Adapted from [1, 2]

AVERAGE VALUE OF A SEQUENCE

Consider a *sequence* of numbers $a = [a_k]_1^n$. The *average value of the sequence*, $\bar{a}$, is the arithmetic average of the sequence series:

$$\bar{a} = \frac{1}{n} \sum_k a_k \, .$$

By analogy, extend this idea to find *function* average values.

AVERAGE VALUE OF A FUNCTION

Consider a function f integrable over the interval $a \leq x \leq b$. Then, estimate the function average value $\bar{f}_{\text{est}}$ by partitioning the interval into subintervals of width $\triangle x = (b-a)/n$, by picking a point x_k in each subinterval, by calculating the function values $\{f[x_k]\}$ at each x_k and by averaging such values:

$$\bar{f}_{\text{est}} = \frac{1}{n} \sum_k f[x_k] \, .$$

Note that, as n increases, the estimate improves – a hint to work with calculus.

Multiply and divide thus the last equation by $\triangle x$, then use $n \triangle x = (b-a)$ to have

$$\bar{f}_{\text{est}} = \frac{1}{b-a} \sum_k f[x_k] \triangle x \, .$$

Calculate next the average value of f by taking the limit of the last equation, provided such a limit exists:

$$\bar{f} = \frac{1}{b-a} \lim_{n\to\infty} \sum_{k=1}^{n} f[x_k] \triangle x = \frac{1}{b-a} \int_{[a,b]} f[x] \, \mathrm{d}x \, .$$

Finally, define the *average value* of a function f integrable over the interval $i = [a,b]$ as

$$\bar{f} \doteq \frac{1}{b-a} \int_i f \, ;$$

N.B. : functional notation for the integral, [3, p. 69]

that is, *the average value of a function over an interval equals the integral of the function divided by the size of the interval.*

MONTE-CARLO INTEGRATION

Monte-Carlo integration is a procedure to estimate a value for the integral of a function over an interval not by partitioning the interval and picking values at the subintervals, but by *randomly* picking numbers within the whole interval and with them calculating function values. The process of picking random numbers within the interval is called *random sampling*.

The process is the inverse to that of finding the average value using integration. In Monte-Carlo integration, the average value of a function is estimated first and then the value of the integral estimated by

$$\int_i f = \bar{f}\,(b-a)\ .$$

This procedure is formalized as follows.

Integration

Suppose we wish to estimate the value of the integral $l = \int_{[a,b]} f$ of a function f.

First, *randomly* choose n points $\{x_k\}$ within $a \leq x \leq b$, use these to calculate the values of f, $\{f\,[x_k]\}$ and then estimate the average value of f:

$$\bar{f}_{\text{est}} = \frac{1}{n}\sum_{k=1}^{n} f\,[x_k]$$

Estimate finally the value of the integral as

$$l_{\text{est}} = (b-a)\,\bar{f}_{\text{est}}\ .$$

Integration uncertainty

The *central limit theorem* of probability theory provides with an estimate for the *uncertainty* in Monte-Carlo integration.

Suppose the average value of a function f is estimated by random sampling n numbers, *aka* the *sample size*,

$$\bar{f}_{\text{est}} = \frac{1}{n}\sum_{k=1}^{n} f\,[x_k]\ .$$

Then, the *variance* of the estimated average is

$$\operatorname{var}\bar{f}_{\text{est}} = \frac{\sigma^2}{n}\ ,$$

where σ is the variance of f.

Measure the uncertainty u by the standard deviation:

$$u = \frac{\sigma}{\sqrt{n}}\ .$$

Note that the uncertainty goes to zero like $1/\sqrt{n}$; *i.e.*, for example, to decrease the uncertainty by a factor of 1000, increase the sample size by a factor of 1 000 000.

Example

Estimate the value of the integral

$$l = \int_{[0,\tau]} \exp[-x] \sin[x] \, \mathrm{d}x \,,$$

where $\tau \doteq 2\pi$.

With the aid of computer generated (pseudo) random numbers (see appendix A for the computer source code), it was possible to estimate the value of the integral and its uncertainty as 0.498 19(37), for a sample size of 100 000. The result of the Monte-Carlo integration differs in 0.20% from the reference value of 0.499 066 3.

N.B. : The reference value is $(1 - \exp[-2\pi])/2 \sim 0.4990663$.

[1] Q. Fang, *Integral Properties and Average Value* (2014).

[2] Unknown, *Monte-Carlo Integration Simulation*, AMTH142 (2007).

[3] T. M. Apostol, *Calculus, One-Variable Calculus with an Introduction to Linear Algebra* (Xerox, 1967).

A MONTE-CARLO INTEGRATION (PROCEDURAL)

This section contains the Ruby code used to approximate the integral in the example of Monte-Carlo integration. The programming paradigm used was *Procedural*.

The code was written, tested and run using `ruby 2.1.2p95 (2014-05-08 revision 45877) [x86_64-darwin13.0]`.

```
#!/usr/local/bin/ruby
#--
# Have faith in the way things are.
#
# monte-carlo.rb
# date: 2014.08.05
#++

# == Description
# Estimate integral: int{exp(-x)sin(x) dx}{0 to tau} by Monte-Carlo
#
# == Third-party libs
# The code depends on the 'descriptive_statistics' gem. To install it
# $ gem install 'descriptive_statistics'
#
# == Algorithm
# define the integrand
# define the sample size = n
# for every n
#   randomly choose n points within [0, tau],
#   use them to calculate values for the integrand
#   store the values in an array
# calculate the function average value from the array
# calculate the function integral value
# calculate the uncertainty
# print results
#
# == Author
# rimbaud1854
#
# == Copyright
# Copyright (c) 2014 rimbaudcode
# Licensed under GPLv3+. No warranty provided.

def integrand x
  exp(-x) * sin(x)
end

def main args
  include Math
  require 'descriptive_statistics'

  tau        = 2.0 * PI
```

```
  sample_size = 100_000

  low_bound = 0.0
  up_bound  = tau
  interval  = (low_bound..up_bound)

  function_values = []
  sample_size.times do
      random_point = rand interval
      function_values << integrand(random_point)
    end

  average_function = function_values.mean
  average_integral = (up_bound - low_bound) * average_function
  uncertainty      = function_values.standard_deviation / sqrt(sample_size
       )
  result           = [average_integral, uncertainty]

  p result

  exit
end

if $0 == __FILE__
  begin
    exit main $*
  rescue
    $stderr.puts "#{$!}"
    $@.each do |item| $stderr.puts item end
    abort
  ensure
    #
  end
end
```

B MONTE-CARLO INTEGRATION (OBJECT ORIENTED PROGRAMMING)

This section contains the `Ruby` code used to approximate the integral in the example of Monte-Carlo integration. The programming paradigm used was *Object Oriented*.

The code was written, tested and run using `ruby 2.1.2p95 (2014-05-08 revision 45877) [x86_64-darwin13.0]`.

```
#!/usr/local/bin/ruby
#--
# Have faith in the way things are.
#
# monte-carlo.rb
# date: 2014.08.05
#++

# == Description
# Estimation of integrals by Monte-Carlo (integration)
#
# == File
# This file contains the main function
#
# == Author
# rimbaud1854
#
# == Copyright
# Copyright (c) 2014 rimbaudcode
# Licensed under GPLv3+. No warranty provided.

$LOAD_PATH << File.expand_path(File.join(__dir__, '../lib'))

def integrand x
  Math::exp(-x) * Math::sin(x)
end

def main args
  require 'monte-carlo'

  tau         = 2.0 * Math::PI
  sample_size = 100_000
  interval    = (0.0..tau)
  monte       = MonteCarlo.new integrand(0.0), sample_size, interval

  monte.calculate_integrand_values
  monte.calculate_average_integrand
  monte.calculate_average_integral
  monte.calculate_uncertainty
  p monte.average_integral_uncertainty

  exit
```

```
end

if $0 == __FILE__
  begin
    exit main $*
  rescue
    $stderr.puts "#{$!}"
    $@.each do |item| $stderr.puts item end
    abort
  ensure
    #
  end
end

#!/usr/local/bin/ruby
#--
# Have faith in the way things are.
#
# monte-carlo.rb
# date: 2014.08.05
#++

# == Description
# Class to model Monte-Carlo integration
#
# == Third-party libs
# The code depends on the 'descriptive_statistics' gem:
# $ gem install 'descriptive_statistics'
#
# == Algorithm
# define the integrand
# define the sample size = n
# for every n
#   randomly choose n points within [0, tau],
#   use them to calculate values for the integrand
#   store the values in an array
# calculate the function average value from the array
# calculate the function integral value
# calculate the uncertainty
# print results
#
# == Author
# rimbaud1854
#
# == Copyright
# Copyright (c) 2014 rimbaudcode
# Licensed under GPLv3+. No warranty provided.

class MonteCarlo < Object

  require 'descriptive_statistics'

  def initialize integrand, sample_size, interval
    @integrand   = integrand
    @sample_size = sample_size
    @interval    = interval
  end

  def calculate_integrand_values
    @integrand_values = []

    @sample_size.times do
      @integrand_values << integrand(rand @interval)
    end
```

```
  end

  def calculate_average_integrand
    @average_integrand = @integrand_values.mean
  end

  def calculate_average_integral
    @average_integral = (@interval.last - @interval.first) *
         @average_integrand
  end

  def calculate_uncertainty
    @uncertainty = @integrand_values.standard_deviation / Math::sqrt(
         @sample_size)
  end

  def average_integral_uncertainty
    [@average_integral, @uncertainty]
  end

end
```