Class Prep 8 | 4.1.1 – 4.2.1

Riley Primeau

2022-10-23

Section 4.1.1 Linear Interpolation

```
linterp = function (x1, y1, x2, y2)
{
  m = (y2-y1)/(x2 - x1)
  b = y2 - m * x2

  return(c(b,m))
}
(p = linterp(2,3,0,-1))
```

```
## [1] -1  2
```

```
library(cmna)
```

```
##
## Attaching package: 'cmna'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     linterp
```

```
horner(1,p)
```

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

Section 4.1.2 Higher-Order Polynomial Interpolation

```
polyinterp = function(x, y)
{
  if(length(x) != length(y))
    stop("Length of x and y vectors must be the same")
  n = length(x) -1
  vandermonde = rep(1, length(x))
  for(i in 1:n)
  {
    xi = x^i
    vandermonde = cbind(vandermonde, xi)
  }
  beta = solve(vandermonde, y)

  names(beta) = NULL
  return(beta)
}
```

```
x = c(-1, 1, 0)
y = c(-2, 2, -1)
(p = polyinterp(x, y))
```

```
## [1] -1  2  1
```

```
horner(-2, p)
```

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

```
x = c(2,0)
y = c(3,-1)
(p = polyinterp(x, y))
```

```
## [1] -1  2
```

```
x = c(-3, -1, 3, 1)
y = c(1 - tan(1), 1 - tan(1/2), 1 + tan(1/2), 1)
(p = polyinterp(x, y))
```

```
## [1]  0.755898927  0.263467854 -0.029050172  0.009683391
```

Section 4.2.1 Piecewise Linear Interpolation

```
pwiselinterp = function(x,y)
{
  n = length(x) - 1

  y = y[order(x)]
  x = x[order(x)]

  mvec = bvec = c()

  for(i in 1:n) {
    p = linterp(x[i], y[i], x[i+1], y[i+1])
    mvec = c(mvec, p[2])
    bvec = c(bvec, p[1])
  }
  return(list(m = mvec, b = bvec))
}

x = c(-2, -1, 0, 1)
y = c(-1, -2, -1, 2)
pwiselinterp(x, y)
```

```
## $m
## [1] -1  1  3
##
## $b
## [1] -3 -1 -1
```

```
f = approxfun(x, y)
f(0)

## [1] -1

f(.5)

## [1] 0.5
```