

Test NUM

Contents

1	Exact Analytical Value	2
2	Midpoint Rule Method	2
3	Error Computation	3
4	Visualizing Error Behavior	3
5	Least Squares Approximation of the Error Function	4
6	Computing and Interpreting the Coefficients	5
7	Evaluation and Plotting of the Approximation	5

1 Exact Analytical Value

The exact value of the integral is known and serves as a benchmark for evaluating the numerical results:

$$I_{\text{exact}} = \ln(2)$$

In R, this is computed using:

```
1 I_analyticke <- log(2)
```

Listing 1: Exact value of the integral

2 Midpoint Rule Method

The midpoint rule approximates the integral of a function over an interval $[a, b]$ by dividing it into n equally sized subintervals of width $h = (b - a)/n$. For each subinterval, the function value at the midpoint is multiplied by the width:

$$I_N = \sum_{i=0}^{n-1} h \cdot f\left(a + ih + \frac{h}{2}\right)$$

This method is particularly effective for smooth, continuous functions and is simple to implement. In R:

```
1 f <- function(x) {  
2   1 / x  
3 }  
4  
5 midpoint_integral <- function(f, a, b, n) {  
6   h <- (b - a) / n  
7   I <- 0  
8  
9   for (i in 0:(n - 1)) {  
10    xi <- a + i * h + h / 2  
11    Si <- h * f(xi)  
12    I <- I + Si  
13  }  
14  return(I)  
15 }
```

Listing 2: Midpoint rule function in R

3 Error Computation

To evaluate how the midpoint approximation converges, we compute the error for increasing values of $n = 2^i$, with i ranging from 1 to 20. The error is computed as:

$$y(i) = I_N(i) - I_{\text{exact}}$$

This shows how the numerical result deviates from the true value.

```
1 i_val <- 1:20
2 y_val <- numeric(length(i_val))
3
4 for (j in seq_along(i_val)) {
5   i <- i_val[j]
6   n <- 2^i
7   I_numericke <- midpoint_integral(f, 1, 2, n)
8   y_val[j] <- I_numericke - I_analyticke
9   if (j>1 && abs(y_val[j]) > abs(y_val[j-1])) {
10     print(j)
11   }
12 }
```

Listing 3: Compute midpoint error for i

4 Visualizing Error Behavior

The error is plotted as a function of i . The error should generally decrease with increasing n :

```
1 plot(i_val, y_val, type = "b", pch = 19, col = "blue",
2       xlab = "i", ylab = expression(y(i) == I[N](i) - I[A
3       ]))
3 abline(h = 0, col = "red", lty = 2)
```

Listing 4: Plot the numerical error

5 Least Squares Approximation of the Error Function

We want to approximate the numerical error $y(i)$ using a linear combination of basis functions:

$$y(i) \approx c_0 \cdot \varphi_0(i) + c_1 \cdot \varphi_1(i) + c_2 \cdot \varphi_2(i) + c_3 \cdot \varphi_3(i)$$

where:

- $\varphi_0(i) = 1$
- $\varphi_1(i) = \frac{1}{2^i}$ (exponential decay)
- $\varphi_2(i) = \frac{1}{i}$ (harmonic decay)
- $\varphi_3(i) = \frac{1}{i^2}$ (polynomial decay)

We construct a design matrix A where $A[i, j] = \varphi_j(i)$, and solve the normal equations:

$$(A^T A) \vec{c} = A^T \vec{y}$$

```
1 least_squares_approx <- function(x, y, basis_functions)
  {
2   if (length(x) != length(y)) stop("x and y must have the
    same length")
3
4   m <- length(x)
5   n <- length(basis_functions)
6
7   A <- matrix(0, nrow = m, ncol = n)
8   for (j in 1:n) {
9     A[, j] <- basis_functions[[j]](x)
10  }
11  coeff <- solve(t(A) %*% A, t(A) %*% y)
12  return(as.vector(coeff))
13 }
```

Listing 5: Least squares approximation function

6 Computing and Interpreting the Coefficients

The coefficients represent the best-fit parameters for the chosen basis functions to approximate the error behavior.

```
1 poly_basis <- list(  
2   function(x) 1,  
3   function(x) 1 / (2^x),  
4   function(x) 1 / x,  
5   function(x) 1 / (x^2)  
6 )  
7  
8 coeff_poly <- least_squares_approx(i_val, y_val, poly_  
   basis)  
9 print(coeff_poly)
```

Listing 6: Define basis functions and compute coefficients

7 Evaluation and Plotting of the Approximation

The approximated function is evaluated at each point i :

```
1 aprox_y_val <- sapply(i_val, function(xi) {  
2   sum(mapply(function(c, phi) c * phi(xi), coeff_poly,  
   poly_basis))  
3 })  
4  
5 lines(i_val, aprox_y_val, col = "darkgreen", lwd = 2)
```

Listing 7: Evaluate and plot the approximation