3.4 Cubic Spine Interpolation

1.d construct free cubic spline for the following data.

Using the spline function gives the following tables:

> x=c(3.0,3.1,3.2)

> y=c(-4.240058,-3.496909,-2.596792)

> A=cbind(x, y)

> source('~/math311/3.4CubicSpline.R')

> p1d = spline(A)

> p1d

x a b c d

[1,] 3.0 -4.240058 7.03907 0.0000 39.242

[2,] 3.1 -3.496909 8.21633 11.7726 -39.242

[3,] 3.2 -2.596792 NA NA NA

This table defines the spline as:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ---------------------- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2.d

The interpolated value can be calculated by result from exercise #1

> p1d[2, 2:5]

a b c d

-3.496909 8.216330 11.772600 -39.242000

> interp = sum(p1d[2, **2:5**] \* c(1, cumprod( rep( pi-3.1, 3))))

> interp

[1] -3.137628

> f = function(x) { x\*cos(x)-x^2\*sin(x) }

> actual = f(pi)

> actual

[1] -3.141593

> error = abs (actual - interp)

> error

[1] 0.003965031

To find approximation for the derivative at the point, simply take the derivative of the cubic polynomial and evaluate at that point.

> fp = function(x) { 8.21633+ 23.5452\*(x - 3.1) - 117.726\*(x-3.1)^2}

> interp = fp(pi)

> interp

[1] 8.991977

> fpReal = function(x){-3\*x\*sin(x) + cos(x) - x^2 \*cos(x)}

> actual = fpReal(pi)

> actual

[1] 8.869604

> error = abs( actual - interp)

> error

[1] 0.122373

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ---------------------- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3.g Construct the clamped cubic spline using the data of Exercise 1:

> x=c(1, 1.1, 1.2, 1.3, 1.4)

> y=c(1.684370, 1.949477, 2.199796, 2.439189, 2.670324)

> A = cbind(x, y)

> p3g = spline(A, "clamped", 2.742245, 2.276919)

> p3g

x a b c d

[1,] 1.0 1.684370 2.742245 -1.0059261 0.9417607

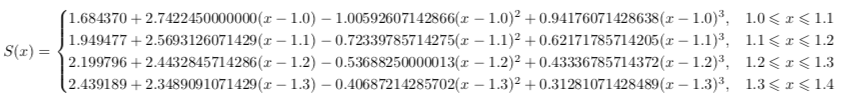
[2,] 1.1 1.949477 2.569313 -0.7233979 0.6217179

[3,] 1.2 2.199796 2.443285 -0.5368825 0.4333679

[4,] 1.3 2.439189 2.348909 -0.4068721 0.3128107

[5,] 1.4 2.670324 NA NA NA

The piece wise function this clamped spline defines is:



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ---------------------- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4.g

Since we are interpolating at f(1.15), where x=1.15 is between 1.1 and 1.2,

so we choose the function:



This R code is the dot product of the coefficients with the polynomial

> xs = 1.15

> interp = sum(p3g[2, 2:5] \* c(1, cumprod(rep(xs - 1.1, 3))))

> interp

[1] 2.076212

> f = function(x){log(exp(2\*x) - 2) }

> actual = f(xs)

> actual

[1] 2.076209

> error = abs( actual - interp)

> error

[1] 2.720533e-06

To find the approximation of the derivative at that point, simply take the derivate of the cubic polynomial and evaluate at that point. Again, using a dot product in R is easiest, with the added product of the exponents.

> p3g[2,3:5]

b c d

2.5693126 -0.7233979 0.6217179

> fxsnterp=sum(p3g[2, 3:5]\*c(1,2,3)\*c(1,cumprod(rep(xs-1.1,2))))

> fxsnterp

[1] 2.501636

> fp=function(x) { 2\*exp(2\*x)/(exp(2\*x)-2) }

> fxsnterp=sum(p3g[2,3:5]\*c(1,2,3)\*c(1,cumprod(rep(xs-1.1,2))))

> fxsnterp

[1] 2.501636

> actual = fp(xs)

> actual

[1] 2.501619

> error = abs( actual - fxsnterp)

> error

[1] 1.688454e-05

\*