THERMESH

A Finite Element Code for 1D Heat Conduction

Wouter Grouve

University of Twente, Faculty of Engineering Technology
Mechanics of Solids, Surfaces and Systems, Chair of Production Technology

1 INTRODUCTION

bla

2 DERIVATION

2.1 Governing partial differential equation and weak form

In the case of 1D heat conduction, the governing partial equation reads:

$$\rho c_p \frac{\partial T}{\partial z} - k_z \frac{\partial^2 T}{\partial z^2} - \dot{Q} = 0 \tag{1}$$

with T the temperature, ρc_p the volumetric heat capacity, k_z the thermal conductivity, and $\dot{Q}$ an internal source or sink, e.g. due to a phase transformation. The partial differential equation is subject to boundary conditions at the two ends of the domain, which can be of the Dirichlet type:

$$T(0,t) = T_{\text{left}}(t), \qquad T(L,t) = T_{\text{right}}(t),$$

or of the Neumann type:

$$-k_z \left.\frac{\partial T}{\partial z}\right|_{z=0} = q_{\text{left}}(t), \qquad -k_z \left.\frac{\partial T}{\partial z}\right|_{z=L} = q_{\text{right}}(t),$$

with T_{left} and T_{right} an imposed temperature and q_{left} and q_{right} an imposed heat flux density. The latter could be either directly imposed $\hat{q}$, for example as a result of laser heating, or be the result of convection:

$$q = h(T_\infty - T),$$

in which h and T_∞ are the heat transfer coefficient and the far field temperature, or the result of thermal radiation:

$$q = \epsilon\sigma(T_\infty^4 - T^4),$$

where ϵ is the surface emissivity and σ is Stefan's constant.

We can approximate the solution $T(z,t)$ of the partial differential equation using the weighted residual method:

$$\int_L w \left(\rho c_p \frac{\partial T}{\partial t} - k \frac{\partial^2 T}{\partial z^2} - \dot{Q} \right) dz = 0, \tag{2}$$

with w a weighting function.

2.2 Discretization in space

The domain in N is now split up in smaller elements, as is indicated in Figure 1 which shows an element (e) of length ℓ that is bounded by the nodes i and j. The weighted residual (Equation 2) can now be rewritten as:

$$\sum_{e=1}^{N} \int_{\ell} w \left(\rho c_p \frac{\partial T}{\partial t} - k \frac{\partial^2 T}{\partial z^2} - \dot{Q} \right) dz = 0. \tag{3}$$

We will now make sure that the weighted residual vanishes for each element. Further, we get rid of the second derivative with respect to z in the second term using integration by parts:

$$\int_{\ell} w \rho c_p \frac{\partial T}{\partial t} dz + \int_{\ell} \frac{dw}{dz} k \frac{\partial T}{\partial z} dz - wk \frac{\partial T}{\partial z} \Bigg|_{z_i}^{z_j} - \int_{\ell} w \dot{Q} dz = 0,$$

with z_i and z_j the element end points. Realizing that the third term represents the flux ($q = -k\partial T/\partial z$), the equation can be rewritten as:

$$\int_{\ell} w \rho c_p \frac{\partial T}{\partial t} dz + \int_{\ell} \frac{dw}{dz} k \frac{\partial T}{\partial z} dz = \int_{\ell} w \dot{Q} dz - wq \Big|_{z_i}^{z_j}. \tag{4}$$

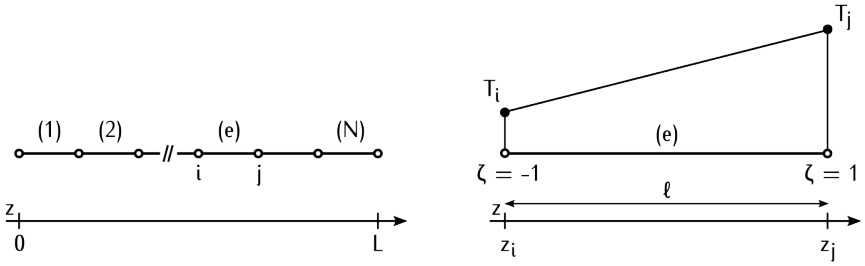

Figure 1: Left: Elements in 1D domain. Right: Definition of local coordinate system and linear interpolation of temperature.

2.2.1 *Linear shape functions*

The temperature inside an element is approximated as a linear interpolation between the bounding node temperatures as:

$$T(\zeta) = N_i(\zeta) T_i + N_j(\zeta) T_j, \tag{5}$$

with ζ a local coordinate, as is illustrated in Figure 1, defined as:

$$\zeta(z) = 2 \frac{z - (z_j - z_i)/2}{\ell},$$

while the two shape functions are:

$$N_i(\zeta) = \frac{1 - \zeta}{2} \quad \text{and} \quad N_j(\zeta) = \frac{1 + \zeta}{2}. \tag{6}$$

The spatial derivative of the temperature with respect to z can now be calculated as:

$$\frac{\partial T}{\partial z} = \frac{\partial T}{\partial \zeta} \frac{\partial \zeta}{\partial z},$$

with:

$$\frac{\partial \zeta}{\partial z} = \frac{2}{\ell} \quad \text{and} \quad \frac{\partial T}{\partial \zeta} = \frac{T_j - T_i}{2},$$

such that:

$$\frac{\partial T}{\partial z} = \frac{T_j - T_i}{\ell}, \tag{7}$$

which also intuitively makes sense of course. Further, noting that the shape functions do not depend on time, we can rewrite the time derivative as:

$$\frac{\partial T}{\partial t} = N_i(\zeta)\frac{\partial T_i}{\partial t} + N_j(\zeta)\frac{\partial T_j}{\partial t}. \tag{8}$$

Following the Galerkin method, we choose our weighting function w to be our shape functions. The equation for the weighted residual for an element (Equation 4) can now be rewritten as:

$$\int_\ell N_k \rho c_p \frac{\partial T}{\partial t} dz + \int_\ell \frac{dN_k}{dz} k \frac{\partial T}{\partial z} dz = \int_\ell N_k \dot{Q} dz - N_k q \Big|_{z_i}^{z_j} \quad \text{for: } k = 1, 2. \tag{9}$$

with N_i and N_j the two shape functions as defined in Equation 6.

Matrix-vector notation

Before evaluating the integrals, we first rewrite the expressions from the previous section into a matrix-vector form. Starting with Equation 5, which can be written as:

$$T(\zeta) = \mathbf{NT},$$

in which:

$$\mathbf{N} = [N_i(\zeta), N_j(\zeta)] \quad \text{and} \quad \mathbf{T} = \begin{Bmatrix} T_i \\ T_j \end{Bmatrix}$$

The spatial derivative of the temperature (Equation 7) yields:

$$\frac{\partial T}{\partial z} = \frac{\partial T}{\partial \zeta}\frac{\partial \zeta}{\partial z} = \frac{2}{\ell}\frac{\partial \mathbf{NT}}{\partial \zeta} = \frac{2}{\ell}\frac{\partial \mathbf{N}}{\partial \zeta}\mathbf{T} = \mathbf{BT},$$

with:

$$\mathbf{B} = \frac{2}{\ell}\frac{\partial \mathbf{N}}{\partial \zeta} = \frac{2}{\ell}\left[\frac{\partial N_i}{\partial \zeta}, \frac{\partial N_j}{\partial \zeta}\right] = \left[-\frac{1}{\ell}, \frac{1}{\ell}\right],$$

while the time-derivative of the temperature can be rewritten as:

$$\frac{\partial T}{\partial t} = \mathbf{N}\dot{\mathbf{T}}.$$

Further, for convenience, we will write our weighting functions as:

$$w = \mathbf{N}^T = \begin{Bmatrix} N_i \\ N_j \end{Bmatrix}.$$

We can now evaluate the integrals, starting with the first term:

$$\int_\ell w \rho c_p \frac{\partial T}{\partial t} dz = \rho c_p \int_\ell \mathbf{N}^T \mathbf{N} dz\, \dot{\mathbf{T}}.$$

We can rewrite this integral in terms of ζ, by making use of the derivative of ζ with respect to z:

$$\frac{d\zeta}{dz} = \frac{2}{\ell} \quad \rightarrow \quad dz = \frac{\ell}{2} d\zeta,$$

such that:

$$\rho c_p \int_\ell \mathbf{N}^T \mathbf{N} dz\, \dot{\mathbf{T}} = \frac{\ell \rho c_p}{2} \int_{-1}^{1} \mathbf{N}^T \mathbf{N} d\zeta\, \dot{\mathbf{T}} = \mathbf{C}\dot{\mathbf{T}}, \tag{10}$$

with:

$$\mathbf{C} = \frac{\ell \rho c_p}{2} \int_{-1}^{1} \mathbf{N}^T \mathbf{N} d\zeta = \frac{\ell \rho c_p}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

In the same manner, the second term yields:

$$\int_\ell \frac{dN_k}{dz} k \frac{\partial T}{\partial z} dz = \frac{\ell k}{2} \int_{-1}^{1} \mathbf{B}^T \mathbf{B} d\zeta \, \mathbf{T} = \mathbf{K}\mathbf{T}, \tag{11}$$

with:

$$\mathbf{K} = \frac{\ell k}{2} \int_{-1}^{1} \mathbf{B}^T \mathbf{B} d\zeta = \frac{k}{\ell} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

The first term on the right hand side yields:

$$\int_\ell \mathbf{N}^T \dot{Q} dz = \frac{\ell}{2} \int_{-1}^{1} \mathbf{N} d\zeta \, \dot{Q} = \frac{\dot{Q}\ell}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix},$$

with $\dot{Q}$ the heat source term for the element between nodes i and j. The second term with the heat flux q on the boundary is first expanded to include both a direct heat flux $\hat{q}$ and a flux due to convection:

$$q = \hat{q} + h(T_\infty - T),$$

which yields:

$$N_k q \Big|_{z_i}^{z_j} = N_k \hat{q} \Big|_{z_i}^{z_j} + N_k h (T_\infty - T) \Big|_{z_i}^{z_j}.$$

The term with the direct heat flux $\hat{q}$ is evaluated as:

$$N_k \hat{q} \Big|_{z_i}^{z_j} = \begin{Bmatrix} N_i(z_j) q_j - N_i(z_i) \hat{q}_i \\ N_j(z_j) q_j - N_j(z_i) \hat{q}_i \end{Bmatrix} = \begin{Bmatrix} -\hat{q}_i \\ \hat{q}_j \end{Bmatrix},$$

with $\hat{q}_k$ the heat flux on the k-th node. The convective term can be accounted for using a stiffness matrix for convection:

$$N_k h T \Big|_{z_i}^{z_j} = \mathbf{H}\mathbf{T} \quad with: \quad \mathbf{H} = h \begin{bmatrix} N_i N_i & N_i N_j \\ N_j N_i & N_j N_j \end{bmatrix}, \tag{12}$$

and an additional term in the force vector:

$$N_k h T_\infty \Big|_{z_i}^{z_j} = h \begin{Bmatrix} -T_{\infty,i} \\ T_{\infty,j} \end{Bmatrix}.$$

As an example for the stiffness matrix $\mathbf{H}$, in case of a convective boundary condition at the j-th node, where $N_i = 0$, this term would evaluate as:

$$\mathbf{H} = \begin{bmatrix} N_i N_i & N_i N_j \\ N_j N_i & N_j N_j \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix},$$

which intuitively makes sense. The force vector is now combined as:

$$\mathbf{f} = \int_\ell N_k \dot{Q} dz - N_k q \Big|_{z_i}^{z_j} - N_k h T_\infty \Big|_{z_i}^{z_j} = \frac{\dot{Q}\ell}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} + \begin{Bmatrix} \hat{q}_i \\ -\hat{q}_j \end{Bmatrix} + h \begin{Bmatrix} T_{\infty,i} \\ -T_{\infty,j} \end{Bmatrix}. \tag{13}$$

The final element equation can now be assembled from by substituting Equations 10, 11, 12 and 13 in Equation 9:

$$\mathbf{C}\dot{\mathbf{T}} + (\mathbf{K} + \mathbf{H})\mathbf{T} = \mathbf{f}.$$

With the local damping and stiffness matrices determined for each element, we can assemble the global matrices using the node locations and element connectivity in the global system.

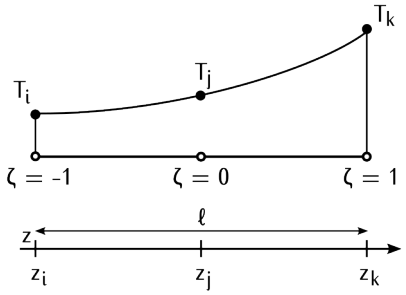

Figure 2: Definition of a regular local coordinate system and quadratic interpolation of temperature.

2.2.2 *Quadratic shape functions*

In the case of quadratic shape functions, the temperature inside an element is approximated as:

$$T(\zeta) = N_i(\zeta)T_i + N_j(\zeta)T_j + N_k(\zeta)T_k, \tag{14}$$

with reference to 2 for the node locations. The three shape functions are now defined as:

$$N_i(\zeta) = -\frac{1}{2}\zeta(1-\zeta) \quad \text{and} \quad N_j(\zeta) = 1-\zeta^2 \quad \text{and} \quad N_j(\zeta) = \frac{1}{2}\zeta(1+\zeta), \tag{15}$$

or:

$$\mathbf{N} = \left[-\frac{1}{2}\zeta(1-\zeta), 1-\zeta^2, \frac{1}{2}\zeta(1+\zeta)\right].$$

The mapping between the local coordinate ζ and the global coordinate x is achieved by:

$$z(\zeta) = N_i(\zeta)z_i + N_j(\zeta)z_j + N_i(\zeta)z_k.$$

Here, for convenience, we will consider a regular element which means that:

$$z_j = \frac{z_i + z_k}{2}.$$

The Jacobian is now evaluated as:

$$\frac{dz}{d\zeta} = (\zeta - \frac{1}{2})z_i - 2\zeta z_j + (\zeta + \frac{1}{2})z_k = \frac{\ell}{2}.$$

The temperature gradient with respect to z can be written as:

$$\frac{dT}{dz} = \frac{d\zeta}{dz}\frac{\partial T}{\partial \zeta} = \frac{2}{\ell}\frac{\partial \mathbf{N}}{\partial \zeta}\mathbf{T} = \mathbf{BT},$$

with:

$$\mathbf{B} = \frac{2}{\ell}\frac{\partial \mathbf{N}}{\partial \zeta} = \frac{2}{\ell}\left[\frac{\partial N_i}{\partial \zeta}, \frac{\partial N_j}{\partial \zeta}, \frac{\partial N_k}{\partial \zeta}\right] = \left[\frac{2\zeta-1}{\ell}, -\frac{4\zeta}{\ell}, \frac{2\zeta+1}{\ell}\right].$$

Now we can derive the damping matrix $\mathbf{C}$ and the stiffness matrix $\mathbf{K}$, in the same manner as we have done for the linear shape functions:

$$\mathbf{C} = \frac{\ell \rho c_p}{2}\int_{-1}^{1} \mathbf{N}^T\mathbf{N}d\zeta = \frac{\ell \rho c_p}{30}\begin{bmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{bmatrix},$$

$$\mathbf{K} = \frac{\ell k}{2}\int_{-1}^{1} \mathbf{B}^T\mathbf{B}d\zeta = \frac{k}{3\ell}\begin{bmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{bmatrix}.$$

2.3 Temporal discretization

The final step is to integrate the equation with time. For this purpose, we will discretize the temporal variable will using the so-called Θ-method:

$$\mathbf{C}\frac{\mathbf{T}_{n+1}-\mathbf{T}_n}{\Delta t}+(1-\Theta)(\mathbf{K}+\mathbf{H})\mathbf{T}_n+\Theta(\mathbf{K}+\mathbf{H})\mathbf{T}_{n+1}=(1-\Theta)\mathbf{f}_n+\Theta\mathbf{f}_{n+1}, \qquad (16)$$

where $\Theta \in [0,1]$. Common values of Θ are:

$$\begin{aligned}\Theta &= 0, && \text{(Explit Euler)}\\ \Theta &= 1/2, && \text{(Crank Nicolson)}\\ \Theta &= 1, && \text{(Implicit Euler).}\end{aligned}$$

Equation 16 can be rearranged as:

$$\Big(\mathbf{C}+\Delta t\Theta(\mathbf{K}+\mathbf{H})\Big)\mathbf{T}_{n+1}=\Big(\mathbf{C}-\Delta t(1-\Theta)(\mathbf{K}+\mathbf{H})\Big)\mathbf{T}_n+\Delta t(1-\Theta)\mathbf{f}_n+\Delta t\Theta\mathbf{f}_{n+1}.$$

3 VALIDATION

The finite elements derived here were implemented in an object-oriented Python code called `thermesh`, which can be found on the author's Github page. Please note, with the focus on readability, the code is far from optimized. Nevertheless, I believe it should still be more than fast enough for most problems. This section presents three short validation cases to show that the code is implemented correctly. Please note that, although all presented cases use linear elements, the code was also validated for the quadratic elements Details on material properties and dimensions used for each case can be found in the respective Python file.

3.1 Step temperature at boundary

Consider a domain of length L with a uniform initial temperature T_0. For $t > 0$ the temperature at one end is raised to a value of T_{end}, while the other end is kept at the initial temperature:

$$\begin{aligned}T(x,0) &= T_0\\ T(0,t) &= T_0\\ T(L,t) &= T_{end}\end{aligned}$$

In case the initial temperature equals 0.0 °C , the analytical solution[1] yields:

$$T(x,t)=\frac{T_{end}x}{L}+\frac{2}{\pi}\sum_{N=1}^{\infty}\frac{T_{end}\cos N\pi}{N}\sin\left(\frac{N\pi x}{L}\right)\exp\left(-\alpha N^2\pi^2 t/L^2\right),$$

with $\alpha = k/\rho c_p$ the thermal diffusivity. The left graph in Figure 3 shows the temperature distribution at different times. Code listing 1 illustrates how to solve this problem using `thermesh`. The right graphs in Figure 3 shows the finite element solution for 10 linear elements of equal length. Good comparison is obtained between the numerical and analytical solution. The code for this comparison is available in `step_change.py`.

1 The Mathematics of Diffusion, Crank, 1975, pp 49-50.

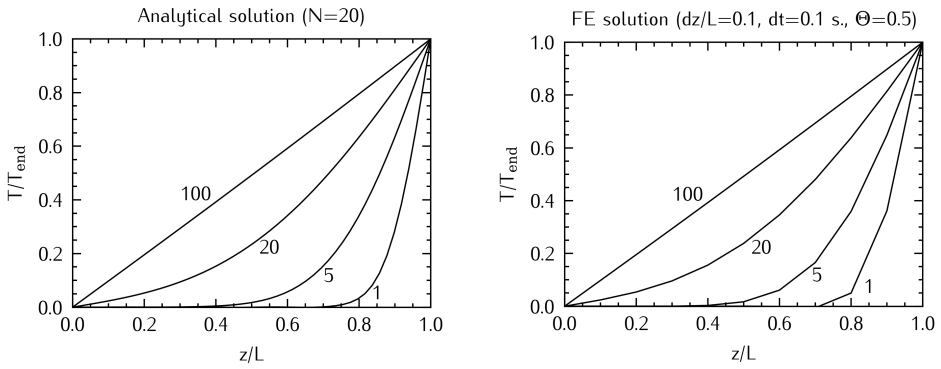

Figure 3: Comparison of the analytical and FE solution at different times for a step change in temperature at one end. The numbers in the graphs indicate the time in seconds.

Listing 1: Thermesh example for a step change at one end.

```
import numpy as np
import thermesh as tm

# Domain information
L = 0.01
k, rho, cp = 0.72, 1560, 1450
cpeek = tm.isothermal_model(k, rho, cp)  # constitutive model

# Mesh generation using linear elements
nn = 11  # number of nodes
z = np.linspace(0, L, nn)  # node locations
mesh = tm.Mesh(z, tm.LinearElement)

# Boundary conditions
bc = [{"T": 0.0},  # T on the left
      {"T": 1.0}]  # T on the right

# Domain generation and initialization
domain = tm.Domain(mesh, cpeek, bc)
domain.set_T(np.zeros(nn))

# Solve
solver = {"dt": 0.1, "t_end": 100.0, "theta": 0.5}  # settings
t, T = tm.solve_ht(domain, solver)
```

3.2 Constant heat flux at boundary of semi-infinite solid

Consider a semi-infinite domain with a uniform initial temperature. The domain is subjected to a constant heat flux $\hat{q}$ at the boundary which causes the temperature to increase. The analytical solution[2] for the temperature increase ΔT in the domain yields:

$$T(x,t) - T_0(x,t) = \Delta T(x,t) = \frac{\hat{q}}{k}\left[\sqrt{\frac{4\alpha t}{\pi}}\exp\left(-\frac{x^2}{4\alpha t}\right) - x\,\mathrm{erfc}\left(\frac{x}{2\sqrt{\alpha t}}\right)\right],$$

with erfc the complementary error function. The left graph in 4 shows the temperature increase at 2, 10 and 25 seconds. Code listing 2 illustrates the code to solve

2 Heat Transfer, Nillis & Klein, 2008, p 362.

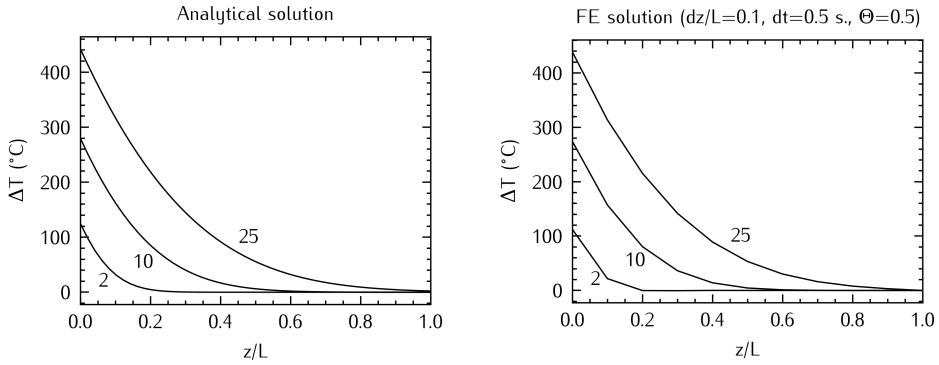

Figure 4: Comparison of the analytical and FE solution at different times in the case of a constant heat flux at one end. The numbers in the graphs indicate the time in seconds.

this problem using `thermesh`. Of course, one has to make sure that the domain is large enough (or the time short enough) to be considered a semi-infinite solid in this case. The right graphs in Figure 4 shows the finite element solution for 10 linear elements of equal length. Good comparison is obtained between the numerical and analytical solution. The code for this comparison is available in the Python file `constant_heat_flux.py`.

Listing 2: Thermesh example for a constant heat flux at one end.

```
# Clear all solution data
domain.clear()

# Set new boundary conditions
bc = [{"q": 1E5},  # q on the left
      {"T": 0.0}]  # T on the right

# Solve
solver = {"dt": 0.1, "t_end": 25.0, "theta": 0.5}  # settings
t, T = tm.solve_ht(domain, solver)
```

3.3 Convective boundary condition

In the last example, we again consider a semi-infinite domain with a uniform initial temperature. The domain is now subjected to a convective boundary condition with a heat transfer coefficient h and a far field temperature T_∞. The analytical solution[3] for the temperature increase ΔT in the domain yields:

$$\Delta T(x,t) = (T_\infty - T_0)\left[\operatorname{erfc}\left(\frac{x}{2\sqrt{\alpha t}}\right) - \exp\left(\frac{hx}{k} + \frac{h^2\alpha t}{k^2}\right)\operatorname{erfc}\left(\frac{x}{2\sqrt{\alpha t}} + \frac{h}{k}\sqrt{\alpha t}\right)\right].$$

The left graph in 5 shows the temperature increase at 2, 10 and 25 seconds. Code listing 3 illustrates the code to solve this problem using `thermesh`. The right graphs in Figure 4 shows the finite element solution for 10 linear elements of equal length. As can be seen, also here good comparison is obtained between the numerical and analytical solution. The code for this comparison is available in the Python file `convective_bc.py`.

3 See footnote 2.

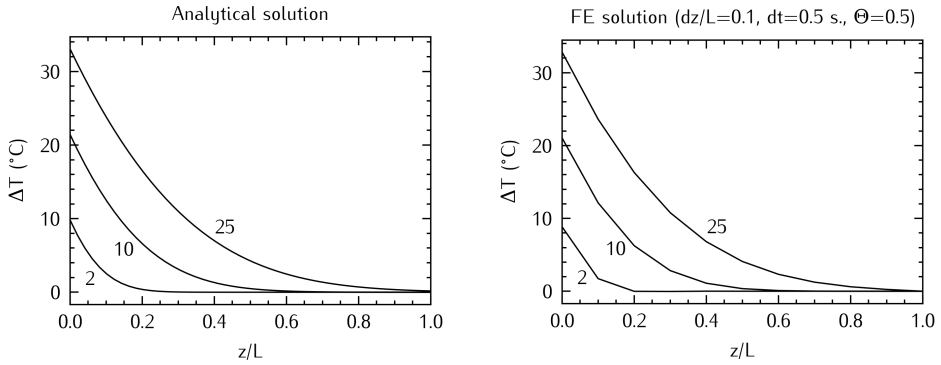

Figure 5: Comparison of the analytical and FE solution at different times in the case of a convective boundary condition at one end. The numbers in the graphs indicate the time in seconds.

Listing 3: Thermesh example for a constant heat flux at one end.

```
# Clear all solution data
domain.clear()

# Set new boundary conditions
bc = [{"h": 20,
       "T_inf": 400},  # h and T_inf on the left
      {"T": 0.0}]      # T on the right

# Solve
solver = {"dt": 0.1, "t_end": 25.0, "theta": 0.5}  # settings
t, T = tm.solve_ht(domain, solver)
```

4 FINAL REMARKS