Computing Group 5 HW 6

November 22, 2020

Exercises 129, 132, 136-140, 143, 149-150, 152, 154, 156, 158-159, 163

1 126.

```
cos_fixed = function(x, tol, i=0){
  while(signif(x, tol) != signif(cos(x), tol)){
    x=cos(x)
    i=i+1
  }
  return(c(x, cos(x), i))
}

# (x, cos(x), number of iterations)
cos_fixed(0.5, 2)

## [1]  0.7350063  0.7418265 10.0000000

cos_fixed(0.5, 3)

## [1]  0.7387045  0.7393415 16.0000000

cos_fixed(0.5, 4)

## [1]  0.7391091  0.7390690 23.0000000

cos_fixed(0.7, 2)

## [1] 0.7444212 0.7354802 5.0000000

cos_fixed(0.7, 3)

## [1]  0.7387487  0.7393117 12.0000000

cos_fixed(0.7, 4)

## [1]  0.7391318  0.7390537 17.0000000
```

```r
cos_fixed(0, 2)
```

```
## [1]  0.7442374  0.7356047 11.0000000
```

```r
cos_fixed(0, 3)
```

```
## [1]  0.7387603  0.7393039 18.0000000
```

```r
cos_fixed(0, 4)
```

```
## [1]  0.7391302  0.7390548 23.0000000
```

2 132.

```r
hornerpoly <- function(x,a) {
  res <- numeric(length(x))
  for(j in 1:length(x)) {
    v <- a
    for(i in (length(a)-1):1) {
      v[i] <- (v[i+1]*x[j]) + a[i]
    }
    res[j]<-v[1]
  }
  return(res)
}

polyderiv=function(beta){
  beta_deriv=beta*(seq(1:length(beta))-1)
  return(beta_deriv[-1])
}

newton_poly = function(x, beta, eps){
  counter = 0
  while(abs(hornerpoly(x, beta))>eps){
    x = x-hornerpoly(x, beta)/hornerpoly(x, polyderiv(beta))
    counter = counter + 1
  }
  return(c(x, counter))
}

newton_poly(0, c(5, 0.0005, 605, 0.0605, 10600, 1.06, 10000, 1), eps=0.0001)
```

```
## [1] -10000      1
```

```r
#The solution is found after just 1 iteration.
```

3 136.

To Do

4 137.

$$f(x) = 1/x - y$$
$$\Rightarrow f'(x) = -1/x^2$$
$$\Rightarrow \frac{f(x)}{f'(x)} = \frac{1/x - y}{-1/x^2} = -(1/x - y)x^2 = yx^2 - x$$
$$\Rightarrow x_{n+1} = x_n - (yx_n^2 - x_n) = 2x_n - yx_n^2$$

Applying this method iteratively will find the reciprocal of y.

5 138.

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$
$$= \frac{x_k(f(x_k) - f(x_{k-1})) - f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$
$$= \frac{f(x_k)x_k - f(x_{k-1})x_k - f(x_k)x_k + f(x_k)x_{k-1}}{f(x_k) - f(x_{k-1})}$$
$$= \frac{x_{k-1}f(x_k) - x_k f(x_{k-1})}{f(x_K) - f(x_{k-1})}$$

For $x_k \approx x_{k-1}$, the iterative method may have representation errors, whereas the secant method would not.

6 139.

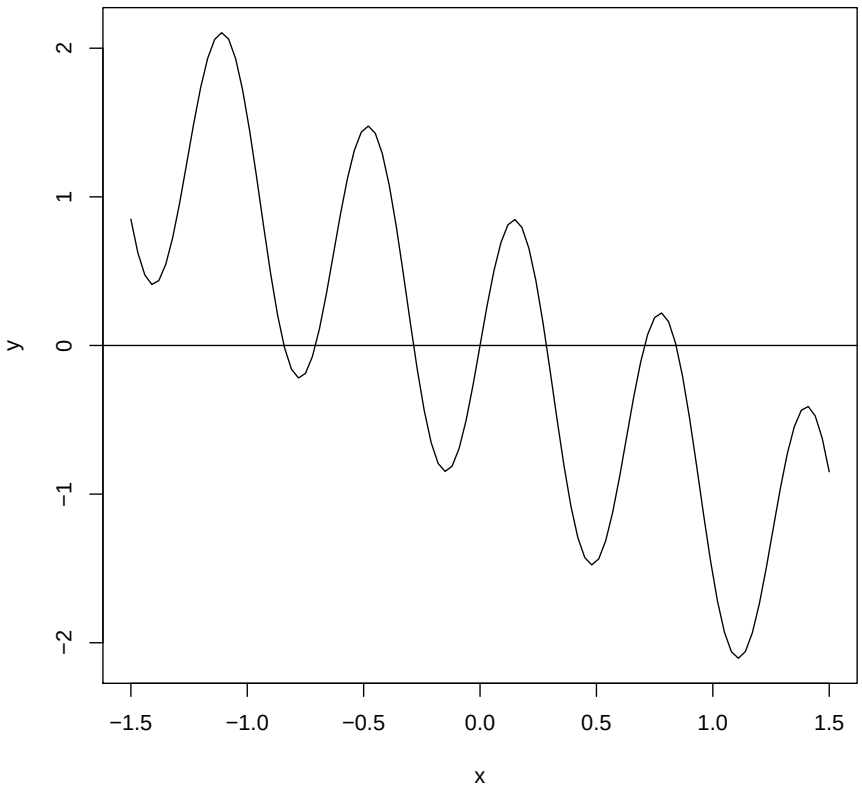
$$x_1 - 1 = 0$$
$$x_1x_2 - 1 = 0 \iff x_1x_2 = 1 \iff x_2 = \frac{1}{x_1}$$
$$f(x) = \frac{1}{x}$$
$$f'(x) = -\frac{1}{x^2}$$
$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = x_1 + \frac{\frac{1}{x_1}}{\frac{1}{x_1^2}} = x_1 + \frac{x_1^2}{x_1} = 2x_1$$

If we start with any ordered pair where $x_1 = 0$, x_2 will always be 0 after any number of iterations. Therefore, we won't reach the solution $(1, 1)$.

7 140.

```
curve(sin(10*x)-x, ylab="y", from=-1.5, to=1.5)
abline(add=TRUE, h=0)

## Warning in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...):
"add" is not a graphical parameter
```




```
f=function(x) abs(sin(10*x)-x)
root=numeric(7)
root[1]=optim(-.85, f)$par

## Warning in optim(-0.85, f):  one-dimensional optimization by Nelder-Mead
is unreliable:
## use "Brent" or optimize() directly
```

```r
root[2]=optim(-.7, f)$par
```

```
## Warning in optim(-0.7, f):  one-dimensional optimization by Nelder-Mead
is unreliable:
## use "Brent" or optimize() directly
```

```r
root[3]=optim(-.3, f)$par
```

```
## Warning in optim(-0.3, f):  one-dimensional optimization by Nelder-Mead
is unreliable:
## use "Brent" or optimize() directly
```

```r
root[4]=optim(0, f)$par
```

```
## Warning in optim(0, f):  one-dimensional optimization by Nelder-Mead
is unreliable:
## use "Brent" or optimize() directly
```

```r
root[5]=optim(.3, f)$par
```

```
## Warning in optim(0.3, f):  one-dimensional optimization by Nelder-Mead
is unreliable:
## use "Brent" or optimize() directly
```

```r
root[6]=optim(.7, f)$par
```

```
## Warning in optim(0.7, f):  one-dimensional optimization by Nelder-Mead
is unreliable:
## use "Brent" or optimize() directly
```

```r
root[7]=optim(.85, f)$par
```

```
## Warning in optim(0.85, f):  one-dimensional optimization by Nelder-Mead
is unreliable:
## use "Brent" or optimize() directly
```

```r
cat("The function has 7 zeroes, the roots are: \n", root, "\n")
```

```
## The function has 7 zeroes, the roots are:
##  -0.8423204 -0.7068174 -0.2852342 0 0.2852342 0.7068174 0.8423204
```