Class Prep 9: 5.1.1 to 5.3.2

# Chapter 5: Differentiation and Integration

## Section 3.1.1: Finite Differences

library(cmna)  
library(pracma)

##   
## Attaching package: 'pracma'

## The following objects are masked from 'package:cmna':  
##   
## cubicspline, horner, newton, nthroot, romberg, secant, wilkinson

finddiff <- function(f, x, h = x\*sqrt(.Machine$double.eps)) {  
 return((f(x + h) - f(x)) / h)  
}  
  
f <- function(x) {3\*x - 1}  
  
finddiff(f, 4)

## [1] 3

finddiff(f, 4, h=1)

## [1] 3

finddiff(f, 4, h=1e-6)

## [1] 3

finddiff(sin, pi/4, h=1)

## [1] 0.2699545

finddiff(sin, pi/4, h=0.5)

## [1] 0.5048857

finddiff(sin, pi/4, h=0.)

## [1] NaN

symdiff <- function(f, x, h = x\* .Machine$double.eps^(1/3)) {  
 return((f(x + h) - f(x - h)) / (2 \* h))  
}

finddiff(sin, pi/4, h=1e-6)

## [1] 0.7071064

finddiff(sin, pi/4, h=1e-10)

## [1] 0.7071077

finddiff(sin, pi/4, h=1e-14)

## [1] 0.7105427

finddiff(sin, pi/4, h=1e-15)

## [1] 0.7771561

finddiff(sin, pi/4, h=1e-1)

## [1] 0.670603

finddiff(sin, pi/4)

## [1] 0.7071068

f <- function(x) {x^2 + 3\*x - 4}  
  
finddiff(sin, 2)

## [1] -0.4161469

symdiff(sin, pi/4, h=0.01)

## [1] 0.707095

symdiff(sin, pi/4, h=0.001)

## [1] 0.7071067

symdiff(sin, pi/4, h=0.0001)

## [1] 0.7071068

symdiff(sin, pi/4)

## [1] 0.7071068

## Section 5.1.2: The Second Derivative

finddiff2 <- function(f, x, h) {  
 return((f(x + h) - 2\*f(x) + f(x - h)) / h^2)  
}  
  
finddiff2(sin, pi/4, h = 1e-4); -sin(pi/4)

## [1] -0.7071068

## [1] -0.7071068

finddiff2(sin, 3, h = 1e-4); -sin(3)

## [1] -0.14112

## [1] -0.14112

## Section 5.2.1: Multipanel Interpolation Rules

midpt <- function(f, a, b, m = 100) {  
 nwidth = (b - a) / m  
 x = seq(a, b-nwidth, length.out = m) + nwidth / 2  
 y = f(x)  
   
 area = sum(y) \* abs(b - a) / m  
 return(area)  
}  
  
f <- function(x) { x^2 }  
  
midpt(f, 0, 1, m = 2)

## [1] 0.3125

midpt(f, 0, 1, m = 10)

## [1] 0.3325

midpt(f, 0, 1, m = 100)

## [1] 0.333325

midpt(f, 0, 1, m = 1000)

## [1] 0.3333332

trap <- function(f, a, b, m = 100) {  
 x = seq(a, b, length.out = m+1)  
 y = f(x)  
   
 p.area = sum((y[2: (m+1)] + y[1:m]))  
 p.area = p.area \* abs(b - a) / (2 \* m)  
 return(p.area)  
}  
  
simp <- function(f, a, b, m = 100) {  
 x.ends = seq(a, b, length.out = m+1)  
 y.ends = f(x.ends)  
 x.mids = (x.ends[2: (m + 1)] - x.ends[1:m]) / 2 + x.ends[1:m]  
 y.mids = f(x.mids)  
   
 p.area = sum(y.ends[2: (m+1)] + 4\*y.mids[1:m] + y.ends[1:m])  
 p.area = p.area \* abs(b - a) / (6\*m)  
 return(p.area)  
}  
  
f <- function(x) { x^2 }  
  
trap(f, 0, 1, m = 2)

## [1] 0.375

trap(f, 0, 1, m = 10)

## [1] 0.335

trap(f, 0, 1, m = 100)

## [1] 0.33335

trap(f, 0, 1, m = 1000)

## [1] 0.3333335

simp38 <- function(f, a, b, m = 100) {  
 x.ends = seq(a, b, length.out = m + 1)  
 y.ends = f(x.ends)  
 x.midh = (2 \* x.ends[2: (m+1)] + x.ends[1:m]) / 3  
 x.midl = (x.ends[2: (m+1)] + 2 \* x.ends[1:m]) / 3  
 y.midh = f(x.midh)  
 y.midl = f(x.midl)  
   
 p.area = sum(y.ends[2:(m+1)] + 3 \* y.midh[1:m] + 3 \* y.midl[1:m] + y.ends[1:m])  
 p.area = p.area \* abs(b - a) / (8 \* m)  
   
 return(p.area)  
}  
  
midpt(f, 0, 1, m=1)

## [1] 0.25

trap(f, 0, 1, m=1)

## [1] 0.5

simp(f, 0, 1, m=1)

## [1] 0.3333333

simp38(f, 0, 1, m=1)

## [1] 0.3333333

f <- function(x) { x^4 - x^2 + 1 }

midpt(f, 0, 1, m=1)

## [1] 0.8125

trap(f, 0, 1, m=1)

## [1] 1

simp(f, 0, 1, m=1)

## [1] 0.875

simp38(f, 0, 1, m=1)

## [1] 0.8703704

f <- function(x) { sin(x) + cos(x) }  
  
midpt(f, 0, 1, m=10)

## [1] 1.301711

trap(f, 0, 1, m=10)

## [1] 1.300084

simp(f, 0, 1, m=10)

## [1] 1.301169

simp38(f, 0, 1, m=10)

## [1] 1.301169

simp(f, 0, pi, m=2)

## [1] 2.00456

simp(f, 0, pi, m=5)

## [1] 2.00011

simp(f, 0, pi, m=10)

## [1] 2.000007

simp(f, 0, pi, m=100)

## [1] 2

simp38(f, 0, pi, m = 2)

## [1] 2.00201

simp38(f, 0, pi, m = 5)

## [1] 2.000049

simp38(f, 0, pi, m = 10)

## [1] 2.000003

simp38(f, 0, pi, m = 100)

## [1] 2

## Section 5.2.3: Newton-Cotes Forms, Generally

trap(log, 0, 1, m=10)

## [1] -Inf

simp(log, 0, 1, m=10)

## [1] -Inf

midpt(f, 0, 1, m=10)

## [1] 1.301711

midpt(f, 0, 1, m=100)

## [1] 1.301174

midpt(f, 0, 1, m=1000)

## [1] 1.301169

f <- function(x) { 1/x }  
  
trap(f, 0, 1, m=100)

## [1] Inf

midpt(f, 0, 1, m=10)

## [1] 4.266511

midpt(f, 0, 1, m=100)

## [1] 6.568684

midpt(f, 0, 1, m=1000)

## [1] 8.871265

## Section 5.3.2: Implementation Details

gaussint <- function(f, x, w) {  
 y <- f(x)  
   
 return(sum(y \* w))  
}  
  
  
w = c(1, 1)  
x = c(-1 / sqrt(3), 1 / sqrt(3))  
f <- function(x) { x^3 + x + 1 }  
gaussint(f, x, w)

## [1] 2

trap(f, -1, 1, m = 1)

## [1] 2

gaussint(cos, x, w)

## [1] 1.675824

trap(cos, -1, 1, m = 1)

## [1] 1.080605