MATH5824 Assessed Practical (Level 5 part)

Name: Dominic Lee
Student ID: 201691989
Date of submission: 03/03/2023

The first thing to do is pick where to place the knots, and how many to choose. Something that seems "natural" to do would be to to place n knots at the $0 * \frac{100}{n-1}, 1 * \frac{100}{n-1}, 3 * \frac{100}{n-1}, \ldots, (n-1)\frac{100}{n-1}$ quantiles of the engine size data, since if there is lots of data in a region, then the there should be more knots allowing for more fine variation in the function. We could also look at the data and see if there are any obvious places. Fig.1 shows a plot of engine wear vs size, with a line of best fit just to aid the eye. From looking at the graph, and guessing roughly the average wear for sizes with multiple data points, it seems there should be at least 2 knots, one between 1.58 and 1.78, and one between 2.13 and 2.32. However, since this project is not about the knots, I will use the data points as knots to simplify things, particularly the code.

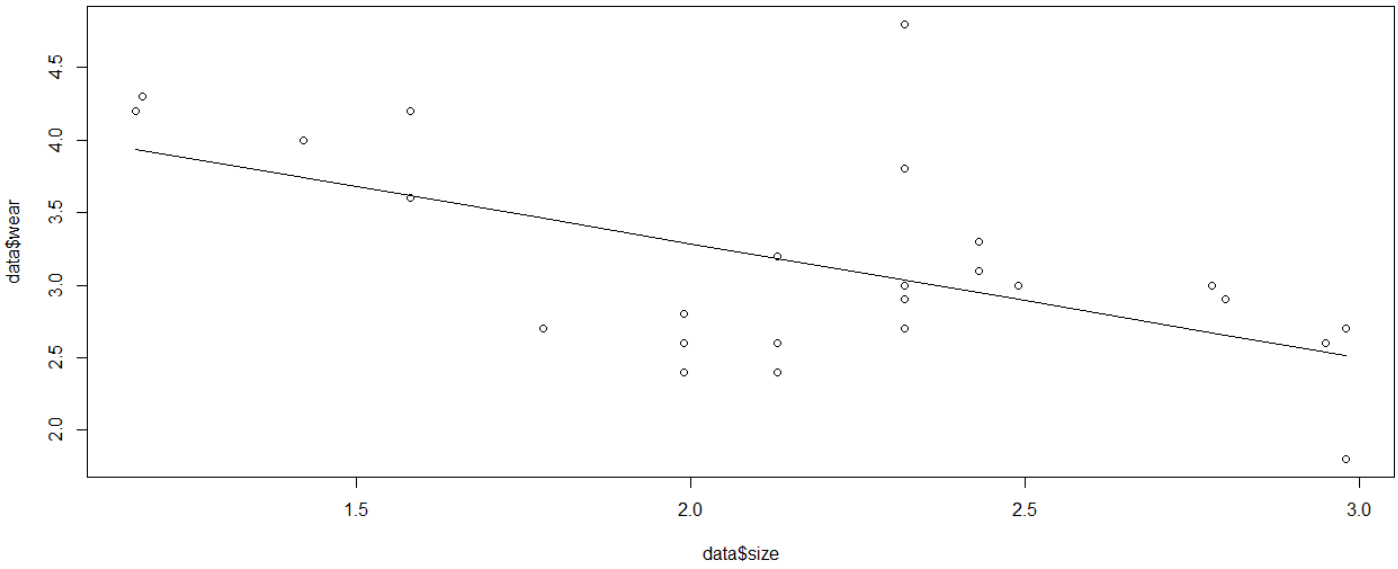

Figure 1: Plot of wear vs size, with line of best fit

First, I plotted the splines for certain values of lambda to see what seems a reasonable range (fig. 2). To do this, I used the solutions to last year's exercises to help me with the code for getting the coefficients of the interpolating spline of a data set given λ. I then plotted the interpolating spline over a graph of the data points to see what looked reasonable visually.

To me the highest λ value of 1, (pink line) seemed to not be a good enough fit for the data points, so we should use a lower value. On the othe hand, the lowest value (0.000001) seemed too "wiggly" so we should use some value inbetween. To make the plot clearer I removed those 2 lines (fig. 3).

We can see that the higher the value of λ is, the worse the fit gets, but the closer the line is to being straight, this is because the $\lambda J_2(\hat{f})$ term now carries more weight compared to the residuals, so the optimal function optimises more for smaller variations rather than closer fits. This means that as $\lambda \to \infty$, the function optimises "only" for less variation so we get $J_2(\hat{f}) \to 0$ i.e. $\int_{-\infty}^{\infty} f''(t)^2 dt \to 0$, i.e. $f''(t) \to 0$, i.e. $f(t) \to at + b$ for constants a, b, i.e. f tends to a linear function. This means the GCV will tend to a high number.

Conversely, as $\lambda \to 0$, the "penalty term" goes to 0, so the optimal function only optimises for sum of squares of residuals, so the function will go the points perfectly where it can, and where there are multiple values of engine wear for the same engine

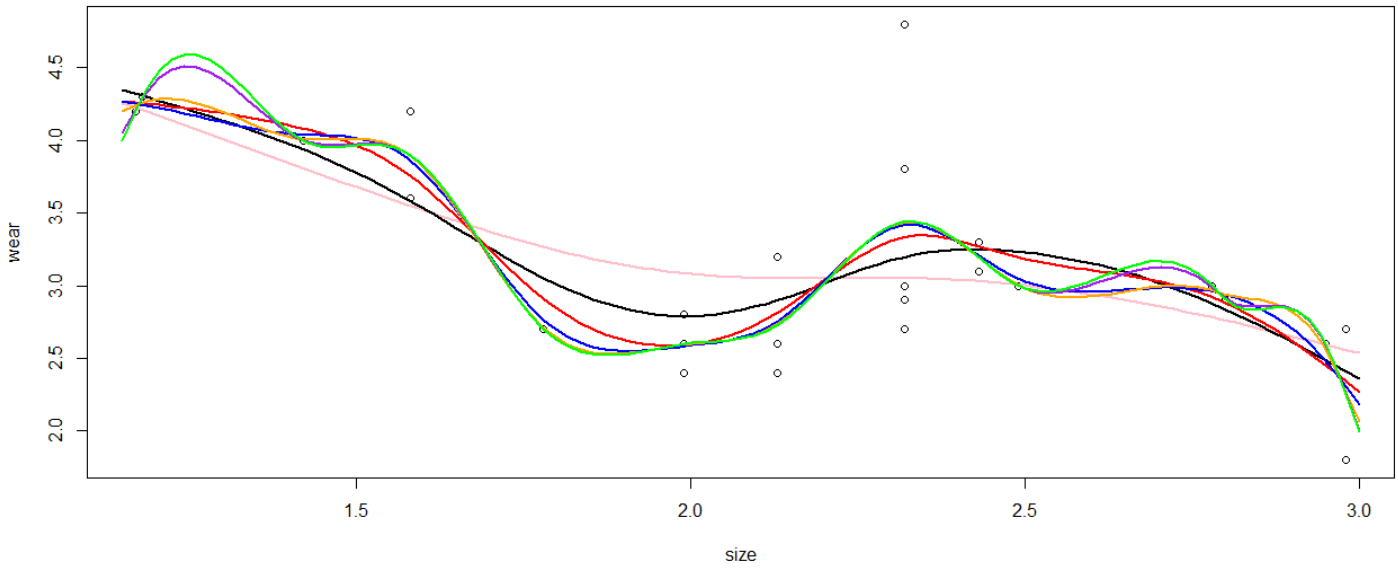

Figure 2: Plot of wear vs size, with splines with $\lambda = 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}$ with colours pink, black, red, blue, orange, purple, green respectively

size, the function will go through the mean of the wear values Note there are enough degrees of freedom for the function to be able to do this. This means the GCV will approach some low number, but not 0 as it can't go through multiple wear values at the same size value.

One way of checking how good a smoothness parameter is is to work out the GCV or OCV. However, since OCV is ambiguous when there are multiple engine wear values for the same engine size value, which there are here, I decided to use GCV. Alternatively I could have used a mean of the wear values for each size value, and then used OCV. I decided to plot a graph of GCV of the model using each λ value for $\lambda = 10^{-x}$, for each $x = 2, 2.1, 2.2, \ldots, 10$.

As can be seen from fig. 4, the gcv seems to be minimised at around 0.001, this, combined with the fact it seems to strike a balance between reflecting the true behaviour of the points and not being too "wiggly" (see blue line in fig. 5), means I think the best single λ value would be 0.001, but anything in the range 0.1 to 0.00001 would be acceptable in my opinion.

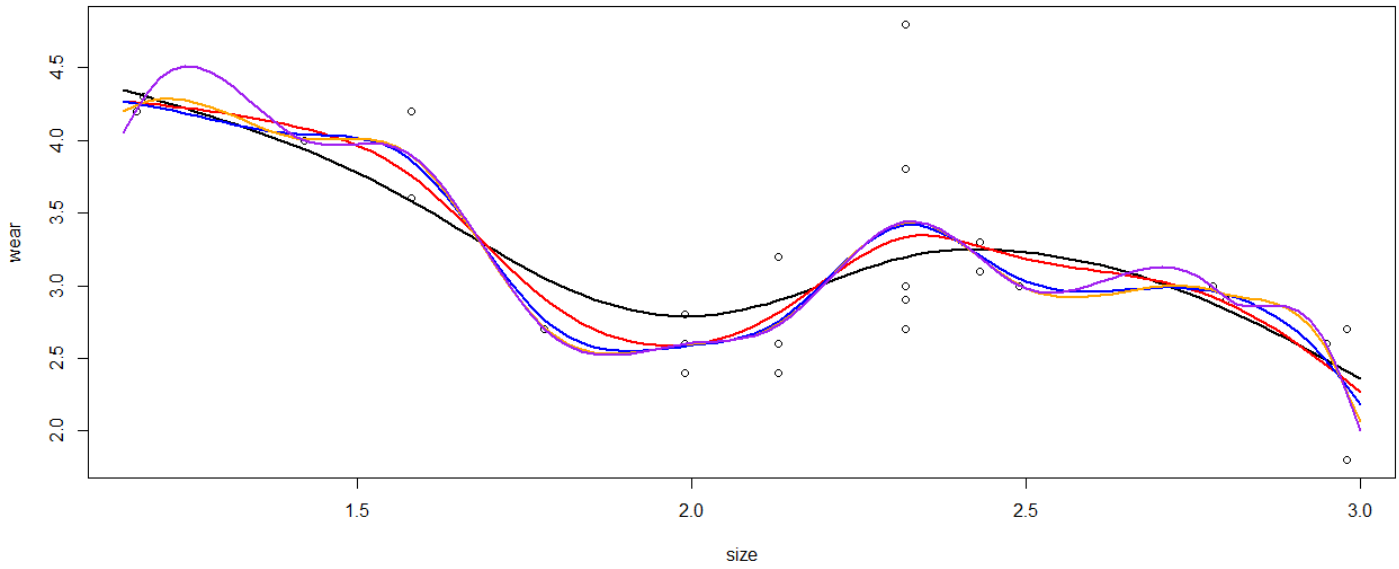

Figure 3: Plot of wear vs size, with splines with $\lambda = 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}$ with colours black, red, blue, orange, purple respectively

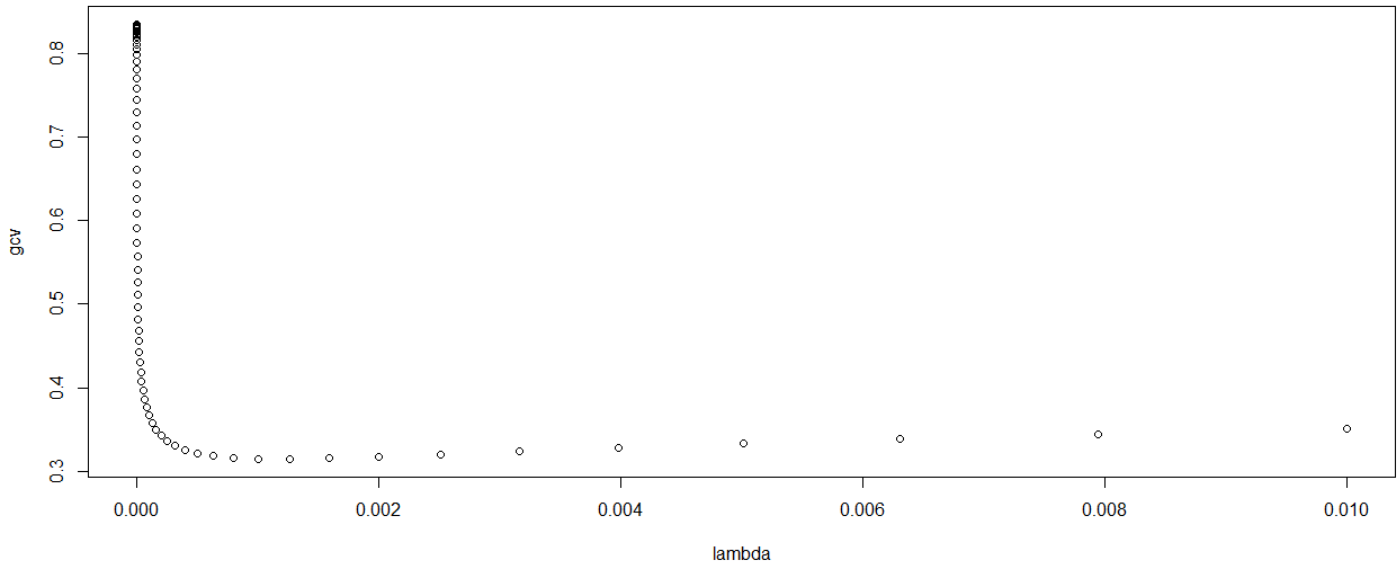

Figure 4: Plot of GCV of model with smoothness parameter lambda vs lambda

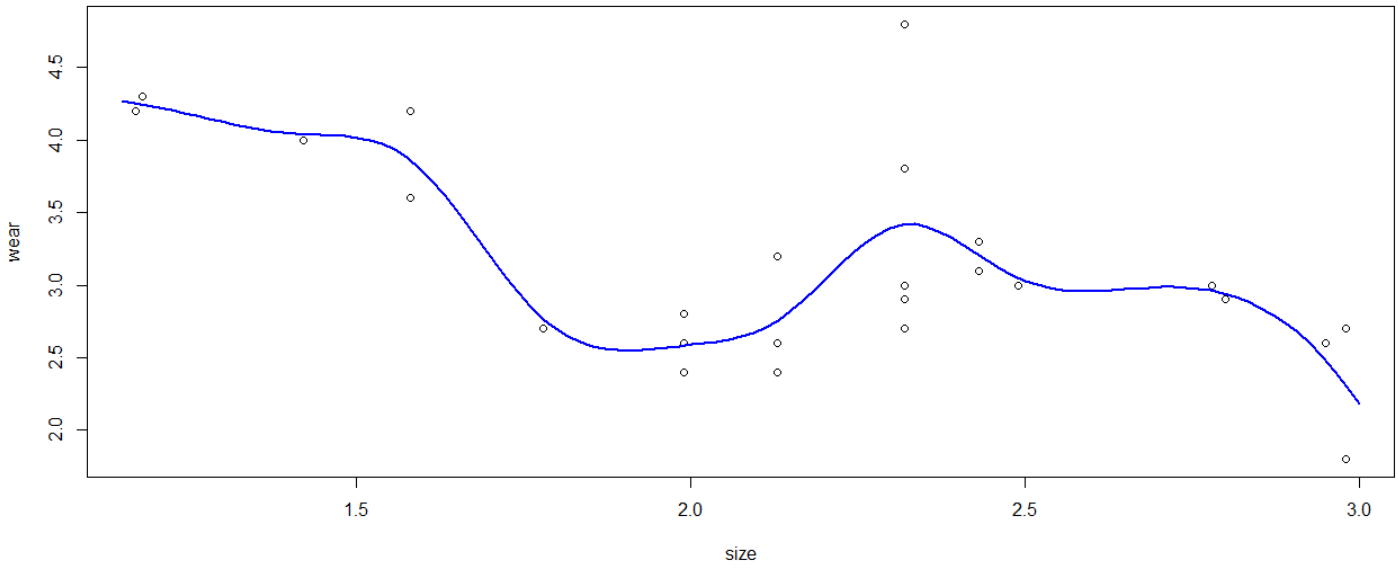

Figure 5: Plot of GCV of spline with smoothness parameter 0.001 on top of actual data

1 Appendix

```
data=data = read.csv("C:\\Users\\xboxl\\OneDrive\\GLAM practical\\enginewear.csv")
m1 = lm(wear ~ size, data=data)
plot(data$size,data$wear)
lines(data$size, predict(m1))


get.coef = function(x, y, lambda){
    n = length(x)
    absdiff3 = function(x,y){abs(x-y)^3}
    K = outer(x, x, absdiff3) + lambda * diag(n)
    L = cbind(rep(1, n), x)
    M = rbind(cbind(K, L), cbind(t(L),matrix(0, nrow=2, ncol=2)))

    coef = solve(M) %*% c(y, 0, 0)
    return(coef)
}

nat.cubic.spline = function(x, coef, knots){
    a0 = coef[length(coef)-1]
    a1 = coef[length(coef)]
    b = coef[1:(length(coef)-2)]
    return(a0 + a1*x + sum(b * abs(x - knots)^3))
}


f.coef0000001 =  get.coef(x=data$size, y=data$wear, lambda=0.000001)
f.coef000001 =  get.coef(x=data$size, y=data$wear, lambda=0.00001)
f.coef00001 =  get.coef(x=data$size, y=data$wear, lambda=0.0001)
f.coef0001 =  get.coef(x=data$size, y=data$wear, lambda=0.001)
f.coef001 =  get.coef(x=data$size, y=data$wear, lambda=0.01)
f.coef01 =  get.coef(x=data$size, y=data$wear, lambda=0.1)
f.coef1 =  get.coef(x=data$size, y=data$wear, lambda=1)
x = seq(1.15, 3, length=101)
y0000001 = y000001 = y00001 = y0001 = y001= y01 = y1 = numeric(length(x))


for(i in 1:101){
    y1[i] = nat.cubic.spline(x[i], f.coef1, data$size)
    y01[i] = nat.cubic.spline(x[i], f.coef01, data$size)
    y001[i] = nat.cubic.spline(x[i], f.coef001, data$size)
    y0001[i] = nat.cubic.spline(x[i], f.coef0001, data$size)
    y00001[i] = nat.cubic.spline(x[i], f.coef00001, data$size)
    y000001[i] = nat.cubic.spline(x[i], f.coef000001, data$size)
    y0000001[i] = nat.cubic.spline(x[i], f.coef0000001, data$size)
}

plot(x=data$size, y=data$wear, xlab = "size", ylab = "wear")

lines(y1~x, col="pink", lwd=2)
lines(y01~x, col="black", lwd=2)
lines(y001~x, col="red", lwd=2)
lines(y0001~x, col="blue", lwd=2)
lines(y00001~x, col="orange", lwd=2)
lines(y000001~x, col="purple", lwd=2)
lines(y0000001~x, col="green", lwd=2)


plot(x=data$size, y=data$wear, xlab = "size", ylab = "wear")
```

```
lines(y01~x, col="black", lwd=2)
lines(y001~x, col="red", lwd=2)
lines(y0001~x, col="blue", lwd=2)
lines(y00001~x, col="orange", lwd=2)
lines(y000001~x, col="purple", lwd=2


gcv=0
lambda = 10^(-rev(seq(from = 2, to = 10, by = 0.1)))
for(i in 1:81){
model=smooth.spline(data$size, data$wear, all.knots = TRUE, lambda=lambda[i])
gcv[i] = model$cv.crit}

plot(lambda,gcv)


plot(x=data$size, y=data$wear, xlab = "size", ylab = "wear")
lines(y0001~x, col="blue", lwd=2)
```