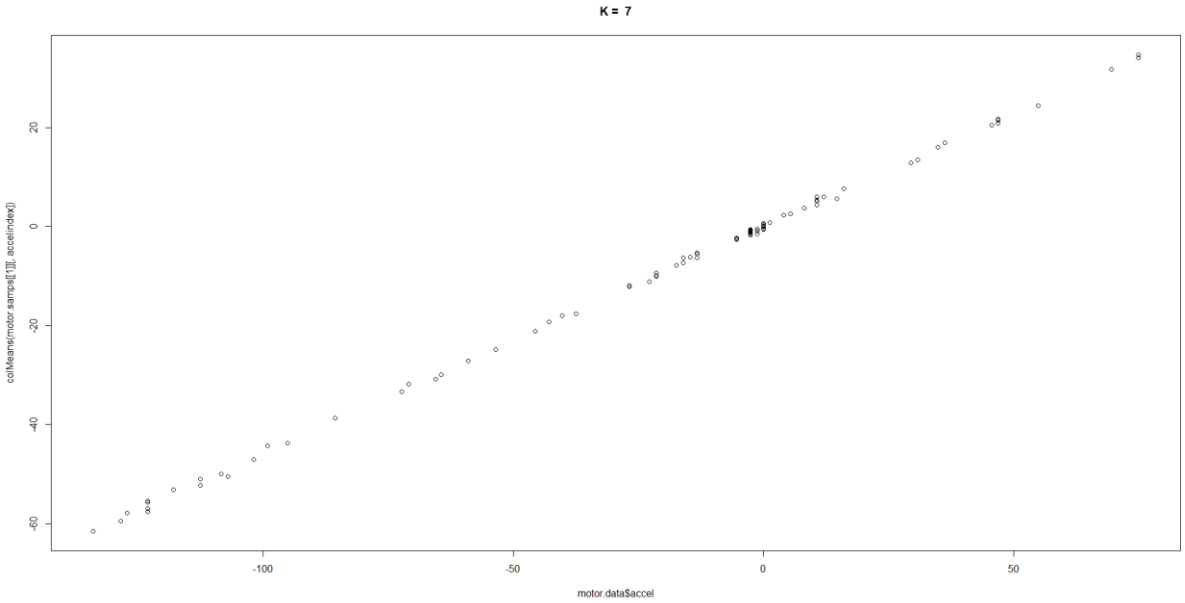
JAGS Smoothing

I have tried to complete this analysis however based on the steps provided. I have completed MCMC for different values of K in splines. Doing those calculation I got following values for DIC based on K. Starting K was 4 as R was not able to calculate X for K lower than 4.

K	DIC
4	1022
5	1054
6	3275470
7	214132
8	2113911
9	2316205
10	12243

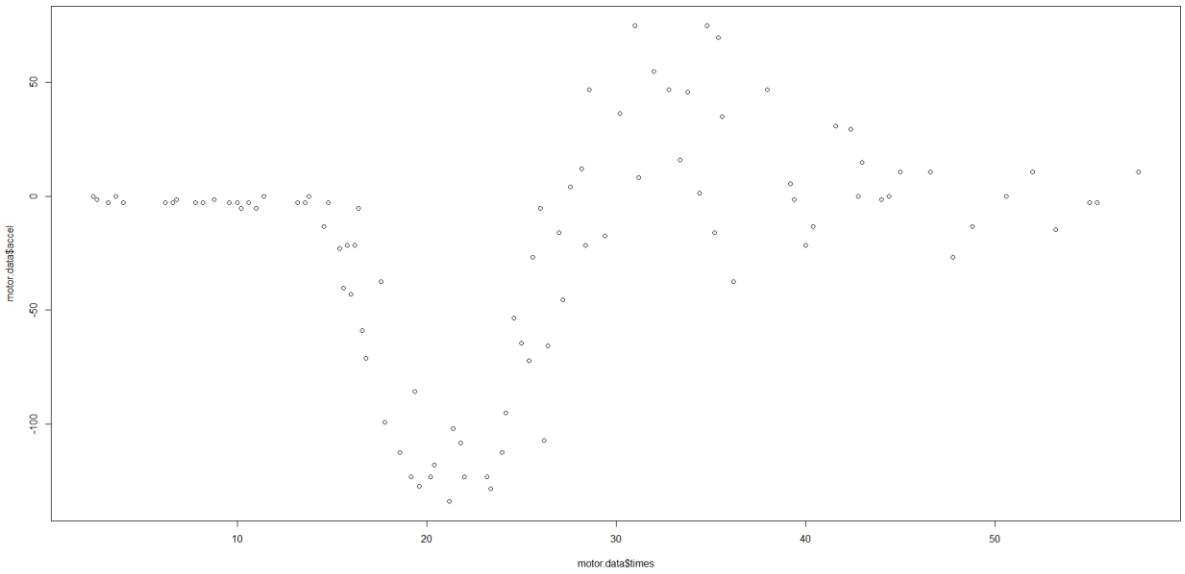
However, when rest of analysis is completed, and in particular estimating accelstar we don't get valid estimates and they are not aligned with expected (measured) values. Due to this I have selected the K=7 which best aligns accelstar to accel measured values. Next figure shows that graph:



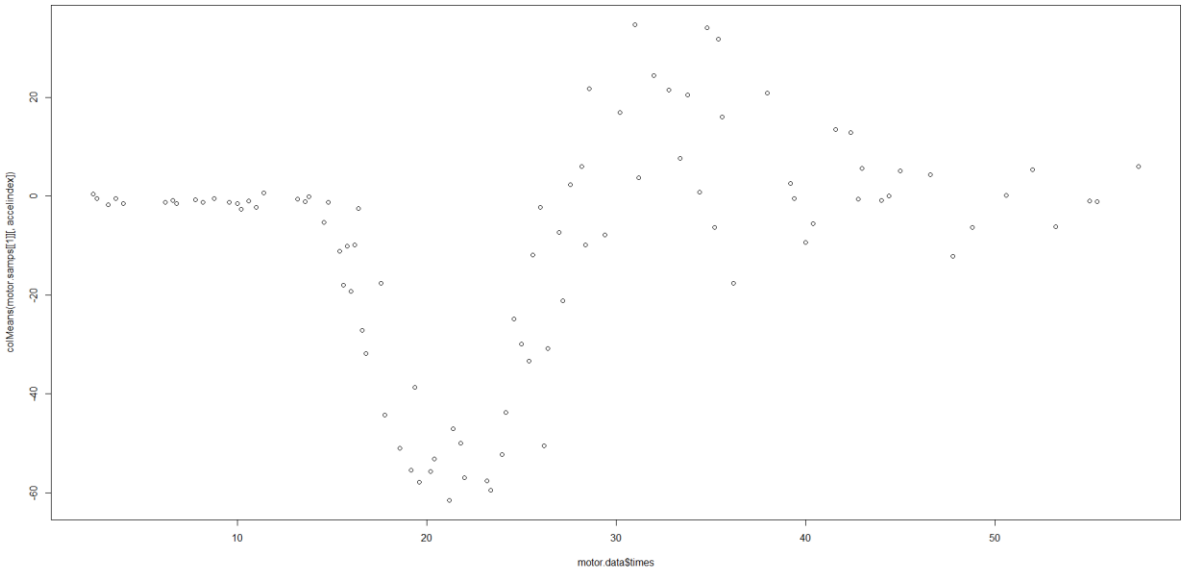
While that graph is the closest to alignment we can get we still are not aligned on the values. It is only alignment up to the constant of proportionality.

Additionally, gelman methods when run would not finish on my laptop.

Next figure shows the plot of observed data over time.



Similarly, I have produced a graph of predicted values over time and it can be seen on the figure below.



I was not sure how to calculate credible intervals for the curve. I have calculated them for the tau and lambda and they are

	Mean	Lower CI	Upper CI
Tau	7.103923	0.0001896741	0.8094826
Lambda	0.3475846	9.01196e-05	0.0481515

I understand that I probably have mistakes in my code and model but I was not able to find them and would appreciate feedback from you.

Appendix

R Code

```
require(rjags)
require(splines)
motor = read.table("motor.dat",header=T)
motor = append(list(N=nrow(motor)), motor)
K = 20
dic = list()
for (k in (4:K)) # starting at 4 as R complains everything lower is too small
{
  X = bs(motor$times, k, intercept = TRUE)
  motor.data = append(list(X=X),motor)
  motor.data = append(list(K=k),motor.data)

  motor.model = jags.model("motor.model", motor.data,n.chains=2)
  motor.samps = coda.samples(motor.model, variable.names=c("tau","lambda","b",
"accelstar"),1e4)
  motor.gelmman = gelman.diag(motor.samps,transform=TRUE)
  motor.dic = dic.samples(motor.model, 1e4)
  dic = append(dic, list(motor.dic))
  accelindex=grep("accelstar",colnames(motor.samps[[1]])) # find which variables in the
output relate to ystar
  plot(motor.data$accel, colMeans(motor.samps[[1]][,accelindex]), main = paste("K = ", k))
# plot the mean of the posterior samples against the data
  abline(0, 1)
}

dic # check DIC values. Smaller dic values are giving bad results

k = 7

X = bs(motor$times, k, intercept = TRUE)
motor.data = append(list(X=X),motor)
motor.data = append(list(K=k),motor.data)

motor.model = jags.model("motor.model", motor.data,n.chains=2)
motor.samps = coda.samples(motor.model, variable.names=c("tau","lambda","b",
"accelstar"),1e4)

# I tried to run gelman.diag and gelman.plot but it would not finish on my machine
#motor.gelmman = gelman.diag(motor.samps,transform=TRUE)
#acfplot(motor.samps)
#gelman.plot(motor.samps,ask=TRUE,transform=TRUE) # examine the BGR statistics graphically

accelindex=grep("accelstar",colnames(motor.samps[[1]])) # find which variables in the
output relate to ystar
plot(motor.data$accel, colMeans(motor.samps[[1]][,accelindex]), main = paste("K = ", k)) #
plot the mean of the posterior samples against the data
abline(0, 1)

chain=1
plot(motor.data$times, motor.data$accel)
plot(motor.data$times, colMeans(motor.samps[[1]][,accelindex]))

chain=1
cat("Posterior mean of mean tau = ",  mean(sapply(motor.samps,function(x)
x[,grep("tau",colnames(x))])), "\n")
cat("mean tau C.I. =",
HPDinterval(motor.samps[[chain]],prob=0.95)[grep("tau",colnames(motor.samps[[chain]])),],"\
n")
cat("Posterior mean of mean llambda = ",  mean(sapply(motor.samps,function(x)
x[,grep("lambda",colnames(x))])), "\n")
```

```
cat("mean beta C.I. =",
HPDinterval(motor.samps[[chain]],prob=0.95)[grep("lambda",colnames(motor.samps[[chain]])),]
,"\n")
```

Model

```
model{

  for (t in 1:N){
    accel[t] ~ dnorm(X[t, ] %*% b[,t], tau)
    for (k in 1:K) {
      b[k,t] ~ dnorm(0, lambda)
    }
  }

  tau ~ dgamma(0.001, 0.001)
  lambda ~ dgamma(0.001, 0.001)

  for(t in 1:N) {
                    accelstar[t] ~ dnorm(X[t, ] %*% b[,t], tau)
          }
}
```