Class Prep 9: 5.1.1 to 5.3.2

Chapter 6: Root Finding and Optimization

Section 6.1.1: Bisection Method

```
library(cmna)
library(pracma)

##
## Attaching package: 'pracma'

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

bisection <- function(f, a, b, tol = 1e-3, m = 100) {
  iter <- 0
  f.a <- f(a)
  f.b <- f(b)

  while(abs(b - a) > tol) {
    iter <- iter + 1
    if (iter > m) {
      warning("iterations max exceeded")
      break
    }
    xmid <- (a+b)/2
    ymid <- f(xmid)
    if(f.a*ymid > 0) {
      a <- xmid
      f.a <- ymid
    } else {
      b <- xmid
      f.b <- ymid

    }
  }

  root <- (a+b)/2
  return(root)
}


f <- function(x) {
  return(x^2 - 1)
}
```

```r
bisection(f, .5, 1.25, tol = 1e-3)
```

```
## [1] 0.9998779
```

```r
bisection(f, .5, 1.25, tol = 1e-6)
```

```
## [1] 0.9999999
```

```r
f <- function(x) {
  return(x^3 - x)
}

bisection(f, -2, 1.25, tol = 1e-6)
```

```
## [1] -0.9999997
```

```r
bisection(f, -.5, 1.25, tol = 1e-6)
```

```
## [1] 1.788139e-07
```

```r
bisection(f, -2, 1.25, tol = 1e-6)
```

```
## [1] -0.9999997
```

```r
bisection(sin, 1, 7, tol = 1e-6)
```

```
## [1] 3.141593
```

```r
bisection(sin, -50, 100, tol = 1e-6)
```

```
## [1] -9.424778
```

```r
bisection(sin, -1000, 2000, tol = 1e-6)
```

```
## [1] 1721.593
```

```r
bisection(tan, 1, 2)
```

```
## [1] 1.570801
```

```r
bisection(tan, -1, 1)
```

```
## [1] -0.0004882812
```

Section 6.1.2: Newton-Raphson Method

```
newton <- function(f, fp, x, tol = 1e-3, m = 100) {
  iter <- 0

  oldx <- x
  x <- oldx + 10 * tol

  while(abs(x - oldx) > tol) {
    iter <- iter + 1
    if(iter > m) {
      stop("no solutions found")
    }
    oldx <- x
    x <- x - f(x) / fp(x)
  }

  return(x)
}

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

fp <- function(x) {
  return(2*x)
}

newton(f, fp, 1.25, tol = 1e-3)
```

```
## [1] 1
```

```
newton(f, fp, -1100, tol = 1e-6)
```

```
## [1] -1
```

```
newton(f, fp, 1e-6, tol = 1e-9)
```

```
## [1] 1
```

```
f <- function(x) {
  return(x^2 - 2*x + 1)
}

fp <- function(x) {
  return(2*x - 2)
}

newton(f, fp, 1.25, tol = 1e-3)
```

```
## [1] 1.000508
```

```
newton(f, fp, -1100, tol = 1e-6)
## [1] 0.9999995
newton(f, fp, 1e-6, tol = 1e-9)
## [1] 1
newton(f, fp, 0, tol = 1e-3)
## [1] 0.9990332
newton(sin, cos, 2, tol = 1e-6)
## [1] 3.141593
newton(sin, cos, pi, tol = 1e-6)
## [1] 3.141593
newton(sin, cos, pi/2, tol = 1e-6)
## [1] 99978.04
cos(pi/2)
## [1] 6.123032e-17
```

Section 6.1.3: Secant Method

```
secant <- function(f, x, tol = 1e-3, m = 100) {
  i <- 0

  oldx <- x
  oldfx <- f(x)

  x <- oldx + 10 * tol

  while(abs(x - oldx) > tol) {
    i <- i + 1
    if(i > m) {
      stop("No solution found")
    }

    fx <- f(x)
    newx <- x - fx * ((x - oldx) / (fx - oldfx))
    oldx <- x
    oldfx <- fx
    x <- newx
  }

  return(x)
}



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

secant(f, 1.25, tol = 1e-3)

## [1] 1

secant(f, -1100, tol = 1e-6)

## [1] -1

secant(f, 1e-6, tol = 1e-9)

## [1] 1

secant(sin, 2, tol = 1e-6)

## [1] 3.141593

secant(sin, pi, tol = 1e-6)

## [1] 3.141593
```