5 The Finite Difference Method

A **finite difference** for a function $f(x)$ is an expression of the form $f(x+s) - f(x+t)$. Finite differences can give a good approximation of derivatives.

Suppose we have a function $u(x)$, defined on an interval $[a, b]$. Let $a = x_0, x_1, \ldots x_{n-1}, x_n = b$ be a grid of $n+1$ evenly spaced points, with $x_{i+1} - x_i = h$, where $h = (b-a)/n$.

You are used to seeing the derivative $u'(x)$, which can written in centered-difference form as:

$$u'(x) = \lim_{h \to \infty} \frac{u(x+h) - u(x-h)}{2h}.$$

Suppose we are interested in knowing the value of the derivative at the points $\{x_i\}$. Even if we don't have a formula for $u'(x)$, we can approximate it using finite differences. We first write the Taylor polynomial expansion of $u(x+h)$ and $u(x-h)$ centered at x. This gives

$$u(x+h) = u(x) + u'(x)h + \frac{1}{2}u''(x)h^2 + \frac{1}{6}u'''(x)h^3 + \mathcal{O}(h^4) \tag{5.1}$$

$$u(x-h) = u(x) - u'(x)h + \frac{1}{2}u''(x)h^2 - \frac{1}{6}u'''(x)h^3 + \mathcal{O}(h^4) \tag{5.2}$$

Subtracting (5.2) from (5.1) and rearranging gives

$$u'(x) = \frac{u(x+h) - u(x-h)}{2h} + \mathcal{O}(h^2).$$

In terms of our grid points $\{x_i\}$, we have:

$$u'(x_i) \approx \frac{u(x_i+h) - u(x_i-h)}{2h} = \frac{u(x_{i+1}) - u(x_{i-1})}{2h}.$$

We won't worry about the derivative at the endpoints, $u'(x_0)$ and $u'(x_n)$. This allows us to approximate the values $\{u'(x_i)\}$ as the solution to a system of equations:

$$\frac{1}{2h}\underbrace{\begin{bmatrix} -1 & 0 & 1 & & & \\ & -1 & 0 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 0 & 1 \\ & & & & -1 & 0 & 1 \end{bmatrix}}_{(n-1)\times(n+1)} \cdot \underbrace{\begin{bmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_{n-1}) \\ u(x_n) \end{bmatrix}}_{(n+1)\times 1} \approx \underbrace{\begin{bmatrix} u'(x_1) \\ u'(x_2) \\ \vdots \\ u'(x_{n-2}) \\ u'(x_{n-1}) \end{bmatrix}}_{(n-1)\times 1}. \tag{5.3}$$

This can be rewritten with a $(n-1)\times(n-1)$ tridiagonal matrix instead:

$$\frac{1}{2h}\underbrace{\begin{bmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -1 & 0 \end{bmatrix}}_{(n-1)\times(n-1)} \cdot \underbrace{\begin{bmatrix} u(x_1) \\ u(x_2) \\ \vdots \\ u(x_{n-2}) \\ u(x_{n-1}) \end{bmatrix}}_{(n-1)\times 1} + \underbrace{\begin{bmatrix} -u(x_0)/(2h) \\ 0 \\ \vdots \\ 0 \\ u(x_n)/(2h) \end{bmatrix}}_{(n-1)\times 1} \approx \underbrace{\begin{bmatrix} u'(x_1) \\ u'(x_2) \\ \vdots \\ u'(x_{n-2}) \\ u'(x_{n-1}) \end{bmatrix}}_{(n-1)\times 1}. \tag{5.4}$$

Next, we will consider the approximation for $u''(x)$. If we let

$$u'(x) \approx \frac{u(x+\frac{h}{2}) - u(x-\frac{h}{2})}{h}$$

then

$$u''(x) \approx \frac{u'(x+\frac{h}{2}) - u'(x-\frac{h}{2})}{h} \approx \frac{\frac{u((x+\frac{h}{2})+\frac{h}{2}) - u((x+\frac{h}{2})-\frac{h}{2})}{h} - \frac{u((x-\frac{h}{2})+\frac{h}{2}) - u((x-\frac{h}{2})-\frac{h}{2})}{h}}{h}$$

$$= \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}$$

You can achieve the same result by again consider the Taylor polynomial expansion and adding (5.1) and (5.2) and rearranging. Thus

$$u''(x_i) \approx \frac{u(x_i+h) - 2u(x_i) + u(x_i-h)}{h^2} = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2}$$

Again ignoring the second derivative at the endpoints, this can be written in matrix form as

$$\frac{1}{h^2}\underbrace{\begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 & 1 \end{bmatrix}}_{(n-1)\times(n+1)} \cdot \underbrace{\begin{bmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_{n-1}) \\ u(x_n) \end{bmatrix}}_{(n+1)\times 1} \approx \underbrace{\begin{bmatrix} u''(x_1) \\ u''(x_2) \\ \vdots \\ u''(x_{n-2}) \\ u''(x_{n-1}) \end{bmatrix}}_{(n-1)\times 1}. \tag{5.5}$$

This can also be written with a $(n-1)\times(n-1)$ tridiagonal matrix:

$$\frac{1}{h^2}\underbrace{\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}}_{(n-1)\times(n-1)} \cdot \underbrace{\begin{bmatrix} u(x_1) \\ u(x_2) \\ \vdots \\ u(x_{n-2}) \\ u(x_{n-1}) \end{bmatrix}}_{(n-1)\times 1} + \underbrace{\begin{bmatrix} u(x_0)/h^2 \\ 0 \\ \vdots \\ 0 \\ u(x_n)/h^2 \end{bmatrix}}_{(n-1)\times 1} = \underbrace{\begin{bmatrix} u''(x_1) \\ u''(x_2) \\ \vdots \\ u''(x_{n-2}) \\ u''(x_{n-1}) \end{bmatrix}}_{(n-1)\times 1} \tag{5.6}$$

Problem 1. Let $u(x) = \sin((x+\pi)^2 - 1)$. Use (5.3) - (5.6) to approximate $\frac{1}{2}u'' - u'$ at the grid points where $a = 0$, $b = 1$, and $n = 10$. Graph the result.

The previous equations are not only useful for approximating derivatives, but they can be also used to solve differential equations. Suppose that instead of knowing the function $u(x)$, we know that $\frac{1}{2}u'' - u' = f$, where the function $f(x)$ is given. How do we solve for $u(x)$?

Finite Difference Methods

Numerical methods for differential equations seek to approximate the exact solution $u(x)$ at some finite collection of points in the domain of the problem. Instead of analytically solving the original differential equation, defined over an infinite-dimensional function space, they use a well-chosen finite system of algebraic equations to approximate the original problem.

Consider the following differential equation:

$$\begin{aligned} &\varepsilon u''(x) - u(x)' = f(x), \quad x \in (0,1), \\ &u(0) = \alpha, \quad u(1) = \beta. \end{aligned} \tag{5.7}$$

Equation (5.7) can be written $Du = f$, where $D = \varepsilon \frac{d^2}{dx^2} - \frac{d}{dx}$ is a differential operator defined on the infinite-dimensional space of functions that are twice continuously differentiable on $[0,1]$ and satisfy $u(0) = \alpha$, $u(1) = \beta$.

We look for an approximate solution $\{U_i\}$, where

$$U_i \approx u(x_i)$$

on an evenly spaced grid of points, $a = x_0, x_1, \ldots, x_n = b,$. Our finite difference method will replace the differential operator $D = \varepsilon \frac{d^2}{dx^2} - \frac{d}{dx}$, (which is defined on an infinite-dimensional space), with finite difference operators (defined on a finite dimensional space). To do this, we replace derivative terms in the differential equation with appropriate difference expressions.

Recalling that

$$\begin{aligned} \frac{d^2}{dx^2}u(x_i) &= \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + \mathcal{O}(h^2), \\ \frac{d}{dx}u(x_i) &= \frac{u(x_{i+1}) - u(x_{i-1})}{2h} + \mathcal{O}(h^2). \end{aligned}$$

we define the finite difference operator D_h by

$$D_h U_i = \varepsilon \frac{1}{h^2}\left(U_{i+1} - 2U_i + U_{i-1}\right) - \frac{1}{2h}\left(U_{i+1} - U_{i-1}\right). \tag{5.8}$$

Thus we discretize equation (5.7) using the equations

$$\frac{\varepsilon}{h^2}(U_{i+1} - 2U_i + U_{i-1}) - \frac{1}{2h}\left(U_{i+1} - U_{i-1}\right) = f(x_i), \quad i = 1, \ldots, n-1,$$

along with boundary conditions $U_0 = \alpha$, $U_n = \beta$.

This gives $n+1$ equations and $n+1$ unknowns, and can be written in matrix form as

$$\frac{1}{h^2}\underbrace{\begin{bmatrix} h^2 & 0 & 0 & \ldots & 0 \\ (\varepsilon + h/2) & -2\varepsilon & (\varepsilon - h/2) & \ldots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \ldots & (\varepsilon + h/2) & -2\varepsilon & (\varepsilon - h/2) \\ 0 & \ldots & & 0 & h^2 \end{bmatrix}}_{(n+1)\times(n+1)} \cdot \underbrace{\begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_{n-1} \\ U_n \end{bmatrix}}_{(n+1)\times 1} = \underbrace{\begin{bmatrix} \alpha \\ f(x_1) \\ \vdots \\ f(x_{n-1}) \\ \beta \end{bmatrix}}_{(n+1)\times 1}.$$

As before, we can remove two equations to modify the system to obtain an $(n-1)\times(n-1)$ tridiagonal

system:

$$\frac{1}{h^2}\underbrace{\begin{bmatrix} -2\varepsilon & (\varepsilon - h/2) & 0 & \dots & 0 \\ (\varepsilon + h/2) & -2\varepsilon & (\varepsilon - h/2) & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & (\varepsilon + h/2) & -2\varepsilon & (\varepsilon - h/2) \\ 0 & \dots & & (\varepsilon + h/2) & -2\varepsilon \end{bmatrix}}_{(n-1)\times(n-1)} \cdot \underbrace{\begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_{n-2} \\ U_{n-1} \end{bmatrix}}_{(n-1)\times 1} = \underbrace{\begin{bmatrix} f(x_1) - \alpha(\varepsilon + h/2)/h^2 \\ f(x_2) \\ \vdots \\ f(x_{n-2}) \\ f(x_{n-1}) - \beta(\varepsilon - h/2)/h^2 \end{bmatrix}}_{(n-1)\times 1}. \tag{5.9}$$

Problem 2. Use equation (5.9) to solve the singularly perturbed BVP (5.7) with $\varepsilon = 1/10$, $f(x) = -1$, $\alpha = 1$, and $\beta = 3$ on a grid with $n = 30$ subintervals. Graph the solution. This BVP is called singularly perturbed because of the location of the parameter ε. For $\varepsilon = 0$ the ODE has a drastically different character - it then becomes first order, and can no longer support two boundary conditions.

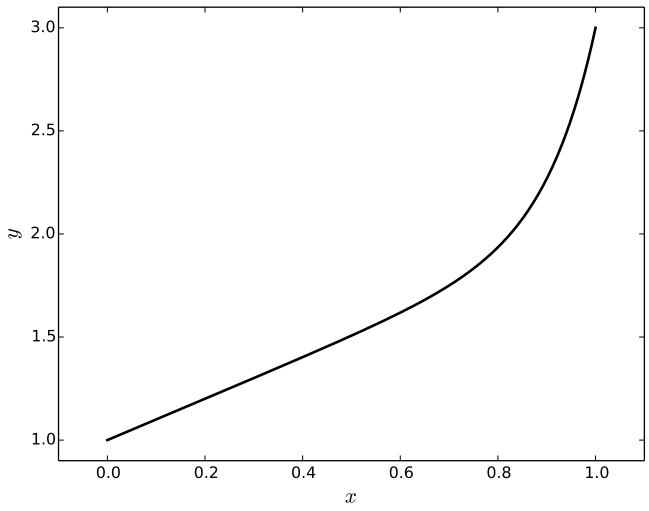

Figure 5.1: The solution to Problem 2. The solution gets steeper near $x = 1$ as ε gets small.

A heuristic test for convergence

The finite differences used above are second order approximations of the first and second derivatives of a function. It seems reasonable to expect that the numerical solution would converge at a rate of

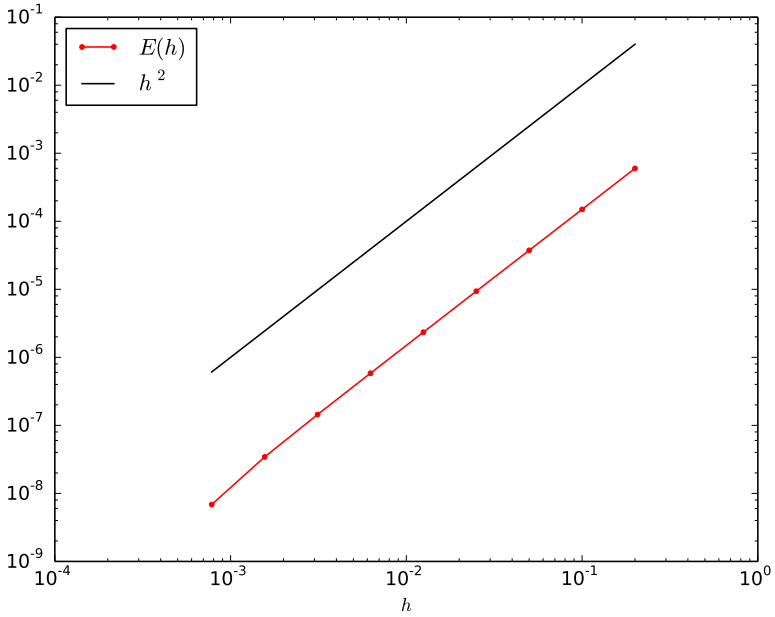

Figure 5.2: Demonstration of second order convergence for the finite difference approximation (5.8) of the BVP given in (5.7) with $\varepsilon = .5$.

about $\mathcal{O}(h^2)$. How can we check that a numerical approximation is reasonable?

Suppose a finite difference method is $\mathcal{O}(h^p)$ accurate. This means that the error $E(h) \approx Ch^p$ for some constant C as $h \to 0$ (in other words, for $h > 0$ small enough).

So compute the approximation y_k for each stepsize h_k, $h_1 > h_2 > \ldots > h_m$. y_m should be the most accurate approximation, and will be thought of as the true solution. Then the error of the approximation for stepsize $h_k, k < m$, is

$$E(h_k) = \max(|y_k - y_m|) \approx Ch_k^p,$$
$$\log(E(h_k)) = \log(C) + p\log(h_k).$$

Thus on a log-log plot of $E(h)$ vs. h, these values should be on a straight line with slope p when h is small enough to start getting convergence. We should note that demonstrating second-order convergence does NOT imply that the numerical approximation is converging to the correct solution.

Problem 3. Visualize the $\mathcal{O}(h^2)$ convergence of this finite difference method by producing a loglog plot similar to Figure 5.2, except in the case $\varepsilon = .1$. Implement a function `singular_bvp` to compute the finite difference solution to 5.7. Using $n = 5 \times 2^0, 5 \times 2^1, \ldots, 5 \times 2^9$ subintervals, compute 10 approximate solutions.

To produce the plot, treat the approximation with $n = 5 \times 2^9$ subintervals as the "true solution", and measure the error for the other approximations against it. Note that, since the number of subintervals for each approximation is a multiple of 2, we can compute the L_∞ error for the $n = 5 \times 2^j$ approximation by using the `step` argument in the array slicing syntax:

```
# best approximation
sol_best = singular_bvp(eps,alpha,beta,f,5*(2**9))

# approximation with 5*(2^j) intervals
sol_approx = singular_bvp(eps,alpha,beta,f,5*(2**j))

# approximation error
error = np.max(np.abs(sol_approx - sol_best[::2**(9-j)]))
```

Problem 4. Extend your finite difference code to the case of a general second order linear BVP with boundary conditions:

$$a_1(x)y'' + a_2(x)y' + a_3(x)y = f(x), \quad x \in (a,b),$$
$$y(a) = \alpha, \quad y(b) = \beta.$$

Use your code to solve the boundary value problem

$$\varepsilon y'' - 4(\pi - x^2)y = \cos x,$$
$$y(0) = 0, \quad y(\pi/2) = 1,$$

for $\varepsilon = 0.1$ on a grid with $n = 30$ subintervals. Be sure to modify the finite difference operator D_h in (5.8) correctly.

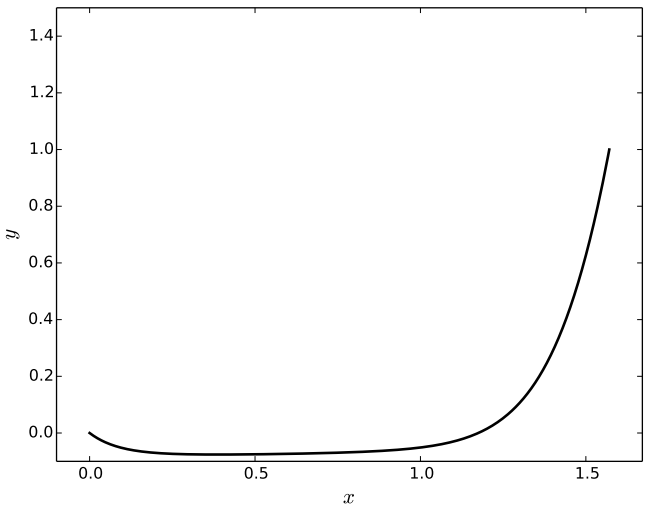

Figure 5.3: The solution to Problem 4.

The next few problems will help you test your finite difference code.

Problem 5. Numerically solve the boundary value problem

$$\varepsilon y'' + xy' = -\varepsilon\pi^2\cos(\pi x) - \pi x\sin(\pi x),$$
$$y(-1) = -2, \quad y(1) = 0,$$

for $\varepsilon = 0.1, 0.01$, and 0.001. Use a grid with $n = 150$ subintervals.

Problem 6. Numerically solve the boundary value problem

$$(\varepsilon + x^2)y'' + 4xy' + 2y = 0,$$
$$y(-1) = 1/(1+\varepsilon), \quad y(1) = 1/(1+\varepsilon),$$

for $\varepsilon = 0.05, 0.02$. Use a grid with $n = 150$ subintervals.

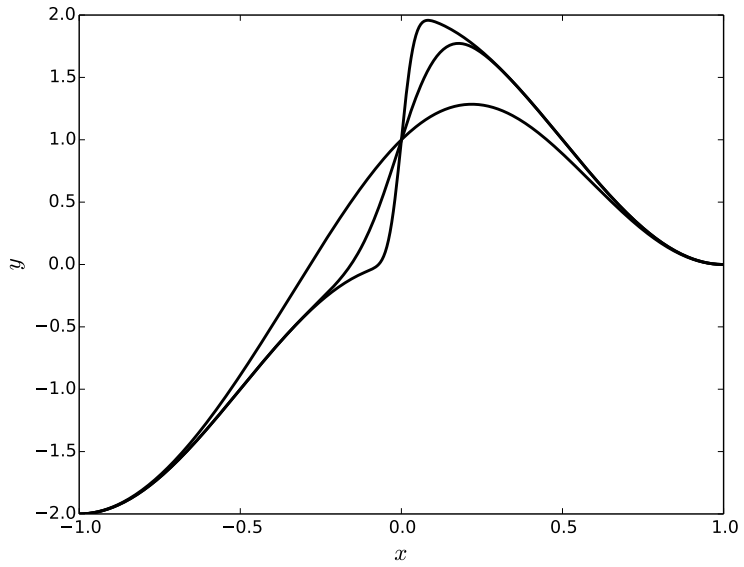

Figure 5.4: The solution to Problem 5.

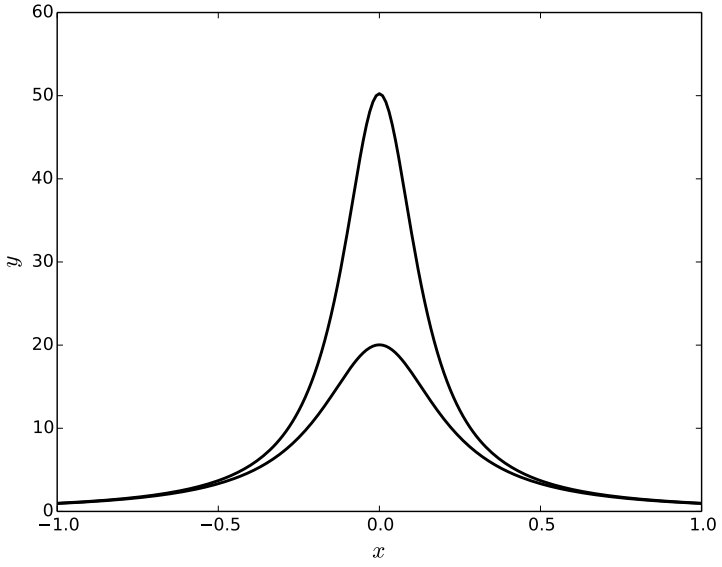

Figure 5.5: The solution to Problem 6.