MATH5824M Assessed Practical

Faye Williams
ID - 201308646

July 23, 2023

1 Data Set

Due to the nature of the dataset, we have removed some reading with the same engine size and combined them together to give the average index of wear out of these plots, corresponding to the engine size. This means that our matrices which we require for building the spline are not singular, which therefore allows us to generate the spline when we would not be able to otherwise. Therefore we have the plots as follows

Size	1.42	1.58	1.78	1.99	2.13	2.32	2.43	2.78	2.98
Wear	4	4.2	2.5	2.6	2.733	3.625	3.2	3	2.25

1.1 Interpolating spline

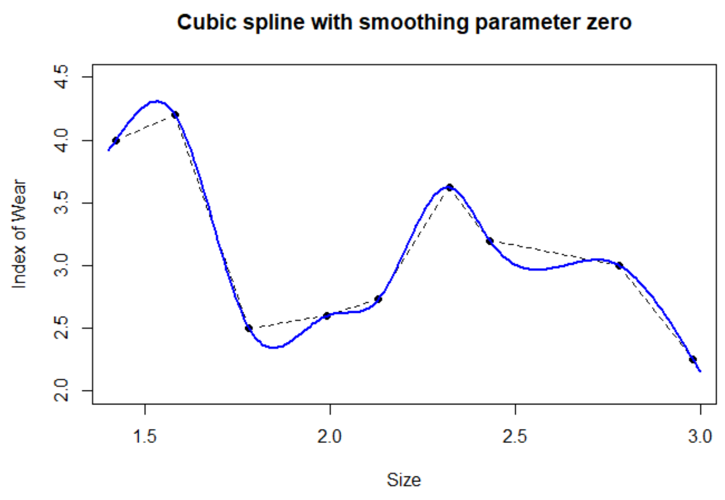


Figure 1: Interpolating Spline

I first plotted the cubic spline with a base value for lambda as zero using the R code in figures 3 and 4 in the appendix, which gave the unique interpolating spline without the smoothing parameter as shown in figure 1. We see that the spline indeed interpolates all our data points, and its curves between knots do not stray too far from the linear line connecting these knots.
Despite this, we see the most deviance of the spline from the connecting lines in between the first two knots, the third and fourth knot, and the seventh and eighth. This is partially due to the requirement of the gradient on both sides of each knot needing to be equal, therefore some restraint are applied which deviate the spline from the original plot, but help us find a smoother fit to the data.
We want to look further into how we can make this parameter smoother, which we do by introducing the smoothing parameter, λ, to our formula for building a spline, as we will investigate next.

1.2 Smoothing spline

Next we have a plot of cubic splines for this problem with a range of values for the smoothing parameter, as indicated by the key in the top right corner. This plot was given by the R code, as shown in figures 5 and 6 in the appendix.
As we can see, the only values of λ that showed any suitable spline were very small, and by the point λ reaches 1, we have almost a linear spline. This is partially due to us having to multiply our smoothing parameter by 12, as this is the constant given for cubic splines, however despite this we

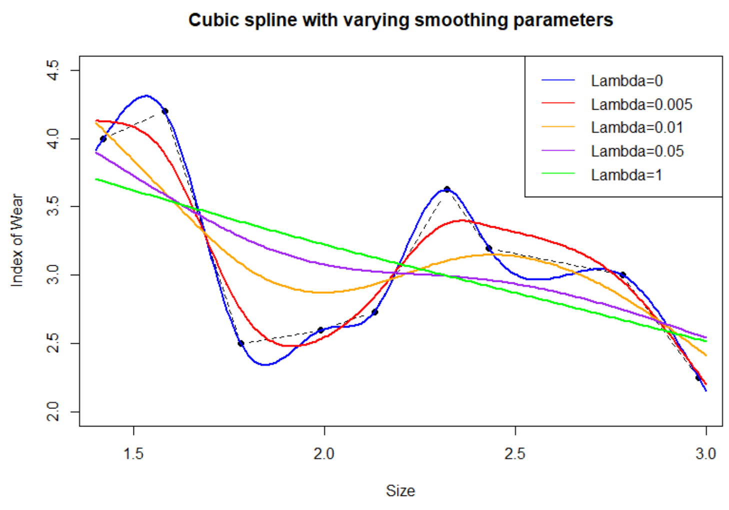


Figure 2: Smoothing Spline

would still have to use very small values for lambda to fit a suitable spline.
In fact, the red spline shows some similarity to our original blue spline, in the sense that they both start at a high index of wear for low values of engine size, then dip in index of wear when the engine is around 1.85. We then see a steady increase up to around 3.5 on the index of wear, occurring at around 2.4 engine size for both these splines. Lastly they both decrease and trail off at an index of wear at around 2.2, both with a size of 3. Therefore, in both these splines we see three main turning points where the gradient is zero.
However, in comparison to the smoother spline given by $\lambda = 0.01$, we see only two turning points occurring at different points to those of the red and blue splines. The local minimum is seen at around $(1.9, 3.0)$ and we see a local maximum at $(2.5, 3.1)$. as we can see there is a small difference between the index of wear of these points, this suggests the spline is in fact becoming smoother, as we see less variation in our data. Making this even smoother, we have the green and purple splines, with parameters lambda=0.05 and 1.
The green and purple splines show a similar shape, except the purple spline has slightly more curvature due to the value of λ being lower here. Even though the value of λ for the purple line is half of lambda for the green spline, we don't see as much difference in these splines as we would perhaps expect. Despite this, they both end at similar points, being higher than the readings of the other splines taken at engine size=3. This is due to these splines having an increased smoothing influence, which means for more extreme values, like $(2.98, 2.25)$, we are less likely to see the spline interpolate this point and stray further away from it to match the general trend of the data.

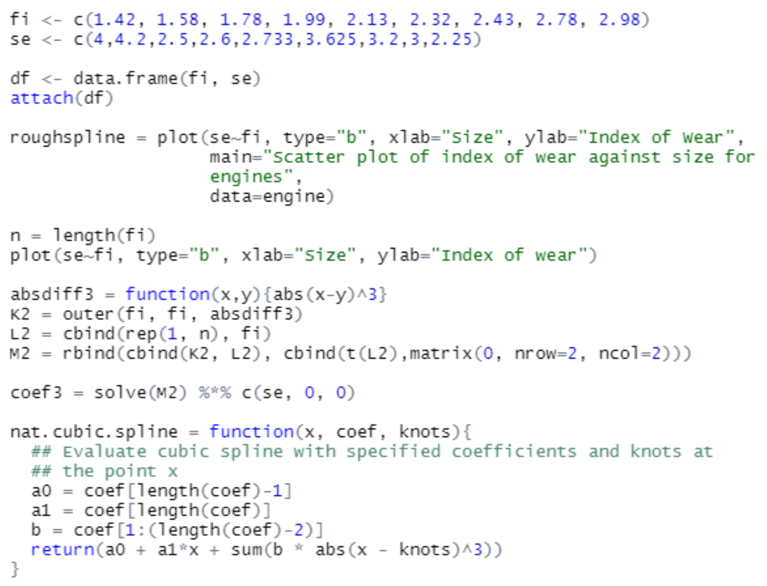
1.2.1 Varying lambda

We also see from this graph that as the smoothing parameter, λ, tends to zero we begin to see the original cubic spline, as shown in figure 1, that does not involve any smoothing aspect. This is what we would expect as substituting $\lambda = 0$ into our K^2 matrix would simply give us the K^1 matrix, which we use to build our interpolating spline. Although this spline is accurate in the sense that it travels through all of the knots, it is also rather complex and would be difficult to apply in certain aspects. Instead we would require a less complex model that still comes near to each of the knots, which we can see when increasing the value of λ.
In particular, as the smoothing parameter tends to infinity, we would see a linear line plotted from around $(1.4, 3.75)$ to roughly $(3, 2.5)$. As we can see this spline does not pass through any of the knots, nor come close to most of them. Therefore, despite its smoothness giving us a simpler model, it would not be a suitable spline as it is does not provide us with a good representation of the spread of the data.

1.2.2 Choice of λ

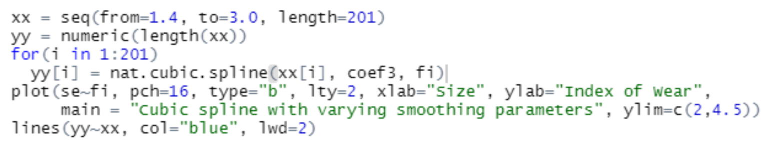
Thus, to find a suitable value for the smoothing parameter that balances these two aspects of giving us a simpler model but still an accurate one, we shall choose the smoothing parameter to be $\lambda = 0.005$, as represented by the red line.

A Appendices



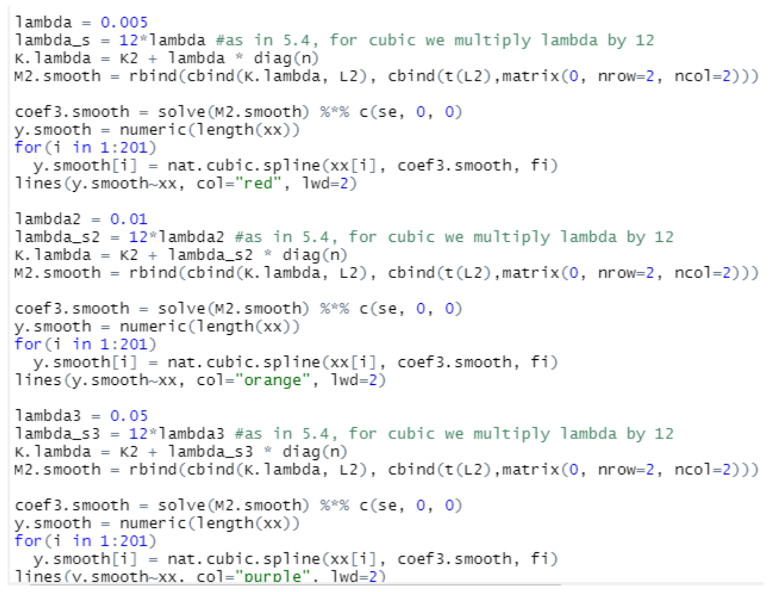
```
fi <- c(1.42, 1.58, 1.78, 1.99, 2.13, 2.32, 2.43, 2.78, 2.98)
se <- c(4,4.2,2.5,2.6,2.733,3.625,3.2,3,2.25)

df <- data.frame(fi, se)
attach(df)

roughspline = plot(se~fi, type="b", xlab="Size", ylab="Index of wear",
                   main="Scatter plot of index of wear against size for
                   engines",
                   data=engine)

n = length(fi)
plot(se~fi, type="b", xlab="Size", ylab="Index of wear")

absdiff3 = function(x,y){abs(x-y)^3}
K2 = outer(fi, fi, absdiff3)
L2 = cbind(rep(1, n), fi)
M2 = rbind(cbind(K2, L2), cbind(t(L2),matrix(0, nrow=2, ncol=2)))

coef3 = solve(M2) %*% c(se, 0, 0)

nat.cubic.spline = function(x, coef, knots){
  ## Evaluate cubic spline with specified coefficients and knots at
  ## the point x
  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))
}
```

Figure 3: Code for figure 1 (part 1)



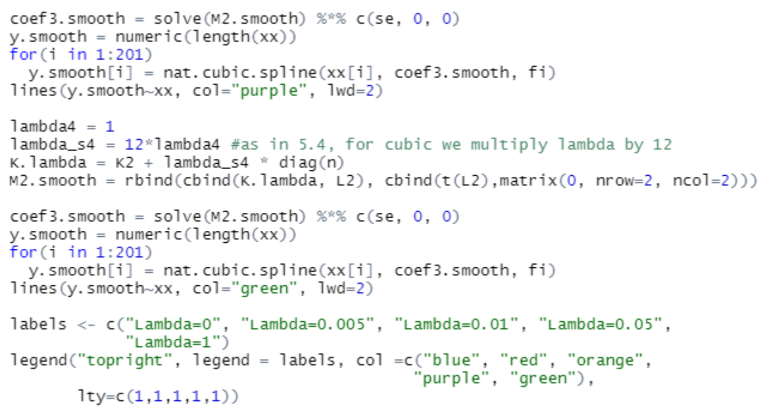
```
xx = seq(from=1.4, to=3.0, length=201)
yy = numeric(length(xx))
for(i in 1:201)
  yy[i] = nat.cubic.spline(xx[i], coef3, fi)
plot(se~fi, pch=16, type="b", lty=2, xlab="Size", ylab="Index of wear",
     main = "Cubic spline with varying smoothing parameters", ylim=c(2,4.5))
lines(yy~xx, col="blue", lwd=2)
```

Figure 4: Code for figure 1 (part 2)



```
lambda = 0.005
lambda_s = 12*lambda #as in 5.4, for cubic we multiply lambda by 12
K.lambda = K2 + lambda * diag(n)
M2.smooth = rbind(cbind(K.lambda, L2), cbind(t(L2),matrix(0, nrow=2, ncol=2)))

coef3.smooth = solve(M2.smooth) %*% c(se, 0, 0)
y.smooth = numeric(length(xx))
for(i in 1:201)
  y.smooth[i] = nat.cubic.spline(xx[i], coef3.smooth, fi)
lines(y.smooth~xx, col="red", lwd=2)

lambda2 = 0.01
lambda_s2 = 12*lambda2 #as in 5.4, for cubic we multiply lambda by 12
K.lambda = K2 + lambda_s2 * diag(n)
M2.smooth = rbind(cbind(K.lambda, L2), cbind(t(L2),matrix(0, nrow=2, ncol=2)))

coef3.smooth = solve(M2.smooth) %*% c(se, 0, 0)
y.smooth = numeric(length(xx))
for(i in 1:201)
  y.smooth[i] = nat.cubic.spline(xx[i], coef3.smooth, fi)
lines(y.smooth~xx, col="orange", lwd=2)

lambda3 = 0.05
lambda_s3 = 12*lambda3 #as in 5.4, for cubic we multiply lambda by 12
K.lambda = K2 + lambda_s3 * diag(n)
M2.smooth = rbind(cbind(K.lambda, L2), cbind(t(L2),matrix(0, nrow=2, ncol=2)))

coef3.smooth = solve(M2.smooth) %*% c(se, 0, 0)
y.smooth = numeric(length(xx))
for(i in 1:201)
  y.smooth[i] = nat.cubic.spline(xx[i], coef3.smooth, fi)
lines(y.smooth~xx, col="purple", lwd=2)
```

Figure 5: Code for figure 2 (part 1)



```
coef3.smooth = solve(M2.smooth) %*% c(se, 0, 0)
y.smooth = numeric(length(xx))
for(i in 1:201)
  y.smooth[i] = nat.cubic.spline(xx[i], coef3.smooth, fi)
lines(y.smooth~xx, col="purple", lwd=2)

lambda4 = 1
lambda_s4 = 12*lambda4 #as in 5.4, for cubic we multiply lambda by 12
K.lambda = K2 + lambda_s4 * diag(n)
M2.smooth = rbind(cbind(K.lambda, L2), cbind(t(L2),matrix(0, nrow=2, ncol=2)))

coef3.smooth = solve(M2.smooth) %*% c(se, 0, 0)
y.smooth = numeric(length(xx))
for(i in 1:201)
  y.smooth[i] = nat.cubic.spline(xx[i], coef3.smooth, fi)
lines(y.smooth~xx, col="green", lwd=2)

labels <- c("Lambda=0", "Lambda=0.005", "Lambda=0.01", "Lambda=0.05",
            "Lambda=1")
legend("topright", legend = labels, col =c("blue", "red", "orange",
                                           "purple", "green"),
       lty=c(1,1,1,1,1))
```

Figure 6: Code for figure 2 (part 2)