Numerical Integration

David Eberly
Magic Software, Inc.
http://www.magic-software.com

Created: March 2, 1999

1 Richardson Extrapolation

This method is very powerful. The key idea is to get high–order accuracy by using low–order formulas. It is used in Romberg integration (which my integration code is based on), but it is also the key idea in the adaptive Runge–Kutta differential equation solver.

Let Q be an unknown quantity which is approximated by $A(h)$ with approximation error of order $O(h^2)$. That is,

$$Q = A(h) + C_1 h^2 + O(h^4) = A(h) + O(h^2) \tag{1}$$

for some constant C_1. We can use this formula to produce a (possibly) more accurate approximation. Replacing h by $h/2$ in the formula yields

$$Q = A\left(\frac{h}{2}\right) + \frac{C_1}{2}h^2 + O(h^4). \tag{2}$$

Of course I used $O(h^2/4) = O(h^2)$. Taking four times equation (2) and substracting equation (2), then dividing by three yields

$$Q = \frac{4A\left(\frac{h}{2}\right) - A(h)}{3} + O(h^4). \tag{3}$$

The goal is for the $O(h^4)$ terms in equations (1) and (3) to be about the same size. If so, equation (3) is more accurate since it does not have the h^2 term in it.

Define $A_1(h) = A(h)$ and $A_2(h) = (4A_1(h/2) - A_1(h))/3$. Other approximations can be written in an `extrapolation table`

$$\begin{array}{cc}
A_1(h) & \\
A_1\left(\frac{h}{2}\right) & A_2(h) \\
A_1\left(\frac{h}{4}\right) & A_2\left(\frac{h}{2}\right) \\
A_1\left(\frac{h}{8}\right) & A_2\left(\frac{h}{4}\right) \\
\vdots & \vdots \\
A_1\left(\frac{h}{2^n}\right) & A_2\left(\frac{h}{2^{n-1}}\right)
\end{array}$$

The approximation $A_1(h/2^k)$ is order $O(h^2)$ and the approximation $A_2(h/2^k)$ is order $O(h^4)$.

If the original approximation is written as

$$Q = A(h) + C_1 h^2 + C_2 h^4 + O(h^6),$$

then the extrapolation table has an additional column

$$\begin{array}{lll}
A_1(h) & & \\
A_1\left(\frac{h}{2}\right) & A_2(h) & \\
A_1\left(\frac{h}{4}\right) & A_2\left(\frac{h}{2}\right) & A_3(h) \\
A_1\left(\frac{h}{8}\right) & A_2\left(\frac{h}{4}\right) & A_3\left(\frac{h}{2}\right) \\
\vdots & \vdots & \\
A_1\left(\frac{h}{2^n}\right) & A_2\left(\frac{h}{2^{n-1}}\right) & A_3\left(\frac{h}{2^{n-2}}\right)
\end{array}$$

where

$$A_3(h) = \frac{16A_2\left(\frac{h}{2}\right) - A_2(h)}{15}.$$

The approximation $A_3(h/2^k)$ is order $O(h^6)$.

In general the extrapolation table is an $n \times m$ lower triangular matrix $T = [T_{rc}]$ where

$$T_{rc} = A_c\left(\frac{h}{2^{r-1}}\right)$$

and

$$A_c(h) = \frac{4^{c-1}A_{c-1}\left(\frac{h}{2}\right) - A_{c-1}(h)}{4^{c-1} - 1}.$$

1.1 Trapezoid Rule

An approximation for $\int_a^b f(x)\,dx$ can be computed by first approximating $f(x)$ by the linear function

$$L(x) = \frac{x-b}{a-b}f(a) + \frac{x-a}{b-a}f(b)$$

and using $h[f(b)+f(a)]/2 = \int_a^b L(x)\,dx \doteq \int_a^b f(x)\,dx$. Some calculus shows that

$$\int_a^b f(x)\,dx = \frac{f(b)+f(a)}{2}h + O(h^3).$$

When $f(x) > 0$, the approximation is the area of a trapezoid with vertices at $(a, 0)$, $(a, f(a))$, $(b, 0)$, and $(b, f(b))$.

The integration interval $[a, b]$ can be divided into N subintervals over which the integration can be composited. Define $h = (b-a)/N$ and $x_j = a + jh$ for $0 \le j \le N$. It can be shown that

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

Note that the order of the approximation decrease by a power of one.

2 Romberg Integration

This method of integration uses the trapezoid rule to obtain preliminary approximations to the integral followed by Richardson extrapolation to obtain improvements.

Define $h_k = (b-a)/2^{k-1}$ for $k \geq 1$. The trapezoidal approximations corresponding to the interval partitions are

$$T_{k,1} = \frac{h_k}{2}\left[f(a) + 2\left(\sum_{j=1}^{2^{k-1}-1} f(a+jh_k)\right) + f(b)\right]$$

and so

$$\int_a^b f(x)\,dx = T_{k,1} + O(h_k^2)$$

for all $k \geq 1$. The following recursion formula can be shown to hold:

$$2T_{k,1} = T_{k-1,1} + h_{k-1}\sum_{j=1}^{2^{k-2}} f(a+(j-0.5)h_{k-1}) \tag{4}$$

for $k \geq 2$.

Richardson extrapolation can be applied; that is, generate the table

$$T_{i,j} = \frac{4^{j-1}T_{i,j-1} - T_{i-1,j-1}}{4^{j-1}-1}$$

for $2 \leq j \leq i$. It can be shown that

$$\lim_{k\to\infty} T_{k,1} = \int_a^b f(x)\,dx \text{ if and only if } \lim_{k\to\infty} T_{k,k} = \int_a^b f(x)\,dx.$$

The second limit typically converges much faster than the first. The idea now is to choose a value n and use $T_{n,n}$ as an approximation to the integral.

3 Implementation

The code is shown below.

```
class mgcIntegrate
{
public:
    typedef float (*Function)(float);

    mgcIntegrate () {;}

    float RombergIntegral (float a, float b, Function F);
};
```

```
float mgcIntegrate::
RombergIntegral (float a, float b, Function F)
{
    const int order = 5;

    float rom[2][order];
    float h = b-a;

    // initialize T_{1,1} entry
    rom[0][0] = h*(F(a)+F(b))/2;

    for (int i = 2, ipower = 1; i <= order; i++, ipower *= 2, h /= 2) {
        // calculate summation in recursion formula for T_{k,1}
        float sum = 0;
        for (int j = 1; j <= ipower; j++)
            sum += F(a+h*(j-0.5));

        // trapezoidal approximations
        rom[1][0] = (rom[0][0]+h*sum)/2;

        // Richardson extrapolation
        for (int k = 1, kpower = 4; k < i; k++, kpower *= 4)
            rom[1][k] = (kpower*rom[1][k-1] - rom[0][k-1])/(kpower-1);

        // save extrapolated values for next pass
        for (j = 0; j < i; j++)
            rom[0][j] = rom[1][j];
    }

    return rom[0][order-1];
}
```

I have arbitrarily chosen $n = 5$ (as the variable `integral_order`). The values of $T_{i,j}$ are stored in `rom[2][order]`. Note that not all the values must be saved to build the next ones (so the first dimension of `rom` does not have to be `order`). This follows from the recursion given in equation (4).