NUMERICAL INTEGRATION AND DIFFERENTIAL EQUATIONS

Key words: Interpolating functions, numerical integration, trapezoidal rule, Simpson's rule, Euler's method, RungeKutta methods.

6.1 INTRODUCTION

So far, we have considered interpolation, curve fitting and matrix operations. Now we consider the problem of numerical integration and differential equations. Taking derivatives from a set of data $(x_1, y_1), (x_2, y_2) \ldots \ldots (x_n, y_n)$ is a bit harder than integration, unless we first have an interpolating polynomial or a fitting function for the data. While the same problems persists for taking an integral over these points, the error in the integral (area under the curve) is smaller than the values of slopes of the curve in different ranges of the data (use the definition of the derivative and the integral to rationalise this feature).We will not deal with numerical derivatives, but suggest that you use a suitable interpolating polynomial or a "best fit" and take the derivatives of these functions at the point of interest.

We will consider two elementary methods of numerical integration and two methods for solving ordinary differential equations. We begin with numerical integration.

Key words: Interpolating functions, numerical integration, trapezoidal rule, Simpson's rule.

6.2 NUMERICAL INTEGRATION

Given a set of data points $\big(x_0, f(x_0)\big), \big(x_1, f(x_1)\big), \ldots . (x_n, f(x_n))$

We want an estimate of the integral I,

$$I = \int_{x_0}^{x_n} f(x)dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx (6.1)$$

For convenience, consider the situation wherein the interval between adjacent points x_i is a constant, h. Since we do not know the value of x at intermediate points other than the data points, we can approximate the function by interpolating polynomials of different degrees and get approximations to I. for a linear interpolating polynomial (a straight line between x_i and x_{i+1})

$$P_1^i(x) = f(x_i) + \frac{\Delta f(x_i)}{h}(x - x_i) \ \forall \ x_i \leq x \leq \ x_{i+1}) \text{Where } \Delta f(x_i) = \frac{[f(x_{i+1}) - f(x_i)]}{} (6.2)$$

$$\mathrm{I}_i = \int_{x_i}^{x_{i+1}} P_1^i(x)dx = \frac{\Delta f(x_i)}{h} \int_{x_i}^{x_{i+1}} (x - x_i)dx + f(x_i)h (6.3)$$

$$= f(x_i)h + \frac{\Delta f(x_i)}{h} . \frac{h^2}{2} = \frac{h}{2}(f(x_{i+1}) + f(x_i)) (6.4)$$

This is the area of the trapezoid (trapezium)shown in fig 1. Hence this is called the trapezoidal rule.

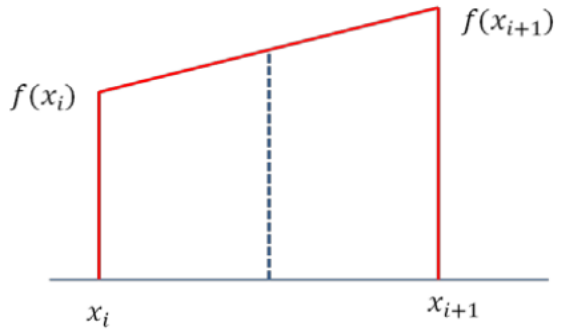

Fig 6.1 An illustration of the trapezoidal rule between two adjacent points.

If there are (n+1) data points, there will be n intervening trapezoids between $(x_0, x_1)(x_1, x_2) \ldots \ldots (x_{n-1}, x_n)$ and the area of all these trapezoids is

$$\mathrm{I} = h \sum_{i=0}^{n-1} \frac{f(x_{i+1}) + f(x_i)}{2} = \frac{hf(x_0)}{2} + h \sum_{i=1}^{n-1} f(x_i) + \frac{hf(x_n)}{2} (6.5)$$

The last two points have the coefficients h/2 and all the intermediate points having the coefficient h.

The next improvement over the trapezoidal rule is the Simpson's rule, wherein we interpolate between x_i and x_{i+1} using a quadratic polynomial. The integration for the i^{th} segment is

$$\mathrm{I} = \int_{x_{2i}}^{x_{2i+2}} P_2^i(x)dx \quad (6.6)$$

Here, we have divided the intervals into $(x_0, x_2), (x_2, x_4) \ldots (x_{n-2}, x_n)$ or (x_{2i}, x_{2i+2}) with $i = 0,1,2 \ldots (\frac{n}{2} - 1)$. As in trapezoidal rule, we need (n+1) values of x_i and $f(x_i)$, i.e. the values of n= even and the total number of data points n+1 is odd.

Using the second order Newton's forward interpolating polynomial between x_{2i} and x_{2i+2}, we have

$$\mathrm{I}_i = \int_{x_{2i}}^{x_{2i+2}} [f(x_{2i}) + \frac{\Delta^1 f(x_{2i})}{h}(x - x_{2i}) + \frac{\Delta^2 f(x_i)}{2h^2}(x - x_{2i})(x - x_{2i+1})] \quad (6.7)$$

Where $\Delta^1 f(x_i) = f(x_{i+1}) - f(x_i)$

And $\Delta^2 f(x_i) = \Delta^1 f(x_{i+1}) - \Delta^1 f(x_i) = f(x_{i+2}) - 2f(x_{i+1}) - f(x_i)$

$$\mathrm{I}_i = h[2f(x_{2i}) + 2\Delta^1 f(x_{2i}) + \frac{1}{3}\Delta^2 f(x_{2i})] \quad (6.8)$$

$$= h/3[2f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2})] \quad (6.9)$$

For the full interval between x_0 *and* x_n, the integral is

$$\mathrm{I} = \frac{\mathrm{h}}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) \ldots + 4f(x_{n-1}) + f(x_n)\right] \quad (6.10)$$

6.3 PROGRAM FOR SIMPSON'S RULE

Programming for the formula of eq. (6.10) is very straightforward. We need an odd numbered data set of (x_i, y_i), $i = 1, n$ where n is odd.If n is even, the integration between the last two data points

can be done by trapezoidal rule. This is a straight forward program and it is given below for two situations.

(Since do loops normally start from I = 1,n, the notation has been switched from x_0, x_1…… to x_1, x_2….)

```
c    These programs are for well defined functions. Same logic will
c     apply for equally spaced data: x(i), y(i), i = 1,2,.....N (N odd)
c     If n is even, use trapezoidal rule for two points either at the end
c     or at the beginning, wherever the values of y(i) are small
c     SIMPSON'S METHOD OD INTEGRATION (ARRAY GIVEN)
         PROGRAM Simpson
cSimpson's method of integration
dimension x(1000),y(1000)
write(*,*)'To find the value of integral of a function using
1  Simpsons rule. '
write(*,*)'Enter the limits of integration:'
read(*,*)a,b
write(*,*)'Enter the number of subintervals(an even no.) :'
read(*,*)n
h=(b-a)/n
x(1)=a
y(1)=f(x(1))
DO 10 i=2,n+1
x(i)=x(i-1)+h
y(i)=f(x(i))
10       continue
s=y(1)+y(n+1)
        DO 20 i=2,n,2
         s=s+4*y(i)
20       continue
        DO 30 i=3,n-1,2
           s=s+2*y(i)
30         continue
        s=h*s/3
write(*,40)s
40      format(1x,'The value of the integral is: ',F10.4)
        STOP
        END
function f(x)
f=sin(x)
c   This can be generalised to any other function or simply
c   a set of data points x(i) and y(i), i = 1, n
c   After testing for a few functions, you can use the program
c   for any set of data
return
       END
```

6.4 DIFFERENTIAL EQUATIONS

We will consider only first order differential equations. Higher order differential equations and partial differential equations will not be considered although important applications of numerical methods in chemistry involve not only higher order equations, but also equations involving both derivatives and integrals!

A typical first order differential equation can be represented by

$$y'(x) = \frac{dy}{dx} = f(x, y) \tag{6.11}$$

Many of the schemes of integration use the Taylor series expansion of the function

$$y(x_{n+1}) = y(x_n) + hy^{(1)}(x_n) + \frac{h^2}{2!}y^{(2)}(x_n) \ldots\ldots + \frac{h^n}{n!}y^{(n)}(x) + T_{n+1} \tag{6.12}$$

Here T_{n+1} is the remainder after approximating the function after n terms. The value of the error T_{n+1} is

$$T_{n+1} = \frac{h^{n+1}}{(n+1)!}y^{(n+1)}(\bar{x}_n), \qquad x_n < \bar{x}_n < x_{n+1} \tag{6.13}$$

Here, $y^{(k)}$ refers to the k^{th} derivative of y.

In a one-step method of integration, the value of y at x_{n+1} is determined from the value of y at x_n and its derivative at x_n

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} y'(x)dx \tag{6.14}$$

If we use only the first derivative in eq. (6.12), then

$$y(x_{n+1}) = y(x_n) + hy^{(1)}(x_n) \tag{6.15}$$

This is the Euler's method of one step category and is the easiest method. This can be extended to a series of points. The errors accumulate to orders of h^2 and thus h has to be very small for using this algorithm.

Examples for this method and other methods are more easily constructed using other softwares such as scilab and will be considered in the next chapter.

6.5 RUNGE KUTTA (R K) METHODS

A natural extension of the one step Euler's method which involved only $y^{(1)}$ is to use $y^{(2)}$ and higher derivatives of the function $y' = f(x, y)$. Instead of higher derivatives of $f(x, y)$, it is more convenient to use additional points of $f(x, y)$ in the vicinity of (x_n, y_n) and these are referred to as RUNGE KUTTA methods. These are equivalent to nth order Taylor's method.

In the second order RUNGE KUTTA method, y_{n+1} is obtained in terms of y_n and two undermined coefficients k_1, k_2, α and β as follows

$$y_{n+1} = y_n + h[k_1 f(x_n, y_n) + k_2 f(x_n + \alpha h, y_n + \beta h) f(x_n, y_n)] \quad (6.16)$$

To determine the coefficients, equate eq. (6.16) to the Tailor's second order expression which is given by

$$y_{n+1} = y_n + h[f(x_n, y_n) + \frac{h}{2} f'(x_n, y_n)] \quad (6.17)$$

Equating eqns. (6.16) and (6.17), we have

$$k_1 f(x_n, y_n) + k_2 f(x_n + \alpha h, y_n + \beta h) f(x_n, y_n)] = f(x_n, y_n) + \frac{h}{2} f'(x_n, y_n) \quad (6.18)$$

Expand the second term on the left hand side by Taylor expansion

$$f(x_n + \alpha h, y_n + \beta h) f(x_n, y_n) = f(x_n, y_n) + \alpha h f_x(x_n, y_n) + \beta h f_x(x_n, y_n) f_y(x_n, y_n) \quad (6.19)$$

Since $y' = f(x, y)$, $y'' = f' = f_y\, y' = f_x + f f_y$ as f depends on both x and y.

Substitute eq. (6.19) into eq. (6.18), we get

$$k_1 f(x_n, y_n) + k_2 f(x_n, y_n) + \alpha h f x_n y_n) + \beta \text{hf}(x_n, y_n) f_y(x_n, y_n)$$

$$= f(x_n, y_n) + \frac{h}{2}[f_x(x_n, y_n) + f(x_n, y_n) f_y(x_n, y_n) \quad (6.20)$$

Where, we have again used $f^{\text{'}} = f_x + f f_y$ on the right hand side.

Equating the coefficients in eq. (6.20), we get three equations ink_1, k_2, α and β. These are

$$k_1 \alpha = \frac{1}{2},\ k_2 \beta = \frac{1}{2} \text{ and } k_1 + k_2 = 1 \quad (6.21)$$

For each value ofk_2, we get an algorithm as follows

$$k_1 = 1 - k_2, \alpha = \frac{1}{2k_2} \beta = \frac{1}{2k_2} \quad (6.22)$$

If $k_2 = \frac{1}{2}$ we get the second order RK algorithm as

$$y_{n+1} = y_n + \frac{h}{2}[f_n + f(x_{n+1}, y_n + h f_n)] \quad (6.23)$$

Where $x_{n+1} = x_n + h$.

The fourth order RUNGE KUTTA method, which is very popular, is outlined below

$$y^{\text{'}} = f(x, y) \text{with} \quad (6.24)$$

$$y_{n+1} = y_n + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \quad (6.25)$$

Where

$$k_1 = h\, f(x_n, y_n)$$

$$k_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$$

$$k_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \tag{6.26}$$

$$k_4 = hf(x_n + h, y_n + k_3)$$

6.7 SUMMARY

In this chapter, we have studied numerical integration and the solution of first order differential equations. We have considered a set of discrete equidistant points(x_n, y_n). If the data is not equidistant, it is better and often more convenient to use interpolation methods to regenerate an equidistant collection of data$(\bar{x}_n, \bar{y}_n)$. A linear interpolation scheme between all adjacent points leads to the trapezoidal rule of numerical integration. If the spacing between adjacent points is *h*, the error in the scheme is of the order of h^2. Simpson's rule uses the interpolation formula to second order in *h* and thus is more accurate and commonly used.

We have considered elementary methods of solving first order differential equations. These methods are also derived using interpolating polynomials. The RUNGE KUTTA methods are quite popular. For higher order differential equations and partial different equations, you may either refer to the books on numerical analysis or other NPTEL courses that deal with these topics.

PROBLEMS

1. Write a program to integrate the given set of data (even) using trapezoidal and Simpson's rule.
2. Write a program to find sin(x) and cos(x) using tailor series.