



# Komputasi Statistika

#12-13 Meeting

Numerical Integration :  
Trapezoidal, Simpson's, dan Adaptive quadrature

Ferdian Bangkit Wijaya, S.Stat., M.Si  
NIP. 199005202024061001





# Numerical Integration

## Trapezoidal Rule

Dalam kalkulus dan analisis matematika, integral tentu  $\int_a^b f(x)dx$  merepresentasikan luas daerah di bawah kurva fungsi  $f(x)$  yang dibatasi oleh interval  $[a, b]$ . Secara teoritis, nilai ini dapat dihitung menggunakan Teorema Dasar Kalkulus dengan mencari antiturunan (integral tak tentu) dari fungsi tersebut.

Salah satu metode integrasi numerik yang paling fundamental adalah **Aturan Trapezoidal**. Metode ini bekerja dengan prinsip aproksimasi geometris. Alih-alih menghitung luas area di bawah kurva yang melengkung secara langsung, metode ini mendekati area tersebut menggunakan bangun datar **trapesium**.



# Numerical Integration

## Trapezoidal Rule

### Konsep dan Rumus Umum

Aturan Trapezoidal menghampiri nilai integral tentu  $\int_a^b f(x)dx$  dengan membagi area di bawah kurva menjadi  $n$  segmen trapesium.

#### Lebar Segmen ( $h$ ):

$$h = \frac{b - a}{n}$$

#### Rumus Integral ( $I$ ):

$$I \approx \frac{h}{2} \left[ f(x_0) + 2 \left( \sum_{i=1}^{n-1} f(x_i) \right) + f(x_n) \right]$$



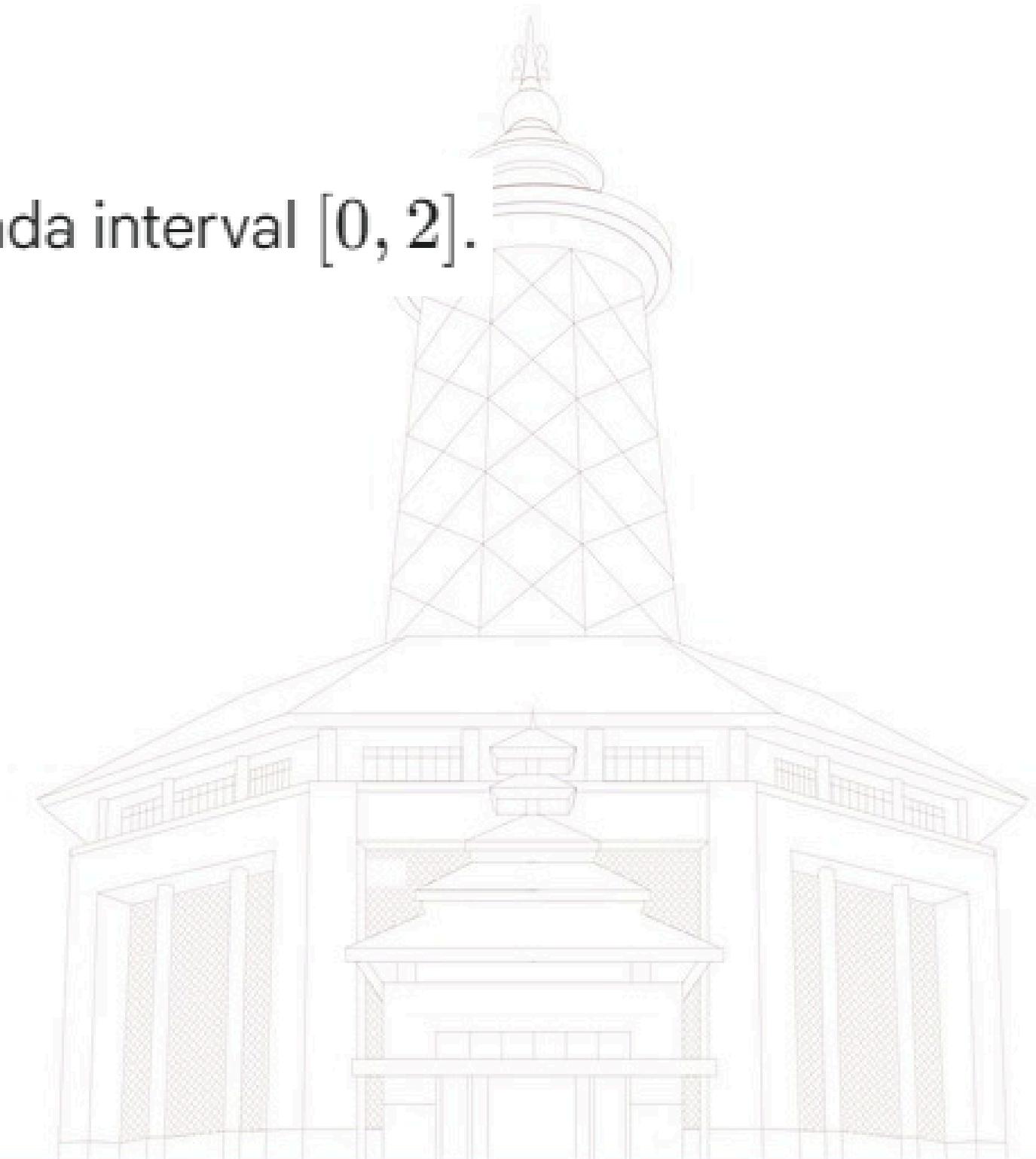
# Numerical Integration

## Trapezoidal Rule (Contoh 1)

**Kasus:** Hitung integral fungsi berikut pada interval  $[0, 2]$ .

$$f(x) = x^2$$

Nilai Eksak (Aljabar): 2.666667



# Numerical Integration

## Trapezoidal Rule (Contoh 1)

$$I \approx \frac{h}{2} \left[ f(x_0) + 2 \left( \sum_{i=1}^{n-1} f(x_i) \right) + f(x_n) \right]$$

### Iterasi 1: Menggunakan 1 Segmen ( $n = 1$ )

- Lebar segmen ( $h$ ):  $(2 - 0)/1 = 2.0$
- Evaluasi Fungsi:  $f(0) = 0, \quad f(2) = 4$
- Hitung Integral ( $I_1$ ):

$$I_1 \approx \frac{2.0}{2} [0 + 4] = 4.000000$$

- Estimasi Error: -

### Iterasi 2: Menggunakan 2 Segmen ( $n = 2$ )

- Lebar segmen ( $h$ ):  $(2 - 0)/2 = 1.0$
- Titik Grid ( $x$ ):  $0, 1, 2$
- Evaluasi Fungsi:  $f(0) = 0, \quad f(1) = 1, \quad f(2) = 4$
- Hitung Integral ( $I_2$ ):

$$I_2 \approx \frac{1.0}{2} [0 + 2(1) + 4] = 0.5 \times 6 = 3.000000$$

- Estimasi Error:  $|3.000 - 4.000| = 1.000000$

# Numerical Integration

## Trapezoidal Rule (Contoh 1)

$$I \approx \frac{h}{2} \left[ f(x_0) + 2 \left( \sum_{i=1}^{n-1} f(x_i) \right) + f(x_n) \right]$$

### Iterasi 3: Menggunakan 4 Segmen ( $n = 4$ )

- Lebar segmen ( $h$ ):  $(2 - 0)/4 = 0.5$
- **Penentuan Titik Grid ( $x$ ):** Titik didapat dengan menambahkan  $h$  secara berulang dari batas bawah ( $a = 0$ ).
  - $x_0 = 0.0$  (Awal)
  - $x_1 = 0.0 + 0.5 = 0.5$  (Tengah)
  - $x_2 = 0.5 + 0.5 = 1.0$  (Tengah)
  - $x_3 = 1.0 + 0.5 = 1.5$  (Tengah)
  - $x_4 = 1.5 + 0.5 = 2.0$  (Akhir)

- **Evaluasi Fungsi ( $f(x) = x^2$ ):**
  - Komponen Ujung:  $f(0) + f(2) = 0 + 4 = 4.00$
  - Komponen Tengah 1:  $f(0.5) = 0.5^2 = 0.25$
  - Komponen Tengah 2:  $f(1.0) = 1.0^2 = 1.00$
  - Komponen Tengah 3:  $f(1.5) = 1.5^2 = 2.25$
  - Jumlah Tengah:  $0.25 + 1.00 + 2.25 = 3.50$

- **Hitung Integral ( $I_4$ ):**

$$I_4 \approx \frac{0.5}{2} [4.00 + 2(3.50)] = 0.25 \times [4 + 7] = 0.25 \times 11 = 2.750000$$

- **Estimasi Error:**  $|2.750 - 3.000| = 0.250000$



# Numerical Integration

## Trapezoidal Rule (Contoh 2)

**Kasus:** Hitung integral fungsi berikut pada interval  $[0, 2]$ .

$$f(x) = e^{-x^2}$$

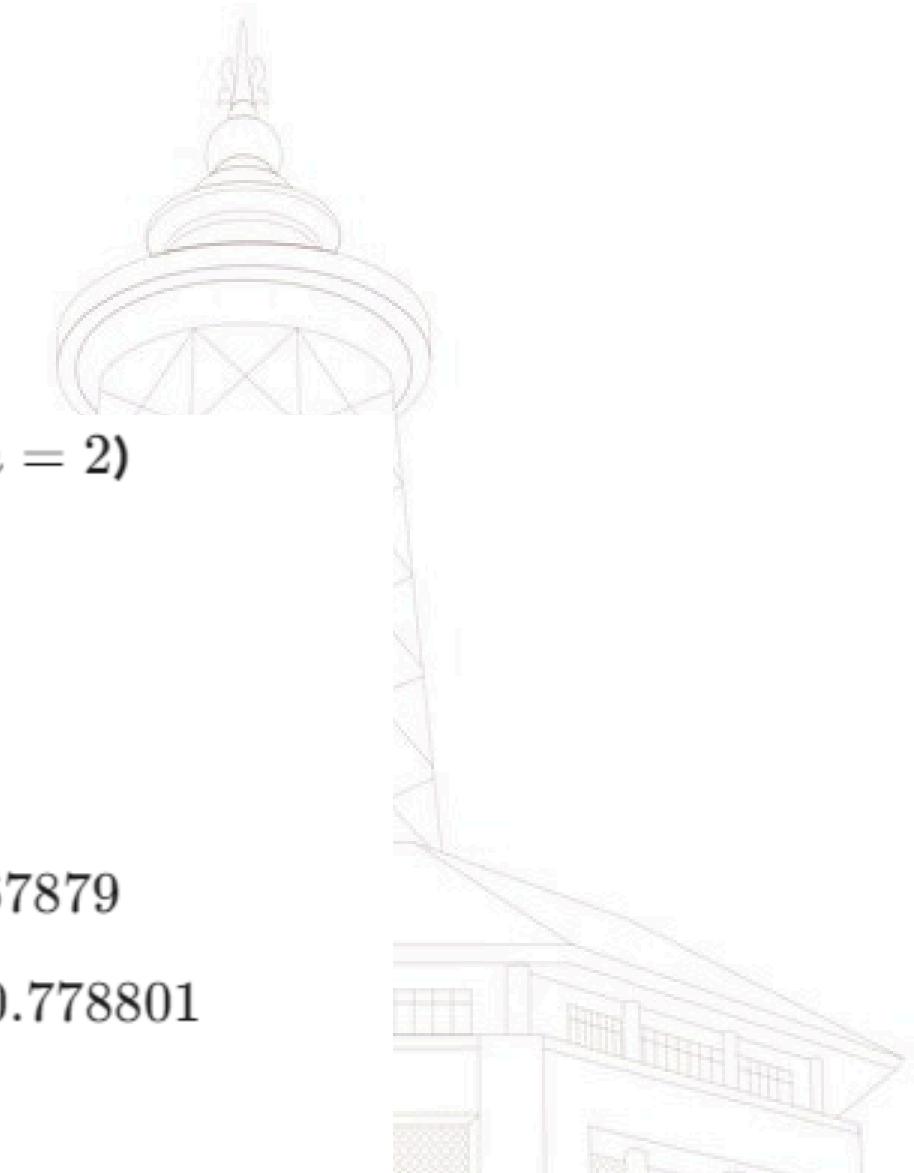
**Nilai Referensi:** 0.746824



# Numerical Integration

## Trapezoidal Rule (Contoh 2)

$$I \approx \frac{h}{2} \left[ f(x_0) + 2 \left( \sum_{i=1}^{n-1} f(x_i) \right) + f(x_n) \right]$$



### Iterasi 1: Menggunakan 1 Segmen ( $n = 1$ )

- Lebar segmen ( $h$ ): 1.0
- Evaluasi Fungsi:  $f(0) = 1$ ,  $f(1) = 0.367879$
- Hitung Integral ( $I_1$ ):

$$I_1 \approx \frac{1.0}{2} [1 + 0.367879] = 0.683940$$

- Estimasi Error: -

### Iterasi 2: Menggunakan 2 Segmen ( $n = 2$ )

- Lebar segmen ( $h$ ): 0.5
- Titik Grid ( $x$ ): 0, 0.5, 1
- Evaluasi Fungsi:
  - Ujung:  $1 + 0.367879 = 1.367879$
  - Tengah:  $f(0.5) = e^{-0.25} = 0.778801$
- Hitung Integral ( $I_2$ ):

$$I_2 \approx \frac{0.5}{2} [1.367879 + 2(0.778801)] = 0.25 \times 2.925481 = 0.731370$$

- Estimasi Error:  $|0.731370 - 0.683940| = 0.047430$

# Numerical Integration

## Trapezoidal Rule (Contoh 2)

$$I \approx \frac{h}{2} \left[ f(x_0) + 2 \left( \sum_{i=1}^{n-1} f(x_i) \right) + f(x_n) \right]$$

**Iterasi 3: Menggunakan 4 Segmen ( $n = 4$ )**

- Lebar segmen ( $h$ ):  $(1 - 0)/4 = 0.25$
- **Penentuan Titik Grid ( $x$ ):** Titik didapat dengan menambahkan  $h = 0.25$  dari batas bawah ( $a = 0$ ).
  - $x_0 = 0.00$  (Awal)
  - $x_1 = 0.00 + 0.25 = 0.25$  (Tengah)
  - $x_2 = 0.25 + 0.25 = 0.50$  (Tengah)
  - $x_3 = 0.50 + 0.25 = 0.75$  (Tengah)
  - $x_4 = 0.75 + 0.25 = 1.00$  (Akhir)

- **Evaluasi Fungsi ( $f(x) = e^{-x^2}$ ):**
  - Komponen Ujung ( $f(0) + f(1)$ ):  $1.000000 + 0.367879 = 1.367879$
  - Komponen Tengah 1:  $f(0.25) = e^{-0.0625} = 0.939413$
  - Komponen Tengah 2:  $f(0.50) = e^{-0.2500} = 0.778801$
  - Komponen Tengah 3:  $f(0.75) = e^{-0.5625} = 0.569783$
  - Jumlah Tengah:  $0.939413 + 0.778801 + 0.569783 = 2.287997$
- **Hitung Integral ( $I_4$ ):**

$$I_4 \approx \frac{0.25}{2} [1.367879 + 2(2.287997)]$$

$$I_4 \approx 0.125 \times [1.367879 + 4.575994]$$

$$I_4 \approx 0.125 \times 5.943873 = 0.742984$$

- **Estimasi Error:**  $|0.742984 - 0.731370| = 0.011614$



# Numerical Integration

## Trapezoidal Rule

```
# --- DEFINISI FUNGSI ---
f1 <- function(x) { return(x^2) }
f2 <- function(x) { return(exp(-x^2)) }
# --- FUNGSI TRAPEZOIDAL DASAR ---
hitung_trapezoidal <- function(f, a, b, n) {
  h <- (b - a) / n
  x <- seq(a, b, by = h)
  y <- f(x)
  sum_tengah <- if(n==1) 0 else sum(y[2:n])
  integral <- (h / 2) * (y[1] + 2 * sum_tengah + y[n + 1])
  return(integral)
}
# --- ALGORITMA PENCARIAN N OPTIMUM ---
cari_n_optimal <- function(f, a, b, toleransi = 1e-6) {
  cat(sprintf("Mencari n optimum (Target Perubahan <
%g)...\\n", toleransi))
  cat(sprintf("%-8s %-15s %-15s\\n", "n", "Integral",
"Perubahan (Error)"))
  cat("-----\\n")
```

```
# Langkah Awal (n=1)
n <- 1
integral_lama <- hitung_trapezoidal(f, a, b, n)
  cat(sprintf("%-8d %-15.6f %-15s\\n", n,
integral_lama, "-"))

max_iter <- 20
for(i in 1:max_iter) {

# Lipat gandakan segmen (n -> 2n)
n_baru <- n * 2
integral_baru <- hitung_trapezoidal(f, a, b, n_baru)
  # Hitung Estimasi Error (Selisih dengan hasil
sebelumnya)
perubahan <- abs(integral_baru - integral_lama)
  cat(sprintf("%-8d %-15.6f %-15.6f\\n", n_baru,
integral_baru, perubahan))
```



# Numerical Integration

## Trapezoidal Rule

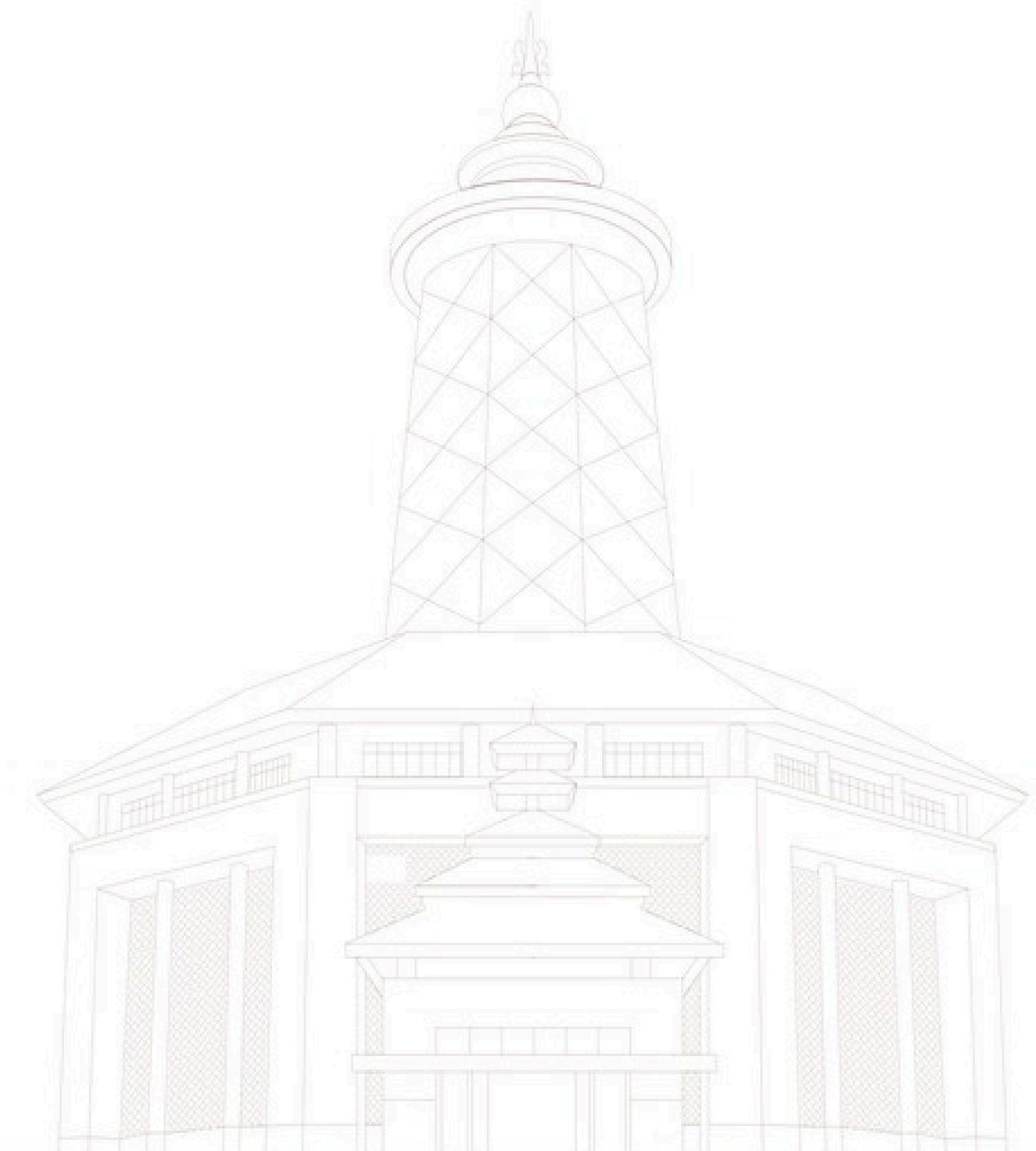
```
# Cek Konvergensi
if (perubahan < toleransi) {
  cat("-----\n")
  cat("KONVERGEN! Hasil stabil pada n =", n_baru, "\n")
  cat("Hasil Akhir Integral =", integral_baru, "\n\n")
  return(integral_baru)
}

# Update untuk putaran berikutnya
n <- n_baru
integral_lama <- integral_baru
}

#
# --- EKSEKUSI ---

cat("==> CONTOH 1: Polinomial x^2 [0, 2] ==>\n")
hasil1 <- cari_n_optimal(f1, 0, 2, toleransi=1e-5)

cat("==> CONTOH 2: Gaussian e^(-x^2) [0, 1] ==>\n")
hasil2 <- cari_n_optimal(f2, 0, 1, toleransi=1e-6)
```





# Numerical Integration

## Trapezoidal Rule

Mencari n optimum (Target Perubahan < 1e-05)...

| n    | Integral | Perubahan (Error) |
|------|----------|-------------------|
| 1    | 4.000000 | -                 |
| 2    | 3.000000 | 1.000000          |
| 4    | 2.750000 | 0.250000          |
| 8    | 2.687500 | 0.062500          |
| 16   | 2.671875 | 0.015625          |
| 32   | 2.667969 | 0.003906          |
| 64   | 2.666992 | 0.000977          |
| 128  | 2.666748 | 0.000244          |
| 256  | 2.666687 | 0.000061          |
| 512  | 2.666672 | 0.000015          |
| 1024 | 2.666668 | 0.000004          |

KONVERGEN! Hasil stabil pada n = 1024

Hasil Akhir Integral = 2.666668

Mencari n optimum (Target Perubahan < 1e-06)...

| n   | Integral | Perubahan (Error) |
|-----|----------|-------------------|
| 1   | 0.683940 | -                 |
| 2   | 0.731370 | 0.047431          |
| 4   | 0.742984 | 0.011614          |
| 8   | 0.745866 | 0.002882          |
| 16  | 0.746585 | 0.000719          |
| 32  | 0.746764 | 0.000180          |
| 64  | 0.746809 | 0.000045          |
| 128 | 0.746820 | 0.000011          |
| 256 | 0.746823 | 0.000003          |
| 512 | 0.746824 | 0.000001          |

KONVERGEN! Hasil stabil pada n = 512

Hasil Akhir Integral = 0.7468239



# Numerical Integration

## Simpson's 1/3 Rule

Aturan Simpson 1/3 adalah metode integrasi numerik yang menghampiri luas daerah di bawah kurva menggunakan **polinomial kuadrat (parabola)**. Metode ini umumnya memberikan akurasi yang jauh lebih tinggi dibandingkan Aturan Trapezoidal untuk jumlah segmen yang sama.

Syarat utama metode ini adalah jumlah segmen ( $n$ ) harus berupa **bilangan genap**.

Rumus Lebar Segmen ( $h$ ):

$$h = \frac{b - a}{n}$$

Rumus Integral Simpson 1/3:

$$I \approx \frac{h}{3} \left[ f(x_0) + 4 \sum f(x_{ganjil}) + 2 \sum f(x_{genap}) + f(x_n) \right]$$



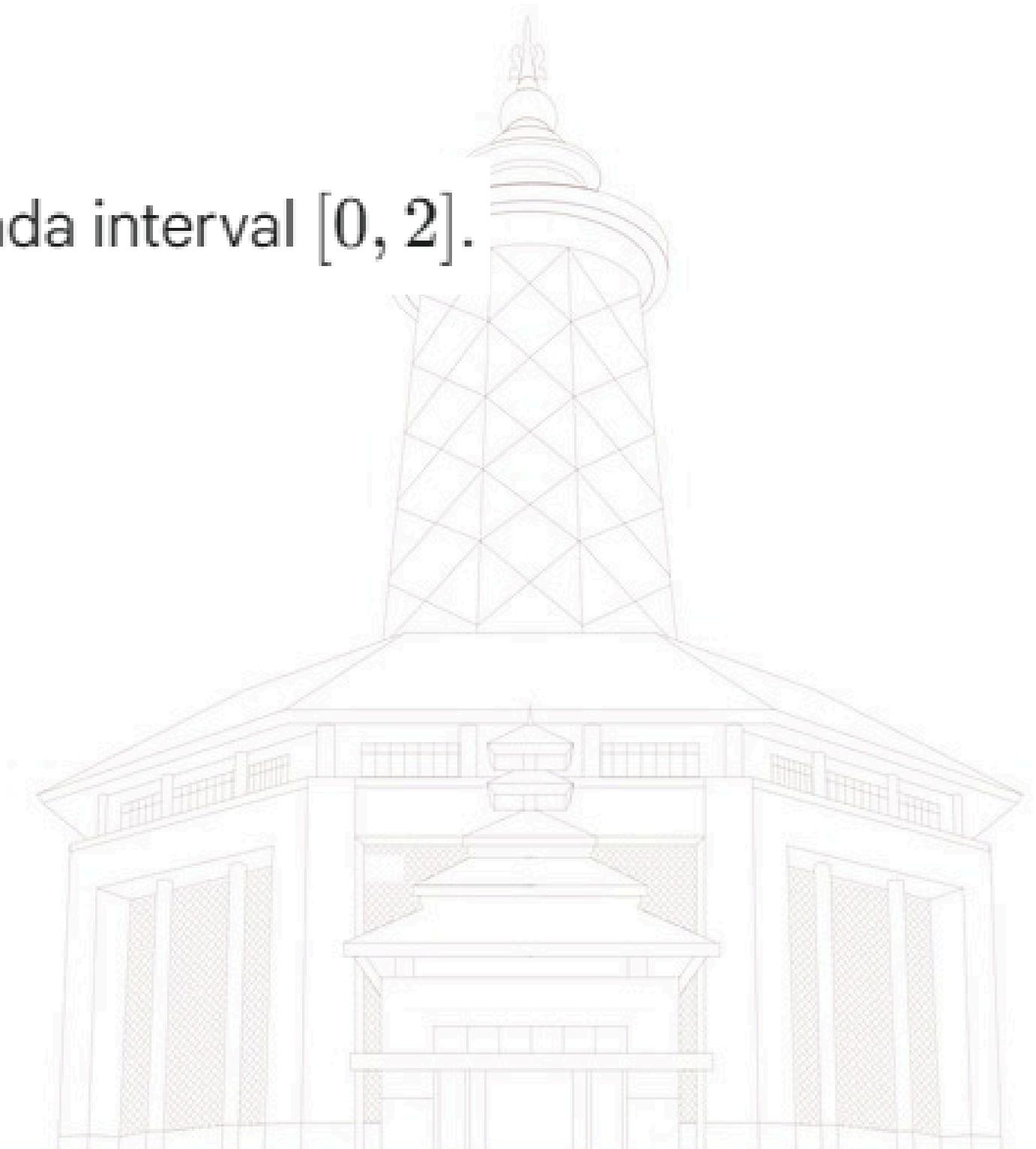
# Numerical Integration

## Simpson's 1/3 Rule (Contoh 1)

**Kasus:** Hitung integral fungsi berikut pada interval  $[0, 2]$ .

$$f(x) = x^2$$

Nilai Eksak (Aljabar): 2.666667



# Numerical Integration

## Simpson's 1/3 Rule (Contoh 2)

**Iterasi 1: Menggunakan 2 Segmen ( $n = 2$ ) Syarat  $n$  genap terpenuhi.**

- Lebar segmen ( $h$ ):  $(2 - 0)/2 = 1.0$

**• Penentuan Titik Grid:**

- $x_0 = 0.0$  (Awal)
- $x_1 = 1.0$  (Ganjil)
- $x_2 = 2.0$  (Akhir)

**• Evaluasi Fungsi:**

- Ujung:  $f(0) = 0, f(2) = 4$
- Ganjil ( $x_1$ ):  $f(1) = 1^2 = 1$

**• Hitung Integral ( $I_2$ ):**

$$I_2 \approx \frac{1.0}{3}[f(0) + 4(f(1)) + f(2)]$$

$$I_2 \approx \frac{1}{3}[0 + 4(1) + 4] = \frac{1}{3}[8] = 2.666667$$

**• Estimasi Error:** -

**Iterasi 2: Menggunakan 4 Segmen ( $n = 4$ )**

- Lebar segmen ( $h$ ):  $(2 - 0)/4 = 0.5$

**• Penentuan Titik Grid:**

- $x_0 = 0.0$  (Awal)
- $x_1 = 0.5$  (Ganjil)
- $x_2 = 1.0$  (Genap)
- $x_3 = 1.5$  (Ganjil)
- $x_4 = 2.0$  (Akhir)

**• Evaluasi Fungsi:**

- Ujung:  $f(0) = 0, f(2) = 4$
- Ganjil ( $x_1, x_3$ ):  $f(0.5) = 0.25, f(1.5) = 2.25$

**• Hitung Komponen:**

- $\sum f_{ganjil} = 0.25 + 2.25 = 2.50$
- $\sum f_{genap} = 1.00$

**• Hitung Integral ( $I_4$ ):**

$$I_4 \approx \frac{0.5}{3}[0 + 4(2.50) + 2(1.00) + 4]$$

$$I_4 \approx 0.166667 \times 16 = 2.666667$$

**• Estimasi Error:**  $|2.666667 - 2.666667| = 0.000000$





# Numerical Integration

## Simpson's 1/3 Rule (Contoh 2)

**Kasus:** Hitung integral fungsi berikut pada interval  $[0, 2]$ .

$$f(x) = e^{-x^2}$$

**Nilai Referensi:** 0.746824



# Numerical Integration

## Simpson's 1/3 Rule (Contoh 2)

### Iterasi 1: Menggunakan 2 Segmen ( $n = 2$ )

- Lebar segmen ( $h$ ):  $(1 - 0)/2 = 0.5$
- Titik Grid: 0.0, 0.5, 1.0
- Evaluasi Fungsi:
  - Ujung:  $f(0) = 1.000000$ ,  $f(1) = 0.367879$
  - Ganjil ( $x_1 = 0.5$ ):  $f(0.5) = 0.778801$
- Hitung Integral ( $I_2$ ):

$$I_2 \approx \frac{0.5}{3}[1.000000 + 4(0.778801) + 0.367879]$$

$$I_2 \approx 0.166667 \times 4.483083 = 0.747180$$

- Estimasi Error: -

### Iterasi 2: Menggunakan 4 Segmen ( $n = 4$ )

- Lebar segmen ( $h$ ):  $(1 - 0)/4 = 0.25$
- Penentuan Titik Grid:
  - Awal:  $x_0 = 0.0$
  - Ganjil:  $x_1 = 0.25$ ,  $x_3 = 0.75$
  - Genap:  $x_2 = 0.50$
  - Akhir:  $x_4 = 1.0$
- Evaluasi Fungsi ( $f(x) = e^{-x^2}$ ):
  - Ujung:  $1.000000 + 0.367879 = 1.367879$
  - Ganjil 1 ( $x_1$ ):  $e^{-0.0625} = 0.939413$
  - Ganjil 2 ( $x_3$ ):  $e^{-0.5625} = 0.569783$
  - Genap 1 ( $x_2$ ):  $e^{-0.2500} = 0.778801$

### Komponen Rumus:

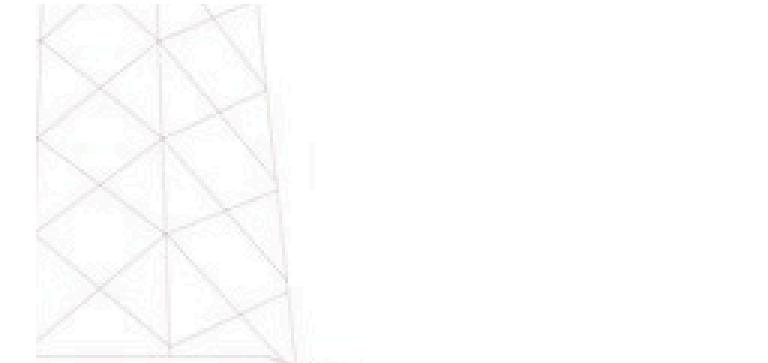
- Sum Ganjil ( $f(x_1) + f(x_3)$ ) = 1.509196
- Sum Genap ( $f(x_2)$ ) = 0.778801

### Hitung Integral ( $I_4$ ):

$$I_4 \approx \frac{0.25}{3}[1.367879 + 4(1.509196) + 2(0.778801)]$$

$$I_4 \approx 0.083333 \times 8.962265 = 0.746855$$

### Estimasi Error: $|0.746855 - 0.747180| = 0.000325$





# Numerical Integration

## Simpson's 1/3 Rule (Contoh 2)

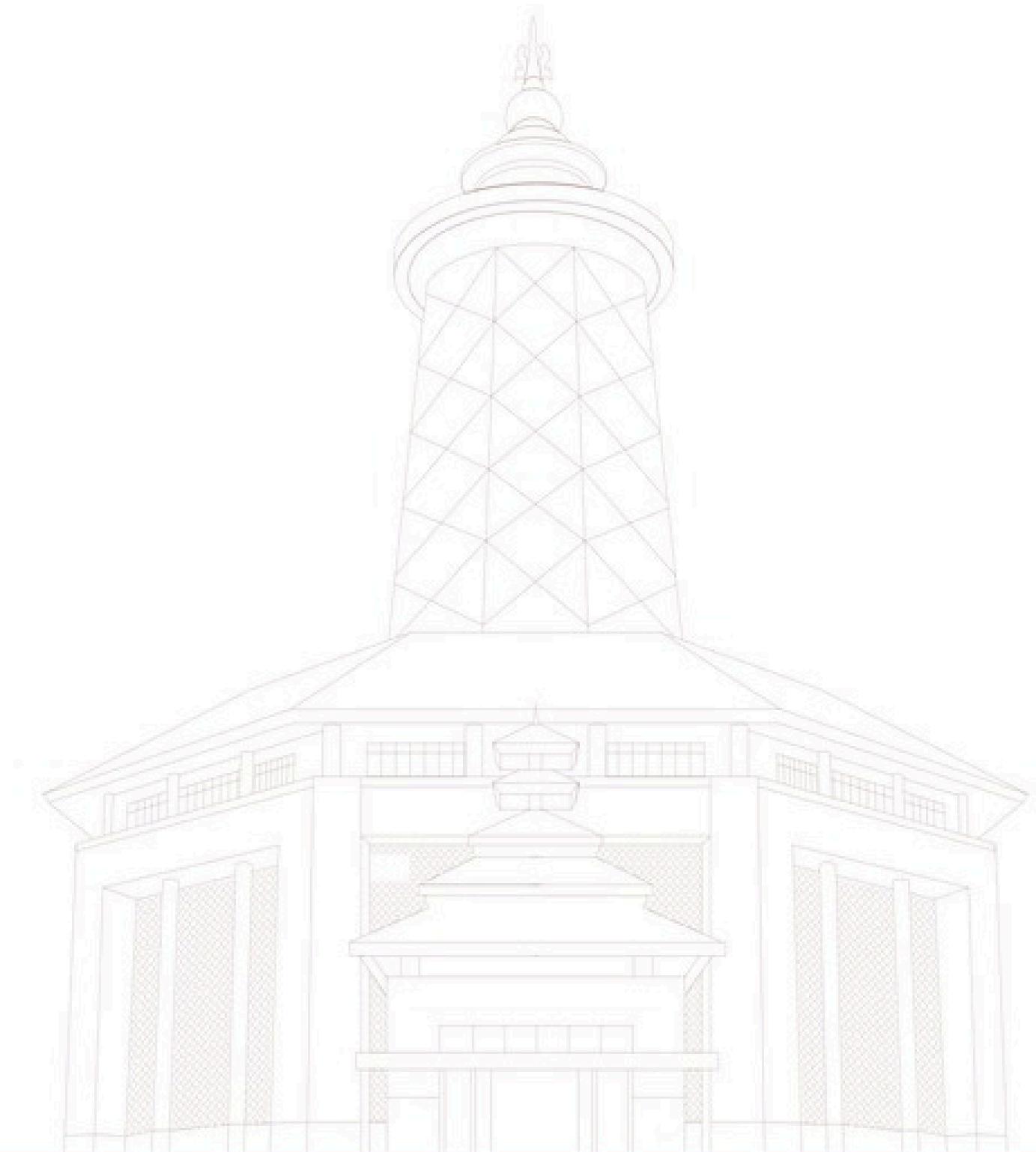
Iterasi 3: Menggunakan 8 Segmen ( $n = 8$ )

- Lebar segmen ( $h$ ): 0.125
- **Komponen Hasil Evaluasi (Ringkasan):**
  - Ujung: 1.367879
  - $4 \times$  Sum Ganjil: 11.959148
  - $2 \times$  Sum Genap: 4.594009
- **Hitung Integral ( $I_8$ ):**

$$I_8 \approx \frac{0.125}{3} [1.367879 + 11.959148 + 4.594009]$$

$$I_8 \approx 0.041667 \times 17.921036 = 0.746826$$

- **Estimasi Error:**  $|0.746826 - 0.746855| = 0.000029$



# Numerical Integration

## Simpson's 1/3 Rule

```
# --- DEFINISI FUNGSI ---
```

```
f1 <- function(x) { return(x^2) }
```

```
f2 <- function(x) { return(exp(-x^2)) }
```

```
# --- FUNGSI SIMPSON 1/3 (PERBAIKAN) ---
```

```
hitung_simpson <- function(f, a, b, n) {
```

```
  # Validasi n harus genap
```

```
  if (n %% 2 != 0) stop("Metode Simpson 1/3
```

```
mewajibkan n genap!")
```

```
h <- (b - a) / n
```

```
x <- seq(a, b, by = h)
```

```
y <- f(x)
```

```
# 1. Hitung Jumlah Bagian Ganjil (x1, x3, x5...)
```

```
  # Dalam R index dimulai dari 1, jadi x0=index1,
```

```
x1=index2
```

```
  # Maka x1, x3, x5... ada di index 2, 4, 6...
```

```
idx_ganjil <- seq(2, n, by=2)
```

```
sum_ganjil <- sum(y[idx_ganjil])
```

```
# 2. Hitung Jumlah Bagian Genap (x2, x4, x6...)
```

```
# Index R: 3, 5, 7...
```

```
  # Jika n=2, tidak ada index genap di tengah  
(langsung ujung).
```

```
sum_genap <- 0
```

```
if (n > 2) {
```

```
  idx_genap <- seq(3, n-1, by=2)
```

```
  sum_genap <- sum(y[idx_genap])
```

```
}
```

```
# Rumus: h/3 * (Ujung + 4*Ganjil + 2*Genap)
```

```
integral <- (h / 3) * (y[1] + 4 * sum_ganjil + 2 *
```

```
sum_genap + y[n + 1])
```

```
return(integral)
```

```
}
```



# Numerical Integration

## Simpson's 1/3 Rule

```
# --- ALGORITMA PENCARIAN N OPTIMUM ---
cari_n_optimal_simpson <- function(f, a, b, toleransi =
1e-6){

  cat(sprintf("Mencari n optimum (Target Perubahan <
%g)...\\n", toleransi))
  cat(sprintf("%-8s %-15s %-15s\\n", "n", "Integral",
"Perubahan (Error)"))
  cat("-----\\n")

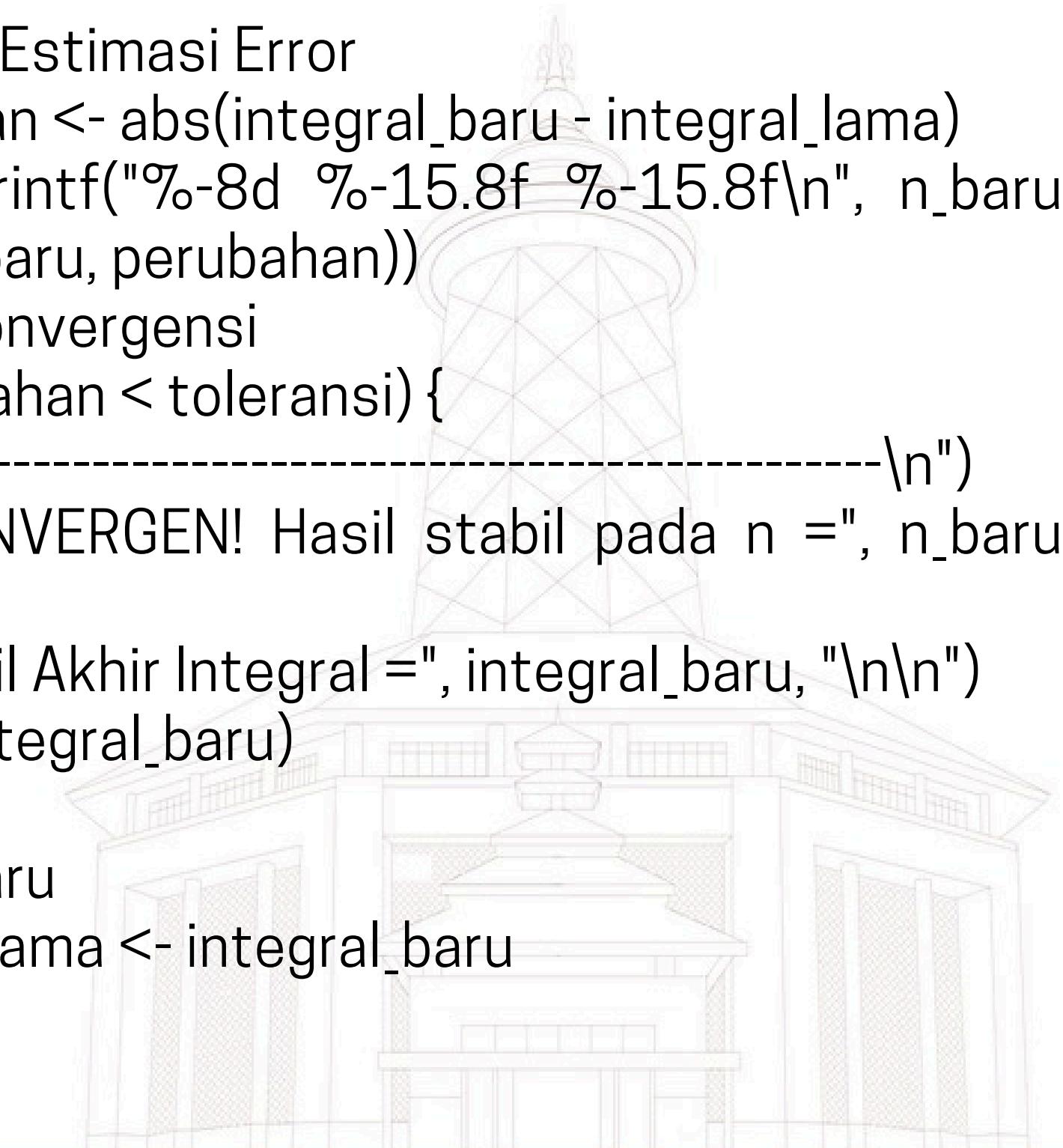
  # Langkah Awal (n=2)
  n <- 2
  integral_lama <- hitung_simpson(f, a, b, n)
  cat(sprintf("%-8d %-15.8f %-15s\\n", n, integral_lama,
"-"))

  max_iter <- 20
  for(i in 1:max_iter){
    # Lipat gandakan segmen (2 -> 4 -> 8...)
    n_baru <- n * 2
    integral_baru <- hitung_simpson(f, a, b, n_baru)
```

```
# Hitung Estimasi Error
perubahan <- abs(integral_baru - integral_lama)
cat(sprintf("%-8d %-15.8f %-15.8f\\n", n_baru,
integral_baru, perubahan))

# Cek Konvergensi
if (perubahan < toleransi) {
  cat("-----\\n")
  cat("KONVERGEN! Hasil stabil pada n =", n_baru,
"\n")
  cat("Hasil Akhir Integral =", integral_baru, "\\n\\n")
  return(integral_baru)
}

n <- n_baru
integral_lama <- integral_baru
}
```





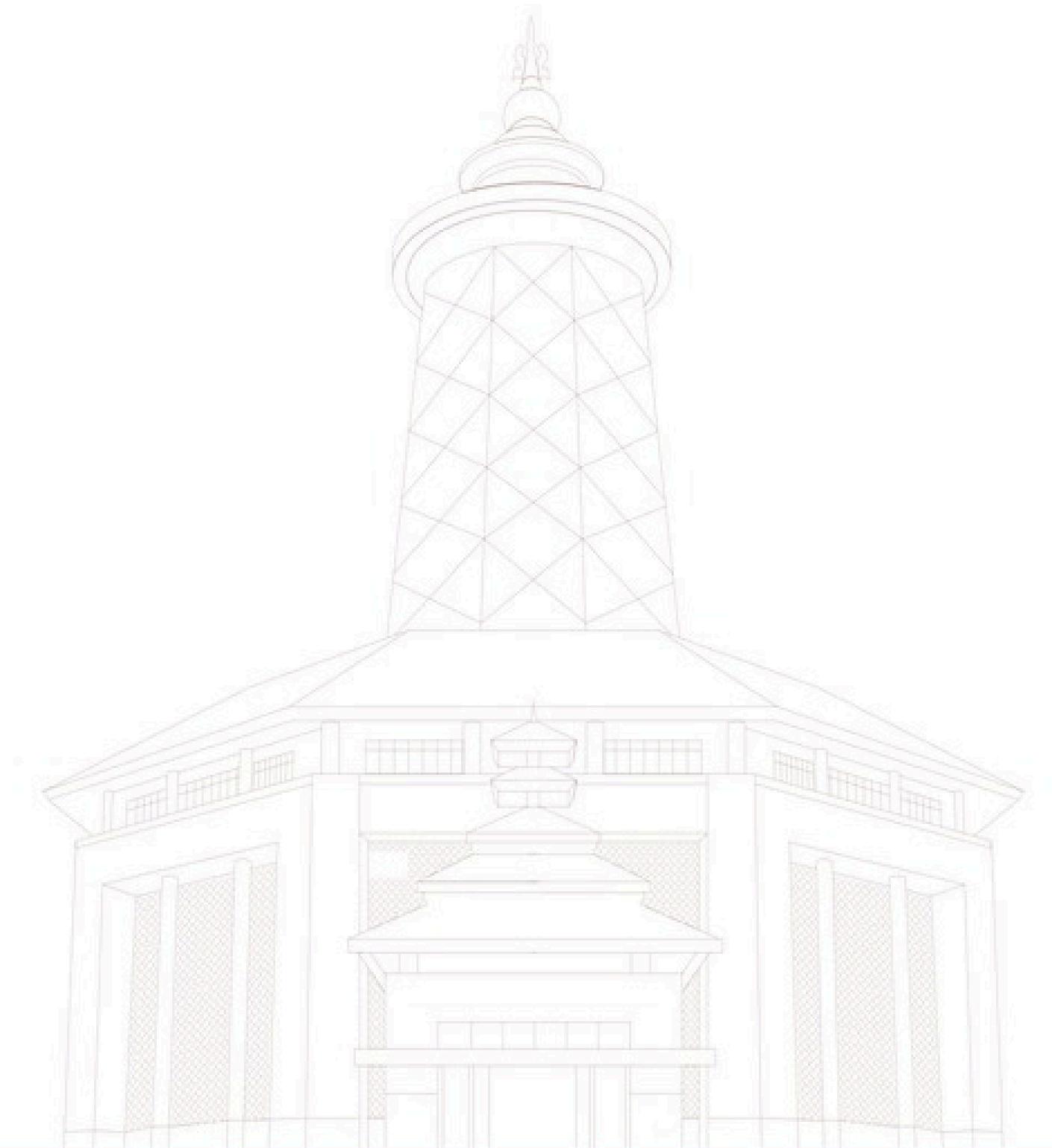
# Numerical Integration

## Simpson's 1/3 Rule

# --- EKSEKUSI ---

```
cat("==> CONTOH 1: Polinomial x^2 [0, 2] ==>\n")
hasil1 <- cari_n_optimal_simpson(f1, 0, 2, toleransi=1e-
8)
```

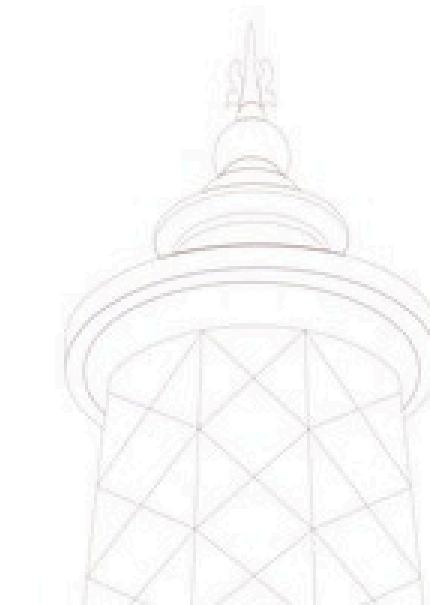
```
cat("\n==> CONTOH 2: Gaussian e^(-x^2) [0, 1] ==>\n")
hasil2 <- cari_n_optimal_simpson(f2, 0, 1, toleransi=1e-
6)
```



# Numerical Integration

## Simpson's 1/3 Rule

```
> hasil1 <- cari_n_optimal_simpson(f1, 0, 2, toleransi=1e-8)
Mencari n optimum (Target Perubahan < 1e-08)...
n      Integral      Perubahan (Error)
-----
2      2.66666667      -
4      2.66666667      0.00000000
-----
KONVERGEN! Hasil stabil pada n = 4
Hasil Akhir Integral = 2.666667
```



```
> hasil2 <- cari_n_optimal_simpson(f2, 0, 1, toleransi=1e-6)
Mencari n optimum (Target Perubahan < 1e-06)...
n      Integral      Perubahan (Error)
-----
2      0.74718043      -
4      0.74685538      0.00032505
8      0.74682612      0.00002926
16     0.74682426      0.00000186
32     0.74682414      0.00000012
-----
KONVERGEN! Hasil stabil pada n = 32
Hasil Akhir Integral = 0.7468241
```



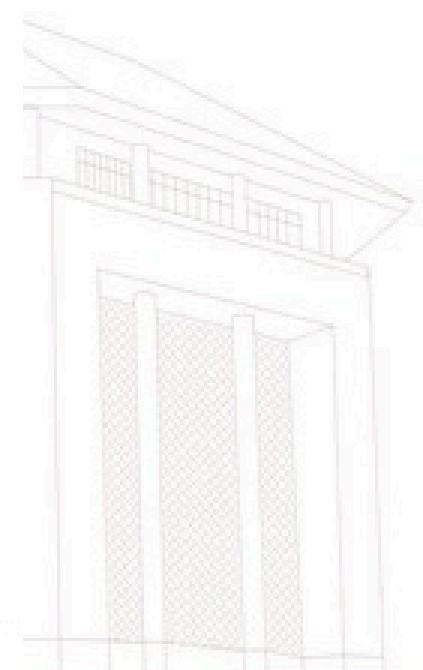
# Numerical Integration

## Adaptive Quadrature - Simpson

Integrasi Adaptif adalah strategi untuk mengefisienkan perhitungan integral numerik. Pada metode konvensional (seperti Simpson dengan  $n$  tetap), lebar segmen ( $h$ ) dipukul rata untuk seluruh interval. Hal ini tidak efisien jika fungsi memiliki perilaku yang berbeda-beda (ada bagian datar, ada bagian yang fluktuatif tajam).

Prinsip kerja Integrasi Adaptif (berbasis Simpson) adalah "**Divide and Conquer**" (Bagi dan Taklukkan):

1. Hitung integral pada interval  $[a, b]$  (Sebut saja  $S_1$ ).
2. Bagi interval menjadi dua:  $[a, c]$  dan  $[c, b]$ . Hitung integral masing-masing dan jumlahkan (Sebut saja  $S_2$ ).
3. **Cek Error:** Bandingkan  $S_1$  dan  $S_2$ .
  - Jika perbedaannya kecil (di bawah toleransi), maka  $S_2$  diterima sebagai hasil.
  - Jika perbedaannya besar, maka interval dibagi lagi (rekursif) hanya pada bagian yang bermasalah.





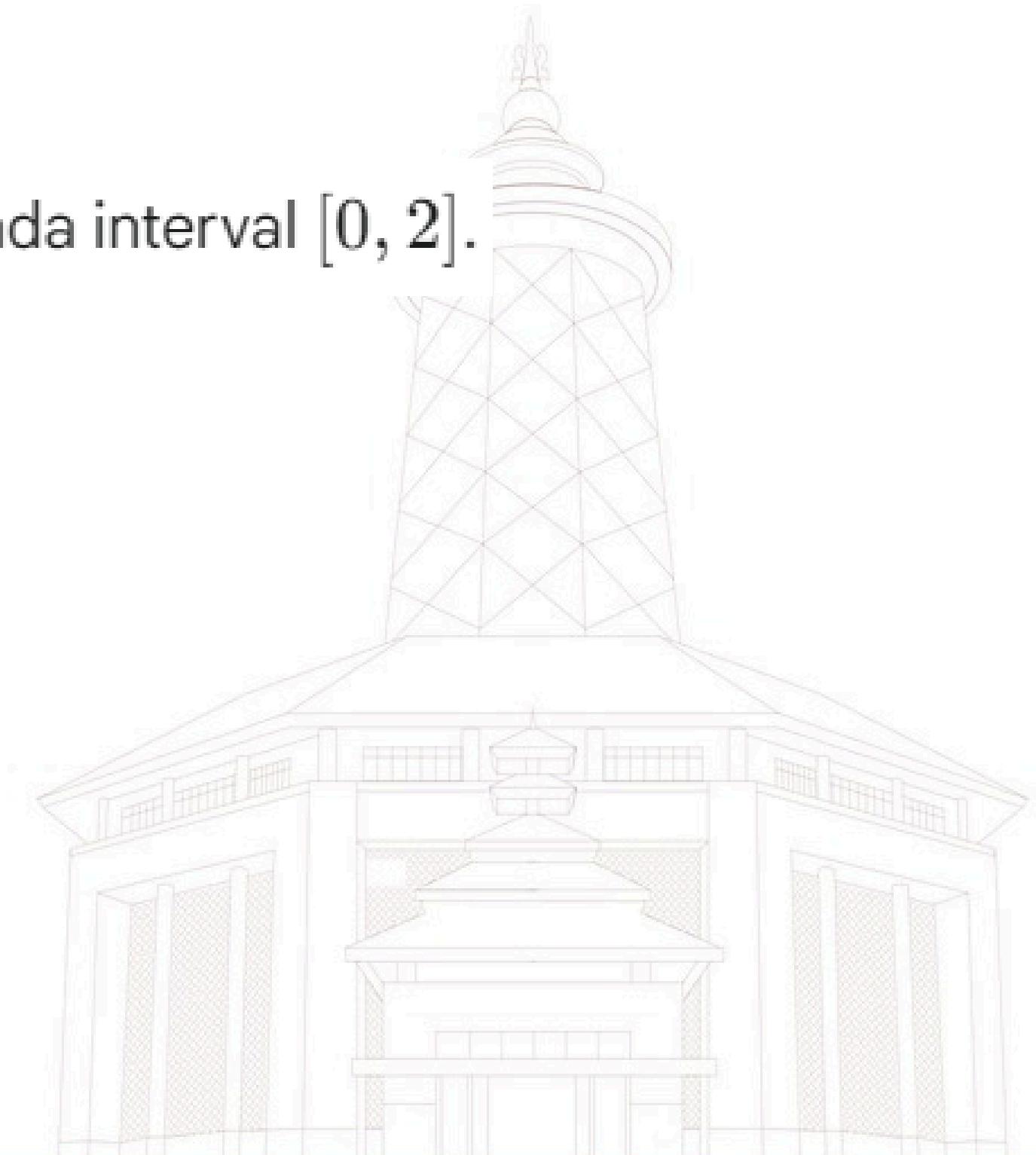
# Numerical Integration

## Adaptive Quadrature - Simpson (Contoh 1)

**Kasus:** Hitung integral fungsi berikut pada interval  $[0, 2]$ .

$$f(x) = x^2$$

Nilai Eksak (Aljabar): 2.666667

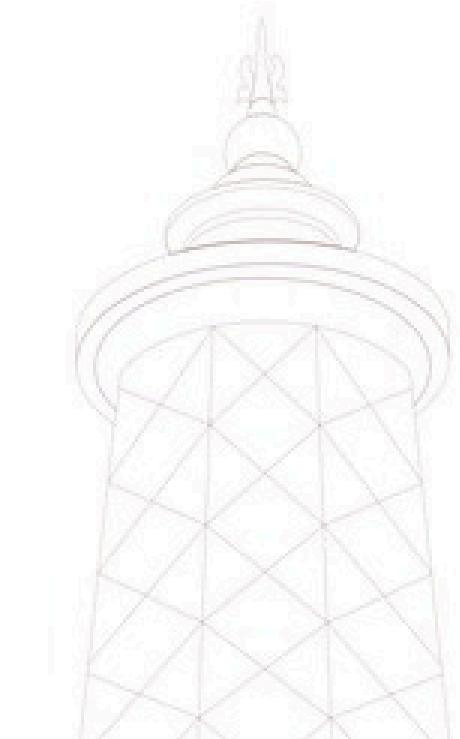


# Numerical Integration

## Adaptive Quadrature - Simpson (Contoh 1)

Langkah 1: Estimasi Kasar ( $S_1$ ) pada Interval  $[0, 2]$  Menggunakan Aturan Simpson satu langkah ( $h = 1.0$ ). Titik evaluasi:  $f(0)=0$ ,  $f(1)=1$ ,  $f(2)=4$ .

$$S_1 = \frac{1.0}{3}[0 + 4(1) + 4] = 2.666667$$



Langkah 2: Estimasi Halus ( $S_2$ ) pada Sub-Interval Interval dibagi dua menjadi  $[0, 1]$  dan  $[1, 2]$  dengan  $h = 0.5$ .

Sub-interval Kiri  $[0, 1]$ : Titik evaluasi:  $f(0)=0$ ,  $f(0.5)=0.25$ ,  $f(1)=1$ .

$$S_{kiri} = \frac{0.5}{3}[0 + 4(0.25) + 1] = 0.333333$$

Sub-interval Kanan  $[1, 2]$ : Titik evaluasi:  $f(1)=1$ ,  $f(1.5)=2.25$ ,  $f(2)=4$ .

$$S_{kanan} = \frac{0.5}{3}[1 + 4(2.25) + 4] = 2.333333$$

$$\text{Total } S_2 = 0.333333 + 2.333333 = 2.666667$$

Langkah 3: Evaluasi Error

$$\text{Error} = \frac{1}{15}|2.666667 - 2.666667| = 0$$

Keputusan: Error < Toleransi. Iterasi berhenti.





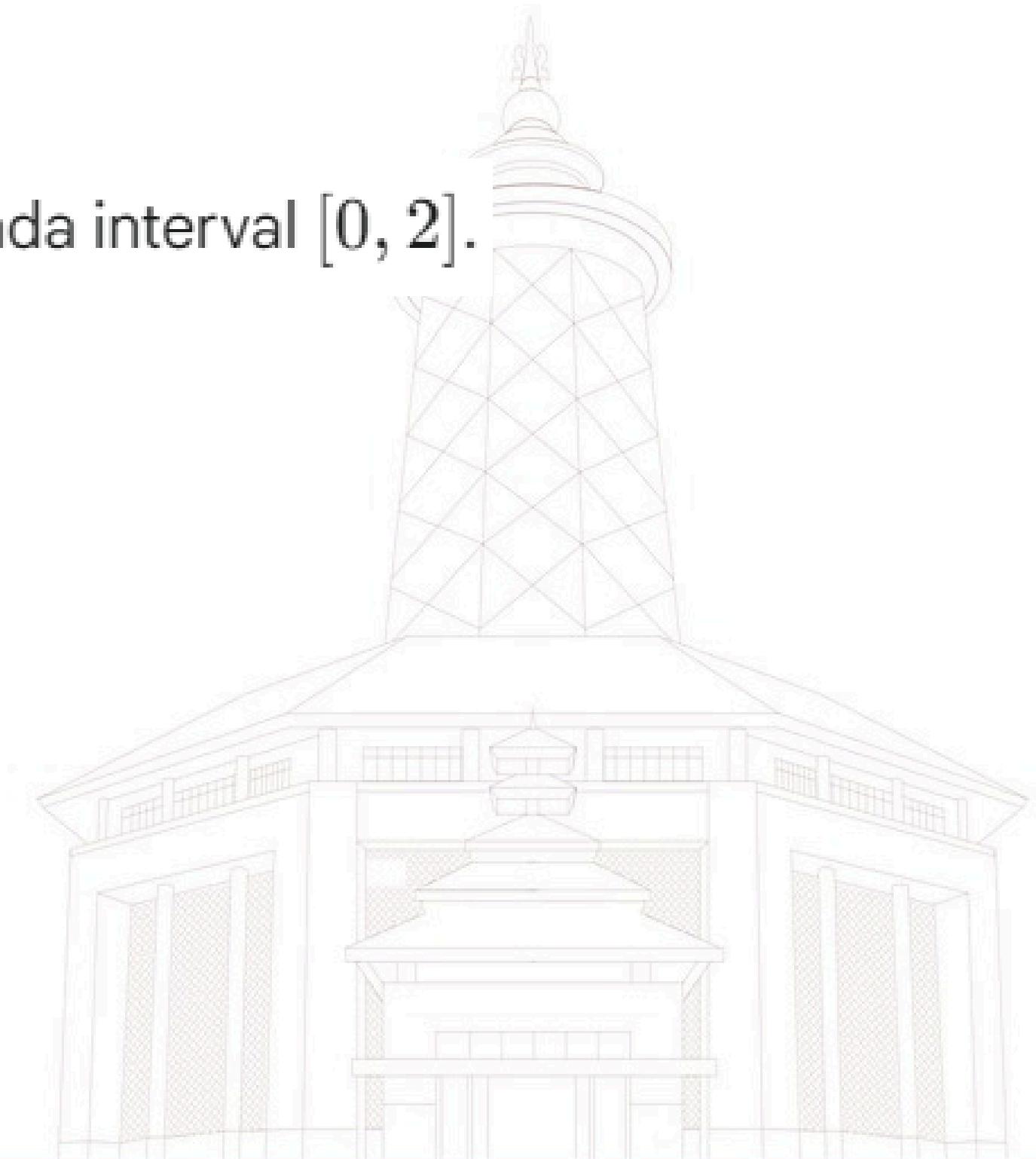
# Numerical Integration

## Adaptive Quadrature - Simpson (Contoh 2)

**Kasus:** Hitung integral fungsi berikut pada interval  $[0, 2]$ .

$$f(x) = e^{-x^2}$$

**Nilai Referensi:** 0.746824



# Numerical Integration

## Adaptive Quadrature - Simpson (Contoh 2)

Langkah 1: Estimasi Kasar ( $S_1$ ) pada Interval  $[0, 1]$  Menggunakan Simpson dengan  $h = 0.5$ . Titik evaluasi:  $f(0)=1$ ,  $f(0.5)=0.778801$ ,  $f(1)=0.367879$ .

$$S_1 = \frac{0.5}{3}[1 + 4(0.778801) + 0.367879] = 0.747180$$

Langkah 2: Estimasi Halus ( $S_2$ ) pada Sub-Interval Interval dibagi menjadi  $[0, 0.5]$  dan  $[0.5, 1.0]$  dengan  $h = 0.25$ .

Sub-interval Kiri  $[0, 0.5]$ : Titik evaluasi:  $f(0)=1$ ,  $f(0.25)=0.939413$ ,  $f(0.5)=0.778801$ .

$$S_{kiri} = \frac{0.25}{3}[1 + 4(0.939413) + 0.778801] = 0.461133$$

Sub-interval Kanan  $[0.5, 1.0]$ : Titik evaluasi:  $f(0.5)=0.778801$ ,  $f(0.75)=0.569783$ ,  $f(1)=0.367879$ .

$$S_{kanan} = \frac{0.25}{3}[0.778801 + 4(0.569783) + 0.367879] = 0.285722$$

$$\text{Total } S_2 = 0.461133 + 0.285722 = 0.746855$$

Langkah 3: Evaluasi Error Selisih =  $|0.746855 - 0.747180| = 0.000325$

$$\text{Error} = \frac{1}{15}(0.000325) = 0.000021$$

Akurasi belum tercapai

pembagian interval secara rekursif



# Numerical Integration

## Adaptive Quadrature - Simpson

```
# Definisi Fungsi Target
f1 <- function(x) { return(x^2) }
f2 <- function(x) { return(exp(-x^2)) }

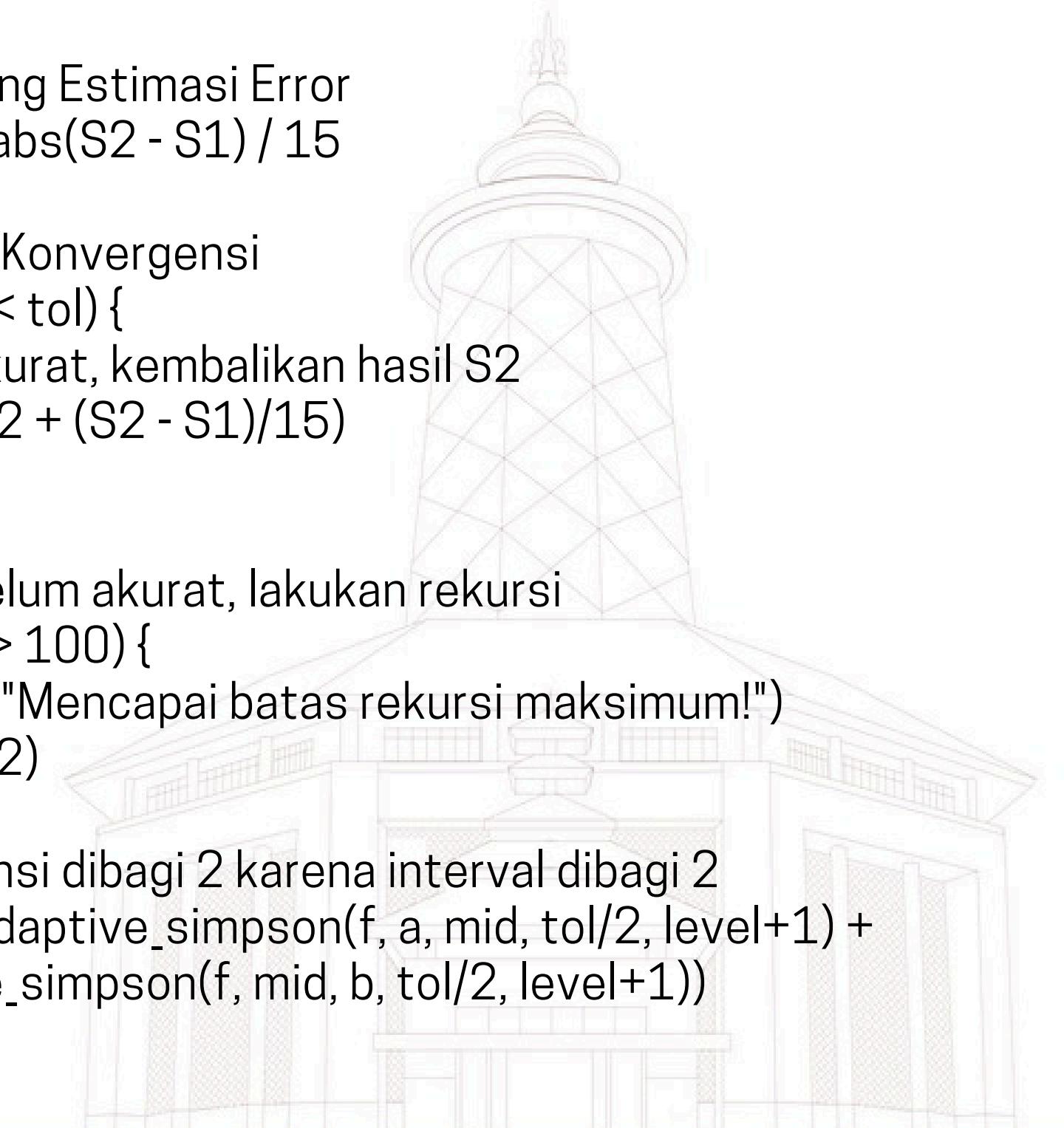
# Fungsi Dasar Simpson (3 Titik)
basic_simpson <- function(f, a, b) {
  c <- (a + b) / 2
  h <- (b - a) / 2
  return( (h/3) * (f(a) + 4*f(c) + f(b)) )
}

# Algoritma Adaptive Quadrature (Rekursif)
adaptive_simpson <- function(f, a, b, tol = 1e-6, level =
1) {
  # 1. Hitung S1 (Kasar - Seluruh Interval)
  S1 <- basic_simpson(f, a, b)

  # 2. Hitung S2 (Halus - Interval Dibagi Dua)
  mid <- (a + b) / 2
  S_left <- basic_simpson(f, a, mid)
  S_right <- basic_simpson(f, mid, b)
  S2 <- S_left + S_right

  # 3. Hitung Estimasi Error
  error <- abs(S2 - S1) / 15

  # 4. Cek Konvergensi
  if (error < tol) {
    # Jika akurat, kembalikan hasil S2
    return(S2 + (S2 - S1)/15)
  } else {
    # Jika belum akurat, lakukan rekursi
    if (level > 100) {
      warning("Mencapai batas rekursi maksimum!")
      return(S2)
    }
    # Toleransi dibagi 2 karena interval dibagi 2
    return(adaptive_simpson(f, a, mid, tol/2, level+1) +
adaptive_simpson(f, mid, b, tol/2, level+1))
  }
}
```





# Numerical Integration

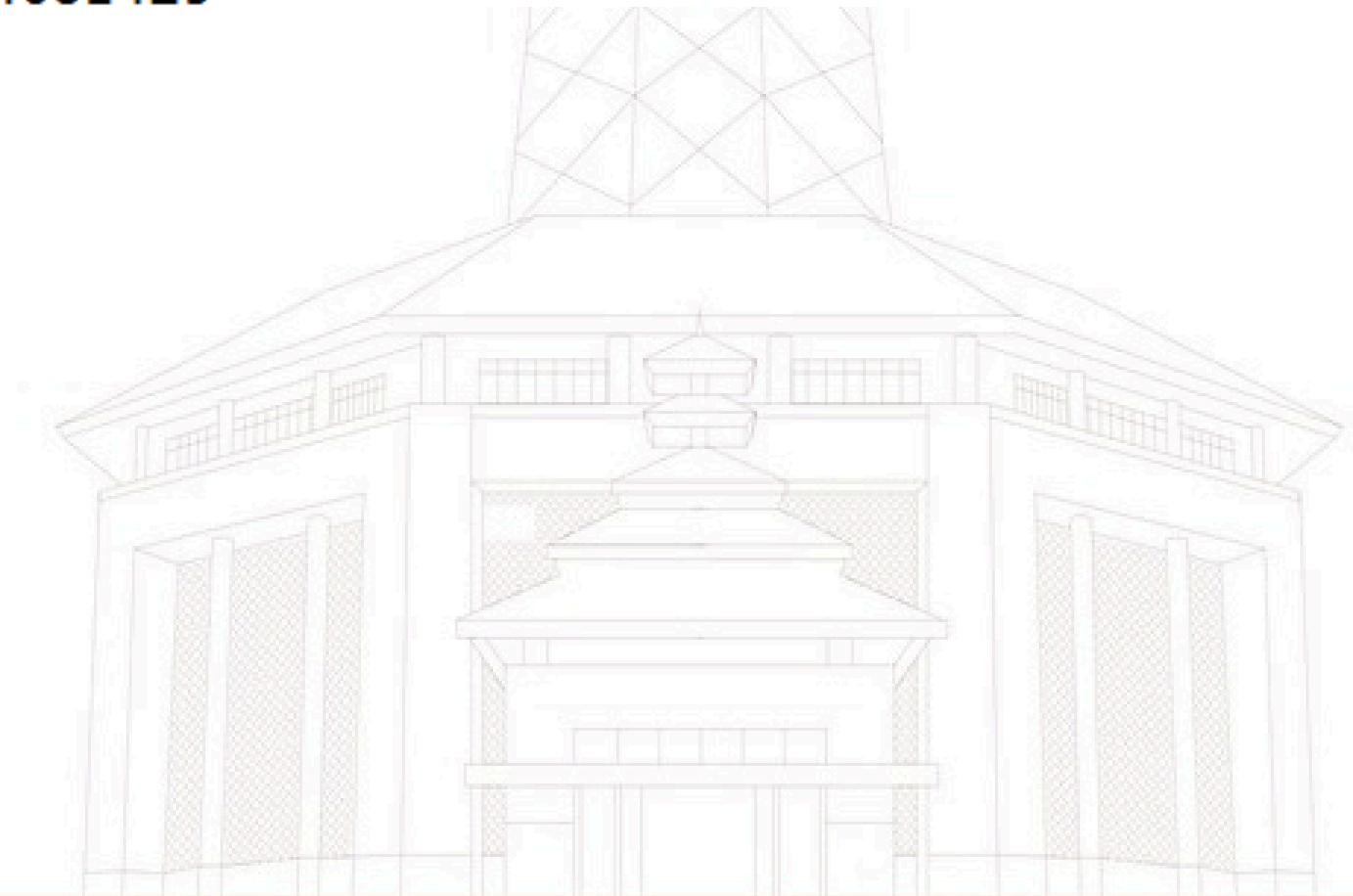
## Adaptive Quadrature - Simpson

```
# Eksekusi Kasus
cat("==> CONTOH 1: Polinomial x^2 [0, 2] ==>\n")
hasil1 <- adaptive_simpson(f1, 0, 2, tol=1e-6)
print(hasil1)

cat("\n==> CONTOH 2: Gaussian e^(-x^2) [0, 1] ==>\n")
hasil2 <- adaptive_simpson(f2, 0, 1, tol=1e-8)
print(hasil2, digits=8)
```

```
==> CONTOH 1: Polinomial x^2 [0, 2] ==
> hasil1 <- adaptive_simpson(f1, 0, 2, tol=1e-6)
> print(hasil1)
[1] 2.666667

==> CONTOH 2: Gaussian e^(-x^2) [0, 1] ==
> hasil2 <- adaptive_simpson(f2, 0, 1, tol=1e-8)
> print(hasil2, digits=8)
[1] 0.74682413
```





# SEE YOU NEXT WEEK !

Ferdian Bangkit Wijaya, S.Stat., M.Si

NIP. 199005202024061001

[ferdian.bangkit@untirta.ac.id](mailto:ferdian.bangkit@untirta.ac.id)

