MAT 772: Numerical Analysis

James V. Lambers

August 23, 2016

Contents

Chapter 1

Solution of Equations by Iteration

1.1 Nonlinear Equations

The solution of a single linear equation is an extremely simple task. We now explore the much more difficult problem of solving nonlinear equations of the form

$$f(\mathbf{x}) = \mathbf{0},$$

where $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$ can be any known function. A solution $\mathbf{x}$ of such a nonlinear equation is called a *root* of the equation, as well as a *zero* of the function f.

1.1.1 Existence and Uniqueness

For simplicity, we assume that the function $f : \mathbb{R}^n \to \mathbb{R}^m$ is continuous on the domain under consideration. Then, each equation $f_i(\mathbf{x}) = 0$, $i = 1, \ldots, m$, defines a hypersurface in $\mathbb{R}^m$. The solution of $f(\mathbf{x}) = 0$ is the intersection of these hypersurfaces, if the intersection is not empty. It is not hard to see that there can be a unique solution, infinitely many solutions, or no solution at all.

For a general equation $f(\mathbf{x}) = \mathbf{0}$, it is not possible to characterize the conditions under which a solution exists or is unique. However, in some situations, it is possible to determine existence analytically. For example, in one dimension, the Intermediate Value Theorem implies that if a continuous function $f(x)$ satisfies $f(a) \leq 0$ and $f(b) \geq 0$ where $a < b$, then $f(x) = 0$ for some $x \in (a, b)$.

Similarly, it can be concluded that $f(x) = 0$ for some $x \in (a, b)$ if the function $(x - z)f(x) \geq 0$ for $x = a$ and $x = b$, where $z \in (a, b)$. This condition can be generalized to higher dimensions. If $S \subset \mathbb{R}^n$ is an open, bounded set, and $(\mathbf{x} - \mathbf{z})^T f(\mathbf{x}) \geq 0$ for all $\mathbf{x}$ on the boundary of S and for some $\mathbf{z} \in S$, then $f(\mathbf{x}) = \mathbf{0}$ for some $\mathbf{x} \in S$. Unfortunately, checking this condition can be difficult in practice.

One useful result from calculus that can be used to establish existence and, in some sense, uniqueness of a solution is the *Inverse Function Theorem*, which states that if the Jacobian of f is nonsingular at a point $\mathbf{x}_0$, then f is invertible near $\mathbf{x}_0$ and the equation $f(\mathbf{x}) = \mathbf{y}$ has a unique solution for all $\mathbf{y}$ near $f(\mathbf{x}_0)$.

If the Jacobian of f at a point $\mathbf{x}_0$ is singular, then f is said to be *degenerate* at $\mathbf{x}_0$. Suppose that $\mathbf{x}_0$ is a solution of $f(\mathbf{x}) = \mathbf{0}$. Then, in one dimension, degeneracy means $f'(x_0) = 0$, and we say that x_0 is a *double root* of $f(x)$. Similarly, if $f^{(j)}(x_0) = 0$ for $j = 0, \ldots, m-1$, then x_0 is a root of multiplicity m. We will see that degeneracy can cause difficulties when trying to solve nonlinear equations.

1.1.2 Sensitivity

The *absolute condition number* of a function $f(x)$ is a measure of how a perturbation in x, denoted by $x + \epsilon$ for some small ϵ, is amplified by $f(x)$. Using the Mean Value Theorem, we have

$$|f(x + \epsilon) - f(x)| = |f'(c)(x + \epsilon - x)| = |f'(c)||\epsilon|$$

where c is between x and $x + \epsilon$. With ϵ being small, the absolute condition number can be approximated by $|f'(x)|$, the factor by which the perturbation in x (ϵ) is amplified to obtain the perturbation in $f(x)$.

In solving a nonlinear equation in one dimension, we are trying to solve an *inverse problem*; that is, instead of computing $y = f(x)$ (the *forward problem*), we are computing $x = f^{-1}(0)$, assuming that f is invertible near the root. It follows from the differentiation rule

$$\frac{d}{dx}[f^{-1}(x)] = \frac{1}{f'(f^{-1}(x))}$$

that the condition number for solving $f(x) = 0$ is approximately $1/|f'(x^*)|$, where x^* is the solution. This discussion can be generalized to higher dimensions, where the condition number is measured using the norm of the Jacobian.

Using *backward error analysis*, we assume that the approximate solution $\hat{x} = \hat{f}^{-1}(0)$, obtained by evaluating an approximation of f^{-1} at the exact input $y = 0$, can also be viewed as evaluating the exact function f^{-1} at a nearby input $\hat{y} = \epsilon$. That is, the approximate solution $\hat{x} = f^{-1}(\epsilon)$ is the exact solution of a nearby problem.

From this viewpoint, it can be seen from a graph that if $|f'|$ is large near x^*, which means that the condition number of the problem $f(x) = 0$ is small (that is, the problem is *well-conditioned*), then even if ϵ is relatively large, $\hat{x} = f^{-1}(\epsilon)$ is close to x^*. On the other hand, if $|f'|$ is small near x^*, so that the problem is *ill-conditioned*, then even if ϵ is small, $\hat{x}$ can be far away from x^*.

1.2 Simple Iteration

A nonlinear equation of the form $f(x) = 0$ can be rewritten to obtain an equation of the form

$$g(x) = x,$$

in which case the solution is a *fixed point* of the function g. This formulation of the original problem $f(x) = 0$ will leads to a simple solution method known as *fixed-point iteration*, or *simple iteration*. Before we describe this method, however, we must first discuss the questions of existence and uniqueness of a solution to the modified problem $g(x) = x$. The following result answers these questions.

Theorem (Brouwer's Fixed Point Theorem) *Let g be a continuous function on the interval $[a, b]$. If $g(x) \in [a, b]$ for each $x \in [a, b]$, then g has a fixed point in $[a, b]$.*

Given a continuous function g that is known to have a fixed point in an interval $[a, b]$, we can try to find this fixed point by repeatedly evaluating g at points in $[a, b]$ until we find a point x for which $g(x) = x$. This is the essence of the method of fixed-point iteration, the implementation of which we now describe.

Algorithm (Fixed-Point Iteration) Let g be a continuous function defined on the interval $[a, b]$. The following algorithm computes a number $x^* \in (a, b)$ that is a solution to the equation $g(x) = x$.

Choose an initial guess x_0 in $[a, b]$.
for $k = 0, 1, 2, \ldots$ **do**
    $x_{k+1} = g(x_k)$

```
    if |x_{k+1} − x_k| is sufficiently small then
        x* = x_{k+1}
        return x*
    end
end
```

Under what circumstances will fixed-point iteration converge to a fixed point x^*? We say that a function g that is continuous on $[a, b]$ satisfies a *Lipschitz condition* on $[a, b]$ if there exists a positive constant L such that

$$|g(x) - g(y)| \leq L|x - y|, \quad x, y \in [a, b].$$

The constant L is called a *Lipschitz constant*. If, in addition, $L < 1$, we say that g is a *contraction* on $[a, b]$.

If we denote the error in x_k by $e_k = x_k - x^*$, we can see from the fact that $g(x^*) = x^*$ that if $x_k \in [a, b]$, then

$$|e_{k+1}| = |x_{k+1} - x^*| = |g(x_k) - g(x^*)| \leq L|x_k - x^*| \leq L|e_k| < |e_k|.$$

Therefore, if g satisfies the conditions of the Brouwer Fixed-Point Theorem, and g is a contraction on $[a, b]$, and $x_0 \in [a, b]$, then fixed-point iteration is *convergent*; that is, x_k converges to x^*.

Furthermore, the fixed point x^* must be *unique*, for if there exist two distinct fixed points x^* and y^* in $[a, b]$, then, by the Lipschitz condition, we have

$$0 < |x^* - y^*| = |g(x^*) - g(y^*)| \leq L|x^* - y^*| < |x^* - y^*|,$$

which is a contradiction. Therefore, we must have $x^* = y^*$. We summarize our findings with the statement of the following result.

Theorem (Contraction Mapping Theorem) Let g be a continuous function on the interval $[a, b]$. If $g(x) \in [a, b]$ for each $x \in [a, b]$, and if there exists a constant $0 < L < 1$ such that

$$|g(x) - g(y)| \leq L|x - y|, \quad x, y \in [a, b],$$

then g has a unique fixed point x^* in $[a, b]$, and the sequence of iterates $\{x_k\}_{k=0}^{\infty}$ converges to x^*, for any initial guess $x_0 \in [a, b]$.

In general, when fixed-point iteration converges, it does so at a rate that varies inversely with the Lipschitz constant L. If the smallest possible Lipschitz constant on an interval containing x^* actually approaches zero as the width of the interval converges to zero, then the iteration can converge

much more rapidly. We will discuss convergence behavior of various methods for solving nonlinear equations in a later lecture.

Often, there are many ways to convert an equation of the form $f(x) = 0$ to one of the form $g(x) = x$, the simplest being $g(x) = x - \phi(x)f(x)$ for any function ϕ. However, it is important to ensure that the conversion yields a function g for which fixed-point iteration will converge.

Example We use fixed-point iteration to compute a fixed point of $g(x) = \cos x$ in the interval $[0, 1]$. Since $|\cos x| \leq 1$ for all x, and $\cos x \geq 0$ on $[0, \pi/2]$, and $\pi/2 > 1$, we know that $\cos x$ maps $[0, 1]$ into $[0, 1]$. Since $\cos x$ is continuous for all x, we can conclude that $\cos x$ has a fixed point in $[0, 1]$. Because $g'(x) = -\sin x$, and $|-\sin x| \leq |-\sin 1|$ on $[0, 1]$, we can apply the Mean Value Theorem to obtain

$$|\cos x - \cos y| = |-\sin c||x - y| \leq |-\sin 1||x - y|,$$

for any $x, y \in [0, 1]$, where c lies between x and y. As $|\sin 1| < 1$, we conclude that $\cos x$ is a contraction on $[0, 1]$, and therefore it has a unique fixed point on $[0, 1]$.

To use fixed-point iteration, we first choose an initial guess x_0 in $[0, 1]$. As discussed above, fixed-point iteration will converge for any initial guess, so we choose $x_0 = 0.5$. The table on page 4 shows the outcome of several iterations, in which we compute $x_{k+1} = \cos x_k$ for $k = 0, 1, 2, \ldots$. As the table shows, it takes nearly 30 iterations to obtain the fixed point to five decimal places, and there is considerable oscillation in the first iterations before a reasonable approximate solution is obtained. This oscillation is shown in Figure 1.1.

As x_k converges, it can be seen from the table that the error is reduced by a factor of roughly 2/3 from iteration to iteration. This suggests that $\cos x$ is a relatively poor choice for the iteration function $g(x)$ to solve the equation $g(x) = \cos x$. □

In general, nonlinear equations cannot be solved in a finite sequence of steps. As linear equations can be solved using *direct methods* such as Gaussian elimination, nonlinear equations usually require *iterative methods*. In iterative methods, an approximate solution is refined with each iteration until it is determined to be sufficiently accurate, at which time the iteration terminates. Since it is desirable for iterative methods to converge to the solution as rapidly as possible, it is necessary to be able to measure the speed with which an iterative method converges.

To that end, we assume that an iterative method generates a sequence of iterates $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ that converges to the exact solution $\mathbf{x}^*$. Ideally, we

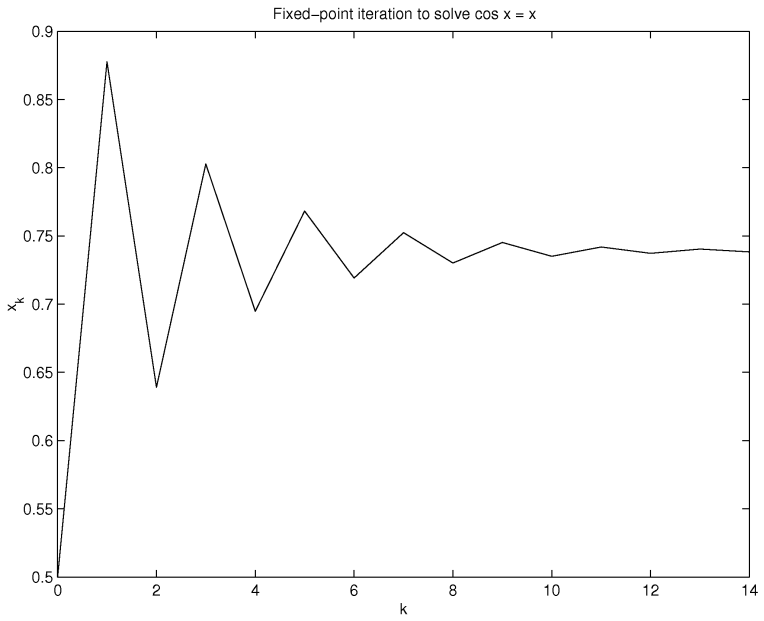

Figure 1.1: Fixed-point iteration applied to the equation $\cos x = x$, with $x_0 = 0.5$.

would like the error in a given iterate $\mathbf{x}_{k+1}$ to be much smaller than the error in the previous iterate $\mathbf{x}_k$. For example, if the error is raised to a power greater than 1 from iteration to iteration, then, because the error is typically less than 1, it will approach zero very rapidly. This leads to the following definition.

Definition (Order and Rate of Convergence) Let $\{\mathbf{x}_k\}_{k=0}^{\infty}$ be a sequence in $\mathbb{R}^n$ that converges to $\mathbf{x}^* \in \mathbb{R}^n$ and assume that $\mathbf{x}_k \neq \mathbf{x}^*$ for each k. We say that the **order of convergence** of $\{\mathbf{x}_k\}$ to $\mathbf{x}^*$ is **order** r, with **asymptotic error constant** C, if

$$\lim_{k\to\infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|^r} = C,$$

where $r \geq 1$. If $r = 1$, then the number $\rho = -\log_{10} C$ is called the **asymptotic rate of convergence**.

If $r = 1$, and $0 < C < 1$, we say that convergence is *linear*. If $r = 1$ and $C = 0$, or if $1 < r < 2$ for any positive C, then we say that convergence is

superlinear. If $r = 2$, then the method converges *quadratically*, and if $r = 3$, we say it converges *cubically*, and so on. Note that the value of C need only be bounded above in the case of linear convergence.

When convergence is linear, the asymptotic rate of convergence ρ indicates the number of correct decimal digits obtained in a single iteration. In other words, $\lfloor 1/\rho \rfloor + 1$ iterations are required to obtain an additional correct decimal digit, where $\lfloor x \rfloor$ is the "floor" of x, which is the largest integer that is less than or equal to x.

If g satisfies the conditions of the Contraction Mapping Theorem with Lipschitz constant L, then Fixed-point Iteration achieves at least linear convergence, with an asymptotic error constant that is bounded above by L. This value can be used to estimate the number of iterations needed to obtain an additional correct decimal digit, but it can also be used to estimate the *total* number of iterations needed for a specified degree of accuracy.

From the Lipschitz condition, we have, for $k \geq 1$,

$$|x_k - x^*| \leq L|x_{k-1} - x^*| \leq L^k|x_0 - x^*|.$$

From

$$|x_0 - x^*| \leq |x_0 - x_1 + x_1 - x^*| \leq |x_0 - x_1| + |x_1 - x^*| \leq |x_0 - x_1| + L|x_0 - x^*|$$

we obtain

$$|x_k - x^*| \leq \frac{L^k}{1 - L}|x_1 - x_0|.$$

Therefore, in order to satisfy $|x_k - x^*| \leq \epsilon$, the number of iterations, k, must satisfy

$$k \geq \frac{\ln|x_1 - x_0| - \ln(\epsilon(1 - L))}{\ln(1/L)}.$$

That is, we can bound the number of iterations after performing a single iteration, as long as the Lipschitz constant L is known.

We know that Fixed-point Iteration will converge to the unique fixed point in $[a, b]$ if g satisfies the conditions of the Contraction Mapping Theorem. However, if g is differentiable on $[a, b]$, its derivative can be used to obtain an alternative criterion for convergence that can be more practical than computing the Lipschitz constant L. If we denote the error in x_k by $e_k = x_k - x^*$, we can see from Taylor's Theorem and the fact that $g(x^*) = x^*$ that

$$\begin{aligned} e_{k+1} &= x_{k+1} - x^* = g(x_k) - g(x^*) = g'(x^*)(x_k - x^*) + \frac{1}{2}g''(\xi_k)(x_k - x^*)^2 \\ &= g'(x^*)e_k + O(e_k^2), \end{aligned}$$

where ξ_k lies between x_k and x^*. Therefore, if $|g'(x^*)| \leq k$, where $k < 1$, then Fixed-point Iteration is *locally convergent*; that is, it converges if x_0 is chosen sufficiently close to x^*. This leads to the following result.

Theorem (Fixed-point Theorem) Let g be a continuous function on the interval $[a, b]$, and let g be differentiable on $[a, b]$. If $g(x) \in [a, b]$ for each $x \in [a, b]$, and if there exists a constant $k < 1$ such that

$$|g'(x)| \leq k, \quad x \in (a, b),$$

then the sequence of iterates $\{x_k\}_{k=0}^{\infty}$ converges to the unique fixed point x^* of g in $[a, b]$, for any initial guess $x_0 \in [a, b]$.

It can be seen from the preceding discussion why $g'(x)$ must be bounded away from 1 on (a, b), as opposed to the weaker condition $|g'(x)| < 1$ on (a, b). If $g'(x)$ is allowed to approach 1 as x approaches a point $c \in (a, b)$, then it is possible that the error e_k might not approach zero as k increases, in which case Fixed-point Iteration would not converge.

Suppose that g satisfies the conditions of the Fixed-Point Theorem, and that g is also *continuously* differentiable on $[a, b]$. We can use the Mean Value Theorem to obtain

$$e_{k+1} = x_{k+1} - x^* = g(x_k) - g(x^*) = g'(\xi_k)(x_k - x^*) = g'(\xi_k)e_k,$$

where ξ_k lies between x_k and x^*. It follows from the continuity of g' at x^* that for any initial iterate $x_0 \in [a, b]$, Fixed-point Iteration converges linearly with asymptotic error constant $|g'(x^*)|$, since, by the definition of ξ_k and the continuity of g',

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|} = \lim_{k\to\infty} |g'(\xi_k)| = |g'(x^*)|.$$

Recall that the conditions we have stated for linear convergence are nearly identical to the conditions for g to have a unique fixed point in $[a, b]$. The only difference is that now, we also require g' to be continuous on $[a, b]$.

The derivative can also be used to indicate why Fixed-point Iteration might *not* converge.

Example The function $g(x) = x^2 + \frac{3}{16}$ has two fixed points, $x_1^* = 1/4$ and $x_2^* = 3/4$, as can be determined by solving the quadratic equation $x^2 + \frac{3}{16} = x$. If we consider the interval $[0, 3/8]$, then g satisfies the conditions of the Fixed-point Theorem, as $g'(x) = 2x < 1$ on this interval, and therefore Fixed-point Iteration will converge to x_1^* for any $x_0 \in [0, 3/8]$.

On the other hand, $g'(3/4) = 2(3/4) = 3/2 > 1$. Therefore, it is not possible for g to satisfy the conditions of the Fixed-point Theorem. Furthemore, if x_0 is chosen so that $1/4 < x_0 < 3/4$, then Fixed-point Iteration will converge to $x_1^* = 1/4$, whereas if $x_0 > 3/4$, then Fixed-point Iteration *diverges.* □

The fixed point $x_2^* = 3/4$ in the preceding example is an *unstable fixed point* of g, meaning that *no* choice of x_0 yields a sequence of iterates that converges to x_2^*. The fixed point $x_1^* = 1/4$ is a *stable fixed point* of g, meaning that *any* choice of x_0 that is sufficiently close to x_1^* yields a sequence of iterates that converges to x_1^*.

The preceding example shows that Fixed-point Iteration applied to an equation of the form $x = g(x)$ can fail to converge to a fixed point x^* if $|g'(x^*)| > 1$. We wish to determine whether this condition indicates nonconvergence in general. If $|g'(x^*)| > 1$, and g' is continuous in a neighborhood of x^*, then there exists an interval $|x - x^*| \leq \delta$ such that $|g'(x)| > 1$ on the interval. If x_k lies within this interval, it follows from the Mean Value Theorem that

$$|x_{k+1} - x^*| = |g(x_k) - g(x^*)| = |g'(\eta)||x_k - x^*|,$$

where η lies between x_k and x^*. Because η is also in this interval, we have

$$|x_{k+1} - x^*| > |x_k - x^*|.$$

In other words, the error in the iterates *increases* whenever they fall within a sufficiently small interval containing the fixed point. Because of this increase, the iterates must eventually fall outside of the interval. Therefore, it is not possible to find a k_0, for any given δ, such that $|x_k - x^*| \leq \delta$ for all $k \geq k_0$. We have thus proven the following result.

Theorem Let g have a fixed point at x^*, and let g' be continuous in a neighborhood of x^*. If $|g'(x^*)| > 1$, then Fixed-point Iteration does not converge to x^* for any initial guess x_0 except in a finite number of iterations. □

Now, suppose that in addition to the conditions of the Fixed-point Theorem, we assume that $g'(x^*) = 0$, and that g is twice continuously differentiable on $[a, b]$. Then, using Taylor's Theorem, we obtain

$$e_{k+1} = g(x_k) - g(x^*) = g'(x^*)(x_k - x^*) + \frac{1}{2}g''(\xi_k)(x_k - x^*)^2 = \frac{1}{2}g''(\xi_k)e_k^2,$$

where ξ_k lies between x_k and x^*. It follows that for any initial iterate $x_0 \in [a, b]$, Fixed-point Iteration converges at least quadratically, with asymptotic error constant $|g''(x^*)/2|$. Later, this will be exploited to obtain a quadratically convergent method for solving nonlinear equations of the form $f(x) = 0$.

1.3 Iterative Solution of Equations

Now that we understand the convergence behavior of Fixed-point Iteration, we consider the application of Fixed-point Iteration to the solution of an equation of the form $f(x) = 0$. When rewriting this equation in the form $x = g(x)$, it is essential to choose the function g wisely. One guideline is to choose $g(x) = x - \phi(x)f(x)$, where the function $\phi(x)$ is, ideally, nonzero except possibly at a solution of $f(x) = 0$. This can be satisfied by choosing $\phi(x)$ to be constant, but this can fail, as the following example illustrates.

Example Consider the equation

$$x + \ln x = 0.$$

By the Intermediate Value Theorem, this equation has a solution in the interval $[0.5, 0.6]$. Furthermore, this solution is unique. To see this, let $f(x) = x + \ln x$. Then $f'(x) = x + 1/x > 0$ on the domain of f, which means that f is increasing on its entire domain. Therefore, it is not possible for $f(x) = 0$ to have more than one solution.

We consider using Fixed-point Iteration to solve the equivalent equation

$$x = x - (1)(x + \ln x) = -\ln x.$$

However, with $g(x) = -\ln x$, we have $|g'(x)| = |-1/x| > 1$ on the interval $[0.5, 0.6]$. Therefore, by the preceding theorem, Fixed-point Iteration will fail to converge for any initial guess in this interval. We therefore apply $g^{-1}(x) = e^{-x}$ to both sides of the equation $x = g(x)$ to obtain

$$g^{-1}(x) = g^{-1}(g(x)) = x,$$

which simplifies to

$$x = e^{-x}.$$

The function $g(x) = e^{-x}$ satisfies $|g'(x)| < 1$ on $[0.5, 0.6]$, as $g'(x) = -e^{-x}$, and $e^{-x} < 1$ when the argument x is positive. By narrowing this interval to $[0.52, 0.6]$, which is mapped into itself by this choice of g, we can apply the Fixed-point Theorem to conclude that Fixed-point Iteration will converge to the unique fixed point of g for any choice of x_0 in the interval. □

1.4 Relaxation

As previously discussed, a common choice for a function $g(x)$ to use with Fixed-point Iteration to solve the equation $f(x) = 0$ is a function of the form $g(x) = x - \phi(x)f(x)$, where $\phi(x)$ is nonzero. Clearly, the simplest choice of $\phi(x)$ is a constant function $\phi(x) \equiv \lambda$, but it is important to choose λ so that Fixed-point Iteration with $g(x)$ will converge.

Suppose that x^* is a solution of the equation $f(x) = 0$, and that f is continuously differentiable in a neighborhood of x^*, with $f'(x^*) = \alpha > 0$. Then, by continuity, there exists an interval $[x^* - \delta, x^* + \delta]$ containing x^* on which $m \leq f'(x) \leq M$, for positive constants m and M. It follows that for any choice of a positive constant λ,

$$1 - \lambda M \leq 1 - \lambda f'(x) \leq 1 - \lambda m.$$

By choosing

$$\lambda = \frac{2}{M + m},$$

we obtain

$$1 - \lambda M = -k, \quad 1 - \lambda m = k, \quad k = \frac{M - m}{M + m},$$

which satisfies $0 < k < 1$. Therefore, if we define $g(x) = x - \lambda f(x)$, we have $|g'(x)| \leq k < 1$ on $[x^* - \delta, x^* + \delta]$.

Furthermore, if $|x - x^*| \leq \delta$, then, by the Mean Value Theorem,

$$|g(x) - x^*| = |g(x) - g(x^*)| = |g'(\xi)||x - x^*| < \delta,$$

and therefore g maps the interval $[x^* - \delta, x^* + \delta]$ into itself. We conclude that the Fixed-point Theorem applies, and Fixed-point Iteration converges linearly to x^* for any choice of x_0 in $[x^* - \delta, x^* + \delta]$, with asymptotic error constant $|1 - \lambda\alpha| \leq k$.

In summary, if f is continuously differentiable in a neighborhood of a root x^* of $f(x) = 0$, and $f(x^*)$ is nonzero, then there exists a constant λ such that Fixed-point Iteration with $g(x) = x - \lambda f(x)$ converges to x^* for x_0 chosen sufficiently close to x^*. This approach to Fixed-point Iteration, with a constant ϕ, is known as *relaxation*.

Convergence can be accelerated by allowing λ to vary from iteration to iteration. Intuitively, an effective choice is to try to minimize $|g'(x)|$ near x^* by setting $\lambda = 1/f'(x_k)$, for each k, so that $g'(x_k) = 1 - \lambda f'(x_k) = 0$. This results in linear convergence with an asymptotic error constant of 0, which indicates faster than linear convergence. We will see that convergence is actually quadratic.

1.5 Newton's Method

To develop a more effective method for solving this problem of computing a solution to $f(x) = 0$, we can address the following questions:

- Are there cases in which the problem easy to solve, and if so, how do we solve it in such cases?
- Is it possible to apply our method of solving the problem in these "easy" cases to more general cases?

In this course, we will see that these questions are useful for solving a variety of problems. For the problem at hand, we ask whether the equation $f(x) = 0$ is easy to solve for any particular choice of the function f. Certainly, if f is a linear function, then it has a formula of the form $f(x) = m(x-a)+b$, where m and b are constants and $m \neq 0$. Setting $f(x) = 0$ yields the equation

$$m(x - a) + b = 0,$$

which can easily be solved for x to obtain the unique solution

$$x = a - \frac{b}{m}.$$

We now consider the case where f is not a linear function. Using Taylor's theorem, it is simple to construct a linear function that approximates $f(x)$ near a given point x_0. This function is simply the first Taylor polynomial of $f(x)$ with center x_0,

$$P_1(x) = f(x_0) + f'(x_0)(x - x_0).$$

This function has a useful geometric interpretation, as its graph is the tangent line of $f(x)$ at the point $(x_0, f(x_0))$.

We can obtain an approximate solution to the equation $f(x) = 0$ by determining where the linear function $P_1(x)$ is equal to zero. If the resulting value, x_1, is not a solution, then we can repeat this process, approximating f by a linear function near x_1 and once again determining where this approximation is equal to zero. The resulting algorithm is *Newton's method*, which we now describe in detail.

Algorithm (Newton's Method) Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. The following algorithm computes an approximate solution x^* to the equation $f(x) = 0$.

Choose an initial guess x_0
for $k = 0, 1, 2, \ldots$ **do**
    if $f(x_k)$ is sufficiently small **then**
        $x^* = x_k$
        return x^*
    end
    $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$
    if $|x_{k+1} - x_k|$ is sufficiently small **then**
        $x^* = x_{k+1}$
        return x^*
    end
end

When Newton's method converges, it does so very rapidly. However, it can be difficult to ensure convergence, particularly if $f(x)$ has horizontal tangents near the solution x^*. Typically, it is necessary to choose a starting iterate x_0 that is close to x^*. As the following result indicates, such a choice, if close enough, is indeed sufficient for convergence.

Theorem *(Convergence of Newton's Method) Let f be twice continuously differentiable on the interval $[a, b]$, and suppose that $f(c) = 0$ and $f'(c) = 0$ for some $c \in [a, b]$. Then there exists a $\delta > 0$ such that Newton's Method applied to $f(x)$ converges to c for any initial guess x_0 in the interval $[c - \delta, c + \delta]$.*

Example We will use of Newton's Method in computing $\sqrt{2}$. This number satisfies the equation $f(x) = 0$ where

$$f(x) = x^2 - 2.$$

Since $f'(x) = 2x$, it follows that in Newton's Method, we can obtain the next iterate x_{n+1} from the previous iterate x_n by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - 2}{2x_n} = x_n - \frac{x_n^2}{2x_n} + \frac{2}{2x_n} = \frac{x_n}{2} + \frac{1}{x_n}.$$

We choose our starting iterate $x_0 = 1$, and compute the next several iterates as follows:

$$\begin{aligned} x_1 &= \frac{1}{2} + \frac{1}{1} = 1.5 \\ x_2 &= \frac{1.5}{2} + \frac{1}{1.5} = 1.41666667 \end{aligned}$$

$$\begin{aligned} x_3 &= 1.41421569 \\ x_4 &= 1.41421356 \\ x_5 &= 1.41421356. \end{aligned}$$

Since the fourth and fifth iterates agree in to eight decimal places, we assume that 1.41421356 is a correct solution to $f(x) = 0$, to at least eight decimal places. The first two iterations are illustrated in Figure 1.2. □

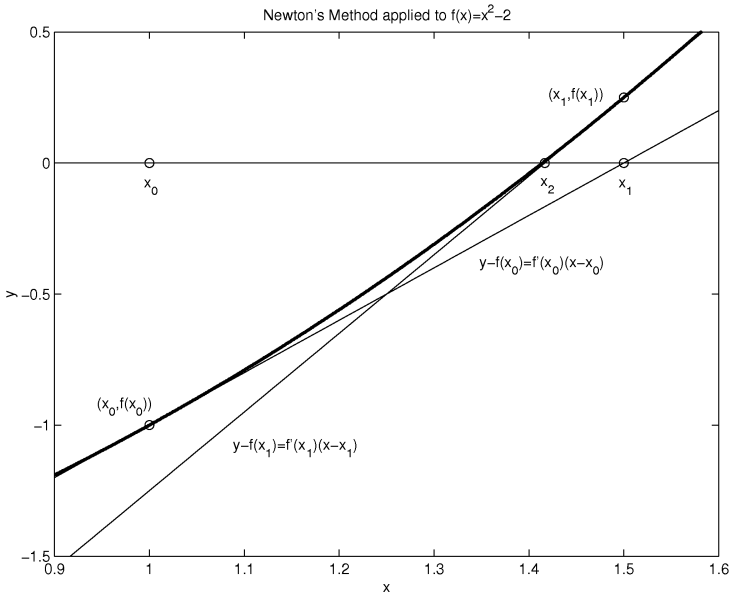

Figure 1.2: Newton's Method applied to $f(x) = x^2 - 2$. The bold curve is the graph of f. The initial iterate x_0 is chosen to be 1. The tangent line of $f(x)$ at the point $(x_0, f(x_0))$ is used to approximate $f(x)$, and it crosses the x-axis at $x_1 = 1.5$, which is much closer to the exact solution than x_0. Then, the tangent line at $(x_1, f(x_1))$ is used to approximate $f(x)$, and it crosses the x-axis at $x_2 = 1.41\overline{6}$, which is already very close to the exact solution.

Example Newton's Method can be used to compute the reciprocal of a number a *without performing any divisions*. The solution, $1/a$, satisfies the equation $f(x) = 0$, where

$$f(x) = a - \frac{1}{x}.$$

Since

$$f'(x) = \frac{1}{x^2},$$

it follows that in Newton's Method, we can obtain the next iterate x_{n+1} from the previous iterate x_n by

$$x_{n+1} = x_n - \frac{a - 1/x_n}{1/x_n^2} = x_n - \frac{a}{1/x_n}^2 + \frac{1/x_n}{1/x_n^2} = 2x_n - ax_n^2.$$

Note that no divisions are necessary to obtain x_{n+1} from x_n. This iteration was actually used on older IBM computers to implement division in hardware.

We use this iteration to compute the reciprocal of $a = 12$. Choosing our starting iterate to be 0.1, we compute the next several iterates as follows:

$$\begin{aligned} x_1 &= 2(0.1) - 12(0.1)^2 = 0.08 \\ x_2 &= 2(0.12) - 12(0.12)^2 = 0.0832 \\ x_3 &= 0.0833312 \\ x_4 &= 0.08333333333279 \\ x_5 &= 0.08333333333333. \end{aligned}$$

We conclude that 0.08333333333333 is an accurate approximation to the correct solution.

Now, suppose we repeat this process, but with an initial iterate of $x_0 = 1$. Then, we have

$$\begin{aligned} x_1 &= 2(1) - 12(1)^2 = -10 \\ x_2 &= 2(-10) - 12(-10)^2 = -1220 \\ x_3 &= 2(-1220) - 12(-1220)^2 = -17863240 \end{aligned}$$

It is clear that this sequence of iterates is not going to converge to the correct solution. In general, for this iteration to converge to the reciprocal of a, the initial iterate x_0 must be chosen so that $0 < x_0 < 2/a$. This condition guarantees that the next iterate x_1 will at least be positive. The contrast between the two choices of x_0 are illustrated in Figure 1.3 for $a = 8$. □

We now analyze the convergence of Newton's Method applied to the equation $f(x) = 0$, where we assume that f is twice continuously differentiable near the exact solution x^*. Using Taylor's Theorem, we obtain

$$e_{k+1} = x_{k+1} - x^*$$

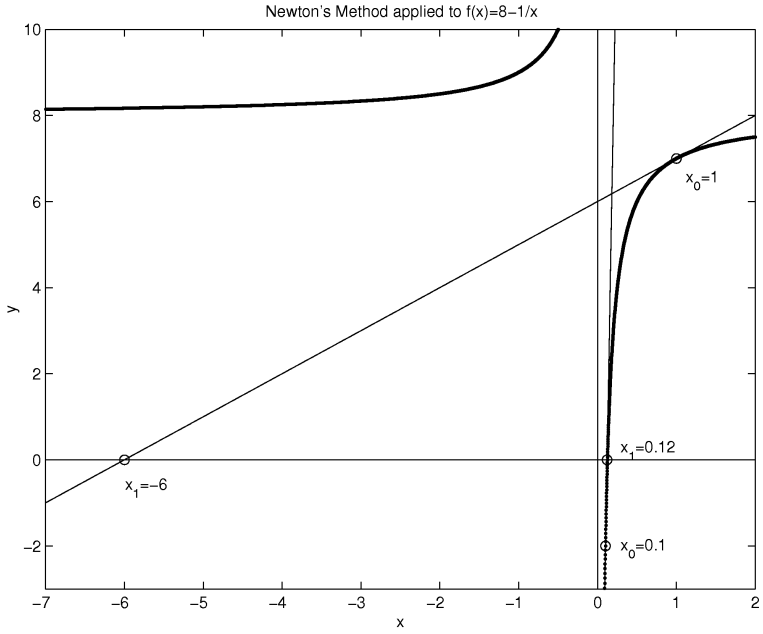

Figure 1.3: Newton's Method used to compute the reciprocal of 8 by solving the equation $f(x) = 8 - 1/x = 0$. When $x_0 = 0.1$, the tangent line of $f(x)$ at $(x_0, f(x_0))$ crosses the x-axis at $x_1 = 0.12$, which is close to the exact solution. When $x_0 = 1$, the tangent line crosses the x-axis at $x_1 = -6$, which causes searcing to continue on the wrong portion of the graph, so the sequence of iterates does not converge to the correct solution.

$$
\begin{aligned}
&= x_k - \frac{f(x_k)}{f'(x_k)} - x^* \\
&= e_k - \frac{f(x_k)}{f'(x_k)} \\
&= e_k - \frac{1}{f'(x_k)}\left[f(x^*) - f'(x_k)(x^* - x_k) - \frac{1}{2}f''(\xi_k)(x_k - x^*)^2\right] \\
&= e_k + \frac{1}{f'(x_k)}\left[f'(x_k)(x^* - x_k) + \frac{1}{2}f''(\xi_k)(x_k - x^*)^2\right] \\
&= e_k + \frac{1}{f'(x_k)}\left[-f'(x_k)e_k + \frac{1}{2}f''(\xi_k)e_k^2\right] \\
&= e_k - e_k + \frac{f''(\xi_k)}{2f'(x_k)}e_k^2
\end{aligned}
$$

$$= \frac{f''(\xi_k)}{2f'(x_k)}e_k^2$$

where ξ_k is between x_k and x^*.

Suppose that $f'(x^*) \neq 0$. Then, by the continuity of f' and f'', there exists a $\delta > 0$ such that on the interval $I_\delta = [x^* - \delta, x^* + \delta]$, there is a constant A such that

$$\frac{|f''(x)|}{|f'(y)|} \leq A, \quad x, y \in I_\delta.$$

If $x_0 \in I_\delta$ is chosen so that $|x_0 - x^*| \leq 1/A$, then, by the above Taylor expansion,

$$|x_1 - x^*| \leq \frac{1}{2}A|x_0 - x^*|^2 \leq \frac{1}{2}|x_0 - x^*|.$$

Continuing by induction, we obtain

$$|x_k - x^*| \leq 2^{-k}\min\{1/A, \delta\},$$

and therefore Newton's method converges to x^*, provided that x_0 is chosen sufficiently close to x^*.

Because, for each k, ξ_k lies between x_k and x^*, ξ_k converges to x^* as well. By the continuity of f'' on I_δ, we conclude that Newton's method converges quadratically to x^*, with asymptotic error constant

$$C = \frac{|f''(x^*)|}{2|f'(x^*)|}.$$

Example Suppose that Newton's Method is used to find the solution of $f(x) = 0$, where $f(x) = x^2 - 2$. We examine the error $e_k = x_k - x^*$, where $x^* = \sqrt{2}$ is the exact solution. We have

k	x_k	$\lvert e_k\rvert$
0	1	0.41421356237310
1	1.5	0.08578643762690
2	1.41666666666667	0.00245310429357
3	1.41421568627457	0.00000212390141
4	1.41421356237469	0.00000000000159

We can determine analytically that Newton's Method converges quadratically, and in this example, the asymptotic error constant is $|f''(\sqrt{2})/2f'(\sqrt{2})| \approx 0.35355$. Examining the numbers in the table above, we can see that the

number of correct decimal places approximately doubles with each iteration, which is typical of quadratic convergence. Furthermore, we have

$$\frac{|e_4|}{|e_3|^2} \approx 0.35352,$$

so the actual behavior of the error is consistent with the behavior that is predicted by theory. □

It is easy to see from the above analysis, however, that if $f'(x^*)$ is very small, or zero, then convergence can be very slow, or may not even occur.

Example The function

$$f(x) = (x-1)^2 e^x$$

has a double root at $x^* = 1$, and therefore $f'(x^*) = 0$. Therefore, the previous convergence analysis does not apply. Instead, we obtain

$$\begin{aligned}
e_{k+1} &= x_{k+1} - 1 \\
&= x_k - \frac{f(x_k)}{f'(x_k)} - 1 \\
&= x_k - \frac{(x_k-1)^2 e^{x_k}}{[2(x_k-1) + (x_k-1)^2]e^{x_k}} - 1 \\
&= e_k - \frac{e_k^2}{2e_k + e_k^2} \\
&= \left[1 - \frac{1}{2+e_k}\right] e_k \\
&= \left[1 - \frac{1}{x_k+1}\right] e_k \\
&= \frac{x_k}{x_k+1} e_k.
\end{aligned}$$

It follows that if we choose $x_0 > 0$, then Newton's method converges to $x^* = 1$ *linearly*, with asymptotic error constant $C = \frac{1}{2}$. □

Normally, convergence of Newton's method is only assured if x_0 is chosen sufficiently close to x^*. However, in some cases, it is possible to prove that Newton's method converges on an interval, under certain conditions on the sign of the derivatives of f on that interval. For example, suppose that on the interval $I_\delta = [x^*, x^* + \delta]$, $f'(x) > 0$ and $f''(x) > 0$, so that f is increasing and concave up on this interval.

Let $x_k \in I_\delta$. Then, from

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},$$

we have $x_{k+1} < x_k$, because f, being equal to zero at x^* and increasing on I_δ, must be positive at x_k. However, because

$$x_{k+1} - x^* = \frac{f''(\xi_k)}{2f'(x_k)}(x_k - x^*)^2,$$

and f' and f'' are both positive at x_k, we must also have $x_{k+1} > x^*$.

It follows that the sequence $\{x_k\}$ is monotonic and bounded, and therefore must be convergent to a limit $x_* \in I_\delta$. From the convergence of the sequence and the determination of x_{k+1} from x_k, it follows that $f(x_*) = 0$. However, f is positive on $(x^*, x^* + \delta]$, which means that we must have $x_* = x^*$, so Newton's method converges to x^*. Using the previous analysis, it can be shown that this convergence is quadratic.

1.6 The Secant Method

One drawback of Newton's method is that it is necessary to evaluate $f'(x)$ at various points, which may not be practical for some choices of f. The *secant method* avoids this issue by using a finite difference to approximate the derivative. As a result, $f(x)$ is approximated by a *secant line* through two points on the graph of f, rather than a tangent line through one point on the graph.

Since a secant line is defined using two points on the graph of $f(x)$, as opposed to a tangent line that requires information at only one point on the graph, it is necessary to choose two initial iterates x_0 and x_1. Then, as in Newton's method, the next iterate x_2 is then obtained by computing the x-value at which the secant line passing through the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$ has a y-coordinate of zero. This yields the equation

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_2 - x_1) + f(x_1) = 0$$

which has the solution

$$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)}$$

which can be rewritten as follows:

$$\begin{aligned} x_2 &= x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} \\ &= x_1 \frac{f(x_1) - f(x_0)}{f(x_1) - f(x_0)} - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} \\ &= \frac{x_1(f(x_1) - f(x_0)) - f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} \\ &= \frac{x_1 f(x_1) - x_1 f(x_0) - x_1 f(x_1) + x_0 f(x_1)}{f(x_1) - f(x_0)} \\ &= \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)}. \end{aligned}$$

This leads to the following algorithm.

Algorithm (Secant Method) Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous function. The following algorithm computes an approximate solution x^* to the equation $f(x) = 0$.

Choose two initial guesses x_0 and x_1
for $k = 1, 2, 3, \ldots$ **do**
    if $f(x_k)$ is sufficiently small **then**
        $x^* = x_k$
        return x^*
    end
    $x_{k+1} = \frac{x_{k-1} f(x_k) - x_k f(x_{k-1})}{f(x_k) - f(x_{k-1})}$
    if $|x_{k+1} - x_k|$ is sufficiently small **then**
        $x^* = x_{k+1}$
        return x^*
    end
end

Like Newton's method, it is necessary to choose the starting iterate x_0 to be reasonably close to the solution x^*. Convergence is not as rapid as that of Newton's Method, since the secant-line approximation of f is not as accurate as the tangent-line approximation employed by Newton's method.

Example We will use the Secant Method to solve the equation $f(x) = 0$, where $f(x) = x^2 - 2$. This method requires that we choose two initial iterates x_0 and x_1, and then compute subsequent iterates using the formula

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}, \quad n = 1, 2, 3, \ldots.$$

We choose $x_0 = 1$ and $x_1 = 1.5$. Applying the above formula, we obtain

$$\begin{aligned} x_2 &= 1.4 \\ x_3 &= 1.41379310344828 \\ x_4 &= 1.41421568627451. \end{aligned}$$

As we can see, the iterates produced by the Secant Method are converging to the exact solution $x^* = \sqrt{2}$, but not as rapidly as those produced by Newton's Method. □

We now prove that the Secant Method converges if x_0 is chosen sufficiently close to a solution x^* of $f(x) = 0$, if f is continuously differentiable near x^* and $f'(x^*) = \alpha \neq 0$. Without loss of generality, we assume $\alpha > 0$. Then, by the continuity of f', there exists an interval $I_\delta = [x^* - \delta, x^* + \delta]$ such that

$$\frac{3\alpha}{4} \leq f'(x) \leq \frac{5\alpha}{4}, \quad x \in I_\delta.$$

It follows from the Mean Value Theorem that

$$\begin{aligned} x_{k+1} - x^* &= x_k - x^* - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \\ &= x_k - x^* - \frac{f'(\theta_k)(x_k - x^*)}{f'(\varphi_k)} \\ &= \left[1 - \frac{f'(\theta_k)}{f'(\varphi_k)}\right](x_k - x^*), \end{aligned}$$

where θ_k lies between x_k and x^*, and φ_k lies between x_k and x_{k-1}. Therefore, if x_{k-1} and x_k are in I_δ, then so are φ_k and θ_k, and x_{k+1} satisfies

$$|x_{k+1} - x^*| \leq \max\left\{\left|1 - \frac{5\alpha/4}{3\alpha/4}\right|, \left|1 - \frac{3\alpha/4}{5\alpha/4}\right|\right\}|x_k - x^*| \leq \frac{2}{3}|x_k - x^*|.$$

We conclude that if $x_0, x_1 \in I_\delta$, then all subsequent iterates lie in I_δ, and the Secant Method converges at least linearly, with asymptotic rate constant $2/3$.

The order of convergence of the Secant Method can be determined using a result, which we will not prove here, stating that if $\{x_k\}_{k=0}^{\infty}$ is the sequence of iterates produced by the Secant Method for solving $f(x) = 0$, and if this sequence converges to a solution x^*, then for k sufficiently large,

$$|x_{k+1} - x^*| \approx S|x_k - x^*||x_{k-1} - x^*|$$

for some constant S.

We assume that $\{x_k\}$ converges to x^* of order α. Then, dividing both sides of the above relation by $|x_k - x^*|^\alpha$, we obtain

$$\frac{|x_{k+1} - x^*|}{|x_k - x^*|^\alpha} \approx S|x_k - x^*|^{1-\alpha}|x_{k-1} - x^*|.$$

Because α is the order of convergence, the left side must converge to a positive constant C as $k \to \infty$. It follows that the right side must converge to a positive constant as well, as must its reciprocal. In other words, there must exist positive constants C_1 and C_2

$$\frac{|x_k - x^*|}{|x_{k-1} - x^*|^\alpha} \to C_1, \quad \frac{|x_k - x^*|^{\alpha-1}}{|x_{k-1} - x^*|} \to C_2.$$

This can only be the case if there exists a nonzero constant β such that

$$\frac{|x_k - x^*|}{|x_{k-1} - x^*|^\alpha} = \left(\frac{|x_k - x^*|^{\alpha-1}}{|x_{k-1} - x^*|}\right)^\beta,$$

which implies that

$$1 = (\alpha - 1)\beta \quad \text{and} \quad \alpha = \beta.$$

Eliminating β, we obtain the equation

$$\alpha^2 - \alpha - 1 = 0,$$

which has the solutions

$$\alpha_1 = \frac{1 + \sqrt{5}}{2} \approx 1.618, \quad \alpha_2 = \frac{1 - \sqrt{5}}{2} \approx -0.618.$$

Since we must have $\alpha \geq 1$, the rate of convergence is 1.618.

1.7 The Bisection Method

Suppose that $f(x)$ is a continuous function that changes sign on the interval $[a, b]$. Then, by the Intermediate Value Theorem, $f(x) = 0$ for some $x \in [a, b]$. How can we find the solution, knowing that it lies in this interval?

The method of *bisection* attempts to reduce the size of the interval in which a solution is known to exist. Suppose that we evaluate $f(m)$, where $m = (a + b)/2$. If $f(m) = 0$, then we are done. Otherwise, f must change sign on the interval $[a, m]$ or $[m, b]$, since $f(a)$ and $f(b)$ have different signs. Therefore, we can cut the size of our search space in half, and continue

this process until the interval of interest is sufficiently small, in which case we must be close to a solution. The following algorithm implements this approach.

Algorithm (Bisection) Let f be a continuous function on the interval $[a, b]$ that changes sign on (a, b). The following algorithm computes an approximation p^* to a number p in (a, b) such that $f(p) = 0$.

```
for j = 1, 2, ... do
    p_j = (a + b)/2
    if f(p_j) = 0 or b − a is sufficiently small then
        p* = p_j
        return p*
    end
    if f(a)f(p_j) < 0 then
        b = p_j
    else
        a = p_j
    end
end
```

At the beginning, it is known that (a, b) contains a solution. During each iteration, this algorithm updates the interval (a, b) by checking whether f changes sign in the first half (a, p_j), or in the second half (p_j, b). Once the correct half is found, the interval (a, b) is set equal to that half. Therefore, at the beginning of *each* iteration, it is known that the current interval (a, b) contains a solution.

The test $f(a)f(p_j) < 0$ is used to determine whether f changes sign in the interval (a, p_j) or (p_j, b). This test is more efficient than checking whether $f(a)$ is positive and $f(p_j)$ is negative, or vice versa, since we do not care which value is positive and which is negative. We only care whether they have different signs, and if they do, then their product must be negative.

In comparison to other methods, including some that we will discuss, bisection tends to converge rather slowly, but it is also guaranteed to converge. These qualities can be seen in the following result concerning the accuracy of bisection.

Theorem *Let f be continuous on $[a, b]$, and assume that $f(a)f(b) < 0$. For each positive integer n, let p_n be the nth iterate that is produced by the bisection algorithm. Then the sequence $\{p_n\}_{n=1}^{\infty}$ converges to a number p in*

(a, b) such that $f(p) = 0$, and each iterate p_n satisfies

$$|p_n - p| \leq \frac{b-a}{2^n}.$$

It should be noted that because the nth iterate can lie anywhere within the interval (a, b) that is used during the nth iteration, it is possible that the error bound given by this theorem may be quite conservative.

Example We seek a solution of the equation $f(x) = 0$, where

$$f(x) = x^2 - x - 1.$$

Because $f(1) = -1$ and $f(2) = 1$, and f is continuous, we can use the Intermediate Value Theorem to conclude that $f(x) = 0$ has a solution in the interval $(1, 2)$, since $f(x)$ must assume every value between -1 and 1 in this interval.

We use the method of *bisection* to find a solution. First, we compute the midpoint of the interval, which is $(1 + 2)/2 = 1.5$. Since $f(1.5) = -0.25$, we see that $f(x)$ changes sign between $x = 1.5$ and $x = 2$, so we can apply the Intermediate Value Theorem again to conclude that $f(x) = 0$ has a solution in the interval $(1.5, 2)$.

Continuing this process, we compute the midpoint of the interval $(1.5, 2)$, which is $(1.5+2)/2 = 1.75$. Since $f(1.75) = 0.3125$, we see that $f(x)$ changes sign between $x = 1.5$ and $x = 1.75$, so we conclude that there is a solution in the interval $(1.5, 1.75)$. The following table shows the outcome of several more iterations of this procedure. Each row shows the current interval (a, b) in which we know that a solution exists, as well as the midpoint of the interval, given by $(a + b)/2$, and the value of f at the midpoint. Note that from iteration to iteration, only one of a or b changes, and the endpoint that changes is always set equal to the midpoint.

a	b	$m = (a + b)/2$	$f(m)$
1	2	1.5	-0.25
1.5	2	1.75	0.3125
1.5	1.75	1.625	0.015625
1.5	1.625	1.5625	-0.12109
1.5625	1.625	1.59375	-0.053711
1.59375	1.625	1.609375	-0.019287
1.609375	1.625	1.6171875	-0.0018921
1.6171875	1.625	1.62109325	0.0068512
1.6171875	1.62109325	1.619140625	0.0024757
1.6171875	1.619140625	1.6181640625	0.00029087

The correct solution, to ten decimal places, is 1.6180339887, which is the number known as the *golden ratio*. □

For this method, it is easier to determine the order of convergence if we use a different measure of the error in each iterate x_k. Since each iterate is contained within an interval $[a_k, b_k]$ where $b_k - a_k = 2^{-k}(b - a)$, with $[a, b]$ being the original interval, it follows that we can bound the error $x_k - x^*$ by $e_k = b_k - a_k$. Using this measure, we can easily conclude that bisection converges linearly, with asymptotic error constant $1/2$.

1.8 Safeguarded Methods

It is natural to ask whether it is possible to combine the rapid convergence of methods such as Newton's method with "safe" methods such as bisection that are guaranteed to converge. This leads to the concept of *safeguarded methods*, which maintain an interval within which a solution is known to exist, as in bisection, but use a method such as Newton's method to find a solution within that interval. If an iterate falls outside this interval, the safe procedure is used to refine the interval before trying the rapid method.

An example of a safeguarded method is the *method of Regula Falsi*, which is also known as the *method of false position*. It is a modification of the secant method in which the two initial iterates x_0 and x_1 are chosen so that $f(x_0) \cdot f(x_1) < 0$, thus guaranteeing that a solution lies between x_0 and x_1. This condition also guarantees that the next iterate x_2 will lie between x_0 and x_1, as can be seen by applying the Intermediate Value Theorem to the secant line passing through $(x_0, f(x_0))$ and $(x_1, f(x_1))$.

It follows that if $f(x_2) \neq 0$, then a solution must lie between x_0 and x_2, or between x_1 and x_2. In the first scenario, we use the secant line passing through $(x_0, f(x_0))$ and $(x_2, f(x_2))$ to compute the next iterate x_3. Otherwise, we use the secant line passing through $(x_1, f(x_1))$ and $(x_2, f(x_2))$. Continuing in this fashion, we obtain a sequence of smaller and smaller intervals that are guaranteed to contain a solution, as in bisection, but interval is updated using a superlinearly convergent method, the secant method, rather than simply being bisected.

Algorithm (Method of Regula Falsi) Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous function that changes sign on the interval (a, b). The following algorithm computes an approximate solution x^* to the equation $f(x) = 0$.

repeat

```
    c = (af(b) − bf(a)) / (f(b) − f(a))
    if f(c) = 0 or b − a is sufficiently small then
        x* = c
        return x*
    end
    if f(a) · f(c) < 0 then
        b = c
    else
        a = c
    end
end
```

Example We use the Method of Regula Falsi (False Position) to solve $f(x) = 0$ where $f(x) = x^2 - 2$. First, we must choose two initial guesses x_0 and x_1 such that $f(x)$ changes sign between x_0 and x_1. Choosing $x_0 = 1$ and $x_1 = 1.5$, we see that $f(x_0) = f(1) = -1$ and $f(x_1) = f(1.5) = 0.25$, so these choices are suitable.

Next, we use the Secant Method to compute the next iterate x_2 by determining the point at which the secant line passing through the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$ intersects the line $y = 0$. We have

$$\begin{aligned} x_2 &= x_0 - \frac{f(x_0)(x_1 - x_0)}{f(x_1) - f(x_0)} \\ &= 1 - \frac{(-1)(1.5 - 1)}{0.25 - (-1)} \\ &= 1 + \frac{1.5 - 1}{0.25 + 1} \\ &= 1 + \frac{0.5}{1.25} \\ &= 1.4. \end{aligned}$$

Computing $f(x_2)$, we obtain $f(1.4) = -0.04 < 0$. Since $f(x_2) < 0$ and $f(x_1) > 0$, we can use the Intermediate Value Theorem to conclude that a solution exists in the interval (x_2, x_1). Therefore, we compute x_3 by determining where the secant line through the points $(x_1, f(x_1))$ and $f(x_2, f(x_2))$ intersects the line $y = 0$. Using the formula for the Secant Method, we obtain

$$x_3 = x_1 - \frac{f(x_1)(x_2 - x_1)}{f(x_2) - f(x_1)}$$

$$\begin{aligned} &= 1.5 - \frac{(0.25)(1.4 - 1.5)}{-0.04 - 0.25} \\ &= 1.41379. \end{aligned}$$

Since $f(x_3) < 0$ and $f(x_2) < 0$, we do not know that a solution exists in the interval (x_2, x_3). However, we do know that a solution exists in the interval (x_3, x_1), because $f(x_1) > 0$. Therefore, instead of proceeding as in the Secant Method and using the Secant line determined by x_2 and x_3 to compute x_4, we use the secant line determined by x_1 and x_3 to compute x_4. □

k	x_k	$g(x_k)$
0	0.50000000000000	0.87758256189037
1	0.87758256189037	0.63901249416526
2	0.63901249416526	0.80268510068233
3	0.80268510068233	0.69477802678801
4	0.69477802678801	0.76819583128202
5	0.76819583128202	0.71916544594242
6	0.71916544594242	0.75235575942153
7	0.75235575942153	0.73008106313782
8	0.73008106313782	0.74512034135144
9	0.74512034135144	0.73500630901484
10	0.73500630901484	0.74182652264325
11	0.74182652264325	0.73723572544223
12	0.73723572544223	0.74032965187826
13	0.74032965187826	0.73824623833223
14	0.73824623833223	0.73964996276966
15	0.73964996276966	0.73870453935698
16	0.73870453935698	0.73934145228121
17	0.73934145228121	0.73891244933210
18	0.73891244933210	0.73920144413580
19	0.73920144413580	0.73900677978081
20	0.73900677978081	0.73913791076229
21	0.73913791076229	0.73904958059521
22	0.73904958059521	0.73910908142053
23	0.73910908142053	0.73906900120401
24	0.73906900120401	0.73909599983575
25	0.73909599983575	0.73907781328518
26	0.73907781328518	0.73909006398825
27	0.73909006398825	0.73908181177811
28	0.73908181177811	0.73908737057104
29	0.73908737057104	0.73908362610348
30	0.73908362610348	0.73908614842288

Chapter 2

Polynomial Interpolation

2.1 Lagrange Interpolation

Calculus provides many tools that can be used to understand the behavior of functions, but in most cases it is necessary for these functions to be continuous or differentiable. This presents a problem in most "real" applications, in which functions are used to model relationships between quantities, but our only knowledge of these functions consists of a set of discrete data points, where the data is obtained from measurements. Therefore, we need to be able to construct continuous functions based on discrete data.

The problem of constructing such a continuous function is called *data fitting*. In this lecture, we discuss a special case of data fitting known as *interpolation*, in which the goal is to find a linear combination of n known functions to fit a set of data that imposes n constraints, thus guaranteeing a unique solution that fits the data exactly, rather than approximately. The broader term "constraints" is used, rather than simply "data points", since the description of the data may include additional information such as rates of change or requirements that the fitting function have a certain number of continuous derivatives.

When it comes to the study of functions using calculus, polynomials are particularly simple to work with. Therefore, in this course we will focus on the problem of constructing a polynomial that, in some sense, fits given data. We first discuss some algorithms for computing the unique polynomial $p_n(x)$ of degree n that satisfies $p_n(x_i) = y_i$, $i = 0, \ldots, n$, where the points (x_i, y_i) are given. The points $x_0, x_1, \ldots, x_n$ are called *interpolation points*. The polynomial $p_n(x)$ is called the *interpolating polynomial* of the data (x_0, y_0), (x_1, y_1), ..., (x_n, y_n). At first, we will assume that the interpolation points

are all distinct; this assumption will be relaxed in a later lecture.

If the interpolation points $x_0, \ldots, x_n$ are distinct, then the process of finding a polynomial that passes through the points (x_i, y_i), $i = 0, \ldots, n$, is equivalent to solving a system of linear equations $A\mathbf{x} = \mathbf{b}$ that has a unique solution. However, different algorithms for computing the interpolating polynomial use a different A, since they each use a different basis for the space of polynomials of degree $\leq n$.

The most straightforward method of computing the interpolation polynomial is to form the system $A\mathbf{x} = \mathbf{b}$ where $b_i = y_i$, $i = 0, \ldots, n$, and the entries of A are defined by $a_{ij} = p_j(x_i)$, $i, j = 0, \ldots, n$, where $x_0, x_1, \ldots, x_n$ are the points at which the data $y_0, y_1, \ldots, y_n$ are obtained, and $p_j(x) = x^j$, $j = 0, 1, \ldots, n$. The basis $\{1, x, \ldots, x^n\}$ of the space of polynomials of degree $n+1$ is called the *monomial basis*, and the corresponding matrix A is called the *Vandermonde matrix* for the points $x_0, x_1, \ldots, x_n$. Unfortunately, this matrix can be ill-conditioned, especially when interpolation points are close together.

In *Lagrange interpolation*, the matrix A is simply the identity matrix, by virtue of the fact that the interpolating polynomial is written in the form

$$p_n(x) = \sum_{j=0}^{n} y_j \mathcal{L}_{n,j}(x),$$

where the polynomials $\{\mathcal{L}_{n,j}\}_{j=0}^{n}$ have the property that

$$\mathcal{L}_{n,j}(x_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}.$$

The polynomials $\{\mathcal{L}_{n,j}\}$, $j = 0, \ldots, n$, are called the *Lagrange polynomials* for the interpolation points x_0, x_1, ..., x_n. They are defined by

$$\mathcal{L}_{n,j}(x) = \prod_{k=0, k\neq j}^{n} \frac{x - x_k}{x_j - x_k}.$$

As the following result indicates, the problem of polynomial interpolation can be solved using Lagrange polynomials.

Theorem *Let $x_0, x_1, \ldots, x_n$ be $n+1$ distinct numbers, and let $f(x)$ be a function defined on a domain containing these numbers. Then the polynomial defined by*

$$p_n(x) = \sum_{j=0}^{n} f(x_j) \mathcal{L}_{n,j}$$

is the unique polynomial of degree n that satisfies

$$p_n(x_j) = f(x_j), \quad j = 0, 1, \ldots, n.$$

The polynomial $p_n(x)$ is called the *interpolating polynomial* of $f(x)$. We say that $p_n(x)$ *interpolates* $f(x)$ at the points $x_0, x_1, \ldots, x_n$.

Example We will use Lagrange interpolation to find the unique polynomial $p_3(x)$, of degree 3 or less, that agrees with the following data:

i	x_i	y_i
0	-1	3
1	0	-4
2	1	5
3	2	-6

In other words, we must have $p_3(-1) = 3$, $p_3(0) = -4$, $p_3(1) = 5$, and $p_3(2) = -6$.

First, we construct the Lagrange polynomials $\{\mathcal{L}_{3,j}(x)\}_{j=0}^{3}$, using the formula

$$\mathcal{L}_{n,j}(x) = \prod_{i=0, i\neq j}^{3} \frac{(x - x_i)}{(x_j - x_i)}.$$

This yields

$$\begin{aligned}
\mathcal{L}_{3,0}(x) &= \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} \\
&= \frac{(x-0)(x-1)(x-2)}{(-1-0)(-1-1)(-1-2)} \\
&= \frac{x(x^2-3x+2)}{(-1)(-2)(-3)} \\
&= -\frac{1}{6}(x^3-3x^2+2x) \\
\mathcal{L}_{3,1}(x) &= \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} \\
&= \frac{(x+1)(x-1)(x-2)}{(0+1)(0-1)(0-2)} \\
&= \frac{(x^2-1)(x-2)}{(1)(-1)(-2)} \\
&= \frac{1}{2}(x^3-2x^2-x+2)
\end{aligned}$$

$$\begin{aligned}
\mathcal{L}_{3,2}(x) &= \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} \\
&= \frac{(x+1)(x-0)(x-2)}{(1+1)(1-0)(1-2)} \\
&= \frac{x(x^2-x-2)}{(2)(1)(-1)} \\
&= -\frac{1}{2}(x^3-x^2-2x) \\
\mathcal{L}_{3,3}(x) &= \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} \\
&= \frac{(x+1)(x-0)(x-1)}{(2+1)(2-0)(2-1)} \\
&= \frac{x(x^2-1)}{(3)(2)(1)} \\
&= \frac{1}{6}(x^3-x).
\end{aligned}$$

By substituting x_i for x in each Lagrange polynomial $\mathcal{L}_{3,j}(x)$, for $j = 0, 1, 2, 3$, it can be verified that

$$\mathcal{L}_{3,j}(x_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}.$$

It follows that the Lagrange interpolating polynomial $p_3(x)$ is given by

$$\begin{aligned}
p_3(x) &= \sum_{j=0}^{3} y_j \mathcal{L}_{3,j}(x) \\
&= y_0\mathcal{L}_{3,0}(x) + y_1\mathcal{L}_{3,1}(x) + y_2\mathcal{L}_{3,2}(x) + y_3\mathcal{L}_{3,3}(x) \\
&= (3)\left(-\frac{1}{6}\right)(x^3-3x^2+2x) + (-4)\frac{1}{2}(x^3-2x^2-x+2) + (5)\left(-\frac{1}{2}\right)(x^3-x^2-2x) + \\
&\quad (-6)\frac{1}{6}(x^3-x) \\
&= -\frac{1}{2}(x^3-3x^2+2x) + (-2)(x^3-2x^2-x+2) - \frac{5}{2}(x^3-x^2-2x) - (x^3-x) \\
&= \left(-\frac{1}{2}-2-\frac{5}{2}-1\right)x^3 + \left(\frac{3}{2}+4+\frac{5}{2}\right)x^2 + (-1+2+5+1)\,x - 4 \\
&= -6x^3+8x^2+7x-4.
\end{aligned}$$

Substituting each x_i, for $i = 0, 1, 2, 3$, into $p_3(x)$, we can verify that we obtain $p_3(x_i) = y_i$ in each case. □

While the Lagrange polynomials are easy to compute, they are difficult to work with. Furthermore, if new interpolation points are added, all of the Lagrange polynomials must be recomputed. Unfortunately, it is not uncommon, in practice, to add to an existing set of interpolation points. It may be determined after computing the kth-degree interpolating polynomial $p_k(x)$ of a function $f(x)$ that $p_k(x)$ is not a sufficiently accurate approximation of $f(x)$ on some domain. Therefore, an interpolating polynomial of higher degree must be computed, which requires additional interpolation points.

To address these issues, we consider the problem of computing the interpolating polynomial *recursively*. More precisely, let $k > 0$, and let $p_k(x)$ be the polynomial of degree k that interpolates the function $f(x)$ at the points $x_0, x_1, \ldots, x_k$. Ideally, we would like to be able to obtain $p_k(x)$ from polynomials of degree $k-1$ that interpolate $f(x)$ at points chosen from among $x_0, x_1, \ldots, x_k$. The following result shows that this is possible.

Theorem *Let n be a positive integer, and let $f(x)$ be a function defined on a domain containing the $n+1$ distinct points $x_0, x_1, \ldots, x_n$, and let $p_n(x)$ be the polynomial of degree n that interpolates $f(x)$ at the points $x_0, x_1, \ldots, x_n$. For each $i = 0, 1, \ldots, n$, we define $p_{n-1,i}(x)$ to be the polynomial of degree $n-1$ that interpolates $f(x)$ at the points $x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. If i and j are distinct nonnegative integers not exceeding n, then*

$$p_n(x) = \frac{(x - x_j)p_{n-1,j}(x) - (x - x_i)p_{n-1,i}(x)}{x_i - x_j}.$$

This result leads to an algorithm called *Neville's Method* that computes the value of $p_n(x)$ at a given point using the values of lower-degree interpolating polynomials at x. We now describe this algorithm in detail.

Algorithm Let $x_0, x_1, \ldots, x_n$ be distinct numbers, and let $f(x)$ be a function defined on a domain containing these numbers. Given a number x^*, the following algorithm computes $y^* = p_n(x^*)$, where $p_n(x)$ is the nth interpolating polynomial of $f(x)$ that interpolates $f(x)$ at the points $x_0, x_1, \ldots, x_n$.

for $j = 0$ **to** n **do**
    $Q_j = f(x_j)$
end
for $j = 1$ **to** n **do**
    for $k = n$ **to** j **do**
        $Q_k = [(x - x_k)Q_{k-1} - (x - x_{k-j})Q_k]/(x_{k-j} - x_k)$
    end

end
$y^* = Q_n$

At the jth iteration of the outer loop, the number Q_k, for $k = n, n-1, \ldots, j$, represents the value at x of the polynomial that interpolates $f(x)$ at the points $x_k, x_{k-1}, \ldots, x_{k-j}$.

The preceding theorem can be used to compute the polynomial $p_n(x)$ itself, rather than its value at a given point. This yields an alternative method of constructing the interpolating polynomial, called *Newton interpolation*, that is more suitable for tasks such as inclusion of additional interpolation points.

2.2 Convergence

In some applications, the interpolating polynomial $p_n(x)$ is used to fit a known function $f(x)$ at the points $x_0, \ldots, x_n$, usually because $f(x)$ is not feasible for tasks such as differentiation or integration that are easy for polynomials, or because it is not easy to evaluate $f(x)$ at points other than the interpolation points. In such an application, it is possible to determine how well $p_n(x)$ approximates $f(x)$.

To that end, we assume that x is *not* one of the interpolation points $x_0, x_1, \ldots, x_n$, and we define

$$\varphi(t) = f(t) - p_n(t) - \frac{f(x) - p_n(x)}{\pi_{n+1}(x)}\pi_{n+1}(t),$$

where

$$\pi_{n+1}(x) = (x - x_0)(x - x_1)\cdots(x - x_n)$$

is a polynomial of degree $n + 1$. Because x is not one of the interpolation points, it follows that $\varphi(t)$ has at least $n + 2$ zeros: x, and the $n + 1$ interpolation points $x_0, x_1, \ldots, x_n$. Furthermore, $\pi_{n+1}(x) \neq 0$, so $\varphi(t)$ is well-defined.

If f is $n + 1$ times continuously differentiable on an interval $[a, b]$ that contains the interpolation points and x, then, by the Generalized Rolle's Theorem, $\varphi^{(n+1)}$ must have at least one zero in $[a, b]$. Therefore, at some point $\xi(x)$ in $[a, b]$, that depends on x, we have

$$0 = \varphi^{(n+1)}(\xi(x)) = f^{(n+1)}(t) - \frac{f(x) - p_n(x)}{\pi_{n+1}(x)}(n + 1)!,$$

which yields the following result.

Theorem *(Interpolation error) If f is $n+1$ times continuously differentiable on $[a, b]$, and $p_n(x)$ is the unique polynomial of degree n that interpolates $f(x)$ at the $n+1$ distinct points $x_0, x_1, \ldots, x_n$ in $[a, b]$, then for each $x \in [a, b]$,*

$$f(x) - p_n(x) = \prod_{j=0}^{n}(x - x_j)\frac{f^{(n+1)}(\xi(x))}{(n+1)!},$$

where $\xi(x) \in [a, b]$.

It is interesting to note that the error closely resembles the Taylor remainder $R_n(x)$.

If the number of data points is large, then polynomial interpolation becomes problematic since high-degree interpolation yields oscillatory polynomials, when the data may fit a smooth function.

Example Suppose that we wish to approximate the function $f(x) = 1/(1 + x^2)$ on the interval $[-5, 5]$ with a tenth-degree interpolating polynomial that agrees with $f(x)$ at 11 equally-spaced points $x_0, x_1, \ldots, x_{10}$ in $[-5, 5]$, where $x_j = -5 + j$, for $j = 0, 1, \ldots, 10$. Figure 2.1 shows that the resulting polynomial is not a good approximation of $f(x)$ on this interval, even though it agrees with $f(x)$ at the interpolation points. The following MATLAB session shows how the plot in the figure can be created.

```
>> % create vector of 11 equally spaced points in [-5,5]
>> x=linspace(-5,5,11);
>> % compute corresponding y-values
>> y=1./(1+x.^2);
>> % compute 10th-degree interpolating polynomial
>> p=polyfit(x,y,10);
>> % for plotting, create vector of 100 equally spaced points
>> xx=linspace(-5,5);
>> % compute corresponding y-values to plot function
>> yy=1./(1+xx.^2);
>> % plot function
>> plot(xx,yy)
>> % tell MATLAB that next plot should be superimposed on
>> % current one
>> hold on
>> % plot polynomial, using polyval to compute values
>> % and a red dashed curve
>> plot(xx,polyval(p,xx),'r--')
```

```
>> % indicate interpolation points on plot using circles
>> plot(x,y,'o')
>> % label axes
>> xlabel('x')
>> ylabel('y')
>> % set caption
>> title('Runge''s example')
```

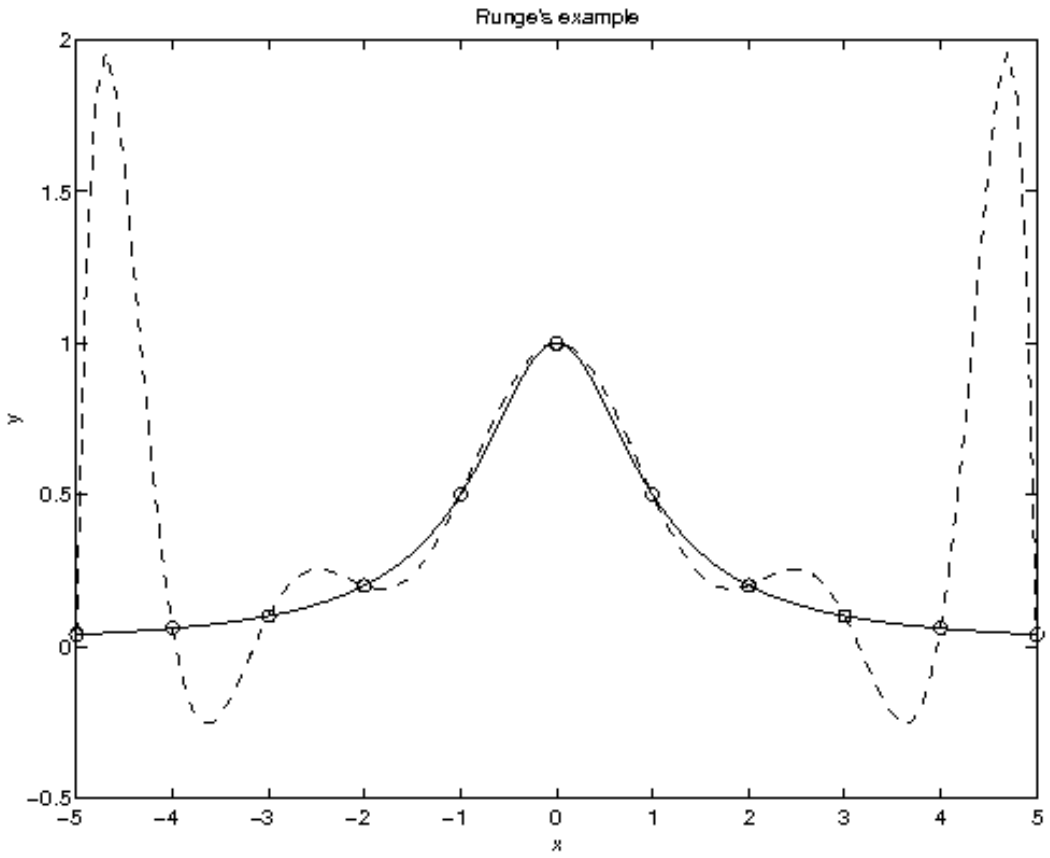

Figure 2.1: The function $f(x) = 1/(1 + x^2)$ (solid curve) cannot be interpolated accurately on $[-5, 5]$ using a tenth-degree polynomial (dashed curve) with equally-spaced interpolation points. This example that illustrates the difficulty that one can generally expect with high-degree polynomial interpolation with equally-spaced points is known as *Runge's example*.

In general, it is not wise to use a high-degree interpolating polynomial and equally-spaced interpolation points to approximate a function on an interval $[a, b]$ unless this interval is sufficiently small. The example shown in Figure 2.1 is a well-known example of the difficulty of high-degree polynomial interpolation using equally-spaced points, and it is known as *Runge's example*. □

Is it possible to choose the interpolation points so that the error is minimized? To answer this question, we introduce the *Chebyshev polynomials*

$$T_k(x) = \cos(k \cos^{-1}(x)),$$

which satisfy the three-term recurrence relation

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x), \quad T_0(x) \equiv 1, \quad T_1(x) \equiv x.$$

These polynomials have the property that $|T_k(x)| \leq 1$ on the interval $[-1, 1]$, while they grow rapidly outside of this interval. Furthermore, the roots of these polynomials lie within the interval $[-1, 1]$. Therefore, if the interpolation points $x_0, x_1, \ldots, x_n$ are chosen to be the images of the roots of the $(n+1)$st-degree Chebyshev polynomial under a linear transformation that maps $[-1, 1]$ to $[a, b]$, then it follows that

$$\left| \prod_{j=0}^{n} (x - x_j) \right| \leq \frac{1}{2^n}, \quad x \in [a, b].$$

Therefore, the error in interpolating $f(x)$ by an nth-degree polynomial is determined entirely by $f^{(n+1)}$.

2.3 Hermite Interpolation

Suppose that the interpolation points are perturbed so that two neighboring points x_i and x_{i+1}, $0 \leq i < n$, approach each other. What happens to the interpolating polynomial? In the limit, as $x_{i+1} \to x_i$, the interpolating polynomial $p_n(x)$ not only satisfies $p_n(x_i) = y_i$, but also the condition

$$p_n'(x_i) = \lim_{x_{i+1} \to x_i} \frac{y_{i+1} - y_i}{x_{i+1} - x_i}.$$

It follows that in order to ensure uniqueness, the data must specify the value of the derivative of the interpolating polynomial at x_i.

In general, the inclusion of an interpolation point x_i k times within the set $x_0, \ldots, x_n$ must be accompanied by specification of $p_n^{(j)}(x_i)$, $j = 0, \ldots, k-1$, in order to ensure a unique solution. These values are used in place of divided differences of identical interpolation points in Newton interpolation.

Interpolation with repeated interpolation points is called *osculatory interpolation*, since it can be viewed as the limit of distinct interpolation points approaching one another, and the term "osculatory" is based on the Latin word for "kiss".

In the case where each of the interpolation points $x_0, x_1, \ldots, x_n$ is repeated exactly once, the interpolating polynomial for a differentiable function $f(x)$ is called the *Hermite polynomial* of $f(x)$, and is denoted by $p_{2n+1}(x)$, since this polynomial must have degree $2n+1$ in order to satisfy the $2n+2$ constraints

$$p_{2n+1}(x_i) = f(x_i), \quad p'_{2n+1}(x_i) = f'(x_i), \quad i = 0, 1, \ldots, n.$$

To satisfy these constraints, we define, for $i = 0, 1, \ldots, n$,

$$\begin{aligned} H_i(x) &= [L_i(x)]^2(1 - 2L_i'(x_i)(x - x_i)), \\ K_i(x) &= [L_i(x)]^2(x - x_i), \end{aligned}$$

where, as before, $L_i(x)$ is the ith Lagrange polynomial for the interpolation points $x_0, x_1, \ldots, x_n$.

It can be verified directly that these polynomials satisfy, for $i, j = 0, 1, \ldots, n$,

$$\begin{aligned} H_i(x_j) = \delta_{ij}, \quad H_i'(x_j) = 0, \\ K_i(x_j) = 0, \quad K_i'(x_j) = \delta_{ij}, \end{aligned}$$

where δ_{ij} is the *Kronecker delta*

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}.$$

It follows that

$$p_{2n+1}(x) = \sum_{i=0}^{n} [f(x_i)H_i(x) + f'(x_i)K_i(x)]$$

is a polynomial of degree $2n+1$ that satisfies the above constraints.

To prove that this polynomial is the *unique* polynomial of degree $2n+1$, we assume that there is another polynomial $\tilde{p}_{2n+1}$ of degree $2n+1$ that

satisfies the constraints. Because $p_{2n+1}(x_i) = \tilde{p}_{2n+1}(x_i) = f(x_i)$ for $i = 0, 1, \ldots, n$, $p_{2n+1} - \tilde{p}_{2n+1}$ has at least $n + 1$ zeros. It follows from Rolle's Theorem that $p'_{2n+1} - \tilde{p}'_{2n+1}$ has n zeros that lie within the intervals (x_{i-1}, x_i) for $i = 0, 1, \ldots, n - 1$.

Furthermore, because $p'_{2n+1}(x_i) = \tilde{p}'_{2n+1}(x_i) = f'(x_i)$ for $i = 0, 1, \ldots, n$, it follows that $p'_{2n+1} - \tilde{p}'_{2n+1}$ has $n + 1$ additional zeros, for a total of at least $2n + 1$. However, $p'_{2n+1} - \tilde{p}'_{2n+1}$ is a polynomial of degree $2n$, and the only way that a polynomial of degree $2n$ can have $2n + 1$ zeros is if it is identically zero. Therefore, $p_{2n+1} - \tilde{p}_{2n+1}$ is a constant function, but since this function is known to have at least $n + 1$ zeros, that constant must be zero, and the Hermite polynomial is unique.

Using a similar approach as for the Lagrange interpolating polynomial, combined with ideas from the proof of the uniqueness of the Hermite polynomial, the following result can be proved.

Theorem Let f be $2n + 2$ times continuously differentiable on $[a, b]$, and let p_{2n+1} denote the Hermite polynomial of f with interpolation points $x_0, x_1, \ldots, x_n$ in $[a, b]$. Then there exists a point $\xi(x) \in [a, b]$ such that

$$f(x) - p_{2n+1}(x) = \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!}(x - x_0)^2(x - x_1)^2 \cdots (x - x_n)^2.$$

The representation of the Hermite polynomial in terms of Lagrange polynomials and their derivatives is not practical, because of the difficulty of differentiating and evaluating these polynomials. Instead, one can construct the Hermite polynomial using a Newton divided-difference table, in which each entry corresponding to two identical interpolation points is filled with the value of $f'(x)$ at the common point. Then, the Hermite polynomial can be represented using the Newton divided-difference formula.

2.4 Differentiation

We now discuss how polynomial interpolation can be applied to help solve a fundamental problem from calculus that frequently arises in scientific applications, the problem of computing the derivative of a given function $f(x)$.

2.4.1 Finite Difference Approximations

Recall that the derivative of $f(x)$ at a point x_0, denoted $f'(x_0)$, is defined by

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}.$$

This definition suggests a method for approximating $f'(x_0)$. If we choose h to be a small positive constant, then

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}.$$

This approximation is called the *forward difference formula*.

To estimate the accuracy of this approximation, we note that if $f''(x)$ exists on $[x_0, x_0 + h]$, then, by Taylor's Theorem, $f(x_0 + h) = f(x_0) + f'(x_0)h + f''(\xi)h^2/2$, where $\xi \in [x_0, x_0 + h]$. Solving for $f'(x_0)$, we obtain

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{f''(\xi)}{2}h,$$

so the error in the forward difference formula is $O(h)$. We say that this formula is *first-order accurate*.

The forward-difference formula is called a *finite difference approximation* to $f'(x_0)$, because it approximates $f'(x)$ using values of $f(x)$ at points that have a small, but finite, distance between them, as opposed to the definition of the derivative, that takes a limit and therefore computes the derivative using an "infinitely small" value of h. The forward-difference formula, however, is just one example of a finite difference approximation. If we replace h by $-h$ in the forward-difference formula, where h is still positive, we obtain the *backward-difference formula*

$$f'(x_0) \approx \frac{f(x_0) - f(x_0 - h)}{h}.$$

Like the forward-difference formula, the backward difference formula is first-order accurate.

If we average these two approximations, we obtain the *centered difference formula*

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h}.$$

To determine the accuracy of this approximation, we assume that $f'''(x)$ exists on the interval $[x_0 - h, x_0 + h]$, and then apply Taylor's Theorem again to obtain

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{f''(x_0)}{2}h^2 + \frac{f'''(\xi_+)}{6}h^3,$$

$$f(x_0 - h) = f(x_0) - f'(x_0)h + \frac{f''(x_0)}{2}h^2 - \frac{f'''(\xi_-)}{6}h^3,$$

where $\xi_+ \in [x_0, x_0+h]$ and $\xi_- \in [x_0-h, x_0]$. Subtracting the second equation from the first and solving for $f'(x_0)$ yields

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} - \frac{f'''(\xi_+) + f'''(\xi_-)}{12}h^2.$$

By the Intermediate Value Theorem, $f'''(x)$ must assume every value between $f'''(\xi_-)$ and $f'''(\xi_+)$ on the interval (ξ_-, ξ_+), including the average of these two values. Therefore, we can simplify this equation to

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} - \frac{f'''(\xi)}{6}h^2,$$

where $\xi \in [x_0-h, x_0+h]$. We conclude that the centered-difference formula is *second-order accurate*. This is due to the cancellation of the terms involving $f''(x_0)$.

Example Consider the function

$$f(x) = \frac{\sin^2\left(\frac{\sqrt{x^2+x}}{\cos x - x}\right)}{\sin\left(\frac{\sqrt{x}-1}{\sqrt{x^2+1}}\right)}.$$

Our goal is to compute $f'(0.25)$. Differentiating, using the Quotient Rule and the Chain Rule, we obtain

$$\begin{aligned} f'(x) &= \frac{2\sin\left(\frac{\sqrt{x^2+x}}{\cos x-x}\right)\cos\left(\frac{\sqrt{x^2+x}}{\cos x-x}\right)\left[\frac{2x+1}{2\sqrt{x^2+1}(\cos x-x)} + \frac{\sqrt{x^2+1}(\sin x+1)}{(\cos x-x)^2}\right]}{\sin\left(\frac{\sqrt{x}-1}{\sqrt{x^2+1}}\right)} - \\ &\quad \frac{\sin\left(\frac{\sqrt{x^2+x}}{\cos x-x}\right)\cos\left(\frac{\sqrt{x}-1}{\sqrt{x^2+1}}\right)\left[\frac{1}{2\sqrt{x}\sqrt{x^2+1}} - \frac{x(\sqrt{x}-1)}{(x^2+1)^{3/2}}\right]}{\sin^2\left(\frac{\sqrt{x}-1}{\sqrt{x^2+1}}\right)}. \end{aligned}$$

Evaluating this monstrous function at $x = 0.25$ yields $f'(0.25) = -9.066698770$.

An alternative approach is to use a *centered difference* approximation,

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

Using this formula with $x = 0.25$ and $h = 0.005$, we obtain the approximation

$$f'(0.25) \approx \frac{f(0.255) - f(0.245)}{0.01} = -9.067464295,$$

which has absolute error 7.7×10^{-4}. While this complicated function must be evaluated twice to obtain this approximation, that is still much less work than using differentiation rules to compute $f'(x)$, and then evaluating $f'(x)$, which is much more complicated than $f(x)$. $\square$

While Taylor's Theorem can be used to derive formulas with higher-order accuracy simply by evaluating $f(x)$ at more points, this process can be tedious. An alternative approach is to compute the derivative of the interpolating polynomial that fits $f(x)$ at these points. Specifically, suppose we want to compute the derivative at a point x_0 using the data

$$(x_{-j}, y_{-j}), \ldots, (x_{-1}, y_{-1}), (x_0, y_0), (x_1, y_1), \ldots, (x_k, y_k),$$

where j and k are known nonnegative integers, $x_{-j} < x_{-j+1} < \cdots < x_{k-1} < x_k$, and $y_i = f(x_i)$ for $i = -j, \ldots, k$. Then, a finite difference formula for $f'(x_0)$ can be obtained by analytically computing the derivatives of the Lagrange polynomials $\{\mathcal{L}_{n,i}(x)\}_{i=-j}^{k}$ for these points, where $n = j + k + 1$, and the values of these derivatives at x_0 are the proper weights for the function values $y_{-j}, \ldots, y_k$. If $f(x)$ is $n+1$ times continuously differentiable on $[x_{-j}, x_k]$, then we obtain an approximation of the form

$$f'(x_0) = \sum_{i=-j}^{k} y_i \mathcal{L}'_{n,i}(x_0) + \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=-j, i\neq 0}^{k} (x_0 - x_i),$$

where $\xi \in [x_{-j}, x_k]$.

Among the best-known finite difference formulas that can be derived using this approach is the second-order-accurate three-point formula

$$f'(x_0) = \frac{-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)}{2h} + \frac{f'''(\xi)}{3} h^2, \quad \xi \in [x_0, x_0 + 2h],$$

which is useful when there is no information available about $f(x)$ for $x < x_0$. If there is no information available about $f(x)$ for $x > x_0$, then we can replace h by $-h$ in the above formula to obtain a second-order-accurate three-point formula that uses the values of $f(x)$ at x_0, $x_0 - h$ and $x_0 - 2h$.

Another formula is the five-point formula

$$f'(x_0) = \frac{f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h)}{12h} + \frac{f^{(5)}(\xi)}{30} h^4, \quad \xi \in [x_0 - 2h, x_0 + 2h],$$

which is fourth-order accurate. The reason it is called a five-point formula, even though it uses the value of $f(x)$ at four points, is that it is derived from

the Lagrange polynomials for the five points $x_0 - 2h, x_0 - h, x_0, x_0 + h$, and $x_0 + 2h$. However, $f(x_0)$ is not used in the formula because $\mathcal{L}'_{4,0}(x_0) = 0$, where $\mathcal{L}_{4,0}(x)$ is the Lagrange polynomial that is equal to one at x_0 and zero at the other four points.

If we do not have any information about $f(x)$ for $x < x_0$, then we can use the following five-point formula that actually uses the values of $f(x)$ at five points,

$$f'(x_0) = \frac{-25f(x_0) + 48f(x_0 + h) - 36f(x_0 + 2h) + 16f(x_0 + 3h) - 3f(x_0 + 4h)}{12h} + \frac{f^{(5)}(\xi)}{5}h^4,$$

where $\xi \in [x_0, x_0 + 4h]$. As before, we can replace h by $-h$ to obtain a similar formula that approximates $f'(x_0)$ using the values of $f(x)$ at $x_0, x_0 - h, x_0 - 2h, x_0 - 3h$, and $x_0 - 4h$.

The strategy of differentiating Lagrange polynomials to approximate derivatives can be used to approximate higher-order derivatives. For example, the second derivative can be approximated using a centered difference formula,

$$f''(x_0) \approx \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2},$$

which is second-order accurate.

Example We will construct a formula for approximating $f'(x)$ at a given point x_0 by interpolating $f(x)$ at the points x_0, $x_0 + h$, and $x_0 + 2h$ using a second-degree polynomial $p_2(x)$, and then approximating $f'(x_0)$ by $p'_2(x_0)$. Since $p_2(x)$ should be a good approximation of $f(x)$ near x_0, especially when h is small, its derivative should be a good approximation to $f'(x)$ near this point.

Using Lagrange interpolation, we obtain

$$p_2(x) = f(x_0)\mathcal{L}_{2,0}(x) + f(x_0 + h)\mathcal{L}_{2,1}(x) + f(x_0 + 2h)\mathcal{L}_{2,2}(x),$$

where $\{\mathcal{L}_{2,j}(x)\}_{j=0}^2$ are the Lagrange polynomials for the points x_0, $x_1 = x_0 + h$ and $x_2 = x_0 + 2h$. Recall that these polynomials satisfy

$$\mathcal{L}_{2,j}(x_k) = \delta_{jk} = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{otherwise} \end{cases}.$$

Using the formula for the Lagrange polynomials,

$$\mathcal{L}_{2,j}(x) = \prod_{i=0, i\neq j}^{2} \frac{(x - x_i)}{(x_j - x_i)},$$

we obtain

$$\begin{aligned}
\mathcal{L}_{2,0}(x) &= \frac{(x-(x_0+h))(x-(x_0+2h))}{(x_0-(x_0+h))(x_0-(x_0+2h))} \\
&= \frac{x^2-(2x_0+3h)x+(x_0+h)(x_0+2h)}{2h^2}, \\
\mathcal{L}_{2,1}(x) &= \frac{(x-x_0)(x-(x_0+2h))}{(x_0+h-x_0)(x_0+h-(x_0+2h))} \\
&= \frac{x^2-(2x_0+2h)x+x_0(x_0+2h)}{-h^2}, \\
\mathcal{L}_{2,2}(x) &= \frac{(x-x_0)(x-(x_0+h))}{(x_0+2h-x_0)(x_0+2h-(x_0+h))} \\
&= \frac{x^2-(2x_0+h)x+x_0(x_0+h)}{2h^2}.
\end{aligned}$$

It follows that

$$\begin{aligned}
\mathcal{L}'_{2,0}(x) &= \frac{2x-(2x_0+3h)}{2h^2} \\
\mathcal{L}'_{2,1}(x) &= -\frac{2x-(2x_0+2h)}{h^2} \\
\mathcal{L}'_{2,2}(x) &= \frac{2x-(2x_0+h)}{2h^2}
\end{aligned}$$

We conclude that $f'(x_0) \approx p'_2(x_0)$, where

$$\begin{aligned}
p'_2(x_0) &= f(x_0)\mathcal{L}'_{2,0}(x_0) + f(x_0+h)\mathcal{L}'_{2,0}(x_0) + f(x_0+2h)\mathcal{L}'_{2,0}(x_0) \\
&\approx f(x_0)\frac{-3}{2h} + f(x_0+h)\frac{2}{h} + f(x_0+2h)\frac{-1}{2h} \\
&\approx \frac{3f(x_0)+4f(x_0+h)-f(x_0+2h)}{2h}.
\end{aligned}$$

Using Taylor's Theorem to write $f(x_0+h)$ and $f(x_0+2h)$ in terms of Taylor polynomials centered at x_0, it can be shown that the error in this approximation is $O(h^2)$, and that this formula is exact when $f(x)$ is a polynomial of degree 2 or less. □

In a practical implementation of finite difference formulas, it is essential to note that roundoff error in evaluating $f(x)$ is bounded independently of the spacing h between points at which $f(x)$ is evaluated. It follows that the roundoff error in the approximation of $f'(x)$ actually *increases* as h decreases, because the errors incurred by evaluating $f(x)$ are divided by h.

Therefore, one must choose h sufficiently small so that the finite difference formula can produce an accurate approximation, and sufficiently large so that this approximation is not too contaminated by roundoff error.

Chapter 3

Numerical Integration

3.1 Integration

Numerous applications call for the computation of the integral of some function $f : \mathbb{R} \to \mathbb{R}$ over an interval $[a, b]$,

$$I(f) = \int_a^b f(x)\,dx.$$

In some cases, $I(f)$ can be computed by applying the Fundamental Theorem of Calculus and computing

$$I(f) = F(b) - F(a),$$

where $F(x)$ is an *antiderivative* of f. Unfortunately, this is not practical if an antiderivative of f is not available. In such cases, numerical techniques must be employed instead.

3.2 Well-Posedness

The integral $I(f)$ is defined using sequences of *Riemann sums*

$$R_n = \sum_{i=1}^{n} f(\xi_i)(x_{i+1} - x_i), \quad x_i \leq \xi_i \leq x_{i+1},$$

where $a = x_1 < x_2 < \cdots < x_n = b$. If any such sequence $\{R_n\}_{n=1}^{\infty}$ converges to the same finite value R as $n \to \infty$, then f is said to be *Riemann integrable* on $[a, b]$, and $I(f) = R$. A function is Riemann integrable if and only if it is

bounded and continuous except on a set of measure zero. For such functions, the problem of computing $I(f)$ has a unique solution, by the definition of $I(f)$.

To determine the sensitivity of $I(f)$, we define the ∞-norm of a function $f(x)$ by

$$\|f\|_\infty = \max_{x\in[a,b]} |f(x)|$$

and let $\hat{f}$ be a perturbation of f that is also Riemann integrable. Then the condition number of the problem of computing $I(f)$ can be approximated by

$$\begin{aligned}
\frac{|I(\hat{f}) - I(f)|/|I(f)|}{\|\hat{f} - f\|_\infty/\|f\|_\infty} &= \frac{|I(\hat{f} - f)|/|I(f)|}{\|\hat{f} - f\|_\infty/\|f\|_\infty} \\
&\leq \frac{I(|\hat{f} - f|)/|I(f)|}{\|\hat{f} - f\|_\infty/\|f\|_\infty} \\
&\leq \frac{(b-a)\|\hat{f} - f\|_\infty/|I(f)|}{\|\hat{f} - f\|_\infty/\|f\|_\infty} \\
&\leq (b-a)\frac{\|f\|_\infty}{|I(f)|},
\end{aligned}$$

from which it follows that the problem is fairly well-conditioned in most cases, since, if $I(f)$ is small, we should then use the absolute condition number, which is bounded by $(b-a)$. Similarly, perturbations of the endpoints a and b do not lead to large perturbations in $I(f)$, in most cases.

3.2.1 Newton-Cotes Quadrature

Clearly, if f is a Riemann integrable function and $\{R_n\}_{n=1}^\infty$ is any sequence of Riemann sums that converges to $I(f)$, then any particular Riemann sum R_n can be viewed as an approximation of $I(f)$. However, such an approximation is usually not practical since a large value of n may be necessary to achieve sufficient accuracy.

Instead, we use a *quadrature rule* to approximate $I(f)$. A quadrature rule is a sum of the form

$$Q_n(f) = \sum_{i=1}^{n} f(x_i)w_i,$$

where the points x_i, $i = 1, \ldots, n$, are called the *nodes* of the quadrature rule, and the numbers w_i, $i = 1, \ldots, n$, are the *weights*. We say that a quadrature

rule is *open* if the nodes do not include the endpoints a and b, and *closed* if they do.

The objective in designing quadrature rules is to achieve sufficient accuracy in approximating $I(f)$, for any Riemann integrable function f, while using as few nodes as possible in order to maximize efficiency. In order to determine suitable nodes and weights, we consider the following questions:

- For what functions f is $I(f)$ easy to compute?
- Given a general Riemann integrable function f, can $I(f)$ be approximated by the integral of a function g for which $I(g)$ is easy to compute?

One class of functions for which integrals are easily evaluated is the class of polynomial functions. If we choose n nodes $x_1, \ldots, x_n$, then any polynomial $p_{n-1}(x)$ of degree $n-1$ can be written in the form

$$p_{n-1}(x) = \sum_{i=1}^{n} p_{n-1}(x_i)\mathcal{L}_{n-1,i}(x),$$

where $\mathcal{L}_{n-1,i}(x)$ is the ith Lagrange polynomial for the points $x_1, \ldots, x_n$. It follows that

$$\begin{aligned} I(p_{n-1}) &= \int_a^b p_{n-1}(x)\,dx \\ &= \int_a^b \sum_{i=1}^{n} p_{n-1}(x_i)\mathcal{L}_{n-1,i}(x)\,dx \\ &= \sum_{i=1}^{n} p_{n-1}(x_i)\left(\int_a^b \mathcal{L}_{n-1,i}(x)\,dx\right) \\ &= \sum_{i=1}^{n} p_{n-1}(x_i)w_i \\ &= Q_n(p_{n-1}) \end{aligned}$$

where

$$w_i = \int_a^b \mathcal{L}_{n-1,i}(x)\,dx, \quad i = 1, \ldots, n,$$

are the weights of a quadrature rule with nodes $x_1, \ldots, x_n$.

Therefore, *any* n-point quadrature rule computes $I(f)$ *exactly* when f is a polynomial of degree less than n. For a more general function f, we can use this quadrature rule to approximate $I(f)$ by $I(p_{n-1})$, where p_{n-1}

is the polynomial that interpolates f at the points $x_1, \ldots, x_n$. Quadrature rules that use the weights defined above for given nodes $x_1, \ldots, x_n$ are called *interpolatory* quadrature rules. We say that an interpolatory quadrature rule has *degree of accuracy* n if it integrates polynomials of degree n exactly, but is not exact for polynomials of degree $n+1$.

If the weights w_i, $i = 1, \ldots, n$, are nonnegative, then the quadrature rule is stable, as its absolute condition number can be bounded by $(b-a)$, which is the same absolute condition number as the underlying integration problem. However, if any of the weights are negative, then the condition number can be arbitrarily large.

The family of *Newton-Cotes* quadrature rules consists of interpolatory quadrature rules in which the nodes are equally spaced points within the interval $[a, b]$. The most commonly used Newton-Cotes rules are

- The *Midpoint Rule*, which is an open rule with one node, is defined by

$$\int_a^b f(x)\,dx \approx (b-a) f\left(\frac{a+b}{2}\right).$$

  It is of degree one, and it is based on the principle that the area under $f(x)$ can be approximated by the area of a rectangle with width $b-a$ and height $f(m)$, where $m = (a+b)/2$ is the midpoint of the interval $[a, b]$.

- The *Trapezoidal Rule*, which is a closed rule with two nodes, is defined by

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}[f(a) + f(b)].$$

  It is of degree one, and it is based on the principle that the area under $f(x)$ from $x = a$ to $x = b$ can be approximated by the area of a trapezoid with heights $f(a)$ and $f(b)$ and width $b-a$.

- *Simpson's Rule*, which is a closed rule with three nodes, is defined by

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right].$$

  It is of degree three, and it is derived by computing the integral of the quadratic polynomial that interpolates $f(x)$ at the points a, $(a+b)/2$, and b.

Example Let $f(x) = x^3$, $a = 0$ and $b = 1$. We have

$$\int_a^b f(x)\,dx = \int_0^1 x^3\,dx = \left.\frac{x^4}{4}\right|_0^1 = \frac{1}{4}.$$

Approximating this integral with the Midpoint Rule yields

$$\int_0^1 x^3\,dx \approx (1-0)\left(\frac{0+1}{2}\right)^3 = \frac{1}{8}.$$

Using the Trapezoidal Rule, we obtain

$$\int_0^1 x^3\,dx \approx \frac{1-0}{2}[0^3 + 1^3] = \frac{1}{2}.$$

Finally, Simpson's Rule yields

$$\int_0^1 x^3\,dx \approx \frac{1-0}{6}\left[0^3 + 4\left(\frac{0+1}{2}\right)^3 + 1^3\right] = \frac{1}{6}\left[0 + 4\frac{1}{8} + 1\right] = \frac{1}{4}.$$

That is, the approximation of the integral by Simpson's Rule is actually exact, which is expected because Simpson's Rule is of degree three. On the other hand, if we approximate the integral of $f(x) = x^4$ from 0 to 1, Simpson's Rule yields 5/24, while the exact value is 1/5. Still, this is a better approximation than those obtained using the Midpoint Rule (1/16) or the Trapezoidal Rule (1/2). □

In general, the degree of accuracy of Newton-Cotes rules can easily be determined by expanding the integrand $f(x)$ in a Taylor series. This technique can be used to show that n-point Newton-Cotes rules with an odd number of nodes have degree n, which is surprising since, in general, interpolatory n-point quadrature rules have degree $n-1$. This extra degree of accuracy is due to the cancellation of the high-order error terms in the Taylor expansion used to determine the error. Such cancellation does not occur with Newton-Cotes rules that have an even number of nodes.

3.3 Error Estimates

The error in any interpolatory quadrature rule defined on an interval $[a, b]$, such as a Newton-Cotes rule or a Clenshaw-Curtis rule can be obtained by computing the integral from a to b of the error in the polynomial interpolant on which the rule is based.

For the Trapezoidal Rule, which is obtained by integrating a linear polynomial that interpolates the integrand $f(x)$ at $x = a$ and $x = b$, this approach to error analysis yields

$$\int_a^b f(x)\,dx - \frac{b-a}{2}[f(a)+f(b)] = \int_a^b \frac{f''(\xi(x))}{2}(x-a)(x-b)\,dx,$$

where $\xi(x)$ lies in $[a,b]$ for $a \le x \le b$. The function $(x-a)(x-b)$ does not change sign on $[a,b]$, which allows us to apply the Weighted Mean Value Theorem for Integrals and obtain a more useful expression for the error,

$$\int_a^b f(x)\,dx - \frac{b-a}{2}[f(a)+f(b)] = \frac{f''(\eta)}{2}\int_a^b (x-a)(x-b)\,dx = -\frac{f''(\eta)}{12}(b-a)^3,$$

where $a \le \eta \le b$. Because the error depends on the second derivative, it follows that the Trapezoidal Rule is exact for any linear function.

A similar approach can be used to obtain expressions for the error in the Midpoint Rule and Simpson's Rule, although the process is somewhat more complicated due to the fact that the functions $(x-m)$, for the Midpoint Rule, and $(x-a)(x-m)(x-b)$, for Simpson's Rule, where in both cases $m = (a+b)/2$, change sign on $[a,b]$, thus making the Weighted Mean Value Theorem for Integrals impossible to apply in the same straightforward manner as it was for the Trapezoidal Rule.

We instead use the following approach, illustrated for the Midpoint Rule. We assume that f is twice continuously differentiable on $[a,b]$. First, we make a change of variable

$$x = \frac{a+b}{2} + \frac{b-a}{2}t, \quad t \in [-1,1],$$

to map the interval $[-1,1]$ to $[a,b]$, and then define $F(t) = f(x(t))$. The error in the Midpoint Rule is then given by

$$\int_a^b f(x)\,dx - (b-a)f\left(\frac{a+b}{2}\right) = \frac{b-a}{2}\left[\int_{-1}^1 F(\tau)\,d\tau - 2F(0)\right].$$

We now define

$$G(t) = \int_{-t}^t F(\tau)\,d\tau - 2tF(0).$$

It is easily seen that the error in the Midpoint Rule is $\frac{1}{2}(b-a)G(1)$. We then define

$$H(t) = G(t) - t^3 G(1).$$

Because $H(0) = H(1) = 0$, it follows from Rolle's Theorem that there exists a point $\xi_1 \in (0, 1)$ such that $H'(\xi_1) = 0$. However, from

$$H'(0) = G'(0) = [F(t) + F(-t)]|_{t=0} - 2F(0) = 2F(0) - 2F(0) = 0,$$

it follows from Rolle's Theorem that there exists a point $\xi_2 \in (0, 1)$ such that $H''(\xi_2) = 0$.

From

$$H''(t) = G''(t) - 6tG(1) = F'(t) - F'(-t) - 6tG(1),$$

and the Mean Value Theorem, we obtain, for some $\xi_3 \in (-1, 1)$,

$$0 = H''(\xi_2) = 2\xi_2 F''(\xi_3) - 6\xi_2 G(1),$$

or

$$G(1) = \frac{1}{3}F''(\xi_3) = \frac{1}{3}\left(\frac{b-a}{2}\right)^2 f''(x(\xi_3)).$$

Multiplying by $(b - a)/2$ yields the error in the Midpoint Rule.

The result of the analysis is that for the Midpoint Rule,

$$\int_a^b f(x)\,dx - (b-a)f\left(\frac{a+b}{2}\right) = \frac{f''(\eta)}{24}(b-a)^3,$$

and for Simpson's Rule,

$$\int_a^b f(x)\,dx - \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] = -\frac{f^{(4)}(\eta)}{90}\left(\frac{b-a}{2}\right)^5,$$

where, in both cases, η is some point in $[a, b]$.

It follows that the Midpoint Rule is exact for any linear function, just like the Trapezoidal Rule, even though it uses one less interpolation point, because of the cancellation that results from choosing the midpoint of $[a, b]$ as the interpolation point. Similar cancellation causes Simpson's Rule to be exact for polynomials of degree three or less, even though it is obtained by integrating a quadratic interpolant over $[a, b]$.

3.4 The Runge Phenomenon Revisited

Unfortunately, Newton-Cotes rules are not practical when the number of nodes is large, due to the inaccuracy of high-degree polynomial interpolation

using equally spaced points. Furthermore, for $n \geq 11$, n-point Newton-Cotes rules have at least one negative weight, and therefore such rules can be ill-conditioned. This can be seen by revisiting *Runge's Example*, and attempting to approximate

$$\int_{-5}^{5} \frac{1}{1+x^2}\,dx$$

using a Newton-Cotes rule. As n increases, the approximate integral does not converge to the exact result; in fact, it increases without bound.

3.5 Composite Formulae

When using a quadrature rule to approximate $I(f)$ on some interval $[a, b]$, the error is proportional to h^r, where $h = b - a$ and r is some positive integer. Therefore, if the interval $[a, b]$ is large, it is advisable to divide $[a, b]$ into smaller intervals, use a quadrature rule to compute the integral of f on each subinterval, and add the results to approximate $I(f)$. Such a scheme is called a *composite quadrature rule*.

It can be shown that the approximate integral obtained using a composite rule that divides $[a, b]$ into n subintervals will converge to $I(f)$ as $n \to \infty$, provided that the maximum width of the n subintervals approaches zero, and the quadrature rule used on each subinterval has a degree of at least zero. It should be noted that using closed quadrature rules on each subinterval improves efficiency, because the nodes on the endpoints of each subinterval, except for a and b, are shared by two quadrature rules. As a result, fewer function evaluations are necessary, compared to a composite rule that uses open rules with the same number of nodes.

We will now state some of the most well-known composite quadrature rules. In the following discussion, we assume that the interval $[a, b]$ is divided into n subintervals of equal width $h = (b-a)/n$, and that these subintervals have endpoints $[x_{i-1}, x_i]$, where $x_i = a+ih$, for $i = 0, 1, 2, \ldots, n$. Given such a partition of $[a, b]$, we can compute $I(f)$ using the *Composite Midpoint Rule*

$$\int_a^b f(x)\,dx \approx 2h[f(x_1) + f(x_3) + \cdots + f(x_{n-1})], \quad n \text{ is even},$$

the *Composite Trapezoidal Rule*

$$\int_a^b f(x)\,dx \approx \frac{h}{2}[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)],$$

or the *Composite Simpson's Rule*

$$\int_a^b f(x)\,dx \approx \frac{h}{3}[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+\cdots+2f(x_{n-2})+4f(x_{n-1})+f(x_n)],$$

for which n is required to be even, as in the Composite Midpoint Rule.

To obtain the error in each of these composite rules, we can sum the errors in the corresponding basic rules over the n subintervals. For the Composite Trapezoidal Rule, this yields

$$\begin{aligned} E_{trap} &= \int_a^b f(x)\,dx - \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right] \\ &= -\sum_{i=1}^{n} \frac{f''(\eta_i)}{12}(x_i - x_{i-1})^3 \\ &= -\frac{h^3}{12}\sum_{i=1}^{n} f''(\eta_i) \\ &= -\frac{h^3}{12} n f''(\eta) \\ &= -\frac{f''(\eta)}{12}(b-a)h^2, \end{aligned}$$

where, for $i = 1, \ldots, n$, η_i belongs to $[x_{i-1}, x_i]$, and $a \leq \eta \leq b$. The replacement of $\sum_{i=1}^{h} f''(\eta_i)$ by $nf''(\eta)$ is justified by the Intermediate Value Theorem, provided that $f''(x)$ is continuous on $[a, b]$. We see that the Composite Trapezoidal Rule is *second-order accurate*. Furthermore, its *degree of accuracy*, which is the highest degree of polynomial that is guaranteed to be integrated exactly, is the same as for the basic Trapezoidal Rule, which is one.

Similarly, for the Composite Midpoint Rule, we have

$$E_{mid} = \int_a^b f(x)\,dx - 2h\sum_{i=1}^{n/2} f(x_{2i-1}) = \sum_{i=1}^{n/2} \frac{f''(\eta_i)}{24}(2h)^3 = \frac{f''(\eta)}{6}(b-a)h^2.$$

Although it appears that the Composite Midpoint Rule is less accurate than the Composite Trapezoidal Rule, it should be noted that it uses about half as many function evaluations. In other words, the Basic Midpoint Rule is applied $n/2$ times, each on a subinterval of width $2h$. Rewriting the Composite Midpoint Rule in such a way that it uses n function evaluations,

each on a subinterval of width h, we obtain

$$\int_a^b f(x)\,dx = h\sum_{i=1}^{n} f\left(x_{i-1} + \frac{h}{2}\right) + \frac{f''(\eta)}{24}(b-a)h^2,$$

which reveals that the Composite Midpoint Rule is generally more accurate.

Finally, for the Composite Simpson's Rule, we have

$$E_{simp} = -\sum_{i=1}^{n/2} \frac{f^{(4)}(\eta_i)}{90}h^5 = -\frac{f^{(4)}(\eta)}{180}(b-a)h^4,$$

because the Basic Simpson Rule is applied $n/2$ times, each on a subinterval of width $2h$. We conclude that the Simpson's Rule is *fourth-order accurate*.

Example We wish to approximate

$$\int_0^1 e^x\,dx$$

using composite quadrature, to 3 decimal places. That is, the error must be less than 10^{-3}. This requires choosing the number of subintervals, n, sufficiently large so that an upper bound on the error is less than 10^{-3}.

For the Composite Trapezoidal Rule, the error is

$$E_{trap} = -\frac{f''(\eta)}{12}(b-a)h^2 = -\frac{e^\eta}{12n^2},$$

since $f(x) = e^x$, $a = 0$ and $b = 1$, which yields $h = (b-a)/n = 1/n$. Since $0 \leq \eta \leq 1$, and e^x is increasing, the factor e^η is bounded above by $e^1 = e$. It follows that $|E_{trap}| < 10^{-3}$ if

$$\frac{e}{12n^2} < 10^{-3} \quad \Rightarrow \quad \frac{1000e}{12} < n^2 \quad \Rightarrow \quad n > 15.0507.$$

Therefore, the error will be sufficiently small provided that we choose $n \geq 16$.

On the other hand, if we use the Composite Simpson's Rule, the error is

$$E_{simp} = -\frac{f^{(4)}(\eta)}{180}(b-a)h^4 = -\frac{e^\eta}{180n^4}$$

for some η in $[0,1]$, which is less than 10^{-3} in absolute value if

$$n > \left(\frac{1000e}{180}\right)^{1/4} \approx 1.9713,$$

so $n = 2$ is sufficient. That is, we can approximate the integral to 3 decimal places by setting $h = (b - a)/n = (1 - 0)/2 = 1/2$ and computing

$$\int_0^1 e^x \, dx \approx \frac{h}{3}[e^{x_0} + 4e^{x_1} + e^{x_2}] = \frac{1/2}{3}[e^0 + 4e^{1/2} + e^1] \approx 1.71886,$$

whereas the exact value is approximately 1.71828. □

3.6 Richardson Extrapolation

We have seen that the accuracy of methods for computing integrals or derivatives of a function $f(x)$ depends on the spacing between points at which f is evaluated, and that the approximation tends to the exact value as this spacing tends to 0.

Suppose that a uniform spacing h is used. We denote by $F(h)$ the approximation computed using the spacing h, from which it follows that the exact value is given by $F(0)$. Let p be the order of accuracy in our approximation; that is,

$$F(h) = a_0 + a_1 h^p + O(h^r), \quad r > p,$$

where a_0 is the exact value $F(0)$. Then, if we choose a value for h and compute $F(h)$ and $F(h/q)$ for some positive integer q, then we can neglect the $O(h^r)$ terms and solve a system of two equations for the unknowns a_0 and a_1, thus obtaining an approximation that is rth order accurate. If we can describe the error in this approximation in the same way that we can describe the error in our original approximation $F(h)$, we can repeat this process to obtain an approximation that is even more accurate.

This process of extrapolating from $F(h)$ and $F(h/q)$ to approximate $F(0)$ with a higher order of accuracy is called *Richardson extrapolation*. In a sense, Richardson extrapolation is similar in spirit to Aitken's Δ^2 method, as both methods use assumptions about the convergence of a sequence of approximations to "solve" for the exact solution, resulting in a more accurate method of computing approximations.

Example Consider the function

$$f(x) = \frac{\sin^2\left(\frac{\sqrt{x^2+x}}{\cos x - x}\right)}{\sin\left(\frac{\sqrt{x}-1}{\sqrt{x^2+1}}\right)}.$$

Our goal is to compute $f'(0.25)$ as accurately as possible. Using a centered difference approximation,

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2),$$

with $x = 0.25$ and $h = 0.01$, we obtain the approximation

$$f'(0.25) \approx \frac{f(0.26) - f(0.24)}{0.02} = -9.06975297890147,$$

which has absolute error 3.0×10^{-3}, and if we use $h = 0.005$, we obtain the approximation

$$f'(0.25) \approx \frac{f(0.255) - f(0.245)}{0.01} = -9.06746429492149,$$

which has absolute error 7.7×10^{-4}. As expected, the error decreases by a factor of approximately 4 when we halve the step size h, because the error in the centered difference formula is of $O(h^2)$.

We can obtain a more accurate approximation by applying *Richardson Extrapolation* to these approximations. We define the function $N_1(h)$ to be the centered difference approximation to $f'(0.25)$ obtained using the step size h. Then, with $h = 0.01$, we have

$$N_1(h) = -9.06975297890147, \quad N_1(h/2) = -9.06746429492149,$$

and the exact value is given by $N_1(0) = -9.06669877124279$. Because the error in the centered difference approximation satisfies

$$N_1(h) = N_1(0) + K_1 h^2 + K_2 h^4 + K_3 h^6 + O(h^8),$$

where the constants K_1, K_2 and K_3 depend on the derivatives of $f(x)$ at $x = 0.25$, it follows that the new approximation

$$N_2(h) = N_1(h/2) + \frac{N_1(h/2) - N_1(h)}{2^2 - 1} = -9.06670140026149,$$

has fourth-order accuracy. Specifically, if we denote the exact value by $N_2(0)$, we have

$$N_2(h) = N_2(0) + \tilde{K}_2 h^4 + \tilde{K}_3 h^6 + O(h^8),$$

where the constants $\tilde{K}_2$ and $\tilde{K}_3$ are independent of h.

Now, suppose that we compute

$$N_1(h/4) = \frac{f(x+h/4) - f(x-h/4)}{2(h/4)} = \frac{f(0.2525) - f(0.2475)}{0.005} = -9.06689027527046,$$

which has an absolute error of 1.9×10^{-4}, we can use extrapolation again to obtain a second fourth-order accurate approximation,

$$N_2(h/2) = N_1(h/4) + \frac{N_1(h/4) - N_1(h/2)}{3} = -9.06669893538678,$$

which has absolute error of 1.7×10^{-7}. It follows from the form of the error in $N_2(h)$ that we can use extrapolation on $N_2(h)$ and $N_2(h/2)$ to obtain a *sixth*-order accurate approximation,

$$N_3(h) = N_2(h/2) + \frac{N_2(h/2) - N_2(h)}{2^4 - 1} = -9.06669877106180,$$

which has an absolute error of 1.8×10^{-10}. □

3.7 The Euler-Maclaurin Expansion

In the previous example, it was stated, without proof, that the error in the centered difference approximation could be expressed as a sum of terms involving even powers of the spacing h. We would like to use Richardson Extrapolation to enhance the accuracy of approximate integrals computed using the Composite Trapezoidal Rule, but first we must determine the form of the error in these approximations. We have established that the Composite Trapezoidal Rule is second-order accurate, but if Richardson Extrapolation is used once to eliminate the $O(h^2)$ portion of the error, we do not know the order of what remains.

Suppose that $g(t)$ is differentiable on $(-1, 1)$. From integration by parts, we have

$$\int_{-1}^{1} g(t)\,dt = tg(t)|_{-1}^{1} - \int_{-1}^{1} tg'(t)\,dt = [g(-1) + g(1)] - \int_{-1}^{1} tg'(t)\,dt.$$

The first term on the right side of the equals sign is the basic Trapezoidal Rule approximation of the integral on the left side of the equals sign. The second term on the right side is the error in this approximation. If g is $2k$-times differentiabe on $(-1, 1)$, and we repeatedly apply integration by

parts, $2k-1$ times, we obtain

$$\begin{aligned}\int_{-1}^{1} g(t)\,dt - [g(-1)+g(1)] &= \left[q_2(t)g'(t) - q_3(t)g''(t) + \cdots + q_{2k}(t)g^{(2k-1)}(t)\right]\Big|_{-1}^{1} - \\ &\quad \int_{-1}^{1} q_{2k}(t)g^{(2k)}(t)\,dt,\end{aligned}$$

where the sequence of polynomials $q_1(t), \ldots, q_{2k}(t)$ satisfy

$$q_1(t) = -t, \quad q'_{r+1}(t) = q_r(t), \quad r = 1, 2, \ldots, 2k-1.$$

If we choose the constants of integration correctly, then, because $q_1(t)$ is an odd function, we can ensure that $q_r(t)$ is an odd function if r is odd, and an even function if r is even. Furthermore, we can ensure that $q_r(-1) = q_r(1) = 0$ if r is odd. This yields

$$\int_{-1}^{1} g(t)\,dt - [g(-1)+g(1)] = \sum_{r=1}^{k} q_{2r}(1)[g^{(2r-1)}(1) - g^{(2r-1)}(-1)] - \int_{-1}^{1} q_{2k}(t)g^{(2k)}(t)\,dt.$$

Using this expression for the error in the context of the Composite Trapezoidal Rule, applied to the integral of a $2k$-times differentiable function $f(x)$ on a general interval $[a, b]$, yields the *Euler-Maclaurin Expansion*

$$\begin{aligned}\int_a^b f(x)\,dx &= \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{n-1} f(x_i) + f(b)\right] + \\ &\quad \sum_{r=1}^{k} c_r h^{2r}[f^{(2r-1)}(b) - f^{(2r-1)}(a)] - \left(\frac{h}{2}\right)^{2k} \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} q_{2k}(t) f^{(2k)}(x)\,dx,\end{aligned}$$

where, for each $i = 1, 2, \ldots, n$, $t = -1 + \frac{2}{h}(x - x_{i-1})$, and the constants

$$c_r = \frac{q_{2r}(1)}{2^{2r}} = -\frac{B_{2r}}{(2r)!}, \quad r = 1, 2, \ldots, k$$

are closely related to the *Bernoulli numbers* B_r.

It can be seen from this expansion that the error $E_{trap}(h)$ in the Composite Trapezoidal Rule, like the error in the centered difference approximation of the derivative, has the form

$$E_{trap}(h) = K_1 h^2 + K_2 h^4 + K_3 h^6 + \cdots + O(h^{2k}),$$

where the constants K_i are independent of h, provided that the integrand is at least $2k$ times continuously differentiable. This knowledge of the error

provides guidance on how Richardson Extrapolation can be repeatedly applied to approximations obtained using the Composite Trapezoidal Rule at different spacings in order to obtain higher-order accurate approximations.

It can also be seen from the Euler-Maclaurin Expansion that the Composite Trapezoidal Rule is particularly accurate when the integrand is a periodic function, of period $b - a$, as this causes the terms involving the derivatives of the integrand at a and b to vanish. Specifically, if $f(x)$ is periodic with period $b - a$, and is at least $2k$ times continuously differentiable, then the error in the Composite Trapezoidal Rule approximation to $\int_a^b f(x)\,dx$, with spacing h, is $O(h^{2k})$, rather than $O(h^2)$. It follows that if $f(x)$ is infinitely differentiable, such as a finite linear combination of sines or cosines, then the Composite Trapezoidal Rule has an *exponential* order of accuracy, meaning that as $h \to 0$, the error converges to zero more rapidly than *any* power of h.

3.8 Romberg Integration

Richardson extrapolation is not only used to compute more accurate approximations of derivatives, but is also used as the foundation of a numerical integration scheme called *Romberg integration*. In this scheme, the integral

$$I(f) = \int_a^b f(x)\,dx$$

is approximated using the Composite Trapezoidal Rule with step sizes $h_k = (b-a)2^{-k}$, where k is a nonnegative integer. Then, for each k, Richardson extrapolation is used $k-1$ times to previously computed approximations in order to improve the order of accuracy as much as possible.

More precisely, suppose that we compute approximations $T_{1,1}$ and $T_{2,1}$ to the integral, using the Composite Trapezoidal Rule with one and two subintervals, respectively. That is,

$$\begin{aligned} T_{1,1} &= \frac{b-a}{2}[f(a) + f(b)] \\ T_{2,1} &= \frac{b-a}{4}\left[f(a) + 2f\left(\frac{a+b}{2}\right) + f(b)\right]. \end{aligned}$$

Suppose that f has continuous derivatives of all orders on $[a, b]$. Then, the Composite Trapezoidal Rule, for a general number of subintervals n, satisfies

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b)\right] + \sum_{i=1}^{\infty} K_i h^{2i},$$

where $h = (b-a)/n$, $x_j = a + jh$, and the constants $\{K_i\}_{i=1}^{\infty}$ depend only on the derivatives of f. It follows that we can use Richardson Extrapolation to compute an approximation with a higher order of accuracy. If we denote the exact value of the integral by $I(f)$ then we have

$$\begin{aligned} T_{1,1} &= I(f) + K_1 h^2 + O(h^4) \\ T_{2,1} &= I(f) + K_1 (h/2)^2 + O(h^4) \end{aligned}$$

Neglecting the $O(h^4)$ terms, we have a system of equations that we can solve for K_1 and $I(f)$. The value of $I(f)$, which we denote by $T_{2,2}$, is an improved approximation given by

$$T_{2,2} = T_{2,1} + \frac{T_{2,1} - T_{1,1}}{3}.$$

It follows from the representation of the error in the Composite Trapezoidal Rule that $I(f) = T_{2,2} + O(h^4)$.

Suppose that we compute another approximation $T_{3,1}$ using the Composite Trapezoidal Rule with 4 subintervals. Then, as before, we can use Richardson Extrapolation with $T_{2,1}$ and $T_{3,1}$ to obtain a new approximation $T_{3,2}$ that is fourth-order accurate. Now, however, we have two approximations, $T_{2,2}$ and $T_{3,2}$, that satisfy

$$\begin{aligned} T_{2,2} &= I(f) + \tilde{K}_2 h^4 + O(h^6) \\ T_{3,2} &= I(f) + \tilde{K}_2 (h/2)^4 + O(h^6) \end{aligned}$$

for some constant $\tilde{K}_2$. It follows that we can apply Richardson Extrapolation to these approximations to obtain a new approximation $T_{3,3}$ that is *sixth-order* accurate. We can continue this process to obtain as high an order of accuracy as we wish. We now describe the entire algorithm.

Algorithm (Romberg Integration) Given a positive integer J, an interval $[a, b]$ and a function $f(x)$, the following algorithm computes an approximation to $I(f) = \int_a^b f(x)\,dx$ that is accurate to order $2J$.

$h = b - a$
for $j = 1, 2, \ldots, J$ **do**
    $T_{j,1} = \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{2^{j-1}-1} f(a + ih) + f(b)\right]$ (Composite Trapezoidal Rule)
    for $k = 2, 3, \ldots, j$ **do**
        $T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{4^{k-1}-1}$ (Richardson Extrapolation)
    end
    $h = h/2$
end

It should be noted that in a practical implementation, $T_{j,1}$ can be computed more efficiently by using $T_{j-1,1}$, because $T_{j-1,1}$ already includes more than half of the function values used to compute $T_{j,1}$, and they are weighted correctly relative to one another. It follows that for $j > 1$, if we split the summation in the algorithm into two summations containing odd- and even-numbered terms, respectively, we obtain

$$\begin{aligned} T_{j,1} &= \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{2^{j-2}} f(a+(2i-1)h) + 2\sum_{i=1}^{2^{j-2}-1} f(a+2ih) + f(b)\right] \\ &= \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{2^{j-2}-1} f(a+2ih) + f(b)\right] + \frac{h}{2}\left[2\sum_{i=1}^{2^{j-2}} f(a+(2i-1)h)\right] \\ &= \frac{1}{2}T_{j-1,1} + h\sum_{i=1}^{2^{j-2}} f(a+(2i-1)h). \end{aligned}$$

Example We will use *Romberg integration* to obtain a sixth-order accurate approximation to

$$\int_0^1 e^{-x^2}\,dx,$$

an integral that *cannot* be computed using the Fundamental Theorem of Calculus. We begin by using the Trapezoidal Rule, or, equivalently, the Composite Trapezoidal Rule

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left[f(a) + \sum_{j=1}^{n-1} f(x_j) + f(b)\right], \quad h = \frac{b-a}{n}, \quad x_j = a + jh,$$

with $n = 1$ subintervals. Since $h = (b-a)/n = (1-0)/1 = 1$, we have

$$R_{1,1} = \frac{1}{2}[f(0) + f(1)] = 0.68393972058572,$$

which has an absolute error of 6.3×10^{-2}.

If we bisect the interval $[0, 1]$ into two subintervals of equal width, and approximate the area under e^{-x^2} using two trapezoids, then we are applying the Composite Trapezoidal Rule with $n = 2$ and $h = (1-0)/2 = 1/2$, which yields

$$R_{2,1} = \frac{0.5}{2}[f(0) + 2f(0.5) + f(1)] = 0.73137025182856,$$

which has an absolute error of 1.5×10^{-2}. As expected, the error is reduced by a factor of 4 when the step size is halved, since the error in the Composite Trapezoidal Rule is of $O(h^2)$.

Now, we can use Richardson Extrapolation to obtain a more accurate approximation,

$$R_{2,2} = R_{2,1} + \frac{R_{2,1} - R_{1,1}}{3} = 0.74718042890951,$$

which has an absolute error of 3.6×10^{-4}. Because the error in the Composite Trapezoidal Rule satisfies

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f(a) + \sum_{j=1}^{n-1} f(x_j) + f(b)\right] + K_1 h^2 + K_2 h^4 + K_3 h^6 + O(h^8),$$

where the constants K_1, K_2 and K_3 depend on the derivatives of $f(x)$ on $[a, b]$ and are independent of h, we can conclude that $R_{2,1}$ has fourth-order accuracy.

We can obtain a second approximation of fourth-order accuracy by using the Composite Trapezoidal Rule with $n = 4$ to obtain a third approximation of second-order accuracy. We set $h = (1 - 0)/4 = 1/4$, and then compute

$$R_{3,1} = \frac{0.25}{2}\left[f(0) + 2[f(0.25) + f(0.5) + f(0.75)] + f(1)\right] = 0.74298409780038,$$

which has an absolute error of 3.8×10^{-3}. Now, we can apply Richardson Extrapolation to $R_{2,1}$ and $R_{3,1}$ to obtain

$$R_{3,2} = R_{3,1} + \frac{R_{3,1} - R_{2,1}}{3} = 0.74685537979099,$$

which has an absolute error of 3.1×10^{-5}. This significant decrease in error from $R_{2,2}$ is to be expected, since both $R_{2,2}$ and $R_{3,2}$ have fourth-order accuracy, and $R_{3,2}$ is computed using half the step size of $R_{2,2}$.

It follows from the error term in the Composite Trapezoidal Rule, and the formula for Richardson Extrapolation, that

$$R_{2,2} = \int_0^1 e^{-x^2}\,dx + \tilde{K}_2 h^4 + O(h^6), \quad R_{2,2} = \int_0^1 e^{-x^2}\,dx + \tilde{K}_2\left(\frac{h}{2}\right)^4 + O(h^6).$$

Therefore, we can use Richardson Extrapolation with these two approximations to obtain a new approximation

$$R_{3,3} = R_{3,2} + \frac{R_{3,2} - R_{2,2}}{2^4 - 1} = 0.74683370984975,$$

which has an absolute error of 9.6×10^{-6}. Because $R_{3,3}$ is a linear combination of $R_{3,2}$ and $R_{2,2}$ in which the terms of order h^4 cancel, we can conclude that $R_{3,3}$ is of sixth-order accuracy. □

Chapter 4

Polynomial Approximation in the ∞-norm

4.1 Normed Linear Spaces

Previously we have considered the problem of polynomial *interpolation*, in which a function $f(x)$ is approximated by a polynomial $p_n(x)$ that agrees with $f(x)$ at $n+1$ distinct points, based on the assumption that $p_n(x)$ will be, in some sense, a good approximation of $f(x)$ at other points. As we have seen, however, this assumption is not always valid, and in fact, such an approximation can be quite poor, as demonstrated by Runge's example.

Therefore, we consider an alternative approach to approximation of a function $f(x)$ on an interval $[a, b]$ by a polynomial, in which the polynomial is not required to agree with f at any specific points, but rather approximate f well in an "overall" sense, by not deviating much from f at *any* point in $[a, b]$. This requires that we define an appropriate notion of "distance" between functions that is, intuitively, consistent with our understanding of distance between numbers or points in space.

To that end, let $\mathcal{V}$ be a vector space over the field of real numbers $\mathbb{R}$. A *norm* on $\mathcal{V}$ is a function $\|\cdot\| : \mathcal{V} \to \mathbb{R}$ that has the following properties:

1. $\|f\| \geq 0$ for all $f \in \mathcal{V}$, and $\|f\| = 0$ if and only if f is the zero vector of $\mathcal{V}$.

2. $\|cf\| = |c|\|f\|$ for any vector $f \in \mathcal{V}$ and any scalar $c \in \mathbb{R}$.

3. $\|f + g\| \leq \|f\| + \|g\|$ for all $f, g \in \mathcal{V}$.

The last property is known as the *triangle inequality*. A vector space $\mathcal{V}$, together with a norm $\|\cdot\|$, is called a *normed vector space* or *normed linear space*.

Example The space $C[a,b]$ of functions that are continuous on the interval $[a,b]$ is a normed vector space with the norm

$$\|f\|_\infty = \max_{a\le x\le b} |f(x)|,$$

known as the ∞-*norm* or *maximum norm*. □

Example The space $C[a,b]$ can be equipped with a different norm, such as

$$\|f\|_2 = \left(\int_a^b |f(x)|^2 w(x)\,dx\right)^{1/2},$$

where the *weight function* $w(x)$ is positive and integrable on (a,b). It is allowed to be singular at the endpoints, as will be seen in certain examples. This norm is called the *2-norm* or *weighted 2-norm*. □

The 2-norm and ∞-norm are related as follows:

$$\|f\|_2 \le W\|f\|\infty, \quad W = \|1\|_2.$$

However, unlike the ∞-norm and 2-norm defined for the vector space $\mathbb{R}^n$, these norms are not *equivalent* in the sense that a function that has a small 2-norm necessarily has a small ∞-norm. In fact, given any $\epsilon > 0$, no matter how small, and any $M > 0$, no matter how large, there exists a function $f \in C[a,b]$ such that

$$\|f\|_2 < \epsilon, \quad \|f\|_\infty > M.$$

4.2 Best Approximation in the ∞-norm

We now consider the problem of approximating a function $f \in C[a,b]$ by a polynomial p such that $\|f-p\|_\infty$ is small. Such an approximation does exist; in fact, for any $\epsilon > 0$, no matter how small, there exists a polynomial p such that

$$\|f-p\|_\infty \le \epsilon.$$

This classical result can be proved by considering the interval $[0,1]$ and using the approximation

$$p_n(x) = \sum_{k=0}^{n} p_{nk}(x) f(k/n), \quad x \in [0,1],$$

where the polynomials

$$p_{nk}(x) = \left(\begin{array}{c} n \\ k \end{array} \right) x^k (1-x)^{n-k}$$

are known as the *Bernstein polynomials*. It can be shown that for any given error tolerance ϵ, there exists a degree n, dependent on ϵ, such that $\|f - p_n\|_\infty \leq \epsilon$.

We now fix the degree n and consider the problem of approximating $f \in C[a,b]$ by a polynomial $p_n \in \mathcal{P}_n$, where $\mathcal{P}_n$ is the space of polynomials on $[a,b]$ of degree at most n, such that $\|f - p_n\|_\infty$ is minimized. In fact, it can be shown that $\|f - p_n\|_\infty$, as a function of the $n+1$ coefficients of p_n, does have a minimum on $\mathbb{R}^{n+1}$ that can be attained, as it is a continuous function of the coefficients, and there exists a compact, nonempty subset of $\mathbb{R}^{n+1}$ such that $\|f - p_n\| \leq \|f\|_\infty + 1$ for any polynomial p_n whose coefficients lie in this subset. Therefore, this subset must contain a minimum.

A polynomial $p_n \in \mathcal{P}_n$ that minimizes $\|f - p_n\|_\infty$, the maximum absolute value of $f(x) - p_n(x)$ on $[a,b]$, is called the *minimax polynomial*. It is, in the ∞-norm sense, the best approximation of f on $[a,b]$ by a polynomial of degree n. This polynomial, as we will see later, is in fact unique.

Example Let $f \in C[a,b]$. Then f has a minimum at a point $\xi \in [a,b]$, and a maximum at $\eta \in [a,b]$. Then the minimax polynomial of f of degree 0 is the constant function

$$p_0(x) = \frac{1}{2}[f(\xi) + f(\eta)].$$

□

We note that the error $|f(x) - p_0(x)|$ is maximized at two points, at $x = \xi$ and $x = \eta$. We also note that at $x = \xi$,

$$f(x) - p_0(x) = f(\xi) - \frac{1}{2}[f(\xi) + f(\eta)] = \frac{1}{2}[f(\xi) - f(\eta)] < 0,$$

while at $x = \eta$,

$$f(x) - p_0(x) = f(\eta) - \frac{1}{2}[f(\xi) + f(\eta)] = \frac{1}{2}[f(\eta) - f(\xi)] > 0.$$

Not only are the errors at these points of opposite sign; they are also equal in magnitude to the error on the entire interval, $\|f - p_0\|_\infty$.

A similar result holds for higher-degree approximations. The *Oscillation Theorem* states that if $p_n \in \mathcal{P}_n$ is the minimax polynomial of degree n for

$f \in C[a,b]$, then there exist $n+2$ points $a \leq x_0 < x_1 < \cdots < x_{n+1} \leq b$ such that

$$|f(x_i) - p_n(x_i)| = \|f - p_n\|_\infty, \quad i = 0, 1, \ldots, n+1,$$

and

$$f(x_i) - p_n(x_i) = -(f(x_{i+1}) - p_n(x_{i+1})), \quad i = 0, 1, \ldots, n.$$

These points are called the *critical points* of f on $[a,b]$.

This result can be used to prove the uniqueness of the minimax polynomial. It can also be used to compute it. It follows from the Oscillation Theorem that if $f \in C[a,b]$ is continuously differentiable on (a,b), and if f' is monotonic on (a,b), then the minimax polynomial of degree one, $p_1(x) = c_0 + c_1 x$, can be obtained by first noting that because f' does not change sign on $[a,b]$, $f - p_1$ must assume its maximum and minimum values at $x = a$, $x = b$, and $x = d$, where $d \in (a,b)$.

Because p_1 is a minimax polynomial, it follows that

$$f(a) - (c_0 + c_1 a) = A,$$

$$f(d) - (c_0 + c_1 d) = -A,$$

$$f(b) - (c_0 + c_1 b) = A.$$

It follows from the first and third equations that

$$c_1 = \frac{f(b) - f(a)}{b - a}.$$

That is, the graph of $p_1(x)$ is parallel to the secant line passing through $(a, f(a))$ and $(b, f(b))$.

From the first and second equations, we obtain

$$c_0 = \frac{1}{2}[f(a) + f(d) - c_1(a + d)],$$

where d is the point at which the slope of the tangent line is equal to c_1, the existence of which is guaranteed by the Mean Value Theorem. This point is also unique, because f' is assumed to be monotonic. We conclude that p_1 is the linear function whose graph is a line that is parallel to the secant line and the tangent line at d, both of which have slope c_1, and is halfway between these lines.

4.3 Chebyshev Polynomials

Previously we have learned how to compute minimax polynomials for certain special cases, but in general this is quite difficult. One minimax problem that can be solved is the problem of computing the minimax polynomial of a function $f(x)$ that is itself a polynomial, where the approximation must have lower degree. Another related problem is that of finding a polynomial approximation that is at least "near" the minimax polynomial in some sense, when it is not practical to compute the minimax polynomial itself.

A class of polynomials that is helpful for these problems is the sequence of *Chebyshev polynomials*, defined by

$$T_k(x) = \cos(k \cos^{-1} x), \quad -1 \leq x \leq 1.$$

From this definition, we obtain

$$T_0(x) = 1, \quad T_1(x) = x.$$

Additional polynomials can be obtained using the trigonometric identities

$$\cos((k+1)\theta) = \cos k\theta \cos\theta - \sin k\theta \sin\theta,$$

$$\cos((k-1)\theta) = \cos k\theta \cos\theta + \sin k\theta \sin\theta.$$

Adding these identities, and setting $\theta = \cos^{-1} x$, yields

$$T_{k+1}(x) + T_{k-1}(x) = 2xT_k(x),$$

from which we obtain a *three-term recurrence relation*

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x).$$

It can easily be seen from this relation, and the first two Chebyshev polynomials, that $T_k(x)$ is in fact a polynomial for all integers $k \geq 0$.

The Chebyshev polynomials have the following properties of interest:

1. The leading coefficient of $T_k(x)$ is 2^{n-1}.
2. $T_k(x)$ is an even function of k is even, and an odd function if k is odd.
3. The zeros of $T_k(x)$, for $k \geq 1$, are

$$x_j = \cos\frac{(2j-1)\pi}{2k}, \quad j = 1, 2, \ldots, k.$$

4. The extrema of $T_k(x)$ on $[-1,1]$ are

$$\tilde{x}_j = \cos\frac{j\pi}{k}, \quad j = 0, 1, \ldots, k,$$

and the corresponding extremal values are ± 1.

5. $|T_k(x)| \leq 1$ on $[-1,1]$ for all $k \geq 0$.

Now, consider the problem of finding the minimax polynomial of degree n for $f(x) = x^{n+1}$ on $[-1,1]$. If we define

$$p_n(x) = x^{n+1} - 2^{-n}T_{n+1}(x),$$

then, because the leading coefficient of $T_{n+1}(x)$ is 2^n, $p_n(x)$ is a polynomial of degere n. Furthermore, because

$$x^{n+1} - p_n(x) = 2^{-n}T_{n+1}(x),$$

it follows that $x^{n+1} - p_n(x)$ attains its maxima and minima at the $n+2$ points that are the extrema of $T_{n+1}(x)$ on $[-1,1]$, and the corresponding extreme values alternate in sign. Therefore, by the Oscillation Theorem, $p_n(x)$ is the nth-degree minimax polynomial for $f(x)$ on $[-1,1]$.

A consequence of this result is that if we denote by $\mathcal{P}_n^1$ the set of all *monic* polynomials of degree n (that is, all polynomials of degree n with leading coefficient 1), then

$$\min_{r\in\mathcal{P}_{n+1}^1} \|r\|_\infty = \min_{q\in\mathcal{P}_n} \|x^{n+1} - q\| = \|x^{n+1} - (x^{n+1} - 2^{-n}T_{n+1}(x))\| = \|2^{-n}T_{n+1}(x)\|.$$

That is, $2^{-n}T_{n+1}(x)$ is the monic polynomial of degree $n+1$ with smallest ∞-norm on $[-1,1]$.

4.4 Interpolation

Let $f(x)$ be a function that is $(n+1)$ times continuously differentiable on $[a,b]$. If we approximate $f(x)$ by a nth-degree polynomial $p_n(x)$ that interpolates $f(x)$ at the $n+1$ roots of the Chebyshev polynomial $T_{n+1}(x)$, mapped from $[-1,1]$ to $[a,b]$,

$$\xi_j = \frac{1}{2}(b-a)\cos\frac{(2j+1)\pi}{2n+2} + \frac{1}{2}(a+b),$$

then the error in this approximation is

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \left(\frac{b-a}{2}\right)^{n+1} 2^{-n} T_{n+1}(t(x)),$$

where

$$t(x) = \frac{2x - a - b}{b - a}$$

is the linear map from $[a, b]$ to $[-1, 1]$. This is because

$$\prod_{j=0}^{n}(x - \xi_j) = \left(\frac{b-a}{2}\right)^{n+1} \prod_{j=0}^{n}(t(x) - \tau_j) = \left(\frac{b-a}{2}\right)^{n+1} 2^{-n} T_{n+1}(t(x)),$$

where τ_j is the jth root of $T_{n+1}(t)$. We conclude that

$$|f(x) - p_n(x)| \leq \frac{(b-a)^{n+1}}{2^{2n+1}(n+1)!} \max_{\xi \in [a,b]} |f^{(n+1)}(\xi)|.$$

Because the interpolation error exhibits $(n+1)$ sign changes on $[a, b]$, the Chebyshev interpolant is called a *near-minimax* polynomial. In other words, it is an approximating polynomial for which the error, in the ∞-norm sense, is not minimized, but is still small. Furthermore, the Chebyshev interpolant is readily computed, and as such it is often used as a viable alternative to the minimax polynomial.

Chapter 5

Polynomial Approximation in the 2-norm

5.1 Best Approximation in the 2-norm

Suppose that we wish to obtain a function $f_n(x)$ that is a linear combination of given functions $\{\phi_j(x)\}_{j=0}^n$, and best fits a function $f(x)$ at a *discrete* set of data points $\{(x_i, f(x_i))\}_{i=1}^m$ in a least-squares sense. That is, we wish to find constants $\{c_j\}_{j=0}^n$ such that

$$\sum_{i=1}^{m} [f_n(x_i) - f(x_i)]^2 = \sum_{i=1}^{m} \left[\sum_{j=0}^{n} c_j \phi_j(x_i) - f(x_i) \right]^2$$

is minimized. This can be accomplished by solving a system of $n+1$ linear equations for the $\{c_j\}$, known as the *normal equations*.

Now, suppose we have a *continuous* set of data. That is, we have a function $f(x)$ defined on an interval $[a, b]$, and we wish to approximate it as closely as possible, in some sense, by a function $f_n(x)$ that is a linear combination of given functions $\{\phi_j(x)\}_{j=0}^n$. If we choose m equally spaced points $\{x_i\}_{i=1}^m$ in $[a, b]$, and let $m \to \infty$, we obtain the *continuous least-squares problem* of finding the function

$$f_n(x) = \sum_{j=0}^{n} c_j \phi_j(x)$$

that minimizes

$$E(c_0, c_1, \ldots, c_n) = \int_a^b [f_n(x) - f(x)]^2 \, dx = \int_a^b \left[\sum_{j=0}^{n} c_j \phi_j(x) - f(x) \right]^2 dx.$$

To obtain the coefficients $\{c_j\}_{j=0}^n$, we can proceed as in the discrete case. We compute the partial derivatives of $E(c_0, c_1, \ldots, c_n)$ with respect to each c_k and obtain

$$\frac{\partial E}{\partial c_k} = \int_a^b \phi_k(x) \left[\sum_{j=0}^{n} c_j \phi_j(x) - f(x) \right] dx,$$

and requiring that each partial derivative be equal to zero yields the *normal equations*

$$\sum_{j=0}^{n} \left[\int_a^b \phi_k(x) \phi_j(x) \, dx \right] c_j = \int_a^b \phi_k(x) f(x) \, dx, \quad k = 0, 1, \ldots, n.$$

We can then solve this system of equations to obtain the coefficients $\{c_j\}_{j=0}^n$. This system can be solved as long as the functions $\{\phi_j(x)\}_{j=0}^n$ are *linearly independent*. That is, the condition

$$\sum_{j=0}^{n} c_j \phi_j(x) \equiv 0, \quad x \in [a, b],$$

is only true if $c_0 = c_1 = \cdots = c_n = 0$. In particular, this is the case if, for $j = 0, 1, \ldots, n$, $\phi_j(x)$ is a polynomial of degree j. This can be proved using a simple inductive argument.

Example We approximate $f(x) = e^x$ on the interval $[0, 5]$ by a fourth-degree polynomial

$$f_4(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4.$$

The normal equations have the form

$$\sum_{j=0}^{n} a_{ij} c_j = b_i, \quad i = 0, 1, \ldots, 4,$$

or, in matrix-vector form, $A\mathbf{c} = \mathbf{b}$, where

$$a_{ij} = \int_0^5 x^i x^j \, dx = \int_0^5 x^{i+j} \, dx = \frac{5^{i+j+1}}{i+j+1}, \quad i, j = 0, 1, \ldots, 4,$$

$$b_i = \int_0^5 x^i e^x \, dx, \quad i = 0, 1, \ldots, 4.$$

Integration by parts yields the relation

$$b_i = 5^i e^5 - i b_{i-1}, \quad b_0 = e^5 - 1.$$

Solving this system of equations yields the polynomial

$$f_4(x) = 2.3002 - 6.226x + 9.5487x^2 - 3.86x^3 + 0.6704x^4.$$

As Figure 5.1 shows, this polynomial is barely distinguishable from e^x on $[0, 5]$.

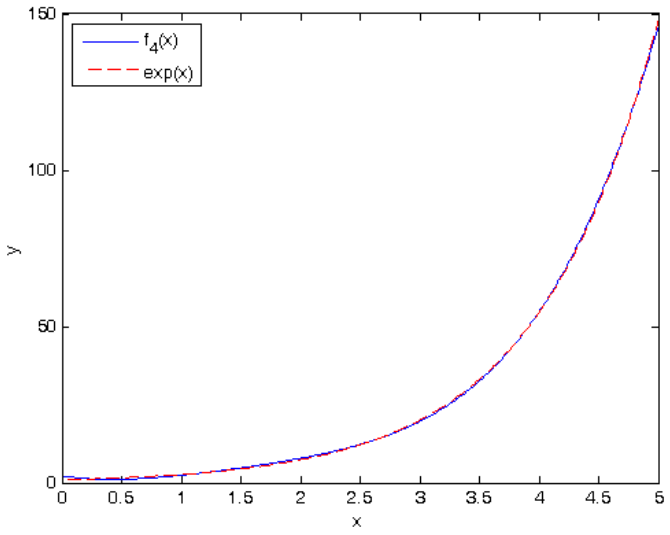

Figure 5.1: Graphs of $f(x) = e^x$ (red dashed curve) and 4th-degree continuous least-squares polynomial approximation $f_4(x)$ on $[0, 5]$ (blue solid curve)

However, it should be noted that the matrix A is closely related to the $n \times n$ *Hilbert matrix* H_n, which has entries

$$[H_n]_{ij} = \frac{1}{i + j - 1}, \quad 1 \leq i, j \leq n.$$

This matrix is famous for being highly *ill-conditioned*, meaning that solutions to systems of linear equations involving this matrix that are computed

using floating-point arithmetic are highly sensitive to roundoff error. In fact, the matrix A in this example has a *condition number* of 1.56×10^7, which means that a change of size ϵ in the right-hand side vector $\mathbf{b}$, with entries b_i, can cause a change of size $1.56\epsilon \times 10^7$ in the solution $\mathbf{c}$. $\square$

5.2 Inner Product Spaces

As the preceding example shows, it is important to choose the functions $\{\phi_j(x)\}_{j=0}^n$ wisely, so that the resulting system of normal equations is not unduly sensitive to round-off errors. An even better choice is one for which this system can be solved analytically, with relatively few computations. An ideal choice of functions is one for which the task of computing $f_{n+1}(x)$ can reuse the computations needed to compute $f_n(x)$.

To that end, recall that two m-vectors $\mathbf{u} = \langle u_1, u_2, \ldots, u_m \rangle$ and $\mathbf{v} = \langle v_1, v_2, \ldots, v_m \rangle$ are *orthogonal* if

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{m} u_i v_i = 0,$$

where $\mathbf{u} \cdot \mathbf{v}$ is the *dot product*, or *inner product*, of $\mathbf{u}$ and $\mathbf{v}$.

By viewing functions defined on an interval $[a, b]$ as infinitely long vectors, we can generalize the inner product, and the concept of orthogonality, to functions. To that end, we define the *inner product* of two real-valued functions $f(x)$ and $g(x)$ defined on the interval $[a, b]$ by

$$\langle f, g \rangle = \int_a^b f(x)g(x)\,dx.$$

Then, we say f and g are *orthogonal* with respect to this inner product if $\langle f, g \rangle = 0$.

In general, an inner product on a vector space $\mathcal{V}$ over $\mathbb{R}$, be it continuous or discrete, has the following properties:

1. $\langle f + g, h \rangle = \langle f, h \rangle + \langle g, h \rangle$ for all $f, g, h \in \mathcal{V}$
2. $\langle cf, g \rangle = c\langle f, g \rangle$ for all $c \in \mathbb{R}$ and all $f \in \mathcal{V}$
3. $\langle f, g \rangle = \langle g, f \rangle$ for all $f, g \in \mathcal{V}$
4. $\langle f, f \rangle \geq 0$ for all $f \in \mathcal{V}$, and $\langle f, f \rangle = 0$ if and only if $f = \mathbf{0}$.

This inner product can be used to define the *norm* of a function, which generalizes the concept of the magnitude of a vector to functions, and therefore provides a measure of the "magnitude" of a function. Recall that the magnitude of a vector $\mathbf{v}$, denoted by $\|\mathbf{v}\|$, can be defined by

$$\|\mathbf{v}\| = (\mathbf{v} \cdot \mathbf{v})^{1/2}.$$

Along similar lines, we define the *2-norm* of a function $f(x)$ defined on $[a, b]$ by

$$\|f\|_2 = (\langle f, f \rangle)^{1/2} = \left(\int_a^b [f(x)]^2 \, dx \right)^{1/2}.$$

As we will see, it can be verified that this function does in fact satisfy the properties required of a norm. The continuous least-squares problem can then be described as the problem of finding

$$f_n(x) = \sum_{j=0}^{n} c_j \phi_j(x)$$

such that

$$\|f_n - f\|_2 = \left(\int_a^b [f_n(x) - f(x)]^2 \, dx \right)^{1/2}$$

is minimized. This minimization can be performed over $C[a, b]$, the space of functions that are continuous on $[a, b]$, but it is not necessary for a function $f(x)$ to be continuous for $\|f\|_2$ to be defined. Rather, we consider the space $L^2(a, b)$, the space of real-valued functions such that $|f(x)|^2$ is *integrable* over (a, b).

One very important property that $\|\cdot\|_2$ has is that it satisfies the *Cauchy-Schwarz inequality*

$$|\langle f, g \rangle| \leq \|f\|_2 \|g\|_2, \quad f, g \in \mathcal{V}.$$

This can be proven by noting that for any scalar $c \in \mathbb{R}$,

$$c^2 \|f\|_2^2 + 2c\langle f, g \rangle + \|g\|_2^2 = \|cf + g\|_2^2 \geq 0.$$

The left side is a quadratic polynomial in c. In order for this polynomial to not have any negative values, it must either have complex roots or a double real root. This is the case if the discrimant satisfies

$$4\langle f, g \rangle^2 - 4\|f\|_2^2 \|g\|_2^2 \leq 0,$$

from which the Cauchy-Schwarz inequality immediately follows. By setting $c = 1$ and applying this inequality, we immediately obtain the triangle-inequality property of norms.

Suppose that we can construct a set of functions $\{\phi_j(x)\}_{j=0}^n$ that is *orthogonal* with respect to the inner product of functions on $[a, b]$. That is,

$$\langle \phi_k, \phi_j \rangle = \int_a^b \phi_k(x)\phi_j(x)\,dx = \begin{cases} 0 & k \neq j \\ \alpha_k > 0 & k = j \end{cases}.$$

Then, the normal equations simplify to a trivial system

$$\left[\int_a^b [\phi_k(x)]^2\,dx\right] c_k = \int_a^b \phi_k(x) f(x)\,dx, \quad k = 0, 1, \ldots, n,$$

or, in terms of norms and inner products,

$$\|\phi_k\|_2^2 c_k = \langle \phi_k, f \rangle, \quad k = 0, 1, \ldots, n.$$

It follows that the coefficients $\{c_j\}_{j=0}^n$ of the least-squares approximation $f_n(x)$ are simply

$$c_k = \frac{\langle \phi_k, f \rangle}{\|\phi_k\|_2^2}, \quad k = 0, 1, \ldots, n.$$

If the constants $\{\alpha_k\}_{k=0}^n$ above satisfy $\alpha_k = 1$ for $k = 0, 1, \ldots, n$, then we say that the orthogonal set of functions $\{\phi_j(x)\}_{j=0}^n$ is *orthonormal*. In that case, the solution to the continuous least-squares problem is simply given by

$$c_k = \langle \phi_k, f \rangle, \quad k = 0, 1, \ldots, n.$$

Next, we will learn how sets of orthogonal polynomials can be computed.

5.3 Orthogonal Polynomials

Previously, we learned that the problem of finding the polynomial $f_n(x)$, of degree n, that best approximates a function $f(x)$ on an interval $[a, b]$ in the least squares sense, i.e., that minimizes

$$\|f_n - f\|_2 = \left(\int_a^b [f_n(x) - f(x)]^2\,dx\right)^{1/2},$$

is easy to solve if we represent $f_n(x)$ as a linear combination of *orthogonal polynomials*,

$$f_n(x) = \sum_{j=0}^n c_j p_j(x).$$

Each polynomial $p_j(x)$ is of degree j, and the set of polynomials $p_0(x), p_1(x), \ldots, p_n(x)$ are orthogonal with respect to the *inner product*

$$\langle f, g \rangle = \int_a^b f(x)g(x)\,dx.$$

That is,

$$\langle p_k, p_j \rangle = \int_a^b p_k(x)p_j(x)\,dx = 0, \quad k \neq j.$$

Given this sequence of orthogonal polynomials, the coefficients c_j in the linear combination used to compute $f_n(x)$ are given by

$$c_j = \frac{\langle p_j, f \rangle}{\langle p_j, p_j \rangle}, \quad c_j = 0, 1, \ldots, n.$$

Now, we focus on the task of finding such a sequence of orthogonal polynomials.

Recall the process known as *Gram-Schmidt orthogonalization* for obtaining a set of orthogonal vectors $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$ from a set of linearly independent vectors $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$:

$$\begin{aligned}
\mathbf{p}_1 &= \mathbf{a}_1 \\
\mathbf{p}_2 &= \mathbf{a}_2 - \frac{\mathbf{p}_1 \cdot \mathbf{a}_2}{\mathbf{p}_1 \cdot \mathbf{p}_1}\mathbf{p}_1 \\
&\vdots \\
\mathbf{p}_n &= \mathbf{a}_n - \sum_{j=0}^{n-1} \frac{\mathbf{p}_j \cdot \mathbf{a}_n}{\mathbf{p}_j \cdot \mathbf{p}_j}\mathbf{p}_j.
\end{aligned}$$

By *normalizing* each vector $\mathbf{p}_j$, we obtain a unit vector

$$\mathbf{q}_j = \frac{1}{|\mathbf{p}_j|}\mathbf{p}_j,$$

and a set of *orthonormal* vectors $\{\mathbf{q}_j\}_{j=1}^n$, in that they are orthogonal ($\mathbf{q}_k \cdot \mathbf{q}_j = 0$ for $k \neq j$), and unit vectors ($\mathbf{q}_j \cdot \mathbf{q}_j = 1$).

We can use a similar process to compute a set of orthogonal polynomials. For simplicitly, we will require that all polynomials in the set be *monic*; that is, their leading (highest-degree) coefficient must be equal 1. We then define $p_0(x) = 1$. Then, because $p_1(x)$ is supposed to be of degree 1, it must have the form $p_1(x) = x - \alpha_1$ for some constant α_1. To ensure that $p_1(x)$ is orthogonal to $p_0(x)$, we compute their inner product, and obtain

$$0 = \langle p_0, p_1 \rangle = \langle 1, x - \alpha_1 \rangle,$$

so we must have

$$\alpha_1 = \frac{\langle 1, x\rangle}{\langle 1, 1\rangle}.$$

For $j > 1$, we start by setting $p_j(x) = xp_{j-1}(x)$, since p_j should be of degree one greater than that of p_{j-1}, and this satisfies the requirement that p_j be monic. Then, we need to subtract polynomials of lower degree to ensure that p_j is orthogonal to p_i, for $i < j$. To that end, we apply Gram-Schmidt orthogonalization and obtain

$$p_j(x) = xp_{j-1}(x) - \sum_{i=0}^{j-1} \frac{\langle p_i, xp_{j-1}\rangle}{\langle p_i, p_i\rangle} p_i(x).$$

However, by the definition of the inner product, $\langle p_i, xp_{j-1}\rangle = \langle xp_i, p_{j-1}\rangle$. Furthermore, because xp_i is of degree $i+1$, and p_{j-1} is orthogonal to *all* polynomials of degree less than j, it follows that $\langle p_i, xp_{j-1}\rangle = 0$ whenever $i < j-1$.

We have shown that sequences of orthogonal polynomials satisfy a *three-term recurrence relation*

$$p_j(x) = (x - \alpha_j)p_{j-1}(x) - \beta_{j-1}^2 p_{j-2}(x), \quad j > 1,$$

where the *recursion coefficients* α_j and β_{j-1}^2 are defined to be

$$\alpha_j = \frac{\langle p_{j-1}, xp_{j-1}\rangle}{\langle p_{j-1}, p_{j-1}\rangle}, \quad j > 1,$$

$$\beta_j^2 = \frac{\langle p_{j-1}, xp_j\rangle}{\langle p_{j-1}, p_{j-1}\rangle} = \frac{\langle xp_{j-1}, p_j\rangle}{\langle p_{j-1}, p_{j-1}\rangle} = \frac{\langle p_j, p_j\rangle}{\langle p_{j-1}, p_{j-1}\rangle} = \frac{\|p_j\|_2^2}{\|p_{j-1}\|_2^2}, \quad j \geq 1.$$

Note that $\langle xp_{j-1}, p_j\rangle = \langle p_j, p_j\rangle$ because xp_{j-1} differs from p_j by a polynomial of degree at most $j-1$, which is orthogonal to p_j. The recurrence relation is also valid for $j = 1$, provided that we define $p_{j-1}(x) \equiv 0$, and α_1 is defined as above. That is,

$$p_1(x) = (x - \alpha_1)p_0(x), \quad \alpha_1 = \frac{\langle p_0, xp_0\rangle}{\langle p_0, p_0\rangle}.$$

If we also define the recursion coefficient β_0 by

$$\beta_0^2 = \langle p_0, p_0\rangle,$$

and then define

$$q_j(x) = \frac{p_j(x)}{\beta_0\beta_1\cdots\beta_j},$$

then the polynomials $q_0, q_1, \ldots, q_n$ are also orthogonal, and

$$\langle q_j, q_j \rangle = \frac{\langle p_j, p_j \rangle}{\beta_0^2 \beta_1^2 \cdots \beta_j^2} = \langle p_j, p_j \rangle \frac{\langle p_{j-1}, p_{j-1} \rangle}{\langle p_j, p_j \rangle} \cdots \frac{\langle p_0, p_0 \rangle}{\langle p_1, p_1 \rangle} \frac{1}{\langle p_0, p_0 \rangle} = 1.$$

That is, these polynomials are *orthonormal*.

If we consider the inner product

$$\langle f, g \rangle = \int_{-1}^{1} f(x) g(x)\, dx,$$

then a sequence of orthogonal polynomials, with respect to this inner product, can be defined as follows:

$$\begin{aligned} L_0(x) &= 1, \\ L_1(x) &= x, \\ L_{j+1}(x) &= \frac{2j+1}{j+1} x L_j(x) - \frac{j}{j+1} L_{j-1}(x), \quad j = 1, 2, \ldots \end{aligned}$$

These are known as the *Legendre polynomials*. One of their most important applications is in the construction of Gaussian quadrature rules. Specifically, the roots of $L_n(x)$, for $n \geq 1$, are the nodes of a Gaussian quadrature rule for the interval $[-1, 1]$. However, they can also be used to easily compute continuous least-squares polynomial approximations, as the following example shows.

Example We will use Legendre polynomials to approximate $f(x) = \cos x$ on $[-\pi/2, \pi/2]$ by a quadratic polynomial. First, we note that the first three Legendre polynomials, which are the ones of degree 0, 1 and 2, are

$$L_0(x) = 1, \quad L_1(x) = x, \quad L_2(x) = \frac{1}{2}(3x^2 - 1).$$

However, it is not practical to use these polynomials directly to approximate $f(x)$, because they are orthogonal with respect to the inner product defined on the interval $[-1, 1]$, and we wish to approximate $f(x)$ on $[-\pi/2, \pi/2]$.

To obtain orthogonal polynomials on $[-\pi/2, \pi/2]$, we replace x by $2t/\pi$, where t belongs to $[-\pi/2, \pi/2]$, in the Legendre polynomials, which yields

$$\tilde{L}_0(t) = 1, \quad \tilde{L}_1(t) = \frac{2t}{\pi}, \quad \tilde{L}_2(t) = \frac{1}{2}\left(\frac{12}{\pi^2} t^2 - 1\right).$$

Then, we can express our quadratic approximation $f_2(x)$ of $f(x)$ by the linear combination

$$f_2(x) = c_0 \tilde{L}_0(x) + c_1 \tilde{L}_1(x) + c_2 \tilde{L}_2(x),$$

where

$$c_j = \frac{\langle f, \tilde{L}_j \rangle}{\langle \tilde{L}_j, \tilde{L}_j \rangle}, \quad j = 0, 1, 2.$$

Computing these inner products yields

$$\begin{aligned}
\langle f, \tilde{L}_0 \rangle &= \int_{-\pi/2}^{\pi/2} \cos t \, dt \\
&= 2, \\
\langle f, \tilde{L}_1 \rangle &= \int_{-\pi/2}^{\pi/2} \frac{2t}{\pi} \cos t \, dt \\
&= 0, \\
\langle f, \tilde{L}_2 \rangle &= \int_{-\pi/2}^{\pi/2} \frac{1}{2} \left(\frac{12}{\pi^2} t^2 - 1 \right) \cos t \, dt \\
&= \frac{2}{\pi^2}(\pi^2 - 12), \\
\langle \tilde{L}_0, \tilde{L}_0 \rangle &= \int_{-\pi/2}^{\pi/2} 1 \, dt \\
&= \pi, \\
\langle \tilde{L}_1, \tilde{L}_1 \rangle &= \int_{-\pi/2}^{\pi/2} \left(\frac{2t}{\pi} \right)^2 dt \\
&= \frac{8\pi}{3}, \\
\langle \tilde{L}_2, \tilde{L}_2 \rangle &= \int_{-\pi/2}^{\pi/2} \left[\frac{1}{2} \left(\frac{12}{\pi^2} t^2 - 1 \right) \right]^2 dt \\
&= \frac{\pi}{5}.
\end{aligned}$$

It follows that

$$c_0 = \frac{2}{\pi}, \quad c_1 = 0, \quad c_2 = \frac{2}{\pi^2}\frac{5}{\pi}(\pi^2 - 12) = \frac{10}{\pi^3}(\pi^2 - 12),$$

and therefore

$$f_2(x) = \frac{2}{\pi} + \frac{5}{\pi^3}(\pi^2 - 12) \left(\frac{12}{\pi^2} x^2 - 1 \right) \approx 0.98016 - 0.4177 x^2.$$

This approximation is shown in Figure 5.2. □

It is possible to compute sequences of orthogonal polynomials with respect to other inner products. A generalization of the inner product that

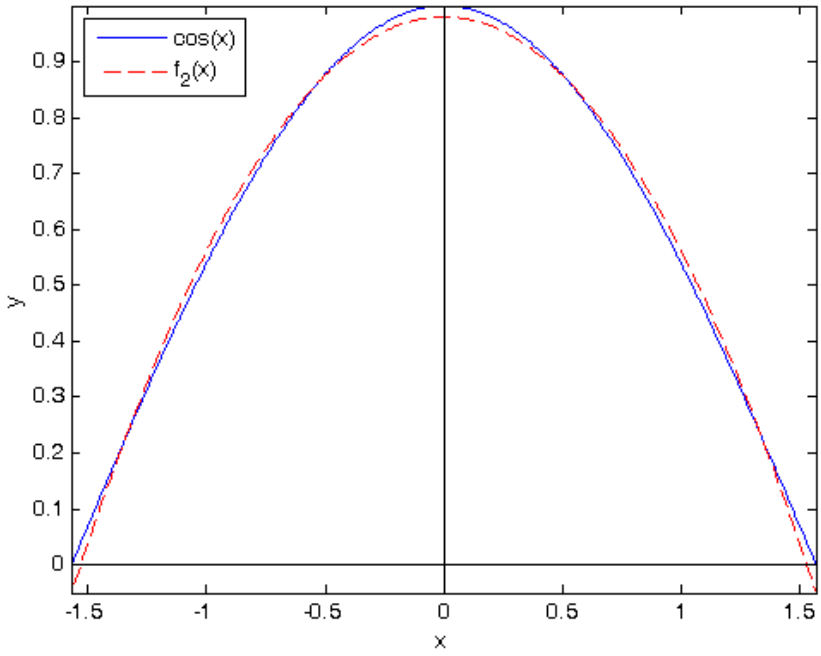

Figure 5.2: Graph of $\cos x$ (solid blue curve) and its continuous least-squares quadratic approximation (red dashed curve) on $[-\pi/2, \pi/2]$

we have been using is defined by

$$\langle f, g \rangle = \int_a^b f(x)g(x)w(x)\,dx,$$

where $w(x)$ is a *weight function*. To be a weight function, it is required that $w(x) \geq 0$ on (a, b), and that $w(x) \neq 0$ on any subinterval of (a, b). So far, we have only considered the case of $w(x) \equiv 1$.

Another weight function of interest is

$$w(x) = \frac{1}{\sqrt{1-x^2}}, \quad -1 < x < 1.$$

A sequence of polynomials that is orthogonal with respect to this weight function, and the associated inner product

$$\langle f, g \rangle = \int_{-1}^1 f(x)g(x)\frac{1}{\sqrt{1-x^2}}\,dx$$

is the sequence of *Chebyshev polynomials*

$$\begin{aligned} C_0(x) &= 1, \\ C_1(x) &= x, \\ C_{j+1}(x) &= 2xC_j(x) - C_{j-1}(x), \quad j = 1, 2, \ldots \end{aligned}$$

which can also be defined by

$$C_j(x) = \cos(j \cos^{-1} x), \quad -1 \leq x \leq 1.$$

It is interesting to note that if we let $x = \cos\theta$, then

$$\begin{aligned} \langle f, C_j \rangle &= \int_{-1}^{1} f(x) \cos(j \cos^{-1} x) \frac{1}{\sqrt{1-x^2}}\, dx \\ &= \int_0^{\pi} f(\cos\theta) \cos j\theta \, d\theta. \end{aligned}$$

In later lectures, we will investigate continuous and discrete least-squares approximation of functions by linear combinations of *trigonometric polynomials* such as $\cos j\theta$ or $\sin j\theta$, which will reveal one of the most useful applications of Chebyshev polynomials.

Previously, we established that if $f \in L^2(a, b)$, and $f_n(x)$ is a polynomial of degree n such that

$$f_n(x) = \sum_{j=0}^{n} c_j q_j(x),$$

where the polynomials $q_0(x), q_1(x), \ldots, q_n(x)$ are *orthonormal*, in the sense that

$$\langle q_j, q_k \rangle = \int_a^b q_j(x) q_k(x) w(x)\, dx = \delta_{jk}, \quad j, k = 0, 1, \ldots, n,$$

and the constants c_j are defined by

$$c_j = \langle f, q_j \rangle, \quad j = 0, 1, \ldots, n,$$

then the 2-norm approximation error

$$\|f - p\|_2 = \left(\int_a^b |f(x) - p(x)|^2 w(x)\, dx \right)^{1/2}, \quad p \in \mathcal{P}_n,$$

is minimized by f_n.

It follows that if $p \in \mathcal{P}_n$, then

$$\begin{aligned}
\langle f - f_n, p\rangle &= \langle f, p\rangle - \langle f_n, p\rangle \\
&= \left\langle f, \sum_{k=0}^{n} \langle p, q_k\rangle q_k \right\rangle - \left\langle \sum_{j=0}^{n} \langle f, q_j\rangle q_j, \sum_{k=0}^{n} \langle p, q_k\rangle q_k \right\rangle \\
&= \sum_{k=0}^{n} \langle p, q_k\rangle\langle f, q_k\rangle - \sum_{j=0}^{n}\sum_{k=0}^{n} \langle f, q_j\rangle\langle p, q_k\rangle\langle q_j, q_k\rangle \\
&= \sum_{k=0}^{n} \langle p, q_k\rangle\langle f, q_k\rangle - \sum_{k=0}^{n} \langle f, q_k\rangle\langle p, q_k\rangle \\
&= 0.
\end{aligned}$$

That is, the error in the approximation of f by f_n is orthogonal to *any* polynomial in $\mathcal{P}_n$.

Now, we prove the converse: that if $f_n \in \mathcal{P}_n$ satisfies

$$\langle f - f_n, p\rangle = 0, \quad p \in \mathcal{P}_n,$$

then $\|f - f_n\|_2$ is minimized over $\mathcal{P}_n$. First, we note that if $p \in \mathcal{P}_n$, then $p - f_n \in \mathcal{P}_n$ as well, so

$$0 = \langle f - f_n, p - f_n\rangle = \langle f - f_n, p - f_n + f - f\rangle = \langle f - f_n, f - f_n\rangle + \langle f - f_n, p - f\rangle,$$

or, by the Cauchy-Schwarz inequality,

$$\|f - f_n\|_2^2 = \langle f - f_n, f - p\rangle \leq \|f - f_n\|_2 \|f - p\|_2.$$

Clearly, if $f = f_n$, then $\|f - f_n\|_2 = 0$ is minimized, because a norm must be nonnegative. Otherwise, we have

$$\|f - f_n\|_2 \leq \|f - p\|_2.$$

We conclude that f_n is the closest polynomial in $\mathcal{P}_n$ to f in the 2-norm sense.

Finally, we prove one property of orthogonal polynomials that will prove useful in our upcoming discussion of the role of orthogonal polynomials in numerical integration. Let $\varphi_j(x)$ be a polynomial of degree $j \geq 1$ that is orthogonal to all polynomials of lower degree, with respect to the inner product

$$\langle f, g\rangle = \int_a^b f(x)g(x)w(x)\,dx,$$

and let the points $\xi_1, \xi_2, \ldots, \xi_k$ be the points in (a, b) at which $\varphi_j(x)$ changes sign. This set of points cannot be empty, because φ_j, being a polynomial of degree at least one, is orthogonal a constant function, which means

$$\int_a^b \varphi_j(x) w(x)\, dx = 0.$$

Because $w(x)$ is a weight function, it does not change sign. Therefore, in order for the integral to be zero, $\varphi_j(x)$ must change sign at least once in (a, b).

If we define

$$\pi_k(x) = (x - \xi_1)(x - \xi_2) \cdots (x - \xi_k),$$

then $\varphi_j(x)\pi_k(x)$ does not change sign on (a, b), because π_k changes sign at exactly the same points in (a, b) as φ_j. Because both polynomials are also nonzero on (a, b), we must have

$$\langle \varphi_j, \pi_k \rangle = \int_a^b \varphi_j(x)\pi_k(x) w(x)\, dx \neq 0.$$

If $k < j$, then we have a contradiction, because φ_j is orthogonal to *any* polynomial of lesser degree. Therefore, $k \geq j$. However, if $k > j$, we also have a contradiction, because a polynomial of degree j cannot change sign more than j times on the entire real number line, let alone an interval. We conclude that $k = j$, which implies that all of the roots of φ_j are real and distinct, and lie in (a, b).

5.4 Comparisons

Let $f_n \in \mathcal{P}_n$ be the best approximation of $f \in L^2(a, b)$ in the 2-norm sense, and let $\xi_1, \xi_2, \ldots, \xi_k$ be the points in (a, b) at which the error $f - f_n$ changes sign. Then, if we define π_k as before, we have

$$\langle f - f_n, \pi_k \rangle \neq 0.$$

However, $\langle f - f_n, p \rangle = 0$ for any polynomial p of degree n or less. It follows that π_k must have degree at least $n + 1$, so the error must change sign at least $n + 1$ times in (a, b). It follows that the best 2-norm approximation is also a *near-minimax* approximation of f. That is, the error in the ∞-norm sense is nearly minimized.

One particularly useful property of the 2-norm approximation is that it is easy to obtain an approximation of higher degree. Let f_n be the best nth-degree approximation of $f \in L^2(a,b)$, and suppose that we wish to obtain f_{n+1}, the best approximation of degree $n+1$. Given a sequence of orthonormal polynomials $q_0, q_1, \ldots, q_{n+1}$, we have

$$f_n(x) = \sum_{j=0}^{n} c_j q_j(x), \quad c_j = \langle f, q_j \rangle, \quad j = 0, 1, \ldots, n,$$

and

$$f_{n+1}(x) = \sum_{j=0}^{n+1} c_j q_j(x), \quad c_j = \langle f, q_j \rangle, \quad j = 0, 1, \ldots, n+1.$$

It follows that

$$f_{n+1}(x) = f_n(x) + c_{n+1} q_{n+1}(x), \quad c_{n+1} = \langle f, q_{n+1}.$$

That is, the nth-degree approximation can be reused. On the other hand, the nth-degree minimiax approximation, or the nth-degree Lagrange interpolating polynomial, cannot be used to efficiently obtain the corresponding approximation of degree $n+1$. However, it should be noted that if Newton interpolation is used instead, such an update is feasible.

Chapter 6

Numerical Integration - II

6.1 Construction of Gauss Quadrature Rules

Previously, we learned that a Newton-Cotes quadrature rule with n nodes has degree at most n. Therefore, it is natural to ask whether it is possible to select the nodes and weights of an n-point quadrature rule so that the rule has degree greater than n. *Gaussian quadrature rules* have the surprising property that they can be used to integrate polynomials of degree $2n-1$ exactly using only n nodes.

Gaussian quadrature rules can be constructed using a technique known as *moment matching*, or *direct construction*. For any nonnegative integer k, the k^{th} moment is defined to be

$$\mu_k = \int_a^b x^k \, dx.$$

For given n, our goal is to select weights and nodes so that the first $2n$ moments are computed exactly; i.e.,

$$\mu_k = \sum_{i=0}^{n-1} w_i x_i^k, \quad k = 0, 1, \ldots, 2n-1.$$

Since we have $2n$ free parameters, it is reasonable to think that appropriate nodes and weights can be found. Unfortunately, this system of equations is nonlinear, so it can be quite difficult to solve.

Suppose $g(x)$ is a polynomial of degree $2n-1$. For convenience, we will write $g \in \mathcal{P}_{2n-1}$, where, for any natural number k, $\mathcal{P}_k$ denotes the space of polynomials of degree at most k. We shall show that there exist weights

$\{w_i\}_{i=0}^{n-1}$ and nodes $\{x_i\}_{i=0}^{n-1}$ such that

$$\int_a^b g(x)\,dx = \sum_{i=0}^{n-1} w_i g(x_i).$$

Furthermore, for more general functions, $G(x)$,

$$\int_a^b G(x)\,dx = \sum_{i=0}^{n-1} w_i G(x_i) + E[G]$$

where

1. x_i are real, distinct, and $a < x_i < b$ for $i = 0, 1, \ldots, n-1$.
2. The weights $\{w_i\}$ satisfy $w_i > 0$ for $i = 0, 1, \ldots, n-1$.
3. The error $E[G]$ satisfies $E[G] = \frac{G^{(2n)}(\xi)}{(2n)!} \int_a^b \prod_{i=0}^{n-1}(x - x_i)^2\,dx$.

Notice that this method is exact for polynomials of degree $2n-1$ since the error functional $E[G]$ depends on the $(2n)^{th}$ derivative of G.

To prove this, we shall construct an *orthonormal family* of polynomials $\{q_i(x)\}_{i=0}^{n}$ so that

$$\int_a^b q_r(x)q_s(x)\,dx = \begin{cases} 0 & r \neq s, \\ 1 & r = s. \end{cases}$$

This can be accomplished using the fact that such a family of polynomials satisfies a *three-term recurrence relation*

$$\beta_j q_j(x) = (x-\alpha_j)q_{j-1}(x) - \beta_{j-1}q_{j-2}(x), \quad q_0(x) = (b-a)^{-1/2}, \quad q_{-1}(x) = 0,$$

where

$$\alpha_j = \int_a^b x q_{j-1}(x)^2\,dx, \quad \beta_j^2 = \int_a^b x q_j(x)q_{j-1}(x)\,dx, \quad j \geq 1, \quad \beta_0 = 1.$$

We choose the nodes $\{x_i\}$ to be the roots of the n^{th}-degree polynomial in this family, which are real, distinct and lie within (a, b), as proved earlier. Next, we construct the interpolant of degree $n-1$, denoted $p_{n-1}(x)$, of $g(x)$ through the nodes:

$$p_{n-1}(x) = \sum_{i=0}^{n-1} g(x_i)\mathcal{L}_{n-1,i}(x),$$

where, for $i = 0, \ldots, n-1$, $\mathcal{L}_{n-1,i}(x)$ is the ith Lagrange polynomial for the points $x_0, \ldots, x_{n-1}$. We shall now look at the interpolation error function

$$e(x) = g(x) - p_{n-1}(x).$$

Clearly, since $g \in \mathcal{P}_{2n-1}$, $e \in \mathcal{P}_{2n-1}$. Since $e(x)$ has roots at each of the roots of $q_n(x)$, we can factor e so that

$$e(x) = q_n(x)r(x),$$

where $r \in \mathcal{P}_{n-1}$. It follows from the fact that $q_n(x)$ is orthogonal to *any* polynomial in $\mathcal{P}_{n-1}$ that the integral of g can then be written as

$$\begin{aligned} I(g) &= \int_a^b p_{n-1}(x)\,dx + \int_a^b q_n(x)r(x)\,dx \\ &= \int_a^b p_{n-1}(x)\,dx \\ &= \int_a^b \sum_{i=0}^{n-1} g(x_i)\mathcal{L}_{n-1,i}(x)\,dx \\ &= \sum_{i=0}^{n-1} g(x_i) \int_a^b \mathcal{L}_{n-1,i}(x)\,dx \\ &= \sum_{i=0}^{n-1} g(x_i)w_i \end{aligned}$$

where

$$w_i = \int_a^b \mathcal{L}_{n-1,i}(x)\,dx, \quad i = 0, 1, \ldots, n-1.$$

For a more general function $G(x)$, the error functional $E[G]$ can be obtained from the expression for the interpolation error presented earlier.

Example We will use *Gaussian quadrature* to approximate the integral

$$\int_0^1 e^{-x^2}\,dx.$$

The particular Gaussian quadrature rule that we will use consists of 5 nodes x_0, x_1, x_2, x_3 and x_4, and 5 weights w_0, w_1, w_2, w_3 and w_4. To determine the proper nodes and weights, we use the fact that the nodes and weights of a 5-point Gaussian rule for integrating over the interval $[-1, 1]$ are given by

i	Nodes $r_{5,i}$	Weights $c_{5,i}$
0	0.9061798459	0.2369268850
1	0.5384693101	0.4786286705
2	0.0000000000	0.5688888889
3	−0.5384693101	0.4786286705
4	−0.9061798459	0.2369268850

To obtain the corresponding nodes and weights for integrating over $[0, 1]$, we can use the fact that in general,

$$\int_a^b f(x)\,dx = \int_{-1}^{1} f\left(\frac{b-a}{2}t + \frac{a+b}{2}\right)\frac{b-a}{2}\,dt,$$

as can be shown using the change of variable $x = [(b-a)/2]t + (a+b)/2$ that maps $[a, b]$ into $[-1, 1]$. We then have

$$\begin{aligned}\int_a^b f(x)\,dx &= \int_{-1}^{1} f\left(\frac{b-a}{2}t + \frac{a+b}{2}\right)\frac{b-a}{2}\,dt \\ &\approx \sum_{i=0}^{4} f\left(\frac{b-a}{2}r_{5,i} + \frac{a+b}{2}\right)\frac{b-a}{2}c_{5,i} \\ &\approx \sum_{i=0}^{4} f(x_i)w_i,\end{aligned}$$

where

$$x_i = \frac{b-a}{2}r_{5,i} + \frac{a+b}{2}, \quad w_i = \frac{b-a}{2}c_{5,i}, \quad i = 0, \ldots, 4.$$

In this example, $a = 0$ and $b = 1$, so the nodes and weights for a 5-point Gaussian quadrature rule for integrating over $[0, 1]$ are given by

$$x_i = \frac{1}{2}r_{5,i} + \frac{1}{2}, \quad w_i = \frac{1}{2}c_{5,i}, \quad i = 0, \ldots, 4,$$

which yields

i	Nodes x_i	Weights w_i
0	0.95308992295	0.11846344250
1	0.76923465505	0.23931433525
2	0.50000000000	0.28444444444
3	0.23076534495	0.23931433525
4	0.04691007705	0.11846344250

It follows that

$$\begin{aligned}\int_0^1 e^{-x^2}\,dx &\approx \sum_{i=0}^{4} e^{-x_i^2} w_i \\ &\approx 0.11846344250e^{-0.95308992295^2} + 0.23931433525e^{-0.769234655052} + \\ &\quad 0.28444444444e^{-0.5^2} + 0.23931433525e^{-0.230765344952} + \\ &\quad 0.11846344250e^{-0.046910077052} \\ &\approx 0.74682412673352.\end{aligned}$$

Since the exact value is 0.74682413281243, the absolute error is -6.08×10^{-9}, which is remarkably accurate considering that only fives nodes are used. □

The high degree of accuracy of Gaussian quadrature rules make them the most commonly used rules in practice. However, they are not without their drawbacks:

- They are not progressive, so the nodes must be recomputed whenever additional degrees of accuracy are desired. An alternative is to use *Gauss-Kronrod rules*. A $(2n+1)$-point Gauss-Kronrod rule uses the nodes of the n-point Gaussian rule. For this reason, practical quadrature procedures use both the Gaussian rule and the corresponding Gauss-Kronrod rule to estimate accuracy.

- Because the nodes are the roots of a polynomial, they must be computed using traditional root-finding methods, which are not always accurate. Errors in the computed nodes lead to lost degrees of accuracy in the approximate integral. In practice, however, this does not normally cause significant difficulty.

6.2 Error Estimation for Gauss Quadrature

We have learned how to approximate

$$I[G] = \int_a^b G(x)\,dx$$

using a *Gaussian* quadrature rule

$$\mathcal{G}_n[G] = \sum_{i=0}^{n-1} G(x_i) w_i$$

that is exact whenever $G(x) \in \mathcal{P}_{2n-1}$; that is, G is a polynomial of degree at $2n-1$ or less.

It is easy to show that the weights w_i are positive. Since the interpolation basis functions $\mathcal{L}_{n-1,i}$ belong to $\mathcal{P}_{n-1}$, it follows that $\mathcal{L}_{n-1,i}^2 \in \mathcal{P}_{2n-2}$, and therefore

$$0 < \int_a^b \mathcal{L}_{n-1,i}^2(x)\,dx = \sum_{j=0}^{n-1} w_j \mathcal{L}_{n-1,i}^2(x_j) = w_i.$$

Note that we have thus obtained an alternative formula for the weights. This formula also arises from an alternative approach to constructing Gaussian quadrature rules, from which a representation of the error can easily be obtained.

We construct the Hermite interpolating polynomial $G_{2n-1}(x)$ of $G(x)$, using the Gaussian quadrature nodes as interpolation points, that satisfies the $2n$ conditions

$$G_{2n-1}(x_i) = G(x_i), \quad G'_{2n-1}(x_i) = G'(x_i), \quad i = 0, 1, \ldots, n-1.$$

This interpolant has the form

$$G_{2n-1}(x) = \sum_{i=0}^{n-1} G(x_i)H_i(x) + \sum_{i=0}^{n-1} G'(x_i)K_i(x),$$

where, as in our previous discussion of Hermite interpolation,

$$H_i(x_j) = \delta_{ij}, \quad H_i'(x_j) = 0, \quad K_i(x_j) = 0, \quad K_i'(x_j) = \delta_{ij}, \quad i, j = 0, 1, \ldots, n-1.$$

Then, we have

$$\int_a^b G_{2n-1}(x)\,dx = \sum_{i=0}^{n-1} G(x_i) \int_a^b H_i(x)\,dx + \sum_{i=0}^{n-1} G'(x_i) \int_a^b K_i(x)\,dx.$$

We recall that

$$H_i(x) = \mathcal{L}_{n-1,i}(x)^2[1-2\mathcal{L}'_{n-1,i}(x_i)(x-x_i)], \quad K_i(x) = \mathcal{L}_{n-1,i}(x)^2(x-x_i), \quad i = 0, 1, \ldots, n-1,$$

and for convenience, we define

$$\pi_n(x) = (x - x_0)(x - x_1)\cdots(x - x_{n-1}),$$

and note that

$$\mathcal{L}_{n-1,i}(x) = \frac{\pi_n(x)}{(x - x_i)\pi_n'(x_i)}.$$

We then have

$$\begin{aligned}\int_a^b H_i(x)\,dx &= \int_a^b \mathcal{L}_{n-1,i}(x)^2\,dx - 2\mathcal{L}'_{n-1,i}(x_i)\int_a^b \mathcal{L}_{n-1,i}(x)^2(x-x_i)\,dx \\ &= \int_a^b \mathcal{L}_{n-1,i}(x)^2\,dx - \frac{2\mathcal{L}'_{n-1,i}(x_i)}{\pi'_n(x_i)}\int_a^b \mathcal{L}_{n-1,i}(x)\pi_n(x)\,dx \\ &= \int_a^b \mathcal{L}_{n-1,i}(x)^2\,dx,\end{aligned}$$

as the second term vanishes because $\mathcal{L}_{n-1,i}(x)$ is of degree $n-1$, and $\pi_n(x)$, a polynomial of degree n, is orthogonal to all polynomials of lesser degree.

Similarly,

$$\int_a^b K_i(x)\,dx = \int_a^b \mathcal{L}_{n-1,i}(x)^2(x-x_i)\,dx = \frac{1}{\pi'_n(x_i)}\int_a^b \mathcal{L}_{n-1,i}(x)\pi_n(x)\,dx = 0.$$

We conclude that

$$\int_a^b G_{2n-1}(x)\,dx = \sum_{i=0}^{n-1} G(x_i)w_i,$$

where, as before,

$$w_i = \int_a^b \mathcal{L}_{n-1,i}(x)^2\,dx = \int_a^b \mathcal{L}_{n-1,i}(x)\,dx.$$

The equivalence of these formulas for the weights can be seen from the fact that the difference $\mathcal{L}_{n-1,i}(x)^2 - \mathcal{L}_{n-1,i}(x)$ is a polynomial of degree $2n-2$ that is divisible by $\pi_n(x)$, because it vanishes at all of the nodes. The quotient, a polynomial of degree $n-2$, is orthogonal to $\pi_n(x)$. Therefore, the integrals of $\mathcal{L}_{n-1,i}(x)^2$ and $\mathcal{L}_{n-1,i}(x)$ must be equal.

We now use the error in the Hermite interpolating polynomial to obtain

$$\begin{aligned}E[G] &= \int_a^b G(x)\,dx - \sum_{i=0}^{n-1} G(x_i)w_i \\ &= \int_a^b [G(x) - G_{2n-1}(x)]\,dx \\ &= \int_a^b \frac{G^{(2n)}(\xi(x))}{(2n)!}\pi_n(x)^2\,dx \\ &= \frac{G^{(2n)}(\xi)}{(2n)!}\int_a^b \pi_n(x)^2\,dx,\end{aligned}$$

where $\xi \in (a, b)$. The last step is obtained using the Weighted Mean Value Theorem for Integrals, which applies because $\pi_n(x)^2$ does not change sign.

In addition to this error formula, we can easily obtain qualitative bounds on the error. For instance, if we know that the even derivatives of g are positive, then we know that the quadrature rule yields a lower bound for $I(g)$. Similarly, if the even derivatives of g are negative, then the quadrature rule gives an upper bound.

Now, we show that if $f \in C[a, b]$, that the n-node Gaussian quadrature approximation of $I[f]$ converges to $I[f]$ as $n \to \infty$. That is,

$$\lim_{n\to\infty} \mathcal{G}_n[f] = I[f].$$

First, we recall that by the Weierstrass Theorem, for any $\epsilon_0 > 0$, there exists a polynomial $p(x)$ such that $\|f - p\|_\infty \leq \epsilon_0$. We let N denote the degree of this polynomial, and from this point on refer to this approximating polynomial as $p_N(x)$.

We now have

$$\begin{aligned} I[f] - \mathcal{G}_n[f] &= I[f] - \int_a^b p_N(x)\,dx + \\ &\quad \int_a^b p_N(x)\,dx - \mathcal{G}_n[p_N] + \\ &\quad \mathcal{G}_n[p_N] - \mathcal{G}_n[f]. \end{aligned}$$

For the first two terms, we have

$$\left|I[f] - \int_a^b p_N(x)\,dx\right| = \left|\int_a^b [f(x) - p_N(x)]\,dx\right| \leq \int_a^b |f(x) - p_N(x)|\,dx \leq \epsilon_0 W,$$

where

$$W = \int_a^b dx = b - a.$$

Next, we have

$$\int_a^b p_N(x)\,dx - \mathcal{G}_n[p_N] = 0,$$

if n is chosen so that $n \geq (N + 1)/2$, since an n-node Gaussian rule is exact for all polynomials of degree $2n - 1$.

Finally, we have

$$|\mathcal{G}_n[p_N] - \mathcal{G}_n[f]| = \left|\sum_{i=0}^{n-1} [p_N(x_i) - f(x_i)]w_i\right| \leq \sum_{i=0}^{n-1} |p_N(x_i) - f(x_i)||w_i| \leq \epsilon_0 \sum_{i=0}^{n-1} |w_i| \leq \epsilon_0 W,$$

because all of the weights are positive, and therefore their sum is equal to the integral of the function $f(x) \equiv 1$ over $[a, b]$.

It follows that

$$|I[f] - \mathcal{G}_n[f]| \leq 2\epsilon_0 W,$$

and therefore if we choose $\epsilon_0 = \epsilon/(2W)$, where $\epsilon > 0$ is arbitrary, we obtain

$$|I[f] - \mathcal{G}_n[f]| \leq \epsilon$$

for n sufficiently large. We conclude that $\mathcal{G}_n[f]$ converges to $I[f]$ in the limit as $n \to \infty$.

It is important to note that such a result does not apply to Newton-Cotes quadrature rules, because they can have negative weights. This means that the sum of the absolute value of the weights is not necessarily bounded, as is the case for Gaussian rules. We also note that all of the above error analysis and convergence analysis applies when a nonnegative weight function $w(x)$ is included in the integral.

6.3 Composite Gauss Formulae

Just as the Trapezoidal, Midpoint and Simpson's Rules can be applied on subintervals of $[a, b]$ to obtain composite rules, Gaussian rules can also be used to construct composite rules. We assume that the integrand $f(x)$ is $2n$-times continuously differentiable on (a, b). Let $[a, b]$ be divided into m subintervals of width $h = (b-a)/m$ with endpoints $[x_{i-1}, x_i]$, $i = 1, 2, \ldots, m$, where $x_i = a + ih$, and let ξ_j and w_j, $j = 0, 1, \ldots, n-1$, be the Gaussian quadrature nodes and weights, respectively, for $[-1, 1]$. Then, the composite Gaussian quadrature rule can be defined by

$$\int_a^b f(x)\,dx = \frac{h}{2} \sum_{i=1}^{m} \sum_{j=0}^{n-1} f\left(x_{i-1} + \frac{h}{2}(\xi_j + 1)\right) w_j + \mathcal{E}_{m,n},$$

where $\mathcal{E}_{m,n}$ is the error in the approximate integral.

A formula for this error can be obtained by summing the error on each subinterval, which yields

$$\begin{aligned} \mathcal{E}_{m,n} &= \sum_{i=1}^{m} \frac{f^{(2n)}(\eta_i)}{(2n)!} \int_{x_{i-1}}^{x_i} \prod_{j=0}^{n-1} \left(x - x_{i-1} - \frac{h}{2}(\xi_j + 1)\right)^2 dx \\ &= \frac{h^{2n+1}}{2^{2n+1}} \sum_{i=1}^{m} \frac{f^{(2n)}(\eta_i)}{(2n)!} \int_{-1}^{1} \pi_n(t)^2\,dt \end{aligned}$$

$$= \frac{(b-a)^{2n+1}}{m^{2n}2^{2n+1}} \frac{f^{(2n)}(\eta)}{(2n)!} \int_{-1}^{1} \pi_n(t)^2 \, dt,$$

where $\eta \in (a, b)$ and $\pi_n(t) = (t - \xi_0)(t - \xi_1) \cdots (t - \xi_{n-1})$.

When $n = 1$, the only Gaussian quadrature node on $[-1, 1]$ is located at the midpoint, 0. It follows that the composite 1-node Gaussian rule is actually the composite Midpoint Rule, which, as previously shown, has error

$$\mathcal{E}_{m,1} = \frac{(b-a)^3}{m^2 2^3} \frac{f''(\eta)}{2} \int_{-1}^{1} t^2 \, dt = \frac{(b-a)h^2}{24} f''(\eta).$$

6.4 Radau and Lobatto Quadrature

Often, variations of Gaussian quadrature rules are used in which one or more nodes are prescribed. For example, *Gauss-Radau rules* are rules in which either of the endpoints of the interval $[a, b]$ are chosen to be a node, and n additional nodes are determined by a procedure similar to that used in Gaussian quadrature, resulting in a rule of degree $2n$. In *Gauss-Lobatto rules*, both endpoints of $[a, b]$ are nodes, with n additional nodes chosen in order to obtain a rule of degree $2n+1$. It should be noted that Gauss-Lobatto rules are closed, whereas Gaussian rules are open.

Chapter 7

Piecewise Polynomial Approximation

7.1 Linear Interpolating Splines

We have seen that high-degree polynomial interpolation can be problematic. However, if the fitting function is only required to have a few continuous derivatives, then one can construct a *piecewise polynomial* to fit the data. We now precisely define what we mean by a piecewise polynomial.

Definition *(Piecewise polynomial) Let $[a, b]$ be an interval that is divided into subintervals $[x_i, x_{i+1}]$, where $i = 0, \ldots, n-1$, $x_0 = a$ and $x_n = b$. A* **piecewise polynomial** *is a function $p(x)$ defined on $[a, b]$ by*

$$p(x) = p_i(x), \quad x_{i-1} \leq x \leq x_i, \quad i = 1, 2, \ldots, n,$$

where, for $i = 1, 2, \ldots, n$, each function $p_i(x)$ is a polynomial defined on $[x_{i-1}, x_i]$. The **degree** *of $p(x)$ is the maximum degree of each polynomial $p_i(x)$, for $i = 1, 2, \ldots, n$.*

It is essential to note that by this definition, a piecewise polynomial defined on $[a, b]$ is equal to some polynomial on each subinterval $[x_{i-1}, x_i]$ of $[a, b]$, for $i = 1, 2, \ldots, n$, but a different polynomial may be used for each subinterval.

We first consider one of the simplest types of piecewise polynomials, a piecewise linear polynomial. Let $f \in C[a, b]$. Given the points $x_0, x_1, \ldots, x_n$ defined as above, the *spline, linear* $s_{\mathrm{L}}(x)$ that interpolates f at these points is defined by

$$s_{\mathrm{L}}(x) = f(x_{i-1})\frac{x - x_i}{x_{i-1} - x_i} + f(x_i)\frac{x - x_{i-1}}{x_i - x_{i-1}}, \quad x \in [x_{i-1}, x_i], \quad i = 1, 2, \ldots, n.$$

The points $x_0, x_1, \ldots, x_n$ are the *knots* of the spline.

Before we study the accuracy of linear splines, we introduce some terminology and notation. First, we say that a function f is *absolutely continouous* on $[a, b]$ if its derivative is finite almost everywhere in $[a, b]$ (meaning that it is not finite on at most a subset of $[a, b]$ that has *measure zero*), is integrable on $[a, b]$, and satisfies

$$\int_a^x f'(s)\,dx = f(x) - f(a), \quad a \leq x \leq b.$$

Any continuously differentiable function is absolutely continuous, but the converse is not necessarily true.

Example For example, $f(x) = |x|$ is absolutely continuous on any interval of the form $[-a, a]$, but it is not continuously differentiable on such an interval. □

Next, we define the *Sobolev spaces* $H^k(a, b)$ as follows. The space $H^1(a, b)$ is the set of all absolutely continuous functions on $[a, b]$ whose derivatives belong to $L^2(a, b)$. Then, for $k > 1$, $H^k(a, b)$ is the subset of $H^{k-1}(a, b)$ consisting of functions whose $(k-1)$st derivatives are absolutely continuous, and whose kth derivatives belong to $L^2(a, b)$. If we denote by $C^k[a, b]$ the set of all functions defined on $[a, b]$ that are k times continuously differentiable, then $C^k[a, b]$ is a proper subset of $H^k(a, b)$. For example, any linear spline belongs to $H^1(a, b)$, but does not generally belong to $C^1[a, b]$.

Example The function $f(x) = x^{3/4}$ belongs to $H^1(0, 1)$ because $f'(x) = \frac{3}{4}x^{-1/4}$ is integrable on $[0, 1]$, and also *square-integrable* on $[0, 1]$, since

$$\int_0^1 |f'(x)|^2\,dx = \int_0^1 \frac{9}{16}x^{-1/2} = \frac{9}{8}x^{1/2}\Big|_0^1 = \frac{9}{8}.$$

However, $f \notin C^1[a, b]$, because $f'(x)$ is singular at $x = 0$. □

Now, if $f \in C^2[a, b]$, then by the error in Lagrange interpolation, on each subinterval $[x_{i-1}, x_i]$, for $i = 1, 2, \ldots, n$, we have

$$f(x) - s_{\mathrm{L}}(x) = \frac{f''(\xi)}{2}(x - x_{i-1})(x - x_i).$$

If we let $h_i = x_i - x_{i-1}$, then the function $(x - x_{i-1})(x - x_i)$ achieves its maximum absolute value at $x = (x_{i-1} + x_i)/2$, with a maximum value of $h_i^2/4$. If we define $h = \max_{1 \leq i \leq n} h_i$, then we have

$$\|f - s_{\mathrm{L}}\|_\infty \leq \frac{1}{8}h^2\|f''\|_\infty,$$

where $\|\cdot\|_\infty$ denotes the ∞-norm over $[a,b]$.

One of the most useful properties of the linear spline $s_{\rm L}(x)$ is that among all functions in $H^1(a,b)$ that interpolate $f(x)$ at the knots $x_0, x_1, \ldots, x_n$, it is the "flattest". That is, for any function $v \in H^1(a,b)$ that interpolates f at the knots,

$$\|s'_{\rm L}\|_2 \leq \|v'\|_2.$$

To prove this, we first write

$$\|v'\|_2^2 = \|v' - s'_{\rm L}\|_2^2 + 2\langle v' - s'_{\rm L}, s'_{\rm L}\rangle + \|s'_{\rm L}\|_2^2.$$

Then, applying integration by parts, we obtain

$$\begin{aligned}
\langle v' - s'_{\rm L}, s'_{\rm L}\rangle &= \int_a^b [v'(x) - s'_{\rm L}(x)]s'_{\rm L}(x)\,dx \\
&= \sum_{i=1}^n \int_{x_{i-1}}^{x_i} [v'(x) - s'_{\rm L}(x)]s'_{\rm L}(x)\,dx \\
&= \sum_{i=1}^n \left\{ [v(x) - s_{\rm L}(x)]s'_{\rm L}(x)\big|_{x_{i-1}}^{x_i} - \int_{x_{i-1}}^{x_i} [v(x) - s_{\rm L}(x)]s''_{\rm L}(x)\,dx \right\}.
\end{aligned}$$

However, $s_{\rm L}$ is a linear function on each subinterval $[x_{i-1}, x_i]$, so $s''_{\rm L}(x) \equiv 0$ on each subinterval. Furthermore, because both $v(x)$ and $s_{\rm L}(x)$ interpolate $f(x)$ at the knots, the bounday terms vanish, and therefore $\langle v' - s'_{\rm L}, s'_{\rm L}\rangle = 0$, which establishes the result.

7.2 Basis Functions for Linear Splines

Lagrange interpolation allows the unique polynomial $p_n(x)$ of degree n that interpolates $f(x)$ at the knots $x_0, x_1, \ldots, x_n$ to be expressed in the convenient form

$$p_n(x) = \sum_{i=0}^n f(x_i)\mathcal{L}_{n,i}(x).$$

A similar form can be obtained for the linear spline $s_{\rm L}(x)$ using *linear basis splines*, which are piecewise linear functions that are equal to one at one of the knots, and equal to zero at all other knots.

These functions, known as *hat functions* due to the shapes of their graphs, are defined as follows:

$$\varphi_0(x) = \begin{cases} (x_1 - x)/h_1 & x_0 \leq x < x_1, \\ 0 & x_1 \leq x \leq x_n \end{cases},$$

$$\varphi_i(x) = \begin{cases} 0 & x_0 \le x < x_{i-1}, \\ (x - x_{i-1})/h_i & x_{i-1} \le x < x_i, \\ (x_{i+1} - x)/h_{i+1} & x_i \le x < x_{i+1}, \\ 0 & x_{i+1} \le x \le x_n \end{cases}, \quad i = 1, 2, \ldots, n-1,$$

$$\varphi_n(x) = \begin{cases} 0 & x_0 \le x < x_{n-1}, \\ (x - x_{n-1})/h_n & x_{n-1} \le x \le x_n \end{cases}.$$

Then, the linear spline can be expressed as

$$s_{\mathrm{L}}(x) = \sum_{i=0}^{n} f(x_i)\varphi_i(x).$$

7.3 Cubic Splines

Typically, piecewise polynomials are used to fit smooth functions, and therefore are required to have a certain number of continuous derivatives. This requirement imposes additional constraints on the piecewise polynomial, and therefore the degree of the polynomials used on each subinterval must be chosen sufficiently high to ensure that these constraints can be satisfied.

7.3.1 Cubic Spline Interpolation

A *spline* is a piecewise polynomial of degree k that has $k-1$ continuous derivatives. The most commonly used spline is a *cubic spline*, which we now define.

Definition *(Cubic Spline) Let $f(x)$ be function defined on an interval $[a, b]$, and let $x_0, x_1, \ldots, x_n$ be $n+1$ distinct points in $[a, b]$, where $a = x_0 < x_1 < \cdots < x_n = b$. A* **cubic spline***, or* **cubic spline interpolant***, is a piecewise polynomial $s(x)$ that satisifes the following conditions:*

1. *On each interval $[x_{i-1}, x_i]$, for $i = 1, \ldots, n$, $s(x) = s_i(x)$, where $s_i(x)$ is a cubic polynomial.*
2. *$s(x_i) = f(x_i)$ for $i = 0, 1, \ldots, n$.*
3. *$s(x)$ is twice continuously differentiable on (a, b).*
4. *Either of the following boundary conditions are satisfied:*
   (a) *$s''(a) = s''(b) = 0$, which is called* **free** *or* **natural boundary conditions***, and*

(b) $s'(a) = f'(a)$ *and* $s'(b) = f'(b)$, *which is called* **clamped boundary conditions**.

If $s(x)$ *satisfies free boundary conditions, we say that* $s(x)$ *is a* **natural spline**. *The points* $x_0, x_1, \ldots, x_n$ *are called the* **nodes** *of* $s(x)$.

Clamped boundary conditions are often preferable because they use more information about $f(x)$, which yields a spline that better approximates $f(x)$ on $[a, b]$. However, if information about $f'(x)$ is not available, then free boundary conditions must be used instead.

7.3.2 Constructing Cubic Splines

Suppose that we wish to construct a cubic spline interpolant $s(x)$ that fits the given data (x_0, y_0), $(x_1, y_1), \ldots, (x_n, y_n)$, where $a = x_0 < x_1 < \cdots < x_n = b$, and $y_i = f(x_i)$, for some known function $f(x)$ defined on $[a, b]$. From the preceding discussion, this spline is a piecewise polynomial of the form

$$s(x) = s_i(x) = d_i(x-x_{i-1})^3+c_i(x-x_{i-1})^2+b_i(x-x_{i-1})+a_i, \quad i = 1, 2, \ldots, n, \quad x_{i-1} \leq x \leq x_i.$$

That is, the value of $s(x)$ is obtained by evaluating a different cubic polynomial for each subinterval $[x_{i-1}, x_i]$, for $i = 1, 2, \ldots, n$.

We now use the definition of a cubic spline to construct a system of equations that must be satisfied by the coefficients a_i, b_i, c_i and d_i for $i = 1, 2, \ldots, n$. We can then compute these coefficients by solving the system. Because $s(x)$ must fit the given data, we have

$$a_i = y_{i-1}, \quad i = 1, 2, \ldots, n.$$

If we define $h_i = x_i - x_{i-1}$, for $i = 1, 2, \ldots, n$, and define $a_{n+1} = y_n$, then the requirement that $s(x)$ is continuous at the interior nodes implies that we must have $s_i(x_i) = s_{i+1}(x_i)$ for $i = 1, 2, \ldots, n-1$. Furthermore, because $s(x)$ must fit the given data, we must also have $s(x_n) = s_n(x_n) = y_n$. These conditions lead to the constraints

$$d_i h_i^3 + c_i h_i^2 + b_i h_i + a_i = a_{i+1}, \quad i = 1, 2, \ldots, n.$$

To ensure that $s(x)$ has a continuous first derivative at the interior nodes, we require that $s_i'(x_i) = s_{i+1}'(x_i)$ for $i = 1, 2 \ldots, n-1$, which imposes the constraints

$$3d_i h_i^2 + 2c_i h_i + b_i = b_{i+1}, \quad i = 1, 2, \ldots, n-1.$$

Similarly, to enforce continuity of the second derivative at the interior nodes, we require that $s_i''(x_i) = s_{i+1}''(x_i)$ for $i = 1, 2, \ldots, n-1$, which leads to the constraints

$$3d_i h_i + c_i = c_{i+1}, \quad i = 1, 2, \ldots, n-1.$$

There are $4n$ coefficients to determine, since there are n cubic polynomials, with 4 coefficients each. However, we have only prescribed $4n-2$ constraints, so we must specify 2 more in order to determine a unique solution. If we use free boundary conditions, then these constraints are

$$\begin{aligned} c_1 &= 0, \\ 3d_n h_n + c_n &= 0. \end{aligned}$$

On the other hand, if we use clamped boundary conditions, then our additional constraints are

$$\begin{aligned} b_1 &= z_0, \\ 3d_n h_n^2 + 2c_n h_n + b_n &= z_n, \end{aligned}$$

where $z_i = f'(x_i)$ for $i = 0, 1, \ldots, n$.

Having determined our constraints that must be satisfied by $s(x)$, we can set up a system of linear equations $A\mathbf{x} = \mathbf{b}$ based on these constraints, and then solve this system to determine the coefficients a_i, b_i, c_i, d_i for $i = 1, 2 \ldots, n$. In the case of free boundary conditions, A is an $(n+1) \times (n+1)$ matrix is defined by

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & \ddots & & \vdots \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & h_{n-1} & 2(h_{n-1}+h_n) & h_n \\ 0 & \cdots & \cdots & 0 & 0 & 1 \end{bmatrix}$$

and the $(n+1)$-vectors $\mathbf{x}$ and $\mathbf{b}$ are

$$\mathbf{x} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n+1} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ \frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1) \\ \vdots \\ \frac{3}{h_n}(a_{n+1} - a_n) - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \\ 0 \end{bmatrix},$$

where $c_{n+1} = s''(x_n)/2$.

In the case of clamped boundary conditions, we have

$$A = \begin{bmatrix} 2h_1 & h_1 & 0 & \cdots & \cdots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & \ddots & & \vdots \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & h_{n-1} & 2(h_{n-1}+h_n) & h_n \\ 0 & \cdots & \cdots & 0 & h_n & 2h_n \end{bmatrix}$$

and

$$\mathbf{x} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n+1} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \frac{3}{h_1}(a_2 - a_1) - 3z_0 \\ \frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1) \\ \vdots \\ \frac{3}{h_n}(a_{n+1} - a_n) - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \\ 3z_n - \frac{3}{h_n}(a_{n+1} - a_n) \end{bmatrix}.$$

Once the coefficients $c_1, c_2, \ldots, c_{n+1}$ have been determined, the remaining coefficients can be computed as follows:

1. The coefficients $a_1, a_2, \ldots, a_{n+1}$ have already been defined by the relations $a_i = y_{i-1}$, for $i = 0, 1, \ldots, n+1$.

2. The coefficients $b_1, b_2, \ldots, b_n$ are given by
$$b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(2c_i + c_{i+1}), \quad i = 1, 2, \ldots, n.$$

3. The coefficients $d_1, d_2, \ldots, d_n$ can be obtained using the constraints
$$3d_i h_i + c_i = c_{i+1}, \quad i = 1, 2, \ldots, n.$$

Example We will construct a cubic spline interpolant for the following data on the interval $[0, 2]$.

j	x_j	y_j
0	0	3
1	1/2	−4
2	1	5
3	3/2	−6
4	2	7

The spline, $s(x)$, will consist of four pieces $\{s_j(x)\}_{j=1}^4$, each of which is a cubic polynomial of the form

$$s_j(x) = a_j + b_j(x - x_{j-1}) + c_j(x - x_{j-1})^2 + d_j(x - x_{j-1})^3, \quad j = 1, 2, 3, 4.$$

We will impose free, or natural, boundary conditions on this spline, so it will satisfy the conditions $s''(0) = s''(2) = 0$, in addition to the "essential" conditions imposed on a spline: it must fit the given data and have continuous first and second derivatives on the interval $[0, 2]$.

These conditions lead to the following system of equations that must be solved for the coefficients c_1, c_2, c_3, c_4, and c_5, where $c_j = s''(x_{j-1})/2$ for $j = 1, 2, \ldots, 5$. We define $h = (2 - 0)/4 = 1/2$ to be the spacing between the interpolation points.

$$\begin{aligned} c_1 &= 0 \\ \frac{h}{3}(c_1 + 4c_2 + c_3) &= \frac{y_2 - 2y_1 + y_0}{h} \\ \frac{h}{3}(c_2 + 4c_3 + c_4) &= \frac{y_3 - 2y_2 + y_1}{h} \\ \frac{h}{3}(c_3 + 4c_4 + c_5) &= \frac{y_4 - 2y_3 + y_2}{h} \\ c_5 &= 0. \end{aligned}$$

Substituting $h = 1/2$ and the values of y_j, and also taking into account the boundary conditions, we obtain

$$\begin{aligned} \frac{1}{6}(4c_2 + c_3) &= 32 \\ \frac{1}{6}(c_2 + 4c_3 + c_4) &= -40 \\ \frac{1}{6}(c_3 + 4c_4) &= 48 \end{aligned}$$

This system has the solutions

$$c_1 = 516/7, \quad c_2 = -720/7, \quad c_3 = 684/7.$$

Using the relation $a_{j+1} = y_j$, for $j = 0, 1, 2, 3$, and the formula

$$b_j = \frac{a_{j+1} - a_j}{h} - \frac{h}{3}(2c_j + c_{j+1}), \quad j = 1, 2, 3, 4,$$

we obtain

$$b_1 = -184/7, \quad b_2 = 74/7, \quad b_3 = -4, \quad b_4 = -46/7.$$

Finally, using the formula

$$d_j = \frac{c_{j+1} - c_j}{3h}, \quad j = 1, 2, 3, 4,$$

we obtain

$$d_1 = 344/7, \quad d_2 = -824/7, \quad d_3 = 936/7, \quad d_4 = -456/7.$$

We conclude that the spline $s(x)$ that fits the given data, has two continuous derivatives on $[0, 2]$, and satisfies natural boundary conditions is

$$s(x) = \begin{cases} \frac{344}{7}x^3 - \frac{184}{7}x^2 + 3 & \text{if } x \in [0, 0.5] \\ -\frac{824}{7}(x - 1/2)^3 + \frac{516}{7}(x - 1/2)^2 + \frac{74}{7}(x - 1/2) - 4 & \text{if } x \in [0.5, 1] \\ \frac{936}{7}(x - 1)^3 - \frac{720}{7}(x - 1)^2 - 4(x - 1) + 5 & \text{if } x \in [1, 1.5] \\ -\frac{456}{7}(x - 3/2)^3 + \frac{684}{7}(x - 3/2)^2 - \frac{46}{7}(x - 3/2) - 6 & \text{if } x \in [1.5, 2] \end{cases}.$$

The graph of the spline is shown in Figure 7.1. □

7.3.3 Well-Posedness and Accuracy

For both boundary conditions, the system $A\mathbf{x} = \mathbf{b}$ has a unique solution, which leads to the following results.

Theorem *Let $x_0, x_1, \ldots, x_n$ be $n + 1$ distinct points in the interval $[a, b]$, where $a = x_0 < x_1 < \cdots < x_n = b$, and let $f(x)$ be a function defined on $[a, b]$. Then f has a unique cubic spline interpolant $s(x)$ that is defined on the nodes $x_0, x_1, \ldots, x_n$ that satisfies the natural boundary conditions $s''(a) = s''(b) = 0$.*

Theorem *Let $x_0, x_1, \ldots, x_n$ be $n + 1$ distinct points in the interval $[a, b]$, where $a = x_0 < x_1 < \cdots < x_n = b$, and let $f(x)$ be a function defined on $[a, b]$ that is differentiable at a and b. Then f has a unique cubic spline interpolant $s(x)$ that is defined on the nodes $x_0, x_1, \ldots, x_n$ that satisfies the clamped boundary conditions $s'(a) = f'(a)$ and $s'(b) = f'(b)$.*

Just as the linear spline is the "flattest" interpolant, in an average sense, the natural cubic spline with the least "average curvature". Specifically, if $s_2(x)$ is the natural cubic spline for $f \in C[a, b]$ on $[a, b]$ with knots $a = x_0 < x_1 < \cdots < x_n = b$, and $v \in H^2(a, b)$ is any interpolant of f with these knots, then

$$\|s_2''\|_2 \leq \|v''\|_2.$$

This can be proved in the same way as the corresponding result for the linear spline.

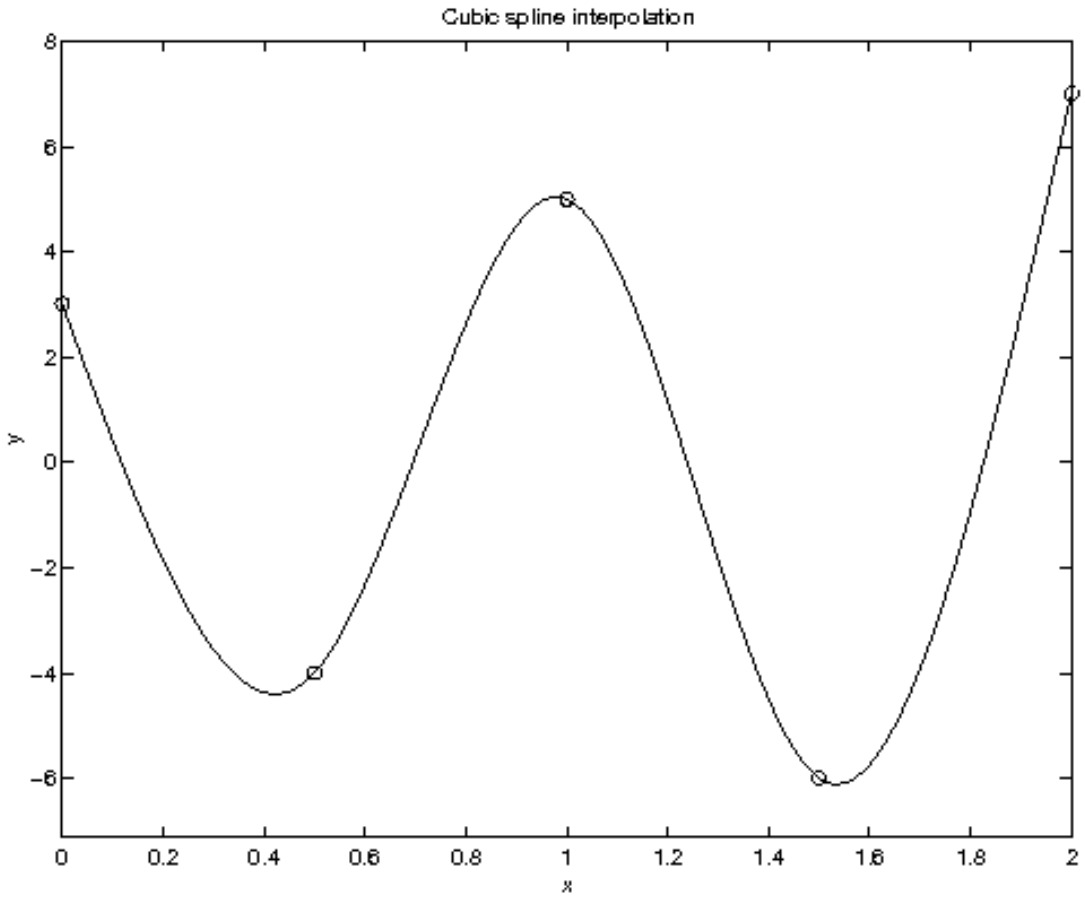

Figure 7.1: Cubic spline that passing through the points $(0, 3)$, $(1/2, -4)$, $(1, 5)$, $(2, -6)$, and $(3, 7)$.

It is this property of the natural cubic spline, called the *smoothest interpolation property*, from which splines were named. A spline is a flexible curve-drawing aid that is designed to produce a curve $y = v(x)$, $x \in [a, b]$, through prescribed points in such a way that the strain energy

$$E(v) = \int_a^b \frac{|v''(x)|^2}{(1 + |v'(x)|^2)^3}\, dx$$

is minimized over all functions that pass through the same points, which is the case if the curvature is small on $[a, b]$.

7.4 Hermite Cubic Splines

We have seen that it is possible to construct a piecewise cubic polynomial that interpolates a function $f(x)$ at knots $a = x_0 < x_1 < \cdots < x_n = b$, that belongs to $C^2[a, b]$. Now, suppose that we also know the values of $f'(x)$ at the knots. We wish to construct a piecewise cubic polynomial $s(x)$ that agrees with $f(x)$, and whose derivative agrees with $f'(x)$ at the knots. This piecewise polynomial is called a *Hermite cubic spline*.

Because $s(x)$ is cubic on each subinterval $[x_{i-1}, x_i]$ for $i = 1, 2, \ldots, n$, there are $4n$ coefficients, and therefore $4n$ degrees of freedom, that can be used to satisfy any criteria that are imposed on $s(x)$. Requiring that $s(x)$ interpolates $f(x)$ at the knots, and that $s'(x)$ interpolates $f'(x)$ at the knots, imposes $2n+2$ constraints on the coefficients. We can then use the remaining $2n - 2$ degrees of freedom to require that $s(x)$ belong to $C^1[a, b]$; that is, it is continuously differentiable on $[a, b]$.

The following result provides insight into the accuracy with which a Hermite cubic spline interpolant $s(x)$ approximates a function $f(x)$.

Theorem *Let f be four times continuously differentiable on $[a, b]$, and assume that $\|f^{(4)}\|_\infty = M$. Let $s(x)$ be the unique Hermite cubic spline interpolant of $f(x)$ on the nodes $x_0, x_1, \ldots, x_n$, where $a = x_0 < x_1 < \cdots < x_n < b$. Then*

$$\|f(x) - s(x)\|_\infty \leq \frac{5M}{384} \max_{1 \leq i \leq n} h_i^4,$$

where $h_i = x_i - x_{i-1}$.

This can be proved in the same way as the error bound for the linear spline, except that the error formula for Hermite interpolation is used instead of the error formula for Lagrange interpolation.

7.5 Basis Functions for Cubic Splines

An alternative method of computing splines to fit given data involves constructing a basis for the vector space of splines defined on the interval $[a, b]$, and then solving a system of linear equations for the coefficients of the desired spline in this basis. The basis functions are known as *B-splines*, where the letter B is due to the fact that these splines form a basis, and the fact that they tend to have bell-shaped graphs.

One advantage of using B-splines is that the system of linear equations that must be solved for the coefficients of a spline in the basis is banded, and therefore can be solved very efficiently. Furthermore, because each B-spline has compact support, it follows that a change in the data value y_i only causes the coefficients of a few B-splines to be changed, whereas in cubic spline interpolation, such a change forces all of the coefficients of each polynomial $s_i(x)$ to be recomputed.

To define a B-spline of degree n, we first introduce the following notation:

$$x_+ = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}.$$

We say that x_+ is the *positive part* of x. Then, we define a spline of degree n, centered at 0, by

$$S_{(n)}(x) = \sum_{k=0}^{n+1} (-1)^k \begin{pmatrix} n+1 \\ k \end{pmatrix} (x - kh)_+^n.$$

To see that this is in fact a spline of degree n, we first note that on each interval of the form $[ih, (i+1)h]$, for any integer i, $S_{(n)}(x)$ is a linear combination of at most $n+2$ polynomials of degree n. Furthermore, it can be determined directly each function of the form $(x-kh)_+^n$ has $n-1$ continuous derivatives.

However, $S_{(n)}(x)$ is only practically useful if it is only nonzero on a finite interval. To that end, we note that for $x < 0$, $S_{(n)}(x) = 0$, because $x - kh < 0$ for $k \geq 0$. For $x \geq kh$, we have

$$S_{(n)}(x) = \sum_{k=0}^{n+1} (-1)^k \begin{pmatrix} n+1 \\ k \end{pmatrix} (x - kh)^n.$$

It follows that

$$S_{(n)}(x) = \sum_{k=0}^{n+1} (-1)^k \begin{pmatrix} n+1 \\ k \end{pmatrix} (x - kh)^n$$

$$
\begin{aligned}
&= \sum_{k=0}^{n+1}(-1)^k \left(\begin{array}{c} n+1 \\ k \end{array} \right) \sum_{j=0}^{n} \left(\begin{array}{c} n \\ j \end{array} \right) x^j (-kh)^{n-j} \\
&= \sum_{j=0}^{n}(-h)^{n-j} \left(\begin{array}{c} n \\ j \end{array} \right) \left[\sum_{k=0}^{n+1}(-1)^k \left(\begin{array}{c} n+1 \\ k \end{array} \right) k^{n-j} \right] x^j \\
&= 0,
\end{aligned}
$$

since the quantity in square brackets, by the theory of finite differences, vanishes. This is closely related to the fact that the $(n+1)$-st derivative of a polynomial of degree n or less is equal to zero.

By scaling and shifting $S_{(n)}(x)$, we obtain basis functions that are equal to 1 at one of the knots, and 0 at all other knots, so that a spline can be expressed in a similar form as the Lagrange interpolating polynomial: a linear combination of basis functions, where the constants are the values of $f(x)$ at the interpolation points.

Chapter 8

Initial Value Problems for ODEs

8.1 Theory of Initial-Value Problems

Consider the *initial value problem*

$$y' = f(t, y), \quad t_0 \leq t \leq T, \tag{8.1}$$

$$y(t_0) = y_0 \tag{8.2}$$

We would like to have an understanding of when this problem can be solved, and whether any solution that can be obtained is unique. The following notion of continuity is helpful for this purpose.

Definition A function $f(t, y)$ satisfies a *Lipschitz condition* in y on $D \subset \mathbb{R}^2$ if

$$|f(t, y_2) - f(t, y_1)| \leq L|y_2 - y_1|, \quad (t, y_1), (t, y_2) \in D,$$

for some constant $L > 0$, which is called a *Lipschitz constant* for f.

If, in addition, $\partial f/\partial y$ exists on D, we can also conclude that $|\partial f/\partial y| \leq L$ on D.

When solving a problem numerically, it is not sufficient to know that a unique solution exists. If a small change in the problem data can cause a substantial change in the solution, then the problem is *ill-conditioned*, and a numerical solution is therefore unreliable, because it could be unduly influenced by roundoff error. The following definition characterizes problems for which numerical solution is feasible.

Definition The initial value problem (8.1), (8.2) is said to be *well-posed* if a unique solution $y(t)$ exists, and depends continuously on the data y_0 and $f(t, y)$.

We are now ready to describe a class of initial-value problems that can be solved numerically.

Theorem (Existence-Uniqueness, Well-Posedness) Let $D = [t_0, T] \times \mathbb{R}$, and let $f(t, y)$ be continuous on D. If f satisfies a Lipschitz condition on D in y, then the initial value problem (8.1), (8.2) has a unique solution $y(t)$ on $[t_0, T]$. Furthermore, the problem is well-posed.

8.2 One-Step Methods

Numerical methods for the initial-value problem (8.1), (8.2) can be developed using Taylor series. We wish to approximate the solution at times t_n, $n = 1, 2, \ldots$, where

$$t_n = t_0 + nh,$$

with h being a chosen *time step*. Taking a Taylor expansion of the exact solution $y(t)$ at $t = t_{n+1}$ around the center t_n, we obtain

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(\xi),$$

where $t_n < \xi < t_{n+1}$.

Using the fact that $y' = f(t, y)$, we obtain a numerical scheme by truncating the Taylor series after the second term. The result is a *difference equation*

$$y_{n+1} = y_n + hf(t_n, y_n),$$

where each y_n, for $n = 1, 2, \ldots$, is an approximation of $y(t_n)$. This method is called *Euler's method*, the simplest example of what is known as a *one-step method*.

We now need to determine whether this method *converges*; that is, whether

$$\lim_{h \to 0} \max_{0 \leq n \leq T/h} |y(t_n) - y_n| = 0.$$

To that end, we attempt to bound the error at time t_n. We begin with a comparison of the difference equation and the Taylor expansion of the exact solution,

$$y_{n+1} = y_n + hf(t_n, y_n)$$

$$y(t_{n+1}) \quad = \quad y(t_n) + hf(t_n, y(t_n)) + \frac{h^2}{2}y''(\xi).$$

It follows that if we define $e_n = y(t_n) - y_n$, then

$$e_{n+1} = e_n + h\frac{\partial f}{\partial y}(t_n, \eta_n)e_n + \frac{h^2}{2}y''(\xi).$$

Therefore,

$$|e_{n+1}| \leq (1 + hL)|e_n| + \frac{h^2 M}{2}, \tag{8.3}$$

where

$$|y''(t)| \leq M, \quad t_0 \leq t \leq T,$$

and L is the Lipschitz constant for f in y on $[t_0, T] \times \mathbb{R}$.

Applying the relationship (8.3) repeatedly yields

$$\begin{aligned} |e_n| &\leq (1 + hL)^n |e_0| + \frac{h^2 M}{2} \sum_{i=0}^{n-1} (1 + hL)^i \\ &\leq \frac{h^2 M}{2} \frac{(1 + hL)^n - 1}{(1 + hL) - 1} \\ &\leq \frac{h^2 M}{2} \frac{[e^{hL}]^n - 1}{hL} \\ &\leq \frac{hM}{2L}[e^{L(t_n - t_0)} - 1]. \end{aligned}$$

We conclude that for $t_0 \leq t_n \leq T$,

$$|y(t_n) - y_n| \leq \frac{hM}{2L}[e^{L(t_n - t_0)} - 1] \leq \frac{hM}{2L}[e^{L(T - t_0)} - 1].$$

That is, as $h \to 0$, the solution obtained using Euler's method converges to the exact solution, and the convergence is $O(h)$; that is, first-order.

This convergence analysis, however, assumes exact arithmetic. To properly account for roundoff error, we note that the approximate solution values $\tilde{y}_n$, $n = 0, 1, 2, \ldots$, satisfy the modified difference equation

$$\tilde{y}_{n+1} = \tilde{y}_n + hf(t_n, \tilde{y}_n) + \delta_{n+1}, \quad \tilde{y}_0 = y_0 + \delta_0,$$

where, for $n = 0, 1, 2, \ldots$, $|\delta_n| \leq \delta$, which is $O(\mathbf{u})$, where $\mathbf{u}$ is the the machine precision (i.e., unit roundoff). Note that even the initial value $\tilde{y}_0$ has an error term, which arises from representation of y_0 in the floating-point system.

Repeating the convergence analysis yields the error bound

$$|y(t_n) - \tilde{y}_n| \leq \frac{1}{L}\left(\frac{hM}{2} + \frac{\delta}{h}\right)[e^{L(t_n - t_0)} - 1] + \delta e^{L(t_n - t_0)}.$$

We see that as $h \to 0$, the error actually *increases*, but an optimal value for h can be chosen to minimize this error and also keep it sufficiently small, provided that δ is sufficiently small.

We conclude our discussion of Euler's method with an example of how the previous convergence analyses can be used to select a suitable time step h.

Example Consider the IVP

$$y' = -y, \quad 0 < t < 10, \quad y(0) = 1.$$

We know that the exact solution is $y(t) = e^{-t}$. Euler's method applied to this problem yields the difference equation

$$y_{n+1} = y_n - hy_n = (1 - h)y_n, \quad y_0 = 1.$$

We wish to select h so that the error at time $T = 10$ is less than 0.001. To that end, we use the error bound

$$|y(t_n) - y_n| \leq \frac{hM}{2L}[e^{L(t_n - t_0)} - 1],$$

with $M = 1$, since $y''(t) = e^{-t}$, which satisfies $0 < y''(t) < 1$ on $[0, 10]$, and $L = 1$, since $f(t, y) = -y$ satisfies $|\partial f/\partial y| = |-1| \equiv 1$. Substituting $t_n = 10$ and $t_0 = 0$ yields

$$|y(10) - y_n| \leq \frac{h}{2}[e^{10} - 1] \approx 1101.27h.$$

Ensuring that the error at this time is less than 10^{-3} requires choosing $h < 9.08 \times 10^{-8}$. However, the bound on the error at $t = 10$ is quite crude. Applying Euler's method with this time step yields a solution whose error at $t = 10$ is 2×10^{-11}.

Now, suppose that we include roundoff error in our error analysis. The optimal time step is

$$h = \sqrt{\frac{2\delta}{M}},$$

where δ is a bound on the roundoff error during any time step. We use $\delta = 2\mathbf{u}$, where $\mathbf{u}$ is the unit roundoff, because each time step performs only

two floating-point operations. Even if $1 - h$ is computed once, in advance, its error still propagates to the multiplication with y_n. In a typical double-precision floating-point number system, $\mathbf{u} \approx 1.1 \times 10^{-16}$. It follows that the optimal time step is

$$h = \sqrt{\frac{2\delta}{M}} = \sqrt{\frac{2(1.1 \times 10^{-16})}{1}} \approx 1.5 \times 10^{-8}.$$

With this value of h, we find that the error at $t = 10$ is approximately 3.55×10^{-12}. This is even more accurate than with the previous choice of time step, which makes sense, because the new value of h is smaller. □

8.3 Consistency and Convergence

We have learned that the numerical solution obtained from Euler's method,

$$y_{n+1} = y_n + hf(t_n, y_n), \quad t_n = t_0 + nh,$$

converges to the exact solution $y(t)$ of the initial value problem

$$y' = f(t, y), \quad y(t_0) = y_0,$$

as $h \to 0$.

We now analyze the convergence of a general one-step method of the form

$$y_{n+1} = y_n + h\Phi(t_n, y_n, h),$$

for some continuous function $\Phi(t, y, h)$. We define the *local truncation error* of this one-step method by

$$T_n(h) = \frac{y(t_{n+1}) - y(t_n)}{h} - \Phi(t_n, y(t_n), h).$$

That is, the local truncation error is the result of substituting the exact solution into the approximation of the ODE by the numerical method.

As $h \to 0$ and $n \to \infty$, in such a way that $t_0 + nh = t \in [t_0, T]$, we obtain

$$T_n(h) \to y'(t) - \Phi(t, y(t), 0).$$

We therefore say that the one-step method is *consistent* if

$$\Phi(t, y, 0) = f(t, y).$$

A consistent one-step method is one that converges to the ODE as $h \to 0$.

We then say that a one-step method is *stable* if $\Phi(t, y, h)$ is Lipschitz continuous in y. That is,

$$|\Phi(t, u, h) - \Phi(t, v, h)| \leq L_\Phi |u - v|, \quad t \in [t_0, T], \quad u, v \in \mathbb{R}, \quad h \in [0, h_0],$$

for some constant L_Φ.

We now show that a consistent and stable one-step method is convergent. Using the same approach and notation as in the convergence proof of Euler's method, and the fact that the method is stable, we obtain the following bound for the *global error* $e_n = y(t_n) - y_n$:

$$|e_n| \leq \left(\frac{e^{L_\Phi (T - t_0)} - 1}{L_\Phi} \right) \max_{0 \leq m \leq n-1} |T_m(h)|.$$

Because the method is consistent, we have

$$\lim_{h \to 0} \max_{0 \leq n \leq T/h} |T_n(h)| = 0.$$

It follows that as $h \to 0$ and $n \to \infty$ in such a way that $t_0 + nh = t$, we have

$$\lim_{n \to \infty} |e_n| = 0,$$

and therefore the method is convergent.

In the case of Euler's method, we have

$$\Phi(t, y, h) = f(t, y), \quad T_n(h) = \frac{h}{2} f''(\tau), \quad \tau \in (t_0, T).$$

Therefore, there exists a constant K such that

$$|T_n(h)| \leq Kh, \quad 0 < h \leq h_0,$$

for some sufficiently small h_0. We say that Euler's method is *first-order accurate*. More generally, we say that a one-step method has *order of accuracy* p if, for any sufficiently smooth solution $y(t)$, there exists constants K and h_0 such that

$$|T_n(h)| \leq Kh^p, \quad 0 < h \leq h_0.$$

We now consider an example of a higher-order accurate method.

8.4 An Implicit One-Step Method

Suppose that we approximate the equation

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} y'(s)\,ds$$

by applying the Trapezoidal Rule to the integral. This yields a one-step method

$$y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1})],$$

known as the *trapezoidal method*.

It follows from the error in the Trapezoidal Rule that

$$T_n(h) = \frac{y(t_{n+1}) - y(t_n)}{h} - \frac{1}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1})] = -\frac{1}{12}h^2 y'''(\tau_n), \quad \tau_n \in (t_n, t_{n+1}).$$

Therefore, the trapezoidal method is *second-order accurate*.

To show convergence, we must establish stability by finding a suitable Lipschitz constant L_Φ for the function

$$\Phi(t, y, h) = \frac{1}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1})],$$

assuming that L_f is a Lipschitz constant for $f(t, y)$ in y. We have

$$\begin{aligned} |\Phi(t, u, h) - \Phi(t, v, h)| &= \frac{1}{2}|f(t, u) + f(t + h, u + h\Phi(t, u, h)) - f(t, v) - f(t + h, v + h\Phi(t, v, h)| \\ &\leq L_f|u - v| + \frac{h}{2}L_f|\Phi(t, u, h) - \Phi(t, v, h)|. \end{aligned}$$

Therefore

$$\left(1 - \frac{h}{2}L_f\right)|\Phi(t, u, h) - \Phi(t, v, h) \leq L_f|u - v|,$$

and therefore

$$L_\Phi \leq \frac{L_f}{1 - \frac{h}{2}L_f},$$

provided that $\frac{h}{2}L_f < 1$. We conclude that for h sufficiently small, the trapezoidal method is stable, and therefore convergent, with $O(h^2)$ global error.

The trapezoidal method constrasts with Euler's method because it is an *implicit* method, due to the evaluation of $f(t, y)$ at y_{n+1}. It follows that

it is generally necessary to solve a nonlinear equation to obtain y_{n+1} from y_n. This additional computational effort is offset by the fact that implicit methods are generally more stable than *explicit* methods such as Euler's method. Another example of an implicit method is *backward Euler's method*

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}).$$

Like Euler's method, backward Euler's method is first-order accurate.

8.5 Runge-Kutta Methods

We have seen that Euler's method is first-order accurate. We would like to use Taylor series to design methods that have a higher order of accuracy. First, however, we must get around the fact that an analysis of the global error, as was carried out for Euler's method, is quite cumbersome. Instead, we will design new methods based on the criteria that their local truncation error, the error committed during a single time step, is higher-order in h.

Using higher-order Taylor series directly to approximate $y(t_{n+1})$ is cumbersome, because it requires evaluating derivatives of f. Therefore, our approach will be to use evaluations of f at carefully chosen values of its arguments, t and y, in order to create an approximation that is just as accurate as a higher-order Taylor series expansion of $y(t+h)$.

To find the right values of t and y at which to evaluate f, we need to take a Taylor expansion of f evaluated at these (unknown) values, and then *match* the resulting numerical scheme to a Taylor series expansion of $y(t+h)$ around t. To that end, we state a generalization of Taylor's theorem to functions of two variables.

Theorem Let $f(t,y)$ be $(n+1)$ times continuously differentiable on a convex set D, and let $(t_0, y_0) \in D$. Then, for every $(t,y) \in D$, there exists ξ between t_0 and t, and μ between y_0 and y, such that

$$f(t,y) = P_n(t,y) + R_n(t,y),$$

where $P_n(t,y)$ is the *nth Taylor polynomial* of f about (t_0, y_0),

$$\begin{aligned} P_n(t,y) &= f(t_0,y_0) + \left[(t-t_0)\frac{\partial f}{\partial t}(t_0,y_0) + (y-y_0)\frac{\partial f}{\partial y}(t_0,y_0)\right] + \\ &\quad \left[\frac{(t-t_0)^2}{2}\frac{\partial^2 f}{\partial t^2}(t_0,y_0) + (t-t_0)(y-y_0)\frac{\partial^2 f}{\partial t \partial y}(t_0,y_0) + \frac{(y-y_0)^2}{2}\frac{\partial^2 f}{\partial y^2}(t_0,y_0)\right] + \\ &\quad \cdots + \left[\frac{1}{n!}\sum_{j=0}^{n}\binom{n}{j}(t-t_0)^{n-j}(y-y_0)^j\frac{\partial^n f}{\partial t^{n-j}\partial y^j}(t_0,y_0)\right], \end{aligned}$$

and $R_n(t, y)$ is the *remainder term* associated with $P_n(t, y)$,

$$R_n(t,y) = \frac{1}{(n+1)!} \sum_{j=0}^{n+1} \left(\begin{array}{c} n+1 \\ j \end{array} \right) (t-t_0)^{n+1-j} (y-y_0)^j \frac{\partial^{n+1} f}{\partial t^{n+1-j} \partial y^j}(\xi, \mu).$$

We now illustrate our proposed approach in order to obtain a method that is second-order accurate; that is, its local truncation error is $O(h^2)$. This involves matching

$$y + hf(t,y) + \frac{h^2}{2}\frac{d}{dt}[f(t,y)] + \frac{h^3}{6}\frac{d^2}{dt^2}[f(\xi,y)]$$

to

$$y + ha_1 f(t + \alpha_1, y + \beta_1),$$

where $t \leq \xi \leq t + h$ and the parameters a_1, α_1 and β_1 are to be determined. After simplifying by removing terms or factors that already match, we see that we only need to match

$$f(t,y) + \frac{h}{2}\frac{d}{dt}[f(t,y)] + \frac{h^2}{6}\frac{d^2}{dt^2}[f(t,y(t))]$$

with

$$a_1 f(t + \alpha_1, y + \beta_1),$$

at least up to terms of $O(h)$, so that the local truncation error will be $O(h^2)$.

Applying the multivariable version of Taylor's theorem to f, we obtain

$$\begin{aligned} a_1 f(t + \alpha_1, y + \beta_1) &= a_1 f(t,y) + a_1\alpha_1 \frac{\partial f}{\partial t}(t,y) + a_1\beta_1 \frac{\partial f}{\partial y}(t,y) + \\ &\quad \frac{\alpha_1^2}{2}\frac{\partial^2 f}{\partial t^2}(\xi,\mu) + \alpha_1\beta_1 \frac{\partial^2 f}{\partial t \partial y}(\xi,\mu) + \frac{\beta_1^2}{2}\frac{\partial^2 f}{\partial y^2}(\xi,\mu), \end{aligned}$$

where ξ is between t and $t + \alpha_1$ and μ is between y and $y + \beta_1$. Meanwhile, computing the full derivatives with respect to t in the Taylor expansion of the solution yields

$$f(t,y) + \frac{h}{2}\frac{\partial f}{\partial t}(t,y) + \frac{h}{2}\frac{\partial f}{\partial y}(t,y) f(t,y) + O(h^2).$$

Comparing terms yields the equations

$$a_1 = 1, \quad a_1\alpha_1 = \frac{h}{2}, \quad a_1\beta_1 = \frac{h}{2} f(t,y),$$

which has the solutions

$$a_1 = 1, \quad \alpha_1 = \frac{h}{2}, \quad \beta_1 = \frac{h}{2} f(t, y).$$

The resulting numerical scheme is

$$y_{n+1} = y_n + hf\left(t_n + \frac{h}{2}, y_n + \frac{h}{2} f(t_n, y_n)\right).$$

This scheme is known as the *midpoint method*, or the *explicit midpoint method*. Note that it evaluates f at the midpoint of the intervals $[t_n, t_{n+1}]$ and $[y_n, y_{n+1}]$, where the midpoint in y is approximated using Euler's method with time step $h/2$.

The midpoint method is the simplest example of a *Runge-Kutta method*, which is the name given to any of a class of time-stepping schemes that are derived by matching multivaraiable Taylor series expansions of $f(t, y)$ with terms in a Taylor series expansion of $y(t + h)$. Another often-used Runge-Kutta method is the *modified Euler method*

$$y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, y_n + hf(t_n, y_n)],$$

which resembles the Trapezoidal Rule from numerical integration, and is also second-order accurate.

However, the best-known Runge-Kutta method is the *fourth-order Runge-Kutta method*, which uses *four* evaluations of f during each time step. The method proceeds as follows:

$$\begin{aligned}
k_1 &= hf(t_n, y_n), \\
k_2 &= hf\left(t_n + \frac{h}{2}, y_n + \frac{1}{2}k_1\right), \\
k_3 &= hf\left(t_n + \frac{h}{2}, y_n + \frac{1}{2}k_2\right), \\
k_4 &= hf\left(t_{n+1}, y_n + k_3\right), \\
y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}$$

In a sense, this method is similar to Simpson's Rule from numerical integration, which is also fourth-order accurate, as values of f at the midpoint in time are given four times as much weight as values at the endpoints t_n and t_{n+1}.

Example We compare Euler's method with the fourth-order Runge-Kutta scheme on the initial value problem

$$y' = -2ty, \quad 0 < t \leq 1, \quad y(0) = 1,$$

which has the exact solution $y(t) = e^{-t^2}$. We use a time step of $h = 0.1$ for

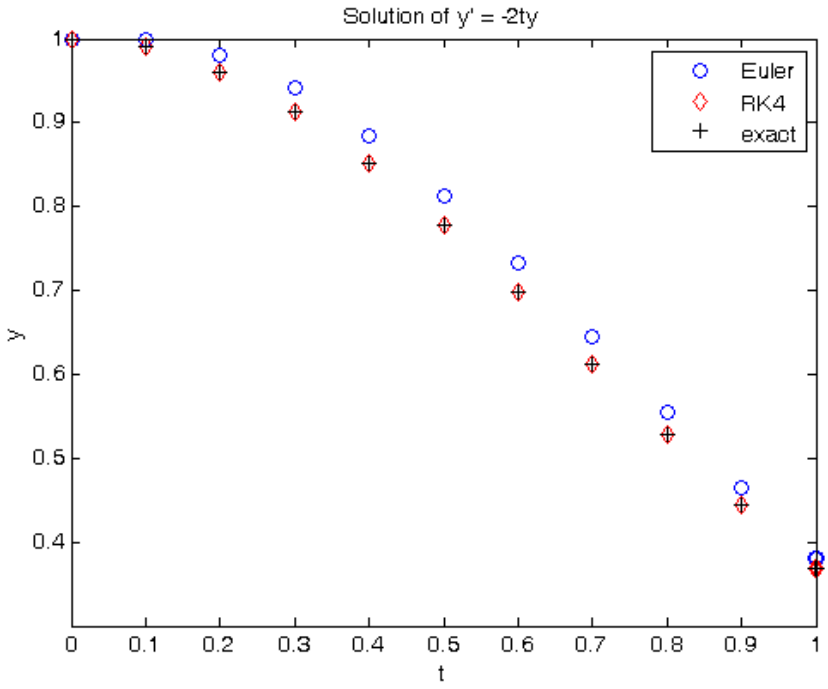

Figure 8.1: Solutions of $y' = -2ty$, $y(0) = 1$ on $[0, 1]$, computed using Euler's method and the fourth-order Runge-Kutta method

both methods. The computed solutions, and the exact solution, are shown in Figure 8.1.

It can be seen that the fourth-order Runge-Kutta method is far more accurate than Euler's method, which is first-order accurate. In fact, the solution computed using the fourth-order Runge-Kutta method is visually indistinguishable from the exact solution. At the final time $T = 1$, the relative error in the solution computed using Euler's method is 0.038, while the relative error in the solution computing using the fourth-order Runge-Kutta method is 4.4×10^{-6}. □

8.6 Multistep Methods

All of the numerical methods that we have developed for solving initial value problems are *one-step methods*, because they only use information about the solution at time t_n to approximate the solution at time t_{n+1}. As n increases, that means that there are additional values of the solution, at previous times, that could be helpful, but are unused.

Multistep methods are time-stepping methods that do use this information. A general multistep method has the form

$$\sum_{i=0}^{s} \alpha_i y_{n+1-i} = h \sum_{i=0}^{s} \beta_i f(t_{n+1-i}, y_{n+1-i}),$$

where s is the number of steps in the method ($s = 1$ for a one-step method), and h is the time step size, as before.

By convention, $\alpha_0 = 1$, so that y_{n+1} can be conveniently expressed in terms of other values. If $\beta_0 = 0$, the multistep method is said to be *explicit*, because then y_{n+1} can be described using an explicit formula, whereas if $\beta_0 \neq 0$, the method is *implicit*, because then an equation, generally nonlinear, must be solved to compute y_{n+1}.

To ensure that the multistep method is *consistent* with the underlying ODE $y' = f(t, y)$, for general f, it is necessary that the constants α_i satisfy the constraints

$$\sum_{i=0}^{s} \alpha_i = 0.$$

This ensures that if $f(t, y) \equiv 0$, a constant solution is produced by the numerical scheme, and as $h \to 0$, the numerical method converges to the ODE itself.

Most multistep methods fall into two broad categories:

- *Adams methods*: these involve the integral form of the ODE,

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(s, y(s))\, ds.$$

  The general idea behind Adams methods is to approximate the above integral using polynomial interpolation of f at the points $t_{n+1-s}, t_{n+2-s}, \ldots, t_n$ if the method is explicit, and t_{n+1} as well if the method is implicit. In all Adams methods, $\alpha_0 = 1$, $\alpha_1 = -1$, and $\alpha_i = 0$ for $i = 2, \ldots, s$.

- *Backward differentiation formulas (BDF)*: these methods also use polynomial interpolation, but for a different purpose–to approximate the *derivative* of y at t_{n+1}. This approximation is then equated to $f(t_{n+1}, y_{n+1})$. It follows that all methods based on BDFs are implicit, and they all satisfy $\beta_0 = 1$, with $\beta_i = 0$ for $i = 1, 2, \ldots, s$.

In our discussion, we will focus exclusively on Adams methods.

Explicit Adams methods are called *Adams-Bashforth methods*. To derive an Adams-Bashforth method, we interpolate f at the points $t_n, t_{n-1}, \ldots, t_{n-s+1}$ with a polynomial of degree $s-1$. We then integrate this polynomial exactly. It follows that the constants β_i, $i = 1, \ldots, s$, are the integrals of the corresponding Lagrange polynomials from t_n to t_{n+1}, divided by h, because there is already a factor of h in the general multistep formula.

The local truncation error of the resulting method is $O(h^s)$. To see this, we first note that because all of the interpolation points have a spacing of h, and the degree of the interpolating polynomial is $s-1$, the interpolation error is $O(h^s)$. Integrating this error over an interval of width h yields an error in y_{n+1} that is $O(h^{s+1})$, but because the definition of local truncation error calls for dividing both sides of the difference equation by h, the local truncation error turns out to be $O(h^s)$.

Example We derive the three-step Adams-Bashforth method,

$$y_{n+1} = y_n + h[\beta_1 f(t_n, y_n) + \beta_2 f(t_{n-1}, y_{n-1}) + \beta_3 f(t_{n-2}, y_{n-2}).$$

The constants β_i, $i = 1, 2, 3$, are obtained by evaluating the integral from t_n to t_{n+1} of a polynomial $p_2(t)$ that passes through $f(t_n, y_n)$, $f(t_{n-1}, y_{n-1})$, and $f(t_{n-2}, y_{n-2})$.

Because we can write

$$p_2(t) = \sum_{i=0}^{2} f(t_{n-i}, y_{n-i}) L_i(t),$$

where $L_i(t)$ is the ith Lagrange polynomial for the interpolation points t_n, t_{n-1} and t_{n-2}, and because our final method expresses y_{n+1} as a linear combination of y_n and values of f, it follows that the constants β_i, $i = 1, 2, 3$, are the integrals of the Lagrange polynomials from t_n to t_{n+1}, divided by h.

However, using a change of variable $u = (t_{n+1} - s)/h$, we can instead interpolate at the points $u = 1, 2, 3$, thus simplifying the integration. If we define $\tilde{p}_2(u) = p_2(s) = p_2(t_{n+1} - hu)$ and $\tilde{L}_i(u) = L_i(t_{n+1} - hu)$, then we

obtain

$$\begin{aligned}
\int_{t_n}^{t_{n+1}} f(s, y(s))\, ds &= \int_{t_n}^{t_{n+1}} p_2(s)\, ds \\
&= h \int_0^1 \tilde{p}_2(u)\, du \\
&= h \int_0^1 f(t_n, y_n)\tilde{L}_0(u) + f(t_{n-1}, y_{n-1})\tilde{L}_1(u) + f(t_{n-2}, y_{n-2})\tilde{L}_2(u)\, du \\
&= h \left[f(t_n, y_n) \int_0^1 \tilde{L}_0(u)\, du + f(t_{n-1}, y_{n-1}) \int_0^1 \tilde{L}_1(u)\, du + \right. \\
&\quad \left. f(t_{n-2}, y_{n-2}) \int_0^1 \tilde{L}_2(u)\, du \right] \\
&= h \left[f(t_n, y_n) \int_0^1 \frac{(u-2)(u-3)}{(1-2)(1-3)}\, du + f(t_{n-1}, y_{n-1}) \int_0^1 \frac{(u-1)(u-3)}{(2-1)(2-3)}\, du + \right. \\
&\quad \left. f(t_{n-2}, y_{n-2}) \int_0^1 \frac{(u-1)(u-2)}{(3-1)(3-2)}\, du \right] \\
&= h \left[\frac{23}{12} f(t_n, y_n) - \frac{4}{3} f(t_{n-1}, y_{n-1}) + \frac{5}{12} f(t_{n-2}, y_{n-2}) \right].
\end{aligned}$$

We conclude that the three-step Adams-Bashforth method is

$$y_{n+1} = y_n + \frac{h}{12}[23f(t_n, y_n) - 16f(t_{n-1}, y_{n-1}) + 5f(t_{n-2}, y_{n-2})].$$

This method is third-order accurate. □

The same approach can be used to derive an implicit Adams method, which is known as an *Adams-Moulton method*. The only difference is that because t_{n+1} is an interpolation point, after the change of variable to u, the interpolation points $0, 1, 2, \ldots, s$ are used. Because the resulting interpolating polynomial is of degree one greater than in the explicit case, the error in an s-step Adams-Moulton method is $O(h^{s+1})$, as opposed to $O(h^s)$ for an s-step Adams-Bashforth method.

An Adams-Moulton method can be impractical because, being implicit, it requires an iterative method for solving nonlinear equations, such as fixed-point iteration, and this method must be applied during every time step. An alternative is to pair an Adams-Bashforth method with an Adams-Moulton method to obtain an *Adams-Moulton predictor-corrector method*. Such a method proceeds as follows:

- *Predict:* Use the Adams-Bashforth method to compute a first approximation to y_{n+1}, which we denote by $\tilde{y}_{n+1}$.

- *Evaluate:* Evaluate f at this value, computing $f(t_{n+1}, \tilde{y}_{n+1})$.
- *Correct:* Use the Adams-Moulton method to compute y_{n+1}, but instead of solving an equation, use $f(t_{n+1}, \tilde{y}_{n+1})$ in place of $f(t_{n+1}, y_{n+1})$ so that the Adams-Moulton method can be used as if it was an *explicit* method.
- *Evaluate:* Evaluate f at the newly computed value of y_{n+1}, computing $f(t_{n+1}, y_{n+1})$, to use during the next time step.

Example We illustrate the predictor-corrector approach with the two-step Adams-Bashforth method

$$y_{n+1} = y_n + \frac{h}{2}[3f(t_n, y_n) - f(t_{n-1}, y_{n-1})]$$

and the two-step Adams-Moulton method

$$y_{n+1} = y_n + \frac{h}{12}[5f(t_{n+1}, y_{n+1}) + 8f(t_n, y_n) - f(t_{n-1}, y_{n-1})].$$

First, we apply the Adams-Bashforth method, and compute

$$\tilde{y}_{n+1} = y_n + \frac{h}{2}[3f(t_n, y_n) - f(t_{n-1}, y_{n-1})].$$

Then, we compute $f(t_{n+1}, \tilde{y}_{n+1})$ and apply the Adams-Moulton method, to compute

$$y_{n+1} = y_n + \frac{h}{12}[5f(t_{n+1}, \tilde{y}_{n+1}) + 8f(t_n, y_n) - f(t_{n-1}, y_{n-1})].$$

This new value of y_{n+1} is used when evaluating $f(t_{n+1}, y_{n+1})$ during the next time step. □

One drawback of multistep methods is that because they rely on values of the solution from previous time steps, they cannot be used during the first time steps, because not enough values are available. Therefore, it is necessary to use a one-step method, with the same order of accuracy, to compute enough *starting values* of the solution to be able to use the multistep method. For example, to use the three-step Adams-Bashforth method, it is necessary to first use a one-step method such as the fourth-order Runge-Kutta method to compute y_1 and y_2, and then the Adams-Bashforth method can be used to compute y_3 using y_2, y_1 and y_0.

8.7 Consistency and Zero-Stability

For multistep methods, the notion of convergence is exactly the same as for one-step methods. However, we must define consistency and stability slightly differently, because we must account for the fact that a multistep method requires starting values that are computed using another method.

Therefore, we say that a multistep method is consistent if its own local truncation error $T_n(h)$ approaches zero as $h \to 0$, *and* if the one-step method used to compute its starting values is also consistent. We also say that a s-step multistep method is stable, or *zero-stable*, if there exists a constant K such that for any two sequences of values $\{y_k\}$ and $\{z_k\}$ produced by the method with step size h from different sets of starting values $\{y_0, y_1, \ldots, y_{s-1}\}$ and $\{z_0, z_1, \ldots, z_{s-1}\}$,

$$|y_n - z_n| \leq K \max_{0 \leq j \leq s-1} |y_j - z_j|,$$

as $h \to 0$.

To compute the local truncation error of Adams methods, integrate the error in the polynomial interpolation used to derive the method from t_n to t_{n+1}. For the explicit s-step method, this yields

$$T_n(h) = \frac{1}{h} \int_{t_n}^{t_{n+1}} \frac{f^{(s)}(\xi, y(\xi))}{s!}(t - t_n)(t - t_{n-1}) \cdots (t - t_{n-s+1})\, dt.$$

Using the substitution $u = (t_{n+1} - t)/h$, and the Weighted Mean Value Theorem for Integrals, yields

$$T_n(h) = \frac{1}{h} \frac{f^{(s)}(\xi, y(\xi))}{s!} h^{s+1} (-1)^s \int_0^1 (u - 1)(u - 2) \cdots (u - s)\, du.$$

Evaluating the integral yields the constant in the error term. We also use the fact that $y' = f(t, y)$ to replace $f^{(s)}(\xi, y(\xi))$ with $y^{(s+1)}(\xi)$. Obtaining the local truncation error for an implicit, Adams-Moulton method can be accomplished in the same way, except that t_{n+1} is also used as an interpolation point.

For a general multistep method, we substitute the exact solution into the method, as in one-step methods, and obtain

$$T_n(h) = \frac{\sum_{j=0}^{s} \alpha_j y(t_{n+1-j}) - h \sum_{j=0}^{s} \beta_j f(t_{n+1-j}, y(t_{n+1-j}))}{h \sum_{j=0}^{s} \beta_j},$$

where the scaling by $h \sum_{j=0}^{s} \beta_j$ is designed to make this definition of local truncation error consistent with that of one-step methods.

By replacing each evaluation of $y(t)$ by a Taylor series expansion around t_n, we find that $T_n(h) \to 0$ as $h \to 0$ only if

$$\sum_{j=0}^{s} \alpha_j = 0, \quad \sum_{j=0}^{s-1} (s-j)\alpha_j - \sum_{j=0}^{s} \beta_j = 0.$$

In other words, we must have

$$\rho(1) = 0, \quad \rho'(1) = \sigma(1) \neq 0,$$

where

$$\rho(z) = \sum_{j=0}^{s} \alpha_j z^{s-j}, \quad \sigma(z) = \sum_{j=0}^{s} \beta_j z^{s-j}$$

are the *first* and *second characteristic polynomials*, respectively, of the multistep method.

However, further analysis is required to obtain the local truncation error of a predictor-corrector method that is obtained by combining two Adams methods. The result of this analysis is the following theorem.

Theorem Let the solution of the initial value problem

$$y' = f(t, y), \quad t_0 < t \leq T, \quad y(t_0) = y_0$$

be approximated by the Adams-Moulton s-step predictor-corrector method with predictor

$$\tilde{y}_{n+1} = y_n + h \sum_{i=1}^{s} \tilde{\beta}_i f_{n+1-i}$$

and corrector

$$y_{n+1} = y_n + h \left[\beta_0 f(t_{n+1}, \tilde{y}_{n+1}) + \sum_{i=1}^{s} \beta_i f_{n+1-i} \right].$$

Then the local truncation error of the predictor-corrector method is

$$S_n(h) = \tilde{T}_n(h) + T_n(h)\beta_0 \frac{\partial f}{\partial y}(t_{n+1}, y(t_{n+1}) + \xi_{n+1})$$

where $\tilde{T}_n(h)$ and $T_n(h)$ are the local truncation errors of the predictor and corrector, respectively, and ξ_{n+1} is between 0 and $hT_n(h)$. Furthermore, there exist constant α and β such that

$$|y(t_n) - y_n| \leq \left[\max_{0 \leq i \leq s-1} |y(t_i) - y_i| + \beta S(h) \right] e^{\alpha(t_n - t_0)},$$

where $S(h) = \max_{s \leq n \leq (T-t_0)/h} |S_n(h)|$. It follows that the predictor-corrector method is convergent if it is consistent.

It is interesting to note that the preceding theorem only requires consistency for convergence, as opposed to both consistency and stability. This is because such a predictor-corrector method is already guaranteed to be stable. To see this, we examine the stability of a general s-step multistep method of the form

$$\sum_{i=0}^{s} \alpha_i y_{n+1-i} = h \sum_{i=0}^{s} \beta_i f(t_{n+1-i}, y_{n+1-i}).$$

If this method is applied to the initial value problem

$$y' = 0, \quad y(t_0) = y_0, \quad y_0 \neq 0,$$

for which the exact solution is $y(t) = y_0$, then for the method to be stable, the computed solution must remain bounded.

It follows that the computed solution satisfies the m-term recurrence relation

$$\sum_{i=0}^{s} \alpha_i y_{n+1-i} = 0,$$

which has a solution of the form

$$y_n = \sum_{i=0}^{s} c_i n^{p_i} \lambda_i^n,$$

where the c_i and p_i are constants, and the λ_i are the roots of the *characteristic equation*

$$\alpha_0 \lambda^s + \alpha_1 \lambda^{s-1} + \cdots + \alpha_{s-1} \lambda + \alpha_s = 0.$$

When a root λ_i is distinct, $p_i = 0$. Therefore, to ensure that the solution does not grow exponentially, the method must satisfy the *root condition:*

- All roots must satisfy $|\lambda_i| \leq 1$.
- If $|\lambda_i| = 1$ for any i, then it must be a *simple root*, meaning that its multiplicity is one.

It can be shown that a multistep method is zero-stable if and only if it satisfies the root condition. Furthermore, $\lambda = 1$ is always a root, because in order to be consistent, a multistep method must have the property that $\sum_{i=0}^{s} \alpha_i = 0$. If this is the only root that has absolute value 1, then we say

that the method is *strongly stable*, whereas if there are multiple roots that are distinct from one another, but have absolute value 1, then the method is said to be *weakly stable*.

Because all Adams methods have the property that $\alpha_0 = 1$, $\alpha_1 = -1$, and $\alpha_i = 0$ for $i = 2, 3, \ldots, s$, it follows that the roots of the characteristic equation are all zero, except for one root that is equal to 1. Therefore, all Adams methods are strongly stable. This explains why only consistency is necessary to ensure convergence. In general, a consistent multistep method is convergent if and only if it is zero-stable.

Example A multistep method that is neither an Adams method, nor a backward differentiation formula, is an implicit 2-step method known as *Simpson's method*:

$$y_{n+1} = y_{n-1} + \frac{h}{3}[f_{n+1} + 4f_n + f_{n-1}].$$

Although it is only a 2-step method, it is fourth-order accurate, due to the high degree of accuracy of Simpson's Rule.

This method is obtained from the relation satisfied by the exact solution,

$$y(t_{n+1}) = y(t_{n-1}) + \int_{t_{n-1}}^{t_{n+1}} f(t, y(t))\, dt.$$

Since the integral is over an interval of width $2h$, it follows that the coefficients β_i obtained by polynomial interpolation of f must satisfy the condition

$$\sum_{i=0}^{s} \beta_i = 2,$$

as opposed to summing to 1 for Adams methods.

For this method, we have $s = 2$, $\alpha_0 = 1$, $\alpha_1 = 0$ and $\alpha_2 = -1$, which yields the characteristic polynomial $\lambda^2 - 1$. This polymomial has two distinct roots, 1 and -1, that both have absolute value 1. It follows that Simpson's method is only weakly stable. □

8.8 Stiff Differential Equations

To this point, we have evaluated the accuracy of numerical methods for initial-value problems in terms of the rate at which the error approaches zero, when the step size h approaches zero. However, this characterization of accuracy is not always informative, because it neglects the fact that the

local truncation error of any one-step or multistep method also depends on higher-order derivatives of the solution. In some cases, these derivatives can be quite large in magnitude, even when the solution itself is relatively small, which requires that h be chosen particularly small in order to achieve even reasonable accuracy.

This leads to the concept of a *stiff* differential equation. A differential equation of the form $y' = f(t, y)$ is said to be *stiff* if its exact solution $y(t)$ includes a term that decays exponentially to zero as t increases, but whose derivatives are much greater in magnitude than the term itself. An example of such a term is e^{-ct}, where c is a large, positive constant, because its kth derivative is $c^k e^{-ct}$. Because of the factor of c^k, this derivative decays to zero much more slowly than e^{-ct} as t increases. Because the error includes a term of this form, evaluated at a time less than t, the error can be quite large if h is not chosen sufficiently small to offset this large derivative. Furthermore, the larger c is, the smaller h must be to maintain accuracy.

Example Consider the initial value problem

$$y' = -100y, \quad t > 0, \quad y(0) = 1.$$

The exact solution is $y(t) = e^{-100t}$, which rapidly decays to zero as t increases. If we solve this problem using Euler's method, with step size $h = 0.1$, then we have

$$y_{n+1} = y_n - 100hy_n = -9y_n,$$

which yields the exponentially *growing* solution $y_n = (-9)^n$. On the other hand, if we choose $h = 10^{-3}$, we obtain the computed solution $y_n = (0.9)^n$, which is much more accurate, and correctly captures the qualitative behavior of the exact solution, in that it rapidly decays to zero. □

The ODE in the preceding example is a special case of the *test equation*

$$y' = \lambda y, \quad y(0) = 1, \quad \operatorname{Re}\lambda < 0.$$

The exact solution to this problem is $y(t) = e^{\lambda t}$. However, as λ increases in magnitude, the problem becomes increasingly stiff. By applying a numerical method to this problem, we can determine how small h must be, for a given value of λ, in order to obtain a qualitatively accurate solution.

When applying a one-step method to the test equation, the computed solution has the form

$$y_{n+1} = Q(h\lambda)y_n,$$

where $Q(h\lambda)$ is a polynomial in $h\lambda$ if the method is explicit, and a rational function if it is implicit. This polynomial is meant to approximate $e^{h\lambda}$, since the exact solution satisfies $y(t_{n+1}) = e^{h\lambda}y(t_n)$. However, to obtain a qualitatively correct solution, that decays to zero as t increases, we must choose h so that $|Q(h\lambda)| < 1$.

Example Consider the modified Euler method

$$y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_n + h, y_n + hf(t_n, y_n))].$$

Setting $f(t, y) = \lambda y$ yields the computed solution

$$y_{n+1} = y_n + \frac{h}{2}[\lambda y_n + \lambda(y_n + h\lambda y_n)] = \left(1 + h\lambda + \frac{1}{2}h^2\lambda^2\right) y_n,$$

so $Q(h\lambda) = 1 + h\lambda + \frac{1}{2}(h\lambda)^2$. If we assume λ is real, then in order to satisfy $|Q(h\lambda)| < 1$, we must have $-2 < h\lambda < 0$. It follows that the larger $|\lambda|$ is, the smaller h must be. □

The test equation can also be used to determine how to choose h for a multistep method. The process is similar to the one used to determine whether a multistep method is stable, except that we use $f(t, y) = \lambda y$, rather than $f(t, y) \equiv 0$.

Given a general multistep method of the form

$$\sum_{i=0}^{s} \alpha_i y_{n+1-i} = h \sum_{i=0}^{s} \beta_i f_{n+1-i},$$

we substitute $f_n = \lambda y_n$ and obtain the *recurrence relation*

$$\sum_{i=0}^{s} (\alpha_i - h\lambda\beta_i) y_{n+1-i} = 0.$$

It follows that the computed solution has the form

$$y_n = \sum_{i=1}^{s} c_i n^{p_i} \mu_i^n,$$

where each μ_i is a root of the *stability polynomial*

$$Q(\mu, h\lambda) = (\alpha_0 - h\lambda\beta_0)\mu^s + (\alpha_1 - h\lambda\beta_1)\mu^{s-1} + \cdots + (\alpha_{s-1} - h\lambda\beta_{s-1})\mu + (\alpha_s - h\lambda\beta_s).$$

The exponents p_i range from 0 to the multiplicity of μ_i minus one, so if the roots are all distinct, all p_i are equal to zero. In order to ensure that the numerical solution y_n decays to zero as n increases, we must have $|\mu_i| < 1$ for $i = 1, 2, \ldots, s$. Otherwise, the solution will either converge to a nonzero value, or grow in magnitude.

Example Consider the 3-step Adams-Bashforth method

$$y_{n+1} = y_n + \frac{h}{12}[23f_n - 16f_{n-1} + 5f_{n-2}].$$

Applying this method to the test equation yields the stability polynomial

$$Q(\mu, h\lambda) = \mu^3 + \left(-1 - \frac{23}{12}h\lambda\right)\mu^2 + \frac{4}{3}h\lambda\mu - \frac{5}{12}h\lambda.$$

Let $\lambda = -100$. If we choose $h = 0.1$, so that $\lambda h = -10$, then $Q(\mu, h\lambda)$ has a root approximately equal to -18.884, so h is too large for this method. On the other hand, if we choose $h = 0.005$, so that $h\lambda = -1/2$, then the largest root of $Q(\mu, h\lambda)$ is approximately -0.924, so h is sufficiently small to produce a qualitatively correct solution.

Next, we consider the 2-step Adams-Moulton method

$$y_{n+1} = y_n + \frac{h}{12}[5f_{n+1} + 8f_n - f_{n-1}].$$

In this case, we have

$$Q(\mu, h\lambda) = \left(1 - \frac{5}{12}h\lambda\right)\mu^2 + \left(-1 - \frac{2}{3}h\lambda\right)\mu + \frac{1}{12}h\lambda.$$

Setting $h = 0.05$, so that $h\lambda = -5$, the largest root of $Q(\mu, h\lambda)$ turns out to be approximately -0.906, so a larger step size can safely be chosen for this method. □

In general, larger step sizes can be chosen for implicit methods than for explicit methods. However, the savings achieved from having to take fewer time steps can be offset by the expense of having to solve a nonlinear equation during every time step.

The *region of absolute stability* of a one-step method or a multistep method is the region R of the complex plane such that if $h\lambda \in R$, then a solution computed using h and λ will decay to zero, as desired. That is, for a one-step method, $|Q(h\lambda)| < 1$ for $h\lambda \in R$, and for a multistep method, the roots $\mu_1, \mu_2, \ldots, \mu_s$ of $Q(\mu, h\lambda)$ satisfy $|\mu_i| < 1$.

Because a larger region of absolute stability allows a larger step size h to be chosen for a given value of λ, it is preferable to use a method that has as large a region of absolute stability as possible. The ideal situation is when a method is *A-stable*, which means that its region of absolute stability contains the entire left half-plane, because then, the solution will decay to zero regardless of the choice of h.

An example of an A-stable one-step method is the *Backward Euler method*

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}),$$

an implicit method. For this method,

$$Q(h\lambda) = \frac{1}{1 - h\lambda},$$

and since $\operatorname{Re}\lambda < 0$, it follows that $|Q(h\lambda)| < 1$ regardless of the value of h. The only A-stable multistep method is the *implicit trapezoidal method*

$$y_{n+1} = y_n + \frac{h}{2}[f_{n+1} + f_n],$$

because

$$Q(\mu, h\lambda) = \left(1 - \frac{h\lambda}{2}\right)\mu + \left(-1 - \frac{h\lambda}{2}\right),$$

which has the root

$$\mu = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}}.$$

The numerator and denominator have imaginary parts of the same magnitude, but because $\operatorname{Re}\lambda > 0$, the real part of the denominator has a larger magnitude than that of the numerator, so $|\mu| < 1$, regardless of h.

Implicit multistep methods, such as the implicit trapezoidal method, are often used for stiff differential equations because of their larger regions of absolute stability. Because y_{n+1} appears on both sides of the difference equation for such methods, it is necessary to use an iterative method such as Newton's method to compute y_{n+1}. For a general implicit multistep method, for which $\beta_0 \neq 0$, Newton's method is applied to the function

$$F(y) = \alpha_0 y + \sum_{i=1}^{s} \alpha_i y_{n+1-i} - h\beta_0 f(t_{n+1}, y) - h\sum_{i=1}^{s} \beta_i f_{n+1-i}.$$

The resulting iteration is

$$\begin{aligned} y_{n+1}^{(k+1)} &= y_{n+1}^{(k)} - \frac{F(y_{n+1}^{(k)})}{F'(y_{n+1}^{(k)})} \\ &= y_{n+1}^{(k)} - \frac{\alpha_0 y_{n+1}^{(k)} + \sum_{i=1}^{s} \alpha_i y_{n+1-i} - h\beta_0 f(t_{n+1}, y_{n+1}^{(k)}) - h\sum_{i=1}^{s} \beta_i f_{n+1-i}}{\alpha_0 - h\beta_0 f_y(t_{n+1}, y_{n+1}^{(k)})}, \end{aligned}$$

with $y_{n+1}^{(0)} = y_n$.

8.9 Dahlquist's Theorems

We conclue our discussion of multistep methods with some important results, due to Germund Dahlquist, concerning the consistency, zero-stability, and convergence of multistep methods.

Theorem (Dahlquist's Equivalence Theorem) A consistent multistep method with local truncation error $O(h^p)$ is convergent with global error $O(h^p)$ if and only if it is zero-stable.

This theorem shows that local error provides an indication of global error only for zero-stable methods.

The second theorem imposes a limit on the order of accuracy of zero-stable methods.

Theorem (Dahlquist's Barrier Theorem) The order of accuracy of a zero-stable s-step method is at most $s+1$, if s is odd, or $s+2$, if s is even.

For example, because of this theorem, it can be concluded that a 6th-order accurate three-step method cannot be zero stable, whereas a 4th-order accurate, zero-stable two-step method has the highest order of accuracy that can be achieved.

Finally, we state a result concerning absolute stability that highlights the trade-off between explicit and implicit methods.

Theorem (Dahlquist's Second Barrier Theorem) No explicit multistep method is A-stable. Furthermore, no A-stable multistep method can have an order of accuracy greater than 2. The second-order accurate, A-stable multistep method with the smallest asymptotic error constant is the trapezoidal method.

In order to obtain A-stable methods with higher-order accuracy, it is necessary to relax the condition of A-stability. Backward differentiation for-

mulae (BDF), mentioned previously in our initial discussion of multistep methods, are efficient implicit methods that are high-order accurate and have a region of absolute stability that includes a large portion of the negative half-plane, including the entire negative real axis.

8.10 Analysis of Multistep Methods

We illustrate the concepts and results of the preceding sections through examples in which the convergence properties of multistep methods are analyzed.

Example Consider the three-step method

$$y_{n+1} = -\frac{7}{2}y_n + 9y_{n-1} - \frac{9}{2}y_{n-2} + \frac{h}{6}[25f_n - 8f_{n-1} - 11f_{n-2}],$$

where $f_n = f(t_n, y_n)$. This method is neither an Adams method, nor a backward differentiation formula. It is explicit, because there is no dependence on f_{n+1}. Therefore, by Dahlquist's Second Barrier Theorem, it cannot be A-stable.

To determine its order of accuracy, we substitute the exact solution $y(t)$ into the method, and expand $y(t)$ in a Taylor series arount $t = t_n$. For a general multistep method of the form

$$\sum_{j=0}^{s} \alpha_j y_{n+1-j} = h \sum_{j=0}^{s} \beta_j f_{n+1-j},$$

this expansion reveals that the method has order of accuracy p if and only if all terms in the expansion of order $O(h^q)$ vanish for $q \leq p$, which is the case if the equations

$$\begin{aligned}
\sum_{j=0}^{s} \alpha_j &= 0, \\
\sum_{j=0}^{s-1} (s-j)\alpha_j - \sum_{j=0}^{s} \beta_j &= 0, \\
\sum_{j=0}^{s-1} \frac{(s-j)^2}{2}\alpha_j - \sum_{j=0}^{s} (s-j)\beta_j &= 0, \\
&\vdots
\end{aligned}$$

$$\sum_{j=0}^{s-1} \frac{(s-j)^p}{p!}\alpha_j - \sum_{j=0}^{s-1} \frac{(s-j)^{p-1}}{(p-1)!}\beta_j = 0,$$

are satisfied. From these equations, it can be determined that the above explicit three-step method is fourth-order accurate. From Dahlquist's First Barrier Theorem, this is the maximum order of accuracy that can be achieved by a zero-stable, three-step method.

To determine whether this method is zero-stable, we consider the first characteristic polynomial of the method,

$$\rho(z) = \sum_{j=0}^{s} \alpha_j z^{s-j} = z^3 + \frac{7}{2}z^2 - 9z + \frac{9}{2}.$$

Unfortunately, this method is not zero-stable, because one of its roots, $z \approx -5.34233$, lies outside of the unit circle. □

Example We next consider an implicit three-step method,

$$y_{n+1} = y_n + \frac{h}{24}[9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}],$$

which is an Adams method. It can easily be seen to be implicit, because the coefficient of f_{n+1} is nonzero. Checking order of accuracy as before, we find that this method is also fourth-order accurate. It is also strongly zero-stable, as all Adams methods are, because its first characteristic polynomial is $\rho(z) = z^3 - z^2$, which has a double root of zero and a simple root of 1. By the Dahlquist Equivalence Theorem, because this method is consistent and zero-stable, it is convergent.

By Dahlquist's First Barrier Theorem, this method cannot be A-stable, because its order of accuracy is greater than two. To determine its region of absolute stability, we consider its *stability polynomial* $Q(\mu, h\lambda)$ that is the characteristic polynomial of the recurrence relation that results from applying the method to the test equation $y' = \lambda y$ with step size h. This polynomial is obtained from the first and second characteristic polynomials of the method as follows:

$$\begin{aligned} Q(\mu, h\lambda) &= \rho(\mu) - h\lambda\sigma(\mu) \\ &= \sum_{j=0}^{s} \alpha_j \mu^{s-j} - h\lambda \sum_{j=0}^{s} \beta_j \mu^{s-j} \\ &= \left(1 - \frac{3h\lambda}{8}\right)\mu^3 - \left(1 + \frac{19h\lambda}{24}\right)\mu^2 + \frac{5h\lambda}{24}\mu - \frac{h\lambda}{24}. \end{aligned}$$

The region of absolute stability is the region in the complex plane consisting of all values of $h\lambda$ for which all of the roots of $Q(\mu, h\lambda)$ are less than one in absolute value. A portion of the region of absolute stability for the above three-step implicit Adams method is shown in Figure 8.2. It follows that for

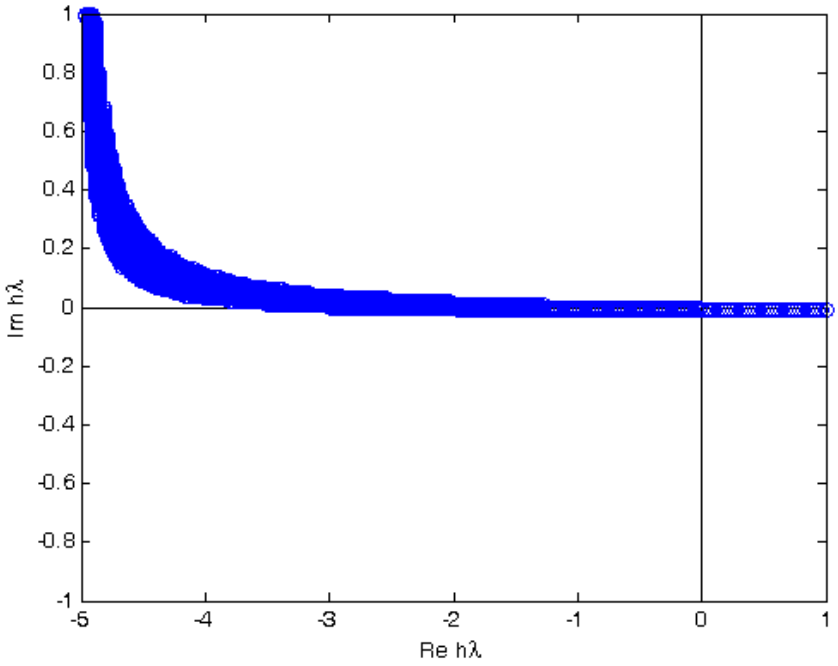

Figure 8.2: Region of absolute stability for a three-step implicit Adams method

the test equation $y' = \lambda y$, where λ is a negative real number, the step size h must be chosen so that $h < 4/|\lambda|$ in order to guarantee that the computed solution decays to zero, as the exact solution does. □

Index