Click here to view the PDF version.

```julia
begin
    using Plots
    using PlutoUI
    using PlutoTeachingTools
    using LaTeXStrings
    using LinearAlgebra
    using HypertextLiteral: @htl, @htl_str
end
```

Table of Contents

Boundary value problems

The goal of boundary value problems is to reconstruct the function values $u(x, t)$ of a function u on an entire domain $x \in [a, b]$ and for all times t knowing only two things

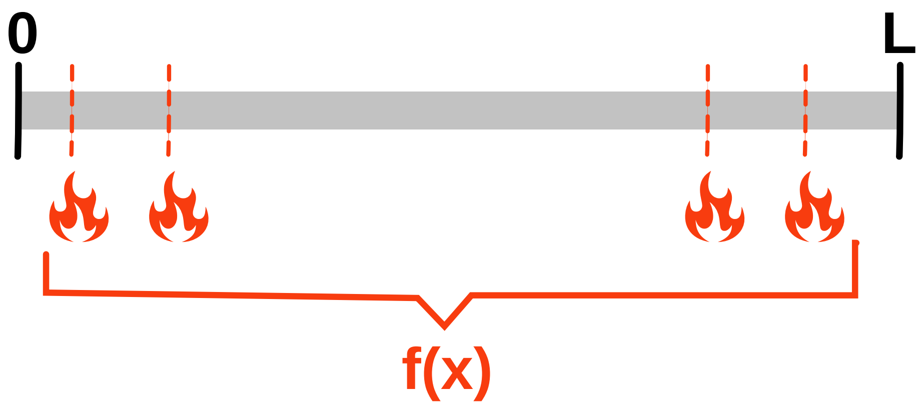
- The behaviour of (some) derivatives of u and
- Some information about u at the boundary $x = a$ and $x = b$ for the initial time $t = 0$. For example we may know the values $u(a, 0)$ and $u(b, 0)$.

Classic examples of boundary value problems include

- the heat equation, where $u(x, t)$ is the temperature at time t and position x of a material, or
- Poisson's equation where $u(x, t)$ is the electrostatic potential at position x and time t corresponding to a given time-dependent density of electronic charges, or
- the diffusion equation where $u(x, t)$ is the concentration of some solute at time t at position x in the solution.

In this lecture, to simplify matters and to make this more concrete, we will only consider one example, namely the **heat equation**.

Stationary heat equation in one dimension



We consider a metal bar extending over the interval $[0, L]$. For a given moment in time t we denote the **temperature** at a point $x \in [0, L]$ by $u(x, t)$. The **heat flux** we fruther denote by $\phi(x, t)$ and allow for an **external source of heat** $f(x)$, e.g. by one more more heat guns pointed at the metal bar.

The **rate of change of the internal heat** $\frac{\partial Q}{\partial t}$ at point x of the rod is proportional to the change in temperature $\frac{\partial u}{\partial t}$, that is

$$\frac{\partial Q}{\partial t} = c\rho \, \frac{\partial u}{\partial t}(x,t)$$

where proportionality constants are the heat capacity c and the mass density ρ.

Due to **conservation of energy** the change of internal heat of a slice $[x, x + \delta x]$ is equal to the incoming heat $f(x)\,\delta x + \phi(x,t)$ minus the outgoing heat $\phi(x+\delta x, t)$, such that we obtain

$$\frac{\partial Q}{\partial t}\delta x = \phi(x,t) + f(x)\,\delta x - \phi(x+\delta x, t).$$

Dividing by δx and **taking the limit** $\delta x \to 0$ thus yields

$$\frac{\partial Q}{\partial t} = -\frac{\partial \phi}{\partial x}(x,t) + f(x).$$

Combining with **Fourier's law**

$$\phi(x,t) = -k\frac{\partial u}{\partial x}(x,t)$$

where k is the thermal conductivity, we obtain the equation

$$c\rho \, \frac{\partial u}{\partial t}(x,t) = k\frac{\partial^2 u}{\partial x^2}(x,t) + f(x) \qquad x \in (0,L), \; t > 0$$

which is the **heat equation** in one dimension, a **partial differential equation**. The heat equation describes the **evolution of the temperature in a system** (here the metal bar) and has as its unknown the temperature distribution $u(x,t)$ at any location at any point in time.

To simplify matters in this lecture **we restrict** ourselves to the case of the **stationary heat equation**, i.e. the case where we assume the **temperature** $u(x,t)$ to be in a state where it **no longer changes with time**, i.e. $\frac{\partial u}{\partial t} = 0$. Dropping thus the zero terms and the dependency on t we obtain the **stationary heat equation**

$$-k\,\frac{\partial^2 u}{\partial x^2}(x) = f(x) \qquad x \in (0,L).$$

To fully specify the problem and find a unique solution we still **need to indicate what happens at the extremities of the bar**, i.e. at $u(0)$ and at $u(L)$. Since these two points sit at the boundary of our computational domain $[0, L]$ we usually call these conditions on u **boundary conditions**.

We sketch the two most common cases:

- Consider the case where the bar is in contact with heat baths of constant temperature. A temperature b_0 for the left-hand side bath and b_L for the right-hand side bath. In this case we know $u(0) = b_0$ and $u(L) = b_L$. We obtain the **heat equation using Dirichlet boundary conditions**:

$$\begin{cases} -k\,\dfrac{\partial^2 u}{\partial x^2}(x) = f(x) \qquad x \in (0, L). \\ \qquad u(0) = b_0 \\ \qquad u(L) = b_L \end{cases} \tag{1}$$

- The bar may also be insulating at the boundary, such that no heat may be allowed to flow in or out of the extremities. In this case the boundary conditions are $0 = -k\frac{\partial u}{\partial x}(0)$ and $0 = -k\frac{\partial u}{\partial x}(L)$ (zero heat flux). We obtain a **heat equation with Neumann boundary conditions**, where the flux at the boundary is controlled:

$$\begin{cases} -k\,\dfrac{\partial^2 u}{\partial x^2}(x) = f(x) \qquad x \in (0, L). \\ \dfrac{\partial u}{\partial x}(0) = -\dfrac{\phi_0}{k} \\ \dfrac{\partial u}{\partial x}(L) = -\dfrac{\phi_L}{k} \end{cases} \tag{2}$$

  where in the insulating case $\phi_0 = \phi_L = 0$. We will not consider Neumann problems any further here.

Since for such kind of problems knowing the boundary is imperative to obtain a unique solution one refers to such problems as **boundary value problems**.

Finite difference approximation

We focus on the case of a Dirichlet boundary (1) with $k = 1$. Our goal is thus to find a function $u : [0, L] \to \mathbb{R}$ with

$$\begin{cases} -\dfrac{\partial^2 u}{\partial x^2}(x) = f(x) & x \in (0, L) \\ u(0) = b_0, \quad u(L) = b_L, \end{cases}$$

were $b_0, b_L \in \mathbb{R}$. Similar to our approach when solving initial value problems (chapter 12) we **divide the full interval** $[0, L]$ **into** $N+1$ **subintervals** $[x_j, x_{j+1}]$ of uniform size h, i.e.

$$x_j = jh \quad j = 0, \ldots, N+1, \qquad h = \frac{L}{N+1}.$$

Our goal is thus to find approximate points u_j such that $u_j \approx u(x_j)$ at the nodes x_j.

⚠ TODO ⚠

IVP is now after BCP, adjust reference accordingly

```
TODO("IVP is now after BCP, adjust reference accordingly")
```

Due to the Dirichlet boundary conditions $u(0) = b_0$ and $u(L) = b_L$. To satisfy these we neccessarily need $u_0 = b_0$ and $u_{N+1} = b_L$. The unknowns of our problems are therefore those u_j with $1 \leq j \leq N$ — the **internal nodes** of the interval.

These internal nodes $u(x_j)$ need to satisfy

$$-\frac{\partial^2 u}{\partial x^2}(x_j) = f(x_j) \qquad \forall \, 1 \leq j \leq N.$$

As the derivatives of u are unknown to us we employ a **central finite-difference formula** to replace this derivative by the approximation

$$f(x_j) = -\frac{\partial^2 u}{\partial x^2}(x_j) \approx -\frac{u(x_{j-1}) - 2u(x_j) + u(x_{j+1})}{h^2} \qquad \forall \, 1 \leq j \leq N \tag{3}$$

or in terms of u_j:

$$\frac{-u_{j-1} + 2u_j - u_{j+1}}{h^2} = f(x_j) \qquad \forall \, 1 \leq j \leq N. \tag{4}$$

Due to our **imposed boundary conditions** $u_0 = b_0$ and $u_{N+1} = b_L$, such that we can simplify the equations for $j = 1$ and $j = N$ as follows:

$$\begin{aligned} j = 1: &\qquad \frac{-b_0 + 2u_1 - u_2}{h^2} = f(x_1) &&\Leftrightarrow \quad \frac{2u_1 - u_2}{h^2} = f(x_1) + \frac{b_0}{h^2} \\ j = N: &\qquad \frac{-u_{N-1} + 2u_N - b_L}{h^2} = f(x_N) &&\Leftrightarrow \quad \frac{-u_{N-1} + 2u_N}{h^2} = f(x_N) + \frac{b_L}{h^2} \end{aligned}$$

Collecting everything we obtain the system of equations

$$\begin{cases} \dfrac{2u_1 - u_2}{h^2} = f(x_1) + \dfrac{b_0}{h^2} & j = 1 \\ \dfrac{-u_{j-1} + 2u_j - u_{j+1}}{h^2} = f(x_j) & \forall\, 2 \leq j \leq N-1 \\ \dfrac{-u_{N-1} + 2u_N}{h^2} = f(x_N) + \dfrac{b_L}{h^2} & j = N \end{cases} \tag{5}$$

▶ **Optional: Does the problem (5) always have a solution ?**

Finally, to write problem (5) more compactly we introduce a **vector of all unknowns** $\mathbf{u} = (u_1, \ldots, u_N)^T \in \mathbb{R}^N$, define the **system matrix** $\mathbf{A} \in \mathbb{R}^{N \times N}$ as well as the **right-hand side** $\mathbf{b} \in \mathbb{R}^N$ with

$$\mathbf{A} = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} f(x_1) + \frac{b_0}{h^2} \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) + \frac{b_L}{h^2} \end{pmatrix}. \tag{7}$$

With these objects the problem can be written as a linear system

$$\mathbf{A}\mathbf{u} = \mathbf{b}. \tag{8}$$

which is to be solved for the unknows $\mathbf{u}$.

We notice that $\mathbf{A}$ is **symmetric and tridiagonal**. Additionally one can show $\mathbf{A}$ to be **positive definite**. Problem (8) can therefore be **efficiently solved** using direct methods based on (sparse) LU factorisation (chapter 5) or an iterative approaches (chapter 6), e.g. the conjugate gradient method.

We summarise:

Algorithm 1: Finite differences for Dirichlet bounary value problems

Given a boundary value problem with Dirichlet boundary conditions

$$\begin{cases} -\dfrac{\partial^2 u}{\partial x^2}(x) = f(x) & x \in (0, L) \\ u(0) = b_0, \quad u(L) = b_L, \end{cases} \tag{9}$$

and nodes $x_j = j\,h$ with $j = 0, \ldots, N$ and $h = \frac{L}{N+1}$ solve the linear system $\mathbf{A}\mathbf{u} = \mathbf{b}$ with $\mathbf{A}$ and $\mathbf{b}$ given in (7) for $\mathbf{u} = (u_1, \ldots, u_N)^T$ and $u_0 = b_0$, $u_{N+1} = b_L$.

The values $u_0, u_1, \ldots, u_{N+1}$ approximate the solution $u(x)$ at the nodal points $\{x_j\}_{j=0}^{N+1}$.

One implementation of Algorithm 1, which solves the linear system using LU factorisation is:

```
fd_dirichlet (generic function with 1 method)
```

```julia
function fd_dirichlet(f, L, b₀, bₗ, N)
    # f:  Function describing the external heat source
    # L:  Length of the metal rod
    # b₀: Left-hand side boundary value
    # bₗ: Right-hand side boundary value
    # N:  Number of nodal points for the finite-differences scheme

    h = L / (N+1)              # Step size
    x = [i * h for i in 1:N]   # Nodal points

    # Build system matrix A
    diagonal     = 2ones(N)    / h^2
    offdiagonal = -ones(N-1) / h^2
    A = SymTridiagonal(diagonal, offdiagonal)

    # Build right-hand side
    b = [f(xⱼ) for xⱼ in x]  # Evaluate function f at nodal points

    # Set terms due to boundary conditions
    b[1] = f(x[1]) + b₀ / h^2
    b[N] = f(x[N]) + bₗ / h^2

    # Solve problem using LU factorisation
    u = A \ b

    (; u, x, h, A, b)  # Return intermediates and results
end
```

Let us consider the example

$$\begin{cases} -\dfrac{\partial^2 u}{\partial x^2}(x) = \sin(x) \qquad x \in (0, 2\pi) \\ \qquad u(0) = 1, \quad u(L) = 2, \end{cases}$$

i.e. the case of

```julia
begin
    b₀ = 1
    bₗ = 2
    L = 2π
    f(x) = sin(x)
end;
```

In this case the exact solution can be found analytically as

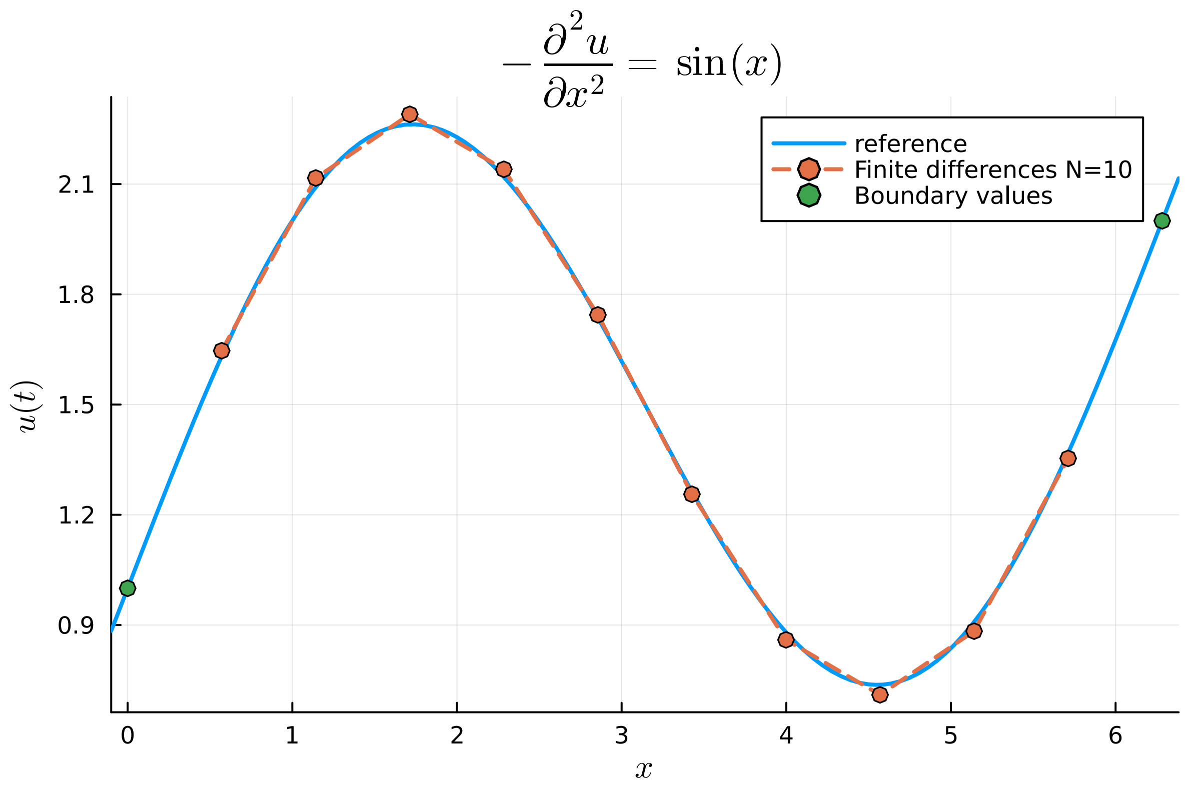
```
u_exact (generic function with 1 method)
```

```julia
u_exact(x) = sin(x) + x/2π + 1
```

Let's solve it with finite differences

```julia
res_fd = fd_dirichlet(f, L, b₀, bₗ, N);
```

and plot the result:



```julia
let
    plot(u_exact; xlims=(-0.1, L+0.1), legend=:topright,
         label="reference", lw=2, xlabel=L"x", ylabel=L"u(t)",
         title=L"-\frac{\partial^2 u}{\partial x^2} = \sin(x)")

    plot!(res_fd.x, res_fd.u; label="Finite differences N=$N", mark=:o, lw=2,
    ls=:dash)

    scatter!([0, L], [b₀, b₁], label="Boundary values", mark=:o, c=3)
end
```

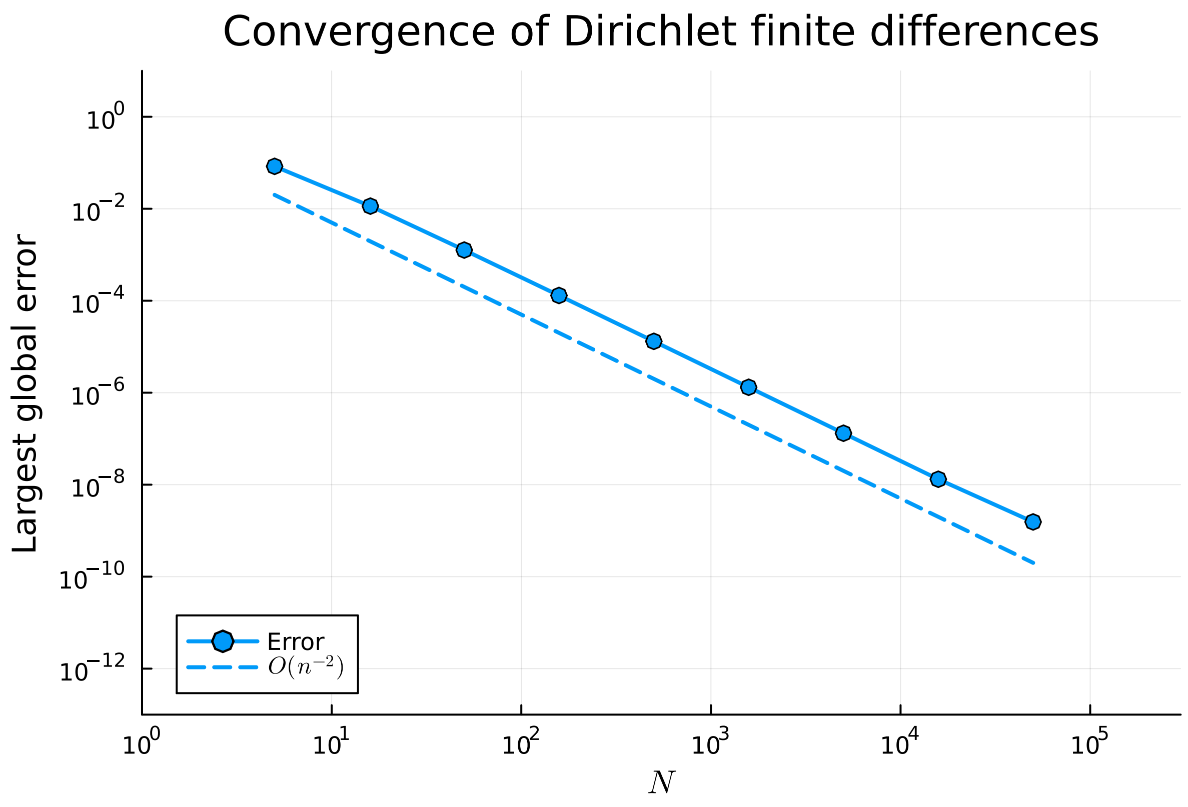
N = 10

Note that the plot of the finite-difference solution does not include the nodal point at the boundaries ($x = 0$ and $x = 2\pi$ — shown in green in the plot).

We observe that as the number of points `N` is increased, the solution visually becomes more and more accurate. The output of our numerical procedure are the computed points $u_1, u_2, \ldots u_N$. These points are meant to approximate the true function values $u(x_1), u(x_2), \ldots, u(x_N)$ of u evaluated at the nodal points. A measure for the accuracy of our obtained solution is thus the maximal deviation any of these computed points has from from the true $u(x_j)$. This measure is called the **global error** defined by

$$|e_j| = |u(x_j) - u_j| \qquad \text{for } j = 1, \ldots, N.$$

Numerically, by observing how this global error changes as N increases, we observe **quadratic convergence**:



```julia
let
    Ns = [ round(Int, 5 * 10^k) for k in 0:0.5:4 ]
    errors = Float64[]
    for N in Ns
        res_fd = fd_dirichlet(f, L, b₀, bₗ, N)
        x = res_fd.x
        u = res_fd.u
        error = [ u_exact(x[j]) - u[j] for j in 1:length(x)]
        push!(errors, maximum(error))
    end

    p = plot(; title="Convergence of Dirichlet finite differences",
               xaxis=:log, yaxis=:log,
               xlabel=L"N", ylabel="Largest global error",
               legend=:bottomleft,
               xlims=(1, 3e5), ylims=(1e-13, 10))
    plot!(p, Ns, errors; lw=2, mark=:o, c=1, label="Error")
    plot!(p, Ns, 0.02(first(Ns) ./ Ns).^2, ls=:dash, lw=2, label=L"O(n^{-2})", c=1)

    xticks!(p, 10.0 .^ (0:1:5))
    yticks!(p, 10.0 .^ (0:-2:-12))
    p
end
```

In line with this discussion we define convergence for numerical schemes for boundary value problems as follows:

Definition: Convergence order for boundary value problems

A numerical scheme approximating a boundary value problem (9) **converges with order p** if there exists a constant $\alpha > 0$ such that

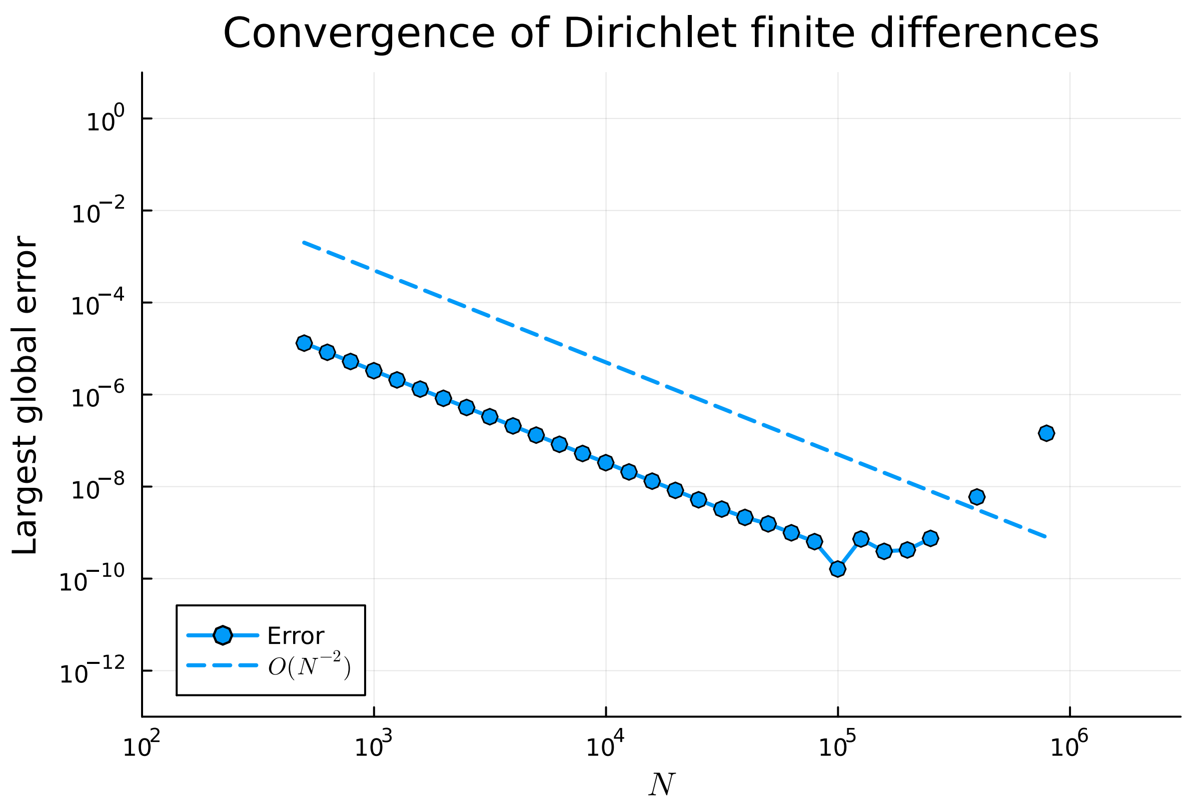
$$\max_{j=0,\ldots,N+1} |u(x_j) - u_j| \leq \alpha\, h^p$$

provided that the solution u is sufficiently regular.

For the **central finite difference scheme** discussed here $p = 2$.

Numerical stability

Let us push the convergence plot from above a little further:

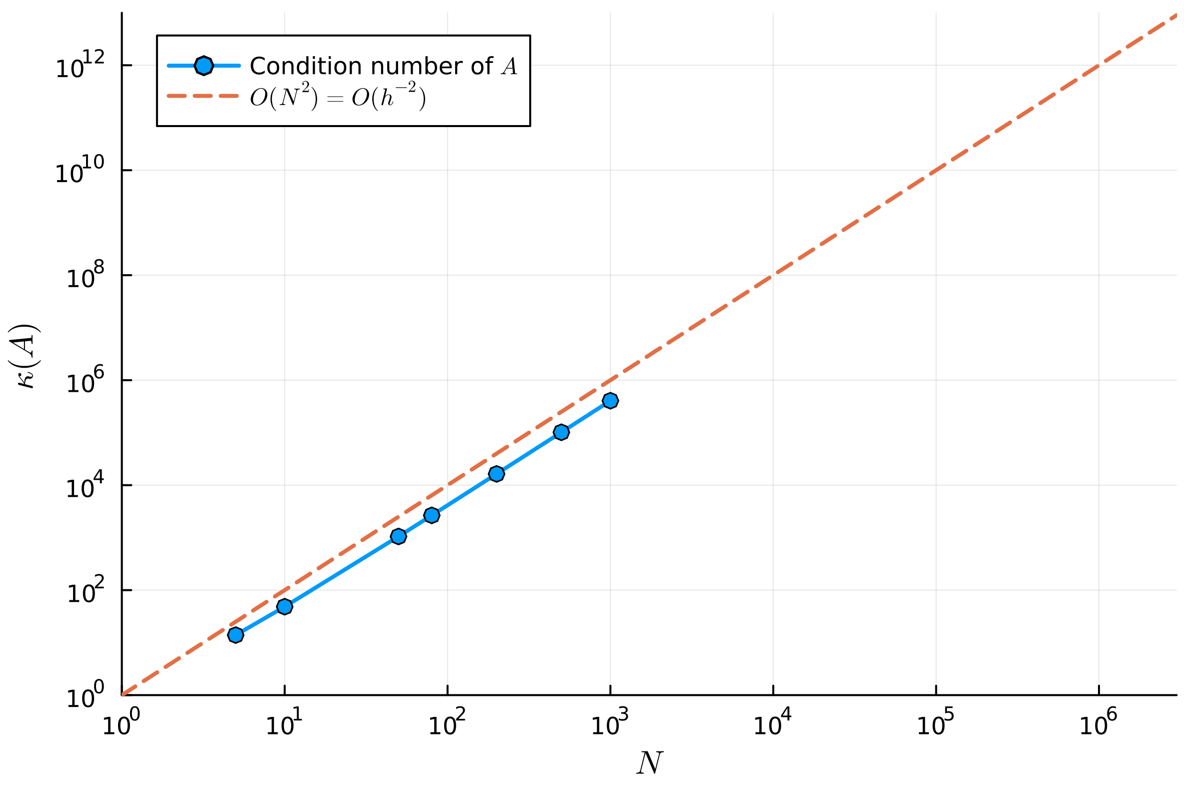


While initially the convergence thus nicely follows the expected convergence curve, **for larger N the convergence degrades and the error starts increasing again**.

Similar to our discussion on numerical stability in the chapter on numerical differentiation this error plot is the result of a balance between two error contributions:

- The **discretisation error** due to the choice of N, where as N gets larger this error **decreases** as $O(N^{-2})$.
- The **error due to finite floating-point precision**, which turns out to increase as N increases.

With respect to the second error contribution the underlying problem is related to solving the linear system $\mathbf{A}\mathbf{u} = \mathbf{b}$ in Algorithm 1. As it turns out the **condition number of the system matrix $\mathbf{A}$ grows** as $O(N^2) = O(h^{-2})$:



As a result **the finer we make the discretisation**, i.e. the **larger N**, the harder it becomes to solve the system $\mathbf{A}\mathbf{u} = \mathbf{b}$, such that the **floating-point error in computing the solution $\mathbf{u}$ itself** increases quadratically.

To make this clear let's consider $N = 10^5$. At this stage the condition number is about 10^{10}, such that the typical 10^{-16} error we make when representing any floating-point operation (such as the computation of $\mathbf{b}$) gets amplified to a relative error in the solution $\mathbf{u}$ of around 10^{-6}: at most around 6 digits of the solution can be known exactly.

As a result around $N = 10^5$ the floating-point error becomes the leading error contribution, such that increasing N beyond 10^5 thus causes the total observed error of the numerical procedure to increase again.

Optional: Error analysis of the discretisation error

Let us come back to the observed quadratic convergence in the regime of small N where the discretisation error dominates. Ignoring thus the floating-point error we conclude that the global error

$$|e_j| = |u(x_j) - u_j| \qquad \text{for } j = 1, \dots, N.$$

should scale as $\max_{j=1,\dots,N} |u(x_j) - u_j| \leq \alpha\, h^2$ for some constant α. In this section we will make this more quantitative.

First, since the value u_j at each nodal point is determined by solving (4), i.e. by replacing the *exact* function values $u(x_{j-1})$, $u(x_j)$ and u_{j+1} by the approximations u_{j-1}, u_j and u_{j+1}, there are in fact two contributions to the global error:

1. The error due to employing the finite difference formula in (3) instead of the exact partial derivative $\frac{\partial^2 u}{\partial x^2}$.
2. The propagation of error from the neighbours u_{j-1} / u_{j+1} to u_j it self and vice versa.

We start by understanding the first contribution, which is the **local error** due to employing the finite difference formula (3).

Definition: Local truncation error

Given the exact solution $u(x_j)$ of problem (9) evaluated at the nodal points $x_j = j\,h$ for $j = 0, \dots, N+1$ of the finite difference scheme of Algorithm 1, we define the **local truncation error** at the node x_j as

$$\tau_j^h = \frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2} - f(x_j) \qquad j = 1, \dots, N. \tag{10}$$

That is the local truncation error is the residual of the numerical scheme (4) when we replace the approximated solution u_j by the exact solution $u(x_j)$.

In our case one can show for the local truncation error:

Lemma 2: Bound on the local truncation error

If the solution u is four times differentiable the local truncation error (10) satisfies

$$\max_{j=1,\dots,N} |\tau_j^h| \leq \frac{h^2}{12} \max_{x\in[0,L]} |u''''(x)| = \frac{h^2}{12} \|u''''\|_\infty. \tag{11}$$

Proof sketch: Since u is the exact solution to (9) we have that $f(x_j) = -\frac{\partial^2 u}{\partial x^2}(x_j)$ for all $j = 1, \dots, N$. Therefore

$$\begin{aligned}
\tau_j^h &= \frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2} - f(x_j) \\
&= \frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2} + \frac{\partial^2 u}{\partial x^2}(x_j) \\
&= -\frac{u(x_j - h) - 2u(x_j) + u(x_j + h)}{h^2} + u''(x_j)
\end{aligned}$$

Considering a Taylor expansion to fourth order of $u(x_j \pm h)$ around x_j than leads to the result.

Next we tackle the second aspect, namely how this local truncation error propagates to give global error.

By rearranging the definition of the local truncation error (10), we find that the exact solution satisfies the discrete problem

$$\frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2} = f(x_j) + \tau_j^h \qquad j = 1, \dots, N. \tag{12}$$

In contrast according to (4) the approximate solution satisfies

$$\frac{-u_{j-1} + 2u_j - u_{j+1}}{h^2} = f(x_j) \qquad j = 1, \dots, N. \tag{4}$$

Subtracting (4) from (12) thus leads to

$$\frac{-e_{j-1} + 2e_j - e_{j+1}}{h^2} = \tau_j^h \qquad j = 1, \dots, N,$$

where we used the definition of the error $e_j = u(x_j) - u_j$. At the boundary, where we set $u_0 = b_0 = u(0)$ and $u_{N+1} = b_L = u(L)$ we have $e_0 = e_{N+1} = 0$, such that the error satisfies

$$\begin{cases} \dfrac{2e_1 - e_2}{h^2} = \tau_1^h & j = 1 \\ \dfrac{-e_{j-1} + 2e_j - e_{j+1}}{h^2} = \tau_j^h & \forall\, 2 \leq j \leq N-1 \\ \dfrac{-e_{N-1} + 2e_N}{h^2} = \tau_N^h & j = N, \end{cases} \tag{13}$$

which is again a discretised Dirichlet bounary value problem (5) with the conditions that $f(x_j) = \tau_j^h$ and $b_0 = b_L = 0$.

We can therefore apply equation (14) of Theorem 1 (see the folded section *Optional: Does the problem (5) always have a solution ?*) to problem (13) and conclude

$$\max_{j=1,\dots,N} |e_j| \leq \frac{1}{8} \max_{j=1,\dots,N} |\tau_j^h|, \tag{15}$$

which relates the local truncation error to the error at each nodal point.

Combining this result with Lemma 2 yields

Theorem 3: Global error of finite differences

Under the assumption that the exact solution u of the Dirichlet boundary value problem (9) is four times differentiable, the finite differences scheme of Algorithm 1 converges as

$$\max_{j=1,\dots,N} |u(x_j) - u_j| \leq \alpha\, h^2 \tag{16}$$

where $\alpha = \frac{1}{96} \|u''''\|_\infty$. It thus achieves **quadratic convergence**.

⚠ TODO ⚠

Show application of finite difference method to a second example Boundary Value Problem. A good example could be the Allan-Cahn equation (Demo 10.5.5 in Driscoll Brown)

```
TODO("Show application of finite difference method to a second example Boundary
Value Problem.

     A good example could be the Allan-Cahn equation (Demo 10.5.5 in Driscoll Brown)

")
```

Galerkin methods

Let us return to the general heat equation using Dirichlet boundary conditions:

$$\begin{cases} -k \dfrac{\partial^2 u}{\partial x^2}(x) = f(x) \qquad x \in (0, L). \\ \qquad u(0) = b_0, \quad u(L) = b_L \end{cases} \tag{17}$$

We saw that **with finite differences** the increasing condition number of the system matrix $\mathbf{A}$ as we increase N makes it **impossible to obtain the solution to arbitrary accuracy**. In the above example we were unable to obtain a solution to higher accuracy than 10^{-10}, irrespective of what value for N we chose. In this section we will develop an alternative approach to solve boundary value problems, which is more general.

Let us assume we are given some smooth function $\psi(x)$, the so-called **test function**, for which we will additionally assume that it **vanishes at the boundary**, i.e. $\psi(0) = \psi(L) = 0$. If we multiply the first line of (17) by this function and integrate over the full computational domain $[0, L]$, this yields:

$$\begin{aligned} \int_0^L f(x)\psi(x)\,dx &= \int_0^L -k\,u''(x)\psi(x)\,dx \\ &\overset{(*)}{=} \left[-k\,u'(x)\psi(x)\right]_0^L + \int_0^L ku'(x)\psi'(x)\,dx \\ &= \int_0^L ku'(x)\psi'(x)\,dx \end{aligned} \tag{18}$$

where we used partial integration in step $(*)$ and the property $\psi(0) = \psi(L) = 0$ in the final step.

Let us define the **set of all test functions** as

$$V_0 = \Big\{ \psi : [0, L] \to \mathbb{R} \;\Big|\; \psi \text{ smooth and } \psi(0) = \psi(L) = 0 \Big\}.$$

If $u : [0, L] \to \mathbb{R}$ is a solution to (17) than our discussion implies that (18) is satisfied for all functions from V_0:

$$\begin{cases} \displaystyle\int_0^L k\, u'(x)\, \psi'(x)\, dx = \int_0^L f(x)\, \psi(x)\, dx & \forall\, \psi \in V_0 \\ u(0) = b_0, \quad u(L) = b_L \end{cases} \tag{19}$$

Equation (19) is known as the **weak form** of the 1D heat equation equation (17) and solving it is interesting in its own right:

Definition: Weak form and weak solution

If a function $u : [0, L] \to \mathbb{R}$ satisfies

$$\begin{cases} \displaystyle\int_0^L k\, u'(x)\, \psi'(x)\, dx = \int_0^L f(x)\, \psi(x)\, dx & \forall\, \psi \in V_0 \\ u(0) = b_0, \quad u(L) = b_L, \end{cases}$$

for all choices of the test function ψ we call u a **weak solution** of the boundary value problem (17).

The original problem (17) is additionally called the **strong form** of the heat equation BVP and its solution u the **strong solution**.

Observation: Strong and weak solutions

If u is a solution to the strong form (17) of a BVP, than it is also a solution to the weak form (19). However, **not every weak solution is also a strong solution**.

The weak form (19) looks unusual at first sight and additionally one might wonder why it is useful, since it can produce solutions, which do not satisfy the original strong problem (17).

But surprisingly for many problems from physics or engineering the situation turns out to be the reverse: it turns out that the somewhat "looser" form provided by **the weak formulation is actually more physical**. In some cases — such as the atomistic modelling of materials — using the strong form can lead to unphysical artifacts, which are in fact avoided when using the weak form.

While we do not have time to discuss this further in this course, the interested reader is referred to the master course MATH-500: Error control in scientific modelling where some of this is discussed.

Discretisation of the weak form

In order to solve (19) our goal is to rewrite the problem in the form of linear algebra — vectors and matrices. For simplicity we will only consider the case $b_0 = b_L = 0$ in this section, i.e. we restrict ourselves to the **heat equation with a zero Dirichlet boundary**

$$\begin{cases} -k\,\dfrac{\partial^2 u}{\partial x^2}(x) = f(x) \qquad x \in (0, L). \\ \qquad u(0) = u(L) = 0 \end{cases} \tag{20}$$

and **corresponding weak form**

$$\begin{cases} \displaystyle\int_0^L k\,u'(x)\,\psi'(x)\,dx = \int_0^L f(x)\,\psi(x)\,dx \qquad \forall\,\psi \in V_0 \\ \qquad u(0) = u(L) = 0. \end{cases} \tag{21}$$

One difficulty in this equation is that the condition $\forall \psi \in V_0$ corresponds to an infinite number of constraints to satisfy (as the set V_0 has infinitely many members). One idea is to approximate this condition by satisfying only finitely many constraints.

To do so we assume that ψ **can be written as a linear combination**

$$\psi(x) = \sum_{i=1}^m \xi_i\,\varphi_i(x), \tag{22}$$

where $\varphi_1, \varphi_2, \ldots, \varphi_m$ are a selection of m linearly independent functions from V_0, i.e. smooth functions which each satisfy $\varphi_i(0) = \varphi_i(L) = 0$. Inserting (22) into the first line of (21) leads to

$$\begin{aligned} 0 &= \int_0^L k\,u'(x)\,\psi'(x) - f(x)\psi(x)\,dx \\ &= \sum_{i=1}^m \xi_i \left[\int_0^L k\,u'(x)\,\varphi_i'(x) - f(x)\,\varphi_i(x)\,dx\right] \end{aligned}$$

which should be true **independent of the values of** ξ_i. As a result the above condition can be achieved if and only if the term in the **square bracket is zero all our functions** φ_i, that is if

$$\forall i = 1, \dots, m: \qquad \int_0^L k\, u'(x)\, \varphi_i'(x)\, dx = \int_0^L f(x)\, \varphi_i(x)\, dx. \tag{23}$$

Notice, that the condition (23) is independent of the actual values taken by the coefficients $\{\xi_i\}_{i=1}^m$. Provided that (23) holds we thus do not need to worry about determining the values of $\{\xi_i\}_{i=1}^m$. In fact in all following development these coefficients will play no further role.

With this development we can replace the infinite number of constraints encoded by the $\forall \psi \in V_0$ in equation (21) by the approximate **version (23)**, which **only involves** $m < \infty$ **conditions** to satisfy.

Considering $u(x)$ we make an additional approximation, namely that — similar to ψ — this function can **also be approximated by a linear combination of finitely many functions**. Here, for simplicity we employ the same selection of functions, that we used for ψ leading to an ansatz

$$u(x) = \sum_{j=1}^m c_j \varphi_j(x). \tag{24}$$

Inserting (24) into (23) we obtain

$$\forall i = 1, \dots, m: \qquad \int_0^L k \sum_{j=1}^m c_j \varphi_j'(x)\, \varphi_i'(x)\, dx = \int_0^L f(x)\, \varphi_i(x)\, dx$$

or arranged differently

$$\forall i = 1, \dots, m: \qquad \sum_{j=1}^m c_j \int_0^L k\, \varphi_j'(x)\, \varphi_i'(x)\, dx = \int_0^L f(x)\, \varphi_i(x)\, dx \tag{25}$$

Since $\varphi_j(0) = \varphi_j(L) = 0$ an immediate consequence is that such a u satisfies the boundary condition $u(0) = u(L) = 0$ independent of the choice of the coefficients c_j. Solving equation (25) therefore automatically ensures that the second line of the weak problem (21) is satisfied. Additionally this equation satisfies (23) — our approximation to the first line of (21).

In summary a **solution to equation (25)** thus **satisfies** (our approximation to) **both conditions required to be a solution to the weak problem** (21). Equation (25) is thus given a special name:

Definition: Galerkin conditions

Given a basis $\varphi_1, \varphi_2, \ldots, \varphi_m$ of m linearly independent functions from the test function set V_0, the linear combination $u = \sum_{j=1}^m c_j \varphi_j$ is an approximate solution to the weak problem (21) if the coefficients $c_1, c_2, \ldots, c_m$ satisfy the **Galerkin conditions**

$$\forall i = 1, \ldots, m: \qquad \sum_{j=1}^m c_j \int_0^L k\, \varphi_j'(x)\, \varphi_i'(x)\, dx = \int_0^L f(x)\, \varphi_i(x)\, dx \tag{25}$$

Collecting the unknown coefficients into a vector $\mathbf{c} = (c_1, c_2, \ldots, c_m)^T \in \mathbb{R}^m$ and introducing the matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ and the vector $\mathbf{f} \in \mathbb{R}^m$ with elements

$$A_{ij} = \int_0^L k\, \varphi_i'(x)\, \varphi_j'(x)\, dx, \qquad f_i = \int_0^L f(x)\, \varphi_i(x)\, dx$$

we can also write the Galerkin conditions more conveniently as the linear system

$$\mathbf{A}\mathbf{c} = \mathbf{f}, \tag{26}$$

which one needs to solve for $\mathbf{c}$.

Example: Sine basis

Let us consider the Dirichlet heat equation

$$\begin{cases} -\dfrac{\partial^2 u}{\partial x^2}(x) = x \qquad x \in (0, \pi) \\ \qquad u(0) = u(L) = 0, \end{cases}$$

i.e. where $k = 1$, $L = \pi$ and $f(x) = x$.

For the basis functions φ_j we select the sine basis

$$\varphi_j(x) = \sin(j\,x) \qquad \text{for } j = 1, \ldots, m.$$

It is easy to see that all of them satisfy $\varphi_j(0) = \varphi_j(2\pi) = 0$. With these basis functions we can obtain the entries of $\mathbf{A}$ and $\mathbf{f}$ in equation (26) as

$$
\begin{aligned}
A_{ij} &= \int_0^\pi \frac{d}{dx}\big[\sin(i\,x)\big]\,\frac{d}{dx}\big[\sin(j\,x)\big]\,dx \\
&= i\,j\int_0^\pi \cos(i\,x)\cos(j\,x)\,dx = \frac{\pi\,i^2}{2}\,\delta_{ij} = \begin{cases} \frac{\pi\,i^2}{2} & i=j \\ 0 & i\neq j \end{cases}
\end{aligned}
$$

$$
f_i = \int_0^\pi x\sin(i\,x)\,dx = \frac{(-1)^{i+1}\,\pi}{i}
$$

For example for $m=5$ we find the matrix and vector

$$
\mathbf{A} = \frac{\pi}{2}\begin{pmatrix} 1 & & & & \\ & 4 & & & \\ & & 9 & & \\ & & & 16 & \\ & & & & 25 \end{pmatrix} \qquad \mathbf{f} = \pi\begin{pmatrix} 1 \\ -\frac{1}{2} \\ \frac{1}{3} \\ -\frac{1}{4} \\ \frac{1}{5} \end{pmatrix}.
$$

Since $\mathbf{A}$ is diagonal we can directly compute the coefficient vector $\mathbf{c}$ as

$$
\mathbf{c} = \begin{pmatrix} f_1/A_{11} \\ f_2/A_{22} \\ f_3/A_{33} \\ f_4/A_{44} \\ f_5/A_{55} \end{pmatrix} = \begin{pmatrix} 2 \\ -\frac{2}{8} \\ \frac{2}{27} \\ -\frac{2}{64} \\ \frac{2}{125} \end{pmatrix}
$$

and the approximate solution becomes

$$
u(x) = 2\sin(x) - \frac{2}{8}\sin(2x) + \frac{2}{27}\sin(3x) - \frac{2}{64}\sin(4x) + \frac{2}{125}\sin(5x).
$$

Generalising to arbitrary m we find the solution coefficients

$$
c_i = \frac{f_i}{A_{ii}} = \frac{2\,(-1)^{i+1}}{i^3} \qquad \forall\, i = 1,\dots,m
$$

leading to the solution function

$$
u(x) = \sum_{i=1}^{m} \frac{2\,(-1)^{i+1}}{i^3}\sin(i\,x), \tag{27}
$$

which can be implemented as:

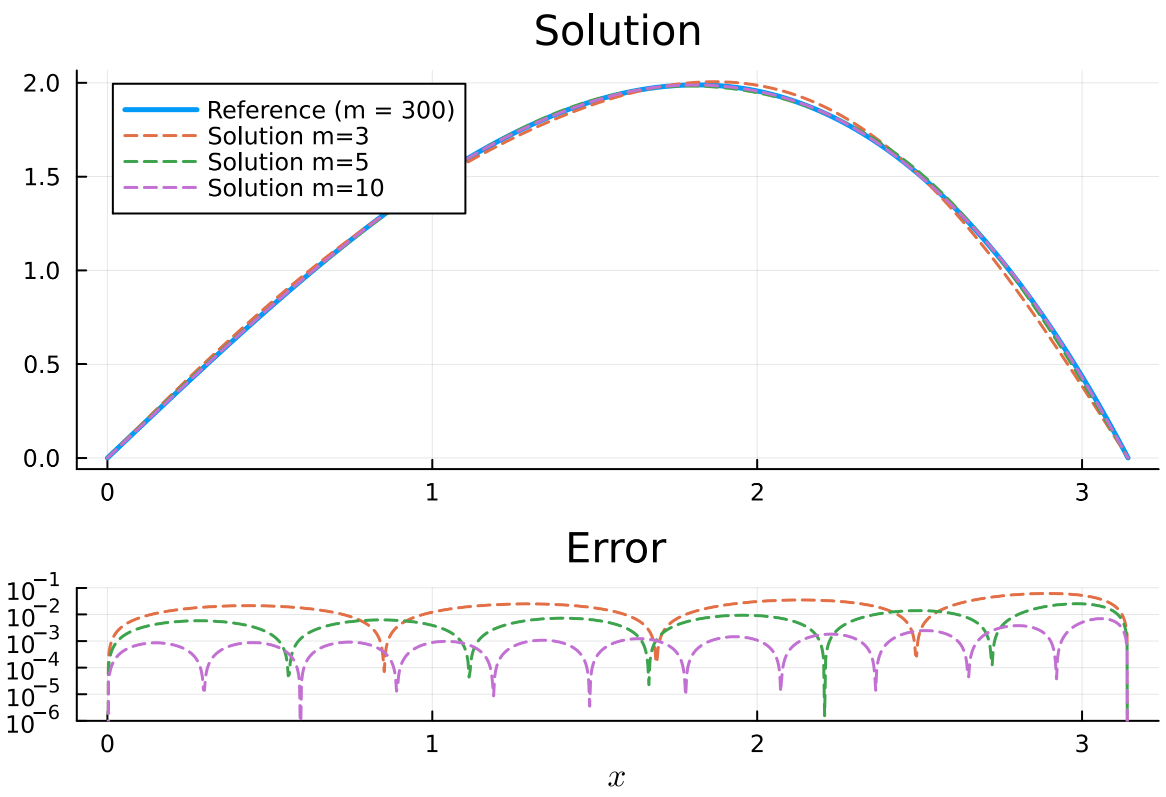
```
sine_solution (generic function with 1 method)
```

```
function sine_solution(m, x)
    # Compute the function value of the numerical solution
    # for given m and x by evaluating (27)

    result = 0.0
    for i in 1:m
        result += 2 * (-1)^(i+1) / i^3 * sin(i * x)
    end
    result
end
```

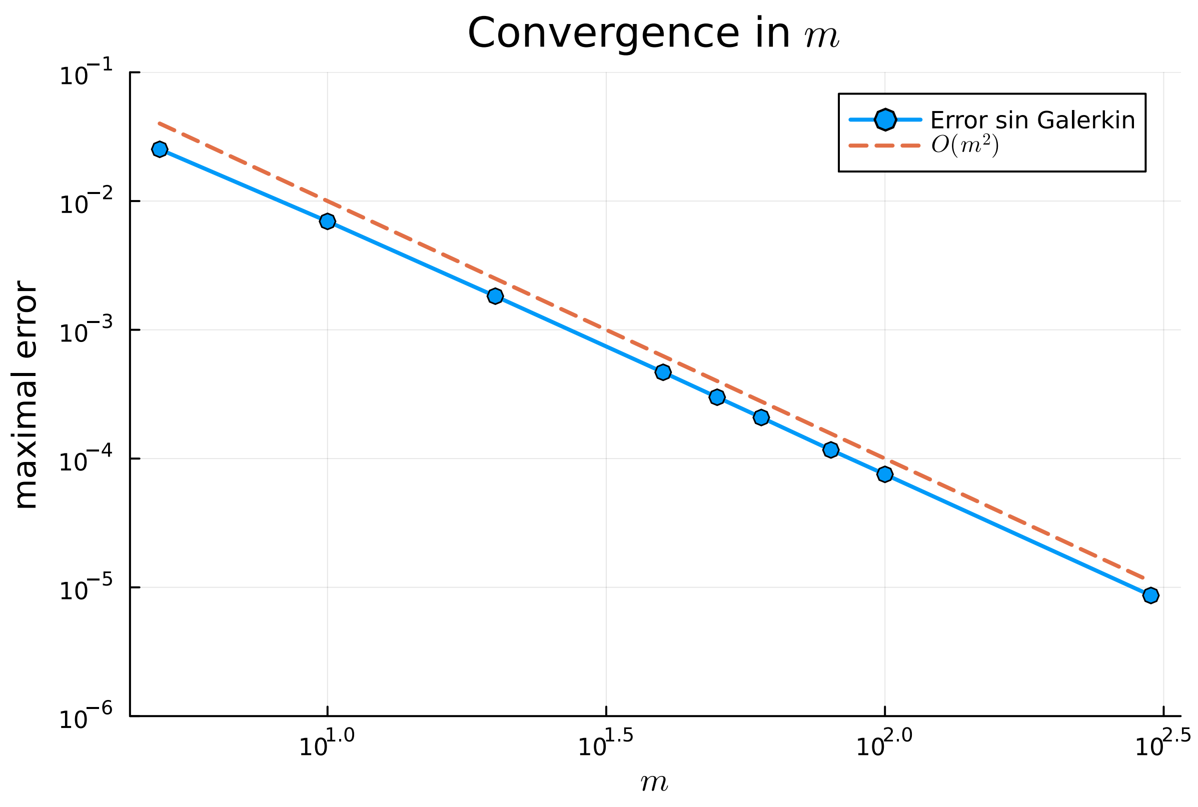
Choosing different values for m we obtain:

- Show reference: ✓
- Show $m = 10$: ✓



We see that already for 5 $\sin$ basis functions it becomes visually very hard to see the difference to the reference solution with 300 basis functions.

Numerically we observe a **quadratic convergence** in a log-log plot:



```julia
let
    x = range(0, π; length=1000)
    reference(x) = sine_solution(800, x)

    ms = [5, 10, 20, 40, 50, 60, 80, 100, 300]
    errors = Float64[]
    for m in ms
        error = [abs(sine_solution(m, xᵢ) - reference(xᵢ)) for xᵢ in x]
        push!(errors, maximum(error))
    end

    p = plot(ms, errors, yaxis=:log, xaxis=:log, lw=2, mark=:o, title=L"Convergence
    in $m$", xlabel=L"m", ylabel="maximal error", label="Error sin Galerkin")
    plot!(p, ms, 1 ./ms.^2, ls=:dash, lw=2, label=L"O(m^2)")

    ylims!(p, (1e-6, 1e-1))
    xticks!(p, 10.0 .^ (0:0.5:3))
    yticks!(p, 10.0 .^ (0:-1:-6))

    p
end
```

Comparison and conclusion

⚠ TODO ⚠

Some conclusion chapter to provide direct comparison between finite differences and Galerkin methods and from that some conclusion why Galerkin methods can be better (numerical stability, less unknowns for a targeted accuracy etc.)

```
TODO("Some conclusion chapter to provide direct comparison between finite
differences and Galerkin methods and from that some conclusion why Galerkin methods
can be better (numerical stability, less unknowns for a targeted accuracy etc.)")
```

⚠ TODO ⚠

To introduce FEM we need the concept of a mass matrix in the Galerkin development above as the basis functions are not orthonormal. See chapter 10.5 of Driscoll Brown

```
TODO("To introduce FEM we need the concept of a mass matrix in the Galerkin development above as the basis functions are not orthonormal.

     See chapter 10.5 of Driscoll Brown
     ")
```

Optional: Finite elements

A widely employed set of basis functions for Galerkin approximations are the hat functions $\varphi_i = H_i$, which we already discussed in the chapter on Interpolation (chapter 7). Recall, that given a set of nodes $x_0 < x_1 < \cdots < x_n$ the hat functions are defined as

$$H_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & \text{if } i > 0 \text{ and } x \in [x_{i-1}, x_i] \\ \frac{x_{i+1}-x}{x_{i+1}-x_i} & \text{if } i < n \text{ and } x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise} \end{cases}$$

for $i = 0, \ldots, n$. In this discussion we will again only consider equispaced nodes for simplicity, i.e. we take the case with $0 = x_0 < x_1 < \cdots < x_n = L$ and where $x_{i+1} - x_i = \frac{L}{n}$. Defining $h = \frac{L}{n}$ the hat functions can thus equally be expressed as

$$H_i(x) = \begin{cases} \frac{x-x_{i-1}}{\color{red}{h}} & \text{if } i > 0 \text{ and } x \in [x_{i-1}, x_i] \\ \frac{x_{i+1}-x}{\color{red}{h}} & \text{if } i < n \text{ and } x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

for $i = 0, \ldots, n$.

Due to the cardinality property of the hat functions $\varphi_i = H_i(x)$, i.e. $H_i(x_j) = \delta_{ij}$, the expansion (24) of the unknown solution function can be simplified to

$$u(x) = \sum_{j=1}^{m} c_j \varphi_j(x) = \sum_{i=\color{red}{1}}^{\color{red}{n-1}} u_i \, H_i(x), \tag{29}$$

where $u_i = u(x_i)$, i.e. the numerical solution at the nodal points. Note that the last sum in (29) is deliberately truncated to omit the hat functions H_0 and H_n. The nodal points of these two functions are $x_0 = 0$ and $x_n = L$, respectively, where these functions take by definition the value 1. As a result these functions are unable to satisfy the condition $0 = \varphi_j(0) = \varphi_j(L)$ underlying V_0 and are thus excluded.

The importance of the hat functions for Galerkin methods stems from the fact that each hat function $H_i(x)$ is only non-zero in the interval $[x_{i-1}, x_{i+1}]$. As a result when evaluating the Galerkin conditions (26) one never has to integrate over the entire domain $[0, L]$, but only a much smaller subset of this interval where the involved hat functions are non-zero — a noteworthy reduction of the computational effort.

As an example consider the evaluation of the elements of the matrix $\mathbf{A} \in \mathbb{R}^{m\times m} = \mathbb{R}^{(n-1)\times(n-1)}$. First note that the derivatives H_i' are non-zero only where H_i itself is non-zero, i.e. $[x_{i-1}, x_{i+1}]$. Therefore the product $H_i' \, H_j'$ is only non-zero in the intersection of $[x_{i-1}, x_{i+1}]$ and $[x_{j-1}, x_{j+1}]$.

This intersection is empty, thus the matrix element A_{ij} zero, whenever $|i-j| > 2$, i.e. when the two indices are more than two apart from each other. For $|i-j| = 2$ the intersection contains only a single point, which similarly implies $A_{ij} = 0$. As a consequence

$$A_{i,i+d} = \int_0^L k \, H_i'(x) \, H_{i+d}'(x) \, dx = 0 \qquad \text{if } d \geq 2.$$

where $i = 1, \ldots, n-3$ and similarly by symmetry

$$A_{i+d,i} = \int_0^L k \, H_{i+d}'(x) \, H_i'(x) \, dx = 0 \qquad \text{if } d \geq 2.$$

again for $i = 1, \ldots, n-3$

Now we consider the remaining three cases, namely

$$
\begin{aligned}
A_{ii} &= \int_0^L k\, H_i'(x)\, H_i'(x)\, dx \\
&= \int_{x_{i-1}}^{x_i} k\, H_i'(x)\, H_i'(x)\, dx + \int_{x_i}^{x_{i+1}} k\, H_i'(x)\, H_i'(x)\, dx \\
&= \int_{x_{i-1}}^{x_i} k \cdot \frac{1}{h} \cdot \frac{1}{h}\, dx + \int_{x_i}^{x_{i+1}} k \cdot \frac{-1}{h} \cdot \frac{-1}{h}\, dx \\
&= \frac{k}{h^2}\left[\int_{x_{i-1}}^{x_i} 1\, dx + \int_{x_i}^{x_{i+1}} 1\, dx\right] \\
&= \frac{2k}{h}
\end{aligned}
$$

for $i = 1, \ldots, n-1$ as well as

$$
\begin{aligned}
A_{i+1,i} = A_{i,i+1} &= \int_0^L k\, H_i'(x)\, H_{i+1}'(x)\, dx \\
&= \int_{x_i}^{x_{i+1}} k\, H_i'(x)\, H_{i+1}'(x)\, dx \\
&= \int_{x_i}^{x_{i+1}} k \cdot \frac{-1}{h} \cdot \frac{1}{h}\, dx \\
&= -\frac{k}{h}.
\end{aligned}
$$

for $i = 1, \ldots, n-2$. We recover a tridiagonal matrix

$$
\mathbf{A} = \frac{k}{h} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{(n-1)\times(n-1)} \tag{30}
$$

⚠ TODO ⚠

Properly work the mass matrix stuff into this.

```
TODO("Properly work the mass matrix stuff into this.")
```

Mass matrix

$$
\begin{aligned}
M_{ii} &= \int_0^L H_i(x)\, H_i(x)\, dx \\
&= \int_{x_{i-1}}^{x_i} H_i(x)\, H_i(x)\, dx + \int_{x_i}^{x_{i+1}} H_i(x)\, H_i(x)\, dx \\
&= \int_{x_{i-1}}^{x_i} \frac{(x - x_{i-1})(x - x_{i-1})}{h^2}\, dx + \int_{x_i}^{x_{i+1}} \frac{(x_{i+1} - x)(x_{i+1} - x)}{h^2}\, dx \\
&= \frac{(x_i - x_{i-1})^3}{3h^2} + \frac{(x_{i+1} - x_i)^3}{3h^2} \\
&= \frac{h^3}{3h^2} + \frac{h^3}{3h^2} \\
&= \frac{2}{3} h
\end{aligned}
$$

and

$$
\begin{aligned}
M_{i,i+1} &= \int_0^L H_i(x)\, H_{i+1}(x)\, dx = \int_{x_i}^{x_{i+1}} H_i(x)\, H_{i+1}(x)\, dx \\
&= \int_{x_i}^{x_{i+1}} \frac{(x_{i+1} - x)(x - x_{i+1})}{h^2}\, dx \\
&= -\frac{(x_{i+1} - x_i)^3}{3h^2} \\
&= -\frac{h}{3}
\end{aligned}
$$

Finally we consider the elements of the vector $\mathbf{f}$. Since again H_i is only non-zero in $[i-1, i+1]$, we need to perform the two integrals

$$f_i = \int_0^L f(x)\, H_i(x)\, dx = \int_{x_{i-1}}^{x_i} f(x)\, H_i(x)\, dx + \int_{x_i}^{x_{i+1}} f(x)\, H_i(x)\, dx$$

for $i = 1, \ldots, n-1$. One can show that to a very good approximation one can replace $f(x)$ by the average value of the function over the respective integrals, i.e.

$$\begin{aligned} \int_{x_{i-1}}^{x_i} f(x)\, H_i(x)\, dx &\approx \frac{f(x_{i-1}) + f(x_i)}{2} \int_{x_{i-1}}^{x_i} H_i(x)\, dx = \frac{f(x_{i-1}) + f(x_i)}{2} \cdot \frac{h}{2} \\ \int_{x_i}^{x_{i+1}} f(x)\, H_i(x)\, dx &\approx \frac{f(x_i) + f(x_{i+1})}{2} \int_{x_i}^{x_{i+1}} H_i(x)\, dx = \frac{f(x_i) + f(x_{i+1})}{2} \cdot \frac{h}{2}. \end{aligned}$$

In particular this approximation converges quadratically as $h \to 0$, which turns out to be exactly the order of approximation of the finite element method itself. Therefore putting in additional efforts to compute these integrals more accurately would not even improve the accuracy of our final result.

Putting these developments tothere we obtain for $i = 1, \ldots, n-1$ that

$$\begin{aligned} f_i &= \frac{f(x_i) + f(x_{i-1})}{2} \frac{h}{2} + \frac{f(x_{i+1}) + f(x_i)}{2} \frac{h}{2} \\ &= \frac{h}{4} \Big(f(x_{i-1}) + 2 f(x_i) + f(x_{i+1}) \Big). \end{aligned} \tag{31}$$

By solving the linear system

$$(\mathbf{A} + \mathbf{M})\, \mathbf{u} = \mathbf{f}$$

we then obtain the coefficients $\mathbf{u} = (u_1, u_2, \ldots, u_{n-1})^T$ in the expansion (29). Notably, due to the cardinality property of the hat functions $u_i = u(x_i)$, i.e. these coefficients are equal to evaluating our numerical approximation $u(x)$ at the nodal points $\{x_i\}_{i=1}^{n-1}$.

An implementation of this approch is given below:

```
heat_equation_1d_fem (generic function with 1 method)
```

```julia
function heat_equation_1d_fem(f, k, L, n)
    # f:  Function describing the external heat source
    # k:  thermal conductivity
    # L:  Length of the metal rod
    # N:  Number of nodal points for the finite element scheme

    h = L / n                      # Step size
    x = [i * h for i in 0:n]       # Nodal points (x₀, x₁, ..., xₙ)
    x_inner = x[2:n]               # Inner nodal points (x₁, ..., xₙ₋₁)


    # Build A and f: Note that these are a (n-1) × (n-1) matrix
    # respectively a vector of length (n-1)
    A = k/h * SymTridiagonal(2ones(n-1), -ones(n-2))
    M = h/3 * SymTridiagonal(2ones(n-1), -ones(n-2))
    f = h/4 * [f(x[i-1]) + 2f(x[i]) + f(x[i+1])
               for i in 2:n]    # Skip first/last nodal point, i.e. x[1] = x₀

    u = (A + M) \ f
    (; x=x_inner, u)
end
```

Let us return to the example we considered with the sine basis, i.e.

$$\begin{cases} -\dfrac{\partial^2 u}{\partial x^2}(x) = x \qquad x \in (0, \pi) \\ \qquad u(0) = u(L) = 0, \end{cases}$$

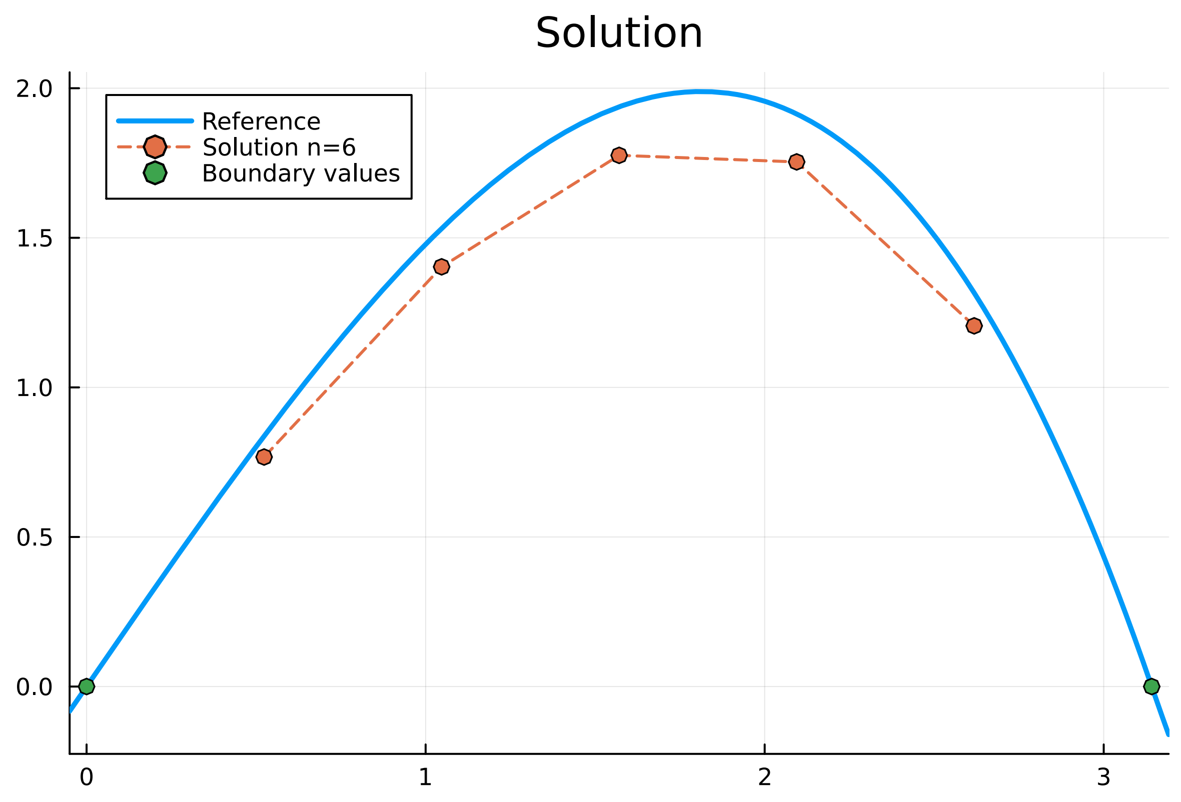
i.e. the Dirichlet heat equation (20) with $k = 1$, $L = \pi$ and $f(x) = x$.

As a reference solution we consider the solution using the sine basis with $m = 300$, which we know to be very accurate for this problem:

```
reference (generic function with 1 method)
```

```julia
reference(x) = sine_solution(300, x)
```

- n = ● 6

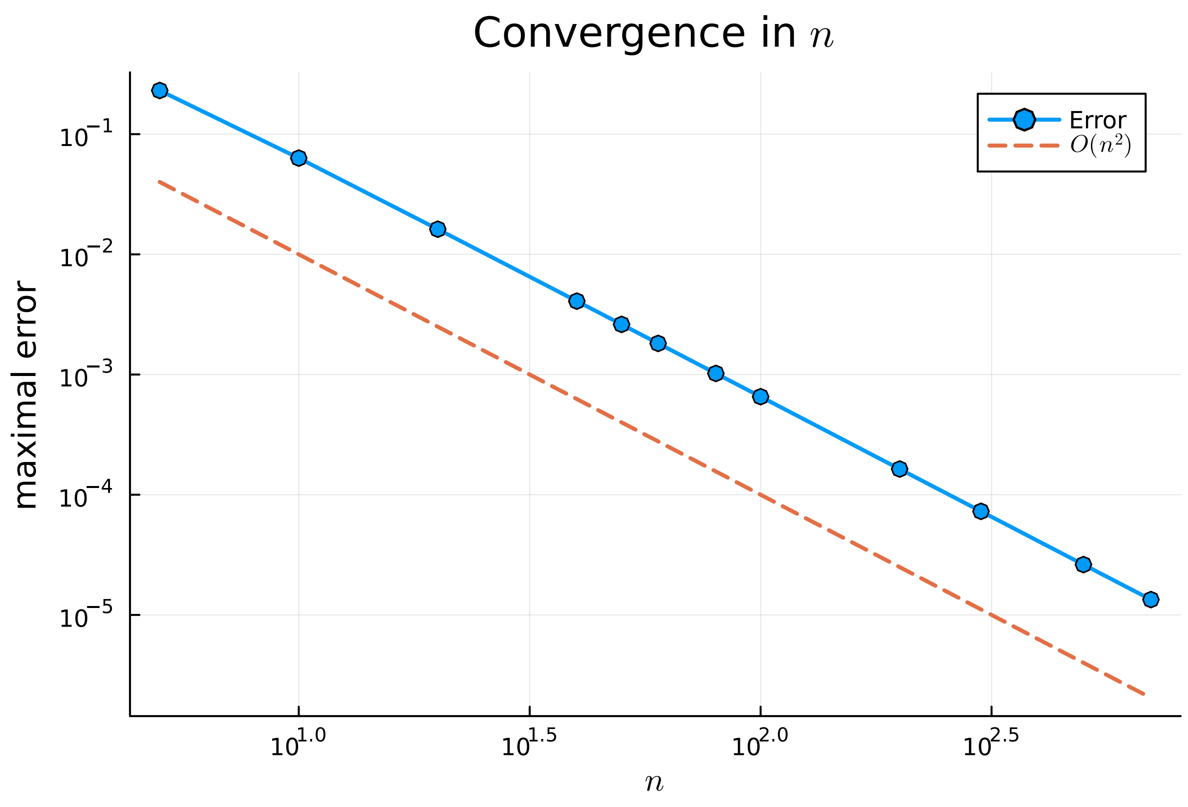


```julia
let
    # Parameters of the problem
    f(x) = x
    k = 1
    L = π

    # Plot the reference
    p = plot(reference; lw=2.5, label="Reference", title="Solution", xlims=(-0.05,
    L + 0.05))

    # Compute FEM solution and plot it
    result = heat_equation_1d_fem(f, k, L, n)
    plot!(p, result.x, result.u; lw=1.5, ls=:dash, label="Solution n=$n", mark=:o)

    scatter!([0, L], [0, 0], label="Boundary values", mark=:o, c=3)
end
```



```julia
let
    f(x) = x
    k = 1
    L = π

    ns = [5, 10, 20, 40, 50, 60, 80, 100, 200, 300, 500, 700]
    errors = Float64[]
    for n in ns
        result = heat_equation_1d_fem(f, k, L, n)
        error = abs.(result.u - reference.(result.x))
        push!(errors, maximum(error))
    end

    p = plot(ns, errors, yaxis=:log, xaxis=:log, lw=2, mark=:o, title=L"Convergence
    in $n$", xlabel=L"n", ylabel="maximal error", label="Error")
    plot!(p, ns, 1 ./ns.^2, ls=:dash, lw=2, label=L"O(n^2)")

    xticks!(p, 10.0 .^ (0:0.5:3))
    yticks!(p, 10.0 .^ (0:-1:-6))
end
```

The finite element method is one of the most widely employed approaches in science and engineering to solve differential equations. With this short instroduction we only scratched the surface.

More information on the approach can be found for example in chapter 10.6 of Driscoll, Brown: *Fundamentals of Numerical Computation*.