Trapezoidal, Simpson, and Romberg Integration Rules

Hannah B. Labana, Jomar M. Leaño, Christian Anthony C. Stewart

2024-02-29

$\int_1^2 sin(x^2)dx$

1. Trapezoid Rule

```
f = function(x) {sin(x^2)}

# Define the limits of integration
a = 1
b = 2

# Number of intervals
n = 18

# Width of each interval
h = (b - a) / n

# Trapezoid rule formula
int = (h / 2) * (f(a) + 2 * sum(sapply(1:(n - 1), function(i) f(a + i * h))) + f(b))

actual = integrate(f, lower = 1, upper = 2)
```

```
## [1] "Estimated value:  0.493556795377587"
```

```
## [1] "Actual value:  0.494508187620375"
```

```
## [1] "Error:  0.000951392242788007"
```

for $n = 18$, we are guaranteed

$$\left| \int_1^2 sin(x^2) - M_n \right| \leq 0.001$$

take note that this took around 18 intervals between 1 and 2 which is a lot.

2. Simpson's Rule

```r
f = function(x) {sin(x^2)}

# Define the limits of integration
a = 1
b = 2

# Number of intervals
n = 6

simpsons_rule <- function(f, a, b, n) {
  if (n %% 2 != 0) {
    stop("Number of subintervals must be even.")
  }

  h <- (b - a) / n
  integral <- f(a) + f(b)  # endpoints

  # Odd indexed points
  odd_sum <- sum(sapply(seq(1, n, by = 2), function(i) f(a + i * h)))
  # Even indexed points
  even_sum <- sum(sapply(seq(2, n-1, by = 2), function(i) f(a + i * h)))

  integral <- integral + 4 * odd_sum + 2 * even_sum
  integral <- integral * h / 3

  return(integral)
}

actual = integrate(f, lower = 1, upper = 2)
```

```
## [1] "Estimated value:  0.494850220366639"
```

```
## [1] "Actual value:  0.494508187620375"
```

```
## [1] "Error:  0.000342032746263599"
```

for $n = 6$, we are guaranteed

$$\left| \int_1^2 sin(x^2) - M_n \right| \leq 0.001$$

take note that this took around 6 intervals between 1 and 2 which is significantly more efficient compared to Trapezoid Rule which took 18 intervals.

Romberg Integration

```r
romberg_integration = function(num_of_iter, func, lower_bound, upper_bound) {
    deltax_array = array(data = NA, dim = num_of_iter+1)
    T_array = array(data = NA, dim = num_of_iter+1)
```

```r
    Tx_array = array(data = NA, dim = num_of_iter+1)

    ptr = 1
    prev = 1
    after = 2
    deltax = (upper_bound - lower_bound) / 2

    while(ptr < num_of_iter+2) {
        deltax_array[ptr] = deltax
        num_of_subintervals = 2^num_of_iter
        temp = 0
        x = 1

        while(x < 2^ptr) {
            temp = temp + 2 * func(lower_bound + x * deltax)
            x = x + 1
        }

        T_d = (deltax/2) * (func(lower_bound) + temp + func(upper_bound))
        T_array[ptr] = T_d

        if(ptr > 1) {
            Tx = ((2^(2*prev)) * T_array[after] - T_array[prev]) / (2^(2*prev) - 1)
            Tx_array[prev] = Tx
            prev = prev + 1
            after = after + 1
        }

        ptr = ptr + 1
        deltax = deltax / 2
    }

    result_df <- data.frame(deltax = deltax_array, T = T_array, Tx = Tx_array + 1)
    result_df$deltax <- deltax_array
    result_df$T <- T_array
    result_df$Tx <- Tx_array

    return(list(Tx_array, result_df))
}

func_to_integrate = function(x) {
    return(sin(x^2))
}

result = romberg_integration(6, func_to_integrate, 1, 2)
answer = result[1]
result_dataframe = result[2]
```

```r
result_dataframe
```

```
## [[1]]
##      deltax         T        Tx
## 1 0.5000000 0.4102037 0.4963932
```

```
## 2 0.2500000 0.4748458 0.4906597
## 3 0.1250000 0.4896713 0.4933614
## 4 0.0625000 0.4933037 0.4942109
## 5 0.0312500 0.4942074 0.4944332
## 6 0.0156250 0.4944330 0.4944894
## 7 0.0078125 0.4944894        NA
```

```
answer
```

```
## [[1]]
## [1] 0.4963932 0.4906597 0.4933614 0.4942109 0.4944332 0.4944894        NA
```

We can see the estimated answer at the 6th iteration = 0.4944894
If we use the pracma library, it gives 0.4945082 with 6 iterations.

```r
f <- function(x) {
  sin(x^2)
}

rom <- romberg(f, 1, 2)
rom
```

```
## $value
## [1] 0.4945082
##
## $iter
## [1] 6
##
## $rel.error
## [1] 1.088019e-14
```

Error = |Expected - Actual|

```r
0.4945082 - 0.4944894
```

```
## [1] 1.88e-05
```

```r
integrate(f,1,2)
```

```
## 0.4945082 with absolute error < 7.5e-15
```