OLD DOMINION UNIVERSITY

Numerical Integration I

A. Godunov and R. Briceno

1. Overview
2. Numerical integration: basic concepts
3. Methods based on Riemann sum
4. Newton-Cotes formulae

updated 15 February 2022

1

Part 1:
Overview

2

Motivation

Integrals are everywhere in physics

Some examples:

The kinetic energy– work theorem

$$\Delta K = K_f - K_i = \int_i^f \vec{F} \cdot d\vec{r}$$

Maxwell's equations, e.g., Gauss's law

$$\oint \vec{E} \cdot d\vec{A} = \frac{1}{\epsilon_0} q_{enq}$$

Fourier transforms

$$F(\omega) = \int_{\omega_i}^{\omega_f} e^{-2\pi\omega x} f(x) dx$$

3

3

Methods for evaluating integrals

Integrals can be evaluated using

- exact integration (analytically when possible)
- simple numerical methods
- advanced numerical methods

We use numerical integration when

- if we cannot get an analytic answer using (many known functions, do not have an exact integral)
- the function $f(x)$, which is to be integrated, may be a set of discrete data.

4

4

Exact integration

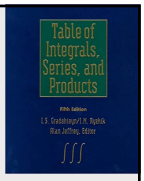
Standard techniques of integration:

- substitution rule, integration by parts, using identities, …
- Tables of integrals
- Computer algebra systems

5

5

Tables of integrals



- Table of Integrals, Series and Products by Gradshteyn I. S. and Ryzhik I. M. Academic Press, 1994 (many editions since 195x)
- Integral and Series, vol.1-4, by Prudnikov A P, Brychkov Yu A and Marichev A I Gordon and Breach, New York, 1986
- Tables of Integrals and Other Mathematical Data by Herbert B. Dwight (very simple integrals, many editions)
- and many more …

6

6

Computer algebra systems

- MatLab - available at ODU
- Mathematica - available at ODU
- WolframAlpha - https://www.wolframalpha.com/
  very easy to use
- Python - [Clunky]
- Maple
- Scientific Workplace
- Derive

7

7

Matlab examples

Here are some simple usage of the int function in MATLAB.

```
% symbolic integrals
syms x a

expr = x^2*cos(x);
F = int(expr)
sin(x)*(x^2 - 2) + 2*x*cos(x)

F = int(expr,x)
expr = exp(-a*x)
F = -exp(-a*x)/a

expr = exp(-x^2);
F = int(expr)
(pi^(1/2)*erf(x))/2

expr = exp(-x^2)
F = int(expr,0,inf)
F = pi^(1/2)/2
```

8

8

Matlab examples

Here are some simple usage of the Integrate function in Mathematica.

```
In[2]:= Integrate[x^n, x]
Out[2]= x^(1+n)/(1+n)

In[3]:= Integrate[1/x, x]
Out[3]= Log[x]

In[4]:= Integrate[1/x^2, x]
Out[4]= -1/x

In[7]:= Integrate[Exp[x * n], x]
Out[7]= e^(n x)/n
```

9

9

Part 2:

Numerical integration: basic concepts

10

Concept 1: Use the definition as a Riemann sum

The definition as a Riemann sum

$$I = \int_a^b f(x)dx = \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i^*)(x_i - x_{i-1}) \quad a = x_0 < x_1 < \cdots < x_n = b$$

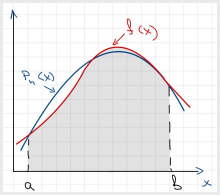
11

11

Concept 2: Integrating approximating functions

Numerical integration (quadrature) formulas can be developed by fitting approximating functions (e.g., polynomials) to discrete data and integrating the approximating function:

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



12

12

Integrating approximating functions (cont.)

CASE 1: The function to be integrated is known only at a finite set of discrete points.

Parameters under control – the degree of approximating polynomial

CASE 2: The function to be integrated is known.

Parameters under control:

- The total number of discrete points
- The degree of the approximating polynomial to represent the discrete data.
- The locations of the points at which the known function is discretized

13

13

Example: integrating direct fit polynomials

Imagine you have some discrete data in some region, and you have successfully found a polynomial to interpolate the data.

In other words, you determine the values of the coefficient, such that

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^n = \sum_{i=0}^{n} a_i x^i$$

Then, the integral in this region is "straightforward",

$$I = \int_a^b f(x)df \approx \int_a^b P_n(x)dx = \sum_{i=0}^{n} \int_a^b a_i x^i \, dx = \sum_{i=0}^{n} a_i \frac{x^{i+1}}{i+1}\Big|_a^b$$

14

14

Integrating direct fit polynomials

- This procedure only relies having determined the coefficients on the interpolating function. In other words it does not depend on whether the original data was *uniformly spaced or not.*
- Because our interpolating routines assume the data can be described by polynomials, one can imagine that this only useful for distributions whose underlying functions are non-singular.

15

15

Part 3:

Methods based on Riemann sum

equal subintervals

16

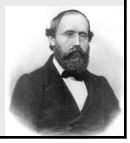
Riemann integral

If $f(x)$ is a continuous function defined for $a \le x \le b$ and we divide the interval into n subintervals of equal width $\Delta x = (b-a)/n$, then the definite integral is

$$I = \int_a^b f(x)dx = \lim_{n\to\infty} \sum_{i=0}^{n} f(x_i^*)\Delta x$$

Note, for a finite n, how well the Riemann sum approximates the integral depends on, among other things, how we write the discrete sums.

The Riemann integral can be interpreted as a net area under the curve $y = f(x)$ from a to b.



Bernhard Riemann [1826-1866]

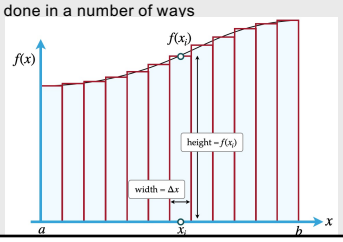
17

An area under the curve

Evaluating integrals exactly as an area under the curve is practically numerically impossible.

Solution: Replace this with the a "large number" of finite-sized and easy-to-evaluate areas (e.g. rectangles), and have your code sum over these areas. Take the limit when these areas become increasingly small.

It can be done in a number of ways



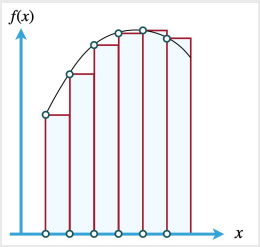
18

18

Primitive rule 1

The left endpoint Riemann sum

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} f(x_i)\Delta x, \qquad \Delta x = \frac{b-a}{n}$$



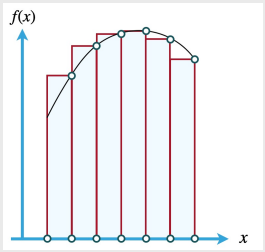
19

19

Primitive rule 2

The right endpoint Riemann sum

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(x_i)\Delta x, \qquad \Delta x = \frac{b-a}{n}$$



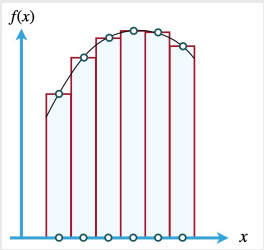
20

20

Primitive rule 3

The midpoint Riemann sum

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f\left(\frac{x_{i-1} + x\_i}{2}\right)\Delta x, \qquad \Delta x = \frac{b-a}{n}$$



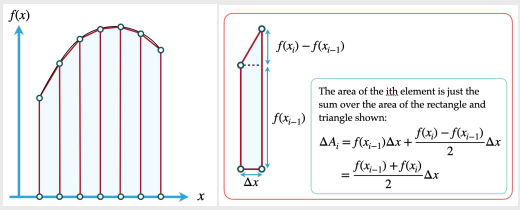
21

21

Primitive rule 4

The trapezoid rule

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} \left(\frac{f(x_i) + f(x_{i-1})}{2} \right) \Delta x, \qquad \Delta x = \frac{b-a}{n}$$



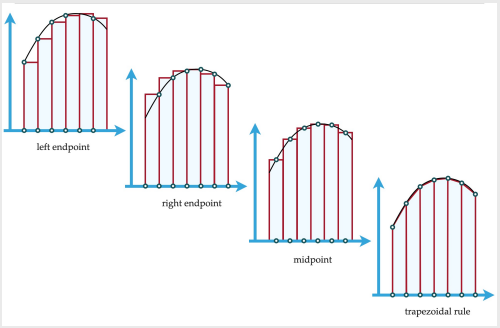
$$\int_a^b f(x)dx \approx \frac{1}{2}\Delta x(f_0 + 2f_1 + 2f_2 + \cdots + 2f_{n-1} + f_n)$$

22

22

Primitive rules for the Riemann sums

Which one should give a better approximation to the integral?



23

23

Example: C++ $\int_a^b f(x)dx \approx \frac{1}{2}\Delta x(f_0 + 2f_1 + 2f_2 + \cdots + 2f_{n-1} + f_n)$

```
// Integration by the trapezoidal rule of f(x) on [a,b]
// f - Function to integrate (supplied by a user)
//    a - Lower limit of integration
//    b - Upper limit of integration
//    r - Result of integration (out)
//    n - number of intervals
double int_trap(double(*f)(double),
                double a, double b, int n)
{
    double r, dx, x;
    r = 0.0;
    dx = (b-a)/n;
    for (int i = 1; i <= n-1; i=i+1)
    {
        x = a + i*dx;
        r = r + f(x);
    }
    r = (r + (f(a)+f(b))/2.0)*dx;
    return r;
}
```

24

24

Part 3b:

Riemann sum and … interpolation

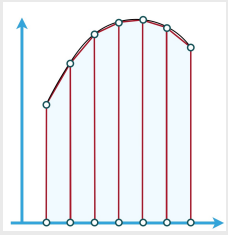
25

Let us talk about interpolation

It turns out that there is a connections between interpolations and the "Trapezoid rule".

This is perhaps evident by looking at the figure below.

It appears that the function is approximated by its linear interpolator, which is indeed the case.



26

26

Let us talk about interpolation – linear interpolation

Consider the first-order interpolation for the i^{th} subinterval:

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

Then, the integral for the i^{th} subinterval

$$\int_a^b f(x)dx = \int_{x_{i-1}}^{x_i} \left[f_{i-1} + \frac{f(x_i) - f(x_{i-1})}{\Delta x}(x - x_{i-1}) \right] dx = \frac{f(x_i) + f(x_{i-1})}{2}\Delta x$$

which is exactly the Trapezoidal approximation!

27

27

Let us talk about interpolation – quadratic interpolation

Using the three-point interpolation, e.g., 3-point Lagrange interpolation for equal subintervals, one may write Simpson's Rule for integration

$$\int_a^b f(x)dx = \frac{\Delta x}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)]$$

Number of n intervals should be even. If n is odd then the last interval should be treated by some other way (e.g. using the trapezoid rule).

Thomas Simpson [1710-1761]

* Useful exercise: Derive the Simpson rule with a pair of slices with an equal interval by using second-order interpolation for $f(x)$ in the region $[x_{i-1}, x_{i+1}]$.

28

Example: C++ $\int_a^b f(x)dx = \frac{\Delta x}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)]$

```
// integration by Simpson rule of f(x) on [a,b]
// f - Function to integrate (supplied by a user)
//    a - Lower limit of integration
//    b - Upper limit of integration
//    s - Result of integration (out)
//    n - number of intervals
double int_simp(double(*f)(double),
                double a, double b, int n)
{
    double s, dx, x;
// if n is odd we add +1 interval to make it even
    if((n/2)*2 != n) {n=n+1;}
    s = 0.0;
    dx = (b-a)/n;
 for ( int i=2; i<=n-1; i=i+2)
    {
        x = a+static_cast<float>(i)*dx;
        s = s + 2.0*f(x) + 4.0*f(x+dx);
    }
    s = (s + f(a) + f(b )+ 4.0*f(a+dx))*dx/3.0;
    return s;
}
```

29

29

Example: Python

```
"""
Simpson integrator for f(x) in [a,b]

f – Function to integrate (supplied by a user)
a – Lower limit of integration
b – Upper limit of integration
s – Result of integration (out)
n – number of intervals

"""

"Trapezoid rule"
def int_trap(f, a, b, n):
    """
    To assure n is a multiple of two
    we can just use the fact that int(n/2) will floor n/2.

    Therefore, f int(n/2) is n, the n is a multiple of two,
    otherwise we should replace n = int((n+1)/2)

    """

    if int(n/2)<n/2.0:
        n = int((n+1)/2)

    s = 0.0
    dx = (b-a)/n;

    for i0 in np.arange(1,n):
        x = a + i0 * dx
        s = s + f(x)

    """
    we have to remember to add the pieces not included above,
    namely the points as x=a, a+dx, b
    """
    s = s + ((f(a) + f(b))/2.0)

    "finally, we need to multiply by dx"

    return s*dx
```

same thing,
slightly different syntax

30

30

Part 4:

Newton-Cotes formulae

31

Newton polynomials

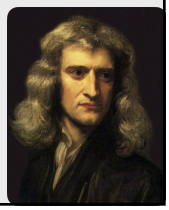
The direct fit polynomial procedure requires a significant amount of effort in the evaluation of the polynomial coefficients. When the function to be integrated is known at equally spaced points, the Newton difference polynomial can be fit to the discrete data with much less effort.

Newton polynomials on equally spaced intervals

$$P_n(x) = f_0 + s\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0 + \frac{s(s-1)(s-2)}{3!}\Delta^3 f_0 + \cdots$$
$$+ \frac{s(s-1)(s-2)\ldots(s-(n-1))}{n!}\Delta^n f_0 + Error$$

where $s = \frac{x-x_0}{\Delta x} = \frac{x-x_0}{h}$, $x = x_0 + sh$

$$Error = \left(\frac{s}{n+1}\right) h^{n+1} f^{n+1}(\xi), \qquad x_0 \le x \le x_n$$



Isaac Newton, 1643 – 1727

32

Newton polynomials

$$I = \int_{x_0}^{b} f(x)df \approx \int_{a}^{b} P_n(x)dx$$

The integral requires that the approximating polynomial be an explicit function of x, whereas the polynomial from the slide before is implicit in x. However, the integral can be transformed into an explicit function of s.

$$s = \frac{x - x_0}{h}, \qquad x = x_0 + sh, \qquad dx = hds$$

$$I = \int_{a}^{b} f(x)df \approx \int_{a}^{b} P_n(x)dx = h\int_{s(a)}^{s(b)} P_n(s)ds$$

For polynomial order n we integrate from $a = x_0$ to $b = x_0 + nh$, then

$$\Delta I = h\int_{0}^{n} P_n(x_0 + sh)\, ds$$

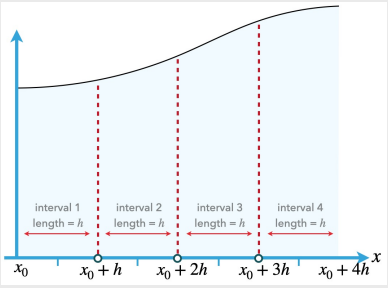
33

33

Pictorial representation

Intervals are points separated by a distance h.

Each choice of the degree n of the interpolating polynomial yields a different Newton-Cotes formula (for nh intervals).



34

34

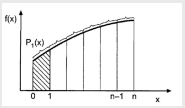
Newton-Cotes: 1st degree polynomial

Let's consider a first degree polynomial for a single interval. In other words, we have just two points:

$$\Delta I \approx h\int_0^1 P_n(x_0 + sh)ds = h\int_0^1 (f_0 + s\Delta f_0)ds = h\left(sf_0 + \frac{s^2}{2}\Delta f_0\right)\Big|_0^1$$

where $s = 1, h = \Delta x$, and $\Delta f_0 = (f_1 - f_0)$, so we get

$$\Delta I \approx \frac{h}{2}(f_0 + f_1)$$



Applying this to all intervals gives

$$I \approx \sum_{i=0}^{n} \Delta I_i = \sum_{i=0}^{n} \frac{h}{2}(f_i + f_{i+1}) = \frac{h}{2}(f_0 + 2f_1 + 2f_2 + \cdots + 2f_{n-1} + f_n)$$

This is again the Trapezoidal approximation!

35

35

Newton-Cotes: 1st degree polynomial – error estimate

The error estimation can be done by integrating the error term in the polynomial.

For a single interval the local error is

$$Error = h\int_0^1 \frac{s(s-1)}{2}h^2 f''(\zeta)ds = -\frac{1}{12}h^3 f''(\zeta) = O(h^3)$$

The total error for equally spaced data is given by

$$\sum_{i=0}^{n-1} -\frac{1}{12}h^3 f''(\zeta) = n\left(-\frac{1}{12}h^3 f''(\zeta)\right)$$

since $n = (x_n - x_0)/h$, Total Error is

$$-\frac{1}{12}(x_n - x_0)h^2 f''(\zeta) = O(h^2)$$

36

36

Newton-Cotes: 2nd degree polynomial

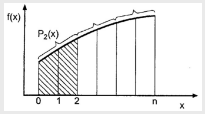
Simpson's 1/3 rule = a second degree polynomial for two increments (three equally spaced points). The upper limit of integration for a single interval is the $s = 2$

$$\Delta I \approx h\int_0^2 \left(f_0 + s\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0\right) ds = \frac{h}{3}(f_0 + 4f_1 + f_2)$$

where we have used: $\Delta f_0 = (f_1 - f_0)$, $\Delta^2 f_0 = f_2 - 2f_1 + f_0$

Applying this to all intervals we get

$$I = \frac{h}{3}[f_0 + 4f_1 + 2f_2 + \cdots + 2f_{n-2} + 4f_{n-1} + f_n]$$



This is again Simpson's rule!

The factor of 1/3 defines the name for "Simpson's 1/3 rule".

The total number of interval must be even.

37

37

Newton-Cotes: 2nd degree polynomial – error estimate

The error estimation can be done by integrating the error term in the polynomial.

For a single interval the local error is

$$Error = h\int_0^2 \frac{s(s-1)(s-2)}{6} h^3 f'''(\zeta) ds = 0$$

The cubic term is identically zero! And the error is in the next degree polynomial

$$Error = h\int_0^2 \frac{s(s-1)(s-2)(s-3)}{24} h^4 f''''(\zeta) ds = -\frac{1}{90} h^5 f''''(\zeta)$$

The error estimation gives the global error $O(h^4)$

Remember that for the trapezoid rule the global error is $O(h^2)$

38

38

Newton-Cotes: 3rd degree polynomial

Simpson's 3/8 rule = a third degree polynomial for four equally spaced points. The upper limit of integration for a single interval is $s = 3$

$$\Delta I \approx h\int_0^3 \left(f_0 + s\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0 + \frac{s(s-1)(s-2)}{3!}\Delta^3 f_0\right) ds$$

Using $\Delta^3 f_0 = f_3 - 3f_2 + 3f_1 - f_0$ we get $\Delta I = \binom{3}{8} h(f_0 + 3f_1 + 3f_2 + f_3)$

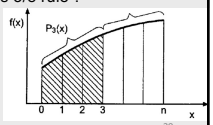
$$I = \frac{3h}{8}[f_0 + 3f_1 + 3f_2 + 2f_3 + 3f_4 + \cdots + 3f_{n-1} + f_n]$$

The factor of 3/8 defines the name for "Simpson's 3/8 rule".

The total number of interval must be a multiple of three

The error estimation gives (again!)

Global error $O(h^4)$ as in the Simpson's 1/3 rule.



39

39

Simpson's 1/3 rule vs. Simpson's 3/8 rule

Both rules have the same order for local error

$$Error_{\frac{1}{3}} = -\frac{1}{90}h^5 f''''(\zeta)$$

$$Error_{\frac{3}{8}} = -\frac{3}{80}h^5 f''''(\zeta)$$

The coefficient in the local error for 1/3 rule is even less than in the local error in 3/8 rule. Hence, Simpson's 1/3 rule should be more accurate.

Is there any value to use Simpson's 3/8 rule?

The 3/8 rule is useful when the total number of intervals is odd. Three intervals can be evaluated by the 3/8 rule and the remaining even number of intervals can be evaluated by the 1/3 rule.

But, normally, it is not worth doing it.

40

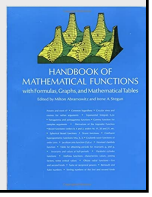
40

Higher-order Newton-Cotes formulae

Numerical integration formulas based on equally spaced increments are called Newton- Cotes formulas. The first 10 Newton-Cotes formulas are presented in Abramowitz and Stegun (1964)*. Newton-Cotes formulas can be expressed in the general form:

$$I = \int_a^b f(x)dx = n\beta h(\alpha_0 f_0 + \alpha_1 f_1 + \cdots) + Error$$

where n denotes both the number of increments and the degree of the polynomial, β and α_i are coefficients



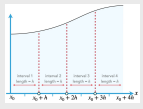
* Abramowitz and Stegun "Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables" (multiple editions) see section 25 "Numerical analysis"

41

Other view on numerical integration

Quadrature is weighted sum of finite number of sample values of integrand function

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$



Then

Rule	degree	weights	global error
Trapezoid rule	n=1	$w_i = \left(\frac{h}{2}, \frac{h}{2}\right)$	$O(h^2)$
Simpson's 1/3 rule	n=2	$w_i = \left(\frac{h}{3}, \frac{4h}{3}, \frac{h}{3}\right)$	$O(h^4)$
Simpson's 3/8 rule	n=3	$w_i = \left(\frac{3h}{8}, \frac{9h}{8}, \frac{9h}{8}, \frac{3h}{8}\right)$	$O(h^4)$
Milne's rule	n=4	$w_i = \left(\frac{14h}{45}, \frac{64h}{45}, \frac{24h}{45}, \frac{64h}{45}, \frac{14h}{45}\right)$	$O(h^6)$

42

42

Integrating error

Generally as N increases, the precision of a method increases

However, as N increases, the round-off error increases

Some evaluations (not exact but gives an idea)*:

Number of steps for highest accuracy:

trapezoid rule	**steps**	**error**
single precision	631	$3*10^{-6}$
double precision	10^{6}	10^{-12}
Simpson's rule	**steps**	**error**
single precision	36	$6*10^{-7}$
double precision	2154	$5*10^{-14}$

* see details in R.H. Landau & M.J.Paez, An Introduction to Computational Physics

43

43

Integrating error (cont.)

The best numerical evaluation of an integral can be obtained with a relatively small number is sub-intervals ($N \sim 1000$ or much less) (not with $N \to \infty$)

It is possible to get an error close to machine precision with Simpson's rule and with other higher-order methods (Newton-Cotes quadratures)

44

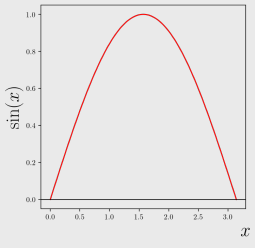
44

Example

Let us consider a couple of examples. First, let's evaluate the following integral using by both the Trapezoid and Simpson's 1/3 rules.

$$\int_0^{\pi} \sin x \, dx$$

Intervals	Trapez.	Simpson
2	1.570796	2.094395
4	1.896119	2.004560
8	1.974232	2.000269
16	1.993570	2.000017
32	1.998393	2.000001
64	1.999598	2.000000
128	1.999900	2.000000
256	1.999975	2.000000
512	1.999994	2.000000
1024	1.999998	2.000000
2048	2.000000	2.000000



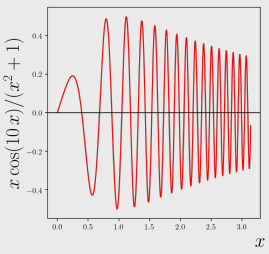
45

45

Example

$$\int_0^{\pi} \frac{x\cos(10x^2)}{x^2+1}dx$$

Intervals	Trapez.	Simpson
2	0.578769	0.811200
4	0.813285	0.891458
8	0.688670	0.647131
16	0.285919	0.151669
32	0.049486	-0.029325
64	0.004360	-0.010682
128	0.001183	0.000124
256	0.000526	0.000306
512	0.000368	0.000315
1024	0.000329	0.000316
2048	0.000319	0.000316
4096	0.000316	0.000316
8192	0.000316	0.000316
16384	0.000316	0.000316
32768	0.000316	0.000316



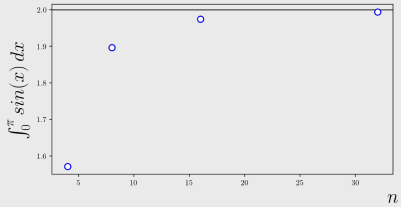
46

46

Extrapolation and Romberg integration

Key idea – use the error estimation by evaluating the integral using two different interval sizes, one with h and a second with a finer value of h/R where $R > 1$. We label these as $I(h)$ and $I(h/R)$.

As one can expect, the difference between these can give us an estimate of the correction needed to added to $I(h/R)$ so as more accurately estimate the integral.



47

47

Extrapolation and Romberg integration

The error estimate can be used both for error control and extrapolation.

Consider a numerical algorithm which approximates an exact calculation with an error that depends on an increment, h

$$f_{exact} = f(h) + Ah^n + Bh^{n+m} + Ch^{n+2m} + \cdots$$

where n is the order of the leading error term and m is the increment in the order of the following error terms. Applying the algorithm at two increment sizes, $h_1 = h$ and, gives $h_2 = h/R$ gives

$$f_{exact} = f(h) + Ah^n + O(h^{n+m})$$

$$f_{exact} = f(h/R) + A(h/R)^n + O(h^{n+m})$$

then, the difference

$$0 = f(h) - f(h/R) + Ah^n - A(h/R)^n + O(h^{n+m})$$

48

48

Extrapolation and Romberg integration (cont.)

$$0 = f(h) - f(h/R) + Ah^n - A(h/R)^n + O(h^{n+m})$$

$$0 = f(h) - f(h/R) + Ah^n(1 - 1/R^n) + O(h^{n+m})$$

Solving for the error term Ah^n gives

$$Error(h) = Ah^n = \frac{R^n}{R^n - 1}\left(f(h/R) - f(h)\right) + O(h^{n+m})$$

$$Error(h/R) = A(h/R)^n = \frac{1}{R^n - 1}\left(f(h/R) - f(h)\right) + O(h^{n+m})$$

The error estimates can be added to the approximate results to yield an improved approximation.

$$f_{exact} = f(h/R) + A(h/R)^n + O(h^{n+m})$$

This process is called *extrapolation*

$$Extrapolated\ value = f(h/R) + \frac{1}{R^n - 1}\left(f(h/R) - f(h)\right) + O(h^{n+m})$$

49

49

Extrapolation and Romberg integration (cont.)

When extrapolation is applied to numerical integration by the trapezoid rule, the result is called Romberg integration.

It can be shown that the error of the composite trapezoid rule has the functional form (the leading term is with $n = 2$)

$$Error = C_1h^2 + C_2h^4 + C_3h^3 + \cdots$$

Let's apply the trapezoid rule for $h/2$ $(R = 2)$

$$Error(h/2) = \frac{1}{2^2 - 1}[I(h/2) - I(h)]$$

Applying the extrapolation formula gives

$$Extrapolated\ value = I(h/2) + Error(h/2) + O(h^4) =$$

$$Extrapolated\ value = I(h/2) + \frac{1}{2^2 - 1}[I(h/2) - I(h)] + O(h^4)$$

Thus we increased the accuracy of the trapezoid rule.

50

50

Summary

1. Numerical integration is mostly based on interpolation (not on Riemann sum definition).
2. The Newton-Cotes formulas, which are based on Newton forward-difference polynomials, give simple integration formulas for equally spaced data.
3. Besides, Newton-Cotes formulae provide error estimates.

51

51