7 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 $a = x_{-1}, x_0, x_1, \ldots x_{n-1} = b$ be a grid of $n+1$ evenly spaced points, with $x_i = a + (i+1)h$, $h = (b-a)/n$.

You are used to seeing the derivative $u'(x)$ which can be written as

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

Since we are interested in the derivative at certain fixed points x_i, we can consider the approximation of $u'(x)$ using finite differences. We first write the Taylor polynomial expansion of $u(x+h)$ and $u(x-h)$ centered at x. This gives

$$u(x+h) = u(x) + u'(x)h + \frac{1}{2}u''(x)h^2 + \frac{1}{6}u'''(x)h^3 + \mathcal{O}(h^4) \tag{7.1}$$

$$u(x-h) = u(x) - u'(x)h + \frac{1}{2}u''(x)h^2 - \frac{1}{6}u'''(x)h^3 + \mathcal{O}(h^4) \tag{7.2}$$

Subtracting (7.2) from (7.1) and rearranging gives

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

From the Taylor expansion, this term has error $E(h) = \mathcal{O}(h^2)$. In terms of our grid points $\{x_i\}$, we can rewrite $u'(x)$ as $u'(x_i)$ and

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

We won't worry about the derivative at the endpoints, $u'(x_{-1})$ and $u'(x_{n-1})$. This allows us to write the set of points $\{u'(x_i)\}$ as the solution to a system of equations

$$\frac{1}{2h} \underbrace{\begin{bmatrix} -1 & 0 & 1 & & & & \\ & -1 & 0 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 0 & 1 & \\ & & & & -1 & 0 & 1 \end{bmatrix}}_{(n-1)\times(n+1)} \cdot \underbrace{\begin{bmatrix} u(x_{-1}) \\ u(x_0) \\ \vdots \\ u(x_{n-2}) \\ u(x_{n-1}) \end{bmatrix}}_{(n+1)\times 1} = \underbrace{\begin{bmatrix} u'(x_0) \\ u'(x_1) \\ \vdots \\ u'(x_{n-3}) \\ u'(x_{n-2}) \end{bmatrix}}_{(n-1)\times 1}. \tag{7.3}$$

This can be rewritten with an $(N-1)\times(N-1)$ tridiagonal matrix on the left.

$$\frac{1}{2h}\underbrace{\begin{bmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -1 & 0 \end{bmatrix}}_{(n-1)\times(n-1)} \cdot \underbrace{\begin{bmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_{n-3}) \\ u(x_{n-2}) \end{bmatrix}}_{(n-1)\times 1} + \underbrace{\begin{bmatrix} -u(x_{-1})/(2h) \\ 0 \\ \vdots \\ 0 \\ u(x_{n-1})/(2h) \end{bmatrix}}_{(n-1)\times 1} = \underbrace{\begin{bmatrix} u'(x_0) \\ u'(x_1) \\ \vdots \\ u'(x_{n-3}) \\ u'(x_{n-2}) \end{bmatrix}}_{(n-1)\times 1}. \tag{7.4}$$

Next we will consider the matrix representation for $u''(x)$. If we let

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

then

$$\begin{aligned} u''(x) = \frac{u'(x+\frac{h}{2}) - u'(x-\frac{h}{2})}{h} &= \frac{\frac{u((x+\frac{h}{2})+\frac{h}{2})-u((x+\frac{h}{2})-\frac{h}{2})}{h} - \frac{u((x-\frac{h}{2})+\frac{h}{2})-u((x-\frac{h}{2})-\frac{h}{2})}{h}}{h} \\ &= \frac{u(x+h)-2u(x)+u(x-h)}{h^2}, \end{aligned}$$

with error $E(h) = \mathcal{O}(h^3)$. You can achieve the same result by again consider the Taylor polynomial expansion and adding (7.1) and (7.2) and rearranging. Thus

$$u''(x_i) = \frac{u(x_i+h) - 2u(x_i) + u(x_i-h)}{h^2} = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2}, \quad i = 0, \dots, n-2.$$

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

$$\frac{1}{h^2}\underbrace{\begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 & 1 \end{bmatrix}}_{(n-1)\times(n+1)} \cdot \underbrace{\begin{bmatrix} u(x_{-1}) \\ u(x_0) \\ \vdots \\ u(x_{n-2}) \\ u(x_{n-1}) \end{bmatrix}}_{(n+1)\times 1} = \underbrace{\begin{bmatrix} u''(x_0) \\ u''(x_1) \\ \vdots \\ u''(x_{n-3}) \\ u''(x_{n-2}) \end{bmatrix}}_{(n-1)\times 1}. \tag{7.5}$$

This can also be written as an $(N-1)\times(N-1)$ tridiagonal matrix on the left.

$$\frac{1}{h^2}\underbrace{\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}}_{(n-1)\times(n-1)} \cdot \underbrace{\begin{bmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_{n-3}) \\ u(x_{n-2}) \end{bmatrix}}_{(n-1)\times 1} + \underbrace{\begin{bmatrix} u(x_{-1})/h^2 \\ 0 \\ \vdots \\ 0 \\ u(x_{n-1})/h^2 \end{bmatrix}}_{(n-1)\times 1} = \underbrace{\begin{bmatrix} u''(x_0) \\ u''(x_1) \\ \vdots \\ u''(x_{n-3}) \\ u''(x_{n-2}) \end{bmatrix}}_{(n-1)\times 1} \tag{7.6}$$

Problem 1. Let $u(x) = \sin((x+\pi)^2 - 1)$. Use (7.3) - (7.6) to approximate $\frac{1}{2}u'' - u'$ at the grid points where $a = 0$, $b = 1$, and $n = 10$. Graph the result.

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 at the grid points?

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 simpler finite system of algebraic equations to approximate the original problem.

Consider the following differential equation:

$$\begin{aligned} &\epsilon u''(x) - u(x)' = f(x), \quad x \in (0,1), \\ &u(0) = \alpha, \quad u(1) = \beta. \end{aligned} \tag{7.7}$$

Equation (7.7) can be written $Du = f$, where $D = \epsilon \frac{d^2}{dx^2} - \frac{d}{dx}$ 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 $\{U_i\}_{i=-1}^{N-1}$, where

$$U_i = u(x_i)$$

on an evenly spaced grid of N subintervals, $a = x_{-1}, x_0, \ldots, x_{N-1} = b$ with $h = x_{i+1} - x_i$ for each i. Our finite difference method will replace the differential operator $D = \epsilon \frac{d^2}{dx^2} - \frac{d}{dx}$, defined on an infinite-dimensional space of functions, with difference operators defined on a finite vector space (the space of grid functions $\{U_i\}_{i=-1}^{N-1}$). 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

$$D_h U_i = \epsilon \frac{1}{h^2} \left(U_{i+1} - 2U_i + U_{i-1} \right) - \frac{1}{2h} \left(U_{i+1} - U_{i-1} \right). \tag{7.8}$$

Thus we discretize equation (7.7) using the equations

$$\frac{\epsilon}{h^2}(U_{i+1} - 2U_i + U_{i-1}) - \frac{1}{2h} \left(U_{i+1} - U_{i-1} \right) = f(x_i), \quad i = 0, \ldots, N-2,$$

along with boundary conditions $U_{-1} = \alpha$, $U_{N-1} = \beta$.

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

$$\frac{1}{h^2} \underbrace{\begin{bmatrix} h^2 & 0 & 0 & \ldots & 0 \\ (\epsilon + h/2) & -2\epsilon & (\epsilon - h/2) & \ldots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \ldots & (\epsilon + h/2) & -2\epsilon & (\epsilon - h/2) \\ 0 & \ldots & & 0 & h^2 \end{bmatrix}}_{(N+1)\times(N+1)} \cdot \underbrace{\begin{bmatrix} U_{-1} \\ U_0 \\ \vdots \\ U_{N-2} \\ U_{N-1} \end{bmatrix}}_{(N+1)\times 1} = \underbrace{\begin{bmatrix} U_{-1} \\ f(x_0) \\ \vdots \\ f(x_{N-2}) \\ U_{N-1} \end{bmatrix}}_{(N+1)\times 1}.$$

We can further modify the system to obtain an $(N-1)\times(N-1)$ tridiagonal matrix on the left:

$$
\frac{1}{h^2}\underbrace{\begin{bmatrix} -2\epsilon & (\epsilon-h/2) & 0 & \dots & 0 \\ (\epsilon+h/2) & -2\epsilon & (\epsilon-h/2) & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & (\epsilon+h/2) & -2\epsilon & (\epsilon-h/2) \\ 0 & \dots & & (\epsilon+h/2) & -2\epsilon \end{bmatrix}}_{(N-1)\times(N-1)} \cdot \underbrace{\begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_{N-3} \\ U_{N-2} \end{bmatrix}}_{(N-1)\times 1}
= \underbrace{\begin{bmatrix} f(x_0) - U_{-1}(\epsilon+h/2)/h^2 \\ f(x_1) \\ \vdots \\ f(x_{N-3}) \\ f(x_{N-2}) - U_{n-1}(\epsilon-h/2)/h^2 \end{bmatrix}}_{(N-1)\times 1}. \tag{7.9}
$$

Problem 2. Use equation (7.9) to solve the singularly perturbed BVP (7.7) with $\epsilon = 1/10$, $f(x) = -1$, $\alpha = 1$, and $\beta = 3$. Graph the solution. This BVP is called singularly perturbed because of the location of the parameter ϵ. For $\epsilon = 0$ the ODE has a drastically different character - it then becomes first order, and can no longer support two boundary conditions.

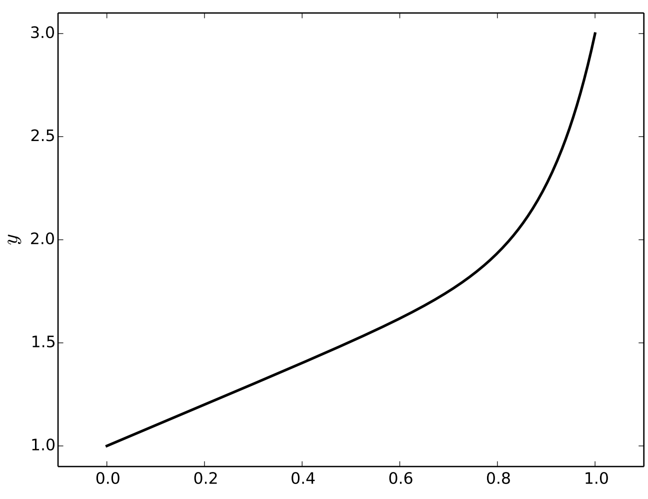

Figure 7.1: The solution to Problem 2. The solution gets steeper near $x = 1$ as ϵ gets small.

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

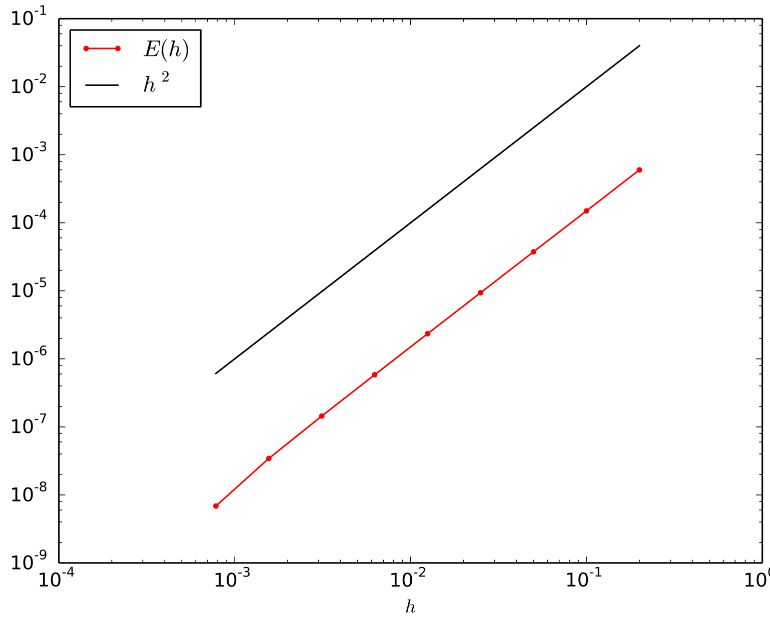

Figure 7.2: Demonstration of second order convergence for the finite difference approximation (7.8) of the BVP given in (7.7) with $\epsilon = .5$.

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

$$
\begin{aligned}
E(h_k) &= \max(|y_k - y_m|) \approx Ch_k^p, \\
\log(E(h_k)) &= \log(C) + p\log(h_k).
\end{aligned}
$$

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. Return to problem 2. How many subintervals are needed to obtain 4 digits of accuracy?

This is a question about the convergence of your solution. The following code generates the log-log plot in Figure 7.2, and demonstrates second-order convergence for our finite difference approximation of (7.7). Use this code to determine what h (and hence what N) is needed for the error to be less than 10^{-4}. You don't need to return the value of h, but make sure you understand by looking at the plot.

NOTE: The function `bvp` is not provided; you need to use your code from problem 2 to define it. Make sure your function is compatible with the code below. It must take 5 parameters as input and return the solution.

```
num_approx = 10 # Number of Approximations
N = 5*np.array([2**j for j in range(num_approx)])
h, max_error = (1.-0)/N[:-1], np.ones(num_approx-1)

# Best numerical solution, used to approximate the true solution.
# bvp returns the grid, and the grid function, approximating the solution
# with N subintervals of equal length.
num_sol_best = bvp(lambda x:-1, epsilon=.1, alpha=1, beta=3, N=N[-1])
for j in range(len(N)-1):
    num_sol = bvp(lambda x:-1, epsilon=.1, alpha=1, beta=3, N=N[j])
    max_error[j] = np.max(np.abs(num_sol-num_sol_best[::2**(num_approx-j-1)]))
plt.loglog(h,max_error,'.-r',label="$E(h)$")
plt.loglog(h,h**(2.),'-k',label="$h^{\, 2}$")
plt.xlabel("$h$")
plt.legend(loc='best')
plt.show()
print("The order of the finite difference approximation is about ",
    (np.log(max_error[0])-np.log(max_error[-1]))/(np.log(h[0])-np.log(h[-1])),
    ".")
```

Problem 4. Extend your finite difference code to the case of a general second order linear BVP with boundary conditions (These boundary conditions are sometimes called Dirichlet conditions):

$$\begin{aligned} &a_1(x)y'' + a_2(x)y' + a_3(x)y = f(x), \quad x \in (a,b),\\ &y(a) = \alpha, \quad y(b) = \beta. \end{aligned}$$

Use your code to solve the boundary value problem

$$\begin{aligned} \epsilon y'' - 4(\pi - x^2)y &= \cos x,\\ y(0) = 0, \quad y(\pi/2) &= 1, \end{aligned}$$

for $\epsilon = 0.1$. Be sure to modify the finite difference operator D_h in (7.8) correctly.

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

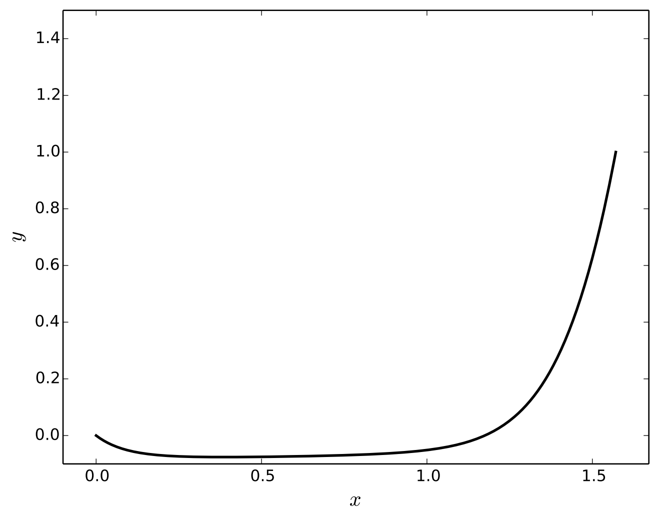

Figure 7.3: The solution to Problem 4.

Problem 5. Numerically solve the boundary value problem

$$\begin{aligned} \epsilon y'' + xy' &= -\epsilon\pi^2\cos(\pi x) - \pi x\sin(\pi x), \\ y(-1) &= -2, \quad y(1) = 0, \end{aligned}$$

for $\epsilon = 0.1, 0.01$, and 0.001.

Problem 6. Numerically solve the boundary value problem

$$\begin{aligned} (\epsilon + x^2)y'' + 4xy' + 2y &= 0, \\ y(-1) = 1/(1+\epsilon), \quad y(1) &= 1/(1+\epsilon), \end{aligned}$$

for $\epsilon = 0.05, 0.02$.

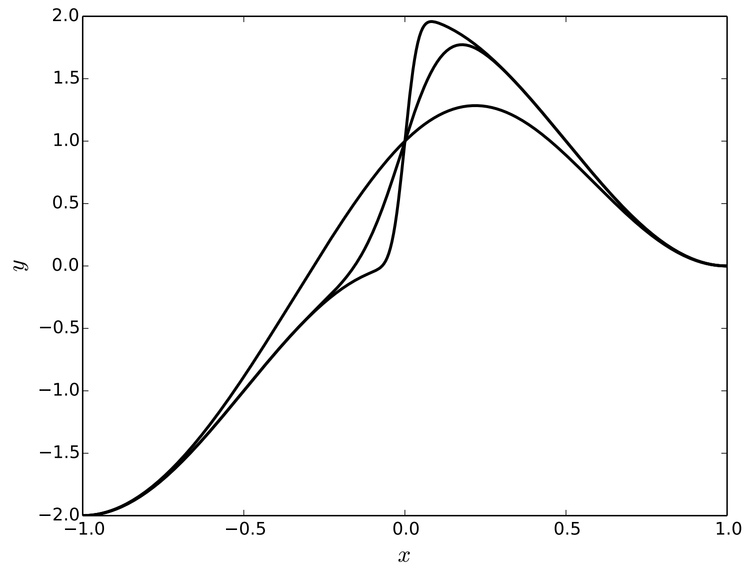

Figure 7.4: The solution to Problem 5.

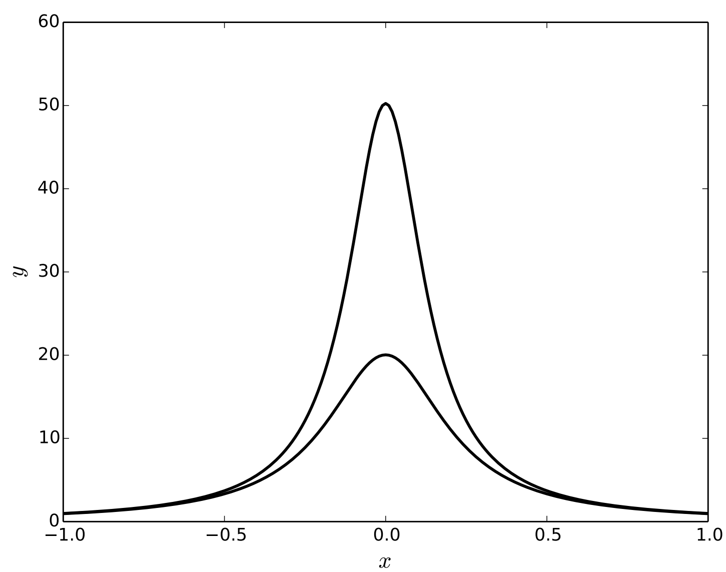

Figure 7.5: The solution to Problem 6.