12 The Finite Element method

Lab Objective: *The finite element method is commonly used for numerically solving partial differential equations. We introduce the finite element method via a simple BVP describing the steady state distribution of heat in a pipe as fluid flows through.*

Advection-Diffusion of Heat in a Fluid

We wish to study the distribution of heat in a fluid that is moving at some constant speed a. Let y denote the temperature of the fluid at any given location and time. The equation modeling this situation can be obtained from the differential form of the conservation law, where the flux is the sum of a diffusive term $-\varepsilon y_x$ and an advection (or transport) term ay:

$$J = ay - \varepsilon y_x$$

The one-dimension conservation law states that y must then obey the partial differential equation $y_t + J_x = f(x)$, where f represents heat sources in the system. Since $J_x = ay_x - \varepsilon y_{xx}$, we obtain the *advection-diffusion equation*

$$y_t + ay_x = \varepsilon y_{xx} + f(x).$$

As time progresses, we expect the temperature of the fluid in the pipe to reach a steady state distribution, with $y_t = 0$. Once this steady state has been reached, the heat distribution y then satisfies the ODE

$$\varepsilon y'' - ay' = -f(x).$$

We consider the scenario of a fluid flowing through a pipe from $x = 0$ to $x = 1$ with speed $a = 1$, and as it travels it is warmed at a constant rate $f(x) = 1$. Note that since this a second-order ODE, we need two boundary conditions. Suppose that the fluid is already at a known temperature $y = 2$ as it enters the pipe. This imposes the boundary condition $y(0) = 2$. Suppose further that a device is installed on the end of the pipe that nearly instantaneously brings the heat of the water up to $y = 4$. Physically, we expect this extra heat that is introduced at $x = 1$ to diffuse backward through the water in the pipe and thus influence the steady-state temperature. Putting this together leads to a well defined BVP:

$$\begin{aligned} &\varepsilon y'' - y' = -1, \quad 0 < x < 1, \\ &y(0) = 2, \quad y(1) = 4. \end{aligned} \tag{12.1}$$

The analytic solution for $\varepsilon = 0.1$ is shown in Figure 12.1.

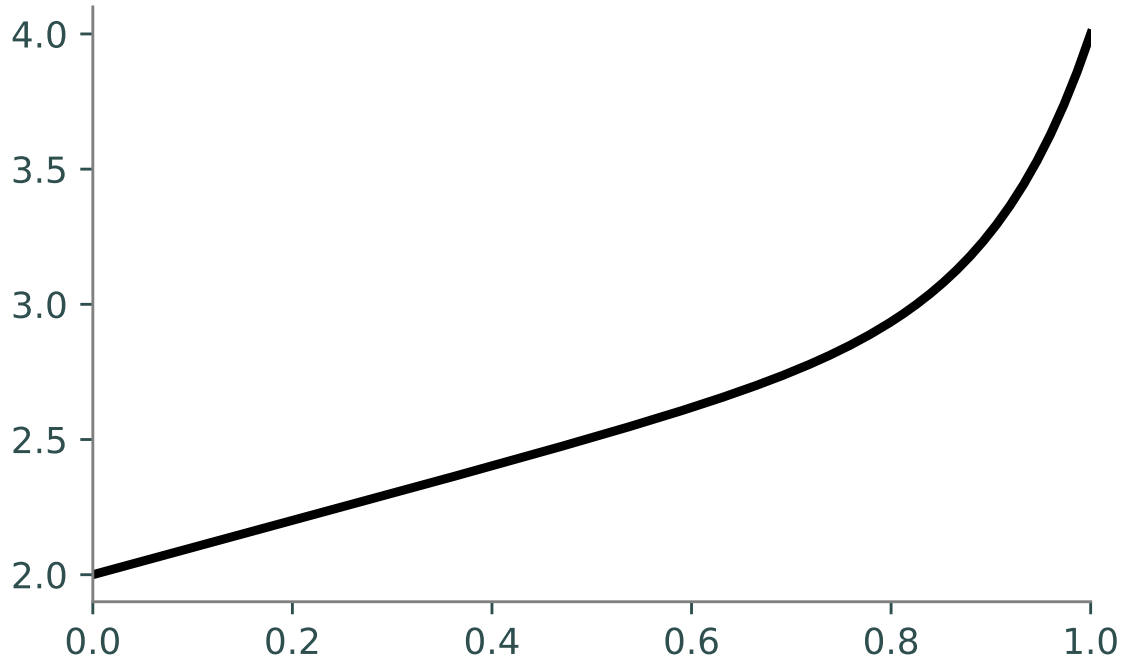

Figure 12.1: The analytic solution of (12.1) for $\varepsilon = 0.1$.

The Weak Formulation

Stepping back momentarily, consider the equation

$$\begin{aligned} &\varepsilon y'' - y' = -f, \quad 0 < x < 1, \\ &y(0) = \alpha, \quad y(1) = \beta. \end{aligned} \tag{12.2}$$

To approximate the solution y using the finite element method, we reframe the problem into one involving integrals, known as its *weak formulation*.

Let w be a smooth function on $[0, 1]$ satisfying $w(0) = w(1) = 0$. Multiplying (12.2) by w and integrating over $[0, 1]$ yields

$$\begin{aligned} \int_0^1 -fw\,dx &= \int_0^1 (\varepsilon y''w - y'w)\,dx, \\ &= \int_0^1 (-\varepsilon y'w' - y'w)\,dx, \end{aligned}$$

where the second equality follows by integration by parts. For notational convenience, define the functionals a and l by

$$\begin{aligned} a(y, w) &= \int_0^1 -\varepsilon y'w' - y'w\,dx, \\ l(w) &= \int_0^1 -fw\,dx. \end{aligned}$$

Then, any solution to (12.2) will also satisfy

$$a(y, w) = l(w) \tag{12.3}$$

This equation is the *weak formulation* of (12.2). Note that any solution to the original ODE is also a solution to the weak formulation. However, solutions to the weak formulation need not be solutions to the original ODE, as they may not even be differentiable everywhere. While this may seem like an undesirable property, it allows us to use a wider variety of functions to approximate the true solution.

Now, we choose some appropriate vector space V of functions, and consider the problem of finding a function $y \in V$ that satisfies the weak formulation (12.3) for all $w \in V_0 = \{w \in V | w(0) = w(1) = 0\}$. The *finite element method* consists of choosing V to be some set of piecewise polynomial functions. In this lab, we will consider the case of using piecewise linear functions.

The Finite Element Method

Let P_n be some partition of $[0,1]$, $0 = x_0 < x_1 < \ldots < x_n = 1$, and let V_n be the set of continuous linear piecewise functions v on $[0,1]$ such that v is linear on each subinterval $[x_j, x_{j+1}]$. These subintervals are the finite elements for which this method is named. Note that V_n has dimension $n+1$, since each of the continuous piecewise linear functions in V are uniquely determined by their values at the $n+1$ points $x_0, x_1 \ldots, x_n$. Let $V_{n,0}$ be the subspace of V_n of dimension $n-1$ whose elements are zero at the endpoints of $[0,1]$.

Let the ϕ_i be the hat functions

$$\phi_i(x) = \begin{cases} (x - x_{i-1})/h_i & \text{if } x \in [x_{i-1}, x_i] \\ (x_{i+1} - x)/h_{i+1} & \text{if } x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise} \end{cases}$$

where $h_i = x_i - x_{i-1}$; see Figures 12.2 and 12.3. These hat functions form a basis for V_n. Note that the points $x_0, \ldots, x_n$ need not be evenly spaced, and the h_i do not need to be equal. This is in fact one of the major strengths of this approach, as it allows adapting the points in the partition to the problem, which can reduce the error in the approximation. When applied to PDEs, it also is a simple way to handle unusually-shaped domains.

We now can write our approximate solution for y and the arbitrary function w as a linear combination of these basis elements, which will enable us to solve the system numerically. In particular, we can write $\hat{y}(x) = \sum_{i=0}^{n} k_i \phi_i(x)$, where the k_i are to be determined.

To make things more concrete, consider the case of $n = 5$ with the partition $\mathrm{P}_5 = \{x_0, x_1, \ldots, x_5\}$. We look for an approximation $\hat{y} = \sum_{i=0}^{5} k_i \phi_i \in V_5$ of the true solution y; to do this, we must determine appropriate values for the constants k_i. We impose the condition on $\hat{y}$ that

$$a(\hat{y}, w) = l(w)$$

for all $w \in V_{5,0}$. This can be written equivalently as

$$a\left(\sum_{i=0}^{5} k_i \phi_i, \phi_j\right) = l(\phi_j) \quad \text{for } j = 1, 2, 3, 4,$$

since a and l are linear in w and $\phi_1, \phi_2, \phi_3, \phi_4$ form a basis for $V_{5,0}$. Since a is also linear in y, we further obtain

$$\sum_{i=0}^{5} k_i a(\phi_i, \phi_j) = l(\phi_j) \quad \text{for } j = 1, 2, 3, 4.$$

To satisfy the boundary conditions, we necessarily have that $k_0 = \alpha$, $k_5 = \beta$. These equations can be written together in matrix form as

$$AK = \Phi, \tag{12.4}$$

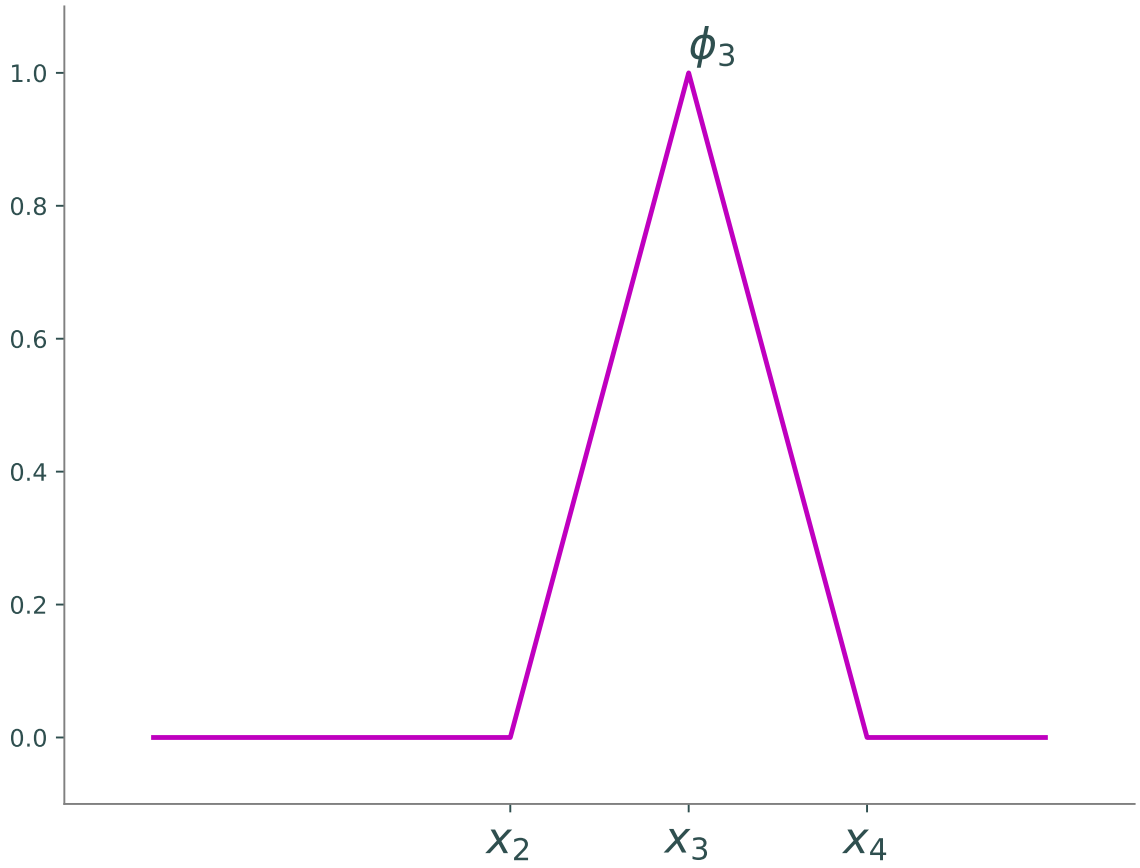

Figure 12.2: The basis function ϕ_3, when the x_i are evenly spaced.

where

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ a(\phi_0, \phi_1) & a(\phi_1, \phi_1) & a(\phi_2, \phi_1) & 0 & 0 & 0 \\ 0 & a(\phi_1, \phi_2) & a(\phi_2, \phi_2) & a(\phi_3, \phi_2) & 0 & 0 \\ 0 & 0 & a(\phi_2, \phi_3) & a(\phi_3, \phi_3) & a(\phi_4, \phi_3) & 0 \\ 0 & 0 & 0 & a(\phi_3, \phi_4) & a(\phi_4, \phi_4) & a(\phi_5, \phi_4) \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$K = \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \end{bmatrix}, \quad \Phi = \begin{bmatrix} \alpha \\ l(\phi_1) \\ l(\phi_2) \\ l(\phi_3) \\ l(\phi_4) \\ \beta \end{bmatrix}.$$

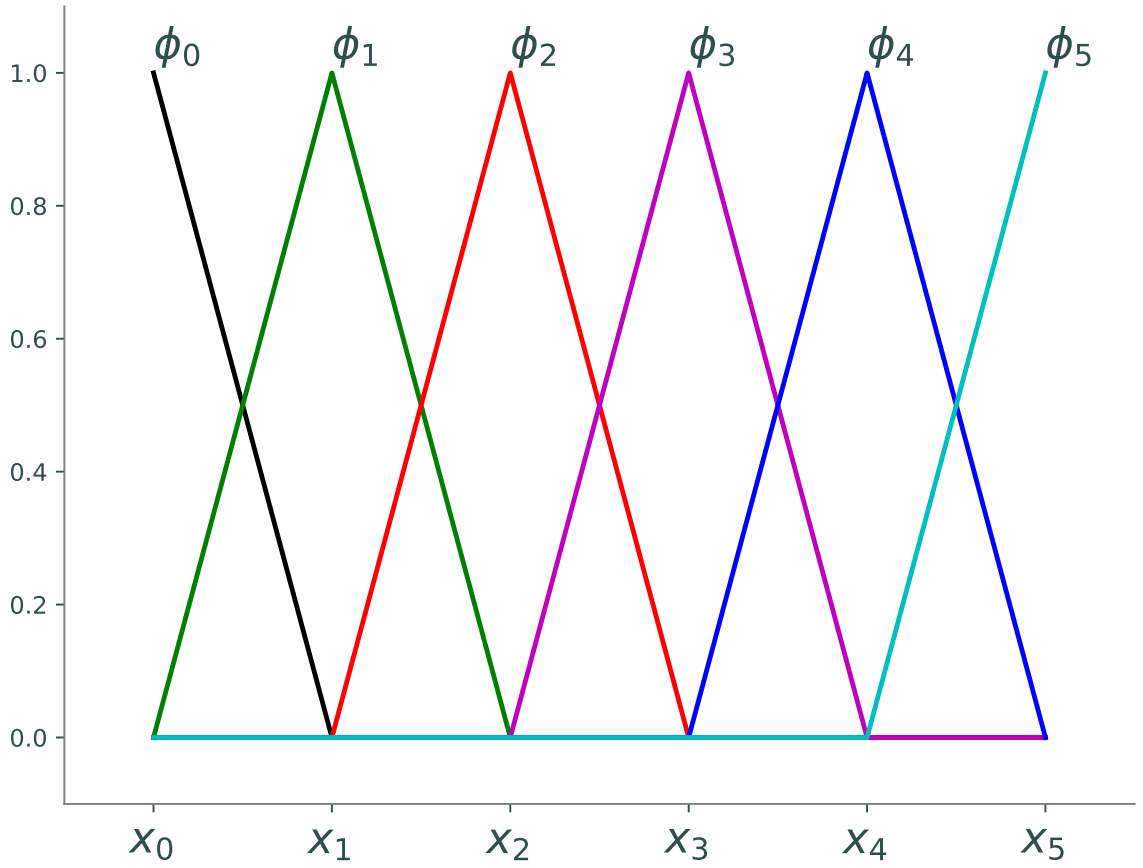

Figure 12.3: The six basis functions for V_5, when the x_i are evenly spaced.

Note that since $a(\phi_i, \phi_j) = 0$ for most values of i, j (in particular, when the hat functions do not have overlapping domains), the finite element method results in a sparse linear system. To compute the coefficients of (12.4) we begin by evaluating some integrals. Since

$$\phi_i(x) = \begin{cases} (x - x_{i-1})/h_i & \text{if } x \in [x_{i-1}, x_i] \\ (x_{i+1} - x)/h_{i+1} & \text{if } x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_i'(x) = \begin{cases} 1/h_i & \text{for } x_{i-1} < x < x_i, \\ -1/h_{i+1} & \text{for } x_i < x < x_{i+1}, \\ 0 & \text{otherwise,} \end{cases}$$

we obtain

$$\int_0^1 \phi_i'\phi_j' = \begin{cases} -1/h_{i+1} & \text{if } j = i+1, \\ 1/h_i + 1/h_{i+1} & \text{if } j = i, \\ 0 & \text{otherwise,} \end{cases}$$

$$\int_0^1 \phi_i'\phi_j = \begin{cases} -1/2 & \text{if } j = i+1, \\ 1/2 & \text{if } j = i-1, \\ 0 & \text{otherwise,} \end{cases}$$

which can be put together to obtain (for $f(x) = 1$)

$$a(\phi_i, \phi_j) = \begin{cases} \varepsilon/h_{i+1} + 1/2 & \text{if } j = i+1, \\ -\varepsilon/h_i - \varepsilon/h_{i+1} & \text{if } j = i, \\ \varepsilon/h_i - 1/2 & \text{if } j = i-1, \\ 0 & \text{otherwise,} \end{cases}$$

$$l(\phi_j) = -\frac{1}{2}(h_j + h_{j+1}).$$

Equation (12.4) may now be solved using any standard linear solver. To handle the large number of elements required for Problem 3, you will want to use sparse matrices from `scipy.sparse`.

Problem 1. Use the finite element method to solve

$$\begin{aligned} \varepsilon y'' - y' &= -1, \\ y(0) = \alpha, \quad y(1) &= \beta, \end{aligned} \tag{12.5}$$

where $\alpha = 2, \beta = 4$, and $\varepsilon = 0.02$. Use $N = 100$ finite elements (101 grid points). Compare your solution with the analytic solution

$$y(x) = \alpha + x + (\beta - \alpha - 1)\frac{e^{x/\varepsilon} - 1}{e^{1/\varepsilon} - 1}.$$

Hint: One additional nice consequence of this setup is that the approximation $\hat{y}$ is exactly the piecewise linear function that connects the points (x_i, k_i). This means that the solution can be plotted very simply using `plt.plot(x, k)`, where `x` and `k` are arrays of the x_i and k_i.

Problem 2. One of the strengths of the finite element method is the ability to generate grids that better suit the problem. The solution of (12.5) changes most rapidly near $x = 1$. Compare the numerical solution when the grid points are unevenly spaced versus when the grid points are clustered in the area of greatest change; see Figure 12.4. Specifically, use the grid points defined by

```
even_grid = np.linspace(0,1,15)
clustered_grid = np.linspace(0,1,15)**(1./8)
```

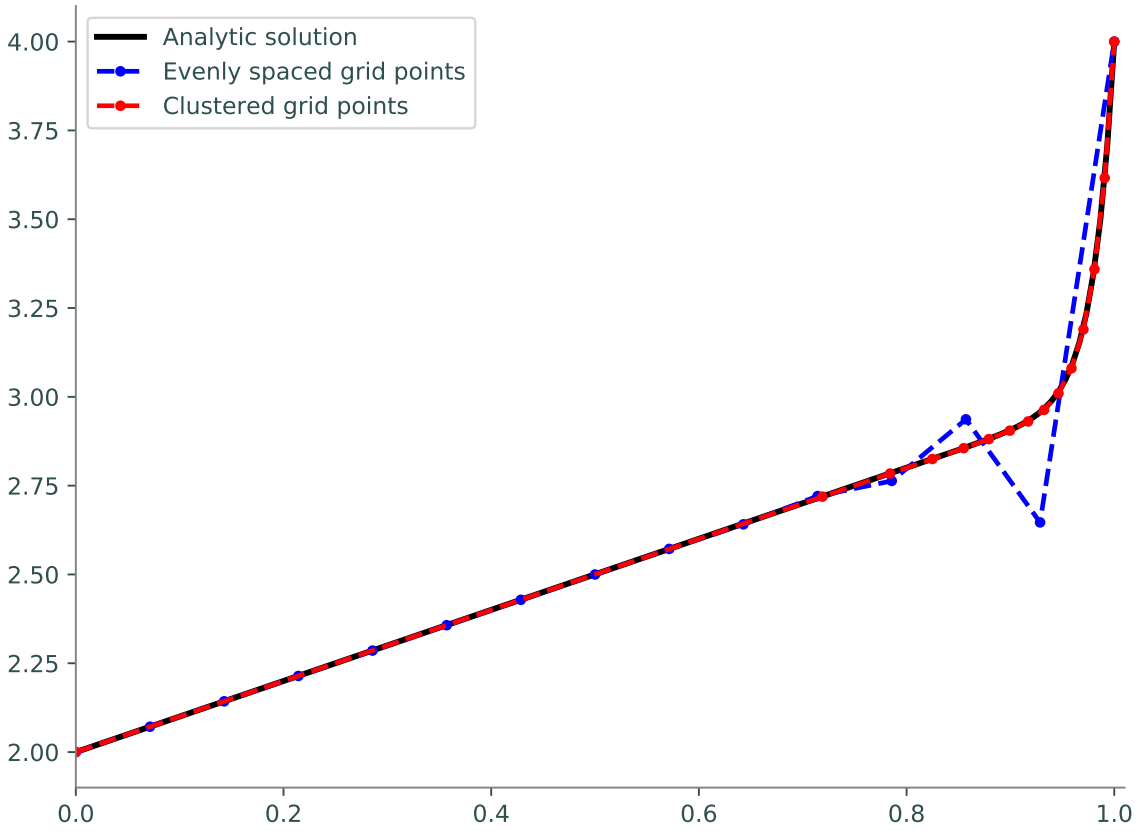

Figure 12.4: Two finite element approximations using 15 grid points, with different spacings.

Problem 3. Higher order methods promise faster convergence, but typically require more work to code. So why do we use them when a low order method will converge just as well, albeit with more grid points? The answer concerns the roundoff error associated with floating point arithmetic. Low order methods generally require more floating point operations, so roundoff error has a much greater effect.

The finite element method introduced here is a second order method, even though the approximate solution is piecewise linear. (To see this, note that if the grid points are evenly spaced, the matrix A in (12.4) is exactly the same as the matrix for the second order centered finite difference method.)

Solve (12.5) with the finite element method using $N = 2^i$ evenly-spaced finite elements, $i = 4, 5, \ldots, 21$. Remember to use sparse matrices, as this greatly reduces the memory and computation needed for the larger N. Compute the error as the maximum absolute value of the difference of the values of the approximate and true solutions at each of the x_i. Use a log-log plot to graph the error, and compare with Figure 12.5.

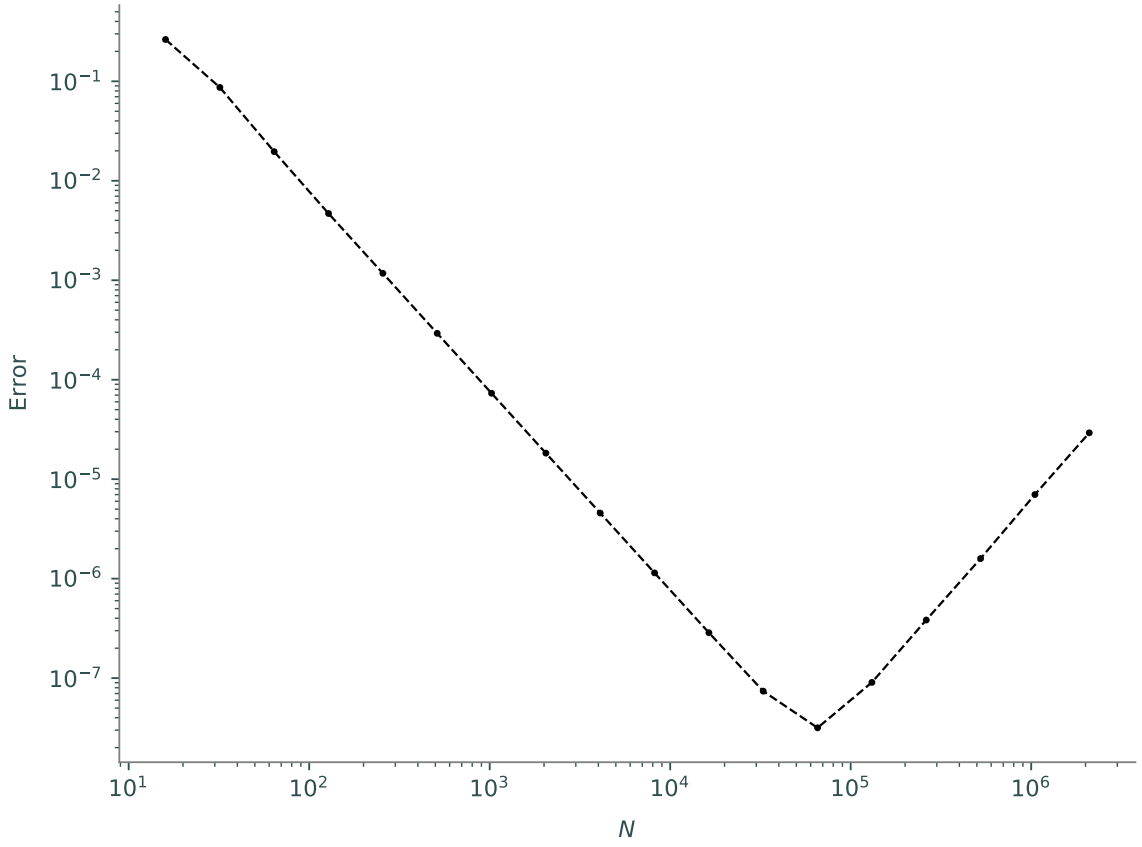

Figure 12.5: Error for the second order finite element method, as the number of subintervals N grows. Round-off error eventually overwhelms the approximation.