# Chapter 3

# Numerical Quadrature

## 3.1 Midpoint Rule and Simpson's Rule

Now that we have seen what the finite element is, we turn our attention to its implementation. We begin by considering the computation of the stiffness matrix $S$, where $S_{ji} = \mathcal{L}(\psi_i, \psi_j)$ and $\{\psi_i\}$ is the basis for the finite element space being used. Recall from the previous chapter that $\mathcal{L}(\psi_i, \psi_j) = \int_0^1 a\psi_i'\psi_j'\,dx + \int_0^1 b\psi_i'\psi_j\,dx + \int_0^1 c\psi_i\psi_j\,dx$. Thus in order to compute $S$, we must compute a number of integrals. In this chapter, we study efficient ways of accurately approximating integrals. We shall use numerical quadrature (or numerical integration) to approximate integrals.

We first emphasize the point that we shall only give ways to approximate integrals here, not compute them exactly (although our rules will be able to exactly compute integrals of polynomials having small enough degree). The exact evaluation of general integrals, when it is possible at all, is much more computationally expensive than their numerical approximation. The dual (and often competing) goals in scientific computing are efficiency and accuracy. The correct choice of numerical quadrature methods yields sufficient accuracy with high efficiency and is thus the preferred method of computing integrals in many applications, including finite element computations.

Our goal is to approximate $\int_I f(x)\,dx$, where $I$ is some interval in $\mathbb{R}$. We're going to start by breaking $I$ into intervals $I_i$ of length $k_i$, $i = 1, ..., M$, so $I = \cup_{i=1}^N I_i$ and $\int_I f(x) = \sum_{i=1}^M \int_{I_i} f(x)\,dx$. We will approximate each $\int_{I_i} f(x)\,dx$ and sum the resulting approximations to approximate $\int_I f(x)\,dx$. We call this sort of approximation a *composite rule* because an integral over an interval $I$ is approximated by a composite of approximations of integrals over subintervals. To begin our discussion, we analyze two very common quadrature techniques, the midpoint rule and Simpson's rule.

Before stating any of the methods or giving error bound for them, we recall Taylor's theorem.

**Theorem 3.1.1 (Taylor's Theorem)** *Assume $f(x) \in C^{n+1}([a,b])$. Let $T^n(x) = \sum_{i=0}^n \frac{1}{i!} f^{(i)}(a)(x-a)^i$, and $R^n(x) = f(x) - T^n(x)$. Then for each point $x \in [a,b]$,*

$$|R^n(x)| \leq \frac{1}{(n+1)!} \|f^{(n+1)}\|_{L_\infty([a,b])} |x-a|^{n+1}$$

Taylor's theorem will be used repeatedly in this and the subsequent chapters.

Let $I = [a, b]$ be an interval in $\mathbb{R}$ with midpoint $\frac{a+b}{2}$ and length $k = b - a$. The midpoint rule is then

$$\int_I f(x)\,dx \approx kf(\frac{a+b}{2}).$$

Our error bound for the midpoint rule is as follows.

**Proposition 3.1.2** *Assume $f \in C^2(I)$. Then $\int_I f(x)\,dx - kf(\frac{a+b}{2}) \leq \frac{k^3}{24}\|f''\|_{L_\infty(I)}$.*

*Proof.* We first use Taylor's theorem to find that for $x \in I$, $f(x) = f(\frac{a+b}{2}) + f'(\frac{a+b}{2})(x - \frac{a+b}{2}) + R^1(x)$, so

$$\int_I f(x)\,dx = \int_I f(\frac{a+b}{2}) + f'(\frac{a+b}{2})x + R^n(x)\,dx = kf(\frac{a+b}{2}) + \int_I R^1(x)\,dx$$

(note that $\int_I f'(0)(x - \frac{a+b}{2})\,dx = 0$ because $(x - \frac{a+b}{2})$ is antisymmetric about $\frac{a+b}{2}$). Thus

$$\int_I f(x)\,dx - kf(\frac{a+b}{2}) = \int_I R^n(x)\,dx\,.$$

Using Taylor's Theorem, we find that

$$\begin{array}{rcl} |\int_I f(x)\,dx - kf(\frac{a+b}{2})| & \leq & \int_I |R^n(x)|\,dx \leq \int_I \frac{1}{2!}\|f''\|_{L_\infty(I)}(x - \frac{a+b}{2})^2\,dx \\ & = & \frac{1}{2}\|f''\|_{L_\infty(I)}\frac{1}{3}(x - \frac{a+b}{2})^3|_a^b = \frac{k^3}{24}\|f''\|_{L_\infty(I)}. \end{array}$$

□

We note that something special has happened here. Since we chose to approximate our integral by evaluating $f$ at the midpoint of the interval, the first-degree term in the Taylor polynomial fell out by symmetry. If we had chosen to evaluate $f$ at any other point within the inverval $I_i$, the linear term in the Taylor polynomial would not have fallen out, and we would have only gotten a factor of $k^2$ in our estimate instead of a factor of $k^3$ (note we're thinking of taking $k$ to be small, so that $k^2$ is much larger than $k^3$). Also, assume that we are approximating $\int_0^1 f(x)\,dx$ by $\sum_{j=1}^N kf(m_j)$, where $k = \frac{1}{N}$ (i.e., the $I_j$'s are uniform) and $m_j$ is the midpoint of $I_j$. Then

$$\begin{array}{rcl} |\int_0^1 f(x)\,dx - \sum_{j=1}^N kf(m_j)| & = & |\sum_{j=1}^N \int_{I_i} f(x)\,dx - kf(m_j)| \leq \sum_{j=1}^N \frac{k^3}{24}\|f''\|_{L_\infty(I_j)} \\ & \leq & N\frac{k^3}{24}\|f''\|_{L_\infty(I)} = \frac{k^2}{24}\|f''\|_{L_\infty(I)} \end{array}.$$

Thus the order of convergence for the composite midpoint rule is $k^2$.

We state Simpson's rule and error bounds for it here, but we do not prove the error bounds. Simpson's rule is derived as follows. We first approximate the curve $y = f(x)$ on $I$ by the parabola $P(x)$ passing through the curve $f(x)$ at the points $a$, $\frac{a+b}{2}$, and $b$. Let $P(x) = A(x - \frac{a+b}{2})^2 + B(x - \frac{a+b}{2}) + C$. It is easy to see that $C = f(\frac{a+b}{2})$, and inserting $x = a$ and $x = b$ into $P$, we find that

$$\begin{array}{rcl} P(a) & = & (\frac{k}{2})^2 A - \frac{k}{2}B + f(\frac{a+b}{2}) = f(a), \\ P(b) & = & (\frac{k}{2})^2 A + \frac{k}{2}B + f(\frac{a+b}{2}) = f(b), \end{array}$$

so that solving we find that $A = \frac{2(f(a) - 2f(\frac{a+b}{2}) + f(b))}{k^2}$ and $B = \frac{f(b) - f(a)}{k}$. We now approximate $\int_I f(x)\,dx$ by

$$\begin{array}{rcl} \int_I P(x)\,dx & = & \frac{A}{3}(x - \frac{a+b}{2})^3 + \frac{B}{2}(x - \frac{a+b}{2})^2 + C(x - \frac{a+b}{2})|_a^b \\ & = & \frac{k^3 A}{12} + kC = \frac{k(f(a) - 2f(\frac{a+b}{2}) + f(b))}{6} + kf(\frac{a+b}{2}) = \frac{1}{6}(f(a) + 4f(\frac{a+b}{2}) + f(b)). \end{array}$$

Thus we use the approximation $\int_a^b f(x)\,dx \approx \frac{k}{6}(f(a) + 4f(\frac{a+b}{2}) + f(b))$. The error bound for the composite Simpson's rule is as follows.

**Proposition 3.1.3** *Assume that $f \in C^4(I)$, where $I = [a,b]$. Divide $I$ into $M$ subintervals of length $k = \frac{b-a}{M}$ and then approximate $\int_I f(x)\,dx$ by the composite Simpson's rule (i.e., adding up the result of Simpson's rule on each of these subintervals). The error in this procedure is bounded by $\frac{k^4(b-a)}{90}\|f^{(4)}\|_{L_\infty(I)}$.*

## 3.2 Gaussian Quadrature

We next present Gaussian quadrature. Its development may be a little less transparent than that of the midpoint and Simpson's rules, but as we shall see, it is worth the effort.

Instead of directly attacking the problem of approximating $\int_a^b f(x)\,dx$, it turns out to be much more convenient to first approximate $\int_{-1}^1 f(x)\,dx$, then handle other intervals by a change of variables. Our goal will be to build an approximation $\sum_{i=0}^M w_i f(x_i) \approx \int_{-1}^1 f(x)\,dx$. Here the values $w_i$ are called weights and the points $x_i$ are called quadrature points. We note that our analysis of the midpoint rule was based on the fact that it could "knock out" (or integrate exactly) the first two terms of the Taylor expansion of $f$, and the analysis of Simpson's rule would similarly show that it can integrate exactly the first four terms of the Taylor expansion of $f$. We thus attempt to pick the weights $w_i$ and the quadrature points $w_i$ so that $\sum_{i=0}^M w_i f(x_i)$ integrates exactly as high of order of polynomial as possible, that is, $\int_{-1}^1 x^j = \sum_{i=0}^M w_i (x_i)^j$ for $j$ as large as possible.

We note that $\sum_{i=0}^M w_i f(x_i)$ contains $2(M+1)$ degrees of freedom $\{w_i\}_{i=0,\ldots,M}$ and $\{x_i\}_{i=0,\ldots,M}$. We thus may guess that we can integrate exactly polynomials $a_0 + a_1 x + a_2 x^2 + \ldots + a_{2M+1}x^{2M+1}$ of up to degree $2M+1$ with the rule $\sum_{i=0}^M w_i f(x_i)$. This is indeed the case, although constructing $w_i$ and $x_i$ will take some work. We note that the midpoint rule is in fact the lowest-order Gaussian rule (it uses one function evaluation, and it will exactly integrate linear functions). Simpson's rule, on the other hand, requires function evaluation at three points and will exactly integrate cubics (a fact which we didn't show), while the Gaussian rule to integrate cubics ($M = 1$) requires two function evaluations and is thus more efficient.

In order to derive the weights and quadrature points, we shall need to use the Legendre polynomials $L_j(x)$, $j = 0, 1, 2, \ldots$ Their defininining property is that they satisfy

$$\int_{-1}^1 L_i(x) L_j(x)\,dx = \delta_{ij}.$$

Thus $\{L_i\}_{i=0,\ldots,\ell}$ forms an orthonormal basis for $P^\ell(0,1)$ with respect to the inner product $(u,v) = \int_{-1}^1 u(x)v(x)\,dx$. We may apply the Gram-Schmidt orthogonalization procedure to determine $L_i$, which would lead to

$$\begin{aligned} L_0(x) &= \tfrac{1}{\sqrt{2}}, \\ L_1(x) &= \sqrt{\tfrac{3}{2}}x, \\ L_2(x) &= \sqrt{\tfrac{5}{2}}\tfrac{1}{2}(3x^2 - 1). \end{aligned}$$

It turns out that $L_j$ is even if $j$ is even and odd if $j$ is odd. In general, the Rodriguez formula

$$L_j(x) = \sqrt{\frac{2j+1}{2}} \frac{1}{j!2^j} \frac{d^j}{dx^j}[(x^2-1)^j]$$

gives $L_j(x)$. It should also be noted that $L_M$ is orthogonal to any polynomial of degree less than $M$ since $(L_M, L_i) = 0$, $i = 0, ..., M-1$, and $\{L_i\}_{i=0,...,M_1}$ forms a basis for $P^{M-1}$.

We now prove a lemma about $L_M$.

**Lemma 3.2.1** *$L_M$ has $M$ zeros, all of which are simple and all of which lie inside of $(-1,1)$.*

*Proof.* Let $r$ be the number of sign changes of $L_M$ lying inside of $(-1,1)$. Since $L_M$ is a polynomial of degree $M$, we must have $r \leq M$. Since we wish to show that $r = M$, we shall assume $r < M$ and reach a contradiction. If $r < M$, then $L_M$ has $r$ sign changes inside of $(-1,1)$ occuring at $\xi_1, ..., \xi_r$. Thus inside of $(-1,1)$, the polynomial $p(x) = (x-\xi_1)(x-\xi_2)\cdot\cdots\cdot(x-\xi_r)$ will have exactly the same (or perhaps exactly the opposite) sign as $L_M$. Thus $(p, L_M) \neq 0$. However, we must have $(p, L_M) = 0$ since the degree of $p$ is less than $M$. This is a contradiction; thus $L_M$ has precisely $M$ sign changes in $(-1,1)$ and thus $M$ simple zeros in $(-1,1)$. □

We choose the $M+1$ quadrature points $x_0, ..., x_M$ to be the $M+1$ zeros of $L_{M+1}$.

In order to choose the weights $w_i$, we shall use Lagrangian interpolation.

**Lemma 3.2.2** *Given distinct points $\{x_i\}_{i=0,...,n}$ and values $\{g_i\}_{i=0,...,n}$, there exists a unique polynomial $q$ of degree $n$ such that $q(x_i) = g_i$, $i = 0, ..., n$.*

*Proof.* We first prove uniqueness. Assume that $q_1(x_i) = g_i$ and $q_2(x_i) = g_i$, $i = 0, ..., n$, and that $q_1$ and $q_2$ are both polynomials of degree $n$. Then $q_1 - q_2$ is also a polynomial of degree $n$, and it has $n+1$ zeros $\{x_i\}_{i=0,...,n}$. A polynomial of degree $n$ with $n+1$ zeros must be 0 everywhere, so $q_1 = q_2$.

We now construct $q$. Let $\ell_i(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}$. $\ell_i$ is then a polynomial of degree $n$ satisfying $\ell_i(x_j) = \delta_{ij}$. $q(x) = \sum_{i=0}^n g_i \ell_i(x)$ then has the desired properties. □

Recall that we are seeking to choose the $w_i$'s so that $\sum_{i=0}^M w_i (x_i)^j = \int_{-1}^1 x^j\,dx$, $j = 0, ..., 2M+1$, i.e., so that our quadrature rule is exact on all polynomials of degree $2M+1$ or less. Let $\{\ell_i(x)\}$ be the polynomials of degree $M$ satisfying $\ell_i(x_j) = \delta_{ij}$ for the $M+1$ quadrature points $\{x_i\}_{i=0,...,M}$. We then define $p_{M,j}(x) = \sum_{i=0}^M (x_i)^j \ell_i(x)$, where $0 \leq j \leq 2M+1$. That is, $p_{M,j}$ interpolates $x^j$ at the quadrature points. Let $\psi_{M,j} = x^j - p_{M,j}(x)$. Note that $\psi_{M,j}(x_i) = 0$, $i = 0, ..., M$, since $p_{M,j}$ interpolates $x^j$ at these points. Thus $\psi_{M,j}(x) = L_{M+1}(x)G(x)$, where $L_{M+1}$ is the Legendre polynomial of degree $M+1$ and $G$ is some polynomial of degree $j-(M+1) \leq 2M+1-(M+1) = M$. We then have $x^j = p_{M,j}(x) + \psi_{M,j}(x)$, so that $\int_{-1}^1 x^j\,dx = \int_{-1}^1 p_{M,j}(x)\,dx + \int_{-1}^1 \psi_{M,j}(x)\,dx$. But $\int_{-1}^1 p_{M,j}(x)\,dx = \int_{-1}^1 \sum_{i=0}^M (x_i)^j \ell_i(x)\,dx = \sum_{i=0}^M (x_i)^j \int_{-1}^1 \ell_i(x)\,dx$. Also, $\int_{-1}^1 \psi_{M,j}(x)\,dx = \int_{-1}^1 L_{M+1}(x)G(x)\,dx = 0$ because $L_{M+1}$ is orthogonal to all polynomials of degree $M$ or less. Thus $\int_{-1}^1 x^j\,dx = \sum_{i=0}^M (x_i)^j \int_{-1}^1 \ell_i(x)\,dx$. We therefore choose $w_i = \int_{-1}^1 \ell_i(x)\,dx$. As we have just shown, our quadrature rule $\sum_{i=0}^M w_i f(x_i)$ is thus exact when $f(x)$ is a polynomial of degree $2M+1$ or less.

We next write down the composite Gaussian quadrature rule. We wish to use a composite rule to estimate $\int_A^B f(x)\,dx$. Let $[A,B] = \cup_{j=1}^N I_j$, where the $I_j$ are disjoint intervals having midpoints $m_j$ and length $k_j$. Then $\int_A^B f(x)\,dx = \sum_{j=1}^N \int_{I_j} dx$. We next note that $\int_{I_j} f(x)\,dx =$

$\frac{k_j}{2}\int_{-1}^{1} f(\frac{k_j}{2}x + m_j)\,dx \approx \frac{k_j}{2}\sum_{i=0}^{M} w_i f(\frac{k_j}{2}x_i + m_j)$. Thus our composite rule is: $\int_A^B f(x)\,dx \approx \sum_{j=1}^{N}\sum_{i=0}^{M} \frac{k_j}{2} w_i f(\frac{k_j}{2}x_i + m_j)$. Note that a translated and scaled polynomial of given degree is still a polynomial of that degree, so the composite Gaussian rule is exact on polynomials of degree up to $2M+1$–in fact, it's exact on piecewise polynomials of degree up to $2M+1$ since the integrals over each subinterval $I_i$ are approximated independently. (This is in contrast to Simpson's rule, which required function evaluations at endpoints of intervals, meaning it only makes sense for functions which are at least continuous.)

Before stating and proving an error bound for this quadrature rule, we would like to observe that $\sum_{i=0}^{M} w_i = 2$ and $w_i > 0$, $i = 0, ..., M$. To see that $\sum_{i=0}^{M} w_i = 2$, let $f(x) = 1$ so that $2 = \int_{-1}^{1} 1\,dx = \sum_{i=0}^{M} w_i$. To see that the second fact is true, note that $\ell_i^2$ is a polynomial of degree $2M$, so our rule is exact for this function. That is, $\int_{-1}^{1} \ell_i^2(x)\,dx = \sum_{j=0}^{M} w_j \ell_i^2(x_j) = w_i$. Thus $w_i$ may be represented as the integral of the square of a function not identically zero, and $w_i$ must thus be positive.

**Theorem 3.2.3**

$$|\int_{I_j} f(x)\,dx - \frac{k_j}{2}\sum_{i=0}^{M} f(\frac{k_j}{2}x_i + m_j)| \le C_{2M+2} k_j^{2M+2} \|f^{(2M+2)}\|_{L_1(I_j)} \tag{3.1}$$

*and*

$$|\int_A^B f(x)\,dx - \sum_{j=1}^{N}\sum_{i=0}^{M} \frac{k_j}{2} f(\frac{k_j}{2}x_i + m_j)| \le C_{2M+2} k^{2M+2} \|f^{(2M+2)}\|_{L_1([A,B])}, \tag{3.2}$$

*where* $k = \max_{j=1,...,N} k_j$.

*Proof.* The essential ingredient of the proof is that Gaussian quadrature integrates exactly polynomials of a given degree $(2M+1)$, so that it in essence "knocks out" the first $2M+1$ terms of the Taylor expansion of $f$. We call this property *polynomial invariance.*

We shall first prove (3.1). In this proof, we shall need a slightly nonstandard version of Taylor's theorem for a Taylor expansion of $f$ about $m_j$, which is $f(x) = T^n(x) + R^n(x)$, where $R^n(x) = \int_{m_j}^{x} \frac{(x-t)^n}{(n)!} f^{(n+1)}(t)\,dt$. Next let $E^M(f) = \int_{I_j} f(x)\,dx - \frac{k_j}{2}\sum_{i=0}^{M} f(\frac{k_j}{2}x_i + m_j)$. Note that $E^M(p) = 0$ for any polynomial $p$ having degree less than or equal to $2M+1$, and note also that $E^M$ is linear, i.e., $E^M(f + cg) = E^M(f) + cE^M(g)$ for functions $f$, $g$ and constant $c$. Using Taylor's formula, we find that $E^M(f) = E^M(T^{2M+1} + R^{2M+1}) = E^M(T^{2M+1}) + E^M(R^{2M+1}) = E^M(R^{2M+1})$. Thus we must bound $|E^M(R^{2M+1})| = |\int_{I_j} R^{2M+1}(x)\,dx - \frac{k_j}{2}\sum_{i=0}^{M} w_i R^{2M+1}(\frac{k_j}{2}x_i + m_j)|$. Note first that

$$|\int_{I_j} R^{2M+1}(x)\,dx\,| \le \int_{I_j} |R^{2M+1}(x)|\,dx\,.$$

But

$$\begin{aligned} |R^{2M+1}(x)| &= |\int_{m_j}^{x} \frac{(x-t)^{2M+1}}{(2M+1)!} f^{2M+2}(t)\,dt\,| \le \int_{I_j} (\frac{k_j}{2})^{2M+1} \frac{1}{(2M+1)!} |f^{2M+2}(t)|\,dt \\ &\le \frac{k_j^{2M+1}}{2^{2M+1}(2M+1)!} \|f^{2M+2}\|_{L_1(I_j)} \end{aligned}\,. \tag{3.3}$$

Thus

$$|\int_{I_j} R^{2M+1}(x)\,dx\,| \le \int_{I_j} \frac{k_j^{2M+1}}{2^{2M+1}(2M+1)!}\|f^{2M+2}\|_{L_1(I_j)}\,dx = \frac{k_j^{2M+2}}{2^{2M+1}(2M+1)!}\|f^{2M+2}\|_{L_1(I_j)}. \tag{3.4}$$

Using (3.3) and recalling that $\sum_{i=0}^{M}|w_i| = \sum_{i=0}^{M} w_i = 2$, we next find that

$$\begin{aligned} |\tfrac{k_j}{2}\textstyle\sum_{i=0}^{M} w_i R^{2M+1}(\tfrac{k_j}{2}x_i + m_j)| &\le \tfrac{k_j}{2}\|R^{2M+1}\|_{L_\infty(I_j)}\textstyle\sum_{i=0}^{M}|w_i| \\ &\le \tfrac{k_j}{2}\tfrac{k_j^{2M+1}}{2^{2M+1}(2M+1)!}\|f^{2M+2}\|_{L_1(I_j)}\cdot 2 \\ &\le \tfrac{k_j^{2M+2}}{2^{2M+1}(2M+1)!}\|f^{2M+2}\|_{L_1(I_j)}. \end{aligned} \tag{3.5}$$

Thus combining (3.4) and (3.5), we find that

$$\begin{aligned} |E^M(f)| &= |\textstyle\int_{I_j} R^{2M+1}(x)\,dx - \tfrac{k_j}{2}\sum_{i=0}^{M} R^{2M+1}(\tfrac{k_j}{2}x_i + m_j)| \\ &\le |\textstyle\int_{I_j} R^{2M+1}(x)\,dx\,| + |\tfrac{k_j}{2}\sum_{i=0}^{M} R^{2M+1}(\tfrac{k_j}{2}x_i + m_j)| \\ &\le 2\tfrac{k_j^{2M+2}}{2^{2M+1}(2M+1)!}\|f^{2M+2}\|_{L_1(I_j)} = \tfrac{k_j^{2M+2}}{2^{2M}(2M+1)!}\|f^{2M+2}\|_{L_1(I_j)}. \end{aligned}$$

This completes the proof of (3.1) with $C_{2M+2} = \frac{1}{2^{2M}(2M+1)!}$.

In order to complete the proof of (3.2), we simply sum over the intervals $I_j$:

$$\begin{aligned} |\textstyle\int_A^B f(x)\,dx &- \sum_{j=1}^{N}\sum_{i=0}^{M}\tfrac{k_j}{2}w_i f(\tfrac{k_j}{2}x_i + m_j)| \le \sum_{j=1}^{M}|E_j^M(f)| \\ &\le \textstyle\sum_{j=1}^{N}\tfrac{k_j^{2M+2}}{2^{2M}(2M+1)!}\|f^{2M+2}\|_{L_1(I_j)} \le \sum_{j=1}^{N}\tfrac{k^{2M+2}}{2^{2M}(2M+1)!}\|f^{2M+2}\|_{L_1(I_j)} \\ &\le \tfrac{k^{2M+2}}{2^{2M}(2M+1)!}\textstyle\sum_{j=1}^{N}\|f^{2M+2}\|_{L_1(I_j)} \le \tfrac{k^{2M+2}}{2^{2M}(2M+1)!}\|f^{2M+2}\|_{L_1([A,B])}. \end{aligned}$$

□

We next make a couple of notes about (3.1) and (3.2). First we note that the constant $C_{2M+2}$ we obtained is not the best possible, though it will decrease very quickly as $M$ increases even as it is. We won't worry about finding the best possible constant, however. What we're most concerned about is the exponent of $k$, which we call the rate of convergence. This is perhaps the most important piece of information gleaned from these error bounds because they tell us how fast the error decreases as we use smaller and smaller subintervals in our quadrature rule. Another note we make about the error bounds we have derived is that they are called *a priori* error bounds because they tell us before we have done any computations what kind of error we can expect. It is often the case in real applications that the $(2M+2)$-th derivatives of $f$ are difficult or impossible to compute. Thus in reality, a priori error bounds don't really tell us how small the error will be–they only give us an idea of how fast it will decrease. Another type of bounds, called *a posteriori* error bounds, use information gleaned from computations to estimate how large the error is. (That is, they don't involve unknown quantities such as $2M+2$-th derivatives.) In a posteriori bounds, constants generally do matter, in contrast to a priori bounds.

A priori bounds suggest a very important tool for checking codes, which is the observed rate of convergence. An observed rate of convergence is computed as follows. First, pick a function $f$ for which $\int_A^B f(x)\,dx$ is known exactly. We only know that the error in Gaussian quadrature is bounded by $Ck^{2M+2}$ (where $C$ now depends on $f$ as well as $M$); assume instead that the error is actually

equal to $Ck^{2M+2}$. Denote by $E_k$ the error obtained by using uniform mesh intervals of size $k$. Then

$$\frac{E_k}{E_{k/2}} = \frac{Ck^{2M+2}}{C(k/2)^{2M+2}} = 2^{2M+2}.$$

Thus

$$\log_2 \frac{E_k}{E_{k/2}} = \log_2 2^{2M+2} = 2M + 2.$$

Thus if we compute $E_k$ and $E_{k/2}$ and compute $\log_2$ of their ratio, we should get $2M + 2$. This observation is based on the assumption that the error is exactly $Ck^{2M+2}$, not just bounded by this quantity, but in practice this test usually gives an observed rate of convergence close to the theoretical rate of convergence for the proper values of $k$. Although we do not justify why this is so, the proper values of $k$ are those which are "small enough" (what "small enough" means will depend on the situation, but numbers like $k = 1$ usually aren't) but not so small that roundoff error will factor heavily into computations. The observed rate of convergence is one of the first tests that should be performed on code, if possible. If the theoretical proven rate is not observed, there is likely something wrong with the code. If possible, codes should also always be tested on problems for which they should give exact answers. For example, in the case of Gaussian quadrature one should test the code on polynomials of degree $2M + 1$ and piecewise polynomials of degree $2M + 1$. The error in this case should be something like machine precision ($10^{-16}$); if it isn't, something is probably wrong.

## 3.3 Exercises

The exercise for this section is to program and compare the midpoint rule and the Gauss rule obtained for $M = 5$. The quadrature points $x_i$ and weights $w_i$ for $M = 5$ are given in Table 3.1. Program the composite midpoint rule and the composite Gaussian rule with $M = 5$. Your code

Table 3.1:

| Quadrature points ($\pm x_i$) | Weights ($w_i$) |
|---|---|
| .238619186083197 | .467913934572691 |
| .661209386466265 | .360761573048139 |
| .932469514203152 | .171324492379170 |

should approximately integrate from $A$ to $B$. You should check your code using different meshes, both uniform and non-uniform. First test your code on functions it should integrate exactly; make sure that both methods integrate polynomials of up to order $2M + 1$ exactly for both uniform and non-uniform meshes. You don't need to hand in results showing that your code integrates polynomials exactly; the parts you do need to hand in are:
(a) Use your code to calculate estimated rates of convergence for both quadrature rules. Think carefully about how to pick a function that will give clear results; try several different functions if necessary. Use a uniform mesh. Your answer should consist of the function or functions you used to get results along with numerical results showing estimated rates of convergence for both rules.

(b) Use first the midpoint rule, then the Gaussian rule with $M = 5$ to approximately integrate $\sin(\frac{1}{x})$ from .01 to 1. Your answer should consist of your approximations to $\int_{.01}^{1} \sin\frac{1}{x}\,dx$ and a clear description of the meshes you used to obtain your approximations (including how many mesh intervals each method required and how the mesh intervals were positioned). Remember to balance carefully the goals of efficiency and accuracy, that is, make sure your answer is correct, but don't put too many mesh intervals where they aren't needed. (Not to hand in: Can you think of a way to automate the chore of choosing a good mesh? )

## 3.4 Programming Project, Part II

In this part of the project, you will assemble the stiffness matrix $S$ and the right-hand-side vector $F$. After completing this assignment, you will be only a linear system solver away from a working finite element code. Recall that the finite element method requires you to solve a system of the form $SU = F$, where $S_{ji} = \mathcal{L}(\psi_i, \psi_j)$ and $\{\psi_i\}$ is a basis for $S_h$ (or $S_h^{k,\mu}$). Computing $S_{ji}$ thus involves computing some integrals, which you are now well-prepared to do. Recall that $\mathcal{L}(\psi_i, \psi_j) = 0$ if $i$ and $j$ are far enough apart (how far depends on the method), and that $\mathcal{L}(\psi_i, \psi_j)$ will involve integrals over at most two mesh intervals in any case. You should apply Gaussian quadrature with $M = 5$ directly on each mesh interval (that is, no composite rule is needed). Thus if you are computing an integral involving two mesh intervals, you will need to apply the quadrature rule twice, once on each interval. Compute and assemble the stiffness matrix $S$ and the right-hand-side vector $F$ (with entries $\int_0^1 f\psi_i\,dx$). You should use a sparse matrix structure to store $S$; otherwise you end up storing a whole bunch of zeros. It's fine if you store the diagonals of $S$ in vectors, at least for now, or else you may use Matlab's "sparse" matrix data structure.

Using $M = 5$ will give you WAY more accuracy than you need for the piecewise linear method, and plenty of accuracy for the finite element method using cubic splines. Using such a high order of accuracy in quadrature would probably be considered "overkill" by most practical finite element practitioners, but our goal is to construct our code in such a way that the quadrature error can be ignored–and it essentially can be with $M = 5$.