LECTURE 4: Numerical integration

September 26, 2011

1 Introduction

The *magnetic field* induced by a current flowing in a circular loop of wire has intensity

$$H(x) = \frac{4Ir}{r^2 - x^2} \int_0^{\pi/2} \left[1 - \left(\frac{x}{r} \right)^2 \sin^2 \theta \right]^{1/2} d\theta$$

where I is the current, r is the radius of the loop, and x is the distance from the center of the loop. The integral is not expressible in terms of familiar functions, but it can be evaluated *numerically*.

In this chapter, we discuss the subject of numerical integration, also called *quadrature*. The numerical methods that are covered here are based, one way or another, on adding up the value of the integrand at a sequence of points along the x-axis. The idea is to obtain the integral as accurately as possible with the smallest number of function evaluations. The Richardson extrapolation formalism (discussed in the previous chapter) is used to construct a general method known as the Romberg algorithm. Multidimensional integration is touched only briefly here. We return to the subject later in connection with Monte Carlo methods.

1.1 Lower and upper sums

The definite integral is defined by means of two concepts: the *lower sums* and the *upper sums* of f. These are approximations of the area under the graph of f on a closed interval $[a,b]$.

Let P be a *partition* of the interval $[a,b]$:

$$P = \{a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b\}$$

with partition points x_i $(0 \leq i \leq n)$ that divide the interval $[a,b]$ into n subintervals $[x_i, x_{i+1}]$.

Denote by m_i the *greatest lower bound* (*infimum*) of $f(x)$ on $[x_i, x_{i+1}]$:

$$m_i = \inf\{ f(x) : x_i \leq x \leq x_{i+1} \}$$

Likewise, denote by M_i the *least upper bound* (*supremum*) of $f(x)$ on $[x_i, x_{i+1}]$:

$$M_i = \sup\{ f(x) : x_i \leq x \leq x_{i+1} \}$$

The *lower sums* and *upper sums* of f corresponding to a given partition P are defined to be

$$L(f;P) = \sum_{i=0}^{n-1} m_i (x_{i+1} - x_i)$$
$$U(f;P) = \sum_{i=0}^{n-1} M_i (x_{i+1} - x_i)$$

If f is positive, then these two quantities can be interpreted as estimates of the area under the curve for f.

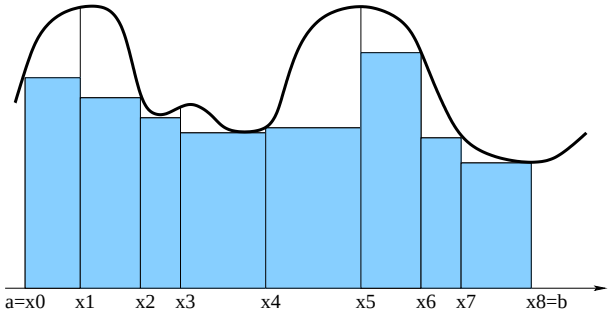

Figure 1: Lower sums.

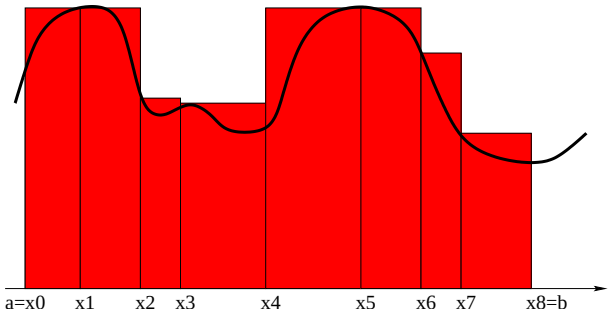

Figure 2: Upper sums.

1.2 Riemann-integrable functions

Consider the least upper bound obtained when P is allowed to range over all partitions of the interval $[a,b]$. This is abbreviated $\sup_P L(f;P)$. Similarly, the greatest lower bound, when P ranges over all partitions P on $[a,b]$, is abbreviated $\inf_P L(f;P)$.

If these two numbers are the same,

$$\sup_P L(f;P) = \inf_P L(f;P)$$

then f is Riemann integrable on $[a,b]$.

Theorem. Every continuous function defined on a closed interval of the real line is Riemann integrable.

The Riemann integral of a continuous function on $[a,b]$ can be obtained by two limits:

$$\lim_{n\to\infty} \sup_P L(f;P_n) = \int_a^b f(x)dx = \lim_{n\to\infty} \inf_P L(f;P_n)$$

Here the length of the largest subinterval in P_n converges to zero as $n \to \infty$.

Direct translation of this definition into a numerical algorithm is possible, but the resulting algorithm converges very slowly. The situation can be improved dramatically by using more sophisticated methods such as the *Romberg's algorithm*.

2 Trapezoid rule

The *trapezoid method* is based on an estimation of the area under a curve using trapezoids. First the interval $[a,b]$ is divided into subintervals according to the partition $P = \{a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b\}$.

A typical trapezoid has the subinterval $[x_i, x_{i+1}]$ as its base, and the two vertical sides are $f(x_i)$ and $f(x_{i+1})$. The area is equal to the base times the average height. Thus we obtain the *basic trapezoid rule:*

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx A_i = \frac{1}{2}(x_{i+1} - x_i)[f(x_i) + f(x_{i+1})]$$

The total area under all of the trapezoids is given by the *composite trapezoid rule:*

$$\int_a^b f(x)dx \approx T(f;P) = \sum_{i=0}^{n-1} A_i = \frac{1}{2}\sum_{i=0}^{n-1}(x_{i+1} - x_i)[f(x_i) + f(x_{i+1})]$$

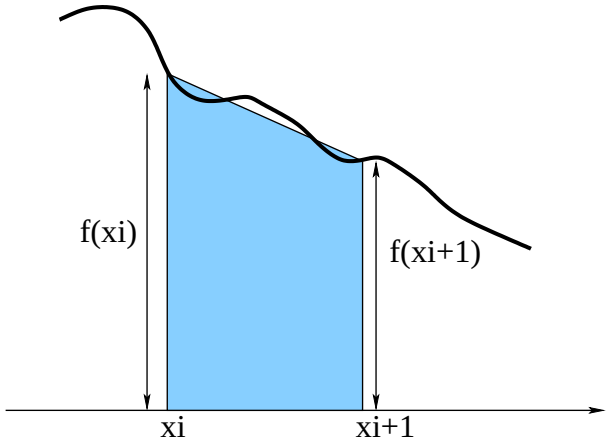

Figure 3: The basic trapezoid rule.

2.1 Uniform spacing

In practice, the trapezoid rule is used with uniform partition of the interval. The division points x_i are equally spaced: $x_i = a + ih$, where $h = (b-a)/n$ and $0 \leq i \leq n$. This gives a simpler formula

$$\int_a^b f(x)dx \approx T(f;P) = \frac{h}{2}\sum_{i=0}^{n-1}[f(x_i) + f(x_{i+1})]$$

A computationally preferable form is written as

$$\int_a^b f(x)dx \approx T(f;P) = h\left\{\sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}[f(x_0) + f(x_n)]\right\}$$

Implementation

The following procedure *Trapezoid* uses the trapezoid method with uniform spacing to estimate the definite integral of a function f on a closed interval $[a,b]$.

```
double Trapezoid(double (*f)(double x),
                 double a, double b, int n)
{
  int i;
  double h, result;

  h = (b-a)/(double)n;
  result = 0.5*(f(a)+f(b));

  for(i=1; i<n; i++)
    result += f(a+i*h);

  return(h*result);
}
```

Output

For $f = e^{-x^2}$ on [0,1], we obtain the result 0.7468071 with $n = 60$ and 0.7468238 with $n = 500$. The correct answer is 0.7468241 with seven decimals.

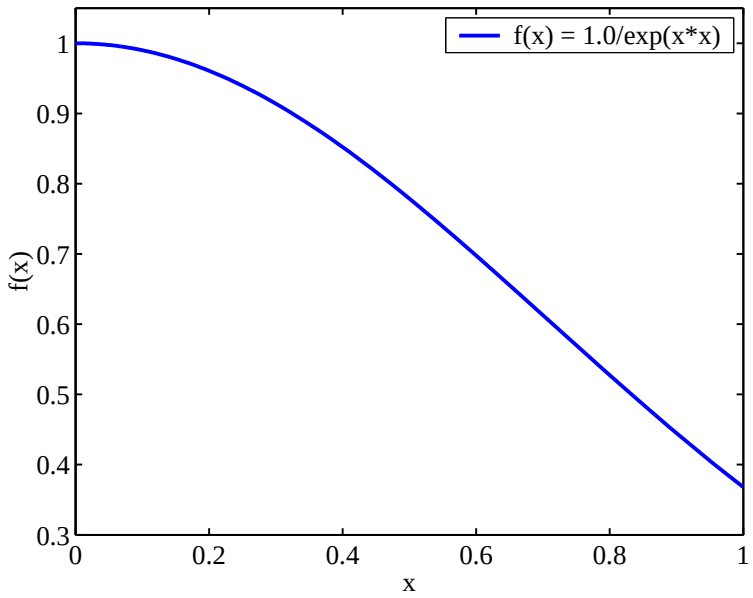

Figure 4: Plot of the function $f = e^{-x^2}$.

2.2 Error analysis

Theorem on precision of trapezoid rule.
If f'' exists and is continuous on the interval $[a,b]$, and if the composite trapezoid rule T with uniform spacing h is used to estimate the integral $I = \int_a^b f(x)dx$, then for some ξ in (a,b),

$$I - T = -\frac{1}{12}(b-a)h^2 f''(\xi) = O(h^2)$$

To prove this use the error formula for polynomial interpolation with $n = 1$, $x_0 = 0$, and $x_1 = 1$: $f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{i=0}^{n}(x - x_i)$ plus mean-value theorem $\int_a^b f(x)g(x)dx = f(\xi)\int_a^b g(x)dx$. **Example.** If the trapezoid rule is used to compute

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

with an error at most $\frac{1}{2} \times 10^{-4}$, how many points should be used?

The error formula is

$$I - T = -\frac{1}{12}(b-a)h^2 f''(\xi)$$

In this example,
$f(x) = e^{-x^2}$
$f'(x) = -2xe^{-x^2}$
$f''(x) = (4x^2 - 2)e^{-x^2}$

Thus, $|f''(x)| \leq 2$ on the interval $[0,1]$, and

$$|I - T| \leq \frac{1}{6}h^2$$

To have an error of at most $\frac{1}{2} \times 10^{-4}$, we require that $h = (b-a)/n \leq 0.01732$ or $n \geq 58$.

Using the procedure *Trapezoid* with $n = 58$, we obtain 0.7468059 which is incorrect in the fifth digit.

2.3 Recursive trapezoid formula

We now introduce a formula for the composite trapezoid rule when the interval $[a,b]$ is subdivided into 2^n equal parts. We have

$$T(f;P) = h\sum_{i=1}^{n-1} f(x_i) + \frac{h}{2}[f(x_0)+f(x_n)]$$
$$= h\sum_{i=1}^{n-1} f(a+ih) + \frac{h}{2}[f(a)+f(b)]$$

If we now replace n by 2^n and use $h=(b-a)/2^n$, the formula becomes

$$R(n,0) = h\sum_{i=1}^{2^n-1} f(a+ih) + \frac{h}{2}[f(a)+f(b)]$$

Here we have introduced the notation $R(n,0)$ which is used in the next section in connection with the Romberg algorithm. The notation $R(n,0)$ denotes the result of applying the composite trapezoid rule with 2^n equal subintervals.

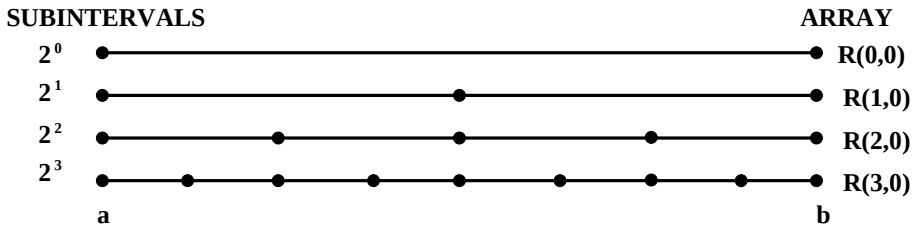

Figure 5: The recursive trapezoid rule.

For the Romberg algorithm, we need a method for computing $R(n,0)$ from $R(n-1,0)$ without unnecessary evaluations of f. We use the identity

$$R(n,0) = \frac{1}{2}R(n-1,0) + [R(n,0) - \frac{1}{2}R(n-1,0)]$$

It is desirable to compute the bracketed expression with as little extra computation as possible. We have

$$R(n,0) = h\sum_{i=1}^{2^n-1} f(a+ih) + C$$
$$R(n-1,0) = 2h\sum_{j=1}^{2^{n-1}-1} f(a+2jh) + 2C$$

where

$$C = \frac{h}{2}[f(a)+f(b)]$$

Notice that the size of the subintervals for $R(n-1,0)$ are twice the size of those for $R(n,0)$.

By subtraction we get

$$R(n,0) - \frac{1}{2}R(n-1,0) = h\sum_{i=1}^{2^n-1} f(a+ih) - h\sum_{j=1}^{2^{n-1}-1} f(a+2jh)$$
$$= h\sum_{k=1}^{2^{n-1}} f[a+(2k-1)h]$$

Here we have taken into account that each term in the first sum that corresponds to an *even* value of i is *canceled* by a term in the second sum. This leaves only terms that correspond to *odd* values of i.

This gives us the **recursive trapezoid formula.**

If $R(n-1,0)$ is available, then $R(n,0)$ can be computed by the formula

$$R(n,0) = \frac{1}{2}R(n-1,0) + h\sum_{k=1}^{2^{n-1}} f[a+(2k-1)h] \quad (n \geq 1)$$

using $h = (b-a)/2^n$ and $R(0,0) = \frac{1}{2}(b-a)[f(a)+f(b)]$.

Implementation

The following procedure *RecTrapezoid* uses the recursive trapezoid formula to calculate a sequence of approximations to the definite integral of function f on the interval $[a,b]$.

```
void RecTrapezoid(REAL (*f)(REAL x), REAL a, REAL b,
                  int n, REAL *R)
{
  int i, j, k, kmax=1;
  REAL h, sum;

  h = b-a;
  /* Value of R(0,0) */
  R[0] = (h/2.0)*(f(a)+f(b));

  /* Successive approximations R(n,0) */
  for(i=1; i<=n; i++) {
    h = h/2.0;
    sum = 0;
    kmax = kmax*2;
    for(k=1; k<=kmax-1; k+=2)
      sum += f(a+k*h);
    R[i] = 0.5*R[i-1]+sum*h;
  }
}
```

Example.

The procedure *RecTrapezoid* was used to calculate the value of π by evaluating the integral

$$\pi \approx \int_0^1 \frac{4}{1+x^2} dx$$

The following output is obtained with $n = 9$:

```
R(0,0) =   3.000000000000
R(1,0) =   3.100000000000
R(2,0) =   3.131176470588
R(3,0) =   3.138988494491
R(4,0) =   3.140941612041
R(5,0) =   3.141429893175
R(6,0) =   3.141551963486
R(7,0) =   3.141582481064
R(8,0) =   3.141590110458
R(9,0) =   3.141592017807
```

The approximation is correct in the sixth decimal. The correct value is 3.141592654 with nine decimals.

Next, we introduce the Romberg algorithm which uses the numbers $R(n,0)$ and improves the estimates in a fashion similar to that of Richardson extrapolation.

3 Romberg algorithm

3.1 Description

The Romberg algorithm produces a triangular array of numbers, all of which are numerical estimates of the definite integral $\int_a^b f(x)dx$. The array is denoted by

$$\begin{array}{llllll} R(0,0) & & & & & \\ R(1,0) & R(1,1) & & & & \\ R(2,0) & R(2,1) & R(2,2) & & & \\ R(3,0) & R(3,1) & R(3,2) & R(3,3) & & \\ \vdots & \vdots & \vdots & \vdots & & \\ R(n,0) & R(n,1) & R(n,2) & R(n,3) & \ldots & R(n,n) \end{array}$$

The first column contains estimates $R(k,0)$ obtained by applying the trapezoid rule with 2^k equal subintervals. The first one is obtained using the formula

$$R(0,0) = \frac{1}{2}(b-a)[f(a)+f(b)]$$

$R(n,0)$ is obtained easily from $R(n-1,0)$ using the formula developed in the previous section:

$$R(n,0) = \frac{1}{2}R(n-1,0) + h\sum_{k=1}^{2^{n-1}} f[a+(2k-1)h]$$

where $h = (b-a)/2^n$ and $n \geq 1$.

The second and successive columns are generated by the following extrapolation formula:

$$R(n,m) = R(n,m-1) + \frac{1}{4^m-1}[R(n,m-1) - R(n-1,m-1)]$$

with $n \geq 1$ and $m \geq 1$. This formula results from the application of the Richardson extrapolation theorem.

3.2 Derivation of the Romberg algorithm

To briefly sketch where the iterative equation behind Romberg algorithm comes from we start from the following formula where the error is expressed in the trapezoid rule with 2^{n-1} equal subintervals

$$\int_a^b f(x)dx = R(n-1,0) + a_2h^2 + a_4h^4 + a_6h^6 + \dots$$

This is one form of Euler-Maclaurin formula (see the next page). (A proof can be found for instance in Gregory, R.T, and Kearney, D., *A Collection of Matrices for Testing Computational Algorithms* (Wiley, 1969).) Here $h = (b-a)/2^n$ and the coefficients a_i depend on f but not on h. $R(n-1,0)$ is one of the trapezoidal elements in the Romberg array. For our purposes, it is not necessary to know the definite expressions for the coefficients. For the theory to work smoothly, we assume that f possesses derivatives of all orders on the interval $[a,b]$.

Now recall the theory of Richardson extrapolation. We can use the same procedure here due to the form of the equation above. Replacing n with $n+1$ and h with $h/2$, we have

$$\int_a^b f(x)dx = R(n,0) + \frac{1}{4}a_2h^2 + \frac{1}{16}a_4h^4 + \frac{1}{64}a_6h^6 + \dots$$

Subtracting the first equation from the second equation multiplied by 4 gives

$$\int_a^b f(x)dx = R(n,1) - \frac{1}{4}a_4h^4 - \frac{5}{16}a_6h^6 - \dots$$

where

$$R(n,1) = R(n,0) + \frac{1}{3}[R(n,0) - R(n-1,0)] \quad (n \geq 1)$$

Note that this is the first case ($m = 1$) of the extrapolation formula that is used to generate the Romberg array.

The term $R(n,1)$ should be considerably more accurate than $R(n,0)$ or $R(n-1,0)$ since the error series is now $O(h^4)$. This process can be repeated to further eliminate higher terms in the error series.

The only assumption made is that the first equation with the error series is valid for the function f. In practice, we use only a modest number of rows in the Romberg algorithm, which means that the assumption is likely to be valid. The situation is governed by a theorem called the Euler-Maclaurin formula.

3.3 Euler-Maclaurin formula and error term

Theorem.
If $f^{(2m)}$ exists and is continuous on the interval $[a,b]$, then

$$\int_a^b f(x)dx = \frac{h}{2}\sum_{i=0}^{n-1}[f(x_i)+f(x_{i+1})]+E$$

where $h=(b-a)/n$, $x_i = a+ih$ for $0 \le i \le n$, and

$$E = \sum_{k=1}^{m-1} A_{2k}h^{2k}[f^{(2k-1)}(a)-f^{(2k-1)}(b)] - A_{2m}(b-a)h^{2m}f^{(2m)}(\xi)$$

for some ξ in the interval (a,b).

In this theorem, the A_k's are constants and they can be defined by the equation (see e.g. D.M. Young and R.T. Gregory, A survey of Numerical Mathematics, Vol.1, p. 374, (Dover) for details):

$$\frac{x}{e^x-1} = \sum_{k=0}^{\infty} A_k x^k$$

The points to notice are that the right-hand side of the Euler-Maclaurin formula contains the trapezoid rule and an error term E. Furthermore, the error term can be expressed as a finite sum in ascending powers of h^2.

This theorem gives the formal justification to the Romberg algorithm.

3.4 Implementation

The following procedure *Romberg* uses the extrapolation formula to calculate numerical estimates of the definite integral $\int_a^b f(x)dx$. The input is the function f, the endpoints of the interval $[a,b]$, the number of iterations n and the Romberg array $(R)_{0:n\times 0:n}$.

```
void Romberg(double (*f)(double x),
             double a, double b, int n,
             double R[][MAXN])
{
  int i, j, k, kmax=1;
  double h, sum;

  h = b-a;
  R[0][0] = (h/2.0)*(f(a)+f(b));

  for(i=1; i<=n; i++) {
    h = h/2.0;
    sum = 0;
    kmax = kmax*2;

    /* Trapezoidal estimate R(i,0) */
    for(k=1; k<=kmax-1; k+=2)
      sum += f(a+k*h);
    R[i][0] = 0.5*R[i-1][0]+sum*h;

    /* Successive R(i,j) */
    for(j=1; j<=i; j++)
      R[i][j] = R[i][j-1]
              +(R[i][j-1]-R[i-1][j-1])/(pow(4.0,(double)j)-1.0);
  }
}
```

Here the elements of the array $R(i,j)$ are computed row by row up to the specified number of rows, n (this number need not to be very large because the method converges very quickly, e.g., $n = 4\ldots5$ is often suitable).

Output

Using same example as before, we calculate the value of π by evaluating the integral

$$\pi \approx \int_0^1 \frac{4}{1+x^2} dx$$

(Correct value is 3.141592654.)

The output using double-precision:

```
3.0000000000
3.1000000000 3.1333333333
3.1311764706 3.1415686275 3.1421176471
3.1389884945 3.1415925025 3.1415940941 3.1415857838
3.1409416120 3.1415926512 3.1415926611 3.1415926384 3.1415926653
3.1414298932 3.1415926536 3.1415926537 3.1415926536 3.1415926536 3.1415926536
```

And using single-precision:

```
3.0000000000
3.0999999046 3.1333332062
3.1311764717 3.1415686607 3.1421177387
3.1389884949 3.1415925026 3.1415941715 3.1415858269
3.1409416199 3.1415927410 3.1415927410 3.1415927410 3.1415927410
3.1414299011 3.1415927410 3.1415927410 3.1415927410 3.1415927410 3.1415927410
```

We notice that the algorithm converges very quickly (with double precision, $n = 5$ is enough to obtain nine decimals of precision). With single precision, the accuracy of the calculation is limited to six decimals.

4 Adaptive Simpson's scheme

We now proceed to discussing a method known as the *Simpson's rule* and develop an *adaptive* scheme for obtaining a numerical approximation for the integral $\int_a^b f(x)dx$. In the adaptive algorithm, the partitioning of the interval $[a,b]$ is not selected beforehand but automatically determined.

4.1 Simpson's rule

A considerable improvement over the trapezoid rule can be achieved by approximating the function within each of the n subintervals by some polynomial. When second-order polynomials are used, the resulting algorithm is called the Simpson's rule.

Dividing the interval $[a,b]$ into two equal subintervals with partition points a, $a+h$ and $a+2h=b$ results in the **basic Simpson's rule**:

$$\int_a^{a+2h} f(x)dx \approx \frac{h}{3}[f(a)+4f(a+h)+f(a+2h)]$$

Proof. Approximating the function with a polynomial of degree ≤ 2 we can write $\int_a^b f(x)dx \approx Af(a)+Bf(\frac{a+b}{2})+Cf(b)$, where $f(x)$ is assumed continuous on the interval $[a,b]$. The coefficients A, B and C are chosen such that the approximation will give correct values for the integral when f is aquadratic polynomial. Let $a=-1$ and $b=1$: $\int_{-1}^1 f(x)dx \approx Af(-1)+Bf(0)+Cf(1)$. The following equations must hold:
$f(x)=1: \int_{-1}^1 dx = 2 = A+B+C$
$f(x)=x: \int_{-1}^1 xdx = 0 = -A+C$
$f(x)=x^2: \int_{-1}^1 x^2dx = 2/3 = A+C$
$\Rightarrow A=1/3$, $B=4/3$, and $C=1/3$. $\Rightarrow \int_{-1}^1 f(x)dx \approx \frac{1}{3}[f(-1)+4f(0)+f(1)]$. Using the linear mapping $y=(b-a)/2+(a+b)/2$ from $[-1,1]$ to $[a,b]$ we obtain the basic Simpson's rule: $\int_a^b f(x)dx \approx \frac{1}{6}(b-a)[f(a)+4f(\frac{a+b}{2})+f(b)]$. □

The error can be estimated using Taylor series:

$$f(a+h) = f+hf'+\frac{1}{2!}h^2f''+\frac{1}{3!}h^3f'''+\ldots$$
$$f(a+2h) = f+2hf'+2h^2f''+\frac{4}{3}h^3f'''+\ldots$$

From these we obtain a series representation for the right-hand side of the Simpson's rule:

$$\frac{h}{3}[f(a)+4f(a+h)+f(a+2h)] = 2hf+2h^2f'+\frac{4}{3}h^3f''+\frac{2}{3}h^4f'''+\ldots(*)$$

Now consider the left-hand side.

$$\int_a^{a+2h} f(x)dx = F(a+2h)-F(a)$$

with

$$F(x) = \int_a^x f(t)dt$$

Thus, $F'=f$, $F''=f'$, $F'''=f''$ and so on.

Now use the Taylor series for $F(a+2h)$:

$$F(a+2h) = F(a) + 2hF'(a) + 2h^2F''(a) + \frac{4}{3}h^3F'''(a) + \ldots$$
$$= F(a) + 2hf(a) + 2h^2f'(a) + \frac{4}{3}h^3f''(a) + \ldots (**)$$

$F((a) = 0$, so $F(a+2h) = \int_a^{a+2h} f(x)dx$.
Subtracting (**) from (*) gives us an estimate of the error:

$$\int_a^{a+2h} f(x)dx - \frac{h}{3}[f(a) + 4f(a+h) + f(a+2h)] = -\frac{h^5}{90}f^{(4)} - \ldots$$

This is due to the fact that all lower order terms cancel out.

So, the **basic Simpson's rule** over the interval $[a,b]$ is

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

with the error term

$$-\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi)$$

which is $O(h^5)$ for some ξ in (a,b).

4.2 Adaptive Simpson's algorithm

In the adaptive process, the interval $[a,b]$ is first divided into two subintervals. Then we decide whether each of the two subintervals is further divided into more subintervals. The process continues until some specified accuracy is reached.

We now develop the test for deciding whether subintervals should continue to be divided. The Simpson's rule over the interval $[a,b]$ can be written as

$$I \equiv \int_a^b f(x)dx = S(a,b) + E(a,b)$$

where S is given by the basic Simpson's rule

$$S(a,b) = \frac{(b-a)}{6}[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)]$$

and E is the error term

$$E(a,b) = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi)$$

We assume here that $f^{(4)}$ remains constant throughout the interval $[a,b]$. Letting $h = b-a$, we have

$$I = S^{(1)} + E^{(1)}$$

where

$$S^{(1)} = S(a,b) \quad \text{and} \quad E^{(1)} = -\frac{1}{90}\left(\frac{h}{2}\right)^5 f^{(4)}$$

Two applications of the Simpson's rule give

$$I = S^{(2)} + E^{(2)}$$

where

$$S^{(2)} = S(a,c) + S(c,b)$$

and

$$E^{(2)} = -\frac{1}{90}\left(\frac{h/2}{2}\right)^5 f^{(4)} - \frac{1}{90}\left(\frac{h/2}{2}\right)^5 f^{(4)} = \frac{1}{16}E^{(1)}$$

Here $c = (a+b)/2$.

We can now subtract the first evaluation from the second one:

$$\begin{aligned} I - I = 0 &= S^{(2)} + E^{(2)} - S^{(1)} - E^{(1)} \\ S^{(2)} - S^{(1)} &= E^{(2)} - E^{(1)} = 15E^{(2)} \\ E^{(2)} &= \frac{1}{15}(S^{(2)} - S^{(1)}) \end{aligned}$$

We can now write the second application of the Simpson's rule as

$$I = S^{(2)} + E^{(2)} = S^{(2)} + \frac{1}{15}(S^{(2)} - S^{(1)})$$

This can be used to evaluate the quality of our approximation of I. We can use the following inequality to test whether to continue the splitting process:

$$\frac{1}{15}|S^{(2)} - S^{(1)}| < \varepsilon$$

If the test is not satisfied, the interval $[a,b]$ is split into two subintervals $[a,c]$ and $[c,b]$. On each of these subintervals we perform the test again with ε replaced by $\varepsilon/2$ so that the resulting tolerance will be ε over the entire interval $[a,b]$.

4.3 Algorithm

The adaptive Simpson's algorithm is programmed using a **recursive procedure**.

We define two variables `one_simpson` and `two_simpson` that are used to calculate two Simpson approximations: one with two subintervals another with four half-width subintervals.

`one_simpson` is given by the basic Simpson's rule:

$$S^{(1)} = S(a,b) = \frac{h}{6}[f(a) + 4f(c) + f(b)]$$

`two_simpson` is given by two Simpson's rules:

$$S^{(2)} = S(a,c) + S(c,b) = \frac{h}{12}[f(a) + 4f(d) + 2f(c) + 4f(e) + f(b)]$$

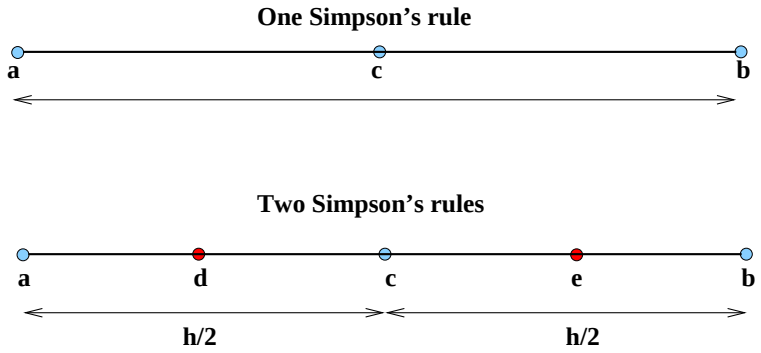

Figure 6: Simpson's rules used in the adaptive algorithm.

After evaluating $S^{(1)}$ and $S^{(2)}$, we use the division test

$$\frac{1}{15}|S^{(2)} - S^{(1)}| < \varepsilon$$

to check whether to divide the subintervals further.

If the criterion is not fulfilled, the procedure calls itself with `left_simpson` corresponding to the left side $[a,c]$ and `right_simpson` corresponding to the right side $[c,b]$. At each recursive call, the value of ε is divided in half. Variable `level` keeps track of the level of the current interval division. The maximum allowed level is given by `level_max`.

The final estimate for each subinterval is given by the Two Simpson's rules (when the criterion is fulfilled). At each upper level, the answer is given by the sum of the left side and the right side.

Implementation

The following C implementation of the recursive procedure *Simpson* calculates the definite integral $\int_a^b f(x)dx$ for the function f specified by an external function f (given as input).

```
double Simpson(double a, double b, double eps,
               int level, int level_max)
{
  int i, j, k, kmax=1;
  double c, d, e, h, result;
  double one_simpson, two_simpson;
  double left_simpson, right_simpson;

  h = b-a;
  c = 0.5*(a+b);
  one_simpson = h*(f(a)+4.0*f(c)+f(b))/6.0;
  d = 0.5*(a+c);
  e = 0.5*(c+b);
  two_simpson = h*(f(a)+4.0*f(d)+2.0*f(c)+4.0*f(e)+f(b))/12.0;
  /* Check for level */
  if(level+1 >= level_max) {
    result = two_simpson;
    printf("Maximum level reached\n");
  }
  else{
    /* Check for desired accuracy */
    if(fabs(two_simpson-one_simpson) < 15.0*eps)
      result = two_simpson + (two_simpson-one_simpson)/15.0;
    /* Divide further */
    else {
      left_simpson = Simpson(a,c,eps/2.0,level+1,level_max);
      right_simpson = Simpson(c,b,eps/2.0,level+1,level_max);
      result =  left_simpson + right_simpson;
    }
  }
  return(result);
}
```

Application example

As an example, the program is used to calculate the integral

$$\int_0^{2\pi} \left[\frac{cos(2x)}{e^x} \right] dx$$

The desired accuracy is set to $\varepsilon = \frac{1}{2} \times 10^{-4}$. An external function procedure is written for f and its name is given as the first argument to *Simpson*.

Running the program with double precision floating-point numbers, we obtain the result 0.1996271. It is correct within the tolerance we set beforehand.

We can use Matlab or Maple to determine the correct answer. The following Matlab commands

```
syms t
res = int(cos(2*t)./exp(t),0,2*pi)
eval(res)
```

first return the exact value $\frac{1}{5}(1 - e^{-2\pi})$ and then evaluate the integral giving (with ten digits) 0.19962 65115.

5 Gaussian quadrature formulas

We have already seen that the various numerical integration formulas all have the common feature that the integral is approximated by the sum of its functional values at a set of points, multiplied by certain aptly chosen weighting coefficients. We have seen that by a better selection of the weights, we can gain integration formulas of higher and higher order. The idea of *Gaussian quadrature formulas* is to give ourselves the freedom to choose not only the weighing factors but also the points where the function values are calculated. They will no longer be equally spaced.

Now the catch: high order is not the same as high accuracy! High order translates to high accuracy only when the integrand is very smooth, in the sense of being well-approximated by a polynomial.

5.1 Description

Most numerical integration formulas conform to the following pattern:

$$\int_a^b f(x)dx \approx A_0 f(x_0) + A_1 f(x_1) + \ldots + A_n f(x_n)$$

To use such a formula, it is only necessary to know the nodes $x_0, x_1, \ldots, x_n$ and the weights $A_0, A_1, \ldots, A_n$.

The theory of polynomial interpolation has greatly influenced the development of integration formulas. If the nodes have been fixed, there is a corresponding Lagrange interpolation formula:

$$p_n(x) = \sum_{i=0}^{n} l_i(x) f(x_i) \quad \text{where} \quad l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j} \right)$$

This formula provides a polynomial p that interpolates f at the nodes. If circumstances are favorable, p will be a good approximation to f, and also

$$\int_a^b f(x)dx \approx \int_a^b p(x)dx = \sum_{i=0}^{n} f(x_i) \int_a^b l_i(x)dx = \sum_{i=0}^{n} A_i f(x_i)$$

where

$$A_i = \int_a^b l_i(x)dx$$

Example.

Determine the quadrature formula when the interval is $[-2,2]$ and the nodes are -1, 0 and 1.

Solution. The cardinal functions are given by

$$l_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j} \right)$$

We get $l_0(x) = \frac{1}{2}x(x-1)$, $l_1(x) = -(x+1)(x-1)$ and $l_2(x) = \frac{1}{2}x(x+1)$.

The weights are obtained by integrating these functions. For example

$$A_0 = \int_{-2}^{2} l_0(x)dx = \frac{1}{2}\int_{-2}^{2} (x^2 - x)dx = \frac{8}{3}$$

Similarly, $A_1 = -4/3$ and $A_2 = 8/3$.

The quadrature formula is

$$\int_{-2}^{2} f(x)dx \approx \frac{8}{3}f(-1) - \frac{4}{3}f(0) + \frac{8}{3}f(1)$$

5.2 Gaussian nodes and weights

The mathematician Karl Friedrich Gauss (1777-1855) discovered that by a special placement of the nodes the accuracy of the numerical integration process could be greatly increased.

Gaussian quadrature theorem.

Let q be a nontrivial polynomial of degree $n+1$ such that

$$\int_a^b x^k q(x)dx = 0 \quad (0 \leq k \leq n)$$

Let $x_0, x_1, \ldots, x_n$ be the zeros of q. The formula

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} A_i f(x_i) \quad \text{where} \quad A_i = \int_a^b l_i(x)dx \quad (*)$$

with these x_i's as nodes will be exact for all polynomials of degree at most $2n+1$. Furthermore, the nodes lie in the open interval (a,b).

To summarize: With arbitrary nodes, the equation (*) is exact for all polynomials of degree $\leq n$. With the Gaussian nodes, the equation (*) is exact for all polynomials of degree $\leq 2n+1$.

The quadrature formulas that arise as applications of this theorem are called **Gaussian** or **Gauss-Legendre quadrature formulas**.

Example

As an example, we derive a Gaussian quadrature formula that is not too complicated. We determine the formula with **three Gaussian** nodes and **three weights** for the integral $\int_{-1}^{1} f(x)dx$.

We must find the polynomial q and compute its roots. The degree of q is 3, so it has the form

$$q(x) = c_0 + c_1x + c_2x^2 + c_3x^3$$

The conditions that q must satisfy are

$$\int_{-1}^{1} q(x)dx = \int_{-1}^{1} xq(x)dx = \int_{-1}^{1} x^2q(x)dx = 0$$

If we let $c_0 = c_2 = 0$, then $q(x) = c_1x + c_3x^3$. The integral of an odd function over a symmetric interval is 0 and so we have

$$\int_{-1}^{1} q(x)dx = \int_{-1}^{1} x^2q(x)dx = 0$$

To obtain c_1 and c_3, we impose the condition

$$\int_{-1}^{1} x(c_1x + c_3x^3)dx = 0$$

A convenient solution of this is $c_1 = -3$ and $c_3 = 5$. Hence

$$q(x) = 5x^3 - 3x$$

The roots are 0 and $\pm\sqrt{3/5}$. These are the desired Gaussian nodes for the quadrature formula.

To obtain the weights A_0, A_1 and A_2, we use a procedure known as the **method of undetermined coefficients**. Consider the formula

$$\int_{-1}^{1} f(x)dx \approx A_0 f\left(-\sqrt{\frac{3}{5}}\right) + A_1 f(0) + A_2 f\left(\sqrt{\frac{3}{5}}\right)$$

We want to select the coefficients A_i in such a way that the approximate equality ($\approx$) is an exact equality (=) whenever f is of the form $ax^2 + bx + c$.

Since integration is a linear process, the above formula will be exact for all polynomials of degree ≤ 2 if the formula is exact for 1, x and x^2. In tabular form

f	left-hand side	right-hand side
1	$\int_{-1}^{1} dx = 2$	$A_0 + A_1 + A_2$
x	$\int_{-1}^{1} x dx = 0$	$-\sqrt{\frac{3}{5}}A_0 + \sqrt{\frac{3}{5}}A_2$
x^2	$\int_{-1}^{1} x^2 dx = \frac{2}{3}$	$\frac{3}{5}A_0 + \frac{3}{5}A_2$

Thus we get

$$\begin{cases} A_0 + A_1 & + A_2 = 2 \\ A_0 & - A_2 = 0 \\ A_0 & + A_2 = 10/9 \end{cases}$$

The weights are $A_0 = A_2 = 5/9$ and $A_1 = 8/9$.

Therefore, the final formula is

$$\int_{-1}^{1} f(x)dx \approx \frac{5}{9} f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\sqrt{\frac{3}{5}}\right)$$

This will integrate all polynomial of degree 4 or less correctly.
For example, $\int_{-1}^{1} x^4 dx$ = 2/5. The formula yields the same value.

Example 2 - Application to a nonpolynomial

With the transformation $t = [2x - (b+a)]/(b-a)$, a Gaussian quadrature rule of the form

$$\int_{-1}^{1} f(t)dt \approx \sum_{i=0}^{n} A_i f(t_i)$$

can be used over an arbitrary interval $[a, b]$. The substitution gives

$$\int_{a}^{b} f(x)dx = \frac{1}{2}(b-a) \int_{-1}^{1} f\left[\frac{1}{2}(b-a)t + \frac{1}{2}(b+a)\right] dt$$

We can now use the obtained formula

$$\int_{-1}^{1} f(x)dx \approx \frac{5}{9} f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\sqrt{\frac{3}{5}}\right)$$

and the transformation to approximate the integral

$$\int_{0}^{1} e^{-x^2} dx$$

We have

$$\int_0^1 f(x)dx = \frac{1}{2}\int_{-1}^1 f\left(\frac{1}{2}t+\frac{1}{2}\right)dt$$
$$\approx \frac{1}{2}\left[\frac{5}{9}f\left(\frac{1}{2}-\frac{1}{2}\sqrt{\frac{3}{5}}\right)+\frac{8}{9}f\left(\frac{1}{2}\right)+\frac{5}{9}f\left(\frac{1}{2}+\frac{1}{2}\sqrt{\frac{3}{5}}\right)\right]$$

Letting $f(x) = e^{-x^2}$, we get

$$\int_0^1 e^{-x^2}dx \approx 0.746814584$$

The true solution is $\frac{1}{2}\text{erf}(1) \approx 0.7468241330$. Thus the error in the approximation given by the quadrature formula is of the order 10^{-5}. This is excellent, considering that only three function evaluations were made!

5.3 Tables of nodes and weights

The numerical values of nodes and weights for several values of n have been tabulated and can be found in references.

Table 5.1 in Cheney and Kincaid gives the values for $n \leq 4$. They can be applied for the Gaussian quadrature formula

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

As an example, the case for $n = 2$ was discussed in the previous example; i.e. for $x_i = -\sqrt{3/5}, 0, \sqrt{3/5}$ and $A_i = 5/9, 8/9, 5/9$.

5.4 Multidimensional integration

In general, multidimensional integration is more difficult and computationally much more time consuming than evaluating one-dimensional integrals. The subject is discussed only briefly here. We return to the subject later on in connection with Monte Carlo methods.

If the area of integration is relatively simple and the integrand is smooth, we can use a sequence of one-dimensional integrals:

$$I = \iiint f(x,y,z)dxdydz = \int_{x_1}^{x_2} dx \int_{y_1(x)}^{y_2(x)} dy \int_{z_1(x,y)}^{z_2(x,y)} dz\, f(x,y,z)$$

Denote the innermost integral by $G(x,y)$:

$$G(x,y) = \int_{z_1(x,y)}^{z_2(x,y)} f(x,y,z)dz$$

and the integral of this function by $H(x)$:

$$H(x) = \int_{y_1(x)}^{y_2(x)} G(x,y)dy$$

The final result is given by

$$I = \int_{x_1}^{x_2} H(x)dx$$

For simplicity, we illustrate with the trapezoid rule for the interval [0,1], using $n+1$ equally spaced points. The step size is therefore $h = 1/n$. The composite trapezoid rule is

$$\int_0^1 f(x)dx \approx \frac{1}{2h}[f(0) + 2\sum_{i=1}^{n-1} f(\frac{i}{n}) + f(1)]$$

We write this in the form

$$\int_0^1 f(x)dx \approx \sum_{i=0}^{n} A_i f(\frac{i}{n})$$

The error is $O(h^2) = O(n^{-2})$ for functions having a continuous second derivative.

If we now want to evaluate a **two-dimensional integral over the unit square**, we can apply the trapezoid rule twice:

$$\begin{aligned}
\int_0^1 \int_0^1 f(x,y)dxdy &\approx \int_0^1 \sum_{i=0}^{n} A_i f(\frac{i}{n},y)dy \\
&= \sum_{i=0}^{n} A_i \int_0^1 f(\frac{i}{n},y)dy \\
&\approx \sum_{i=0}^{n} A_i \sum_{j=0}^{n} A_j f(\frac{i}{n},\frac{j}{n}) \\
&= \sum_{i=0}^{n}\sum_{j=0}^{n} A_i A_j f(\frac{i}{n},\frac{j}{n})
\end{aligned}$$

The error is again $O(h^2)$ because each of the two applications of the trapezoid rule entails this error.

In the same way, we can integrate a function of k variables. The trapezoid rule in the general case for integration over the multidimensional unit hypercube is given by

$$\int_{[0,1]^k} f(\mathbf{x})d\mathbf{x} \approx \sum_{\alpha_1=0}^{n} \dots \sum_{\alpha_k=0}^{n} A_{\alpha_1} \dots A_{\alpha_k} f(\frac{\alpha_1}{n}, \dots, \frac{\alpha_k}{n})$$

Now consider the computational effort that is required for evaluating the integrals using the trapezoid rule. In the one-dimensional case, the work involved is $O(n)$. In the two-variable case, it is $O(n^2)$, and $O(n^k)$ for k variables.

Thus for a constant number of nodes (constant effort) the quality of the numerical approximation to the value of the integrals declines very quickly as the number of variables increases. This explains why the Monte Carlo methods for numerical integration become an attractive option for high-dimensional integration.

Example.

In order to evaluate the integral of $f(x,y) = xye^{-x^2y}$ over the square $x,y \in [0,1]$, we can modify the procedure *Trapezoid* to obtain a two-dimensional version:

```
double Trapezoid_2D(double (*f)(double x,double y),
                    double ax, double bx,
                    double ay, double by, int n)
{
  int i, j;
  double hx, hy, result;


  hx = (bx-ax)/(double)n;
  hy = (by-ay)/(double)n;

  result = 0.5*(f(ax,ay)+f(bx,by));

  for(i=1; i<n; i++){
    for(j=1; j<n; j++){
      result += f(ax+i*hx,ay+j*hy);
    }
  }
  result= hx*hy*result;

  return(result);
}
```

Using the procedure *Trapezoid_2D* with $n = 100$ (overall 10^4 iterations), we get the following estimate of the integral

$$I \approx 0.183649773143$$

The correct value is

$$I = \int_0^1 \int_0^1 xye^{-x^2y} dxdy = \frac{1}{2e} \approx 0.18393972$$

Thus the error is approximately 3×10^{-4}.

It is easy to see that obtaining more accurate estimates quickly becomes computationally intensive.