1 Chapter 1: Solutions to Exercises

1. **(Second-order central difference approximation to the second derivative)** Show that if $f \in C^4[x_{j-1}, x_{j+1}]$, then

$$f''(x_j) = \frac{f(x_{j+1}) - 2f(x_j) + f(x_{j-1})}{h^2} + \mathcal{O}(h^2), \qquad h \to 0.$$

It follows from Taylor's theorem that there exists an $\xi_1 \in [x_j, x_{j+1}]$ such that

$$f(x_{j+1}) = f(x_j) + hf'(x_j) + \frac{h^2}{2}f''(x_j) + \frac{h^3}{6}f'''(x_j) + \frac{h^4}{24}f^{(4)}(\xi_1)$$

where we used the fact that $x_{j+1} - x_j = h$. Similarly, there exists an $\xi_2 \in [x_{j-1}, x_j]$ such that

$$f(x_{j-1}) = f(x_j) - hf'(x_j) + \frac{h^2}{2}f''(x_j) - \frac{h^3}{6}f'''(x_j) + \frac{h^4}{24}f^{(4)}(\xi_2)$$

Adding these two equations and solving for $f''(x_j)$, we have that

$$f''(x_j) = \frac{f(x_{j+1}) - 2f(x_j) + f(x_{j-1})}{h^2} + \frac{h^2}{24}\left(f^{(4)}(\xi_1) + f^{(4)}(\xi_2)\right)$$

Since $f \in C^4[x_{j-1}, x_{j+1}]$, there exists an $M < \infty$ such that

$$M = \sup_{x \in [x_{j-1}, x_{j+1}]} |f^{(4)}(x)|$$

and therefore

$$\left| f''(x_j) - \frac{f(x_{j+1}) - 2f(x_j) + f(x_{j-1})}{h^2} \right| \leq \frac{M}{12}h^2$$

and hence the result follows.

2. Run the Julia code and / or the Matlab code for this chapter (see the Blackboard page of this module) on your own machine. To run the Julia code, see the document titled Julia on Blackboard. To run the Matlab code, you'll need to use Chebfun (see the instructions for how to do this here).

3. Construct, by hand, a polynomial that interpolates $f(x) = \mathrm{e}^x$ at $x = 0, 1, 2$.

Using the formula for the Lagrange interpolating polynomial, we have that

$$p_3(x) = \frac{(x-1)(x-2)}{2}\mathrm{e}^0 - (x-0)(x-2)\mathrm{e}^1 + \frac{(x-0)(x-1)}{2}\mathrm{e}^2$$

4. **(Second-order backward difference approximation)** Construct a Lagrange interpolating polynomial $p_3(x)$ that interpolates $f(x)$ at x_{j-2}, x_{j-1} and x_j, where $x_j - x_{j-1} = x_{j-1} - x_{j-2} = h$, and derive the following one-sided finite difference approximation:

$$f'(x_j) \approx p_3'(x_j) = \frac{3f(x_j) - 4f(x_{j-1}) + f(x_{j-2})}{2h}$$

Using the formula for the Lagrange interpolating polynomial, it follows that

$$p_3(x) = \frac{(x-x_{j-1})(x-x_j)}{2h^2} f(x_{j-2}) - \frac{(x-x_{j-2})(x-x_j)}{h^2} f(x_{j-1}) + \frac{(x-x_{j-2})(x-x_{j-1})}{2h^2} f(x_j).$$

Since

$$p_3'(x) = \frac{(x-x_{j-1})+(x-x_j)}{2h^2} f(x_{j-2}) - \frac{(x-x_{j-2})+(x-x_j)}{h^2} f(x_{j-1}) + \frac{(x-x_{j-2})+(x-x_{j-1})}{2h^2} f(x_j)$$

we have that

$$p_3'(x_j) = \frac{h}{2h^2} f(x_{j-2}) - \frac{2h}{h^2} f(x_{j-1}) + \frac{2h+h}{2h^2} f(x_j)$$

and the result follows.

5. In Julia or Matlab, compute the maximum error of the finite difference approximation

$$f'(x_j) \approx \frac{f(x_{j-2}) - 8f(x_{j-1}) + 8f(x_{j+1}) - f(x_{j+2})}{12h},$$

at the points $x_j = jh$ with $j = 0, \ldots, n-1$ and $h = 2\pi/n$ for the function $f(x) = 1/(2 + \cos(x))$ for various values of n. Then plot the errors on a log-log plot. Comment on your results.

Since $f(x)$ is periodic, the differentiation matrix is as follows:

$$\begin{pmatrix} f'(x_0) \\ f'(x_1) \\ \\ \vdots \\ \\ f'(x_{n-2}) \\ f'(x_{n-1}) \end{pmatrix} \approx \begin{pmatrix} p'(x_0) \\ p'(x_1) \\ \\ \vdots \\ \\ p'(x_{n-2}) \\ p'(x_{n-1}) \end{pmatrix} = \frac{1}{h} \begin{bmatrix} 0 & \frac{2}{3} & -\frac{1}{12} & & & \frac{1}{12} & -\frac{2}{3} \\ -\frac{2}{3} & 0 & \frac{2}{3} & \ddots & & & \frac{1}{12} \\ \frac{1}{12} & -\frac{2}{3} & 0 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} \\ -\frac{1}{12} & & & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{12} & & & \frac{1}{12} & -\frac{2}{3} & 0 \end{bmatrix} \begin{pmatrix} f(x_0) \\ f(x_1) \\ \\ \vdots \\ \\ f(x_{n-2}) \\ f(x_{n-1}) \end{pmatrix}.$$

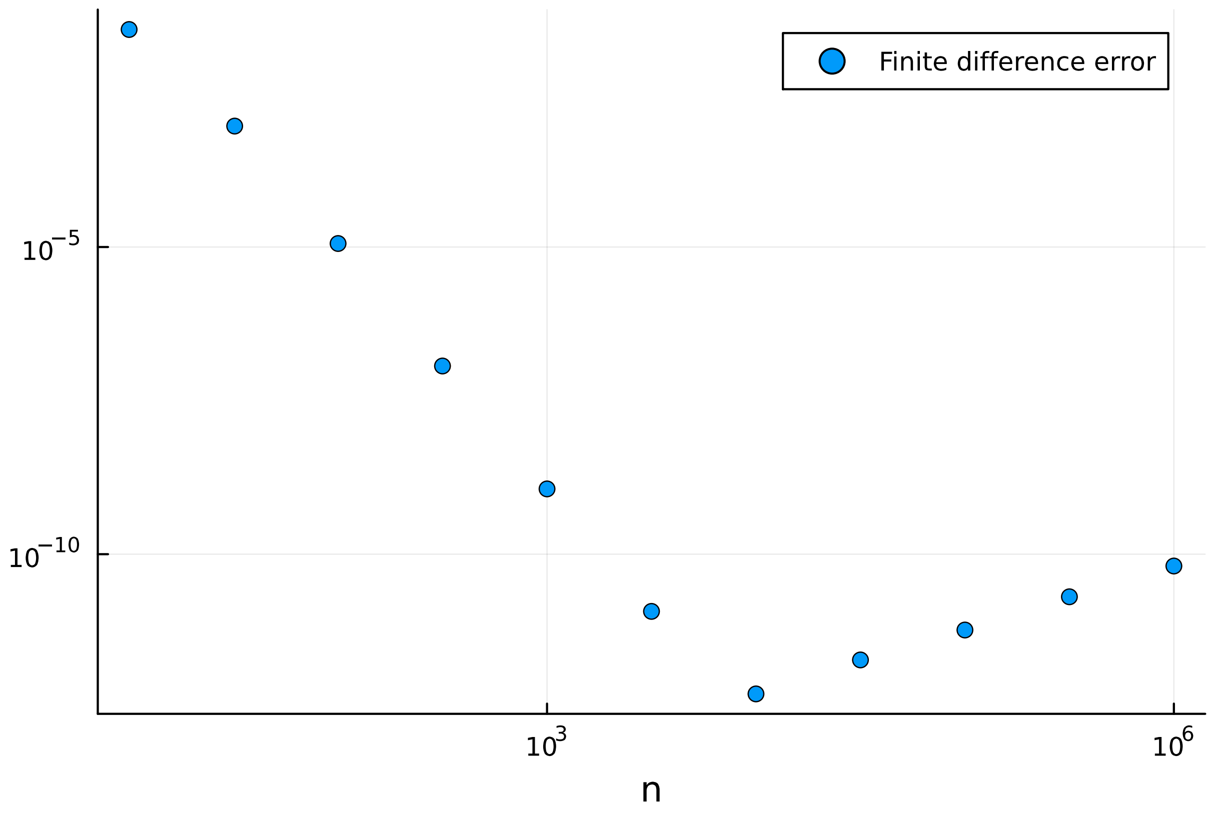
Here is how we can build the differentiation matrix:

```
using LinearAlgebra, Plots, SparseArrays
n = 10
Dn = spdiagm(1=>fill(2/3,n-1),2=>fill(-1/12,n-2),n-2=>fill(1/12,2),n-1=>fill(-2/3,1))
Dn = Dn - Dn'

10×10 SparseArrays.SparseMatrixCSC{Float64, Int64} with 40 stored entries:
   ⋅           0.666667   -0.0833333  …    ⋅           0.0833333  -0.666667
 -0.666667      ⋅          0.666667        ⋅           ⋅           0.0833333
  0.0833333  -0.666667      ⋅              ⋅           ⋅            ⋅
   ⋅           0.0833333  -0.666667        ⋅           ⋅            ⋅
   ⋅            ⋅          0.0833333      ⋅           ⋅            ⋅
   ⋅            ⋅            ⋅          …  -0.0833333    ⋅            ⋅
   ⋅            ⋅            ⋅              0.666667   -0.0833333    ⋅
   ⋅            ⋅            ⋅              ⋅           0.666667   -0.0833333
 -0.0833333    ⋅            ⋅             -0.666667    ⋅            0.666667
  0.666667   -0.0833333    ⋅               0.0833333  -0.666667    ⋅
```

```
f = x -> 1/(2 + cos(x))
Df = x -> sin(x)/(2 + cos(x))^2  # derivative
# compute the errors
nv = Int64.(round.(10 .^(1:0.5:6)))
@time begin
errs =
[( h = 2π/n;
   # differentiation matrix:
   Dn =
1/h*spdiagm(1=>fill(2/3,n-1),2=>fill(-1/12,n-2),n-2=>fill(1/12,2),n-1=>fill(-2/3,1));
   Dn = Dn - Dn';
   x = range(0,2π;length=n+1)[1:end-1]; # the nodes x_j, j = 0, ..., n-1
   # compute the maximum error at the nodes
   norm(Dn*f.(x) - Df.(x),Inf) ) for n = nv]
end;
# plot the errors on a log-log scale
scatter(nv,errs;xscale=:log10,yscale=:log10,label="Finite difference error",xlabel="n")
```

```
1.643177 seconds (514.34 k allocations: 497.255 MiB, 52.42% gc time, 24.6
7% compilation time)
```



The errors decrease as n is increased but for large enough n, the errors increase due to cancellation error in double precision. To estimate the rate of convergence, we need to estimate the slope of the line along which the errors decrease. Here is an estimate of the slope:

```
(log(errs[6])-log(errs[4]))/(log(nv[6]) - log(nv[4]))
```

```
-3.9935549986049828
```

The slope is approximately -4, hence the error decreases as $\mathcal{O}(n^{-4})$, $n \to \infty$ or $\mathcal{O}(h^4)$, $h \to 0$.