A Supplementary Material: Examples using R

A.1 RKHS solution applied to Cubic Smoothing Spline

We apply the cubic smoothing spline within the RKHS framework on the benchmark motorcycle accident data used in Silverman (1985). These data come from a computer simulation of motorcycle accidents. The response is a series of measurements of head acceleration over time in a simulated motorcycle accident used to test crash helmets.

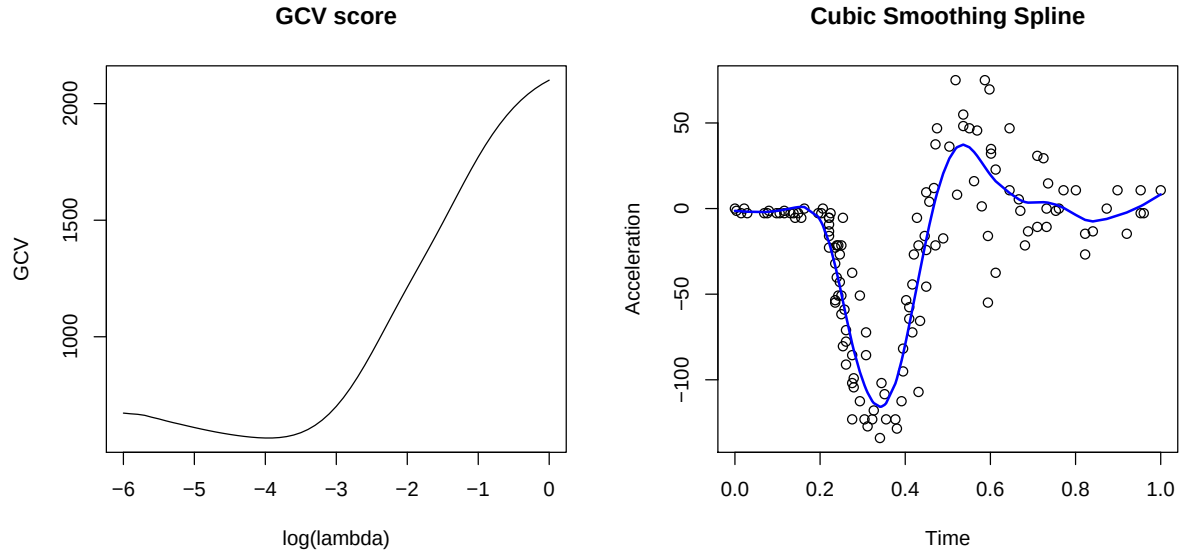
The following code imports the data, creates a function to calculate the cubic smooting spline estimate, finds the optimal GCV λ value and corresponding cubic smoothing spline estimator, and creates the GCV plot and fitted line plot in Figure 2.

```
rm(list=ls(all=TRUE))

#### Data ####
library(MASS)    # Package that contains the mcycle dataset
x<-mcycle$times
x<-(x-min(x))/(max(x)-min(x))  ## need to scale predictor to [0,1]
y<-mcycle$accel

#### Reproducing Kernel for <f,g>=int_0^1 f''(x)g''(x)dx #####
rk.1<-function(s,t){
return( (1/2)*min(s,t)^2*max(s,t) - (1/6)*(min(s,t))^3 )
}

#### Function to obtain the Gram Matrix ####
get.gram.1<-function(X){
n<-dim(X)[1] ;
Gram<-matrix(0,n,n) #initializes Gram array ;
#i=index for rows
#j=index for columns
Gram<-as.matrix(Gram) # Gram matrix ;
for (i in 1:n){
        for (j in 1:n){
                Gram[i,j]<-rk.1(X[i,],X[j,])
                    }
              }
return(Gram) }
```

```
#### Function to find a stable gen-inverse of a symmetric nonneg def ####
####  matrix to avoid numerical problems for small lambdas           ####
gen.inv<-function(X,eps=1e-12){
eig.X<-eigen(X,symmetric=T) ;
P<-eig.X[[2]] ;
lambda<-eig.X[[1]] ;
ind<-lambda>eps ;
lambda[ind]<-1/lambda[ind] ;
lambda[!ind]<-0 ;
ans<-P%*%diag(lambda,nrow=length(lambda))%*%t(P) ;
return(ans) }

#### Cubic Smoothing Spline Function ####
smoothing.spline<-function(X,y,lambda){
X<-as.matrix(X)
Gram<-get.gram.1(X) #Gram matrix (nxn) ;
n<-dim(X)[1] # n=length of y ;
J<-matrix(1,n,1) # vector of ones dim ;
T<-cbind(J,X) # matrix with a basis for the null space of the penalty ;
Q<-cbind(T,Gram) # design matrix ;
m<-dim(T)[2] # dimension of the null space of the penalty ;
S<-matrix(0,n+m,n+m) #initialize S ;
S[(m+1):(n+m),(m+1):(n+m)]<-Gram #non-zero part of S ;
M<-(t(Q)%*%Q+lambda*S) ;
M.inv<-gen.inv(M) # gen-inverse of M ;
gamma.hat<-crossprod(M.inv,crossprod(Q,y)) ;
f.hat<-Q%*%gamma.hat ;
A<-Q%*%M.inv%*%t(Q) ;
tr.A<-sum(diag(A)) #trace of hat matrix ;
rss<-t(y-f.hat)%*%(y-f.hat) #residual sum of squares ;
gcv<-n*rss/(n-tr.A)^2 #obtain GCV score ;
return(list(f.hat=f.hat,gamma.hat=gamma.hat,gcv=gcv))
}

#### Find an optimal lambda using GCV... ####
log.lambda<-seq(-6,0,by=.1) ;
V<-rep(0,length(log.lambda)) ;
for (i in 1:length(log.lambda)){
V[i]<-smoothing.spline(x,y,10^log.lambda[i])$gcv #obtain GCV score ;
}
min.ind<-order(V)[1] # extract index of min(V) ;
opt.mod.1<-smoothing.spline(x,y,10^log.lambda[min.ind]) #fit optimal model ;
```

```
### Plot of GCV ####
plot(log.lambda,V,type="l",main="GCV score",xlab="log(lambda)",ylab="GCV") ;

#### Fitted Line Plot for Cubic Smoothing Spline ####
plot(x,y,xlab="x",ylab="response",main="Cubic Smoothing Spline") ;
lines(x,opt.mod.1$f.hat,type="l",lty=1,lwd=2,col="blue") ;
```

Figure 2: Cubic Smoothing Spline GCV plot (left) and optimal fitted line plot (right)



A.2 RKHS solution applied to Ridge Regression

Here an example of ridge regression in the RKHS framework is provided using the labor statistics data of Longley (1967) which is known for having high collinearity. We seek to model Employment based on Prices, GNP, Unemployment, Military Size, Population Size, and Year. The following code, imports the data, creates a function to calculate the ridge regression estimate, finds the optimal GCV λ value and corresponding ridge regression estimator, creates the GCV plot in Figure 3, and finally converts the RKHS estimator back to slope coefficients as in (27) and compares to the ordinary least squares solution.

```
#### Data ####
data(longley)
y <- longley[,1] ;
X <- as.matrix(longley[,2:7]) ;
for(j in 1:ncol(X))  # scale each predictor to [0,1]
X[,j]<-(X[,j]-min(X[,j]))/(max(X[,j])-min(X[,j])) ;

#### Reproducing Kernel #####
rk.2<-function(s,t){
p<-length(s) ;
rk<-0 ;
for (i in 1:p){
rk<-s[i]*t[i]+rk
}
return( (rk) )
}

#### Gram matrix #####
get.gram.2<-function(X){
n<-dim(X)[1];
Gram<-matrix(0,n,n);
#initializes Gram array;
#i=index for rows;
#j=index for columns;
for(i in 1:n){
        for (j in 1:n){
                Gram[i,j]<-rk.2(X[i,],X[j,])
                     }
              }
return(Gram) }

#### Ridge Regression Function ####
ridge.regression<-function(X,y,lambda){
Gram<-get.gram.2(X) #Gram matrix (nxn) ;
n<-dim(X)[1] # n=length of y ;
J<-matrix(1,n,1) # vector of ones dim ;
Q<-cbind(J,Gram) # design matrix ;
m<-1 # dimension of the null space of the penalty ;
S<-matrix(0,n+m,n+m) #initialize S ;
S[(m+1):(n+m),(m+1):(n+m)]<-Gram #non-zero part of S ;
M<-(t(Q)%*%Q+lambda*S) ;
M.inv<-gen.inv(M) # gen-inverse of M ;
```

```
gamma.hat<-as.numeric(crossprod(M.inv,crossprod(Q,y))) ;
f.hat<-Q%*%gamma.hat ;
A<-Q%*%M.inv%*%t(Q) ;
tr.A<-sum(diag(A)) #trace of hat matrix ;
rss<-t(y-f.hat)%*%(y-f.hat) #residual sum of squares ;
gcv<-n*rss/(n-tr.A)^2 #obtain GCV score ;
beta.hat.0<-gamma.hat[1] #intercept ;
beta.hat<-c(beta.hat.0,gamma.hat[-1]%*%X) # slopes ;
return(list(f.hat=f.hat,gamma.hat=gamma.hat,gcv=gcv,beta.hat=beta.hat))
}

#### Find an optimal lambda using GCV... ####
log.lambda<-seq(-6,0,by=.1) ;
V<-rep(0,length(log.lambda)) ;
beta.mat<-matrix(0,length(log.lambda),ncol(X)+1) ;
for (i in 1:length(log.lambda)){
ans.i<-ridge.regression(X,y,10^log.lambda[i]) ;
V[i]<-ans.i$gcv #obtain GCV score ;
beta.mat[i,]<-ans.i$beta.hat #obtain beta estimates ;
}

#### Plot of GCV scores ####
plot(log.lambda,V,type="l",main="GCV score",xlab="log(lambda)",ylab="GCV") ;

#### Plot betas across lambda ####
plot(log.lambda,beta.mat[,2],main="Betas Across Lambda",xlab="log(lambda)",
ylab="Beta",col=0, ylim=range(beta.mat[,-1])) ;
for(j in 1:ncol(X))
lines(log.lambda,beta.mat[,j+1],col=j) ;
legend(-2,80,legend=paste("Beta",1:ncol(X)),lty=rep(1,ncol(X)),col=1:ncol(X)) ;

### Fit GCV optimal ridge regression ####
min.ind<-order(V)[1] # extract index of min(V) ;
opt.mod.2<-ridge.regression(X,y,10^log.lambda[min.ind]) ;

#### Fit ordinary least squares regression (i.e., lambda=0, no penalty) ####
ols.mod.2<-ridge.regression(X,y,0) ;

#### Print Ridge Regression Estimates ####
round(opt.mod.2$beta.hat,2) ;
#### Print OLS Estimates ####
round(ols.mod.2$beta.hat,2) ;
```

The last two print statments (which include the intercept $\hat{\beta}_0$) give

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{\text{opt}} &= [82.78,\ 54.17,\ \ 5.36,\ 1.38,\ -28.79,\ \ \ 5.40,\ -0.61]' \\ \hat{\boldsymbol{\beta}}_{\text{ols}} &= [81.64,\ 84.49,\ 10.71,\ 2.39,\ -39.04,\ -21.28,\ \ 2.40]'\end{aligned}$$

Figure 3: Ridge Regression GCV plot (left) and change in $\hat{\beta}_j$ across λ (right)