Section 6 - Numerical Integration and Differentiation

Two general types of integrations are solved numerically, ones where the function is presented as tabulated data or as a complicated function.

6.1 Newton-Cotes Formulas

The Newton-Cotes formulas are based on a strategy of replacing the complicated function or tabulated data with an easy-to-integrate approximating function

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

where

$$f_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

Closed and open forms of the Newton-Cotes are available. Closed form are those in which the data points at the beginning and end of the limits of integration are known while open forms are employed when the integration limits extend beyond the range of data, similar to extrapolation.

6.1.1 The Trapezoidal Rule

The Trapezoidal Rule is the first of the Newton-Cotes formulas, connecting each adjacent pair of data points by a straight line:

$$I = \int_a^b f(x)dx \cong \int_a^b \left[f(a) + \frac{f(b) - f(a)}{b - a}(x - a) \right] dx$$

$$I \cong (b - a)\frac{f(a) + f(b)}{2}$$

which has error $E_t = -\frac{1}{12} f''(\xi)(b - a)^3$ where ξ is between a and b. This formula is exact for linear functions (the error vanishes if $f'' = 0$, therefore first order functions are computed exactly), but introduces significant error for higher order functions $f(x)$.

Multiple-Application Trapezoidal Rule

One straightforward way to improve accuracy is to divide the integration interval $[a,b]$ into a number of segments and apply the rule to each. Let $h = (b-a)/n$ where n = number of equal-width segments between n+1 data points. Therefore,

$$I \cong h\frac{f(x_0)+f(x_1)}{2} + h\frac{f(x_1)+f(x_2)}{2} + \ldots + h\frac{f(x_{n-1})+f(x_n)}{2}$$

$$I \cong \frac{(b-a)}{n}\frac{f(x_0)+2\sum_{i=1}^{n-1} f(x_i)+f(x_n)}{2} = h\frac{f(x_0)+2\sum_{i=1}^{n-1} f(x_i)+f(x_n)}{2}$$

Which has error $E_t = -\frac{(b-a)^3}{12n^3}\sum^{n} f''(\xi) = -\frac{(b-a)^3}{12n^2}\bar{f}''$. As the number of intervals n doubles, the error E_t decreases by a factor of 4.

6.1.2 Simpson's Rules

One can increase the accuracy of the approximation by applying the multiple-application Trapezoidal Rule many times, slicing the data finer and finer. This can result, however, in the need for more tabulated data or more function evaluations and lead to more round-off error due to more calculations. As an alternative, Simpson used higher-order functions to approximate the integral.

Simpson's 1/3 Rule
Approximate the function between x_0 and x_2, with intermediate point x_1, with a second-order Lagrange polynomial

$$I \cong \int_{x_0}^{x_2}\left[\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_1)(x_2-x_0)}f(x_2)\right]dx$$

$$I \cong \frac{h}{3}\left[f(x_0)+4f(x_1)+f(x_2)\right] \quad \text{where } h=(b-a)/2$$

$$I \cong (b-a)\frac{f(x_0)+4f(x_1)+f(x_2)}{6}$$

with error $E_t = -\frac{1}{90}f^{(4)}(\xi)h^5$. Note that this rule is accurate to 3rd-order even though the function was approximated with a 2nd order polynomial. This rule can be applied to an even number of multiple segments

$$I \cong \frac{h}{3}\left[f(x_0)+4\sum_{i=1,3,5\ldots}^{n-1} f(x_i)+2\sum_{i=2,4,6\ldots}^{n-1} f(x_i)+f(x_n)\right]$$

where $h=\frac{(b-a)}{n}$ and $E_t = -\frac{(b-a)^5}{180n^4}\bar{f}^{(4)}$

Simpson's 3/8 Rule

If one fits a 3rd order Lagrange polynomial to 4 data points (3 intervals) and integrates, Simpson's 3/8 Rule results:

$$I \cong \frac{3h}{8}\left[f(x_0)+3f(x_1)+3f(x_2)+f(x_3)\right]$$

With $h=\frac{(b-a)}{3}$ and error $E_t=-\frac{(b-a)^5}{6480}f^{(4)}(\xi)$, so as n doubles the error decreases by a factor of 16. This error is similar to that of Simpson's 1/3 Rule. Simpson's 1/3 Rule is preferred, but the 3/8 Rule is useful when the number of intervals is odd. In that case, apply the 3/8 Rule to the three intervals at the end (or the beginning) of the tabulated data and an even number of intervals remain to be evaluated with the 1/3 Rule.

6.1.3 Higher-Order Newton-Cotes Formulas

There are many higher-order (and seldom used) Newton-Cotes formulas, for example Boole's Rule:

$$I \cong \frac{(b-a)}{90}\left[7(x_0)+32f(x_1)+12f(x_2)+32f(x_3)+7f(x_4)\right] \;,E_t=-\frac{8}{945}h^7 f^{(6)}(\xi)$$

6.1.4 Integration with Unequal Segments

To integrate across tabulated data that is not evenly spaced, there are couple of strategies to try, for example:

1. Use the Trapezoidal Rule throughout, accepting the errors
2. If 3 adjacent segments have the same width, use Simpson's 3/8 Rule
   If 2 adjacent segments have the same width, use Simpson's 1/3 Rule
   If the next segment is not the same width as the one following, use the Trapezodial Rule

6.1.5 Open Integration Formulas

Open integration formulas are most often used to evaluate improper integrals, to be discussed later, and in multistep methods of solving ordinary differential equations. Here we will simply list them for future reference:

2 segments n, 1 point (the *midpoint method*), truncation error $E_t=(1/3)h^3 f^{(2)}(\xi)$

$$I \cong (b-a) f(x_1)$$

2 segments n, 2 points, truncation error $E_t = (3/4) h^3 f^{(2)}(\xi)$

$$I \cong (b-a) \frac{f(x_1) + f(x_2)}{2}$$

4 segments n, 3 points, truncation error $E_t = (14/45) h^5 f^{(4)}(\xi)$

$$I \cong (b-a) \frac{2f(x_1) - f(x_2) + 2f(x_3)}{3}$$

5 segments n, 4 points, truncation error $E_t = (95/144) h^5 f^{(4)}(\xi)$

$$I \cong (b-a) \frac{11f(x_1) + f(x_2) + f(x_3) + 11f(x_4)}{24}$$

6 segments n, 5 points, truncation error $E_t = (41/140) h^7 f^{(6)}(\xi)$

$$I \cong (b-a) \frac{11f(x_1) - 14f(x_2) + 26f(x_3) - 14f(x_4) + 11f(x_5)}{20}$$

6.1.6 Multiple Integrals

Multiple integrals are evaluated in exactly the same manner as single integrals. For example

$$I = \int_c^d \left[\int_a^b f(x,y) dx \right] dy = \int_a^b \left[\int_c^d f(x,y) dy \right] dx$$

Evaluate the inside integral at discrete values, then evaluate the next integral using the values obtained in the first step.

Pseudocode – Trapezoidal Rule, Single integral

```
SUB SingleTrap(ax, bx, xints, result)
    intx = 0
    FOR ix = 0, xints
        x = ax + (bx - ax) * ix / xints
        IF (ix = 0 OR ix = xints) THEN
            intx = intx + f(x)   ' trapezoidal rule
        ELSE
            intx = intx + 2 * f(x)
        END IF
    END FOR
    result = intx * (bx - ax) / (2 * xints)
END SUB
```

Pseudocode – Trapezoidal Rule, Double integral

```
SUB DoubleTrap(ax, bx, xints, ay, by, yints, result)
      inty = 0
      FOR iy = o, yints ‘ the integration in y
            y = ay + (by – ay) * iy / yints
            intx = 0
            FOR ix = 0, xints ‘ the integration in x
                  x = ax + (bx - ax) * ix / xints
                  IF (ix = 0 OR ix = xints) THEN
                        intx = intx + f(x, y)
                  ELSE
                        intx = intx + 2 * f(x, y)
                  END IF
            END FOR
            intx = intx * (bx - ax) / (2 * xints)
            IF (iy = 0 OR iy = yints) THEN
                  inty = inty + intx
            ELSE
                  inty = inty + 2 * intx
            END IF
      END FOR
      result = inty * (by – ay) / (2 * yints)
END SUB
```

Pseudocode – Trapezoidal Rule, Triple integral

```
SUB TripleTrap(ax, bx, xints, ay, by, yints, az, bz, zints, result)
      intz = 0
      FOR iz = 0, zints ‘ the integration in z
            z = az + (bz – az) * iz / zints
            inty = 0
            FOR iy = 0, yints ‘ the integration in y
                  y = ay + (by – ay) * iy / yints
                  intx = 0
                  FOR ix = 0, xints ‘ the integration in x
                        x = ax + (bx - ax) * ix / xints
                        IF (ix = 0 OR ix = xints) THEN
                              intx = intx + f(x, y, z)
                        ELSE
                              intx = intx + 2 * f(x, y, z)
                        END IF
                  END FOR
                  intx = intx * (bx - ax) / (2 * xints)
                  IF (iy = 0 OR iy = yints) THEN
                        inty = inty + intx
                  ELSE
                        inty = inty + 2 * intx
                  END IF
            END FOR
            inty = inty * (by – ay) / (2 * yints)
```

```
            IF (iz = 0 OR iz = zints) THEN
                  intz = intz + inty
            ELSE
                  intz = intz + 2 * inty
            END IF
      END FOR
      result = intz * (bz - az) / (2 * zints)
END SUB
```

6.2 Integration of Equations

If we are faced with the task of numerically integrating a function instead of a table of values, we can exploit the fact that we can generate as many function evaluations as we need to obtain acceptable accuracy.

6.2.1 Newton-Cotes Algorithms for Equations

Pseudocodes are given that take advantage of function calls instead of tabulated values. The basic limitation of these formulas is that to improve accuracy more and more segments must be added, eventually leading to unacceptable round-off errors. In those cases, either use higher-order formulas or a better strategy.

6.2.2 Romberg Integration

Richardson Extrapolation

Recall that, for a multiple application trapezoidal rule,

$$I = I(h) + E(h)$$

Where I is the exact value of the integral, $I(h)$ is the approximation of the integral as a function of segment width $h = (b-a)/n$ and $E(h)$ is the error associated with the approximation. If two separate estimates are done at segment widths h_1 and h_2, then

$$I = I(h_1) + E(h_1) = I(h_2) + E(h_2)$$

Recall that the error in a multiple application trapezoidal rule is $E \cong \dfrac{b-a}{12} h^2 \bar{f}''$. If we assume that the average value of the second derivative is the same for both estimates, then

$$E(h_1)/E(h_2) \cong h_1^2 / h_2^2 \quad \text{or} \quad E(h_1) \cong E(h_2)(h_1/h_2)^2$$

so that

$$I(h_1) + E(h_2)(h_1 / h_2)^2 \cong I(h_2) + E(h_2)$$

or

$$E(h_2) \cong \frac{I(h_1) - I(h_2)}{1 - (h_1 / h_2)^2}$$

And

$$I \cong I(h_2) + \frac{I(h_2) - I(h_1)}{(h_1 / h_2)^2 - 1}$$

The error associated with this new estimate is $O(h^4)$. Therefore, we have combined two trapezoidal rule estimates of $O(h^2)$ to yield a new estimate of $O(h^4)$. If the interval is halved (that is, $h_2 = h_1/2$) then this becomes

$$I \cong I(h_2) + \frac{1}{2^2 - 1}\left[I(h_2) - I(h_1)\right]$$
$$I \cong \frac{4}{3} I(h_2) - \frac{1}{3} I(h_1)$$

In a similar manner, two estimates of $O(h^4)$ can be combined to yield an estimate of $O(h^6)$:

$$I \cong \frac{16}{15} I_m - \frac{1}{15} I_l$$

where m = more accurate and l = less accurate, and two estimates at $O(h^6)$ can be combined to yield an estimate of $O(h^8)$:

$$I \cong \frac{64}{63} I_m - \frac{1}{63} I_l$$

Romberg Integration Algorithm

The results above can be generalized to

$$I_{j,k} = \frac{4^{k-1} I_{j+1,k-1} - I_{j,k-1}}{4^{k-1} - 1}$$

where $I_{j+1,k-1}$ and $I_{j,k-1}$ are the more/less accurate integrals, respectively, and $I_{j,k}$ is the improved integral. The index k is the level of integration (k=1 is the original trapezoidal rule, k=2= $O(h^4)$, k=3= $O(h^6)$, etc) and j distinguishes between more (j+1) and less (j) accurate estimates. It is possible to successfully apply this strategy to certain tabulated data of the proper form.

6.2.3 Gauss Quadrature

A characteristic of the Newton-Cotes formulas is that they rely on fixed-values of the function. A strategy that starts by determining the best place for these function evaluations is the basis for a class of techniques called *Gauss Quadrature*.

To see how these formulas are developed, consider the case of approximating the integral of a 3rd order function between -1 and 1. We want to determine the points x_0 and x_1 to evaluate the function, and the coefficients c_0 and c_1, so that the approximation

$$I \cong c_0 f(x_0) + c_1 f(x_1)$$

exactly represents a 3rd order curve. To find these 4 parameters we need 4 conditions:

$$\begin{aligned}
c_0 f(x_0) + c_1 f(x_1) &= \int_{-1}^{1} 1 dx = 2 \\
c_0 f(x_0) + c_1 f(x_1) &= \int_{-1}^{1} x dx = 0 \\
c_0 f(x_0) + c_1 f(x_1) &= \int_{-1}^{1} x^2 dx = 2/3 \\
c_0 f(x_0) + c_1 f(x_1) &= \int_{-1}^{1} x^3 dx = 0
\end{aligned}$$

Solving these equations yields the parameters x_0, x_1, c_0 and c_1.

In general, the functions are evaluated at the roots of the Legendre polynomials of the 1st kind. Legendre polynomials are an orthogonal set of polynomials which are solutions of the differential equation

$$(1-x^2)\frac{d^2 y}{dx^2} - 2x\frac{dy}{dx} + r(r+1)y = 0 \qquad -1 < x < 1$$

$$P_r(x) = \frac{1}{2^r r!}\frac{d^r}{dx^r}\left[(x^2-1)^r\right]$$

The first several Legendre polynomials of the 1st kind are

$$P_2(x) = (1/2)(3x^2 - 1)$$
$$P_3(x) = (1/2)(5x^3 - 3x)$$
$$P_4(x) = (1/8)(35x^4 - 30x^2 + 3)$$
$$P_5(x) = (1/8)(63x^5 - 70x^3 + 15x)$$
$$P_6(x) = (1/16)(231x^6 - 315x^4 + 105x^2 - 5)$$
$$P_7(x) = (1/16)(429x^7 - 693x^5 + 315x^3 - 35x)$$

The coefficients are

$$c_k = \frac{2(1-u_k^2)}{(r+1)^2 P_{r+1}^2(u_k)} \qquad \text{where } u_k \text{ are roots of } P_r$$

Points	**Weighting Factors**	**Function Arguments**	**Truncation Error**
2	c_0 = 1.000000000000000 c_1 = 1.000000000000000	x_0 = -0.57735 02691 89626 x_1 = 0.57735 02691 89626	$\cong f^{(4)}(\xi)$
3	c_0 = 0.55555 55555 55556 c_1 = 0.88888 88888 88889 c_2 = 0.55555 55555 55556	x_0 = -0.77459 66692 41483 x_1 = 0.0 x_2 = 0.77459 66692 41483	$\cong f^{(6)}(\xi)$
4	c_0 = 0.34785 48451 37454 c_1 = 0.65214 51548 62546 c_2 = 0.65214 51548 62546 c_3 = 0.34785 48451 37454	x_0 = -0.86113 63115 94053 x_1 = -0.33998 10435 84856 x_2 = 0.33998 10435 84856 x_3 = 0.86113 63115 94053	$\cong f^{(8)}(\xi)$
5	c_0 = 0.23692 68850 56189 c_1 = 0.47862 86704 99366 c_2 = 0.56888 88888 88889 c_3 = 0.47862 86704 99366 c_4 = 0.23692 68850 56189	x_0 = -0.90617 98459 38664 x_1 = -0.53846 93101 05683 x_2 = 0.0 x_3 = 0.53846 93101 05683 x_4 = 0.90617 98459 38664	$\cong f^{(10)}(\xi)$
6	c_0 = 0.17132 44923 79170 c_1 = 0.36076 15730 48139 c_2 = 0.46791 39345 72691 c_3 = 0.46791 39345 72691 c_4 = 0.36076 15730 48139 c_5 = 0.17132 44923 79170	x_0 = -0.93246 95142 03152 x_1 = -0.66120 93864 66265 x_2 = -0.23861 91860 83197 x_3 = 0.23861 91860 83197 x_4 = 0.66120 93864 66265 x_5 = 0.93246 95142 03152	$\cong f^{(12)}(\xi)$

Integrating $\int_a^b f(x)dx$ involves a change of variable

$$x = \frac{1}{2}(b+a) + \frac{1}{2}(b-a)u \ , \quad dx = \frac{1}{2}(b-a)du$$

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

$$= \frac{1}{2}(b-a)\left[c_0 f(x_0) + c_1 f(x_1) + \ldots + c_n f(x_n)\right] + R$$

Gauss Quadrature provides a highly efficient way to obtain very accurate results with minimum function evaluations. This method can also be used in a multiple-application approach.

6.2.4 Improper Integrals

If faced with evaluating an improper integral whose lower limit is -∞ or whose upper limit is ∞, the following identity allows for a convenient transformation of variable and works whenever the function tends to zero at least as fast as $1/x^2$:

$$\int_a^b f(x)dx = \int_{1/a}^{1/b} \frac{1}{t^2} f\left(\frac{1}{t}\right)dt \quad \text{for } ab > 0$$

Note the restriction. This can be used if $a = \infty$ and b is positive, or $a = -\infty$ and b is negative. For the case when $a = -\infty$ and $b = \infty$, implement the integral in two steps:

$$\int_{-\infty}^{b} f(x)dx = \int_{-\infty}^{-A} f(x)dx + \int_{-A}^{b} f(x)dx$$

where A is sufficiently large so the function is approaching zero at least as fast as $1/x^2$.

To avoid evaluating a function at one of the limits, use one of the open forms of the Newton-Cotes formulas given in the Section 6.1.5.

6.3 Numerical Differentiation

Numerical differentiation was introduced when we began our exploration of numerical methods as an example of the Taylor series. There are three general types of divided difference differentiation approximations: forward, backward and central:

Forward:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} \quad ; \quad O(h)$$

$$f'(x_i) = \frac{-f(x_{i+2}) + 4f(x_{i+1}) - 3f(x_i)}{2h} \quad ; \quad O(h^2)$$

Backward:

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} \quad ; \quad O(h)$$

$$f'(x_i) = \frac{3f(x_i) - 4f(x_{i-1}) + f(x_{i-2})}{2h} \quad ; \quad O(h^2)$$

Central:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} \quad ; \quad O(h^2)$$

$$f'(x_i) = \frac{-f(x_{i+2}) + 8f(x_{i+1}) - 8f(x_{i-1}) + f(x_{i-2})}{12h} \quad ; \quad O(h^4)$$

Corresponding equations for the second derivatives are

Forward:

$$f''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2} \quad ; \quad O(h)$$

$$f''(x_i) = \frac{-f(x_{i+3}) + 4f(x_{i+2}) - 5f(x_{i+1}) + 2f(x_i)}{h^2} \quad ; \quad O(h^2)$$

Backward:

$$f''(x_i) = \frac{f(x_i) - 2f(x_{i-1}) + f(x_{i-2})}{h^2} \quad ; \quad O(h)$$

$$f''(x_i) = \frac{2f(x_i) - 5f(x_{i-1}) + 4f(x_{i-2}) - f(x_{i-3})}{h^2} \quad ; \quad O(h^2)$$

Central:

$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} \quad ; \quad O(h^2)$$

$$f''(x_i) = \frac{-f(x_{i+2}) + 16f(x_{i+1}) - 30f(x_i) + 16f(x_{i-1}) - f(x_{i-2})}{12h^2} \quad ; \quad O(h^4)$$

6.3.1 Richardson Extrapolation

Similar to what we saw for integration, we can use two estimates to compute a third, more accurate, approximation. If the two estimates are computed with step sizes h_1 and h_2, where $h_2 = h_1/2$, then

$$D \cong \frac{4}{3}D(h_2) - \frac{1}{3}D(h_1)$$

As for integration, this can be applied iteratively using the Romberg algorithm until the result achieves an acceptable error criterion.

6.3.2 Derivatives of Unequally-Spaced Data

Experimental or field data that is collected at equal intervals can be differentiated using interpolation. For example, fit a second-order Lagrange interpolating polynomial to each set of three adjacent points, then differentiate the result analytically to find

$$f'(x) = \frac{f(x_{i-1})(2x - x_i - x_{i+1})}{(x_{i-1} - x_i)(x_{i-1} - x_{x+1})} + \frac{f(x_i)(2x - x_{i-1} - x_{i+1})}{(x_i - x_{i-1})(x_i - x_{x+1})} + \frac{f(x_{i+1})(2x - x_{i-1} - x_i)}{(x_{i+1} - x_{i-1})(x_i - x_{x+1})}$$

which has the same error as a central divided difference.

6.3.3 Derivatives and Integrals for Data with Errors

Due to its subtractive nature, numerical differentiation magnifies errors that are present in raw data. One way to treat noisy data is to first perform a low-order polynomial regression on the data, then perform the subsequent analysis on the fitted curves.

Always remember that numerical differentiation tends to be unstable, amplifying errors while numerical integration is forgiving and tends to smooth out errors in the data. Since integration is additive, random positive and negative small errors tend to cancel out, while differentiation tends to make such errors worse.