DOING PHYSICS WITH MATLAB

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

math_1d_integration.m
mscript to evaluate the integral

$$\int_a^b f(x)\,dx$$

using Simpson's 1/3 rule

simpson1d.m
Function used to estimate of the integral 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.

View more detailed notes on a numerical approach to integration

---

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***.

$$I = \frac{h}{3}\left\{\left(f_1 + f_N + 4(f_2 + f_4 + ... + f_{N-2}\right) + 2(f_3 + f_5 + ... + f_{N-1})\right\}$$

Simpson's rule can be written vector form as

$$I = \frac{h}{3}\mathbf{c}\mathbf{f}^{\mathrm{T}}$$

where $\mathbf{c} = \left[1\;4\;2\;4\;...\;2\;4\;1\right]$ and $\mathbf{f} = \left[\,f_1\;f_2\;...\;f_N\,\right]$.

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;
```

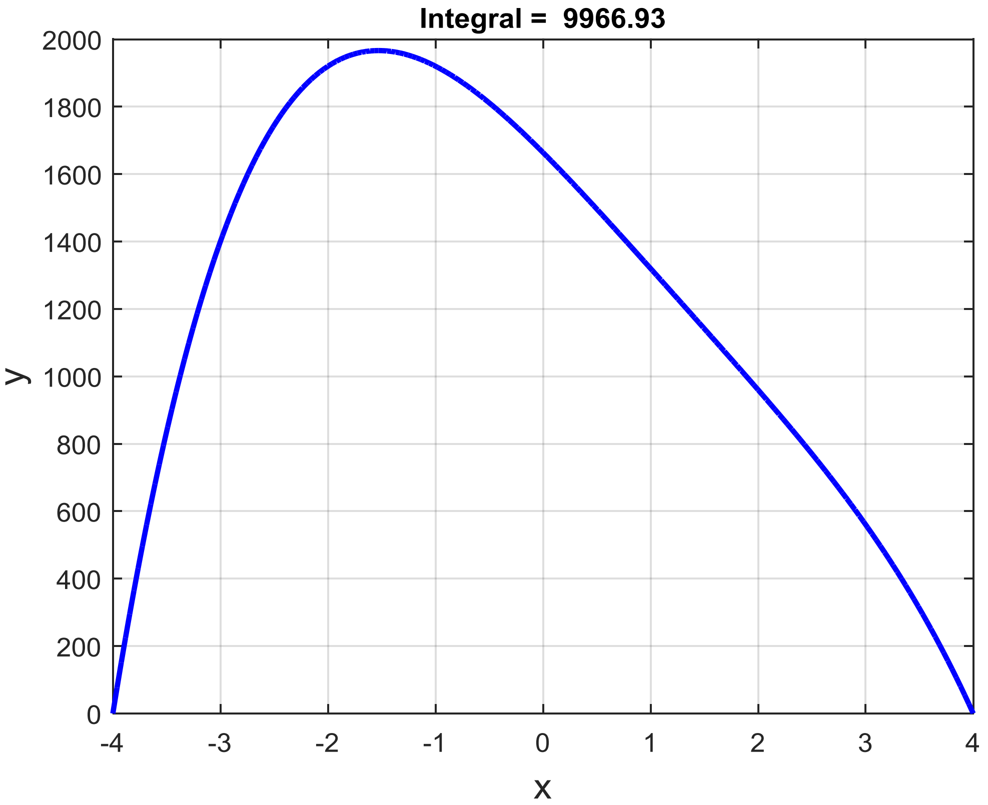
math_1d_integration.m

You need to modify the mscript to evaluate the integral of your function.
Input parameters to define the function

N	number of partitions (must be an odd number)
a	lower bound of integral
b	upper bound of integral
y	function to be integrated
tx	title for X-axis
ty	title for Y-axis

The mscript outputs the value of the integral in the Command Window. A graph of the function is plotted in a Figure Window.

```
clear all
close all
clc
format long


% INPUTS
===============================================================

% Number of partitions
    N = 9999;
% Lower bound
    a = -4;
% Upper bound
    b = 4;

             x = linspace(a,b,N);
% Function to  be integrated
   y = -4.*x.^4  + 20.*x.^3 - 40.*x.^2 - 320.*x + 1664;

% X and Y label for graph axes
    tx = 'x';
    ty = 'y';

% OUTPUTS
================================================================

% Simpson's 1/3 rule ------------------------------------------
    Integral = simpson1d(y,a,b);

    disp('  ');
    format long
    disp('Integral =  ');
    disp(Integral);

% title
   t1 = 'Integral =  ';
   t2 = num2str(Integral,6);
   tm = [t1 t2];

% Graphics ----------------------------------------------------
    figure(1)
    fs = 14;
    plot(x,y,'b','linewidth',2);
    xlabel(tx,'fontsize',fs); ylabel(ty,'fontsize',fs);
    title(tm);
    box on;
    grid on;
```



Integral =
    9.966933333333331e+03