CHAPTER 5

Numerical Quadrature

5.1. Introduction

The bulk of material that I want to cover in this chapter is in sections 5.2 through 5.4. You should read section 5.7 on your own. As time permits, I may talk about the material in sections 5.5, 5.6 and 5.9.

Read section 5.1 in the book on your own.

The problem we want to investigate in this chapter is that of finding the definite integral of a function, that is, evaluating

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

numerically. We will only consider the simplest case: a one-dimensional integral, a finite interval $[a, b]$, and a continuous function $f(x)$. If time permits, I may say a few things about infinite intervals and/or discontinuous functions. We will not cover multidimensional integrals.

Many numerical analysts, including the authors of your textbook, feel that the word *integration* should be reserved for finding the mathematically exact antiderivative of a function or the area under a curve, while the numerical approximation to that should be called *quadrature*.

Personally, I have no problem with the phrase *numerical integration*. I will use the word *quadrature* in these notes, but in class I will probably use *integration* and *quadrature* interchangeably.

5.2. One-dimensional Quadrature Rules and Formulas

I will cover book sections 5.2 and 5.4 in this section of my lecture notes.

The problem we want to solve is the following:

Given a function $f(x)$ on a finite interval $[a, b]$, calculate numerically an approximation to the definite integral

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

We assume that the function f is at least continuous, and that we can evaluate it at any point in $[a, b]$. For the error estimates later, we have to assume that f is sufficiently often differentiable.

What form can a quadrature rule take? It is obvious that we can only evaluate f at a finite number of points x_i. Also, we note that integration is a linear process, which means

$$\int [\alpha f(x) + \beta g(x)]\,dx = \alpha \int f(x)\,dx + \beta \int g(x)\,dx.$$

A quadrature algorithm should have the same property. This means that our quadrature formula has to be a linear combination of the $f(x_i)$

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n} w_i f(x_i).$$

The x_i are called the *nodes*, the w_i are called the *weights*. It is desirable that all weights are positive, otherwise we may run into problems with cancellation.

The methods I want to consider are based on the following approaches:

- Choose the x_i equally spaced, and determine the w_i so that the formula is exact for polynomials of as high a degree as possible. This leads to the *Newton-Cotes formulas*.
- Choose both the x_i and the w_i so that the formula is exact for polynomials of as high a degree as possible. This leads to *Gaussian quadrature*.

- Choose the x_i any way you want, interpolate a spline through these points, and integrate the spline. Subroutine `PCHQA` in this chapter is based on this approach. We will not discuss this method in detail.

5.2.1. Newton-Cotes Formulas. The idea behind the Newton-Cotes formulas is to choose the x_i equally spaced throughout $[a, b]$, and to determine the w_i so that the formula is exact for polynomials of as high a degree as possible.

The w_i can be calculated in at least two ways:

(1) We replace the function f by the polynomial which interpolates f at the points x_i, and integrate the polynomial.
(2) We write down a series of equations:

$$\begin{aligned}\text{numerical integral of } x^0 &= \text{exact integral of } x^0\\ \text{numerical integral of } x^1 &= \text{exact integral of } x^1\\ \text{numerical integral of } x^2 &= \text{exact integral of } x^2\\ &\dots\end{aligned}$$

and solve this system.

I will illustrate both methods in detail when we derive Simpson's rule.

Newton-Cotes formulas come in two flavors: open and closed.

The *closed* formulas use the end points of subintervals. If we use n subintervals, the stepsize is $h = (b - a)/n$, and we get $(n + 1)$ points

$$x_0 = a, \quad x_1 = a + h, \quad \dots, \quad x_{n-1} = b - h, \quad x_n = b.$$

The *open* formulas use the midpoints of subintervals. With n subintervals, the stepsize is again $h = (b-a)/n$, but we get n points

$$x_1 = a + h/2, \quad x_2 = a + 3h/2, \quad \dots, \quad x_n = b - h/2.$$

The open formulas are useful if the function cannot be evaluated at a and/or b.

Newton-Cotes formulas are not recommended for more than 7 points, because the weights become large, and some of them are negative, which leads to cancellation. This is related to the instability of polynomial interpolation we observed before.

There is still an easy way to use more points, for higher accuracy: we simply subdivide $[a, b]$ into smaller intervals, and use a lower order Newton-Cotes formula on each subinterval. These are the *repeated* or *compound Newton-Cotes formulas.*

The word "repeated" is often left out. If you read about the "trapezoidal rule" in a book, the author usually means the repeated trapezoidal rule.

We will only consider three types of Newton-Cotes formulas in detail: the midpoint rule (open, $n = 1$), the trapezoidal rule (closed, $n = 1$), and Simpson's rule (closed, $n = 2$). Using these rules repeatedly, together with some simple extrapolation, is actually more efficient than using higher order rules. If you ever need the higher order rules, you can look them up in a book.

The midpoint rule is is the open Newton-Cotes formula for $n = 1$. The only node is $x_1 = (a + b)/2$, the midpoint of the interval.

With only one point, we can only interpolate a polynomial of degree 0, i.e. a constant. From drawing a picture, we see right away

$$\int_a^b f(x)\,dx \approx (b - a)f(x_1).$$

Amazingly enough, this formula also integrates polynomials of degree 1 correctly. (Lucky coincidence).

The error is

$$\begin{aligned}\text{error} &= \int_a^b f(x)\,dx - (b - a)f(x_1)\\ &= \frac{(b-a)^3}{24} f''(x_1) + \frac{(b-a)^5}{1920} f^{(4)}(x_1) + \dots\end{aligned}$$

assuming that f is smooth enough. (See p. 141/142 in the book for the derivation).

The repeated midpoint rule uses points

$$x_1 = a + h/2, \quad x_2 = a + 3h/2, \quad \ldots, \quad x_n = b - h/2,$$

where $h = (b-a)/n$. The formula is

$$\int_a^b f(x)\,dx \approx h\left[f(x_1) + f(x_2) + \cdots + f(x_n)\right].$$

The error on the subinterval $[a+ih, a+(i+1)h]$ is $(h^3/24)f''(x_i) + \ldots$. When we add up the local errors, we get

$$\text{error} = \frac{h^3}{24}\left[f''(x_1) + f''(x_2) + \cdots + f''(x_n)\right] + \ldots$$

With a little fancy mathematical footwork, we can replace the sum by a single term $nf''(\xi)$, where ξ is some unknown point in $[a,b]$. Since $n = (b-a)/h$, we lose one power of h for a final error estimate of

$$\text{error} \approx \frac{b-a}{24} h^2 f''(\xi).$$

More generally, we can show

$$\text{error} = c_2 h^2 + c_4 h^4 + c_6 h^6 + \ldots$$

Example: Evaluate numerically

$$\int_1^2 \log(x)\,dx.$$

The true answer is $2\log(2) - 1 \approx 0.386294361$.

Let us talk about extrapolation briefly.

Except for evaluating $f(x_i)$, computer time for numerical quadrature is absolutely negligible. There are only a few multiplications and additions. The computer time required to calculate $f(x_i)$ is unknown, and could potentially be large. An efficient algorithm tries to minimize the number of function evaluations, even at the cost of some other calculations.

Therefore, we want to arrange the extrapolation table so that we can re-use points as much as possible.

If we look at the repeated midpoint rule from this angle, it is apparent that we have to *triple* the number of points between levels, instead of doubling it. In this example, I will evaluate f at 9 points, which lets me calculate the repeated midpoint rule for 1, 3, and 9 points.

Extrapolation using factors of 3 works almost the same way as the extrapolation using factors of 2 we had before. If the error is $O(h^p)$, we use

$$\text{true result} \approx \frac{3^p A(h/3) - A(h)}{3^p - 1}.$$

For extrapolating twice, we use first $p = 2$, then $p = 4$ (see form of error above). This gives the following table:

h	repeated midpoint rule		
1	0.405465108		
		0.386473708	
1/3	0.388583864		0.386294955
		0.386297162	
1/9	0.386551240		

h	error		
1	-0.019170747		
		-0.000179347	
1/3	-0.002289503		-0.000000594
		-0.000002801	
1/9	-0.000256879		

□

The trapezoidal rule is the closed Newton-Cotes formula for $n = 1$. The nodes are $x_0 = a$, $x_1 = b$ (the endpoints).

We have two points, so we can interpolate a polynomial of degree 1, i.e. a straight line. By drawing a picture, we see right away that

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\left[f(x_0) + f(x_1)\right].$$

This formula integrates polynomials of degree 1 correctly. The area under the line looks like a trapezoid, which is where the formula got its name.

The error is

$$\begin{aligned}\text{error} &= \int_a^b f(x)\,dx - \frac{b-a}{2}\left(f(x_0) + f(x_1)\right) \\ &= -\frac{(b-a)^3}{12} f''(x_{1/2}) - \frac{(b-a)^5}{860} f^{(4)}(x_{1/2}) + \dots\end{aligned}$$

assuming that f is smooth enough. The notation $x_{1/2}$ is pretty common: it means the point halfway between x_0 and x_1. This is simply the midpoint again, just like in the midpoint rule.

The repeated trapezoidal rule uses points

$$x_0 = a, \quad x_1 = a + h, \quad \dots, \quad x_n = b.$$

where $h = (b-a)/n$. The formula is

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

With the same kind of math as before, the total error estimate becomes

$$\text{error} \approx -\frac{b-a}{12} h^2 f''(\xi),$$

or in general

$$\text{error} = c_2 h^2 + c_4 h^4 + c_6 h^6 + \dots$$

For the single rule, the trapezoidal rule uses twice as many points as the midpoint rule (two versus one), and the error is twice as large. That does not look very promising.

The error for the repeated trapezoidal rule is still twice as large as for the repeated midpoint rule, but the difference in work is negligible ($(n+1)$ points versus n). The fact that we can now extrapolate in steps of 2 instead of 3 makes the repeated trapezoidal rule superior.

The method of using the repeated trapezoidal rule for stepsizes h, $h/2$, $h/4$ etc. and extrapolating is called *Romberg quadrature.*

Example: Same integral as before, again 9 points ($n = 8$). With these points, we can calculate values for 1, 2, 4, and 8 subintervals, and extrapolate more effectively.

h	repeated trapezoidal rule			
1	0.346573590			
		0.385834602		
1/2	0.376019349		0.386287894	
		0.386259563		0.386294309
1/4	0.383699509		0.386294209	
		0.386292043		
1/8	0.385643910			

h	error			
1	0.039720771			
		0.000459759		
1/2	0.010275012		0.000000468	
		0.000034798		0.000000052
1/4	0.002594852		0.000000152	
		0.000002318		
1/8	0.000650451			

□

Simpson's rule is the closed Newton-Cotes formula for $n = 2$. The nodes are $x_0 = a$, $x_1 = (a + b)/2$, $x_2 = b$ (endpoints and midpoint).

We have three points, so we can interpolate a polynomial of degree 2, i.e. a parabola. I will show the details here, so you can see how the derivation of Newton-Cotes rules works in general.

For simplicity, we use $[a, b] = [0, 1]$, and scale the result later (see section 5.3). The points are $x_0 = 0$, $x_1 = 0.5$, $x_2 = 1$.

Method 1. We calculate the interpolating polynomial and integrate that. Any method for finding the polynomial can be used; I will use Lagrange polynomials.

$$\begin{aligned} L_{20}(x) &= \frac{(x-1/2)(x-1)}{(0-1/2)(0-1)} = 2(x-\frac{1}{2})(x-1) \\ L_{21}(x) &= -4x(x-1) \\ L_{22}(x) &= 2x(x-\frac{1}{2}) \end{aligned}$$

We integrate those

$$\begin{aligned} \int_0^1 L_{20}(x)\,dx &= \int_0^1 (2x^2-3x+1)\,dx = \frac{1}{6}, \\ \int_0^1 L_{21}(x)\,dx &= \frac{4}{6}, \\ \int_0^1 L_{22}(x)\,dx &= \frac{1}{6}. \end{aligned}$$

If $p(x)$ interpolates $f(x)$ at x_0, x_1, x_2, we get the formula

$$\begin{aligned} \int_0^1 f(x)\,dx \approx \int_0^1 p(x)\,dx &= \int_0^1 \left[f(x_0)L_{20}(x) + f(x_1)L_{21}(x) + f(x_2)L_{22}(x)\right]\,dx \\ &= \left[f(x_0)\int_0^1 L_{20}(x)\,dx + f(x_0)\int_0^1 L_{20}(x)\,dx + f(x_0)\int_0^1 L_{20}(x)\,dx\right] \\ &= \frac{1}{6}\left[f(x_0) + 4f(x_1) + f(x_2)\right]. \end{aligned}$$

Method 2. We know that we want a formula

$$\int_0^1 f(x)\,dx \approx w_0 f(x_0) + w_1 f(x_1) + w_2 f(x_2)$$

which can integrate polynomials of degree 2 or less correctly. We write out

$$\begin{aligned} w_0 x_0^0 + w_1 x_1^0 + w_2 x_2^0 &= \int_0^1 x^0\,dx \\ w_0 x_0^1 + w_1 x_1^1 + w_2 x_2^1 &= \int_0^1 x^1\,dx \\ w_0 x_0^2 + w_1 x_1^2 + w_2 x_2^2 &= \int_0^1 x^2\,dx \end{aligned}$$

simplified

$$\begin{aligned} w_0 + w_1 + w_2 &= 1 \\ \frac{1}{2}w_1 + w_2 &= \frac{1}{2} \\ \frac{1}{4}w_1 + w_2 &= \frac{1}{3} \end{aligned}$$

This leads to the same result.

When scaling to $[a, b]$, we pick up an extra factor of $(b - a)$ (see section 5.3). The final formula is

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left[f(x_0) + 4f(x_1) + f(x_2)\right].$$

This formula integrates polynomials of degree 2 correctly, by construction, and also works for polynomials of degree 3 (lucky accident, again).

Actually, it can be proved in general that every other Newton-Cotes formula is one degree more accurate than it ought to be. The closed Newton-Cotes rule for $n = 3$ (called the *3/8-rule*) also works for polynomials of degree 3 only. The one after that works for polynomials of degree 5, and so on.

In Germany, Simpson's rule is also known as the *barrel rule.* Barrel makers used it to calculate the volume of a barrel very accurately from the circumference in the middle and at the ends.

The error in Simpson's rule is

$$\begin{aligned} \text{error} &= \int_a^b f(x)\,dx - \frac{b-a}{6}\left(f(x_0) + 4f(x_1) + f(x_2)\right) \\ &= -\frac{(b-a)^5}{2880} f^{(4)}(x_1) + \ldots \end{aligned}$$

assuming that f is smooth enough.

The repeated Simpson rule uses points

$$x_0 = a, \quad x_1 = a + h, \quad \ldots, \quad x_{2n} = b.$$

where $h = (b - a)/(2n)$. The formula is

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots + 2f(x_{2n-2}) + 4f(x_{2n-1}) + f(x_{2n})\right].$$

The total error estimate becomes

$$\text{error} \approx -\frac{b-a}{180} h^4 f^{(4)}(\xi),$$

or in general

$$\text{error} = c_4 h^4 + c_6 h^6 + c_8 h^8 + \ldots$$

How did the number 6 in the single formula turn into 3 in the repeated formula? Likewise, how did the number 2880 in the error estimate turn into 180? Well, the h in the repeated formula corresponds to *half* the subinterval width. The missing factors of 2 and $2^4 = 16$ are hiding in the h and h^4.

Example: Same integral as before, again using 9 points ($n = 8$). With these points, we can calculate values for 1, 2, and 4 subintervals, and extrapolate.

h	repeated Simpson rule		
1/2	0.385834602		
		0.386287894	
1/4	0.386259563		0.386294309
		0.386294209	
1/8	0.386292043		

h	error		
1/2	0.000459759		
		0.000000468	
1/4	0.000034798		0.000000052
		0.000000152	
1/8	0.000002318		

These numbers look remarkably like the extrapolation columns from the repeated trapezoidal rule. This is no coincidence. Let's prove it.

We can use extrapolation on entire formulas, instead of just on the numbers these formulas produce. Suppose we have $f(x)$ given at x_0, x_1, x_2, stepsize h. If $A(2h)$ is the result of using the single trapezoidal rule on $[x_0, x_2]$, and $A(h)$ is the result of using two trapezoidal rules on the subintervals, then

$$\begin{aligned} A(2h) &= h\left[f(x_0) + f(x_2)\right] \\ A(h) &= h\left[\frac{1}{2}f(x_0) + f(x_1) + \frac{1}{2}f(x_2)\right] \\ \text{extrapolated value} &= \frac{4A(h) - A(2h)}{3} \\ &= \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right]. \end{aligned}$$

In other words: one step of extrapolation based on the (single or repeated) trapezoidal rule produces the (single or repeated) Simpson's rule! Extrapolating Simpson's rule produces the rule for $n = 4$, called the *Milne rule*, but I don't want to pursue this. □

5.2.2. Gaussian Quadrature. Gaussian quadrature is based on polynomial interpolation, just like the Newton-Cotes formulas, except the choice of nodes is different.

In the Newton-Cotes formulas, the idea was: let the x_j be given, and determine the w_j so that we can integrate polynomials up to a certain order exactly. With n points, we have n degrees of freedom, and we can hope to integrate polynomials up to degree $(n-1)$ exactly. That is indeed what happens, except that in some formulas we get lucky and can do it up to degree n.

In Gaussian quadrature, we let both the x_j and the w_j be negotiable. With n points, we now have $2n$ degrees of freedom, and we hope to be able to integrate polynomials up to degree $(2n-1)$ exactly. This is indeed possible.

The details of the derivation are beyond the scope of this course. The result is that for quadrature on $[-1, 1]$, the nodes x_i should be chosen to be the zeros of the n-th Legendre polynomial $P_n(x)$ (whatever that is). The weights are then calculated from a system of equations, like before. For quadrature on intervals other than $[-1, 1]$, nodes and weights are scaled (see section 5.3).

A table with some nodes and weights is on page 147 in the textbook.

The error for the n-point rule can be shown to be

$$\frac{(b-a)^{2n+1}(n!)^4}{(2n+1)\left[(2n)!\right]^3} f^{(2n)}(\xi),$$

as long as f has $(2n)$ continuous derivatives. A repeated n-point Gaussian rule would have global error $O(h^{2n})$.

Example: Same integral as before.

For $n = 2$, the scaled nodes and weights are

i	x_i	w_i
1	1.211324865	0.5
2	1.788675135	0.5

(See section 5.3 for the derivation).

The approximate integral is 0.386594944, with an error of 0.00300583. This is very good compared to other rules with only two points.

For $n = 4$, the scaled nodes and weights are

i	x_i	w_i
1	1.069431844	0.173927423
2	1.330009478	0.326072577
3	1.669990522	0.326072577
4	1.930568156	0.173927423

The approximate integral is 0.386294497, with an error of -0.000000136. This is excellent accuracy for only four points. $\square$

5.3. Change of Interval

Assume we have a quadrature rule on the interval $[\alpha, \beta]$ with nodes ξ_i and weights ω_i

$$\int_\alpha^\beta f(x)\,dx \approx \sum \omega_i f(\xi_i).$$

We now want to integrate a function on a different interval $[a, b]$. How do we transform the nodes and weights?

With the substitution

$$\begin{aligned} t &= a + \frac{b-a}{\beta-\alpha}(x-\alpha) \\ x &= \alpha + \frac{\beta-\alpha}{b-a}(t-a) \end{aligned}$$

we get

$$\begin{aligned} \int_a^b f(t)\,dt &= \frac{b-a}{\beta-\alpha}\int_\alpha^\beta f\left(a + \frac{b-a}{\beta-\alpha}(x-\alpha)\right)\,dt \\ &\approx \sum \left(\frac{b-a}{\beta-\alpha}\omega_i\right) f\left(a + \frac{b-a}{\beta-\alpha}(\xi_i-\alpha)\right) \\ &= \sum w_i f(x_i), \end{aligned}$$

so we see

$$\begin{aligned} w_i &= \frac{b-a}{\beta-\alpha}\omega_i \\ x_i &= a + \frac{b-a}{\beta-\alpha}(\xi_i-\alpha) \end{aligned}$$

Example: The nodes and weights for 4-point Gaussian quadrature on $[-1, 1]$ are

i	x_i	w_i
1	-0.861136311594052	0.347854845137454
2	-0.339981043584856	0.652145154862546
3	0.339981043584856	0.652145154862546
4	0.861136311594052	0.347854845137454

If we want to use these for quadrature on $[1, 2]$, we have $\alpha = -1$, $\beta = 1$, $a = 1$, $b = 2$. The formulas become

$$\begin{aligned} w_i &\to \frac{2-1}{1-(-1)}w_i = \frac{1}{2}w_i \\ x_i &\to 1 + \frac{1}{2}(x_i+1) = \frac{3}{2} + \frac{1}{2}x_i \end{aligned}$$

and we get the new nodes/weights I used above

i	x_i	w_i
1	1.069431844	0.173927423
2	1.330009478	0.326072577
3	1.669990522	0.326072577
4	1.930568156	0.173927423

□

5.4. Compund Quadrature Rules and Error Estimates

Read this section in the book. I have included the material from this section in my section 5.2.

5.5. Gauss-Kronrod Quadrature Rules

Before I go into Gauss-Kronrod rules, let me talk a little about the uses of extrapolation.

So far, we have used extrapolation as a way to improve the accuracy of numerical results. A second use is as an error estimate.

Suppose we calculate a result twice, with different step sizes, and extrapolate. Assume the original results are $A(h)$ and $A(h/2)$, and X is the extrapolated value. We expect X to be noticeably more accurate than either one of the original results, so

$$\text{error in } A(h/2) = |A(h/2) - \text{true result}| \approx |A(h/2) - X|.$$

Since X is expected to be more accurate than $A(h/2)$, its error should be even smaller. Thus, we use X as the approximate answer, and $|A(h/2) - X|$ as an error bound. This error bound will usually be quite pessimistic, and we may be able to find a more realistic bound empirically. “Empirically” means by trial and error. We do a whole bunch of test examples and compare the error bound and the actual error, to see if we can derive a more realistic error bound that works most of the time. (Here is where numerical analysis turns into an art, instead of a science).

Now, let us go back to Gaussian quadrature.

As I have discussed before, a good quadrature routine tries to minimize the number of function evaluations.

When we do extrapolation on the Newton-Cotes formulas, we can re-use old points if we decrease the stepsize by a factor of 2 (closed rules) or 3 (open rules). For repeated Gaussian quadrature, there is no regular spacing, and standard extrapolation requires a whole new set of points every time.

Kronrod found a way around this. He started with an n-point Gaussian quadrature rule

$$\int_{-1}^{1} f(x)\,dx \approx \sum w_i f(x_i)$$

and tried to add m other points y_j and find a new formula

$$\int_{-1}^{1} f(x)\,dx \approx \sum a_i f(x_i) + \sum b_j f(x_j)$$

The x_i are fixed, the y_j, a_i and b_j are to be determined. We have $(2m+n)$ degrees of freedom, so we hope we can integrate polynomials up to degree $(2m+n-1)$ exactly.

For the choice $m = (n+1)$ and some values of n, this actually works. The smallest choice that works is $n = 7$, where the details are given in the textbook. Other choices that work are $n = 10$, $n = 15$, $n = 20$, $n = 25$, $n = 30$. (Kronrod must have used up a lot of paper deriving these).

Now, this is not really extrapolation. For the basic Gauss-Kronrod rule, we have a 7-point rule whose error depends on $f^{(14)}$, and a 15-point rule whose error depends on $f^{(23)}$. We cannot combine the two to form a more accurate estimate.

We can still use the two results for an error estimate: we use the 15-point rule as the more accurate result, and the difference between the 7-point rule and the 15-point rule as the error estimate. A more realistic empirical estimate is shown on page 154.

Thus, the $(7,15)$ Gauss-Kronrod pair is really a 15-point rule that comes with an error estimate derived from 7 of these points.

5.6. Automatic and Adaptive Quadrature Algorithms

An *automatic* algorithm is one that will accept a requested accuracy and will attempt to produce a result whose error estimate is less than the requested accuracy. This shifts the burden of error estimation from the user to the subroutine.

An *adaptive* algorithm is one that will try to adapt to the local behavior of the problem. In the case of quadrature, that means that the algorithm will use a small stepsize where the function seems to vary rapidly, and a larger stepsize where the function seems to be smooth.

An automatic adaptive algorithm could be programmed about like this: First, the programmer selects a pair of basic quadrature rules. This could be a Gauss-Kronrod pair, or two Simpson rules with different stepsizes. When applied to a subinterval, this pair of rules will produce an approximate integral, along with an error estimate.

Then, we apply the basic rules to the whole interval. Suppose the requested accuracy is a number ϵ. If the error estimate for the whole interval is less than ϵ, we are done. If not, we divide the interval into two and try to integrate each half with an accuracy of $\epsilon/2$. For some pairs of basic rules, we can even re-use some of the previous points here. We keep subdividing until we achieve the required accuracy on each subinterval. (There should be some other stopping criterion, too, to avoid endless calculations).

The routine `Q1DA` in your textbook uses a similar approach, based on the 15-point Gauss-Kronrod rule.

5.7. Subroutines Q1DA and QK15

Read this section on your own.

5.8. Data Integration

We skip this subsection. (Now, shouldn't that be *Data Quadrature*?)

5.9. Improper Integrals

A *proper* integral is the definite integral of a continuous function over a bounded interval. Anything else is *improper*. Thus, either we have a discontinuous function, or the interval is infinite (or both).

Discontinuities come in many forms, but the types that show up most frequently in practice are simple jumps and points where the function goes to infinity. Jumps are harmless: just subdivide the interval there. Likewise, if the function blows up in the middle somewhere, we subdivide the interval, so that we only need to worry about blowups at the endpoints.

By subdividing the interval, we only need to worry about two types of improper integrals: integrals where the function blows up at one endpoint, and integrals over infinite intervals.

There are three basic ways for handling integrals over infinite intervals:

- Truncate the interval to a finite one.
- Transform the interval to a finite one.
- Use a special quadrature rule for infinite intervals.

Two of these techniques can also be applied to integrals in which the function blows up at an endpoint:

- Use a transformation that makes the singularity disappear.
- Use a special quadrature rule for the singularity.

Let us look at these techniques in more detail.

5.9.1. Truncation.

If the integrand decreases fast enough, we can cut off the integration.

Example: Evaluate $\int_0^\infty e^{-x}\cos^2(x^2)\,dx$ numerically.

We estimate

$$\int_A^\infty e^{-x}\cos^2(x^2)\,dx < \int_0^\infty e^{-x}\,dx = e^{-A}.$$

If we want an absolute error of $\epsilon = 10^{-5}$, we could use $A = 11.5$, since $e^{-11.5} \approx 10^{-5}$. Thus,

$$\int_0^\infty e^{-x}\cos^2(x^2)\,dx \approx \int_0^1 1.5e^{-x}\cos^2(x^2)\,dx,$$

which can be evaluated by a quadrature routine. □

Example: Evaluate $\int_0^\infty \dfrac{dx}{1+x^2}$ numerically.
We estimate

$$\int_A^\infty \frac{dx}{1+x^2} < \int_a^\infty \frac{dx}{x^2} = \frac{1}{A}.$$

For an accuracy of $\epsilon = 10^{-5}$, we would need $A = 10^5$. This does not seem practical. □

5.9.2. Transformation. This technique works both for infinite intervals and for endpoint singularities in the integrand.

Example: Evaluate $\int_0^\infty e^{-x}\cos^2(x^2)\,dx$ numerically.
We could use the transformation $x = -\log t$, so that

$$\int_0^\infty e^{-x}\cos^2(x^2)\,dx = \int_0^1 \cos^2(\log^2 t)\,dt.$$

This does not work, since the integrand oscillates wildly near 0. □

Example: Evaluate $\int_0^\infty e^{-x}\cos^2(x^2)\,dx$ numerically.
Let us try $x = -2\log t$. In this case

$$\int_0^\infty e^{-x}\cos^2(x^2)\,dx = \int_0^1 2t\cos^2(4\log^2 t)\,dt.$$

This works better. The integrand still oscillates rapidly, but at least it goes to zero near 0. (You can't get rid of the oscillations, since the original function oscillates at infinity). □

Example: Evaluate $\int_0^1 \dfrac{\cos x}{\sqrt{x}}$ numerically.
With the substitution $x = t^2$, we get

$$\int_0^1 \frac{\cos x}{\sqrt{x}} = \int_0^1 \cos t^2\,dt,$$

which is easy to evaluate numerically. □

5.9.3. Weighted Quadrature Rules. Many improper integrals are of the form

$$\int_a^b f(x)w(x)\,dx,$$

where $w(x)$ is a *weight function*, which could be "bad", and $f(x)$ is a "nice" function. Standard weight functions are powers of x, logarithms, and exponentials, or combinations of these.

The idea is to derive quadrature formulas which incorporate the weight functions, of the form

$$\int_a^b f(x)w(x)\,dx \approx \sum w_i f(x_i).$$

If a suitable weight function can be extracted from the integrand, this approach would be my first choice.

Let us look at a couple of examples.

Example: Evaluate $\int_0^1 \dfrac{\cos x}{\sqrt{x}}$ by a modified Simpson rule. Here $f(x) = \cos x$, and $w(x) = 1/\sqrt{x}$.

The modified Simpson rule is derived in the same way the standard Simpson rule was derived. We pick $x_0 = 0$, $x_1 = 1/2$, $x_2 = 1$, and try to find weights w_i so that the rule

$$\int_0^1 f(x)\frac{1}{\sqrt{x}}\,dx \approx w_0 f(x_0) + w_1 f(x_1) + w_2) f(x_2)$$

is exact for $f(x) = x^0$, x^1, x^2. This leads to a system of equations

$$\begin{aligned} w_0 x_0^0 + w_1 x_1^0 + w_2 x_2^0 &= \int_0^1 x^0 \frac{1}{\sqrt{x}}\,dx \\ w_0 x_0^1 + w_1 x_1^1 + w_2 x_2^1 &= \int_0^1 x^1 \frac{1}{\sqrt{x}}\,dx \\ w_0 x_0^2 + w_1 x_1^2 + w_2 x_2^2 &= \int_0^1 x^2 \frac{1}{\sqrt{x}}\,dx, \end{aligned}$$

simplified

$$\begin{aligned} w_0 + w_1 + w_2 &= 2 \\ \frac{1}{2} w_1 + w_2 &= \frac{2}{3} \\ \frac{1}{4} w_1 + w_2 &= \frac{2}{5} \end{aligned}$$

with solution $w_0 = 12/15$, $w_1 = 16/15$, $w_2 = 2/15$.

We get

$$\int_0^1 \frac{\cos x}{\sqrt{x}} \approx \frac{12}{15}\cos 0 + \frac{16}{15}\cos\frac{1}{2} + \frac{2}{15}\cos 1 = 1.808128373\ldots$$

The true value is 1.809048476, the error is 0.000920103, which is not bad. □

Example: Evaluate $\int_0^\infty e^{-x} x^{1.2}\,dx$ numerically (see book, page 167).

The modified Gaussian quadrature rules with weight function e^{-x} on the interval $[0, \infty)$ are called *Gauss-Laguerre rules*. Look at the derivation in the book (pages 166/167).

The numerical value is 1.088468113, the exact value is 1.101802491, so the error is 0.0133344. Pretty good for only two points.

Remark: my textbook gives the exact integral as 1.046. That is wrong: 1.046 ... is the value for a power of 1.1. □