

**Stat 648: Assignment 3 Solutions** (85 points)

**(HTF 6.2)** (14 pts.) For local linear regression:

$$\begin{aligned} \sum_{i=1}^N (x_i - x_0) l_i(x_0) &= b(x_0)^T (\mathbf{B}^T \mathbf{W}(x_0) \mathbf{B})^{-1} \mathbf{B}^T \mathbf{W}(x_0) \begin{bmatrix} x_1 - x_0 \\ x_2 - x_0 \\ \vdots \\ x_N - x_0 \end{bmatrix} \\ &= b(x_0)^T (\mathbf{B}^T \mathbf{W}(x_0) \mathbf{B})^{-1} \mathbf{B}^T \mathbf{W}(x_0) \mathbf{B} \begin{bmatrix} -x_0 \\ 1 \end{bmatrix} \\ &= b(x_0)^T \begin{bmatrix} -x_0 \\ 1 \end{bmatrix} = [1 \quad x_0] \begin{bmatrix} -x_0 \\ 1 \end{bmatrix} = 0 \end{aligned}$$

For local polynomial regression of any degree:

First,  $b(x_0)^T = [1 \quad x_0 \quad x_0^2 \quad \cdots \quad x_0^k]$  and  $\mathbf{B} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^k \end{bmatrix}$ . So,

$$\begin{aligned} b_0(x_0) &= \sum_{i=1}^N (x_i - x_0)^0 l_i(x_0) = \sum_{i=1}^N l_i(x_0) \\ &= b(x_0)^T (\mathbf{B}^T \mathbf{W}(x_0) \mathbf{B})^{-1} \mathbf{B}^T \mathbf{W}(x_0) \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \\ &= b(x_0)^T (\mathbf{B}^T \mathbf{W}(x_0) \mathbf{B})^{-1} \mathbf{B}^T \mathbf{W}(x_0) \mathbf{B} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= b(x_0)^T \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 1 \end{aligned}$$

For local polynomial regression of degree  $k$ :

Suppose  $j \in \{1, 2, \dots, k\}$ . By the Binomial Theorem,

$$(x_i - x_0)^j = \sum_{m=0}^j \binom{j}{m} (-x_0)^{j-m} x_i^m = [1 \quad x_i \quad x_i^2 \quad \cdots \quad x_i^k] \begin{bmatrix} \binom{j}{0} (-x_0)^j \\ \binom{j}{1} (-x_0)^{j-1} \\ \vdots \\ \binom{j}{j} (-x_0)^0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

$$\text{So, } \begin{bmatrix} (x_1 - x_0)^j \\ (x_2 - x_0)^j \\ \vdots \\ (x_N - x_0)^j \end{bmatrix} = \mathbf{B} \begin{bmatrix} \binom{j}{0} (-x_0)^j \\ \binom{j}{1} (-x_0)^{j-1} \\ \vdots \\ \binom{j}{j} (-x_0)^0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \text{ and}$$

$$b_0(x_0) = \sum_{i=1}^N (x_i - x_0)^j l_i(x_0) = b(x_0)^T (\mathbf{B}^T \mathbf{W}(x_0) \mathbf{B})^{-1} \mathbf{B}^T \mathbf{W}(x_0) \mathbf{B} \begin{bmatrix} \binom{j}{0} (-x_0)^j \\ \binom{j}{1} (-x_0)^{j-1} \\ \vdots \\ \binom{j}{j} (-x_0)^0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$= \sum_{m=0}^j \binom{j}{m} (-x_0)^{j-m} x_0^m = (x_0 - x_0)^j = 0$$

For a local polynomial regression of degree  $k$  the bias depends only on terms that are of degree  $k + 1$  or higher.

**(HTF 7.3)** (12 pts.) (a) Let  $\hat{f}^{-i}$  be the minimizer of

$$\sum_{i \leq j \leq N, j \neq i} (y_j - f(x_j))^2 + \text{penalty}(f)$$

where the penalty is 0 for least squares and  $\lambda \int \{f''(t)\}^2 dt$  for cubic smoothing splines, i.e.  $\hat{f}^{-i}$  is the minimizer for the fit of all but the  $i^{\text{th}}$  observation. Now consider the case where all  $x$ 's are used and  $y_i$  is replaced with  $\hat{f}^{-i}(x_i)$ . So  $\mathbf{X}$  remains the same and  $\mathbf{y}$  is replaced with  $\mathbf{y}^{-i} = [y_1, \dots, y_{i-1}, \hat{f}^{-i}(x_i), y_{i+1}, \dots, y_N]^T$ . Then  $\hat{f}^{-i}$  will also be the minimizer of

$$\sum_{i \leq j \leq N} (y_j - f(x_j))^2 + \text{penalty}(f)$$

since  $y_i^{-i} - \hat{f}^{-i}(x_i) = 0$  adds nothing to the sum of squares. Thus,  $\hat{f}^{-i} = \mathbf{S}\mathbf{y}^{-i}$  and

$$\begin{aligned} \hat{f}^{-i}(x_i) &= \sum_{j \neq i} S_{ij} y_j + S_{ii} \hat{f}^{-i}(x_i) \\ &= \sum_{j=1}^N S_{ij} y_j - S_{ii} y_i + S_{ii} \hat{f}^{-i}(x_i) \\ &= \hat{f}(x_i) - S_{ii} y_i + S_{ii} \hat{f}^{-i}(x_i) \end{aligned}$$

So,

$$-\hat{f}(x_i) = -\hat{f}^{-i}(x_i) - S_{ii}y_i + S_{ii}\hat{f}^{-i}(x_i)$$

and

$$\begin{aligned} y_i - \hat{f}(x_i) &= y_i - \hat{f}^{-i}(x_i) - S_{ii}y_i + S_{ii}\hat{f}^{-i}(x_i) \\ &= (1 - S_{ii})(y_i - \hat{f}^{-i}(x_i)) \end{aligned}$$

Therefore, the result holds.

(b) For the least squares case  $\mathbf{S}$  is idempotent and for the cubic smoothing spline case  $\mathbf{S} \geq \mathbf{S}^2$ . Also,  $\mathbf{S}$  is symmetric for both cases. Since  $\mathbf{S} \geq \mathbf{S}^2$ ,

$$S_{ii} \geq \sum_{j=1}^N S_{ij}S_{ji} = \sum_{j=1}^N S_{ij}^2 = S_{ii}^2 + \sum_{j \neq i} S_{ij}^2.$$

Thus  $S_{ii} \geq 0$  and  $S_{ii} \geq S_{ii}^2$ , i.e.  $S_{ii} \leq 1$ . Therefore, by part (a)  $|y_i - \hat{f}^{-i}(x_i)| \geq |y_i - \hat{f}(x_i)|$ .

(c) As long as  $\mathbf{S}$  only depends on  $\mathbf{X}$  and not  $\mathbf{y}$  the result (7.64) will hold.

**(HTF 7.6)** (5 pts.) For  $k$ -nearest neighbors

$$\hat{\mathbf{y}} = \sum_{x_i \in N_k(x)} \frac{y_i}{k}$$

where  $N_k(x)$  is the neighborhood containing the  $k$  closest points to  $x$ . A point is always its own nearest neighbor, so in matrix form we have  $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$  where every diagonal entry of  $\mathbf{S}$  is  $1/k$ . Thus,

$$df = \text{tr}(\mathbf{S}) = \sum_{i=1}^N \frac{1}{k} = \frac{N}{k}.$$

**(8)** i) (7 pts.) We know that

$$\begin{aligned} \gamma_i \phi_i(x) &= \int_0^1 \phi_i(z)K(z, x)dz \\ &= \int_0^1 \phi_i(z)(1 + 2zx + z^2x^2)dz \\ &= \int_0^1 \phi_i(z)dz + 2x \int_0^1 x\phi_i(z)dz + x^2 \int_0^1 z^2\phi_i(z)dz \\ &= a_i + b_ix + c_ix^2 \text{ for constants } a_i, b_i, c_i. \end{aligned}$$

Thus,  $\phi_i(x)$  is quadratic for all  $i$ .

Since  $\phi_i(x) = a_i + b_ix + c_ix^2$ , we have

$$\gamma_i(a_i + b_ix + c_ix^2) = \int_0^1 (a_i + b_iz + c_iz^2)(1 + 2zx + z^2x^2)dz.$$

By doing the integration we find that

$$\gamma_i(a_i + b_ix + c_ix^2) = (a_i + 1/2b_i + 1/3c_i) + (a_i + 2/3b_i + 1/2c_i)x + (1/3a_i + 1/4b_i + 1/5c_i)x^2.$$

Thus

$$\begin{aligned} 0 &= (1/5 - \gamma_i)c_i + 1/4b_i + 1/3a_i \\ 0 &= 1/2c_i + (2/3 - \gamma_i)b_i + a_i \\ 0 &= 1/3c_i + 1/2b_i + (1 - \gamma_i)a_i \end{aligned}$$

or

$$\begin{bmatrix} (1 - \gamma_i) & \frac{1}{2} & \frac{1}{3} \\ 1 & (\frac{2}{3} - \gamma_i) & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{4} & (\frac{1}{5} - \gamma_i) \end{bmatrix} \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

In order for this system of equations to have a nonzero solution, the determinant of

$$A \equiv \begin{bmatrix} (1 - \gamma_i) & \frac{1}{2} & \frac{1}{3} \\ 1 & (\frac{2}{3} - \gamma_i) & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{4} & (\frac{1}{5} - \gamma_i) \end{bmatrix}$$

must be 0. Thus, the  $\gamma_i$ 's are the eigenvalues of A, i.e.

$$\begin{aligned} \gamma_1 &= 1.71292 \\ \gamma_2 &= 0.15014 \\ \gamma_3 &= 0.00360 \end{aligned}$$

Now the eigenvectors of the matrix provide scaled coefficients for the quadratic eigenfunctions. We also know that  $\int_0^1 \phi_i(x)^2 dx = 1$  for  $\phi_i(x) = a_i + b_i x + c_i x^2$ . Therefore,

$$a_i^2 + a_i b_i + 2/3 a_i c_i + 1/3 b_i^2 + 1/2 b_i c_i + 1/5 c_i^2 = 1.$$

Using R, the eigenfunctions can be found to be

$$\begin{aligned} \phi_1(x) &= -0.5644618 - 0.6502907x - 0.2318203x^2 \\ \phi_2(x) &= -1.698942 + 1.999564x + 1.332234x^2 \\ \phi_3(x) &= 2.407276 - 13.695943x + 13.348086x^2 \end{aligned}$$

To verify that

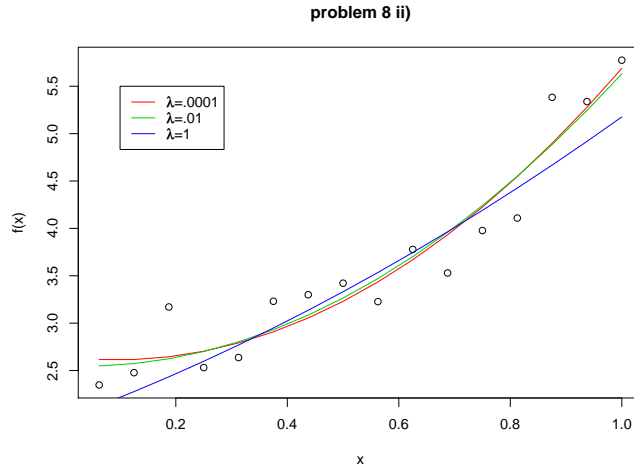
$$K(z, x) = \sum_{i=1}^3 \gamma_i \phi_i(z) \phi_i(x) = \sum_{i=1}^3 \gamma_i (a_i + b_i z + c_i z^2)(a_i + b_i x + c_i x^2)$$

we need to show that  $\sum_{i=1}^3 \gamma_i c_i^2 = 1$ ,  $\sum_{i=1}^3 \gamma_i b_i^2 = 2$ ,  $\sum_{i=1}^3 \gamma_i a_i^2 = 1$ ,  $\sum_{i=1}^3 \gamma_i a_i b_i = 0$ ,  $\sum_{i=1}^3 \gamma_i a_i c_i = 0$ , and  $\sum_{i=1}^3 \gamma_i b_i c_i = 0$ . These were all verified in R.

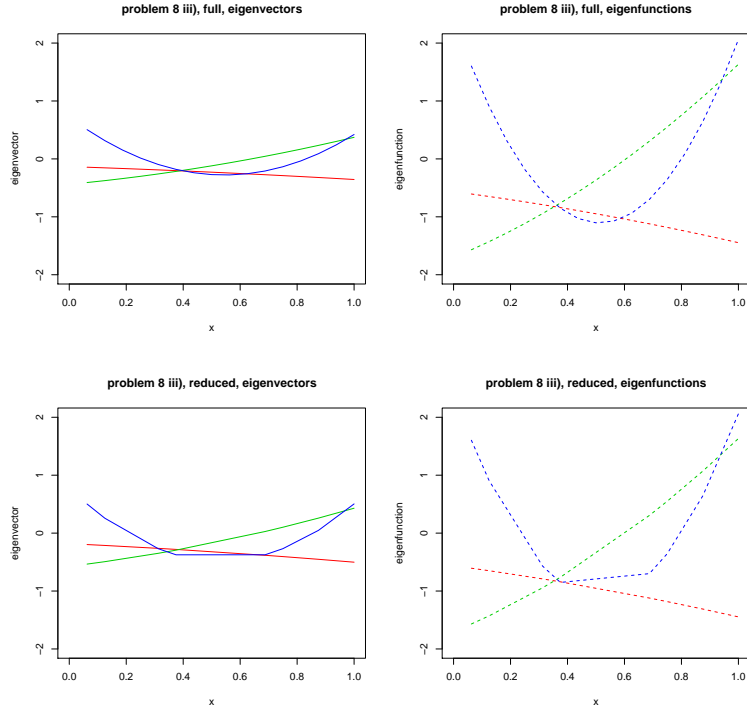
ii) (6 pts.) Plots of the fits are given below.

To find the degrees of freedom: We have  $f_\lambda(x_j) = \sum_{i=1}^N b_{\lambda i} K(x_j, x_i) = (\mathbf{Kb}_\lambda)_j$  (i.e. the  $j^{\text{th}}$  element of the column  $\mathbf{Kb}_\lambda$ ). Then,  $\hat{\mathbf{Y}} = \mathbf{Kb}_\lambda = \mathbf{K}(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{Y}$  and the df's can be computed as the trace of  $\mathbf{K}(\mathbf{K} + \lambda \mathbf{I})^{-1}$ .

The degrees of freedom are found to be 2.998098, 2.838628, and 1.721616.



iii) (4 pts.) The (nonzero) eigenvalues for the full matrix  $\mathbf{K}$  are 28.976667, 2.377899, and 0.053760. For the reduced  $\mathbf{K}$  that uses only observations  $i = 1, 2, 5, 6, 11, 12, 14, 16$  the eigenvalues are 14.756499, 1.485195, and 0.031057. Plots of the eigenvectors and eigenfunctions against the  $x$ 's are given below. The plots are similar in shape but different in scale, suggesting that the eigenvectors are approximately proportional to the eigenfunctions. The eigenfunctions of  $K$  are larger than the eigenvectors of  $\mathbf{K}$ , while the eigenvalues for  $K$  are smaller than the eigenvalues for  $\mathbf{K}$  (but still approximately proportional to one another).



iv) (7 pts.)

$$\begin{aligned} \int_0^1 g(x)K(z, x)dx &= \int_0^1 g(x) \sum_{i=1}^3 \gamma_i \phi_i(x) \phi_i(z) dx \\ &= \sum_{i=1}^3 \gamma_i \phi_i(z) \int_0^1 g(x) \phi_i(x) dx = 0. \end{aligned}$$

So,  $g$  is in the null space of the kernel  $K$ .

Next, we know that

$$\int_0^1 g(x)K(x, z)dx = \int_0^1 (a + bx + cx^2 + dx^3)(1 + 2zx + z^2x^2)dx = 0.$$

By setting  $a = 1$ , we find that  $b = -12$ ,  $c = 30$ , and  $d = -20$ . So  $g(x) = 1 - 12x + 30x^2 - 20x^3$ .

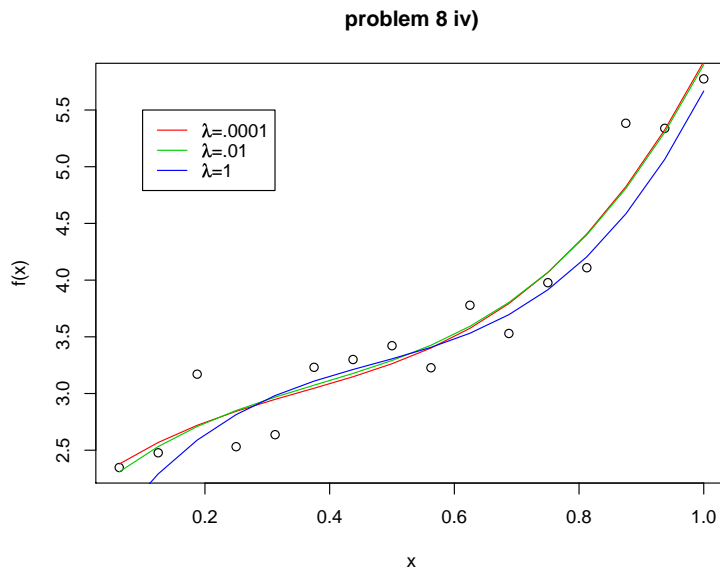
Now, for  $f \in \mathcal{H}_K$  we can write  $f(x) = c_1\phi_1(x) + c_2\phi_2(x) + c_3\phi_3(x)$ , and we can minimize  $\sum_{i=1}^{16} (y_i - (\alpha g(x_i) + f(x_i)))^2 + \lambda \|f\|_{\mathcal{H}_K}^2$  by finding

$$\min_{\alpha, c_1, c_2, c_3} \left[ \sum_{i=1}^{16} (y_i - \alpha g(x_i) - c_1\phi_1(x_i) - c_2\phi_2(x_i) - c_3\phi_3(x_i))^2 + \lambda \sum_{j=1}^3 c_j^2 / \gamma_j \right]$$

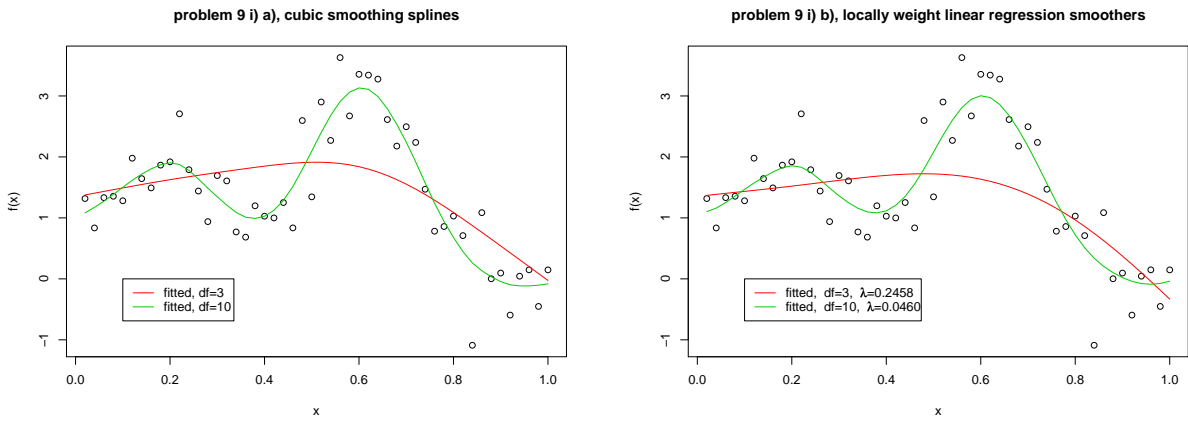
By using R, we arrive at the following:

$\lambda$	$\alpha$	$\alpha g(x)$	$f(x)$
0.0001	-0.35707	$-0.35708 + 4.28490x - 10.71226x^2 + 7.14151x^3$	$2.49579 - 0.00622x + 3.07956x^2$
0.01	-0.38249	$-0.38249 + 4.58993x - 11.47484x^2 + 7.64989x^3$	$2.41171 + 0.41200x + 2.69302x^2$
1	-0.54731	$-0.54731 + 6.56776x - 16.41940x^2 + 10.94627x^3$	$1.96000 + 2.22356x + 0.93597x^2$

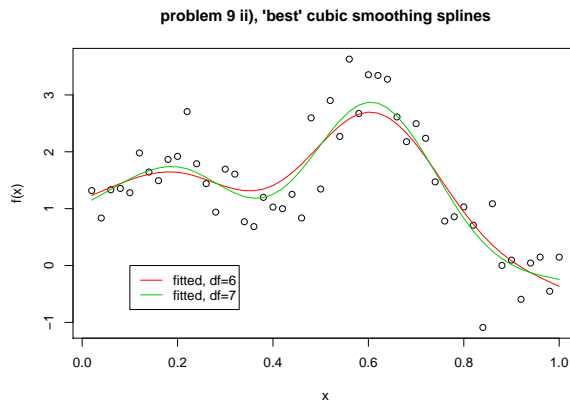
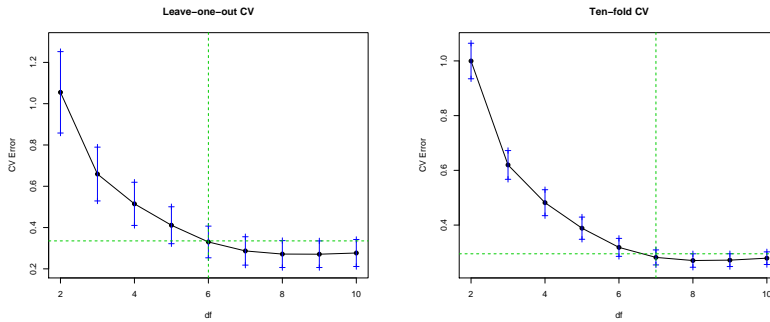
The fitted curves are given in the figure below.



(9) i) (6 pts.) Plots are given below for both a) and b). Note that for part b), the degrees of freedom 3 and 10 correspond to  $\lambda$  values of 0.2458 and 0.0460, respectively.



ii) (6 pts.) Plots of the CV error are given below. For leave-one-out CV we choose  $df=6$ . For ten-fold CV we choose  $df=7$ . The fitted curves for these  $df$ 's are also provided below.

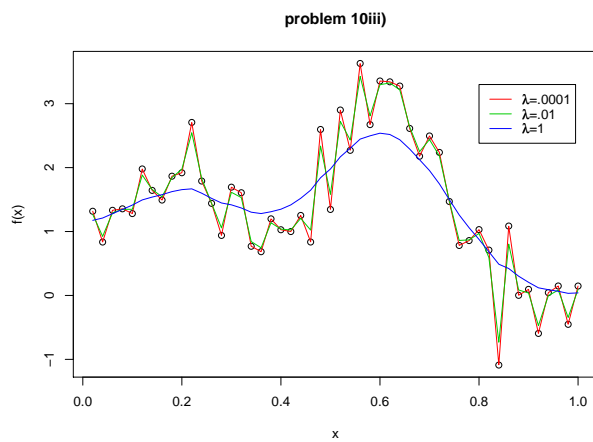
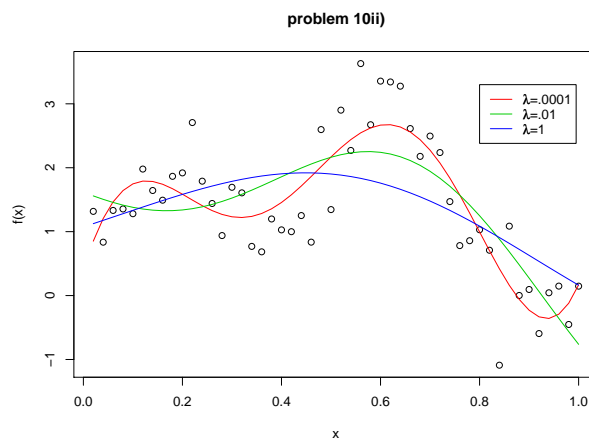
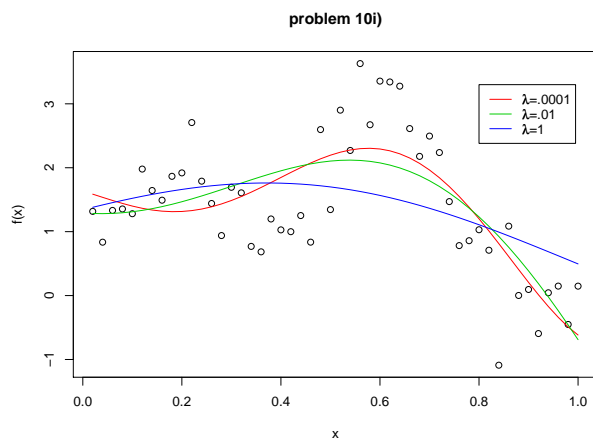


(10) Plots for i)-iii) are given below.

i) (6 pts.) degrees of freedom: 4.977618, 3.679214, and 2.158616

ii) (6 pts.) degrees of freedom: 5.906136, 4.367702, and 2.564012

iii) (6 pts.) degrees of freedom: 49.877790, 40.978458, and 7.293336



## Code for computing problems

```
#####problem 8#####
####i)
J=matrix(c(1,1,1/3,1/2,2/3,1/4,1/3,1/2,1/5),nrow=3)
#eigenvectors & eigenvalues
e=eigen(J)$values
E=eigen(J)$vectors

#finds the eigenfunctions
E2=E
for(i in 1:3){
con=1/sqrt(E[1,i]^2+E[1,i]*E[2,i]+2/3*E[1,i]*E[3,i]+1/3*E[2,i]^2+1/2*E[2,i]*E[3,i]+1/5*E[3,i]^2)
E2[,i]=E[,i]*con}

#verify the claim
sum(e*E2[3,]^2)
sum(e*E2[2,]^2)
sum(e*E2[1,]^2)
sum(e*E2[1,]*E2[3,])
sum(e*E2[1,]*E2[2,])
sum(e*E2[2,]*E2[3,])

####ii)
K=function(x,z){(1+z*x)^2}
Km=matrix(nrow=16,ncol=16)
for(i in 1:16){
for(j in 1:16){
Km[i,j]=K(p8[i,1],p8[j,1])}}

I=diag(16)
lambda=c(.0001,.01,1)
df=rep(0,3)

plot(p8[,1],p8[,2],main="problem 8 ii",xlab="x",ylab="f(x)")
for(j in 1:3){
lam=lambda[j]
M=solve(Km+lam*I)
b=M%*%p8[,2]
lines(p8[,1],Km%*%b,col=(j+1))
df[j]=sum(diag(Km%*%M))}
legend(.1,5.5,legend=c(expression(paste(lambda,"=.0001")),expression(paste(lambda,"=.01")),
expression(paste(lambda,"=1"))),lty=rep(1,3),col=c(2,3,4))

####iii)
phi1=function(x){E2[1,1]+E2[2,1]*x+E2[3,1]*x^2}
phi2=function(x){E2[1,2]+E2[2,2]*x+E2[3,2]*x^2}
phi3=function(x){E2[1,3]+E2[2,3]*x+E2[3,3]*x^2}
eigen(Km)$values
V=eigen(Km)$vectors
X=p8[,1]
par(mfrow=c(1,2))
plot(seq(0,1,,100),seq(-2,2,,100),type="n",xlab="x",ylab="eigenvector",
main="problem 8 iii), full, eigenvectors")
```

```

lines(X,V[,1],col=2)
lines(X,V[,2],col=3)
lines(X,V[,3],col=4)
plot(seq(0,1,,100),seq(-2,2,,100),type="n",xlab="x",ylab="eigenfunction",
main="problem 8 iii), full, eigenfunctions")
lines(X,phi1(X),lty=2,col=2)
lines(X,phi2(X),lty=2,col=3)
lines(X,phi3(X),lty=2,col=4)

#with only data for i=1,2,5,6,11,12,14,16
index=c(1,2,5,6,11,12,14,16)
Kmred=Km[index,index]
eigen(Kmred)$values
V=eigen(Kmred)$vectors
X=p8[index,1]
par(mfrow=c(1,2))
plot(seq(0,1,,100),seq(-2,2,,100),type="n",xlab="x",ylab="eigenvector",
main="problem 8 iii), reduced, eigenvectors")
lines(X,V[,1],col=2)
lines(X,V[,2],col=3)
lines(X,V[,3],col=4)
plot(seq(0,1,,100),seq(-2,2,,100),type="n",xlab="x",ylab="eigenfunction",
main="problem 8 iii), reduced, eigenfunctions")
lines(X,phi1(X),lty=2,col=2)
lines(X,phi2(X),lty=2,col=3)
lines(X,phi3(X),lty=2,col=4)

####iv)
#finding b,c,d for a=1
H=matrix(c(1/2,2/3,1/4,1/3,1/2,1/5,1/4,2/5,1/6),nrow=3)
coef=solve(H,b=c(-1,-1,-1/3))
g=function(x){1+coef[1]*x+coef[2]*x^2+coef[3]*x^3}

#function for finding alpha and f
F=function(t){
a=t[1]
c1=t[2]
c2=t[3]
c3=t[4]
sum((p8[,2]-a*g(X)-c1*phi1(X)-c2*phi2(X)-c3*phi3(X))^2)+lam*sum(t[2:4]^2/e)}
X=p8[,1]

plot(p8[,1],p8[,2],main="problem 8 iv)",xlab="x",ylab="f(x)")

lam=.0001
A.0001=optim(rep(1,4),F)$par
A.0001[1]
#coefficients for f
A.0001[2]*E2[,1]+A.0001[3]*E2[,2]+A.0001[4]*E2[,3]
#coefficients for alpha*g
A.0001[1]*c(1,-12,30,-20)
#coefficients for alpha*g+f
est=c(A.0001[2]*E2[,1]+A.0001[3]*E2[,2]+A.0001[4]*E2[,3],0)+A.0001[1]*c(1,-12,30,-20)

```

```

agf=function(x){est[1]+est[2]*x+est[3]*x^2+est[4]*x^3}
lines(X,agf(X),col=2)

lam=.01
A.01=optim(rep(1,4),F)$par
A.01[1]
#coefficients for f
A.01[2]*E2[,1]+A.01[3]*E2[,2]+A.01[4]*E2[,3]
#coefficients for alpha*g
A.01[1]*c(1,-12,30,-20)
#coefficients for alpha*g+f
est=c(A.01[2]*E2[,1]+A.01[3]*E2[,2]+A.01[4]*E2[,3],0)+A.01[1]*c(1,-12,30,-20)
agf=function(x){est[1]+est[2]*x+est[3]*x^2+est[4]*x^3}
lines(X,agf(X),col=3)

lam=1
A1=optim(rep(1,4),F)$par
A1[1]
#coefficients for f
A1[2]*E2[,1]+A1[3]*E2[,2]+A1[4]*E2[,3]
#coefficients for alpha*g
A1[1]*c(1,-12,30,-20)
#coefficients for alpha*g+f
est=c(A1[2]*E2[,1]+A1[3]*E2[,2]+A1[4]*E2[,3],0)+A1[1]*c(1,-12,30,-20)
agf=function(x){est[1]+est[2]*x+est[3]*x^2+est[4]*x^3}
lines(X,agf(X),col=4)
legend(.1,5.5,legend=c(expression(paste(lambda,"=.0001")),expression(paste(lambda,"=.01")),
expression(paste(lambda,"=1"))),lty=rep(1,3),col=c(2,3,4))

#####problem 9#####
####i)a)
cubic3=smooth.spline(p9[,1],p9[,2],df=3)
cubic10=smooth.spline(p9[,1],p9[,2],df=10)
plot(p9[,1],p9[,2],main="problem 9 i) a), cubic smoothing splines",xlab="x",ylab="f(x)")
lines(predict(cubic3)$x,predict(cubic3)$y,col=2)
lines(predict(cubic10)$x,predict(cubic10)$y,col=3)
legend(.1,0,legend=c("fitted, df=3","fitted, df=10"),lty=rep(1,2),col=c(2,3))

####i)b)
B=matrix(c(rep(1,50),p9[,1]),ncol=2)
yhat=rep(0,50)
plot(p9[,1],p9[,2],main="problem 9 i) b), locally weight
linear regression smoothers",xlab="x",ylab="f(x)")

t=c(3,10)
for(m in 1:2){
Lfind=function(d){
Kn=function(x,z){dnorm((x-z)/d)}
for(i in 1:50){
for(j in 1:50){

```

```

Knm[i,j]=Kn(p9[i,1],p9[j,1])}}
tr=0
for(i in 1:50){
W=diag(Knm[i,])
S=matrix(c(1,i/50),nrow=1)%*%solve(t(B)%*%W)%*%B)%*%t(B)%*%W
tr=tr+S[,i]}
return(tr-t[m])}
lm=uniroot(Lfind,c(.01,3))$root
tr=0
Kn=function(x,z){dnorm((x-z)/lm)}
for(i in 1:50){
for(j in 1:50){
Knm[i,j]=Kn(p9[i,1],p9[j,1])}}
for(i in 1:50){
W=diag(Knm[i,])
S=matrix(c(1,i/50),nrow=1)%*%solve(t(B)%*%W)%*%B)%*%t(B)%*%W
yhat[i]=S%*%p9[,2]
tr=tr+S[,i]}
cat(tr,lm,"\n")
lines(p9[,1],yhat,col=m+1)}
legend(.1,0,legend=c(expression(paste("fitted, df=3, ",lambda,"=0.2458")),
expression(paste("fitted, df=10, ",lambda,"=0.0460"))),lty=rep(1,2),col=c(2,3))

####ii)
##Leave-one-out CV
n=10
perr=matrix(nrow=n-1,ncol=50)
for(i in 2:n){
for(j in 1:50){
dat=p9[-j,]
cub=smooth.spline(dat[,1],dat[,2],df=i)
perr[i-1,j]=(predict(cub,x=j/50)$y-p9[j,2])^2}}
se=apply(perr,1,sd)/sqrt(50)
err=apply(perr,1,mean)
plot(seq(2,n,,100),seq(0.2,1.3,,100),xlab="df",ylab="CV Error",type="n",main="Leave-one-out CV")
points(2:n,err,pch=19)
lines(2:n,err)
points(2:n,err+se,pch=3,col=4)
points(2:n,err-se,pch=3,col=4)
for(i in 1:10){
lines(rep(i+1,2),c((err+se)[i],(err-se)[i]),col=4)}
abline(h=min(err+se),col=3,lty=2)
abline(v=6,col=3,lty=2)

##Ten-fold CV
n=10
perr=matrix(nrow=n-1,ncol=10)
for(i in 2:n){
for(j in 1:10){
dat=p9[-c(j,j+10,j+20,j+30,j+40),]
datl=p9[c(j,j+10,j+20,j+30,j+40),]
cub=smooth.spline(dat[,1],dat[,2],df=i)

```

```

perr[i-1,j]=mean((predict(cub,x=dat1[,1])$y-dat1[,2])^2)}
se=apply(perr,1,sd)/sqrt(50)
err=apply(perr,1,mean)
plot(seq(2,n,,100),seq(0.24,1.07,,100),xlab="df",ylab="CV Error",type="n",main="Ten-fold CV")
points(2:n,err,pch=19)
lines(2:n,err)
points(2:n,err+se,pch=3,col=4)
points(2:n,err-se,pch=3,col=4)
for(i in 1:10){
lines(rep(i+1,2),c((err+se)[i],(err-se)[i]),col=4)}
abline(h=min(err+se),col=3,lty=2)
abline(v=7,col=3,lty=2)

##plots of selected fits
cubic6=smooth.spline(p9[,1],p9[,2],df=6)
cubic7=smooth.spline(p9[,1],p9[,2],df=7)
plot(p9[,1],p9[,2],main="problem 9 ii), 'best' cubic smoothing splines",xlab="x",ylab="f(x)")
lines(predict(cubic6)$x,predict(cubic6)$y,col=2)
lines(predict(cubic7)$x,predict(cubic7)$y,col=3)
legend(.1,0,legend=c("fitted, df=6","fitted, df=7"),lty=rep(1,2),col=c(2,3))

#####problem 10####
Ka=function(x,z){exp(-(x-z)^2)}
Kb=function(x,z){exp(-2*(x-z)^2)}
Kc=function(x,z){exp(-2*abs(x-z))}

Kam=Kbm=Kcm=matrix(nrow=50,ncol=50)
for(i in 1:50){
for(j in 1:50){
Kam[i,j]=Ka(p9[i,1],p9[j,1])
Kbm[i,j]=Kb(p9[i,1],p9[j,1])
Kcm[i,j]=Kc(p9[i,1],p9[j,1])}}

I=diag(50)
lambda=c(.0001,.01,1)

dfa=dfb=dfc=rep(0,3)
plot(p9[,1],p9[,2],main="problem 10i)",xlab="x",ylab="f(x)")
for(j in 1:3){
lam=lambda[j]
M=solve(Kam+lam*I)
b=M%*%p9[,2]
lines(p9[,1],Kam%*%b,col=(j+1))
TT=Kam%*%M
dfa[j]=sum(diag(Kam%*%M))}
legend(.8,3.3,legend=c(expression(paste(lambda,"=.0001")),expression(paste(lambda,"=.01")),
expression(paste(lambda,"=1"))),lty=rep(1,3),col=c(2,3,4))

plot(p9[,1],p9[,2],main="problem 10ii)",xlab="x",ylab="f(x)")
for(j in 1:3){
lam=lambda[j]
M=solve(Kbm+lam*I)

```

```

b=M%%p9[,2]
lines(p9[,1],Kbm%%b,col=(j+1))
dfb[j]=sum(diag(Kbm%%M))}
TT2=Kbm%%M
legend(.8,3.3,legend=c(expression(paste(lambda,"=.0001")),expression(paste(lambda,"=.01")),
expression(paste(lambda,"=1"))),lty=rep(1,3),col=c(2,3,4))

plot(p9[,1],p9[,2],main="problem 10iii",xlab="x",ylab="f(x)")
for(j in 1:3){
lam=lambda[j]
M=solve(Kcm+lam*I)
b=M%%p9[,2]
lines(p9[,1],Kcm%%b,col=(j+1))
dfc[j]=sum(diag(Kcm%%M))}
TT3=Kcm%%M
legend(.8,3.3,legend=c(expression(paste(lambda,"=.0001")),expression(paste(lambda,"=.01")),
expression(paste(lambda,"=1"))),lty=rep(1,3),col=c(2,3,4))

```