RiemannIntegral.R

ATE

Thu Aug 24 14:57:57 2017

```
#
# Computation of the Riemann integral using the limit of the lower
# or the upper Riemann Sum.
#
library(ggplot2)

# 0. Define the boundaries between which we will compute the integral.
# 1. Define the partition (Not forcly equals length)
# 2. Compute the max length partition (just for fun at first purpose)
# 3. Define the Ys points which will serve to compute the integral.
# 4. Compute the sum.

# -> It would be usefull to take the same length between each partition points.
# So that we could manage the limit just by minimize the max length between
# each partition point. This max length will be denoted by [pi_max]

#
# 0. The boundaries:
#
# [a]: lower boundary
# [b]: Upper boundary
#
a <- 0
b <- 4

#
# 1. 2. Define the partition point + the length partition.
#
# [partion_point]: vector with length =
# [max_length]: max length between the partition point. For this example we will
#               set the same length between all partition point. So the max_length
#               variable could be simply named [length] ...
# [partition_tmp]: Temporary variable used to define the partition size.
#                  In my example I want a size of 100 items for the partition.
#                  This variable will also define the size of the vector [partition_point]
#
# In the following example I divide the segment's length by one hundred:
#
partition_tmp <- 100
max_length <- (b-a) / partition_tmp
partition_point <- seq(a, b, by = max_length)

#
# 3. Define the Ys points which will serve to compute the integral.
# For this point I will define a dummy function continuous on R (x^2)
#  (i): The Upper Riemann integral take the max point (yaxis)
#  (ii): The Lower Riemann integral take the min point (yaxis) between the integral(xaxis)
```

```r
#
x_square <- function(x) x^2
upper <- lower <- vector()
for(i in 2:length(partition_point))
  upper[i-1] <- optimise(x_square, lower = partition_point[i-1],
                         upper = partition_point[i],
                         maximum = T)$objective

for(i in 2:length(partition_point))
  lower[i-1] <- optimise(x_square, lower = partition_point[i-1],
                         upper = partition_point[i])$objective

#
# Finally and because all the gap between the partition point have the same
# length, we could use max_length instead of computing the length between
# all the points.
#
# By consequence the Upper Riemann integral is:
upper_riemann_sum <- sum(upper * max_length)
# And the lower:
lower_riemann_sum <- sum(lower * max_length)
```