The University of Nottingham

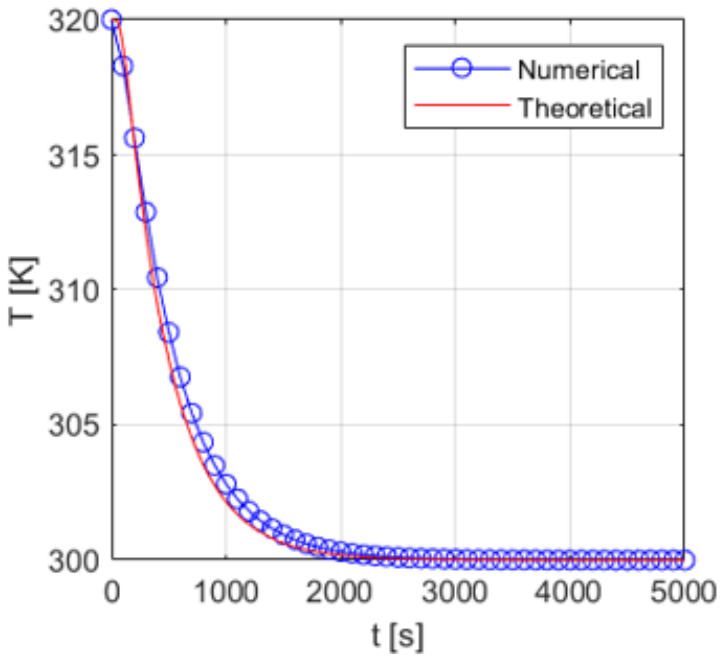
UNITED KINGDOM · CHINA · MALAYSIA

Computer Modelling Techniques

MMME3086 UNUK, 2023/24

Numerical Methods: Solution of the 1D unsteady diffusion equation

Author: Mirco Magnini
Office: Coates B100a
Email: mirco.magnini@nottingham.ac.uk



Contents

1 Introduction

In Lecture 1 we have seen how to discretise the diffusion equation in steady-state conditions using the finite-volume method. In this lecture, we extend this methodology to learn how to discretise the unsteady diffusion equation, that is, a diffusion equation where both spatial and temporal dependencies appear.

For simplicity, we will focus on a 1D case, although the methodology that we will outline is applicabile to 2D and 3D cases as well without (almost) any further complication. We will focus on the unsteady heat equation, although the same method applies to the time-dependent diffusion of any other variable, e.g. momentum, chemical species, etc.

The unsteady heat conduction equation in 1D takes the following form:

$$\frac{\partial(\rho c_p T)}{\partial t} = \frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right) + S(x,t,T), \qquad 0 < x < L, \quad t > 0 \tag{1}$$

where ρ is the density of the material (units: kg/m^3) and c_p is the constant pressure specific heat of the material (units: J/(kg $\cdot$ K)). Now the derivatives are partial derivatives because $T = T(x,t)$, and therefore Eq. (1) is a PDE. Note that the source term now can be a function of time too. Essentially, Eq. (1) states that the temperature at a specific location in the system will vary in time if an unbalance between the diffusion and the source term exists; otherwise, if diffusion and source terms balance out, the temporal derivative will be zero, which means that temperature does not vary in time and the system has reached steady-state. In diffusion problems, the time-derivative appears as a first-order derivative, and therefore the governing PDE is classified as a *parabolic* equation; recall Sec. 1 of Lecture 1.

We know from Lecture 1 that Eq. (1) needs two spatial boundary conditions (at $x = 0$ and $x = L$) to solve the problem. But what about the temporal boundary conditions, now that time enters the problem?

The solution of a parabolic equation has a very distinctive behaviour versus time: disturbances can only travel towards the $+t$ direction, and not backward ($-t$), i.e. time is a one-way coordinate. This means that Eq. (1) needs only one temporal boundary condition, which is the temperature field at the initial time $t = 0$, $T(x, t = 0)$. As such, the temporal boundary condition is also called *initial condition*. From the point of view of the numerical solution, the parabolic behaviour with time allows us to obtain the temporal evolution of the system using a *time marching* procedure: starting from an initial condition at $t = 0$, where $T(x)$ is known, the solution at any time $t > 0$ is found by marching the solution forward, from the time instant t to the next time instant $t + \Delta t$, without any need to know what happens at any future time instant $t > t + \Delta t$.

In brief, the solution procedure for the unsteady diffusion equation proceeds as follows:

- $t = 0$: We need a known initial condition $T(x, t = 0)$.
- $t = 1\Delta t$: We solve the linear system $\mathbf{A} \cdot \mathbf{T} = \mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ contain the information of the solution at $t = 0$. The discretisation equation for the unsteady problem (and related system) will be seen in the next sections. Solution of the system yields $T(x, t = 1\Delta t)$.

- $t = 2\Delta t$: $T(x, t = 1\Delta t)$ is our new initial condition. We solve the linear system $\mathbf{A} \cdot \mathbf{T} = \mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ now contain the information of the solution at $t = 1\Delta t$. Solution of the system yields $T(x, t = 2\Delta t)$.
- ...$t = t_{final}$: The procedure continues until time reaches the desired end value.

Therefore, there is an important difference from steady-state problems: while in steady problems we solve the linear system only once, for unsteady problems we must solve the linear system at every *time-step*.

Below, we will first show in Sec. 2 how to use the finite-volume method to discretise the 1D heat conduction equation in both space and time. Then, in Sections 3, 4 and 5, we will see three different temporal discretisation schemes. Finally, in Section 6, we conclude with a tutorial of the implementation of these numerical methods in Matlab.

Further reading: Patankar [1], Sec. 2.2.3.

2 Finite-volume method for unsteady PDE

In this section we see how to derive the discretisation for Eq. (1), according to the finite-volume method; for simplicity, we discard the source term. Since time is a one-way coordinate, we obtain the solution by marching in time from a given initial distribution of temperature. Thus, at a given time instant t where we know the values of T at each grid point, we want to find the new values of T at time $t + \Delta t$.

We proceed by integrating Eq. (1), discarding the source term, in both space and time. For an internal control volume centred in x_P, and at a given time instant t, we obtain:

$$\rho c_p \int_t^{t+\Delta t} \int_V \frac{\partial T}{\partial t} dV\, dt = \int_t^{t+\Delta t} \int_V \frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right) dV\, dt, \tag{2}$$

where for simplicity we are considering that both ρ and c_p are constant. Let's now manipulate the

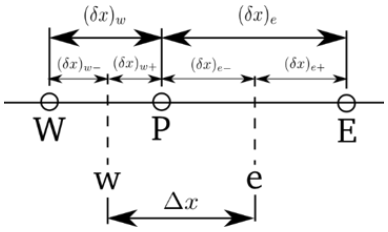


Figure 1: Generic internal CV and notation used in this document: circles refer to control volume centroids, denoted as P (central node), E (east neighbour), W (west neighbour); vertical dashed lines refer to the CV faces, with e and w denoting the locations of the east and west faces of the CV centred in P; δx identifies the distance between the CV centres, whereas $\Delta x = x_e - x_w$ is the distance between the east and west faces, which coincides with the CV width.

single terms, starting from the temporal term:

$$\rho c_p \int_t^{t+\Delta t} \int_V \frac{\partial T}{\partial t} dV\, dt = \rho c_p \int_w^e \left[\int_t^{t+\Delta t} \frac{\partial T}{\partial t} dt \right] A dx = \rho c_p \int_w^e \left[T \right]_t^{t+\Delta t} A dx = \rho c_p \left(T_P^1 - T_P^0 \right) A \Delta x. \tag{3}$$

What have we done here? First, we have swapped the order of the integrals and we have replaced dV with $dV = Adx$, with the spatial integration limits now becoming the west and east faces of the control volume, just like we did in the steady case; see Fig. 1 for the notation. Then, in the inner temporal integral, the dt at the numerator and the ∂t at the denominator cancel out and the integral of the differential ∂T becomes the function T evaluated at the integration boundaries. We call $T(t) \equiv T^0$ (old, known value) and $T(t+\Delta t) \equiv T^1$ (new, unknown value) and therefore $[T]_t^{t+\Delta t} = T^1 - T^0$. Finally, the spatial integral of $T^1 - T^0$ is solved by considering that the temperature within the control volume can be written in terms of T_P, so that $\int_w^e (T^1 - T^0) dx = \left(T_P^1 - T_P^0 \right) \Delta x$.

For the diffusion term, we follow the steady-state practice:

$$\int_t^{t+\Delta t} \int_V \frac{\partial}{\partial x} \left(\lambda \frac{\partial T}{\partial x} \right) dV\, dt = \int_t^{t+\Delta t} \int_w^e \left(\lambda \frac{\partial T}{\partial x} \right) A dx\, dt = \int_t^{t+\Delta t} \left[\lambda_e \frac{T_E - T_P}{\delta x_e} - \lambda_w \frac{T_P - T_W}{\delta x_w} \right] A dt. \tag{4}$$

Therefore, the semi-discretised version of Eq. (1) is (A is constant and cancels out):

$$\rho c_p \left(T_P^1 - T_P^0 \right) \Delta x = \int_t^{t+\Delta t} \left[\lambda_e \frac{T_E - T_P}{\delta x_e} - \lambda_w \frac{T_P - T_W}{\delta x_w} \right] dt, \tag{5}$$

where we call it "semi-discretised" because we do not know yet how to calculate the temporal integrals at the right-hand side. In order to integrate, we need to make an assumption of how $T(t)$ varies between t and $t + \Delta t$. In general, we can imagine that the integral of T_P (and T_W, T_E) between t and $t + \Delta t$ will be a function of the old value T_P^0 and new value T_P^1; the simplest function that we can use is a linear function of the kind:

$$\int_t^{t+\Delta t} T_P dt = \left[f T_P^1 + (1 - f) T_P^0 \right] \Delta t, \tag{6}$$

where f is a weighting factor between 0 and 1 and Δt is our time-step, which can be intended as a temporal-mesh size. Depending on the choice of f, we have the following time-discretisation schemes:

- $f = 0$, **explicit scheme**. This assumes that $T_P \equiv T_P^0$ between t and $t + \Delta t$, as sketched in Fig. 2, so that $\int_t^{t+\Delta t} T_P dt = T_P^0 \Delta t$. This scheme is first-order accurate and conditionally stable, as we will see later on.

- $f = 1$, **fully-implicit scheme**. This assumes that $T_P \equiv T_P^1$ between t and $t + \Delta t$, as sketched in Fig. 2, so that $\int_t^{t+\Delta t} T_P dt = T_P^1 \Delta t$. This scheme is first-order accurate and unconditionally stable.

- $f = 1/2$, **Crank-Nicolson scheme**. This assumes that T_P varies linearly from T_P^0 to T_P^1 between t and $t + \Delta t$, as sketched in Fig. 2, so that $\int_t^{t+\Delta t} T_P dt = \left[0.5 T_P^0 + 0.5 T_P^1 \right] \Delta t$. This scheme is second-order accurate and unconditionally stable, but may give rise to oscillations of the solution (will see in the tutorial).

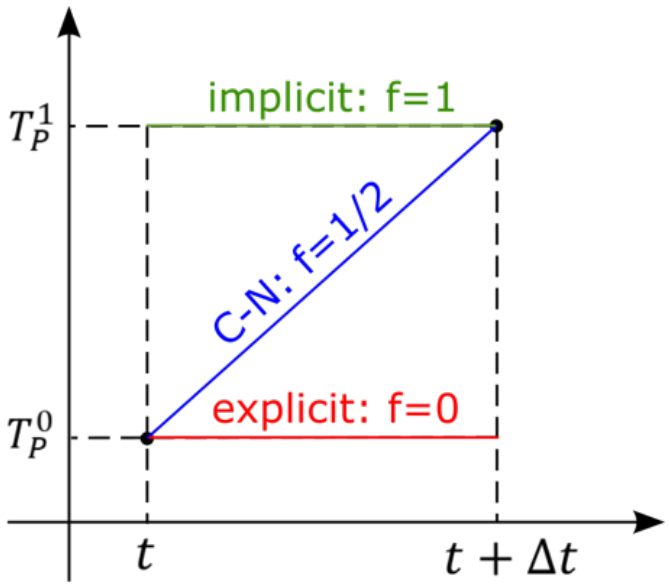


Figure 2: Variation of temperature with time for the three time-discretisation schemes.

Below, we will derive the general discretisation equation by retaining f as a parameter, and we will treat the implications of the specific choice of time scheme in later sections.

We further develop the right-hand side of Eq. (5) by replacing the temporal integral of T_P, T_W, and T_E with the expression in Eq. (6), this yields:

$$\rho c_p \left(T_P^1 - T_P^0\right)\Delta x = \\ = \left\{ f\left[\lambda_e \frac{T_E^1 - T_P^1}{\delta x_e} - \lambda_w \frac{T_P^1 - T_W^1}{\delta x_w}\right] + (1-f)\left[\lambda_e \frac{T_E^0 - T_P^0}{\delta x_e} - \lambda_w \frac{T_P^0 - T_W^0}{\delta x_w}\right] \right\} \Delta t, \tag{7}$$

where the right-hand side includes the contribution of the diffusion term at the new time level (superscripts 1), weighted by f, and that of the old time level (superscripts 0), weighted by $(1-f)$. Now, in this equation T_W^1, T_P^1 and T_E^1 represent the unknowns, whereas T_W^0, T_P^0 and T_E^0 are the old temperatures and are known. Therefore, separating the unknown terms on the left-hand side and the known terms on the right-hand side, and dropping the superscript 1, we obtain:

$$-f\frac{\lambda_w}{\delta x_w} T_W + \left[\rho c_p \frac{\Delta x}{\Delta t} + f\left(\frac{\lambda_w}{\delta x_w} + \frac{\lambda_e}{\delta x_e}\right)\right] T_P - f\frac{\lambda_e}{\delta x_e} T_E = \\ = \rho c_p \frac{\Delta x}{\Delta t} T_P^0 + (1-f)\left[\frac{\lambda_w}{\delta x_w} T_W^0 + \frac{\lambda_e}{\delta x_e} T_E^0 - \left(\frac{\lambda_w}{\delta x_w} + \frac{\lambda_e}{\delta x_e}\right) T_P^0\right], \tag{8}$$

which can be written in compact form as follows:

$$a_W T_W + a_P T_P + a_E T_E = b, \tag{9}$$

with the coefficients:

$$a_W = -f\frac{\lambda_w}{\delta x_w}, \quad a_P = \rho c_p \frac{\Delta x}{\Delta t} + f\left(\frac{\lambda_w}{\delta x_w} + \frac{\lambda_e}{\delta x_e}\right), \quad a_E = -f\frac{\lambda_e}{\delta x_e}, \tag{10}$$

and source term:

$$b = \rho c_p \frac{\Delta x}{\Delta t} T_P^0 + (1-f)\left[\frac{\lambda_w}{\delta x_w} T_W^0 + \frac{\lambda_e}{\delta x_e} T_E^0 - \left(\frac{\lambda_w}{\delta x_w} + \frac{\lambda_e}{\delta x_e}\right) T_P^0\right]. \tag{11}$$

Note that b contains old temperature values T^0, which are the temperatures evaluated at the previous time instant t. Therefore, as you march the solution in time, these values will need to be updated after each time-step.

The discretisation equation for 1D unsteady heat conduction, Eq. (9), looks the same as that for the steady case, the difference is in the coefficients and source term. Note that all the information about the previous time instant temperatures T^0 enter in the source term, as these are known. You can imagine that assembling the discretisation equations for the n control volumes of the domain, leads once again to a linear system $\mathbf{A} \cdot \mathbf{T} = \mathbf{B}$, where $\mathbf{A}$ is the $n \times n$ tridiagonal matrix of the coefficients, $\mathbf{T}$ is the $n \times 1$ vector of unknowns and $\mathbf{B}$ is the $n \times 1$ vector of known terms.

However, this time the linear system must be solved at each time step, i.e. for $t = \Delta t, t = 2\Delta t, t = 3\Delta t$, and so on until the final time instant desired. At each time instant, T^0 will represent the solution at the previous time instant, and thus the vector $\mathbf{B}$ will need to be updated before solving the linear system. Things will be more clear with the Matlab tutorials at the end of this document.

The derivation above was done for an internal control volume, but what about boundary control volumes? If you have Dirichlet conditions at both boundaries, then the discretisation equations for the boundary control volumes are straightforward and are the same as those for the steady case, see Sec. 4.4.1 in Lecture 1; the coefficient b does not change during the temporal marching, because the boundary temperatures T_a and T_b are fixed. If you have Neumann boundary condition (fixed heat flux) at one boundary, you can easily derive the discretisation equation by integrating Eq. (1) in both space and time, but referring to a boundary control volume. The procedure to embed the known heat flux value within the discretised equation is the same as that outlined in Sec. 4.4.2 of Lecture 1, with the addition of the temporal term. In the case of Neumann conditions, the old boundary temperature values will enter the coefficient b and this will need to be updated at the end of each time step.

Below, we take a closer look at each of the three time-discretisation methods defined above.

Further reading: Patankar [1], Sec. 4.3.

3 Explicit time-scheme

For the explicit time-scheme, $f = 0$ and thus:

$$\int_t^{t+\Delta t} T_P dt = T_P^0 \Delta t. \tag{12}$$

This is equivalent to a Taylor expansion of the temperature as a function of time truncated after the first-order term, thus meaning that the time-explicit scheme has a truncation error of order 1, thus first-order accuracy with respect to time. If we reduce the time-step by a factor 2, the truncation error associated with the temporal discretisation reduces by a factor of $2^1 = 2$. Replacing $f = 0$ in Eq. (8) leads to the discretisation equation:

$$a_W T_W + a_P T_P + a_E T_E = b, \tag{13}$$

with coefficients:

$$a_W = 0, \quad a_P = \rho c_p \frac{\Delta x}{\Delta t}, \quad a_E = 0, \tag{14}$$

and source term:

$$b = \frac{\lambda_w}{\delta x_w} T_W^0 + \frac{\lambda_e}{\delta x_e} T_E^0 - \left(\frac{\lambda_w}{\delta x_w} + \frac{\lambda_e}{\delta x_e} - \rho c_p \frac{\Delta x}{\Delta t} \right) T_P^0. \tag{15}$$

Note that, since a_E and a_W are now zero, the matrix of the coefficients will be a diagonal matrix, i.e. a matrix with only one nonzero diagonal. In such a case, we do not even need to solve a linear system, but we can simply solve the equations sequentially from control volume 1 to n, because the equations for each control volume are now decoupled, i.e. they do not depend on the neighbours' temperature values at the new time instant, but only on the neighbour values at the old time instant, which are known. This is the main advantage of the time-explicit scheme.

However, the discretisation equation for the time-explicit method is subject to a strict stability condition. If you remember Lecture 1, Section 4.5, a condition for the stability of the solution procedure was that the coefficients of the neighbours, when appearing at the left-hand side of the equation, had to be negative. In the unsteady case, T_P^0 represents a neighbour of T_P^1 in a temporal sense, and thus stability requires its coefficient to be negative. Taking a look at Eq. (15), this means that if we take the term multiplying T_P^0 to the left-hand side of the discretisation equation, this term must be negative:

$$\left(\frac{\lambda_w}{\delta x_w} + \frac{\lambda_e}{\delta x_e} - \rho c_p \frac{\Delta x}{\Delta t} \right) < 0. \tag{16}$$

Considering, as an example, a situation where λ is constant, and the mesh uniform such that $\delta x_e = \delta x_w = \Delta x$, the stability condition above imposes the following restriction on the time-step size:

$$\Delta t < \frac{\rho c_p (\Delta x)^2}{2\lambda}. \tag{17}$$

This represent a very strict condition, because it means that if we refine the mesh by decreasing the nodes distance by a factor of 2, stability requires that we decrease the time-step by a factor of 4.

In summary, the time-explicit scheme:

- Has the advantage that the equations for each control volume are decoupled, and therefore there is no need to solve a linear system at each time instant. This makes the algorithm simpler, and calculations faster.
- It is a first-order scheme, so in order to achieve sufficient accuracy we need a small time-step.
- It is subject to a strong limitation on the time-step, see Eq. (17), which slows down the calculation of the temporal evolution of the system.

4 Crank-Nicolson time-scheme

For the Crank-Nicolson time-scheme, $f = 1/2$ and thus:

$$\int_t^{t+\Delta t} T_P dt = \frac{T_P^1 + T_P^0}{2}\Delta t. \tag{18}$$

This is equivalent to a Taylor expansion of the temperature as a function of time truncated after the second-order term, thus meaning that the Crank-Nicolson scheme has a truncation error of order 2, thus second-order accuracy with respect to time. If we reduce the time-step by a factor 2, the truncation error associated with the temporal discretisation reduces by a factor of $2^2 = 4$. Replacing $f = 1/2$ in Eq. (8) leads to the discretisation equation:

$$a_W T_W + a_P T_P + a_E T_E = b, \tag{19}$$

with coefficients:

$$a_W = -\frac{1}{2}\frac{\lambda_w}{\delta x_w}, \quad a_P = \rho c_p \frac{\Delta x}{\Delta t} + \frac{1}{2}\left(\frac{\lambda_w}{\delta x_w} + \frac{\lambda_e}{\delta x_e}\right), \quad a_E = -\frac{1}{2}\frac{\lambda_e}{\delta x_e}, \tag{20}$$

and source term:

$$b = \frac{1}{2}\left[\frac{\lambda_w}{\delta x_w}T_W^0 + \frac{\lambda_e}{\delta x_e}T_E^0 - \left(\frac{\lambda_w}{\delta x_w} + \frac{\lambda_e}{\delta x_e} - 2\rho c_p \frac{\Delta x}{\Delta t}\right) T_P^0\right]. \tag{21}$$

You can see that with the Crank-Nicolson scheme $a_W, a_E \neq 0$, which means that the scheme is partially implicit because there is a dependence of T_P on the new temperature values of the neighbours, and thus a linear system needs to be solved at each time-step.

Owing to the stability condition seen in the previous section, there is still a limitation on the time-step size:

$$\Delta t < \frac{\rho c_p (\Delta x)^2}{\lambda}, \tag{22}$$

which only differs from that for a time-explicit scheme by a factor of 2. This is still a strong limitation on the time-step. However, because of the implicit dependence on the neighbours' new temperature values, time-steps above the stability condition will still work. Nonetheless, for large Δt, the solution will oscillate a bit, as we will see in the tutorial, which is an undesired effect.

In summary, the Crank-Nicolson scheme:

- Is second-order accurate with respect to time, which makes it the most accurate among the three schemes seen in this lecture.
- Requires the solution of a linear system at every time-step, which makes it more complicated than the explicit scheme.
- It is still subject to a quite strong limitation on the time-step size.

5 Fully-implicit time-scheme

For the explicit time-scheme, $f = 1$ and thus:

$$\int_t^{t+\Delta t} T_P dt = T_P^1 \Delta t. \tag{23}$$

This is equivalent to a Taylor expansion of the temperature as a function of time truncated after the first-order term, thus meaning that the time-explicit scheme has first-order accuracy with respect to time. Replacing $f = 1$ in Eq. (8) leads to the discretisation equation:

$$a_W T_W + a_P T_P + a_E T_E = b, \tag{24}$$

with coefficients:

$$a_W = -\frac{\lambda_w}{\delta x_w}, \quad a_P = \rho c_p \frac{\Delta x}{\Delta t} + \left(\frac{\lambda_w}{\delta x_w} + \frac{\lambda_e}{\delta x_e} \right), \quad a_E = -\frac{\lambda_e}{\delta x_e}, \tag{25}$$

and source term:

$$b = \rho c_p \frac{\Delta x}{\Delta t} T_P^0. \tag{26}$$

Now, from Eq. (26) you can see that the coefficient multiplying T_P^0, when taken to the left-hand side of the discretisation equation, will be always negative irrespective of the time-step size, this makes the fully-implicit scheme an unconditionally stable scheme.

In summary, the fully-implicit scheme:

- Is not subjected to any time-step restriction and thus is stable for any size of the time-step.
- Is still first-order accurate with respect to time, and thus less accurate than the Crank-Nicolson scheme when the same time-step size is used.
- Requires the solution of a linear system at every time-step, which makes it more complicated than the explicit scheme.

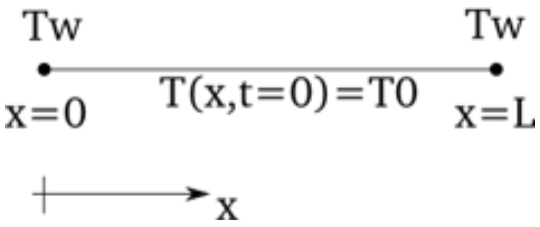


Figure 3: Sketch of the domain and boundary and initial conditions for the tutorials.

6 Matlab tutorials: finite-volume solution of the unsteady 1D heat equation

6.1 Worked example 1: Fully-implicit time-scheme

Implement a FV code in Matlab to solve the unsteady 1D heat conduction equation using a fully-implicit method, with the following conditions:

- $L = 1\,\mathrm{m}$.
- Constant thermal conductivity, $\lambda = 400\,\mathrm{W/(m\,K)}$, density $\rho = 4000\,\mathrm{kg/m^3}$, specific heat $c_p = 400\,\mathrm{J/(kg\,K)}$.
- $n = 21$ equidistant nodes.
- Dirichlet boundary conditions at both boundaries, $T(x = 0) = T(x = L) = T_w = 300\,\mathrm{K}$.
- Initial condition $T(x, t = 0) = 320\,\mathrm{K}$.
- Time-step size $\Delta t = 100\,\mathrm{s}$.
- End time of the simulation $t_{end} = 5000\,\mathrm{s}$.

Compare your solution with the analytical solution:

$$T(x,t) = T_w + \frac{2(T_0 - T_w)}{\pi} \sum_{i=1}^{\infty} \frac{\left[1 - (-1)^i\right]}{i} e^{-\alpha \mu_i^2 t} \sin(\mu_i x), \quad \text{with}\, \mu_i = \frac{i\pi}{L},\ \alpha = \frac{\lambda}{\rho c_p}. \tag{27}$$

Solution

Unlike the steady case, now we need to solve a linear system at each time instant. With $\Delta t = 100\,\mathrm{s}$ and $t_{end} = 5000\,\mathrm{s}$, we have $5000/100 = 50$ time-steps and therefore we will have to solve the linear system 50 times. The cofficient matrix $\mathbf{A}$, see Eq. (25), includes only constants and therefore it will not change during the time marching procedure. For any internal control volume:

$$a_{i,i-1} = -\frac{\lambda}{\delta x}, \quad a_{i,i} = \rho c_p \frac{\Delta x}{\Delta t} + 2\frac{\lambda}{\delta x}, \quad a_{i,i+1} = -\frac{\lambda}{\delta x}, \tag{28}$$

for the first control volume:

$$a_{1,1} = 1, \quad a_{1,2} = 0, \tag{29}$$

and for the n-th control volume:

$$a_{n,n-1} = 0, \quad a_{n,n} = 1. \tag{30}$$

The known terms vector $\mathbf{B}$ includes old temperature values, see Eq. (26), and therefore will change at each time instant. Let's set $k = 1$ as the iteration corresponding to $t = 0$. When $k = 1$, there is nothing to do, because the temperatures in all the nodes are known as the initial condition T_0. For any generic temporal iteration $k > 1$, and internal control volume ($i = 2, ..., n - 1$):

$$b_i = \rho c_p \frac{\Delta x}{\Delta t} T_i^{k-1}, \tag{31}$$

where the previous temperature value T_i^{k-1} is known from the previous solution of the linear system; note that at the time-step corresponding to $t = 1\Delta t$, i.e. $k = 2$ because we set $k = 1$ as the initial condition, the old temperature values correspond to the initial condition, $T_i^{k-1=1} = T_0$. The boundary coefficients $b_1 = T_w$ and $b_n = T_w$ do not change during the calculation. You can imagine that, in order to update the values of $\mathbf{B}$, you will now need a `for` cycle that runs through all the time-steps.

We are now ready to take a look at the Matlab code, see next page. Lines 1-7 should be familiar from the past exercises. In line 8, you see that I am adding `f=1`. The definition of f would not be necessary if I use a fully-implicit method, however I want to write a general code, despite what the exercise is asking, and make the choice of the time-scheme a parameter to be introduced by the user via f; this way, you can use the same code for the next suggested exercises. In line 13, the initial condition is implemented by writing a column vector of temperatures with n elements, each element equal to T_0. Lines 15-18 initialise the matrices, and set the coefficients of $\mathbf{A}$ and $\mathbf{B}$ for the boundary nodes; this is done out of the temporal loop, because the boundary coefficients do not change for Dirichlet conditions (but they do change for Neumann conditions, although we will not see this).

```
1   %%%%% Computational Modelling Techniques - Part 1: Numerical Methods
2   %%%%% Lecture 3, Worked example 1 - Solution of the 1D unsteady heat equation
3
4   clear all; close all; clc; % clears workspace, figures, command window
5   %%%%%%%%%%%%%% Physical parameters
6   L=1; Tw=300; T0=320; lambda=400; rho=4000; cp=400; tEnd=5000;
7   %%%%%%%%%%%%%% Numerical parameters
8   n=21; dt=100; f=1;
9   %%%%%%%%%%%%%% Spatial/temporal Grid generation
10  x0=linspace(0,L,n); dx=L/(n-1); % x0: nodes positions
11  time=0:dt:tEnd; % time is a vector which includes all the time instants
12  %%%%%%%%%%%%%% Initial conditions
13  T=T0*ones(n,1);
14  %%%%%%%%%%%%%% Creating the matrix
15  A=zeros(n,n); B=zeros(n,1); % Fill matrix with zeros
16  %%%%%%%%%%%%%% Time marching
17  A(1,1)=1; B(1)=Tw; % Node 1; this does not change as time marches
18  A(n,n)=1; B(n)=Tw; % Node n: this does not change as time marches
```

Next, we start the temporal loop. We take $k = 1$ as the iteration referring to the initial condition, therefore our loop will start from $k = 2$, and because it starts from 2 it will run till $k = 51$:

```
20  figure('color','w')
21  for k=2:numel(time)
22      for i=2:n-1
23          A(i,i-1)=-f*lambda/dx; A(i,i)=rho*cp*dx/dt+2*f*lambda/dx; A(i,i+1)=-f*lambda/dx;
24          B(i)=(1-f)*(lambda/dx*T(i+1,k-1)+lambda/dx*T(i-1,k-1)-2*lambda/dx*T(i,k-1))+rho*cp*dx/dt*T(i,k-1);
25      end
26      T(:,k)=A\B;
27      plot(x0,T(:,k),'-o'); axis([0 L 0.95*Tw 1.05*T0]); xlabel('x [m]'); ylabel('T [K]'); grid on;
28      F = getframe(gcf);
29  end
30  %%%%%%%%%%%%%% Comparison with theoretical solution at x=L/2
31  I=floor((n+1)/2); % identifies x=L/2; needs n to be an odd number
32  figure('color','w'); plot(time,T(I,:),'b-o'); grid on; xlabel('t [s]'); ylabel('T [K]'); axis([0 tEnd Tw T0])
33  %%%%%%%%%%%%%% Theoretical solution
34  timeTeo=0:1:tEnd;
35  for i=1:1000
36      Sn(i,:)=(1-(-1)^i)/i*sin(i*pi/L*x0(I))*exp(-lambda/rho/cp*(i*pi/L)^2*timeTeo);
37  end
38  Tteo=Tw+2*(T0-Tw)/pi*sum(Sn,1);
39  hold on; plot(timeTeo,Tteo,'r'); legend('Numerical','Theoretical')
```

In line 20, we open a figure where we will display the solution during runtime. You see that, within the temporal loop, in lines 22-25 there is the loop that updates **A** and **B** at each time-step. The coefficients for **A** in the general case (no specific time-scheme chosen) are those reported in Eq. (10); there was no real need to put line 23 within the temporal loop, because **A** does not change during the time marching, however I have done this to make the code more compact, otherwise I should have added an extra `for` loop before the temporal loop. Line 24 defines **B** according to Eq. (11). You see that now the temperature is defined as a matrix. Specifically, it is a matrix where the temperature values at each time instant will be added as new columns from left to right, so each column (of n elements) represents the solution at a specific time instant. For example, `T(:,1)` is the vector of temperatures at time instant $k = 1$, which coincides with the initial condition defined in line 13; `T(:,2)` is the vector of temperatures at time instant $k = 2$, and in general `T(:,k)` is the vector of temperatures at time instant k. Therefore, at the end of the entire temporal loop, all the transient values of the temperature in the domain will be stored in a matrix that has n rows and 51 columns (one for each time instant). So, in line 24, `T(i+1,k-1)` represents T_E^0, `T(i,k-1)` is T_P^0 and `T(i-1,k-1)` is T_W^0. Storing all the temperature values for each time instant in a matrix has two advantages: (i) we retain all the transient solutions so that we can plot them at the end of the calculation; (ii) we can quickly retrieve the temperature values of the old time-step (superscript 0) by simply using `k-1` as second index, without the need to define a new vector of old temperature values. In line 26, the linear system is solved using Matlab's backslash operator, but feel free to use another self-developed method from Lecture 2. Line 27 plots the solution and line 28 is a precious command that tells matlab to plot the solution during runtime. Once the temporal loop is done, line 29, we have available **T** as a matrix with a number of rows coinciding with the number of control volumes, and a number of columns coinciding with the number of time-steps.

Lines 30-39 process the results to compare with the analytical solution in Eq. (27). My intention is to plot $T(x = L/2, t)$, the temperature at the domain centre versus time. First, in line 31 we define the index `I` that locates the index of the control volume at the centre of the domain. We use the `floor` function, which returns the closest integer to its argument towards minus infinity, because we want `I` to be an integer, as we will use it in line 32 as an index to extract data from the matrix **T**. In order to have a smooth theoretical solution, we define a new temporal vector with steps of 1 s in line 34. The analytical solution involves an infinite sum of terms, here we account for the first 1000 terms of the sum, using the cycle of line 35-37. Lines 35-38 evaluate the analytical solution versus time at $x = L/2$, and compare it with the numerical solution in line 39. The figure output is shown in Fig. 4.

6.2 Suggested exercises

1. Repeat Worked example 1, but now using the explicit scheme. You will have to reduce the time-step to make sure you respect the restriction in Eq. (17). Try time-steps that are below, equal to, and above the limit, to see how the solution evolves with time. You will see that when Δt is above the threshold for stability, your solution will oscillate and after a few time-steps will shoot out of the figure boundaries.

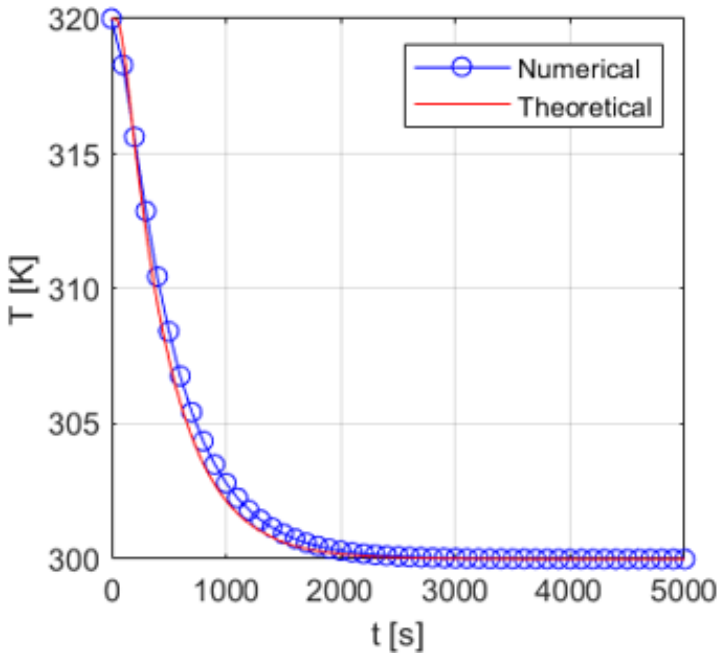


Figure 4: Temperature at $x = L/2$ versus time for worked example 1.

2. Repeat Worked example 1, but now using the Crank-Nicolson scheme. Try time-steps that are below, equal to, and above the limit in Eq. (22), to see how the solution evolves with time. You will notice that the Crank-Nicolson scheme tolerates time-steps above the threshold for stability, because of the implicit contributions to a_W and a_E, however for very large time-steps the solution will becomes oscillatory.

3. Repeat Worked example 1, for different, smaller values of the time step Δt, down to what your computer can achieve. The time- implicit scheme has first-order accuracy with respect to time, and therefore the deviation between the numerical and theoretical solutions, calculated for example in the centre of the domain $x = L/2$ at a time $t = 1000\,\mathrm{s}$ (when the solution has not yet reached steady-state), should decrease with order 1 as Δt is decreased. Note that there exists also the error related to the spatial discretisation. This error may be quite large when n is small, and thus "mask" the error related to the temporal discretisation. In order to rule out the spatial discretisation error, you should perform all the tests with a large value of n, for example $n = 1001$. This way, we can assume that the spatial discretisation error is negligible and the dominant error component in our numerical solution is that related to the temporal discetisation.

References

[1] S. V. Patankar. *Numerical Heat Transfer and Fluid Flow.* Hemisphere Publishing, New York, 1980. NUsearch; Download(may not work).