ONE-DIMENSIONAL HEAT EQUATION

Model PDE:

$$\frac{\partial u}{\partial t} = a\,\frac{\partial^2 u}{\partial x^2} + f, \quad 0 < x < L,\ 0 < t < T$$

supplemented by an initial condition

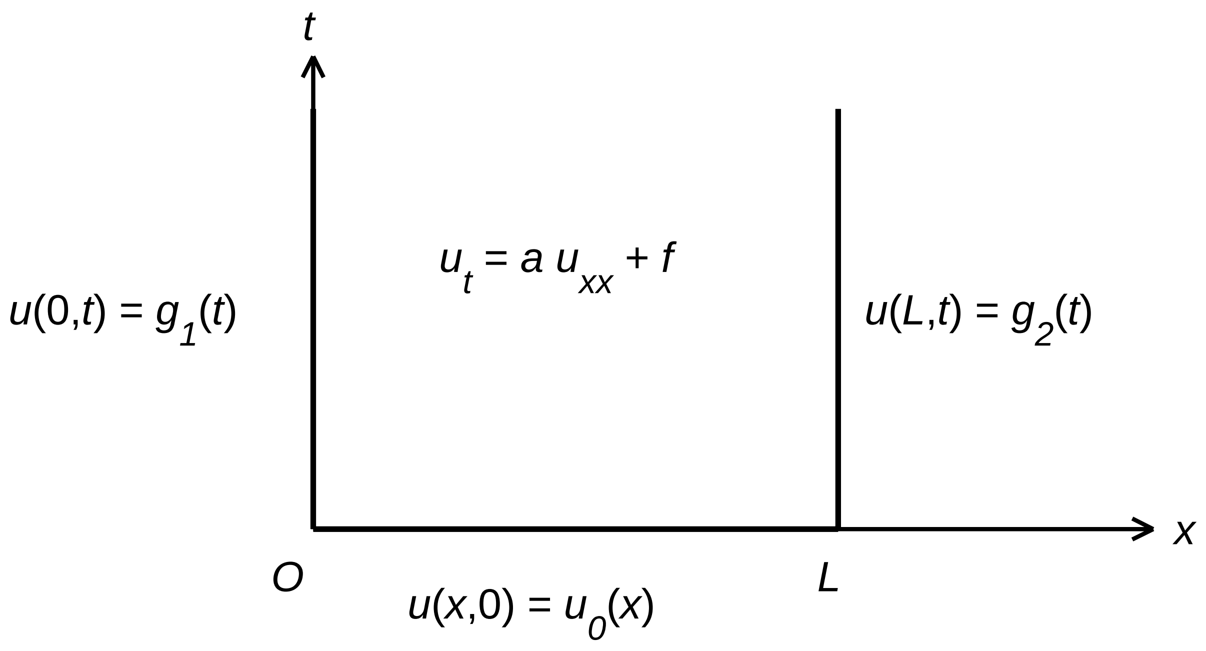
$$u(x,0) = u_0(x), \quad 0 \le x \le L$$

and boundary conditions

$$u(0,t) = g_1(t),\ u(L,t) = g_2(t), \quad 0 \le t \le T$$

The given data are: coefficient $a > 0$, interval lengths $L > 0$ and $T > 0$, function $f(x,t)$ for $0 \le x \le L$ and $0 \le t \le T$, function $u_0(x)$ for $0 \le x \le L$, functions $g_1(t)$ and $g_2(t)$ for $0 \le t \le T$.

The differential equation, the initial and boundary conditions together form an initial boundary value problem.



NUMERICAL METHODS

Two approaches are possible in developing numerical methods for the initial boundary value problem.

In the first approach, the derivation of numerical methods consists of two steps: step 1 is a discretization of the spatial derivatives, leading to a semi-discrete system, a system of ordinary differential equations in the time variable; in step 2, an ODE solver is employed to solve the ODE system.

In the second approach, we discretize with respect to both variables x and t simultaneously, and obtain fully discrete schemes that are ready to be used and solved.

Numerical methods to be discussed:
Semi-discretization
Explicit full discretization
Implicit full discretization

SEMI-DISCRETIZATION

First introduce a partition of the spatial interval $[0, L]$. We divide it into n_x equal parts, and define grid size $h_x = L/n_x$ and grid points $x_i = (i-1)\,h_x$, $1 \le i \le n_x + 1$. Let $u_i(t)$ be an approximation of $u(x_i, t)$, $1 \le i \le n_x + 1$.

Then at an interior grid point x_i, $2 \le i \le n_x$, we consider the differential equation and use the three point central difference formula

$$\frac{\partial^2 u}{\partial x^2}(x_i, t) \approx \frac{u(x_{i+1}, t) - 2\,u(x_i, t) + u(x_{i-1}, t)}{h_x^2}$$

to do the approximation. Note that the three point central difference is a second-order approximation of the second-derivative (i.e., the approximation error is $O(h_x^2)$), leading to a second-order accurate method.

The result is the ODEs

$$\dot{u}_i(t) = a\,\frac{u_{i+1}(t) - 2\,u_i(t) + u_{i-1}(t)}{h_x^2} + f_i(t)$$

for $2 \le i \le n_x$, where $f_i(t) \equiv f(x_i, t)$, and $\dot{u}_i(t)$ is the time derivative of $u_i(t)$. These equations are to be supplemented by values of $u_1(t)$ and $u_{n_x+1}(t)$ from the boundary conditions:

$$u_1(t) = g_1(t), \quad u_{n_x+1}(t) = g_2(t)$$

and by the initial condition:

$$u_i(0) = u_0(x_i), \quad 1 \le i \le n_x + 1$$

The main advantage of this approach is that state-of-the-art ODE solvers can be employed to solve the resulting ODE system.

In matrix/vector notation, the ODE system can be written as

$$\begin{aligned}\dot{\boldsymbol{u}}(t) &= A\,\boldsymbol{u}(t) + \boldsymbol{F}(t)\\ \boldsymbol{u}(0) &= \boldsymbol{u}_0\end{aligned}$$

where, $\boldsymbol{u}(t) = [u_2(t), \cdots, u_{n_x}(t)]^T$ is the vector of unknowns,

$$A = \frac{a}{h_x^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}_{(n_x-1)\times(n_x-1)}$$

$$\boldsymbol{F}(t) = \Big[f_2(t) + \frac{a}{h_x^2}g_1(t), f_3(t), \cdots, \\ f_{n_x-1}(t), f_{n_x}(t) + \frac{a}{h_x^2}g_2(t)\Big]^T$$

$$\boldsymbol{u}_0 = [u_0(x_2), \cdots, u_0(x_{n_x})]^T$$

The ODE system is increasingly stiffer as n_x increases. So it should be solved by an ODE solver that is effective for stiff equations.

EXPLICIT FULL DISCRETIZATION

In addition to the partition of the spatial interval $[0, L]$ introduced earlier, we need a partition of the time interval $[0, T]$. Let n_t be a positive integer. Then the time stepsize is $h_t = T/n_t$. Denote $t_k = (k-1)\, h_t$, $1 \le k \le n_t + 1$, and by u_i^k the finite difference approximation value of $u(x_i, t_k)$.

Now discretize the differential equation at (x_i, t_k), $2 \le i \le n_x$, $2 \le k \le n_t$. Use

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_k) \approx \frac{u_{i+1}^k - 2\, u_i^k + u_{i-1}^k}{h_x^2}$$

$$\frac{\partial u}{\partial t}(x_i, t_k) \approx \frac{u_i^{k+1} - u_i^k}{h_t}$$

Then, we obtain the forward Euler formula

$$\frac{u_i^{k+1} - u_i^k}{h_t} = a\, \frac{u_{i+1}^k - 2\, u_i^k + u_{i-1}^k}{h_x^2} + f_i^k$$

It is convenient to introduce the ratio

$$\gamma = \frac{a\, h_t}{h_x^2}$$

The complete description of the method, incorporating the initial and boundary conditions, is

$$u_i^1 = u_0(x_i), \quad 1 \le i \le n_x + 1$$

and for $k \ge 1$,

$$\begin{aligned} u_i^{k+1} &= \gamma\, u_{i-1}^k + (1 - 2\,\gamma)\, u_i^k + \gamma\, u_{i+1}^k + h_t f_i^k \\ &\qquad 2 \le i \le n_x \\ u_1^{k+1} &= g_1(t_{k+1}), \quad u_{n_x+1}^{k+1} = g_2(t_{k+1}) \end{aligned}$$

For this method, once the approximate solution at $t = t_k$ is known, we can compute the solution at the next time level $t = t_{k+1}$ directly. So the method is called an explicit method. The main advantage of an explicit method is that there is no need to solve linear systems, as the approximate solution at time level $t = t_{k+1}$ is computed directly from that at the previous time steps.

Stability and Convergence

Consider stability of the method with an analysis of error propagation through time advancing of the method. Suppose that the computed solution at level $t = t_k$ is $\tilde{u}_i^k = u_i^k + \epsilon_i^k$, $|\epsilon_i^k| \le \epsilon^k$, $1 \le i \le n_x + 1$. Then, the computed numerical solution (assuming no further error is introduced) is

$$\begin{aligned} \tilde{u}_i^{k+1} &\equiv u_i^{k+1} + \epsilon_i^{k+1} \\ &= \gamma \tilde{u}_{i+1}^k + (1 - 2\,\gamma)\, \tilde{u}_i^k + \gamma \tilde{u}_{i-1}^k + h_t f_i^k \end{aligned}$$

Hence, for $2 \le i \le n_x$,

$$\epsilon_i^{k+1} = \gamma \epsilon_{i+1}^k + (1 - 2\,\gamma)\, \epsilon_i^k + \gamma \epsilon_{i-1}^k$$

is the error associated with the solution $\tilde{u}_i^{k+1}$. So if

$$\gamma \equiv \frac{a\, h_t}{h_x^2} \le \frac{1}{2} \tag{1}$$

then for $2 \le i \le n_x$,

$$|\epsilon_i^{k+1}| \le \gamma |\epsilon_{i+1}^k| + (1 - 2\,\gamma)\, |\epsilon_i^k| + \gamma |\epsilon_{i-1}^k| \le \epsilon^k$$

i.e., the error is not amplified in time advancing and the numerical method is stable. It can be shown that (1) is actually both a necessary and sufficient condition for the forward Euler method to be stable.

Recall that the difference scheme is derived with a second-order approximation ($O(h_x^2)$) of $\frac{\partial^2 u}{\partial x^2}(x_i, t_k)$, and a first-order approximation ($O(h_t)$) of $\frac{\partial u}{\partial t}(x_i, t_k)$.

It can be proved that under the stability condition (1), if the true solution u has several continuous partial derivatives, the following error bound holds:

$$\max_{\substack{1 \le i \le n_x+1 \\ 1 \le k \le n_t+1}} |u(x_i, t_k) - u_i^k| = O(h_t + h_x^2)$$

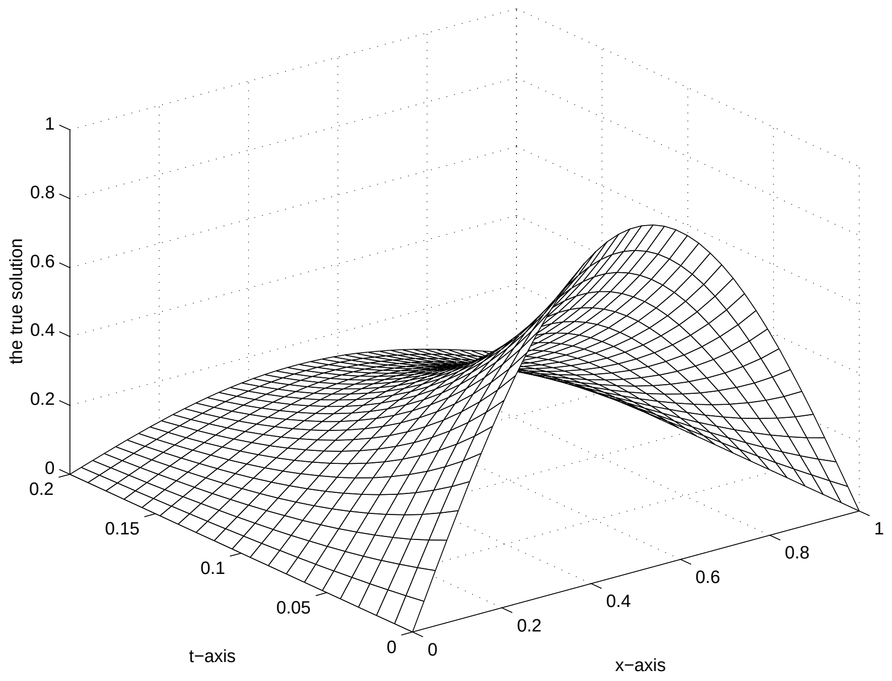
i.e., the scheme is of second-order accuracy in h_x and of first-order accuracy in h_t.

Numerical Example

Let us solve the following initial-boundary value problem:

$$\begin{cases} u_t = u_{xx}, \quad x \in (0,1),\ t \in (0, 0.2) \\ u(x,0) = \sin(\pi x), \quad x \in [0,1] \\ u(0,t) = u(1,t) = 0, \quad t \in [0, 0.2] \end{cases}$$

The true solution is $u(x,t) = e^{-\pi^2 t} \sin(\pi x)$, shown in the following figure.



In this example, $a = 1$, $L = 1$, $T = 0.2$. Then $h_t = 0.2/n_t$, $h_x = 1/n_x$. So the stability condition (1) is

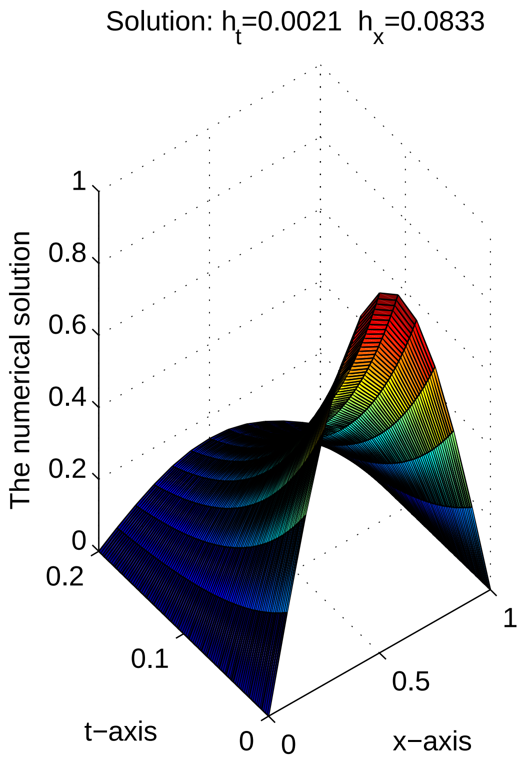
$$\gamma = \frac{0.2\, n_x^2}{n_t} \leq \frac{1}{2}$$

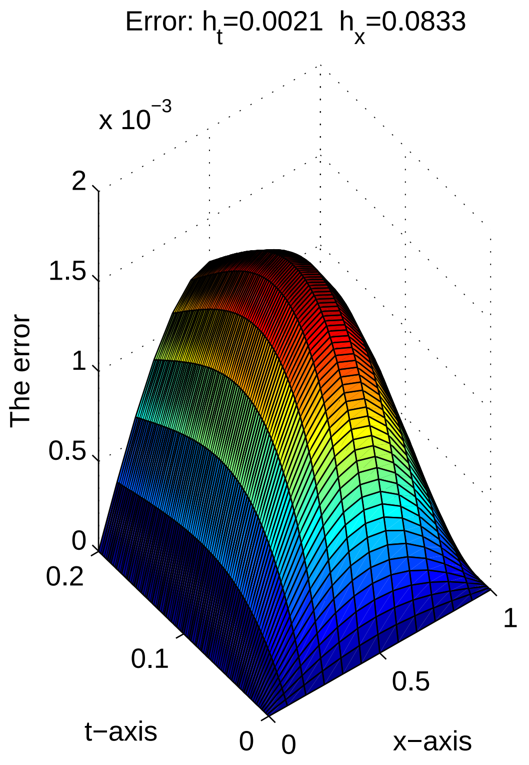
We choose $n_x = 3, 6, 12$ and correspondingly $n_t = 6, 24, 96$ so that $\gamma = 0.3$ and the stability condition is satisfied. We list the maximum errors $\max_{1 \leq i \leq n_x+1} |u(x_i, t_k) - u_i^k|$ for $t_k = 0.2$ in the following table.

n_x	n_t	Max. Error
3	6	1.8414E–2
6	24	5.0829E–3
12	96	1.2573E–3

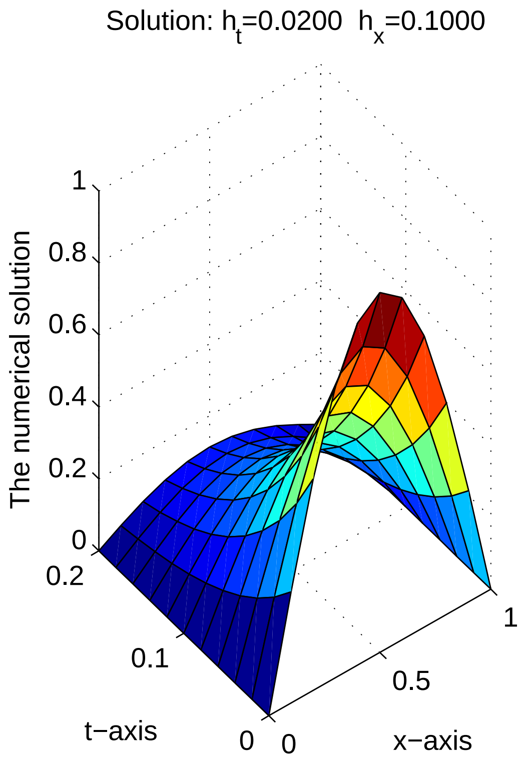
Observe that downward in the table, the maximum error is decreased by a factor around 4.

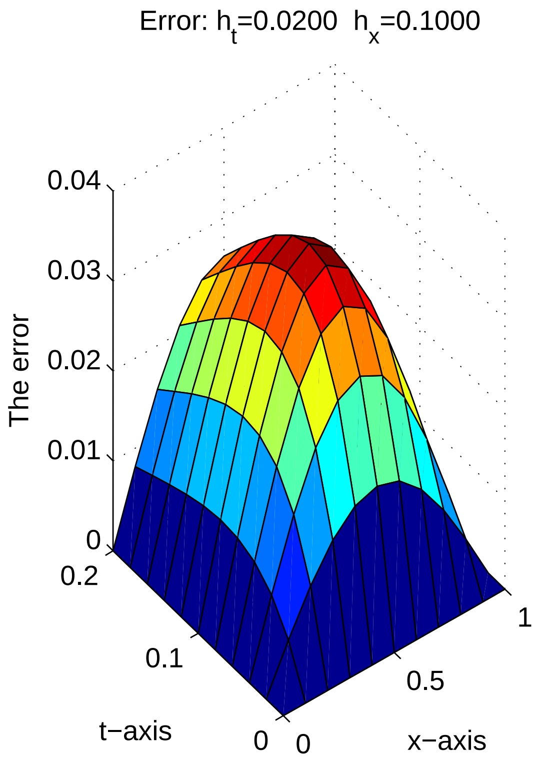
The numerical solution and the corresponding error for $n_x = 12$ and $n_t = 96$ are shown in the following figure.



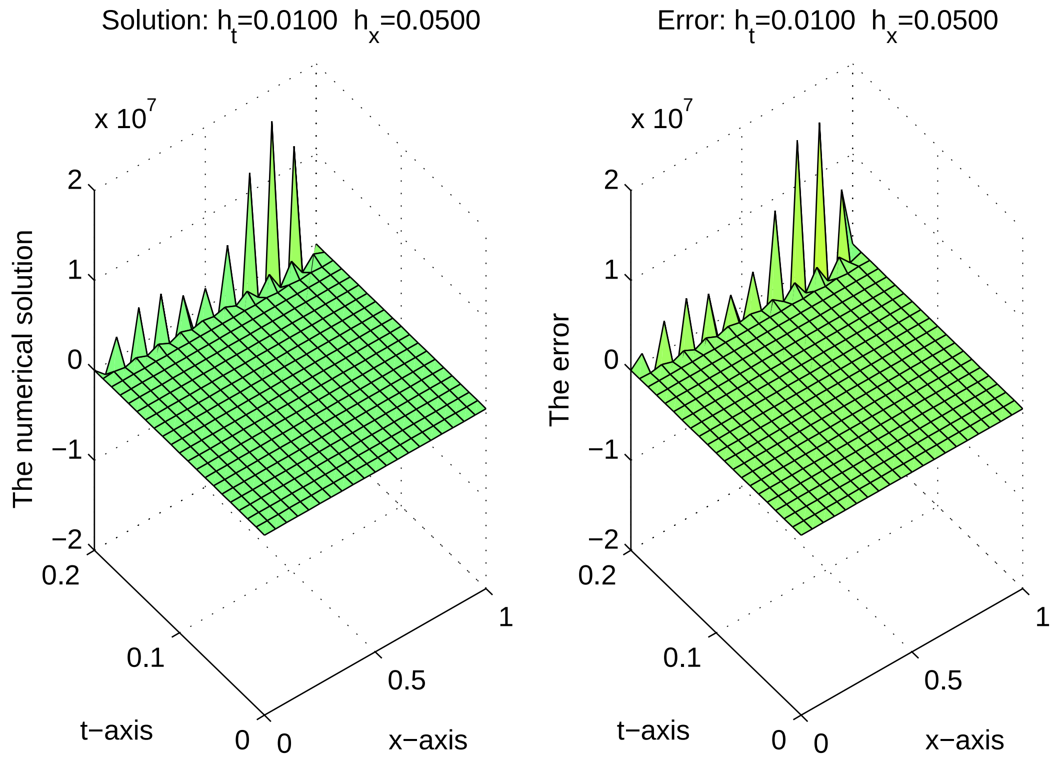


To see the role played by the stability condition, we take other values of n_x and n_t. We first take $n_x = 10$ and $n_t = 10$. The results are shown in the figure. In this case, the ratio $\gamma = 2$, and the scheme is not stable. However, since the difference scheme is applied only 10 times ($n_t = 10$), the accumulation of the roundoff errors is not apparent.





To have a more dramatic illustration, take $n_x = 20$ and $n_t = 20$. The ratio is $\gamma = 4$. The results are shown in the figure. We observe that the effect of roundoff error accumulation becomes rather evident for t close to its upper bound 0.2.



This shows the importance of maintaining the stability condition.

IMPLICIT FULL DISCRETIZATION

The main weakness of the explicit method is the requirement of the stability condition, that imposes a restriction on the relative size of h_t with respect to h_x:

Suppose the numerical solution corresponding to the current values of h_t and h_x is not accurate enough. It is natural to try smaller values for h_t and h_x. Assume we (nearly) double the number of spatial grid points, $\bar{h}_x = h_x/2$. To have the same value of the ratio γ in order to maintain the stability, we need to choose $\bar{h}_t$ such that

$$\frac{a\,\bar{h}_t}{\bar{h}_x^2} = \frac{a\,h_t}{h_x^2}$$

i.e.,

$$\bar{h}_t = \frac{1}{4}\,h_t$$

In other words, whenever h_x is halved, h_t must be quartered. This may lead to the use of prohibitively small time stepsize.

Does there exist a difference method that is always stable?

Let us keep the notations for the partitions of the spatial and time intervals:

$$h_x = L/n_x, \quad x_i = (i-1)\,h_x,\ 1 \le i \le n_x + 1$$
$$h_t = T/n_t, \quad t_k = (k-1)\,h_t,\ 1 \le k \le n_t + 1$$

Again, denote by u_i^k the finite difference approximation value of $u(x_i, t_k)$, and use the three point central difference approximation

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_k) \approx \frac{u_{i+1}^k - 2\,u_i^k + u_{i-1}^k}{h_x^2}$$

for the second-order spatial derivative. For the first-order time derivative, we use instead the backward difference:

$$\frac{\partial u}{\partial t}(x_i, t_k) \approx \frac{u_i^k - u_i^{k-1}}{h_t}$$

The result is the backward Euler method

$$\frac{u_i^k - u_i^{k-1}}{h_t} = a\,\frac{u_{i+1}^k - 2\,u_i^k + u_{i-1}^k}{h_x^2} + f_i^k$$

for $2 \le i \le n_x$.

Again, denote the ratio $\gamma = a\,h_t/h_x^2$. Then the numerical scheme is

$$u_i^1 = u_0(x_i), \quad 1 \le i \le n_x + 1$$

and for $k \ge 2$,

$$\begin{cases} -\gamma\,u_{i-1}^k + (1 + 2\,\gamma)\,u_i^k - \gamma\,u_{i+1}^k = u_i^{k-1} + h_t f_i^k \\ \qquad\qquad 2 \le i \le n_x \\ u_1^k = g_1(t_k), \quad u_{n_x+1}^k = g_2(t_k) \end{cases}$$

Unlike the explicit method, here for given approximate solution at $t = t_{k-1}$, we need to solve a linear system to find the approximate solution at the next time level $t = t_k$. Such a method is called an implicit method.

Implementation of Implicit Method

At each time level $t = t_k$, $k \geq 2$, we need to solve a linear system:

$$A\,\boldsymbol{u}^k = \boldsymbol{F}^k$$

Here, the unknown vector $\boldsymbol{u}^k = [u_2^k, \cdots, u_{n_x}^k]^T$, and the right-hand side vector

$$\begin{aligned}\boldsymbol{F}^k = [&u_2^{k-1} + h_t f_2^k + \gamma\, g_1(t_k), u_3^{k-1} + h_t f_3^k, \cdots, \\ &u_{n_x-1}^{k-1} + h_t f_{n_x-1}^k, u_{n_x}^{k-1} + h_t f_{n_x}^k + \gamma\, g_2(t_{n_x+1})]^T\end{aligned}$$

The $(n_x - 1) \times (n_x - 1)$ coefficient matrix

$$A = \begin{bmatrix} 1+2\gamma & -\gamma & & & \\ -\gamma & 1+2\gamma & -\gamma & & \\ & \ddots & \ddots & \ddots & \\ & & -\gamma & 1+2\gamma & -\gamma \\ & & & -\gamma & 1+2\gamma \end{bmatrix}$$

is the same for any k. So we only need to compute its LU factorization once which is stored and used to solve the linear systems for all k. Since A is tridiagonal, we can use the program `tridiag` for its LU factorization.

Stability and Convergence

It can be shown that for any values of h_x and h_t, error propagation with the time advancing of the implicit method is well controlled. So the implicit scheme is unconditionally stable.

Recall that the difference scheme is derived with a second-order approximation ($O(h_x^2)$) of $\frac{\partial^2 u}{\partial x^2}(x_i, t_k)$, and a first-order approximation ($O(h_t)$) of $\frac{\partial u}{\partial t}(x_i, t_k)$. The following error bound can be proved:

$$\max_{\substack{1 \le i \le n_x+1 \\ 1 \le k \le n_t+1}} |u(x_i, t_k) - u_i^k| = O(h_t + h_x^2)$$

i.e., the backward Euler method is second-order accurate in space and first-order accurate in time, provided the true solution u has several continuous partial derivatives.

Numerical Example

Let us apply the backward Euler method to solve the same initial-boundary value problem:

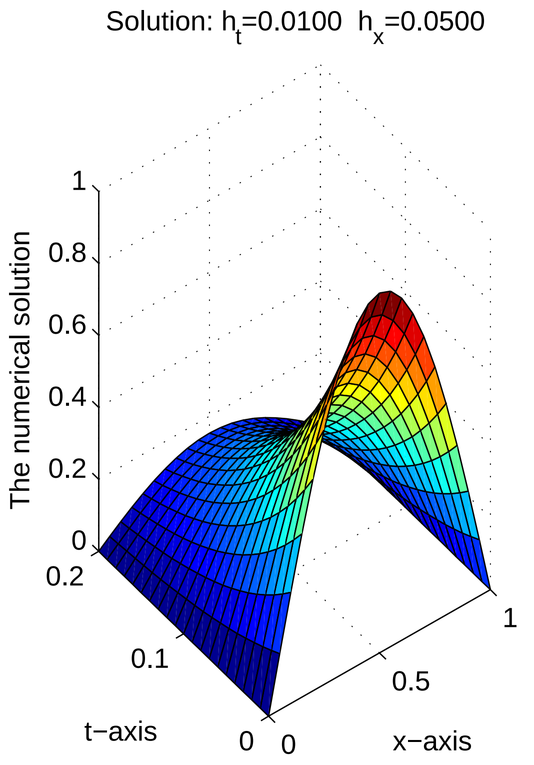
$$\begin{cases} u_t = u_{xx}, \quad x \in (0,1),\ t \in (0,0.2) \\ u(x,0) = \sin(\pi x), \quad x \in [0,1] \\ u(0,t) = u(1,t) = 0, \quad t \in [0,0.2] \end{cases}$$

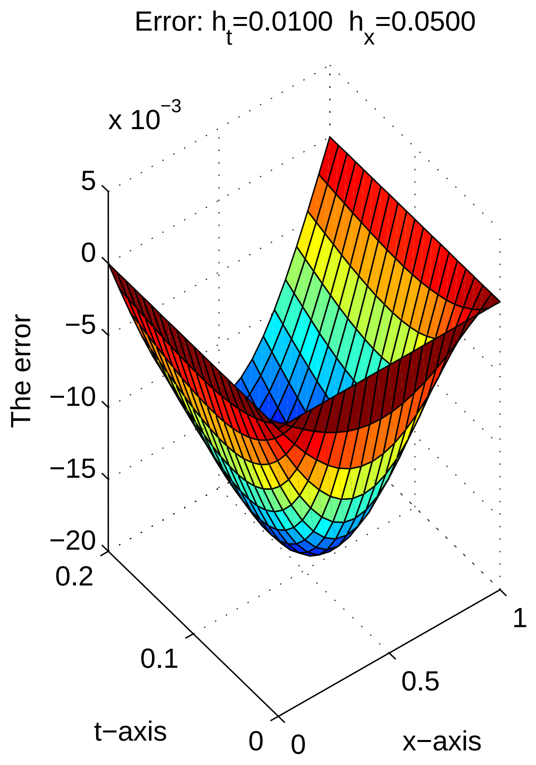
that was solved by the forward Euler method.

Recall that the true solution is

$$u(x,t) = e^{-\pi^2 t} \sin(\pi x)$$

Numerical results with $(n_x, n_t) = (20, 20)$ are shown in the figure. We see that there is no stability problem with the numerical solutions.





To have a closer look at the error behavior, we give a table of the numerical solution errors with $n_t = n_x = 5$ and the ratios of these solution errors with those for several other pairs of n_x and n_t. It is evident that as we double both values of n_t and n_x, the errors are reduced by factors approximately 2, indicating a linear convergence behavior. When we double the value of n_x and quadruple the value of n_t, the numerical solution errors are reduced by factors nearly 4 (in the table, 3.82). If we quadruple both n_t and n_x, the error reduction factors are again close to 4 (in the table, 4.29). When the starting values of n_t and n_x are larger, the ratios 3.82 and 4.29 given in the table will be closer to 4. This phenomenon is consistent with the theoretical result that the method is of first order in h_t and second order in h_x.

x	$n_t = 5$	$n_t = 10$	$n_t = 20$	$n_t = 20$
	$n_x = 5$	$n_x = 10$	$n_x = 10$	$n_x = 20$
0.2	−3.50E − 2	2.09	3.82	4.29
0.4	−5.66E − 2	2.09	3.82	4.29