points where the temperature sensors are placed in the experimental setup.

5. The program is validated by comparing the numerical solution to the analytical solution

$$T(x,t) = 1 + \sin(\pi x)\exp(-\alpha\pi^2 t) \qquad \text{(in reduced units)} \tag{2}$$

to the diffusion equation (10) for initial condition $T_{\rm initial}(x) = 1+\sin(\pi x)$ and Dirichlet boundary conditions

$$T(0,t) = T(L,t) = 1 \qquad \text{(for all } t\text{)}. \tag{3}$$

The program computes the relative error

$$\epsilon = \frac{|T - T_{\rm analytic}|}{T_{\rm analytic}} \tag{4}$$

of the numerical solution at the mid-point $x = 1/2$ when the temperature of this point should have reached the value $3/2$ according to the analytic solution, i.e. after a simulation time τ defined by the condition $T(1/2,\tau) = 3/2$.

6. (Comparison with experiments) Plots are produced to compare the simulation results, stored in file `Tsensors_sim.txt`, with the corresponding experimental data. *Are there systematic deviations? Why? How could the model be improved, while keeping a 1-dimensional representation of the bar?*

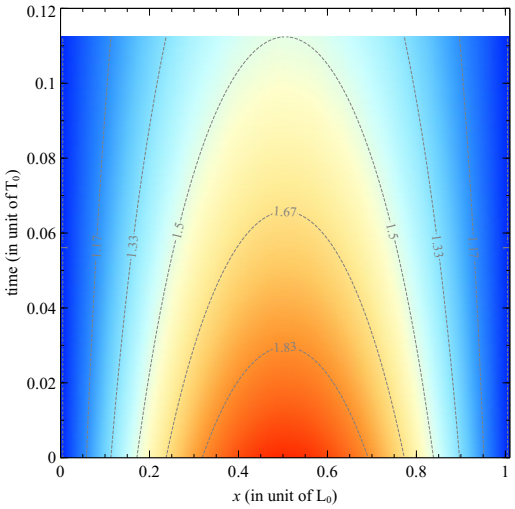

Figure 5: Temperature profile (in reduced units) along the bar as function of time according to the analytical solution (2) when $\alpha = 1$. The temperatures are shown by a color scale ranging from $T = 1$ (blue) to $T = 2$ (red).

• **Modelisation in 3d** (program `heat_3d.py`)

7. A program is written to solve, using the forward Euler scheme, the heat diffusion equation in a homogeneous bar for a grid modelled in 3 dimensions, with grid size N_x, N_y, N_z. Since no heat loss is modelled at this stage, the results of this program should coincide with those of the program `heat_1d.py` if the energy flows only along the x direction.

Version beta

• **Modelisation in 1d**

8. The program `heat_1d.py` (or a new program `heat_1d_implicit.py`) can solve the heat diffusion equation using the **backward Euler scheme** (see §6.5). The resulting tridiagonal linear system of equations is solved using the Thomas algorithm (see §7))
   Since this scheme is stable for any value of the Fourier parameter r, the restriction $r < 1/2$, introduced in point 3. of the alpha version, can now be lifted.
   → *What scheme (forward or backward Euler) is more efficient in terms of time of computation when solving numerically the heat equation at high accuracy? And at low accuracy?*
   For this comparison, one can assume that both scheme provide in general the same accuracy when using the same time step Δt.

9. A Robin (or convection) boundary condition can be applied at any end of the bar.[3] A Robin condition at $x = L$ corresponds to setting $\phi(L,t) = -\lambda \frac{\partial T}{\partial x} = h(T(L,t) - T_{\text{room}})$ for all t, where the coefficient h (in $\mathrm{W\,m^{-2}\,K^{-1}}$) is the convection coefficient[4] at the bar-air interface.
   → *Can we get a better agreement with some experimental curves when using a Robin boundary condition ? With what value of the convection coefficient?*

10. To check whether the heat loss by electromagnetic radiation is important in the experimental setup, the program has an option to account for such losses.

11. (Numerical accuracy) The accuracy of the two integration schemes (forward or backward Euler) is compared by plotting the measured relative error (4) as function of the time step Δt (for Δt in the range 10^{-7} to 10^{-1} in reduced units), the grid spacing Δx being kept fixed.
   A log-log scale should be used for this plot. Interpretation of the results?

• **Modelisation in 3d** (program `heat_3d.py`)

12. The program is able to simulate inhomogeneous bars made up of a Cu and Fe domains with some typical domain size d (this size is the same for the two kinds of metals). For simplicity, these domains are assumed to occupy small cubic or parallelepiped volumes. The spatial distribution of these domains can be either random or alternating (i.e. arranged as in a checkerboard).

13. A plot showing the average temperature of the end surface of the bar as function of time, for the different microscopic structures considered, is produced and interpreted. The bar is subjected to the same conditions as in the experimental setup, with the fan turned off.
   → *What are the effective diffusivities of the inhomogeneous bars?*

Version gold

A gold version of the programs or of the scientific analysis can have features like

- Integration of the heat equation using the Crank-Nicolson scheme.

[3] Since a Robin condition involves both the value of the function and its derivative at the boundary, it is sometimes called a mixed boundary condition. See eq. (24) for the implementation of a boundary condition that sets the derivative $\frac{\partial T}{\partial x}\big|_L = T'(L)$ of the function to a given value (denoted u'_N) which can depend itself on T.

[4] See `https://en.wikipedia.org/wiki/Heat_transfer_coefficient`

- The numerical error ϵ of the various schemes obey to

$$\epsilon = a\Delta t + b(\Delta x)^2 \qquad \text{(forward Euler scheme)}$$
$$\epsilon = c\Delta t + d(\Delta x)^2 \qquad \text{(backward Euler scheme)}$$
$$\epsilon = e(\Delta t)^2 + f(\Delta x)^2 \qquad \text{(Crank-Nicolson scheme)}$$

  An analysis of the measured relative errors of the numerical solution is performed to determine numerically the value of the coefficients a, b, c, d, e and f.

- Scientific analysis of the diffusion of heat in inhomogeneous media. For a bar made with 50% copper and 50% iron, is it possible to get an effective heat diffusivity that is equal to 75% of the one of copper by making a suitable choice of domain size and arrangement?

- The program for heat diffusion in 3 dimensions (`heat_3d.py`) is able to account for heat loss on the surface by radiation or convection.

4 Development constraints

- The Thomas algorithm, which will be used to solve tridiagonal linear system of equations, must be programmed in a function that is well decoupled from the main program. This function should be tested on a simple system, for instance

$$\begin{bmatrix} 1 & 8 & 0 & 0 \\ 2 & 3 & 7 & 0 \\ 0 & 4 & 5 & 9 \\ 0 & 0 & 6 & 10 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \tag{5}$$

  whose solution is $(x_1, x_2, x_3, x_4) = (94, -7, -13, 23) \cdot \frac{1}{38}$. The program that runs this test must be provided.

- Care should be taken to ensure that the various programs are easily readable by a human.

5 Scientific exploitation

In addition to the programs to be written in accordance with the previous specifications, a scientific exploitation of the numerical results must be realised. Several scientific questions have been asked in particular in the section detailing the program specifications. All plots must of course be correctly labelled and the observed phenomena must be briefly commented and interpreted. The size d of the Cu or Fe domains should be chosen so that interesting phenomena are observed while keeping a reasonable computation cost for the simulation.

6 Physical notions and methods necessary to the project

6.1 Fourier's law and the heat equation

The heat flux density is denoted by $\vec{\phi}(\vec{r}, t)$. In SI its units are watts per square metre ($\mathrm{W\,m^{-2}}$). Fourier's law states that

$$\vec{\phi}(\vec{r}, t) = -\lambda \vec{\nabla} T(\vec{r}, t), \tag{6}$$

where T is the temperature and λ (sometimes denoted k) is the thermal conductivity.

Heat diffusion equation For diffusion along a single direction, this equation reads (see e.g. the derivation in wikipedia)

$$\frac{\partial}{\partial t}T(x,t) = \alpha \frac{\partial^2}{\partial x^2}T(x,t) + \tilde{f}(x,t) \tag{7}$$

where $\alpha = \lambda/(\rho c_p)$ is the thermal diffusivity, c_p is the specific heat capacity and ρ is the density (mass per unit volume). The source and sink term is defined as $\tilde{f}(x,t) = f(x,t)/(\rho c_p)$ where $f(x,t)$ is the amount of heat lost or received per unit time and per unit volume (the SI unit of f is $\mathrm{W\,m^{-3}}$).

In 3 dimensions, for a medium with a thermal conductivity $\lambda = \lambda(\vec{r})$ that can be non-uniform, the heat equation becomes

$$\frac{\partial}{\partial t}T = \left[\mathrm{div}(\lambda(\vec{r})\vec{\nabla}T) + f\right]\frac{1}{\rho c_p}. \tag{8}$$

or more explicitely, in cartesian coordinates,

$$\frac{\partial}{\partial t}T = \left[\frac{\partial}{\partial x}\left(\lambda\frac{\partial}{\partial x}T\right) + \frac{\partial}{\partial y}\left(\lambda\frac{\partial}{\partial y}T\right) + \frac{\partial}{\partial z}\left(\lambda\frac{\partial}{\partial z}T\right) + f\right]\frac{1}{\rho c_p}. \tag{9}$$

In the medium is homogeneous (uniform thermal conductivity λ), the heat equation simplifies into $\frac{\partial}{\partial t}T = \alpha\nabla^2 T + \tilde{f}$, that is, in cartesian coordinates,

$$\frac{\partial}{\partial t}T(x,y,z,t) = \alpha\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)T(x,y,z,t) + \tilde{f}(x,y,z,t). \tag{10}$$

6.2 Black body radiation

The total power radiated by a body at thermal equilibrium at temperature T is $A\sigma T^4$, where σ is the Stefan-Boltzmann constant and A is the surface area of the body. An object receives also energy from the electromagnetic radiation emitted by the surrounding objects that are at temperature T_{room}, i.e. it receives the power $A\sigma T_{\mathrm{room}}^4$. Such energy gain or loss can be accounted for the source/sink term $\tilde{f}(x,t)$ in the diffusion equation (7).

6.3 Reduced units

Let T_0 be a characteristic temperature, L_0 a characteristic length and t_0 a characteristic time. In terms of the reduced quantities

$$T^* = T/T_0, \qquad x^* = x/L_0, \qquad t^* = t/t_0, \tag{11}$$

the heat equation (7) becomes

$$\frac{\partial}{\partial t^*}T^* = \alpha^*\frac{\partial^2}{\partial x^{*2}}T^* + \tilde{f}^* \tag{12}$$

where

$$\alpha^* = \alpha\frac{t_0}{L_0^2} \qquad \text{and} \qquad \tilde{f}^* = \frac{t_0}{\rho c_p T_0}f. \tag{13}$$

are dimensionless coefficients. Notice that by choosing suitably the unit of length L_0 and the unit of time t_0, one can always set $\alpha^* = 1$. The program should use, internally, reduced units though we have dropped the notation with a star in all following formulae.

6.4 Discretisation via finite differences

A partial differential equation, like the heat equation, can be discretized on a grid. Let us consider a one-dimensional grid with $N+1$ points and grid spacing Δx. The i^{th} point has cartesian coordinate $x_i = i\Delta x$ $(i = 0, ..., N)$. We denote by u_i the value of a function $u(x)$ at the point x_i: $u_i = u(x_i) = u(i\Delta x)$.

The first derivative of a function $u(x)$ can be estimated numerically by one of the following formulas

$$\left.\frac{\partial u}{\partial x}\right|_i \simeq \begin{cases} \dfrac{u_{i+1}-u_i}{\Delta x} & \text{(forward finite difference)} \quad (14a) \\ \dfrac{u_i-u_{i-1}}{\Delta x} & \text{(backward finite difference)} \quad (14b) \\ \dfrac{u_{i+1}-u_{i-1}}{2\Delta x} & \text{(centred finite difference of step } 2\Delta x) \quad (14c) \\ \dfrac{u_{i+\frac{1}{2}}-u_{i-\frac{1}{2}}}{\Delta x} & \text{(centred finite difference of step } \Delta x \text{ involving off-grid points)} \quad (14d) \end{cases}$$

The second derivative can be approximated by applying twice eq. (14d):

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i \simeq \frac{\left.\frac{\partial u}{\partial x}\right|_{i+1/2} - \left.\frac{\partial u}{\partial x}\right|_{i-1/2}}{\Delta x} \simeq \frac{u_{i-1}-2u_i+u_{i+1}}{(\Delta x)^2}. \quad (15)$$

6.5 Integration schemes

When discretising the heat diffusion equation with finite differences, several schemes can be devised to perform the time integration of this equation.
The time is discretised on a grid made with $M+1$ points and grid spacing Δt. We let $t_n = n\Delta t$. The value of a function $u(x,t)$ at position and time (x_i, t_n) is denoted by

$$u_i^n = u(x_i, t_n) = u(i\Delta x, n\Delta t), \qquad (\text{for } i = 0, ..., N \text{ and } n = 0, ..., M), \quad (16)$$

i.e. spatial positions are indicated in subscript and time is denoted in superscript.

• Forward Euler scheme

Applying formula (14a) to the time derivative and formula (15) to the second derivative $\frac{\partial^2 T}{\partial x^2}$ in eq. (7) leads to

$$\frac{u_i^{n+1}-u_i^n}{\Delta t} = \alpha \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{(\Delta x)^2} + \tilde{f}_i^n \quad (17)$$

where $u = T$ denotes the temperature and $\tilde{f} = f/(\rho c_p) = f\alpha/\lambda$. Therefore:

$$\boxed{u_i^{n+1} = u_i^n + r\left(u_{i-1}^n - 2u_i^n + u_{i+1}^n\right) + \tilde{f}_i^n \Delta t} \qquad (i = 1, ..., N-1) \quad (18)$$

where

$$r = \alpha \frac{\Delta t}{(\Delta x)^2}. \quad (19)$$

Given the temperature u_i^n $(i = 0, ..., N)$ of the medium at time t_n, the temperatures at time $t_{n+1} = t_n + \Delta t$ can be calculated by applying eq. (18) at all internal points of the grid. The solution to the heat equation is therefore constructed by applying repeatedly this time propagation step, starting from the initial temperature profile $u_i^0 = T_{\text{initial}}(x_i)$.

This integration scheme is **explicit**: the temperatures in the medium at the next time step are given by explicit formulas that depend only on the temperatures at the previous time step (and on the heat generation or loss term f if it doesn't vanish) . This scheme is stable only if $r < 1/2$. The generalisation of this scheme to the diffusion equation in 3 dimensions is straightforward:

$$\begin{aligned} u_{i,j,k}^{n+1} = u_{i,j,k}^n &+ r(u_{i-1,j,k}^n - 2u_{i,j,k}^n + u_{i+1,j,k}^n) \\ &+ r(u_{i,j-1,k}^n - 2u_{i,j,k}^n + u_{i,j+1,k}^n) \\ &+ r(u_{i,j,k-1}^n - 2u_{i,j,k}^n + u_{i,j,k+1}^n) + \tilde{f}_{i,j,k}^n \Delta t \qquad (i = 1, ..., N-1) \end{aligned} \quad (20)$$

Boundary conditions The end points $x = 0$ and $x = L$ need to be treated separately. If Dirichlet boundary conditions are used, the situation is trivial since the value of the function at the end points (u_0 and u_N) stay unchanged at all times. To implement a Neumann boundary condition at (say) $x = 0$, i.e. a fixed value u'_0 for the gradient $\frac{\partial u}{\partial x}\big|_{x=0}$, one could write, using eq. (14a),

$$\frac{u_1^n - u_0^n}{\Delta x} = u'_0, \tag{21}$$

($u'_0 = 0$ for an insulating boundary condition) and calculate therefore the new value of the function at the boundary via $u_0^{n+1} = u_1^{n+1} - u'_0 \Delta x$. This is however suboptimal and a better way of implementing a Neumann boundary condition is to apply eq. (14c) at the boundary point $i = 0$, i.e.

$$\frac{u_1^n - u_{-1}^n}{2\Delta x} = u'_0 \quad , \qquad \Longrightarrow u_{-1}^n = u_1^n - u'_0 2\Delta x \tag{22}$$

together with eq. (18) for $i = 0$ in which u_{-1}^n is replaced by the above value. This leads to

$$\boxed{u_0^{n+1} = u_0^n + r\left(2u_1^n - 2u'_0\Delta x - 2u_0^n\right) + \tilde{f}_0^n \Delta t} \qquad \text{(Neumann BC for point } i = 0) \tag{23}$$

Similarly

$$\boxed{u_N^{n+1} = u_N^n + r\left(2u_{N-1}^n + 2u'_N\Delta x - 2u_N^n\right) + \tilde{f}_N^n \Delta t} \qquad \text{(Neumann BC for point } i = N) \tag{24}$$

Eqs. (23) and (24) apply to end points along the x axis in a one-dimensional model of diffusion. The generalisation to end points along the y or z axis is straightforward and is performed as in the generalisation of eq. (18) into eq. (20).

In the case of an inhomogeneous system, one applies eq. (18) [or eq. (20) in 3d] with the local thermal diffusivity. For a point i located at the interface between two media with thermal diffusivities α_1 and α_2, eq. (18) is generalised into

$$u_i^{n+1} = u_i^n + r_1 u_{i-1}^n - (r_1 + r_2)u_i^n + r_2 u_{i+1}^n + \tilde{f}_i^n \Delta t \tag{25}$$

where $r_i = \alpha_i \Delta t/(\Delta x)^2$.

• Backward Euler scheme

In this scheme, one discretises the heat eq. (7) at time t_{n+1} by applying formula (15) to the second derivative and formula (14b) to the time derivative. This leads to the equation

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{(\Delta x)^2} + \tilde{f}_i^{n+1}. \tag{26}$$

Moving the unknowns quantities to the left hand side of the equation gives

$$u_i^{n+1} - r\left(u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}\right) = u_i^n + \tilde{f}_i^{n+1}\Delta t \qquad (i = 1, ..., N-1) \tag{27}$$

If Dirichlet boundary conditions are used, the temperatures $u_0 = u_0^0$ and $u_N = u_N^0$ at the end points are known at all times. Eq. (27) provides then a system of $N - 1$ linear equations for the $N - 1$ unknowns temperatures $\{u_i^{n+1}\}$ of the inner points. This scheme is called **implicit** because equations must be solved to find the "new" values of the function. Denoting these unknown temperatures by a vector

$$\vec{u}_{\text{int}} = \begin{bmatrix} u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}, \tag{28}$$

the system of equations (27) can be written in matrix form

$$\begin{bmatrix} 1+2r & -r & & & 0 \\ -r & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -r \\ 0 & & & -r & 1+2r \end{bmatrix} \vec{u}_{\text{int}}^{n+1} = \vec{u}_{\text{int}}^{n} + \vec{\tilde{f}}_{\text{int}}^{n+1}\Delta t + r \begin{bmatrix} u_0 \\ 0 \\ \vdots \\ 0 \\ u_N \end{bmatrix} \tag{29}$$

In this scheme, the temperatures in the medium at time $t_{n+1} = t_n + \Delta t$ are thus calculated from the temperatures at time t_n by solving this linear system of equations.
If Neumann boundary conditions are used and implemented in the form

$$u_0^{n+1} = u_1^{n+1} - u'_1\Delta x \tag{30}$$
$$u_N^{n+1} = u_{N-1}^{n+1} + u'_N\Delta x \tag{31}$$

with u'_1 and u'_N fixed (=0 for an insulating boundary condition), the system of equations becomes

$$\begin{bmatrix} 1+r & -r & & & 0 \\ -r & 1+2r & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & 1+2r & -r \\ 0 & & & -r & 1+r \end{bmatrix} \cdot \vec{u}_{\text{int}}^{n+1} = \vec{u}_{\text{int}}^{n} + \vec{\tilde{f}}_{\text{int}}^{n+1}\Delta t + \begin{bmatrix} -ru'_1\Delta x \\ 0 \\ \vdots \\ 0 \\ ru'_N\Delta x \end{bmatrix} \tag{32}$$

The backward Euler scheme is stable for any value of r.

• The Crank-Nicolson method

The Crank-Nicolson method is a combination of the forward Euler method at time t_n and of the backward Euler method at time t_{n+1} and is explained in wikipedia. For convenience, we write below the time propagation formula for the Crank-Nicolson method in matrix form in the case of Dirichlet boundary conditions:

$$\begin{bmatrix} 1+2\tilde{r} & -\tilde{r} & & & 0 \\ -\tilde{r} & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -\tilde{r} \\ 0 & & & -\tilde{r} & 1+2\tilde{r} \end{bmatrix} \vec{u}_{\text{int}}^{n+1} = \vec{u}_{\text{int}}^{n} + \frac{\Delta t}{2}(\vec{\tilde{f}}_{\text{int}}^{n} + \vec{\tilde{f}}_{\text{int}}^{n+1}) + \tilde{r} \begin{bmatrix} u_0 \\ 0 \\ \vdots \\ 0 \\ u_N \end{bmatrix} + \tilde{r} \begin{bmatrix} u_0^n - 2u_1^n + u_2^n \\ \vdots \\ u_{i-1}^n - 2u_i^n + u_{i+1}^n \\ \vdots \\ u_{N-2}^n - 2u_{N-1}^n + u_N^n \end{bmatrix} \tag{33}$$

where $\tilde{r} = r/2$.

7 Algorithms

The Python package `numpy` contains generic functions, for instance `numpy.solve()`, to solve linear systems of equations. Since the system of interest [see eq. (29)] is tridiagonal, it is better to use a method optimised for tridiagonal systems, namely the Thomas algorithm, which is well explained in wikipedia.