[**DOING PHYSICS WITH MATLAB**](http://www.physics.usyd.edu.au/teach_res/mp/mphome.htm)

**MATHEMATICAL ROUTINES**

# COMPUTATION OF ONE-DIMENSIONAL

# INTEGRALS

Ian Cooper

School of Physics, University of Sydney

ian.cooper@sydney.edu.au

[**DOWNLOAD DIRECTORY FOR MATLAB SCRIPTS**](http://www.physics.usyd.edu.au/teach_res/mp/mscripts)

**math\_integration\_1D.m**

Demonstration mscript evaluating the integral of two functions using a number of different methods

**simpson1d.m**

Function to give the integral of a function using Simpson’s 1/3 rule.

**NUMERICAL INTEGRATION**

**COMPUTATION OF ONE-DIMENSIONAL INTEGRALS**

The function **simpson1d.m** is a very versatile , accurate and easy to implement function that can be used to evaluate a definite integral of a function between a lower bound and an upper bound. It is easier to use than the standard Matlab integration functions such as quad. The function **simpson1d.m** is described in detail below.

We want to compute a number expressing the definite integral of the function *f*(*x*) between two specific limits *a* and *b*



The evaluation of such integrals is often called ***quadrature***.

We will consider the following integrands to test the accuracy of different integration procedures:

(1) 

(2) 

|  |
| --- |
|  |

The integrals are evaluated over a quarter cycle in an attempt to find a minimum number of partitions of the domain that produce accurate results within a reasonable time. For the testing procedures, the following integrals are to be evaluated from

*a* = 0 to *b* = π/2

 and 

These integrals can be evaluated analytically and the exact answers are





The numerical procedures assume that neither the integrand *f*(*x*) nor any of its derivatives become infinite at any point over the domain of the integration [*a*, *b*], and that the limits of integration *a* and *b* are finite. Furthermore, we assume that *f*(*x*) can be computed, or its values are known at *N* points *xc* where *c* = 1, 2, …, *N*  that are distributed in some manner over the domain [*a*, *b*] with *x*1 = *a* and *xN* = *b*.

The qualifier *closed* signifies that the integration involves the values of the integrand at both the end-points. If only one or none of the end points are included, the qualifiers used are *semi-closed* and *open* respectively.

A major problem that arises with non-adaptive methods is that the number *N* of partitions of the function required to provide a given accuracy is initially unknown. One approach to this problem is to successively double the number of partitions, and compare the results as the number of partitions increase.

# Closed Rectangle Rule

The region from *a* to *b* is divided into *N* rectangles of equal width Δ*x* ≡ *h* where



with each rectangle centre occurring at the points *x*1 = *a*, *x*2 = *x*1+ *h*, … *x*N = *b*.

For example, the plot below shows the domain divided into 4 rectangles, *N* = 4. However, the sub-division of the domain by this means extends from

*a* - *h*/2 to *b* + *h*/2 and not simply from *a* to b.

The integral is approximated by the contribution from each rectangle, such that



For the case shown in the plot above where *N* = 4, the integral is approximated by

 where .

# Open Midpoint Rule

The mid-point rule is the first member of a family of open Newton-Cotes rules corresponding to quadratic, cubic and higher-order interpolating polynomials with evenly spaced points. In the open midpoint rule, the area of each rectangle is added to find the integral. The domain *a* to *b* in divided into *N* partitions, with the width of each partition being *h* = (*b* – *a*)/*N*. The function is evaluated at the midpoint *x*i of each rectangle where *x*i = *a* + (*h*/2)(2*N*-1). The value of the integral is then given by



# Trapezoidal Rule

The function *f*(*x*) is approximated by a straight line segment connecting adjacent points. The area under the curve is then approximated by adding the area of each trapezium. The interval *a* to *b* is divided into *N*-1 partitions of width *h* (*h* ≡ Δ*x*) where *h* = (*b* – *a*)/(*N* - 1). The area of each partition is simply the area of a trapezium which is its base times its mean height. The area of the ith trapezium is

 where *f*i ≡ *f*(*x*i) and *i* = 1, 2, …..*N*-1

Summing all the trapezia gives the ***composite, closed trapezoidal rule***



For evenly distributed spacings, the composite rule is equivalent to the trapezoidal version of the closed ***Newton-cotes rule***.

The Matlab function, trapz implements a procedure to calculate the integral by the trapezoidal rule. For example, the integration of the function *y*1 w.r.t the variable *x*

Integral\_1 = trapz(x,y1) % estimate of the integral

# Simpson’s 1/3 rule

This rule is based on using a quadratic polynomial approximation to the function *f*(*x*) over a pair of partitions. *N*-1 is the number of partitions where *N* must be **odd** and

*h* = (*b* – *a*) / (*N*-1). The integral is expressed below and is known as ***composite Simpson’s 1/3 rule***.



Simpson’s rule can be written vector form as



where .

Simpson’s rule is an example of a ***closed Newton’s-Cotes*** formula for integration. Other examples can be obtained by fitting higher degree polynomials through the appropriate number of points. In general we fit a polynomial of degree *N* through *N* +1 points. The resulting polynomials can them be integrated to provide an integration formula. Because of the lurking oscillations associated with the Gibbs effect, higher-order formulas are not used for practical integration.

**simpson1d.m**

The function *f* and the lower bound *a* and the upper bound *b* are passed onto the function (in the order *f*, *a*, *b*) and the function returns the value of the integral

function integral = simpson1d(f,a,b)

% [1D] integration - Simpson's 1/3 rule

% f function a = lower bound b = upper bound

% Must have odd number of data points

% Simpson's coefficients 1 4 2 4 ... 2 4 1

numS = length(f); % number of data points

sc = 2\*ones(numS,1);

sc(2:2:numS-1) = 4;

sc(1) = 1; sc(numS) = 1;

h = (b-a)/(numS-1);

integral = (h/3) \* f \* sc;

**EXAMPLES**

(1)  (2) 

|  |  |  |
| --- | --- | --- |
| **Method** | **Estimate** | **N** |
| Closed Rectangle | (1) 1.0950  (2) 1.0950 + 1.0950i | 9  99 |
| (1) 1.0080  (2) 1.0080 + 1.0080i |
| Open-Midpoint | (1) 0.9978  (2) 0.9978 + 0.9978i | 9  99 |
| (1) 0.9968  (2) 0.9733 + 0.9733i |
| Trapezoidal | (1) 1.0000  (2) 1.0000 + 1.0000i | 9  99 |
| (1) 1.0000  (2) 1.0000 + 1.0000i |
| Simpson’s 1/3 | (1) 1.0000  (2) 1.0000 + 1.0000i | 9 |

All the integrations took less than one second on a fast Windows computer. Even with only 9 points, the Simpson’s 1/3 rule estimate was equal to the exact values.