4 Finite Difference Methods

Taylor series and Taylor's theorem play a fundamental role in finite difference method approximations of derivatives. We begin this chapter with a review of these important results.

Theorem 17 (Taylor series)**.** *If a function f is infinitely differentiable on $\mathbb{R}$, then for any chosen expansion point x_0, it holds that*

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f'''(x_0)}{3!}(x - x_0)^3 + ...$$

for every real number x.

Theorem 18 (Taylor's theorem)**.** *If a function f is n+1 times differentiable in an interval around a chosen expansion point x_0, then for every real number x in the interval, there is a number $c \in [x, x_0]$ such that*

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 \\ + ... + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + \frac{f^{(n+1)}(c)}{(n + 1)!}(x - x_0)^{n+1}.$$

The importance of Taylor's theorem is that we can pick a point x_0, and then represent the function $f(x)$, no matter how 'messy' it is, by the polynomial $f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + ... + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$. The error is given by the single term $\frac{f^{(n+1)}(c)}{(n+1)!}(x - x_0)^{n+1}$. Although we do not know c, we do know it lives in $[x, x_0]$. In many cases, this term can be estimated to give an upper bound on the error in approximating a function by a Taylor polynomial.

One way to think of these Taylor polynomials is as the tangent polynomials for a function f at a point x_0. If $n = 1$, we recover the tangent line to f at x_0, and if $n = 2$ we would obtain the tangent parabola. As we learned in Calculus I, the tangent line is quite accurate for x close to x_0 and can be quite inaccurate if x is far away from x_0. This is exactly what we see in the error term $\frac{f^{(2)}(c)}{(2)!}(x - x_0)^2$; the difference $(x - x_0)$ is squared, which directly shows large error for x far away from x_0. In general, Taylor polynomials are useful approximations when points are close together and bad approximations when points are far apart. In what we do in this chapter, which includes applications to differential equations, the points will be close together, and thus, Taylor polynomials will be a useful tool.

4.1 Convergence terminology

For a given mathematical problem, assume there is a solution u. If we use a numerical algorithm to approximate u, then we will get a numerical solution $\tilde{u}$ (it is extremely rare for $u = \tilde{u}$). The fundamental question is: How close is $\tilde{u}$ to u? It is our job to quantify this difference.

In many cases, the error in an approximation depends on a parameter. For example, in Newton's method, the error typically depends on how many iterations are performed. If one is approximating a derivative of $f'(x)$ by calculating $\frac{f(x+h)-f(x)}{h}$ for a user chosen h, the error will naturally depend on h. Hence in this case, we will want to quantify the error in terms of h. That is, we want to be able to write

$$\left| \frac{f(x+h)-f(x)}{h} - f'(x) \right| \leq Ch^k,$$

where C is a problem dependent constant (independent of h and k), and we want to determine k. If $k > 0$, then as h decreases, we are assured the error will go to 0. The larger k is, the faster it will go to zero. Often, the value of k will allow us to compare different algorithms in order to determine which is more accurate.

Definition 19 (Big O notation)**.** *Suppose u is the true solution to a mathematical problem, and $\tilde{u}(h)$ is an approximation to the solution that depends on a parameter h. If it holds that*

$$|u - \tilde{u}(h)| \leq Ch^k,$$

with C being a constant independent of h and k, then we write

$$|u - \tilde{u}(h)| = O(h^k).$$

This is interpreted as "The error is on the order of h^k."

For first order convergence ($k = 1$), the error is reduced proportional to the reduction of h. In other words, if h gets cut in half, you can expect the error to be approximately cut in half also. For second order convergence, i.e. $O(h^2)$, however, if h gets cut in half, the error gets cut by $\frac{1}{2}^2 = \frac{1}{4}$, which is obviously much better.

Example 20. *Suppose we have an algorithm where the error depends on h, and for a sequence of h's $\{1, 1/2, 1/4, 1/8, 1/16, 1/32, ...\}$, the sequence of errors is given by*

$$10,\ 5,\ 2.5,\ 1.25,\ 0.625,\ 0.3125$$

and converges with first order accuracy, i.e. $O(h)$*. This is because when* h *gets cut in half, so do the errors. For the same* h*'s, the sequence of errors*

$$100,\ 25,\ 6.25,\ 1.5625,\ 0.390625,\ 0.09765625$$

converges with second order accuracy, i.e. $O(h^2)$*, since the errors get cut by a factor of* $4 = 2^2$ *when* h *is cut in half.*

In the example above, it was clear that the exponents of h were 1 and 2, but in general, the k in $O(h^k)$ need not be an integer. To approximate k from two approximation errors e_1, e_2 corresponding to parameters h_1, h_2, we treat the error bound as a close approximation and start with

$$e \approx Ch^k$$

with C independent of h and k. Then we have that

$$e_1 \approx Ch_1^k, \quad e_2 \approx Ch_2^k.$$

Solving for C in both equations and setting them equal gives

$$\frac{e_1}{h_1^k} \approx \frac{e_2}{h_2^k} \implies \left(\frac{h_2}{h_1}\right)^k \approx \frac{e_2}{e_1} \implies k \log\left(\frac{h_2}{h_1}\right) \approx \log\left(\frac{e_2}{e_1}\right) \implies k \approx \frac{\log\left(\frac{e_2}{e_1}\right)}{\log\left(\frac{h_2}{h_1}\right)}.$$

Given a sequence of h's and corresponding errors, we calculate k's for each successive error, and typically, k will converge to a number.

4.2 Approximating the first derivative

4.2.1 Forward and backward differences

Consider a discretization of an interval [a,b] with N+1 equally spaced points, call them x_0, x_1, $\ldots$, x_N. Call the point spacing h, so that $h = x_{i+1} - x_i$. Suppose we are given function values

$$f(x_0),\ f(x_1),\ \ldots,\ f(x_N),$$

so we know just the points, but not the entire curve, as in the plot in Figure 4.1.

Suppose we want to know $f'(x_i)$ from just this limited information. Recalling the definition of the derivative is

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h},$$

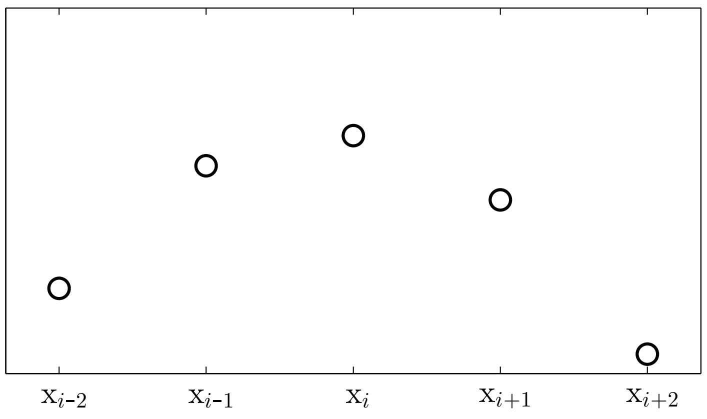


Fig. 4.1: Set of example points.

a first guess at approximating $f'(x_i)$ is to use one of

$$\textbf{2 point forward difference:} \qquad f'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{h},$$

$$\textbf{2 point backward difference:} \qquad f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{h}.$$

Illustrations of these ideas are shown in Figure 4.2. These two finite difference approximations to the derivative are simply the slopes between the adjacent points.

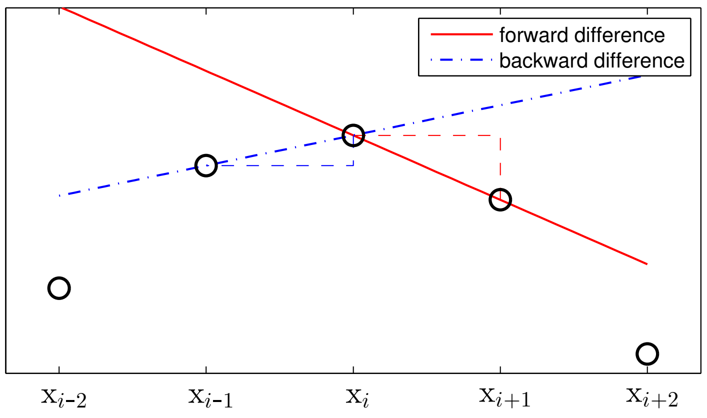


Fig. 4.2: The forward and backward finite difference approximations to $f'(x_i)$.

Important questions to ask about these approximations are "How accurate are they?" and "Is one of these approximations any better than the other?"

Answers to these questions can be determined from Taylor series. Consider the spacing of the x points, h, to be a parameter. We expect that as h gets small, the forward and backward difference approximations should approach the true value of $f'(x_i)$ (assuming no significant roundoff error, which should not be a problem if h is not too small). Using Taylor's theorem with expansion point x_i and choosing $x = x_{i+1}$ gives us

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(c)}{2}(x_{i+1} - x_i)^2,$$

for some $x_i \le c \le x_{i+1}$. Then, since $h = x_{i+1} - x_i$, this reduces to

$$f(x_{i+1}) = f(x_i) + hf'(x_i) + \frac{h^2 f''(c)}{2}.$$

Now some simple algebra yields

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

Note that the left hand side is precisely the difference between the forward difference approximation and the actual derivative $f'(x_i)$. Thus we have proved the following result.

Theorem 21. *Let $f \in C^2([x_i, x_{i+1}])$ [1]. Then, the error in approximating $f'(x_i)$ with the forward difference approximation is bounded by*

$$\left| \frac{f(x_{i+1}) - f(x_i)}{h} - f'(x_i) \right| \le \frac{h}{2} \max_{x_i \le x \le x_{i+1}} |f''(x)| = Ch,$$

where C is a constant independent of h.

The theorem above tells us the relationship between the error and the point spacing h is linear. In other words, if we cut h in half, we can expect the error to get cut (approximately) in half as well.

For the backward difference approximation, we can perform a similar analysis as for the forward difference approximation and will get the following result.

Theorem 22. *Let $f \in C^2([x_{i-1}, x_i])$. Then, the error in approximating $f'(x_i)$ with the backward difference approximation is bounded by*

$$\left| \frac{f(x_i) - f(x_{i-1})}{h} - f'(x_i) \right| \le Ch,$$

where C is a constant independent of h and depends on the value of the second derivative of f near x_i.

1 This means f is a twice differentiable function on the interval $[x_i, x_{i+1}]$.

Example 23. *Use the forward difference method, with varying h, to approximate the derivative of $f(x) = e^{\sin(x)}$ at $x = 0$. Verify numerically that the error is $O(h)$.*

We type the following commands into MATLAB and get the following output for the forward difference calculations and the associated error (note $f'(0) = 1$) for a series of decreasing h's.

```
>> h = (1/2).^[1:10];
>> fprime_fd = ( exp(sin(0+h))-exp(sin(0)))./h;
>> error = fprime_fd - 1;
>> [h',fprime_fd',error',[0,error(1:end-1)./error(2:end)]']

ans =

    5.0000e-01   1.2303e+00   2.3029e-01            0
    2.5000e-01   1.1228e+00   1.2279e-01   1.8756e+00
    1.2500e-01   1.0622e+00   6.2240e-02   1.9728e+00
    6.2500e-02   1.0312e+00   3.1218e-02   1.9937e+00
    3.1250e-02   1.0156e+00   1.5621e-02   1.9985e+00
    1.5625e-02   1.0078e+00   7.8120e-03   1.9996e+00
    7.8125e-03   1.0039e+00   3.9062e-03   1.9999e+00
    3.9062e-03   1.0020e+00   1.9531e-03   2.0000e+00
    1.9531e-03   1.0010e+00   9.7656e-04   2.0000e+00
    9.7656e-04   1.0005e+00   4.8828e-04   2.0000e+00
```

The first column is h, the second is the forward difference approximation, the third column is the error, and the fourth column is the ratios of the successive errors. To verify the method is $O(h)$, we expect that the errors get cut in half when h gets cut in half. The fourth column verifies this is true, since the ratios of successive errors converges to 2.

4.2.2 Centered difference

There are (many) more ways to approximate $f'(x_i)$ using finite differences. One thing to notice about the forward and backward differences is that if f has curvature, then, for smaller h, it must be true that one of the methods is an overestimate and the other is an underestimate. Then it makes sense that an average of the two methods might produce a better approximation. After averaging, we get a formula that could also arise from using a finite difference of the values at x_{i-1} and x_{i+1}:

$$\textbf{2 point Centered difference:} \qquad f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$$

A graphical illustration is given in Figure 4.3.

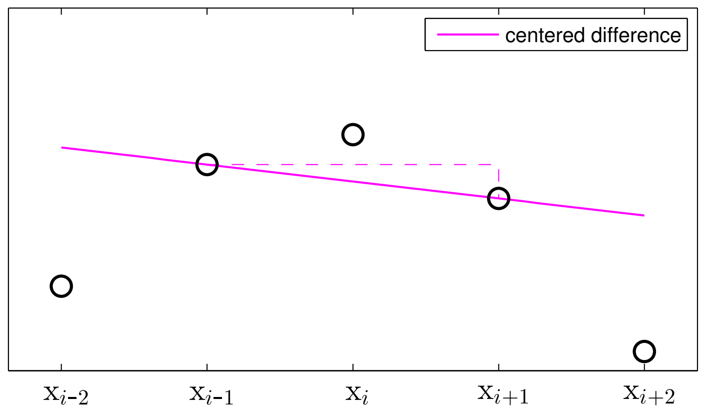


Fig. 4.3: The centered difference approximation to $f'(x_i)$.

Example 24. *Using $h = 0.1$, approximate $f'(1)$ with $f(x) = x^3$ using forward, backward, and centered difference methods. Note that the true solution is $f'(1) = 3$.*

For h=0.1, we get the approximations

$$\begin{aligned} FD &= \frac{f(1.1)-f(1)}{0.1} = 3.31 \qquad (|error| = 0.31) && (4.1)\\ BD &= \frac{f(1)-f(0.9)}{0.1} = 2.71 \qquad (|error| = 0.29) && (4.2)\\ CD &= \frac{f(1.1)-f(0.9)}{0.2} = 3.01 \qquad (|error| = 0.01) && (4.3)\end{aligned}$$

We see in the example that the centered difference method is more accurate.

As it is dangerous to draw conclusions based on one example, let's look at what some mathematical analysis tells us:

Consider the Taylor series of a function f expanded about x_i:

$$f(x) = f(x_i) + f'(x_i)(x-x_i) + \frac{f''(x_i)}{2!}(x-x_i)^2 + \frac{f'''(x_i)}{3!}(x-x_i)^3 + ...$$

Choose $x = x_{i+1}$ and then $x = x_{i-1}$ to get the two equations

$$\begin{aligned} f(x_{i+1}) &= f(x_i) + hf'(x_i) + h^2\frac{f''(x_i)}{2!} + h^3\frac{f'''(x_i)}{3!} + ...\\ f(x_{i-1}) &= f(x_i) - hf'(x_i) + h^2\frac{f''(x_i)}{2!} - h^3\frac{f'''(x_i)}{3!} + ...\end{aligned}$$

Subtracting the equations cancels out the odd terms on the right hand sides and leaves

$$f(x_{i+1}) - f(x_{i-1}) = 2hf'(x_i) + 2h^3\frac{f'''(x_i)}{3!} + ...,$$

and thus, using some algebra, we get

$$\frac{f(x_{i+1}) - f(x_{i-1})}{2h} - f'(x_i) = h^2 \frac{f'''(x_i)}{3!} + ...$$

We now see that the leading error term in the centered difference approximation is $h^2 \frac{f'''(x_i)}{3!}$. Thus, we repeat this procedure, but use Taylor's theorem to truncate at the third derivative terms. This gives us

$$\frac{f(x_{i+1}) - f(x_{i-1})}{2h} - f'(x_i) = h^2 \frac{f'''(c_1) + f'''(c_2)}{12}.$$

We have proved the following theorem:

Theorem 25. *Let $f \in C^3([x_{i-1}, x_{i+1}])$. Then, the error in approximating $f'(x_i)$ with the centered difference approximation is bounded by*

$$\left| \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - f'(x_i) \right| \leq Ch^2 = O(h^2),$$

where C is a constant with respect to h and depends on the value of the third derivative of f near x_i.

This is a big deal! We have proven that the error in centered difference approximations is $O(h^2)$, whereas the error in forward and backward differences is $O(h)$! So, if h gets cut by a factor of 10, we expect FD and BD errors to each get cut by a factor of 10, but the CD error will get cut by a factor of 100. Hence, we can expect much better answers using CD, as we suspected. Note that Theorem 25 requires more regularity of f (one more derivative needs to exist).

Example 26. *Use the centered difference method, with varying h, to approximate the derivative of $f(x) = \sin(x)$ at $x = 1$. Verify numerically that the error is $O(h^2)$.*

We type the following commands into MATLAB and get the following output for the centered difference calculations and the associated error (note $f'(1) = \cos(1)$):

```
>> h = (1/2).^[1:10];
>> fprime_cd = ( sin(1+h)-sin(1-h))./(2*h);
>> error = abs( fprime_cd - cos(1) );
>> [h',fprime_cd',error',[0,error(1:end-1)./error(2:end)]']

ans =

   5.0000e-01   1.2303e+00   2.2233e-02            0
```

```
2.5000e-01   1.1228e+00  5.6106e-03   3.9627e+00
1.2500e-01   1.0622e+00  1.4059e-03   3.9906e+00
6.2500e-02   1.0312e+00  3.5169e-04   3.9977e+00
3.1250e-02   1.0156e+00  8.7936e-05   3.9994e+00
1.5625e-02   1.0078e+00  2.1985e-05   3.9999e+00
7.8125e-03   1.0039e+00  5.4962e-06   4.0000e+00
3.9062e-03   1.0020e+00  1.3741e-06   4.0000e+00
1.9531e-03   1.0010e+00  3.4351e-07   4.0000e+00
9.7656e-04   1.0005e+00  8.5879e-08   4.0000e+00
```

As in the previous example, the first column is h, the second is the forward difference approximation, the third column is the error, and the fourth column is the ratios of the successive errors. To verify the method is $O(h^2)$, we expect that the errors get cut by four when h gets cut in half, and the fourth column verifies this is true.

4.2.3 Three point difference formulas

The centered difference formula offers a clear advantage in accuracy over the backward and forward difference formulas. However, the centered difference formula cannot be used at the endpoints. Hence if one desires to approximate $f'(x_0)$ or $f'(x_N)$ with accuracy greater than $O(h)$, we have to derive new formulas. The idea in the derivations is to use Taylor series approximations with more points - if we use only two points, we cannot do better than forward or backward difference formulas.

Hence consider deriving a formula for $f'(x_0)$ based on the points $(x_0, f(x_0))$, $(x_1, f(x_1))$, and $(x_2, f(x_2))$. Since we are going to use Taylor series approximations, the obvious choice of the expansion point is x_0. Note that this is the only way to get the equations to contain $f'(x_0)$. As for which x-points to plug in, we have already decided to use $x_0,\ x_1,\ x_2$, and since we choose x_0 as the expansion point, consider Taylor series for $x = x_1$ and $x = x_2$:

$$\begin{aligned}
f(x_1) &= f(x_0) + hf'(x_0) + h^2\frac{f''(x_0)}{2!} + h^3\frac{f'''(x_0)}{3!} + ... \\
f(x_2) &= f(x_0) + 2hf'(x_0) + (2h)^2\frac{f''(x_0)}{2!} + (2h)^3\frac{f'''(x_0)}{3!} + ...
\end{aligned}$$

The goal is to add scalar multiples of these equations together so that we get a formula in terms of $f(x_0)$, $f(x_1)$ and $f(x_2)$ that is equal to $f'(x_0) + O(h^k)$ where k is as large as possible. The idea is thus to ‘kill off’ as many terms after the $f'(x_0)$ term as possible. For these two equations, this is achieved by adding

$-4\times$ equation 1 + equation 2. This will kill the f'' terms, but will leave the f''' term. Hence we truncate the two Taylor series at the f''' term and then combine to get the equation

$$3f(x_0) - 4f(x_1) + f(x_2) = -2hf'(x_0) + \frac{h^3}{6}(-4f'''(c_1) + 8f'''(c_2)),$$

where $x_0 \le c_1 \le x_1$ and $x_0 \le c_2 \le x_2$. Assuming that f is three times differentiable near x_0, dividing through by $-2h$ gives

$$\frac{3f(x_0) - 4f(x_1) + f(x_2)}{-2h} = f'(x_0) + C(h^2),$$

with C being a constant depending on f''' but independent of h. A formula for $f'(x_N)$ can be derived analogously, and note that we could have used any three consecutive x-points and gotten an analogous result (but it typically only makes sense to do it at the endpoints, as otherwise one would just use centered difference). Thus, we have proven the following:

Theorem 27. *Let f be a three times differentiable function near x_i. Then, the error in approximating $f'(x_i)$ with the three point, one-sided difference approximation satisfies*

$$\begin{aligned} \left| \frac{-3f(x_i) + 4f(x_{i+1}) - f(x_{i+2})}{2h} - f'(x_i) \right| &= O(h^2), \\ \left| \frac{3f(x_i) - 4f(x_{i-1}) + f(x_{i-2})}{2h} - f'(x_i) \right| &= O(h^2). \end{aligned}$$

Thus, if we wanted second order accurate formulas for $f'(x_i)$, we would use the centered difference formula at all interior points and the three point formulas at the endpoints.

4.2.4 Further notes

There are some important notes to these approximations that should be considered.

- If you use more points in the formulas, and assume that higher order derivatives of f exist, you can derive higher order accurate formulas. However, even more complicated formulas will be needed at the boundary to obtain the higher order accuracy. The $O(h^2)$ formulas are by far the most common in practice.

- If you do not have equal point spacing, $O(h^2)$ formulas can still be derived at each point x_i, using Taylor series as in this section.
- If you are trying to approximate f' for given data, if the data is noisy, using these methods is probably not a good idea. A better alternative is to find a 'curve of best fit' function for the noisy data, then take the derivative of the function.

4.3 Approximating the second derivative

Similar ideas as those used for the first derivative approximations can be used for the second derivative. The procedure for deriving the formulas still use Taylor polynomials, but now we must be careful that our formulas for f'' are only in terms of function values and not in terms of f'. The following is a 3 point centered difference approximation to $f''(x_i)$:

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

We can prove this formula is $O(h^2)$.

Theorem 28. *Assume f is four times differentiable in an interval containing x_i. Then,*

$$\left| \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1})}{h^2} - f''(x_i) \right| \leq Ch^2,$$

where C is independent of h.

Proof. Let x_i be the expansion point in a Taylor series, and get 2 equations from this series by choosing x values of x_{i+1} and x_{i-1}. Then, truncate the series at the fourth derivative terms. This gives us

$$\begin{aligned} f(x_{i+1}) &= f(x_i) + hf'(x_i) + h^2\frac{f''(x_i)}{2!} + h^3\frac{f'''(x_i)}{3!} + h^4\frac{f''''(c_1)}{4!}, \\ f(x_{i-1}) &= f(x_i) - hf'(x_i) + h^2\frac{f''(x_i)}{2!} - h^3\frac{f'''(x_i)}{3!} + h^4\frac{f''''(c_2)}{4!}, \end{aligned}$$

with $x_{i-1} \leq c_2 \leq x_i \leq c_1 \leq x_{i+1}$. Now adding the formulas together and doing a little algebra provides

$$\frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1})}{h^2} = f''(x_i) + h^2\frac{f''''(c_1) + f''''(c_2)}{4!}.$$

The assumption that f is four times differentiable in an interval containing x_i means that for h small enough, the term $\frac{f''''(c_1)+f''''(c_2)}{4!}$ will be bounded above, independent of h. This completes the proof. □

More accurate (and more complicated) formulas can be derived for approximating the second derivative by using more x points in the formula. However, we will stop our derivation of formulas here.

4.4 Application: Initial value ODE's using the forward Euler method

Consider the initial value ODE: For a given f and initial data (t_0, y_0), find a function $y(t)$ satisfying

$$\begin{aligned} y'(t) &= f(t,y) \quad t_0 < t \leq T, \\ y(t_0) &= y_0. \end{aligned}$$

In a sophomore level differential equations course, students learn how to analytically solve a dozen or so special cases. But how do you solve the other infinitely many cases? Finite difference methods can be an enabling technology for such problems.

We discuss now the forward Euler method for approximating solutions to the ODE above. This is the first and simplest solver for initial value ODEs, and we will discuss and analyze this and more complex (and more accurate) methods in detail in a later chapter.

We begin by discretizing time with $N+1$ points, and let

$$t_0 < t_1 < t_2 < t_3 < ... < t_N = T$$

be these points. For simplicity, assume the points are equally spaced and call the point spacing Δt. Since we cannot find a function $y(t)$, we wish to approximate the solution $y(t)$ at each t_i; call these approximations y_i. These are the unknowns, and we want them to satisfy $y_i \approx y(t_i)$ in some sense.

Consider our ODE at a particular t, $t = t_n > t_0$. The equation must hold at t_n for $n > 0$ since it holds for all $t_0 < t \leq T$. Thus we have that

$$y'(t_n) = f(t_n, y(t_n)).$$

Applying the forward difference formula gives us

$$\frac{y(t_{n+1}) - y(t_n)}{\Delta t} \approx f(t_n, y(t_n)).$$

The **Forward Euler timestepping algorithm** is created by replacing the approximation signs by equals signs and the true solutions $y(t_n)$ by their approximations y_n:

Step 1: Given y_0

Steps n=1,2,3,...,N-1: $y_{n+1} = y_n + \Delta t f(t_n, y_n)$

This is a simple and easy-to-implement algorithm that will yield a set of points

$$(t_0, y_0),\ (t_1, y_1),\ ...,\ (t_N, y_N)$$

to approximate the true solution $y(t)$ on $[t_0, T]$. The question of accuracy will be addressed in more detail in a later chapter, but for now we just state the accuracy is $O(\Delta t)$. We know that the forward difference approximation itself is only first order, so we could not expect forward Euler to be any better than that, but it does reach this accuracy. Hence we can be assured that as we use more and more points (by cutting the timestep Δt), the error will shrink, and the forward Euler solution will converge to the true solution as $\Delta t \to 0$.

The forward Euler code is shown below, which inputs a right hand side function named func, a vector of t's (which discretize the interval $[t_0, T]$), and the initial condition.

```
function y = forwardEuler(func,t,y1)
N = length(t);
y = zeros(N,1);

% Set initial condition:
y(1)=y1;

% use forward Euler to find y(i+1) using y(i)
for i=1:N-1
    y(i+1) = y(i) + ( t(i+1) - t(i) ) * func(t(i),y(i));
end
```

Example 29. *Given the initial value problem:*

$$y' = \frac{1}{t^2} - \frac{y}{t} - y^2, \quad y(1) = -1,$$

use the forward Euler method to approximate the solution on [1,2]. Run it with 11 points, 51 points, and 101 points (so Δt = 0.1, 0.02, and 0.01 respectively), and compare your solutions to the true solution $y(t) = \frac{-1}{t}$.

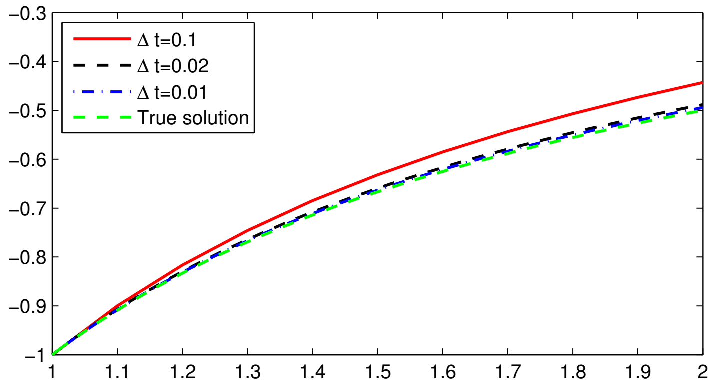
We first need to implement the right hand side $f(t,y) = \frac{1}{t^2} - \frac{y}{t} - y^2$ (here we wrote it as an inline function):

```
f = @(t,y) 1/(t^2) - y/t - y^2
```

One can call the solver via

```
t101 = linspace(1,2,101);
y101 = forwardEuler(f,t101,-1);
```

We plot the computed solutions below, along with the true solution, and observe converge of the forward Euler solutions to the true solution.



4.5 Application: Boundary value ODE's

Consider next the 1D diffusion equation with given boundary data: Find the function $u(x)$ satisfying

$$\begin{aligned} u''(x) &= f(x) \quad \text{on } a < x < b, && (4.4)\\ u(a) &= u_a, && (4.5)\\ u(b) &= u_b, && (4.6) \end{aligned}$$

for a given function f and boundary values u_a and u_b. Finite difference methods can be used to approximate solutions to equations of this form. Similar to initial value problems, the first step is to discretize the domain, so we pick N equally spaced points on [a,b], with point spacing h:

$$a = x_1 < x_2 < ... < x_N = b.$$

Since $u''(x) = f(x)$ on all of the interval (a, b), then it must be true that

$$\begin{aligned} u''(x_2) &= f(x_2),\\ u''(x_3) &= f(x_3),\\ u''(x_4) &= f(x_4),\\ \vdots \quad & \vdots \quad \vdots\\ u''(x_{N-1}) &= f(x_{N-1}). \end{aligned}$$

Now for each equation, we approximate the left hand side using the second order finite difference method. This changes the system of equations to a system of approximations

$$\begin{aligned}
u(x_1) - 2u(x_2) + u(x_3) &\approx h^2 f(x_2),\\
u(x_2) - 2u(x_3) + u(x_4) &\approx h^2 f(x_3),\\
u(x_3) - 2u(x_4) + u(x_5) &\approx h^2 f(x_4),\\
\vdots \quad &\vdots \quad \vdots\\
u(x_{N-2}) - 2u(x_{N-1}) + u(x_N) &\approx h^2 f(x_{N-1}).
\end{aligned}$$

Denote by u_n the approximation to $u(x_n)$ (for $1 \le n \le N$), and let these approximations exactly satisfy the system above. This gives us a system of linear equations for approximations u_1, u_2, ..., u_N:

$$\begin{aligned}
u_1 - 2u_2 + u_3 &= h^2 f(x_2),\\
u_2 - 2u_3 + u_4 &= h^2 f(x_3),\\
u_3 - 2u_4 + u_5 &= h^2 f(x_4),\\
\vdots \quad &\vdots \quad \vdots\\
u_{N-2} - 2u_{N-1} + u_N &= h^2 f(x_{N-1}).
\end{aligned}$$

Recall that we know the values of u_1 and u_N since they are given, so we can plug in their values and move them to the right hand sides of their equations. Now the system becomes

$$\begin{aligned}
-2u_2 + u_3 &= h^2 f(x_2) - u_a\\
u_2 - 2u_3 + u_4 &= h^2 f(x_3)\\
u_3 - 2u_4 + u_5 &= h^2 f(x_4)\\
\vdots \quad &\vdots \quad \vdots\\
u_{N-2} - 2u_{N-1} &= h^2 f(x_{N-1}) - u_b.
\end{aligned}$$

This is a system of N-2 linear equations with N-2 unknowns, which means it can be written in matrix-vector form as $\mathbf{Ax} = \mathbf{b}$, and we can use Gaussian elimination to solve it.

Example 30. *Use the finite difference method above to approximate a solution on* $[-1, 1]$ *to*

$$u''(x) = 2e^x + xe^x,$$

with boundary values $u(-1) = -e^{-1}$ *and* $u(1) = e$.

First, we proceed to setup the linear system above. Define the function f, meshwidth h, the x-points, and the boundary values. We choose N=100 points.

```
>> f = @(x) (2+x).*exp(x);
>> ua = -exp(-1);
>> ub = exp(1);
>> N=100;
>> x = linspace(-1,1,N);
```

The linear system has a $(N-2)\times(N-2)$ matrix with -2's on the diagonal, 1's on the first upper and lower diagonals, and the rest of the entries are 0. The right hand side is created with $h^2 f(x_i)$, where i goes from 2 to N-1.

```
>> d1 = -2*ones(N-2,1);
>> d2 = ones(N-3,1);
>> A = diag(d1) + diag(d2,1) + diag(d2,-1);
>> h = x(2)-x(1);
>> b = h^2 * f(x(2:N-1));
>> b(1) = b(1) - ua;
>> b(N-2) = b(N-2) - ub;
```

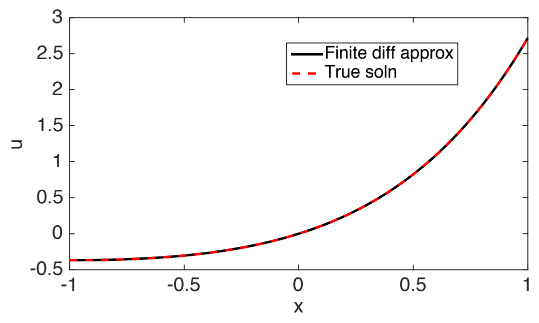
We can now solve the linear system, and then add on the endpoints to get a finite difference approximate solution at all N points:

```
>> u = A \ b';
>> u = [ua; u; ub];
```

Note that our solve is inefficient, since the matrix is very sparse, but we are treating it as a full matrix. Fixing this is an exercise.

For this test example, we know the true solution is $u_{true}(x) = xe^x$, and so we plot our solution together with the true solution to show it is accurate.

```
>> plot(x,u,'k-',x,x.*exp(x),'r--','LineWidth',2.5);
>> xlabel('x','FontSize',20)
>> ylabel('u','FontSize',20)
>> legend('Finite diff approx','True soln')
>> set(gca,'FontSize',20)
```



We observe from the plot that the two curves lay on top of each other, which means success! Note that if we did not know the true solution, we would run the code with several different values of N (say 1000, 2000, and 5000), and then make sure all three plots are the same. The method will converge with $O(h^2)$, but you must make sure your h is small enough that your numerical solution is sufficiently converged.

Lastly, we will illustrate that the method converges with $O(h^2)$. This accuracy is expected, since this is the accuracy of the finite difference approximations that were made to create the method. We calculate the solution using point spacing of h =0.01, 0.005, 0.0025, and 0.00125, and then, compare to the true solution. Shown below are the errors and error ratios for these h's (which correspond to N=201, 401, 801, and 1601):

```
N      errors        ratios
 201   1.9608e-05             0
 401   4.9020e-06    4.0000e+00
 801   1.2255e-06    4.0000e+00
1601   3.0638e-07    4.0000e+00
```

The errors were calculated for each solution as the maximum absolute error of the approximate solution at each node. The ratios of 4 indicate $O(h^2)$ convergence.

The same procedure can easily be extended to advection-diffusion-reaction system, i.e. equations of the form:

$$\alpha u''(x) + \beta u'(x) + \gamma u(x) = f(x) \quad \text{on } a < x < b, \tag{4.7}$$

$$u(a) = u_a, \tag{4.8}$$

$$u(b) = u_b, \tag{4.9}$$

where $\alpha,\ \beta,\ \gamma$ are constants. Here, we approximate the second derivative in the same way, and for the first derivative, we use the centered difference method. Before applying the boundary conditions, the resulting linear system is

$$\begin{aligned}
\alpha\frac{u_1 - 2u_2 + u_3}{h^2} + \beta\frac{u_3 - u_1}{2h} + \gamma u_2 &= f(x_2) \\
\alpha\frac{u_2 - 2u_3 + u_4}{h^2} + \beta\frac{u_4 - u_2}{2h} + \gamma u_3 &= f(x_3) \\
\alpha\frac{u_3 - 2u_4 + u_5}{h^2} + \beta\frac{u_5 - u_3}{2h} + \gamma u_4 &= f(x_4) \\
\vdots \quad &\vdots \quad \vdots \\
\alpha\frac{u_{N-2} - 2u_{N-1} + u_N}{h^2} + \beta\frac{u_N - u_{N-2}}{2h} + \gamma u_{N-1} &= f(x_{N-1})
\end{aligned}$$

Next, using that $u_1 = u_a$ and $u_N = u_b$ and doing some algebra, we get the linear system

$$\begin{aligned}
\left(\frac{-2\alpha}{h^2}+\gamma\right)u_2 + \left(\frac{\alpha}{h^2}+\frac{\beta}{2h}\right)u_3 &= f(x_2) - \frac{\alpha u_a}{h^2} + \frac{\beta u_a}{2h} \\
\left(\frac{\alpha}{h^2}-\frac{\beta}{2h}\right)u_2 + \left(\frac{-2\alpha}{h^2}+\gamma\right)u_3 + \left(\frac{\alpha}{h^2}+\frac{\beta}{2h}\right)u_4 &= f(x_3) \\
\left(\frac{\alpha}{h^2}-\frac{\beta}{2h}\right)u_3 + \left(\frac{-2\alpha}{h^2}+\gamma\right)u_4 + \left(\frac{\alpha}{h^2}+\frac{\beta}{2h}\right)u_5 &= f(x_4) \\
\vdots \quad & \vdots \quad \vdots \\
\left(\frac{\alpha}{h^2}-\frac{\beta}{2h}\right)u_{N-2} + \left(\frac{-2\alpha}{h^2}+\gamma\right)u_{N-1} &= f(x_{N-1}) - \frac{\alpha u_b}{h^2} - \frac{\beta u_b}{2h}.
\end{aligned}$$

Notice that the coefficients in each equation form a pattern, and this can be exploited to easily generate the resulting matrix. It is left as an exercise to write a MATLAB program that solves such a boundary value problem.

4.6 Exercises

1. Find Taylor series approximations using quadratic polynomials ($n = 2$) for the function $f(x) = e^x$ at x=0.9, using the expansion point $x_0 = 1$. Find an upper bound on the error using Taylor's theorem, and compare it to the actual error.
2. Prove Theorem 22. State any assumptions that you make.
3. Prove the second result in Theorem 27. State any assumptions that you make.
4. Approximate the derivative of $f(x) = \sin^2(x)$ at $x = 1$, using backward, forward and centered difference approximations. Use the same h as in the examples in this section. Verify numerically that the convergence orders are $O(h)$, $O(h)$ and $O(h^2)$, respectively.
5. Suppose a function f is twice differentiable but not three times differentiable. What would you expect for its accuracy? Would it still be $O(h^2)$? Why or why not?
6. Show that
$$\left| \frac{f(x_{n+3}) - 9f(x_{n+1}) + 8f(x_n)}{-6h} - f'(x_n) \right| = O(h^2).$$
What assumptions are needed to get this result?
7. Find a formula for $f'(x_n)$ that is $O(h^4)$ accurate that uses
$$f(x_{n-2}),\ f(x_{n-1}),\ f(x_{n+1}),\ f(x_{n+2}).$$

Test it on $f(x) = \sin(x)$ at $x = 1$ using several h values to show it is indeed fourth order accurate.

8. Write a program that uses the finite difference method to solve $u''(x) = f(x)$ on $[a, b]$, given boundary values and the number of points N. Your code should input f, a, b, u_a, u_b, and N and should output the vectors $\mathbf{x}$ and $\mathbf{u}$. The above example for $u'' = f$ is a good template to start from. Your code should solve the linear system more efficiently by having a sparse matrix A created using `spdiags` (instead of `diag`).
   Test your code on $u'' = 8\sin(2x)$ with boundary values $u(0) = 0$ and $u(\pi) = 0$. The true solution is $-2\sin(2x)$, and your code should converge to it with rate $O(h^2)$.
9. Use the finite difference method to approximate a solution to the boundary value problem
   $$y'' - 5y' - 2y = x^2 e^x, \quad -1 < x < 2, \quad y(-1) = 0, \quad y(2) = 0.$$
   Here, you must write your own code. Run the code using $N = 11$, 21, 51, 101, 201, and 401 points. Plot all the solutions on the same graph. Has it converged by N=401?
10. The differential equation
   $$W''(x) - \frac{S}{D}W(x) = \frac{-ql}{2D}x + \frac{q}{2D}x^2, \quad 0 \le x \le l$$
   describes plate deflection under an axial tension force, where l is the plate length, D is rigidity, q is intensity of the load, and S is axial force. Approximate the deflection of a plate if $q = 200\ lb/in^2$, $S = 200\ lb/in$, $D = 1,000,000\ lb/in$ and $l = 12\ in$, assuming $W(0) = W(l) = 0$. Plot your solution, and discuss why you think it is correct.
11. Solve the initial value problem
   $$y'(t) = -e^t \sin(e^t - 1), \ y(0) = 0,$$
   on $[0, 2]$, using the forward Euler method. You should run your code with several choices of Δt to make sure it converges.